Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Mojeskolka.info – tvorba uzavřené komunitní sítě pro školky

Bc. Jan Bradáč

Vedoucí práce: Ing. Jiří Hunka

26. června 2015

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Jiřímu Hunkovi za trpělivost a snahu pomoci, se kterou odpovídal na mé dotazy, ale také za poskytnutí praktických rad a znalostí, které významně přispěly k řadě správných rozhodnutí. Dále bych rád poděkoval panu Jaroslavu Zetelovi za výbornou myšlenku, jejíž realizací se tato práce snaží být.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 26. června 2015 .....

Odkaz na tuto práci

Bradáč, Jan. *Mojeskolka.info – tvorba uzavřené komunitní sítě pro školky.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá analýzou, návrhem, implementací a následným testováním informačního systému mojeskolka.info. Tento projekt realizuje softwarové řešení uzavřené komunitní sítě, která je určená pro školky a pro rodiče dětí, které tam dochází. Výsledný produkt má vyřešit obousměrnou komunikaci mezi rodiči a školkou a usnadnit předávání informací mezi oběma stranami.

Klíčová slova Mojeskolka.info, webová aplikace, návrh GUI, PHP, Laravel

Abstract

This thesis contains the analysis, design, implementation and the following testing of an information system mojeskolka.info (mykindergarten.info). The project aims at creating a software solution for a closed social network targetet at kindergartens and the parents of kids that are being educated there. The resulting product should improve communication between the parents and the kindergarten.

Keywords Mojeskolka.info, web application, GUI design, PHP, Laravel

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Mojeskolka.info

Projekt mojeskolka.info je softwarové řešení uzavřené komunitní sítě pro mateřské školky, určené pro pracovníky školek a pro rodiče dětí, které tam dochází. Cílem projektu je vyřešit a výrazně usnadnit obousměrnou komunikaci mezi rodiči a školkou.

Zjednodušení a zpřehlednění komunikace by mělo být dosaženo přesunutím informací a upozornění pro rodiče do elektornické podoby. Tedy přechodem od tištěných, či psaných dokumentů umístěných na dveřích a nástěnkách ve školce, jak „systém“ funguje nyní, k přehledným notifikacím a zprávám v informačním systému mojeskolka.info. To umožní snadné předávání informací rodičům, kteří pak nebudou muset každý den prohledávat prostory školky a kontrolovat, zda jim neunikly žádné důležité zprávy od učitelů.

Systém by se, z povahy zadání, měl skládat minimálně ze dvou částí, tedy z části pro učitele, kde jsou informace do systému zadávány, a z části pro rodiče, kde jsou jim v co možná nejpřehlednější formě prezentovány.

Ambicí projektu do budoucna je stát se oficiálním informačním kanálem, který tyto školky budou používat jako primární způsob sdílení informací mezi učiteli a rodiči.

Počátky projektu

Prvotní myšlenka na tvorbu podobného systému vznikla právě z hromadné nespokojenosti rodičů se současným stavem, který nejsnáze vystihuje nejistota, špatná informovanost a nutnost často se spoléhat pouze na ústní šíření informací mezi rodiči.

Zadavatel projektu, majitel firmy působící v oblasti informačních technologií, je sám rodič dvou dětí a zaznamenal neuspokojivou situaci ve školkách,

včetně nespokojenosti ostatních rodičů a absence existujícího řešení, které by školky využívaly.

Zároveň zadavatel navázal kontakt s vedením školky, do kterého jeho děti dochází. Mohl se tak utvrdit v tom, že i z této strany by byl zájem o podobný nástroj, který by usnadňoval předávání informací a fungovaní školky, především po stránce spolupráce s rodiči, včetně získávání jejich zpětné vazby a dalších funkcí.

S ranou myšlenkou proto v létě 2014 oslovil autora této práce, zda by měl zájem na podobném projektu spolupracovat. Vzhledem k tomu, že se jedná o poměrně zajímavý nápad, který má potenciál být velmi užitečný, došlo k dohodě o spolupráci na projektu a prvotní práce začaly již v zimním semestru akademického roku 2014/2015, v rámci předmětu Návrh Uživatelského Rozhraní – takzvaný NUR.

Dosavadní práce na projektu

Od léta roku 2014 se projekt začal rozvíjet a od září se na něm začalo intenzivně pracovat, a to ve dvou paralelních rovinách:

Komunikace se zadavatelem

Projekt byl průběžně doplňován a upřesňován pravidelnými diskusemi se zadavatelem, na jejichž základě byly jednak upřesňovány požadavky na systém, a jednak byly zákazníkovi průběžně prezentovány výsledky práce, které mohl připomínkovat a nad nimiž se zhotovitelem, tedy autorem této práce, diskutoval. Na základě těchto pravidelných diskusí byly do projektu zanášeny nové poznatky, díky kterým byl projekt velmi blízko zadavatelově představě již ve fázi návrhu.

Detaily spolupráce se zadavatelem-zákazníkem budou blíže rozebrány dále v této práci v sekci Spolupráce se zadavatelem v kapitole Analýza.

Týmová spolupráce v předmětu NUR

V tomto předmětu zpracovával čtyřčlenný tým pod vedením autora této práce detailní návrh uživatelského rozhraní pro rodičovskou část systému (jedná se o část systému určenou pro prezentaci informací rodičům. O rozdělení systému více v příslušné sekci Rozdělení na jednotlivá rozhraní).

Složení týmu:

- Bradáč Jan (autor této práce)
- Duchač Jan
- Fajfr Marek

- Plocová Martina

V rámci této týmové spolupráce byly dle metodik tvorby uživatelského rozhraní nejprve zpracovány požadavky na systém a jednoduché případy užití, následně byly vytvořeny návrhy jednotlivých stránek v systému, ve formě takzvaných wireframů, a na jejich základě byl vytvořen prototyp s demonstrativní funkcionalitou. Prototyp byl poté využit pro finální otestování práce jednak reálnými osobami, jednak metodikami specifickými pro návrh uživatelského rozhraní, ze kteréhožto testování byla sestavena závěrečná zpráva s požadavky a doporučeními na změny v uživatelském rozhraní aplikace.

Celý tým fungoval velmi dobře – i ostatní členové týmu (kteří na projektu nemuseli mít tak velký zájem jako autor této práce) pracovali s velkým nasazením a snažili se o vysokou kvalitu výsledného návrhu a všech jeho součástí. Velkým povzbuzením pro tým byla skutečnost, že nejen samotní jeho členové, ale i rodiče a spolužáci, kteří testovali výsledný prototyp aplikace, chválili nápad a celkovou koncepci projektu, který vyplňuje komunikační mezeru mezi školkami a rodiči pomocí moderních technologií cílených na koncové uživatele.

Využití výsledků z NUR

Díky prvotřídní spolupráci týmu byly výstupem z předmětu NUR kvalitní materiály, které mohou posloužit jako základ pro tvorbu mnohých částí této práce: především budou využity závěry testování, hotový prototyp aplikace, návrhy jednotlivých stránek, rešerše a případy užití. Všechny tyto položky sice bude nutné dále přepracovat a upravit pro potřeby této práce, ale poslouží jako hrubá kostra pro další postup, a v příslušných sekcích této práce bude na výsledky týmového úsilí z předmětu NUR odkazováno.

Vymezení pojmů

V textu této práce budou opakovaně užívány pojmy související s rozebíranou tematikou nebo jinak spojené s touto prací, proto je níže možné nalézt detailnější upřesnění jednotlivých výrazů a jejich významu.

Systém, projekt, mojeskolka, mojeskolka.info – pokud není v kontextu uvedeno jinak, jedná se o synonyma k výrazu „projekt mojeskolka.info“, jehož prvotní realizace by měla být výstupem této práce.

Uživatel – dle kontextu se jedná o rodiče, administrátora nebo učitele ve významu uvedeném níže. V obecném slova smyslu se jedná o jakéhokoli člověka, který aplikaci používá.

Rodič – uživatel, který má přístup do rodičovského rozhraní. Nemusí se fyzicky jednat o rodiče dítěte, může se jednat o zákonného zástupce, prarodiče či jiného rodinného příslušníka dítěte. Podstatná je možnost přístupu tohoto uživatele do rodičovského rozhraní a jeho zájem o zjišťování informací ohledně daného dítěte.

Učitel – uživatel, který má přístup do učitelského rozhraní. Může se jednat o učitele, ředitele, či jiného pracovníka školky, který má oprávnění k přístupu do systému nebo jeho příslušné části.

Administrátor – uživatel s rozšířeným přístupem k systému, jinak také správce systému, tedy pracovník školky nebo externí správce, například zaměstnanec zadavatele.

Notifikace – univerzální označení pro informaci, oznámení či upozornění, které do systému zadává školka a je zobrazováno uživateli. Může se jednat například o upozornění, že dítěti chybí přezůvky, že určitý den nebude ve školce výuka z důvodu plánovaných oprav nebo o informaci, že v daný den pořádá školka výlet a potřebné detaily o tom, jak se na to mají rodiče připravit.

Entita – obecný pojem označující objekty vystupující v systému, které v daném kontextu sdílí podobné charakteristiky či chování. Například se může jednat o rodiče, dítě, školku a třídu v kontextu jejich správy, jelikož se jedná o objekty v systému, které mají nějaké vlastní údaje (jméno, adresa, ..., ale nemusí se jednat o stejnou množinu), vazby na jiné objekty a je možné procházet jejich seznam, vytvářet je, editovat nebo mazat, tudíž k nim z našeho pohledu můžeme přistupovat podobně.

Rodičovské rozhraní – rozhraní pro uživatele v roli rodiče. Jedná se o hlavní část celého systému, ve které budou rodičům prezentovány informace o jejich dětech, případně další informace a funkce.

Učitelské rozhraní – rozhraní určené pro uživatele v roli učitele (dle definice výše), nebo administrátora (který v něm má přístupné další ovládací prvky). Je oddělené od rodičovského rozhraní, se kterým však sdílí informace. Taktéž **rozhraní školky**.

Rozbor zadání

1. **Pomocí metod softwarového inženýrství analyzujte požadavky na systém a vyberte vhodné technologie pro jeho realizaci.**

Na úvod práce je třeba vyhodnotit požadavky na systém a s ohledem na poznatky získané během NUR sestavit seznam funkčních a nefunkčních požadavků na systém, následovaný seznamem případů použití. Díky této analýze budeme mít lepší představu o požadavcích na systém, entitách, které se v něm budou vyskytovat a uživatelích, kteří s ním budou pracovat.

Avšak vzhledem k doporučením ohledně metodiky výběru vhodných technologií, zmíněným mimo jiné v [1] ještě v rámci analýzy definitivně nerozhodneme o konkrétních technologiích, které budou použity pro realizaci, pouze vybereme obecnou podobu systému a programovací jazyk pro jeho implementaci. Při výběru frameworku, ale obecně i dalších technologií, které budou použity při tvorbě prototypu, bude vhodnější, vybereme-li nejprve skupinu kandidátů, které detailněji vyhodnotíme a na základě práce s nimi, jejich obecných vlastností, ale i názorů uživatelů daného nástroje vybereme ten nejlepší pro náš projekt.

Toto rozdělení na více fází je v souladu s iterativním přístupem k vývoji, který bude přiblížen na konci této kapitoly a bude se objevovat napříč celou prací. Během prvotní analýzy tedy pouze omezíme počet možností v oblasti volby technologií a až po přípravě hrubého návrhu systému (návrh poslouží jako kontext pro detailnější srovnání technologií z hlediska vhodnosti pro jeho realizaci) vybereme v pozdější části projektu technologii pro něj vhodnou.

2. **Navrhněte architekturu systému a způsob jeho rozdělení na rozhraní pro školku a rozhraní pro rodiče, včetně rozboru chování systému v extrémní zátěži a možností přípravy na takovouto zátěž.**

   Tento proces bude umístěn právě mezi fáze prvotního a finálního výběru technologií zmíněné výše. Rozvržení jednotlivých hlavních částí – rodičovské a učitelské – je z principu fungování aplikace poměrně jasné a budeme se jím zabývat již na úvod této práce v sekci Rozdělení na jednotlivá rozhraní. Systém bude dále zkoumán z hlediska možného budoucího rozdělení na jednotlivé moduly a oddělené součásti, které by umožňovaly rozložení zátěže a zvýšení odolnosti systému vůči náporu požadavků, který by mohl nastat při potencionální adopci systému velkým počtem školek. Na závěr práce budou také provedeny zátěžové testy výsledného prototypu, které by měly pomoci analyzovat jeho slabá místa a umožnit zdokonalení systému do budoucna.

3. **Na základě předchozích bodů vypracujte návrh systému.**

   Po dokončení všech předchozích bodů, tedy i výběru vhodných technologií pro realizaci projektu, budou připraveny téměř všechny podklady

pro správnou tvorbu finálního návrhu systému před implementací jeho prototypu.

Pokud se však chceme držet metodiky návrhu uživatelského rozhraní, pak bychom měli před finálním návrhem systému nejprve dokončit návrh uživatelského rozhraní. Tím zaručíme, že se vyhneme typické chybě, kdy se vývojář snaží „napasovat“ uživatelské rozhraní na systém, který leží pod ním, a výsledek je tak zaměřen spíše na systém samotný, nežli na uživatele, pro kterého je určen. Proto při tvorbě konečného návrhu systému a jeho třídního modelu zohledníme finální návrh UI (finální alespoň v rámci této práce), který předtím vypracujeme.

Výsledkem bude návrh systému, entitní model a další specifikace, podle kterých následně bude implementován prototyp učitelské i rodičovské části.

4. **Navrhněte kvalitní uživatelská rozhraní aplikace podle metod návrhu uživatelského rozhraní.**

   Jak již bylo zmíněno výše, důležitým cílem práce je zaměření na uživatele a tvorba prostředí příjemných a vhodných pro člověka, který je skutečně bude používat (jedná se o jeden z nejdůležitějších nefunkčních požadavků). Taktéž bylo výše zmíněno, že by tato část měla být vypracována před finálním návrhem systému, aby navrhování systému bylo ovlivněno představou o jeho skutečném užívání, a nikoli obráceně.

   Při vypracování tohoto bodu tedy bude navázáno na předchozí práci z předmětu NUR, budou dokončeny detaily jednotlivých stránek s ohledem na problémy, se kterými se uživatelé při jejich testování potýkali, budou připraveny návrhy nových stránek, které se v systému vyskytují. Výstupem bude druhá verze prototypu uživatelského rozhraní a výběr vhodného nástroje pro jeho realizaci po vizuální stránce.

5. **Oba předchozí body vypracujte s ohledem na budoucí rozvoj systému, jeho údržbu a případnou distribuci na další platformy.**

   Hlavním účelem tohoto bodu je, aby při práci na projektu, tedy především při výběru technologií, návrhu architektury systému a tvorbě entitního modelu, bylo myšleno i na budoucnost projektu. Především je třeba zaměřit se na kvalitní návrh umožňující snadné úpravy systému a provádění změn v závislosti na nových požadavcích zákazníků; škálovatelnost systému do budoucna, která souvisí s odoláváním zátěži; a možnost převodu aplikací na další platformy; ale jedná se i o možnosti dalšího zdokonalování produktu, od kterých se odvíjí nutnost sbírat statistické údaje ze systému a být schopný analyzovat jeho fungování a způsob, jak s ním uživatelé pracují. Minimálně je třeba při tvorbě systému tyto požadavky zohlednit.

Tento bod tedy bude vnímán spíše jako doplňující požadavek k ostatním stěžejním bodům, ale jeho význam by měl být cítit napříč celou touto prací a bude na něj v příslušných sekcích poukazováno ve formě výhledů do budoucna.

6. **Implementujte prototyp aplikace, který bude demonstrovat její funkcionalitu a možnosti.**

   Cílem tohoto bodu je dokázat, že byl navržen skutečně realizovatelný systém a že podle předchozí analýzy a návrhu můžeme postupovat. Prototyp nebude dokonalým a bezchybným komerčním produktem, ale měl by dodržovat všechny definované zásady, splňovat funkční požadavky a umožňovat realizaci případů užití.

7. **Hotové řešení podrobte vhodným testům a vyhodnoťte výsledky testování.**

   Výsledek tvorby softwarového produktu by měl být náležitě otestován a to jak automatickými testy, tak za přítomnosti skutečných uživatelů.

   Tento bod se tedy zaměřuje na ověření funkčnosti výsledného programového prototypu a otevírá cestu pro další vývoj projektu, založený na opravě chyb nalezených ve vytvořeném produktu. Zároveň budou vypracovány automatizované testy, které budou pokrývat vybranou nejdůležitější funkcionalitu produktu a budou ověřovat správné fungování prototypu.

Cíl práce a projektu

Tato práce si dává za cíl položit kvalitní základy projektu Mojeskolka – důsledně analyzovat výchozí situaci a požadavky na systém, kvalitně sestavit jeho návrh včetně klíčového návrhu uživatelského rozhraní, vybrat vhodné technologie pro realizaci projektu a implementovat jeho prototyp, který bude následně otestován.

Rozsah celého projektu je nad rámec této práce, a tak by výstupem měla být prvotní verze systému, klasifikovatelná jako alfa verze určená pro testování na uzavřeném okruhu uživatelů. Některé funkce, klíčové pro budoucí fungování systému, nebudou implementovány (například administrátorské rozhraní pro správu školek), nebo budou implementovány v omezené míře (kupříkladu nápověda), ale do budoucna bude s nimi, nebo některými funkcemi, které mají realizovat, v projektu počítáno. Jedná se o snahu nalézt kompromis mezi omezeným časovým harmonogramem této práce a připraveností na budoucí podobu systému.

Důležitějším výstupem, než je samotná prvotní verze, by však měla být analýza a návrh systému, které poslouží pro jeho další rozvoj a další verze implementace. Zvláště důležitý je návrh uživatelského rozhraní, jelikož právě

schopnost oslovit koncové uživatele a nabídnout jim kvalitní a dobře použitelný nástroj bude rozhodovat o úspěchu, či neúspěchu projektu. Proto bude jedním z důležitých cílů této práce silné zaměření na uživatele a schopnost výsledného produktu vyrovnat se moderním aplikacím kvalitou uživatelského rozhraní a použitelností aplikace. Tyto kvality budou dále ověřeny a zkoumány testováním, na základě jehož výsledků bude moci být implementace dále zdokonalována.

Cílem celého projektu Mojeskolka by měl být dlouhodobě fungující a rostoucí systém, vycházející z výsledků zpracovaných v této práci. Samotný projekt by měl dále být součástí většího softwarového řešení, jež by mělo mít za cíl usnadnit komunikaci nejen v rámci školek, ale i dalších míst, kde komunikaci brzdí použité technologie. Kvůli těmto plánům budou na některých místech v této práci zmiňována opatření, nebo prováděny složité výběry, které se mohou zdát pro tuto práci příliš komplikované, nebo příliš orientované do budoucnosti, ale právě v tomto ohledu mají své opodstatnění.

Samozřejmě je možné, že projekt nebude úspěšný, nebo na něj z jiného důvodu nepůjde navázat, ale pokud se o dosažení dlouhodobého cíle alespoň nepokusíme a nebudeme na něj připraveni, pak se ochuzujeme o velkou část potenciálu projektu.

Iterativní přístup k analýze a návrhu

Na základě inspirace zkušenostmi jiných (mimo jiné [1], kde je zmíněno rozhodování se o správné volbě technologií až na základě skutečného kontaktu s nimi v kontextu aplikace, nikoli předem v analýze, a [2], kde je zmiňován pozitivní vliv kratších pracovních cyklů) a uvědomění si, že snaha plánovat do daleké budoucnosti většinou naráží na neprozkoumanost a proměnlivost reality, se budeme snažit činit důležitá rozhodnutí nikoli daleko dopředu, ale postupně naše rozhodnutí zdokonalovat, jak budeme mít o daném problému čím dál tím více informací.

Jak bylo naznačeno v předchozí kapitole, tento „iterativní přístup“ se projeví při volbě technologií a při návrhu systému, navíc byl využíván i při analýze požadavků a při tvorbě prvotního návrhu UI, kdy byly změny a jejich výsledky v krátkých cyklech konzultovány se zadavatelem, což mělo na projekt pozitivní efekt.

V praxi to znamená, že například volba technologií bude probíhat postupným zužováním možných kandidátů, což poskytuje možnosti detailnější analýzy těch vybraných z nich, oproti nutnosti zaobírat se příliš širokým výběrem příliš povrchně.

Stejně tak budou rozhodnutí o volbě použitých technologií prokládána postupným návrhem jednotlivých částí systému, aby daná část návrhu byla vypracována s ohledem na použité nástroje a tudíž co možná nejefektivněji.

Kapitola 1

Analýza

V této kapitole nejprve přiblížíme zdroje a způsob sběru informací nutných pro provedení analýzy, a poté rozebereme funkční a nefunkční požadavky následované případy užití, které byly sestavené na základě sebraných informací.

Na závěr této kapitoly bude proveden rozbor technologií a budou učiněna prvotní omezení možností, ze kterých budou dále v práci vybrány technologie užité pro realizaci projektu, jmenovitě vybrána podoba aplikace a programovací jazyk.

1.1 Získávání a sběr podkladů

Pro úvodní analýzu softwarového produktu je třeba získat informace o jeho účelu a způsobu, jakým by měl být používán. Proto bylo třeba na úvod projektu Mojeskolka sladit představu o projektu se zadavatelem a získat od něj co možná nejvíce informací o plánovaném produktu, i o současném stavu ve školkách. O významu a metodice získávání těchto podkladů bude hovořit tato kapitola.

Spolupráce se zadavatelem

Efektivní komunikace se zákazníkem a úspěšné vystižení jeho představy o výsledném produktu je obecně považováno za klíčovou součást úspěchu softwarového projektu. Představy zákazníka může být v určitých případech potřeba přiměřeně usměrnit v oblastech, ve kterých mu mohou chybět technologické znalosti, či korektní představa o způsobu realizace, ale je to právě spokojenost zákazníka, co z velké části určuje hranici mezi úspěšným a neúspěšným softwarovým projektem.

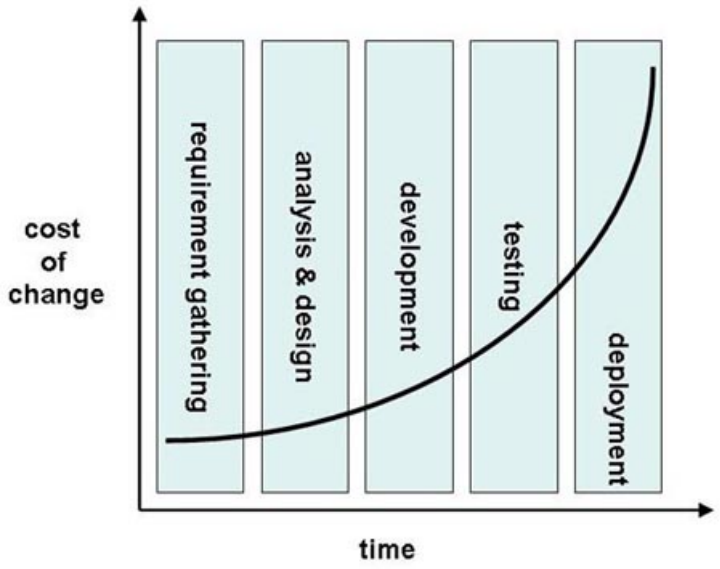
Proto byl v rámci tohoto projektu, i prací jemu předcházejících, kladen důraz na kontakt se zákazníkem, pozorné naslouchání jeho popisu produktu a diskuse nad problematickými částmi projektu.

Zároveň byla – v souladu se závěrečnou sekcí úvodu této práce – snaha o krátké a efektivní vývojové cykly doprovázené průběžnou komunikací se zákazníkem. O významu tohoto přístupu pojednává následující sekce.

Motivace pro průběžnou komunikaci

Důležitým prvkem při efektivní tvorbě softwarového produktu je nacházet problémy (ať už se jedná o nepochopení zákazníka, návrhové nedostatky, nebo chyby v programu) co možná nejdříve, aby cena, kterou budeme muset zaplatit za změny nutné pro vyřešení tohoto problému, byla co nejnižší.

Tuto myšlenku v kontextu softwarového inženýrství uvádí mimo jiné doktor Boehm ve své práci Ekonomika Softwarového Inženýrství (Software Engineering Economics), ze které pochází i následující ilustrace.



Obrázek 1.1: Rostoucí cena za provedení změny v produktu.

Tato ilustrace zachycuje vztah mezi časem, kdy je chyba odhalena (na vodorovné ose) a cenou (čas, finance, ...), kterou je nutné za opravu této chyby či nesrovnalosti zaplatit (na ose svislé).

Ačkoli se jedná o diagram z roku 1981 a existují diskuse ohledně jeho relevantnosti pro současný styl vývoje softwaru – máme modernější vývojová prostředí, možnost šíření aktualizací po internetu, což navíc mnohdy ani není potřeba, protože veškerý kód se často nachází na straně serveru, ... (touto diskusí se zabývá například článek [2]) –, pak obecně panuje názor, že stále platí, že čím dříve problém odhalíme, tím snazší a levnější je jeho řešení, ať už pro definici slova „levnější“ použijeme jakékoli metriky.

Z tohoto modelu prudce rostoucích výdajů na řešení problémů, které v projektu vznikly v raných fázích jeho vývoje, jednoznačně vyplývá, že je vhodné

investovat z počátku projektu dost času do pochopení požadavků a představ zákazníka. A proto byla komunikace se zadavatelem projektu mojeskolka vedena právě v tomto duchu.

Rozbor spolupráce

Jak bylo uvedeno výše v úvodu, se zadavatelem tedy byly vedeny pravidelné diskuse o stavu projektu, což znamená, že mu byly prezentovány jak mockupy jednotlivých stránek, tak později hotový prototyp aplikace a byly s ním probírány požadavky na funkcionalitu a jeho představa o účelu a fungování systému.

Výsledek jednání se zákazníkem

Výsledkem této spolupráce jsou níže uvedené funkční a nefunkční požadavky na systém a z nich vycházející případy užití, které byly se zadavatelem průběžně konzultovány. Výsledkem by tedy měl být vývoj již pevně definovaného produktu s jasným zadáním. Ačkoli existuje rčení, že „jediné, na co se v zadání projektu můžeme spolehnout, je jistota, že se zadání změní“, pak bychom tímto výše uvedeným přístupem měli minimalizovat množství zbytečných změn zadání v průběhu tvorby systému Mojeskolka.

1.2 Rozbor požadavků

Požadavky na systém jsou základním, ale klíčovým krokem pro prvotní uchopení koncepce produktu. Jejich cílem je hrubě specifikovat, jaký je význam daného projektu, tedy „k čemu má vlastně sloužit“. Může se zdát, že se jedná o triviální záležitost, ale právě správné pochopení, čeho by zákazník rád dosáhl a co skutečně potřebují koncoví uživatelé systému, je základním požadavkem na systém, který má plnit svou funkci a nefrustrovat uživatele.

Klíčovým problémem, na který cílí rozbor požadavků na systém, je tedy mezera mezi vývojáři, kteří netuší, jakým způsobem funguje organizace, pro kterou produkt vyvíjí a jaké procesy v ní probíhají, a uživateli produktu, kteří nevědí, co jim moderní software může nabídnout a umožnit.

Při rozboru požadavků na systém budeme vycházet jednak ze znalostí nabytých studiem na FIT ČVUT, a jednak z odborných článků k tomuto subjektu, tedy například z [3].

Požadavky se obecně dělí na dvě skupiny – funkční a nefunkční.

Funkční požadavky shrnují, jaké úkony má systém umět vykonávat. Mají za úkol překlenout výše zmíněnou mezeru a přiblížit vývojářům svět zákazníka a uživatelů budoucího systému. Druhou složku tohoto překlenutí tvoří pravidelná komunikace se zákazníkem a detailní diskuse nad jeho představou,

zmíněné výše, mající za úkol přiblížit zákazníkovi možnosti softwarového řešení. Funkční požadavky tedy říkají, *co* by měl systém dělat, ale nepopisují, *jak* by to měl dělat.

Nefunkční požadavky naopak upřesňují, jakým způsobem má být systém realizován ve smyslu požadovaného výkonu a případných omezení na jeho vývoj.

Doplňují je **návrhové cíle**, které určují, jaké charakteristiky systému by měly být vývojáři preferovány.

1.2.1 Funkční požadavky

- **Notifikace:** Systém umožní školce předávat rodičům informace týkající se jejich dětí. Tato komunikace bude primárně probíhat formou takzvaných notifikací, které budou vždy obsahovat informace k jednomu danému úkolu, události, nebo upozornění.
- **Zpětná vazba:** Systém bude umožňovat získávat od rodičů zpětnou vazbu, například k jednotlivým notifikacím. Reakci rodičů bude možné i vyžadovat, kupříkladu formou potvrzení o přečtení notifikace, čímž rodič závazně potvrdí, že je s danou informací srozuměný.
- **Zprávy:** Systém bude umožňovat komunikaci mezi rodiči a učiteli prostřednictvím zpráv, které mohou sloužit k ujasnění nesrovnalostí ohledně notifikací, upozornění na problém týkající se jednoho dítěte, nebo pro osobní komunikaci.
- **Kontaktní informace:** Systém bude umět poskytovat kontaktní informace týkající se daného dítěte a to jak rodičům kontakt na školku a učitele, tak učiteli kontakt na rodiče. Pro splnění tohoto bodu by měl systém umět uchovávat a prezentovat i adresu školky nebo další informace o ní, stejně tak jako o dalších entitách, jako je dítě, rodič a další.
- **Správa uživatelů:** Systém bude umožňovat spravovat uživatelské účty a v souvislosti s předchozím bodem tedy i jejich kontaktní informace a další podstatné údaje.
- **Správa dětí:** Systém bude umožňovat přidávání dětí do systému a jejich správu. Především v kontextu přiřazování dětí do konkrétní školky, vazby dítěte na jeho rodiče a případně přidávání dítěte do skupin, jako jsou třídy, kroužky a další.

1.2.2 Nefunkční požadavky

Vzhledem k tomu, že tato práce představuje velmi ranou fázi projektu Mojeskolka, jak je zmíněno v sekci Cíl práce a projektu, na systém zatím nejsou

kladeny zcela přesně definované nároky ohledně výkonu a dalších specifikací systému. Níže uvedené požadavky jsou tedy poněkud na hranici s obecnými cíli, ale vzhledem k jejich technické povaze jsou uvedeny v této kapitole.

- **Bezpečnost dat:** Systém bude uchovávat osobní údaje o mnoha rodičích a jejich dětech, kteréžto informace by potenciálně mohly být zneužity. Systém proto musí zaručit zabezpečení a ochranu dat v něm uložených.
- **Spolehlivost a dostupnost:** Úspěšný systém potřebuje svým uživatelům sloužit vždy, když jej potřebují. Systém Mojeskolka tedy potřebuje být spolehlivý z hlediska nepřerušovaného běhu a dostupnosti jím poskytovaných služeb, jinak k němu uživatelé nebudou mít důvěru a nebudou jej využívat. Pro maximální dostupnost a minimalizaci výpadků služeb je tedy potřeba vyhýbat se chybám v programu a připravit systém na odolávání zátěži, které bude běžným užíváním vystaven. Toho lze docílit například pomocí automatických, respektive zátěžových testů.
- **Škálovatelnost:** V případě úspěchu systému a jeho nasazení do mnoha školek bude třeba řešit přístup mnoha uživatelů ve stejnou dobu (kupříkladu ráno před odchodem do školky a večer před přípravou věcí na druhý den) při zachování výkonu a stability. Systém by tedy měl být od počátku navržen tak, aby počítal s možností vysoké zátěže a mohl jí být v budoucnu přizpůsoben vhodnými úpravami jeho částí. Tento požadavek se v některých oblastech překrývá s předchozím bodem.
- **Snadná údržba a rozvoj:** Systém by měl být navržen kvalitně a čistě, tedy v souladu s metodikami softwarového inženýrství. Cílem je, aby jeho rozšíření o nové funkce a entity byl co možná nejsnazší a případné opravy či úpravy systému byly snadno proveditelné. Zároveň by neměl vyžadovat složitou pravidelnou údržbu. Při návrhu systému by mělo být pomýšleno i na plánovanou tvorbu klienta pro mobilní operační systémy. Tento požadavek se v některých oblastech překrývá s předchozím bodem.

1.2.3 Obecné cíle

V případě projektu Mojeskolka se jedná hned o několik doplňujících požadavků, jelikož funkcionalita poskytovaná systémem není sama o sobě tolik komplexní, zatímco složitost a obtížnost do projektu vnáší silné zaměření na uživatele a potřeba konkurovat moderním aplikacím kvalitou uživatelského rozhraní a použitelností aplikace.

- **Zaměření na uživatele:** Uživatelská přívětivost systému a spokojenost jeho uživatelů je klíčem k úspěchu projektu. Tento cíl zahrnuje celkový dojem na uživatele (takzvaný UX, „user experience“), jmenovitě kvalitní návrh UI, užitečnou nápovědu a další.

- **Dostupnost a pohodlnost:** Aplikaci je možné užívat i za běžného denního provozu, tedy v situacích, které nemusí být přímo optimální pro práci se softwarovým produktem. Například při jízdě tramvají, vypravování do školky a v dalších případech. Tento bod implikuje jednak důraz na mobilní zařízení, ale také nutnost minimalizovat možnost chybných voleb uživatele („přehmatů“) a navrhnout rozhraní pokud možno použivatelné i v zhoršených podmínkách (to zahrnuje například větší tlačítka, přehledné obrazovky, málo ovládacích prvků a další. O této tématice blíže pojednává kapitola 2).

- **Skutečné zlepšení:** Systém výrazně usnadňuje komunikaci mezi rodiči a školkou. Pro obě strany je velmi jednoduchý na používání, aby motivoval jeho rychlou adopci mezi uživateli, a zároveň skutečně v uživatelích vyvolával pocit, že jim usnadňuje vzájemnou spolupráci.

- **Rychlost ovládání:** Podstatné informace prezentované uživateli jsou dostupné hned na první obrazovce systému, případně po minimálním počtu přechodů na jinou obrazovku. Cílem je umožnit uživateli velmi rychle zjistit, co mu školka sděluje, a nemuset hledáním aktuálních informací trávit zbytečně mnoho času.

- **Kvalita:** Zpracování systému by celkově mělo být na dostatečně vysoké úrovni, aby se časem mohl pro jednotlivé školky stát oficiálním komunikačním kanálem.

1.3 Rozdělení na jednotlivá rozhraní

Jak vyplývá z předchozích bodů i z rozboru zadání, systém bude muset být rozdělen na dvě částečně separovaná rozhraní a subsystémy:

Rodičovské rozhraní

To by mělo sloužit k rychlé a přehledné prezentaci informací pro rodiče a zadávání zpětné vazby od rodičů.

Učitelské rozhraní

V něm budou do systému vkládány notifikace týkající se daného dítěte, nebo skupiny dětí.Tyto notifikace se budou zobrazovat v rodičovském rozhraní uživatelům přiřazeným jako rodiče těchto dětí. V učitelském rozhraní bude dále možné spravovat entity (rodiče, učitelé, děti, třídy a případně další), tedy vytvářet je, vypisovat je, editovat je a vytvářet mezi nimi vazby, o čemž pojednávají příslušné body níže v sekci Případy užití.[1]

[1] Správa entit zahrnuje i správu uživatelských účtů.

Obecné charakteristiky

Oba subsystémy budou pracovat nad částečně sdílenou množinou dat, která si mezi sebou budou muset vyměňovat. Jinak se k nim však bude přistupovat přes rozdílná rozhraní a pro uživatele budou zdánlivě oddělené.

Tato dvě rozhraní bude muset v budoucnosti doplnit rozhraní třetí, tedy administrátorské, ve kterém bude možné vytvářet školky, centrálně spravovat uživatele, přidělovat a upravovat jim práva a vykonávat další nadřazené úkony. Jak však bylo zmíněno výše v kapitole Cíl práce a projektu, administrátorské rozhraní nebude nyní implementováno.

Dle rozdělení na rodičovské a učitelské rozhraní budou dále strukturovány i některé další části této práce.

Priorita

V rámci této práce bude kladen důraz především na tvorbu kvalitního rodičovského rozhraní, jelikož právě rodiče jsou primárními koncovými uživateli systému, které projekt chce oslovit. Zároveň musí být rozhraní pro rodiče více inovativní a nápadité, zatímco rozhraní pro učitele může vycházet ze struktury a vzhledu klasického CMS (Content Management System, tedy aplikace určené pro správu obsahu webových informačních systémů).

1.4 Případy užití

Případy užití (takzvané „usecases“) reprezentují výchozí bod pro řízení designu a vývoje orientovaného na koncového uživatele. Poskytují detailní pohled, jak bude produkt užíván při reálném nasazení, čímž dále rozšiřují principy přiblížení vývojářů k zákazníkům, které byly popsány výše v rozboru požadavků.

Případy užití zachycují průběh nějaké konkrétní činnosti uživatele, typicky jsou pojmenovávány podle úspěšného závěru této činnosti, a popisují každý krok daného procesu včetně vstupů, výstupů, alternativ, chyb a výjimek. Typicky jsou psány formou akce aktéra – očekávaná reakce (odpověď) systému a možné výsledky.

Při tvorbě případů užití z informací poskytnutých zákazníkem budeme vycházet z [4]. V tomto dokumentu je zároveň zmíněno, že ne vždy je třeba všech detailů uvedených v doporučované šabloně. Proto budeme užívat pouze některé podstatné části z šablony v dokumentu uvedené a některé části budou vypouštěny, pokud jsou jasné z kontextu (například u méně složitých případů užití nebude uveden „Cíl“, pokud je triviální – typicky pokud se přímo shoduje s názvem daného případu užití).

Pro zlepšení struktury případů užití budeme vycházet především z dokumentu [5]. Ten rozebírá možnosti, jak strukturovat případy užití tak, aby

bylo dosaženo maximální přehlednosti, znovupoužitelnosti a možnosti pozměnit případy užití na základě nových informací.

Toho dosáhneme především užíváním vztahů mezi jednotlivými případy užití, tedy možnosti rozšiřovat (extend), zahrnovat (include) a dědit od (inherit from) jiných scénářů.

Dokument mimo jiné zmiňuje, že pro jednoduché alternativy v průběhu případu užití můžeme užívat blok „alternativní postup“ (v této práci označován slovem „Alternativy“), ale pro složitější alternativní možnosti postupu bychom měli tento postup vyčlenit do samostatného případu užití, který bude rozšiřovat ten původní. Proto budeme pro jednoduché odbočky užívat sekci „Alternativy“ a pro složitější vyčleníme daný alternativní průběh do samostatného případu užití.

Cílem těchto nástrojů a strukturálních změn je co nejvíce zjednodušit případy užití, nikoli je komplikovat, proto je budeme užívat pouze ve vhodné míře.[2]

1.4.1 Seznam případů užití

Případy užití společné pro rodičovské rozhraní i učitelské rozhraní

1. Přihlášení do systému
2. Vyřešení problémů při přihlášení
3. Napsání zprávy uživateli či školce
4. Opravení chybného vstupu

Případy užití pro rodičovské rozhraní

1. Rychlé zjištění aktuálních úkolů a poznámek týkajících se dětí daného uživatele
2. Zobrazení detailů notifikace
3. Dohledání kontaktních informací o školce, či učiteli
4. Zobrazení všech notifikací daného dítěte
5. Potvrzení přečtení notifikace
6. Skrytí notifikace, se kterou už je uživatel srozuměný

[2] V dokumentu je také uvedeno, že bychom měli nejprve sestavit první návrh případů užití, a pak teprve mezi nimi vytvářet výše uvedené vztahy a vazby. Tohoto postupu bylo dodrženo a případy užití byly po prvotním sepsání ještě přeformulovány a provázány.

Případy užití pro učitelské rozhraní

1. Vložení notifikace do systému
2. Dodatečná změna notifikace
3. Přidat entitu (uživatele, dítě, třídu) do systému
4. Předání informací o účtu jeho majiteli
5. Upravení detailů instance dané entity

1.4.2 Společné případy užití

Přihlášení do systému

Cíl: umožnit uživateli co možná nejjednodušeji začít používat systém po přidělení účtu.

Průběh

1. Školka poskytne uživateli přihlašovací údaje do systému.
2. Uživatel vstoupí na stránky systému.
3. Uživatel se pokusí přihlásit.
4. Uživatel se úspěšně přihlásí do aplikace.
5. Systém mu předloží úvodní stránku.

Alternativy

5.a. Uživatel se do systému přihlásil poprvé.

5.a.1. Systém zobrazí uvítací obrazovku s úvodními informacemi pro nového uživatele.

5.a.2. Uživatel si pročte předložené informace. V okamžiku, kdy je spokojený, přejde na hlavní stranu.

Poznámka: Týká se učitelského i rodičovského rozhraní.

Vyřešení problémů při přihlášení

Příčina: Uživatel se není schopen do systému přihlásit. To může být způsobeno chybou na straně školky (špatně vytvořila účet), systému (chyba programu či úložiště) nebo samotného uživatele (zapomenuté heslo, přihlašovací jméno či obojí).

Cíl: Umožnit uživateli přístup do aplikace.

Rozšiřuje: „Přihlášení do systému“. Navazuje v bodě 3. „Uživatel se pokusí přihlásit“ v případě, že toho uživatel není schopen.

Průběh

1. Systém odmítne přihlášení uživatele, který si nepamatuje své heslo, nebo e-mail pro přihlášení.
2. Uživatel přejde na stránku pomoci s přihlášením.
3. Uživatel si přečte informace o tom, jak postupovat v případě problémů s přihlášením.
4. Uživatel vyplní formulář pro zaslání hesla.
5. Systém pošle uživateli na uvedený e-mail požadované údaje společně s varováním jak postupovat, pokud o ně nezažádal.
6. Uživatel opětovně postupuje dle scénáře „Přihlášení do systému“.

Alternativy

1.a. Uživatel si pamatuje své heslo i e-mail, ale chyba je na straně systému, nebo školky, postup je však stejný až do bodu 4.

4.a. Uživatel napíše na uvedenou e-mailovou adresu žádost o vyřešení problému.

5.a. Školka (či správce systému) uživatele kontaktuje na poskytnuté číslo či e-mail, aby mu bylo umožněno přihlášení do systému.

3.b. Uživatel pouze udělal chybu při přihlašování (měl zapnutý Caps Lock, nebo se jednalo o jiný triviální problém).

3.b.1. Uživatel se opětovně pokusí přihlásit.

3.b.2. Pokus je tentokrát úspěšný a uživatel je přihlášen.

Napsání zprávy uživateli či školce

Příčina: Uživatel by rád s jiným uživatelem hovořil o informaci, dotazu, či problému, který není zcela nutné řešit ihned (velmi aktuální události (nemoc, nečekaná absence dítěte, ...) by měly být řešeny pomocí přímého telefonního kontaktu na učitele, či školku, nikoli pomocí zpráv).

Průběh

1. Uživatel přejde na stránku zpráv.
2. Systém mu zobrazí výpis odeslaných i přijatých zpráv a umožní napsání nové zprávy.
3. Uživatel zvolí novou zprávu, vyplní požadovaná pole (předmět, text, komu, ...).
4. Uživatel nechá zprávu odeslat.
5. Systém odešle zprávu vybranému příjemci a potvrdí uživateli úspěšné odeslání zprávy.

Alternativy

5.a. Pokud uvedený příjemce nebyl nalezen, systém nedovolí zprávu odeslat.

5.a.1. Uživatel opraví pole příjemce a pokračuje bodem 4.

Poznámky: Koncept zpráv by měl představovat obdobu interní e-mailové komunikace, která bude přehledně zachycena a prezentována uvnitř aplikace, tedy v kontextu, kde uživatele zajímá (a není ztracená v e-mailové schránce mezi mnoha různými zprávami z různých zdrojů).

Opravení chybného vstupu

Příčina: Při užívání formulářů vyplňovaných uživatelem je nutné kontrolovat vstup uživatele, aby nebyla poškozena integrita stavu systému.

Průběh

1. Uživatel udělal chybu při vyplňování formuláře.
2. Systém při kontrole formuláře zjistí chybu ve formuláři.
3. Systém upozorní uživatele a ukáže, kde se problém nachází.
4. Uživatel opraví chybu.
5. Pokračování v činnosti, jejíž průběh přerušila chyba uživatele.

Poznámka: Tento scénář je určen pro zahrnutí do dalších scénářů.

1.4.3 Případy užití pro rodičovské rozhraní

Rychlé zjištění aktuálních notifikací týkajících se dětí daného uživatele

Výchozí pozice: Uživatel je přihlášen v systému.

Cíl: Rychle a stručně zjistit, co by měl v následujících dnech uživatel očekávat, na co se připravit apod. ohledně svých dětí.

Průběh

1. Uživatel přejde na hlavní stranu aplikace.
2. Systém uživateli hned na hlavní straně prezentuje všechny důležité informace (ve formě notifikací) pro nadcházející časové období o všech jeho dětech.

Alternativy

1.a. Uživatel chce zobrazit pouze informace týkající se jednoho dítěte.

1.a.1. Uživatel přejde na stránku daného dítěte.

1.a.2. Systém poskytne všechny aktuální informace týkající se tohoto dítěte.

Poznámky: Jedná se o jednu z hlavních a nejdůležitějších funkcí systému, proto by na ni měl být kladen náležitý důraz (Vychází hned z několika cílů uvedených výše v Rozbor požadavků).

Kvůli maximální přehlednosti a zároveň co nejvyšší kvalitě poskytovaných informací je otázkou, co by měl znamenat výraz „aktuálních informací“, tedy co přesně by mělo být přednostně zobrazováno na hlavní straně. Touto tématikou se detailně budeme zabývat níže v subsekci 2.8.3.

Zobrazení detailů notifikace

Výchozí pozice: Uživatel má zobrazený seznam notifikací (na stránce dítěte, na hlavní straně, či jinde, k přechodu na tuto stránku slouží příslušné scénáře).

Průběh

1. Uživatel si chce přečíst detaily konkrétní zobrazené notifikace.
2. Uživatel si otevře danou notifikaci kliknutím do jejího těla.
3. Systém zobrazí detaily vybrané notifikace.

Dohledání kontaktních informací o školce, či učiteli

Příčina: Uživatel by potřeboval zavolat učiteli daného dítěte, či zavolat přímo do školky. Jedná se o řešení akutních situací, kde komunikace prostřednictvím zpráv nestačí.

Výchozí pozice: Uživatel je přihlášen v systému.

Průběh

1. Uživatel přejde na stránku kontaktů týkajících se vybrané entity.
2. Systém prezentuje výpis relevantních informací (telefon na učitele daného dítěte, telefon do školky, ...).
3. Uživatel si dané číslo zkopíruje do schránky, nebo, pokud je na mobilním zařízení, na ně může ihned volat.

Alternativy

3.a. Uživatel nechce telefonní kontakt, ale kupříkladu korespondenční adresu pro zaslání dokumentů poštou, e-mailový kontakt, nebo jiné informace související s kontaktováním dané entity.

3.a.1. Uživatel si na stránce kontaktů dohledá požadované informace a opíše si je, či zkopíruje dle potřeby.

Zobrazení všech notifikací daného dítěte

Příčina: Uživatel by rád viděl i úkoly a informace ze vzdálenější budoucnosti, nebo by se naopak chtěl podívat na historii notifikací týkajících se daného dítěte.

Průběh

1. Uživatel přejde na takzvanou „nástěnku“ dítěte.
2. Systém mu zobrazí notifikace od nejaktuálnějších, až po ty nejvíce budoucí.

Alternativy

3.a. Uživatel by rád vyděl i skryté, či staré notifikace.

3.a.1. Uživatel si nechá zobrazit nastavení nástěnky.

3.a.2. Uživatel zvolí zobrazovat i staré či zobrazovat i skryté notifikace.

3.a.3. Systému mu poskytne navíc i vybrané skupiny notifikací.

3.b. Uživatel pouze udělal chybu při přihlašování (měl zapnutý Caps Lock, nebo se jednalo o jiný triviální problém).

3.b.1. Uživatel se opětovně pokusí přihlásit.

3.b.2 Pokus je tentokrát úspěšný a uživatel je přihlášen.

Potvrzení přečtení notifikace

Zahrnuje: „Zobrazení detailů notifikace“

Průběh

1. Rodič si zobrazí detail dané notifikace a pozorně si ji přečte.
2. Na stránce detailů notifikace rodič přejde k sekci určené k potvrzení, že danou notifikaci přečetl.
3. Rodič potvrdí, že notifikaci přečetl.
4. Systém informaci uloží a náležitě zpracuje, po čemž o tom uživatele informuje.

Poznámka: Souvisí s alternativním průběhem případu užití učitelského rozhraní „Vložení notifikace do systému“, kdy učitel vyžaduje potvrzení o přečtení.

Skrytí notifikace, se kterou už je uživatel srozuměný

Příčina: Uživatel se rozhodne, že danou notifikaci nepotřebuje dále zobrazovat, protože už se podle ní zařídil (například koupil chybějící přezůvky, připravil vše potřebné na školní výlet, . . .).

Cíl: Nerozptylovat uživatele notifikacemi, se kterými už je srozuměný.

Zahrnuje: „Zobrazení detailů notifikace“

Průběh

1. Uživatel si zobrazí detail dané notifikace a pozorně si ji přečte.
2. Uživatel se rozhodne, že danou notifikaci již nechce zobrazovat, a tak tuto možnost zvolí v rámci příslušného formuláře.
3. Systém daný požadavek uloží a notifikaci již uživateli dále nezobrazuje.

Výjimky

V okamžiku, kdy už je notifikace na zítřejší, nebo dnešní den, je uživateli znovu zobrazována v méně výrazné podobě (například šedivě na konci výpisu ostatních notifikací). Cílem této výjimky je zamezit nechtěnému opomenutí notifikace, kterou uživatel četl a nechal si přestat zobrazovat již před delší dobou.

1.4.4 Případy užití učitelského rozhraní

Vložení notifikace do systému

Výchozí pozice: Učitel je na hlavní stráně učitelského rozhraní.

Průběh

1. Učitel vybere třídu, kroužek, nebo skupinu žáků, pro kterou chce notifikaci přidat.

2. Učitel vyplní formulář pro tvorbu nové notifikace.

3. Učitel nechá vložit notifikaci do systému.

4. Systém vytvoří novou notifikaci, poskytne ji rodičům všech dětí, kterých se týká a informuje o tom učitele.

Alternativy

2.a. Učitel v kroku dva zvolí možnost „Vyžadovat potvrzení o přečtení notifikace“, čímž vytvoří notifikaci, jejíž přečtení budou muset rodiče potvrdit.

4.a. Systém bude od všech uživatelů. kterým je notifikace určena, vyžadovat potvrzení o přečtení této notifikace.

Dodatečná změna notifikace

Příčina: Učitel zjistí, že je potřeba pozměnit obsah notifikace, která už byla umístěna v systému (a kterou již někteří rodiče mohli označit za přečtenou, nebo si ji mohli nechat skrýt).

Cíl: Zaručit, že se informace o změně dostane ke všem uživatelům.

Průběh

1. Učitel změní informace obsažené v dané notifikaci.
2. Systém propaguje nově změněnou informaci ke všem uživatelům, kteří by ji měli obdržet.
3. Systém resetuje všechna nastavení „Tuto notifikaci již nezobrazovat“, potvrzení o přečtení a další nastavení související s touto notifikací.
4. Systém k této notifikaci přidá speciální upozornění, že notifikace byla změněna.
5. Systém informuje učitele o úspěšném provedení požadovaných změn.

Alternativy

1.a. Učitel se rozhodne, že změna není dost důležitá na to, aby na ni museli být rodiče upozorňování (oprava překlepu, reformulace věty, ...)

2.a.1. Učitel zvolí, že tato změna nezasahuje do významu poskytovaných informací.

2.a.2. Systém pouze změnu uloží a ta je propagována k uživatelům bez dalších efektů.

Přidat entitu (uživatele, dítě, třídu) do systému

Příčina: Pro správné fungování celé aplikace je potřeba vytvořit strukturu tříd v dané školce, které obsahují děti mající přiřazené rodiče, kterým se budou zobrazovat notifikace týkající se jejich dětí, tedy i celých školek a tříd, do kterých děti chodí. (V tomto kontextu tedy výrazem „entita“ myslíme třídy, děti, rodiče i učitele, kteréžto entity budou vytvářeny.)

Cíl: Vložit do systému entitu s požadovanými parametry.

Výchozí pozice: Uživatel je přihlášen v rozhraní pro školku.

Zahrnuje: „Opravení chybného vstupu“

Průběh

1. Uživatel přejde na stránku pro správu instancí příslušné entity.
2. Uživatel zvolí vytvořit v systému novou instanci této entity.
3. Systém zobrazí formulář pro tvorbu nové instance této entity.

4. Uživatel vyplní příslušný formulář a nechá jej uložit.
5. Systém formulář zkontroluje (případně užití „Opravení chybného vstupu“).
6. Systém formulář uloží, vytvoří novou entitu a informuje o tom uživatele.
7. Systém přejde na stránku se seznamem instancí této entity.

Předání informací o účtu jeho majiteli

Výchozí pozice: Uživatel je přihlášen v rozhraní pro školku a má administrátorské oprávnění.

Aktéři: Uživatel, nazývaný „administrátor“, a systém.

Zahrnuje: „Přidat entitu do systému“

Průběh

1. Administrátor přejde na stránku pro přidání nového uživatele.
2. Systém zobrazí formulář dle scénáře „Přidat entitu do systému“, ve kterémžto formuláři bude možno zvolit, že systém má po vytvoření účtu uživatele informovat o jeho novém účtu v systému Mojeskolka.
3. Administrátor vyplní formulář a zvolí, že uživatel má být dle předchozího bodu informován.
4. Systém vytvoří příslušný účet a pošle informace o tomto účtu jeho majiteli.

Alternativy

3.a. Administrátor nechce informace uživateli posílat, ale raději by je sdělil osobně přes učitele. Proto nezvolí, že by uživatel měl být o účtu informován automaticky.

4.a. Systém pouze vytvoří účet a uživatele o účtu nikterak neinformuje.

Upravení detailů instance vybrané entity

Příčina: Je třeba editovat údaje o vybrané entitě, což zahrnuje i přiřazení dítěte do třídy, rodiče k dítěti a další úkony.

Zahrnuje: „Opravení chybného vstupu“

Průběh

1. Uživatel přejde na stránku pro správu instancí příslušné entity.
2. Uživatel vybere příslušnou instanci této entity.
3. Systém zobrazí detaily o této entitě a možnost je editovat.
4. Uživatel edituje příslušnou entitu a nechá nový stav uložit.
5. Systém formulář zkontroluje (případně užití „Opravení chybného vstupu“).
6. Systém uloží nové informace o dané entitě a upraví příslušné vztahy v systému.

1.5 Volba platformy

Nyní, když máme na základě předchozích kapitol poměrně ucelenou představu o tom, jakým způsobem by systém měl být užíván, je na místě učinit základní rozhodnutí o jeho technické podobě.

Úvodním krokem bude volba platformy, pro kterou bude první verze aplikací (rodičovské a učitelské) vyvíjena a na které bude probíhat prvotní nasazování a testování systému. Z povahy projektu vyplývá, že základem architektury bude model klient-server, jelikož systém vyžaduje přenos informací mezi školkou a rodiči, kteréžto informace musí být uloženy na pevném úložišti dostupném pro obě strany, tedy nějakém serveru. Zároveň musí být informace současně přístupné pro mnoho různých rodičů, kteří však mezi sebou žádné informace nesdílejí, což vylučuje jiné modely komunikace, například peer-to-peer.

Technologie použité pro realizaci serveru nejsou pro tuto práci tolik zásadní, jelikož se v rámci projektu primárně zaměřujeme na koncového uživatele.[3].

Avšak co se realizace klienta týče, situace je odlišná. Volba platformy totiž zároveň určuje, jakou podobu bude klientská aplikace na počátku mít – zda se bude jednat o lokálně instalovaného „tlustého“ klienta, nebo „tenkého“ webového klienta bez potřeby lokální instalace – což bude mít zásadní vliv na prvotní fáze nasazování projektu. Podoba klienta, která vyplývá z volby dané platformy, bude vždy uvedena u příslušného rozboru platformy.

[3] Architektura a návrh serveru však pro tuto práci podstatné jsou a budou řešeny v souvislosti s učitelským rozhraním, které je na ně pevněji vázáno, zvláště kvůli správě uživatelů a entit. O této tematice bude více uvedeno níže, v kapitole 3

1.5.1 Kritéria

Pro korektní výběr vhodné platformy bude třeba určit, na základě jakých parametrů budeme tuto volbu provádět. Snahou je nezaujatě zhodnotit možnosti a vybrat tu nejlepší, pročež potřebujeme jednotnou míru právě v podobě kritérií.

Jednotlivé možnosti výběru platformy budou vzhledem k danému kritériu ohodnoceny na stupnici od $+2$ $(++)$ do -2 $(--)$, v závislosti na tom, jak určitá možnost vyhovuje danému kritériu.

Zároveň nemají všechna kritéria stejnou váhu, jelikož některé charakteristiky ovlivňují projekt více, než jiné. V základu mají všechna kritéria váhu rovnu jedné, v případě jiné váhy bude tento koeficient u daného kritéria uveden. Kritéria vychází do značné míry z rozboru požadavků a případů užití.

Rozšířenost platformy: Počet uživatelů této platformy, které aplikace potenciálně může oslovit. Jedná se o velmi důležitý aspekt, jednak kvůli rychlosti a plynulosti adopce produktu a jednak kvůli možnosti testovat aplikaci od počátku na co nejširším vzorku uživatelů. Proto má tento parametr zvýšenou váhu.
Váha = 2

Instalace a aktualizace: Není důležitý jen potenciální počet uživatelů, které aplikace může oslovit, ale i to, kolik z nich bude ochotných si do svého systému instalovat zvláštní program pro komunikaci se školkou, pokud bude zvolená platforma vyžadovat instalaci speciální aplikace pro užívání systému Mojeskolka. Tento parametr tedy primárně cílí na otázku, kolik se dá z potenciální množiny očekávat těch reálných uživatelů. S tímto bodem souvisí nejen snadnost instalace z pohledu uživatele, ale i možnost aktualizace programu z hlediska vývojáře, jelikož jsou tyto parametry provázány.

Dostupnost a pohodlí: Možnost přistupovat k aplikaci pohodlně kdykoli a odkudkoli je důležitá pro plné využívání její funkcionality, což se spíše pojí s mobilními zařízeními. Zároveň pohodlí spíše evokuje představu stolního počítače s velkou obrazovkou a plnohodnotnou klávesnicí, nežli obraz malého mobilního zařízení. Zvláště znatelný je tento rozdíl z hlediska psaní zpráv, nebo prohlížení většího množství notifikací týkajících se daného dítěte. Vzhledem k velkému dopadu na celkovou spokojenost uživatelů a použitelnost systému má tento parametr zvýšenou váhu.
Váha = 2

Znalosti vývojáře: Pro rychlý a kvalitní vývoj aplikace pro dané prostředí jsou důležité také znalosti a zkušenosti vývojáře s danou platformou. V rámci tohoto parametru budeme hodnotit právě tento atribut v kontextu znalostí

autora této práce. Ačkoli se jedná o částečně subjektivní hodnocení, jeho význam není možné ignorovat.

