FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro správu zaměstnanců
Student:	Jan Picka
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem je navrhnout a implementovat systém pro správu zaměstnanců a jejich agendy. V sytému bude možné spravovat nepřítomnosti zaměstnanců, firemní dokumenty a vytvářet ankety pro zaměstnance. Přístup k jednotlivým funkcionalitám se bude lišit, dle dané role uživatele. Aplikace bude napsána v PHP frameworku Laravel.

1. Proveďte analýzu existujících řešení.
2. Dle zadané specifikace navrhněte aplikaci.
3. Implementujte prototyp aplikace.
4. Aplikaci podrobte vhodným testům.
5. Připravte aplikaci pro nasazení, alespoň do testovacího prostředí.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 14. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém pro správu zaměstnanců

Jan Picka

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Glazar

4. června 2020

Poděkování

Chtěl bych poděkovat Ing. Filipu Glazarovi za vedení práce a všechny rady, firmě LYFLE s. r. o., jmenovitě Jaroslavu Zetelovi a Ing. Janu Bradáčovi za možnost psát tuto bakalářskou práci a rodině za trpělivost a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ustanovení § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisu.

V Praze dne 4. června 2020 . . . . . .

Odkaz na tuto práci

Picka, Jan. *Systém pro správu zaměstnanců*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce popisuje analýzu, návrh a implementaci prototypu systému pro správu zaměstnanců. Tato webová aplikace, napsaná v programovacím jazyce PHP, si klade za cíl zlepšit a zpřehlednit komunikaci mezi zaměstnancem a zaměstnavatelem a zároveň poskytnout i možnosti domluvy mezi jednotlivými zaměstnanci.

Klíčová slova systém pro správu zaměstnanců, správa zaměstnanců online, dotazníky v práci, PHP, Laravel, webová aplikace

Abstract

This thesis describes analysis, design and implementation of prototype of employee management system. Aim of this web application written in programming language PHP is to improve communication between employees and employer as well as provide messaging feature between employees.

Keywords employee management system, employee management online, questionnaires in company, PHP, Laravel, web application

Obsah

Seznam obrázků

Seznam výpisů kódu

Úvod

V dnešní době se každá firma musí potýkat s problémy interakce mezi svými zaměstnanci. To hlavně platí u menších firem, které nemají dost financí, aby mohli zakoupit drahý a komplikovaný software. Takové firmy potřebují jednoduché řešení, webovou aplikaci, která se dá ovládat i z mobilu. Aplikaci, ve které mohou spravovat omluvenky a dovolené zaměstnanců, vypisovat a vyhodnocovat dotazníky, ukládat dokumenty atd.

Systém, který v rámci své bakalářské práce vyvíjím, je svým zaměřením určen malým a středním podnikům. Toto téma jsem si zvolil, protože neexistuje žádná jednoduchá alternativa takové aplikace. Většina aplikací se zaměřuje jen na pracovněprávní vztah, zatímco aplikace, vyvíjená v rámci bakalářské práce, je spíše o komunikaci se zaměstnanci. Klade si za cíl hlavně zpřehlednit komunikaci mezi zaměstnavatelem a zaměstnancem. Malé podniky ještě dnes řeší komunikaci se zaměstnanci prostřednictví emailů a SMS. Další důležitou komunikační metodou jsou dotazníky, díky nimž zaměstnavatel lehce zjistí, co si o daném problému myslí jeho zaměstnanci.

Práce je zaměřena na návrh, základní implementaci systému a napojení na systémy firmy Lyfle. Interní testování, které bude navazovat na tuto práci, proběhne ve firmě Lyfle. V první části práce se zaměřím na analýzu a návrh systému. Popíšu databázové schéma a existující systém Lyfle, na který bude modul Zaměstnání napojen. V praktické části se zaměřím na samotnou implementaci systému.

Po dokončení práce budeme v Lyfle pokračovat v testování systému. Do budoucna plánujeme vzít společné komponenty systému Zaměstnání a dalších systémů, které Lyfle vyvíjí a vytvořit z nich knihovní moduly, které by se přes rozhraní volaly z jednotlivých komponent. Tím bychom předešli zbytečnému kopírování kódu.

Rejstřík důležitých pojmů

Hned ze začátku bych chtěl pro čtenáře ujasnit některé pojmy, které se v rámci práce budou opakovat a mohli by mít na první pohled více významů.

Systém, projekt, aplikace, modul, modul Zaměstnání

Jedná se o systém, jehož prototyp je cílem této práce, pokud není v kontextu uvedeno jinak.

Školka, projekt Školka

Projekt Školka vznikl v rámci diplomové práce Ing. Jana Bradáče na Fakultě informačních technologií ČVUT. Položil základy pro celý modulární systém ve firmě Lyfle, do kterého v rámci této práce integruji prototyp modulu Zaměstnání.

Uživatel, uživatel systému

Uživatelem se myslím člověk, který se přihlašuje do modulu Zaměstnání pod jakoukoliv rolí. Může to tedy být zaměstnanec i administrátor.

Zaměstnanec

Uživatel systému, který má v rámci aplikace roli Zaměstnanec. Dále tento uživatel může nabývat i rolí v oddělení.

Administrátor

Uživatel s rozšířenými právy. Může to být pracovník firmy, pro kterou je konkrétní instalace systému určena, pak se jedná o lokálního administrátora, nebo to může být zaměstnanec firmy Lyfle, pak se jedná o globálního administrátora.

Entita

Objekt v systému, který má nějaké charakteristiky. V rámci tohoto systému se může jednat například o Uživatele nebo Oddělení.

Laravel fasády

Třídy v Laravelu, které poskytují statické metody, sloužící jako abstrakce metod původních tříd. Tedy opravdu slouží jako „fasády“ pro ostatní třídy [1].

Kapitola 1

Cíl práce

Cílem této bakalářské práce je navrhnout a implementovat prototyp systému pro správu zaměstnanců. Tento modul bude součástí systému firmy Lyfle a proto bude jedním z cílů tento projekt do systému zařadit. Rozsah implementace tohoto projektu je příliš velký na tuto práci a proto cílem této práce by měl být prototyp, který bude možné začít interně testovat.

První cílem této práce je analyzovat existující řešení na trhu. Jak si čtenář může přečíst v kapitole 2, nejedná se v tomto případě jen o existující aplikace. Druhým cílem je návrh samotné aplikace. Bude to webová aplikace, která bude integrovaná do systému Lyfle napsaná v programovacím jazyce PHP, za pomoci frameworku Laravel. Nejprve se budu věnovat databázovému návrhu a poté návrhu uživatelského rozhraní, který bude vycházet z již existujícího uživatelského rozhraní produktů firmy Lyfle.

V další části se budu věnovat samotné implementaci prototypu systému. V neposlední řadě bude potřeba podrobit systém vhodným testům, aby se zajistila potřebná bezpečnost a správné fungování za všech podmínek.

Kapitola 2

Analýza

Tento systém integruji do již existujícího systému ve firmě, který sám vzešel z diplomové práce *Mojeskolka.info – tvorba uzavřené komunitní sítě pro školky* Ing. Jana Bradáče [2] napsané na Fakultě informačních technologií ČVUT. V následujícím textu se na tuto práci budu odkazovat, hlavně co se týká analýzy a návrhu systému Školka.

2.1 Rozbor požadavků

Rozbor požadavků je důležitým krokem ke správné implementaci aplikace. Je dobré si sepsat požadavky zadavatele a pravidelně se k nim vracet, aby se docílilo co nejpřesnějšího výsledku projektu. Požadavky se dělí na dva typy – funkční a nefunkční.

Mezi funkční požadavky specifikujeme, co chceme, aby systém uměl, neboli jeho funkčnost. Důležitá otázka je u těchto požadavků „Co?“ a ne „Jak?“. Do nefunkčních požadavků patří další požadavky na systém, ale také návod, jak jich dosáhnout. Jejich cílem je zmenšit počet možných řešení problémů, tedy nějak konkretizovat zadání [3].

2.1.1 Funkční požadavky

- **Správa zaměstnanců**: Systém bude umožňovat ukládat, upravovat a mazat zaměstnance ve firmě.
- **Správa oddělení**: V systému bude možné dělit zaměstnance do oddělení. V rámci těchto oddělení pak budou mít zaměstnanci další práva podle rolí jim určených.
- **Omluvenky**: Zaměstnanci budou mít možnost v rámci systému podat omluvenku jinému zaměstnanci, který omluvenky spravuje. Omluvenka bude tří typů – Dovolená, Sick Day a Omluvenka (vyřízena domluvou, nemá omezený počet).

