FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová administrace informačního systému pro střední školu
Student:	Tomáš Trbola
Vedoucí:	Mgr. Petr Matyáš
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1) Proveďte analýzu požadavků na informační systém pro SŠ a VOŠ reklamní a umělecké tvorby Michael.
2) Navrhněte a vytvořte webovou administraci připravovaného systému, který se bude skládat z více modulů, ty budou vypracovány jako samostatné bakalářské práce.
3) Na administraci jsou kladeny tyto požadavky:
 - správa uživatelů, přihlašovacích účtů, tříd a oborů vyučovaných na škole,
 - rozhraní pro import studentů ze systému Bakaláři, který se na škole využívá,
 - rozhraní pro převod studentů do vyšších ročníků,
 - u studentů bude vedena historie tříd a oborů,
 - poskytnutí API není vyžadováno, všechny části systému budou využívat sdílenou databázi,
 - systém bude připraven na možné budoucí rozšíření.
4) Zvolte ideální technologie pro realizaci projektu.
5) Na základě návrhu implementujte prototyp webové administrace.
6) Proveďte uživatelské otestování výsledku a vyhodnoťte kvality a nedostatky vašeho řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 31. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webová administrace informačního systému pro střední školu

Tomáš Trbola

Katedra softwarového inženýrství
Vedoucí práce: Mgr. Petr Matyáš

11. května 2018

Poděkování

Chtěl bych poděkovat panu Mgr. Petru Matyášovi za vedení mé bakalářské práce. Dále poděkování patří mé rodině a nejbližším přátelům za podporu během celého studia.

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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2018 ……

Odkaz na tuto práci

Trbola, Tomáš. *Webová administrace informačního systému pro střední školu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá analýzou požadavků, návrhem řešení a implementací prototypu administrace připravovaného informačního systému střední školy Michael. Výstupem práce je administrace implementovaná v PHP frameworku Symfony. Přínosem této práce je umožnění jednoduché správy uživatelů, přístupových účtů, tříd a studijních oborů vyučovaných na škole. Administrace dále poskytuje možnost importování studentů ze systému Bakaláři a převod tříd do vyšších ročníků. Výsledné řešení úspěšně realizuje všechny požadavky, jež byly na administraci kladeny.

Klíčová slova Návrh a realizace webové administrace, školní informační systém, modulární architektura, uživatelsky přívětivé rozhraní, podpora mobilních zařízení, import uživatelů ze systému Bakaláři, Symfony framework

Abstract

This work deals with the analysis of requirements, the design solution and implementation of the prototype administration of the upcoming information system of the Michael school. The output of this work is an administration implemented in the PHP framework Symfony. The benefit of this work is to enable easy administration of users, access accounts, classes and study subjects taught at school. The Administration also provides the possibility of importing students from the system Bakaláři and transferring classes to higher grades. The resulting solution successfully fulfills all the requirements that have been placed on the administration.

Keywords Design and implementation of web administration, school information system, modular architecture, user-friendly interface, mobile device support, importing users from the system Bakaláři, Symfony framework

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Úvod

Informační Systémy (IS) se staly běžnou součástí našich každodenních životů. Ne vždy ovšem můžeme těchto systémů využít a občas je třeba obejít se bez nich. V době psaní této práce přichází studenti Střední školy a Vyšší odborné školy reklamní a umělecké tvorby Michael (dále jen škola Michael) během studia do styku se zdlouhavými procesy, které stojí jak studenty, tak učitele drahocenný čas. Mezi tyto procesy patří například zápis volitelných předmětů, správa docházky na praxích či proces odevzdávání závěrečných prací.

Výstupem této práce je administrace IS, jenž je určen pro studenty a zaměstnance školy Michael. Informační systém z části automatizuje výše zmíněné procesy, urychluje jejich průběh a poskytuje uživatelům jednoduché a přehledné webové rozhraní. Výsledný IS si neklade za cíl nahradit Bakaláře, jež se na škole v současné době používají, pouze má doplnit chybějící funkcionality.

Toto téma si autor zvolil, jelikož v současné době není na trhu k dispozici žádný modulární IS s otevřeným zdrojovým kódem, který by řešil danou problematiku a vyhovoval stanoveným kritériím. V budoucnu se očekává rozšíření systému o dodatečnou funkcionalitu.

Autor práce se zaměřil na analýzu požadavků, výběr vhodných technologií, návrh a implementaci prototypu administrace. Ta poskytuje mimo jiné správu uživatelů, přístupových účtů, tříd, firem, jejich zástupců a studijních oborů. Dále byl kladen důraz zejména na uživatelské rozhraní importování studentů ze systému Bakaláři v podobě Extensible Markup Language (XML) souborů či převod studentů do vyšších ročníků.

Tato práce dále pokračuje v následující struktuře: Nejdříve se v části 1 autor věnuje průzkumu současné situace, analýze požadavků a použitelných technologií. Druhá část vybírá vhodné technologie pro implementaci na základě předešlých zjištění a představuje návrh aplikace, jenž je následně implementován v části 3.

Poslední část se zabývá testováním implementovaného prototypu, kdy byla zkoumána přístupnost, odezva systému a bylo provedeno uživatelské testování.

Na základě těchto zjištění byly vyhodnoceny kvality a nedostatky tohoto řešení. Informační systém bude dále tvořen moduly studentů z Fakulty informačních technologií ČVUT v Praze Jana Vožeha a Nguyen Cong Long.

Kapitola 1

Cíl práce

Cílem rešeršní části práce je analýza požadavků, jež jsou kladeny na administraci a představení použitelných technologií, v nichž lze implementovat prototyp. Dále bude provedena analýza již existujících školních informačních systémů a jejich přirovnání k současné situaci. Tato část práce také představí jednotlivé případy užití a doménový model.

Cílem praktické části práce je návrh a realizace prototypu administrace za pomoci vhodně zvolených technologií. Na základě funkčních a nefunkčních požadavků bude pro tento účel vybrána jedna z použitelných technologií představených v rešeršní části práce. Výsledný informační systém se bude skládat z modulů, je tedy třeba navrhnout a využít modulární architekturu.

Mezi základní funkcionality administrace patří správa uživatelů, přístupových účtů, tříd a oborů vyučovaných na škole. Dále administrace poskytne rozhraní pro import studentů ze systému bakaláři v podobě XML souborů a rozhraní pro převod tříd do vyšších ročníků. Tato rozhraní si kladou za cíl usnadnit a urychlit dané procesy, které by jinak musel vykonávat uživatel administrace ručně.

U studentů bude vedena historie studijních oborů a tříd, jichž se tento student účastnil. Studentům bude umožněn omezený přístup do systému pomocí jednorázových přístupových kodů, jež bude možné generovat v administraci.

Kapitola 2

Analýza

2.1 Současný stav a existující řešení

Pro účely vypracování analýzy a návrhu aplikace je nejprve třeba porozumět procesům, které na škole Michael probíhají. Následuje představení Střední školy Michael: „*Střední škola reklamní tvorby Michael vznikla v roce 1994 a je jedinou střední školou poskytující komplexní vzdělání v oblasti reklamy. Založil ji se svým synem Michalem Zdeněk Štěpánek, legenda české reklamy, který sám vypracoval i některá studijní skripta.*“ [1]. Na Střední školu dále navazuje Vyšší odborná škola Michael.

Následující sekce analyzuje současné procesy, které na škole probíhají a které budou řešit jednotlivé moduly IS. Z těchto procesů budou vyvozeny předběžné požadavky na administraci, které dále budou rozvedeny v sekci 2.3. Dále autor práce provede průzkum trhu a analýzu již existujících informačních systémů a funkcionalit, které poskytují.

2.1.1 Současné procesy

Na škole je v současné době nasazen systém Bakaláři, který poskytuje správu uživatelů, zejména pak studentů. Mezi jeho základní funkcionality patří například evidence docházky či klasifikace studentů. Ne všechny procesy, které na škole probíhají, ovšem dokáží Bakaláři pokrýt. Takovéto situace jsou řešeny mimo IS a jsou časově velmi náročné, zároveň zde dochází k častým chybám.

Na základě analýz těchto procesů vzniknou patřičné moduly k novému IS. Administrace bude poskytovat základní správu subjektů, se kterými budou tyto moduly pracovat. Byly identifikovány následující tři procesy, ke kterým vzniknou patřičné moduly.

2.1.1.1 Výběr volitelných předmětů

V současné době probíhá výběr volitelných předmětů v papírové podobě. Studenti obdrží od školy dotazník, kde si zvolí hlavní a vedlejší volitelné předměty, o které mají zájem. Tyto dotazníky jsou následně vyhodnoceny, vytvoří se statistika nejvíce žádaných předmětů a ty se na následující studijní rok vypíšou. Požadavky na zvolené volitelné předměty mohou být různé, ať již dle minimálního počtu zapsaných hodin, či požadovaného počtu zapsaných předmětů. Dále se dotazníky liší dle studijního oboru, studentova ročníku či specializace, jíž se student účastní. Specializace blíže upřesňuje zaměření studijního oboru, ne všechny obory však musí nějakou specializaci mít.

Z tohoto procesu vznikají následující požadavky na administraci:

- správa studentů,
- správa učitelů,
- správa studijních oborů a jejich specializací,
- přiřazování studentů do tříd a správa třídních učitelů.

2.1.1.2 Správa praxí

Na Střední školu Michael navazuje Vyšší odborná škola Michael, jejíž studenti jsou schopni ihned po absolvování nastoupit do zaměstnání v reklamním průmyslu [1]. To je umožněno díky praxím a stážím, jichž se studenti školy účastní. Od studentů se očekává pravidelná docházka, která je evidována v podobě presenčního listu. Na závěr praxe studenti vypracují hodnocení dané firmy, kdy dostanou příležitost zmínit klady a vytknout zápory praxe v dané společnosti. Vedoucí praxe zároveň ohodnotí výkon studentů.

Za účelem správy praxí vznikne v IS nový modul, jenž umožní provádět výše zmíněné úkony. Administrace pro tento modul poskytne základní správu firem a jejich zástupců.

2.1.1.3 Systém odevzdání závěrečných prací

Studenti školy Michael povinně vypracovávají a odevzdávají závěrečné práce, které jsou nedílnou součástí studia. Tento proces v současné době není nijak automatizován a spolu se správou praxí a výběrem volitelných předmětů nemá v systému Bakaláři podporu.

Výsledný modul poskytne kompletní správu závěrečných prací. Studenti si budou moci zvolit jedno z vypsaných zadání, jež bude v systému k dispozici. Bude zde probíhat komunikace s vedoucím práce, úprava a upřesnění zadání a nakonec i závěrečné odevzdání práce. Tento modul na administraci neklade žádné zvláštní požadavky, všechny subjekty již bude mít k dispozici, nebo je bude spravovat sám.

2.1.2 Existující řešení

Na českém trhu v současné době existuje kolem deseti populárních školních IS. Ty jsou zpravidla modulární a některé svou nabídkou modulů pokrývají v podstatě celou agendu vedení školy [2]. Jelikož se ovšem jedná o univerzální nástroje, i přes širokou škálu modulů nemusí být pokryty speciální požadavky škol. Dle [2] patří mezi nejrozšířenější IS následující (seřazeno abecedně):

- aSc Rozvrhy (Applied Software Consultants),
- Bakaláři (Bakaláři Software),
- dm Vysvědčení, dm Evidence, dm Knihovna (dm Software),
- iŠkola (Computer Media),
- RELAX KEŠ (Alis),
- SAS (MP-Soft),
- Škola OnLine (CCA Group).

Dle [3] patří mezi nejpoužívanější IS Bakaláři (39 %), Škola Online (19 %) a dm Software (12 %). Ovšem žádný z výše uvedených IS nemá otevřený zdrojový kód a neumožňuje tvorbu vlastních modulů. Nyní autor analyzuje jejich hlavní charakteristiky, případné výhody a nevýhody a provede jejich přirovnání k současné situaci.

2.1.2.1 Bakaláři

Informační systém Bakaláři patří k nejrozšířenějším u nás, v současné době je provozován na více jak 3 200 školách. Na českém trhu působí již 25 let a pomáhá českým školám zvládat každodenní administrativu. [4] Systém Bakaláři je možné provozovat na školním serveru, nebo zvolit plně cloudové řešení, kdy škola platí pravidelné měsíční poplatky. Existuje i třetí možnost, provozovat systém Bakaláři na serveru školy a příslušnou webovou aplikaci poté v cloudu [5]. Program Bakaláři se skládá z více různých modulů, ke všem datům se lze dostat i online pomocí webové aplikace, která umožňuje jejich rychlejší správu oproti tradičnímu papírování [6]. Mezi základní moduly patří následující:

- evidence žáků a školní matrika,
- internetová žákovská knížka,
- rozvrh hodin, suplování, plán akcí školy, rozpis maturit,

- třídní kniha, tematické plány,
- přijímací zkoušky, knihovna, inventarizace. [6]

Bakaláři poskytují doplňky, jež dále rozšiřují funkcionalitu základních modulů. Jedná se například o evidenci úrazů, evidenci hospitalizací či zápočet praxe zaměstnance. Ovšem ani základní moduly, ani doplňky neposkytují funkcionalitu, kterou škola Michael vyžaduje. Plánuje se tedy souběžný provoz obou IS, kdy nový IS bude doplňovat funkcionalitu Bakalářů.

Hlavní výhodu systému Bakaláři autor této práce spatřuje v jeho popularitě a rozšířenosti, díky čemuž se jedná o skutečně univerzální systém, který až na specifické požadavky splňuje potřeby škol. Za další výhody považuje poskytování cloudového řešení, kdy škola nemusí řešit potřebnou infrastrukturu. Neméně důležitá je uživatelská podpora a nabídka placených služeb, které maximalizují uživatelský komfort.

Bakaláři poskytují mobilní aplikaci pro zařízení s operačním systémem Android a iOS. Dle [7] mají zařízení s operačním systémem Android 74.23% podíl na trhu. Oficiální distribuční službou pro Android zařízení je Google Play Store, jenž byl vyvinut společností Google v roce 2008 [8]. Bakaláři se právě na této platformě těší velké popularitě, v současné době byly nainstalovány na více jak půl milionu zařízeních [9]. Hodnocení ovšem nejsou zcela příznivá, přes 21 % recenzentů udělilo aplikaci nejnižší možné hodnocení [9]. Mobilní aplikaci tedy autor hodnotí jako přínosnou, ovšem je zde jistý prostor pro zlepšení.

Systém Bakaláři ovšem není dokonalý a autor zde nalezl několik nedostatků. Hlavní nevýhodu spatřuje ve skutečnosti, že je systém již 25 let starý a byl navržen dle neméně starého návrhu. To způsobuje problémy, na které si mnozí uživatelé stěžují, ať už se jedná o nevyhovující a nepřehledné uživatelské rozhraní, pomalou odezvu systému či nestabilní chod aplikace, pokud je zařízení připojeno k internetu pomocí bezdrátové sítě [2], [3].

2.1.2.2 Škola OnLine

Škola OnLine je moderní školní IS, jenž umožňuje rychle a efektivně zpracovávat veškerou školní agendu při zachování uživatelského komfortu. Jedná se o webovou aplikaci, což znamená, že je přístupná 24 hodin denně pomocí webového prohlížeče. [10] Škola OnLine je podobně jako Bakaláři modulární systém, jenž umožňuje za poplatek aktivovat jednotlivé funkcionality. Interaktivní ceníky se liší dle typu školy a lze je nalézt stránkách produktu [11].

Škola OnLine obsahuje značné množství modulů, zejména pak:

- jádro systému,
- školní matrika, evidence osob,
- rozvrh a suplování,

- třídní kniha a evidence docházky,
- učební plány.

Ostatní moduly lze nalézt v uživatelské příručce [12]. Tento systém však neumožňuje dodatečné rozšíření funkcionality modulů pomocí doplňků, ani není možné použít moduly vlastní či si nechat takovéto moduly vytvořit.

Výhodu Školy OnLine spatřuje autor práce ve skutečnosti, že je nabízena v podobě cloudového řešení. Tím nevzniká potřeba vlastnit vlastní infrastrukturu, na které by byl systém provozován. Automaticky je tak vyřešen zdlouhavý proces aktualizování aplikace, jelikož uživatel vždy používá aktuální verzi systému. Cloudové řešení přináší další výhody, jako je automatická záloha, zabezpečení dat na stejné úrovni jako bankovní systémy a až 40% úsporu provozních nákladů. [13], [14]

Za slabou stránku produktu lze považovat mobilní aplikaci, o které lze říci, že plně nevyužila svůj potenciál. Aplikace byla nainstalována pouze na zhruba 10 000 zařízeních a nejnižší možné hodnocení udělilo přes 33 % recenzentů [15]. Aplikace App Store od společnosti Apple počet stažení neuvádí, dle [7] lze ovšem předpokládat, že podíl těchto zařízení s aplikací Škola OnLine nebude nijak výrazný.

I přes výrazný počet modulů, které lze zakoupit se ovšem nejedná o ideální náhradu systému Bakaláři, který je na škole v současné době provozován. Takovýto proces by se sebou přinesl nutnost migrace dat a stál by uživatele nezanedbatelný čas, který by byl třeba k seznámení se s novým systémem. Vezme-li se potaz, že by tento systém nepřinesl žádné značné výhody oproti stávajícímu řešení, přechod na tuto platformu je nevýhodný a nežádoucí.

2.1.2.3 dm Software

Produkt dm Software je interaktivní IS pro školy, jenž je poskytován výhradně jako cloudové řešení. Podobně jako Škola OnLine je uživatelům dostupný 24 hodin denně pomocí webového prohlížeče či mobilní aplikace. Software je vhodný pro mateřské, základní, střední a vyšší odborné odborné školy k vedení školní matriky a elektronické agendy spojené s vedením školy. [16]

Jedná se o modulární systém, kde každý modul poskytuje patřičnou funkcionalitu. Systém je nabízen v podobě balíčků, v současné době jsou k dispozici celkem 3 balíčky. Jedná se o balíčky Základ, Standard a Komplet. Ceny za jednotlivé balíčky jsou stanoveny dle platného ceníku. [17], [18]

Systém dm Software poskytuje podobné základní moduly jako IS Škola Online, celkový počet modulů je však omezen. Školy, které mají více různých požadavků na IS pravděpodobně zvolí konkurenční řešení. Toto tvrzení z části podporuje fakt, že mobilní aplikace dm Software byla nainstalována pouze na zhruba 1000 zařízeních [19].

Z analyzovaných řešení se dm Software jeví jako nejméně vhodný. To je způsobeno omezenou nabídkou modulů a způsobem, jakým je nabízen. Pokud škola potřebuje jednu konkrétní funkcionalitu, která je ovšem součástí vyššího balíčku, nelze tento modul dokoupit samostatně a je třeba přejít na vyšší tarif.

2.1.2.4 Shrnutí

Byla provedena analýza dle [3] třech nejvýznamnějších školních IS. Jedná se o systémy Bakaláři, Škola OnLine a dm Software. Všechny výše zmíněné systémy jsou modulární, skládají se tedy z jednotlivých modulů, které rozšiřují základní funkcionalitu systému. Žádné z těchto řešení ovšem není k dispozici s otevřeným zdrojovým kódem a neposkytuje možnost vytvoření vlastního modulu, který by umožnil pokrýt všechny speciální potřeby školy.

Bakaláři nabízejí jak cloudové, tak lokální řešení, kdy je systém provozován na infrastruktuře patřící škole. Funkcionalitu nabízených modulů lze dále rozšiřovat doplňkovými aplikacemi, které jsou zpravidla bezplatně k dispozici [6]. IS Bakaláři je v současné době na škole Michael provozován, avšak neposkytuje všechny požadované funkcionality. Tyto funkcionality nelze získat ani doplňkovými aplikacemi, bude tedy vytvořen nový IS, který bude provozován souběžně s Bakaláři a bude poskytovat chybějící funkcionality.

Škola Online a dm Software jsou dva sobě velice podobné IS. Oba jsou nabízeny pouze jako cloudové řešení, kdy škola nemusí vlastnit ani spravovat žádnou infrastrukturu. Přístup do systémů je umožněn pomocí webového prohlížeče či mobilní aplikace. Žádný z výše dvou uvedených systémů nepodporuje funkcionality, které škola Michael vyžaduje, případný přechod na tyto systémy by tedy nepřinesl žádné výhody.

2.2 Použitelné technologie

Sekce použitelné technologie se zaměřuje na představení a analýzu technologií, které jsou k dispozici pro implementaci projektu na straně serveru. Autor práce se zde bude zabývat programovacími jazyky, druhy databází a jejich populárními zástupci. Uvedeny budou jejich hlavní charakteristiky, představeny klady a zápory a bude provedeno jejich vzájemné porovnání.

Volba konkrétních technologií, jež budou použity pro realizaci IS bude provedena v sekci 3.1 na základě funkčních a nefunkčních požadavků analyzovaných v sekci 2.3 a případů užití analyzovaných v sekci 2.4. V sekci 3.1.4 budou představeny prezentační technologie, které budou použity zejména na straně klienta.

2.2.1 Programovací jazyky

V dnešní době jsou k dispozici stovky programovacích jazyků, kdy každý jazyk má své specifické využití a typické případy užití. Programovací jazyk bývá

typicky navržen k použití na straně klienta či na straně serveru, existují zde ovšem i výjimky, kdy programovací jazyk může být použit na obou stranách. Typickým zástupcem takového jazyka bývá uváděn Javascript, který lze použít v internetovém prohlížeči na straně klienta či na straně serveru, typicky v běhovém prostředí Node.js [20].

Dle [21] patří mezi oblíbené programovací jazyky na straně serveru následující:

- C#,
- Go,
- Java,
- Node.js (JavaScript),
- Python,
- PHP,
- Ruby.

Z výše uvedených jazyků vybral autor práce 4 zástupce a nyní provede jejich analýzu. Zaměří se mimo jiné na jejich historii, specifické vlastnosti či silné a slabé stránky.

2.2.1.1 PHP

První blíže představenou a analyzovanou technologií bylo zvoleno PHP. PHP je interpretovaný, široce používaný otevřený skriptovací jazyk, jenž má mnoho využití, především se pak jedná o vhodný prostředek k vývoji webových aplikací.

PHP tak, jak ho známe dnes, je nástupcem PHP/FI (formulářový interpret), jenž napsal v roce 1994 Dánsko–Kanadský programátor Rasmus Lerdorf. První verze PHP byla veřejnosti poskytnuta k dispozici poprvé v roce 1995 a než ho v roce 1998 nahradilo PHP 3.0, bylo použito na zhruba 60 000 doménách, což v té době tvořilo zhruba 1 % všech domén. PHP 3.0 bylo zhruba rok po představení nahrazeno PHP 4.0, jenž v roce 2004 ustoupilo před PHP 5.0. [22] V současné době je aktuální verze PHP 7.2.4, která mimo jiné přinesla opravu značného množství chyb [23].

Dle [24] vidíme klesající popularitu PHP mezi programovacími jazyky, oproti minulému roku se PHP propadlo o jednu pozici ze 6. na 7. místo. Údaje z roku 2015 dle [25] tvrdí, že přes 75 % webových aplikací je napsáno na straně serveru v jazyce PHP. Lze očekávat, že díky mírně klesající popularitě PHP bude aktuální číslo menší.

Použití PHP s sebou přináší mnohé výhody, za zmínku stojí především následující:

- nízká vstupní bariéra,
- podpora funkcionálního i objektového programování,
- široký ekosystém,
- veliké množství dostupných knihoven,
- podpora automatizujících nástrojů,
- početná vývojářská komunita. [26]

PHP ovšem není bez chyb, jedna z významnějších vyplývá z podstaty interpretovaných jazyků. Ty je nutné před spuštěním zkompilovat do operačního kódu, jenž může být zpracován procesorem počítače. Tato kompilace ovšem trvá nezanedbatelný čas, po který musí klient čekat. Opakované kompilování jde do určité míry omezit uložením zkompilovaného operačního kódu do mezipaměti počítače, ovšem i tato operace zabere čas, zvláště pak u větších aplikací. Díky nízké vstupní bariéře může být také problémové nalezení opravdového PHP specialisty, než tomu je u ostatních programovacích jazyků. [26]

2.2.1.2 Java

Java je vyšší programovací jazyk vyvinutý společností Sun Microsystems. Syntaxe Javy připomíná C++, jedná se však o čistě objektově orientovaný jazyk. Java je více striktní než C++, všechny proměnné a funkce musí být explicitně deklarované. [27]

„*Zdrojový kód je nejprve přeložen do tzv. mezikódu, kterému se u Javy říká bytecode. Jedná se v podstatě o strojový (binární) kód, který má ale o poznání jednodušší instrukční sadu a přímo podporuje objektové programování. Tento mezikód je potom díky jednoduchosti relativně rychle interpretovatelný tzv. virtuálním strojem (tedy interpretem, v případě Javy je to tzv. JVM - Java Virtual Machine). Výsledkem je strojový kód pro náš procesor.*“ [28]

Java vznikla na začátku 90. let pod týmem vedeným Jamesem Goslingem spadajícím pod společnost Sun Microsystems, dnes vlastněnou společností Oracle. Java byla původně navržena pro použití v mobilních zařízeních, avšak v roce 1996 při oznámení verze 1.0 se její zaměření přesunulo k využití na internetu. Nicméně od verze 1.0 se Java dočkala mnoha aktualizací, jako příklad lze uvézt 2SE 1.3 v roce 2000, J2SE 5.0 v roce 2004, Java SE 8 v roce 2014 či Java SE 10 v roce 2018. [29]

Java je dle [24] nejpopulárnější programovací jazyk, na horních pozicích žebříčku se drží již zhruba od roku 2003. V prostředí webových aplikací je popularita Javy slabší, dle [25] je Java použita jako programovací jazyk na

straně serveru pouze v malém měřítku, zde dominuje PHP s více jak 75 %. Za zajímavost můžeme považovat skutečnost, že v roce 2015 bylo dle [25] 83.3 % webových aplikací s Javou jako programovacím jazykem na straně serveru označeno za aplikace s vysokou návštěvností, kdežto pro PHP tak bylo označeno pouze 14.2 % aplikací.