Interakce se systémem: Pro maximální spokojenost uživatele s aplikací je také důležitá možnost této aplikace interagovat s operačním systémem, ve kterém je spuštěná. To umožňuje například snadné volání na kontaktní čísla uvedená v aplikaci, upozornění uživatele na nové zprávy, či notifikace v Mojeskolka, nebo další úkony. Různé způsoby realizace, tedy cílení na různé platformy, však má dopad na schopnost interakce se systémem (například čistě webová aplikace spuštěná v prohlížeči obecně bez dalších nástrojů nemůže na většině systémů přistupovat k jejich funkcím, jak je blíže rozvedeno níže).

Persistentní multimediální obsah: Různé způsoby realizace klienta se liší také množstvím neměnného multimediálního obsahu, který mohou obsahovat (obrázky, videa a další), například u webových aplikací jsme limitování klientovým připojením, u mobilních platforem zas pamětí zařízení (bližší specifikace jsou uvedeny níže).

Aplikace do budoucna počítá s instruktážním videem pro nové uživatele, ilustrativními obrázky a na vybraných zařízeních by mohla být i možnost lokálně cachovat (dočasně ukládat, aby nebylo třeba je opakovaně stahovat) obrázky v aplikaci (obrázky jednotlivých dětí, učitelů, školky, atd.), ale dá se očekávat, že by se nemělo jednat o objemné soubory. Systém pro své fungování nepotřebuje s uživatelem sdílet příliš velký objem persistentních dat, která by bylo možné jednou nainstalovat a v budoucnu neměnit a proto má toto kritérium sníženou váhu.
Váha = $\frac{1}{2}$

Forma hodnocení: U každé u možnosti výběru platformy a klienta budou jednotlivá kritéria slovně hodnocena a v závěrečné sekci této kapitoly bude uvedena stručná tabulka s vyhodnocením jednotlivých alternativ.

1.5.2 Výčet možností:

V úvahu přichází následující varianty platforem a klientů:

1. **Tenký webový klient**

   Aplikace poběží pouze ve webovém prohlížeči, s klientem bude tedy komunikovat pomocí jazyků HTML, CSS a Javascript. Jedná se o moderní pojetí aplikace, které je založené na skutečnosti, že webový prohlížeč je instalován prakticky na každém zařízení, které uživatelé běžně používají, a aplikace přístup k internetu potřebuje na každý pád.

   Díky tomu může toto řešení oslovit maximální počet uživatelů a má minimální nároky na instalaci a šíření aktualizací (protože u klienta se

žádný kód aplikace Mojeskolka nenachází). Zároveň se jedná o ideální kompromis z hlediska dostupnosti a pohodlnosti užívání systému, jelikož v případě dodržení správných zásad návrhu UI tento klient plně využívá výhod mobilních i stolních zařízení (o takzvaném responzivním návrhu UI bude více uvedeno v sekci 2.3.1). S vývojem webových aplikací má autor této práce nadprůměrné zkušenosti, ale jeho znalosti jsou omezené pouze na tvorbu ve vybraných jazycích a pro vybrané systémy, proto zvláště v oblasti vhodných frameworků jeho znalosti poněkud pokulhávají.

Naopak možnosti interakce se systémem a množství multimediálního obsahu jsou u webové aplikace značně omezené. Samotná interakce se systémem obecně není možná bez spolupráce s nativní aplikací pro to určenou (například aplikace Pushover umožňuje notifikovat uživatele, ale i tyto možnosti jsou značně omezené). Stejně tak je třeba se u webové aplikace vyhnout objemným souborům, tedy po stránce neměnného multimediálního obsahu webová aplikace pokulhává, což však není pro projekt takovou překážkou.

2. **Tlustý desktopový klient**

   Aplikace by byla stažitelná z internetu a instalovala by se do cílového počítače, odkud by komunikovala se serverem a získávala od něj potřebná data. Multimediální obsah by byl uložen lokálně, stejně tak, jako případná aplikační logika (která je však v Mojeskolka poměrně jednoduchá).

   Jedná se z hlediska současného vývoje na poli informačních technologií o poněkud zpátečnické řešení, ale přesto může poskytovat určité výhody.

   Platforma je to sice velmi rozšířená, stolní počítače se nachází prakticky v každé domácnosti, avšak mnoho uživatelů, zvláště technologických nadšenců, kteří budou nejvíce ochotní novou technologii ve školkách adoptovat, už je zvyklých provádět většinu podobných úkonů na mobilních zařízeních, pročež by desktopové řešení oslovilo o něco méně uživatelů.

   Dostupnost aplikace je sice značně omezená, jelikož se v této variantě jedná o aplikaci pro stolní počítače, ale na druhou stranu je užívání systému na desktopových zařízeních velice pohodlné, jelikož se na velký monitor vejde výrazně více informací a jsou lépe čitelné. Bohužel však nevýhoda nutnosti užívat aplikaci pouze na stolních počítačích převažuje nad výhodami pohodlné práce se systémem na těchto zařízeních Stejně tak nutnost instalace komplikující šíření systému není silnou stránkou tohoto řešení.

   Avšak znalosti autora práce jsou z daných možností u této platformy nejsilnější, což poskytuje výhodu, stejně tak jako skutečnost, že neměnný

multimediální obsah není třeba opakovaně stahovat, jako u webového řešení, a máme na něj dost místa na pevném úložišti, na rozdíl od mnohých mobilních zařízení. Desktopová aplikace také umožňuje užívat systémových nástrojů pro informování uživatele, i když je v tomto případě význam interakce se systémem poněkud omezen, pokud uživatele můžeme o událostech v aplikaci informovat pouze v okamžiku, kdy je doma a na svém stolním počítači.

3. **Klient pro mobilní operační systém**

   Aplikace by byla nainstalována do mobilního zařízení uživatele, kde by podobným způsobem, jako webový klient, umožňovala práci se systémem, navíc by však měla možnost interakce s hostujícím operačním systémem, tedy by například mohla uživatele informovat o nových zprávách, či notifikacích, které vyžadují jeho pozornost.

   Jedná se o velmi rozšířenou platformu, avšak její nevýhodou je rozdělení na různé operační systémy, přičemž aplikace by z počátku mohla být implementována pouze pro jeden mobilní OS, což limituje množství počátečních uživatelů. Co se možností užívání týče, pak mobilní telefon má u sebe neustále většina potenciálních uživatelů a mnozí by mohli k aplikaci přistupovat pomocí tabletů s větším displejem, ale mnoho uživatelů stále preferuje pohodlné ovládání pomocí stolního počítače, zvláště při psaní delšího textu. Aplikace by byla umístěna na úložišti specifickém pro daný systém (AppStore, PlayStore, ...), pročež by distribuce a aktualizace aplikace byly jednodušší, než u desktopového řešení, ale stále je potřeba instalace a aktivní účasti uživatele.

   Zkušenosti autora této práce s vývojem pro mobilní OS jsou značně omezené (jedná se pouze o teoretické znalosti), což tuto platformu poněkud znevýhodňuje. Možnost práce s neměnným multimediálním obsahem je taktéž poněkud omezená, u mnoha zařízení je velikost trvalého úložiště stále nedostačující a uživatelé budou při instalaci aplikace Mojeskolka hledět i na její velikost.

1.5.3 Vyhodnocení

Tabulka 1.1: Shrnutí srovnání platforem

	Rozšířenost platformy	Instalace a aktualizace	Dostupnost a pohodlí	Znalosti vývojáře	Interakce se systémem	Neměnný multimed. obsah
Tenký webový klient	++	++	++	+	−−	−−
Tlustý desktopový klient	+	−−	−	++	+	++
Mobilní klient	+	−	+	−	++	−

Součet hodnocení

- Tenký webový klient: 8 bodů
- Mobilní klient: 3,5 bodu
- Tlustý desktopový klient: 2 body

Ve výsledku tedy ze srovnání jednoznačně nejlépe vychází tenký webový klient a proto bude prvotní verze rodičovského rozhraní implementována právě pro webovou platformu. (Je dobré připomenout různé váhy jednotlivých kritérií, které na výsledné počty bodů mají vliv, ale i bez nich by se stále tenký webový klient nacházel mezi vítězi.)

Rodičovské a učitelské rozhraní

Ačkoli se tato kritéria zaměřují spíše na rodičovské rozhraní, není pádného důvodu vyvíjet učitelské rozhraní pro jinou platformu – jeho primární funkce budou taktéž nejlépe ovládány na stolním počítači (psaní textu notifikací i manipulace s instancemi entit v dané školce využijí dobře vstupu z klávesnice a myši, stejně tak výstupu na velký displej) a jedná se i o poměrně standardní formu moderních aplikací pro školy a školky (o této tématice více níže v návrhu UI, v sekci 2.2).

Budoucnost

Zvoleným řešením prakticky pokrýváme stolní počítače na velmi dobré úrovni, ale z hlediska mobilních zařízení bychom stále postrádali některé výhody nativních aplikací. Proto by v dalších fázích projektu mělo dojít i na tvorbu aplikací pro mobilní operační systémy, kteréžto aplikace však nebudou v rámci této práce vytvářeny.

Ale v zájmu splnění zadání bychom s tímto předpokladem měli v průběhu práce počítat. Jeho vliv by se měl projevit v návrhu systému a jeho částí, který by měl reflektovat připravenost na budoucí přesun některých součástí do aplikace pro mobilní operační systém, což implikuje potřebu pro správnou modularitu aplikace a nízkou provázanost komponent.

To je však v souladu s obecně platnými doporučeními pro návrh a vývoj softwarového produktu, takže to vývoj aplikace nijak neomezuje, naopak to pouze vynucuje korektní postupy.

1.6 Volba programovacího jazyka

Poté, co jsme úspěšně vybrali vhodnou platformu, na kterou bude projekt cílit, držíme v rukou další informace potřebné pro to, abychom podobným způsobem vybrali i programovací jazyk, ve kterém bude aplikace implementována.

Je dobré míti na mysli, že výběr programovacího jazyka zároveň určuje množinu frameworků a dalších nástrojů, ze kterých budeme moci dále v průběhu práce vybírat, což má velmi zásadní dopad na tvorbu prototypu. Při tvorbě této sekce byly proto předběžně zkoumány a analyzovány možné frameworky a nástroje, které jednotlivé programovací jazyky poskytují, kterážto předběžná analýza bude zohledněna níže v kritériu „dostupné technologie“. Tento krok a mnohé další jsou snahou o učinění co možná nejpřesnějšího závěru této kapitoly, abychom pozitivně ovlivnili výsledný prototyp a prvotní verzi aplikace, i budoucí směr rozvoje projektu, které všechny jsou na této volbě do znační míry závislé.

1.6.1 Kritéria

Podobně, jako u předchozího výběru platformy i při volbě programovacího jazyka budeme náš výběr zakládat na kritériích, která by nám měla pomoci nalézt nejvhodnější kandidáty. Stejně tak platí identická pravidla pro hodnocení jednotlivých alternativ a váhu kritérií, jaká byla použita v předchozí sekci.

V případě výběru programovacího jazyka je však problémem získání objektivních a přesných údajů ohledně jednotlivých sledovaných parametrů. Názory se různí, pevná data je obtížné sehnat a jejich důvěryhodnost je mnohdy zpochybnitelná, už jen proto, že na podobné téma neexistuje vhodná literatura zahrnující postačující množství detailů, jelikož dříve, než by případný autor stihl knihu pojednávající o vhodných technologiích sepsat a vydat, informace v ní uvedené už by nebyly aktuální.

Hodnocení u jednotlivých položek bude tedy založeno na dohledaných informacích agregovaných z článků dostupných on-line, z názorů vyučujících, odborníků a z dalších dostupných zdrojů.

Největší váhu bude tedy mít jediné hledisko, které je pevně podložitelné – zkušenosti autora této práce s daným jazykem.

Znalosti vývojáře

Pro úspěch této práce a pro budoucnost projektu bude vhodnější, pokud aplikace bude vyvíjena v programovacím jazyce, ve kterém má autor řečeného projektu dostatečné zkušenosti a znalosti. Ačkoli jsou znalosti a zkušenosti z velké části přenositelné mezi jednotlivými jazyky, rychlost a efektivita vývoje v daném jazyce je – přirozeně – poznamenána zkušenostmi vývojáře. Ze všech výše uvedených důvodů (i kvůli relativní „pevnosti“ tohoto parametru, zmíněné výše) má toto kritérium zvýšenou váhu.

Váha = 2

Cena

Pořizovací cena nutných nástrojů a cena vývoje v daném programovacím jazyce taktéž není nezanedbatelnou položkou. Jednak z hlediska počáteční investice do projektu, jednak z hlediska placení dalších vývojářů, kteří by na rozvoji, pokračování, nebo údržbě projektu měli v budoucnu pracovat. Ohledně této charakteristiky se dají sehnat relevantní údaje, i když jejich důvěryhodnost je často sporná. Jakožto rozumný zdroj se zdál být [6], ze kterého budou některé informace čerpány.

Dostupné technologie

Množství a kvalita technologií, převážně frameworků zaměřených na tvorbu webových aplikací v daném programovacím jazyce, jsou pro nás taktéž důležitým kritériem, jelikož právě z těchto možností bude dále v průběhu práce vybírána použitá technologie. Opět se nejedná o statistiku, kterou je zcela jednoduché uchopit, ale z dosavadních znalostí získaných studiem na FIT a dostatečným procházením on-line zdrojů je možné získat alespoň hrubou představu o tomto kritériu pro jednotlivé jazyky, kterážto představa nám bude postačovat.

Snadnost užívání, rozšířenost a dokumentace

V rámci výběru je nutné zohlednit i obtížnost a předpokládanou dobu vývoje v daném jazyce, jelikož některé programovací jazyky a nástroje, zvláště ze skupiny jazyků vázaných na JVM, jsou určeny pro tvorbu velmi rozsáhlých a komplexních systémů, čemuž však odpovídá velká časová náročnost a složitost práce s těmito nástroji. Tuto vlastnost, jakkoli vhodnou pro jiné projekty, nelze vzhledem k omezenému časovému harmonogramu této práce považovat za výhodu.

Také možnost sehnat nové programátory (v tandemu s cenou těchto programátorů, zohledněnou v příslušném kritériu), možnost získávat on-line informace a rady o vývoji v tomto jazyce a zároveň množství vývojářů, kteří daný jazyk pro tvorbu komerčních webových aplikací používají, reflektuje jeho použitelnost a vhodnost pro projekt Mojeskolka.

Současně s množstvím uživatelů jde ruku v ruce dokumentace a možnost získat pomoc, která je často nejaktuálnější a nejlépe dostupná z komunitních stránek (například velmi známé a velmi užitečné stránky StackOverflow ([7])[4], kde jsou informace dostupné právě od ostatních vývojářů často téměř v reálném čase.

Může být sporné, zda se jedná o zcela „tvrdou“ míru, ale nesporně nese určitou informaci.

[4]Dokonce tak známé, že indexy hodnotící popularitu a rozšířenost různých jazyků, kterážto indexy jsou zmíněny níže v tomto odstavci, berou tuto stránku jako jeden z hlavních ukazatelů.

Je poměrně těžké sehnat podložená a relevantní data, jelikož všechny indexy popularity programovacích jazyků (TIOBE, PYPL, výše zmíněné statistiky na [6] a další) zahrnují obecný zájem o daný jazyk, nikoli data v kontextu tvorby webových aplikací, ale i přesto se opět dá nějakým způsobem dohledat hrubá data, která nám alespoň do jisté míry pomohou s výběrem vhodné volby.

1.6.2 Výčet možných jazyků

Níže následuje výčet a rozbor jednotlivých programovacích jazyků, ze kterých je vybíráno. Jedná se o typické jazyky pro tvorbu aplikační logiky webových aplikací, se kterými je autor této práce alespoň letmo seznámený.

Java a skupina jazyků spouštěných na JVM

Skupina jazyků běžících na JVM (Java Virtual Machine) obsahuje hned několik kandidátů, kteří poskytují možnosti vývoje webových aplikací – za zmínku stojí minimálně Java EE, Groovy a Scalla.

Z těchto jazyků má nepochybně nejstabilnější pozici na trhu samotná Java, která je prověřená mnoha lety užívání a pevně zakořeněná ve světě vývoje webových aplikací, což je pro projekt Mojeskolka, i s ohledem na jeho plánovanou budoucnost, z této skupiny nejvhodnější volba. I ostatní jazyky ze zmíněné množiny nepochybně mají své výhody, například frameworky Play, nebo Grails jsou hodnoceny velmi dobře (kupříkladu v [1]), ale jedná se o nové technologie, se kterými má autor práce pouze omezené zkušenosti (kurz programování ve Scalle), proto budeme dále uvažovat pouze samotnou Javu.

Co se Javy týče, obecně se jedná o velmi rozšířené řešení, které je považováno za rychlé, spolehlivé a bezpečné, ale tvorba webových aplikací v Javě EE je mnohdy označována za poněkud „krkolomnou“ a technologicky náročnou. Ačkoli se jedná o velmi často užívaný a populární jazyk, tvorba webových aplikací v Javě není zcela triviální a rychlý vývoj vyžaduje mnoho zkušeností, které autor této práce v dostatečném rozsahu nemá. A dohledání podpory, užitečné dokumentace či nápovědy pro tyto projekty je značně náročné, kvůli menší a méně sdílné komunitě.[5]

Na druhou stranu, co se frameworků a podpůrných technologií týče, výběr je široký a poměrně kvalitní. Samotný jazyk je možné užívat bez jakýchkoli poplatků, stejně tak je možné užívat neplacené vývojové prostředí na vysoké úrovni, avšak Java programátoři webových aplikací patří k nákladnějším.

Velkou výhodou Javy jsou poměrně rozsáhlé znalosti autora i jeho praktické zkušenosti s vývojem větších aplikací v tomto jazyce, ale tyto zkušenosti a znalosti jsou bohužel orientovány spíše na standardní edici Javy, nikoli na webové aplikace a frameworky pro ně užívané.

[5] Autor vychází i z vlastních zkušenosti s vývojem ve frameworku Spring, či portálovém řešení Liferay.

C# a skupina jazyků platformy .Net

Platforma .Net poskytuje velmi silný a rozsáhlý základ pro tvorbu softwarových projektů, jazyk C# je velmi rozšířený a komunitou kladně přijímaný, nejspíše i proto, že do jeho vývoje jsou investovány nemalé prostředky a jazyk je cíleně navrhován a vyvíjen pod záštitou významné společnosti. Měl by existovat i dostatek nástrojů a dokumentace pro vývoj webových aplikací pro platformu .Net.

Velmi zásadním problémem je však velmi malá zkušenost autora s touto platformou – nejmenší, že všech zmíněných alternativ a důležitým nedostatkem je také nutnost platit za používání platformy, i za vývojové prostředí. To může být výhodou z hlediska jejich kvalit, ale nikoli z hlediska počátků tohoto projektu. Stejně tak vyšší plat programátorů v C# vrhá stín na výběr tohoto jazyka.

PHP

Na programovací jazyk PHP bývá někdy shlíženo svrchu, pro některé jeho nekonvenční charakteristiky a praktickou absenci cíleného návrhu při jeho vzniku (to je možno slyšet od samotného autora jazyka v rozhovoru [8]). Ale i když osobní názor autora a mnoha jiných na úroveň PHP jako programovacího jazyka nemusí být úplně pozitivní, nedá se upřít, že PHP je široce používaný a ustálený nástroj, který svůj účel plní. A tímto účelem od samotného vzniku tohoto jazyka bylo vytvářet snadno a rychle webové aplikace.

Obecně panuje názor, že kvalitu a úroveň těchto aplikací musí zajistit samotný programátor a nejedná se vždy o jednoduchý úkol, ale i přesto je PHP se svou orientací pouze na webové aplikace pro jejich tvorbu vhodným nástrojem. Samotný programovací jazyk je možné svobodně užívat i pro komerční účely, stejně tak je možné pro vývoj v PHP použít nekomerční vývojová prostředí na dostatečně vysoké úrovni.

Skutečnost, že jazyk je velmi široce používán, je reflektována rozsáhlou komunitou, díky které je možné dohledat mnoho návodů a užitečné pomoci s naprostou většinou běžných i méně obvyklých problémů, které při vývoji aplikací mohou nastat. Nevýhodou je méně kvalitní dokumentace k PHP[6] Jak bylo zmíněno výše, s užíváním PHP má autor této práce poměrně bohaté zkušenosti, ale ne tak hluboké teoretické znalosti a problémem je i neveliká znalost frameworků pro vývoj v tomto jazyce.

Ruby

Vývoj webových aplikací v Ruby je úzce spjat s webovým frameworkem Ruby on Rails (RoR), což poněkud omezuje výběr možných technologií. Jedná se

[6] za příklad poslouží oficiální dokumentace k funkci count(), kterou však rozsáhlá a sdílná komunita zvládá vyvažovat.

o velmi omezující podmínku, i když je Ruby on Rails považováno za kvalitní framework, který je užíván mnoha vývojáři. V důsledků tohoto omezení na jediný framework budeme dále v této kapitole mít v souvislosti s Ruby na mysli převážně Ruby on Rails.

Specifickou vlastností RoR, kterou je vhodné zmínit, je primární zaměření na rychlé protypování aplikací, které framework se svým přístupem „convention over configuration“ zvládá skvěle. Je možné instalovat RoR, okamžitě spustit vývoj produktu a při dodržování správné struktury projektu (která navíc vede k čistému a přehlednému návrhu) je možné velmi rychle vytvořit kvalitní prvotní verzi aplikace. Ruby on Rails však skutečně od počátku svého vývoje bylo zaměřeno na prototypování aplikaci, což z principu implikuje nutnost výsledný produkt od základu přepisovat v relativně blízké budoucnosti, což není úplně žádoucí. Tento problém podtrhuje fakt, že jazyk Ruby je obecně spojován s horší škálovatelností. Stojí za zmínku, že například PHP taktéž není spojováno s bleskovou rychlostí, ale je o něco méně náročné na systémové zdroje (srovnání možné nalézt například v [9]).

Na druhou stranu existuje pro Ruby on Rails velmi nadšená komunita, velké množství tutoriálů a kvalitní dokumentace, i když je celkový objem dostupných materiálů menší, než v případě PHP (kvůli celkově menší komunitě, jakkoli nadšené). Zároveň je Ruby on Rails volně použitelné i pro komerční účely, ale drobnou nevýhodou RoR jsou vyšší platy Ruby programátorů a skutečnost, že kvalitní IDE, se kterým má autor práce zkušenosti, je placené.

Co se zkušeností autora této práce týče, jedná se spíše o teoretické znalosti, získané v rámci kurzu programování v Ruby, kteréžto znalosti nejsou podepřené praktickou zkušeností s tvorbou aplikací v RoR, jakkoli bylo setkání s tímto jazykem a frameworkem příjemné.

1.6.3 Vyhodnocení

Tabulka 1.2: Shrnutí srovnání programovacích jazyků

	Znalosti	Cena	Dostupné technologie	Snadnost užívání, rozšířenost a dokumentace
Java	+	+	++	−
Platforma .Net	−−	−−	++	+
PHP	+	++	++	+
Ruby	−	+	+	+

Součet hodnocení

- PHP: 7 bodů
- Java: 4 body

- Ruby: 1 bod
- Platforma .Net: −3 body

Ze srovnání vychází jednoznačně nejlépe programovací jazyk PHP. Je to především kvůli zkušenostem autora, širokému výběru frameworků a nízkým nákladům pro spuštění projektu tvořeného v jazyce PHP. Ostatní jazyky zároveň poněkud zastiňuje obava z úskalí vývoje v nich, zatímco PHP paradoxně pomáhá fakt, že s jeho problematickými stránkami má autor praktické zkušenosti a má představu, jak je řešit.

Na druhém místě se umisťuje Java, o které má sice autor nejhlubší znalosti, avšak primárně v oblasti vývoje desktopových aplikací. Zároveň je velkou slabinou Javy v tomto srovnání relativně vysoká technická náročnost tvorby webových aplikací v tomto jazyce, stejně jako obtížnější dohledávání pomoci s problémy, které při vývoji mohou nastat, což je prakticky pouze další parafráze na obavy z úskalí vývoje zmíněné výše.

V případě, kdy by autor měl možnost na této práci spolupracoval s pokročilejším programátorem majícím dostatečně rozsáhlé praktické zkušenosti v oblasti vývoje webových aplikací v jazyce Java, tyto nedostatky by měly mnohem menší váhu a pravděpodobně by jako finální volba byl zvolen programovací jazyk Java. Za současných podmínek však vysoká technická náročnost a obavy z opakovaného uvíznutí v mrtvých bodech během vývoje řadí tento jazyk až na druhé místo.

1.6.4 Závěr

Tato kapitola rozebírá velmi ošemetné a náročné téma. V rámci tvorby této práce bylo nemálo hodin stráveno dohledáváním informací o jednotlivých jazycích, jejich frameworcích (jejichž výběr na výběr jazyka navazuje) a jejich vhodnosti pro podobný projekt, stejně tak i diskusemi na toto téma s odborníky z akademické sféry i ze sféry komerční.

Jedná se i tak o poměrně těžké rozhodnutí a existuje mnoho pro a proti hovořících o vhodností každé možnosti. Většina dostupných zdrojů se však shoduje v tom, že neexistuje universální nejlepší volba, pouze volba vhodnější a méně vhodná. Jenže ještě lépe řečeno je to volba „vhodná, či nevhodná pro konkrétní projekt“, což nás vrací zpět na začátek s otázkou „Jaký programovací jazyk je pro projekt Mojeskolka nejvhodnější“?.

Pro zodpovězení této otázky proto vznikla celá předchozí sekce, která nám poskytuje tu nejpřesnější odpověď, jakou jsme momentálně schopni dostat – na základě uchopitelných kritérií jsme zhodnotili možnosti a pokusili jsme se o výběr té nejvhodnější. Za zmínku stojí, že mezi uchopitelná kritéria nespadá například „kvalita návrhu jazyka“ – která je u PHP často kritizována – a nespadá do nich právě proto, že ji nejsme schopni snadno vyjádřit a na jejím základě srovnat jednotlivé možnosti. To by vyžadovalo roky praxe a hluboké

znalosti práce s daným jazykem v kontextu webových aplikací, kteréžto znalosti objektivně vzato autor této práce nemůže míti.

Rozhodujeme se tedy na základě informací, které jsou nám momentálně dostupné, a na základě kritérií, které v současnosti dokážeme posoudit.

Kapitola 2

Návrh uživatelského rozhraní

Kvalitní návrh uživatelského rozhraní je klíčový pro úspěch projektu Mojeskolka a je klíčový i pro tuto práci – stačí uvést, že přinejmenším pět bodů ze sekce 1.2.3 má přímou vazbu na uživatelské rozhraní. Je nesporné, že úspěšný softwarový produkt musí být stabilní, bezpečný, mít výborně zpracovanou architekturu umožňující jeho údržbu a další rozvoj, ale všechny tyto kvality by přišly vniveč, pokud výsledný produkt nebudou uživatelé aktivně využívat a nebudou k němu mít pozitivní vztah, což vyžaduje, aby se jim s ním dobře pracovalo díky kvalitnímu návrhu UI a aby je oslovil grafický design (rozdílu a dopadu těchto dvou charakteristik se věnuje příslušná sekce uvedená níže).

Návaznost na předchozí práci

Jak bylo zmíněno výše v kapitole úvod, návrh uživatelského rozhraní nebude v rámci této práce vytvářen od píky, ale bude vycházet z předchozí práce vytvořené v předmětu NUR.

Na úvod práce na této kapitole byly zpracovány poznámky z dosavadního vývoje, celkem se jedná o více, jak 80 stran dokumentů, přičemž tato práce vychází z jejich interní, needitované podoby obohacené o poznámky autorů. Z těchto dokumentů byly vytaženy nutné, nebo důležité požadavky na změny v návrhu UI založené primárně na výsledcích testování, jmenovitě uživatelských testů a heuristické analýzy.

Zároveň budou využity i zhotovené prototypy jednotlivých stránek v systému, na jejich základě bude totiž zpracována nová podoba těchto stránek.

Postup

Výsledkem této kapitoly by měla být další, v pořadí druhá iterace návrhu uživatelského rozhraní, která rozšiřuje a zdokonaluje návrh vytvořený v rámci předmětu NUR.

Pro tvorbu této další iterace nejprve shromáždíme potřebné podklady – zpracujeme rešerše, definujeme obecné priority a pravidla, kterými se budeme

řídit při návrhu UI – a vybereme vhodné technologie a nástroje, které nám později pomohou uživatelské rozhraní implementovat. A na základě všech těchto podkladů vytvoříme novou podobu jednotlivých stránek.

2.1 Odlišení návrhu UI od designu grafiky a jejich význam

V této úvodní sekci si dovolíme čtenáři přiblížit zásadní rozdíl mezi dvěma koncepty, které se často zaměňují, ačkoli se ve skutečnosti jedná o oddělené složky procesu tvorby aplikace a měli by je zpracovávat odlišní lidé.

Jedná se o návrh uživatelského rozhraní a grafický design.[7] Oba tyto pojmy se úzce týkají prezentační vrstvy aplikace – tedy toho, „co uživatel vidí“ – ale každý z jiného úhlu pohledu. Zatímco grafický design řeší barvy, ikony, tvar a další prvky ovlivňující estetické působení aplikace, návrh uživatelského rozhraní se zabývá rozložením ovládacích prvků a prezentovaných informací na obrazovce tak, aby se v nich uživatel co nejsnáze orientoval a užívání aplikace pro něj bylo co možná nejjednodušší.

Pro lepší představu je vhodné uvést, že produktem návrhu uživatelského rozhraní je kostra rozložení ovládacích prvků, kterou velmi dobře zachycují takzvané mockupy (které je možné vidět níže v sekci návrhu jednotlivých stránek v aplikaci) – tedy nákresy zachycující uspořádání jednotlivých ovládacích prvků (tlačítek, odkazů, ...) a bloků (menu, panelů, ...) na obrazovce prezentované uživateli. Zatímco cílem grafického designu je vhodně tyto prvky obarvit, vyplnit obrázky a vhodnými ikonami, aby bylo dosaženo příjemného estetického efektu.

Dopad na uživatele

Grafický design a návrh UI oba nepochybně ovlivňují celkový dojem uživatele aplikace, takzvané UX (user experience), avšak poměr významu těchto dvou složek je předmětem debat.

Ze závěrů populární studie What is beautiful is usable, provedené Noamem Tractinskym a kolektivem v roce 2000 (vycházíme z jejího shrnutí v [10]) vyplývá, že dojem, který v uživateli zanechá uživatelské rozhraní bankomatu, je velmi zásadně závislý na grafickém designu („pohlednosti“) a v mnohem menší míře na kvalitě návrhu uživatelského rozhraní. To by poukazovalo na menší význam návrhu, kterým se má zabývat tato kapitola.

Závěry této studie však byly opakovaně zpochybňovány, například právě v práci [10], ve které se sice mimo jiné uvádí, že na základě všech dostupných studií se zdá, že existuje korelace mezi tím, jak uživatelé hodnotí grafický

[7] Za zmínku stojí, že český jazyk nám odlišení těchto dvou konceptů usnadňuje, jelikož máme výrazy „návrh“ a „design“, které však jsou v angličtině reprezentovány stejným slovem.

design (estetiku) stránek a jak hodnotí návrh uživatelského rozhraní, avšak různé studie už se rozchází v tom, jaký je poměr jejich vlivu a která stránka má větší vliv na celkový dojem uživatele ze systému. Závěrem zmiňované práce je naopak tvrzení, že vztah mezi „krásou“ a „použitelností“ uživatelského rozhraní je obrácený, než jak se domnívala studie What is beautiful is usable – tedy že hodnocení pohlednosti uživatelského rozhraní (jeho grafického designu) je ovlivněno použitelností tohoto rozhraní (uživatel frustrovaný nelogickým, nepřehledným, nebo z jiného důvodu špatně použitelným rozhraním jej označí za „ošklivé“, ačkoli je po stránce grafického designu stejné, jako rozhraní, které jiní uživatelé hodnotí jako „pohledné“). A naopak – „ošklivý“ design neměl příliš negativní vliv na hodnocení použitelnosti produktu.

Na základě těchto zdrojů je tedy možné usuzovat, že na dojem uživatele z produktu a jeho hodnocení použitelnosti (usability) produktu má vliv i grafický design, i návrh UI, ale že význam kvality návrhu uživatelského rozhraní je výrazně vyšší, než říkala studie Noama Tractinského a kolektivu, a pro výsledný dojem z produktu je snadná použitelnost a srozumitelnost produktu velmi zásadní. To potvrzuje naše tvrzení o významu této kapitoly a zdůvodňuje, proč je tato kapitola zpracována značně detailně.

Závěr

V rámci projektu Mojeskolka bude návrh uživatelského rozhraní úkolem autora této práce, zatímco návrh grafického designu bude po dokončení této práce svěřen do rukou profesionálního grafika a následně zakomponován do další verze aplikace. Z toho vyplývá, že prototypy vytvořené v této práci nebudou mít dokonalý grafický design a důraz bude kladen na kvalitu návrhu uživatelského rozhraní.

2.2 Rešerše

Dle metodik návrhu GUI, vyučovaných v rámci předmětu NUR, je vhodné analyzovat podobné, v optimálním případě přímo konkurenční aplikace, a začlenit získané poznatky do návrhu GUI vlastní aplikace. Můžeme tak zahrnout jejich kladné stránky a inovativní nápady, zatímco uvědomění si jejich chyb nám může pomoci vyvarovat se podobných přehmatů v našem vlastním produktu. To poukazuje na skutečnost, která bude v této práci zmíněna i v dalším kontextu – není vždy vhodné snažit se samostatně vyřešit problémy, které někdo řešil a dost možná vyřešil před námi.

V rámci tvorby GUI je nepochybně vhodné snažit se o jistou míru originality, už jen proto, abychom se nesnažili náš produkt „napasovat“ na cizí rozhraní, které se nám zalíbí, zatímco jeho vhodnost pro realizaci našeho projektu je pochybná. Ale naše vlastní kreativita měla možnost se projevit již v prvotním návrhu GUI v rámci NUR, kterýžto návrh byl vytvářen dříve, než byly prováděny jakékoli rešerše cizích stránek. O zbytek originality by se měl

postarat převážně grafický design, ale z hlediska návrhu uživatelského rozhraní je důležitější použitelnost produktu, než snaha o originalitu. Navíc sami uživatelé často uvítají spíše známé prvky uživatelského rozhraní (například bubliny označující počet příchozích zpráv, ale i základní principy jako červená varování a zelená označení úspěchu), než snahu o originalitu. Na základě poznatků z předmětu NUR, ale i průběhu předchozích testování aplikace, můžeme dokonce tvrdit, že kvalitnější řešení UI, na které však uživatel není zvyklý, může paradoxně vyvolat negativní reakci v porovnání s variantou, která je nedokonalá, ale uživatel je na ni zvyklý.

Proto je vhodné pochopit užívané vzory a způsoby, jakými již existující aplikace řeší jednotlivé problémy v návrhu uživatelského rozhraní a vzít si z nich pozitivní, i negativní poučení.

Vzhledem k tomu, že jsme jako podobu aplikace zvolili webového klienta s důrazem kladeným na mobilní zařízení, je vhodné analyzovat jednak kvalitní aplikace optimalizované pro přenosná zařízení, kvůli inspiraci po stránce responzivního UI, a jednak konkurenční aplikace, které poskytnou jak přehled z hlediska návrhu a designu UI užívaného v dané oblasti, tak inspiraci po stránce funkcionality. Navíc získáme schopnost srovnat vlastní produkt s konkurenčními produkty, což může být například při nabízení systému zákazníkovi velmi vítaným bonusem.

2.2.1 Výtah z předchozí práce

V rámci zpracování rešerší opět využijeme již hotové práce z předmětu NUR. Vyjdeme z druhé iterace semestrální práce, v rámci které byly vypracovány rešerše na několik různých webových stránek a aplikací. Primární zaměření bylo na úspěšné nebo kvalitně zpracované webové aplikace, povětšinou s Responzivním designem (zmíněným níže), se kterými byla současně zkoumána i příslušná mobilní aplikace, pokud takovou aplikaci daný produkt nabízel. Zároveň byly při výběru aplikací pro rešerše preferovány takové, které cílí na podobné zákazníky, jako projekt Mojeskolka – mladé lidi a rodiče dětí v školním, ideálně předškolním věku. Cílem bylo získat co nejvíce relevantní informace a postřehy.

Níže je uveden výčet vybraných pozitivních a negativních prvků zaznamenaných napříč zkoumanými stránkami a aplikacemi. Nejedná se o kompletní seznam, mnoho dalších poznatků bylo zakomponováno již do tvorby prototypu v rámci NUR, ale spíše o příklad zajímavých a relevantních informací získaných z předchozích rešerší.

Pozitivní inspirace

V této sekci si přiblížíme pozitivní prvky, na které jsme narazili v dřívějších rešerších, tedy prvky, které může být vhodné zakomponovat do našeho návrhu.

- Obvyklé je užívání čistých a jasně oddělených grafických bloků v kombinaci s barvami, které upřesňují zdroj či význam obsahu příslušných bloků. A tento způsob prezentace je zamýšlen například pro hlavní stranu aplikace Mojeskolka a obecně při prezentaci notifikací. To jednak implikuje správnost naší volby a jednak to přináší výhodu z hlediska pocitu uživatele, kterému se systém zdá být bližší a srozumitelnější již při prvním setkání.

- Velmi hojně jsou také užívány vhodné grafické ikony, které usnadňují orientaci uživatele v aplikaci a zrychlují jeho volbu na základě jednoznačných grafických identifikátorů. Příčinu můžeme spatřit jednak v tom, že je rychlejší shlédnout ikonu, než číst text na tlačítku, a jednak v tom, že často užívané ikony jsou pro uživatele snadno identifikovatelné a opět přispívají k rychlému seznámení se s aplikací a k pocitu uživatele, že prostředí aplikace je mu blízké.

- Na stránkách určených pro rodiče se často vyskytují dětské motivy, fotky dětí, výrazné barvy a další prvky související s dětmi. To nejspíše reflektuje snahu stránek vcítit se do nálady uživatele a stejně tak i vazbu mezi obsahem stránek a grafickým zpracováním. Ačkoli se jedná spíše o grafický prvek (návrh UI ovlivňuje pouze možnost užívání barev pro doplnění obsahu, zmíněná výše), pak je vhodné mít na mysli, že dojem uživatele ze vzhledu aplikace ovlivňuje jeho pocit spokojenosti při práci s ní, což je pro úspěch projektu důležitý parametr.

- Některé stránky umožňující práci s podobnými prvky, jaké v naší aplikaci představují notifikace, umožňují práci s těmito prvky již v jejich obecném přehledu, v našem případě již na hlavní straně, či na nástěnce dítěte. Funkce pro práci s notifikacemi bez nutnosti otevírat jejich detail mohou pokročilejšímu uživateli poskytovat příjemnější možnosti, jak aplikaci užívat. Je však třeba myslet na ovládání na mobilních zařízeních, tedy nutnost omezit možnost nechtěného stisku chybné volby a limitovat množství zobrazovaných prvků v prezentovaných uživateli, proto se tento bod vztahuje spíše na stolní počítače.

- Dle vzhledu mnoha stránek v režimu pro mobilní zařízení, nebo jejich příslušných mobilních aplikací (například ČEZ, Facebook a dalších) se zdá, že dobře udělaný a kvalitní responsive design webových stránek je dobrým výchozím bodem pro tvorbu GUI jejich mobilních verzí. Je dobré zdůraznit, že takto významné firmy považují svůj základní webový design za dostatečně dobrý i pro mobilní aplikace určené na rozdílné mobilní OS (Android, iOS, ...), kteréžto operační systémy se vyznačují výrazně odlišnými uživatelským prostředím. V důsledku byly jejich mobilní aplikace vytvořeny bez nutnosti velkých rozdílů mezi produkty pro

jednotlivé platformy. To je pro nás dobrým znakem, že vycházet od kvalitního Responzivního webu je dobrá volba, která nám umožní snadnější přechod na mobilní zařízení (odkazujeme tím na výběr Responzivního webu jako výchozí platformy, který byl učiněn v úvodu v sekci 1.5).

Negativní inspirace

Tato sekce naopak obsahuje prvky návrhu UI, které na zkoumaných aplikacích působily problémy a kterých bychom se tedy raději měli vyvarovat.

- Příliš dlouhé a špatně tříděné seznamy položek, které byly kvůli nepřehlednosti prakticky nepoužitelné. Například v souvislosti s výpisem dětí v učitelské aplikaci, nebo notifikací v aplikaci rodičovské je dobré míti tuto chybu na paměti a vyvarovat se jí správným tříděním dlouhých seznamů, ideálně do stromové struktury, nebo alespoň umožnit vyhledávání v příslušném seznamu. To, že v testovací verzi aplikace se zdá být málo položek neříká nic o tom, že jich při reálném nasazení nebudou stovky. A rozdíl mezi lineární a logaritmickou asymptotickou složitostí vyhledávání není důležitý jen pro počítače, ale i pro uživatele.

- Prezentace informací podle jejich interní reprezentace a třídění v systému, nikoli podle kritérií blízkých uživateli. Tento bod souvisí s předchozím bodem a s obecnou snahou navrhovat systém orientovaný na uživatele, nikoli sám na sebe. Pokud budou pro uživatele kategorie ve stromové struktuře (ať už se jedná o menu, nebo výpis položek) naprosto nic neříkající, nebo budou prezentovány v nevhodné podobě, prohledávání uživateli stále zabere asymptoticky lineární čas a případ je degradován na předchozí bod. Poučením tedy je netřídit podle interních kategorií, blízkých programátorovi systému, ale podle toho, co uživatel hledá.

- Nápověda vytvořená pouze za účelem splnění zadání zakázky není příliš platná. Nápověda by měla být, stejně tak jako celý systém, skutečně určena pro uživatele. Nic neříkající opis toho, co se na stránce nachází, sice splní požadavky v zadání projektu, ale pro výsledný produkt je to jen přítěží.[8]

- Úvodní stránka bývá často nepřehledná a určená pro stálého uživatele, nikoliv pro uživatele nového, který právě systém otevřel poprvé v životě. Proto je v plánu v rámci aplikace Mojeskolka novému uživateli zobrazit úvodní stránku s rychlým vysvětlením a stručnou průpravou určenou pro takovéhoto uživatele (jak bylo zmíněno výše v 1.4). Zároveň je třeba

[8] V rámci této práce může být tento bod poněkud sporný, protože jak bylo řečeno v úvodu, nápověda bude realizována pouze v omezené míře. Ale význam této informace je především do budoucna a už při vývoji systému je dobré myslet na to, k čemu má nápověda sloužit, i když její obsah zatím nebude kompletní.

myslet na skutečnost, že uživatel zvyklý užívat počítačové systémy by neměl potřebovat nápovědu pokud možnou žádnou, tato úvodní obrazovka by tedy měla sloužit uživatelům obecně nezkušeným s užíváním podobných aplikací a zkušený uživatel musí mít možnost tuto průpravu snadno přeskočit. [9]

Dodatečné postřehy

V této sekci jsou zmíněny z rešerší získané neutrální poznatky, nespadající do předchozích kategorií, které však mohou projektu přinést užitek.

- Stránky školek určené pro veřejnost: školky pro své stránky typicky používají systém umožňující snadnou tvorbu stránek pomocí grafického nástroje (kupříkladu systém Webnode), nebo mají stránky podprůměrné až nízké kvality s velmi zastaralým designem a malou hodnotou pro školky i pro rodiče, které rozhodně nevyužívají plného potenciálu komunikace školky s veřejností. Tento poznatek poukazuje na další možnost rozvoje služeb nabízených v rámci projektu Mojeskolka, jelikož by systém v budoucnu mohl sloužit nejen pro interní komunikaci s rodiči a učiteli, ale mohl by i poskytovat možnosti, jak veřejně prezentovat danou školku (důležité informace o školce, pořádaných akcích i dalších aktivitách školky budou v systému uloženy na každý pád, jejich další využití ve prospěch školky je tak výrazně jednodušší).

- Platforma zkoumaných produktů: je vidět, že zkoumané aplikace potvrzují trend v současném vývoji informačních systému – většina aplikací je nyní webových, případně doplněných o mobilní klienty. To poukazuje na správnost naší volby z hlediska platformy, pro kterou budou aplikace z projektu Mojeskolka vyvíjeny.

Závěr

Předchozí informace nám poskytují spíše obecná doporučení a v mnoha směrech se překrývají s již dříve získanými znalostmi autora, ale i po této stránce se přinejmenším jedná o příjemné utvrzení, že naše představa o výsledném produktu není zcestná. Zároveň je možné nechat se v mnoha ohledech inspirovat a tyto postřehy se budeme dále v práci snažit zakomponovat do našeho návrhu.

[9] V prototypu aplikace bude opět pouze omezený prostor pro realizaci detailní úvodní prezentace systému novému uživateli, ale jedná se o velmi užitečnou informaci pro budoucí rozvoj aplikace.

2.2.2 Provedené rešerše

V rámci této práce byly následně vypracovány rešerše vybraných konkurenčních systému. Jejich detailní rozbor je uveden v příloze Výsledky rešerší konkurenčních systémů.

Stručnější výtah z těchto rešerší poskytuje následující subsekce.

2.2.3 Závěr provedených rešerší

V této sekci se pokusíme učinit závěr charakteristikách zkoumaných systémů, které považujeme z hlediska projektu Mojeskolka za zajímavé nebo vhodné pro potenciální inspiraci. Vzhledem k rozsahu jednotlivých zkoumaných oblastí bude podle nich tato sekce rozdělena do několika částí.

Zhodnocení konkurenceschopnosti

Rešerše poskytly velice užitečný náhled do světa konkurenčního softwaru, jeho možností a kvalit. Ukázalo se, že jediným v současnosti užívaným produktem, který je alespoň částečně blízký plánovanému zaměření Mojeskolka, je Edookit. Zároveň se jedná o jediný produkt, v rámci kterého nebyly nalezeny zásadní nedostatky, má moderní uživatelské rozhraní a dosahuje po stránce uživatelského rozhraní, zpracování a použitelnosti[10] kvalit, které považujeme za skutečně úctyhodné. Může se jednat o subjektivně zabarvený názor, ale ostatní systémy objektivně mají značné nedostatky a ačkoli nepochybně slouží svému účelu (jak může potvrdit jejich rozšířenost), systém Edookit dosahuje podstatně vyšších kvalit. Navíc alespoň částečně cílí i na mateřské školky, proto ho můžeme považovat za nejvýznamnější konkurenci projektu Mojeskolka.

K této rozvaze byla ještě sestavena následující tabulka, která obsahuje charakteristiku, vystihující, jak moc daný systém cílí na mateřské školky. Tato charakteristika byla vyhodnocena na základě komplexního obrázku o zkoumaném systému.Tento obrázek si nyní, po detailní analýze všech systémů, můžeme utvořit.

Tabulka 2.1: Srovnání zkoumaných systémů dle jejich překryvu se zaměřením systému mojeskolka.

Název systému	Zaměření na školky
Edookit	Částečné
Škola OnLine	Zcela minimální
OpenSIS	Žádné

[10]Použitelností máme na mysli dostupnost z mobilních telefonů, užitečnost nabízených funkcí a snadnost jejich používání, které mají velký vliv na možnost aplikaci plně využívat a u systému Edookit působily velmi pozitivně.

Z námi vyvozených závěrů je možné vidět, že projekt Mojeskolka cílí na část trhu, která není ještě zcela zaplněna a zároveň ji vyplňují převážně systémy, kterým by Mojeskolka měla být schopná konkurovat.

Funkcionalita

Zkoumáním konkurenčních systémů byly získány užitečné informace i o jejich funkcionalitě.

Základní funkce definované výše v rámci sekce 1.2.1 byly v jisté formě nalezeny prakticky ve všech systémech a stručně je shrnuje následující blok (nejedná se o nikterak překvapivé zjištění, jelikož je řeč o velmi základních funkcích, ale zároveň se zdá, že nic důležitého neuniklo naší pozornosti).

Potvrzená funkcionalita:

- Evidence žáků
- Správa uživatelů
- Funkce pro upozornění rodičů na nové události (notifikace)
- Kontakty
- Zprávy v systému nahrazující elektronickou poštu

Zároveň rešerše potvrdily, že mnoho funkcí plánovaných do budoucna pro systém Mojeskolka se objevuje ve více příbuzných systémech a tudíž se pravděpodobně jedná o funkcionalitu, kterou skutečně bude vhodné implementovat. Ačkoli zmíněné plány nejsou v této práci v plném rozsahu upřesněny, v rámci této sekce považuje autor za zajímavé uvést i shodu mezi konkurenčními produkty a plány do budoucna, které vznikly nezávisle na nich.

Do budoucna plánovaná funkcionalita nalezená ve zkoumaných systémech:

- Sdílení souborů s rodiči
- Kalendář událostí a úkolů
- Správa financí a plateb od rodičů
- Komentáře, respektive obecně diskuse nad informacemi v systému
- Evidence suplování

Stejně tak je záhodno zmínit i zcela nové podněty, které nám rešerše poskytly. Ne všechny tyto nové poznatky budou využity přímo v této práci, ale z hlediska budoucnosti projektu jsou podstatné a podtrhují význam rešerší.

Nové:

- Omluvenky, především včasné omlouvání dětí z výuky
- Rozvrh, alespoň hrubý, který by umožnil rodičům zjistit přinejmenším hrubou momentální polohu dítěte, kupříkladu pokud jej chce vyzvednout předčasně.
- Provázání e-mailu a zpráv
- Historie činnosti uživatele a posledních navštívených stránek systému
- Záznamy o přihlášení uživatele do systému
- Export dat do dalších aplikací, které školky využívají
- Vazba na sociální sítě
- Štítky, neboli „tagy“ u notifikací a zpráv

Na závěr je vhodné zmínit, že je třeba míti se na pozoru před přílišným důrazem na málo využívané funkce systému a naopak je nutné cílit na takzvanou „core“ funkcionalitu a tu zpracovat na excelentní úrovni. Důslednou analýzou a filtrací nalezených inspirativních funkcí, které jsou implementovány ve zkoumaných systémech, však mohou být objeveny velmi důležité a užitečné funkce, které by systém Mojeskolka mohl do budoucna poskytovat.

Prvky UI

Z velmi rozsáhlého seznamu objevených pozitiv a negativ u jednotlivých stránek v této sekci vybereme ty, které jsou pro nás zásadní a budou mít dopad na tvorbu UI aplikací ze systému Mojeskolka. Jedná se povětšinou o pozitivní inspiraci, jelikož většina záporných poučení pro nás není tolik relevantní a spíše poslouží při zpětném rozboru již hotového návrhu našeho uživatelského rozhraní a kontrole, že neopakujeme v rešerších nalezené chyby.

Barvy:
Jedná se o tolik výraznou složku UI, že jí věnujeme samostatný blok.

- Užití barev pro odlišení včerejšího, dnešního a zítřejšího dne.
- Modré odkazy pro přechod na jinou stránku.
- Zvýraznění přechodů na jinou stránku a shrnování.
- Barevné odlišení výsledku operace – zelený úspěch a červený neúspěch.

Barvy jsou velice užitečné pro doplnění prezentované informace, ale je třeba dát pozor na příliš velké množství barev užitých současně. Vyvstává tedy otázka „Co je příliš?“. Na ni bude třeba hledat odpověď testováním uživatelského rozhraní a opakovaným hledáním vhodné míry.

Další prvky:

- Našeptávání, užitečné obzvláště pro mobilní zařízení.
- Podoba zpráv, velice přehledně řešená v systému Edookit.

Srovnání systémů dle UI

V této sekci se pokusíme porovnat jednotlivé systémy na základě číselného ohodnocení kvality jejich UI. Budeme se snažit ohodnotit výsledky heuristické analýzy, doplněné o klady a zápory. Jedná se o značně hrubou míru, která je však vyvážena jemností detailů samotných rešerší.

Tabulka 2.2: Srovnání zkoumaných systémů dle výsledků heuristické analýzy, pozitiv a negativ UI.

Název	*	1	2	3	4	5	6	7	8	9	10	Další	Celkem
Edookit	N		2(0)		1		1	1(0)	2		1	3(1)	−7
	P		++	++	++	+	++		++			10	+21
Škola OnLine	N		3		3	1	2	1	1			4	−15
	P										+	1	+2
OpenSIS	N		2		3		1	1	4	2	3	3(2)	−17
	P	++		+		+						3(1)	+6

* N značí negativa a problémy nalezené heuristickou analýzou, zatímco P značí pozitiva.
** Čísla v závorkách jsou užita, pokud ne všechny položky jsou dostatečně závažné, a reflektují skutečnou váhu těchto položek dohromady.

Je možné povšimnout si, že ačkoli jednotlivé body nemají stejnou váhu a je dobré zdůraznit, že některé chyby nalezené v systémech OpenSIS a Škola OnLine se vyskytovaly tak často, nebo měly tak zásadní význam, že by zasloužily naopak zvýšenou váhu, pak i takto stručná tabulka poměrně přesně reflektuje výsledky rešerší. Jednak z hlediska relativní kvality jednotlivých systémů, ale například i rozporuplnosti systému OpenSIS, který se na jednu stranu snaží o moderní přístup k řešení problémů, což se promítlo v zisku pozitivních bodů, ale na druhou stranu selhává v dodržení základních pravidel pro návrh UI.

Technologie

U všech detailněji zkoumaných systémů bylo možné přímo, či nepřímo získat informace o technologiích užitých na jejich realizaci a o způsobu poskytování produktu. Tyto údaje přehledně shrnuje následující tabulka.

Tabulka 2.3: Srovnání zkoumaných systémů dle technologií použitých pro jejich realizaci.

Název systému	Programovací jazyk	SaaS
Edookit	PHP	Ano
Škola OnLine	C#, platforma .NET	Ano
OpenSIS	PHP	Volitelně

Je vidět, že realizace projektu Mojeskolka v podobě webové aplikace poskytované jako softwarová služba implementovaná v jazyce PHP vůbec není tak výjimečná ani mezi největšími produkty na trhu, ba právě naopak[11]. To nás může utvrdit v přesvědčení, že zvolený způsob realizace by neměl být nikterak zcestný a zároveň získáváme lepší představu o charakteru konkurence.

Shrnutí

Vypracování rešerší nám dodalo mnoho velice užitečných informací. Jejich dopad na tuto práci značný – potvrdila se nám správnost mnohých učiněných rozhodnutí, i nám byly poskytnuty zajímavé podněty k návrhu UI, včetně obecného přehledu, jaké prvky a „kombinace“ v uživatelském rozhraní fungují dobře a které naopak nikoliv, což bude velmi užitečné pro zpracování uživatelského rozhraní v této práci. Ale mnohem větší význam mají rešerše pro samotný projekt Mojeskolka – rozsah získaných informací a možnost zdravé sebejistoty, kterou jsme nyní mohli nabýt v oblasti konkurenčního softwaru, jsou nesmírně cenné.

2.3 Principy návrhu UI

Cílem této sekce je definovat důležité principy, kterými se v rámci návrhu UI systému Mojeskolka budeme řídit. Nejprve v sekci o takzvaných „design guidelines“ určíme obecný přístup k návrhu uživatelského rozhraní, ze kterého budeme vycházet, díky čemuž získáme základní sadu vhodných doporučení a pravidel pro tvorbu UI. Na základě těchto doporučení, obecných principů dobrého návrhu uživatelského rozhraní (probíraných v předmětu NUR) a informací získaných rešeršemi jiných systémů se pokusíme sestavit sadu principů vhodných pro návrh a tvorbu uživatelského rozhraní.

Důležitým závěrem této sekce tedy budou obecná pravidla pro tvorbu UI, která shrnují náš přístup k jeho návrhu a realizaci v rámci systému Mojeskolka.

[11] Za zmínku stojí, že rešerše skutečně byly provedeny až po vyhodnocení vhodné podoby aplikace a jazyka pro implementaci. Jediným technologickým aspektem, o které jsme doposud nehovořili, je forma distribuce, které se budeme věnovat níže v sekci Forma poskytování aplikace, ale od začátku úvah o projektu se předpokládala realizace jako SaaS.

2.3.1 Design guidelines a responzivní design

Jak již bylo zmíněno v úvodu sekce týkající se rešerší, při návrhu UI (i při jiných úkonech) je dobré využít znalostí těch, kteří daným procesem prošli před námi a v našem případě je také vhodné dodržovat současné trendy v návrhu aplikací. Je tedy na místě vybrat vhodnou metodiku návrhu UI, které se budeme v rámci této práce držet. Vzhledem k tomu, že projekt cílí na webové prohlížeče a na mobilní zařízení, budeme vycházet ze sad doporučení, souhrnně označovaných jako design guidelines, určených právě pro webové a mobilních aplikace. Tyto dvě skupiny aplikací propojuje takzvaný responsive design webových stránek, ze kterého budeme primárně vycházet.

Tento responzivní návrh uživatelského rozhraní se snaží odpovídat na rozmanitost zařízení, ze kterých je v současné době k webovým stránkám přistupováno. V důsledku značně rozdílných vlastností jednotlivých zařízení (mobilních telefonů, tabletů a stolních počítačů) je totiž třeba do jisté míry upravovat i vzhled a rozložení zobrazované stránky tak, aby uživateli vždy byly dostupné důležité funkce systému, aby ovládací prvky byly dostatečně velké a umožňovaly tak pohodlnou interakci a aby prezentovaný obsah byl pro uživatele čitelný a dostupný bez nutnosti přibližovat obrazovku, zvětšovat zobrazení, nebo posouvat obraz do strany, jak tomu je na stránkách, které nejsou „responzivní“.

Tématikou tvorby takovýchto stránek se zabývá mnoho zdrojů, při tvorbě stránek budeme vycházet například z [11] a inspiraci hledat na [12]. Také dále v této kapitole vybereme vhodné nástroje, které by nám s tvorbou takovýchto stránek měli pomoci.

Vzhledem k tomu, že bychom měli myslet i na budoucnost aplikace a její převod na mobilní zařízení, tak při návrhu UI a jednotlivých prvků zohledníme částečně i Android guidelines [13], které shrnují doporučení pro návrh UI pro mobilní zařízení s operačním systémem Android, ale jsou obecně platné pro návrh aplikací pro mobilní zařízení. Řídit se budeme především obecně platnými radami, například o tom, že obrázky a barvy jsou pro uživatele rychleji rozpoznatelné, než slova, že by dotazy a položky ve formulářích měly být krátké, protože uživatelé dlouhý text přeskakují a dalšími. V Android Guidelines zmíněný moderní způsob návrhu uživatelského rozhraní, souhrnně označený jako „Material design“, taktéž poskytuje zajímavé a užitečné podněty, které můžeme zakomponovat do návrhu uživatelského rozhraní.