- **Dotazníky**: V systému bude možné vypsat dotazník pro zaměstnance.

- **Správa dokumentů**: Zaměstnavatel bude mít možnost nahrát dokument pro své zaměstnance. Ti si ho budou moci stáhnout.

2.1.2 Nefunkční požadavky

- **Webová aplikace**: Systém bude webová aplikace. Po nasazení bude systém přístupný z Internetu.

- **Responzivní design**: Systém bude navržen responzivně, aby se dal používat i na tabletech a mobilních zařízeních.

2.2 Volba platformy, programovacího jazyka a frameworku

Jak už jsem v této práci několikrát zmínil, tento systém integruji do systému firmy Lyfle. Tudíž všechny volby, co bych chtěl dělat, budou ovlivněny tímto již existujícím systémem. Proto jsem se rozhodl nejdříve ve zkratce popsat postup a řešení ke kterému dospěl Jan Bradáč ve své diplomové práci a poté k tomu přidat svůj komentář.

2.2.1 Volba platformy

V kapitole 1.5 jeho diplomové práce [2] Jan Bradáč popisuje rozhodování o volbě platformy pro systém Školka. Ve snaze nezaujatě zhodnotit jednotlivé platformy, zvolil Bradáč kritéria jimž přiřadil váhu důležitosti. Mezi nejdůležitější počítal rozšířenost platformy, tedy kolik uživatelů tuto platformu používá a tedy kolik uživatelů může systém oslovit. Dalším důležitým faktorem je dostupnost a pohodlí platformy, tedy jednak použitelnost platformy na mobilním zařízení, ale také na velké obrazovce u počítače.

Po zhodnocení těchto kriterií se rozhodl pro tenkého webového klienta, tedy systém, který budou uživatelé používat v rámci webového prohlížeče. Později se v rámci vývoje systému k webovému klientovi přidaly i mobilní aplikace pro Android a iOS.

Aplikace Zaměstnání bude také webová aplikace hned z několika důvodů. Za prvé to zjednodušuje napojení na existující systém, pokud jsou ty platformy stejné. Za druhé, stejně jako vyšlo Janu Bradáčovi v jeho diplomové práci, webové aplikace jsou stále mnohem více používané než kterákoliv alternativa hlavně kvůli tomu, že webový prohlížeč je na každém uživatelském zařízení, které má přístup k Internetu.

2.2.2 Volba programovacího jazyka

V kapitole 1.6 diplomové práce [2] se Bradáč věnuje zvažování kriterií pro výběr programovacího jazyka. V návaznosti na kapitolu 1.5 se mu tím omezil výběr jazyků na Javu, platformu .Net, PHP a Ruby. Po zvážení kritérií, kde mezi nejvýznamnější patřila znalost jazyka vývojářem aplikace, došel k závěru, že aplikace bude napsaná v jazyce PHP[1].

Podle průzkumu stránky W3Techs je PHP stále nejpoužívanějším programovacím jazykem pro serverovou část aplikací, používá ho 78,7% nejnavštěvovanějších serverů [4].

2.2.3 Volba frameworku

Další volba, která před J. Bradáčem stála, byla volba vhodného frameworku pro PHP. Rozhodl se využít již existující framework oproti skládání vlastního [2, s. 93, kap. 3.7.1]. Po zvážení kritérií u něj zvítězil framework Laravel[2], který byl vybrán hlavně pro velikost komunity a silné zaměření na usnadnění vývoje.

Laravel, vyvíjený od roku 2011 Taylorem Otwellem, je v dnešní době pořád jeden z nejoblíbenějších frameworků pro PHP (například podle [5], [6]). Zároveň je to jediný framework pro PHP, který ovládám natolik, abych byl ochotný v něm psát nějaký větší systém, proto pro mě toto rozhodnutí nebylo těžké.

2.3 Rozbor zadání

1. **Proveďte analýzu existujících řešení.**

   Tato analýza si klade za cíl poznat další softwarová řešení systému pro správu zaměstnanců, ale také je potřeba projít výhody a nevýhody jiných řešení. Díky tomu si ustanovíme požadavky, které tento systém má splňovat, což nám pomůže pro následující návrh a implementaci.

2. **Dle zadané specifikace navrhněte aplikaci.**

   Návrh částečně vychází z existujících aplikací firmy Lyfle. Proto začneme nejdříve obecným přehledem důležitých částí návrhu systému Školka firmy Lyfle a dále se budu věnovat rozílům a přidaným částem systému Zaměstnání, o kterém pojednává tato práce.

3. **Implementujte prototyp aplikace.**

   Podle předchozích dvou kapitol začneme s implementací prototypu aplikace. V této kapitole se hlavně zaměříme na úpravu základu projektu

[1] https://www.php.net/
[2] https://laravel.com/

Školka, abychom k němu mohli postupně přidávat nové funkce a udělat z něj nakonec systém Zaměstnání.

4. **Aplikaci podrobte vhodným testům.**

   Kvůli rychlému vývoji aplikací ve firmě Lyfle bohužel nebyly dobře navrženy na použití automatických testů. Proto se v této kapitole zaměřím na uživatelské testování systému.

5. **Připravte aplikaci pro nasazení, alespoň do testovacího prostředí.**

   Tomuto tématu se krátce věnuji v kapitole 4.5.

2.4 Rešerše

2.4.1 Webová aplikace

Webová aplikace je softwarový program (software), který běží na webovém serveru. Na rozdíl od standardních aplikací (offline/desktop aplikací), které se spouští v lokálním operačním systému, webová aplikace musí být přístupná z webového prohlížeče [7].

Výhody a nevýhody webových aplikací

Výhody i nevýhody webových aplikací lze dále dělit na výhody a nevýhody pro vývojáře a pro uživatele.

Největší výhoda pro programátory je, že není potřeba vyvíjet systém v několika různých programovacích jazycích pro různé operační systémy. Webová aplikace běží na webovém serveru a uživatel k ní přistupuje přes webový prohlížeč. Díky tomu že jsou přístupné z webového prohlížeče, nemusí vývojáři psát aplikace pro jednotlivé operační systémy. Aktualizace obsahu aplikace je také lehčí než v rámci offline/desktop aplikací. Aktualizovat stačí data na webovém serveru a uživatelé při dalším přístupu získají nejaktuálnější data.

Mezi nevýhody pro vývojáře patří větší počáteční náročnost návrhu uživatelského rozhraní. Jelikož má být aplikace přístupná jak z velké počítačové obrazovky, tak z malé obrazovky telefonu, je potřeba řešit rozmístění a velikost grafických prvků [8].

Co se týká uživatelů, těm určitě pomůže v orientaci podobnost uživatelského rozhraní aplikace na různých zařízeních, jelikož webová aplikace není závislá na operačním systému uživatele, ale spíš na prohlížeči. Dále data, která uživatel zadá do webové aplikace jsou uložena na vzdáleném serveru, tudíž k nim má přístup z různých zařízení [7].

Jelikož webové aplikace neběží na lokálním operačním systému, pro náročnější aplikace, jako zpracování videa a obrázků jsou ve většině případů pomalé. Za prvé je zde limitující rychlost uživatelova připojení a za druhé také může

být limitující výkon webového serveru. Pokud na webový server přistupuje zároveň více lidí, může být práce ve webové aplikaci pro uživatele pomalá [7].

Také je zde nutnost připojení k Internetu. Pokud je webová aplikace rozsáhlá a obsahuje například i videa, tak je zde nutnost rychlého a spolehlivého připojení k Internetu [8]. S rychlým vývojem webových technologií se také rychle vyvíjejí webové prohlížeče a zastaralé verze prohlížečů mohou způsobit nekompatibilitu webové aplikace [7].

2.4.2 Správa zaměstnanců

Při provádění literární rešerše mě zaujal vědecký článek o implementaci systému pro sledování a plánování výkonu zaměstnanců v neziskové organizaci [9]. Netýká se sice stejného typu systému jako implementuji já v rámci své bakalářské práce, ale přesto jsem si z toho článku odnesl pár poznatků.

Pokud by tento systém měl v budoucnu obsahovat nějaké metody pro sledování výkonu zaměstnanců, je nutné si uvědomit, že někteří to považují až za škodlivé pro samotný výkon zaměstnanců [9, s. 256].