Java byla navržena s několika klíčovými principy, které lze považovat za přednosti tohoto jazyka:

- snadná použitelnost,
- spolehlivost,
- bezpečnost,
- platformní nezávislost. [29]

I přes tyto výhody ovšem Java často nebývá při výběru programovacího jazyka mezi prvními kandidáty. Může za to skutečnost, že firmy většinou zvolí jeden ze skriptovacích jazyků, jenž se nemusí při každé změně v kódu ručně kompilovat. To šetří čas při vývoji a umožňuje rychlejší vývoj produktu a jeho uvedení na trh. [30]

2.2.1.3 Python

Python je interpretovaný, výkonný a objektově orientovaný programovací jazyk vyšší úrovně s dynamickou sémantikou. Díky jeho vysoce postavené datové struktuře kombinované s dynamickým typováním a vázáním se jedná o velice atraktivní nástroj pro rapidní vývoj aplikací. Je také vhodný pro použití jako skriptovací jazyk či jako jazyk pro propojení více komponent dohromady. [31]

Python vznikl v prosinci 1989 jako dílo nizozemského programátora Guido van Rossuma. Jedná se o programovací jazyk vzniklý na základě ukončeného jazyka ABC holandského výzkumného institutu matematiky a informatiky (CWI). Python byl na svém autorovi závislý až do roku 2000, kdy byl vydán pod týmem BeOpen Python Labs. Pythonu verze 2.7 je dnes stále aktuální a má zaručenou podporu do roku 2020, kdy bude jeho podpora ukončena ve prospěch Pythonu 3.0 vydaného v roce 2008. Jedna z nejdůležitějších vlastností verze 3.0 je kompletní absence zpětné kompatibility, jednalo se o kontroverzní rozhodnutí založené na myšlence vyčištění konstruktů a modulů daného jazyka. [32]

Mezi hlavní přednosti jazyka patří jednoduchost, jednoduchá a přehledná syntaxe ulehčující čtení a umožnující efektivnější údržbu programu [31]. Python je podporován na všech předních platformách včetně Windows, Mac OS, Linuxových distribucí a má i neoficiální podporu na mobilních zařízeních.

Mezi slabé stránky patří pomalost jazyka, špatná podpora mobilních zařízení a omezené možnosti práce s databázemi. Python zároveň není vhodný jazyk pro procesy s vysokou paměťovou náročností a nejedná se ani o ideální volbu k nasazení na systémech s více procesory či výpočetními vlákny. [33]

2.2.1.4 Node.js (JavaScript)

Node.js je asynchronní, JavaScriptové, událostmi řízené běhové prostředí (anglicky runtime environment) navržené ke tvorbě škálovatelných webových aplikací [34]. Aplikace pro Node.js jsou napsány v JavaScriptu a mohou běžet v běhovém prostředí Node.js na platformách Windows, OS X a Linux. Node.js také poskytuje bohatou knihovnu různých JavaScriptových modulů, které do značné míry zjednodušují vývoj webových aplikací [35].

Node vznikl jako dílo autora Ryana Dahla v roce 2009, jedná se tedy o poměrně nový a moderní nástroj. Původní verze byly podporovány pouze na operačních systémech Linux a OS X, na Windows se Node.js dočkal podpory až v roce 2011. Od doby jeho vzniku si prošel mnoha změnami a dle [36], [37] byl adoptován velkým množstvím 500 předních technologických firem.

Značnou výhodu oproti ostatním technologiím představuje jeho architektura, která byla navržena za účelem minimalizace počtu vstupních a výstupních (anglicky I/O) operací. Právě díky této vlastnosti byl Node.js zvolen několika technologickými giganty, jako jsou například Google, PayPal či LinkedIn pro operace vyžadující velký počet I/O operací.

Node.js s sebou ovšem přináší i jisté nevýhody, můžeme zmínit například nevyhovující práci s relačními databázemi. Node.js také není vhodný k provádění výpočetně náročných operací, jeho výhoda spočívá v nízkém počtu vstupně výstupních operací, díky čemuž se jedná o vhodný nástroj pro použití ve webových aplikacích. Programátor bez dostatečné znalosti JavaScriptu může také čelit konceptuálním problémům, které s sebou Node.js přináší. [38]

2.2.1.5 Shrnutí

V této sekci byla provedena analýza čtyř populárních programovacích jazyků, které se dle [21] používají v současné době na straně serveru. U těchto jazyků byly představeny jejich základní charakteristiky, historie vývoje a výhody a nevýhody. Každý z jazyků má své typické případy užití a specifické oblasti nasazení, kde patří v komunitě vývojářů mezi velmi populární nástroje.

PHP je velmi populární volba u produktů, které je třeba rychle uvést na trh či u webových aplikací, kde se neočekává vysoká návštěvnost. I přes skutečnost, že je PHP dle [25] použito na více jak 75 % všech webových aplikacích, pouze 14.2 % z nich lze považovat za webové stránky s vysokou návštěvností. Opačný příklad tvoří Java, jenž je nasazena pouze na malém množství webových aplikací, ovšem 83.3 % z nich lze klasifikovat jako webové aplikace s vysokou návštěvností [25].

Python je díky své jednoduchosti využíván pro rapidní vývoj aplikací či jako jazyk pro propojení více komponent webové aplikace dohromady. Nemá ovšem oficiální podporu na mobilních zařízení a nejedná se ani o ideální volbu pro práci s relačními databázemi. Pro webové aplikace s vysokým počtem vstupně výstupných operací se jeví jako ideální technologie Node.js, který je využíván v řadě technologických gigantů, jako je Google, PayPal či LinkedIn.

2.2.2 Typy Databází

Databáze je kolekce informací organizovaná takovým způsobem, jenž umožňuje snadný přístup k těmto informacím, jejich správu a aktualizování [39]. Všechny webové aplikace, kromě těch s neměnným obsahem, obsahují nějaký typ databáze, jenž umožňuje dynamicky upravovat obsah bez nutnosti zásahu do zdrojového kódu dané aplikace.

Pokud jde o výběr databáze, jedním z největších rozhodnutí je výběr relační (SQL) nebo nerelační (NoSQL) datové struktury [40]. Následující sekce se věnují právě těmto dvěma typům databází, určí jejich charakteristiky a představí nejpopulárnější zástupce.

2.2.2.1 Relační Databáze (SQL)

SQL znamená strukturovaný dotazovací jazyk a pochází z anglického Structured Query Language (SQL). Jedná se o standardní jazyk pro dotazování a manipulování s databázemi [41]. Pro SQL databáze je typické ukládání dat do tabulek, kde je každý záznam reprezentován jedním řádkem a jeho atributy jsou rozděleny do patřičných sloupců. Struktura relační databáze umožňuje propojit informace z více tabulek za pomoci cizích klíčů, které odkazují na jednoznačně identifikované záznamy v tabulkách [42]. SQL je mocný nástroj, jedná se o vhodnou volbu obzvláště pro tvorbu komplexních dotazů, což z něj činí jednu z nejvíce všestranných a nejčastěji používaných technologií,

Na druhou stranu, SQL může být podstatně omezující. Při práci s SQL je vyžadováno dodržení předem definované struktury dat, jejíž případná změna znamená nutnost provést úpravy samotné databáze. [40]

Dle [43] dominují nejpopulárnějším databázím právě SQL databáze, kde první tři pozice obsadily MySQL, SQL Server a PostgreSQL. MySQL je nejpopulárnější databázový systém SQL s otevřeným zdrojovým kódem na světě, jenž byl vyvinut společností Oracle [44]. MySQL je k dispozici zdarma pod veřejnou licencí GPL, pokud ovšem firma vyžaduje profesionální uživatelskou podporu, je nutné zakoupení licence od společnosti Oracle. [45]

2.2.2.2 Nerelační Databáze (NoSQL)

Termín NoSQL je nejčastěji považován za zkratku Not Only SQL, již lze přeložit jako „nejen strukturovaný dotazovací jazyk“ [46]. NoSQL je třída systémů pro správu databází (anglicky DBMS), která nutně nemusí splňovat všechna pravidla relačních DBMS a která nevyužívá tradiční SQL pro manipulování s daty.

„*Systémy založené na NoSQL se typicky používají ve velmi rozsáhlých databázích, které jsou obzvláště náchylné na problémy s výkonem způsobené omezeními SQL a relačního modelu databází.*“ [46] přeložil Tomáš Trbola

Výhodou relačních databází je možnost využívání transakcí, jež zaručují konzistenci dat uložených v databázi. NoSQL ovšem nemusí dodržovat striktní pravidla, kterými se tyto transakce musí řídit. To poskytuje relaxovaný model konzistence, který s sebou spolu s masivní škálovatelností přináší vyšší výkon a dostupnost [47].

Dle [43] se MongoDB stala nejpopulárnější NoSQL databází v roce 2018, v žebříčku všech populárních databází se poté umístila na 4. místě. MongoDB je databáze s otevřeným zdrojovým kódem, na rozdíl od relačních databází však nepoužívá pro ukládání dat tabulky, ale kolekce a dokumenty. Dokumenty obsahují množiny párů klíčů a hodnot, jež tvoří základní jednotku dat v MongoDB. Kolekce obsahují množiny dokumentů a fungují jako ekvivalent tabulek v relačních databázích. [48]

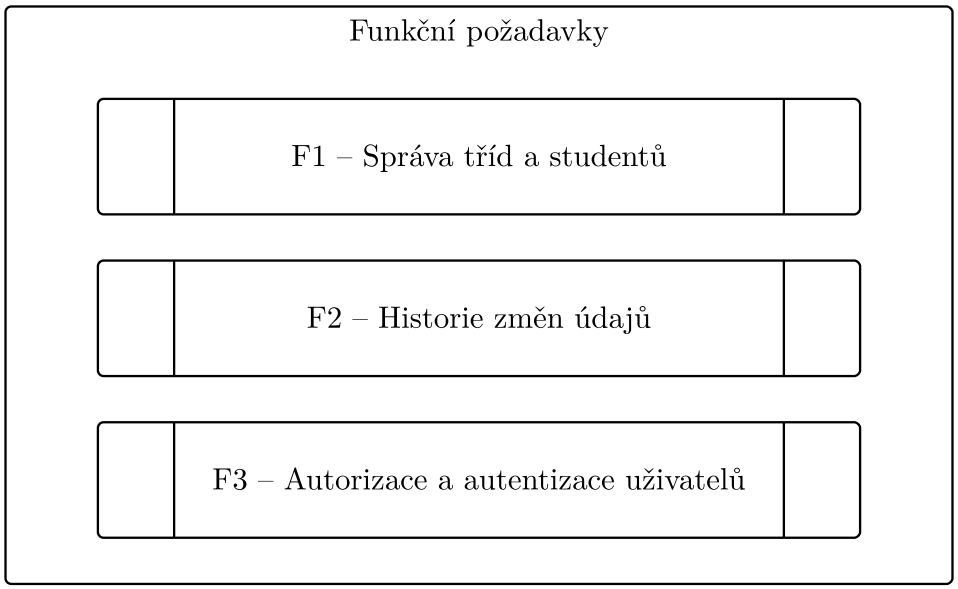
2.3 Funkční a nefunkční požadavky

Při provádění analýzy nároků na systém se setkáváme s pojmy funkční a nefunkční požadavky. Funkční požadavky stanovují, jaké funkcionality by měl systém podporovat, kdežto nefunkční požadavky kladou na systém omezení a upřesňují, jak by měl systém dané akce vykonávat [49]. Následuje analýza funkčních a nefunkčních požadavků, které jsou na administraci kladeny.

2.3.1 Funkční požadavky

Na základě funkčních požadavků vzniknou v sekci 2.4 konkrétní případy užití. Na administraci jsou kladeny následující funkční požadavky, jež zobrazuje diagram na obrázku 2.1.

F1 – Správa tříd a studentů Administrace bude poskytovat základní správu tříd a studentů. Bude možné vytvářet a upravovat třídy, či k nim přiřazovat třídní učitele a studenty. Třídy z předchozích ročníků budou archivovány. Dále administrace poskytne rozhraní pro převod tříd do vyšších ročníků, kdy bude pro každou třídu vytvořen záznam v archivu a následně bude třída přesunuta do vyššího ročníku.



Obrázek 2.1: Funkční požadavky

Administrace dále poskytne správu studentů, ty bude možno vytvářet, upravovat a případně jim odebrat studentský status, kdy dojde k deaktivaci studenta. Bude umožněno přesouvat studenty do tříd či jim přidělovat studijní obory. Studenty bude možno vytvářet ručně či je importovat z IS Bakaláři, pro tento způsob poskytne administrace speciální uživatelské rozhraní.

F2 – Historie změn údajů Pro účely vedení školní agendy je zapotřebí, aby administrace zaznamenávala historii změn důležitých údajů. Pro každého uživatele bude vedena historie jeho jmen a příjmení pro případ, že by došlo ke změně těchto údajů. U studentů bude vedena historie tříd, studijních oborů a přístupových kódů. Bude zaznamenávána historie třídních učitelů. Tyto záznamy bude možné dohledat na profilu daného učitele či na detailu třídy.

Archivace dalších údajů není vyžadována. Jedná se například o historii rolí uživatelů či o historii úkonů, jež uživatel administrace vykonával.

F3 – Autorizace a autentizace uživatelů Administrace poskytne autorizaci a autentizaci uživatelů. Bude se jednat o správu uživatelských rolí a správu přístupových možností do systému. Administrace se člení do několika sekcí, kdy uživatel potřebuje příslušné oprávnění pro přístup do této sekce.

Budou poskytnuty dva způsoby přístupu do informačního systému. Základní způsob představují uživatelské účty. Alternativním způsobem jsou jednorázové přístupové kódy, jejichž použitím uživatel získá přístup do systému s omezenými pravomocemi.

2.3.2 Nefunkční požadavky

Byly zjištěny následující nefunkční požadavky, které jsou na administraci kladeny.

N1 – Multijazyčná lokalizace Na škole Michael učí jak čeští tak zahraniční učitelé a profesoři. Administrace tedy poskytne možnost zvolit si preferovaný jazyk rozhraní, při spuštění IS bude podporován český a anglický jazyk. Do budoucna se očekává přidání podpory více lokalizací.

N2 – Modulární architektura IS se bude skládat z modulů, které budou poskytovat patřičné funkcionality. Administrace by tedy měla dodržovat takzvané nejlepší praktiky (anglicky best practices) jazyka a frameworku, jenž bude použit pro implementaci prototypu. Dále se předpokládá použití snadno čitelného a udržitelného kódu.

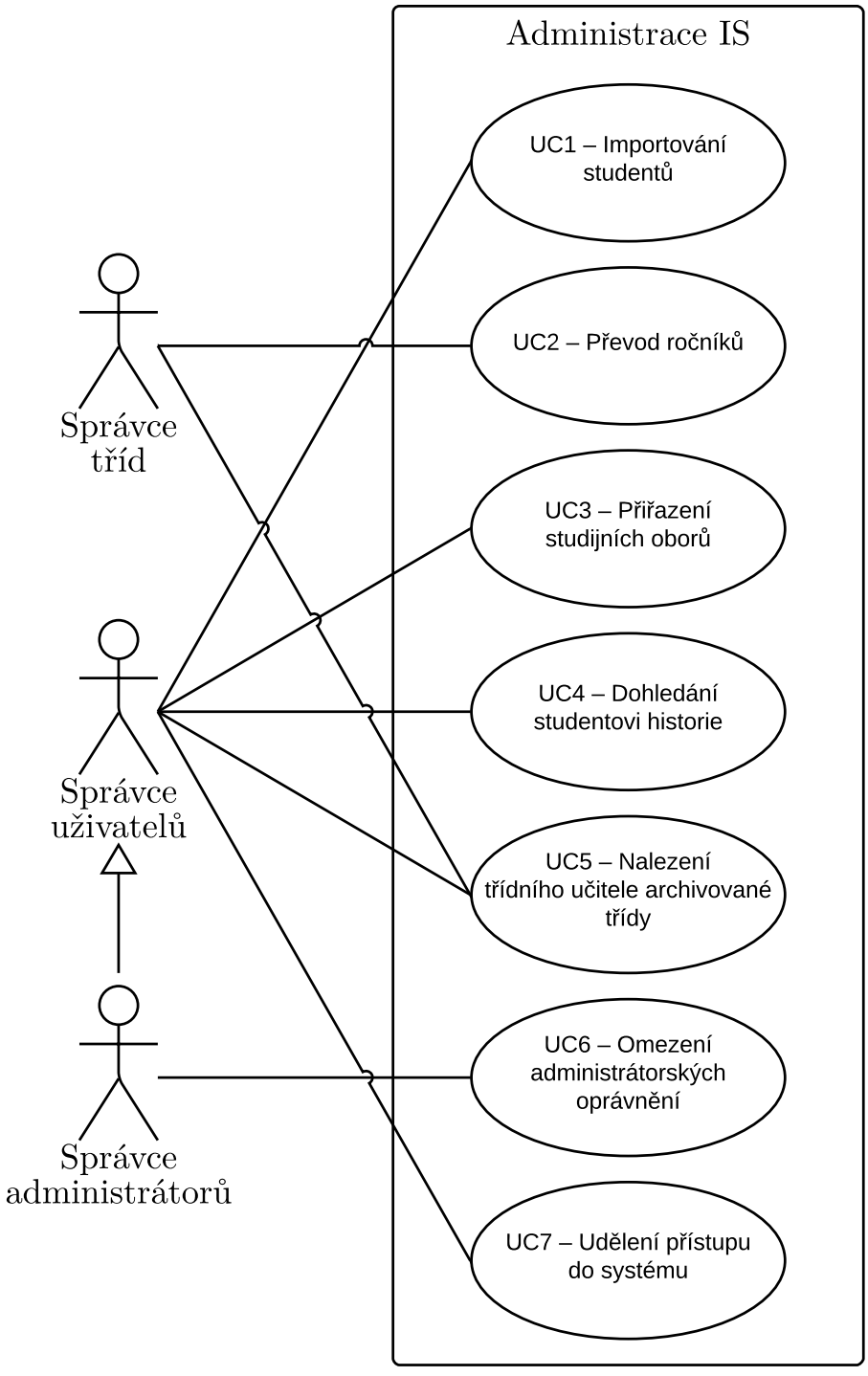
N3 – Zaměření na mobilní zařízení Očekává se, že bude systém využíván ve třídách na tabletech a dalších mobilních zařízeních. Administrace by tedy měla být zaměřena především na tato zařízení a měla by poskytnout responsivní rozhraní spolu s tradičními ovládacími prvky, na které jsou uživatelé těchto zařízení zvyklí.

2.4 Případy užití

Případ užití je sada několika akcí, které vedou k dosažení stanoveného cíle. Definuje tedy jednu funkcionalitu, kterou by měl systém umět. Případ užití již ovšem neřeší vnitřní logiku aplikace. [50]

Tato sekce popisuje jednotlivé případy užití, které vznikly na základě funkčních požadavků. Do administrace má přístup pouze uživatel s rolí administrátor. Administrátoři se ovšem dále člení do skupin dle přiřazených rolí. Diagram na obrázku 2.2 zachycuje případy užití a jejich aktéry.

Je důležité, aby byl každý funkční požadavek realizován alespoň jedním případem užití. Absence takovéto realizace by znamenala situaci, kdy systém neposkytuje požadovanou funkcionalitu. Pokrytí všech funkčních požadavků zobrazuje tabulka 2.1.



Obrázek 2.2: Diagram případů užití

Tabulka 2.1: Pokrytí funkčních požadavků

	UC1	UC2	UC3	UC4	UC5	UC6	UC7
F1	+	+	+				
F2				+	+		
F3						+	+

UC1 – Importování studentů Případ užití importování studentů poskytne administrátorovi možnost importovat studenty ze systému Bakaláři. Importovaná data budou ve formátu XML souborů, kdy bude pro každou třídu vytvořen jeden soubor se seznamem studentů. Bude možné importovat následující studentovy atributy:

- studentovo jméno a příjmení,
- studentova třída,
- studentův email,
- synchronizační klíč.

Jméno, příjmení, třída a synchronizační klíč jsou povinné položky, které musí obsahovat každý záznam. Synchronizační klíč je libovolný unikátní řetězec vygenerovaný systémem Bakaláři. Může se jednat například o vnitřní identifikační kód studenta či například o jeho rodné číslo. Administrace tyto klíče poskytne v čitelné podobě za účelem jejich snadné kontroly či změny v budoucnu.

UC2 – Převod ročníků Administrace poskytne rozhraní pro převod tříd do vyšších ročníků. Při zahájení procesu bude pro všechny aktivní třídy rozhodnuto, zda-li bude daná třída pokračovat, či zda-li bude ukončena. Administrátor bude mít možnost před dokončením převodu rozhodnout jinak. Pokud administrátor nastaví více třídám stejný název, dojde ke sloučení těchto tříd, jejich studentů a třídních učitelů.

UC3 – Přiřazení studijních oborů Studenti si během studia vybírají studijní obor. Administrátor může obory přiřazovat studentům jednotlivě či hromadně. Možnost přiřazení oboru studentovi lze nalézt na profilu daného studenta či na detailu třídy, jíž se účastní. Na detailu této třídy lze také nalézt rozhraní pro hromadné přiřazení oborů.

UC4 – Dohledání studentovy historie Případ užití dohledání studentovy historie umožňuje administrátorovi sledovat průběh studia daného studenta. Pro účely vedení školní agendy je nezbytné, aby bylo možné dohledat

historii studentových tříd a studijních oborů, kterých se účastnil. Jsou evidovány případy, kdy student změnil jméno či příjmení během studia. Administrace tedy poskytne historii všech jmen a příjmení daného studenta. Dále bude pro všechny studenty vedena historie přístupových kódů. Bude tedy možné dohledat, kdy byl daný kód vygenerován a případně aktivován.

UC5 – Nalezení třídního učitele archivované třídy Systém umožní nalézt třídního učitele již archivované třídy. Pokud uživatel zná název dané třídy a její ročník, lze tento údaj nalézt na profilu dané třídy. V opačném případě, pokud administrátor zná jméno učitele a chce nalézt název třídy, kterou v daném roce učil, může tak učinit na profilu tohoto učitele.

UC6 – Omezení administrátorských oprávnění Administrace poskytne možnost omezení administrátorských pravomocí. Bude tedy možné omezit administrátorovi jeho práva pro vykonávání pouze omezeného počtu akcí. Administrace poskytne následující oprávnění, která lze administrátorům přiřadit.

- správa administrátorů,
- správa tříd,
- správa firem,
- správa uživatelů,
- správa studijních oborů,
- super admin – plný přístup.

Tato oprávnění je možné libovolně kombinovat. Jednotlivé sekce administrace vyžadují patřičná oprávnění, bez nichž nebude administrátorovi umožněn přístup do dané sekce. Administrátor nemusí mít žádné administrátorské oprávnění, v takovém případě se jeví jako administrátor, ale nemá žádné pravomoce a nemůže v administraci vykonávat žádné akce.

UC7 – Udělení přístupu do systému Administrace poskytne dvě možnosti přístupu do IS. Základní možností jsou přístupové účty. Každý uživatel nemusí mít žádný, ale může mít i více přístupových účtů. Tyto účty mohou být permanentní nebo časově omezené.

Druhou možnost přístupu představují jednorázové přístupové kódy. Tyto kódy mohou být vygenerovány pouze pro studenty a poskytnou jednorázový přístup do systému s omezenými oprávněními. Kódy bude možné generovat pro každého studenta zvlášť, či pro celou třídu najednou. Administrace poskytne možnost vytisknutí těchto kódů či jejich odeslání na emaily studentů.

2.5 Doménový model

Následující doménový model vznikl za účelem vytvoření přehledu o jednotlivých částech systému a jejich provázanosti. Tento model zároveň tvoří základní předlohu pro vznik databáze, která bude představena v části 3.1.3.

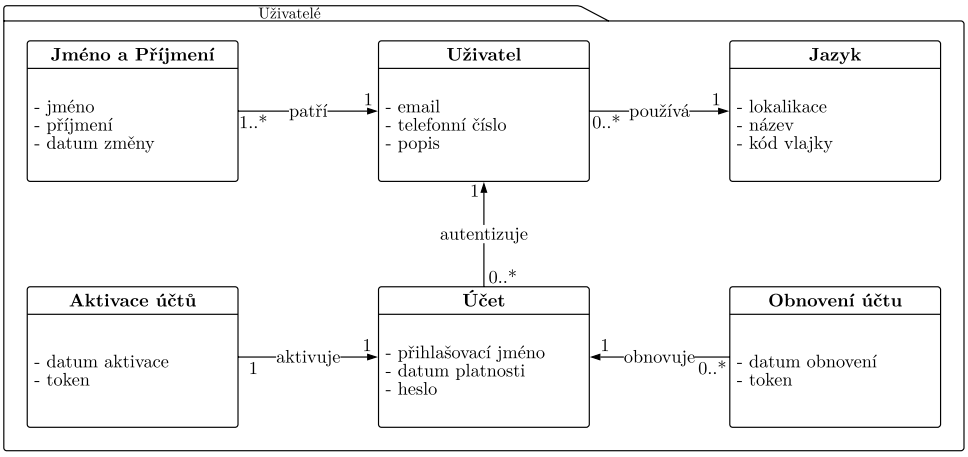
2.5.1 Uživatelé

Entita `Uzivatel` tvoří jeden z hlavních pilířů systému, díky čemuž má vazbu na větší počet entit. Tato a následující dvě sekce jsou proto zaměřeny na tuto entitu a její vztahy.