2.3.2 Rekapitulace požadavků na systém

Je dobré zmínit, že velká část dodatečných požadavků na systém uvedených v sekci 1.2.3 je zaměřena právě na kvalitu UI, což podtrhuje význam návrhu uživatelského rozhraní, uvedený v úvodu této kapitoly.

Jedná se o body „zaměření na uživatele“, podtrhující význam UX jako takového; „dostupnost a pohodlnost“, které se týkají přístupu z mobilních

zařízení; „skutečné zlepšení“, kterýžto bod popisuje nutnost přinášet uživatelům skutečně užitečné funkce v podobě, která je pro ně přívětivá (a tento je naprosto klíčový pro úspěch celého projektu) a bod o rychlosti získávání informací, pro jehož splnění je nutné správně strukturovat zobrazované informace i odkazy na ty informace, které momentálně zobrazované nejsou, ale systém je poskytuje.

Jak je vidět, právě tyto body značně ovlivňují výslednou podobu této práce a značný rozsah kapitoly o uživatelském rozhraní je jen odrazem jejich důležitosti. I z těchto požadavků tedy budeme vycházet při sestavování následující sekce.

2.3.3 Pravidla pro tvorbu UI

Pokud tedy shrneme závěry z řešerší, informace získané v předmětu NUR a doporučení z design guidelines, pak získáváme sadu pravidel, která se budeme snažit v této práci dodržovat, abychom co možná nejlépe splnili požadavky na systém.

Minimalizace počtu kliknutí

Problém zvláště u mobilních zařízení, důležitým parametrem pro pohodlné ovládání aplikace je schopnost dostat se „rychle“, kamkoli uživatel potřebuje. A mírou rychlosti je v našem případě počet obrazovek, které se musí načíst, než se uživatel dostane k požadované informaci, či funkci v systému. Tento parametr opět naráží na logaritmickou asymptotickou složitost korektně tříděných informací, v tomto případě se jedná o hloubku zanoření do pomyslného grafu tvořeného obrazovkami a odkazy v aplikaci.

Dostupnost informací a ovládacích prvků

S předchozím bodem, zaměřeným na průchod obrazovkami, silně souvisí také samotný obsah těchto obrazovek. V zájmu uživatele je, aby mu bylo prezentováno co možná nejvíce informací a nemusel po nich dlouho pátrat. Stejně tak není příjemné hledat daleko funkce pro interakci se systémem a entitami, které se v něm nacházejí, pročež je vhodné uživateli nabízet ovládací prvky na co možná nejobecnějších stránkách (na hlavní straně, nebo v seznamu daných entit) a nenutit ho, aby se musel kvůli nějaké funkci zanořovat příliš hluboko do systému .

Přehlednost

Protiváhu proti předchozím bodům představuje tento požadavek, vycházející z poznatku, že čím méně uživatel vidí, tím rychleji vybírá. Zvláště na mobilních zařízeních je potřeba omezit zobrazovaný obsah a prezentovat tak uživateli jen to nejdůležitější. Je například dobré míti na mysli, že ne všechny ovládací

prvky, které jsou uživateli dostupné ve verzi pro stolní počítače, musí být tak snadno dostupné i ve verzi mobilní. To by vedlo na příliš ochuzené desktopové rozhraní, nebo naopak na přeplněné rozhraní pro menší zařízení. Proto je vhodné například některé volby schovat pod tlačítko, po jehož stisknutí se uživateli zobrazí seznam dalších funkcí, které se jinak na displej nevešly. Tak může zkušený uživatel plnohodnotně aplikaci ovládat, pokud je zvyklý tyto funkce používat, zatímco tyto funkce nezabírají zbytečné místo.

Taktéž usnadnění navigace pomocí tlačítka domů a odkazů na nejdůležitější místa v systému, doplněných o indikátory, kde v systému se uživatel právě teď nachází a snadnou cestu zpět na hlavní stranu.

Velmi důležitým prvkem jsou také již výše zmiňované barvy – pro odlišení dnů, pro odlišení důležitosti prezentované informace apod. Je nutné všeho užívat s mírou, ale na základě rešerší můžeme jasně vidět, že barvy jsou velmi důležitým prvkem nejen grafického designu, ale i návrhu uživatelského rozhraní, ve kterém umožňují uživateli rychle předat informaci, kterou by jinak muselo vystihovat mnoho slov.

Omezení možných chyb uživatele

Nejlepší chyby jsou takové, které uživatel nikdy nemůže udělat. Proto je vhodné zvýraznit (ať už barevně, velikostí, nebo jinak) ovládací prvky, které typicky uživatel chce udělat (odeslat zprávu oproti tlačítku zpět na předchozí obrazovku), nebo naopak takové ovládací prvky, které provádí podstatné změny v systému (například červenou barvou označit tlačítka pro smazání záznamů v systému).

Stejně tak je vhodné mít na mysli, že nesouvisející ovládací prvky by neměly být uskupeny v těsné blízkosti (tak, jako bylo v některých zkoumaných konkurenčních systémech, ale i v předchozím prototypu Mojeskolka, kde volba pro potvrzení přečtení notifikace byla umístěna v sekci pro uzavírání notifikace a uživatelé byli zmatení).

Důležité jsou také jasné a krátké popisy tlačítek – jakou funkci plní a to i s ohledem na ostatní tlačítka (typická kombinace nějaké akce, její negace a tlačítka storno).

Přehlednost a použitelnost i za pochodu

Pokud chceme cílit na mobilní platformu, nemyslíme tím jen mobilní zařízení, ale uživatele, který se skutečně pohybuje – někam jde, jede hromadnou dopravou, nebo je jiným způsobem v pohybu. Je třeba snažit se minimalizovat množství možných chyb a přehmatů, které uživatel může udělat a prezentovat mu jednak informace velice přehledně, ale i ovládací prvky tak, abychom omezili nechtěné přehmaty.

Reflexe skutečného světa a světa uživatele, nikoli interního světa systému

Toto pravidlo platí obecně nejen při návrhu UI, ale i při tvorbě kvalitního objektového návrhu a v dalších situacích. Systém by měl reflektovat skutečný svět a mluvit jazykem uživatele, tedy strukturovat data tak, jak je uživatel zvyklý, používat ikony a označení, která uživatel bude chápat. Bude se tak cítit ve známém prostředí a jeho dojem z užívání systému bude lepší.

Nezávaznost akcí uživatele

Uživatel by neměl mít pocit, že je svazován prostředím, ve kterém jakýkoli přešlap může znamenat nenávratnou ztrátu dat, nebo jiný problém. Naopak by měl mít možnost systém procházet a mít možnost vzít zpět akce, jejichž důsledek mu nevyhovuje. Tento bod vychází z Nielsenovy heuristické analýzy, budeme se na něj snažit pamatovat i při tvorbě systému a především při jeho rozvoji do budoucna[12].

Závěr

V rámci tvorby jednotlivých stránek uživatelského rozhraní se pak budeme na tyto pravidla a principy myslet, abychom na jejich základě činili co možná nejlepší rozhodnutí.

2.4 Technologie

V kapitole 1 bylo rozhodnuto, že učitelská i rodičovská část systému Mojeskolka budou realizovány jako webová aplikace, z čehož vyplývá architektura klient-server, kde klient poběží ve webovém prohlížeči a se serverem bude komunikovat pomocí protokolu HTTP.

V této sekci vybereme nástroje pro implementaci uživatelského rozhraní, které navrhujeme v rámci této kapitoly a bude prezentováno klientovi ve webovém prohlížeči.

Obvyklým způsobem realizace je užití HTML, CSS a Javascriptu pro tvorbu interaktivní webové stránky a vzhledem k masové rozšířenosti tohoto řešení i k vazbě jazyka PHP na zmíněné technologie použijeme pro implementaci právě tyto technologie. Samotnou realizací se zabývá sekce Tvorba prototypu.

[12]Realizace zcela „nezávazných“ akcí je poměrně složitý proces a proto bude moct být v prototypu implementována pouze v omezené míře, ale opět se jedná o důležitý poznatek do budoucna

2.4.1 Výběr grafického frameworku

V rámci iterativního přístupu k návrhu, tedy i návrhu UI, bude nyní vhodné vybrat nástroj, který nám později pomůže s implementací grafického uživatelského rozhraní. Znalosti o něm budeme totiž moci již nyní využít při konkrétním návrhu jednotlivých stránek uživatelského rozhraní, ale zároveň už při výběru tohoto frameworku máme velmi detailní informace o požadavcích na něj kladených, jelikož z předmětu NUR již máme návrhy a prototypy jednotlivých stránek hotové. Jedná se tedy o další příklad iterativního přístupu: nejprve jsme vytvořili hrubý návrh, na základě něj vybereme vhodný nástroj, který následně můžeme zohlednit při dalším, jemnějším návrhu, ze kterého bude vycházet další verze implementace.

Grafický framework by nám měl pomoci s realizací uživatelského rozhraní, tedy v našem případě při tvorbě HTML, CSS a Javascriptu, ve kterých budou implementováni klienti pro přístup do Mojeskolka. Pro tento účel je možné použít mnoho různých grafických knihoven a frameworků, avšak jako nejvhodnější pro realizaci aplikací v systému Mojeskolka se jeví grafický framework Bootstrap, případně některá z šablon v něm vytvořených. Tento framework byl vybrán na základě jeho rozšířenosti, diskusí s odborníky a na základě skutečnosti, že přesně pokrývá naše požadavky na urychlení vývoje UI a kvalitu užitého nástroje. Zároveň bylo v rámci rešerší zjištěno, že systém Edookit, který se jeví jako velmi vyspělá a moderní webová aplikace, taktéž používá Bootstrap, což jen poukazuje na potenciál tohoto frameworku v oblasti produktů podobných projektu Mojeskolka.

Výběr varianty Bootstrapu

V rámci frameworku Bootstrap existuje rozsáhlá sada různých šablon, které se zaměřují na konkrétní použití tohoto nástroje. Po projití stránek nabízejících různé šablony o různé míře specializace (například [14], [15] a [16]) se však zdá, že pokud nechceme použít již velmi specializovanou šablonu, pak pro rodičovské rozhraní není přílišného rozdílu mezi jednotlivými možnostmi. A protože, jak bylo zmíněno výše, grafický design bude v budoucích fázích projektu ponechán profesionálovi, specializovaná šablona není vhodná, jelikož řeší v první řadě právě grafický design.

Výběr varianty určené pro rodičovské rozhraní tedy, po prozkoumání mnoha dalších, padl na jednu z jednodušších možností – šablonu Cerluean (detaily jsou dostupné na jejích stránkách [17]). Jedná se o poměrně jednoduchý základ, který však poskytuje solidní výchozí bod a vhodně stylizované prvky UI pro urychlení vývoje.

Pro učitelské rozhraní je však situace poněkud odlišná. Jedná se o velmi typickou aplikaci, kde zároveň nepotřebujeme klást důraz na originalitu, a proto můžeme použít již existující řešení pro tvorbu rozhraní podobného druhu. Z procházených možností se nejlépe jeví šablona Admin LTE (více o této ša-

bloně je možné najít na příslušných stránkách [18]), která obsahuje již velmi detailně zpracovanou kostru uživatelského rozhraní pro podobný typ aplikace.

Výhodou tohoto řešení je, že CSS styly specifické pro náš návrh a grafický design mohou být aplikovány „přes“ styly vybrané šablony, do kterých nebude nijak zasahováno, pročež by případná náhrada zvolené šablony za jinou neměla být příliš bolestivá, kdyby se naše volba v budoucnu ukázala nevhodnou.

Za zmínku stojí také volba mezi použitím LESS a Saas verze frameworku Bootstrap, tedy výběr použité varianty preprocesoru CSS, což je nástroj, který výrazně usnadňuje tvorbu kaskádových stylů a tedy celého GUI aplikace. Po přečtení článků zaměřených na tuto tématiku (mimo jiné [19], [20]) se zdá, že ačkoli většina autorů zmiňuje větší sílu a širší možnosti Saas, tak pro vývojáře, který nemá v oblasti CSS preprocesorů zkušenosti, je vhodnější volbou LESS kvůli jednoduchosti přechodu od CSS a snadnějšímu užívání. Zároveň je Bootstrap primárně zaměřený na LESS, do Saas je pouze portován. Z obou těchto důvodů bylo rozhodnuto pro LESS jako vhodnější volbu.

CSS preprocesor LESS tedy bude používán dále v tomto projektu pro usnadnění tvorby CSS souborů.

2.5 Obecný úvod k tvorbě uživatelských rozhraní

Nyní máme připraveny všechny podklady a můžeme přistoupit k samotnému návrhu UI. V následujících sekcích bude popsán návrh jednotlivých stránek v rodičovském i učitelském uživatelském rozhraní. Výstupem bude jak slovní popis nutných změn a řešení problémů souvisejících s rozložením obsahu dané stránky, tak grafický návrh v podobě takzvaných mockupů, tedy stručných nákresů zachycujících rozložení prvků na stránce oproštěné od grafického designu.

Mockupy plní obdobnou funkci, jako třídní diagramy a další nástroje užívané při návrhu aplikační logiky – umožňují nám hrubě navrhnout nějaký aspekt systému, který díky tomu můžeme lépe pochopit a v případě nalezení problémů v hrubém návrhu je cena za opravu této chyby výrazně nižší, než poté, co byla daná část systému implementována. Opět se jedná o formu iterativního přístupu, v tomto případě vycházející z metodik návrhu UI.

Návaznost na předchozí práci

Jak již bylo zmíněno v úvodu této kapitoly, jedná se o druhou iteraci návrhu níže uvedených stránek, proto si můžeme dovolit být o něco stručnější a zaměříme se pouze na hrubý popis funkce dané stránky a rozbor nutných změn.

Cílem bude opravit chyby, které byly nalezeny při testování původního prototypu aplikace, a zdokonalit jednotlivé stránky na základě nově nabytých poznatků a opětovného vyhodnocení priorit a požadavků na stránky, ke kterému došlo v předcházejících částech této práce. U některých stránek se může

jednat jen o velmi drobnou sadu změn, proto budou jim příslušné subsekce poměrně stručné.

Stejně tak některé stránky nebudou nikterak zajímavé a bude možné při jejich tvorbě vyjít z typických exemplářů podobných stránek (například stránka pro přihlašování, stránky obsahující pouze prostý text a další), proto nebude potřeba k nim vytvářet mockup návrhu UI.

Responzivní návrh pro různé rozměry zobrazení

V předchozích sekcích zmiňovaný responzivní design byl zvolen jako primární směr při navrhování uživatelského rozhraní. To znamená, že na různých zařízeních budou mít stránky systému Mojeskolka jiné rozložení a tato skutečnost musí být zohledněna při jejich návrhu.

Toho bude dosaženo jednak různými mockupy, jednak za pomoci frameworku Bootstrap.

Co se mockupů týče, návrhy stránek pro malé rozměry displeje byly vytvořeny již v rámci předmětu RUN, kde byly navrženy a uživatelsky otestovány, a v této práci na předchozí mockupy navážeme a vytvoříme verzi pro střední rozměry stránek.

To by samo o sobě nemuselo pro detailní pokrytí postačovat, jelikož některé stránky budou přepracovány a budou navrhovány i stránky zcela nové, avšak responzivní design bude zaručen díky grafickému frameworku Bootstrap. Ten umožňuje, aby stránky byly responzivní „automaticky“ – primárně je totiž zaměřen právě na tvorbu responzivních webových stránek a při dodržování základních pravidel pro tvorbu UI je velice jednoduché takovéto stránky vytvořit.

Vazba návrhu na použití Bootstrapu bude znázorněna na vytvářených mockupech jednotlivých stránek – typicky na nich budou viditelné sloupce, do kterých je obsah stránky rozložen. V okamžiku, kdy by se takováto stránka zobrazovala na obrazovce o menších rozměrech, než je v Bootstrapu definovaná mez, budou tyto sloupce postupně umisťovány vertikálně nad sebe, podle zvolených kritérií. Pro příklad tohoto postupu je možné podívat se na mockupy hlavní stránky rodičovského rozhraní, která se takto bude měnit.

Tento způsob automatického přeuspořádávání prvků na obrazovce bude tedy použit u všech stránek, kde by se prvky jinak nevešly na obrazovku, pokud nebude v textu uvedeno jinak. Tím je zajištěno Responzivní chování stránek a tudíž jejich použitelnost na mobilních zařízeních.

Rozdělení na učitelské a rodičovské rozhraní

Jak bylo zmíněno výše v kapitole Rozdělení na jednotlivá rozhraní, důraz bude kladen více na rodičovské rozhraní, než na učitelské. Důvodem je, že rozhraní pro rodiče je úzce spřažené s doménou a s poskytovanými funkcemi, navíc musí dokázat oslovit uživatele a zaujmout ho, zatímco rozhraní pro učitele řeší

problémy typické pro běžné administrátorské rozhraní, navíc při jeho tvorbě budeme do značné míry vycházet z vybrané šablony grafického frameworku, pročež není třeba mnohé stránky detailně navrhovat.

Komponenty UI

V následujících sekcích budeme kromě celých stránek hovořit i o grafických komponentách, které jsou používány na více různých stránkách. Jedná se například o krátký výpis notifikace, nebo vysunovací menu s funkcemi týkajícími se uživatelského profilu.

Tyto komponenty reprezentují často se opakující prvky, které mají jednotný design napříč všemi stránkami, kde jsou zobrazovány, proto budou popsány pouze na jednom místě a dále už bude v příslušných mockupech s jejich přítomností počítáno, případně budou znázorňovány jako prázdné obdélníky s příslušným názvem.

Vazba návrhu na funkcionalitu

Je dobré připomenout, že ačkoli v této sekci navrhujeme uživatelské rozhraní, tedy grafickou podobou stránky, primárním cílem UI je zpřístupnit funkce systému jeho uživateli, tedy rozložit poskytované informace a ovládací prvky na obrazovce tak, aby systém bylo možné co nejsnáze ovládat. Proto v rámci návrhu jednotlivých stránek zmíníme i funkce realizované na těchto stránkách a budeme z nich vycházet, co se prvků na stránce týče.

2.6 Stránky a komponenty společné pro obě rozhraní

V aplikaci jsou funkce a tudíž i k nim odpovídající stránky, nebo komponenty, které jsou shodné pro obě rozhraní, učitelské i rodičovské. Mohou být zasazené v jiném kontextu menu a postranních panelů a mohou používat jiné CSS styly pro dosažení mírně odlišného vzhledu, ale z hlediska návrhu UI se jedná o stejné prvky.

Tyto stránky a komponenty budou jednotně popsány v následující sekci.

2.6.1 Přihlášení

Jedná se o velmi typickou a nikterak zajímavou stránku, která představuje vstupní bod do systému. Nepřihlášeným uživatelům je poskytnuta pouze stránka „O aplikaci“ a možnost přihlásit se.[13] Jedinou zajímavostí na této stránce je přítomnost odkazu na stránku pro pomoc s přihlašováním, která je uvedena níže.

[13] Do budoucna se počítá s více sdílným portálem pro veřejnost, ale ten není předmětem této práce.

Z hlediska funkce této stránky je dobré zmínit, že bylo potřeba rozhodnout, na základě kterého identifikátoru se budou uživatelé přihlašovat do aplikace (tedy například zda to bude na základě uživatelského jména; id školky a jména rodiče; nebo na základě jiného identifikátoru). Po vzoru moderních aplikací bylo nakonec rozhodnuto, že nejvhodnějším identifikátorem bude e-mail uživatele.

2.6.2 Pomoc s přihlašováním

Tato stránka vychází z případu užití „vyřešení problémů při přihlašování“ a je určena pro uživatele, kteří potřebují pomoc s přihlášením. Cílem je poskytnout uživateli užitečné rady, jak postupovat, pokud se mu nedaří přihlásit se, a na jaké různé subjekty a jakým způsobem se může obrátit s žádostí o pomoc.

Z hlediska UI se jedná o prostou stránku s blokem textu. Vzhledem k tomu, že takovýchto stránek je v aplikaci více, jejich detailnějším rozborem se zabývá následující subsekce.

2.6.3 Prostá stránka s textem

Jedná se o typ jednoduché stránky, jejíž jediným cílem je sdělit uživateli informaci v podobě bloku textu, který nevyžaduje žádné další složité formátování (obsah stránky se dá přirovnat k textovému dokumentu, který obsahuje nadpisy a odstavce, ale jinak je jednolitý).

Ovládací prvky jsou omezeny na případné „vpřed“, „zpět“, nebo podobné tlačítko, ale k žádným jiným úkonům stránka neslouží..

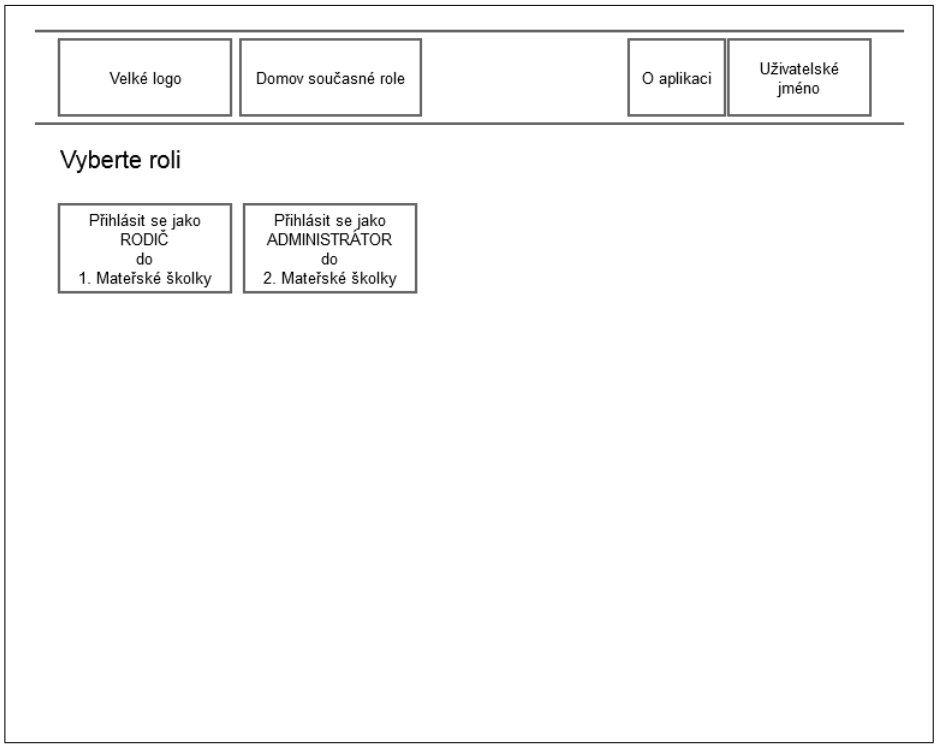
Menu závisí na kontextu, tedy například, zda je uživatel vůbec přihlášený, případně, zda jako rodič, nebo učitel, ale samotné tělo stránky je velice prosté.

2.6.4 Výběr role

Po úspěšném přihlášení do systému je stránka, kterou uživatel uvidí jako první, závislá na jeho přiřazených rolích a oprávněních (o těch blíže pojednává sekce Uživatelské role). Pokud má pouze jednu roli, je přesměrován přímo na domovskou stránku této role, tedy například na hlavní stránku rodičovského rozhraní.

Pokud však má uživatel v systému více rolí (například je administrátor ve školce, kam zároveň chodí jeho děti a proto je v ní zároveň evidován jako rodič), je nejprve potřeba zvolit, v jaké roli chce k systému přistupovat. A právě tento výběr je realizován na této stránce.

Stejně tak se uživatel může rozhodnout, že by rád změnil roli, jakou v daném okamžiku v aplikaci zastává. Proto může přejít na tuto stránku a vybrat si jinou roli, než v jaké byl do té doby přihlášený.



Obrázek 2.1: Mockup stránky pro volbu uživatelské role.

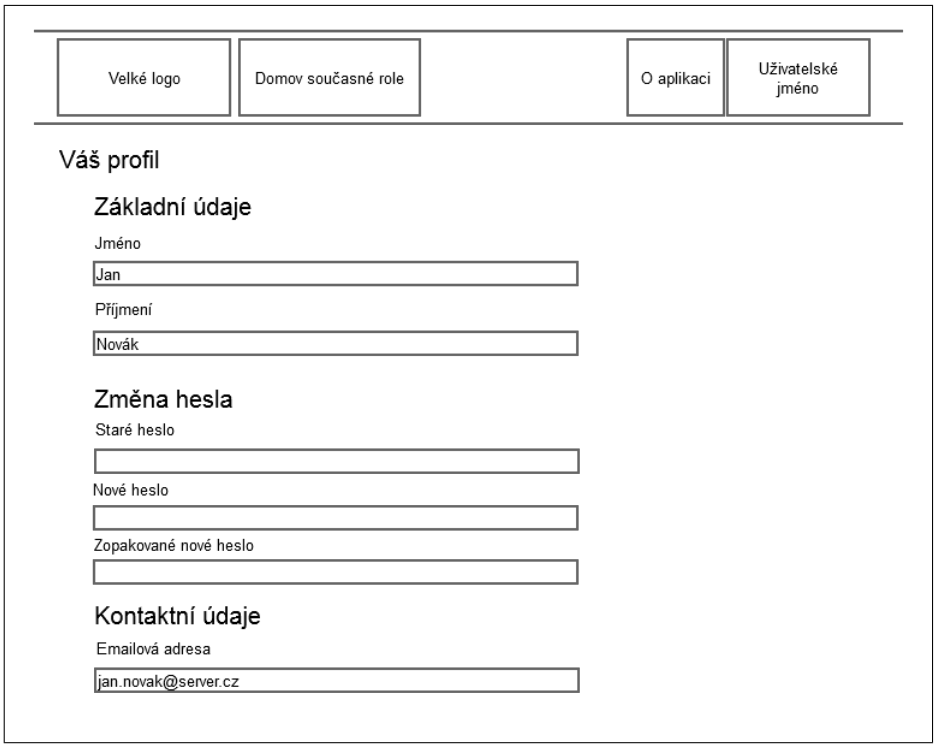
Z hlediska UI se jedná o jednoduchou stránku, jejíž menu vychází z menu použitého v rodičovském rozhraní (kterému se věnuje samostatná subsekce níže), ale obsahuje pouze omezené funkce, sdílené pro všechny role. Odkaz „Domov současné role“ slouží pro rychlý přechod na stránku role, ve které byl uživatel naposledy přihlášený.

2.6.5 Uživatelské funkce

Pro funkce související s vlastním profilem uživatele jsou v aplikaci přítomny jednak celé stránky a jednak samostatné komponenty.

Stránka profil uživatele

Krom výše uvedené stránky pro přihlášení a změnu role má uživatel stránku svého profilu, kde může měnit své kontaktní údaje, heslo a další.



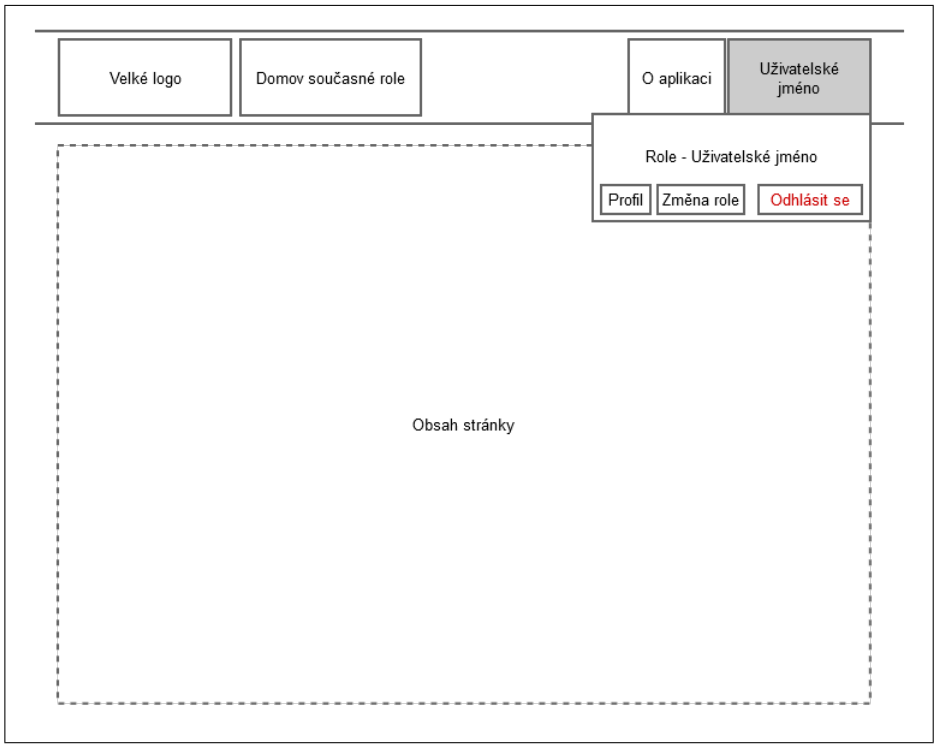
Obrázek 2.2: Mockup stránky profilu uživatele.

Jedná se o poměrně jednoduchou stránku, která plní roli obvyklou v systémech podobných Mojeskolka.

Komponenta uživatelské funkce v menu

Pro přístup ke stránce profilu a k dalším uživatelským funkcím slouží komponenta, která je vždy umístěna v pravém horním rohu obrazovky v horním menu.

Na předchozím nákresu je označována jako Uživatelské jméno, jelikož na tomto místě skutečně bude uvedeno jméno uživatele, ale po kliknutí na toto tlačítko se uživateli zobrazí panel, který zmiňuje roli, ve které je uživatel přihlášen, a poskytuje základní uživatelské funkce.



Obrázek 2.3: Mockup komponenty uživatelských funkcí.

Na místo „Role – Uživatelské jméno“ budou pochopitelně uvedeny korektní údaje, tedy například „Jan Novák – Rodič“.

2.6.6 O aplikaci

Jedná se o další prostou stránku s blokem textu. Jejím účelem je poskytnout základní informace o aplikaci, nikoli z hlediska jejího užívání, k čemuž slouží stránka nápověda, ale z hlediska popisu samotného produktu. Do budoucna se zde může nacházet verze aplikace, kontakt na provozovatele a další informace.

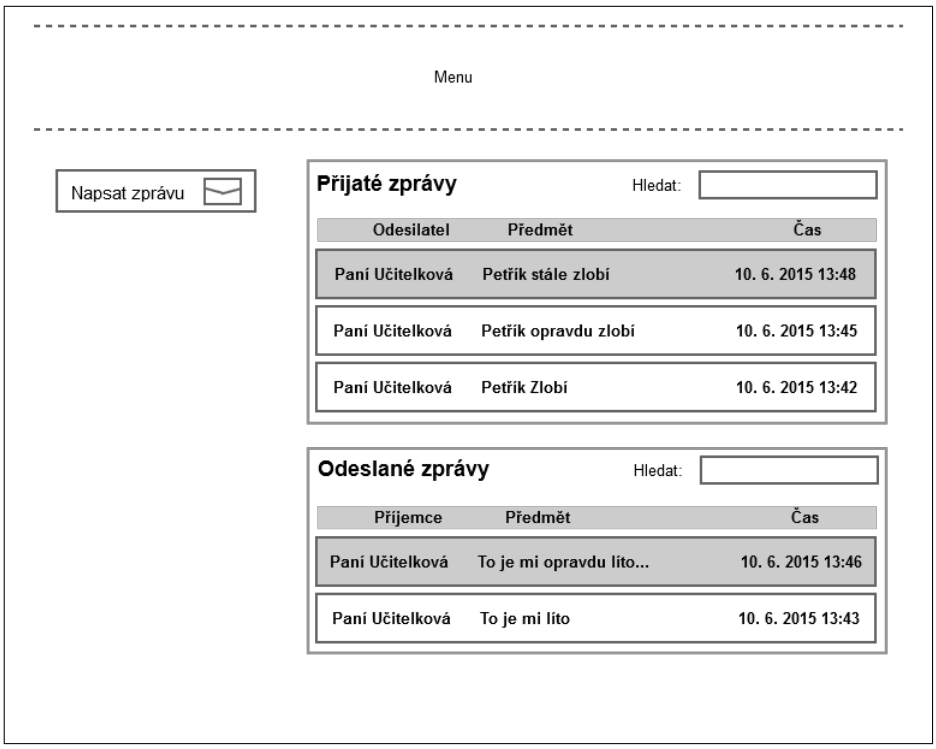
2.6.7 Zprávy

Tato skupina stránek vychází přímo z funkčních požadavků – komunikace mezi učiteli a rodiči pomocí zpráv je pro aplikaci klíčová. Tuto funkci zajišťuje několik stránek a komponent uvedených níže.

U všech mockupů uvedených stránek je menu uvedené v záhlaví závislé na konkrétním rozhraní – ve smyslu rodičovského, nebo učitelského – a proto je pouze naznačeno.

Hlavní strana zpráv

Tato stránka slouží jako rozcestník pro psaní a čtení zpráv. Připomíná jednoduchého e-mailového klienta a má i podobné funkce, až na to, že se jedná o interní zprávy v systému, nikoli klasickou elektronickou poštu.



Obrázek 2.4: Mockup hlavní stránky uživatelských zpráv.

Barevně odlišené řádky na obrázku značí nepřečtené zprávy.

Nová zpráva

Na této stránce je uživateli prezentován formulář pro odeslání nové zprávy. Opět se jedná o podobnou stránku, jakou je možné najít v klientech elektronické pošty.

Obrázek 2.5: Mockup stránky pro napsání nové zprávy.

Za zmínku stojí, že pole „Adresát“ umožňuje vyhledávání, což je na obrázku znázorněno. Takto rozevřená nabídka adresátů překrývá pole „Předmět“, které však je ve formuláři přítomné. Zároveň je tlačítko pro odeslání zprávy, což je očekávaná akce uživatele, výrazně větší, než tlačítko pro návrat zpět.

Detail zprávy

Na této stránce je možné přečíst si odeslanou, nebo přijatou zprávu. Na zprávu je taktéž možné pomocí jednoho tlačítka „Odpovědět“. Na obrazovce není žádný zajímavý prvek, který by vyžadoval uvedení mockupu této stránky, jedná se o prosté zobrazení zprávy ve stylu klienta elektronické pošty.

Komponenta zprávy s ikonou označující počet nepřečtených zpráv

V rámci požadavků na systém jsme definovali cíl, aby byly podstatné informace uživateli předkládány viditelně a aby k nim měl co možná nejrychleji přístup. Také v rámci heuristické analýzy, kterou jsme analyzovaly systémy v rámci rešerší, existuje bod, který říká, že stav systému by měl být vždy viditelný.

Proto by bylo vhodné, aby uživatel byl informován o nových zprávách bez nutnosti kontrolovat stránku přijatých zpráv. Z tohoto důvodu bude v menu

vždy na místo prostého odkazu do sekce zpráv umístěna i ikona obsahující číslo určující počet nepřečtených zpráv určených pro přihlášeného uživatele. Ačkoli bude v učitelské i v rodičovské aplikaci do sekce zpráv přistupováno jiným způsobem, odkaz na zprávy bude umístěn ve viditelném menu a proto bude informovanost uživatele o stavu zpráv zaručena.

2.7 Komponenty v rodičovském rozhraní

Mnohé prvky uživatelského rozhraní určeného pro rodiče se budou opakovat napříč stránkami a jsou natolik důležité, že zasluhují podrobnější popis, kterému je určena tato sekce.

Užití barev

Na tomto místě je vhodné podotknout, že v rodičovském rozhraní budou hojně užívány barvy pro doplnění informací, například ohledně data notifikace; dítěte, kterého se týká nebo důležitosti zprávy, kterou notifikace nese.

Ačkoli se na první pohled může zdát, že barvy jsou záležitostí designu grafiky, v příslušném kontextu se naopak jedná o velmi důležitý prvek návrhu uživatelského rozhraní. Jak bylo zmíněno v sekci Design guidelines a responzivní design, uživatel se mnohem rychleji orientuje podle obrázků a barev, než podle prostého textu.

Nebylo snadné vymyslet systém, který by dokázal využít jasné a výrazné barvy označující různé druhy informací vedle sebe na jedné stránce, ale v kombinaci s vhodným prvkem, který je obarvován (celé panely a jejich ohraničení, oproti oddělujícím horizontálním blokům, nebo pouze jinak barevným ikonám) je takovýto záměr možné realizovat. Snaha byla udržet i konzistentní barevné označení napříč různými stránkami.

2.7.1 Notifikace

Notifikace jsou klíčový koncept v celém systému Mojeskolka a proto je potřeba jim věnovat zvýšené pozornosti. Zároveň se jedná o prvek, který se v různých podobách vyskytuje na více různých stránkách (hlavní strana, nástěnka dítěte a detail notifikace) a ačkoli se jeho podoba pro různé stránky v některých detailech může lišit, základní podoba a problematické části jsou pro všechny stránky shodné.

V aplikaci se vyskytují různé komponenty související s notifikacemi velmi hojně a takovýchto komponent existuje několik druhů:

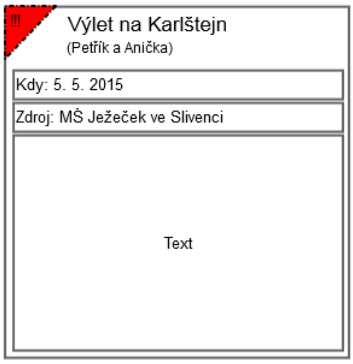
Krátký výpis notifikace

Jedná se o krátký blok stručně zachycující obsah notifikace. Skupiny těchto krátkých výpisů jsou zobrazovány v seznamech notifikací a kliknutím na krátký

výpis je možné přejít na detail příslušné notifikace. Cílem tohoto prvku UI je v rychlosti uživateli prezentovat podstatné informace z dané notifikace, proto by měl být objem podávané informace poměrně velký na takto malou plochu.

Toho je dosaženo několika grafickými prvky. Prvním z nich je rozličné orámování notifikace, které značí, zda se jedná o čerstvou, tedy ještě neotevřenou notifikaci, notifikaci již zobrazenou, nebo notifikaci starou(případně skrytou a znovu zobrazenou). Tato informace je dále doplněna druhým z grafických prvků – ikonou vykřičníku v případě důležitých notifikací. Barva tohoto vykřičníku určuje, zda se jedná o notifikaci, která vyžaduje potvrzení o přečtení a ještě nebyla potvrzena, pak je vykřičník označen červenou barvou. V případě, že již byla notifikace uživatelem potvrzena, vykřičník změní barvu na zelenou. Naopak oranžový vykřičník značí, že daná notifikace byla změněna a uživatel by si ji měl znovu přečíst. Po otevření notifikace oranžový vykřičník zmizí.

Cílem všech těchto mechanismů je informovat uživatele co možná nejlépe o obsahu a významu dané notifikace [14].



Obrázek 2.6: Mockup komponenty zobrazující stručný výpis notifikace.

Za zmínku stojí, že na obrázku je znázorněna notifikace, která vyžaduje potvrzení, proto užití červeného zdůraznění.

Seznam notifikací

Notifikace jsou vypisovány ve skupinách podle toho, k jaké entitě přísluší. Tento seznam zároveň obsahuje informace doplňující vypsané notifikace.

Řazení notifikací

Ze závěru testování prototypu vytvořeného v rámci NUR jasně vyplývalo, že uživatelům velmi vadí, pokud notifikace nejsou seřazené podle data. Ačkoli se jednalo o chybu prototypu a nikoli úmysl, jednoznačně to poukazuje na význam takovéhoto nedostatku.

[14] Tato specifika krátkého výpisu notifikace vysvětluje stránka nápovědy.

Primární rozdělení notifikací do sekcí bude podle entit, ke kterým náleží (panely entit se zabývá následující sekce), ale výpisy notifikací týkajících se dané entity, například dítěte, budou řazeny podle data události, o které je uživatel notifikován.

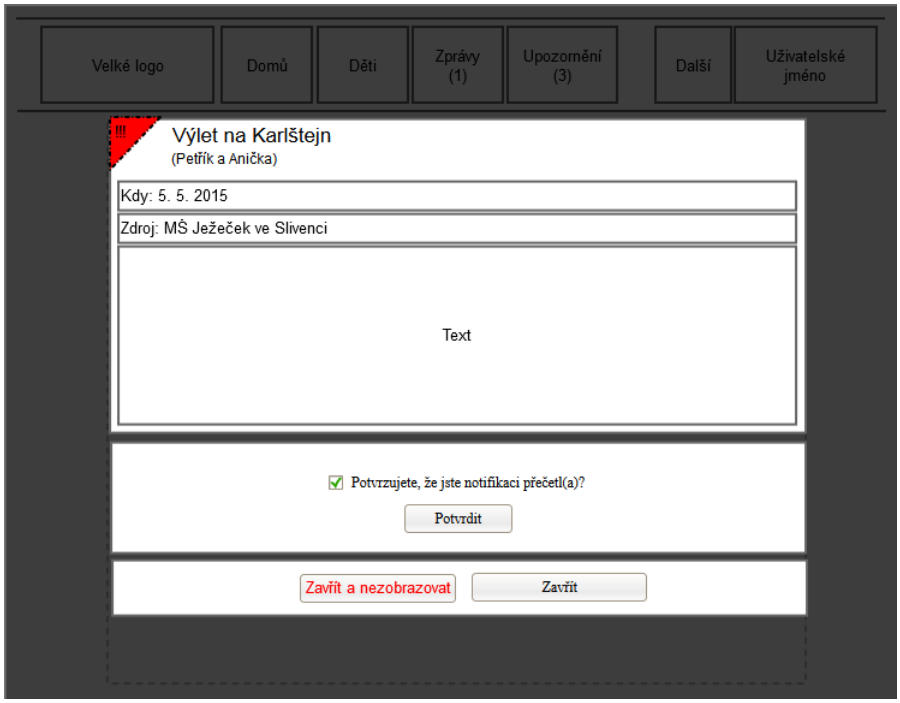
Stojí za zmínku, že v souladu s případem užití „Skrytí notifikace, se kterou již je uživatel srozuměný“ je možné některé notifikace skrýt. To by však mohlo vést na neúmyslné opomenutí důležitých úkolů. Proto v okamžiku, kdy už je notifikace na zítřejší, nebo dnešní den, je uživateli znovu zobrazována v méně výrazné podobě, avšak viditelně. Cílem této výjimky je zamezit nechtěnému opomenutí notifikace, kterou uživatel četl a nechal si přestat zobrazovat již před delší dobou.

Mockup seznamu notifikací je zahrnut níže v mockupu panelu entity.

Detail notifikace

Pro interakci s notifikací, tedy potvrzení o přečtení, nebo skrytí notifikace, slouží detail notifikace, který zároveň ukazuje celou délku textu notifikace i s případnými dalšími přílohami. Komponenta detailu se po kliknutí na krátký výpis příslušné notifikace otevře v podobě dialogu překrývajícího původní stránku.

Tato obrazovka umožňuje realizaci scénářů „Zobrazení detailů notifikace“, „Potvrzení přečtení notifikace“ a „Skrytí notifikace, se kterou již je uživatel srozuměný“ zároveň by však měla být primárně zaměřena na prezentaci důležitých informací. Že to není zcela triviální úděl se potvrdilo při testování předchozího prototypu, kde vyšlo najevo, že uživatelé mají problém s rozlišením mezi potvrzením o přečtení notifikace a mechanismem zavírání a skrývání notifikace. Proto byl při tvorbě nové podoby detailu notifikace kladen důraz na oddělení těchto dvou mechanismů do samostatných sekcí na obrazovce, jak znázorňuje následující mockup.



Obrázek 2.7: Mockup otevřeného detailu notifikace.

Aby bylo možné potvrdit přečtení notifikace tlačítkem, je nejprve nutné zaškrtnout příslušnou volbu nad ním, čímž je zaručeno, že uživatel neudělá tuto volbu omylem.

Zároveň je tlačítko „Zavřít a nezobrazovat“ zcela skryté, dokud uživatel nepotvrdí přečtení notifikace. Po jeho zobrazení je zvýrazněno červenou barvou, aby uživatel omylem neskryl notifikaci, kterou ještě chce vidět.

Nastavení zobrazovaných notifikací

Tato komponenta je použita na stránce dítěte pro zobrazování starých, skrytých, nebo budoucích notifikací, což vychází z případu užití „Zobrazení všech notifikací daného dítěte“.

Při testování předchozího prototypu z předmětu NUR bylo jasně vidět, že mechanismus, který byl použit pro zobrazování skrytých notifikací, uživatelům není blízký a nebyli schopni ho korektně používat.

Proto bylo toto nastavení přesunuto ze zápatí stránky přímo k seznamu notifikací, označeno vhodnou ikonou oka a textem „Zobrazit více“, což by nyní mělo být pro uživatele výrazně srozumitelnější.

Samotné nastavení zobrazovaných notifikací se skládá z seznamu checkboxů odpovídajících jednotlivým skupinám skrytých notifikací – tedy příliš staré, příliš budoucí, nebo uživatelem skryté – a tlačítka „Zobrazit“.

Mockup této komponenty je možné nalézt níže v kontextu jeho používání v subsekci věnované stránce dítěte.

2.7.2 Panel entity

V rodičovském rozhraní je potřeba rozlišovat mezi informacemi, které přísluší různým instancím entit stejného druhu. Jmenovitě mezi jednotlivými dětmi a případně školkami, pokud děti dochází do více různých školek. Zároveň je tato informace, tedy kterého dítěte se týká daná notifikace, nebo přímo celá stránka, velmi důležitá a měla by být uživatelem snadno rozpoznatelná.

Podobně jako u řazení notifikací vyplynulo z testování předchozího prototypu, že uživatelé mají problém rozlišit, kterého dítěte se daná notifikace, nebo celý výpis notifikací týká.

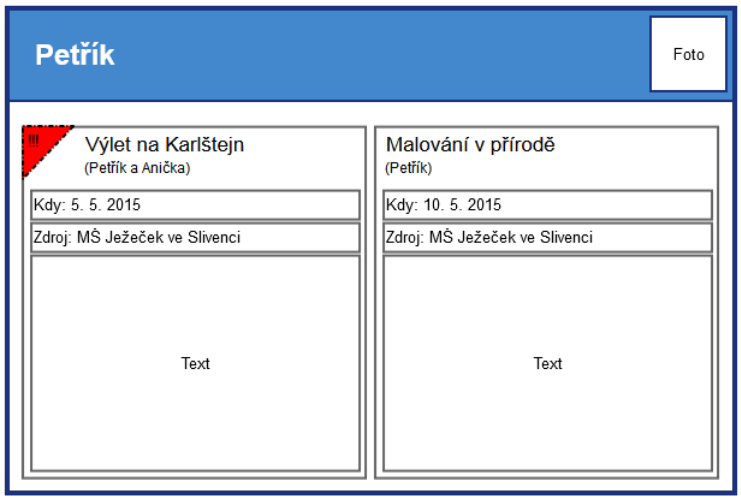
Proto jsou informace příslušící danému dítěti, nebo školce, zobrazovány na stránkách rodičovského rozhraní v rámci panelů, které umožňují snadnou orientaci v prezentovaných informacích.

Tyto panely jednak opticky rámují prezentované notifikace, nebo jiné informace (například adresu u školky, učitele u dítěta a další) a jednak samy o sobě mohou nést informaci pro uživatele. Tou je konzistentní barevné označení dané entity v rámci rozhraní daného uživatele. To například znamená, že když na hlavní straně bude chlapec jménem Petřík označen modrou barvou a v jeho panelu budou umístěny všechny jemu příslušící notifikace, tak jednak uživatel podle rámce okolo seznamu notifikací hned vidí, že se jedná o notifikace pro Petříka, a jednak ví, že po přechodu na stránku dítěte budou informace týkající se Petříka opět ohraničeny modrou barvou. Tímto způsobem se uživatel snadno může orientovat, jakého dítěte se týkají prezentované informace.

Pro odlišení mezi dětmi a školkami (které mají také vlastní sekce a vlastní stránky) je užito jednak odlišné barevné palety, ale především šrafovaného pozadí panelu dané školky. Toto označení by mělo umožňovat snadné odlišení mezi informacemi týkajícími se celé školky a informacemi pro jednotlivé děti.

Podobné panely, ale méně výrazných barev, jsou použity i pro profily učitelů v rodičovském rozhraní a pro orámování jednotlivých notifikací pro jasné vymezení obsahu.

Toto uspořádání do panelů, nebo čistých, výrazných bloků, chceme-li, zároveň poskytuje další výhody. Panel ve svém záhlaví obsahuje místo pro další informace, například fotku dítěte, ale i školky, nebo učitele, či informaci o tom, koho se daná notifikace týká, nebo které děti uživatele daný učitel vyučuje. Celkově by se mělo jednat o prvek, který uživateli výrazně usnadňuje práci se systémem Mojeskolka.



Obrázek 2.8: Mockup komponenty panelu příslušícího instanci entity v rodičovském rozhraní.

Na mockupu je vidět využití barev v rámci designu UI (nejedná se o využití barev pro grafický design). Panel entity s nadpisem „Petřík“ obsahuje seznam jednotlivých krátkých výpisů notifikací. Příklad použití panelu entity v kontextu poskytuje mockup hlavní stránky rodičovského rozhraní[15].

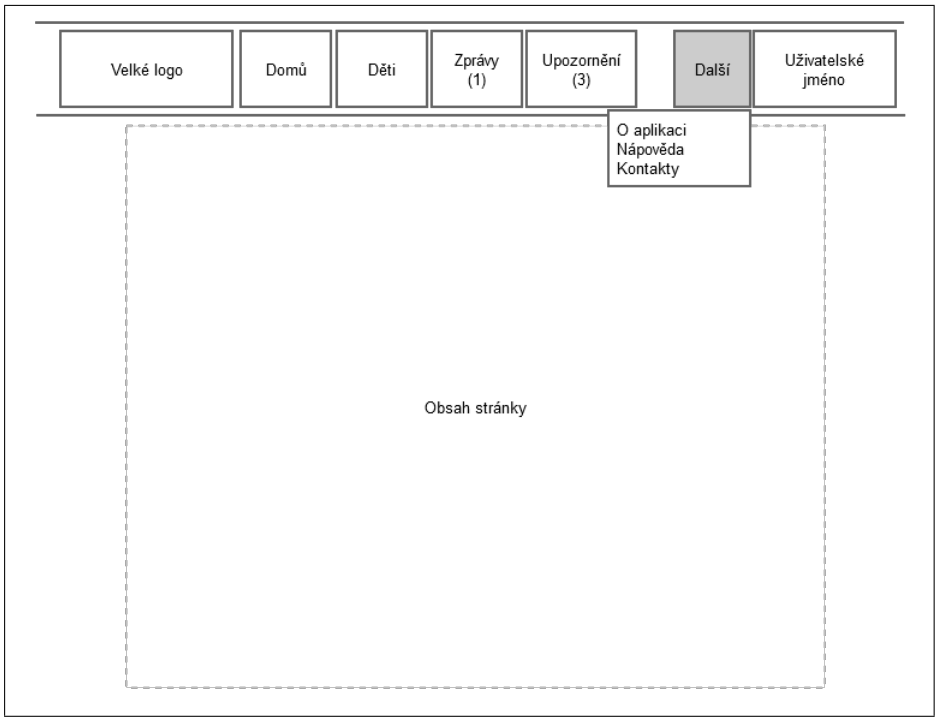
2.7.3 Společné záhlaví

V rámci rodičovského rozhraní je užíváno společné záhlaví, obsahující menu pro navigaci a logo aplikace. Toto logo je samo o sobě odkazem na stránku „domů“, která je navíc v menu uvedena explicitně pro větší přehlednost. U zpráv je uveden počet nepřečtených zpráv, jak bylo zmíněno výše a stejným způsobem je realizováno upozornění na počet nevyřešených důležitých notifikací v aplikaci u položky „Důležité“.

Položka „Další“ obsahuje méně často využívané funkce, jako je odkaz na nápovědu, stránku o aplikaci a stránku kontaktů.

Při návrhu vzhledu menu byl zohledněn použitý framework Bootstrap a jím poskytovaný vzhled menu, což by mělo vést k usnadnění implementace a využití již ověřeného, pro uživatele srozumitelného vzhledu menu.

[15]Toto zobrazení odpovídá panelu dítěte zobrazovaném na jeho vlastní stránce, jelikož notifikace „Výlet na Karlštejn“ je určena i Petříkovi, i Aničce a na hlavní stránce by tak byla zobrazena v sekci „Sdílené“, jak bude vysvětleno níže.



Obrázek 2.9: Mockup menu rodičovského rozhraní.

Mockup znázorňuje menu po kliknutí na položku „Další“, která otevře rozbalovací menu s příslušnými položkami. Stejným způsobem se rozbalí seznam dětí po klepnutí na položku „Děti“.

2.8 Stránky v rodičovském rozhraní

Rodičovské rozhraní je primárním zaměřením projektu, proto je jeho návrhu a tvorbě věnováno největší úsilí. Funkce jednotlivých stránek vychází z případů užití rodičovského rozhraní zmíněných výše v sekci Případy užití.

2.8.1 Úvodní stránka

Tato stránka má za účel slušně přivítat nového uživatele v aplikaci a nabídnout mu možnost projít si krátké seznámení s aplikací. Tato funkce je realizována pomocí dialogu, který se uživatele zeptá, zda by se rád podíval na nápovědu, jejíž popis následuje níže

2.8.2 Nápověda

Cílem této stránky je ukázat uživateli funkce poskytované aplikací a objasnit grafické prvky v ní používané, tedy například barevné odlišení notifikací zmiňované výše. Stránka je v prototypu koncipovaná pouze jako stručný souhrn

dostupných funkcí a je převážně zaměřena na nové uživatele. Plánuje se, že v budoucích verzích bude nápověda mít více různých stránek a sekcí, navíc bude doplněna o interaktivní průvodce a videa podobně, jako bylo možné vidět u konkurenčního software.

Z hlediska UI se jedná o typickou stránku s textem, jediným rozdílem je přítomnost ikon užívaných v menu – ty jsou uvedeny pro zpevnění vazby grafické reprezentace na poskytované funkce – a příkladů „vzorových“ notifikací, které objasňují různým způsobem zdůrazněné notifikace, jejichž výčet byl uveden výše v sekci o UI notifikací (jde o vysvětlení odlišení notifikací pomocí ohraničení a/nebo barevných vykřičníků).

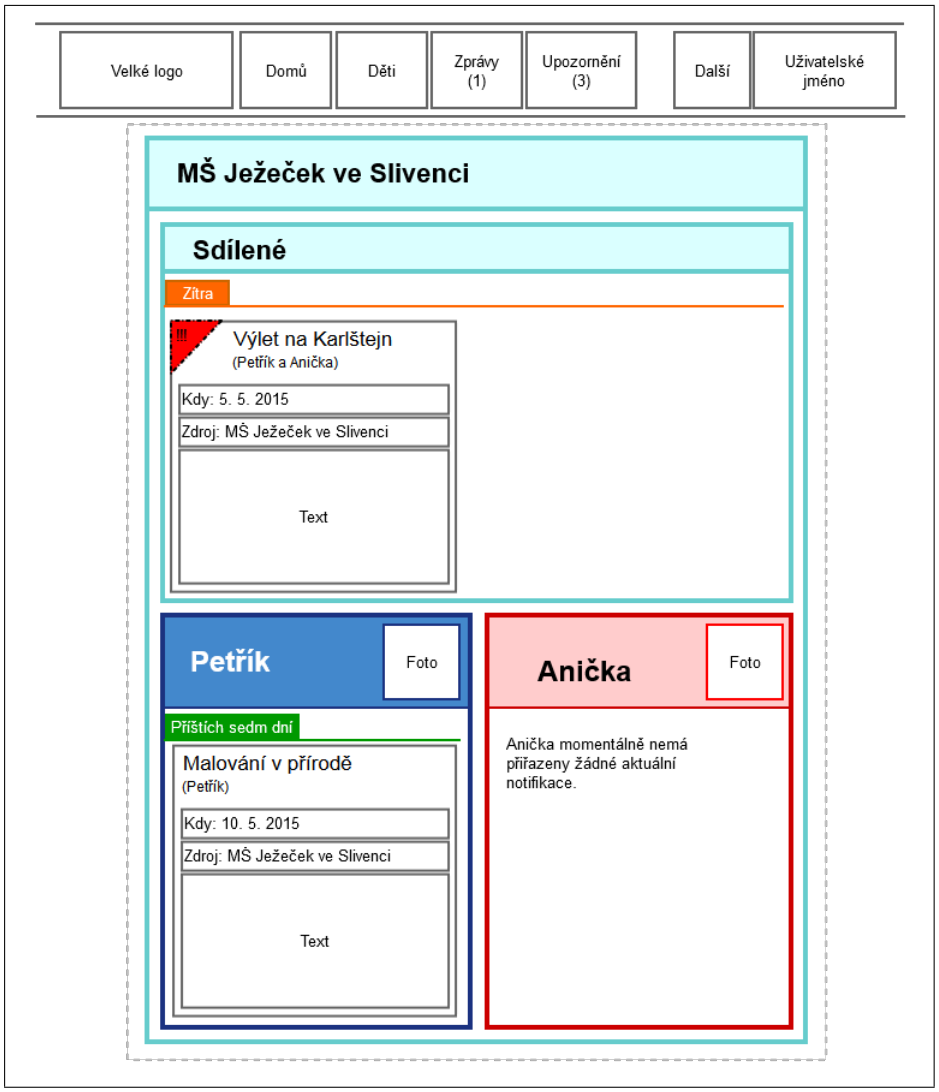
2.8.3 Hlavní strana

Označení této stránky je poměrně příznačné, jelikož by se skutečně mělo jednat o centrální a nejdůležitější obrazovku v systému. Jejím cílem je uživateli prezentovat na jednom místě všechny důležité informace a umožnit mu velmi rychle zjistit vše, co by potřeboval vědět, včetně stavu notifikací a jejich stručného obsahu. Očekává se, že dlouhodobí uživatelé aplikace stráví většinu času právě na této stránce, kde pouze zkontrolují aktuální notifikace a aplikaci opustí.

Funkce poskytované stránkou vychází přímo z usecase „Rychlé zjištění aktuálních úkolů a poznámek týkajících se dětí daného uživatele“, zároveň se jedná o primární přístupový bod k detailům notifikací a dalším funkcím systému.

Všechny výše uvedené komponenty týkající se notifikací a panelů entit cílí primárně na tuto stránku a právě na ní budou plnit svou hlavní funkci, protože hlavní strana je složena z panelů entit obsahujících seznamy notifikací.

Při testování předchozího prototypu měli uživatelé na hlavní straně problémy například s pořadím notifikací a jejich řazením podle data, ale právě na tyto problémy se již zaměřily UI komponenty, které zobrazují notifikace, jejichž rozborem jsme se zabývali výše.