Z článku dále vyplývá, že je důležité si uvědomit rozdíl mezi ziskovými a neziskový organizacemi na úrovni zaměstnanců. V neziskových organizacích zaměstnanci často vymění vyšší peněžní ohodnocení a bonusy, které by získali v jiných společnostech za pocit, že část jejich práce jde na věc, se kterou morálně souhlasí. Proto se většina neziskových firem vyhýbá nasazení systému pro sledování a zlepšování výkonu svých zaměstnanců [9, s. 257].

V dalším článku nazvaném Úskalí implementace systému pro správu zaměstnanců (...) [10] je možné najít několik výhod takového systému. Jako první výhodu autoři článku uvádí zabezpečení dat v systému. Dokumenty o zaměstnancích jsou podle nich ve větším bezpečí v dobře navrženém systému než v zásuvce u stolu v nějaké kanceláři.

Jako další výhodu uvádějí aktuálnost a dostupnost dat. Data v systému jsou upravována zaměstnanci, kteří se o data tohoto typu starají i ve firmě, tím pádem jsou data vždy aktuální. Pokud společnost používá systém pro správu zaměstnanců, který je webovou aplikací, pak je další velkou výhodou dostupnost dat. Každý zaměstnanec má zpravidla vytvořený účet a může kontrolovat a získávat jednoduše svá data přes systém odkudkoliv [10, s. 1201].

2.4.3 Webový dotazník

Dále jsem se při rešerši zaměřil na webové dotazníky. S rozšířením přístupu k Internetu pomocí počítačů a mobilních zařízení se i zadávání dotazníků stává čím dál populárnější na Internetu [11].

I když dotazníky na webech a posílané přes email mají zpravidla menší vyplněnost oproti dotazníkům zasílaným poštou, tak jejich výhody pro většinu lidí převažují. Například možnost kontrolovat správnost vyplnění dotazníku už při jeho vyplňování. Autoři dotazníku se tak nemusí rozhodovat, zdali čás-

tečně vyplněný dotazník započítat do statistik nebo ne. Dále není potřeba dotazníky vyplněné na webu přepisovat ručně do statistického softwaru, většina webových aplikací s dotazníky statistiky tvoří přímo z odpovědí. Tím se eliminuje možnost chyby při přepisování odpovědí.

Asi největší výhodou webových dotazníků je relativně nízká cena. Po vytvoření dotazníku se může rozeslat (emailem nebo odkazem) lidem, což už nestojí nic navíc. Ovšem webové dotazníky jsou ze začátku pracnější na vytvoření než papírové, hlavně kvůli nastavení automatické kontroly odpovědí [12, s. 18]. Velkou nevýhodou webových dotazníků je, že z něj jsou úplně vyloučení lidé, kteří nemají přístup k počítači, potažmo Internetu, což se v dnešní době týká hlavně seniorů.

Někteří lidé mohou považovat webové dotazníky za neosobní, například oproti interview. To se dá částečně zlepšit pomocí přidání videa nebo fotografií k jednotlivým otázkám dotazníku [12, s. 17].

2.4.4 Existující řešení

Papírové řešení

Základní možnost jak řešit správu zaměstnanců je samozřejmě vést si důležité údaje na papír. Toto řešení nejméně nákladné ze všech a pokud je firma malá, tak je i zvládnutelné. Avšak přináší jistá úskalí. Zabezpečení takových informací je hodně nevyhovující. Nemusí se jednat ani o velmi katastrofické scénáře, stačí polít důležitý dokument čajem. Co se týká řešení omlouvání zaměstnanců a řešení dovolených, tak v tomto případě bude nejspíš řešeno pomocí emailů a SMS. Řešení tohoto problému je také nevyhovující, jelikož se špatně udržuje historie dovolených a také aktuální čerpání. Pokud by se firma měla začít rozrůstat na více než jen pár zaměstnanců, bude hledání takových informací prakticky nemožné.

Výhody:

+ bezplatné řešení
+ není potřeba žádná znalost počítačových systémů

Nevýhody:

– vhodné pouze pro velmi malý počet zaměstnanců
– nemožnost přehledného zobrazení historie a aktuálního stavu

Online nástroje – Microsoft Excel, Google Sheets, atd.

Většina online služeb má data zálohovaná, proto je toto řešení lepší aspoň co se týká bezpečnosti. Ovšem přidává další nepřehlednost a tou je spousta nových systémů a online dokumentů, ve kterých se musí zaměstnanec vyznat.

Navíc toto řešení moc nepomáhá snadné dohledatelnosti dat do minulosti. Omlouvání a správa dovolených je v tomto řešení také dost nevyhovující. Zaměstnanec se se svým nadřízeným stále musí domlouvat pomocí jiných komunikačních prostředků (email, SMS) a nemá možnost například jednoduše zjistit, kolik dovolené mu ještě zbývá.

Výhody:

+ bezplatné[3] řešení
+ potřeba jen základních znalostí práce s kancelářskými nástroji

Nevýhody:

– vhodné pouze pro malý počet zaměstnanců
– stále nemožnost přehledného zobrazení historie a aktuálního stavu

Specializované aplikace

Specializovaná aplikace, je podle mě, nejlepší řešení pro tento problém. Většina firem vyvíjejících software pro správu zaměstnanců, míří na větší firmy nebo státní správu. U nich je potřeba nejen software který zvládá správu zaměstnanců, ale i software s další funkcionalitou, například sledování pracovního výkonu, precizní sledování příchodů a odchodů zaměstnanců atd. Díky těmto částem navíc, které pro malé firmy nejsou tak potřebné to přidává nechtěnou složitost systému a navíc i zvyšuje cenu.

Výhody:

+ přehlednost (díky specializaci pro tento účel)
+ doplňující funkce

Nevýhody:

– dražší (záleží na vybraném softwaru)
– potřeba se naučit práci s novým systémem

[3]Balíček programů firmy Microsoft něco stojí, ovšem nepředpokládám, že by firma kupovala tyto programy jen pro vedení docházky svých zaměstnanců.

Kapitola 3

Návrh

3.1 Modulární systém

Systém firmy Lyfle byl navržen modulárně, aby bylo jednodušší přidávat další části (moduly) bez větších, nebo žádných změn v ostatních modulech. Druhá strana modularity systému je větší složitost při návrhu a implementaci, jelikož se musí počítat s co největší separací logiky od ostatních modulů, aby na sobě byly nezávislé. Toho se dosahuje hlavně pomocí rozhraní. Systém Lyfle je z pohledu modularity rozdělen na dvě části, aplikační a databázovou, které spolu souvisí, ale nejsou úplně identické.

3.1.1 Aplikační moduly

Aplikační moduly jsou logické celky v aplikaci, které jsou oddělené od zbytku aplikace právě pomocí výše zmíněných rozhraní. Tím se docílí dobré separace logiky a díky tomu se systém lépe spravuje.

Uživatelé

Uživatelský modul udržuje logiku uživatelů a jejich kontaktních informací. Jak se čtenář dozví v následující podkapitole, zároveň má tento aplikační modul i vlastní databázi.

Zprávy

Zprávy jsou oddělené od uživatelů hlavně kvůli rozložení zátěže na systém, počítá se s tím, že pokud se systém rozšíří mezi hodně firem, zprávy budou hodně vytěžovaná část systému, a proto jsou také v jiné databázi než uživatelé [2, s. 88].

Firmy a role

Dalším logickým celkem aplikace jsou firmy a role. Tento modul udržuje logiku práce s jednotlivými společnostmi a jejich rolemi. Rozhodnutí, že role jsou ve stejném modulu jako firmy, a ne jako uživatelé, je znovu opodstatněné vývojem do budoucna. Počítá se, že jednotlivé firmy si budou moci nadefinovat systémové role podle svých představ, a potom dává smysl je držet dohromady. Zároveň se v tomto modulu také nachází logika oddělení a rolí v odděleních, a to ze stejného důvodu jako u systémových rolí.

Oznámení

Další logický celek tvoří Oznámení (Notifikace) a jejich metadata. Do tohoto modulu zároveň patří i interakce mezi Oznámením a Uživatelem.

Dotazníky

Veškerá logika okolo dotazníků má také svůj aplikační modul. K Dotazníkům se ještě dostaneme při návrhu nových entit v podkapitole 3.3.2.

3.1.2 Databázové moduly

Databázové moduly jsou větší celky v aplikaci, které spolu komunikují jen minimálně, a proto je možné je „fyzicky“ oddělit. Slovo „fyzicky“ je nutné brát s rezervou, jelikož se jedná o rozdělení databází. Prozatím, kdy systém není moc zatížen, jsou tyto databáze na stejném serveru. Do budoucna je ale systém díky těmto modulům připraven na rozdělení na více databázových serverů.