Funkční požadavek F2, jenž byl představen v sekci 2.3.1 vyžaduje, aby bylo možné dohledat případné změny jmen a příjmení uživatelů. Z tohoto důvodu vznikla entita `Jméno a Příjmení`, na kterou musí mít vazbu každý uživatel.

Jazyky, jež jsou podporovány systémem, jsou reprezentovány pomocí vlastní entity. Tato entita obsahuje kód jazyka, název a kód vlajky pro daný jazyk. Překlady pro daný jazyk jsou uloženy v konfiguračních souborech, což umožňuje jejich jednoduché upravování.

Uživatelům bude umožněn přístup do systému, přihlašovací údaje pro každého uživatele bude zprostředkovávat entita `Účet`. Uživatel nemusí mít účet žádný, ale může jich mít i více. Pro každý účet lze nastavit, zda-li je bez data expirace, či zda-li je časově omezený. Toho lze využít například v situaci, kdy bude chtít uživatel dočasně propůjčit svůj účet cizí osobě.



Obrázek 2.3: Doménový model – uživatelé

2.5.2 Uživatelské role

Vzhledem k rozsahu systému bude každá role reprezentována vlastní entitou, jedná se pak zejména o následující:

- administrátor,
- učitel,
- student,
- zástupce firmy.

Uživateli poté budou přiřazeny role na základě existence vztahů s těmito entitami. Tyto entity budou mít mimo jiné atribut `aktivní`, který bude určovat, zda-li je daná role v současné době aktivní. Jednotlivé části systému poté mohou pracovat přímo s těmito entitami.

2.5.3 Studenti

Entita `Student` představuje druhý pilíř systému, oproti entitě uživatele má tedy neméně vztahů. U studenta se rozlišuje, zda-li v současné době aktivně studuje či nikoli. Dále entita `Student` obsahuje synchronizační klíč, který je použit při importování studentů. Student tento klíč mít nemusí, avšak poté nelze provést jeho synchronizaci. Vztah entity `Student` a ostatních entit je zobrazen v diagramu na obrázku 2.4.

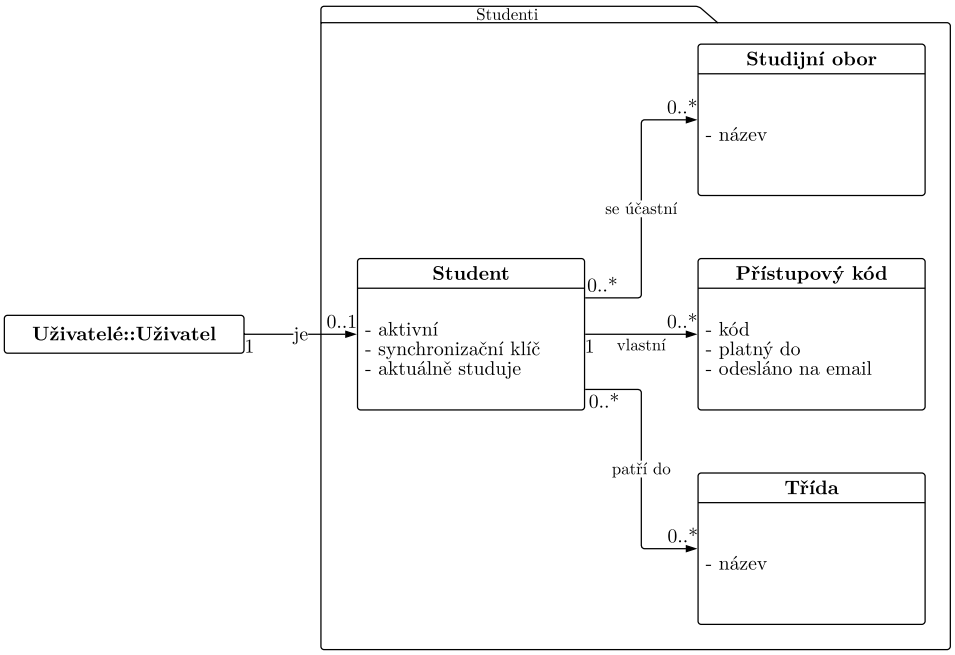
2.5.4 Převod ročníků

Převádět třídy do vyšších ročníků může pouze administrátor s udělenou rolí správa tříd. Entita `Migrace` reprezentuje tyto převody ročníků. Pro každou třídu, jež je v současné době aktivní, je při převodu vytvořen záznam, jenž je reprezentován entitou `Migrační záznam`. Tento záznam odkazuje na stávající třídu a pokud byla tato třída převedena, tak také na třídu nově vzniklou. Každý záznam obsahuje atribut `určeno k archivaci`, jenž určí, zda-li bude daná třída převedena či nikoliv.

2.5.5 Importování studentů

Za účelem snadnějšího pochopení doménového modelu importování studentů je vhodné si nejprve popsat, jak tento proces probíhá. Tento doménový model zobrazuje diagram na obrázku 2.5.

Importování začíná vytvořením nového procesu zvaného pouze import, v systému může být těchto procesů najednou aktivních více. Na detailu importu lze nalézt přehled importovaných tříd a studentů, kteří již byli načteni.

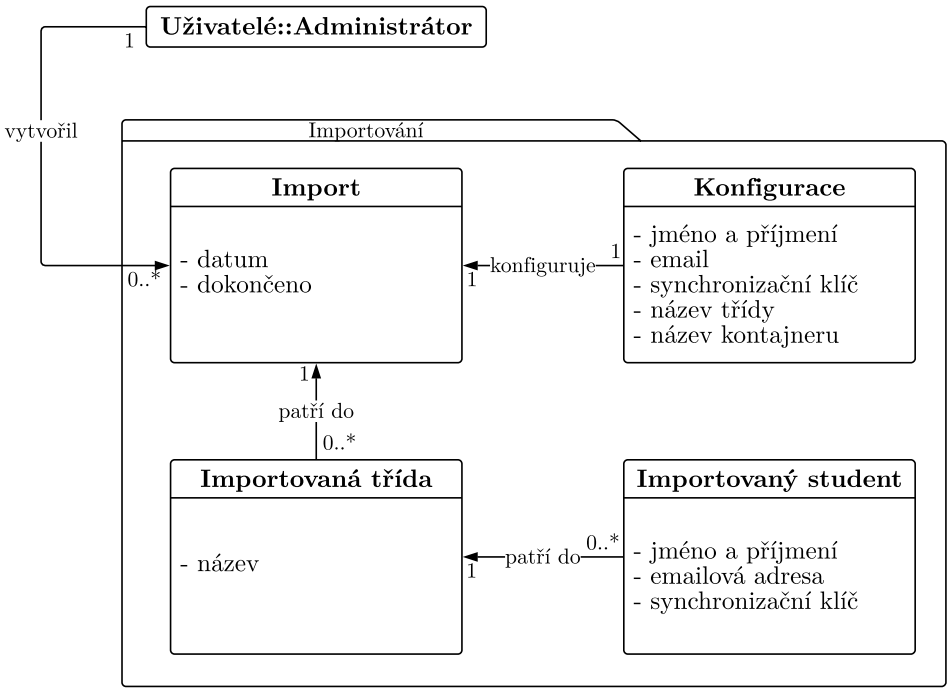


Obrázek 2.4: Doménový model – studenti

Načtení nových studentů probíhá nahráním XML souboru, jenž obsahuje importovaná data. Po zpracování těchto dat jsou vytvořeny záznamy pro jednotlivé studenty a jejich třídy. Proces nahrání souboru lze opakovat, což umožňuje načíst studenty z více různých souborů. Po dokončení importu dojde k aktualizování studentů za pomoci synchronizačních klíčů.

Tento proces je reprezentován entitou `Import`, která má vazbu na administrátora, jenž daný proces zahájil. Entita importu obsahuje čas provedení akce a status, zda-li již byl proces dokončen. Entita `Importovaná třída` obsahuje záznamy pro jednotlivé studenty, jež jsou reprezentovány entitou `Importovaný student`.

Zvláštní pozornost si zde zaslouží především entita `Konfigurace`, která je automaticky vytvořena po založení importu. Její počáteční hodnoty jsou načteny z konfiguračních souborů IS a lze je libovolně upravovat dle potřeby. Účelem této entity je poskytnout názvy jednotlivých elementů, jež jsou obsaženy v importovaných XML souborech, pokud by se měla jejich struktura změnit. Účel všech proměnných v entitě `Konfigurace` lze poznat z jejich názvu, výjimku zde tvoří proměnná `název elementu`, která určuje název elementu, jenž obsahuje samotné údaje o studentovi.



Obrázek 2.5: Doménový model – importování studentů

Kapitola 3

Návrh

3.1 Volba technologií

V předchozí kapitole v sekci 2.2 byly představeny technologie, jež lze použít pro implementaci prototypu na straně serveru. V následující sekci zvolí autor této práce vhodné technologie. Po stanovení programovacího jazyka dojde k výběru jednoho z populárních frameworků pro daný jazyk. Na základě požadavků kladených na systém vybere autor vhodnou databázi. Poslední sekce představí technologie, jež budou použity pro přenos dat ke klientovi a jejich následnou prezentaci.

3.1.1 Programovací jazyk

V sekci 2.2.1 byly představeny 4 populární programovací jazyky, konkrétně PHP, Java, Python a Node.js (JavaScript). Při výběru vhodného jazyka je třeba zohlednit několik kritérií, tato se stanoví na základě charakteristik výsledného IS, kde se bude zkoumat rozsah projektu a jeho nároky na systémové prostředky. Dále bude brán ohled na prostředí, ve kterém bude systém realizován a na možnosti budoucího rozšíření systému.

Připravovaný IS je modulární, lze tedy předpokládat a počítá se s jeho budoucím rozšiřováním. Jazyk, v němž bude systém implementován by měl být dobře čitelný, lehce udržitelný a měl by být vhodný pro realizaci modulární architektury. Je tedy nezbytná podpora programovacích struktur, jako jsou například rozhraní (anglicky interfaces). Jazyk by měl být dále vhodný ke psaní znovu použitelných komponent.

Plánovaný počet dlouhodobě aktivních uživatelů činí pouze několik jedinců, nárazově se poté bude jednat maximálně o desítky studentů. Hlavní funkcionality systému budou spočívat v práci s formuláři, systém tedy klade pouze minimální nároky na výpočetní výkon. To se dále promítne u výběru databáze, který bude rozveden v sekci 3.1.3.

Webový portál školy Michael je napsán v jazyce PHP a je provozován na školním serveru. Ze strany budoucího provozovatele OS (školy) je preferován tento jazyk z důvodu minimalizace nákladů. Tento výběr podporuje i skutečnost, že PHP splňuje všechny výše uvedené požadavky na programovací jazyk a také osobní praktické zkušenosti autora. Ze všech výše zmiňovaných důvodů byl jako vhodný programovací jazyk pro implementaci prototypu IS zvolen jazyk PHP.

3.1.2 Framework

V dnešní době se s pojmem framework setkáváme téměř neustále. Framework je softwarová struktura, kterou může vývojář použít při vývoji. Ke každému frameworku se váže jistá filozofie, která provází vývoj aplikace. Framework často poskytuje nejrůznější funkcionality, které nacházejí uplatnění v nově vznikajících projektech. To vývojáři umožňuje zaměřit se především na vývoj samotné aplikace a nemusí tak trávit čas implementací základních funkcionalit, které tyto frameworky již poskytují.

Použití frameworku není nutností, ovšem přináší to se sebou řadu výhod. Dle [51] je využití frameworku zárukou kvality, udržitelnosti a rozšiřitelnosti aplikace za nižší ceny. Neméně důležitou výhodou při správném používání frameworku je záruka bezpečnosti, kterou s sebou framework přináší.

Pro PHP vznikla velká řada komunitou oblíbených frameworků. Dle [52] patří mezi nejpopulárnější PHP frameworky Laravel, Symfony a CodeIgniter. Za zmínku také stojí Nette Framework z dílny českého vývojáře Davida Grudla, který je v současné době udržován a vyvíjen komunitou vývojářů.

Frameworkem pro implementaci IS bylo zvoleno Symfony, především díky vhodně navržené práci s balíčky (anglicky bundle), které budou využity při implementaci jednotlivých modulů systému. Další výhodu představuje vestavěná podpora objektově relačního mapování, která usnadní práci s entitami. Prototyp bude vyvíjen ve verzi Symfony 3.4. s dlouhodobou podporou. Ta končí v listopadu 2020, opravy bezpečnostních chyb poté budou vycházet do listopadu 2021.

3.1.3 Databáze

V sekci 2.2.2 byly představeny rozdíly mezi relačními a nerelačními databázemi. Symfony neposkytuje komponentu pro práci s databázemi, zato ale poskytuje těsnou integraci s knihovnou třetí strany Doctrine. Doctrine zprostředkovává ukládání entit a poskytuje abstrakční vrstvu nad databází. Doctrine používá ovladač (anglicky driver), který specifikuje aktuální implementaci abstraktní databázové vrstvy (DBAL). To ji činí nezávislou na použité databázi, při případné změně databáze pouze stačí změnit tento ovladač.

V současné době Doctrine podporuje většinu populárních poskytovatelů databází, konkrétně pak:

- MySQL,
- Oracle,
- Microsoft SQL Server,
- PostgreSQL,
- SAP Sybase SQL Anywhere,
- SQLite,
- Drizzle. [53]

NoSQL databáze je vhodné použít v situacích, kdy se pracuje s nestrukturovanými daty, či pokud je třeba databázi rychle škálovat. Na rozdíl od relačních databází je zde vhodné chytře denormalizovat ukládaná data. Nejedná se tedy o vhodný typ databáze pro tento IS, jelikož ani jedna z těchto situací zde nenastává. Prototyp systému bude otestován na MySQL databázi, v případě potřeby ovšem není problém vhodnou změnou ovladače tento typ databáze změnit.

3.1.4 Prezentační vrstva

Byly představeny technologie, jež budou použity na straně serveru. Tato sekce dále představí technologie, jež budou použity k prezentaci dat uživatelům.

Aplikace bude přístupná pomocí webových prohlížečů a bude využívat HTML5, CSS a JavaScript. HTML5 je nejnovější verze Hypertext Markup Language (HTML), která oproti minulé verzi přinesla například sémantické značky či nové možnosti formulářových prvků. Administrace byla vytvořena za pomoci šablony pro poslední verzi Bootstrapu SB Admin. Ta v sobě obsahuje výše zmíněný Bootstrap, JavaScriptovou knihovnu jQuery a značné množství dalších komponent pro snadný vývoj rozhraní administrátorských systémů.

Použitím Bootstrapu bylo docíleno responsivního rozhraní, které je vhodné k použití na mobilních zařízeních. Prototyp administrace ovšem na zařízeních bez podpory JavaScriptu nemusí zcela správně fungovat. Ovlivněna může být funkcionalita navigačních prvků a načítání dat u určitých tabulek. Této problematice je dále věnována sekce 5.1.7, ve které je analyzováno chování systému za použití různých webových prohlížečů, mobilních zařízení a situací, kdy není k dispozici JavaScript.

3.2 Návrh databáze

Na základě analýzy doménového modelu bylo vytvořeno schéma databáze. Každá tabulka je v systému reprezentovaná právě jednou entitou. Výjimku tvoří tabulky, které dekomponují vztah many-to-many. Many-to-many se také značí zkratkou M:M či M:N a je tak označován vztah dvou entit, kdy entita A má vztah s více entitami typu B a vice versa.

Při tvorbě databáze byl kladen zvláštní důraz na dodržování konzistence pojmenování a strukturování entit. S ohledem na konzistenci vůči programovacímu jazyku bylo rozhodnuto použít anglické názvy tabulek a jejich atributů. Názvy tabulek jsou uvedeny v jednotném čísle, neobsahují velká písmena a jednotlivá slova jsou odděleno podtržítkem. Každá entita obsahuje primární klíč, jenž je typu `integer` a je pojmenován `id`. Některé tabulky obsahují alternativní klíče, které by mohly být použity jako klíče primární. Nebylo tak ovšem učiněno kvůli možným změnám, které by mohly v budoucnu změnit vlastnosti těchto sloupců a znemožnit jejich využití pro roli primárních klíčů.

Tato sekce představí realizaci databázového schématu pro uživatele a importování studentů, které byly představeny v analýze doménového modelu. Kompletní databázové schéma lze potom nalézt na přiloženém CD ve složce `/assets`. Za zmínku zde stojí použité typy klíčů v diagramu, kde se vyskytují následující klíče:

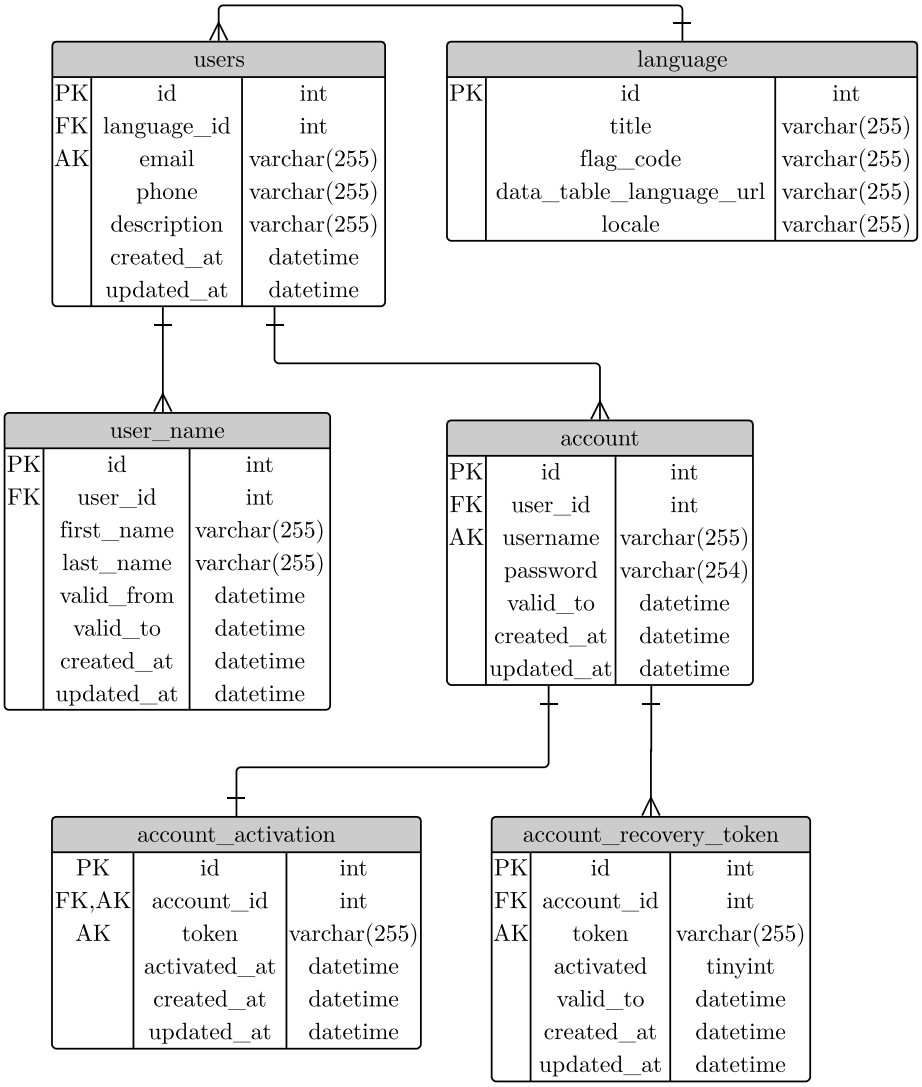
- PK – Primární klíč
- FK – Cizí klíč
- AK – Alternativní klíč

Primární a cizí klíče jsou dobře známé a často používané, výjimku by zde mohly představovat klíče alternativní. Alternativní klíč označuje sloupec v databázi, jenž by mohl být použit jako primární klíč. Jeho hodnoty tedy musí být unikátní.

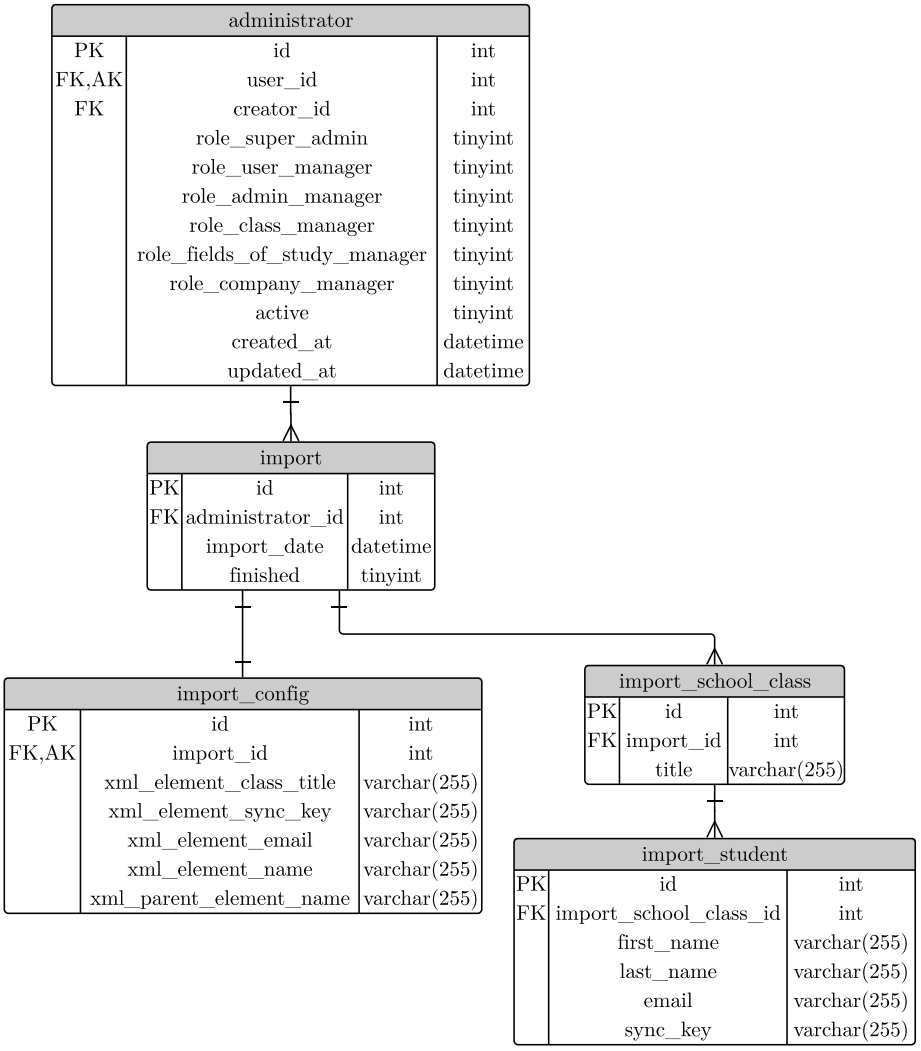
Diagram na obrázku 3.1 zachycuje realizaci doménového modelu uživatelů. Lze pozorovat splnění všech požadavků a kritérií, které byly na tuto část databáze kladeny. Jedná se například o vedení historie jmen a příjmení či podporu více přihlašovacích účtů pro jednoho uživatele. Názvy jednotlivých sloupců by měly svým názvem zachycovat podstatu svého využití. Výjimku zde tvoří sloupec `data_table_language_url` v tabulce `language`. Jedná se o odkazy na soubory s překlady pro plugin DataTables, jenž je v systému využíván.

Tabulky, u kterých se předpokládá pravidelná změna obsahu, dále obsahují sloupce `created_at` a `updated_at`. Hodnoty těchto sloupců reprezentují čas, kdy došlo k vytvoření či změně daného záznamu. Tyto údaje mohou být v budoucnu využity pro tvorbu statistik o využívání systému.

Část schéma databáze zajišťující importování studentů byla realizována dle analýzy doménového modelu a zachycuje ji diagram na obrázku 3.2. Za zmínku



Obrázek 3.1: Schéma databáze – uživatelé



Obrázek 3.2: Schéma databáze – importování studentů

zde stojí zajímavost, že pro importované třídy a studenty nejsou v databázi vynuceny žádné unikátní atributy. Údaje, například název třídy nebo synchronizační klíč studenta, musí být unikátní v rámci jednoho importu. Toho je docíleno na aplikační vrstvě.

3.3 Představení Symfony

Pro účely realizace prototypu IS byl v předchozích sekcích zvolen PHP framework Symfony verze 3.4. Pro správný návrh systému je třeba pochopit filozofii tohoto frameworku a jeho vnitřní strukturu. Tyto znalosti poté najdou uplatnění při návrhu samotné webové aplikace. Tato a následující sekce blíže přiblíží Symfony a představí detailní návrh prototypu.

3.3.1 MVC architektura

Model–View–Controller (MVC) je v současné době velmi populární architektonický vzor, jenž nalézá uplatnění především ve webových aplikacích. Toto tvrzení podporuje skutečnost, že většina populárních PHP frameworků používá právě tuto architekturu. Jedná se například o Laravel či Symfony, mnohé další frameworky poté využívají architektury, které jistým způsobem vychází z MVC. Za zmínku stojí například Nette, které využívá architektury Model–View–Presenter (MVP), který vychází právě z MVC.