Obrázek 2.10: Mockup hlavní strany rodičovského rozhraní[16].

Mockup na obrázku zachycuje rozložení prvků na hlavní straně a barevné odlišení dětí a školky.

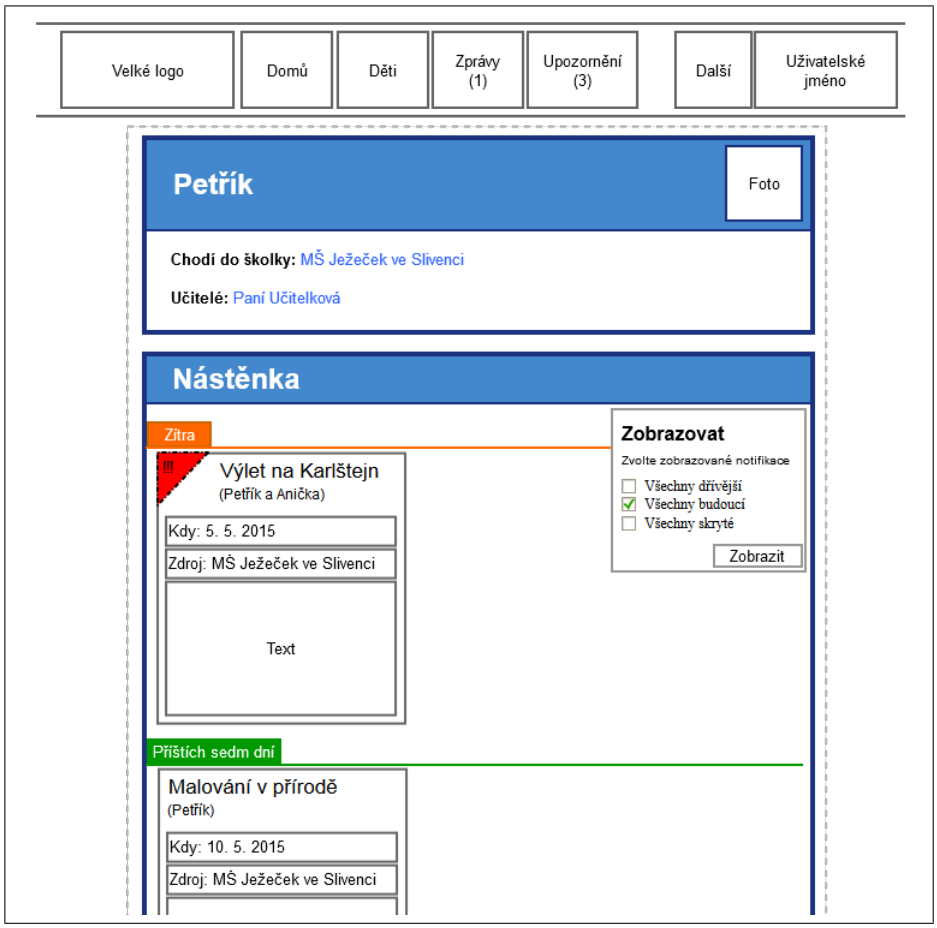
Je možné si povšimnout, že notifikace, které jsou společné pro více dětí v dané školce, jsou uvedeny v sekci „Sdílené“, aby se zamezilo zbytečnému opakování a zároveň dosáhlo větší přehlednosti. To, o jaké děti se konkrétně jedná, je zachyceno v samotném přehledu obsahu notifikace, v tomto případě v podobě textu „(Petřík a Anička)“.

[16]Šrafované pozadí pro odlišení školky na obrázku není znázorněno.

2.8.4 Stránka dítěte

Cílem této stránky je prezentovat veškeré informace o daném dítěti, což zahrunuje jednak jeho notifikace (umístěné na „Nástěnce“) a jednak odkazy na stránky školky a učitelů.

Tato se skládá z dvou panelů entity, které dále rozšiřuje o některé ovládací prvky a další informace. Do panelu „Nástěnka“ je umístěna komponenta „Nastavení zobrazovaných notifikací“, která umožňuje zobrazit skryté notifikace.



Obrázek 2.11: Mockup stránky dítěte.

Na obrázku je vidět, že odkazy na stránku školky a stránku učitele jsou zvýrazněny modrou barvou, stejně tak je vidět rozbalený panel „Zobrazit více“, který byl rozebrán výše v sekci komponent.

Je na místě zamyslet se nad problémem, jak získat fotky dětí, tedy zda je má nahrávat školka, nebo rodiče, a jaké zabezpečení by muselo být přítomno z hlediska ochrany osobních údajů, ale jedná to velice užitečný prvek jednak

pro nové rodiče-uživatele, aby rychle věděli, která sekce se týká kterého jejich dítěte, ale především pro učitele, protože snáz poznají své žáky v učitelské aplikaci (které se věnuje sekce níže). Tím se omezí počet nechtěných voleb při vkládání notifikací, nebo psaní zpráv (při dohledávání uživatele dle dětí), a pokud bude učitel suplovat třídu, snáz děti rozliší.

2.8.5 Stránka školky

Tato stránka je velice podobná stránce dítěte, ale neobsahuje nástěnku a obsahuje navíc kontaktní informace školky. Jinak se jedná o totožné rozložení obsahující místo odkazů na školku a učitele odkazy na děti, které do školky docházejí.

2.8.6 Stránka učitele

V rodičovském rozhraní je možné podívat se na profil učitele, kde může uživatel nalézt kontaktní informace, například telefonní číslo a e-mailovou adresu, pokud by potřeboval učitele kontaktovat.

Jedná se o stránku s totožným rozložením, jakou má stránka školky, které je funkcí velmi podobná. Navíc jsou stránce uvedeny jména dětí uživatele, které daný učitel vyučuje.

2.8.7 Upozornění

Tato stránka slouží pro shrnutí důležitých notifikací, které vyžadují pozornost uživatele, na jedno místo. Takové notifikace jsou jednak notifikace, které vyžadují potvrzení o přečtení, nebo notifikace, které byly změněny a uživatel by si je měl znovu prohlédnout.

Vzhledem k tomu, že se při předchozím testování již původní verze této stránky osvědčila jako bezproblémová, nebylo třeba provádět žádných změn.

Jedinou změnou bylo přejmenování odkazu z „Důležité“ na „Upozornění“, jelikož uživatelé při testování přisuzovali této stránce význam, který má mít spíše hlavní strana.

2.8.8 Kontakty

Tato stránka shrnuje veškeré informace o školkách a o učitelích, které by uživatel mohl chtít kontaktovat. Jedná se o novou stránku, která v dřívěji testované verzi prototypu neexistovala, ani nebyla implementována při prvním testování prototypu vytvořeného v rámci této práce[17]. Ale uživatelé stále měli problémy nalézt kontaktní informace v aplikaci i po přidání stránek školky a korektních odkazů na ně (s čímž byl problém v prvotní testovací verzi v rámci předmětu NUR).

[17]Případ užití „Zobrazení kontaktních informací“ je sám o sobě splněn na stránkách jednotlivých entit.

Z těchto důvodů byla dodatečně[18] přidána stránka kontakty, která sice neobsahuje žádné informace, který by v aplikaci nebyly přístupné na jiných místech, ale měla by být bližší konceptům, na které jsou uživatelé zvyklí.

Po stránce návrhu UI se jedná o jednoduchou stránku, kde jsou uvedeny panely entit školky (školek), kam dochází děti uživatele, a v těchto panelech jsou umístěny kontaktní informace školky a panely učitelů, které obsahují jejich kontaktní informace.

2.9 Úvod k učitelskému rozhraní

Učitelské rozhraní představuje druhý pohled na data v aplikaci, jeho účelem je umožnit učitelům vkládat do systému děti, notifikace, třídy a další entity a spravovat vazby mezi nimi. Učitelské rozhraní představuje po stránce návrhu UI méně zajímavou část systému a to nejen z důvodu již dříve zmiňovaného důrazu spíše na rodičovské rozhraní, ale především kvůli jeho repetitivnosti, kterou shrnuje následující subsekce.

Entity a jim příslušící typy stránek

V učitelském rozhraní je primární funkcí správa entit. To znamená, že toto rozhraní realizuje typické CRUD operace (Create, Read, Update, Delete) nad množinou instancí různých entit, k nimž navíc přidává možnost vytvářet vazby mezi nimi[19].

Entity v aplikaci jsou:

- Školky
- Notifikace[20]
- Děti
- Skupiny dětí
- Uživatelé

[18] Jedná se o stránku přidanou dodatečně, ale přesto její návrh spadá do této kapitoly a této sekce.

[19] Před sestavením následující sekce bylo nutné udělat základní návrh aplikační logiky, tedy vykonat jednu malou vývojovou iteraci, aby bylo jasné, jaké všechny funkce musí učitelské rozhraní realizovat

[20] Ačkoli jsou notifikace natolik důležité, že se na ně vážou některá speciální pravidla zachycená v případech uživatelského rozhraní, z hlediska obecného zacházení s nimi se jedná o typickou entitu.

Pro všechny tyto entity byly dle případů užití vytvořeny následující stránky:

- Seznam instancí dané entity
- Přidání nové instance
- Editování instance
- Zobrazení detailu instance

Až na drobné výjimky jsou tyto stránky pro všechny entity prakticky totožné, pouze se liší dle konkrétních parametrů dané entity. Tyto stránky jsou blíže rozebrány v příslušných sekcích níže.

Uživatelské role

Učitelské rozhraní je určeno pro dvě role uživatelů: administrátory a učitele. Rozhraní zobrazované oběma rolím je prakticky totožné, ale učitelé nemají možnost editovat entity a mají omezené možnosti tvorby nových entit (pravomoce uživatelských rolí rozebírá blíže sekce Uživatelské role).

Využití šablony

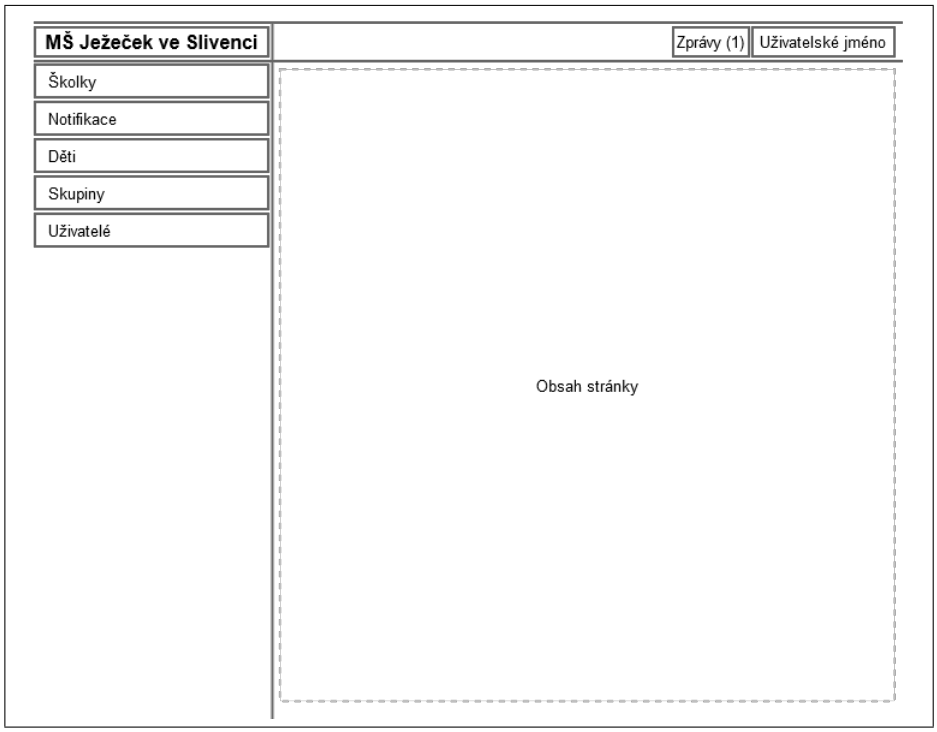
Jak bylo zmíněno v sekci technologie výše v této kapitole, pro usnadnění implementace UI učitelského rozhraní byla vybrána šablona Admin LTE frameworku Bootstrap (dostupná on-line [18]). Vzhled, tedy grafický design, ale i rozložení prvků na obrazovce, především postranního a horního menu, tedy bude vycházet z této šablony.

2.10 Komponenty v učitelské aplikaci

V učitelském rozhraní se opakuje několik komponent, které zasluhují bližší upřesnění:

2.10.1 Menu, navigace a celkové rozložení stránky

Vzhled menu vychází z menu šablony Admin LTE, která je doplněna uživatelským drop-down menu a odkazem do zpráv, jehož součástí je ikona ukazující počet nepřečtených zpráv (obě tyto komponenty byly zmíněny výše).



Obrázek 2.12: Mockup rozložení stránky v učitelském rozhraní.

2.10.2 Tvorba vazby mezi entitami

V učitelském rozhraní není důležitá jen tvorba entit, ale velkou důležitost má i tvorba vazeb mezi nimi – tedy například přiřazování dětí do tříd a kroužků (souhrnně nazývaných skupinami), přiřazení dítěte k jeho rodiči (aby systém příslušnému uživateli zobrazoval notifikace určené pro jeho děti) a další.

Tyto úkony provází procházení potenciálně dlouhých seznamů instancí přiřazované entity (u běžné školky mohou být až stovky dětí) a vybírání příslušných možností, ale zároveň se bude jednat o poměrně typickou operaci, tudíž by měla být pro uživatele co možná nejjednodušší.

Klasický HTML element select proto v tomto případě zvláště kvůli délce seznamů není postačujícím řešením. Pro implementaci bude pravděpodobně vhodné nalézt nástroj, který rozšíří funkcionalitu klasického select elementu o vyhledávání a umožní přehlednější zobrazení vybraných instancí.

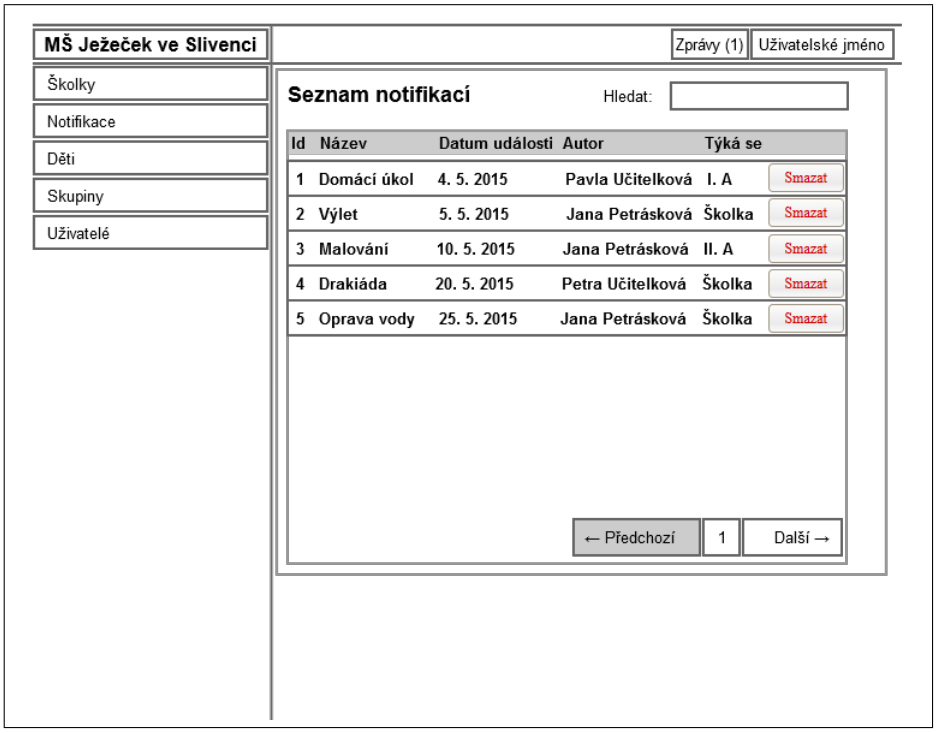
V rámci designu UI budeme počítat s existencí komponenty umožňující takovouto tvorbu vazeb a při implementaci se pokusíme najít vhodný nástroj pro její realizaci.

2.11 Stránky pro práci s instancemi entit

2.11.1 Seznam instancí

Základním rozcestníkem pro práci s již existujícími instancemi je jejich seznam. Jedná se o tabulku, která uživateli poskytuje rychlý přehled existujících instancí a základní funkce pro práci s nimi – odkaz na zobrazení, či editaci instance a tlačítko pro smazání.

Vzhledem k jejich potenciálně velkému počtu je nutné zavést „stránkování“, tedy zobrazovat pouze omezený počet instancí naráz. Zároveň je z podobných důvodů nutné umožnit uživatelům vyhledávat v seznamu entit.



Obrázek 2.13: Mockup seznamu instancí entit v učitelském rozhraní.

Na výše uvedeném mockupu je seznam demonstrován na příkladu notifikací, ale pro jiné entity se bude jednat o velmi podobnou stránku.

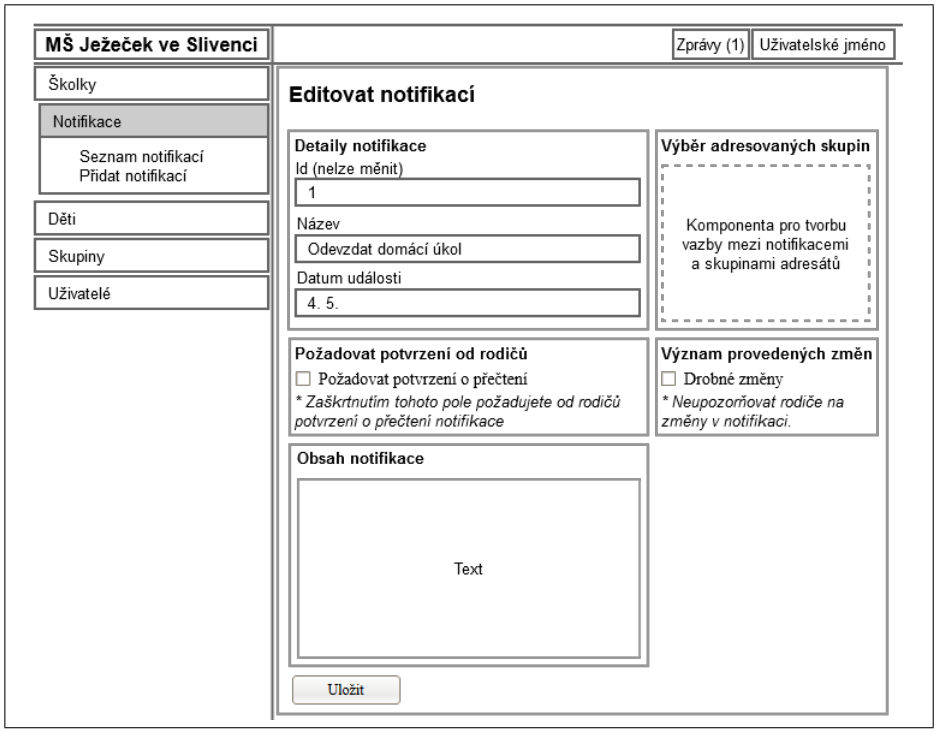
2.11.2 Zobrazit, editovat a přidat instanci

Všechny ostatní stránky jsou si navzájem velmi podobné, pokud ne přímo totožné.

Stránka pro přidání nové instance obsahuje formulář, který reflektuje atributy vytvářené entity, které má uživatel vyplnit. Při editaci této entity však přistupujeme, až na výjimky, k těm samým položkám, tudíž se jedná o stej-

nou stránku, pouze vyplněnou údaji již existující instance. A zobrazení detailu dané instance je opět stejné, pouze uživateli neumožňuje měnit uvedené údaje.

Co se přístupu k jednotlivým stránkám týče, na stránku přidání nové instance odkazuje položka v menu, zatímco stránky pro zobrazení detailů dané entity a její editaci jsou přístupné ze stránky seznamu instancí dané entity, poklepáním na řádek příslušné instance v tabulce. Stránka, která je uživateli zobrazena, závisí na jeho právech, tedy na jeho roli – pokud má právo instanci editovat, je mu zobrazena příslušná stránka, jinak je mu zobrazen pouze detail dané instance, kterou nemůže měnit. O uživatelských rolích a právech je více uvedeno níže v kapitole věnované návrhu.



Obrázek 2.14: Mockup detailu instance v učitelském rozhraní.

Mockup znázorňuje obecnou podobu stránky pro tvorbu, úpravu nebo zobrazení detailů instance entity.

Následující subsekce se věnují specifickým případům, které vyplývají z případů užití.

Tvorba a úprava notifikace

Dle případu užití „Vložení notifikace do systému“ může uživatel zvolit, že se jedná o důležitou notifikaci a vyžadovat její potvrzení.

Dále má dle případu užití „Dodatečná změna notifikace“ uživatel při editaci notifikace možnost zvolit, zda mají být rodiče informování o úpravách notifikace. Pokud nezvolí, že se jedná pouze o drobné změny, u notifikace bude poznamenáno, že byla změněna.

Tvorba uživatelského účtu

V případě tvorby uživatelského účtu je dle příslušného případu užití možno zvolit, že uživatel má být o nově vytvořeném účtu informován e-mailem.

2.12 Závěr

V této kapitole byly stručně zmíněny teoretické základy tvorby uživatelského rozhraní, analyzovány konkurenční produkty po stránce UI a funkcionality, navržena nová podoba uživatelského rozhraní a vybrány vhodné nástroje pro jeho tvorbu.

Tím je zároveň plně rozhodnuto o podobě klientů (rodičovského a učitelského) – bude se jednat o lehkého webového klienta tvořeného pomocí HTML, CSS a Javascriptu, což je v dnešní době velmi typická alternativa.

Následovat bude návrh aplikační logiky, i když, jak bylo zmíněno výše, část tohoto návrhu již musela být hrubě sestavena v rámci tvorby učitelského rozhraní.

Kapitola 3

Návrh aplikační logiky a datového úložiště

V předchozí kapitole bylo navrženo uživatelské rozhraní pro přístup do aplikace Mojeskolka. Tato kapitola, návrh aplikační logiky a datového úložiště, se bude zabývat otázkou, k čemu vlastně je za pomoci UI možné přistupovat.

Nejprve budou učiněna některá rozhodnutí upřesňující podobu systému a poté postupně od návrhu obecného rozdělení aplikace a hrubého modelu entit v aplikaci bude přistoupeno až k výběru technologií (knihoven, či frameworku) pro implementaci, na základě kterých bude určena finální podobu architektury systému a strukturu tříd realizujících aplikační logiku.

3.1 Forma poskytování aplikace

Jednou z otázek, které doposud nebyly zodpovězeny, je způsob poskytování aplikace, tedy zda bude systém prodáván po jednotlivých licencích do školek, které jej budou provozovat na vlastním serveru a budou si muset zajistit jeho správu, nebo zda bude poskytován jako SaaS, ve kterémžto režimu školka pouze platí měsíční paušál za poskytování služeb a správu, zatímco systém běží na serverech provozovaných jeho vlastníky a vývojáři.[21]

Na základě současných trendů v IT, výsledků rešerší, ale i zdravého rozumu, který velí, že pro školky, které typicky zaměstnávající několik málo učitelů, je zbytečnou zátěží provozovat a spravovat vlastní informační systém, je možné poměrně jednoznačně rozhodnout ve prospěch SaaS.

Výhody tohoto přístupu, tedy šetření zátěže na samotné školky (je dobré nezapomínat, že hlavním cílem projektu Mojeskolka je usnadnit situaci rodičům, ale i učitelům v mateřských školkách), nevystavování zdrojového kódu

[21] Jedná se o rozhodnutí na hranici softwarového inženýrství a marketingu, ale vzhledem k jistému podílu autora této práce na podobných rozhodnutích a jeho tematické příslušnosti k této práci můžeme příslušné rozhodnutí skutečně učinit.

zákazníkům, nebo snadná distribuce aktualizací systému, výrazně převažují nad výhodami správy systému samotnými školkami. Další skutečností hovořící ve prospěch SaaS je fakt, že firma, pro kterou je systém Mojeskolka vytvářen, se specializuje na tvorbu datových center a provoz serverů, což eliminuje prakticky veškeré nevýhody zmíněné formy poskytování aplikace.

3.2 Rozdělení na rodičovské a učitelské rozhraní

Rozdělení na jednotlivá rozhraní dává smysl z uživatelského hlediska, tedy z hlediska návrhu UI, i z hlediska vnějšího pohledu na systém, ale z hlediska návrhu jeho interních částí již smyslu pozbývá. Je totiž důležité si povšimnout, že rodiče vidí ty samé notifikace, které učitelé vkládají do systému, jejich děti jsou ty samé, které učitelé přiřazují do skupin podle tříd, kroužků a dalších logických celků, ale také toho, že jeden uživatel může být v jedné školce učitelem, zatímco do druhé chodí jeho děti, od kterých by měl dostávat notifikace. Ale fyzicky se jedná o toho samého uživatele, je zbytečné, aby byl jinak reprezentován v rodičovské a jinak v učitelské části systému. A toto tvrzení je možné vztáhnout na všechny entity.

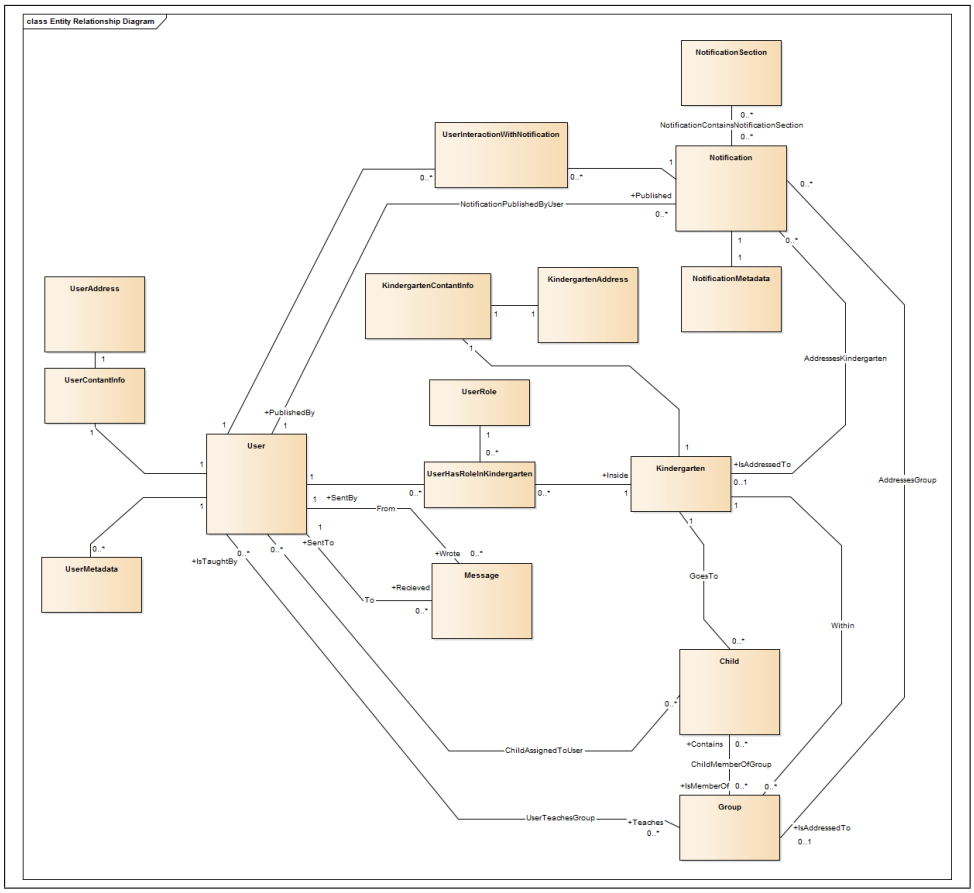
Proto je důležité si uvědomit, že rodičovské a učitelské rozhraní jsou pouze dva pohledy na tatáž data, tedy že obě části aplikace sdílí ten samý entitní model a persistentní data. Proto již dále nebude nutné rozlišovat mezi učitelským a rodičovským rozhraním – pouze pro každé z nich budou vytvořeny jiné pohledy (views) a používány jiné přístupové body (které budou obsluhovat takzvané controllery). O těchto složkách systému je více uvedeno v sekci Výsledná podoba systému.

Jinak však bude vytvořen jednotný návrh entit a datového úložiště, shodný pro obě rozhraní.

3.3 Hrubý entitní model

V rámci iterativního přístupu k tvorbě aplikace a poznatkům uvedeným v Motivace pro průběžnou komunikaci, tedy že růstu ceny za opravu chyby v průběhu projektu roste exponenciálně, nejprve bude sestaven hrubý model vystihující entity figurující v systému a jejich vztahy, a až na základě něj a další rozvahy bude vytvořen více detailní model s atributy a konkrétní podobou vazeb.

Cílem tohoto postupu je nejprve pochopit doménu systému jako celek, korektně vystihnout vazby v ní a až poté přejít k detailům, aby výsledný model dosahoval co nejvyšších kvalit. Doménový model totiž tvoří základ celé aplikace a pokud do něj bude zanesena chyba, pak při následné snaze stavět nad ním aplikační logiku lze očekávat velké množství problémů a komplikací. Zároveň tímto způsobem je možné lépe pochopit systém z jednodušších a přehlednějších schémat.



Obrázek 3.1: Hrubý entitní model.

Jak je možné z obrázku vidět, entitní model poskytuje pohled na doménu systému, tedy entity a jejich vztahy. Při jeho tvorbě bylo vycházeno z případů užití, ale velkou měrou i z návrhu uživatelského rozhraní, který byl vytvořen v předchozí kapitole (přesněji řečeno bylo čerpáno spíše z procesu jeho tvorby, než ze samotného výsledku). Jak bylo zmíněno výše v sekci Entity a jim příslušící typy stránek, návrh UI a návrh aplikační logiky se částečně prolínaly, jelikož pro správný návrh stránek v učitelském rozhraní bylo potřeba mít alespoň základní představu o entitách v systému. Zároveň detailní návrh UI pomohl lépe pochopit požadavky na systém a lépe sestavit tento model, což je pozitivním důsledkem snahy postupovat po malých krocích a krátkých iteracích, ve kterých se střídají různé typy činností vedoucí k tvorbě finální podoby systému.

3.3.1 Poznámky k modelu

V samotném modelu jsou vazby a entity, nad kterými by mohlo být vhodné se blíže pozastavit. Jejich bližšímu rozboru je věnována tato sekce.

- **UserInteractionWithNotification:** jedná se o entitu, která zachycuje speciální vlastnosti této vazby – popisuje totiž akce rodiče vůči dané notifikaci. Za zmínku stojí, že rozhodnutí, zda se danému rodiči tato notifikace zobrazuje, je na této vazbě zcela nezávislé. To je učiněno na základě jeho dětí a jejich vazeb na skupinu, nebo školku, které je notifikace určena (tomuto polymorfnímu vztahu bude věnován odstavec níže). Ale z případů užití vychází, že rodič může danou notifikaci potvrdit, nebo si ji nechat skrýt. A právě tyto interakce je nutné zaznamenávat.

- **Polymorfní vztah notifikace ke školce, nebo ke skupině:** z navrženého modelu je patrné, že notifikace může mít vazbu jak na školku, tak na skupinu. Bylo by vhodné upřesnit, že dle současných plánů se počítá s vazbou notifikace pouze na jednu jinou instanci, tedy na skupinu, nebo na školku, ale nikoli na obě současně a pouze na jednu instanci vybrané entity, jak znázorňuje arita uvedených vazeb.

  Tento polymorfní vztah bude lépe viditelný na detailním entitním modelu a proto bude blíže rozveden ještě v sekci tomuto modelu věnované.

- **Rozdílné kontaktní údaje školek a uživatelů:** kontaktní údaje pro uživatele a pro školku se kvůli absenci atributů mohou zdát na tomto diagramu totožné, ale jsou rozděleny na dvě oddělené entity jednak kvůli korektnímu rozdělení na moduly, kterému je věnována následující sekce, ale také kvůli odlišným položkám, například IČ u školky.

- **Děti ve více školkách:** je dobré povšimnout si, že jednomu uživateli mohou být přiřazeny děti, které chodí do více různých školek, což nejspíše nebude zcela typická situace, ale v reálném světě může nastat, pročež by ji měl umět zachytit i navrhovaný model.

  Možnost docházky jednoho dítě do více školek je však složitější. Mělo by se jednat o velmi atypickou situaci, ale teoreticky může být možné, aby dítě například chodilo na kroužky ve více různých školkách. Tuto situaci však model neumí popsat, jelikož dítě může chodit pouze do jedné školky, jak je dáno aritou této vazby. Důvodem pro toto omezení je možnost oddělené jednotlivých školek do vlastních oddělených webových služeb, které se budeme věnovat v sekci Odolnost systému vůči zátěži.

  Bude třeba ověřit v reálném nasazení, zda tato situace nastává a jakým způsobem by bylo vhodné ji řešit, pokud bude nastávat často. V úvahu připadá duplikování dítěte mezi databázemi školek, které by do budoucna mohly být oddělené, nebo separace kroužků do samostatného modulu, který by podobné situace řešil.

- **Uživatelské role:** Z modelu je vidět, že umožňuje uživateli nabývat různých rolí v různých školkách zcela nezávisle na sobě. To poskytuje plnou flexibilitu, která by mohla být využita například v okamžiku, kdy

nějaký rodič současně v dané školce dělá administrátora, ačkoli v ní nevyučuje, nebo kdy má učitel děti v jiné, nebo té samé školce. Rolím v systému je věnována samostatná sekce Uživatelské role.

- **Metadata:** Tyto entity doplňují záznamy o uživateli nebo notifikaci a slouží k ukládání velmi řídkých informací o instanci dané entity, tedy dodatečných detailů, které pravděpodobně budou přiřazeny jen malé skupině instancí příslušné entity a proto není výhodné pro tyto informace vyhrazovat samostatný sloupec v tabulce, který by u valné většiny instancí nabýval defaultní hodnoty.
- **Sekce notifikací:** Podobným způsobem slouží i sekce notifikací (tedy NotificationSection), které umožňují do systému vkládat notifikace, které mají různou strukturu – jedna notifikace může obsahovat pouze text, zatímco jiná může obsahovat textové pole, anketu a připojené soubory. Tyto dodatečné sekce (různých typů, jak zachycuje příslušný sloupec) budou přiřazeny nějaké notifikaci, jako její volitelná rozšíření (pro realizaci anket, kde rodiče budou moci hlasovat o nějakém rozhodnutí, bude pravděpodobně nutné rozšířit současný model o další položky, ale bude se jednat o stejný princip fungování). Potvrzení o notifikace je teoreticky taktéž sekcí notifikace, ale jedná se o tolik typickou a důležitou funkci v systému, že je pro ni vyhrazeno pole přímo v notifikaci.

3.3.2 Závěr

Byl navržen a rozebrán hrubý entitní model, který slouží k vytvoření základní představy o fungování systému. Na základě tohoto modelu bude následně vytvořen detailní model, který poslouží jednak pro implementaci databáze, jednak jako jádro tříd v prototypu.

Tento model stále obsahuje velké množství entit a pro člověka není zcela snadné se v něm zorientovat. To souvisí s takzvaným Millerovým pravidlem, taktéž známým jako pravidlo sedm, plus mínus dvě, které říká, že lidský mozek je schopný současně vnímat vazby mezi pěti až devíti objekty(podle jeho mentálních schopností a složitosti vazeb mezi entitami), což implikuje, že vytvořený model, který se skládá z šestnácti entit, je stále příliš složitý pro snadné pochopení.

Je však možné si všimnout, že každá daná entita má vazbu pouze na menší množství jiných, díky čemuž je možné systém vnímat po jednotlivých oblastech, v rámci kterých je toto pravidlo již respektováno. Existence těchto oblastí poukazuje na modularitu systému, které se věnuje následující sekce.

3.4 Rozdělení na moduly

Zkoumáním entitního modelu je možné povšimnout si logických celků, které představují skupiny entit, které společně plní nějakou funkci a typicky mají

vazby převážně mezi sebou, zatímco jejich vazby na zbytek modelu jsou omezené, povětšinou přes jednu konkrétní entitu (například uživatel a jemu příslušící entity, ta samá situace nastává i u školek, dětí a notifikací).

Tato představa umožňuje identifikovat logické moduly, které jednak pomohou lépe uchopit doménu systému, ale co je potenciálně velmi důležité do budoucna, představují funkční celky, které by mohly být časem rozděleny do samostatných webových služeb v rámci škálování systému (této problematice je věnována sekce Odolnost systému vůči zátěži níže v této kapitole).

Omezenost vzájemných vazeb mezi těmito celky a jednotlivé entity do nich spadající budou lépe patrné z Detailní entitní model uvedeného v následující sekci. V následujících subsekcích pouze uvedeme jednotlivé logické moduly a stručně je popíšeme.

3.4.1 Model modulů

Samotný model modulů by nebyl tolik vypovídající, jako jeho znázornění v rámci detailního modelu, uvedeného v následující sekci. Toto znázornění totiž zachycuje konkrétní entity v modulech a vazby mezi těmito entitami, díky čemuž je možné lépe pochopit vazby mezi moduly a zároveň je viditelné, že tyto vazby nejsou tolik početné, což odpovídá zvolenému rozdělení do těchto celků.

3.4.2 Seznam modulů

Uživatelé

Centrální součást systému, která má vazby na všechny ostatní moduly. Obsahuje uživatele a jemu příslušící třídy, jako kontaktní informace a metadata.

Na systém je možné pohlížet tak, že této centrální součásti, v jejímž středu je uživatel, poskytují ostatní moduly data.

Zprávy

Oddělení zpráv do samostatného modulu se může zdát sporným, jelikož se jedná pouze o jednu entitu, která má silnou vazbu na uživatele. Avšak v případě, že by bylo žádoucí do budoucna moduly oddělovat do samostatných služeb kvůli rozložení výkonu, dá se očekávat, že zprávy mezi uživateli budou poměrně vytíženou součástí systému, což z ní činí vhodného kandidáta na oddělení do samostatného celku.

Školky a uživatelské role

Vzhledem k tomu, že role uživatele je vázána na konkrétní školku a je nezávislá na ostatních školkách, obsahuje tento modul entitu UserHasRoleInKindergarten určující přiřazení role k danému uživateli a školce.

Otázkou zůstává, v jakém modulu by měly být umístěny samotné uživatelské role, tedy který modul by měl spravovat množinu různých rolí. Jako nejvhodnější se jeví možnost umístit role do modulu školek, čímž by se v případě rozdělování systému umožnilo školkám definovat vlastní uživatelské role v případě potřeby. Tento mechanismus by vyžadoval přidání entity zachycující oprávnění, přiřazená jednotlivým rolím, avšak prozatím toto umístění poslouží minimálně ke zdůraznění, že se jedná o role v rámci školky, nikoli obecné role, která by příslušela uživateli.

Tento modul samozřejmě obsahuje i samotné školky a jejich kontaktní údaje.

Děti a jejich skupiny

Jak název tohoto modulu napovídá, obstarává děti a jejich přiřazení do skupin. Sice má poměrně silnou vazbu na školky a uživatele, kteří jsou rodiči dětí a učiteli skupin, přesto umístění těchto entit do vlastního modulu nejlépe odpovídá jejich vzájemné provázanosti.

Notifikace

Samostatným logickým celkem jsou i notifikace, jejich sekce a metadata. Do tohoto modulu spadá i interakce uživatelů s notifikacemi, jelikož tyto interakce jsou důležité pouze při přístupu k jednotlivým notifikacím, nikoli při zjišťování informací o uživateli, čemuž by odpovídalo umístění v modulu uživatelů.

3.5 Detailní entitní model

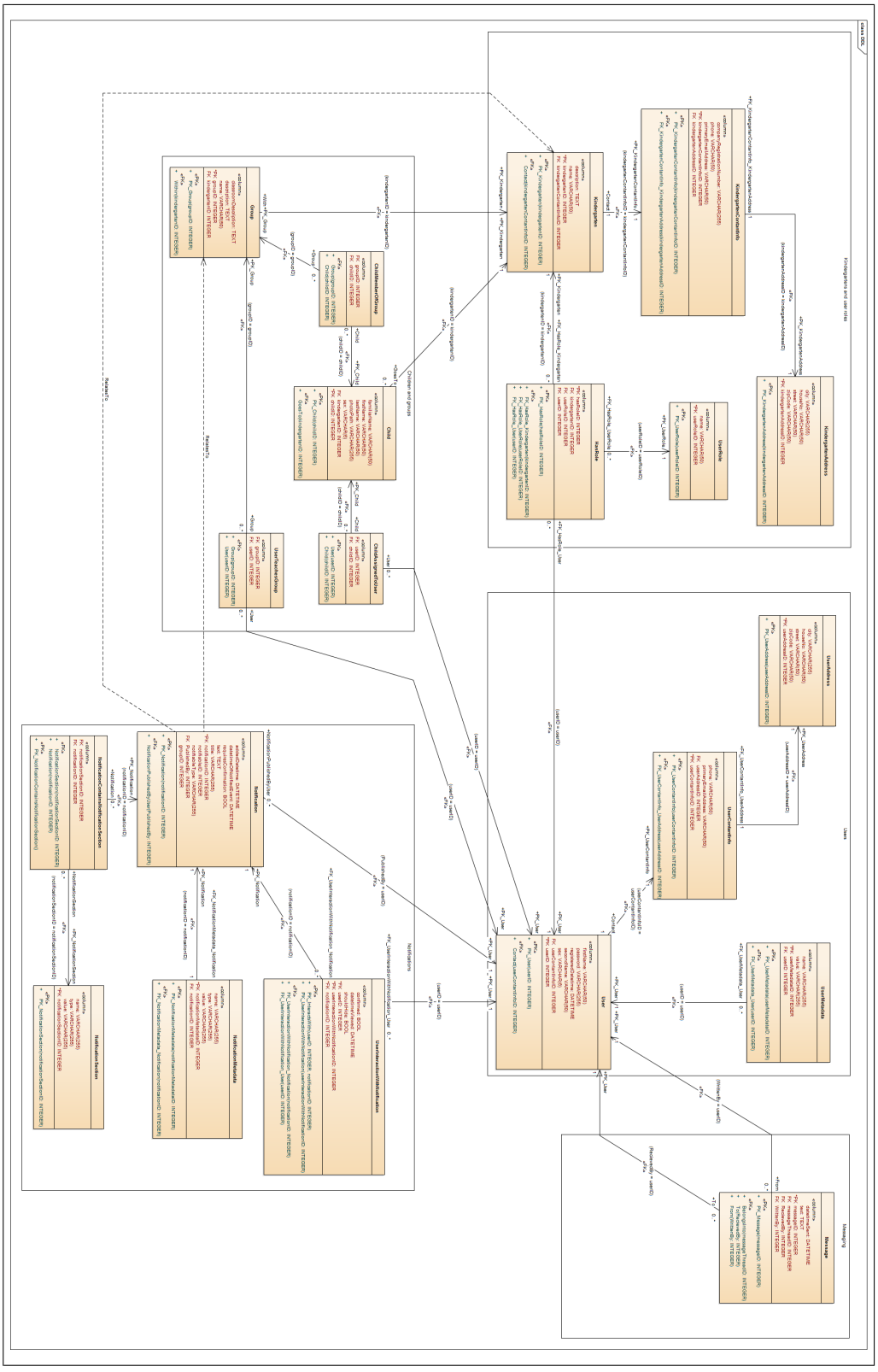
Po důkladnějším rozboru hrubého modelu a analýze logických celků, které se v něm vyskytují, nyní můžeme přistoupit k tvorbě detailního modelu entit, který bude zachycovat návrh relační databáze (jejíž uspořádání odpovídá logickým entitám v systému) a zároveň znázorní rozdělení na moduly, zmíněné výše.

Tento model je rozšířením výše uvedeného hrubého modelu o atributy entit a vazby mezi nimi. Zároveň ukazuje entity, které přísluší jednotlivým modulům, a poměrně nízkou provázanost mezi těmito moduly.

3.5.1 Poznámky k modelu

Většina zajímavých návrhových rozhodnutí, která zachycuje tento model, jako například umístění uživatelských rolí do modulu školek, nebo význam interakcí a metadat, byla již vysvětlena výše.

Jedinou zajímavostí je řešení vztahu mezi notifikací a školkou, nebo skupinou dětí, které bylo vyřešeno za pomoci polymorfní vazby. Tato vazba byla navržena podle způsobu, jakým tento problém řeší moderní webové frameworky,



Obrázek 3.2: Detailní návrh entit v systému, zachycující strukturu relační databáze.

tedy ukládáním identifikátoru typu entity, ke kterému notifikace náleží, tedy řetězce notifiableType, a unikátního identifikátoru (notifiableID) instance této entity. Tento přístup odpovídá spíše objektovému, nežli relačnímu paradigmatu, ale nemělo by se jednat o nežádoucí vlastnost, vzhledem k tomu, že zvažované frameworky, z nichž je v příslušné sekci níže vybíráno, povětšinou využívají ORM, které relační a objektový svět přibližuje.

Alternativou k tomuto přístupu by bylo vytvořit pro každou školku vlastní skupinu se shodným názvem, ale toto řešení neposkytuje takovou flexibilitu a zanáší nežádoucí duplicity.

3.6 Uživatelské role

Jak bylo uvedeno již v případech užití, v systému Mojeskolka budou figurovat uživatelé ve třech různých rolích (a do budoucna možná ve více, jak bude zmíněno níže). Z modelu entit vyplývá, že jeden uživatel může mít ve více různých školkách, dokonce i v jedné a té samé školce, více různých rolí, což je v souladu s reálným světem, i když tento případ nemusí být tolik typický.

Otázkou zůstává, jaké role a především, jaké pravomoci uživatelé v systému mohou mít. Tímto rozborem se budou zabývat následující subsekce.

Rodič

Uživatel v roli rodiče by měl mít přístup ke všem informacím, které mu náleží, ale k žádným dalším.

Uživatel v roli rodiče má oprávnění:

- Prohlížet notifikace přiřazené jeho dětem.
- Zobrazit si kontaktní informace školek, kam chodí jeho děti, a učitelů k nim náležících.
- Dívat se na stránky svých dětí.
- Psát zprávy učitelům a správcům (administrátorům) ze školky, kam chodí jeho děti.

Z hlediska aplikační logiky se jedná o jednodušší případ, jelikož každý rodič má přístup do celého rodičovského rozhraní, i když je omezen pouze na informace, které se týkají jeho dětí.

Učitel

V případě učitelů a administrátorů je však situace odlišná. Uživatelé v obou těchto rolích se přihlašují do stejného rozhraní, avšak mají o něco odlišné pravomoci.

Je otázkou, jaké oprávnění by přesně měli mít učitelé v porovnání s administrátory, a toto nastavení se pravděpodobně do ostrého nasazení systému ještě změní na základě konzultací se školkami a výsledků beta testování (o kterém pojednává příslušná sekce níže v kapitole testování). V rámci této práce budou učitelům přiřazeny minimální pravomoci, které potřebují pro vykonávání jim příslušících úkonů, kteréžto pravomoci mohou být v budoucnu rozšířeny.

Uživatel v roli učitele má oprávnění:

- Zobrazovat detaily všech entit (uživatelů, dětí, skupin, školek a notifikací) v aplikaci.
- Přidávat do systému notifikace.
- Editovat, nebo mazat notifikace, které se týkají skupin, které učiteli náleží.
- Psát zprávy rodičům, učitelům a administrátorům.

Z tohoto seznamu vyplývá, že učitel nemůže měnit, nebo do systému přidávat děti, skupiny dětí, uživatele, ani školky. Také nemůže editovat notifikace, které nenáleží skupinám, které má v systému přiřazené.

Administrátor

Administrátor, nebo také správce, má za úkol spravovat systém na nejvyšší úrovni v rámci dané školky, proto má všechna oprávnění k tomu potřebná.

Uživatel v roli administrátora má oprávnění:

- Zobrazit si detaily všech entit v systému.
- Měnit detaily instancí všech entit v systému.
- Mazat instance všech entit v systému.
- Psát zprávy jakémukoli uživateli.

Je dobré si uvědomit, že v budoucnu bude muset být oddělena role takzvaného site-administrátora a system-administrátora. Zásadní rozdíl je v rozsahu pravomocí, tedy že site-administrátor má maximální pravomoci v rámci dané školky, zatímco system-administrátor má plné pravomoci v rámci celého systému, tedy například i co se týče mazání, editace a přidávání školek. V rámci prototypu existují pouze uživatelé s pravomocemi budoucích system-administrátorů, ale tyto role bude nutné oddělit.

3.7 Výběr technologií pro realizaci

Dříve, než bude možné přistoupit k finálnímu návrhu aplikační logiky a architektury serverové části systému, bylo by vhodné učinit poslední rozhodnutí, která tyto úkony mohou ovlivnit. Konkrétně se jedná o rozhodnutí, jaké technologie budou použity při implementaci.

Jak bylo zmíněno výše v sekci Volba programovacího jazyka, v rámci iterativního přístupu k analýze a návrhu bylo v předcházejících kapitolách a sekcích shromážděno co možná největší množství indícií ovlivňujících toto rozhodnutí. Bylo již rozhodnuto, jakou podobu bude systém mít, jaké požadavky by měl splňovat, byl sestaven detailní model entit v něm figurujících a zbývá výběr konkrétních nástrojů pro jeho implementaci.

Tím nastává poslední fáze iterativního přístupu k volbě použitých technologií. Následovat ji bude už pouze výběr knihoven a dalších komponent nutných pro realizaci částí programu přímo při implementaci.

3.7.1 Typ použité technologie

Na základě výsledků rešerší, diskusí se zkušenějšími vývojáři a zkoumání online zdrojů bylo rozhodnuto, že pro implementaci systému Mojeskolka bude použit již hotový framework, namísto tvorby vlastního, nebo jeho skládání z již existujících knihoven.

Tato sekce se bude zabývat výběrem vhodného softwarového frameworku a v dalších kapitolách bude tento nástroj brán v potaz a bude na tuto volbu odkazováno, jelikož výrazně ovlivní architekturu systému a fungování jeho aplikační logiky.

3.7.2 Úvod k výběru frameworků

Výběr frameworků navazuje na sekci Volba programovacího jazyka. I se při tomto výběru opakují velmi podobné problémy, jaké bylo nutné řešit při výběru vhodného jazyka, s tím rozdílem, že podložených informací o vhodnosti jednotlivých voleb ještě ubylo.

Proto budeme muset opět hodnotit na základě důvěry v informace agregované z velkého množství on-line zdrojů a na základě vlastních kritérií.

Jak bylo zmíněno výše, v rámci celkového rozhodování o vhodném jazyce pro implementaci byly již předběžně zkoumány, analyzovány a srovánávány různé frameworky v různých jazycích, například na základě [1], [21] a mnoha dalších, jejichž výsledky ovlivnily i výběr programovacího jazyka.

Zásadním problémem, se kterým bude při volbě frameworku potřeba vhodně pracovat, je rozličnost názorů a navzájem si protiřečící informace z různých on-line zdrojů. A zásadním faktorem umocňujícím tento problém je, že neexistuje žádný lepší zdroj informací, než on-line články, případně doplněné o diskuse se zkušenějšími vývojáři, kteří však také mají vlastní názor založe-

ných na svých subjektivních zkušenostech. Na toto téma například neexistuje vhodná literatura, jelikož dříve, než by případný autor stihl knihu srovnávající webové frameworky sepsat a vydat, informace v ní uvedené už by nebyly aktuální.

Pro skutečně kvalitní rozhodnutí by bylo nutné za pomoci všech zvažovaných frameworků vytvořit webový informační systém, několik let jej spravovat a rozvíjet, a poté by bylo možné rozhodnout o tom, který z nich je nejlepší. Lépe řečeno byl – v době, kdy byly tyto informační systémy vytvořeny. Protože za dobu podobného testu se situace na poli webových frameworků opět stihne výrazně změnit.

Proto nezbývá, než založit výběr na dostatečném množství on-line článků, definovat důležitá kritéria, na základě kterých budou jednotlivé alternativy srovnávány, a pokusit se nalézt nejvhodnější volbu.

Tato volba pravděpodobně bude muset být učiněna i na základě subjektivních kritérií a kvalifikovaného odhadu, jelikož je dobré mít na mysli, že většina současných frameworků si je navzájem velmi podobná (jak poukazuje například [22]) a výběr konkrétního závisí na kritériích, která budou nyní vyhodnocena jako důležitá, ale není možné zaručit, že budou zcela objektivní.

3.7.3 Kritéria pro výběr

Není tedy snadné definovat, na základě jakých vlastností jednotlivé alternativy porovnávat, jak se snaží vysvětlit předchozí sekce. I přesto však je možné definovat některé parametry, které bude vhodné sledovat, a takové, které naopak sledovat nutné nebude. Na základě těchto kritérií a dalších vlastností zkoumaných frameworků bude učiněno rozhodnutí o nástroji použitém pro implementaci.

Významné

- **Rozšířenost a komunita:** Velmi důležitým aspektem při výběru podobného nástroje je možnost získat návody, rady a případně pomoc ohledně typických problémů, které při práci s daným nástrojem musí vývojáři řešit. Kvalitní on-line komunita šetří čas nutný k ovládnutí nástroje a výrazně redukuje úsilí věnované pročítání manuálů, nebo odstraňování nepříjemných chyb, které může v případě špatně zdokumentovaného problému s málo rozšířeným nástrojem vést na rozsáhlé prostoje ve vývoji produktu.

- **Náročnost ovládnutí nástroje:** Anglicky označovaná jako „learning curve“, tato vlastnost zachycuje schopnost vývojáře, který s daným nástrojem pracuje poprvé, efektivně jej využívat. Vzhledem k relativně omezenému času na tvorbu prototypu a obecné nepříjemnosti práce se zbytečně složitým nástrojem je tento parametr poměrně důležitý.

- **Moderní technologie:** Svět informačních technologií se neustále vyvíjí a je možné sledovat vývoj od starých, složitých nástrojů a paradigmat, z nichž většina je časem překonána a zaniká, k modernějším, flexibilnějším alternativám. Jedná se o poměrně sporné kritérium, ale je možné obecně říci, že mnoho starších nástrojů a frameworků je používáno pouze proto, že existují systémy, které na nich byly vystavěny a tyto systémy jsou natolik úspěšné, že udržují starší technologie ?při životě?, ačkoli jsou již současnou konkurencí překonány. A takovýchto nástrojů bude vhodné se vyvarovat.

  Samozřejmě je potřeba neriskovat příliš a nevolit neověřenou alternativu, která se může ukázat jako neúspěšná a v brzké době zaniknout. Proto budou preferovány frameworky, které již mají stabilní postavení mezi konkurencí, ale zohledňují moderní paradigmata a přístupy, například databázové migrace, ORM, nebo architektonický vzor MVC (který však je základem většiny zvažovaných frameworků).

- **Další parametry:** Ohled bude brán i na bohatost a rozsah poskytovaných funkcí, ale i na obecné zaměření frameworku, jelikož některé jsou určeny spíše pro tvorbu výkonných webových stránek, zatímco jiné cílí skutečně na tvorbu webových informačních systémů, jakým je Mojeskolka. Důvodem, proč tyto důležité parametry nemají větší váhu, je jejich obtížná posouditelnost na základě dostupných zdrojů.

Nevýznamné

- **Cena:** Nelze tvrdit, že by na pořizovací ceně frameworku nezáleželo, ale všechny zvažované zvažované jsou dostupné zdarma jako open source projekty, pročež nemá význam tuto vlastnost u každé z nich uvádět.

3.7.4 Srovnání frameworků

Po omezení možných alternativ a vyloučení nevhodných možností, zmíněných níže, připadají v úvahu následující softwarová řešení: Laravel [23], Phalcon [24], Symfony2 [25] a Yii [26].

Následuje výčet jednotlivých alternativ s jejich pozitivy a negativy. Uvádět bodové srovnání, jaké bylo sestaveno v sekci Volba programovacího jazyka, v tomto případě postrádá smyslu, jelikož váha jednotlivých kladů a záporů je velmi rozličná a jak již bylo zmíněno, výběr bude muset být učiněn i na základě subjektivního rozhodnutí.

Laravel

Klady

- Zaměření na vývojáře a kvalitu kódu – expresivní syntaxe, princip inversion of control a další mechanismy pro usnadnění vývoje.
- Convention over configuration, tedy celkový přístup k tvorbě produktu a užívání nástroje jako takového za pomoci dodržování definovaných pravidel, raději než nastavování složitých konfiguračních souborů. Autor se s ním měl možnost setkat v Ruby on Rails a tento princip ho velmi ho oslovil.
- Snadný na ovládnutí a efektivní používání.
- Velmi silná on-line komunita, dle [21] se jedná v současnosti o nejpopulárnější framework.
- Moderní paradigmata a konvence.
- Spojuje knihovny a nástroje z různých frameworků, například Symfony, snaží se o výběr toho nejlepšího z dostupných nástrojů.

Zápory

- Má tendenci produkovat v některých ohledech ?špagetový? kód, tedy velmi provázaný a hůře udržovatelný.[22]
- Pomalejší, než například Phalcon.
- Relativně nový nástroj a některé jeho součásti nejsou na tak vysoké úrovni, jako ostatní.

Phalcon

Klady

- Výkon – vytvořen jako nativní PHP knihovna napsaná v jazyce C, rychlost tohoto frameworku lze těžko překonat.
- Moderní konvence a paradigmata.

Zápory

- Napsaný jako nativní knihovna v C, což omezuje možnosti zkoumání kódu, ale především jeho debugování za použití nástrojů jako PHP Xdebug.
- Omezená komunita kvůli složitější instalaci a používání frameworku.
- Zaměřený na menší projekty.

[22] Ale jak vyplynulo ze článků zkoumaných během implementace, tento nedostatek byl po předchozí kritice zlepšen v nových verzích frameworku kvůli změně fungování controlleru.

Symfony2

Klady

- Poskytuje mnoho užitečných knihoven a nástrojů. Poměrně populární a rozšířený nástroj.

Zápory

- Často kritizovaný jako příliš rozsáhlý a pomalý.
- Velká část jeho knihoven je používána ostatními frameworky, například Laravelem, které si mohou díky modularitě tohoto frameworku vzít to nejlepší, co nabízí, zatímco nevhodné knihovny a obecně špatné vlastnosti zůstávají v Symfony2.
- Značně náročný na ovládnutí.
- Jedná se o zastaralejší framework.

Yii

Klady

- Správa databáze za pomoci takzvaných active records a migrací, což jsou moderní technologie, se kterými má autor této práce jisté zkušenosti.
- Klade důraz na rychlosti (i když je ze zvažovaných fameworků až na třetím místě, pravděpodobně se snaží o lepší škálovatelnost).
- Klade důraz na bezpečnost.
- Zaměřen na tvorbu sociálních sítí.