Centrální databáze

V centrální databázi se uchovávají odkazy na jednotlivé instance modulů a zprávy uživatelů. Tato databáze se označuje jako `lfl_clm`.

Uživatelská databáze

Tato databáze drží data o všech uživatelích všech instancí. Jedná se tedy o uživatele jak modulu Školka, tak modulu Zaměstnání. Nese označení `lfl_sud` (akronym ze System user database).

Společná databáze modulu

Každý modul (Zaměstnání, Školka) má společnou databázi pro uchování dat o firmách (kontaktní informace a korespondenční adresy). Tato databáze je

označovaná jako `lfl_<slug_modulu>_shared`, kde za *<slug_modulu>* se dosadí unikátní zkrácená verze názvu modulu [13], pro modul Zaměstnání je to *emp*, pro Školku *kinder*.

Databáze instancí modulů

Poslední databáze jsou databáze jednotlivých instancí modulů. Ty se postupně číslují a dodržují systém pojmenování `lfl_<slug_modulu>_<xxx>`, kde za *<xxx>* se dosadí trojmístné číslo, začínající na 001 (například `lfl_emp_001`).

3.2 Návrh uživatelského rozhraní

V rámci bakalářské práce jsem tvořil návrh uživatelského rozhraní pro Dotazníky a Dokumenty. Vycházel jsem z návrhu zbytku systému a výsledky si můžete prohlédnout v příloze D.

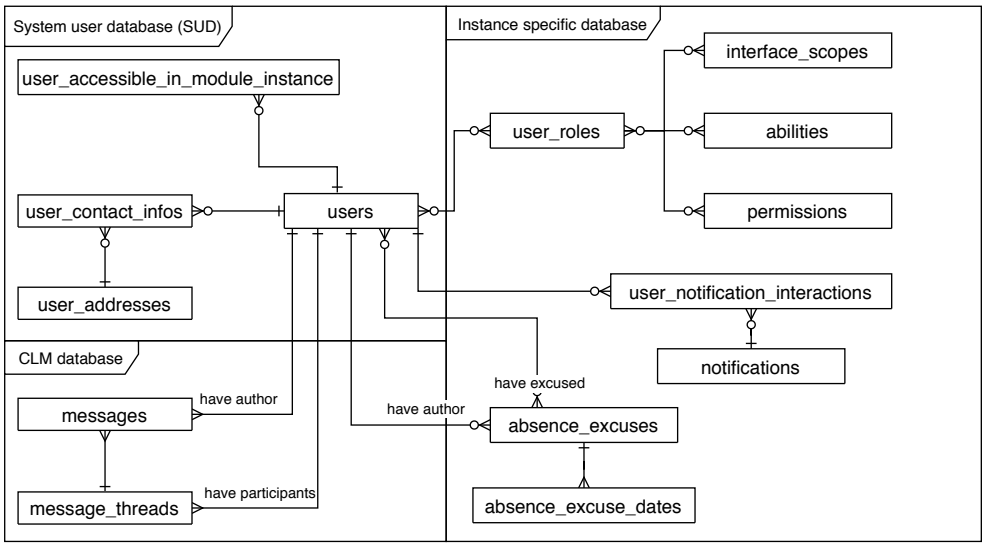
3.3 Návrh entit

3.3.1 Důležité entity pro základ systému

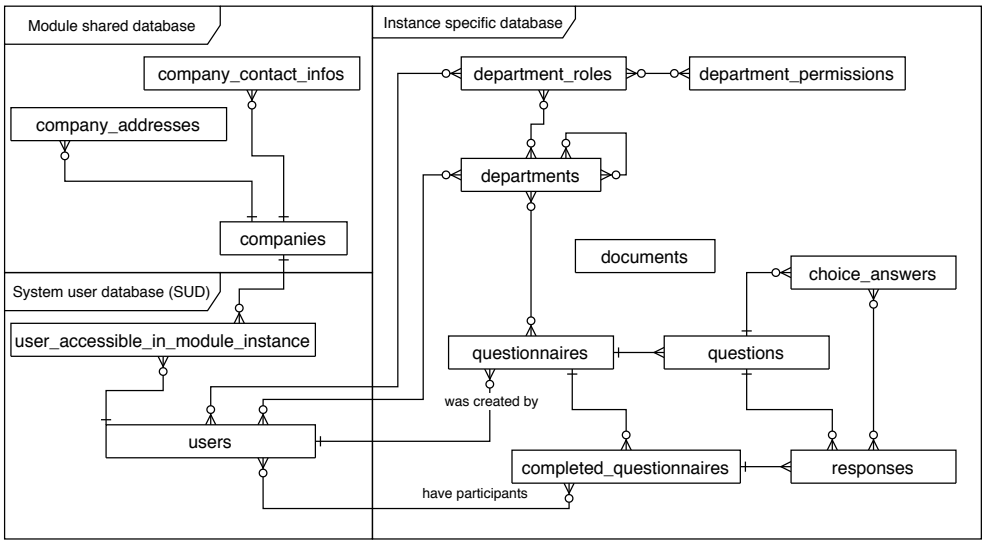
Na diagramu 3.1 jsou vidět entity, díky kterým funguje jádro systému. Rád bych zde zmínil pivot tabulku *user_accessible_in_module_instances*, která udržuje vztahy mezi uživateli a instancemi modulů. Díky ní je možné, aby byl uživatel registrován ve více instancích modulů na jeden uživatelský účet. Systém se po přihlášení zeptá, do jaké instance se chce přihlásit.

3.3.2 Entity související s dotazníky a dokumenty

Na dalším diagramu můžete vidět entity, které souvisí s dotazníky a dokumenty. Diagram dokumentů je velmi jednoduchý, zde nám stačí databázová tabulka, která udržuje jednotlivé záznamy o nahraných dokumentech, konkrétně cestu k nahranému souboru a popis souboru. Pro Dotazníky je entit více. Při implementaci prototypu jsem nevyužil entitu *choice_answers*, která bude sloužit pro uchování odpovědí typu single-choice a multi-choice. V rámci prototypu byly implementovány pouze textové odpovědi.



Obrázek 3.1: Databázový diagram znázorňující důležité systémové entity.



Obrázek 3.2: Databázový diagram znázorňující entity dotazníků a dokumentů.

Kapitola 4

Realizace

V rámci této kapitoly práce vznikl prototyp modulu Zaměstnání. Nejdříve bych rád čtenáře seznámil se základním fungování PHP frameworku Laravelu. Dále popíšu implementaci nových částí, konkrétně Dokumentů a Dotazníků.

4.1 Nástroje

Pro vývoj modulu bylo použito IDE PhpStorm od firmy JetBrains [14]. Už jsem v něm pracoval dříve, ve škole i v práci, a byl jsem s ním velmi spokojen. Dále jsem použil nástroj Wamp [15] pro Windows, který poskytuje prostředí podobající se reálnému serveru. Díky tomuto nástroji, který je ve skutečnosti balíčkem několika známých nástrojů, jako například webový server Apache, nebo databázový server MySQL, je lokální vývoj webových aplikací v PHP jednodušší. Vývojář má možnost zkoušet kód který napsal okamžitě na lokálních testovacích datech.

Pro verzování kódu byl použit systém GIT [16]. Tento systém je v dnešní době hojně používaný vývojáři nejen webových aplikací. Všechny použité nástroje jsem měl osvědčené už z minulosti, proto jsem nepředpokládal, že s nimi nastanou nějaké problémy a také nenastaly.

4.2 Použité technologie

4.2.1 Laravel

Jak vyšlo z analýzy systému, byl pro tento systém použit PHP framework Laravel. Až po chvíli používání frameworku Symfony[4], ze kterého Laravel také obsahuje několik balíčků, jsem zjistil, jak moc se Laravel snaží programátorovi usnadnit vývoj a tvořit čistý, znovupoužitelný kód. Pro správné pochopení

[4] https://symfony.com/

dalších kapitol implementace je nutné čtenáře alespoň okrajově zasvětit, jak tento framework funguje.

Většina PHP frameworků dodržuje návrhový vzor Model View Controller (MVC) a Laravel není výjimkou. Díky MVC se v aplikaci separují tři důležité a velmi rozdílné části a díky tomu vzniká dobře udržovatelná a rozšiřitelná aplikace. Zde jsou popsané části návrhového vzoru MVC:

- **Model**

  O modelovou vrstvu se v Laravelu stará balíček Eloquent ORM[5]. Je to implementace ActiveRecord, tedy systém, kde Model zajišťuje získávání dat z databáze a zároveň základní business logiku [17]. K tomu se vrátím později při popisování návrhového vzoru Repository.