Základní myšlenkou MVC architektury je oddělení logiky od výstupu. Řeší tedy problém takzvaného „špagetového kódu“. Je tak označována situace, kdy se v jednom souboru vyskytuje logika aplikace a zároveň renderování výstupu. Takový soubor poté obsahuje databázové dotazy, aplikační logiku a HTML značky (anglicky HTML tags). To má za následek špatnou čitelnost a udržitelnost takového kódu. [54]

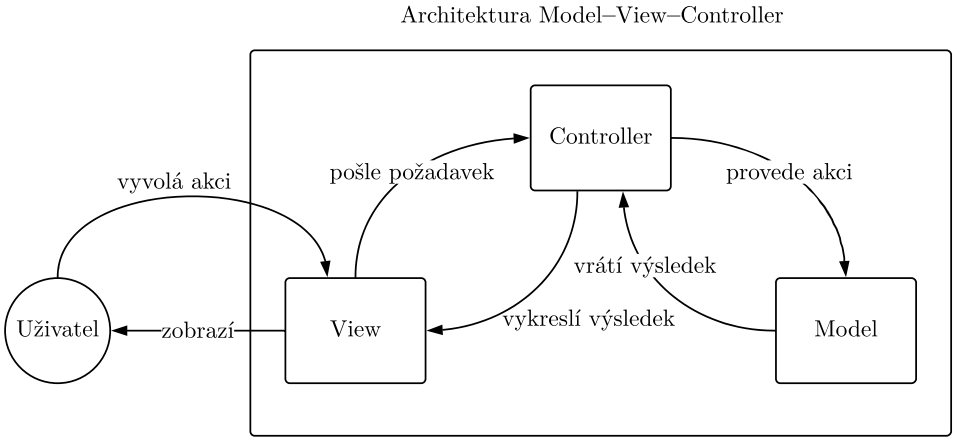
Vhodným výběrem a použitím architektonického vzoru lze takovýmto komplikacím do značné míry předejít. Symfony využívá již zmíněný vzor MVC, který aplikaci rozděluje do tří komponent. Jedná se o datový model aplikace (Model), uživatelské rozhraní či pohled (View) a řídící logiku (Controller). Při návrhu a implementaci těchto komponent je kladen důraz na jejich nezávislost, aby případná změna jedné z komponent ovlivnila své okolí pouze minimálně.

3.3.1.1 Model aplikace

Model obsahuje logiku aplikace a vše, co do ní spadá. Může tedy obsahovat např. databázové dotazy, výpočetní operace či může pracovat s entitami. Díky nezávislosti komponent MVC model neví, jakým způsobem bude se zpracovanými daty naloženo dále ani nezná původní uživatelův požadavek na server.

3.3.1.2 Pohled

Pohled se stará o zobrazení výstupu aplikace uživateli. V našem případě se jedná o použití šablonovacího systému Twig, který má na starosti generování HTML stránek, které jsou poté prezentovány uživateli. Symfony umožňuje použití libovolného kompatibilního šablonovacího systému, pro tvorbu administračního rozhraní byl ovšem použit doporučený systém Twig, který tvoří základní součást Symfony projektu.



Obrázek 3.3: Architektura MVC

Twig kompiluje výsledné stránky z šablon, jež jsou pro administraci uloženy v adresáři `app\Resources\views`. Tyto šablony se vyznačují příponou `.twig` a obsahují speciální syntaxi, která umožňuje rychlou, bezpečnou a flexibilní tvorbu webových stránek. Šablony lze rozšiřovat a navzájem je zahrnovat (includovat) do sebe, což podporuje znovu využitelnost již napsaných šablon.

3.3.1.3 Řídící logika

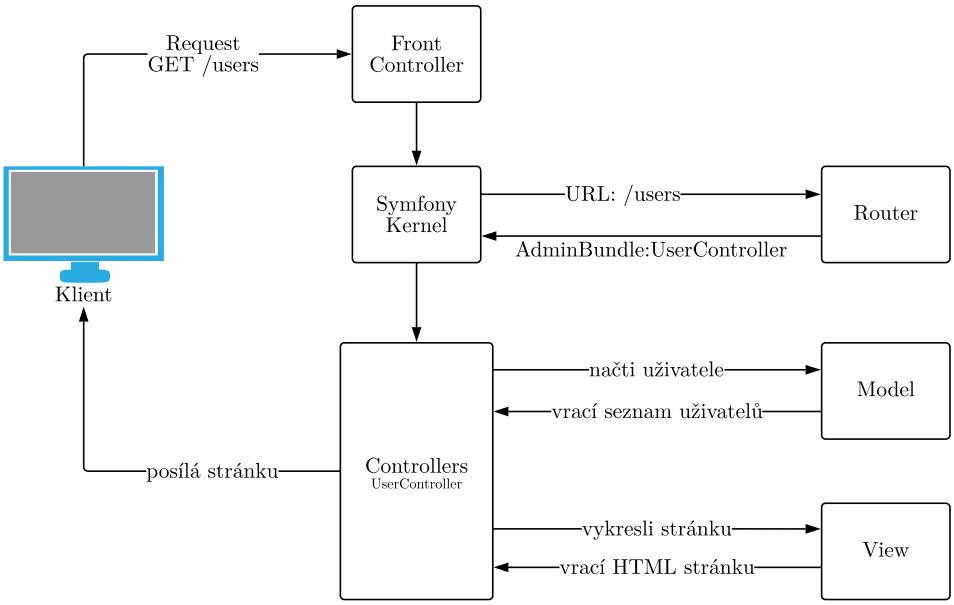
Controller je prvek, který drží celý systém pohromadě. Přijme od uživatele požadavek, zařídí zavolání příslušného modelu a nakonec pomocí pohledu prezentuje uživateli výsledek. Komunikaci jednotlivých komponent zachycuje diagram na obrázku 3.3.

Symfony pro tvorbu controllerů poskytuje třídy *Controller* a *AbstractController*, jejichž děděním lze získat přístup k pomocným funkcím. Administrace obsahuje třídu `BaseController`, od níž dědí všechny ostatní controllery. `BaseController` poskytuje několik funkcionalit a proměnných, které jsou dostupné všem controllerům, jenž od něj dědí. Jedná se například o drobečkovou navigaci, překladač či funkci pro změnu lokalizace.

3.3.2 Životní cyklus požadavku

Diagram na obrázku 3.3 znázorňuje abstraktní pohled na komunikaci jednotlivých komponent. Tato komunikace je blíže upřesněna pomocí analýzy životního cyklu požadavku. Jako příklad pro tuto analýzu byl zvolen požadavek

na zobrazení stránky se seznamem uživatelů, jenž zachycuje diagram na obrázku 3.4. Tento scénář je pouze ukázkový, při implementaci prototypu bylo použito asynchronního načítání uživatelů.



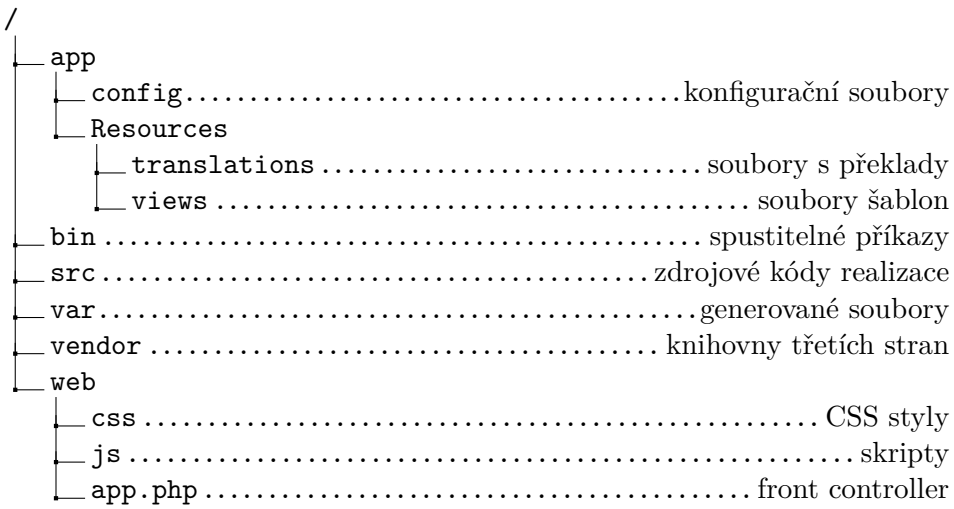
Obrázek 3.4: Životní cyklus požadavku

Životní cyklus požadavku začíná ve chvíli, kdy uživatel vyvolá danou akci ve webovém prohlížeči. Front Controller je návrhový vzor, jedná se o část kódu, kterou musí projít všechny požadavky, které má systém zpracovat. Symfony pro tyto účely poskytuje dva front controllery `app.php` a `app_dev.php`, kdy první z nich je použit v produkčním prostředí, druhý poté při vývoji. Front controller vytvoří instance `AppKernel` a `Request`, následně nechá kernelem zpracovat požadavek a vytvořit odpověď, kterou nakonec odesílá uživateli.

Kernel tvoří jádro Symfony, mimo jiné načítá a inicializuje aplikační balíčky. Kernel následně využije služeb routeru k zjištění názvu příslušného controlleru, kterému bude delegováno zpracování požadavku. V případě načítání seznamu uživatelů je zavolána funkce `indexAction()` controlleru `UserController`. Ta dále načítá seznam uživatelů pomocí příslušné třídy aplikační logické vrstvy. Pro načtená data je vytvořena HTML stránka pomocí šablonovacího systému Twig, která je následně odeslána uživateli.

3.3.3 Adresářová struktura

Adresářová struktura Symfony je flexibilní, ovšem Symfony důrazně doporučuje používat doporučenou strukturu. Administrace neklade na tuto strukturu žádné speciální nároky, pro účely realizace prototypu byla tedy zvolena doporučená adresářová struktura. Použitou adresářovou strukturu zachycuje diagram na obrázku 3.5



Obrázek 3.5: Použitá adresářová struktura

3.4 Návrh systému

Předchozí sekce představila strukturu a filozofii Symfony. Nyní je třeba navrhnout modulární architekturu a adresářovou strukturu, která umožní realizaci výsledného IS.

3.4.1 Modulární architektura

Vyvíjený informační systém bude postaven na modulární architektuře, která umožní jeho lehké rozšíření o dodatečné funkcionality. Je tedy třeba navrhnout, jakým způsobem bude tato architektura realizována. Při návrhu architektury využijeme jednu z nejsilnějších vlastností Symfony, takzvaný budle system.

Na jednotlivé bundly můžeme nahlížet jako na pluginy v ostatních programech. Zde se ovšem nehovoří o pluginech, jelikož se z nich skládá celé Symfony, od jádra frameworku až po komponenty samotné aplikace. Použití bundlů umožňuje zapouzdření jednotlivých komponent a jejich lehké znovu použití a sdílení.

Prototyp IS obsahuje dva základní bundly. Jedná se konkrétně o `DefaultBundle` a `AdminBundle`. `DefaultBundle` poskytuje uživatelské rozhraní nad celým informačním systémem a poskytuje uživateli možnost vstupu do jednotlivých částí systému. Druhý bundle poté obsahuje implementaci administrace, která představuje hlavní cíl této práce. Každý další modul systému bude tvořit nový bundle. Tyto bundly poté stačí zaregistrovat v `AppKernel.php` a lze je ihned využívat.

3.4.2 Adresářová struktura

V sekci 3.4.1 byla představena adresářová struktura projektu. Dále je třeba stanovit pravidla a konvence, které budou dodržovat jednotlivé aplikační moduly. Tímto bude zajištěna konzistence jednotlivých bundlů, ulehčena jejich komunikace a zajištěna vzájemná kompatibilita.

Pro administrační modul byla navržena následující adresářová struktura (znázorněna na obrázku 3.6). Ostatní moduly budou dodržovat tuto strukturu, nemusí však nutně obsahovat všechny její komponenty.

```
AdminBundle
├── ConfigProvider ..... třídy pro distribuci konfiguračních hodnot
├── Controller ..... controllery
├── Entity
│   ├── Interfaces ..... rozhraní entit
│   ├── Traits ..... traity entit
│   ├── DataTables ..... pomocné třídy
│   └── Imports ..... pomocné třídy
├── EventListener ..... nasluchači událostí
├── EventSubscriber ..... odběratelé událostí
├── Exception ..... vyjímky
├── Form ..... formuláře
├── Repository ..... 1. vrstva aplikační logiky
├── Functionality ..... 2. vrstva aplikační logiky
├── Operation ..... 3. vrstva aplikační logiky
├── Security ..... autentizační třídy
├── Voter ..... autorizační třídy
├── Serializer ..... serializéry
└── AdminBundle.php
```

Obrázek 3.6: Struktura administračního modulu

3.4.2.1 Správa konfiguračních hodnot

Při návrhu systému byl kladen důraz na možnost konfigurace co největšího počtu jeho parametrů. Hodnoty těchto parametrů jsou uloženy v konfiguračních

souborech v adresáři `app/config`. Tento adresář obsahuje hierarchii konfiguračních souborů, které jsou navrženy takovým způsobem, aby bylo možné mít pro každé prostředí vlastní konfigurační soubory.

Některé služby administračního modulu ovšem vyžadují přístup k těmto parametrům. Za účelem distribuce konfiguračních hodnot vznikla třída `ConfigProvider.php`, kterou si mohou vyžádat ostatní služby aplikace. Tato třída je zaregistrována jako speciální služba a jsou jí v konstruktoru předány všechny potřebné konfigurační hodnoty. Všechny třídy či služby, které poté vyžadují přístup k těmto parametrům využívají služeb této třídy.

3.4.2.2 Vrstvy aplikační logiky

Vrstva aplikační logiky se v MVC skrývá pod slovem Model a poskytuje samotné funkcionality systému. Informační systém bude využívat tří takovýchto vrstev. Tyto vrstvy mají pevně stanovenou hierarchii a je umožněna pouze jednosměrná komunikace, třídy vyšších vrstev mohou využívat služeb vrstev nižších, nikdy však ne naopak.

Nejnižší vrstvu representují repositáře. Každý repositář je pevně spjat právě s jednou entitou. Repositáře dědí od třídy `EntityRepository` a poskytují dodatečnou funkcionalitu. Jejich hlavním účelem je načítání dat z databáze.

Druhou vrstvu představují třídy adresáře `AdminBundle/Functionality`. Každá `funkcionalita` se váže právě k jedné entitě, existují zde ovšem i výjimky, kdy se `funkcionalita` neváže k entitě žádné. `Funkcionality` poskytují základní funkční logiku s entitami, jako je jejich tvoření, úprava či ukládání. `Funkcionality` na rozdíl od repositářů mohou navzájem komunikovat a využívat služby nižší vrstvy, v tomto případě služby `repositářů`.

Speciální typ `funkcionality` představuje třída `SecurityFunctionality`, která se neváže k žádné entitě. Tato třída poskytuje možnosti generování unikátních, kryptograficky bezpečných řetězců či jejich hashování. Toho je využíváno zvláště u generování bezpečnostních tokenů, které jsou používány například při obnově hesla či slouží jako přístupové kódy studentů.

Nejvyšší vrstvu představují `operace`. `Operace` umožňují provádění složitých procesů, jež jsou realizovány za použití služeb nižších vrstev či ostatních `operací`. Běžně se tak zde vyskytují například funkce na zpracování formulářů či jiné složité procesy, jako je například importování studentů či jejich převod do vyšších ročníků.

3.4.2.3 Entity

Entity se dělí na dva typy, perzistentní a neperzistentní. Perzistentní entity jsou úzce propojeny s doménovým modelem a navrženou databází. Představují reálné objekty, se kterými může uživatel přijít do styku. Entity jsou uloženy v adresáři `AdminBundle/Entity`, kde jsou dále uloženy i příslušné traity a rozhraní.

Každá perzistentní entita má svoje rozhraní, které musí striktně dodržovat. Moduly poté mohou pracovat s těmito rozhraními namísto entit, což je zbavuje nutnosti spoléhat na konkrétní implementaci, která je náchylná ke změnám.

Neperzistentní entity jsou poté pomocné třídy, které hrají podobnou roli jako struktury v C++. Jsou tedy používány zvláště v situacích, kdy je třeba předat větší množství argumentů. Jako příklad lze uvést třídu `AjaxResponse`, kterou lze použít namísto předávání 6 argumentů. Tato třída dále poskytuje pomocné funkce jako je např. tvorba asynchronní odpovědi, která odstraňuje potřebu opakování kódu na více místech v aplikaci.

3.4.2.4 Práce s událostmi

Události (anglicky events) jsou spouštěny jádrem systému při správě HyperText Transfer Protokol (HTTP) požadavků, či je mohou vytvářet aplikační bundly. Symfony umožňuje tvorbu tříd, které s těmito událostmi mohou pracovat. Jsou k dispozici dva možné přístupy, posluchač (anglicky listener) a odběratel (anglicky subscriber). Posluchač je zaregistrován pro určitý typ události, zatímco odběratel říká jádru systému, kterým událostem naslouchá.

Tento systém využívá oba výše uvedené způsoby. Zástupcem posluchače je třída `RequestListener`. Tato třída naslouchá všem událostem, které vznikají při interakci uživatele se systémem. Účelem této třídy je kontrola přístupových oprávnění uživatele během doby, co systém aktivně využívá. Pokud by uživatel tato oprávnění ztratil, Symfony by ho automaticky neodhlásilo, z tohoto důvodu vznikla tato třída.

Systém dále obsahuje jednoho odběratele událostí, jedná se o `UserLocaleSubscriber.php`. Událost, kterou tato třída odebírá vzniká v době přihlášení uživatele do systému. Po úspěšném přihlášení dojde k nastavení správné lokalizace systému na základě uživatelových preferencí.

3.4.2.5 Autorizace a autentizace uživatelů

Zvláštní důraz při návrhu systému byl kladen na bezpečnost. Setkáváme se zde s pojmy autorizace a autentizace uživatelů. Autentizace uživatele spočívá v ověření, zda-li se jedná skutečně o daného uživatele. Skrývá se zde tedy přihlašování a odhlašování uživatelů. v Symfony pro autentizaci uživatelů poskytuje mocné nástroje. Uživatel se může přihlásit pomocí přístupového účtu či dle přístupového kódu, který mu byl vygenerován v administraci. Přístup do administrace je umožněn pouze pomocí přístupového účtu, kdy musí mít

uživatel zároveň dostatečné oprávnění. Oprávnění jsou řešena pomocí uživatelských rolí, které blízce souvisí s autorizací uživatelů.

Autorizace je proces, při němž se zjišťuje, zda-li má uživatel udělen přístup do dané sekce, či zda-li má dostatečná oprávnění pro provedení požadované akce. Tento IS podporuje a využívá hierarchii uživatelských rolí. Do administrace je umožněn pouze přístup uživatelům s rolí `Administrátor` či `Učitel`. Po vstupu do administrace má uživatel udělen vstup pouze do sekcí, ke kterým má patřičné role.

Třídní učitelé mají omezený přístup na profil svých tříd, kde mohou kontrolovat údaje studentů a generovat jim jednotlivě či hromadně přístupové kódy. Učitelé, jenž nejsou třídními učiteli žádné třídy nemají v administraci přístup do žádné sekce. Modul pro správu zápisu předmětů bude rozšiřovat administraci o další sekce, do kterých budou mít přístup všichni učitelé.

Administrátoři získávají plný přistup do sekcí, ke kterým mají patřičné role. Ke zjištění, zda-li má administrátor přístup lze využít dva způsoby. Prvním je již zmíněná kontrola role, která se používá, pokud není třeba přístupy dále omezovat na úrovni vykonávání jednotlivých akcí. Některé sekce ovšem vyžadují právě takovéto kontroly, ke kterým slouží třídy typu `volič` (anglicky voter).

Voliči slouží k rozhodování, zda-li má uživatel oprávnění vykonat požadovanou akci. Každá takováto třída má stanovený subjekt, o kterém může hlasovat a seznam operací, které lze s tímto subjektem vykonat. O udělení oprávnění hlasují všichni voliči a následně se dle nastavené strategie rozhodne o výsledku. Tento projekt využívá výchozí strategii, jedná se o tzv. `potvrzovací strategii`, kdy je přístup udělen ve chvíli, co ho potvrdí libovolný z voličů.

3.4.2.6 Controllery

Jak již bylo řečeno v sekci 3.3.1, controller je třída, která spojuje všechny komponenty aplikace dohromady. Tato třída by neměla vykonávat žádnou aplikační logiku, všechny požadavky deleguje do patřičných vrstev aplikační logiky. Tento princip byl zohledněn při návrhu struktury aplikace, controllery zde tedy nevykonávají žádnou logiku.

Controllery využívají anotací metod, kde lze specifikovat např. název a tvar url adresy či požadované oprávnění pro danou metodu. Po zavolání dané metody dojde k delegaci zpracování požadavku na jednu z vrstev aplikační logiky. Po zpracování tohoto požadavku controller vygeneruje pomocí systému Twig výslednou HTML stránku či odpověď a odešle ji uživateli.

Kapitola 4

Realizace

4.1 Realizace prototypu

Tato kapitola se zabývá realizací a nasazením prototypu vyvíjeného IS. Autor zde přiblíží proces implementace zajímavých funkcionalit a problémů, které se vyskytly během vývoje. Následovat bude implementace uživatelského rozhraní a překladače, který poskytuje překlady systému do jazyka preferovaného uživatelem. Na závěr bude provedeno nasazení implementovaného prototypu do testovacího prostředí a bude přiblížen proces nahrávání nových verzí.

4.1.1 Líné načítání

Během vývoje narazil autor této práce na technická omezení použitých technologií, zejména poté na způsob, jakým jsou načítány objekty z databáze. Doctrine využívá takzvaného líného načítání (anglicky lazy loading), jedná se o způsob načítání dat z databáze, kdy se místo se skutečnou entitou pracuje s jejím zástupcem (anglicky proxy). Skutečná entita a její atributy jsou poté načteny z databáze teprve ve chvíli, kdy je s nimi zapotřebí pracovat. Tento systém výrazně šetří paměť a přináší s sebou mnohé další výhody. Líné načítání ovšem způsobuje komplikace v situacích, kdy je třeba procházet přes desítky, stovky nebo tisíce entit, jelikož pro každou entitu je třeba vytvořit nový databázový dotaz.

Těchto situací se lze vyvarovat více způsoby. Nabízí se dočasné vypnutí líného načítání či tvorba vlastních databázových dotazů, kdy se všechny entity načtou v jednom dotazu. Po vyřešení této situace ovšem rostou nároky na paměť, jelikož se ve stejné době načítá do paměti více entit najednou. Na tyto problémy při načítání velkého množství entit současně narazil autor při více příležitostech, zejména pak v následujících situacích:

- Načítání přehledu všech uživatelů systému.
- Načítání přehledu všech studentů.

- Načítání přehledu všech tříd.
- Načítání přehledu archivovaných tříd.

Tyto komplikace byly nakonec vyřešeny pomocí stránkování výsledků na straně serveru. Uživateli se na začátku pošle pouze omezený počet výsledků a teprve při žádosti o zobrazení dalších se mu odešlou vyžadovaná data.

Administrace využívá pluginu `DataTables.js`, který vytváří interaktivní tabulky na straně uživatele. Tyto tabulky lze poté řadit dle sloupců, vyhledávat v nich a používat stránkování. Řešení na straně klienta má za výhodu rychlou odezvu systému, jelikož uživatel má všechna data již připravena.

`DataTables.js` podporují provedení všech těchto funkcionalit na straně serveru, kdy je pomocí JavaScriptu odeslán asynchronní požadavek na server. Této možnosti je využito ve výše zmíněných situacích, kdy je zapotřebí provádět tyto operace na straně serveru. Vzhledem ke skutečnosti, že tabulky obsahují data získaná spojením více entit najednou, není ovšem možné nahradit plně tyto funkcionality za každé situace. V administraci se tedy vyskytují 4 tabulky, které mají možnosti řazení a vyhledávání omezené na podmnožinu svých sloupců.

4.1.2 Implementace funkcionalit administrace

Na administraci vznikly při analyzování projektu mnohé požadavky, viz. jejich analýza v sekci 2.3. Prototyp administrace všechny tyto požadavky úspěšně splňuje za pomoci implementace mnohých funkcionalit. Tato sekce se zaměřuje na řešení, kterými byly tyto požadavky splněny.

4.1.2.1 Správa uživatelů

Administrace poskytuje kompletní správu uživatelů a jejich rolí, jedná se tedy zejména o vytváření, úpravu a mazání. Uživatele v systému reprezentuje entita `User`, každá uživatelská role je poté reprezentována vlastní entitou, která je s entitou `User` úzce provázána. Každá uživatelská role má v administraci vytvořen vlastní profil, který obsahuje informace týkající se dané role a základních informací o uživateli.

Pro každou uživatelskou roli existuje vlastní controller, který začíná názvem dané role. Pro roli administrátora tedy existuje `AdministratorController`, pro roli studenta poté `StudentController` atd. Každý z těchto controllerů obsahuje funkce, které souvisí se správou dané role uživatele či se správou uživatele samotného v případě `UserControlleru`.

Zvláštní kategorii rolí poté tvoří administrátorská oprávnění, která se vždy vztahují k entitě `Administrator` a nejsou reprezentována vlastní entitou. Seznam těchto oprávnění určuje UC6, jenž byl představen v sekci 2.4.

4.1.2.2 Správa přihlašovacích účtů

Pod správu uživatelských účtů spadají účty a přístupové kódy. Je zde kladen důraz na bezpečnost, pro manipulaci s těmito údaji jsou zapotřebí potřebná oprávnění. Pro správu přihlašovacích účtů je vyžadována role `správce uživatelů`, pro generování přístupových kódů je poté dostatečná i role `správce tříd`.

Přístupová hesla k uživatelským účtům jsou hashována v databázi, což případnému útočníkovi znemožňuje jejich zneužití. Pro účely hashování byl použit kryptograficky bezpečný algoritmus BCrypt, který je zmiňován a doporučován dokumentací Symfony. Mezi silné stránky tohoto algoritmu patří především:

- Implicitní generování soli (anglicky salt), který je použit při hashování hesla. Není zapotřebí si tento salt uchovávat zvlášť v databázi, jelikož tvoří část výsledného řetězce;
- BCrypt je pomalý hashovací algoritmus, což z něj dělá ideálního kandidáta právě pro hashování hesel. Pomalý výpočet hashe téměř zcela eliminuje možnost útoku hrubou výpočetní silou;
- Náročnost výpočtu hashe lze pro BCrypt nastavit pomocí tzv. ceny (anglicky cost). Hodnota ceny se pohybuje v rozmezí 4–31, výchozí hodnota je 13. Každé zvýšení ceny o 1 znamená dvojnásobnou dobu výpočtu oproti předchozí hodnotě;
- Hodnotu ceny výpočtu lze měnit za běhu aplikace, tato cena tvoří součást hesla, čímž je zajištěna zpětná kompatibilita.