Zápory

- Menší komunita, i když údajně aktivní.
- Méně expresivní, než ostatní frameworky (potřeba napsat více kódu pro stejný výsledek).
- Je těžký na ovládnutí a efektivní používání.
- Dle [27] se webové aplikace tvořené za jeho pomoci s narůstající velikostí stávají z hlediska kódu těžko udržovatelnými, což je v rozporu s požadavky na systém Mojeskolka.

Další alternativy

Zvažovány byly také frameworky Zend, Code Igniter, CakePHP, Nette a další, ale z rozličných důvodů byly z výběru vyloučeny. Převážně se jednalo o jiné zaměření, než které má projekt Mojeskolka, menší on-line komunitu, problematickou škálovatelnost, nebo zastaralé technologie.

3.7.5 Zvolený framework

Především na základě silné on-line komunity, silného zaměření na usnadnění vývoje, ale i moderního přístup byl z uvedených alternativ zvolen framework Laravel.

Velmi vážným kandidátem byl i framework Yii s jeho zaměřením na tvorbu sociálních sítí a důrazem na zabezpečení, ale slabší on-line komunita a především často kritizovaná nepřehlednost kódu nakonec rozhodly v jeho neprospěch.

Hlavní uváděnou nevýhodou frameworku Laravel je jeho rychlost, která však ze srovnání v [22]vychází jako druhá nejvyšší. Ale i kdybychom rychlost stále brali jako nevýhodu frameworku Laravel, na základě [28] by tento nedostatek neměl mít tak zásadní dopad, jak se snaží tvrdit například autoři frameworku Phalcon.

Důvodem je skutečnost, že většinu času nutného pro vyřízení požadavku na webový server zabere přístup do databáze a samotná business logika specifická dané doméně, a tak měření rychlosti samotného frameworku není tolik podstatné. Výrazně zajímavější by bylo srovnání rychlosti toho samého systému implementovaného v různých frameworcích, ale těžko si jej představit vzhledem k rozsahu práce nutné pro jeho realizaci a obtížnému udržení stejné kvality a efektivity kódu napříč testovanými frameworky.

Z hlediska ostatních aspektů je Laravel na podobné úrovni, jako ostatní alternativy, pročež rozhodnutí bylo učiněno především na základě silné on-line komunity, snahy vycházet vstříc vývojářům a podporovat tvorbu přehledného kódu.

Co se posouzení učiněného rozhodnutí týče, z důvodů uvedených výše vyplývá, že pouze samotný průběh implementace může ukázat, zda se jednalo o správný výběr. Proto je jeho zhodnocení věnována celá sekce Využití frameworku Laravel v následující kapitole.

3.8 Výsledná podoba systému

Byl tedy zhotoven entitní model, ze kterého bude vycházet podoba datového úložiště a jádra systému, a zvolen nástroj, za pomoci kterého bude systém vyvíjen. Zbývá tedy návrh architektury serverové části a samotné aplikační logiky. Tato fáze návrhu je však do značné míry definována rozhodnutím, že pro implementaci bude použit framework Laravel.

Protože výběr frameworku výrazně ovlivní architekturu systému a jeho vnitřní uspořádání (jelikož diktuje podobu projektu a další pravidla, jak bylo vysvětleno v předchozí sekci), tak pokud nemá být vnitřní podoba výsledného prototypu v rozporu s architektonickým modelem a strukturou frameworku (což by bylo v rozporu s rozhodnutím, že podobný nástroj bude pro implementaci vůbec použit), pak již v návrhu aplikační logiky není mnoho prostoru pro podstatná rozhodnutí.

Prakticky celá vnitřní struktura systému je tedy určena frameworkem Laravel. Proto budou následující sekce věnovány právě tomu, jak je tento framework uvnitř uspořádaný, a tedy i tomu, jak bude vypadat aplikační logika a přístup k datovému úložišti v systému Mojeskolka.

Tomu, zda bylo možné a vhodné strukturu frameworku dodržovat, a jak skutečně probíhala implementace za jeho přičinění, se věnuje příslušná sekce Využití architektury frameworku, která také zmiňuje rozhodnutí, která bylo potřeba učinit mimo rámec architektury definované frameworkem.

3.8.1 MVC

Základem architektury frameworku Laravel, stejně jako většiny „plnohodnotných“ webových frameworků (takzvaných „full-stack“, oproti „lightweight“ frameworkům, ty slouží pro tvorbu méně komplexních webových systémů, kde je důraz kladen na rychlost a efektivitu), je architektonický vzor Model - View - Controller, většinou zkracovaný na „MVC“.

Jádrem tohoto architektonického vzoru je rozdělení aplikace na tři oddělené části, které spolu komunikují a spolupracují, ale jsou separovány tak, že náhrada některé z nich není nikterak obtížná, a zároveň jsou jednotlivé komponenty znovupoužitelné pro realizací různých funkcí uvnitř aplikace.

- **View:** Tato složka MVC slouží k prezentaci dat. Ani u webové aplikace nemusí produkovat pouze HTML kód, je možné mít různé pohledy, které zobrazují ta samá data v různých formátech, například HTML pro fyzické osoby a JSON v rámci webového API. View by mělo obsahovat minimum aplikační logiky, maximálně provádět jednoduché cykly a ověřovat podmínky nutné pro zobrazování dat.

- **Model:** Tato složka reflektuje doménu aplikace, tudíž i reálné objekty, které se v ní vyskytují. Vychází z entitního modelu a má tudíž poměrně přirozenou vazbu na entity v databázi, čehož využívá takzvané ORM, zmíněné níže. Zároveň obsahuje aplikační logiku a mělo by se jednat o jedinou část MVC, která ji obsahuje ve větší, než minimální míře.

- **Controller:** Controllery[23], ve frameworku Laravel slouží pro delegaci

[23]Počešťování anglických slov nečiní autorovi této práce radost, ale český ekvivalent „ovladač“ by evokoval nevhodné asociace, což implikuje skloňování slova controller a controllery.

> požadavků na příslušná view a jejich naplnění daty z modelu, i když o jeho ?pravé? funkci se vedou debaty. Mělo by se jednat o poměrně jednoduché a krátké třídy, které pouze vytváří vazbu mezi požadavkem, odpovídajícím view a příslušnými daty z modelu.

Stručně se dá říci, že tento návrhový vzor řeší korektní oddělení prezentační vrstvy a aplikační logiky.[24]

Do těchto logických celků budou spadat implementované třídy, které budou silně vázány na framework Laravel, jak vysvětluje sekce [**?**].

3.8.2 ORM

Na vrstvy definované v MVC navazuje ve frameworku Laravel, a tudíž bude navazovat i v systému Mojeskolka, takzvané objektově relační mapování (ORM), což je vrstva zajišťující přístup k persistentnímu datovému úložišti. Třída Model, od které příslušné třídy reprezentující model dědí, totiž obsahuje funkce, které umožňují jednoduchý přístup k databázi přímo voláním metod objektů, které jsou instancemi jednotlivých entit v systému.

Je tedy možné velmi dobře čitelným a objektově orientovaným způsobem přistupovat k persistentním datům, což vysvětluje značnou rozšířenost tohoto řešení mezi populárními webovými frameworky.

3.8.3 Absence třídního modelu

Důsledkem předchozích sekcí je absence třídního modelu v této kapitole. Důvodem je právě silná vazba na framework Laravel a skutečnost, že vytvářené třídy budou spadat do kategorií model, controller, nebo helper.[25]

Uvedením detailního modelu tříd by bylo pouze dosaženo mnoha duplicit. V modelové části MVC byl pouze kopírován detailní entitní model, uvedený výše, a v oblasti controlleru kopírován návrh UI a případy užití, jelikož všechny akce, které tyto sekce zmiňují, jsou realizovány v třídách do controlleru spadajících.

Ostatní funkce, jako je delegování požadavků na controllery, logika přístupu k databázi a další, řeší interní třídy frameworku Laravel. Vzhledem k jeho rozsahu a komplexnosti, ale především proto, že se jedná o oddělený

[24] Za zmínku stojí, že v dnešní době již označení MVC nabylo poněkud jiného významu, než jakého mělo v době svého vzniku. (Což však není tolik podivuhodné, jelikož tento architektonický vzor byl poprvé popsán v roce 1979).

Například bylo původní funkcí controlleru zachytávat uživatelský vstup, tedy pohyb myší a stisky klávesnice, jelikož tyto funkce nebyly řešeny pomocí OS, nebo knihoven, jak je tomu dnes. V současnosti však slouží controller ve webových aplikacích povětšinou k účelu, který byl popsán výše.

[25] Této poslední skupině tříd je věnována sekce View helpers, jelikož jejich potřeba a konkrétní podoba nemohla být během návrhu systému známa, protože silně souvisí se strukturou frameworku Laravel.

nástroj, jehož tvorba, ani detailní rozbor nejsou předmětem této práce (pouze v rozsahu, který přímo ovlivňuje její tvorbu a podobu prototypu), nepovažuje autor za rozumné takovýto celkový třídní model uvádět (mimo jiné proto, že takovýto model i s použitými knihovnami obsahuje přes pět set tříd).

Na přiloženém disku je však možné nalézt výslednou dokumentaci, která obsahuje i diagram tříd vytvořených v této práci (vyjmuty z něj jsou pouze drobné třídy příliš specifické pro Laravel, které by působily zcela vytržené z kontextu, například validátory požadavků).

3.9 Dlouhodobý rozvoj projektu

V této kapitole budou navrženy postupy a zvoleny nástroje, které pomohou projektu Mojeskolka s dlouhodobým rozvojem a dalším zdokonalováním. Její podoba vychází ze zadání práce a nefunkčních požadavků na systém, ale především snahy o tvorbu kvalitního a úspěšného produktu, jelikož snadná údržba a možnost budoucího rozvoj jsou důležitou součástí dlouhodobého úspěchu.

3.9.1 Předpoklady budoucího úspěchu

Bude vhodné krátce se pozastavit nad tím, co přesně definuje úspěšný softwarový projekt – schopnost vzbudit zájem uživatelů, užitečná a originální myšlenka projektu, kvalitní zpracování a možnost dalšího růstu, tedy rozšiřování okruhu zákazníků.

První tři body byly rozvedeny v předchozích kapitolách této práce.

1. Snaha vzbudit zájem uživatelů a vycházet jim vstříc byla zachycena v předchozí kapitole Návrh uživatelského rozhraní, která se snaží do detailu navrhnout UI systému tak, aby bylo jeho užívání pro koncové uživatele co možná nejpříjemnější a celkové UX bylo na dobré úrovni. Významnou složkou schopnosti produktu přilákat uživatele bude také marketing a business strategie, které však nejsou předmětem této práce.

2. Ústřední myšlenka projektu byla a je dotazovanými potenciálními uživateli hodnocena pozitivně, ale její tvorba ani změny nejsou předmětem této práce. Pouze bylo v rámci rešerší zkoumány konkurenční systémy, jejichž zaměření jiným směrem, než je plánovaný pro projekt Mojeskolka, do jisté míry potvrzuje originalitu celého nápadu.

3. Prvním krokem ke kvalitnímu zpracování byla důkladná analýza a návrh, vypracované v této práci. Společně s rešeršemi a dalšími vypracovanými podklady by měla tato práce poskytovat solidní základ budoucího produktu.

Zbývá tedy možnost dalšího růstu, tedy schopnost produkt zdokonalovat a dále rozvíjet. Možnosti, jak přispět k úspěšnosti tohoto procesu, jsou rozvedeny níže.

3.9.2 Sběr statistik

Pokud má být produkt v budoucnu zdokonalován, je velmi důležité především zjistit, co vlastně by mělo být zlepšeno a jak. Důležitým nástrojem v tomto ohledu je sběr a analýza rozličných statistických údajů. Především se jedná o odpověď na otázky, jací uživatelé z jakých zařízení k systému přistupují, jak jej používají a jaké části systému jsou nejvíce vytížené.

K těmto účelům mohou jednak posloužit již hotové nástroje, jednak je možné získávat informace z samotného systému, především z databázového stroje.

Využití nástrojů pro sběr statistických dat

Pro analýzu chování uživatelů je možné využít již hotových nástrojů, jakými je například služba Analytics od Google [29], nebo takzvané heatmapy, tedy nástroje, které zkoumají interakci uživatelů se stránkami v systému a shromažďují data o jejich procházení, například [30].

Tyto údaje poskytnou jednak informace o kvalitě UI (například by bylo viditelné, že uživatelé se snaží klikat na neinteraktivní prvky na stránce, nebo že prochází stránky chaoticky a nevyužívají přítomných odkazů, což signalizuje potřebu změny), o rozměrech a typu zařízení, na které by bylo vhodné primárně cílit, nebo o rozložení aktivity uživatelů, tedy v které části dne nastává v systému ?špička? a přistupuje k němu nejvíce uživatelů.

Sledování chodu systému

Zároveň by bylo vhodné sledovat interní chování systému, tedy dotazy do databáze, ale i vytížení systémových zdrojů, minimálně kvůli možnosti přetížení systému, která by signalizovala nutnost zvýšení výkonu, ať už zrychlením systému, nebo užitím výkonnějšího serveru.

Velké potenciální přínosy může přinést analýza databázových dotazů, která dokáže identifikovat problémy týkající se přílišné zátěže některých částí systému, nevhodné indexace tabulek v databázi, nebo nedokonalé práce s přístupem k ní, například v podobě přílišného množství dotazů.

Pro úvodní ladění systému také bude vhodný nástroj Laravel Debugbar [31], což je rozšíření frameworku Laravel, které poskytuje údaje o fungování daného systému. Tento nástroj ukazuje, jaké požadavky byly vyřízeny pro zobrazení dané stránky, jaké pohledy (view) byly vykresleny a jaká data do nich byla předána. Ale především poskytuje seznam všech dotazů do databáze, které byly vykonány, aby daná stránka mohla být zobrazena, včetně konkrét-

ních předávaných dat a návratových hodnot, což poskytuje nesmírně cenné údaje pro optimalizaci systému.

3.9.3 Rozšíření na další platformy

V rámci úvah o budoucnosti systému by bylo vhodné myslet i na rozšíření systému na mobilní platformy. A to nejen v podobě responzivních webových stránek, které budou výstupem této práce, ale i v podobě nativních aplikací pro různé mobilní operační systémy.

Jak bylo zjištěno při tvorbě rešerší, vzhled responzivních stránek bývá dobrým základem pro tvorbu mobilních aplikací, což je prvním krokem k jejich vytvoření, jelikož bude možné využít již hotového návrhu UI, upraveného pouze o vhodné detaily. Také existují již připravené nástroje, které mohou posloužit k snazšímu vytvoření mobilních aplikací – například Native CSS [32] nebo PhoneGap [33] pro převod UI. Jednalo by se však spíše o krátkodobé řešení, jelikož uživatelé jsou zvyklí na plně nativní aplikace vystavěné od základu právě pro daný mobilní operační systém.

3.9.4 Bezpečnost

Dá se očekávat, že každý systém, obzvláště v případě jeho úspěchu a rozšíření mezi uživatele, bude vystaven hrozbě elektronických útoků v jejich mnoha rozličných podobách. Proto by bylo vhodné při tvorbě systému s možností těchto útoků počítat a systém na ně připravit.

Vnější

Vnějšími útoky máme na mysli především DDoS, CRSF a další.[26]

Se zabezpečením proti vnějším útokům do velké míry pomáhá framework Laravel, jelikož například proti CRSF útokům užívá speciálního tokenu, obsaženého ve všech formulářích, bez jehož korektní hodnoty (generované náhodně a platné vždy jen po krátký časový úsek) není daný požadavek autorizován, což efektivně chrání proti těmto typům útoku.

Ochrana proti DDoS útokům vyžaduje robustní serverovou architekturu, která musí být po stránce softwaru i hardwaru připravena na podobný typ útoku. Detailní rozbor takovýchto řešení však leží mimo rámec této práce.

Vnitřní

Útok na systém nemusí přicházet pouze z vnějšího prostředí, naopak se dá očekávat, že někteří uživatelé se budou pokoušet prolomit zabezpečení systému „zevnitř“. Důležitými aspekty, na které je potřeba v rámci prevence

[26] Vysvětlení těchto zkratek je možno nalézt v příloze Seznam použitých zkratek.

bezpečnostních incidentů dbát, je autorizace uživatelů, ošetření jimi zadávaných vstupů a aktivní správa systému, která souvisí se sledováním stavu systému a sběrem údajů o chování uživatelů, tedy například logováním jejich akcí v systému.

Typickými útoky, které hrozí od přihlášeného uživatele, je XSS a SQL injection. Stejně, jako v případě CRSF útoků, pomáhá při obraně proti těmto druhům napadení použití frameworku.

Útokům založeným na SQL injection je možno zabránit použitím knihovních funkcí (obsažených v ORM) při vkládání dat do databáze. Tyto funkce automaticky korektně vyčistí přijatý vstup od SQL příkazů, čímž je zabráněno vykonání škodlivého SQL kódu, který by v přijatých datech mohl být obsažen. XSS útokům je taktéž nutné zabránit za pomoci kontroly uživatelských vstupů a jejich korektního vyčištění.

Do budoucna je vhodné uvažovat i o oddělení databázového stroje od samotného serveru a vložení další bezpečnostní vrstvy, která by v případě, že by nepovolaná osoba získala vysokých oprávnění v systému, zabránila škodám na persistentních datech. Konkrétní detaily takovýchto opatření však leží mimo rámec této práce.

Nástroje pro zvýšení bezpečnosti

Jazyk PHP obecně nebývá spojován s nejvyšší úrovní zabezpečení, a proto se na tento nedostatek snaží zaměřit některé softwarové knihovny. Jedná se například o Suhosin [34] a další nástroje, které umožňují snížit šance na prolomení zabezpečení systému.

Tyto knihovny, společně s dostupnými on-line články o zabezpečení, budou pomáhat při zvýšení celkové úrovně bezpečnosti. Zároveň je v rámci projektu plánována spolupráce s IT firmou, která se na zabezpečení softwarových produktů specializuje, ale detaily této spolupráce opět nejsou předmětem této práce.

3.9.5 Přidávání a zdokonalování funkcí

Za zmínku stojí, že během tvorby této práce byly vytvářeny seznamy funkcí a změn, které by do práce v budoucnu mohly, nebo přímo měly být zapracovány. Tyto seznamy dohromady čítají téměř padesát stran dokumentů, což představuje několik stovek záznamů.

Tím byly zachyceny myšlenky a nápady, které vznikly při tvorbě této práce, ale nemohly být z rozličných důvodů realizovány.

Zároveň je však důležité vyvarovat se snahy přidávat do systému přílišné množství funkcí, jelikož, jak je opakovaně poukazováno, uživatelé mají nejraději jednoduché a prosté systémy, které plní svou funkci a nerozptylují uživatele přebytkem možných akcí. Nalézt rovnováhu mezi množstvím užitečných funkcí systému a jeho jednoduchostí však není snadný úkol.

3.9.6 Závěr

Nepochybně nebude snadné produkt dovést na úroveň skutečně kvalitního informačního systému, jelikož tento proces vyžaduje precizní splnění velkého množství rozličných a nesnadných úkolů, ale poznatky uvedené v této sekci by měly na této cestě pomoci.

Další složkou, která ovlivní budoucnost projektu, je jeho škálovatelnost, tedy schopnost obsluhovat do budoucna velké a rostoucí množství uživatelů, které by se dalo očekávat, pokud systém bude úspěšný. Tomuto aspektu vývoje do budoucna je věnována následující kapitola.

3.10 Odolnost systému vůči zátěži

S možností rozvoje souvisí i schopnost systému odolávat větší zátěži, které bude vystaven v případě, že produkt bude využívat větší množství mateřských školek. Vhodným způsobem, jak systém škálovat do budoucna, je jeho rozdělení na menší logické celky, které spolu navzájem komunikují přes určená rozhraní. Tedy rozdělení na jednotlivé webové služby.

Výhodou takovéhoto rozdělení je, že služby mohou být spuštěny na oddělených serverech, čímž se rozloží zátěž na systém, respektive subsystémy, což umožní zvýšení jeho celkového výkonu.

Jedním z možných rozdělení systému za účelem lepší škálovatelnosti je oddělení jednotlivých modulů do samostatných webových služeb, kteréžto řešení bylo nastíněno v sekci o modulech výše v této kapitole.

Druhou alternativou je však pohled na systém, který jej rozděluje v jiné rovině.

3.10.1 Oddělení jednotlivých školek

Z návrhu systému totiž vyplývá, že neexistují vazby mezi jednotlivými školkami. Děti, skupiny dětí i notifikace náleží pouze do jedné školky, což bylo zmíněno i výše v rozboru hrubého entitního modelu. Toto omezení poskytuje potenciál pro velmi efektivní rozložení zátěže.

Zmíněná nezávislost školek totiž umožňuje dělit systém nikoliv podle skupin entit – do modulů – ale právě podle jednotlivých školek, tedy podle skupin *instancí* daných entit. Tím by bylo dosaženo velmi dobrého rozložení zátěže na jednotlivé servery, které mohou obsluhovat dle potřeby jednu, nebo více školek, které jsou na sobě navzájem nezávislé.

Společným bodem, který propojuje jednotlivé školky, jsou uživatelé. Mimo jiné proto, že návrh umožňuje, aby jeden uživatel měl současně přiřazeny různé role v různých školkách. Z toho plyne, že uživatelé nemohou být evidováni v rámci jednotlivých školek, ale záznamy o nich musí být ukládány v centrálním modulu, který by měl zajišťovat evidenci a přihlašování uživatelů a

sloužit jako vstupní bod do celého systému (což je v souladu s popisem uživatelského modulu uvedeným výše). Tomuto rozdělení odpovídá i umístění záznamů o rolích přidělených danému uživateli (UserHasRoleInKindergarten) do modulu školky.

Pro realizaci tohoto rozdělení by centrální modul musel navíc pouze uchovávat záznamy o tom, v jakých školkách má uživatel nějakou roli, a po přihlášení daného uživatele by se příslušných serverů dotázal, o jaké role se jedná. Následně by uživateli umožnil do systému těchto školek přístup, případně by z nich sám agregoval data, která by si uživatel vyžádal. Záznamy o školkách, ve kterých mají uživatelé přiřazené role, by byly aktualizovány v okamžiku, kdy je uživateli nově přiřazena, nebo odebrána role v nějaké školce, což by však měla být poměrně vzácná operace.

Toto rozdělení by mělo umožňovat velkou škálovatelnost systému, ve kterém by se kritickým bodem z hlediska zátěže stal centrální modul, který by však neměl být ani takto příliš vytížený, zvláště v porovnání s vytížením původního monolitického systému, ve kterém by databáze obsahovala údaje všech školek.

Jedná se o jistou formu řešení logického problému za pomoci metody rozděl a panuj, která dělí velký, hůře zpracovatelný celek na menší části. Jeho realizace je otázkou budoucnosti, implementovaný prototyp bude mít zmiňovanou monolitickou strukturu, ale podstatným bodem je, že navržený systém je na podobné rozdělení připraven a v případě jeho nutnosti by v budoucnu umožňoval přechod z monolitické podoby do podoby separovaných školek.

3.11 Závěr

V této kapitole byl iterativním postupem sestaven model entit a modulů, který zároveň poslouží jako základ pro návrh databáze. Na jeho základě byl vybrán vhodný nástroj pro implementaci systému – PHP framework Laravel.

Dále byly v souladu se zadáním práce analyzovány a navrženy postupy, jak bude možné podpořit dlouhodobý rozvoj systému a připravit jej do budoucna na potenciální zátěž v podobě velkého množství školek a jim příslušících uživatelů.

Kvalita zhotoveného návrhu a učiněných rozhodnutí by měla možnost plně se projevit až v okamžiku, kdy by systém byl ostře nasazen a několik let sloužil reálným zákazníkům. Korektnost učiněných voleb a velká část případných problémů z nich vycházejících by se však měla ukázat, respektive projevit již při implementaci jeho prototypu, pročež budou učiněné kroky a zhotovený návrh v následující kapitole zpětně zhodnoceny.

Kapitola 4

Implementace

V rámci této kapitoly byl vytvořen funkční prototyp systému Mojeskolka na základě návrhu sestaveného v předchozích kapitolách.

Vybraným detailům tohoto procesu a zhodnocení dřívějších rozhodnutí a zvolených nástrojů jsou věnovány následující sekce.

4.1 Tvorba prototypu

Vývoj probíhal v IDE Netbeans [35], které se však ukázalo nedostatečně vybaveným pro práci s frameworkem Laravel, pročež by do budoucna bylo vhodné zvážit přechod k IDE PhpStorm [36], které poskytuje lepší nástroje pro spolupráci se zvoleným frameworkem.[27]

Pro implementaci na lokálním počítači, který provozuje operační systém Microsoft Windows 7, byl využit nástroj Wamp [37], který poskytuje prostředí pro vývoj webových aplikací, a pro který byl následně vytvořen lokální virtuální hosting v systému Windows.

Toto nastavení umožňuje lokálně velmi dobře simulovat provoz skutečného webového serveru, což se vyplatilo v závěru implementace, kdy byl lokálně vyvíjený systém bez problému přesunut na testovací server, kde bez nutnosti změn v nastavení úspěšně fungoval.

Velmi užitečným se prokázalo být PHP rozšíření Xdebug [38] umožňující interaktivní ladění PHP kódu, které výrazně usnadňuje vývoj v tomto programovacím jazyce, zvláště po stránce hledání a opravy chyb.

Pro správu verzí systému byl použit nástroj GIT [39]. Navazovat na jeho funkce měl nástroj pro automatický přesun GIT repozitáře na webový server, avšak snaha zprovoznit příslušný program se nesetkala s úspěchem, kvůli konfliktu verzí PHP.[28] Tento záměr však byl pouze odložen a vzhledem k množství

[27] Jak je možné usoudit na základě zhlédnutých video-tutoriálů, které budou zmíněny níže.

[28] Nástroj pro synchronizaci verzí byl vytvořen v jazyce PHP a nebyl kompatibilní s lokální instalací PHP interpreteru. Se samotným kódem prototypu problém nebyl.

práce, kterou takovéto řešení šetří, je podobný automatizovaný postup plánován do budoucna.

Výstupem implementace je fungující a použitelný prototyp, který splnil požadavky na něj kladené a do značné míry překonal očekávání autora této práce. Použité nástroje se ukázaly být velmi kvalitní a skutečně dobře použitelné, což výrazně urychlilo vývoj prototypu a umožnilo jeho rozsáhlou implementaci.[29] Zhodnocením použitých technologií se detailnější zabývají sekce uvedené níže.

Přesto však některé složitější součásti prototypu, které nebyly v rámci zadání práce vyžadovány, nebylo možné implementovat, povětšinou proto, že byl kladen důraz na jiné aspekty implementace. Tyto nedokončené části jsou zmíněny níže v sekci Nedokončené části.

Výsledný prototyp byl následně prezentován všem osobám spojeným s touto prací, tedy vedoucímu práce, oponentovi a na závěr i zadavateli projektu, tedy autorovi koncepce Mojeskolka. Obrazovky z vytvořeného prototypu je možné nalézt v Ukázky vytvořeného prototypu.

4.2 Využití předchozích kapitol

Účelem této sekce je zhodnotit, zda byl návrh uživatelského rozhraní, aplikační logiky a datového úložiště úspěšný a zda podle něj bylo možné vytvořit funkční prototyp, nebo zda bylo nutné předchozí rozhodnutí měnit.

4.2.1 Uživatelské rozhraní

Implementace byla zvláště v této oblasti poměrně časově náročná, především kvůli rozsahu systému a detailnosti navrženého UI (tedy například odlišení dětí a jednotlivých panelů entit, vysouvání panelů nastavení, animací otevírání notifikace a užití barev pro tvorbu přehledného uživatelského rozhraní), ale mockupy vytvořené při návrhu UI tuto tvorbu výrazně usnadnily a zároveň zaručily, že stránky v systému skutečně byly vytvářeny se zaměřením na jejich snadnou použitelnost.

Předchozí návrh uživatelského rozhraní se tedy projevil jako velmi užitečný podklad, jelikož jednak urychlil tvorbu prototypu, ale především proto, že během intenzivního vývoje má programátor přirozeně tendenci vidět systém jinak, než koncový uživatel. Dá se předpokládat, že by tento „syndrom klapek na očích“ bez předchozí tvorby vhodného návrhu UI vedl na hůře použitelné uživatelské rozhraní.

Responzivní design

Za zmínku stojí, že v průběhu implementace stránek byly všechny stránky rodičovského rozhraní primárně vyvíjeny ve zmenšeném okně prohlížeče, simulujícím rozměry a vlastnosti mobilních zařízení, aby byly zajištěny dobré

[29] Která však i tak zabrala okolo tří set hodin čistého času, vzhledem k rozsahu prototypu.

responsivní vlastnosti. To mělo za následek celkové zaměření výsledného prototypu spíše na mobilní zařízení, nežli na stolní počítače, což by však vzhledem k předpokládanému využívání produktu nemělo býti na závadu.

4.2.2 Aplikační logika a datové úložiště

Model entit se ukázal být velmi dobrým základem pro tvorbu aplikační logiky, zvláště tříd náležících do modelu, jelikož korektně vyřešil problematické vztahy mezi entitami v systému. Bylo potřeba pouze jemných dodatečných úprav modelu entit, aby se opravily drobné nedostatky (způsobené nekorektní představou o fungování dané části systému, což však odpovídá iterativnímu vývoji), nebo přidaly nové prvky umožňující dodatečnou funkcionalitu, která byla shledána důležitou.

Databázový model a migrace

Drobným zklamáním byla nemožnost snadno využít databázového návrhu vytvořeného v programu Enterprise Architect[30] pro naplnění databáze příslušnými tabulkami. Důvodem byl nesnadný převod SQL scriptu pro tvorbu databáze na takzvané migrace, užívané pro definici databáze frameworkem Laravel. Možností bylo nevyužít pro definici databáze konceptu migrací a setrvat u používání SQL scriptu, avšak v souladu se snahou respektovat vnitřní mechanismy fungování frameworku bylo rozhodnuto, že se využije migrací, které budou vytvořeny na základě již hotového databázového modelu.

Toto rozhodnutí se ukázalo býti velmi vhodným, jelikož tvorba migrací zabere zkušenému uživateli minimální množství času, zvláště, pokud využije nástroje Artisan, kterému je věnována samostatná sekce níže v této kapitole. Zároveň je při generaci migrací možné nechat vytvořit odpovídající model a další komponenty systému, což urychluje následnou implementaci.

Dalšími detailům užívání zvoleného frameworku se věnuje následující sekce.

4.3 Využití frameworku Laravel

Framework Laravel, zvolený pro implementaci prototypu, se ukázal být skutečně mocným, ale zároveň snadno použitelným nástrojem. Na tomto nástroji je vidět, že si vzal inspiraci z již existujících populárních řešení, jako je Ruby on Rails (s jeho convention over configuration, migracemi a dalšími aspekty), Java Spring (ze kterého je převzat princip IoC), Symfony2 a dalších, spojil dohromady jejich výhody a s velkou pečlivostí se snažil eliminovat jejich nedostatky.

[30] Jehož výstupem jsou modely uvedené výše v kapitole Návrh aplikační logiky a datového úložiště. Za zmínku stojí, že k tomuto programu byla v průběhu práce zakoupena licence pro tvorbu komerčního produktu, kterým Mojeskolka má v budoucnu býti.

Při používání tohoto nástroje je cítit snaha maximálně usnadnit vývoj softwarového produktu, vyjít programátorovi vstříc a umožnit mu tvorbu krátkého, čistého a přehledného kódu. S nadsázkou se dá říci, že tvorba prototypu neprobíhala ani tolik v jazyce PHP, alespoň ne takovém, jaký je tímto slovem běžně označován, jelikož nevýhody a úskalí tohoto programovacího jazyka framework efektivně odstiňuje a poskytuje velmi příjemné programovací prostředí.[31]

4.3.1 Komunita

Výše zmíněná pozitiva však nebyla při výběru frameworku známa – rozhodnutí upřednostnit Laravel bylo učiněno z velké části na základě jeho rozšířenosti. A toto rozhodnutí se ukázalo být velmi smysluplné, jelikož on-line komunita poskytuje obrovské množství knihoven a zdrojů právě pro tento nástroj a skutečně výrazně pomohla při vývoji prototypu.

Pro řešení typických problémů, na které bylo při tvorbě systému naráženo, již existuje mnoho on-line doporučení a rozborů na stránkách jako StackOverflow [7], nebo Laracasts [40], což je webový portál věnovaný přímo tomuto frameworku, na kterém je možné nalézt velké množství návodů, ale i diskuzní fórum věnované vývoji za použití tohoto nástroje.

Na portálu Laracasts je dokonce dostupná několikahodinová série video tutoriálů, které od začátku do konce popisují a názorně předvádí tvorbu webové aplikace ve frameworku Laravel [41].

Je možné vděčit právě těmto tutoriálům, dostupnosti informací a silné online komunitě za to, že při implementaci prototypu Mojeskolka nevyvstaly žádné zásadní problémy a všechny překážky, očekávatelné při vývoji nového systému ya použití zcela neznámého nástroje, byly bez větších obtíží překonány.

4.3.2 Využití architektury frameworku

V souladu s návrhem aplikační logiky bylo při její implementaci postupováno dle architektonických pravidel definovaných frameworkem Laravel a bylo dbáno i dalších doporučení pro tvorbu aplikací v tomto nástroji.

Důvěra v Laravel se i v tomto případě ukázala býti opodstatněná, jelikož struktura projektu, vycházející z MVC, výrazně omezila potřebu složitých architektonických rozhodnutí, a zároveň poskytovala velice pevnou kostru, kterou bylo možné snadno obalit kódem nutným pro fungování systému Mojeskolka (což je účel návrhového vzoru IoC). Postačovalo implementovat správně třídy spadající do modelu a controlleru a následně vypracovat samotná view, doplněná o podpůrné třídy. Ty byly zhotoveny jednak pro jednotlivé stránky,

[31] Pro autora této práce to bylo poprvé po dlouhé době, kdy přišel k novému nástroji, netušil, co ho čeká, byl šokován použitelností a kvalitou používaného produktu a nejednalo se o šok v negativním slova smyslu...

ale i samostatné UI komponenty (uvedené výše, například v sekci Komponenty v rodičovském rozhraní), které bylo možné pohodlně skládat dohromady a vkládat do view jednotlivých stránek za pomoci mechanismů templatového enginu Blade, který je ve frameworku používán.

Pro typické úkony existují ve frameworku předdefinované mechanismy a je implementující třídy, které lze snadno rozšířit a přidat požadovanou aplikační logiku, což opětsouvisí s principem IoC. Krom výše uvedených součásti MVC se jedná například o třídu Request, jejíž potomci zajišťují validaci uživatelského vstupu z jednotlivých požadavků zasílaných do systému, ViewComposer, jejíž potomci naplňují jednotlivé pohledy v aplikaci sdílenými daty (například přidávají menu, které je společné pro mnoho stránek, pročež by bylo zbytečné tento kód duplikovat ve všech controllerech) a třídy Middleware, které omezují přístup do systému na základě autorizace uživatelů.

View helpers

Jediným zajímavějším oříškem z hlediska struktury projektu se ukázala být vazba mezi modelem a view, tedy problém, jak předat do view z modelu data, která jsou však strukturovaná právě pro toto konkrétní zobrazení (a žádné jiné, nebo jen pro velmi omezenou skupinu zobrazení). Dá se říci, že jde o tvorbu a předání různě rozsáhlého a strukturovaného pohledu na data v modelu, kterýžto pohled je specifický pro daný okruh stránek, nebo grafických prvků.

Typickým příkladem je problém, jak předat hlavní straně aplikace všechny notifikace určené danému uživateli, které jsou roztříděné do sekcí podle data a dětí, kterým náleží. Problémem je zde onen výraz „strukturovaná data“, zmíněný výše, jelikož je třeba pro view připravit informace tak, aby nemuselo provádět žádné složité logické operace, které do view nepatří.

Tento problém je možné řešit zasíláním asociativního pole zachycujícího příslušnou stukturu dat. Ale co když bude třeba tyto notifikace dále filtrovat, či řadit? Nebo přidat další hierarchickou úroveň? A jak je vůbec možné získat všechny notifikace, které náleží danému uživateli, roztříděné podle jeho dětí? To vyžaduje poměrně značné množství aplikační logiky a kódu, který by měl být obsažen v samostatných funkcích a třídách.

A kam by bylo správné tyto funkce nebo třídy umístit? V případě posílání asociativního pole by toto pole musel připravit controller, ale aplikační logiku by na základě pravidel uvedených výše v sekci věnované MVC měl obstarávat model. Navíc by tímto řešením trpěl objektový přístup, jelikož controller, jehož úkolem je jen předávat data z modelu do view, by se měl stát multifunkčním a velmi obsáhlým objektem, který slouží mnoha velmi odlišným účelům. A to není v souladu s korektním objektovým návrhem, což implikuje, že controller by měl být ponechán stranou a pouze tato data předat do view.

Zároveň však není možné všechny podobné metody umístit do samotného modelu, který by měl být velmi obecný – pokud by do něj byly přidávány me-

tody příslušící vždy jen jednomu, nebo několika málo view, pak by se příslušná třída stala nepřehlednou a těžko udržovatelnou.

A view slouží pouze pro prezentaci dat, nemělo by obsahovat aplikační logiku. Zůstává tedy pouze zmiňovaná vazba model – view, respektive nějaký mezičlánek na této cestě.

Zdá se, že právě tato vazba je nejméně vytíženou a používanou částí trojúhelníku M-V-C. A kupodivu pro ni framework Laravel neposkytuje jasné řešení. Pravděpodobně i proto, že soudě dle on-line diskusí[32] a článků se této vazbě někteří vývojáři snaží vyhnout, jelikož podle nich není „korektní“ propojovat model a view. Avšak snaha obcházet problém jinou cestou vede typicky na ještě horší řešení, jak bylo nastíněno výše.

Řešení
Touto problematikou se zabývá článek [42], na základě kterého autor této práce pochopil, že daný problém se v MVC aplikaci vyskytuje zcela běžně, i to, jakým způsobem ho korektně řešit.

Z předchozí rozvahy je viditelné, že kód připravující tato data by očividně bylo vhodné strukturovat do tříd, z nichž některé se starají o načítání notifikací všech dětí daného uživatele, jiné obsahují notifikace podle určitého časového úseku a další slouží k následné filtraci těchto notifikací. Vzniká tím samostatná skupina tříd, kterou je vhodné oddělit do dedikovaného mezičlánku, nazvaného ve výše zmíněném článku jako „View Helpers“, nebo také „View Model“. Tyto třídy umí přijmout data od modelu a strukturovat je tak, aby k nim specifické view mohlo snadno přistupovat, čímž nezatěžují ani model, ani view, a udržují související kód na jednom místě a strukturovaný do přehledných tříd.

Tento problém představoval zajímavou logickou, i architektonickou hádanku, kterou bylo nutné v rámci implementace vyřešit. Po jejím rozluštění však bylo možné lépe pochopit tuto „hranu“ trojúhelníku MVC, refactorovat staré třídy, které se neúspěšně snažily tento problém řešit, a dosáhnout větší architektonické čistoty výsledného kódu.[33]

4.4 Zhodnocení výběru použitých nástrojů

Krom samotného frameworku Laravel byly při vývoji prototypu použity další nástroje, které výrazně usnadnily jeho tvorbu.

4.4.1 Composer

Tento nástroj plní dvě velmi důležité funkce – jednak umožňuje snadnou správu závislostí projektu, tedy automatické stahování a instalaci požado-

[32] Především na respektované stránce StackOverflow, ale i na dalších místech.

[33] Po refactoringu dotčených tříd se kód skutečně zpřehlednil a ulehčilo se jak třídám ve složce controllers, tak modelu, tak autorovi této práce.

vaných PHP knihoven; a zároveň umožňuje automatické načítání tříd používaných v jednotlivých souborech – bez nutnosti užívat PHP konstrukty require_once, include, nebo jim podobné, které velmi znepříjemňují tvorbu PHP projektu.

Pohodlí, které poskytuje automatické načítání tříd v porovnání s útrapami manuálního načítání požadovaných souborů (zvláště při následném refactoringu tříd), je těžko popsatelné.

Tento nástroj je právem považován za jeden z největších průlomů posledních let, co se jazyka PHP týče, jelikož přináší automatizaci zdlouhavých a složitých úkonů, které brzdí rozvoj aplikace.

4.4.2 Artisan

Nástroj Artisan, sloužící pro práci se soubory projektu za pomoci příkazové řádky, je sice součástí frameworku Laravel, ale zasluhuje samostatnou zmínku, díky značně odlišnému přístupu k práci s projektem a velkému množství užitečných funkcí.

Tento nástroj umožňuje pomocí jednoduchých příkazů provést inicializaci databáze na základě vytvořených migrací (definovaných v jazyce PHP), naplnění databáze vzorovými daty, tvorbu modelů a controllerů odpovídajících vytvořeným migracím a mnoho dalších úkonů.

Kombinací nástrojů Artisan a Composer je možné (po přenosu minima nutných zdrojových souborů projektu na server) nechat automaticky stáhnout požadované knihovny, inicializovat na tomto serveru databázi na základě migrací a lokálních konfiguračních souborů a naplnit ji vzorovými daty, která slouží pro testování, nebo inicializaci systému.

Tento postup je velmi rychlý, velmi pohodlný a opět zdůrazňuje kvalitu frameworku Laravel.

4.4.3 Laravel Debugbar

Tento nástroj je vyvíjen nezávisle na samotném frameworku, ale umožňuje nahlédnout do již hotového systému a získat velmi cenné údaje o jeho běhu. Byl již blíže popsán výše v sekci Sledování chodu systému, ale jeho pozitiva se projevila ještě dříve, než předpokládala zmíněná sekce. Již při implementaci a následném testování byl použit pro analýzu fungování systému, usnadnil pochopení fungování frameworku a pomohl s odstraněním některých nedostatků. Taktéž potvrdil domněnku o výkonu systému, zmíněnou v sekci Zvolený framework, tedy že úkony samotného frameworku tvoří relativně malou část celkové doby zpracování požadavku serverem (jedná se o méně, jak jednu třetinu).[34]

[34] Výsledný výkon prototypu je však jednou z jeho největších slabin, jak bude zmíněno níže v kapitole Zátěžové testy.

4.4.4 Bootstrap

Jak bylo zmíněno výše v sekci Technologie, pro tvorbu UI byl použit grafický framework Bootstrap. Ten umožnil velice snadnou tvorbu responsivního webu, který navzdory absenci dedikovaného grafického designu působí poměrně profesionálním dojmem, což je zásluhou především tohoto nástroje.

4.4.5 LESS

Také použití jazyka LESS pro generování CSS souborů splnilo očekávání, jelikož tvorba v tomto jazyce byla výrazně snazší, než je psaní klasických CSS souborů, a umožnila například snadno generovat barevné styly panelů příslušících jednotlivým dětem.

4.5 Nedokončené části

Ne všechny plánované části prototypu bylo možné dokončit v dostačující kvalitě, proto v této sekci budou zmíněny důležité části systému, které bude potřeba implementovat nebo výrazně zdokonalit až dodatečně.

4.5.1 Autorizace

Autentizace uživatelů byla zpracována korektně a systém by neměl dovolovat nepřihlášeným návštěvníkům stránek vykonat jakékoli akce, které přísluší pouze přihlášeným uživatelům (to zajišťuje vhodné nastavení frameworku Laravel).

Problém však představuje následující autorizace, jelikož některé akce je možné vykonávat i nad daty, ke kterým by uživatel neměl mít přístup. Například je možné pomocí změny adresy přistupovat k dětem jiných uživatelů, nebo pomocí manipulace formuláře poslat zprávu i učitelům či rodičům z jiné školky.

Tento problém by měl být adresován přednostně a musí být vyřešen před ostrým spuštěním aplikace, jelikož se jedná o významné bezpečnostní riziko.

4.5.2 Sekce notifikací

Ačkoli byl mechanismus přidávání dalších sekcí do dané notifikace navržen, nebyl v prototypu implementován, jelikož by vyžadoval další úpravy v uživatelském rozhraní, databázi a aplikační logice. Navíc se nejedná o funkcionalitu vyžadovanou zadáním práce nebo případy užití systému, pročež nebyla vyhodnocena jako důležitá priorita.

4.6 Dokumentace

K vytvořenému prototypu byla za pomoci nástroje phpDocumentor [43] vygenerována dokumentace, která zachycuje hierarchii tříd, jejich metody a inline dokumentaci.

Výslednou dokumentaci, obsahující seznam tříd vytvořených při implementaci a jejichstručný rozbor, je možné nalézt v příslušném souboru přiloženém na disku.

4.7 Závěr

Produktem této kapitoly je jednak hotový prototyp, ale také zhodnocení použitelnosti předchozího návrhu a zvolených nástrojů. Celkově je možné výsledek označit za úspěšný, jelikož bylo možné plně využít návrhu vytvořeného v této práci, zvolené nástroje se ukázaly býti velmi užitečnými a výsledný prototyp je použitelný a demonstruje klíčové funkce systému Mojeskolka.

Stav prototypu je však dle očekávání nepostačující pro ostré nasazení v mateřských školkách. Krom nutnosti přidání autorizace i některých dalších funkcí a opravy chyb, které budou nalezeny při testování, je citelná ještě potřeba refactoringu výsledného kódu. Projevila se skutečnost, že se jedná o první setkání autora s tímto frameworkem a s velkým PHP frameworkem obecně. Díky postupnému hlubšímu a lepšímu pochopení fungování frameworku Laravel a jeho architektury, které bylo nabyto v průběhu tvorby prototypu, se po dokončení implementace dostavilo vědomí, že prototyp nepochybně bylo možné naprogramovat lépe a elegantněji. Tyto nedostatky však nejsou z pohledu funkce prototypu v této práci – kterou je demonstrace úspěšné anylýzy a použitelného návrhu – klíčové, což umožňuje výsledný prototyp označit za úspěch.

Krokem následujícím při vývoji softwarového produktu po implementaci je testování. Právě této tematice bude proto věnována následující kapitola.

Kapitola 5

Testování

Důležitou součástí vývoje softwarového produktu je ověření jeho správného fungování za pomoci rozličných testů. Testovat je možné prakticky každý aspekt informačního systému a tímto procesem se zabývá celé odvětví informačních technologií, které je však pro detailní pokrytí v této práci příliš rozsáhlé, pročež budou v této kapitole stručně rozvedeny pouze jeho vybrané oblasti.

Zřetel bude kladen především na uživatelské rozhraní systému, na výkon prototypu a jeho celkovou použitelnost. Budou také vytvořeny automatické testy, které ověří správné fungování prototypu a v budoucnu budou sloužit jako kontrola při refactoringu a dalším rozšiřování programového kódu. Na závěr budou shrnuty výsledky jednotlivých sekcí a zhodnoceno, zda implementace a testování byly úspěšné.

5.1 Úvod k uživatelským testům

Důležitou složkou tvorby uživatelského rozhraní je ověření jeho kvality za pomoci reálných osob. Vývojáři aplikací totiž, byť se mohou sebevíce snažit vžít do role koncového uživatele, vždy uvidí systém jinak, než jak ho bude vnímat skutečný uživatel. Je proto důležité vyhledat skutečné lidi, ideálně co možná nejvíce podobné předpokládaným uživatelům systému (v případě této práce se jedná o rodiče dětí v předškolním věku), a provést s nimi uživatelské testy, které ukáží, jak vnímá UI systému osoba, která ho má doopravdy používat.

V rámci těchto testů bude sledováno, jak jsou zkušební uživatelé (takzvaní „testeři“) schopni systém ovládat, jaké úkony jim činí potíže a zda používají uživatelské rozhraní takovým způsobem, jak si jeho autor představoval.

Dá se očekávat, že tyto testy odhalí nejvýznamnější chyby v návrhu UI a jejich výsledky budou velmi užitečné pro další rozvoj produktu Mojeskolka, jelikož přívětivé a použitelné uživatelské rozhraní je považováno za základ úspěchu celého projektu.

5.1.1 Scénáře

Základem uživatelského testování budou scénáře, tedy sady úkonů, které se snaží simulovat úkony reálného uživatele, který bude aplikaci po jejím nasazení používat. Tyto scénáře vychází z případů užití a zaměřují se na části, které byly vyhodnoceny jako problematické na základě analýzy systému, nebo na základě dřívějšího testování.

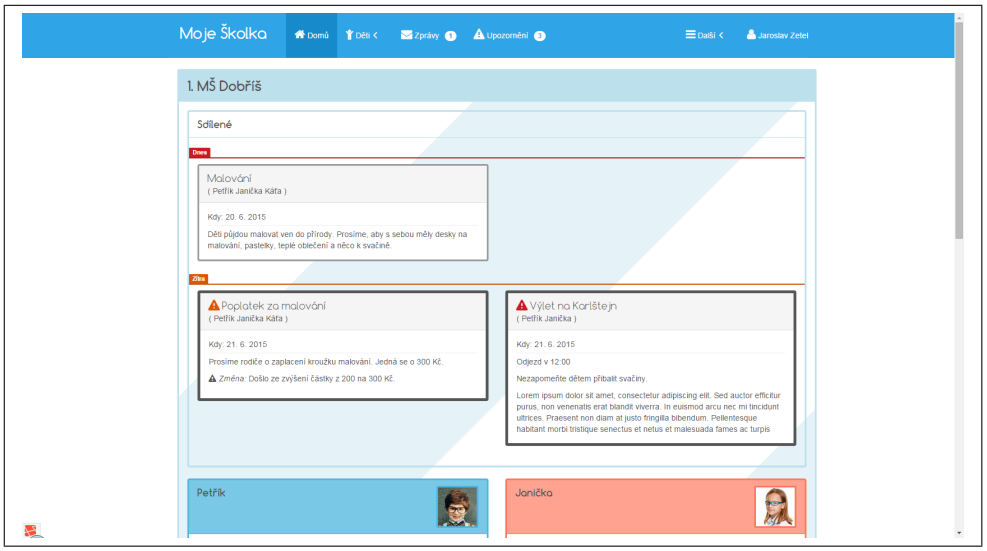
S ohledem na iterativní vývoj byly testy prováděny s postupně upravovanými scénáři, jak byly jednotlivé chyby v návrhu postupně odhalovány a opravovány. Základem je scénář prvního testování, který bude postupně doplňován o testy zaměřené na nově zjištěné problémy a upravován, pokud v něm budou nalezeny nedostatky.

Současná iterace testování navazuje na dřívější poznatky získané v rámci předmětu NUR, proto bude u některých výsledků uvedena informace, zda se podařilo, či nepodařilo vyřešit původní problém objevený při dřívějších uživatelských testech.

Stav systému na počátku testování

Uživatelé se přihlásí do systému, kde na ně bude čekat připravené prostředí naplněné vzorovými daty, která mají simulovat reálné prostředí.

Při testu rodičovského rozhraní jsou uživatelé v roli rodiče, který má tři děti – Petříka, Janičku a Káťu. Tyto děti mají v systému uvedeno několik notifikací, například o výletu na Karlštejn, nebo poplatku za malování. Na tyto předpřipravená data bude odkazováno ve scénářích.



Obrázek 5.1: Hlavní strana při testování prototypu.

Další obrazovky ze systému je možné nalézt v příloze Ukázky vytvořeného prototypu.

Při testu učitelského rozhraní uvidí uživatelé ten samý systém ze strany učitelky, která zadávala notifikace, které testeři viděli v rodičovském rozhraní.

5.2 První uživatelské testování

5.2.1 Scénář

Tento scénář průchodu aplikací tvoří základ všech uživatelských testů provedených v této práci. Snaží se simulovat typické akce uživatele a tak prověřit všechny funkce v systému, přičemž prochází prakticky všechny jeho obrazovky.

Kurzívou uvedený text je sdělován testujícím osobám, zbytek popisuje účel testu a dodatečné informace k němu.

Rodičovské rozhraní

1. Úkolem je přihlásit se do rodičovského rozhraní do 1. MŠ Dobříš (vzorový profil školky) a změnit údaje v uživatelském profilu, čímž se testuje schopnost uživatele zacházet se svým profilem a kontaktními informacemi.

   a. *Přihlaste se do 1. MŠ Dobříš*
   Uživatelský e-mail: test.novak@site.test
   Heslo: Rodic

   b. *Jako první prosíme změňte své kontaktní údaje*

   Kritické body: zda uživatelé postupují přes ikonu uživatele a uživatelské jméno na stránku profilu, nebo zda hledají jinde.

2. Úkolem je otestovat základní orientaci uživatelů v aplikaci. Hledání obecných informací by mělo odpovídat přechodu na hlavní stranu, je otázka, jak uživatelé zareagují a kde budou hledat.

   Podívejte se, co mají vaše děti za aktuální úkoly.

   Kritické body: Vracejí se uživatelé na hlavní stranu? (Při prvotním testování byla tendence jít do „Důležité“, nebo do „Zprávy“).

3. Zaměření na schopnost najít informace o školkách, která se v dřívějších testech ukázala jako problematická.

   Chtěli byste školce, kam chodí Petřík, tedy 1. MŠ Dobříš, napsat mail z vašeho e-mailového účtu a proto byste rádi našli e-mailovou adresu školky (zatím nechcete používat zprávy uvnitř aplikace).

   Kritické části: Pochopí uživatel, že jméno školky je odkaz na její stránku? Kde by se snažil podobné informace hledat?

4. Test zaměřený na upozornění, tedy na notifikace, které vyžadují pozornost uživatele.

   Zajímaly by vás důležité informace, které se týkají Petříka. Chtěli byste tedy najít všechny Petříkovi důležité notifikace.

   Kritické body: zda uživatelé hledají na stránce Petříka, nebo na stránce Upozornění. Pokud na stránce Petříka, pak zda chápou, že vykřičníky značí důležitá upozornění, kterým by měli věnovat pozornost.

5. Zaměření na stránky jednotlivých dětí, hledání specifických informací v aplikaci a práci s jednotlivými notifikacemi. Také je žádoucí zjistit, zda využívají hlavní stranu, nebo spíše stránky jednotlivých dětí.

   Příprava před testem: Pokud uživatelé ještě nejsou na stránce Petříka, pak je poprosit, aby na tuto stránku přešli.

   a. *Najděte všechny Petříkovi notifikace, protože vás zajímají i dřívější notifikace, tedy i ty z minulosti.*

   b. *Kamarádka ve školce si stěžovala, že po vás za docházku do kroužku malování chtějí více peněz. Nejspíše nastala nějaká změna. Vaše sdílná kamarádka zmínila, že v aplikaci jsou změněné notifikace označené ikonou, ale více o nich nevíte.*
   Bylo by dobré, abyste se podívali, o jaké změny se jedná, tedy nalezli tuto notifikaci a otevřeli jste ji.

      Kritické body: Jestli si uživatelé všimnou oranžového vykřičníku a pochopí jeho význam.

   c. *Vzhledem k tomu, že peníze už jste učitelce dali, tak vás tato notifikace o ceně malování dále nezajímá a chtěli byste, aby ji aplikace už dále nezobrazovala.*

   d. *Slyšeli jste, že Petřík příští týden pojede na výlet a že se jedná se o notifikaci, kterou by školka měla ráda potvrzenou. Zjistěte přesné detaily tohoto výletu a potvrďte, že jste si notifikaci přečetli. Zprávu si však chcete ještě později prohlédnout, proto ji nechcete skrývat.*

      Kritické body: Jestli uživatelé naleznou notifikaci, jestli pochopí, že červený vykřičník znamená upozornění a zda pak správně potvrdí přečtení a nedají notifikaci skrýt.

   e. *To je vše ohledně Petříka. Nyní by vás ještě v rychlosti zajímala Janička. Už jste koukali na ostatní děti, ale možná jsou ještě nějaké notifikace, které se týkají pouze Janičky. Najděte a otevřete si notifikaci, která se týká jen Janičky a nikoho jiného.*

      Kritické body: Jestli uživatelé rozumí jménům dětí, která jsou vedená v závorkách pod názvem notifikace a indikují, kterých dětí se notifikace týká.

6. Tento test je zaměřený na interní zprávy v aplikaci. Je třeba ověřit, zda uživatelé chápou význam „bubliny“, tedy kruhové ikony s číslem udávajícím počtem nepřečtených zpráv, a jestli se jim dobře používá rozhraní pro jejich psaní.

    a. *V aplikaci je ještě jedna věc, která by zasluhovala vaši pozornost. Všimli jste si, o co se jedná?* (Jedná se o novou zprávu, na kterou je aplikace upozorňuje číslicí „1“ vedle ikony zpráv v menu)

    b. *Podívejte se na tuto novou zprávu.*

    Kritické body: Zda uživatelé chápou, že nová zpráva je označena tučně a zda jim došlo, že do tabulky je možné klikat.

    c. *Krátce, ale slušně paní učitelce odpovězte.*

    d. *Rádi byste napsali paní Petráskové zprávu, že během hodiny, kterou suplovala, Petrovi někdo sebral pastelky. Jestli by je dotyčnému dítku mohla vzít a vrátit je zpátky Petrovi.*

    Kritické body: Nalezení tlačítka pro psaní nové zprávy. Dále interakce s polem pro vyhledávání adresátů.

Učitelské rozhraní

V rámci testů bylo stručně analyzováno i učitelské rozhraní pomocí jednoduchého scénáře.

1. Přihlásit se jako učitel.

    - Uživatelské jméno: pavla.ucitelkova@site.test
    - Heslo: Ucitel

2. Přidat notifikaci.

    - Třídě II. A poslat notifikaci, že za dva dny se půjde do bazénu

3. Upravit přidanou notifikaci.

4. Poslat zprávu uživateli.

5.2.2 Interakce se zkušebními uživateli

Dle metodik vyučovaných v předmětu NUR je vhodné počítat s přirozeným chováním testujících osob a snažit se je vhodným způsobem usměrnit a motivovat.

Úvod do testování

V rámci úvodu prvního testování, jehož subjekty byly přívětivé, ale nepříliš technicky založené maminky[35] bylo potřeba uživatelky upozornit, aby se chovaly přirozeně a nesnažily se napodobovat chování rodiče, jak si jej představují. Je totiž běžné, že se uživatelé snaží chovat tak, jak si myslí, že by se skutečný uživatel choval, což zkresluje získané informace, jelikož nejlepší přístup k testování, jaký mohou zkušební uživatelé zaujmout, je chování, jako kdyby v aplikaci byly jejich děti a oni se snažily ji skutečně používat.

Jednoduchým důvodem pro toto tvrzení je skutečnost, že právě tyto ženy jsou typickými představiteli budoucích uživatelů aplikace, tudíž se nemusí snažit chovat jako kdokoli jiný.

Dále byly uživatelky upozorněny, že jejich jméno v systému je Tomáš Novák a že mají děti Petříka, Janičku a Káťu. [36]

Po skončení testování

Zároveň byla s testováním spojena snaha věnovat dárek, nebo nabídnout protislužbu, a se zkušebními uživatelkami se na jejich konkrétní podobě předem domluvit. Cílem bylo, aby byly uživatelky motivované a plně se soustředily na testování s vědomím, že z něj budou mít nějaký užitek. V rámci této iniciativy byl tedy zprovozněn jeden starý počítač, nainstalováno několik programů a opraven nefungující přístup do firemního intranetu společnosti British Airways.