- **View**

  Tato vrstva návrhového vzoru MVC se stará o grafickou část aplikace. Laravel přináší silný šablonovací systém Blade, který umožňuje nejen definovat stránky aplikace, ale zároveň také jednotlivé komponenty a šablony. Více o Blade v další části.

- **Controller**

  Controllery v Laravelu slouží ke zpracování požadavků od uživatele a následnému předání vykreslených View.

```
<!-- Stored in resources/views/layouts/app.blade.php -->

<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Výpis kódu 4.1: Blade master template

[5] https://laravel.com/docs/7.x/eloquent

Šablonovací systém Blade

Blade vznikl jako součást frameworku Laravel a od dalších šablonovacích systémů pro PHP se liší tím, že umožňuje používat základní PHP syntaxi přímo ve View. Soubory pro Blade systém mají koncovku `.blade.php` a při spuštění aplikace se zkompilují do čistého PHP kódu, díky čemu zbytečně nezatěžují systém při běhu.

Dvě hlavní výhody šablonovacího systému Blade jsou dědičnost a rozdělení na sekce. Základem pro Blade je master template (hlavní šablona). Ta definuje společné části všech výsledných HTML stránek aplikace (definuje CSS styly aplikace, meta tagy atd.). Do této šablony se pak přidávají jednotlivé sekce stránek potomků pomocí direktivy `@yield(<section-name>)`.

Tyto sekce, které se definují v Blade souborech potomků, označené direktivou `@section(<section-name>)`, utváří základní části stránek aplikace a používají se například pro vykreslení hlavního obsahu, hlavičky, ale také se mohou použít pro přidání dalších stylů a JavaScriptu. Aby Blade věděl, do jaké šablony doplnit definované sekce při zobrazování Views, je nutné rozšířit tuto šablony pomocí direktivy `@extend(<template>)`. Ve výpisu kódu 4.1 můžete vidět jednoduchý příklad master template a poté ve výpisu 4.2 Blade soubor, který master template rozšiřuje [18].

```
<!-- Stored in resources/views/child.blade.php -->

@extends('layouts.app')

@section('title', 'Page Title')

@section('sidebar')
    @parent

    <p>This is appended to the master sidebar.</p>
@endsection

@section('content')
    <p>This is my body content.</p>
@endsection
```

Výpis kódu 4.2: Blade potomek master template

Jazykové mutace

Další výborná funkce Laravelu je, že podporuje jazykové mutace bez nutnosti přidávání dalších balíčků. Definuje fasádu Lang, přes kterou se volají metody třídy Translator. Tato třída poté umožňuje získávat překladové texty ze sou-

```
/resources
/lang
    /en
        messages.php
    /es
        messages.php
```

Výpis kódu 4.3: Struktura lang souborů

borů (lang souborů), které je definují pro jednotlivé jazyky. Jak má vypadat struktura těchto souborů v Laravelu můžete vidět na výpisu kódu 4.3.

Pak stačí v aplikaci použít fasádu `App::setLocale('en')` a všechny překladové texty se berou ze souborů pro daný jazyk. V Lyfle jsme tento model dále rozšířili tak, že každý model má svůj lang soubor a přes fasádu *LflLang* se poté, pomocí názvu třídy tohoto modelu, přímo dostaneme k překladovým textům modelu.

4.2.2 Problémy Laravelu

Bohužel žádný programovací jazyk ani framework není dokonalý. V této kapitole jsem sepsal pár vylepšení frameworku Laravel, které mi pomáhají při vývoji.

View presentery

Jedním z takových vylepšení, který jsem ve své práci využil jsou View presentery. Data z modelů do view v Laravelu přichází jako strukturované pole (pole párů klíč-hodnota). Tento typ přenosu dat je určitě lepší, než kdyby se data přenášela v nestrukturovaném poli (hodnoty v poli jen za sebou), avšak i tento systém přenosu dat zabraňuje jednoduché znovupoužitelnosti kódu. Dobrým příkladem jsou výpisy Zaměstnanců a Oznámení do tabulky v aplikaci. Každá z těchto entit má jiné atributy a proto by se pro každou tuto entitu zvlášť muselo dělat view s tabulkou, kde jediný rozdíl by byl právě v přístupu do pole na různé atributy.

K vyřešení tohoto problému napomáhají View presentery. Jsou to jednoduché obalovací třídy, které definují stejné metody pro všechny modely. Tím zajistí to, že stačí pouze v controlleru vytvořit novou instanci View presenteru pro daný model, poslat mu strukturovaná data z modelu a on už se postará o naformátování dat pro view. Navíc tím zajišťuje, že view zobrazující tabulku entit může být stejné pro všechny.

```php
<?php

namespace App\Http\Controllers;

use ...;

class UserController extends Controller
{
    protected $userRepository;

    public function __construct(UserRepository $userRepository)
    {
        $this->userRepository = $userRepository;
    }

    public function index()
    {
        $users = $this->userRepository->all();
    }
}
```

Výpis kódu 4.4: Ukázka Laravel Dependency injection v Controlleru

Repository

Návrhový vzor Repository slouží jako abstrakce datové vsrtvy v aplikaci. Repository přes interface poskytuje sadu metod, které slouží ke zpracování dat z úložiště a navrácení ve formě objektů – Modelů.

Hlavním přínosem vzoru Repository v Larvelu je separace logiky Modelu od Controlleru. Kvůli tomu, že Model v Laravelu je implementací vzoru ActiveRecord, tak by logika zpracování dat z úložiště měla náležet právě buď metodám v Modelu, nebo v Controlleru. To ovšem do obou těchto částí přidává nechtěnou složitost a v případě Controlleru zbytečnou provázanost s Modely. Při používání databáze jako úložiště dat je při složitějších dotazech potřeba více tabulek (Modelů). Jelikož jsou Controllery v Laravelu určené na zpracování dat od uživatele (a nejsou určeny pro další instancování jinde v kódu), zabraňuje to znovupoužitelnosti takto napsaného databázového dotazu.

Proto používám návrhový vzor Repository. Složitější dotazy jsou zde uloženy v rámci jedné metody přístupné pomocí interface, který Repository implementuje. Díky Laravel service container nástroji je jednoduché získat konkrétní instanci Repository, buď pomocí Dependency injection v konstruktoru třídy (viz kód 4.4) nebo pomocí statické metody `\App::make()`, které se předá jméno požadované třídy (kód 4.5).

```php
<?php

namespace App\Http\Controllers;

use ...;