BCrypt je použit pouze pro hashování hesel a to za pomoci třídy `PasswordEncoder`. Pro hashování jakékoli jiné hodnoty lze využít funkci `hashPlainCode()`, jež poskytuje třída `SecurityFunctionality`. Tato funkce přijímá jeden argument typu řetězec a vrací jeho hash vygenerovaný pomocí algoritmu `sha512`. Jedná se o rychlý a bezpečný algoritmus, který byl zvolen vzhledem k dopředu neznámému seznamu subjektů, které budou této funkce využívat.

Pro potřeby generování unikátních a bezpečných řetězců slouží funkce `generateUniqueCode(…)`, která je dostupná v téže třídě. Tato funkce přijímá tři argumenty, název třídy, název proměnné a délku řetězce v bytech. Generování náhodného řetězce probíhá pomocí funkce `string random_bytes(int $length)`, která je vhodná pro generování kryptograficky bezpečných řetězců.

Po zavolání této funkce se validují předané parametry, tzn. zda-li existuje třída daného názvu, zda-li v ní existuje daná proměnná a zda-li je požadovaná délka delší jak 0. Při porušení jedné z těchto kontrol je vrácena výjimka `InvalidArgumentException`. Pokud jsou všechny argumenty validní, je vy-

tvořen nový bezpečný řetězec, který je unikátní pro danou proměnnou zadané třídy. Vygenerovaný řetězec je nyní bezpečný a unikátní, ovšem aby ho bylo možné použít ve všech situacích, je před navrácením překódován pomocí algoritmu `base64` do čitelné podoby.

4.1.2.3 Správa tříd

Správu tříd zajišťuje třída `SchoolClassController`. Administrace správy umožňuje vytváření, čtení a upravování jednotlivých tříd. Možnost zrušení tříd je nahrazena volbou archivace. K obsluze administrace je třeba, aby měl uživatel přidělen status `správce tříd`.

Sekce Třídy se dělí v administraci na tři části. Uživatel může vybrat volbu aktivní třídy, archivované třídy a všechny třídy. Na základě problematiky, která byla představena v sekci 4.1.1, jsou tabulky obsahující seznam tříd zpracovávány na straně serveru a poskytují tak omezené možnosti vyhledávání a řazení.

V situaci, kdy je třída automaticky archivována z důvodu vypršení její platnosti zůstává všem studentům aktivní studentský status. Studenti ztrácejí tento status pouze v případě, že dojde k archivaci třídy při převodu do vyššího ročníku, či pokud jim je tento status manuálně odebrán. Důvodem tohoto nastavení systému je vyvarování se zrušení statusu student bez vědomí administrátora. Operace převodu do vyššího ročníku musí být provedena manuálně. Na tuto skutečnost je administrátor upozorněn pomocí zobrazených instrukcí.

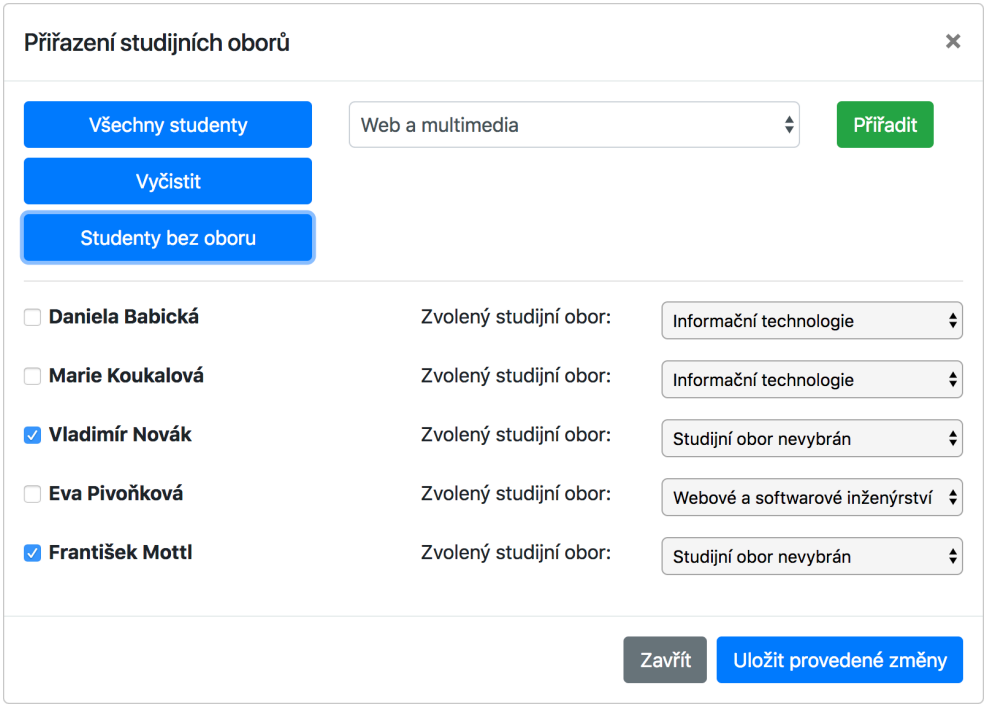
4.1.2.4 Správa studijních oborů

Systém umožňuje kompletní správu agendy studijních oborů s výjimkou jejich zrušení. Ta je nahrazena možností archivace. Při archivaci zůstává tento obor všem studentům, kteří ho měli zapsaný v době ukončení jeho platnosti. Novým studentům však již tento obor zapsán být nemůže.

Specifikem vzdělávacího systému na střední škole Michael je existence tzv. specializací u některých studijních oborů. Tyto specializace se také nazývají „upřesněním oboru“. Pokud obor takovéto upřesnění obsahuje, nelze si ho zapsat přímo a je třeba zvolit jednu z dostupných specializací. Informační systém s těmito specializacemi počítá a poskytuje jejich podporu.

Specializace je charakterizována stejnými parametry jako studijní obor. Je proto realizována pomocí již existující entity, která standardně reprezentuje právě studijní obory. Uživatelské rozhraní poté uživateli zobrazuje možnost výběru ve formátu `název oboru - název specializace`.

Administrace umožňuje přiřazení a výběr jednotlivých oborů studentům více způsoby. Základním způsobem je přiřazení jednoho oboru jednomu studentovi. To je možné realizovat na profilu studenta nebo na detailu třídy, do které student náleží. Obory je také možno přiřazovat hromadně. Rozhraní pro hromadné přiřazení je zachyceno na obrázku 4.1.



Obrázek 4.1: Rozhraní přiřazování studijních oborů

Toto rozhraní je implementováno pomocí vyskakovacího okna (anglicky modal), které obsahuje patřičný formulář. Tento formulář umožňuje přiřazení oborů jednotlivým studentům či více studentům najednou. Studenty lze označit ručně, či pomocí ovládacích prvků formuláře. Pro tyto účely obsahuje formulář tři ovládací tlačítka. Ta poskytují možnost vybrat všechny studenty najednou, vybrat pouze studenty bez přiřazených studijních oborů či zrušení výběru označených studentů.

4.1.2.5 Importování studentů

Importování studentů bylo implementováno na základě doménového modelu, jenž byl představen v sekci 2.5.5. V sekci 3.2 poté bylo představeno schéma databáze, na které budou přímo mapovány jednotlivé entity související s nahráváním studentů.

Proces importování studentů (dále jen import) je reprezentován instancí entity `Import`. Import se může nacházet ve dvou stavech, `Probíhá` a `Dokončeno`. Data studentů jsou uložena do databáze ihned po nahrání a zpracování souboru s těmito daty. Rozpracovanou práci lze tedy opustit bez rizika ztráty neuložených dat.

Studenti jsou nahráváni pomocí XML souborů. Každý z těchto souborů může obsahovat studenty jedné či více tříd. Proces zpracování těchto souborů přiblíží následující sekce. Importovaná data studentů jsou reprezentována pomocí entity `ImportStudent`. Tato entita nemá žádnou přímou vazbu na entitu `Student`, která v systému reprezentuje skutečné studenty.

Nahraná data studentů jsou s opravdovými studenty synchronizována teprve při dokončení importu. K synchronizaci je použit tzv. synchronizační klíč, který musí povinně obsahovat každý importovaný student. Tento klíč může být reprezentován libovolnou unikátní hodnotou, například rodným číslem studenta.

Je-li při synchronizaci studentů pro daný klíč nalezena shoda, proběhne aktualizace odpovídající entity. Není-li však v systému nalezen žádný student s hledaným klíčem, je vytvořen nový.

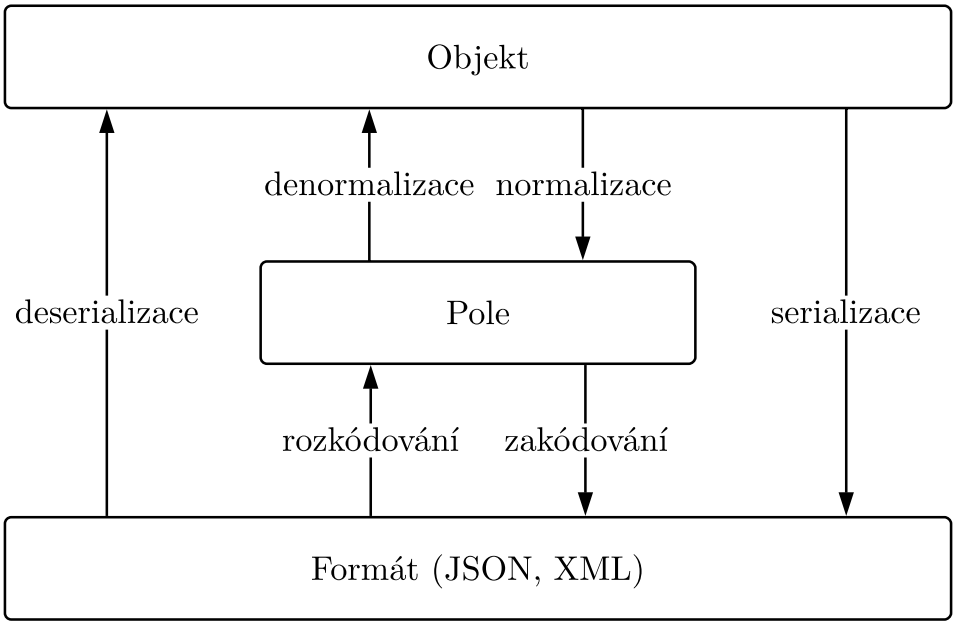
Zpracování XML souboru probíhá pomocí komponenty `Serializer`, která je součástí Symfony frameworku. Tato komponenta umožňuje převádět instance entit do výstupních formátů, jako jsou například JSON či XML. `Serializer` se řídí dle schématu zachyceného na obrázku 4.2, v případě realizovaného prototypu bude převáděn XML soubor zpět na objekt.

Pro účely deserializace importovaných údajů vznikla třída `XmlElementNameNormalizer`, která převádí názvy atributů získaných z rozkódovaných souborů na názvy proměnných třídy `ImportStudent`. Tento normalizér přijímá instanci třídy `ImportConfig`, která obsahuje názvy elementů nahraného XML souboru, struktura těchto souborů se tak může do jisté míry měnit.

4.1.2.6 Převod studentů do vyšších ročníků

Na rozdíl od procesu importování studentů je převod do vyšších ročníků realizován pomocí formuláře. Při jeho implementaci bylo využito jedné z dalších silných stránek Symfony, konkrétně pak možnosti skládat formuláře z jiných formulářů. Hlavní formulář je reprezentován třídou `MigrationType`, která dále obsahuje formuláře typu `MigrationRecordType`, kdy je pro každou aktivní třídu přidán jeden takovýto formulář. `MigrationRecordType` obsahuje indikátor, zda-li bude daná třída archivována či nikoliv.

Při zpracování formuláře je pro každou třídu rozhodnuto, zda-li bude archivována či nikoliv. Pokud je třída určena k archivaci, všichni její studenti ztrácí studentský status a získávají novou roli `bývalý student`. Uživateli s touto rolí není umožněn přístup do systému, slouží pouze k odlišení aktivních studentů od bývalých.



Obrázek 4.2: Schéma komponenty Serializer

Pokud třída nebyla určena k archivaci, je vytvořena nová třída pro další ročník. Tato třída je pojmenována dle odpovídající hodnoty z formuláře, kterou mohl uživatel změnit. Všichni studenti, kteří se aktivně účastnili staré třídy v době převodu jsou přesunuti do třídy následující. Stará třída je poté archivována a lze ji tak dohledat v přehledu archivovaných tříd.

Při rozhodování, zda-li bude třída pokračovat dalším ročníkem je využito služby `ConfigProvider`. Ta načítá z konfiguračních souborů počet ročníků, které jsou na škole standardně vedeny, základní hodnota činí 4. Pokud uživatel při převodu tříd do vyšších ročníků nastaví více třídám stejný název, dojde k jejích spojení. Výsledná třída poté obsahuje všechny studenty a třídní učitele ze sloučených tříd.

4.2 Uživatelské rozhraní

Webové stránky administrace renderuje na straně serveru systém Twig, který byl představen v sekci 3.3.1. Tato sekce se bude zabývat strukturou stránek samotných a technologiemi, které byly použity při jejich implementaci.

Systém využívá nejnovější verzi HTML, konkrétně se jedná o HTML5. Ta s sebou přináší oproti předchozí verzi mnohá vylepšení, jedná se například o nové sémantické značky či nové formulářové prvky. Prototyp hojně využívá především značek, které nacházejí v systému četná uplatnění. Nyní je tedy možné používat konstrukce jako `<header>` namísto `<div class="header">`, které byly typické pro starší verze HTML. Výsledný kód je tak jednodušší, přehlednější a lze se v něm rychleji orientovat.

Při implementaci administrace byla použita nejnovější verze knihovny Bootstrap, která s sebou přináší CSS3 a JQuery. Bootstrap poskytuje řadu HTML komponent, které lze použít při tvorbě webových stránek, administrace se poté skládá téměř výhradně právě z těchto komponent. Jejich využití přineslo celou řadu benefitů. Jedná se především o dosaženou konzistenci uvnitř celého systému, rovnoměrně ve všech sekcích administrace.

4.2.1 Zpracování aktiv

Pro účely zpracování Cascading Style Sheets (CSS) a HTML souborů byl využit populární nástroj Gulp. Gulp je tzv. stavějící nástroj (anglicky build tool), jenž nachází uplatnění při vývoji webových aplikací. Gulp umožňuje automatizaci procesů při vývoji, podporuje optimalizování CSS a JavaScript (JS) souborů a umožňuje použití CSS preprocesorů, jako jsou například Sass či LESS.

Díky skutečnosti, že systém využívá Gulp, mohl být použit Sass preprocesor pro tvorbu stylů a nejnovější ECMAScript standard pro psaní skriptů. Pro kompilaci CSS souborů byla pro Gulp vytvořena úloha `sass-compile`. Po spuštění této úlohy dojde k vykonání následujících operací:

1. Jsou načteny všechny soubory s příponou `.scss` v adresáři `web/css/scss/`;
2. Za pomoci autoprefixeru je zajištěna kompatibilita stylů pro všechny moderní prohlížeče, které mají na trhu větší podíl než 5%;
3. Všechny výsledné styly jsou poté sloučeny do jednoho souboru, který se nazývá `master.js`. Tento soubor obsahuje všechny styly, jež jsou v prototypu administrace použity. Výjimku tvoří styly třetích stran a knihoven, které jsou zpracovány jinou úlohou a uloženy v souboru `libs.js`;
4. Výsledný soubor je dále zpracován pomocí pluginu pro Gulp `uglifycss`. Tento plugin odstraní přebytečné bílé mezery, minimalizuje a optimalizuje výsledný soubor.

Podobně jsou zpracovány i všechny použité scripty. Skripty jsou převedeny do staršího standardu kvůli podpoře všech moderních prohlížečů, následuje jejich optimalizace a minimalizace.

Tabulka 4.1: Podpora Bootstrapu na mobilních zařízení

	Chrome	Firefox	Safari	Android Browser	Edge
Android	+	+		Android v5.0+	+
iOS	+	+	+		+
Windows Mobile					+

Gulp dále obsahuje úlohy pro kopírování potřebných souborů z adresářů třetích stran do veřejného adresáře, odkud jsou tyto zdroje dále načítány pro použití na webových stránkách. Jedná se například o použité fonty nebo ikony vlajek.

4.2.2 Mobilní zařízení

Vzhledem k nefunkčnímu požadavku N3, jenž byl představen v sekci 2.3.2, musí administrace plně podporovat mobilní zařízení. Podporu těchto zařízení přináší s sebou právě Bootstrap, jehož aktuální použitá verze je od základu navržena právě s ohledem na tato zařízení. Podporu nejpopulárnějších prohlížečů dle operačního systému zařízení zachycuje tabulka 4.1.

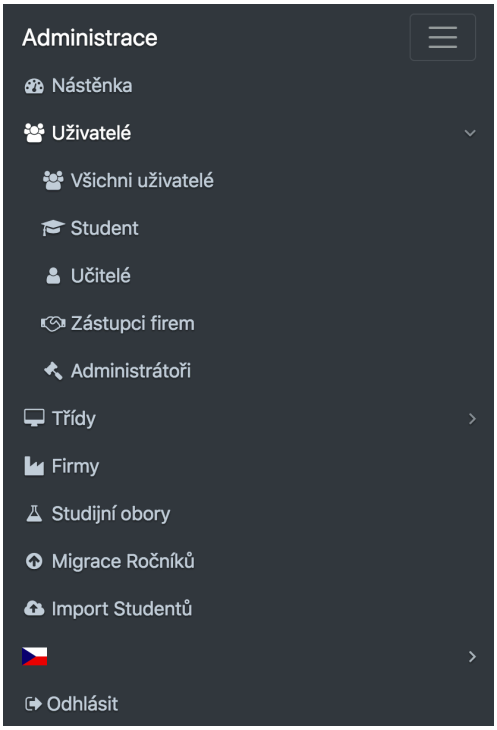
Stávající verze Bootstrapu využívá tzv. flexboxů, jedná se populární způsob návrhu responsivních rozhraní. Flexbox je dle [55] v současné době podporován na všech předních prohlížečích a mobilních zařízeních. Za zmínku zde stojí responzivní chování levého a horního menu, které je na mobilních zařízeních sjednoceno do jednoho. Toto menu lze rozbalit pomocí tzv. „hamburger tlačítka“, které se standardně používá pro tyto účely. Rozbalování menu ovšem vyžaduje podporu JavaScriptu, nabízí se zde tedy prostor pro budoucí vylepšení prototypu. Na obrázku 4.3 je zachyceno rozbalené menu na simulovaném zařízení iPhone 8 Plus.

Prohlížeče bez oficiální podpory mohou fungovat, není však zaručena 100% kompatibilita a některé komponenty mohou vykazovat defekty či neočekávané chování. Testování populárních mobilních zařízení a jejich prohlížečů bude provedeno v následující kapitole.

4.3 Překlady

Na systém je kladen požadavek multijazykové podpory. Během analýzy požadavků bylo stanoveno, že systém bude v době nasazení podporovat český a anglický jazyk. Spolu s dalšími rozšířeními pro IS přibude v budoucnu podpora dalších jazyků. Prototyp administrace je kompletně přeložen do dvou výše uvedených lokalizací. Tato sekce se zabývá implementací překladače a využitím souvisejících technologií.

Symfony framework obsahuje překládací komponentu, pomocí které lze celý proces překladu nastavit a spravovat. Základním krokem je správné nasta-



Obrázek 4.3: Vzhled menu na mobilních zařízeních

vení lokalizace v konfiguračních souborech. Zde je uvedena cesta k souborům s překlady a výchozí jazyk, který je použit v případě, kdy není žádný jazyk explicitně stanoven. Výchozí hodnota lokalizace je v konfiguračním souboru přijímána jako parametr, lze ji tedy nastavit z běhového prostředí, v němž je systém provozován. Výchozím jazykem tohoto projektu byl zvolen český jazyk.

Samotné překlady jsou zprostředkovány pomocí služby překladače, který tvoří jádro překládací komponenty. Překlad probíhá pomocí klíče zprávy, která má být přeložena. Spolu s klíčem může být překladači poskytnut i seznam parametrů, které budou dosazeny do překládané zprávy. Překlady probíhají zavoláním funkce překladače `trans(string $id, array $params = [], string $domain = 'messages')`. Tato funkce navrací požadovanou zprávu v cílovém jazyce. Pokud žádná hodnota pro zadaný klíč neexistuje, ne navrácen klíč samotný.

Překladač podporuje více formátů pro ukládání překladů, tento projekt využívá známý formát `yaml`. Všechny překladové soubory dodržují jmennou konvenci `domena.kod_zeme.yml`. Samotné překlady jsou poté uloženy v těchto souborech jako dvojice ve formátu `klic: "hodnota"`. Obsah těchto souborů je seřazen abecedně pro rychlejší vyhledávání a snadnější údržbu.

Zvláštní pozornost si zaslouží překlady pluginu DataTables, který slouží

pro manipulaci s tabulkami na straně klienta. Tento plugin podporuje překlady, ty lze nastavit dle ukázky kódu 1. Za hodnotu parametru `url` je dosazena URL adresa lokalizačního souboru. Tato hodnota je uložena v záznamu pro daný jazyk v databázi a je dosazena při generování výsledné HTML stránky.

```
<script type="text/javascript">
    $(document).ready(function() {
        $('#example').dataTable( {
            "language": {
                "url": "dataTables.czech.lang"
            }
        });
    });
</script>
```

Výpis kódu 1: Načtení lokalizace pro plugin DataTables

V rámci prototypu administrace tyto lokalizační soubory představují jediný zdroj, který je třeba načíst ze serveru třetí strany. Soubory s překlady pro DataTables nejsou na serveru lokálně uloženy, protože nejsou součástí repositáře, který je dostupný ke stažení. Pokud by je autor stáhl ručně, mohlo by s budoucí verzí tohoto pluginu dojít k jejich desynchronizaci a bylo by třeba tyto překlady stáhnout manuálně znova. Všechny ostatní zdroje, jež jsou zapotřebí pro provoz IS jsou uloženy lokálně na serveru. To umožňuje provoz tohoto systému bez přístupu k síti Internet.

4.4 Nasazení prototypu

Vyvíjený informační systém bude před nasazením do produkčního prostředí provozován v testovacím režimu. Tento testovací provoz bude sloužit k detailnímu dopracování všech potřebných procesů souvisejících se správou systému a k otestování systému podmnožinou koncových uživatelů. Do testovacího prostředí bude nejprve nasazeno jádro systému, tedy administrace. Po dokončení implementace zbylých modulů budou tyto moduly integrovány do výsledného systému.

V době psaní této práce byl do testovacího provozu uveden modul administrace a čeká se na dokončení zbylých modulů. Tato sekce se zaměří na samotné testovací prostředí, proces nasazení aplikace a budoucí využití systému.

4.4.1 Heroku

Heroku je kontejnerová cloudová platforma jako služba (známá pod anglickou zkratkou PaaS). Vývojáři využívají Heroku k nasazení, správě a škálování

moderních aplikací. Heroku umožňuje zaměřit se především na vývoj samotné webové aplikace, samo poté řeší údržbu serverů či správu potřebné infrastruktury.

Heroku poskytuje základní služby zdarma, ty se hodí k testování aplikací či pro experimentování s provozem aplikací v cloudu. Pro produkční nasazení aplikací je poté doporučeno využít jedné z placených alternativ. Heroku nabízí 4 typy kontejnerů, kdy je v každém kontejneru provozována právě jedna aplikace. Pro testovací účely byl zvolen nejnižší kontejner `Free Dyno`, který je poskytován zdarma.

Kontejnery typu `Free Dyno` zůstávají aktivní 30 minut po poslední akci provedené uživatelem, poté dochází k jejich uspání. Každý uživatel má přiřazený limit hodin na měsíc, ze kterého je čerpáno při běhu těchto kontejnerů. Po vyčerpání limitu dochází k zastavení všech běžících kontejnerů typu `Free Dyno`. Účet, pod kterým běží tento IS má k dispozici 1000 hodin na každý měsíc a běží zde pouze jeden kontejner, nemůže tedy dojít k přečerpání limitu a zastavení služby.

Nevýhodu `Free Dyno` představuje skutečnost, že kontejner se musí po prvním přístupu uživatele nejdříve probudit, tato operace může trvat i desítky sekund. Pro účely ukládání dat je použita SQL databáze ClearDB, která je poskytována zdarma, ovšem obsahuje limit 5 MB uložených dat. Celková velikost projektu poté nesmí v rámci použitého kontejneru překročit velikost 500 MB. Výše uvedené omezení činí z tohoto kontejneru nevhodnou volbu pro produkční použití, pro testovací účely ale zcela dostačuje.

Prototyp administrace využívá šifrovanou komunikaci pomocí protokolu `HTTPS`, který je zabezpečen `SSL` certifikátem. Heroku automaticky udržuje tento certifikát platný a dle potřeby prodlužuje jeho platnost. Tato služba je součástí `Free Dyno` kontejneru a je tedy dostupná zdarma. Přístupové údaje do databáze a všechny konfigurační hodnoty jsou uloženy v administraci Heroku, do které lze získat přístup pouze se znalostí přihlašovacího jména a hesla, které zná pouze autor této práce. Testovací prostředí systému je tedy plně zabezpečeno a připraveno k testovacímu provozu.