5.2.3 První uživatel

Detaily testující osoby

- Jméno: Marika V.
- Věk: 40 let
- Děti: dívka 11 let a chlapec 9 let

Uživatelka má středoškolské vzdělání s maturitou, je řadovám uživatelem počítačového software a nemá žádné zkušenosti s prací v oblasti IT.

5.2.4 Druhý uživatel

Detaily testující osoby

- Jméno: Gábina S.

[35] Jak se však ukázalo, samy sice nepracují v IT, ale zkušenosti s mobilními aplikacemi z hlediska koncového uživatele mají značné.

[36] Zvláště se jménem uživatele se však při testování vyskytly problémy.

- Věk: 35 let
- Děti: dívka 6 let a dívka 3,5 roku

Uživatelka má středoškolské vzdělání s maturitou, je řadovým uživatel počítačového software a taktéž nemá žádné zkušenosti s prací v oblasti IT. (Jedná se o sestru Mariky V.)

5.2.5 Závěry prvního testování

Uživatelky plnily většinu úkolů bez obtíží, ale přesto bylo objeveno mnoho podnětů k tomu, co by bylo vhodné v aplikaci změnit, nebo zlepšit. Tyto závěry shrnují následující seznamy.

Detailní průběh testování s jednotlivými uživateli je možné nalézt v příloze Výsledky uživatelského testování.

Poznatky k jednotlivým bodům scénáře

1. Problémy s uživatelským jménem, které neodpovídá jménu testujícího uživatele. Bylo by vhodné změnit v následujících testech.

   Řešení: Vzorová data v aplikaci budou uzpůsobena na míru testující osobě.

2. Bez problémů, uživatelé chápou, že mají přejít „Domů“ a na stránce se orientují.

3. Stále chybí stránka „Kontakty“, problém s nalezením stránky školky bohužel přetrvává. To, že název školky je odkaz, mají uživatelky problém zaregistrovat, navzdory tomu, že jméno dítěte jako odkaz chápou zcela přirozeně. Důvod pro toto odlišné vnímání nejsou uživatelky schopny vysvětlit.

   Řešení: Přidání stránky kontakty a odkazů na ní ze stránek dětí.

4. Uživatelé sice mají problémy pochopit význam oranžového a červeného vykřičníku (které značí pozměněné notifikace, respektive takové, které vyžadují potvrzení), ale po shlédnutí stránky „Upozornění“ je jim z použitých barev jasné, co které upozornění znamená.

   Řešení: Přidání demonstrativních ukázkek notifikací do „Nápovědy“.

5. Velmi silně využívají stránky jednotlivých dětí, raději, než hlavní stranu. Možná je málo přehledná, nebo příliš dlouhá? Nebo k tomu svádí zadání scénářů, ve kterém je uvedeno, že by uživatelé měli najít informace týkající se jejich dětí? Na tyto otázky bude třeba hledat odpovědi dalším testováním.

a. Problémy s nalezením nezobrazovaných notifikací, v této verzi skrytým v nastavení pod ikonou ozubených kol, což podle všeho není srozumitelné.

   Řešení: Změna designu a přejmenování na „Zobrazit více“.

b. Jak poznat změny v notifikaci?

   Možné řešení: Samotné uživatelky uvádí velmi užitečný návrh na přidání „patch“ sekce, tedy stručného shrnutí změn v notifikaci.

c. Dřívější problémy byly vyřešeny, jak je blíže vysvětleno v následujícím bodu.

d. Uživatelé už nemají žádné problémy potvrdit přečtení a odlišit ho od mechanismu zavírání notifikace, což působilo v předchozím testování problémy.

e. Identifikace, komu notifikace náleží, je bez problému

6. Uživatelé nemají nejmenší problém pochopit význam „bubliny“ (ikony s číslem) umístěné vedle zpráv v menu. Obě ženy dokonce od začátku testování upozorňují na skutečnost, že „jim tam svítí zprávy“ a rády by se podívaly, co jim kdo píše. Zdá se, že podobnost s zobrazováním podobných informací v populárních webových aplikacích je v tomto případě úspěšná a velmi žádoucí vlastnost UI.

   Problémy ale působí řazení podle názvu, nikoli podle data přijetí zprávy, které je však neúmyslným nedostatkem prototypu, nikoli návrhu UI.

   Naopak tučné zvýraznění nepřečtené zprávy uživatelé chápou.

   Tlačítko napsání nové zprávy nemusí být ihned viditelné, možná by bylo vhodné sekci zpráv přizpůsobit více obrazu mailového klienta a přidat vlevo nabídku s „Napsat novou zprávu“, „Přijaté“ a „Odeslané“ – pro zpřehlednění aplikace a aby byly volby lépe viditelné. Nejedná se však o kritický problém.

   Při změně adresátů mají uživatelky tendenci klikat na ikonu šipky, která je navíc velice malá, takže mají problém se na ni strefit. Obě zkušební uživatelky, nezávisle na sobě. Jejich vysvětlení je, že když už tam taková ikona je, tak očekávají, že by právě ji měly využívat. Bohužel je nenapadlo, že celá velká plocha určená pro psaní adresáta má jimi požadovanou funkci. Možné řešení: Odstranění šipky...

Další poznatky

- Volí „Zůstat přihlášen“, stejně tak automaticky volí „Požadovat potvrzení od rodičů“.
- V případě problému s CRFS tokenem bude do budoucna nutné zobrazit smysluplnou chybu.

- Problém se zvýrazněním odkazů – toto bylo diskutováno od počátku tvorby prvního prototypu v předmětu NUR a problém stále přetrvává. Je také otázka, jestli by například jména dětí pod notifikací měla být odkazem, pro přehlednost však spíše ne (odkaz na ně je v menu, i na hlavní straně).

5.2.6 Výsledné změny

Na základě výsledků prvního testování byly v prototypu provedeny změny zaměřené na opravu nalezených chyb a nedostatků.

- Přidání stránky „Kontakty“, která byla již dříve zvažována.
- Nápověda byla rozšířena o chybějící informace. Zároveň bude přidán scénář, který otestuje chování uživatelů, když si nevědí rady (Zda jdou na stránku nápověda a rozumí jí).
- Na stránce zpráv byl v poli pro zadání adresáta zobrazen text, že pole umožňuje tuto volbu a že je možné rovnou psát pro vyhledávání. To by mělo pomoci uživateli využít nabízených funkcí a současně snížit optickou přitažlivost malé bezvýznamné šipky, na kterou se uživatelky snažily klikat.
- Změna „Nastavení“ u seznamu notifikací na „Zobrazit více“ pro lepší pochopení uživatelů. Ikona ozubených kol a obecně pojem „nastavení“ pro uživatele nejsou v této souvislosti srozumitelné.
- Změna menu, přidání sekce „Další“ se stránkami „O aplikaci“, „Nápověda“ a „Kontakty“.

5.3 Druhé uživatelské testování

Cílem druhého testování bylo ověřit závěry získané při prvním testování a objevit další problémy v aplikaci. Zároveň se jednalo o částečnou prezentaci stavu aplikace osobám s prací spojeným, pro kterýžto účel slouží scénář použití aplikace velmi dobře.[37]

Z této okolnosti však nepřímo vyplývá, že osoby testované v druhé iteraci testů jsou odborníci na IT a jejich pohled na softwarové produkty je přirozeně zkreslený jejich povoláním. Ale i za daných podmínek přináší tyto uživatelské testy potenciál pro nové poznatky a užitečné závěry, částečně i proto, že naplno využívat aplikaci Mojeskolka budou pravděpodobně právě

[37] Na významu uživatelského testování by tato skutečnost neměla představovat žádnou újmu za předpokladu, že program uživatelé nikdy neviděli, a tato podmínka byla splněna.

spíše technologičtí nadšenci[38], než úplně běžní uživatelé, a bylo by vhodné na tuto skupinu uživatelů nazapomínat.

5.3.1 Scénáře

Scénář druhého testování vychází z toho předcházejícího, který doplňuje o další testy, specificky zaměřené na nově implementované části. Číselně označené body scénáře jsou shodné s scénářem předchozího testování, rozšiřující bod je na záznamech z testování označen jako X1.

X1. Nápověda. Vložen mezi body 4 a 5 – cílem tohoto scénáře je otestovat, zda uživatelům pomůže nápověda a zda je vůbec napadne takovouto pomoc vyhledat.

   a. *Notfikace jsou různě označené, všimli jste si jejich grafického odlišení?*

   b. *Rádi byste věděli, co přesně znamená oranžový, červený a zelený vykřičník u notifikací. Stejně tak, jako co znamená různé zdůraznění ohraničení. Kde byste podobnou informaci hledali?*

   c. *Vraťte se zpět na hlavní stranu a zkuste stručně popsat stav jednotlivých notifikací.*

   Uživatelé by měli přejít do nápovědy a prohlédnout si ji. Úspěšná snaha podívat se na nápovědu kdykoli dříve během testování je považována za úspěšné splnění bodů a. a b. Cílem bodu c. je ověřit, že uživatel je schopný pochopit prezentované informace a zapamatovat si je.

Změny původních bodů scénáře

Dále je třeba upravit scénář 1., aby odpovídal změnám v připraveném prototypu. Změnou je, že uživatelé mají připravené jméno a e-mail odpovídající realitě, tedy jejich jménu a pracovní doméně. Tím by měla být lépe simulována reálná situace a mělo by být ověřeno, zda problémy s přechodem na stránku profilu byly způsobeny chybným jménem, nebo špatným návhem UI.

5.3.2 První uživatel

Detaily testující osoby

- Jméno: Jiří Hunka
- Věk: 30 let
- Děti: jedno dítě, 4 roky

[38] Máme zde na mysli takzvané „powerusers“, tedy uživatele využívající i pokročilejších možností produktu, jakkoli jsou tyto možnosti v prvotní verzi prototypu spíše upozaděné.

Vysokoškolský učitel, člověk se vzděláním a zkušenostmi v oblasti IT. Jedná se o vedoucího této práce, funkční prototyp však měl možnost spatřit úplně poprvé.

5.3.3 Druhý uživatel

Detaily testující osoby

- Jméno: Michal Valenta
- Věk: 44 let
- Děti: 12 a 14 let

Vysokoškolský učitel, vedoucí katedry softwarového inženýrství FIT ČVUT. Avšak sám upozorňuje, že s používáním moderních webových, natož mobilních aplikací tolik zkušeností nemá.

5.3.4 Třetí uživatel

Detaily testující osoby

- Jméno: Jaroslav Zetel
- Věk: 37 let
- Děti: 3,5 a 8 let

Majitel IT firmy a autor celého konceptu produktu mojeskolka.info, prototyp však vidí poprvé. Pravidelný uživatel mobilních aplikací.

Za zmínku stojí, že toto testování probíhalo na tabletu a v jeho průběhu byly stránky používány i na mobilním telefonu (pro demonstraci aktualizace rodičovského rozrhaní při vkládaní a editaci notifikací v tom učitelském, otevřeném na druhém zařízení). Díky tomu se jedná i o letmý test použitelnosti UI na malých displejích.

5.3.5 Závěr druhého testování

Opět bylo nalezeno nemalé množství drobných chyb a podnětů na zlepšení, opět o něco méně významných, než v minulých testech. Některé jednodušší chyby byly opět opraveny, ale už se jednalo pouze o omezený počet položek.

Taktéž došlo k ověření chování UI na mobilních zařízeních, které potvrdilo korektní rozložení prvků na obrazovce a použitelnost rozhraní na malých displejích a mobilných zařízeních celkově.

Stejně jako u první iterace testů je možné detailní průběh testování s jednotlivými uživateli nalézt v příloze Výsledky uživatelského testování.

Podněty a nápady

Tato upozornění, doporučení a objevené drobné nedostatky by měly být adresovány při dalším rozvoji aplikace a měly by pomoci s jejím zdokonalováním.

- Lépe strukturovat data ve výpisu informací entity, raději jen ve formátu piktogram : text, tedy například „ikona telefonu : telefonní číslo“. Také by bylo vhodné zvážit lepší uzpůsobení UI i pro stolní počítače, což by bylo v rámci responzivního designu možné řešit odlišným rozložením stránky pro mobilní zařízení a pro desktopy. Celkově by bylo vhodné stránku více zkompaktnit a zkrátit.

- Pozor na modrou barvu, která evokuje klikatelný odkaz. Zvláště v souvislosti s barevným schématem, jehož základem je modrá barva.

- Bylo by vhodné přidat tooltip s nápovědou k jednotlivým typům notifikací a ostatních grafických prvků. Celkově se opět o něco více zaměřit na stolní počítače.

- Zkušení uživatelé nápovědu nebudou vyhledávat a přeskočí nabídku jejího zobrazení, případně si ji nepročtou. Vyžadují tedy další vysvětlení o prvcích na obrazovce. Souvisí s předchozím bodem.

- Bylo by vhodné přidat do stručného výpisu notifikace tlačítko „Potvrdit přečtení a zobrazit detail“, které zdůrazní, že na panel dá kliknout, a zvýší interaktivitu.

- Přidat oranžovou ikonu vykřičníku i k sekci „Editováno“ u notifikace.[39]

- Uživatelé jsou zvyklí, že když chtějí další záznamy, scrollují dolů na stránce. Proto by bylo vhodné přidat na konec stránky dítěte tlačítko „Zobrazit další“ pro notifikace ze vzdálenější budoucnosti.

- Všichni testeři při potvrzování o přečtení notifikace klikají na malý checkbox, na který mají problém se trefit (přitom mohou kliknout na celý dlouhý text vedle něj). Žádný z uživatelů na to neupozornil, ani tomu nevěnoval pozornost, ale přesto by stálo za úvahu, zda samotný checbox nezvětšit.

- Možnost přidat vedle textu určujícího děti, kterých se notifikace týká (Petřík, Janička) ještě ikony dítěte příslušné barvy. Je dokázáno, že ikony a barvy jsou pro uživatele pohodlnější, proto by barevné ikony chlapce a dívky mohly být vhodnější. Samotný text však funguje dobře, pročež se nejedná o kritický problém.

[39] Tato úprava byla provedena jako rychlý hotfix po testování s prvním uživatelem a viditelně pomohla druhému a třetímu uživateli pochopit koncept editované notifikace.

- Jistě by bylo vhodné zobrazovat v kalendáři dnešní datum. Mělo by se jednat o jednoduchou úpravu, která uživatelům výrazně pomůže.
- Taktéž by bylo vhodné přidat koncept „vlákna“ zpráv pro lepší orientaci uživatele. Podobná funkce již byla zvažována a potvrdilo se, že by alespoň v nějaké podobě měla být implementována. Jako nejjednodušší se jeví prefixace předmětu (tedy „Re: předmět původní zprávy“), protože na tento systém jsou uživatelé zvyklí z textových zpráv. Možno přidat i šipky pro přechod na předchozí a případně další zprávu v daném „vlákně“ konverzace.
- Do zápatí stránky by bylo vhodné přidat další informace a například odkaz na kontakty, který se tam na mnoha webových stránkách vyskytuje.

Opravy

Provedené opravy jsou již značně povrchní, jedná se pouze o rychlé opravy drobných chyb s velkým dopadem.

- Pořadí zpráv ve výpisu přijatých a odeslaných zpráv bylo změněno a nyní jsou zprávy řazeny korektně podle data přijetí – tedy tak, jak jsou uživatelé zvyklí.
- Okno potvrzující skrytí notifikace překrývalo dropdown navigace, proto bylo změněno pořadí vrstev, ve kterých se tyto prvky nachází.
- Posílání zpráv obsahovalo chybu kvůli změněnému typu metody HTTP užité pro napsání zprávy konkrétnímu uživateli. Tato chyba byla opravena změnou na správnou metodu protokolu HTTP.

5.4 Závěr

Další dvě iterace uživatelských testů opět odhalily mnoho chyb v návrhu UI, ale celkově se jedná o čím dál tím drobnější a méně významné chyby, což je v souladu s očekáváním, že UI se postupně zdokonaluje a trend nacházených chyb je od větších k menším.

Celkově toto testování přineslo mnoho důležitých oprav, například přidání uživateli velmi využívané stránky Kontakty (stránku školky využil bez upozornění na její existenci pouze jeden z nich), výraznou změnu UI pro zobrazování skrytých notifikací, zdokonalení nápovědy a lepší prezentaci informací, například díky opravě řazení zpráv. Zároveň poskytují výsledky i dobrý výchozí bod pro další rozvoj aplikace – stejně, jako byly výsledky testování z předmětu NUR použity na začátku tvorby UI v této práci.

Testování také potvrdilo použitelnost návrhu uživatelského rozhraní, ve kterém již nebyly nalezeny žádné zásadní nedostatky, pročež by nyní měl návrh být na dostatečně vysoké úrovni, aby po zapracování získaných podnětů

mohlo rozhraní být používáno reálnými uživateli. Tím však proces jeho tvorby a zdokonalování zdaleka nekončí – výstupy z uživatelského testování UI se potvrdily být velmi užitečnými a i při dalším vývoji produktu Mojeskolka budou prováděny další testy pro zvýšení jeho kvality.

Testování tedy splnilo svůj účel a jeho výstupem je mnoho cenných podnětů a nápadů, jak produkt dále zdokonalovat.

Zároveň přineslo i některá překvapení z hlediska chování uživatelů, z nichž mezi největší dozajista patří nekonzistence mezi vnímáním panelu dítěte a panelu školky[40] a silná tendence uživatelů klikat na bezvýznamné grafické prvky uvnitř velkých bloků, které jsou celé „klikatelné“ – tedy například tendence klikat na vykřičníky u notifikací nebo na šipku pro rozevření seznamu voleb u přiřazování entit nebo volby adresátů, ačkoli se jedná pouze o doplňující grafické elementy, v případě zmiňované šipky naprosto bezvýznamné.

Závěr této sekce je tak nejen přínosný, ale i zajímavý po stránce získaných poznatků. Celkově tedy můžeme označit uživatelské testy za úspěch.

5.5 Heuristická analýza

Heuristická analýza představuje velice užitečnou metodu, pomocí které může vývojář nebo jiný softwarový specialista analyzovat použitelnost uživatelského rozhraní.

Heuristická analýza je založena na ověření platnosti deseti jednoduchých pravidel, která jsou zaměřená na typické nedostatky v uživatelském rozhraní. Její pomocí je možné objevit velkou část chyb v UI, které by jinak musely být objeveny za pomoci mnoha uživatelských testů. Tato metoda je jednak časově a finančně méně náročná (uživatelské testy zaberou poměrně velké množství času a zdrojů) a současně je schopná nalézt i problémy, které subjekty při uživatelských testech mohou podvědomě vnímat, ale neumí je správně identifikovat nebo popsat, protože postrádají znalosti v oblasti softwarového inženýrství.

Při provádění heuristické analýzy prototypu (i při tvorbě rešerší, uvedených výše) bylo využito článků a stránek zaměřených na tuto tématiku, jmenovitě [44], [45] a například i [46].

5.5.1 Výsledky analýzy

Opět bude ve středu zájmu rodičovské rozhraní, jelikož je složitější a provedené uživatelské testy dokázaly, že s učitelským rozhraním nejsou žádné zásadní sproblémy (i díky použití vhodné šablony, i díky jeho repetetivnosti). Proto bude výstup pro učitelské rozhraní pouze velmi stručný.

[40] Tedy, že každého přirozeně napadne na hlavní straně kliknout na jméno Petříka, pokud chce přejít na jemu příslušící stránku, ale nikoho nenapadne kliknout na jméno školky, ačkoli se jedná o stejný typ panelu a nadpis u dítěte je dokonce méně výrazný.

Některé níže zmíněné problémy již byly objeveny v rámci uživatelského testování, v této sekci jsou pro přehlednost ještě jednou uvedeny v kontextu heuristické analýzy a typicky je k nim přidáno i navrhované řešení.

Rodičovské rozhraní

1. Viditelnost stavu systému

    a. Při načítání detailu notifikace a při opětovném načítání seznamu notifikací (poté, co uživatel v příslušném panelu zvolil zobrazovat všechny notifikace) chybí grafický prvek, který by znázorňoval načítání (například klasický progress bar, nebo alespoň ikona točícího se kolečka).

    b. V aplikaci sice jsou barevné zprávy, které uživatele informují o úspěchu, či neúspěchu dané akce („Notifikace Výlet na Karlštejn byla skryta a již se nebude zobrazovat“), ale chybí možnost si tyto zprávy znovu zobrazit poté, co byly automaticky skyty. Řešení tohoto problému není úplně triviální, pokud by mělo být zachováno minimálnosti UI. Možností by bylo přidat tlačítko „Zobrazit upozornění“, které by se „vysunulo“ po najetí na menu. Na mobilních zařízeních by však muselo být podobné tlačítko viditelné neustále.

2. Shoda mezi systémem a realitou

    a. ID uživatele pro něj pravděpodobně není důležité. Naopak odhaluje zbytečně interní fungování systému.

    b. Stránka profilu uživatele je příliš podobná formuláři pro jeho tvorbu, tedy příliš blízká pohledu, který má systém a nikoli uživatel.

3. Minimální zodpovědnost (a stres)

    a. Chybí možnost vzít zpět skrytí notifikace.

    b. Když uživatel udělá chybu při vyplňování obsahu zprávy, sice se zobrazí korektní chybové hlášení, ale smaže se adresát (pokud byl vyplněn).

4. Shoda s použitou platformou a obecnými standardy

    a. Modrý text, použitý v souladu s barevnou šablonou systému i na některé nadpisy, evokuje modré odkazy v prohlížeči, což je nevhodná asociace. Bude třeba změnit barvu těchto nadpisů a případně naopak některé odkazy, kde to bude vhodné, uvést modrou barvou (většina odkazů už tímto způsobem označena je, pokud se nejedná právě o nadpis).

   b. Problém se slovem „Notifikace“, které nejlépe popisuje tento objekt v systému, ale je otázka, jestli bude pro všechny uživatele srozumitelné. Lepší souhrnné označení pro úkoly, informace, požadavky na rodiče a další oznámení, která může školka předávat rodičům, nebylo nalezeno, bude tedy potřeba časem ověřit, že uživatelé systému tento pojem bez problému chápou a zvykli si na něj.

   c. U zpráv chybí označení „vlákna“ konverzace, tedy něco na způsob „Re:“ v předmětu.

5. Prevence chyb

   a. Některé formuláře umožňují vyplnit chybné údaje.

   b. Tlačítko „Zavřít a nezobrazovat“ nevyžaduje potvrzení této akce. Otázka je, jestli je takovéto potvrzení nutné, pokud by bylo možné ihned vzít akci zpět. Pozitivem však například je, že tlačítko pro skrytí notifikace se ani nezobrazí, dokud není potvrzena.

6. Kouknu a vidím

   a. Chybí historie akcí a stránek, které uživatel vykonal, respektive navštívil. Ta by usnadnila uživateli dokončení činností, které musel přerušit.

   b. Problém v menu s položkou „Další“, která sice obsahuje méně často používané akce, tuto akce však uživatelé typicky potřebují poměrně akutně (nutnost zavolat rychle do školky, nebo okamžitá potřeba nápovědy k aktuální obrazovce). Řešením by mohlo být přidat některé akce do zápatí (u kontaktů a nápovědy to není atypické).

   c. Chybí upozornění na to, že celý panel notifikace je interaktivním prvkem, po jehož stisknutí se otevře detail notifikace. Bylo by vhodné přidat tlačítko „Detaily a potvrzení“, které by bylo zvýrazněno při najetí na notifikaci pro zvýšení interaktivity. Po otevření detailu by bylo možné přídat „Tip“, že celý panel notifikace má stejnou funkci, jako toto tlačítko.

7. Flexibilita a efektivita

   a. Chybí klávesové zkratky pro stolní počítače a stejně tak gesta pro mobilní telefony (například „swipe“ gesto pro skrytí notifikace, nebo podobné funkce).

   b. Chybí možnost ovládat notifikace z přehledu, tedy kupříkladu skrýt některé notifikace, nebo je označit jako důležité pro uživatele. U potvrzování o přečtení se jedná o úmysl, uživatel by měl notifikaci otevřít a pročíst, ale v případě skrývání nepotřebných notifikací tento ovládací prvek chybí.

8. Estetika a minimalita (Klapky na očích)

    a. Zbytečně velké odsazení mezi jednotlivými panely, tedy například u dítěte v rámci školky. Jedná se o problém spíše grafického designu, ale omezuje použitelnost, jelikož se na obrazovku vejde výrazně méně informací.

    b. Stejně tak zbytečně dlouhé a nepříliš dobře čitelné kontaktní informace o školce.

9. Smysluplná chybová hlášení

    a. V případě chyby v programu systém pouze zobrazí generické hlášení „Něco se pokazilo“. Chybové hlášky v případě interních chyb v programu jsou zatím určeny jen pro vývojáře, což je potřeba do budoucna změnit.

    b. Chybové hlášky u formulářů jsou česky, ale označení nevyplněných polí je stále v angličtině (bude potřeba přidat další jazykové soubory).

10. Help a dokumentace

    a. Nápověda je přítomná, ale statická, chybí rychlý dynamický úvod pro nové uživatele (i když, jak říká [46], uživatelé si tyto informace nezapamatují, což zvyšuje význam následujícího bodu).

    b. Chybí tooltipy, tedy popisky UI prvků, které se zobrazí po najetí na daný prvek. Na tento problém se narazilo i při uživatelském testování. Vhodné by také bylo přidat „Tipy“, tedy upozornit uživatele na možné akce, které na dané obrazovce může vykonat.

Učitelské rozhraní

Mnoho problémů rodičovského rozhraní je přítomno i v učitelském, například problémy se zprávami, absence klávesových zkratek, nebo chybějící tooltipy. Níže jsou uvedeny jen problémy unikátní pro učitelské rozhraní.

2. Je otázkou, jak budou učitelé vnímat slovo „Skupiny“ souhrnně označující kroužky, třídy a celou školku (tedy různé „okruhy“ dětí, kterým lze posílat notifikace).

3. - Chybí možnost vzít zpět úpravy entity, bylo by vhodné implementovat historii úprav.
   - Pokud udělá uživatel chybu při tvorbě instance entity, pak je sice zobrazena vhodná chybová hláška a zachována většina vyplněných údajů, ale smažou se vazby na další entity (bude například nutné znovu přiřadit dítěte do skupin).

5. Systém se nedotazuje, zda uživatel opravdu chce smazat instanci entity, pouze potvrdí její smazání. Určitě bude vhodné do budoucna přidat dialog vyžadující potvrzení o smazání příslušné instance, aby se zabránilo nechtěným akcím uživatele.

6. U seznamu instancí entit není uveden název akce, která bude provedena po kliknutí na řádek v tabulce (jedná se o zobrazení detailů nebo editaci příslušné entity).

7. V seznamu uživatelů chybí odkazy na napsání zprávy a pod všemi seznamy entit chybí tlačítko pro přidání nové instance.

10. Chybí nápověda pro učitelské rozhraní.

5.5.2 Závěr

Heuristická analýza pomohla odhalit chyby v aplikaci, které bude vhodné při dalším vývoji přednostně napravit. Většina z těchto chyb již byla odhalena v rámci uživatelského testování, nebo si jich byl autor vědom, ale přinejmenším se jedná o velmi dobré shrnutí a faktem je, že některé nalezené nedostatky nebyly dříve zaznamenány.

5.6 Beta testování

Ačkoli tato fáze započne až po dokončení a odevzdání práce, bylo by v souvislosti s testováním vhodné zmínit, že systém Mojeskolka bude od září roku 2015 nasazen v testovacím režimu do mateřské školky, kde bude používán skutečnými učiteli a rodiči. Od svého nasazení by měl plně nahrazovat nástěnky ve školce, což vyžaduje, aby byl od počátku plně funkční.

Tento beta test má za úkol důkladně prověřit kvality systému před tím, než bude uveden jako hotový produkt na trh, což vyžaduje opět o stupeň vyšší úroveň zpracování, než které bylo dosaženo u výsledného prototypu v této práci. Beta testování a minimálně půl roku další práce na systému nabízejí velké možnosti zdokonalení a vyladění problémů nalezených při jednotlivých testech, i nedostatků, které budou objeveny v rámci beta testování. V této fázi by také měly být velmi užitečné nástroje, které byly zmíněny v sekci Dlouhodobý rozvoj projektu.

Dá se očekávat, že během této doby bude systém výrazně zdokonalen a rozšířen o další funkce, ale základem pro přípravu na beta testování a další rozvoj budou právě testy popsané v této kapitole.

5.7 Automatické testy

Z hlediska dlouhodobého rozvoje systému jsou automatické testy velice důležitým nástrojem, který pomáhá udržovat daný systém v konzistentním stavu,

jelikož zabraňuje zanesení nových chyb v průběhu jeho vývoje, údržby a rozšiřování.

Tato sekce se tedy bude věnovat tvorbě a vlastnostem automatických testů pro systém Mojeskolka. Ačkoli jsou zátěžové testy do jisté míry taktéž automatizované, mají jiný účel a bude jim věnována samostatná sekce níže.

5.7.1 Forma testů

Automatické testy jsou velmi široký pojem, dle zaměření se samotné testy dělí do mnoha kategorií. K těm nejdůležitějším patří takzvané unit testy, integrační testy a funkční testy.

Vzhledem k tomu, že primárním cílem automatických testů vytvářených pro Mojeskolka bylo ověřit fungování systému a připravit podklady pro refactoring a další rozvoj, jako nejvhodnější se jeví funkční testy, které slouží právě pro tyto účely. Funkční testy celé aplikace jsou zároveň částečně i integračními testy, jelikož ověřují schopnost systému, složeného z jednotlivých vrstev, fungovat jako celek.

5.7.2 Zaměření

Otázkou stále zůstává, co všechno pomocí funkčních testů ověřovat a jakým způsobem testy koncipovat. Ideální by bylo co nejvíce se přiblížit tomu, jak aplikaci používají skuteční uživatelé – pak je možné mít jistotu, že když automatické funkční testy proběhnou bez problému, aplikaci budou uživatelé moci používat.

Zde přichází na pomoc framework Laravel, jelikož knihovna Laracasts Integrated [47] , využívajících knihovny Symfony DomCrawler [48] a testovacího frameworku PhpUnit [49], umožňuje simulovat akce uživatele za pomoci příkazů typu „přejdi na domovskou stránku“, „stiskni tlačítko s textem Zprávy“, „ověř, že vidíš text Přijaté zprávy“, „ověř, že vidíš text Pavla Učitelková“ a podobných.

Díky tomu je možnés přesně simulovat průchod uživatele aplikací a vytvořit takové testy, které odpovídají jeho pohledu na systém. Při dostatečném pokrytí všech akcí v aplikaci bude možné prohlásit, že když funkční testy proběhnou v pořádku, pak je systém funkční.[41]

Samozřejmě není možné se spoléhat pouze na automatické testy a zvláště při vývoji nových částí systému bude nutné provádět manuální kontroly, stejně tak tyto testy nemohou pokrýt například chybně vykreslené UI, ale po stránce fungování systému se jedná o kvalitní kontrolu.

[41] Tato shoda slov není dílem náhody.

5.7.3 Implementace automatických testů

Za pomoci výše zmíněné knihovny byly na základě případů užití sestaveny testy, které simulují všechny akce, které uživatel může v rodičovském rozhraní vykonat[42]. Jedná se o čtyřicet různých testů, které vykonávají 273 různých ověření (takzvaných asercí), pokrývají všechny stránky a provádí činnosti od přihlášení do systému, přes zobrazování jednotlivých stránek, až po potvrzení notifikace.

Tyto testy probíhají přímo na testovaném počítači ve speciálním režimu, pro který je vytvořena a spuštěna vlastní SQLite databáze v paměti počítače, která umožňuje rychlé testování a nezávislost automatických testů na stavu perzistentní databáze, pročež je teoreticky možné tyto testy spustit i na běžícím serveru. Tato databáze je naplněna vzorovými daty, určenými právě pro funkční testy, a následně PHP interpret provede příslušnou sadu testů. Výhodou tohoto přístupu je i relativní rychlost, tyto testy proběhnou během 75 sekund.

Opravy použitých knihoven

Zajímavostí je, že při tvorbě automatických testů bylo potřeba opravit, nebo alespoň kompenzovat chyby v použité knihovně Laracasts Integrated Tests.

První chyba byla způsobena těsným spojením těchto testů s frameworkem Laravel, jelikož testy probíhají (pro výrazné zrychlení) v rámci jednoho běhu interpreteru PHP, což však má jako vedlejší efekt zachování nastavených hodnot statických proměnných ve volaných interních třídách frameworku. Tyto proměnné bylo tedy potřeba manuálně resetovat, naštěstí pro takovéto případy existují veřejné funkce daných tříd, jedná se tedy spíše o problém testovací knihovny, nikoli samotného frameworku.

Druhá chyba byla způsobena nekompatibilitou s českými, respektive obecně speciálními znaky ze znakové sady UTF-8. Po krátkém zkoumání byl naštěstí objeven návod [50], který je možné aplikovat na použitou knihovnu, a tak bylo pomocí vhodného přetížení jedné metody dosaženo kompatibility s kódováním UTF-8. Možná by autor knihovny mohl ocenit pull request v jejím GIT repozitáři, ale nejprve bude potřeba provést víc testů pro ověření korektnosti provedených oprav.

5.7.4 Testy modelu prováděné při naplňování databáze

Funkční testy jsou doplněny automatickými testy modelu při vkládání vzorových údajů do databáze, například při spouštění samotných funkčních testů. Vzhledem k tomu, jakým způsobem tento proces funguje ve frameworku Laravel, jistá forma testů totiž probíhá již při vkládání vzorových dat. Jedná se

[42]Toto tvrzení je založeno na pokrytí všech URL, příslušících rodičovskému rozhraní, které jsou definovány v souboru routes.php.

o pozitivní vedlejší efekt skutečnosti, že nástroj Eloquent ORM, užívaný pro práci s databází, při ukládání a zpracovávání dat automaticky využívá metod v modelu systému, kteréžto metody definují vztahy mezi entitami v doméně Mojeskolka. Díky tomu se chyby v základním chování entit, například v definici vazeb mezi nimi, projeví již při vkládání dat do databáze a jedná se tak o první úroveň automatického testování. Ta však sama o sobě nepostačuje pro pokrytí požadavků na automatické testy, pročež byla rozšířena o testy funkční.

5.7.5 Závěr

V rámci této sekce tedy byly vytvořeny automatické testy, které umožňují ověřovat správné fungování systému bez nutnosti kompletních manuálních testů po každé jeho úpravě, které jinak výrazně zdržují vývoj. Je dobré míti na mysli, že se nejedná o všelék – už jen proto, že i samotné testy mohou obsahovat chyby (což je notorický známý a značně nepříjemný problém) – ale i tak se jedná o velmi užitečný nástroj, který jednak šetří čas nutný pro ověřování, že stávající funkcionalita nebyla při dalším vývoji systému poškozena, a zároveň výrazně snižuje potenciální množství chyb v systému.

5.8 Zátěžové testy

Tato sekce má za úkol ověřit schopnost systému obsluhovat současně větší množství uživatelů. Tento požadavek je definován v zadání práce Rozbor zadání a vyplývá i z nefunkčních požadavků na systém, konkrétně požadavku na spolehlivost a dostupnost, který vyžaduje, aby se uživatelé do aplikace mohli připojit a nebránil jim v tom například přehlcený server.

Jak bylo zmíněno výše v sekci Dlouhodobý rozvoj projektu, systém je po stránce architektury připraven na potenciálně velký nápor uživatelů pomocí vhodného rozdělení na logické moduly, které je možné časem separovat do samostatných celků a rozdělit tak zátěž na více serverů, což umožňuje větší škálovatelnost.

Přesto bude vhodné podrobit i počáteční, monolitický systém zátěžovým testům. Jednak proto, aby bylo možné vyhodnotit, zda je systém po stránce výkonu připraven na reálné nasazení; jednak proto, aby byly nalezeny problematické stránky a procesy v systému, na které bude potřeba se zaměřit při optimalizaci systému a zvyšování výkonu.

5.8.1 Obecný úvod

Cílem zátěžových testů je simulovat zátěž serveru, kterou uživatelé vyvolávají při používání systému. Tyto testy nám umožňují najít odpověď na otázku „Kolik uživatelů zvládne server obsloužit za minutu?“, nebo možná srozumitelněji: „Kolik uživatelů může server současně obsluhovat, aby ho tito uživatelé mohli plynule a pohodlně používat?“.

Odpověď na tyto otázky je možné najít za pomoci správně sestavených zátěžových testů s různým počtem simulovaných uživatelů, z jejichž výsledků (doba obsluhy jednoho požadavku, počet vyřízených požadavků za minutu a dalších ukazatelů) budeme moci usoudit, kolik reálných uživatelů by měl být server schopen obsluhovat a při kolika již začne být zahlcen a doba obsluhy se stane neúnosnou.

Je dobré zmínit, že výsledky těchto testů samozřejmě nezáleží jen na zkoumaném systému, ale i na výkonu serveru, na kterém tento systém běží. Proto je níže uveden i výkon námi testovaného serveru.

5.8.2 Výběr vhodného nástroje

Stejně, jako u předchozích kapitol, bude vhodné využít již hotový nástroj, který byl pro tento účel vytvořen. V úvahu připadají nástroje jako Grinder, Pylon a další, ale jako nejvhodnější byl vyhodnocen nástroj jMeter [**?**]. Bylo tak učiněno na základě online doporučení, například [51] a praktického, byť velmi zběžného, vyzkoušení několika alternativ.

Apache jMeter je open-source produkt určený pro tvorbu zátěžových a funkčních testů webových aplikací a záznam jejich výsledků. Poskytuje poměrně široké množství funkcí, ale pro potřeby této prace stačí poměrně malý výřez z nich, konkrétně:

- Možnost poslat HTTP požadavky na server.
- Možnost pracovat s cookies, tedy možnost simulovat přihlášení uživatele.
- Schopnost spustit test paralelně v mnoha vláknech pro simulaci přístupu více uživatelů současně.
- Schopnost zaznamenat výsledky a prezentovat je v podobě grafů, tabulek a dalších výstupů.

To vše v sice komplexním a poněkud nepřehledném, ale přesto použitelném uživatelském rozhraní, které sestavování testů výrazně urychluje.

5.8.3 Forma

Otázkou zůstává, jak testování naplánovat z hlediska výpočetní a síťové vytíženosti počítače, případně počítačů, které budou požadavky generovat a posílat. V úvahu připadají distribuované a nedistribuované testy, tedy testy prováděné současně z více počítačů, respektive testy prováděné z jednoho počítače více paralelně spuštěnými vlákny.

Z úvodu článku [52] a stránky [53] je však poměrně jasné, že distribuované zátěžové testy by měly následovat až v okamžiku, kdy výkon nebo síla internetového připojení testujícího počítače nestačí na zahlcení serveru.

Proto budou nejprve provedeny zátěžové testy z jednoho klientského počítače, který by měl být schopen server dostatečně zahltit, a pouze v případě, že by dříve nestačil výpočetní výkon testujícího počítače, než výkon serveru, by se přistoupilo k distribuovaným zátěžovým testům (to by ale vzhledem k poměrně nízkému výkonu serveru, uvedenému níže, nemělo hrozit). Vzhledem k malému množství přenášených dat se dá očekávat, že úzkým hrdlem nebude síla připojení klientského počítače, ale zátěž na server z hlediska výpočetního výkonu a přístupu do databáze.

5.8.4 Výkon testovaného serveru

Pro prvotní testování byla zvolena velmi slabá konfigurace serveru, která by však měla pro základní testy stačit. Jedná se o virtuální server, který má přiřazen jeden jednojádrový procesor s frekvencí 2.0 GHz a 1 GB paměti.

Výkon testujícího počítače je několikanásobně vyšší (a během testování se jej generováním požadavků nepodaří vytížit ani z deseti procent), pročež není důležité jej uvádět.

5.8.5 Nastavení zátěžových testů

Tvorba scénářů

Podobně jako u uživatelských testů budou ty zátěžové vycházet z hrubých scénářů, jejichž cílem je simulovat reálné chování uživatelů. Proto byly sestaveny tři sady úloh, které prováděly různé skupiny vláken, odpovídající úkonům třech různým uživatelských rolí v systému.

Co se samotného průběhu scénářů týče, jedná se o poměrně jednoduché úkony: všechny tři skupiny uživatelů začnou tím, že se přihlásí a zvolí příslušnou roli (administrátor, učitel, nebo rodič).

- Administrátoři si nechají zobrazit výpis uživatelů, vytvoří uživatele[43], zobrazí si znovu seznam uživatelů a smažou uživatele, kterého dříve vytvořili.
- Učitelé si nechají zobrazit seznam dětí, poté detail konkrétního dítěte, následně přejdou na seznam notifikací, vytvoří novou notifikaci a zobrazí si její detaily. Poté přejdou na stránku zpráv a pošlou zprávu uživateli.
- Simulovaní rodiče, kterých je v testech přítomno nejvíce, si po přihlášení do systému v náhodném pořadí nechávají zobrazit svůj profil, stránky svých dětí, kontakty, důležité notifikace, hlavní stranu a detaily notifikace, kterou následně potvrdí.

[43]Pomocí logické funkce v programu jMeter automaticky ověřujeme, že tato akce byla úspěšná, v tomto případě regulárním výrazem, který kontroluje, zda se vytvořený uživatel nachází ve výpisu uživatelů. Tento postup je aplikován u všech požadavků, které mění persistentní data.

Každé roli přísluší vlastní množina vláken, která vykonávají zmíněné úkony formou zasílání úkonu odpovídajících požadavků na server a analýzou obdržené odpovědi. Počty těchto vláken zmiňuje následující sekce.

Nastavení parametrů

Cílem celého scénáře je imitovat přístup uživatelů, proto je nutné správně nastavit parametry, jako je počet spuštěných vláken, doba mezi jednotlivými požadavky na server (přechody na další stránku) a další.

Prodleva mezi akcemi uživatele

Jedním z parametrů, jehož nastavení stojí za zmínku, je prodleva mezi odesílanými požadavky. Pokud má zátěžový test simulovat reálné uživatele, pak je třeba brát v potaz, že skuteční lidé většinou potřebují nějaký čas mezi jednotlivými akcemi, protože cílem reálných osob není zatěžovat server, ale využít informace, které jim poskytuje. Ve snaze napodobit toto chování bude mezi jednotlivé požadavky vložena náhodná prodlevu s normálním rozložením trvání.

Korektní nastavení těchto prodlev je nepochybně zajímavý problém sám o sobě, ale účely provtních zátěžových testů postačí poměrně hrubý odhad, který byl stanoven na střední hodnotě prodlevy šest sekund s rozptylem jedna a půl sekundy. Nemusí se jednat o dokonalé nastavení, ale pro hrubou simulaci chování uživatelů by se mělo jednat o alespoň řádově přesný odhad realistického nastavení.

Počet simulovaných uživatelů

Důležitou součástí nastavení zátěžových testů je také počet vláken, tedy množství simulovaných uživatelů. Vhodné by bylo najít body několik zlomových bodů, jmenovitě takové, kdy server ještě není zatížený, další, kdy je vytížen optimálně (tedy dosahuje maximálního počtu vyřízených požadavků za minutu) a poslední, kdy již server začíná být přetížený.

Tyto parametry bylo třeba ověřit experimentálně, proto bylo na úvod zátěžového testování provedeno několik hrubých testů, na jejichž základě bylo určeno jako vhodné nastavení patnáct, třicet a padesát rodičů, vykonávajících výše uvedené scénáře. K těmto rodičům je přidána vždy přibližně desetina učitelů a dvacetina administrátorů, kteří do systému, na počátku naplněného vzorovými daty, vkládají další záznamy dle výše uvedeného scénáře.[44]

Sady testů stručně shrnuje následující tabulka.

[44]Kvůli přidávaným notifikacím a zprávám, které jsou určeny pro uživatelský účet, na kterém operují rodičovská vlákna, je možné na výsledcích testů pozorovat postupné zvyšování střední doby obsluhy a snižování propustnosti. Před každým testem však byla databáze uvedena do počátečního stavu, aby se tento efekt nekumuloval.

Tabulka 5.1: Počet simulovaných uživatelů při jednotlivých nastaveních zátěžových testů.

Úroveň zátěže	Počet rodičů	Počet učitelů	Počet administrátorů
Nevytížený server	15	2	1
Průměrná zátěž	30	3	2
Přílišná zátěž	50	5	3

Každé vlákno běží ve dvou po sobě jdoucích cyklech, aby bylo dosaženo méně náhodného vzorku dat.

5.8.6 Výsledky

Po několika zkušebních testech, na základě kterých byly vybrány vhodné parametry pro finální testování a provedeny vhodné úpravy pro zrychlení běhu systémů[45], následovaly ostré sady testů, jejichž rozboru se nyní budeme věnovat.

Sledované parametry

Z výsledků testů jsou významné dva základní parametry:

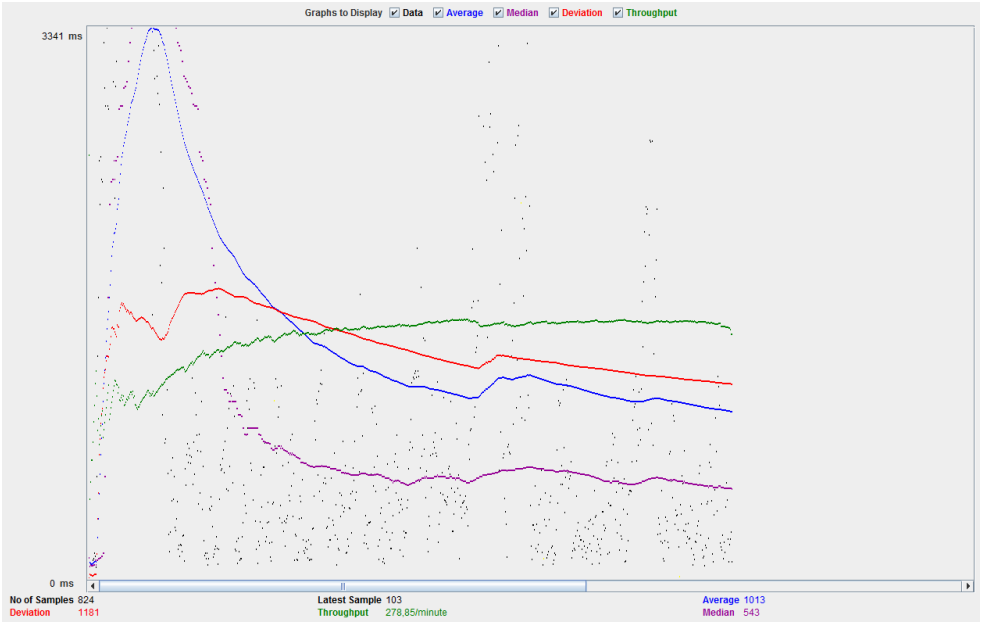
1. Propustnost: Počet vyřízených požadavků za minutu, tedy schopnost serveru obsluhovat více uživatelů současně.

2. Průměrná doba odezvy: Jak dlouho v průměru trvá vyřídit jeden požadavek. Tento parametr je důležitý z hlediska koncového uživatele, jelikož určuje, jak dlouho musí čekat, než se mu stránka zobrazí.

Rozbor výstupu z testů

Interpretace získaných výsledků není na první pohled zcela triviální a forma prezentace v nástroji jMeter situaci nikterak neusnadňuje (ačkoli nástroj jinak funguje velmi dobře, kvalita výstupu není nejlepší a jMeter je za ni online komunitou kritizován). Nejzákladnějším výstupem, který program poskytuje a který zachycuje pro nás důležité údaje, je graf vyřízených požadavků v čase, jehož příklad následuje níže:

[45] Díky vhodnému nastavení frameworku a vypnutí debugovacích modulů určených pro vývojáře byla propustnost zvýšena ze 120 požadavků za minutu na necelých 300.



Obrázek 5.2: Ukázka výstupu z programu Apache jMeter při současném přístupu třiceti rodičů a odpovídajícího počtu učitelů a administrátorů.

Na tomto grafu je možné vidět všechny důležité údaje, tedy průměrnou dosaženou propustnost a průměrnou dobu obsluhy požadavku, i když zanesení těchto dvou údajů do stejného grafu se zcela jinými jednotkami a měřítkem je poněkud nešťastné (to je jeden z důvodů kritiky výstupu z jMeteru, přičemž uvedený obrázek je neupraveným výstupem z tohoto programu).

Graf však věrohodně zachycuje průběh zátěžového testu – na počátku si uživatelé zobrazují stránku pro přihlášení, proto je úplně na začátku testů zátěž nízká. Poté probíhá přihlašování uživatelů a volba rolí, což jsou velmi náročné operace, jedná se tedy o simulaci velké nárazové zátěže a průměrná doba odezvy serveru je v tomto okamžiku nejhorší. Poté se situace uklidní a průměrná doba odezvy postupně klesá, zatímco propustnost roste. Drobné výkyvy jsou způsobeny náhodným načítáním stránek, které zabírají více výkonu, například hlavní strany.

Druhá oblast nárůstu průměrné doby obsluhy odpovídá počátku druhého cyklu (testy byly spuštěny v režimu, kdy automaticky opakují celý test ve dvou po sobě jdoucích cyklech), kde jsou opět prováděny výpočetně náročné operace pro přihlášení uživatelů, volbu role a zobrazení hlavní strany, které po nich následuje. Poté již opět následují dotazy, které server zvládá vyřizovat rychleji a průměrná doba odezvy tak končí na jedné sekundě s tím, že průměrný požadavek (medián) je vyřizován půl sekundy, což lze považovat za dostatečně rychlý výsledek, který by stále měl uživateli poskytovat uspokojivý pocit z používání systému.

Shrnutí výsledků

Ostatní testy měly velice podobný průběh a vlastností, jaké zachycují předchozí grafy, avšak s jinými absolutními hodnotami. Detailní výsledky zátěžových testů je možno nalézt v příloze Výsledky zátěžových testů, následující tabulka uvádí stručný souhrn.

Tabulka 5.2: Výsledky zátěžových testů.

Úroveň zátěže	Propustnost*	Průměrná doba		
		Vyřízení požadavku	Načtení hlavní strany	Odeslání zprávy
Nevytížený server	161	305 ms	405 ms	259 ms
Průměrná zátěž	279	1013 ms	930ms	407 ms
Přílišná zátěž	282	5454 ms	7531 ms	5189 ms

* Jednotkou propustnosti je počet vyřízených požadavků za minutu.

Je vidět, že při 15 rodičích systém ještě není plně vytížen, propustnost je tedy nižší, jelikož nepřichází dost požadavků, které by mohly být zpracovávány. Při 30 rodičích již systém běží na plný výkon a poskytuje maximální propustnost, ale načítání stránek začíná trvat delší dobu. Při 50 rodičích již je server přetížen a průměrná doba nutná pro vyřízení požadavků je neúnosně dlouhá.

5.8.7 Závěr

Výsledky zátěžových testů poukazují na nízkou optimalizaci výkonu systému a soudě dle pomocných nástrojů, například Laravel Debugbar, především v oblasti přístupu do databáze.

Možné zlepšení

Primárním přínosem zátěžových testů je přehled o chování systému v zátěži a informace o problematických stránkách a dotazech, jejichž rozbor umožní systém dále zdokonalovat.

Nejspíše by bylo vhodné se jako první zaměřit na databázi, tedy především optimalizaci dotazů a vhodnou indexaci tabulek. Při důkladnějším zkoumání problematických stránek bylo zjištěno, že výše zmíněný nástroj Laravel Debugbar upozorňuje na příliš velký počet vykonaných dotazů do databáze. Například pro zobrazení hlavní strany je třeba značně přemrštěných sto dotazů. A to se jedná o situaci při „čerstvě naplněné“ databázi, obsahující 5 notifikací, které se uživateli zobrazují. Ale na konci většího automatického testování, kdy se uživateli zobrazuje okolo 30 notifikací, přidaných automatizovanými učitelskými účty, již tento počet vzrostl na čtyřnásobek jen pro načtení hlavní strany, což je neúnosně vysoký počet, který bude nutné výrazně snížit.

Pozitivní však je, že autor této práce si je vědom, kde problém vzniká, a jeho odhalení těmito testy bylo nejdůležitějším krokem k budoucímu zlepšení situace.[46]

Určitě by také bylo vhodné zaměřit se na nastavení samotného databázového stroje, především indexů u jednotlivých tabulek.

Zhodnocení výsledků

Z hlediska absolutních čísel zátěžové testy nedopadly příliš dobře a jsou do jisté míry zklamáním. Určitý podíl na nižší rychlosti má i framework Laravel, který sice velmi usnadnil tvorbu aplikace, ale zároveň se počítá mezi pomalejší webové frameworky, což je cena za rychlost vývoje. Hlavní problém je však nepochybně v přístupu k databázovému stroji.

Na druhou stranu se nejedná o katastrofální závěr, testy naopak otevírají dveře k dalšímu rozvoji a zdokonalení. Zároveň bylo možné podobný výsledek očekávat, jelikož v souladu s výrokem „Premature optimization is the root of all evil.“ byl při implementaci kladen důraz na tvorbu fungujícího prototypu, který bude použitelný po funkční stránce, ale jeho optimalizace proběhne až dodatečně, právě za pomoci rozličných testů, které byly v této práci sestaveny.

Také je dobré připomenout, že systém bude od září roku 2015 nasazen pouze v jedné mateřské školce, kde bude podroben ostrému testování. Systém je i v současném stavu schopný obsluhovat krátkodobě až třicet rodičů, kteří jsou současně aktivní a využívají jeho funkce, což by mělo samo o sobě poskytovat základ, který umožní použití pro jednu malou mateřskou školku. Díky tomu vzniká prostor systém výrazně zrychlit do února 2016, kdy by se případně měl zvyšovat počet školek, které ho využívají. Tudíž zbývají dva měsíce do ostrého nasazení a dalších šest pro optimalizaci výkonu pro více školek, což by měla být postačující doba, zvláště v situaci, kdy existují užitečné závěry z provedených testů.

Z tohoto hlediska jsou tedy zátěžové testy úspěch, jelikož pomocí nich byly odhaleny problémy v aplikaci a je sestavena automatizovaná sada testů, která bude sloužit pro průběžné sledování postupu při zvyšování výkonu systému.

5.9 Závěr

Po skončení provedených prací prošlo uživatelské rozhraní systému Mojeskolka třemi vlnami uživatelských testů, které tvoří dvě plně uzavřené iterace cyklu tvorba UI, testování UI a opravy nalezených chyb. Při posledním, finálním

[46] Jedná se o nesprávné pochopení, jak funguje předpřipravení databázových dotazů ve frameworku Laravel, které má řešit problémy s načítáním dat z databáze v programových cyklech. Právě načítání z databáze v cyklu je příčinou vysokého počtu databázových dotazů, ačkoli by při korektním použití funkcí poskytovaných ORM měl být proveden pouze jeden dotaz na celý cyklus.

testování již nalezené chyby byly relativně drobné a méně závažné, což by mělo potvrzovat poměrně vysokou úroveň výsledného návrhu UI.

Heuristická analýza odhalila některé nedostatky, které však povětšinou spadají do oblastí, na které nebyl při tvorbě UI kladen důraz a počítalo se s jejich nutnými změnami, jejichž potřebu analýza potvrdila.

Vytvořené automatické testy jednak pomohly při ověřování funkcionality prototypu, ale především poskytují velmi důležitý základ pro další zdokonalování systému, které bude nutné jak po stránce zvýšení výkonu, refactoringu kódu a obecně úprav již hotového, tak kvůli přidávání nových funkcí a zdokonalování těch již existujících.

Nejméně uspokojivé výsledky přinesly zátěžové testy, jejichž negativní výsledek však nebyl žádným překvapením, jelikož na optimalizaci výkonu systému nebyl v implementaci kladen důraz a dalo se očekávat, že se tato skutečnost negativně projeví na rychlosti systému.

Celkově je možné říci, že testy splnily svou úlohu především z hlediska budoucího rozvoje, jelikož jsou výchozím bodem pro další zdokonalování systému. Dle očekávání patří nalezené chyby převážně do oblastí, kterým v rámci tvorby prototypu nebyla věnována taková pozornost a které tedy bude nutné zlepšit během dalšího vývoje.

Z hlediska této práce lze výsledky testů označit za úspěch, jelikož potvrzují, že byl vytvořen fungující prototyp, nalezly jeho nedostatky a byly navrženy postupy, jak tyto nedostatky odstranit. Ale skutečný význam provedených testů leží v budoucnosti projektu Mojeskolka.

Závěr

Splnění zadání

1. **Pomocí metod softwarového inženýrství analyzujte požadavky na systém a vyberte vhodné technologie pro jeho realizaci.**

   Na počátku práce byl důkladně zkoumán účel systému Mojeskolka i jeho důležité aspekty a na základě spolupráce se zadavatelem práce byly analyzovány funkční a nefunkční požadavky na systém kladené. Na základě funkčních požadavků byly sestaveny případy užití, které tvořily základ návrhu uživatelského rozhraní, aplikační logiky a datového úložiště v pozdějších fázích práce. Analýza nefunkčních požadavků a obecných cílů projektu naopak ovlivnila tvorbu prototypu, testování a celkový důraz na tvorbu kvalitního uživatelského rozhraní, které bylo vyhodnoceno jako stěžejní bod úspěchu projektu Mojeskolka.

   V průběhu tohoto procesu byly aplikovány metody softwarového inženýrství pro korektní analýzu požadavků, tvorbu případů užití a získávání informací od zadavatele.

   Dále byly postupně v průběhu práce analyzovány a vybírány technologie a nástroje vhodné pro realizaci vytvářeného systému. Bylo postupováno od hrubého výběru možných technologií na začátku práce až po volbu konkrétních frameworků (grafického a aplikačního), doplněných o užitečné knihovny, při tvorbě finálního návrhu systému. Cílem tohoto postupného, iterativního přístupu bylo učinit příslušná rozhodnutí ve vhodné, pokud možno dostatečně pokročilé fázi práce, kdy už jsou známy všechny důležité informace a je proto možné učinit kvalifikovaný výběr.

2. **Navrhněte architekturu systému a způsob jeho rozdělení na rozhraní pro školku a rozhraní pro rodiče, včetně rozboru cho-**

vání systému v extrémní zátěži a možností přípravy na takovouto zátěž.

Nejprve bylo rozhodnuto o rozdělení systému na rodičovské a učitelské přístupové rozhraní, odpovídající potřebám příslušných uživatelských rolí. Z hlediska vnitřního fungování systému však tato rozhraní poskytují pouze dva rozdílné pohledy na ta samá data v systému, což reflektuje situaci v reálném světě.