class UserController extends Controller
{
    public function index()
    {
        $userRepository = \App::make(UserRepositoryInterface::class);
        $users = $userRepository->all();
    }
}
```

Výpis kódu 4.5: Ukázka metody App::make

4.3 Využití předchozích kapitol

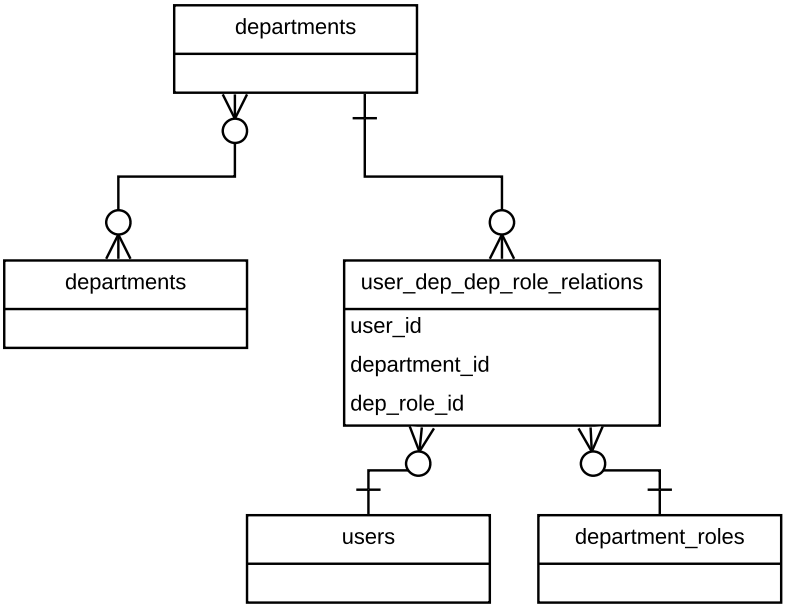
4.3.1 Diagram entit

V rámci implementace mi hodně pomohlo, že jsem měl zpracovaný návrh entit. Tento návrh pak stačilo přepsat do migrací, což jsou soubory, které Laravel používá pro tvorbu databázových tabulek. V průběhu implementace jsem se ovšem setkal s překážkami, které několikrát zapříčinily změnu diagramu entit, v jednom případě i dost razantně. O tom píšu v následující části.

Hierarchie oddělení

Firmy v reálném světe dodržují většinou nějakou hierarchii. Tuto hierarchii jsem jim chtěl umožnit vnést i do systému. Jak je vidět na zjednodušeném diagramu tříd 4.1, již při návrhu jsem se snažil tento problém s hierarchií vyřešit. V mém návrhu může mít oddělení jako členy jak zaměstnance, tak další oddělení. Ovšem v rámci zajištění správné funkcionality a pravomocí v aplikaci jsem chtěl, aby zaměstnanec mohl mít i v rámci oddělení roli. Proto je na diagramu znázorněna tabulka `user_dep_dep_relations` (budu se na ní dále odkazovat jako na Relation), která popisuje vztah mezi zaměstnancem, jeho rolí v oddělení a oddělením.

Kvůli zajištění maximální kompatibility s reálným světem jsem také při návrhu umožnil, aby jak oddělení, tak Relace (potažmo tedy uživatel s rolí) mohli být členy ve více odděleních najednou (proto je také arita mezi Odděleními nastavena na Many-To-Many). Jelikož je tabulka Relace již dekompozicí vztahu Many-To-Many mezi Uživatelem a Oddělením, tak tam je v diagramu arita mezi Oddělením a Relací jen One-To-Many.



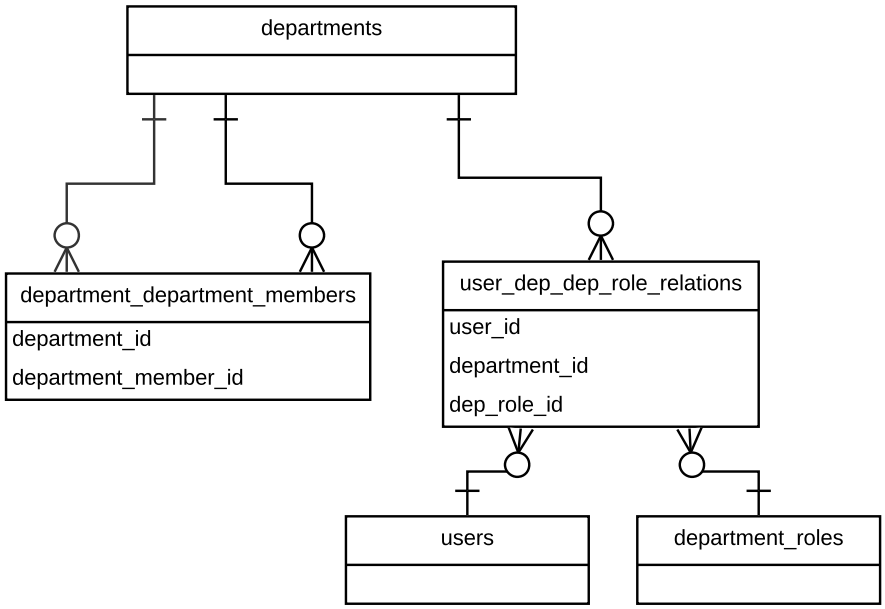
Obrázek 4.1: Diagram tříd znázorňující hierarchie v odděleních.

Toto se ukázalo být problém v implementaci pomocí polymorfního vztahu, jelikož Laravel neumožňuje, aby vztah k jedné entitě byl Many-To-Many a druhý jen One-To-Many. Framework řeší Many-To-Many polymorfní relace pomocí pivot tabulky, která obsahuje primární klíč rodičovské entity, primární klíč potomka a název třídy potomka, aby se zajistilo, že uložený primární klíč odkazuje na správnou entitu. Ovšem One-To-Many polymorfní relaci řeší jen pomocí přidání primárního klíče a typu jako atributů rodičovskému modelu (tabulce).

Bohužel, framework neumožňuje pro One-To-Many relaci také použít pivot tabulku, a proto bylo potřeba návrh přehodnotit. Nakonec jsem se rozhodl pro řešení s tabulkou pro relaci Oddělení <–> Oddělení-člen, jak ukazuje diagram 4.2. Toto řešení se mi nezdá tak elegantní jako pomocí polymorfního vztahu, ale nepotřebuje žádné tabulky navíc oproti polymorfnímu řešení a oddělení a zaměstnanci jsou ve svých tabulkách, což zvyšuje přehlednost.

4.4 Implementace dokumentů a dotazníků

Další částí této práce byla implementace Dotazníků a Dokumentů do systému.



Obrázek 4.2: Diagram konečné implementace hierarchie oddělení

4.4.1 Dokumenty

Implementace této sekce se obešla bez větších komplikací. Na základě databázového návrhu jsem měl dobrou představu o potřebných entitách této části aplikace a proto tvorba migrací a modelů nebyla problém. Začal jsem implementací dokumentů v administrátorském rozhraní. Ze základního CRUDu jsem pro tuto sekci vynechal metodu update, jelikož uživatel nemá možnost u Dokumentu upravit moc atributů, proto by mělo být v pořádku Dokument odstranit a nahrát nový.

Při implementaci této sekce jsem se naučil pracovat s Laravel fasádou Storage, která slouží jako rozhraní k PHP balíčku Flysystem[6]. Práce s touto fasádou je velmi intuitivní a jen potvrzuje tvrzení, že Laravel je framework zaměřený na zpříjemnění práce vývojáře. Soubory dokumentů s ukládají do složky *storage/app/documents*, kde se nadále třídí podle slugů jednotlivých instancí modulů. V cílovém adresáři jsou soubory uloženy pod vygenerovaným názvem a jejich původní název je uložen do databáze, spolu s cestou k souboru.

Zaměstnanecká část aplikace pak obsahuje pouze tabulku nahraných dokumentů, kdy se po kliknutí na záznam v tabulce dokument stáhne.

[6]https://github.com/thephpleague/flysystem

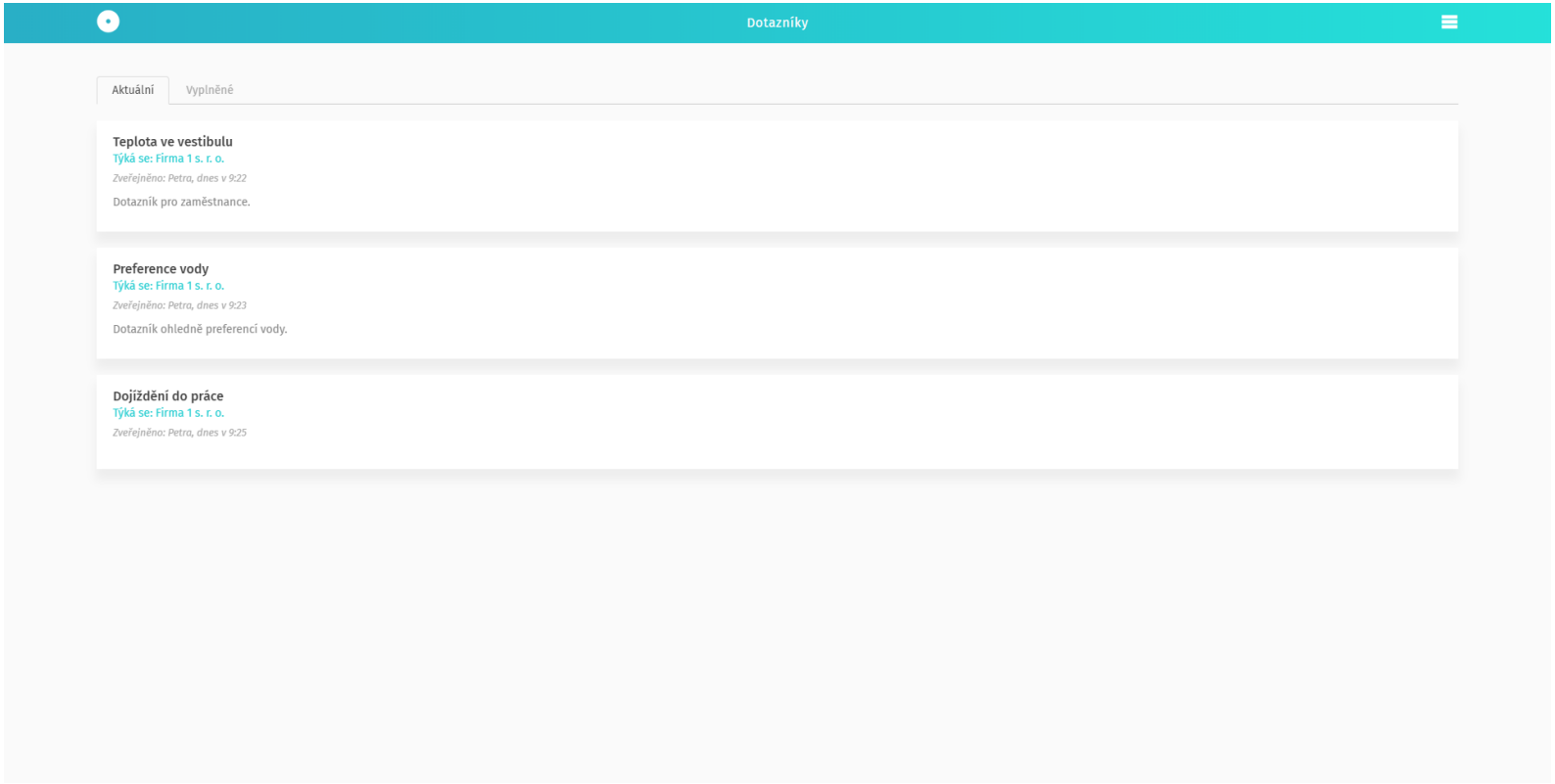
Možná vylepšení

Tento systém bohužel není dokonalý, a nepovedlo se mi v rámci prototypu zprovoznit nahrávání souborů pomocí AJAX. To by nadále umožnilo například zobrazit náhled nahrávaného souboru (pro fotky a obrázky), nebo i nahrávání více souborů. Dále bych do budoucna chtěl, aby se soubory dali nahrávat také jen pro nějaká oddělení, stejně jako například fungují oznámení. Tím by se zabránilo zbytečnému zahlcení přehledů souborů.

4.4.2 Dotazníky

Už při návrhu jsem počítal s tím, že implementace Dotazníků bude náročnější než implementace Dokumentů, a proto jsem jí vyhradil více času. Při prvním návrhu jsem úplně opomněl možnost, že je potřeba nějak ukládat odpovědi uživatelů, proto jsem se musel k návrhu vrátit a přepracovat ho. Napodruhé už bylo vše v pořádku.

Dotazníky se v systému dělí na dvě části, dotazovací a vyplněnou. Dotazovací část obsahuje entity Dotazníku (*Questionnaire*) a otázky (*Question*). Tyto entity mohou vytvářet zatím pouze administrátoři. Druhá část jsou vyplněné dotazníky, tvořené entitami Vyplněný dotazník (*CompletedQuestionnaire*) a Odpověď (*Response*). Tyto entity samozřejmě mohou vytvářet všichni zaměstnanci. Na dalším obrázku je možné vidět obrazovku s přehledem dotazníků v zaměstnaneckém rozhraní.



Obrázek 4.3: Přehled dotazníků v zaměstnaneckém rozhraní.

Možná vylepšení

Jak vyplynulo z uživatelského testování v kapitole 5, tvorba dotazníku uživatelem není úplně šťastně navržena. Nejdříve se po administrátorovi chce, aby vytvořil entitu dotazníku a až po odeslání formuláře se dostane na stránku, kde může přidávat otázky. Tímto postupem jsem se chtěl vyhnout velké metodě na ukládání dotazníku v controlleru, jelikož ten by poté spravoval vytváření entit Dotazníku i Otázek. Ale jak je vidět, uživatelům tenhle postup nepřijde intuitivní a proto je ho potřeba změnit.