4.4.2 Proces nasazení nové verze

Heroku podporuje širokou škálu způsobů nasazení webových aplikací. Při vývoji prototypu administrace byl aktivně využíván verzovací systém Git, který představuje jednu z podporovaných možností. Proces nasazení nové verze je zahájen nahráním zdrojových kódů do speciálního vzdáleného repositáře, který je spravován platformou Heroku. Pro vykonání této akce je třeba nejprve nainstalovat rozhraní příkazového řádku (anglicky CLI) Heroku a v jeho rámci se přihlásit do svého Heroku účtu. Případný útočník, který nezná přihlašovací údaje do Heroku, tedy nemůže nijak ovlivnit zdrojové kódy aplikace.

Heroku obsahuje tzv. sestavující balíčky (anglicky buildpacks). Jedná se o scripty, které jsou automaticky spuštěny při nahrání nové verze aplikace.

Tyto scripty jsou určeny k nainstalování závislostí třetích stran a k nastavení prostředí, v němž bude aplikace provozována. Prototyp administrace má zaregistrovány dva balíčky, konkrétně `heroku/php` a `heroku/nodejs`.

Proces začíná vykonáním příkazu `git push heroku master` v příkazové řádce. Tento příkaz nahraje aktuální verzi kódu do vzdáleného repositáře `heroku`. Po úspěšném nahrání těchto změn jsou postupně vykonány jednotlivé sestavující balíčky.

Prvním spuštěným balíčkem je `heroku/php`. Tento script nainstaluje nejnovější stabilní verzi PHP a webový server `nginx`. Následuje načtení a nainstalování závislostí z `composer.json`. Zde jsou přeskočeny všechny závislosti, jež jsou označeny jako vývojářské, jelikož je nastaveno produkční prostředí.

Po úspěšném dokončení prvního balíčku následuje vykonání `heroku/nodejs`. Výchozím správcem balíčků a závislostí pro Node.js je `npm`, tento systém ovšem využívá modernější a rychlejší `yarn`. Pomocí `yarn` je možné definovat a vytvářet skripty, které ulehčují a urychlují sestavení výsledné aplikace. Sestavovací balíček se pokusí zavolat skript `heroku-postbuild`, pokud takový script existuje. Tento IS pomocí tohoto skriptu provede následující operace:

1. První operaci představuje zavolání `yarn` skriptu `schema-update`. Pokud od poslední nasazené verze došlo ke změně entit, tento skript upraví databázové schéma tak, aby reflektovalo tyto změny.

2. Následující operace provede instalaci všech závislostí. Závislosti, které využívala předchozí verze, jsou načteny z mezipaměti a není třeba je tak stahovat opakovaně.

3. Na závěr je vyvolána úloha `default-production` v systému Gulp. Tato úloha zpracuje aktiva pro produkční prostředí, popis tohoto procesu lze nalézt v sekci 4.2.1.

Všechny výše zmíněné operace jsou provedeny automaticky při zavolání operace `git push`. Pokud by během zpracování jednoho z potřebných kroků došlo k chybě, akce nasazení by byla přerušena a byla by obnovena poslední funkční verze. Heroku dále umožňuje zapnutí režimu údržby, při kterém jsou všichni aktivní uživatelé přesměrování na statickou stránku údržby.

4.4.3 Budoucí využití

V současné době je dokončen prototyp modulu administrace vyvíjeného informačního systému. Tento prototyp je nasazen na platformě Heroku a je připraven pro testovací provoz. Pro účely testovacího provozu bude po dokončení zbylých modulů provedena jejich vzájemná integrace. Po spuštění testovacího provozu tak bude k dispozici prototyp výsledného IS skládajícího se z prototypů jednotlivých modulů.

V tomto testovacím provozu bude využíván jednou, či několika málo třídami. Jednotlivé moduly budou upravovány do konečné podoby dle získaných poznatků. Po ukončení testovací fáze bude systém připraven ve verzi `1.0` k ostrému spuštění.

Při návrhu a implementaci informačního systému byl kladen důraz na jeho modularitu, systém je tak připraven pro budoucí integraci nově vytvořených modulů. Tyto moduly mohou být vytvořeny jako semestrální či závěrečné práce studentů.

Kapitola 5

Testování a vyhodnocení

5.1 Webové prohlížeče a mobilní zařízení

Tato kapitola představí testování systému a naměřené výsledky. Budou testovány tři kategorie, funkcionalita systému v různých webových prohlížečích a mobilních zařízení, testování odezvy systému a uživatelské testování. První sekce představí testování systému průchodem scénářů, kde se autor práce zaměří především na chování systému při použití různých webových prohlížečů a mobilních zařízeních. Dále budou představeny výsledky testování odezvy systému v závislosti na celkovém počtu dat uložených v systému.

Třetí sekce se zaměřuje na uživatelské testování. Na základě jednotlivých případů užití vznikly scénáře, které byly otestovány uživateli. Uživatelského testování se zúčastnilo 6 účastníků, na jejich žádost však budou zjištěné výsledky anonymizovány. Každý účastník prošel všemi scénáři, tato sekce prezentuje zjištěné výsledky a průběh jednotlivých testů.

Na závěr budou vyhodnoceny kvality a nedostatky navrženého a implementovaného systému. Tyto závěry budou vyvozeny na základě výsledků jednotlivých testování.

5.1.1 Testované scénáře

Pro účely testování prohlížečů a mobilních zařízení bylo vytvořeno několik testovacích scénářů. Tyto scénáře se liší od scénářů uživatelského testování, které budou představeny později. Scénáře představené v této sekci se zaměřují na testování různých vlastností a funkcionalit systému v závislosti na velikosti obrazovky a použitém zařízení.

S1 – Vyhledání uživatele a jeho následné smazání Pro tento scénář bylo vytvořeno celkem 150 testovacích subjektů, tedy uživatelů. Cílem je nalezení studenta se jménem John Doe a jeho kompletní odstranění ze systému. K nalezení tohoto uživatele bude použit formulář pro vyhledávání uživatelů

na stránce přehledu všech uživatelů systému. Systém ovšem obsahuje učitele se stejným jménem, je tedy třeba nejprve vyhledat uživatele dle jména a poté na základě role vybrat správného kandidáta ke smazání.

S2 – Importování studentů Scénář S2 otestuje chování systému během provádění akce importování studentů. Tento scénář bude vykonán provedením následujících kroků:

1. Vytvoření nového procesu importování studentů.
2. Testování reakce a chování systému na pokus o nahrání nevalidních souborů s importovanými údaji.
3. Nahrání validního souboru a následná změna emailové adresy studenta Johna Doeho.
4. Dokončení procesu a následná kontrola očekávaných změn v systému.

S3 – Hromadné přiřazení studijních oborů Testovací třída `1.IT` obsahuje 25 studentů a třídního učitele. Sedm z těchto studentů má již zvolený a zapsaný studijní obor, zbylí studenti obor zapsán nemají. Třídní učitel se pokusí o hromadné přiřazení studijního oboru `Web a multimedia` studentům bez studijního oboru.

S4 – Vzhled komponent a jejich responsivita Tento scénář testuje vzhled vybraných komponent systému a jejich responsivní chování napříč webovými prohlížeči a mobilními zařízeními. Otestováno bude hlavní menu a tabulka obsahující seznam všech uživatelů. Na závěr bude testováno, jakým způsobem systém oznámí uživateli skutečnost, že je vyžadován JavaScript pro správnou funkcionalitu aplikace.

5.1.2 Webové prohlížeče

Otestovány budou celkem 4 webové prohlížeče. Jako referenční řešení bude použit průchod scénářů pomocí webového prohlížeče `Google Chrome v66.0` na operačním systému `macOS High Sierra`. Referenční řešení bude podrobně analyzováno a podloženo případnými záznamy obrazovky. Po vyhodnocení referenčního řešení budou jednotlivé scénáře otestovány na zbylých testovaných webových prohlížečích. Jedná se pak o aktuální verze prohlížečů `Microsoft Edge`, `Firefox` a `Opera`. Tyto prohlížeče budou testovány v prostředí `Windows 10`. Každý z těchto prohlížečů bude otestován stejnou metodikou jako referenční řešení, autor této práce poté uvede případné rozdíly a rozchody oproti referenčnímu řešení.

5.1.3 Mobilní zařízení

Scénáře představené v sekci 5.1.1 budou kromě webových prohlížečů otestovány pomocí mobilních zařízení. Pro testování budou použita následující mobilní zařízení:

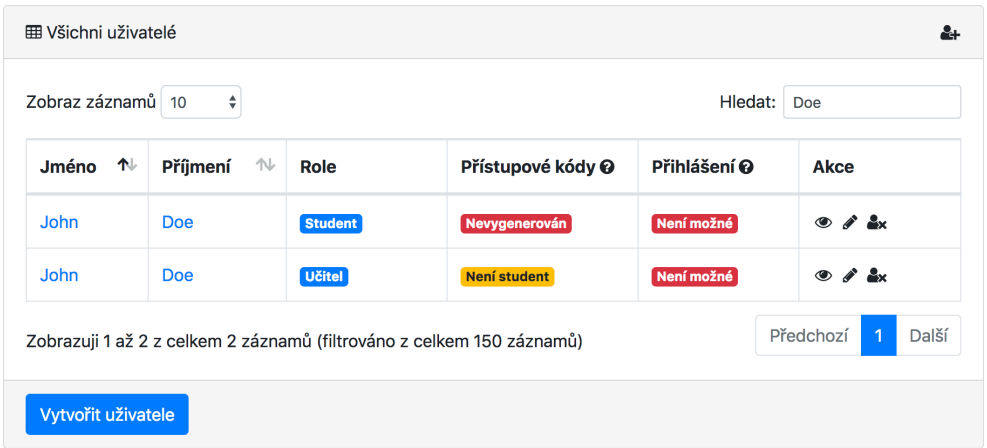
- Xiaomi Mi3
- iPhone 7 Plus
- Pixel 2
- iPad Pro

5.1.4 Testování scénáře S1

Tato sekce představí výsledky testování prvního scénáře. Jedná se o nalezení studenta dle jména a příjmení a jeho následné smazání ze systému. Při průchodu scénáře nejprve vzniklo referenční řešení, vůči kterému se následně testovali další webové prohlížeče a mobilní zařízení.

5.1.4.1 Referenční řešení

Nalezení studenta dle jména lze docílit více způsoby. Pro účely tohoto scénáře bylo použito vyhledávání v tabulce všech uživatelů systému. Po vyhledání příjmení systém prezentuje uživateli tabulku, jež je zobrazena na obrázku 5.1.



Obrázek 5.1: Filtrování uživatelů dle příjmení

Tabulka obsahuje dva záznamy, jelikož se v systému nachází dva uživatelé s příjmením `Doe`. Cílem scénáře bylo nalezení a odstranění studenta, jeho

odlišení od učitele nám umožní zobrazená role, kterou lze nalézt ve zkoumané tabulce. Seznam proveditelných akcí s vyhledaným uživatelem lze nalézt v posledním sloupci prezentované tabulky. Při najetí kurzoru na jednotlivé ikony se ihned zobrazuje jejich popis, poslední ikona poté signalizuje `Smazání uživatele`. Po kliknutí na tuto ikonu je uživatel přesměrován na potvrzovací obrazovku, kde se dočítá upozornění, že daná akce je nevratná. Po závěrečném potvrzení je student `John Doe` úspěšně odstraněn ze systému, uživateli je tato skutečnost oznámena pomocí tzv. zprávy rychlého oznámení (anglicky flash message).

5.1.4.2 Webové prohlížeče

Při testování ostatních prohlížečů nebyly zjištěny žádné anomálie ani jiné nesrovnalosti oproti referenčnímu řešení. Rozhraní aplikace vypadalo jednotně ve všech testovaných prohlížečích, chování aplikace pak také nebylo nijak pozměněno a ve všech testovaných prohlížečích se chovalo korektně. Pozorované výsledky jsou tedy zcela v souladu s očekáváním.

5.1.4.3 Mobilní zařízení

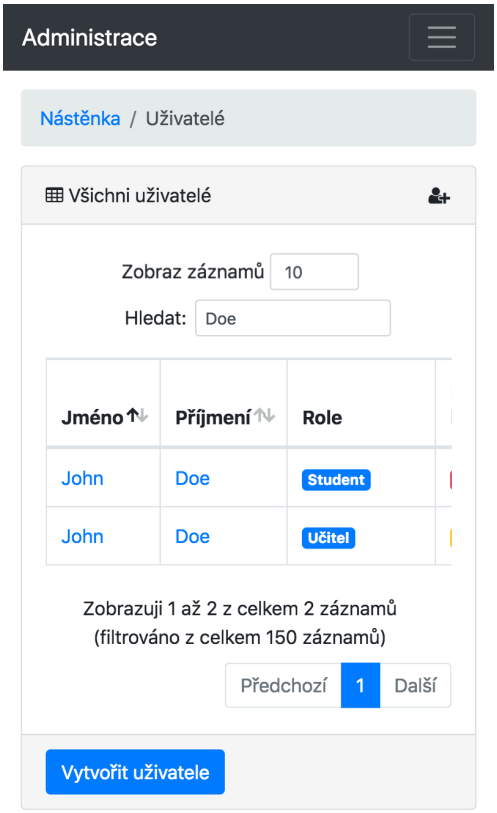
Rozhraní aplikace se podobalo referenčnímu řešení nejvíce během testování `iPadu Pro`. Díky nadstandardnímu rozlišení a rozměru obrazovky tohoto zařízení nedošlo k přepnutí uživatelského rozhraní do kompaktního režimu a uživatel tak mohl aplikaci využívat stejným způsobem, jako na stolním počítači. Jelikož mobilní zařízení neobsahují kurzor, popis akcí u ikon se zobrazí při podržení prstu na dané ikoně. To představuje jistou nevýhodu oproti zařízením, které kurzor podporují.

Při testování jednotlivých mobilních zařízení již došlo k přepnutí rozhraní do kompaktního režimu. V tomto režimu dochází ke schování menu, které lze zobrazit použitím tzv. „hamburger“ tlačítka, které se na mobilních zařízeních standardně používá právě pro tyto účely. Obrázek 5.2 zachycuje obrazovku `iPhone 7 Plus` při vyhledání uživatele dle příjmení. Rozložení rozhraní je oproti referenčnímu řešení podstatně změněno, stále však umožňuje jednoduchou navigaci a snadné ovládání aplikace. Formát zobrazených dat v tabulce zůstal nezměněn, tabulka se ovšem již celá nevejde na obrazovku, lze ji proto posouvat do stran dle potřeby.

Žádné potíže či neočekávané chování nebyly při testování mobilních zařízení zjištěny. Systém se choval zcela dle očekávání a umožnil rychlé nalezení studenta a jeho následné smazání.

5.1.5 Testování scénáře S2

Scénář importování studentů obsahuje 4 kroky, které budou testovány. Nejdříve bude vyhodnoceno referenční řešení, následovat poté bude testování pomocí



Obrázek 5.2: Filtrování uživatelů na iPhone 7 Plus

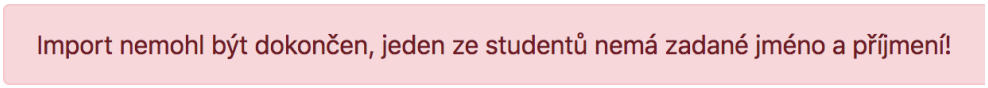
zbylých prohlížečů a mobilních zařízení. Stav systému bude po každém provedeném testování uveden do výchozího stavu.

5.1.5.1 Referenční řešení

Vytvoření nového procesu importování studentů probíhá rychle a intuitivně. Uživatel může tuto akci vykonat v sekci administrace `Importování studentů`. Po založení tohoto procesu je uživatel přesměrován na detail vytvořeného importu, odkud může vykonat další akce.

Systém umožňuje nahrání pouze XML souborů, při pokusu o nahrání souboru v jiném formátu vrací uživateli chybovou zprávu. Pokud je obsah souboru nevalidní, nebo neobsahuje všechny povinné atributy, systém opět upozorní uživatele chybovou zprávou. Obrázek 5.3 zachycuje chybovou zprávu, kdy se uživatel pokusil nahrát XML soubor, který pro jednoho ze studentů neobsahoval jméno a příjmení. Text chybové zprávy se mění dle typu problému, který nastal při zpracování souboru.

Po úspěšném nahrání souboru je uživatel přesměrován zpět na detail prováděného importování. Nahraní studenti jsou rozděleni do skupin dle tříd, což



Obrázek 5.3: Importování studentů – chybějící jméno uživatele

umožňuje jejich rychlé vyhledávání. Změna emailové adresy `Johna Doeho` proběhla pomocí jednoduchého formuláře zcela dle očekávání. Při pokusu o nastavení emailové adresy, kterou již využíval jiný student, vrátil systém správnou chybovou hlášku.

Po změně emailové adresy studenta byl dokončen proces importování studentů. Všichni studenti byli správně importování do systému, proces importování tedy funguje dle očekávání.

5.1.5.2 Webové prohlížeče

Při testování scénáře S2 pomocí zbylých testovaných prohlížečů lze pozorovat stejné výsledky jako u prvního scénáře. Systém se chová zcela dle očekávání, jedinou změnu představují odlišně vypadající formulářové prvky. Jedná se zejména o část formuláře sloužící pro výběr souborů. Funkčnost formuláře však i přes tuto skutečnost zůstává zachována a nepozorujeme tak žádné neočekávané chování.

5.1.5.3 Mobilní zařízení

Rozhraní pro importování studentů je plně responsivní a lze ho tak využít jak na tabletech, tak na mobilních telefonech. Systém se na `iPadu Pro` dle očekávání podobal standardnímu webovému prohlížeči a referenčnímu řešení nejvíce. Pro výběr souboru lze na mobilních zařízeních využít libovolný průzkumník adresářů, k dispozici je poté vždy minimálně základní aplikace, kterou poskytuje daný operační systém.

Při testování scénáře importování studentů pomocí mobilních telefonů nebyl dosažen komfort, který nabízí využití standardních webových prohlížečů či tabletů. To je způsobeno rozměrem obrazovky, kterou jsou mobilní telefony limitovány. Během procesu importování studentů systém prezentuje uživateli velké množství údajů, jejichž rozložení na menších obrazovkách nemůže dosahovat kvalit standardních webových prohlížečů.

Veškeré funkcionality systému a jeho chování ovšem zůstávají i v tomto případě neovlivněny a lze je tak za cenu nižšího uživatelského komfortu úspěšně vykonat. Systém tedy během testování prvních dvou scénářů fungoval zcela dle očekávání, což je do značné míry podpořeno responsivním uživatelským rozhraním, kterého bylo dosaženo i za pomoci Bootstrapu.

5.1.6 Testování scénáře S3

Testovací scénář S3 klade za cíl hromadné nastavení studijních oborů studentům jedné třídy. Funkcionalitu tohoto formuláře poskytuje skript napsaný v jazyce JavaScript, bez jeho podpory není použití tohoto formuláře umožněno.

5.1.6.1 Referenční řešení

Studijní obory lze studentům přiřadit vícero způsoby, pro účely tohoto scénáře je ovšem použit formulář pro hromadné přiřazení oborů. Tento formulář lze nalézt na profilu každé aktivní třídy a vyžaduje roli `Správce uživatelů`.

Formulář se nachází ve vyskakovacím okně, které se vykreslí po kliknutí na tlačítko `Přiřadit studentům obory`. Tento formulář byl představen k sekci 4.1.2, kde byl vyobrazen na obrázku 4.1. Jeho využití je intuitivní a přehledné, nejprve byli označeni všichni studenti bez přiřazeného studijního oboru za pomoci tlačítka `Studenty bez oboru`. Následně byl z poskytnuté nabídky oborů vybrán požadovaný obor, který byl studentům přiřazen po kliknutí na `Přiřadit`.

Po závěrečné kontrole byl formulář odeslán vyvoláním akce `Uložit provedené změny`, jež se nachází v zápatí daného modalu. Uživatel byl následně přesměrován zpět na profil třídy, kde již byly vidět provedené změny u studentů.

5.1.6.2 Webové prohlížeče a mobilní zařízení

Testování zbylých webových prohlížečů a mobilních zařízení proběhlo stejným způsobem, jako u předchozích scénářů. Zároveň autor práce došel k podobným závěrům, kdy se modal s formulářem choval ve všech testovaných prohlížečích totožně. Jediný rozdíl představoval vzhled prvků formuláře pro výběr oborů, který záleží dle konkrétní implementace webového prohlížeče. Funkčnost formuláře však nebyla žádným způsobem ovlivněna.

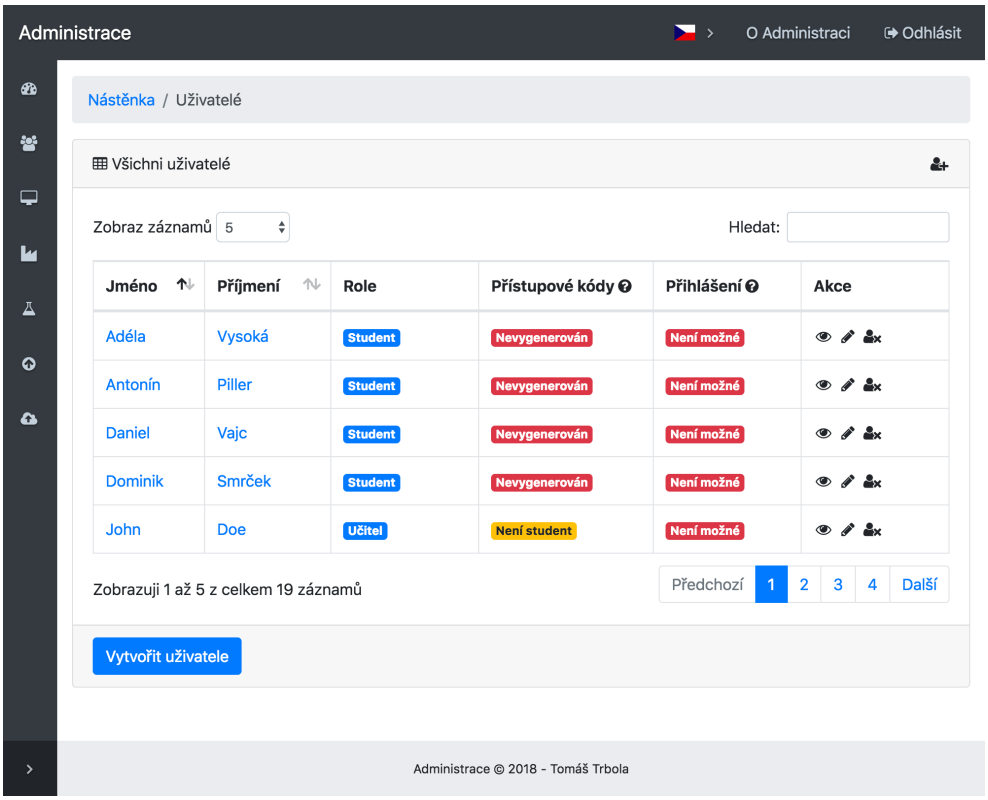
Během testování mobilních zařízení došel autor k podobným závěrům, jako u testování scénáře S2. Veškerá funkcionalita a responsivita byla zachována, ovšem na mobilních telefonech vzhledem k omezenému prostoru na obrazovce nedosahoval uživatelský komfort úrovně dosažené při testování webovými prohlížeči.

5.1.7 Testování scénáře S4

Čtvrtý scénář je zaměřen na testování vzhledu a responsivity komponent aplikace napříč webovými prohlížeči a mobilními zařízeními. Bylo analyzováno menu aplikace a tabulka obsahující všechny uživatele systému, kterou lze nalézt v sekci administrace `Všichni uživatelé`.

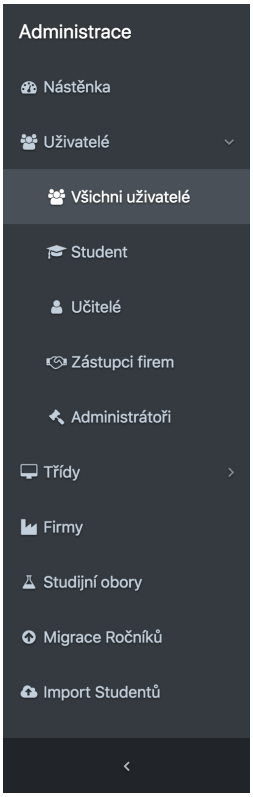
Menu aplikace se skládá ze dvou částí, horní a levé menu. Levé menu slouží jako navigace, obsahuje tedy odkazy na jednotlivé sekce administrace. Horní menu poté obsahuje rozhraní pro změnu lokalizace, odkaz na stránku `O Administraci` a formulář pro odhlášení uživatele.

Levé menu lze dále přepnout do zjednodušeného režimu, kdy dojde ke schování popisků a zůstanou tak zobrazeny pouze ikony jednotlivých sekcí. Obrázek 5.4 zachycuje obě analyzované komponenty, kdy je navíc levé menu ve stavu zjednodušeného režimu. Na obrázku 5.5 se poté nachází levé menu aplikace v plnohodnotném režimu. Lze zde pozorovat jednotlivé podsekce sekce `Uživatelé`. Pro zobrazení těchto podsekcí je ovšem v současné verzi prototypu vyžadován JavaScript, bez jeho podpory se tedy nelze plnohodnotně navigovat skrze IS. Jedná se tedy o jeden z bodů, kterými by se měl vývojář systému před vydáním finální verze systému zabývat, pro účely prototypu se ovšem jedná o zcela dostatečné řešení.