V průběhu návrhu pak byly zkoumány možnosti struktury a architektury systému z hlediska rozdělení na logické celky a moduly. Byly navrženy možnosti budoucího rozdělení systému, které by umožnilo výrazné rozložení zátěže vyvíjené na systém v případě jeho využívání větším počtem školek. V rámci rozboru chování systému v zátěži byly navrženy, provedeny a vyhodnoceny zátěžové testy, které prověřily výkon vytvořeného prototypu.

3. **Na základě předchozích bodů vypracujte návrh systému.**

   Vzhledem k iterativnímu přístupu k tvorbě této práce bylo nejprve – i pro lepší pochopení logické domény systému Mojeskolka – analyzováno a navrženo uživatelské rozhraní systému, které je rozebráno v následujícím bodě.

   Tento přístup umožnil lepší porozumění systému, čehož bylo využito při následném návrhu aplikační logiky, datového úložiště a celkové architektury systému. Výsledná architektura je silně vázána na zvolený framework, což je však v souladu s rozhodnutím, že podobný nástroj bude použit pro urychlení, zjednodušení, ale i zkvalitnění implementace systému.

   Výstupem návrhu je detailní model entit, databázové schéma i schéma rozdělení modulů, které zachycuje vhodný diagram.

4. **Navrhněte kvalitní uživatelská rozhraní aplikace podle metod návrhu uživatelského rozhraní.**

   Jak bylo zmíněno v předchozím bodě, návrh uživatelského rozhraní systému byl zhotoven před návrhem aplikační logiky systému. Důvodem pro toto rozhodnutí byla mimo jiné snaha omezit vliv znalosti interních vlastností systému na návrh jeho uživatelského rozhraní, před čímž varují metodiky návrhu UI, které byly v průběhu práce opakovaně citovány a aplikovány.

   Proces návrhu UI započal analýzou konkurenčních produktů po stránce UI i dalších zajímavých vlastností a zpracováním výsledků předchozí práce v předmětu NUR, která poskytla uživatelskými testy prověřenou první verzi návrhu uživatelského rozhraní. Na základě těchto podkladů byly navrženy jednotlivé obrazovky v systému a jejich detailní podoba ve formě takzvaných mockupů.

5. **Oba předchozí body vypracujte s ohledem na budoucí rozvoj systému, jeho údržbu a případnou distribuci na další platformy.**

   V souladu se zadáním byly předchozí body vypracovány s důrazem na budoucnost projektu.

   To se podepsalo například na tvorbě responzivního uživatelského rozhraní, které je prvním krokem k rozšíření systému na mobilní platformy. V rámci návrhu aplikační logiky a především při návrhu rozložení systému na logické celky byly učiněny některé důležité závěry, které umožní budoucí rozvoj systému, například co se možnosti jeho škálování týče. Pozornost byla také věnována volbě vhodných nástrojů a postupů, které pomohou se správou a rozvojem systému do budoucna.

   Tento bod velmi výrazně ovlivnil celkové ladění práce a jednotlivá rozhodnutí, která při její tvorbě byla učiněna, jelikož při nich vždy bylo pomýšleno na budoucí dopad a důsledky.

6. **Implementujte prototyp aplikace, který bude demonstrovat její funkcionalitu a možnosti.**

   Na základě předchozích bodů byl vypracován funkční prototyp systému, který potvrdil, že analýza a návrh proběhly úspěšně a že je na jejich základě možné implementovat fungující a použitelný prototyp. Implementace také potvrdila kvalitu návrhu systému a volby použitých technologií, jelikož tvorba prototypu probíhala hladce a bez větších obtíží, což poukazuje na vhodnost učiněných rozhodnutí a voleb.

   Díky použitým knihovnám a frameworkům, ale i velkému úsilí věnovanému implementaci má zhotovený prototyp mnohé vlastnosti komerčního produktu, který by na jeho základě měl vzniknout, což lze považovat za úspěch.

7. **Hotové řešení podrobte vhodným testům a vyhodnoťte výsledky testování.**

   Zhotovený prototyp byl podroben pěti různým testům, které důkladně prověřily jeho vlastnosti.

   Byly provedeny dvě sady uživatelských testů celkově s pěti různými uživateli, které potvrdily použitelnost vytvořeného uživatelského rozhraní i celého prototypu. Bylo ověřeno splnění funkčních požadavků na systém a korektnost realizace navržené aplikační logiky a datového úložiště.

   Na základě těchto testů byly taktéž objeveny drobné nedostatky v návrhu uživatelského rozhraní, které bylo následně možné z velké části opravit. Dle očekávání se množství nalezených chyb zmenšovalo s přibývajícími iteracemi testování, oprav a opětovného návrhu UI. Tyto iterace proběhly celkem již tři, pokud je započítána i práce v předmětu NUR,

ze které bylo vycházeno. Tento počet iterací již postačoval k vytvoření kvalitního a použitelného návrhu UI.

Dále byla provedena heuristická analýza uživatelského rozhraní, která odhalila další možnosti, jak systém v budoucnu zdokonalovat.

Provedeny byly také zátěžové testy systému, které sice objevily nedostatky z hlediska výkonu systému, ale je to právě odhalení těchto nedostatků, které otevírá dveře k jejich budoucímu odstranění a dalšímu zdokonalení systému.

V rámci testování byly také implementovány automatické funkční a integrační testy, které jednak pomohly ověřit správnost fungování prototypu, ale především poslouží při dalším rozvoji systému.

Osobní přínos

Práce na systému Mojeskolka pro mne byla obrovským přínosem z hlediska osobního rozvoje a doplnění znalostí, které bych jako budoucí softwarový inženýr měl ovládat. Studium na Fakultě informačních technologií Českého vysokého učení technického mi poskytlo velmi dobré teoretické základy, avšak stále ve mně přebýval pocit, že mi chybí velké množství praktických znalostí, zvláště v oblasti používání moderních technologií pro tvorbu praktických projektů.

A přesně tyto mezery velmi dobře vyplnila tato diplomová práce, při jejíž tvorbě byla využita široká škála nástrojů, jejichž kvality, ale především hodnota z hlediska úspory času při vývoji a zvýšení kvality výsledného softwarového produktu mne velmi pozitivně překvapily.

Také pro mne bylo překvapením, jak účinné jsou některé techniky návrhu UI, například zpracování rešerší, které poskytly značné množství velmi užitečných informací, nebo tvorba mockupů uživatelského rozhraní, které poskytují stejné benefity z hlediska tvorby UI, jako kvalitní návrh systému při implementaci aplikační logiky. V souvislosti s ní mne potěšilo nalezení vhodného rozdělení systému na separované celky, které by mělo umožňovat velkou škálovatelnost systému do budoucna díky jeho vhodnému návrhu.

Hlavním přínosem tedy nepochybně bylo skutečné upevnění znalostí získaných během dosavadního studia a jejich samostatná aplikace na konkrétní problémy, která potvrdila, že jsem schopen navrhnout a realizovat netriviální softwarový produkt. Jedná se z mého pohledu o velmi vhodné završení studia a splnění značně náročné zkoušky.

Budoucnost projektu

S odevzdáním této práce však projekt Mojeskolka ani zdaleka nekončí. V následujících měsících bude třeba značného úsilí, aby byl systém připraven na ostré nasazení a přístup reálných uživatelů.

Proběhne půlroční betatestování, které nesmlouvavě prověří kvalitu výstupu této práce a možnost jeho reálného uplatnění. Budou muset být opraveny všechny chyby a nedostatky, nalezené i nenalezené při testování provedeném v závěru této práce, a zpracováno velké množství dalších podnětů, které přijdou od samotné školky a rodičů, kteří budou systém používat.

Avšak provedená analýza, přizpůsobivost vytvořeného návrhu, vhodné návrhové i programovací techniky, použití moderních technologií a rozsah provedeného testování – tyto všechny složky představují velmi solidní základ pro budoucnost projektu Mojeskolka a nezbývá než doufat, že těchto základech dokáže být postaven kvalitní a úspěšný produkt.

Literatura

[1] Raible, M.: Comparing JVM Web Frameworks - February 2014 [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.slideshare.net/mraible/comparing-jvm-web-frameworks-february-2014

[2] Bird, J.: The Real Cost of Change in Software Development [on-line]. [cit. 2015-25-04]. Dostupné z: http://java.dzone.com/articles/real-cost-change-software

[3] User Requirements [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.coleyconsulting.co.uk/require.htm

[4] Cockburn, A.: Basic use case template [on-line]. [cit. 2015-25-04]. Dostupné z: http://alistair.cockburn.us/Basic+use+case+template

[5] Johnston, R.: The «include» and «extend» Relationships in Use Case Models [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.karonaconsulting.com/downloads/UseCases_IncludesAndExtends.pdf

[6] Joseph, C.: Programming languages - salaries and demand (October 2014) [on-line]. [cit. 2015-25-04]. Dostupné z: https://gooroo.io/GoorooTHINK/Article/16225/Programming-languages--salaries-and-demand-October-2014/17081

[7] Kolektiv přispěvatelů: Stack Overflow [on-line]. [cit. 2015-25-04]. Dostupné z: http://stackoverflow.com/

[8] Yank, K.: Interview – PHP's Creator, Rasmus Lerdorf [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.sitepoint.com/phps-creator-rasmus-lerdorf/

[9] Montanez, J.: Ruby on Rails vs PHP Comparison [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.comentum.com/ruby-on-rails-vs-php-comparison.html

[10] Tuch, A. N.: Is Beautiful Really Usable? Toward Understanding the Relation Between Usability, Aesthetics, and Affect in HCI [on-line]. [cit. 2015-13-06]. Dostupné z: http://www.mmi-basel.ch/download/pictures/cf/8682k9s5be3eui6n560befvd7uy0uo/2012_tuch_chb.pdf

[11] LePage, P.: Multi-Device Layouts [on-line]. [cit. 2015-14-06]. Dostupné z: https://developers.google.com/web/fundamentals/layouts/?hl=en

[12] Uggedal, E.: Media Queries [on-line]. [cit. 2015-14-06]. Dostupné z: http://mediaqueri.es/popular/

[13] Společnost Google: Android design guidelines [on-line]. [cit. 2015-14-06]. Dostupné z: https://developer.android.com/design/index.html

[14] Park, T.: Bootswatch [on-line]. [cit. 2015-20-04]. Dostupné z: https://bootswatch.com/

[15] Iron Summit Media Strategies: Start Bootstrap [on-line]. [cit. 2015-20-04]. Dostupné z: http://startbootstrap.com/

[16] WrapMarket LLC: WrapBootstrap [on-line]. [cit. 2015-20-04]. Dostupné z: https://wrapbootstrap.com/

[17] Park, T.: Bootswatch: Cerulean [on-line]. [cit. 2015-20-04]. Dostupné z: https://bootswatch.com/cerulean/

[18] Almasaeed Studio: Admin LTE [on-line]. [cit. 2015-20-04]. Dostupné z: https://almsaeedstudio.com/preview

[19] Johnson, J.: Battle of the LESS Mixin Libraries: LESS Elements vs. LESS Hat vs. Bootstrap [on-line]. [cit. 2015-21-04]. Dostupné z: http://designshack.net/articles/css/battle-of-the-less-mixin-libraries-less-elements-vs-less-hat-vs-bootstrap/

[20] Zing design: Less vs Sass? It's time to switch to Sass [on-line]. [cit. 2015-21-04]. Dostupné z: http://www.zingdesign.com/less-vs-sass-its-time-to-switch-to-sass/

[21] Skvorc, B.: Best PHP Frameworks for 2014 [on-line]. [cit. 2015-25-04]. Dostupné z: http://www.sitepoint.com/best-php-frameworks-2014/

[22] Drumelis, V.: 20 Best PHP Frameworks for Developers in 2014 [on-line]. [cit. 2015-24-06]. Dostupné z: http://codegeekz.com/20-best-php-frameworks-developers-august-2014/

[23] Laravel [on-line]. [cit. 2015-24-06]. Dostupné z: http://laravel.com/

[24] Phalcon [on-line]. [cit. 2015-24-06]. Dostupné z: http://phalconphp.com/

[25] Symfony2 [on-line]. [cit. 2015-24-06]. Dostupné z: https://symfony.com/

[26] Yii [on-line]. [cit. 2015-24-06]. Dostupné z: http://www.yiiframework.com/

[27] Davis, M.: The Advantages (And Disadvantages) Of The Yii Framework [online]. [cit. 2015-24-06]. Dostupné z: https://www.futurehosting.com/blog/the-advantages-and-disadvantages-of-the-yii-framework/

[28] Ferrara, A.: Thoughts On PECL Frameworks [online]. [cit. 2015-24-06]. Dostupné z: http://blog.ircmaxell.com/2012/08/thoughts-on-pecl-frameworks.html

[29] Google Analytics [on-line]. [cit. 2015-24-06]. Dostupné z: http://www.google.com/analytics/

[30] Heatmap [on-line]. [cit. 2015-24-06]. Dostupné z: https://heatmap.me

[31] Laravel Debugbar [on-line]. [cit. 2015-24-06]. Dostupné z: https://github.com/barryvdh/laravel-debugbar

[32] Laravel Debugbar [on-line]. [cit. 2015-24-06]. Dostupné z: https://nativecss.com/

[33] PhoneGap [on-line]. [cit. 2015-24-06]. Dostupné z: http://phonegap.com/

[34] Suhosin [on-line]. [cit. 2015-24-06]. Dostupné z: http://suhosin.org/stories/index.html

[35] Netbeans [on-line]. [cit. 2015-25-06]. Dostupné z: https://netbeans.org/

[36] PhpStorm [on-line]. [cit. 2015-25-06]. Dostupné z: https://www.jetbrains.com/phpstorm/

[37] Wamp [on-line]. [cit. 2015-25-06]. Dostupné z: http://www.wampserver.com/en/

[38] Xdebug [on-line]. [cit. 2015-25-06]. Dostupné z: http://xdebug.org/index.php

[39] GIT [on-line]. [cit. 2015-25-06]. Dostupné z: https://git-scm.com/

[40] Laracasts [on-line]. [cit. 2015-25-06]. Dostupné z: https://laracasts.com/

[41] Way, J.: Laravel 5 Fundamentals [online]. [cit. 2015-24-06]. Dostupné z: https://laracasts.com/series/laravel-5-fundamentals

[42] Butler, T.: Model-View-Confusion part 1: The View gets its own data from the Model [online]. [cit. 2015-24-06]. Dostupné z: https://r.je/views-are-not-templates.html

[43] phpDocumentor [on-line]. [cit. 2015-25-06]. Dostupné z: http://www.phpdoc.org/

[44] Nielsen, J.: 10 Usability Heuristics for User Interface Design [online]. [cit. 2015-24-06]. Dostupné z: http://www.nngroup.com/articles/ten-usability-heuristics/

[45] Neil, T.: 6 Tips for a Great Flex UX: Part 5 [online]. [cit. 2015-24-06]. Dostupné z: http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5

[46] Budiu, R.: Memory Recognition and Recall in User Interfaces [online]. [cit. 2015-24-06]. Dostupné z: http://www.nngroup.com/articles/recognition-and-recall/

[47] Integrated [on-line]. [cit. 2015-25-06]. Dostupné z: https://github.com/laracasts/integrated

[48] DomCrawler [on-line]. [cit. 2015-25-06]. Dostupné z: http://symfony.com/doc/current/components/dom_crawler.html

[49] PHPUnit [on-line]. [cit. 2015-25-06]. Dostupné z: https://phpunit.de/

[50] Hohner, J.: Crawling UTF-8 pages with Symfony2 DomCrawler [online]. [cit. 2015-24-06]. Dostupné z: https://janhohner.de/crawling-utf-8-pages-with-symfony2-domcrawler/

[51] Performing a Stress Test on Web Application? [on-line]. [cit. 2015-25-06]. Dostupné z: http://stackoverflow.com/questions/7492/performing-a-stress-test-on-web-application

[52] Wiki, J.: How to do remote testing the proper way? [online]. [cit. 2015-24-06]. Dostupné z: http://wiki.apache.org/jmeter/JMeterFAQ

[53] Remote Testing [on-line]. [cit. 2015-25-06]. Dostupné z: http://jmeter.apache.org/usermanual/remote-test.html

[54] Nielsen, J.: How to Conduct a Heuristic Evaluation [on-line]. [cit. 2015-13-06]. Dostupné z: http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[55] Ihsanullah Khan on LinkedIn [on-line]. [cit. 2015-24-06]. Dostupné z: https://sa.linkedin.com/in/ihsanullah12

[56] P?ihlá?ení do systému Edookit [on-line]. [cit. 2015-24-06]. Dostupné z: https://zszdirec.edookit.net/user/login?backlink=boidf

[57] Dokumentace k systému ?kola OnLine [on-line]. [cit. 2015-24-06]. Dostupné z: https://aplikace.skolaonline.cz/dokumentace/KS/zakovska/web/index.html

Příloha **A**

Seznam použitých zkratek

FIT ČVUT Fakulta informačních technologií Českého vysokého učení technického v Praze, tedy akademický subjekt, v rámci kterého byla tato práce zadána a vypracována.

IDE Integrated (interactive) Development Environment je vývojové prostředí pro tvorbu aplikací. Cílem tohoto nástroje je usnadnit práci programátora, typicky se skládá z textového editoru pro psaní kódu a podpůrných nástrojů, jako je nástroj pro ladění kódu, nápoveda, našeptávání při psaní kódu a dalších funkcí. Kvalitní IDE velmi usnadňuje vývoj v daném programovacím jazyce.

UI User Interface, tedy uživatelské rozhraní. Jedná se o souhrn prvků, pomocí kterých uživatel může komunikovat se systémem a ovládat jej. V souvislosti se systémem Mojeskolka tento pojem označuje především grafické uživatelské rozhraní, případně rozšířené o doprovodné zvukové efekty. Tedy typicky obrazovky obsahující jednak prezentované informace (notifikace, informace o dítěti, kontakt na školku,...) a jednak ovládací prvky (formuláře, tlačítka a další).

GUI Grafické UI.

XML Extensible Markup Language, široce používaný jazyk pro záznam strukturovaných dat.

ORM Objektově Relační Mapování. Jedná se o vrstvu mezi aplikační logikou a datovým úložištěm, která usnadňuje přístup z objektově orientovaného systému k relační databázi.

DDoS Distributed Denial of Service, tedy organizovaný útok na webový informační systém. Jeho podstatou je snaha zahltit daný server velkým množstvím požadavků zasílaných z mnoha počítačů současně.

XSS Cross Site Scripting, tedy vložení škodlivého, typicky javascriptového kódu na napadené stránky.

CSRF Cross Site Request Rorgery, tedy kybernetický útok založený na podstrčení adresy se škodlivým efektem na systém uživateli, který má do tohoto systému přístup. Může se jednat například o odkaz, který systém interně používá pro smazání důležitých záznamů. Útočník využívá toho, že uživatel je v systému v daném okamžiku přihlášen (za pomoci souboru cookie uloženého v prohlížeči), tudíž má autorizaci vykonat i úkony, které samotný útočník provést nemůže.

SQL Structured Query Language, dotazovací jazyk používaný pro přístup k relačním databázím.

IoC Inversion of Control, tedy návrhový vzor, který obrací typickou tvorbu softwarového produktu tím, že uživatele-programátora nechá doplnit pouze části kódu specifické pro danou doménu, ale jinak za něj obstarává veškerou ostatní logiku.

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `docs` ..... dokumentace projektu
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

Výsledky rešerší konkurenčních systémů

C.1 Úvod

C.1.1 Výběr aplikací pro rešerše

V rámci NUR jsme analyzovali kvalitně zpracované aplikace, v rámci této práce se proto zaměříme na druhou zmiňovanou skupinu vhodných kandidátů – konkurenční aplikace.

Jako vhodný počáteční bod pro výběr se jeví informační systémy pro školy a školky vyvíjené v České republice, jelikož se dá očekávat, že právě takové aplikace budou případní zákazníci a potenciální uživatelé již nyní používat. Důvodů pro tento předpoklad je několik – vzdělávací systémy v různých zemích mají různá specifika, na která budou typicky cílit i systémy v nich vyvíjené, pročež bude českým vzdělávacím institucím nejbližší český informační systém, a zároveň u českých produktů odpadají problémy s lokalizací a dostupností technické podpory.

Ukazuje se však, že problémem není ani tolik sehnat aplikaci, která by byla zaměřená na vzdělávací systém, problémem je získat do takové aplikace přístup. V Čechách vyvíjené a používané produkty, například Škola OnLine, Edookit, Bakaláři a další, které by mohly představovat systém principiálně podobný zaměření Mojeskolka, totiž neumožňují přístup pro veřejnost. Provozovatelé těchto systémů sice nabízí školám možnost zdarma zažádat o vyzkoušení jejich produktu, pokud zájemci vyplní příslušný formulář na stránkách provozovatele, jiným způsobem však není možné do těchto systémů získat přístup. Autor této práce se pokusil kontaktovat některé provozovatele s žádostí o přístup do jimi vyvíjené aplikace za účelem tvorby rešerší v rámci této práce, ale nedostalo se mu pozitivní odpovědi.

To bohužel výrazně omezuje možnosti tvorby rešerší na příslušné aplikace, ale nikoli úplně. K aplikacím Edookit a Škola OnLine je totiž možné sehnat

materiály, které obsah aplikace velmi detailně dokumentují – jmenovitě uživatelské příručky a instruktážní videa prezentující aplikaci do detailu po stránce GUI, doplněné o poměrně rozsáhlý popis aplikace pro zákazníka, poskytující informace po stránce funkcionality. A právě tyto dvě charakteristiky – GUI a funkcionalita – budou důležité při hodnocení stránek (které bude rozebráno v následující sekci), což nám umožňuje na tyto systémy provést rešerši s dostatečnou vypovídající hodnotou. Z množiny kandidátů tedy budeme hodnotit produkt Škola OnLine a Edookit.

Vedle českých systémů by bylo vhodné pro srovnání vybrat také zástupce systémů vyvíjených v zahraničí. Z několika variant byl nakonec vybrán produkt OpenSIS, který jako jediný ze zvažovaných alternativ umožňuje přístup do systému i pro neregistrované uživatele, jelikož se jedná o open source projekt. Tento program navíc má on-line testovací režim, který umožňuje prohlédnout si demonstrativní verzi aplikace. Kdyby tento režim nepostačoval, pro účely dalšího testování byl tento systém nainstalován a zprovozněn na autorově počítači[47]. Díky tomu bylo zaručeno, že pro rešerši systému budou dostupná všechna relevantní data.

Všechny zkoumané systémy mají na trhu poměrně silnou pozici – Škola OnLine je údajně nejrozšířenější český školní informační systém a se 45 000 uživateli v roce 2009 je možné tomu i věřit, OpenSIS zas nejrozšířenější open-source systém tohoto typu na světě – nebo velký potenciál: Edookit je vyvíjen silnou skupinou a působí mimořádně kvalitně (jak potvrdí i výsledky jeho rešerše, pokud si můžeme dovolit předběhnout). Zároveň však všechny systémy pokrývají školky jen velice okrajově, pokud vůbec. Snaha nalézt systém zaměřený na školky alespoň do stejné míry, jako na vyšší stupně vzdělávání, když už ne primárně, se totiž setkala s neúspěchem. Pokud takovýto software existuje, je velmi málo rozšířený, alespoň v České republice. Proto budeme zkoumat systémy nejbližší našemu zaměření a hledat u nich poučení, které by mělo být i tak přenositelné na náš produkt.

C.1.2 Hodnocení stránek

V rámci jednotlivých rešerší budeme hodnotit kvalitu stránek a snažit se najít pozitivní i negativní prvky v jejich návrhu uživatelského rozhraní. Tím nemáme na mysli zkoumání grafického návrhu (ačkoli zastaralost grafického designu za problém považovat lze, jak bylo zmíněno výše), ale máme tím na mysli uživatelskou přívětivost, tedy jednoduchost, či složitost kroků nutných pro provedení úkonů v aplikaci, přehlednost uživatelského rozhraní a další. Pokusíme se provést i Nielsonovu heuristickou analýzu, která nám poskytne informace o případných chybách v návrhu UI těchto aplikací. Na tuto část hodnocení bude mít vliv omezený přístup k GUI aplikací (pokud u daného produktu musíme vycházet pouze z materiálů k němu), avšak dle [54] je možné

[47] V průběhu instalace se objevily problémy. jejichž oficiální doporučované řešení („vypněte zaznamenávání chyb v PHP“), autora této práce poněkud pobavilo.

systém vyhodnotit i na základě pouhých obrazovek z něj sejmutých, což i takového heuristické analýze přidává na váze. V případě toho, že některé body nepůjde ověřit, nebudeme tyto body rozvádět a budeme uvádět vždy pouze nalezené chyby. O heuristické analýze je možno více detailů nalézt v sekci Heuristická analýza.

Je dobré mít na mysli, že primárním cílem řešerší je nalézt indicie, které by nám mohly pomoci při tvorbě naší vlastní aplikace, proto se zaměříme právě na takovéto prvky. Mezi ně spadají i další informace, které nám rešerše poskytují a přesahují návrh GUI – můžeme se inspirovat i z hlediska zajímavé funkcionality, kterou proto taktéž zahrneme do hodnocení stránek, a můžeme částečně zkoumat i technologickou stránku jednotlivých aplikací, která nám může pomoci udělat závěry o naší volbě použitých technologií a případně tuto volbu přehodnotit. Zkoumat budeme jednak jazyk použitý pro implementaci aplikační logiky, jednak formu, jakou je produkt poskytován – buďto se jedná o lokální instalaci, kterou musí zákazník sám spravovat, nebo se jedná o SaaS, o jejíž zprávu se stará poskytovatel produktu (SaaS formu lze vnímat jako pozitivum a i je tak autory produktů prezentována, jelikož malý zákazník typicky nechce řešit problémy související s instalací a provozem aplikačního serveru). Technické aspekty budeme posuzovat podle dostupné technické dokumentace a podle informací, které aplikace vystavují okolnímu světu.

Následuje stručné shrnutí jednotlivých aspektů hodnocení:

C.1.2.1 Funkcionalita

- Zajímavé funkce – funkce systému, ze kterých bychom si mohli vzít inspiraci.

C.1.2.2 UI

- Pozitiva – zajímavé prvky UI, které bychom mohli využít i v návrhu systému Mojeskolka.
- Negativa – nevhodné elementy UI, který zkoumanému systému škodí a kterých bychom se měli vyvarovat.
- Závěr heuristické analýzy – v rámci heuristické analýzy budeme hledat chyby ve zkoumaném UI. Jednak proto, abychom měli představu o slabinách konkurenčního softwaru, a jednak proto, abychom se z nich do budoucna, pokud možno, mohli poučit.

C.1.2.3 Technologie

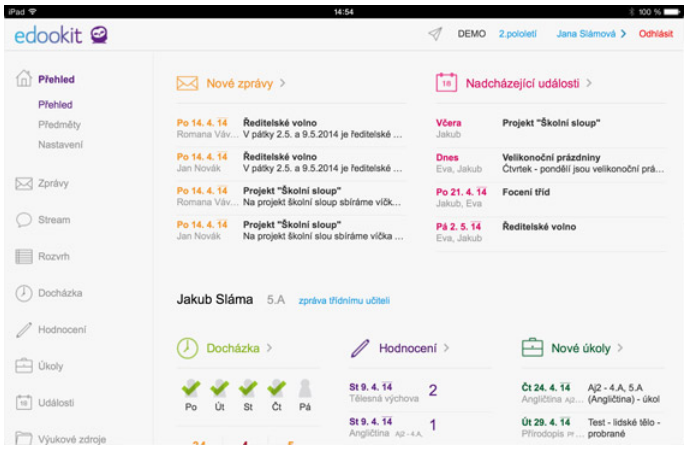
- Jazyk použitý pro realizaci aplikační logiky projektu – jednak pro zhodnocení naší vlastní volby, jednak kvůli představě o stavu trhu by mělo být záhodno zjistit, v jakém programovacím jazyce byly zkoumané (a poměrně úspěšné) systémy vytvořeny.

- SaaS – to, jakým způsobem je produkt distribuován a spravován má pro nás podobný poučný význam, podobně, jako předchozí bod.

C.2 Edookit

Webová aplikace vyvíjená Masarykovou univerzitou v Brně ve spolupráci s institucemi Jihomoravské inovační centrum a Centrum vzdělávání, výzkumu a inovací pro ICT, což samo o sobě napovídá o vysoké úrovni a kvalitě produktu. Edookit je moderní systém s kvalitním designem, rozsáhlou funkcionalitou a velice silným zázemím, pročež se jedná se o konkurenci výjimečných kvalit, která je zároveň se svým zaměřením i na školky ze zkoumaných produktů nejblíže systému Mojeskolka. Proto budeme systému Edookit věnovat nejvíce pozornosti a nejdetailnější rozbor funkcionality i UI.

Na druhou stranu je tento systém, i přes své deklarované zaměření na všechny stupně vzdělávání od školek až po univerzit, určen primárně pro základní a střední školy, na což budeme v této sekci poměrně často narážet. Tato skutečnost umožňuje projektu Mojeskolka jít jinou cestou, než Edookit, a nemuset se tolik bát přímé konkurenceschopnosti ve srovnání s tímto, alespoň jak je na základě dokumentace, videí a obrazů uživatelského rozhraní možné soudit, velmi kvalitním produktem.



Obrázek C.1: Obrázek uživatelského rozhraní aplikace Edookit[48].

Velká část funkcionality se tedy míjí s náplní projektu Mojeskolka – e-learning, on-line testy, digitální výukové materiály, třídní kniha, žákovská kniha se známkami, ani další funkce nejsou určeny pro mateřské školky. Stejně tak velká část prezentované funkcionality produktu již byla v rámci Mojeskolka identifikována a je s ní počítáno v rámci prvotní verze – události (v našem

[48]Prosíme o omluvení nižší kvality obrázků UI. Jedná se o nejvyšší kvalitu, ve které je autoři aplikace poskytují.

systému realizovány rámci notifikací), komunikace prostřednictvím osobních zpráv, správa kontaktních informací entit, evidence žáků a rodičů; nebo je plánována do budoucna – sdílení souborů, kalendář událostí, správa financí, komentáře k příspěvkům, suplování (v Mojeskolka se do budoucna v rámci stránky třídy počítá i s informací o tom, kdo třídu aktuálně učí a v jaké učebně se třída momentálně nachází) a další. Tento překryv plánované funkcionality s již hotovým produktem opět poskytuje indicie o korektnosti analýzu systému, kterou jsme již provedli.

Co se nefunkčních vlastností systému týče, Edookit láká na SaaS [49], nízké náklady (600 Kč měsíčně pro větší mateřské školky), moderní zpracování aplikací a kontakt s rodiči. Ale přesně tyto charakteristiky chce mít i projekt Mojeskolka, tudíž v této oblasti se taktéž zdá, že jsme na správné cestě.

C.2.1 Zajímavé funkce

- **Customizace:** Možnost upravit produkt na míru zákazníkovi – obecný trend v IT, takzvaná „Mass customization“, tedy masové nasazování produktu, který může být dle požadavků zákazníka za příplatu obohacen či upraven dle jeho individuálních potřeb.

- **Docházka žáků:** Vyplňování a evidence docházky žáků je v rámci základních škol (zvláště jejich vyšších ročníků) pro rodiče i učitele důležitá. Vyvstává však otázka, zda pro školky má smysl. Nejspíše pouze částečně při omlouvání žáka z výuky. Rodič by měl mít možnost snadno předem, případně zpětně (pokud to školka vyžaduje) omluvit své dítě z výuky, především kvůli informacím pro samotnou školku. Z počátku by pro tyto účely měly sloužit zprávy v aplikaci, časem by mohla být vytvořena samostatná stránka s formulářem pro omluvení dítěte z výuky.

- **Náplň výuky:** Prezentace osnov předmětu a obsahu probírané látky má význam spíše v rámci základních škol, kde může být rodičům přesně prezentováno, co jsou děti vyučovány, ale i v našem případě by stálo za úvahu, jestli by rodiče nezajímala náplň výukových i odpočinkových aktivit dětí ve školce, například používané metody, denní rozpis činností a další informace. Mohlo by se do budoucna uvažovat o samostatné sekci pro tyto informace jednak v rámci školky, jednak v rámci tříd.

- **Hodnocení práce žáka:** Malé děti není dle moderních výzkumů dobré hodnotit stejným způsobem, jako jsou hodnoceni starší studenti, ale přesto by mohlo být vhodné mít možnost alespoň slovně rodiče informovat o tom, jak se žákům ve školce daří. Pro tyto účely možná lépe

[49] napříč stránkami jsou jednotlivé výhody SaaS rozepsány a zopakovány více, jak deseti různými způsoby, je však na místě podotknout, že pro zákazníka znějí všechny lákavě.

poslouží osobní kontakt s rodičem, pročež tato funkcionalita není vnímána jako tak podstatná, ale další možnost, jak komunikovat s rodiči, by měla být pozitivní.

- **Používání systému učitelem v průběhu výuky:** Opět se jedná o funkcionalitu okrajovou pro mateřské školky, ale stálo by za úvahu zjistit od školek, jestli by učitelům pomohlo, kdyby mohli systém užívat i během výuky, typicky pro vkládání informací, kontaktování rodičů, nebo další úkony.

- **Rozvrh:** Školky typicky mají rozvrh orientovaný spíše ve stylu „kdy se děti zaobírají jakou činností“, nežli „kde přesně se jí zaobírají“ a proto nemusí mít školky vypracovaný tak detailní a přesný rozvrh, jako základní školy. Ale i za těchto podmínek může alespoň hrubý rozvrh poskytovat důležité informace kupříkladu pro rodiče, který chce vyzvednout své dítě předčasně ze školky a chtěl by vědět, jestli ho má hledat ve třídě, na zahradě, nebo jestli je zrovna třída na procházce v parku. Alespoň částečně implementovaný rozvrh je tedy do budoucna nutný, pokud chceme v systému poskytovat výše zmiňované informace o aktuální poloze třídy. Je otázkou, zda by všechny školky chtěly podobnou funkcionalitu využívat, může se proto jednat o volitelnou součást, kterou nemusí vyplňovat a používat.

- **Oblíbené položky:** V aplikaci Edookit je možné přistupovat k často užívaným položkám pomocí ikony hvězdy, typicky užívané pro tyto účely. To může umožnit customizaci aplikace samotným uživatelem a usnadnit orientaci v systému.

- **Integrace e-mailu:** Zprávy by měly nahrazovat e-mail, ale zároveň mnozí uživatelé mohou preferovat možnost vidět zprávy i ve svém e-mailovém klientovi. Nemuselo by tedy být na škodu umožnit uživateli nechat si zprávy přeposílat i na e-mail. Evidence zpráv v systému by probíhala na základě e-mailových zpráv příchozích na adresu uživatele na doméně Mojeskolka, kteréžto zprávy elektronické pošty by byly automaticky systémem evidovány a převedeny do interní podoby zpráv v příslušné aplikaci, čímž by se oba přístupy ke zprávám provázaly.

- **Stránka s historií činnosti uživatele:** Zmíněno už na základě předchozích rešerší, ale v systému Edookit je tato funkcionalita implementována ještě o krok lépe. V sekci „Spolupráce“ máme jednak „Výpis aktualit“, chválený i níže, ale také stránku „Historie vkládání“, kde může uživatel vidět svou poslední aktivitu a například se moci snadno vrátit k chybě, kterou udělal při vkládání notifikace, nebo si ujasnit, jakou práci už dnes udělal a čím by měl pokračovat. Velmi dobré je zde i využití ikon pro odlišení zpráv, domácích úkolů a dalších typů jednotlivých záznamů.

C.2.2 Uživatelské rozhraní

Jak bylo zmíněno v úvodu této sekce, aplikace byla hodnocena na základě videí a sejmutých obrazovek prezentovaných provozovatelem systému edookit. Tento omezený zdroj informací však nezabránil v poměrně detailnímu rozboru systému a provedení až na některé body zcela plnohodnotné heuristické analýzy.

Klady

- **Ikony:** Dobře zpracované, snadno odlišitelné a reprezentativní ikony v souladu se zvyklostmi (žák – hlava se studentskou čepicí, oblíbené položky – hvězda, ...).
- **Grafický design:** Celkově moderní design , čisté uživatelské rozhraní, pohledné a příjemné.
- **Responzivní design:** Elegantní a velmi přehledné řešení jednak pro velmi malá zařízení (velice velké a málo početné ovládací prvky), tak pro tablety (podpora takzvaného „landscape módu“ včetně typických prvků jako je menu vlevo, důsledkem je jasná viditelnost, že tato verze aplikace je skutečně určená pro zařízení velikosti tabletů a má adekvátně upravené uživatelské rozhraní), tak i pro stolní počítače (velké množství ovládacích prvků, které jsou však užitečné a na velké obrazovce neruší).
- **Využití barev:** Různé bloky informací barevně odlišeny – docházka, hodnocení, ..., ale i odhlášení ze systému, nebo datum zpráv a událostí dnes, včera a zítra – takováto označení by bylo možné využít u detailů zpráv a především u notifikací. Do jisté míry barevná konzistence – modrá barva značí odkazy a přechod na jiné stránky, velice užitečné.
- **Zvýraznění přechodů na jinou stránku:** Jak bylo zmíněno v předchozím bodě, velice dobře jsou odlišeny akce, které uživatele přenesou na jinou stranu. Typicky se jedná o odlišení příslušného textu a/nebo šipky modrou barvou.
- **Velice hezky řešené zprávy:** Stránka přijatých zpráv je realizována velice přehledně a elegantně.
- **Způsob otevírání detailů:** Na mobilních telefonech velká oblast pro otevření detailů daného bloku (zprávy, notifikace, nadcházející události, ...), protože je potřeba dbát na možnost přehmatů a viditelnost na malých displejích. Na větších zařízeních vždy pomocí odkazu „Detail“ a/nebo šipky. Zajímavé a přehledné.

- **Našeptávání:** Minimalizuje psaní textu, prakticky všechna pole našeptávají možný obsah (jména žáků, tříd, předmětů, ...), velice užitečné na mobilních zařízeních.

- **Název „Výpis aktualit“:** alternativní název pro sekci nazývanou „Důležité“, možnost inspirovat se vzhledem.

- **Zvuky:** V mobilních verzích jsou použity zvuky pro lepší orientaci uživatele. [50]

Zápory

Většina problémů objevena v rámci heuristické analýzy a proto jsou uvedeny až níže v příslušné sekci. Zde jsou uvedeny spíše drobnosti a postřehy nesouvisející s heuristickou analýzou.

- **Jazyková nekonzistence:** Může se jednat pouze o chybu v demo verzi aplikace na stránkách, ale na obrazovce „Zápis učiva v hodině“ se mísí český text s anglickým („View selected“, „Filter selected“). Jedná se spíše o drobnost, ale ve finálním produktu by to nepochybně byla značná chyba.

- **Problém při zvětšování stránky:** Při zvětšení stránky kvůli čitelnosti textu (nebo obrázků...) překrývají nadpisy sekcí obrázky před nimi. Jedná se o drobnost, ale nikoli pozitivní.

- **Nevhodné kopírování formuláře:** Je vidět, že stejný typ formuláře pro filtrování (umístěný v záhlaví stránek) je použit pro zprávy, i pro docházku. V případě docházky však kolonka „časové rozmezí“ může dávat menší smysl. Bylo by třeba vidět skutečnou funkci v aplikaci, v tomto případě se ani nemusí přímo jednat o chybu, ale obecně je dobré mít na mysli, že programátoři mají tendenci znovu používat stejný kód i v místech, kde by mohlo být lepší ho modifikovat.[51]

C.2.3 Heuristická analýza

2. Nevhodná ikona: ikona hodnocení (tužka) užívána spíše pro editaci stránky, jinak jsou ikony kvalitní.

   Zobrazení interního ID: V historii vkládání je uvedeno i ID zprávy, zadání, nebo jiného záznamu. Jedná se o interní systémovou informaci a dá se očekávat, že v reálném systému tato čísla (id položky) půjdou do

[50] Možnost použití zvuků je jedním z důvodů, proč se tato celá sekce jmenuje pouze „Uživatelské rozhraní“, a nikoli „Grafické uživatelské rozhraní“.

[51] Ačkoli se nemusí jednat o chybu v Edookitu, závěr použitelný pro zlepšení aplikace Mojeskolka je pro nás důležitější, než samotný problém v jeho původním kontextu.

řádu statisíců, což zpochybňuje vhodnost prezentace této informace uživateli. Jedná se však taktéž o velmi drobný nedostatek, který může mít své opodstatnění.

4. Barevná nekonzistence: U zpráv dnes a včera odlišeno barevně, u jiných datovaných záznamů však nikoliv. Dokonce zelená barva jednou značí „dnes“ a jednou „včera“. Avšak skutečnost, že dny v týdnu jsou v rámci rozvrhu a kalendáře barevně odlišeny jinými klíči, než dnes a zítra, lze hodnotit spíše jako pozitivum, nežli jako chybu.

   Nekonzistence je také u ikon v menu napříč různými demonstrativními obrázky uživatelského rozhraní pro různé uživatelské role (učitelé, rodiče a vedení školy). Může se jednat jen o rozdílná vydání aplikace a problém vzniklý postupným doplňovaním obrázků na stránky, ale jinak by v rámci celé aplikace nejspíše bylo vhodné dodržovat stejné barevné, či černobílé schéma.

6. V kalendáři není označen dnešní den v sloupci, pouze vlevo ve výčtu dní. Chybí tedy zvýraznění dnešního dne o v hlavním okně kalendáře.

7. Není možnost užívat klávesové zkratky, nebo není snadná možnost nechat si je zobrazit. Vzhledem k cílení na mobilní aplikace se nejedná o tak velký prohřešek.

8. Na stránce „Zápis učiva v hodině“ je velké množství prezentovaných informací. Takovéto množství je na jednu stranu potřebné, na druhou stranu už na hranici s uživatelskou (ne)přívětivostí. Také velké množství barev může být problém, ale pro zkušeného uživatele dost možná spíše přínos. Pro posouzení těchto rozhodnutí bylo by třeba sledovat systém v praxi.

   V kalendáři jsou pod časy začátků a konců hodin občas (ale ne vždy) vidět také celé hodiny a půl hodiny (tedy 10:00, 13:00 a další). Jedná se o drobnou chybu, ale je v rozporu s čistotou uživatelského rozhraní.

10. Na některých stránkách chybí nápověda.

Body 1, 5, 9 a částečně 10 jsou těžko posouditelné vzhledem k založení analýzy pouze na obrazovkách a videích ze systému. Body 2, 3, 4 (po stránce dodržování konvencí), 6 a 8 jsou až na drobné výjimky obecně velmi dobře zpracovány a splněny.

C.2.4 Technologické aspekty

Pokud ze zdroje technologických informací o Edookitu [55]můžeme korektně usuzovat, pak byl systém Edookit vytvořen v programovacím jazyce PHP. Tento zdroj uvádí použití frameworku Zend, skript na stránce pro přihlášení

do systému [56] obsahuje kód z frameworku Nette, tudíž není zcela jisté, který framework byl pro tvorbu systému Edookit použit, ale s největší pravděpodobností byl pro tvorbu systému použit programovací jazyk PHP.

Ze samotných stránek můžeme poznat, že Edookit používá knihovnu modernizr.js pro zlepšení responzivních vlastnostní stránek (kterážto knihovna byla zmiňována v předmětu NUR), javascriptovou knihovnu jQuery (která je však velmi široce užívána), a grafický framework Bootstrap, který je plánováno využít i pro Mojeskolka. A stejně tak jako většina moderních aplikací používá Edookit službu Google Analytics, která bude využita při sběru údajů i o systému Mojeskolka.

C.2.5 Závěr

Dle očekávání se systém Edookit ukázal být na vysoké úrovni jak po stránce funkcionality, tak po stránce návrhu UI, ve kterém byly nalezeny pouze drobné nedostatky. Detailní zpracování jeho rešerše přineslo mnoho zajímavých podnětů a informací.

Zároveň se potvrdilo celkově silné zaměření systému na vyšší stupně vzdělání, než mateřské školky, což je z hlediska konkurenceschopnosti projektu Mojeskolka pozitivní informace.

C.3 Škola OnLine

Škola OnLine je starší webový informační systém (nejstarší verze fungovala minimálně od roku 2002, současná verze, soudě dle uživatelské dokumentace, je v provozu od roku 2009) určený primárně pro základní a střední školy. Za zmínku stojí, že systém je údajně „nejrozšířenější“, pravděpodobně v rámci České republiky, což zvyšuje šance, že se s tímto systémem setkali budoucí potenciální uživatelé systému Mojeskolka.

V mnoha ohledech je systém Škola OnLine velice podobný systému Edookit. Z hlediska zaměření na vyšší stupně vzdělávání, než mateřské školky, z hlediska realizace formou SaaS a především z hlediska poskytované funkcionality. Dá se hovořit prakticky o absolutním překryvu deklarované funkcionality obou systémů pro všechny skupiny uživatelů (školní matrika, hodnocení, komunikace pomocí zpráv, události ve škole a další), ačkoli jejich realizace nemusí být v jednotlivých systémech na stejné úrovni. Podobné je i opakované zmiňování všech parametrů vyplývajících z SaaS, které bylo možné vidět u Edookitu, i zmiňování modernosti aplikace. Výjimek ve zmiňované funkcionalitě je minimum, za zmínku stojí vazba na další školní aplikace (Škola OnLine uvádí e-learningové LMS a Microsoft Office 365), nebo správa obědů.

Avšak po stránce UI a grafického zpracování aplikace se zdá, že mezi aplikacemi bude poměrně zásadní rozdíl. Vyhodnocení této domněnky však poskytne až závěr sekce Škola OnLine.

Obrázek C.2: Obrázek uživatelského rozhraní aplikace Škola OnLine[52].

Celý systém Škola OnLine se sestává z několika modulů určených pro různé uživatelské role. Vzhledem k tomu, že primárním zaměřením této kapitoly je tvorba rodičovského rozhraní systému Mojeskolka, v rámci rešerše analyzujeme aplikaci Žákovská, určenou v rámci Školy OnLine pro kontakt s rodiči.

C.3.1 Zajímavé funkce

Vzhledem ke značnému překryvu funkcionality se systémem Edookit budou v této sekci zajímavé funkce systému Škola OnLine, které již byly zmíněny v sekci Edookit výše, uvedené jen nadpisem a detaily budou rozvedeny pouze u unikátních funkcí.

- **Docházka žáků**
- **Rozvrh**
- **Možnost sledovat přečtení zprávy:** Tato možnost byla a je zvažována i u systému Mojeskolka. Jedná se o mechanismus, kterým odesilatel zprávy může vidět, zda si ji příjemce již přečetl, podobně, jako například na některých sociálních sítích. Tento přístup má své výhody i nevýhody, ale jistě je vhodné jej nezavrhovat.
- **Správa obědů:** Tato funkcionalita je taktéž do budoucna zvažována a její implementace v systému Škola OnLine poukazuje na vhodnost této myšlenky. V aplikaci Škola OnLine se jedná o vazbu na systému WebKredit firmy Anete a iStravné firmy VIS.

[52]Stejně jako u aplikace Edookit prosíme o omluvení nižší kvality obrázků UI – obrázky vyšší kvality autoři aplikace neposkytují.

- **Záznam přihlášení do aplikace:** Z hlediska bezpečnosti je vhodné uživateli ukazovat, z jakých adres a zařízení se naposledy přihlásil, aby byl schopný odhalit neoprávněný přístup k jeho účtu.

- **Export do dalších aplikací:** V rámci škol a dalších institucí jsou typicky používány kancelářské programy pro evidenci a sdílení informací, jmenovitě Microsoft Excel, Open Office Calc a další. Snaha usnadnit zákazníkovi přechod od starého způsobu fungování do nového výrazně usnadní rozšíření a adopci systému. Za úvahu by mohl stát i import z dokumentů, které školky v současnosti užívají (pokud uchovávají relevantní dokumenty v elektronické formě), ale vzhledem k jeho rozmanitosti a nejednotnému formátu bude nejspíše potřeba podobné operace provádět manuálně.

C.3.2 Uživatelské rozhraní

Při analýze UI budeme vycházet z uživatelské příručky k produktu [57], která obsahuje detailní obrázky dokumentující používání aplikace Žákovská.

Je otázkou, jestli nápověda odpovídá současnému stavu systému (jelikož obrazovky v ní působí poněkud archaicky), ale pokud by dokumentace a nápověda k systému byla poskytována k úplně jiné verzi aplikace, než jaká je momentálně používána, nebo pokud byly v UI provedeny rozsáhlé změny bez aktualizace příslušných sekcí nápovědy, jednalo by se samo o sobě o značnou chybu na straně provozovatele služby Škola OnLine. Proto budeme předpokládat, že uživatelská příručka reflektuje současný stav systému.

Vzhledem k tomu, že v příručce nejsou vždy zachyceny celé obrazovky z aplikace, ale často pouze jejich výřezy, možnost analýzy je vůči Edookitu poněkud omezena, ale stále je možné extrahovat z této dokumentace nemálo užitečných informací.

Klady

- **Snaha o detailní nápovědu:** Pro názornější ukázku používání aplikace a jejích funkcí je k systému dodávána uživatelská příručka, která obsahuje i videa zachycující typické složitější úkony v aplikaci. Instruktážní videa pro nápovědu nezkušeným uživatelům jsou zvažována i pro systém Mojeskolka, zde můžeme vidět, že konkurenční software je užívá také. Nejedná se o jakkoli převratný prvek v dokumentaci kvalitních aplikací a samotná videa jsou značně zastaralá, přesto se jedná o pozitivum systému Škola OnLine.

Zápory

- **Grafický design:** Ačkoli se nejedná o samotný návrh UI, grafický design ovlivňuje spokojenost uživatele s produktem, jak bylo rozebráno výše v sekci 2.1. A grafický návrh aplikace Škola OnLine působí značně zastarale.

- **Responzivní design:** Dostupné stránky systému nemají správně udělaný design pro různé velikosti displeje a zobrazení, například u nápovědy přestane být viditelné menu při zvětšování stránky. Je dobré myslet i na uživatele, kteří si potřebují stránku zvětšit (tento problém se objevil i u prototypů Mojeskolka, nejedná se tedy pouze o kritiku systému Škola OnLine).

- **Barvy:** Téměř nulové užití barev pro odlišení informací podtrhuje špatnou strukturu prezentovaných informací zmiňovanou níže v subsekci věnované heuristické analýze. Například po úspěšné registraci nikde není jediná komponenta obarvena zelenou barvou, uživatel musí celý text přečíst, aby tušil, jestli byla operace úspěšná, nebo nikoliv. Nebo zobrazování absolvovaného testu zelenou barvou, i když student neprospěl. Jedinou pozitivní výjimkou je úspěšné příhlášení k absolventské práci, kde zelená barva textu je použita.

- **Nutné přechody na další stránku pro provední akce:** (nebo také nízká interaktivita systému) – téměř pro každou funkci je třeba přecházet na novou stránku, jak bylo zvykem u starších systému, zvláště vlivem nerozšířenosti technologie AJAX. V moderních aplikacích, za kterou se Škola OnLine prohlašuje, však takovýto přístup není nejšťastnější a užívání na mobilních zařízeních touto vlastností trpí ještě o něco více.

C.3.2.1 Heuristická analýza

Vzhledem k rozsahu nalezených nesrovnalostí a tomu, že mnoho problémů je natolik zásadních, že by spadaly do více kategorií a zabíraly by tak příliš velký duplicitní objem textu, bylo třeba některé informace zestručnit a uvádět je pouze u jednoho bodu, kterého se týkají, ačkoli správně dle metodiky pana Nielsena bychom je měli více rozvádět. Námi zvolený přístup je v našem případě přijatelný, jelikož se nejedná o analýzu našeho systému, na jejímž základě bychom chtěli zkoumaný systém zdokonalovat, ale pouze o zkoumání konkurenčního softwaru.

2. Struktura prezentovaných informací: Na stránce „Ověření pinu“ jsou informace ve formuláři strukturovány velice nepřehledně a je nesnadné se v prezentovaném shluku textu vyznat. Stejně tak zobrazování sekce „Děti ve škole“ pro studenta je poněkud pochybné. Dále třídění jednotlivých kategorií v nastavení do tabulky se sloupci pojmenovanými „Kategorie“, „Název“, (namísto třídění do přehledných sekcí dle „Kategorie“)

není úplně šťastné[53]. Taktéž na stránce exportu do aplikací Microsoft chybí nadpis a oddělení sekce pro výběr sloupců tabulky, které mají být exportovány, což zkušenému uživateli nemusí činit problémy, ale pro nového uživatele to může být matoucí. Struktura a přehlednost této stránky rozhodně není v souladu s metodikami návrhu UI.

Chybí označení pozice v systému: Vzhledem k absenci menu, nebo alespoň označení volby, kterou jsme se na současnou stránku dostali, není možné zjistit, kde se uživatel v systému právě nachází.

Nevhodné názvy: Rozvrh je označen jako „Kalendář“, na jiném místě jako „Týden“ a samotný kalendář jako „Výběr data“. Stejně tak výsledky zkoušení jsou označovány písmenem „Z“ a výukové zdroje písmenem „V“, což poněkud vyvolává zmatek mezi „Z“ – „Výsledky“ a „V“ – „Zdroje“.

4. Chybí tlačítko „Domů“: Ani na webových stránkách, ani v systému není tlačítko „domů“, na které jsou uživatelé zvyklí. Takto budou nuceni hledat cestu zpět na úvodní stranu systému. Pro tyto účely slouží v aplikaci Žákovská ikona kalendáře, což však rozhodně není ihned srozumitelné.

   Chybí menu: V aplikaci Žákovská chybí klasické menu, vše je v panelech funkcí po levé straně, které však nejsou dobře organizovány, například nastavení školního roku, které určuje data zobrazovaná na hlavní straně, je vedle změny hesla v obecném bloku „Nastavení“.

   Nekonzistentní názvy: „Kalendář žáka/studenta“ odkazuje na stejnou stránku, jako „Týden“, což je ve skutečnosti rozvrh (tento nedostatek je rozveden v bodu 2).

5. Tlačítko „Odeslat omluvenku“ vedle „Zobrazit odeslané zprávy“ působí jako dvě nesouvisející tlačítka patřící k bodu 2, ale po bližším zkoumání je možné usoudit, že druhé zmiňované má pravděpodobně funkci „Zpět“. Krom toho, že je toto rozdělení matoucí, pak tlačítko pro návrat zpět je výrazně větší, než tlačítko pro odeslání, ačkoli je to méně častá volba (větší je prostě proto, že obsahuje delší text).

6. Špatně označené odkazy: Uživatel je nucen hledat a pamatovat si stránku, která vede „Domů“, stejně tak hledat funkce v jejich velmi špatně tříděném seznamu po levé straně (jak bylo zmíněno u bodu 4, jelikož je tento návrh v rozporu hned s několika pravidly, opakuje se a je dosti zásadní).

   Malá tlačítka: Tlačítka pro zobrazení detailů jsou sice jednotně označena třemi tečkami, ale mají minimální velikost a nejsou nikterak odlišená od zbytku stránky, ačkoli se často jedná o hlavní funkcionalitu na ní zobrazovanou. Lépe by bylo doplnit text („Detail“, „Zobrazit více“, . . .) a barevně, nebo alespoň velikostí odlišit od zbytku stránky. Dokonce

[53] Význam checkboxu ve sloupci „Pro školní rok“ zůstává záhadou.

i přečtení zprávy, která má v systému nahrazovat funkce elektronické počty, je možné jen po kliknutí na tlačítko o velikosti v řádu desetin centimetru.

7. Dle prohlížených videí se nezdá, že by systém umožňoval používání klávesových zkratek, ani v uživatelské příručce nejsou zmíněny. U systému Eddokit zaměřeného na mobilní platformy má tento nedostatek poněkud menší váhu, ale u systému, který na mobilních zařízeních podle všeho ani nelze používat, je tento nešvar výraznější.

8. Nesouvisející prvky umístěny vedle sebe: Tlačítko „Zrušit vazbu na Live ID“ je pod formulářem „Nastavení vlastního hesla“ (který sám je typicky spíše nazýván „Změna hesla“) hned vedle tlačítka „Uložit“ (tedy uložit nové heslo). Stejně tak informace o roli přihlášeného uživatele je prezentována vedle názvu stránky v sekci určené pro nadpis stránky, což vytváří situace, kdy máme pod sebou informaci „Student: Petr Novák“ a „Žák/Student: Petr Novák“ a těžko soudit, co je role, co je informace týkající se rozvrhu (nadepsaného „Kalendář“) a co je název stránky.

Opět prakticky není možné hodnotit body 1 a 9, jelikož ani z obrazovek systému, ani z videí nejsou posouditelné.

C.3.3 Technologické aspekty

Soudě dle hlavičky stránek obsahující kód s meta informacemi „Microsoft Visual Studio .NET 7.1“ a „CODE_LANGUAGE“ rovno „C#“, dle užité přípony .aspx (užívané pro stránky vyvíjené ve frameworku ASP.NET, tedy webovém frameworku platformy .Net), dle možnosti přihlašovat se přes učet Microsoft i dle cookie souboru se jménem ASP.NET_SessionId se dá poměrně s jistotou usuzovat, že systém Škola OnLine je vybudovaný na platformě .NET. Firma to sama nepřímo potvrzuje, když říká, že „výhradní postavení mezi aplikovanými technologiemi zaujímá společnost Microsoft“.

Stejně jako v případě služby Edookit je Škola OnLine poskytována jako SaaS za měsíční paušál.

Další relevantní detaily o použitých technologiích se nepodařilo získat, ale dostupné informace jsou postačující.

C.3.4 Závěr

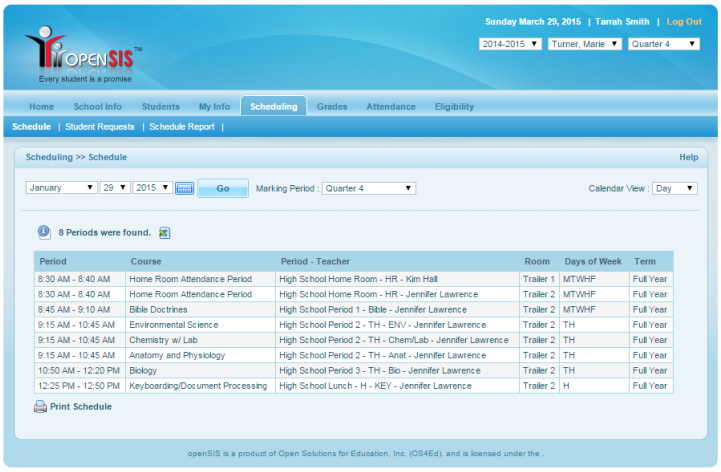
Potvrdilo se, že po stránce základní funkcionality i zaměření na vyšší stupně vzdělávání je systém Škola OnLine poměrně blízký systému Edookit, avšak z hlediska uživatelského rozhraní je rozdíl mezi těmito systémy naprosto markantní. Množství chyb nalezených v uživatelském rozhraní Škola OnLine – a to pouze na základě jeho dokumentace – je překvapivě vysoké, až alarmující. V kombinaci s postarším grafickým designem je celkové vyznění UI aplikace

poměrně negativní. Zároveň lze považovat za poněkud nešťastnou realizaci vybraných funkcí systému a některá rozhodnutí jeho autorů. Za zmínku stojí například potřeba doobjednávat do aplikace další funkcionalitu (a to poměrně základní, jako „Aktuální přehled hodnocení ve všech předmětech“) a platit za ni navíc, což se nejeví jako příliš uživatelsky přívětivé (i když se tento nedostatek netýká přímo UI).