Další vylepšení do budoucna je možnost odložit zveřejnění dotazníku na konkrétní datum a čas. Toho by se dalo docílit pomocí plánování úloh v Laravelu[7].

4.5 Nasazení aplikace do testovacího prostředí

V poslední kapitole implementační části bych rád čtenáři popsal postup nasazení aplikace na testovací server. Díky tomu, že aplikace poběží na stroji velmi podobném produkčnímu serveru, tak se dá mnohdy najít pár dalších chyb. Nejprve je potřeba zkompilovat soubory stylů do produkční verze, na to nám poslouží příkaz `php artisan lfl:com --production`. Na serveru je poté potřeba stáhnout zdrojové kódy aplikace přes Git a nastavit správné prostředí aplikace v souboru *.env*. Poslední částí je založení testovacích databází a naplnění aplikace testovacími daty.

[7] https://laravel.com/docs/7.x/scheduling

Kapitola 5

Testování

Součástí vývoje jakéhokoliv softwaru je jeho testování. Testovat se dá vše, od výkonu systému, až po uživatelské rozhraní. V rámci této práce se zaměřím na uživatelské testování.

5.1 Význam uživatelského testování

Programátor si může myslet, že se dokáže vžít do člověka, který bude jeho systém používat, ale úplná pravda to není. Je proto důležité najít lidi, kteří systém budou používat, nebo kteří by alespoň takový systém mohli používat. Dobře připravené uživatelské testy pak mohou programátorovi prozradit, jestli cílová skupina uživatelů používá systém tak, jak to programátor v první řadě zamýšlel.

Dá se očekávat, že díky testovacím osobám, se odhalí hlavní chyby v návrhu UX. Pokud se takové testování provádí při vývoji, tak je mnohem jednodušší a levnější produkt upravit, než když už se jedná o produkt nasazený na produkci. Proto je doporučováno uživatelské testování provádět při vývoji víckrát [19].

5.2 Testovací scénáře

Hlavní složkou uživatelského testování jsou testovací scénáře. Jedná se o sadu úkolů, které má testovací osoba v systému provést, při kterých ji sleduje vedoucí testu, který si dělá poznámky o průběhu. Tyto scénáře mají simulovat reálné používání aplikace uživatelem po jejím nasazení do produkce [19].

5.3 Testovací skript

Před samotným testování a po rozmyšlení testovacích scénářů, je dobré si sepsat testovací skript. Obsahuje seznam úkonů, jak půjdou po sobě a tím si vý-

vojář může ověřit, že takové pořadí úkonů dává smysl. Skript také napomáhá konzistenci postupu testování, když se má testovat více osob. Navíc, když se takový skript dá testovací osobě do ruky, může se k němu kdykoliv obrátit a ujistit se v tom, co má právě dělat [19].

5.4 Testování modulu Zaměstnání

Pro testování byly vybráni dva uživatelské účty, zaměstnanecký účet a administrátor, aby si testovací osoba vyzkoušela obě rozhraní aplikace. Aplikace byla pokaždé naplněna testovacími daty, které simulovali reálný stav aplikace.

5.4.1 Zaměstnanecké rozhraní

1. Přihlásit se jako zaměstnanec.
2. Vytvořit novou omluvenku.

   V tomto scénáři se testuje orientace testovací osoby v uživatelském rozhraní modulu. Zkoumá se, jestli uživatel najde menu a zde najde položku pro omluvenky. Dále je po testovací osobě požadováno, aby našla tlačítko „+“ pro přidání nové omluvenky, následuje vyplnění omluvenky a odeslání.
3. Smazat jinou omluvenku.

   Zde se zkoumá, jestli testovací osoba využije toho, že je již na obrazovce s přehledem omluvenek a jestli použije tlačítko přímo v seznamu, nebo jestli půjde přes detail omluvenky.
4. Stáhnout dokument „Smlouva o zpracování osobních údajů.pdf“.

   Tento test ověřuje, zda uživatel ihned zamíří do menu, nebo jestli se bude snažit dokumenty hledat jinde. Dále ověřuje, jak uživatel reaguje na to, že má kliknout na řádek tabulky s dokumentem pro jeho stažení.
5. Vyplnit dotazník „Teplota ve vestibulu“.

   Zde se zkoumá, jak uživatel reaguje na rozložení prvků na obrazovce, jestli jednoduše najde tlačítko na vyplnění dotazníku, nebo jestli se snaží klikat na jednotlivé otázky v tabulce.
6. Návrat na domovskou obrazovku (Seznam oznámení).

   Tento test zkoumá, jestli testovací osoba použije logo firmy Lyfle jako odkaz na domovskou obrazovku, nebo jestli radši vyhledá položku oznámení v menu.
7. Odhlášení ze systému.

   Tento scénář ověřuje, jestli si uživatel všiml tlačítka pro odhlášení v menu.

5.4.2 Administrátorské rozhraní

Druhá část testování se zaměřila na administrátorské rozhraní.

1. Přihlásit se jako administrátor.

2. Vytvořit omluvenku pro Petra Novotného.

   Tento scénář zkoumal uživatelovu orientaci v administrátorském rozhraní. Zkoumalo se, jestli uživatel najde tlačítko pro přidání nové omluvenky a orientace na samotné obrazovce sloužící pro přidávání omluvenky.

3. Nahrát dokument „Soubor1.pdf“ s popiskem.

   Zde se testovalo, zda uživatel najde položku v menu pro nahrání dokumentů a jestli pro něj bude tento proces intuitivní.

4. Smazat druhý dokument.

   Zde se po testovací osobě chtělo, aby klikla na tlačítko pro smazání dokumentu, které bylo v tabulce nahraných dokumentů na stejné řádce.

5. Vytvořit dotazník „Preference vody“ s jednou otázkou.

   V tomto bodě se hlavně zkoumalo, jestli přijde testovací osobě intuitivní ovládání vytváření nového dotazníku s otázkami. Dále se po testovací osobě chtělo, aby nově vytvořený dotazník zveřejnila pomocí tlačítka.

6. Zveřejnit dotazník „Nové klávesnice“.

   Tento scénář měl prověřit, jestli testovací osoba po předchozí zkušenosti najde bez problému tlačítko pro zveřejnění již existujícího dotazníku.

7. Přidat uživatele Jana Nováka jako člena IT Oddělení.

   Zde se zkoumalo, kde bude testovací osoba hledat možnost přiřazení uživatele do oddělení. Dále se zkoumalo, jestli testovací osobě přijde intuitivní přidávání nových uživatelů do oddělení na obrazovce pro úpravu oddělení.

8. Navrácení na domovskou obrazovku (Seznam oznámení).