Obrázek 5.4: Rozhraní systému – menu a tabulky

Rozhraní systému je plně responsivní, přizpůsobuje se tedy šířce obrazovky či oknu prohlížeče a za žádné situace je nepřesahuje. Jednotlivé komponenty jsou plynule zužovány, ovšem existují zde jisté body zlomu (anglicky break-

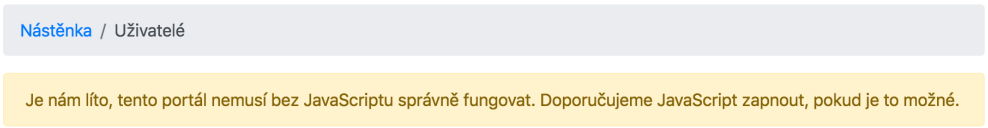


Obrázek 5.5: Rozhraní systému – levé menu aplikace

points), při kterých dochází ke skokové změně rozhraní. Nejvýraznější změna nastává, pokud šíře obrazovky klesne pod `992 px`. Pod touto hranicí dochází k přepnutí rozhraní do tzv. kompaktního režimu, kdy je například horní a levé menu sjednoceno a schováno. K jeho zobrazení poté slouží „hamburger“ tlačítko, které bylo zmíněno již dříve.

Při testování responsivity se menu i tabulka chovaly totožně na všech testovaných prohlížečích. Z mobilních zařízení byl `iPad Pro` jediné zařízení, které přesahovalo šíři `992 px` a podporovalo tak plnohodnotný režim rozhraní. Zbylá testovaná zařízení poté poskytovala rozhraní v kompaktním režimu.

Na závěr testování byly simulovány situace, kdy prohlížeč nepodporuje, či má explicitně zakázaný JavaScript. Systém v tomto případě zobrazil uživateli varovnou zprávu, která je zobrazena na obrázku 5.6. Tato zpráva je zobrazena pomocí HTML značky `<noscript>` a ve všech testovaných prohlížečích se zobrazila korektně.



Obrázek 5.6: Rozhraní systému – vypnutí JavaScriptu

5.2 Přístupnost

Administrace byla již od samotného návrhu vyvíjena s ohledem na dostupnost a přístupnost. Dle [56] by měl být kladen hlavní důraz při návrhu webové aplikace na její jednoduchost, přehlednost a jednoznačnost. Jedním z prvořadých faktorů, který rozhoduje, zda-li v návrhu webu něco funguje či ne, je skutečnost, zda-li uživatel musí u používání aplikace přemýšlet či nikoli [56]. Rozhraní aplikace bylo tedy vytvořeno především s ohledem na tato doporučení. Všechny ovládací prvky jsou označeny tak, aby byl na první pohled zřejmý jejich účel. Dále jsou rozmístěny takovým způsobem, aby je uživatel nemusel hledat, při implementaci rozhraní poté byly dodržovány standardní postupy a praktiky, na které jsou uživatelé zvyklí.

Implementovaný prototyp byl analyzován pomocí diagnostického nástroje `Lighthouse` od společnosti Google. Proběhla analýza ve třech kategoriích, konkrétně pak přístupnosti, nejlepších praktik (anglicky best practices) a optimalizace pro vyhledávače (SEO). Naměřené výsledky jsou zobrazeny na obrázku 5.7.

Obrázek 5.7: Lighthouse – analýza přístupnosti systému

Přístupnost systému byla ohodnocena 91 body ze sta. Zbývajících devět bodů bylo systému strženo za nedostatečný kontrast textu a pozadí tlačítek použitých na nástěnce (titulní stránka administrace). Prototyp pro získání vyššího kontrastu těchto tlačítek využívá stínů písma, ovšem `Lighthouse` tyto stíny nepočítá do výsledného kontrastu. Provedeme-li ruční měření, získáme hodnotu kontrastu `4.69:1`, což představuje dostatečnou hodnotu pro splnění normy `WCAG AA`.

Hodnocení nejlepších praktik či doporučených postupů poté ztratilo celkem 12 bodů. Body byly strženy za nevyužití `HTTP/2` a nepoužití pasivních

posluchačů ve skriptech. Heroku v současné době `HTTP/2` nepodporuje, tento problém tedy v rámci zvoleného testovacího prostředí nelze řešit. Nejedná se však o nijak závažný problém, který by bylo třeba řešit.

Nástroj `Lighthouse` dále udělil plný počet bodů v kategorii optimalizace pro webové vyhledávače. V této kategorii bylo zkoumáno, zda-li aplikace splňuje všechny podmínky pro zajištění ideálního indexování vyhledávači. Pro účely tohoto IS ovšem není tato kategorie stěžejní, jelikož se jedná o systém určený pro uzavřenou komunitu uživatelů.

5.3 Odezva systému

Tato sekce podrobí systém zátěžovému testování, ve kterém bude zkoumána doba odezvy systému v závislosti na množství zpracovávaných dat. Budou tedy simulovány situace, kdy systém obsahuje a spravuje různý počet studentů. Testovány budou tři funkcionality, které by mohly být těmito okolnostmi ovlivněny, konkrétně pak vytvoření nového studenta, vyhledání studentů dle jmen a převod studentů do vyšších ročníků.

Každá z funkcionalit bude testována ve třech situacích, kdy bude systém obsahovat 100, 1 000 a 5 000 studentů. Všichni tito studenti musí mít aktivní studenský status, systém neaktivní studenty ignoruje. Každý z testů bude proveden třikrát, naměřené hodnoty budou následně zprůměrovány.

5.3.1 Vytvoření nového studenta

Scénář vytvoření nového studenta má za úkol otestovat, zda-li počet studentů v systému ovlivní dobu vytvoření nového záznamu. Během procesu vytváření studenta se kontroluje, zda-li jsou zadaná emailová adresa a synchronizační kód unikátní. Tato kontrola unikátnosti je závislá na počtu studentů v databázi, jedná se tedy o vhodný scénář k otestování.

Tabulka 5.1 zobrazuje všechny naměřené hodnoty v průběhu testování a jejich aritmetický průměr. Lze pozorovat skutečnost, že s rostoucím počtem uživatelů systému roste mírně čas vykonávání operace. Nárůst časové náročnosti se ovšem pohybuje v jednotkách až desítkách milisekund, při 50 násobném zvýšení počtu uživatelů se průběh vytvoření nového studenta prodloužil pouze o 9 ms.

Tabulka 5.1: Odezva systému – vytvoření nového studenta

	Průběh č. 1	Průběh č. 2	Průběh č. 3	Průměr
100 uživatelů	191 ms	185 ms	185 ms	187 ms
1 000 uživatelů	187 ms	187 ms	184 ms	186 ms
5 000 uživatelů	188 ms	193 ms	207 ms	196 ms

5.3.2 Filtrování tabulky studentů

Tabulka obsahující přehled studentů provádí modifikace zobrazených dat na straně serveru. Tento scénář testuje rychlost vyhledávání studentů dle jména či příjmení. Testovány byly tři situace, kdy systém obsahoval 100, 1 000 a 5 000 studentů. Naměřené výsledky lze pozorovat v tabulce 5.2.

Tabulka 5.2: Odezva systému – filtrování tabulky studentů

	Průběh č. 1	Průběh č. 2	Průběh č. 3	Průměr
100 uživatelů	239 ms	200 ms	199 ms	213 ms
1 000 uživatelů	247 ms	208 ms	226 ms	227 ms
5 000 uživatelů	277 ms	284 ms	263 ms	275 ms

Z naměřených dat vyplývá, že rostoucí počet uživatelů systému zpomaluje operaci filtrování tabulky. Při zvýšení uživatelů na 10 násobek došlo ke zpomalení filtrování o 6,57 %, při 50 násobném zvýšení se poté jednalo o zpomalení o 29 %. V nejhorším případě činila naměřená hodnota 284 ms, i ta je ovšem pro účely tohoto systému zcela vyhovující.

5.3.3 Převod studentů do vyšších ročníků

Tento scénář testuje rychlost systému při provádění operace převodu studentů a tříd do vyšších ročníků. Pro účely tohoto testování bylo vytvořeno více tříd, kdy se každé třídy účastní 25 studentů a jeden až tři třídní učitele. Pro test s 1 000 studenty bylo tedy vytvořeno tříd 40, pro 5 000 studentů pak bylo tříd vytvořeno 200.

Výsledky jednotlivých průběhů spolu s jejich průměry lze nalézt v tabulce 5.3. Zde lze pozorovat značné zpomalení oproti předchozím scénářům. Při testu se sto uživateli je rychlost provedení operace ještě přiměřená, pro náročnější situace již musí uživatel administrace čekat po dobu několika vteřin.

Tabulka 5.3: Odezva systému – převod do vyšších ročníků

	Průběh č. 1	Průběh č. 2	Průběh č. 3	Průměr
100 uživatelů	492 ms	442 ms	517 ms	484 ms
1 000 uživatelů	1 719 ms	1 872 ms	1 834 ms	1 808 ms
5 000 uživatelů	7 342 ms	7 481 ms	6 952 ms	7 258 ms

Proces převodu studentů do vyšších ročníků je optimalizován, časová náročnost je zde způsobena návrhem systému a způsobem, jakým Doctrine načítá potřebná data. Systém musí při převodu načíst každého dotčeného studenta a aktualizovat potřebné údaje. Konkrétně se jedná o změnu údajů in-

stance entity reprezentující studentovu účast v dané třídě. Tuto entitu je třeba archivovat a vytvořit novou, která bude studenta reprezentovat v následující třídě.

Provádění akce převodu studentů do vyšších ročníků je plánováno jednou ročně, v rámci prototypu lze tedy čas nutný k provedení této akce akceptovat. V produkčním prostředí nebude systém dosahovat počtu studentů, se kterým byl testován, akce bude tedy provedena znatelně rychleji.

Systém při odeslání formuláře uživatele vždy upozorní, že byl jeho požadavek na odeslání akceptován a formulář je zpracováván, uživatel tedy vždy ví, v jakém stavu se aplikace po pokusu o odeslání nachází. Zajímavé řešení této situace by mohl představovat interaktivní prvek, který by uživatele informoval, jaká třída je aktuálně zpracovávána.

5.4 Uživatelské testování

Implementovaný prototyp administrace byl uživatelsky otestován v testovacím režimu. Testování se zúčastnilo 5 dobrovolníků, na žádosti několika z nich ovšem byly zjištěné výsledky anonymizovány. Na základě případů užití vzniklo 7 testovaných scénářů, každý účastník poté prošel všemi těmito scénáři.

Autor seznámil každého z účastníků s prototypem systému, jeho účelem a účely jednotlivých scénářů. Dále bylo účastníkovi oznámeno, že se jedná především o testování systému a ne účastníka samotného. Během provádění jednotlivých scénářů autor do průběhu testování nijak nezasahoval, výjimku představovaly situace, kdy testovaný subjekt zvolil nestandardní postup či se během průchodu scénářem ztratil. Kladené otázky poté byly zaměřeny na pochopení myšlenkového pochodu účastníka.

Po dokončení každého scénáře byl účastník požádán o ohodnocení intuitivnosti průchodu na škále od jedné do deseti, kdy deset bodů znamená maximální intuitivnost. Tato sekce dále u každého účastníka testování uvede případy, kdy se tento subjekt dostal do neočekávané situace, či zvolil nestandardní postup.

5.4.1 Účastníci testování

Správný výběr účastníků tvoří jeden ze základních předpokladů pro úspěšné testování produktu. Jejich nevhodným výběrem, či neodhadnutím optimálního počtu může dojít k nežádoucímu zkreslení výsledků. S ohledem na rozsah testované aplikace a počet testovaných scénářů byl zvolen ideální počet účastníků na čtyři až šest osob.

Dle [57] tvoří 5 testovaných subjektů ideální počet pro uživatelské testování. Při testování do počtu pěti uživatelů přináší jednotliví účastníci nové poznatky a zjištění, nad tento počet se již ovšem čas investovaný do testování dalších účastníků nevyplácí.

Účastník 1 Student ČVUT, 22 let, mezi jeho koníčky patří návrh a realizace webových aplikací, stavění a následné přetaktování počítačů. Převážnou část svého volného času tráví na počítači, ovládá širokou škálu počítačových programů na vysoké až profesionální úrovni. Má zkušenosti s vývojem i správou informačních systémů, především jejich vývoj pak představuje jeho hlavní pracovní náplň.

Účastník 2 Svatební a rodinná fotografka, 45 let, mezi její koníčky patří fotografování a chov domácích zvířat. Ovládá běžné kancelářské programy na základní úrovni, software určený pro úpravu a zpracování fotografií poté na úrovni odborné až profesionální. S informačními systémy a jejich správou má základní zkušenosti, ovšem nevyužívá je pravidelně.

Účastník 3 Zaměstnanec v domově pro osoby se zdravotním postižením, 52 let, mezi jeho koníčky patří cyklistika, fotografování veteránů a jiných mechanických strojů. V oblasti počítačových programů a kancelářských nástrojů má rámcové znalosti, výjimku zde představují programy pro úpravu fotografií, které používá na odborné úrovni. Se správou informačních systémů nemá zkušenosti.

Účastník 4 Zaměstnankyně ve státním úřadu pro jadernou bezpečnost, 26 let, jejími koníčky jsou tanec a hraní na kytaru. Používá počítačové programy na základní úrovni, se specializovanými programy jako AutoCAD či Prezi má odborné zkušenosti. Se správou informačních systémů má pouze rámcové zkušenosti a nespravuje je pravidelně, díky své práci k nim ovšem nemá daleko.

Účastník 5 Projektový manager a vývojář ve společnosti Appmine, 22 let, zajímá se o vývoj aplikací, vědu, techniku a osobní rozvoj. Běžné počítačové programy používá na vysoké úrovni, software specializovaný na vývoj aplikací poté na úrovni profesionální. Do styku s informačními systémy přichází díky práci pravidelně, ovládá je poté na pokročilé úrovni.

5.4.2 Testované scénáře

Druhý důležitý předpoklad pro úspěšné testování představují samotné testované scénáře. Za účelem testování prototypu vzniklo 7 scénářů, které vychází z analyzovaných případů užití, jenž byly představeny v sekci 2.4. Testované scénáře tedy simulují reálné situace, se kterými mohou uživatelé administrace přijít do styku.

Naměřené výsledky a vyvozené závěry budou sloužit pro zhodnocení stavu implementovaného prototypu. Dále budou využity k odstranění nalezených chyb prototypu a zlepšení uživatelské zkušenosti při používání systému.

Každý představený scénář obsahuje zkratku, pod kterou na něj bude později odkazováno (SUT – scénář uživatelského testování). Scénář nejprve uvede výchozí stav a nastavení systému, následovat budou očekávané kroky provedení scénáře. Každý scénář bude zakončen očekávaným stavem systému po úspěšném dokončení scénáře. Délka doby trvání každého ze scénářů by neměla překročit 5–10 minut.

SUT1 – Provedení importu studentů Účelem tohoto scénáře je provedení operace importu studentů. Testovanému subjektu bude před zahájením testování poskytnut XML soubor, jenž obsahuje potřebné údaje pro úspěšné provedení operace. Před začátkem scénáře je systém ve stavu, kdy je uživatel přihlášen pomocí administrátorského účtu s plnými oprávněními a nachází se na domácí stránce administrace.

Očekávané kroky scénáře:

1. Nalezení položky `Import studentů` v levém menu a následný vstup do této sekce.

2. Založení nového procesu importování kliknutím na tlačítko `Vytvořit nový import`.

3. Pokračování kliknutím na tlačítko `Načíst více studentů`.

4. Přečtení instrukcí na obrazovce, zvolení a nahrání požadovaného souboru.

5. Kontrola načtených studentů a následné dokončení procesu kliknutím na tlačítko `Dokončit import`.

Po ukončení scénáře systém oznámí uživateli pomocí zprávy rychlého oznámení skutečnost, že byla operace importování studentů úspěšně dokončena. Systém po dokončení této operace obsahuje všechny importované záznamy.

SUT2 – Převod studentů do vyšších ročníků Účelem tohoto scénáře je provedení operace převodu studentů a tříd do vyšších ročníků. Převod se řídí běžnými pravidly, kdy má být každá třída ve čtvrtém ročníku archivována, zbývající třídy mají být převedeny. Uživateli je však oznámen zvláštní požadavek, kdy má být třída 4.IT převedena do vyššího ročníku na místo archivace. Před začátkem scénáře je uživatel přihlášen pomocí administrátorského účtu s plnými oprávněními a nachází se na detailu dokončeného importu ze scénáře SUT1.

Očekávané kroky scénáře:

1. Nalezení položky `Migrace ročníků` v menu a následný vstup do této sekce.
2. Vytvoření nového procesu převodu pomocí patřičného tlačítka.
3. Nalezení třídy 4.IT v zobrazeném přehledu tříd.
4. Zrušení položky `Archivovat` u této třídy.
5. Dokončení procesu kliknutím na tlačítko `Migrovat`.

Po zakončení scénáře systém signalizuje uživateli úspěšné provedení operace převodu do vyšších ročníků. Všechny třídy označeny pro archivaci budou archivovány, pro zbylé třídy bude vytvořen historický záznam a následně budou převedeny do vyššího ročníku.

SUT3 – Hromadné přiřazení studijních oborů Scénář SUT3 testuje hromadné přiřazení studijních oborů studentům třídy 5.IT, která vznikla provedením scénáře SUT2. Uživatel je před začátkem scénáře stále přihlášen pomocí administrátorského účtu a nachází se na detailu dokončeného procesu převodu studentů do vyšších ročníků. Třída 5.IT obsahuje 25 studentů, z nichž 12 studentů již má zvolený studijní obor `Web a multimedia`. Cílem tohoto scénáře je přiřazení studijního oboru `Znalostní inženýrství` všem zbývajícím studentům s výjimkou studenta John Doe, kterému bude přiřazen studijní obor `Teoretická informatika`.

Očekávané kroky scénáře:

1. Vstup na detail třídy 5.IT, existuje více správných postupů:
   - Vyhledání třídy 5.IT v přehledu migrovaných tříd a následný přístup na její detail pomocí nalezeného odkazu v tabulce.
   - Vstup do sekce `Všechny třídy` či `Aktivní třídy`, vyhledání třídy 5.IT v seznamu tříd a následný vstup na detail této třídy.
2. Pokračování kliknutím na tlačítko `Přiřazení studijních oborů` v horní sekci detailu třídy.
3. Nalezení studenta John Doe a přiřazení oboru `Teoretická informatika`.
4. Označení zbývajících studentů pomocí tlačítka `Studenty bez oboru`, zvolení oboru `Znalostní inženýrství` a přiřazení oboru studentům pomocí tlačítka `Přiřadit`.
5. Dokončení procesu pomocí tlačítka `Uložit provedené změny`.

Systém po provedení akce uživateli oznámí tuto skutečnost pomocí zprávy rychlého oznámení. Uživatel se po dokončení akce nachází na profilu dané třídy, studijní oboru u všech studentů jsou úspěšně změněny.

SUT4 – Dohledání historických údajů studenta Účelem tohoto scénáře je nalezení historických údajů studenta Johna Doeho. Tento student obsahuje dva přístupové kódy, jeden aktivní a jeden již použitý. Uživatel má za úkol nalézt datum aktivace použitého přístupového kódu a název předchozí studentovy třídy. Uživatel se na začátku scénáře nachází na detailu třídy 5.IT a má plná administrátorská oprávnění.

Očekávané kroky scénáře:

1. Přístup na profil studenta Johna Doeho, toho může být docíleno více způsoby:
   - Nalezení studenta na stránce, na níž se uživatel aktuálně nachází pomocí vyhledání v tabulce studentů.
   - Navigace do sekce `Všichni uživatelé` či `Studenti`, následné vyhledání studenta a přístup na jeho profil.
2. Nalezení sekce `Třídy` na profilu studenta, vyhledání dané třídy a zjištění jejího názvu.
3. Nalezení sekce `Přístupové kódy` na profilu studenta, vyhledání požadovaného záznamu a zjištění data aktivace.

SUT5 – Pátrání po třídním učiteli Účelem tohoto scénáře je vypátrání třídního učitele archivované třídy a následná práce s tímto učitelem. Uživatel bude dotázán, aby nalezl třídu 2.IT, která se konala v letech 2013–2014 a zjistil jméno jejího třídního učitele. Dále bude požádán, aby zjistil, zda-li tento učitel byl v zadaném časovém rozmezí třídním učitelem nějakých dalších tříd.

Uživatel se bude před začátkem tohoto scénáře nacházet na domovské stránce administrace, k dispozici bude mít plná administrátorská oprávnění. Očekávané kroky scénáře:

1. Vstup do sekce `Archivované třídy` v levém menu.
2. Vyhledání třídy dle názvu, následný výběr správného záznamu dle data konání třídy.
3. Nalezení třídního učitele, lze nalézt přímo v dané tabulce či na profilu třídy.
4. Vstup na profil třídního učitele.
5. Nalezení sekce `Třídní učitel - učil třídy`, na základě zjištěných údajů konstatovat, že žádné další třídy v roce 2013–2014 tento učitel neučil.

SUT6 – Omezení pravomocí podřízeného administrátora Pro účely tohoto scénáře byli vytvořeni dva administrátoři, Samuel Pilný a Patrik Šimánek. Samuel je podřízeným administrátorem Patrika, účelem tohoto scénáře je omezení jeho pravomocí. Uživateli bude sděleno jméno nadřízeného administrátora, tedy Patrik Šimánek. Systém bude obsahovat přes 100 administrátorů, uživatel tedy bude muset využít funkce vyhledávání.

Uživatel bude přihlášen pomocí účtu s administrátorským oprávněním, scénář bude začínat na domovské stránce administrace. Administrátor Samuel Pilný má udělen plný přístup, cílem tohoto scénáře je nechat mu pouze roli `Správce uživatelů`. Očekávané kroky scénáře:

1. Vstup do sekce `Uživatelé - Administrátoři`.
2. Nalézt administrátora se jménem Patrik Šimánek pomocí funkce vyhledat.
3. Vstup na profil tohoto administrátora.
4. Nalezení sekce `Podřízení administrátoři`.
5. Nalezení podřízeného administrátora v této tabulce, kliknutí na tlačítko `Upravit administrátora`.
6. Zrušení volby `Plný přístup`, zaškrtnutí `Správa uživatelů`.
7. Potvrzení formuláře tlačítkem `Uložit změny`.

Po úspěšném dokončení scénáře systém oznámí uživateli skutečnost, že oprávnění administrátora byly změněny. Uživatel bude automaticky přesměrován na administrátorský profil dotčeného administrátora, kde lze zkontrolovat provedené úpravy.

SUT7 – Hromadné generování přístupových kódů Účelem tohoto scénáře je vygenerování přístupových kódů všem studentům třídy 5.IT. Uživatel bude mít dále za úkol tyto přístupové kódy vytisknout a rozeslat studentům na emailové adresy. Pro testování tohoto scénáře bude uživatel přihlášen pomocí účtu třídního učitele této třídy. Scénář začíná na domovské stránce administrace.

Očekávané kroky scénáře:

1. Nalezení sekce administrace `Aktivní třídy` a následný vstup do této sekce.
2. Vyhledání a navštívení profilu třídy 5.IT.
3. Zvolení možnosti `Generovat přístupové kódy`.

4. Potvrzení doby platnosti přístupových kódů opětovným stiskem tlačítka `Generovat přístupové kódy`.

5. Stisknutí tlačítka `Vytisknout přístupové kódy` následované stisknutím tlačítka `Odeslat na E-mail`.

Scénář je ukončen rozesláním přístupových kódů na emailové adresy studentů. Systém uživateli oznámí úspěšně dokončenou akci pomocí zprávy rychlého oznámení. Uživatel se po ukončení scénáře nachází na detailu třídy, kde kontroluje, zda-li všichni studenti vlastní přístupový kód.

5.4.3 Výsledky testování

Tato sekce představí výsledky uživatelského testování, které byly zjištěny na základě průchodu jednotlivých scénářů. Každý účastník testování ohodnotil intuitivnost průchodu scénáře od 1 do 5 body, kdy udělení 5 bodů znamená maximální intuitivnost. Na základě zjištěných poznatků byly provedeny patřičné úpravy systémy, který je v současné době připraven na zahájení testovacího provozu.

Průchod účastníka č. 1 První účastník má bohaté zkušenosti s Bootstrap frameworkem, vzhledem i chováním systému. Testování probíhalo bez komplikací, u scénáře SUT2 a SUT3 ovšem vznesl účastník několik poznámek. Udělené body tímto účastníkem lze nalézt v tabulce 5.4.

Tabulka 5.4: Uživatelské testování – hodnocení scénářů účastníkem č. 1

	SUT1	SUT2	SUT3	SUT4	SUT5	SUT6	SUT7
Hodnocení	5	4	4	5	5	5	5

První poznámka se týká rozhraní převodu tříd do vyšších ročníků. To je sice přehledné a jednoznačné, mohlo by ale být více kompaktní. Pokud systém obsahuje více aktivních tříd, mohlo by toto rozhraní dále podporovat stránkování výsledků.

Druhá poznámka se týká formuláře hromadného přiřazení studijních oborů. Při první uživatelově interakci s tímto formulářem by mu mohl být nabídnut krátký návod či instrukce pro správnou obsluhu. Jedná se o doporučení testovaného subjektu, situaci tedy není třeba prioritně řešit, systém by ovšem tuto funkcionalitu mohl do vydání verze 1.0 implementovat.

Průchod účastníka č. 2 Účastník č. 2 neměl před zahájením testování předchozí zkušenosti s použitým Bootstrap frameworkem, se vzhledem komponent a rozložením uživatelského rozhraní se však rychle seznámil. Testování

probíhalo hladce, u scénářů SUT2, SUT4, SUT6 a SUT7 nenarazil na žádné nejasnosti. Body, kterými uživatel ohodnotil jednotlivé scénáře lze nalézt v tabulce 5.4.