V součtu s funkcionalitou ještě více zaměřenou pouze na vyšší stupně vzdělávání se by se nemělo jednat o přímou konkurenci produktu Mojeskolka.

C.4 OpenSIS

Poslední zevrubně zkoumaný systém je zahraniční open-source studijní informační systém OpenSIS. Taktéž cílí na vyšší stupně vzdělávání, taktéž poskytuje podobnou funkcionalitu, jako předchozí dva zkoumané systémy (evidence studentů, správa kontaktních informací, tvorba rozvrhu, záznam studijních výsledků i docházky a další) a sdílí i další podobné charakteristiky. Z unikátních funkcí stojí za zmínku důraz na práci s exportem informací ze systému, především možnost tvorby a správy šablon exportovaných dokumentů, kterážto funkcionalita je však pro školky dost možná nadbytečná, a možnost nechat systém vygenerovat vysvědčení a další dokumenty za minulý (půl)rok, což se však také míjí s zaměřením na školky.



Obrázek C.3: Obrázek uživatelského rozhraní aplikace OpenSIS

Po stránce vzhledu GUI se zdá, že se OpenSIS nachází někde mezi prvním a druhým zkoumaným systémem, avšak kvalitu UI samozřejmě bude třeba detailněji analyzovat.

Podobně jako Škola OnLine má OpenSIS více modulů, z nichž hlavním pro nás opět bude rozhraní určené pro rodiče.

C.4.1 Zajímavé funkce

Podobně jako u Škola OnLine uvedeme u funkcí duplicitních s ostatními systémy pouze název.

- **Docházka žáků**
- **Rozvrh**
- **Export do dalších aplikací:** Stejně jako u Škola OnLine je možno stahovat data ve formátu pro tabulkové editory, v OpenSIS je tato funkcionalita ještě rozšířena.

C.4.2 Uživatelské rozhraní

Klady

- **Domovská stránka s „notifikacemi“:** jedná se spíše o kladnou informaci, nežli objektivně pozitivní prvek, ale ihned po vstupu do systému je uživateli prezentováno aktuální dění a různá upozornění týkající se jeho dítěte, což je v souladu s představou o hlavní straně systému Mojeskolka.
- **Barvy:** Užívány výrazně méně, než by mohly být, ale kupříkladu alespoň odhlášení ze systému je barevně odlišeno, stejně tak jako v systému Edookit, což je velmi vhodný prvek UI zabraňující nechtěným volbám.
- **Vyplňování formulářů:** Editace pomocí kliknutí na vypisovaný text může dobře sloužit zkušenému uživateli, zobrazení (a částečný výběr) adresy na mapě a další detaily působí pozitivně.

Zápory

- **Stránky opensis.com:** Stránky projektu OpenSIS působí velmi zmateně a není snadné se v nich orientovat. Na některé součásti stránek dokonce vůbec nejsou uvedeny odkazy – například přístup do demo verze systému je nutné vyhledat pomocí internetového vyhledávače, protože najít na něj odkaz na stránkách se autorovi této práce nepovedlo ani po čtvrt hodině hledání, ačkoli při snaze tento odkaz nalézt věděl jistě, že taková stránka existuje (již ji dříve nalezl, právě pomocí internetového vyhledávání).
- **Nekompletní zápatí:** V zápatí jsou uvedeny základní informace ohledně autorských práv, ale chybí typ licence, ačkoli je pro něj očividně ponecháno místo („. . . and is licensed under the .“). Jedná se o detail, ale nepůsobí příliš profesionálně.

- **Responzivní design:** Lépe řečeno jeho naprostá absence omezuje použitelnost produktu.

C.4.3 Heuristická analýza

2. Nevýrazné označení aktiální pozice uživatele v systému: Ačkoli je v menu a v záložkách označeno, kde se uživatel nachází, jedná se pouze o velmi drobný rozdíl ve váze fontu, kterýžto rozdíl pro snadnou orientaci nepostačuje.

   Reprezentace rozvrhu: Velice nevhodná reprezentace rozvrhu studenta. Místo klasického školního rozvrhu podle dní a hodin ve sloupcích a řádcích je použita tabulka, která je řazená podle času začátku hodiny, zatímco dny, kdy se hodina koná (a do kterých je takováto tabulka běžně roztříděna), popisuje sloupec „Days of Week“ s obsahem ve formátu „MTWHF“ (podle všeho se jedná o první písmena anglických názvů dní v týdnu, ale žádná nápověda, nebo tooltip tento sloupec nijak blíže nevysvětluje). Tato písmena ani nejsou nijak graficky oddělena (ani pomocí mezer v textu), nebo barevně označena, aby byla alespoň částečně zvýšena přehlednost této značně nešťastné prezentace informací. Podobně nevhodně je řešena i reprezentace docházky a dalších částí systému – výpis seznamu do tabulky, která je pro uživatele přehledná téměř stejně, jako hrubá textová data. Nevhodně seřazená textová data.

4. Nekonzistentní ovládání: Při přihlašování do systému stačí omylem pomocí klávesy tabulátor vybrat jako „focus“ pole „Language Preference“ a objeví se velký dialog překrývající většinu obrazovky, který ale už nelze zrušit pomocí žádné klávesy, například escape, a je nutné hledat tlačítko pro jeho zavření. Buďto by dialog mělo být možné zavřít stejným způsobem, jakým byl otevřen (pomocí klávesové zkratky), ale spíše by tento dialog neměl být otevřen po vybrání tohoto tlačítka, ale až po kliknutí na ně.

   Nefungující „Zpět“ v prohlížeči: Přechod o stránku zpět v rámci prohlížeče buďto nemá žádný efekt, nebo nás vrátí do nekonzistentního stavu, kde stránka označená v menu za vybranou neodpovídá zobrazené stránce.

   Nastavení umístěné v sekci „My Info“: Ačkoli umístění nastavení na stránce účtu uživatele je pravděpodobně správné, název „My Info“ není standardním označením pro stránku účtu a je matoucí (lépe možná „My Account“).

6. Nejasný význam prvků UI: U ročníku, jména, rasy a data narození studenta je hvězdička („*“), označující pravděpodobně povinný údaj, nebo jiné dodatečné informace k této položce, ale ani v nápovědě není vysvětleno, jaký přesně má hvězdička význam.

8. Duplicitní položky ve formuláři: Stránka pro vyplnění adresy dítěte obsahuje čtyři různé sekce, kde je zkopírována adresa dítěte. Účel je takový, že pokud je v dané sekci zaškrtnutý checkbox „Same as Home Address“, pak je pro kontaktní adresu, záložní adresu a sekundární záložní adresu použita standardní adresa uvedená výše na stránce. Ovšem i při zvolení, že chceme užívat standardní adresu, systém tato pole zobrazuje (na stránce je tedy pět polí pro vyplnění adresy zobrazeno čtyřikrát, pokaždé se shodným obsahem). Systém navíc umožňuje tato pole vyplňovat, takže se z pohledu uživatele zdá, že mění údaje standardní adresy. Krom toho, že jsou tato pole zbytečná, je navíc nejisté, která ze čtyř různých hodnot se tedy vlastně použije, když všechny označují tu samou adresu (pokud je zvolena volba „Same as Home Address“).

   Nepřehledné členění formuláře: Formulář „Final Grades“ – který neobsahuje seznam získaných známek, jak by se dalo očekávat, ani nápovědu o tom, co vlastně obsahuje – se skládá ze sedmnácti různých checkboxů umístěných v tabulce. Krom toho, že tyto položky formuláře mají zvláštní a nepříliš sdílné popisky (například „Sum Sem“), nemají tooltip a jsou poměrně nepřehledně členěny, jsou navíc jako ovládací prvky použity checkboxy v situaci, kdy by bylo na místě drop-down menu (pokud tedy uživatel nechce zobrazit výsledky svých dětí pouze za první a třetí čtvrtletí současně).

   Výběr data ze seznamu týdnů v roce: Padesát dva položek je více, než je vhodné pro drop-down menu. Obzvláště, když v systému jsou pro takovéto účely v ostatních případech používány kalendáře.

   Nepopsané ovládací prvky: Na stránce „My Info“ v tabulce „School Information“ je checkbox, který nemá žádný nadpis, tooltip, ani jiné označení.

9. Při pokusu stáhnout soubor ze systému nás aplikace osočí, že se snažíme obejít zabezpečení a náš prohřešek byl zaznamenán. Může jít o úmyslnou vlastnost demonstrativní verze, ale nikoli příjemnou a rozhodně se nejedná o vhodné chybové hlášení.

   Řešení chyb v aplikaci: Chybová hláška „V OpenSIS nastala chyba, kterou mohla způsobit: chyba v databázi, chyba v programu, chyba v uživatelském vstupu. Prosíme, udělejte screenshot této obrazovky a pošlete ho vašemu přiřazenému správci OpenSIS.“ porušuje nejedno doporučení ohledně zobrazování užitečných chybových hlášek.

10. Nedostačující nápověda: I když se nápověda v systému snaží býti užitečná, vůbec nepopisuje chování některých ovládacích prvků v systému a jejich použití tak není jasné. Poučením by měla být snaha v rámci nápovědy objasnit všechny ovládací prvky na obrazovce a vysvětlit jejich funkci.

Zvláštní chování nápovědy: Při přechodu ze stránky s otevřenou nápovědou na jinou stránku zůstane otevřené okno nápovědy. To samo o sobě zní jako dobrý nápad, ovšem pokud je v tomto okně ponechána nápověda k předchozí stránce, poněkud to jeho kvality snižuje. Stejně tak nestačí myší kliknout mimo okno nápovědy, nebo zmáčknout klávesu escape, je třeba kliknout na relativně malé tlačítko „Hide help“, které navíc není korektně umístěné a je vidět pouze malou část z něj.

Nerelavantní nápověda: Dostupná nápověda se často vůbec netýká stránky, na které se v uživatel systému nachází (a to i v okamžiku kdy ji korektně otevřeme a nejedná se o problém zmíněný v předchozím bodě).

Co je možné z demonstrativní verze soudit, bod 1 byl splněn pomocí ikon načítání stránky a dalších vhodných indícií o stavu systému, body 3 a 5 také nebyly porušeny, ale jinak bylo objeveno značné množství velmi vážných chyb v UI.

C.4.4 Technické detaily

Jak je možno poměrně s jistotou usoudit ze stažených a nainstalovaných souborů, systém OpenSIS byl vytvořen v programovacím jazyce PHP.

Za zmínku stojí, že systém nepoužívá objektový návrh a celý je tvořen výhradně procedurálním[54] kódem, jehož čitelnost je značně nesnadná jak díky velké provázanosti mezi soubory, tak kvůli úplně absenci pojmenovaných funkcí a celkově nízké úrovni kódu (celé soubory jsou tvořeny jedním blokem kódu bez jakéhokoli členění, kterýžto kód je protkán mnohdy až několika stovkami řádek komentářů obsahujících pouze starý kód). Stejně tak se nezdá, že by systém používal jakýkoliv framework, ani pro vzhled stránek, ani pro tvorbu aplikační logiky.

Co se formy distribuce týče, systém je možno používat zdarma jako lokální instalaci, nebo jako cloudovou SaaS spravovanou poskytovatelem, za kterou je však přirozeně třeba platit.

C.4.5 Závěr

Uživatelské rozhraní působí na první pohled lépe, než u systému Škola OnLine, díky pohlednějšímu grafickému designu, ale toto zdání klame, jelikož chyby nalezené v uživatelském rozhraní programu jsou naprosto zásadní. Nepřehledné formuláře, zcela nevhodná forma prezentace záznamů a problémy s ovládacími prvky prakticky všech netriviálních funkcí v systému se opakují napříč celým UI. Stejně tak technická stránka projektu působí značně rozporuplně, jelikož kód aplikace není nikterak strukturován a jedná se prakticky o monolitický kód rozdělený pouze do souborů.

[54] Pokud lze celé soubory považovat za „procedury“, které jsou opakovaně užívány napříč systémem.

OpenSIS tedy nelze s jeho problémy a cílením na jiný stupeň vzdělávání považovat za silnou konkurenci projektu Mojeskolka.

C.5 Letmá analýza dalších systémů

Při hledání vhodných kandidátů pro provedení rešerše bylo alespoň povrchně analyzováno větší množství systémů. Velká část byla z výběru vyloučena ihned (jednalo se pouze o finanční systémy, nebo systémy s výrazně jiným zaměřením, než projekt Mojeskolka), ale některé byly zkoumány více detailně, než byly z výběru vyloučeny. Jmenovitě se jedná o produkty FAME firmy Focusitservices, Pre-K Information System společnosti Choice20 a sada programů Bakaláři společnosti BAKALÁŘI software s.r.o., která je v České republice poměrně známá.

U těchto systémů byla zkoumána primárně funkcionalita, jelikož o GUI byly k dispozici značně omezené informace a testovací režim aplikací opět typicky nebyl přístupný pro veřejnost. Zároveň se jedná o systémy přirovnatelné spíše k učitelské verzí aplikace, která není naším primárním cílem a pro kterou již máme zjištěno mnoho informací z výše uvedených rešerší.

Pro nás relevantní funkcionalita těchto produktů navíc velmi přesně odpovídá námi plánované – evidence dětí, přiřazování dětí (entit) k učitelům, nebo třídám (jiným enititám), evidence a sdílení informací o nadcházejících událostech (v našem případě řešeno notifikacemi), automatický sběr dat pro administrátory (řešen v rámci této práce v sekci Dlouhodobý rozvoj projektu) a další funkce systémů se překrývají s námi plánovanými. To je do jisté míry známkou naší správné představy o fungování systému, zpracované v požadavcích na systém a v případech užití výše v této práci.

Bylo však možné narazit i na funkcionalitu, která by nás mohla inspirovat a dostupné stránky některých systémů umožňovaly analyzovat i technologické detaily. Přehled takovýchto získaných informací je uveden níže.

C.5.1 Zajímavé funkce

Jedná se o funkce poněkud více vzdálené jádru projektu Mojeskolka, ale některé z nich by mohlo být dobré zvážit do budoucna.

- **Vazba na sociální sítě:** Ať už je možné myslet si o sociálních sítích cokoli, jsou v současnosti velmi důležité pro zviditelnění produktu a kontakt s veřejností. Proto se dá očekávat, že možnost vytvořit skupiny nebo kanály na známých sociálních sítích a sdílet na nich snadno některé události z dění ve školce může školkám pomoci zviditelnit se a oslovit potencionální rodiče-zákazníky, nebo pomoci vybudovat dobrou pověst dané školky.
- **Účetnictví:** V rámci rozšiřování služeb poskytovaných školkám může systém umožňovat evidenci plateb, například v souvislosti s placením

obědů a dalších poplatků. Automatické notifikace upozorňující rodiče na zaplacení pravidelných plateb zároveň ušetří starosti rodičům. Tato funkcionalita byla do budoucna navrhována i samotným zadavatelem projektu.

- **Online přihlášky:** Jedná se spíše o vlastní myšlenku, nežli funkcionalitu poskytovanou některým ze zkoumaných produktů, ale tento nápad vznikl na základě úvah o nástrojích určených pro správu přijímacího řízení, které některé systémy poskytují . V rámci funkcí pro veřejnost, tedy i pro rodiče, kteří ještě nepoužívají aplikaci Mojeskolka, by systém mohl nabízet možnost vyhledání vhodné školky v okolí a následného navázání kontaktu, případně i rezervace místa v dané školce, což by usnadnilo přihlašování dětí do školky jak pro rodiče, tak pro školku.

C.5.2 Technologie

O systému Bakaláři, který je z uvedených produktů nejblíže projektu Mojeskolka, bylo možné získat bližší informace v oblasti užitých technologií. Stránky pro přihlášení do systému totiž ukládají do počítače soubor cookie s názvem „ASP.NET_SessionId“, který se vyskytoval i u systému Škola OnLine a poukazuje na použití platformy .NET, stejně tak, jako na její užití poukazuje přítomnost knihovny DXR.AXD určené pro jazyk C#. Co se distribuce týče, systém Bakaláři není poskytován jako SaaS, ale pouze formou lokální instalace, kterou musí klient spravovat samostatně.

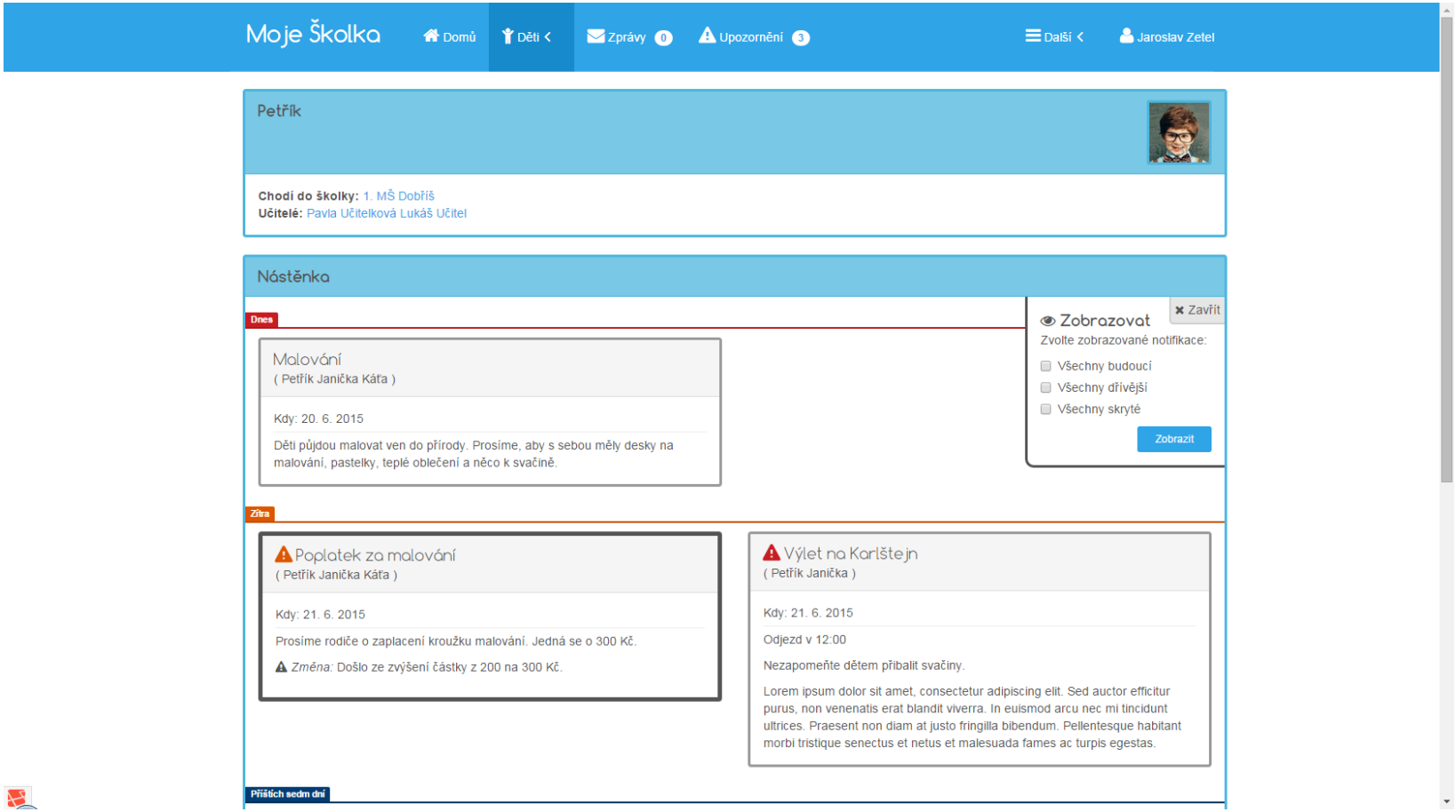
C.6 Závěr

Závěr z uvedených rešerší je možné nalézt výše v kapitole Návrh uživatelského rozhraní, v sekci Závěr provedených rešerší.

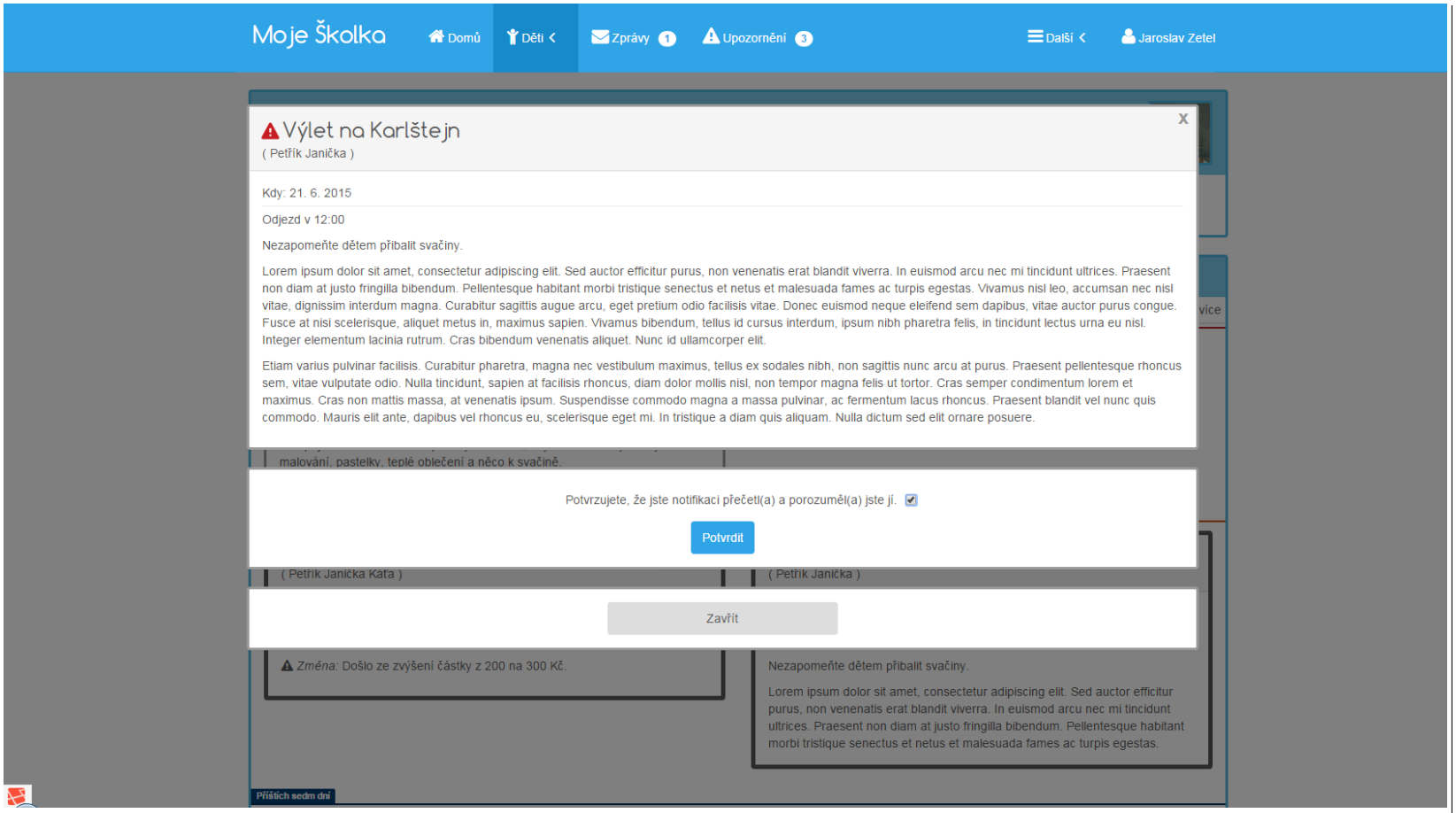
Příloha D

Ukázky vytvořeného prototypu

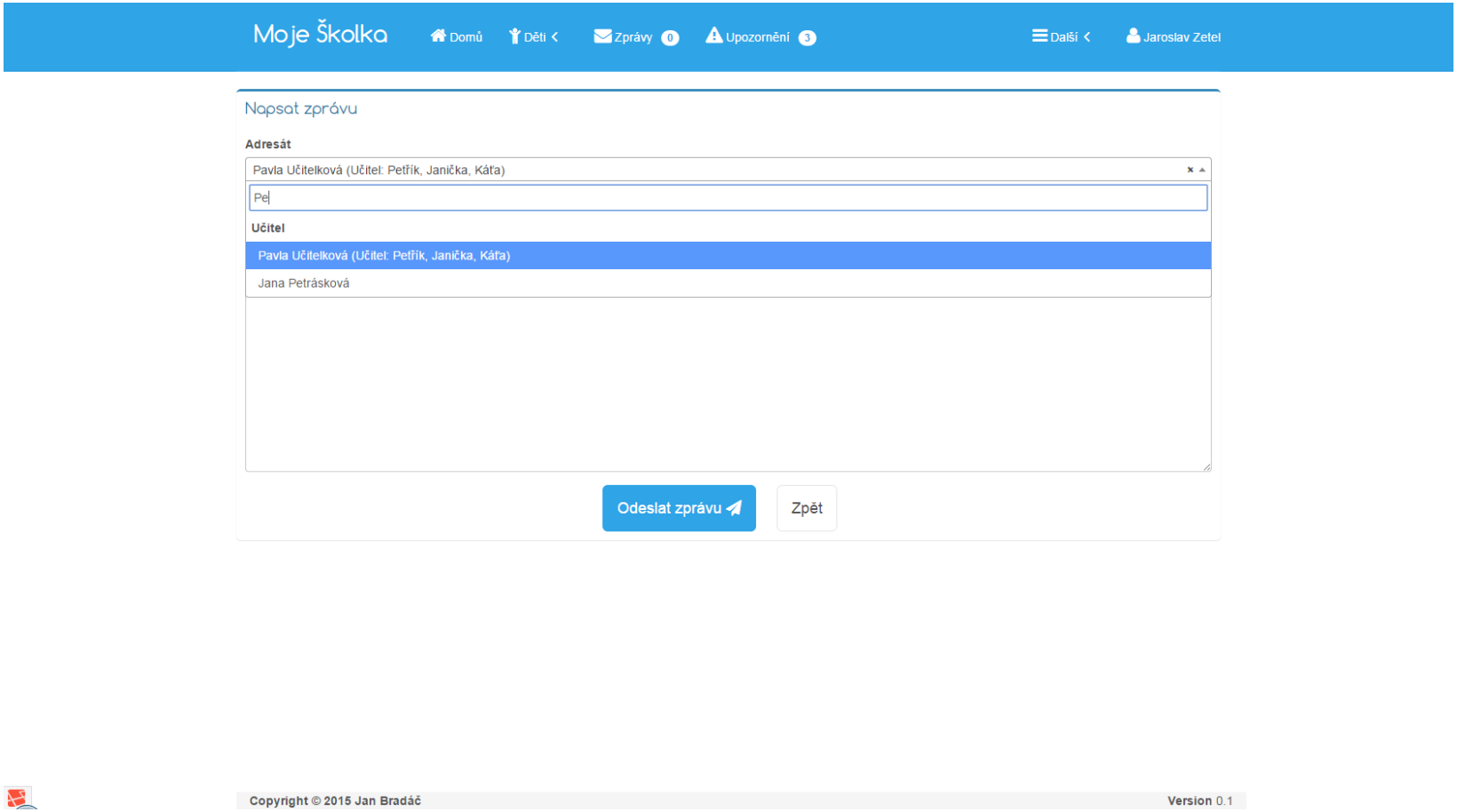
V této příloze jsou uvedeny ukázky obrazovek z vytvořeného prototypu.



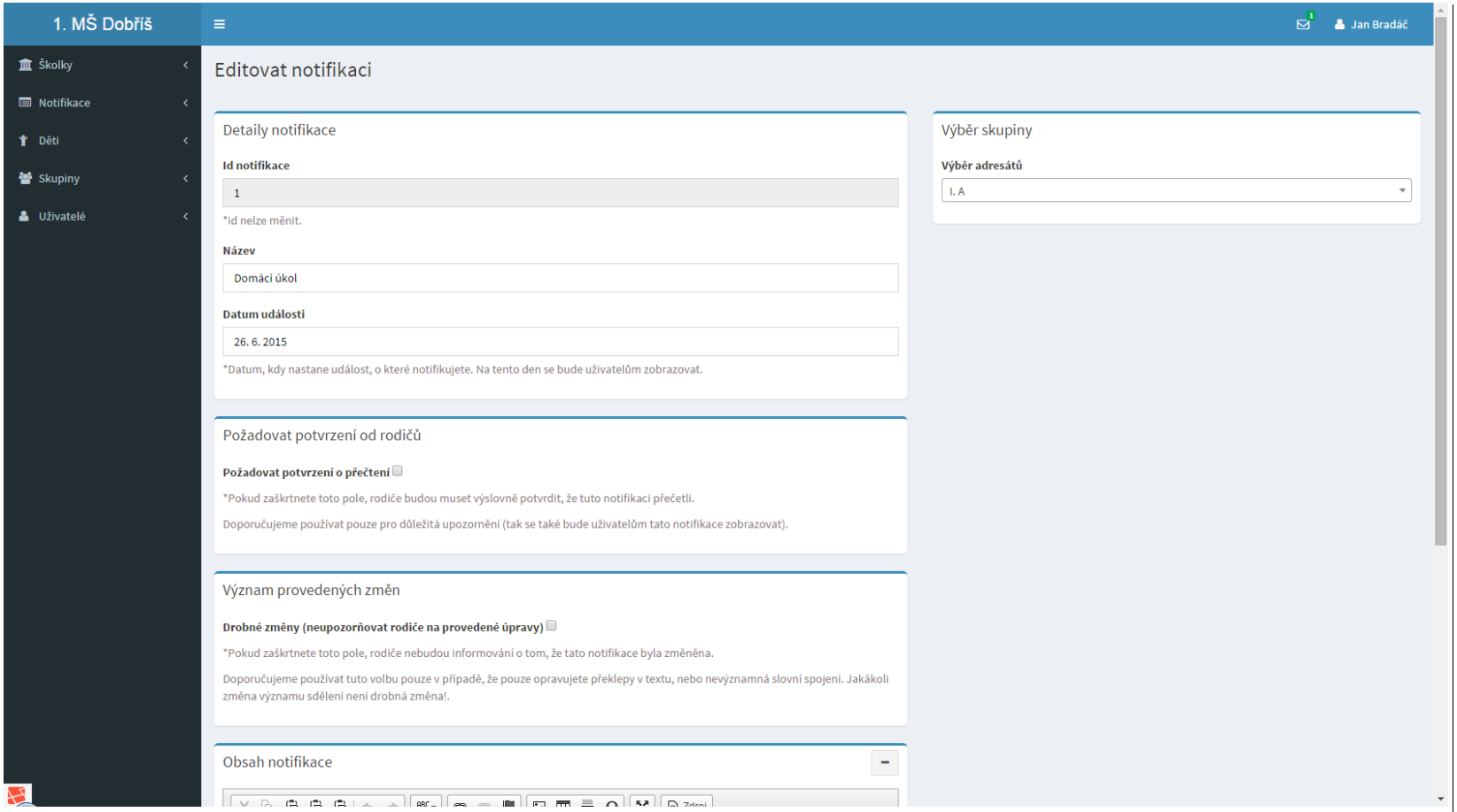
Obrázek D.1: Stránka dítěte v rodičovském rozhraní.



Obrázek D.2: Otevřený detail notifikace v rodičovském rozhraní.



Obrázek D.3: Stránka pro napsání nové zprávy v rodičovském rozhraní.



Obrázek D.4: Ukázka učitelského rozhraní.

Příloha E

Výsledky uživatelského testování

V této příloze je uveden průběh testování uživatelského rozhraní. Detaily testovaných osob, scénáře, podle kterých bylo testováno a závěry z testování je možné nalézt v sekci Testování.

Přepis výsledků je pouze stručný, zaměřuje se na splnění úkolů scénáře a především na problematické části, jelikož primárním cílem je objevit a opravit chyby v prototypu.

E.1 Průchody scénářem při prvním uživatelském testování

E.1.1 První uživatel

Rodičovské rozhraní

1. a. Přihlásí se, ale server vyhodí neošetřenou chybu, protože již je neplatný token proti cross-site request forgery útokům, jelikož byla stránka dlouho načtena (několik hodin). Druhé přihlášení proběhne bez problému.

   b. Má problémy zorientovat se. Hledá profil, ale neví, kde ho najít. Přechází na „Domů“ a další stránky. Je třeba ji upozornit, že je nevhodně vyplněné její jméno a že uživatel se jmenuje „Tomáš Novák“, což je napsáno v pravém horním rohu obrazovky. Okamžitě se situace mění, kliká na příslušnou ikonku, vidí profil a ten bez problému vyplní.

2. Nejprve hledá zpět (možná by bylo vhodné implementovat breadcrumb), velmi rychle přejde na hlavní stranu, kterou prohlíží a aktivně se snaží pochopit prezentované informace. Vykládá si je poměrně správně, akorát upozorňuje, že „na dnešek už je pozdě, ráno nemám čas na podobné věci koukat“, což může být užitečné.

Ihned si všímá, že se jí systém hlásí nepřečtené „zprávy“ a „důležité“, tudíž čísla u těchto položek uživatelé chápou dobře

3. Neví, kde hledat, na jakou stránku přejít. Hledá stránku „Kontakt“, ale nenachází. Je potřeba ji upozornit, že jméno školky je odkaz, poté je problém vyřešen, ale celkově je s tímto bodem značný problém. Uživatelka podotýká, že je pro ni matoucí, že název školky nemá žádné barevné zvýraznění, nebo podtržení, jaké mají běžné odkazy.

4. „To budou upozornění, že?“ táže se. Přechází ihned na příslušnou stránku, kde se seznamuje s barevným odlišením různých typů notifikací. Nejprve však myslí, že červená značí změnu, až poté si všímá nadpisů jednotlivých sekcí. Tento problém by se však měl velmi rychle odbourat u uživatelů, kteří aplikaci nevidí poprvé. Poměrně trefně navrhuje, že by u pozměněné notifikace měla být přidána sekce, která stručně vystihuje provedené změny.

5. a. Přechází na stránku Petříka přes menu - „Děti“.
   b. Bylo řešeno už v předchozím bodě, nebylo tedy potřeba opakovat.
   c. Bez problému skrývá (dříve byly problémy kvůli nejasnosti „Skrýt a nezobrazovat“ a „potvrdit přečtení notifikace“, ale lepším rozložením prvků na stránce byly očividně odstraněny).
   d. Bez problému potvrzuje. Dřívější problémy byly výraznějším oddělením sekcí vyřešeny.
   e. Snaží se kliknout na jméno „Janička“ u jiné notifikace (což však pro přehlednost úmyslně není implementováno), když vidí, že toto nefunguje, přechází na stránku Janičky a identifikuje notifikaci, která patří pouze jí. Chápe jména dětí uvedená pod názvem notifikace.

6. a. Zprávy chtěla celou dobu otevřít a podívat se, kdo jí píše, proto nyní přesně ví, o co se jedná.
   b. Má problémy identifikovat, která zpráva je nová, protože v prototypu je omylem implicitně řazení nikoli podle data, ale podle předmětu zprávy, což dává na první místo v seznamu již přečtenou zprávu.
   c. Bez problému přechází na stránku odpovědět.
   d. Chvíli hledá tlačítko „Napsat novou zprávu“, ale pak komentuje, že se jen špatně podívala a tlačítko přehlédla. Při psaní adresáta má však problém, snaží se kliknout na šipku pro změnu adresáta a nemůže se trefit, jelikož je šipka malá (tato šipka je součást pluginu používaného pro lepší HTML prvek „select“). Přitom je tento úkon zbytečný, jelikož samotná šipka není nikterak potřebná, funkci, kterou od ní očekávala, plní celá plocha pro psaní jména adresáta.

Učitelské rozhraní

1. Bez problému.
2. Bez problému.
3. Taktéž bez problému.
4. Jde do seznamu uživatelů, ráda by přešla na jeho profil a přes něj mu napsala zprávu. Bylo by vhodné přidat tlačítko pro napsání zprávy na stránku uživatele.

   Po chvíli nachází zprávy v pravém horním rohu, ale umístění ikony jí nevyhovuje. Po přechodu na stránku zpráv již bez problému.

E.1.2 Druhý uživatel

Rodičovské rozhraní

1. a. Bez problému
   b. Splní, ale ačkoli testy probíhaly odděleně, tak sama zmiňuje, že z vedlejší místnosti zaslechla, že upozorňuji na nekorektní uživatelské jméno a proto ví, že tam je její profil. Tudíž úspěch v tomto bodě nemá velkou váhu.
2. Bez problému přechází na hlavní stranu a zjišťuje potřebné informace.
3. Hledá stránku „Kontakt“. Korektně poznamenává, že „Zprávy jsou jen pro mne, tam nebudou adresy“ a zkouší jít na hlavní stranu. Nachází Petříka, zkouší hledat kontakt na jeho stránce, ale zde je opět ztracená. Stejně jako u prvního testovaného uživatele je potřeba upozornit na to, že název školky je odkaz na novou stránku (v tomto případě však v kontextu stránky dítěte, nikoli hlavní strany, odkud se původně předpokládalo, že uživatelé budou na stránku školky přecházet). Poté již kontakt nachází, ale opět je tento bod problematický.
4. Přechází automaticky na stranu „Upozornění“, bez problému chápe (přesněji, než první uživatelka).
5. a. Přechází na stránku Petříka. Je vidět, že odkazy zvýrazněné modrou barvou uživatelé ihned chápou a využívají.
   b. Bez problému řeší.
   c. Bez problému.
   d. Taktéž plní bez obtíží.
   e. Přechází přes menu - „Děti“ na stránku Janičky a je jí jasné, že notifikace, která má pod názvem uvedenou jen Janičku, je pouze pro ni.

6. a. Stejně, jako první zkušební uživatelka, už předem upozorňovala na ikonku zpráv, takže ihned ví, že se jedná o zprávy. (Ikony v menu poutají pozornost výrazně více, než bylo původně očekáváno, ale to je pozitivní zpráva)
   b. Chápe, že tučně je nová zpráva, pořadí zpráv ji neruší.
   c. Bez problému.
   d. Bez problému přejde na stránku „Napsat novou zprávu“, ale zde opět zbytečně cvaká na drobnou šipku. Jinak vyplňuje bez problému

Učitelské rozhraní

1. Bez problému.
2. Taktéž nemá problém, učitelské rozhraní je očividně blízké standardním informačním systémům, se kterými běžní uživatelé přichází do kontaktu v zaměstnání.
3. Taktéž bez problému.
4. Váhá, jestli uživatelé nejsou jen učitelé, ale pak přechází na stránku zpráv pomocí ikony v pravém horním rohu a bez problému úkol plní.

E.2 Průchody scénářem při druhém uživatelském testování

E.2.1 První uživatel

Rodičovské rozhraní

1. a. V pořádku, volí „zůstat přihlášen“.
   b. Bez problému kliká na své jméno (což je změna vůči minulému testování) a mění nastavení profilu.
2. Jde na stránku dítěte a nachází úkoly.
3. Jde do „Zpráv“, říká, že e-mail ho nezajímá a raději by psal přímo v aplikaci. Po ujištění, že by skutečně rád napsal e-mail, bez problému přechází na stránku Kontakty a e-mail najde. Negativně však hodnotí uspořádání prezentovaných informací a doporučuje vhodnější rozložení, které bude rozvedeno ve výsledcích tohoto testování.
4. Přechází na stránku Petříka a správně identifikuje důležité notifikace jako ty s ikonou vykřičníku. Jak se však později ukáže, neví, že červená znamená, že notifikace vyžaduje potvrzení.

X1. a. U notifikace mu chybí tooltip nápověda, stránky jsou sice specializovány na mobilní zařízení, ale větší zaměření na stolní počítače by nejspíše bylo vhodnější.

b. Hledá, kouká. Po upozornění na existenci nápovědy je se samotnou stránkou spokojený a chápe z ní, co znamená zvýraznění notifikace, ale upozorňuje, že nápovědu není zvyklý používat, stejně tak, jako většina zkušených uživatelů informačních technologií.

c. Nyní chápe, co znamená červený vykřičník, ale upozorňuje na to, že by pro nové uživatele bylo vhodné přidat tlačítko „Potvrdit přečtení a zobrazit detail“, které zdůrazní, že notifikace se dá otevřít.

5. a. Bez problému nachází nový ovládací prvek pro zobrazení skrytých notifikací, ale doporučuje přidat tlačítko i dolů, uživatelé jsou zvyklí scrollovat a nechat si zobrazit další informace.

b. Hledá a neví, kde hledat, ani co hledat. Nápovědu nepročetl celou (což však bohužel reflektuje většinu uživatelů) a v důsledku nyní neví, že oranžový vykřičník znázorňuje editovanou notifikaci. Hledá ve zprávách, pak usoudí, že by mohla podobná informace být uvedena na stránce „Upozornění“. Avšak i na této stránce tápe a potvrzuje tvrzení, že „Uživatelé nečtou“ – ačkoli je v polovině stránky velký nadpis „Byly změněny“ a pod ním příslušné notifikace, nadpis nevidí, dokud není upozorněn na jeho existenci. Poté chápe, akorát kritizuje málo zvýrazněnou sekci „Opraveno:“ (v této verzi označení jen textem). Doporučuje přidat ikonu.

c. Bez problému zavírá, ale kritizuje (zcela oprávněně), že okno potvrzující skrytí notifikace překrývá dropdown menu dětí. (Tato chyba byla po skončení testů opravena.)

d. Bez problému potvrzuje, ale nemůže se trefit na checkbox (i jako zkušený uživatel kliká na checkbox, ačkoli je možné klepnout i na celý text pro stejný efekt). Stálo by za úvahu zvětšení checboxu.

e. Přechází na stránku Janičky, bez problému nachází notifikaci pouze pro ni. Zmiňuje však, že by bylo možné k notifikacím přidat ikony chlapce nebo dívky, podle toho, komu daná notifikace patří. Podobná funkce je plánována jiným způsobem, ale jedná se o zajímavý nápad.

6. a. Zcela s přehledem úkol plní, uživatelé očividně ikonu „bubliny“ u zpráv chápou.

b. Pozná tučně zvýrazněnou novou zprávu, ale vadí mu pořadí, které je v této verzi prototypu chybné.

c. Po stránce UI bez problému, ale v aplikaci je chyba, způsobená opravou typu HTTP metody použité pro daný požadavek.

d. Ze stejného důvodu není možné zprávu napsat, ačkoli ovládací prvky UI problémy nepůsobí.

Učitelské rozhraní

1. Bez problému.

2. Chybí informace, co je dnes za den, jinak v pořádku. Vyžaduje potvrzení a upozorňuje, že až na obecné informace (děti mohou přijít na drakiádu) většina notifikací nejspíše potvrzení vyžadovat bude.

3. Bez problému.

4. Po stránce UI bez problému, ale naráží na problém se zprávami, zmíněný výše.

E.2.2 Druhý uživatel

Rodičovské rozhraní

1. a. Bez problému. Jakékoli problémy zmizely po změně scénáře zahrnující použití e-mailů a jmen, které testujícím osobám skutečně náleží.

   b. Jde přes profil do nastavení a bez problému mění. Opět se zdá, že skutečné jméno uživatele funguje výrazně lépe, než vzorové jméno, byť by na ně byli uživatelé upozorněni.

2. Jde přes logo „Moje Školka“ na stránku, která v současné verzi prototypu odpovídá funkci „Volba role“, což není zcela šťastná vlastnost prototypu. Možná by bylo vhodné při vyplňování profilu ponechat menu příslušící danému rozhraní, ale pak vyvstávají problémy s nekonzistencí mezi rodičovským a učitelským rozhraním.

3. Zkouší logo Moje Školka, znovu volí roli. Jde do zpráv, chce napsat zprávu, hledá 1. MŠ Dobříš. Po upozornění, že by ho zajímala i adresa školky pochopí, že se jedná o kontakty, chvíli hledá a pak v menu volí Další a na stránku Kontakty, kde už nemá problém. [55]

[55] Když ho autor práce požádá, jestli by mohl přejít na hlavní stránku, sjet níže (aby byly vidět panely dětí) a nyní je dotázán, jak by přešel na stránku Petříka, odpoví, že by kliknul na panel Petříka, což je naprosto správný postup. Když je však dotázán, jak by přešel na stránku s kontakty školky, stejně jako všichni ostatní testeři netuší. Když je na tuto skutečnost upozorněn, vypadá stejně pobavený a zaskočený, jako autor této práce. Je poněkud absurdní, že každého přirozeně napadne kliknout na jméno Petříka, ale nikoho nenapadne kliknout na jméno školky, ačkoli se jedná o stejný typ panelu a zvýraznění panelu dítěte je dokonce menší.

4. Jde přes menu Děti na stránku Petříka, zde chápe, že vykřičníky znamenají upozornění na důležitost, ale zatím nerozumí rozdílu mezi oranžovým a červeným.

X1. a. Grafického odlišení i vykřičníku si všiml.

b. Jde korektně přes Další na stránku Nápověda. Stránka se mu líbí, chválí příklady notifikací.

c. Rozumí nyní tomu, co které označení znamená, vypadá spokojenější, že se v zobrazovaných informacích lépe orientuje.

5. a. Jde na stránku Petříka, sjíždí nejprve dolů (což potvrzuje dříve zmíněnou vhodnost přidání tlačítka „Zobrazit více“ i dolů na stránku), poté vidí příslušné tlačítko a korektně volí zobrazované notifikace.

b. Sjíždí níže, kde se nachází příslušná notifikace, vidí upozornění na změnu, vše v pořádku.

c. Otevírá notifikaci, zajímavé však je, že tak činí kliknutím na oranžový vykřičník (který musí pomoci myši chvíli „zaměřovat“) – očividně není dostatečně jasné, že celá plocha notifikace je „klikatelná“, opět se potvrzuje vhodnost přidání informace o tom, že je možno kliknout na celý panel.

d. Přejde na hlavní stranu, hledá u Petříka a vidí korektně „Výlet na Karlštejn“. Opět kliká na malou ikonu vykřičníku, jinak bod plní bez problému.

e. Jde na stránku Janičky a korektně pozná notifikaci pouze pro ni.

6. a. Nejprve jde na stránku Upozornění, protože stále ještě „svítí“ jedno nevyřízené upozornění v menu. Po jeho vyřešení si všímá, že má přijatou zprávu.

b. Bez problému identifikuje tučnou zprávu jako nově přijatou.

c. Bez problému, zmiňuje však, že by bylo vhodné nějakým způsobem realizovat vlákna zpráv. Například alespoň pomocí „Re: Předmět původní zprávy“, jak je typické u mailových klientů.

d. Opět plní úkol bez problému, ale zbytečně kliká na malou šipku, podobně, jako první testující osoby.

Učitelské rozhraní

1. Bez problému. Nedává „Zůstat přihlášen“, na rozdíl od většiny ostatních.

2. Chybí mu označení dnešního dne, což je chyba na straně UI, jinak plní bez problému. Požaduje přečtení, což je však pochopitelné. Po stránce UI v pořádku, nelze však notifikaci uložit, protože se projevuje problém

v prototypu aplikace, který byl sice opraven, ale tento patch omylem nebyl zkopírován na testovací počítač.

3. Po stránce UI bez problému, opět zavází chyba v prototypu, která však nesouvisí s UI.

4. Jde přes seznam uživatelů na stránku příslušného uživatele a volí Napsat zprávu, což je zcela korektní. Poté si všímá i ikony zpráv v pravém horním rohu.

E.2.3 Třetí uživatel

Rodičovské rozhraní

1. a. Bez problému i na tabletu.

   b. Bez obtíží přechází na stránku profilu a plní úkoly.

2. Přechází na hlavní stranu a prohlíží si notifikace. V průběhu testování tento uživatel využívá hlavní stranu více, než jemu předcházející uživatelé, ale to může souviset s jeho znalostí koncepce produktu.

3. Přejde na hlavní stranu. Hledá v zápatí, pravděpodobně něco jako „Kontakty“, po chvíli uvažování klikne na 1. MŠ Dobříš. Učinil tak jako úplně první z celé řady testerů a i tak působil poněkud nejisté.

4. Přes hlavní stranu přechází na stranu Petříka, kde si prohlíží jeho důležité notifikace.

X1. a. Odlišení notifikací si již dříve všiml.

   b. Otevírá detail notifikace, kliká na vykřičník, když to nepomáhá, hledá dále. Jde domů a odkaz na nápovědu nemůže najít, dokud není upozorněn na položku „Další“, kde se v menu nápověda nachází.

   c. Nyní již chápe, co které označení znamená, pozitivně hodnotí tučné zvýraznění ještě neotevřené notifikace, ale vadí mu nepřítomnost tooltipu.

5. a. Jde na stránku Petříka, používá bez problému „Zobrazit více“, kteréhožto tlačítka si všiml již dříve.

   b. Bez problému.

   c. Bez problému.

   d. Bez problému. Hodnotí pozitivně UI a grafický design[56].

[56]Který je v této verzi stále provizorní, i když už se nejedná o katastrofální stav. Možno posoudit v přílozeUkázky vytvořeného prototypu.

e. Přechází na stránku Janičky, chvíli kouká, pak nachází notifikaci jen pro ni.

6. a. Bez problému.
   b. Vidí tučně označenou zprávu a chápe její význam.
   c. Bez problému.
   d. Bez problému.

Učitelské rozhraní

Celé testy učitelského rozhraní proběhly bez problémů, pan Zetel hodnotí pozitivně návrh UI a v souladu s výše vyslovenými domněnkami podotýká, že učitelské rozhraní je velmi intuitivní a jednoduché, do jisté míry i díky své podobnosti s jinými informačními systémy.

Zhodnocení testování na mobilních zařízeních

Na obou zařízeních – tabletu i mobilním telefonu – se potvrdilo, že UI je skutečně plně responzivní a z hlediska rozložení prvků na stránce nebyl nalezen jediný problém. Lehce neočekávané chování vykazuje pořadí některých Javascriptových animací, které se provedou po kliknutí na panel notifikace (místo po najetí, jak je tomu na stolních počítačích) a brzdí otevření jejího detailu – bude tedy třeba tento problém adresovat. Jinak je celé UI bez problému použitelné, jak by tomu u správného responzivního UI mělo být.

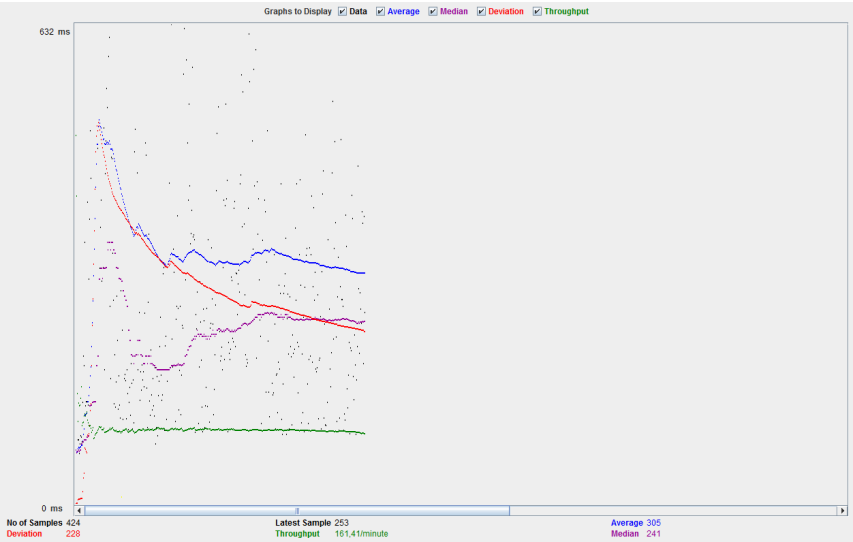
Příloha F

Výsledky zátěžových testů

V této příloze je možné nalézt doplňující informace k zátěžovým testům zmíněným výše, v kapitole o testování.

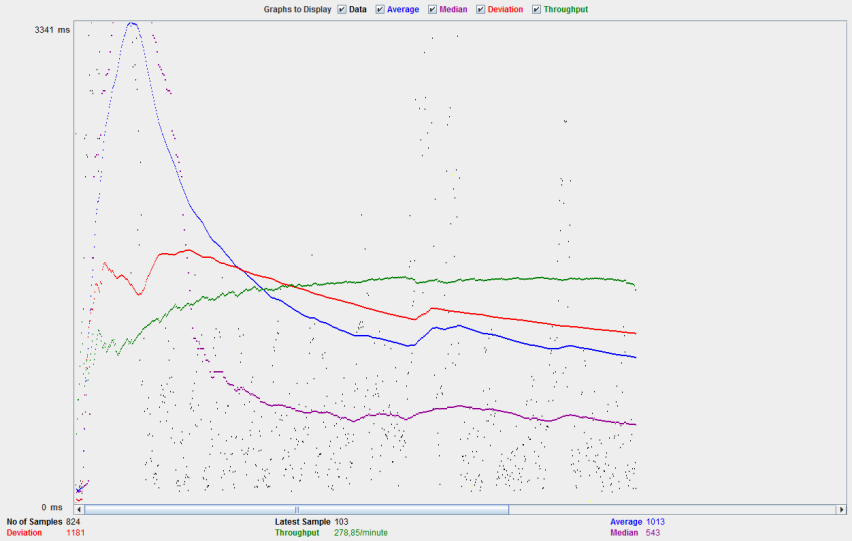
Následující obrázky reprezentují výstup z nástroje Apache jMeter pro různé sady testů s různým nastavením. Jejich výsledky jsou shrnuty v kapitole testování v sekci Zátěžové testy.

Uvedené grafy zachycují časový průběh na vodorovné ose a na ose svislé (průměrnou) dobu nutnou pro vyřízení požadavku a průměrnou propustnost (jedná se o údaje se zcela odlišnou jednotkou, umístěné v jednom grafu, což je však „přirozené“ chování Apache jMeter, níže uvedené obrázky jsou neupravený výstup z tohoto nástroje).



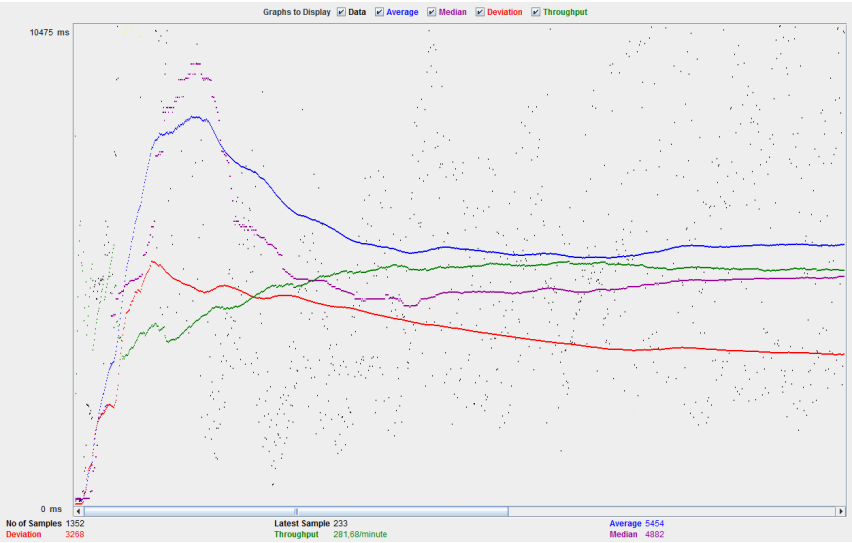
Obrázek F.1: Výsledek zátěžového testu s 15 simulovanými rodiči, 2 učiteli a 1 administrátorem.

Na tomto výsledku je vidět, že v tomto případě server ani není plně vytížen, což se projevuje na velmi nízké průměrné době odezvy, ale i nižší propustnosti, protože požadavky nestíhají server zahltit. Takováto míra vytížení pro server nepředstavuje problém.



Obrázek F.2: Výsledek zátěžového testu s 30 simulovanými rodiči, 3 učiteli a 2 administrátory.

Tento test už je na hranici maximální zátěže, kterou server zvládá a stíhá při ní vyřizovat požadavky dostatečně rychle

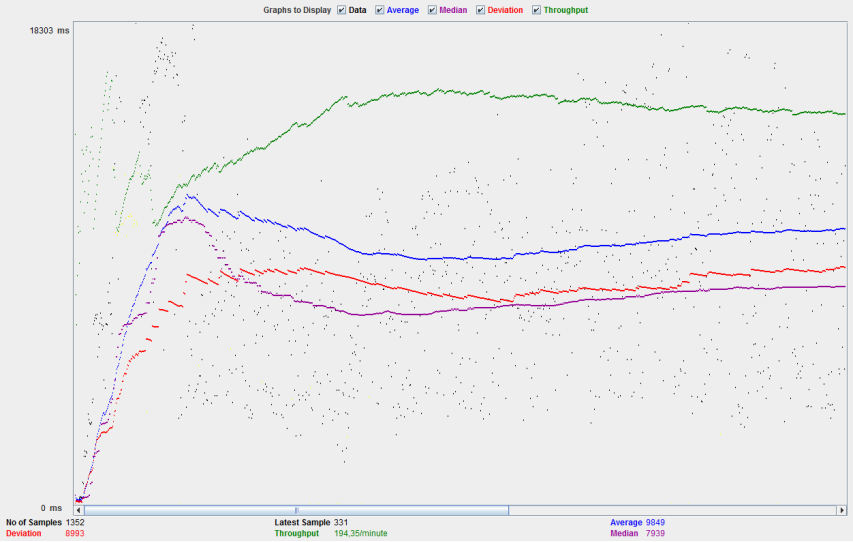


Obrázek F.3: Výsledek zátěžového testu s 50 simulovanými rodiči, 5 učiteli a 3 administrátory.

Jedná se o výsledek testu s maximální zátěží, která byla uvedena v sekci testování.

Výsledky testů před optimalizací

Tyto obrázky zachycují výrazně horší výsledky zátěžových testů, které proběhly před vypnutím na výkon náročných nástrojů pro vývojáře a optimalizací frameworku Laravel pomocí k tomu určeného příkazu, který předpřipraví často používané třídy a správné cachování.



Obrázek F.4: Výsledek neoptimalizovaného zátěžového testu s 50 simulovanými rodiči, 5 učiteli a 3 administrátory.

Na tomto obrázku je možné vidět přibližně dvakrát horší výsledky výše uvedeného testu před provedením optimalizací systému. Tento graf dokládá, že výsledky před optimalizací systému byly výrazně horší.