Tabulka 5.5: Uživatelské testování – hodnocení scénářů účastníkem č. 2

	SUT1	SUT2	SUT3	SUT4	SUT5	SUT6	SUT7
Hodnocení	3	5	4	5	5	5	5

První situace, kdy si uživatel nebyl jistý následujícím krokem nastala ve scénáři importování studentů (SUT1). Po založení procesu importování si testovaný subjekt nebyl jistý, jaká akce by měla následovat. Tento problém bude vyřešen úpravou textu tlačítka `Načíst více studentů`, či přidáním návodu na tuto stránku.

Druhá nejasnost se vyskytla při testování scénáře přiřazování studijních oborů (SUT3). Uživatel byl schopný se samostatně dostat na profil třídy a otevřít formulář pro přiřazení studijních oborů, zde však místo ovládacíche tlačítek začal vybírat jednotlivé studenty bez studijního oboru ručně. Na dotaz přísedícího uvedl, že si nebyl vědom, že se jedná o ovládací tlačítka, z jejich popisu to jasně nevyplývalo. Tento problém bude odstraněn úpravou textu tlačítek a jejich představením v záhlaví daného formuláře.

Poslední incident nastal v scénáři SUT5. Zde byl uživatel schopný provést správně všechny kroky scénáře, upozornil však na zavádějící popis tabulek s historií tříd třídního učitele. Tento problém bude vyřešen provedením vhodné úpravy daného textu.

Průchod účastníka č. 3 Účastník č. 3 kromě již objevených problémů objevil několik nových. Díky absenci zkušeností se správou či používáním informačních systémů se tento účastník ze začátku testování v systému ztrácel a trvalo několik minut, než se začal orientovat. První dva scénáře tedy neprobíhaly zcela optimálně a účastník musel být výjimečně nasměrován správným směrem.

Po dokončení prvních třech scénářů se však již zúčastněný začal v systému plně orientovat a zbývající scénáře dokončil bez dalších komplikací. První komplikace nastala hned při počáteční interakci uživatele se systémem, uživatel si nevšiml možnosti přepnout jazyk rozhraní do českého jazyka. Následně objevil tento účastník již známé problémy, kdy mu např. dělalo potíže rozhraní u převodu ročníků či ovládací tlačítka hromadného přiřazení studijních oborů.

Během testování účastníků č. 2 a č.3 bylo pozorováno podobné chování. Při průchodu prvních scénářů se uživatelé nejdříve museli rozkoukat a seznámit se systémem, poté průchod zbývajících scénářů nedělal žádné problémy. Hodnocení účastníka č. 3 lze nalézt v tabulce 5.6.

Tabulka 5.6: Uživatelské testování – hodnocení scénářů účastníkem č. 3

	SUT1	SUT2	SUT3	SUT4	SUT5	SUT6	SUT7
Hodnocení	1	3	4	5	5	4	5

Průchod účastníka č. 4 Testování čtvrtého účastníka probíhalo hladce, nepřineslo však žádné nové poznatky. Problémové úseky testování objevené dříve se projevily i zde. Systém byl před začátkem testování přepnut do anglické lokalizace, uživatel nebyl schopen nalézt a přepnout jazyk rozhraní.

Převod tříd do vyšších ročníků představoval jediný vážnější problém, uživateli zde dělalo problém zorientování se v použitém uživatelském rozhraní. Po té, co se uživatel zorientoval však již bez problému scénář i všechny zbývající bez problému dokončil. Body udělené scénářům uživatelem lze nalézt v tabulce 5.7.

Tabulka 5.7: Uživatelské testování – hodnocení scénářů účastníkem č. 4

	SUT1	SUT2	SUT3	SUT4	SUT5	SUT6	SUT7
Hodnocení	5	3	4	5	5	5	5

Průchod účastníka č. 5 Účastník č. 5 má široké znalosti v oblasti počítačových programů a informačních systémů. Při průchodu scénářů nenastaly žádné komplikace, díky zkušenosti účastníka s návrhem uživatelského rozhraní ovšem vznesl během testování několik poznámek a návrhů.

Tyto poznámky a návrhy se týkaly problémů, které již objevili předchozí účastníci testování. Jednalo se zejména o popisy tlačítek, která je třeba poupravit a rozhraní pro převod studentů do vyšších ročníků. Testovaný účastník i přes tyto poznámky bez problémů prošel stanovené scénáře, systém ohodnotil body, jenž lze nalézt v tabulce 5.8

Tabulka 5.8: Uživatelské testování – hodnocení scénářů účastníkem č. 5

	SUT1	SUT2	SUT3	SUT4	SUT5	SUT6	SUT7
Hodnocení	5	4	4	5	5	5	5

5.5 Vyhodnocení

Tato sekce se zabývá vyhodnocením implementovaného prototypu. Systém byl otestován v oblasti responsivity rozhraní na mobilních zařízeních, odezvy systému, přístupnosti a na závěr byl uživatelsky otestován. Na základě těchto zjištění budou provedeny patřičné úpravy systému, který poté bude připraven k zahájení testovacího provozu.

První provedené testy testovaly funkce systému a jeho responsivitu na populárních webových prohlížečích a mobilních zařízeních. Díky návrhu systému, použitým knihovnám a vhodné implementaci se systém choval konzistentně na všech testovaných zařízeních. Nejhůře zde vyšly mobilní telefony, kde sice systém fungoval zcela korektně, limitujícím faktorem se však jevily malé obrazovky, které neposkytovaly komfort obrazovek standardních rozlišení.

Přístupnost prototypu byla testována nástrojem Lighthouse ve třech kategoriích a to přístupnosti, nejlepší praktiky a optimalizace pro vyhledávače. Ve všech těchto kategoriích byl systém ohodnocen téměř plným počtem bodů. Zvláštní důraz byl kladen na kontrast textů, kdy systém splňuje minimální doporučený kontrast ve všech situacích.

Po testování přístupnosti bylo provedeno měření odezvy systému. Cílem tohoto testování bylo zjistit, jak se bude systém chovat po několika letech provozu, kdy bude obsahovat tisíce studentů a stovky tříd a učitelů. Všechny testované situace s výjimkou převodu ročníků byly ovlivněny zcela minimálně, docházelo ke zpomalení v řádu několika desítek milisekund. Převod tříd do vyšších ročníků byl ovlivněn nejvíce, zde při 5 000 studentech trvalo provedení operace řádově sekundy. Tato akce bude ovšem prováděna zcela výjimečně (zpravidla jednou do roka). Zpomalení je způsobeno zejména použitými technologiemi, nejedná se však o chybu a v rámci prototypu lze toto chování akceptovat.

Na závěr byl systém podroben uživatelskému testování. Na základě případů užití prezentovaných v sekci 2.4 vzniklo 7 scénářů uživatelského testování. Následoval průchod těchto scénářů 5 dobrovolníky, na jejichž přání ovšem byly naměřené výsledky anonymizovány. Průběh testování všech účastníků obsahoval stejné charakteristiky, kdy se testovaný subjekt během prvních scénářů seznámil se systémem a následně bez komplikací dokončil zbývající scénáře.

Během uživatelského testování bylo zjištěno několik nedostatků, na které upozornilo více účastníků. Obvykle se jednalo o ne zcela ideální texty tlačítek či situace, kdy si uživatel nebyl jist dalším krokem. Všechny uživatelské poznatky byly zaznamenány a aktuálně na jejich základě probíhají patřičné úpravy prototypu.

Systém je k datu odevzdání této práce úspěšně nasazen v testovacím režimu, kdy obsahuje záznamy 8 tříd a celkem 180 studentů. Po osvědčení systému a opravě případných chyb dojde k nasazení do produkčního prostředí, kdy budou do systému importováni všichni stávající žáci školy. Očekává se budoucí rozšíření systému pomocí modulů, které mohou být vypracovány jako semestrální či závěrečné práce jiných studentů.

Závěr

Cílem práce bylo analyzovat požadavky, navrhnout a implementovat prototyp webové administrace připravovaného informačního systému pro Střední školu a Vyšší odbornou školu reklamní a umělecké tvorby Michael. Na administraci byly klady požadavky správy uživatelů, přístupových účtů, tříd a oborů vyučovaných na škole. Dále měla administrace poskytnout rozhraní pro převod tříd a studentů do vyšších ročníků a rozhraní pro import studentů ze systému Bakaláři. Administrace tvoří jádro IS, měla tedy použít modulární architekturu, která umožní budoucí rozšíření systému o další moduly.

Výsledná aplikace úspěšně splnila všechny tyto požadavky. Jejich analýza proběhla v rešeršní části práce, kde byl dále analyzován současný stav a již existující řešení, byly zde představeny použitelné technologie, případy užití a na závěr byl analyzován doménový model.

V návrhu systému byla na základě funkčních a nefunkčních požadavků zvolena ideální technologie pro realizaci prototypu. Dále zde bylo navrženo schéma databáze, modulární architektura a samotný návrh systému a jeho komponent. Na základě tohoto návrhu vznikl prototyp systému, který poskytuje veškeré požadované funkcionality. Mezi tyto funkcionality patří správa uživatelů, přihlašovacích účtů, tříd a oborů vyučovaných na škole. U studentů je dále vedena historie tříd, oborů a použitých přihlašovacích kódů. Administrace podporuje importování studentů ze systému Bakaláři pomocí XML souborů, které jsou tímto systémem generovány. Dále poskytuje rozhraní pro převod tříd do vyšších ročníků, správu firem a jejich zástupců.

Prototyp systému byl na závěr podroben testování přístupnosti, odezvy systému a kompatibility systému při použití populárních webových prohlížečů a mobilních zařízení. Na základě případů užití vznikly scénáře uživatelského testování, které otestovalo 5 dobrovolníků. Výsledky všech testů dopadly zcela dle očekávání, prototyp aplikace bude dále zdokonalen o poznatky z uživatelského testování.

Administrace je v současné době spolu s modulem pro správu závazků provozována v testovacím režimu, kdy obsahuje 8 tříd a celkem 180 studentů. Po ukončení a vyhodnocení testovacího režimu dojde k nasazení systému do produkčního prostředí, kdy budou importování zbývající studenti, třídy a učitelé z této školy. Systém je dále připraven na budoucí rozšíření pomocí modulů, jejichž snadné integrování vychází z navržené a použité architektury.

Bibliografie

1. *Michael - Střední škola a Vyšší odborná škola reklamní a umělecké tvorby - O škole* [online]. 2017 [cit. 2018-04-10]. Dostupné z: `http://www.skolamichael.cz/cz/o_skole`.
2. NEUMAJER, Ondřej. *Školní informační systémy* [online]. 2010 [cit. 2018-04-11]. Dostupné z: `http://ondrej.neumajer.cz/skolni-informacni-systemy/`.
3. NEUMAJER, Ondřej. *Jaký školní informační systém používáte?* [online]. 2015 [cit. 2018-04-11]. Dostupné z: `https://diskuze.rvp.cz/viewtopic.php?f=761&t=24223&view=viewpoll`.
4. BAKALÁŘI S.R.O. *Bakaláři – mezi školou a rodinou* [online]. 2017 [cit. 2018-04-11]. Dostupné z: `https://www.bakalari.cz/`.
5. BAKALÁŘI S.R.O. *Možnosti nasazení systému Bakaláři* [online]. 2017 [cit. 2018-04-11]. Dostupné z: `https://www.bakalari.cz/Static/cloud`.
6. BAKALÁŘI S.R.O. *Bakaláři – vybrané moduly systému* [online]. 2017 [cit. 2018-04-11]. Dostupné z: `https://www.bakalari.cz/Home/Modules`.
7. STATCOUNTER. *Podíl mobilních operačních systémů* [online]. 2017 [cit. 2018-04-11]. Dostupné z: `http://gs.statcounter.com/os-market-share/mobile/worldwide`.
8. MOBILE NATIONS LLC. Everything you need to know about the Google Play Store [online]. 2016 [cit. 2018-04-11]. Dostupné z: `https://www.androidcentral.com/google-play-store/home`.
9. GOOGLE COMMERCE LTD. *Bakaláři – oficiální aplikace (beta) – aplikace na Google Play* [online]. 2018 [cit. 2018-04-11]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.impire.bakalari.student`.

10. ŠKOLA ONLINE A.S. *ŠKOLNÍ INFORMAČNÍ SYSTÉM ŠKOLA ONLINE* [online]. 2018 [cit. 2018-04-11]. Dostupné z: www.skolaonline.cz/Skolni_informacni_system.aspx.

11. ŠKOLA ONLINE A.S. *CENÍK ŠKOLNÍHO INFORMAČNÍ SYSTÉMU ŠKOLA ONLINE* [online]. 2018 [cit. 2018-04-11]. Dostupné z: https://www.skolaonline.cz/Cen%C3%ADk.aspx.

12. ŠKOLA ONLINE A.S. *Škola OnLine - uživatelská příručka - ostatní moduly* [online]. 2017 [cit. 2018-04-11]. Dostupné z: https://aplikace.skolaonline.cz/SOL/dokumentace/KS/zamestnanci/index.html?ostatni.html.

13. ŠKOLA ONLINE A.S. *PŘEDNOSTI ŠKOLNÍHO INFORMAČNÍHO SYSTÉMU ŠKOLA ONLINE* [online]. 2018 [cit. 2018-04-11]. Dostupné z: https://www.skolaonline.cz/O%C5%A0koleOnLine/P%C5%99ednosti.aspx.

14. ŠKOLA ONLINE A.S. *ZAJIŠTĚNÍ BEZPEČNOSTI A DOSTUPNOSTI ŠKOLNÍHO INFORMAČNÍHO SYSTÉMU ŠKOLA ONLINE* [online]. 2018 [cit. 2018-04-11]. Dostupné z: https://www.skolaonline.cz/O%C5%A0koleOnLine/Bezpe%C4%8Dnost.aspx.

15. GOOGLE COMMERCE LTD. *Škola OnLine – aplikace na Google Play* [online]. 2018 [cit. 2018-04-11]. Dostupné z: https://play.google.com/store/apps/details?id=cz.skolaonline.mobile.

16. DM SOFTWARE S.R.O. *dm Software - Uživatelská příručka - Úvod* [online] [cit. 2018-04-11]. Dostupné z: https://aplikace.dmsoftware.cz/SOL/dokumentace/DM/zamestnanci/.

17. DM SOFTWARE S.R.O. *Všeobecné obchodní podmínky poskytování služeb hostování softwarové aplikace dm Software* [online]. 2016 [cit. 2018-04-11]. Dostupné z: https://portal.dmsoftware.cz/Portals/portaldm/smlouva/V%C5%A1eobecn%C3%A9%20obchodn%C3%AD%20podm%C3%ADnky%20poskytov%C3%A1n%C3%AD%20slu%C5%BEeb%20hostov%C3%A1n%C3%AD%20softwarov%C3%A9%20aplikace%20dm%20Software%2020161114.pdf.

18. DM SOFTWARE S.R.O. *Ceník aplikace dm Software* [online]. 2015 [cit. 2018-04-11]. Dostupné z: https://portal.dmsoftware.cz/Portals/portaldm/smlouva/Cen%C3%ADk%2020151201.pdf.

19. GOOGLE COMMERCE LTD. *dm Software – aplikace na Google Play* [online]. 2018 [cit. 2018-04-11]. Dostupné z: https://play.google.com/store/apps/details?id=cz.dmsoftware.mobile.

20. MOZILLA; INDIVIDUAL CONTRIBUTORS. *What is JavaScript? - Learn web development | MDN* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

21. CODE SCHOOL LLC. *Server-side Languages | Code School* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.codeschool.com/beginners-guide-to-web-development/server-side-languages.

22. THE PHP GROUP. *History of PHP* [online]. 2018 [cit. 2018-04-12]. Dostupné z: http://php.net/manual/en/history.php.php.

23. THE PHP GROUP. *PHP: PHP 7 Changelog* [online]. 2018 [cit. 2018-04-12]. Dostupné z: http://php.net/ChangeLog-7.php#7.2.4.

24. TIOBE SOFTWARE BV. *TIOBE Index for April 2018* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.tiobe.com/tiobe-index/.

25. MILLARES, Gerard. *Top 5 Programming Languages Used In Web Development* [online]. 2015 [cit. 2018-04-12]. Dostupné z: http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/.

26. GAMMA, Smart. *What are the pros and cons of using PHP?* [online]. 2016 [cit. 2018-04-12]. Dostupné z: https://medium.com/@smartgamma/what-are-the-pros-and-cons-of-using-php-490553ed8ff2.

27. SHARPENED PRODUCTIONS. *TechTerms - Java definition* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://techterms.com/definition/java.

28. ČÁPKA, David. Úvod do jazyka Java. *IT network* [online]. 2018, č. 1 [cit. 2018-04-12]. Dostupné z: https://www.itnetwork.cz/java/zaklady/java-tutorial-uvod-do-jazyka-java.

29. LEAHY, Paul. What Is Java? *ThoughtCo* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.thoughtco.com/what-is-java-2034117.

30. CHEUNG, Allen. *Why is Java perceived as not cool for startups? We seem to be getting a lot of feedback lately that a startup should be using Ruby on Rails, PHP, Python, etc., if they want to be agile and iterate quickly.* [online]. 2014 [cit. 2018-04-12]. Dostupné z: https://www.quora.com/Why-is-Java-perceived-as-not-cool-for-startups-We-seem-to-be-getting-a-lot-of-feedback-lately-that-a-startup-should-be-using-Ruby-on-Rails-PHP-Python-etc-if-they-want-to-be-agile-and-iterate-quickly.

31. PYTHON SOFTWARE FOUNDATION. *What is Python? Executive Summary* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.python.org/doc/essays/blurb/.

32. WOOD, Sam. A Brief History of Python. *Packt Hub* [online]. 2015 [cit. 2018-04-12]. Dostupné z: https://hub.packtpub.com/brief-history-python/.

33. SAHOUANE, Amine. *The pros and cons of Python* [online]. 2016 [cit. 2018-04-12]. Dostupné z: https://www.supinfo.com/articles/single/3425-the-pros-and-cons-of-python.

34. NODE.JS FOUNDATION. *About Node.js®* [online] [cit. 2018-04-12]. Dostupné z: https://nodejs.org/en/about/.

35. TUTORIALSPOINT.COM. *Node.js - Introduction* [online]. 2018 [cit. 2018-04-12]. Dostupné z: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm.

36. ADAM, Marc F. *The History and Impact of Node.js* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://nixa.ca/en/blog/the-history-and-impact-of-nodejs/.

37. TIME INC. *Fortune 500 Companies 2017: Who Made the List* [online]. 2017 [cit. 2018-04-13]. Dostupné z: http://fortune.com/fortune500/list/.

38. SHAN, Paul. *Node.js – reasons to use, pros and cons, best practices!* [online]. 2014 [cit. 2018-04-13]. Dostupné z: http://voidcanvas.com/describing-node-js/.

39. ROUSE, Margaret; LEAKE, Allan; HUGHES, Adam. *What is database (DB)? - Definition from WhatIs.com* [online]. 2017 [cit. 2018-04-13]. Dostupné z: https://searchsqlserver.techtarget.com/definition/database.

40. XPLENY. *The SQL vs NoSQL Difference: MySQL vs MongoDB* [online]. 2017 [cit. 2018-04-13]. Dostupné z: https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2.

41. REFSNES DATA. *Introduction to SQL* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://www.w3schools.com/sql/sql_intro.asp.

42. HOMAN, Jacqueline. *Relational vs. non-relational databases: Which one is right for you?* [online]. 2014 [cit. 2018-04-13]. Dostupné z: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases.

43. *Developer Survey Results 2018* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://insights.stackoverflow.com/survey/2018#technology-databases.

44. ORACLE CORPORATION AND/OR ITS AFFILIATES. *What is MySQL?* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

45. ORACLE CORPORATION AND/OR ITS AFFILIATES. *MySQL Editions* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://www.mysql.com/products/.

46. TECHOPEDIA INC. *What is NoSQL? - Definition from Techopedia* [online]. 2018 [cit. 2018-04-13]. Dostupné z: https://www.techopedia.com/definition/27689/nosql-database.

47. GATTERMAYER, Josef. *NoSQL, Apache Cassandra, úvod a clusterování* [online]. 2014 [cit. 2018-04-13]. Dostupné z: https://owncloud.cesnet.cz/index.php/s/b8d8aa30c136b48d01c55c290e17b218.

48. ROUSE, Margaret; VAUGHAN, Jack. *What is MongoDB? - Definition from WhatIs.com* [online]. 2014 [cit. 2018-04-15]. Dostupné z: https://searchdatamanagement.techtarget.com/definition/MongoDB.

49. ERIKSSON, Ulf. *FUNCTIONAL VS NON FUNCTIONAL REQUIREMENTS* [online]. 2012 [cit. 2018-04-15]. Dostupné z: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/.

50. ČÁPKA, David. UML - Use Case Diagram. *IT network* [online]. 2018, č. 2 [cit. 2018-04-18]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram.

51. SENSIOLABS. *Why should I use a framework?* [online] [cit. 2018-04-21]. Dostupné z: https://symfony.com/why-use-a-framework.

52. MONUS, Anna. *10 PHP Frameworks For Developers – Best of* [online]. 2018 [cit. 2018-04-21]. Dostupné z: https://www.hongkiat.com/blog/best-php-frameworks/.

53. *Introduction - Database Abstraction Layer (DBAL) - Doctrine* [online] [cit. 2018-04-21]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-dbal/en/2.7/reference/introduction.html.

54. ČÁPKA, David. *MVC architektura* [online]. 2018 [cit. 2018-04-23]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

55. DEVERIA, Alexis. *CSS Flexible Box Layout Module* [online]. 2018 [cit. 2018-04-30]. Dostupné z: https://caniuse.com/#feat=flexbox.

56. KRUG, Steve. *Don't Make Me Think! A Common Sense Approach to Web Usability*. 2. vyd. 1249 Eighth Street, Berkeley, California USA: New Riders, 2006. ISBN 0-321-34475-8.

57. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online]. 2000 [cit. 2018-05-04]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

Dodatek A

Seznam použitých zkratek

CLI Command Line Interface.

CSS Cascading Style Sheets.

HTML Hypertext Markup Language.

HTTP HyperText Transfer Protokol.

IS Informační Systém.

JS JavaScript.

MVC Model–View–Controller.

MVP Model–View–Presenter.

PHP PHP: Hypertext Preprocessor.

SEO Search Engine Optimization.

SQL Structured Query Language.

XML Extensible Markup Language.

Dodatek B

Instalační příručka

Tato instalační příručka si klade za cíl provést uživatele procesem nasazení aplikace. Představeny budou požadavky na systém a prostředí, dále bude následovat nastavení a instalace databáze a webové aplikace. Na závěr bude představena konfigurace aplikace a řešení problémů.

Aplikace nevyžaduje použití HTTPS protokolu, pro produkční nasazení je ovšem jeho zapnutí důrazně doporučeno. Přesměrování na HTTPS lze nastavit v souboru `web/.htaccess`.

(A) Požadavky na systém a prostředí

- Pro instalaci závislostí je zapotřebí program `composer`. Stáhnout ho můžete z `www.getcomposer.org/`.
- Dále je vyžadován `yarn`, jedná se o program pro správu balíčků. Stáhnout ho můžete z `www.yarnpkg.com/en/docs/install`.
- Systém je ve výchozím nastavení připraven pro použití MySQL. Lze použít i jinou databázi, to ovšem vyžaduje úpravu hodnoty `doctrine.dbal.driver` v souboru `app/config/config.yml`.
- Ujistěte se, že je k dispozici PHP minimálně verze 7.1, stáhnout ho lze z `http://php.net/downloads.php`.
- Ujistěte se, že máte povolený zápis a čtení z adresáře `var` včetně všech podadresářů.

(B) Instalace databáze

a) Ujistěte se, že je k dispozici vhodná databáze.

b) Databázi lze vytvořit spuštěním příkazu `yarn db-create`.

c) Schéma databáze vytvoříte spuštěním příkazu `yarn schema-update`, následně můžete ověřit spuštěním `yarn schema-validate`.

d) Do prázdné databáze lze nahrát počáteční hodnoty spuštěním příkazu `yarn populate` či pomocí skriptu `insert.sql`.

(C) Aplikaci nainstalujete spuštěním příkazu `yarn build`.

(D) Konfigurace

a) Vytvořte kopii souboru `app/config/parameters_example.yml`, tu pojmenujte `app/config/parameters.yml`.

b) Ve zkopírovaném souboru se nacházejí konfigurační hodnoty, ty upravte dle potřeby. Význam jednotlivých proměnných obsažených v tomto souboru:

- env(DATABASE_HOST) – adresa databázového serveru
- env(DATABASE_PORT) – port databázového serveru
- env(DATABASE_NAME) – název databáze
- env(DATABASE_USER) – název uživatele pro přístup do databáze
- env(DATABASE_PASSWORD) – heslo daného uživatele pro přístup do databáze
- env(MAILER_TRANSPORT) – typ SMTP serveru
- env(MAILER_HOST) – adresa SMTP serveru
- env(MAILER_USER) – emailová adresa
- env(MAILER_PASSWORD) – heslo pro danou emailovou adresu
- env(SECRET) – bezpečnostní řetězec, důležité změnit

(E) Odstranění chyb

- V případě chyby nedostatečného oprávnění lze využít detailní postup zveřejněný na adrese `https://symfony.com/doc/3.4/setup/file_permissions.html`.

Pokud jste použili příkaz `yarn populate`, či jste využili skriptu `insert.sql`, databáze obsahuje připravený účet administrátora.

jméno main_admin
heslo change_me

Dodatek C

Obsah přiloženého CD

- /
  - src
    - diagrams ..... diagramy
    - implementation ..... zdrojové kódy prototypu
    - screenshots ..... snímky obrazovky prototypu
      - computer ..... snímky obrazovky se standardním rozlišením
      - smartphone ..... snímky obrazovky s malým rozlišením
  - thesis
    - src ..... zdrojová forma práce ve formátu LaTeX
    - thesis.pdf ..... text práce ve formátu PDF
  - readme.txt ..... stručný popis obsahu CD