   Stejně jako u zaměstnaneckého rozhraní se zde testovalo, jestli uživatel pro tuto akci použije logo firmy Lyfle.

9. Odhlášení z aplikace.

   Tento scénář zkoumal, kde bude uživatel hledat tlačítko pro odhlášení, jestli po předchozí zkušenosti v menu, nebo pod jménem aktuálně přihlášeného uživatele.

5.4.3 Testovací osoby

Do testování systému jsem zapojil dvě osoby. Oba jsou zaměstnáni v nějaké firmě a tudíž patří do cílové skupiny této aplikace.

Detaily testovaných osob

- **Jakub O.** (48 let)

  Uživatel je vysokoškolsky vzdělaný. Je normálním uživatelem softwaru a o IT se zajímá jen zběžně.

- **Marie M.** (44 let)

  Uživatelka je také vysokoškolsky vzdělaná, ale s IT nemá skoro žádné zkušenosti a je jen minimálním uživatelem softwaru.

5.4.4 Výsledky testování a navrhované změny

Bylo vidět, že Jakub O. má s webovými technologiemi větší zkušenosti, jelikož plnil testovací scénáře rychleji než Marie. Oba testování celkem zvládali, ale přesto mi poskytli dost informací, na které bych jako vývojář nepomyslel jako na důležité.

Jelikož bylo testování prováděno až těsně před odevzdáním, tak nebyl čas na udělání změn podle výsledků testů. Ovšem má práce na tomto systému nekončí s odevzdáním práce, proto si z tohoto testování vezmu důležité poznatky do budoucna.

Problémy s nalezením menu v zaměstnaneckém rozhraní

Marie M. měla ze začátku problém poznat, že ikona „hamburgeru“ slouží pro otevření menu. Navrhla, že by pomohlo, kdyby vedle té ikony byl nápis *Menu*.

Problémy najít tlačítko „+“ pro přidání omluvenky v zaměstnaneckém rozhraní

Jak vyšlo z testování, po prvním přihlášení do aplikace ani jedna z testovacích osob automaticky neodklikla (tak, jak bych to udělal já) souhlas s používáním cookies. Kvůli tomu rámeček, který informuje o používání cookies, překrýval tlačítko pro přidání omluvenky.

Zdržení při stahování dokumentů

Obě testovací osoby nakonec zkusily kliknout na dokument v tabulce, ale oba řekli, že by radši viděli u dokumentu nějaké tlačítko s nápisem *Stáhnout* nebo s ikonou stahování.

Nepřehledná tvorba omluvenek v administrátorském rozhraní

Marie M. si stěžovala, že tvorba omluvenek je zbytečně roztažená dolů na stránku, když je vpravo tolik místa. Řešením by mohlo být panel pro tvorbu omluvenky roztáhnout na celou obrazovku a přesunout nějaká políčka vedle sebe do řady.

Nepřehledné mazání dokumentů

Tlačítko smazat by mohlo být více zvýrazněné, Jakub si ho na první pohled nevšiml a proto na dokument nejdříve klikl v domnění, že se možná označí pro smazání, ale místo toho se stáhl. Řešením by mohlo být místo odkazu tlačítko.

Tvorba dotazníku a otázek v jednom kroku

Z testování vyplynulo, že uživatelům se více zamlouvá, když tvoří dotazník zároveň s otázkami. Marii zmátlo mé řešení, že se nejdříve vyplní název dotazníku a až posléze na další stránce se mohou přidávat otázky. Proto do budoucna, by bylo lepší implementovat vytváření dotazníku v jednom kroku.

Přidání člena do oddělení

Ani jednu z testovaných osob nenapadlo, že by měli kliknout vedle aktuálních členů oddělení, že se jim poté otevře možnost vybrat další. Kvůli tomu, že oddělení už mělo jednoho člena, tak byl skrytý popisek, že stačí do toho políčka kliknout pro přidání dalšího. Do budoucna by tedy bylo lepší, aby se přidávali zaměstnanci buď pomocí dedikovaného tlačítka „+“, nebo aby tam byl trvalý popisek, jak přidávat další členy.

Světle modré prvky v aplikaci jsou někdy špatně čitelné

Marie poukázala na to, že některé prvky se světle modrým pozadím mohou být špatně čitelné pro špatně vidící lidi. Řešením je tuto barvu trochu ztmavit.

Závěr

Práce na tomto projektu pro mne byla velkým přínosem. Ze školy mám dost teoretických znalostí, nějaké základy programování a softwarového návrhu. To hlavní je ovšem praxe, kterou jsem díky téhle práci úspěšně rozšířil. Na začátku projektu bylo provedeno několik konzultací se zadavatelem, kde jsem si utřídil funkční i nefunkční požadavky na systém.

Dále jsem provedl návrh entit a uživatelského rozhraní, čímž jsem položil základy pro úspěšnou implementaci. Následovala praktická část práce. Začal jsem přenesením existujícího projektu Školky do nového Git repositáře a začal pracovat na editaci projektu, což jak se v průběhu ukázalo byla časově nejnáročnější část praktické části mé bakalářské práce. Dále následovala testovací část, kdy jsem provedl uživatelské testování aplikace pro ověření správně navrženého UX.

Tento systém mou bakalářskou prací teprve začal, do budoucna na něm budu intenzivně pracovat. Začnu od vylepšení, která vzešla z testování. Po interním testování ve firmě, mě následně čeká příprava projektu na testování v dalších společnostech.

Bibliografie

1. OTWELL, Taylor. *Facades* [online]. Laravel LLC, 2020 [cit. 2020-06-04]. Dostupné z: `https://laravel.com/docs/7.x/facades`.
2. BRADÁČ, Jan. *Mojeskolka.info – tvorba uzavřené komunitní sítě pro školky*. Praha, 2015. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
3. COLEY, Phil. *User Requirements* [online]. Coley Consulting, 2019 [cit. 2020-05-20]. Dostupné z: `https://www.coleyconsulting.co.uk/require.htm`.
4. W3TECHS. *Usage statistics of server-side programming languages for websites* [online]. Q-Success, 2020 [cit. 2020-05-30]. Dostupné z: `https://w3techs.com/technologies/overview/programming_language`.
5. LARASHOUT. *What is Laravel and Why You Should Learn it?* [online]. Larashout, 2018 [cit. 2020-05-10]. Dostupné z: `https://www.larashout.com/what-is-laravel-and-why-you-should-learn-it`.
6. INFINIJITH APPS & TECHNOLOGIES. *25 Best PHP Frameworks for Web Development in 2020* [online]. Medium, 2020 [cit. 2020-05-10]. Dostupné z: `https://medium.com/@infinijith/25-best-php-frameworks-for-web-development-in-2020-20dcd85e43ca`.
7. CHRISTENSSON, Per. *Web Application* [online]. TechTerms.com, 2014 [cit. 2020-05-09]. Dostupné z: `https://techterms.com/definition/web_application`.
8. YEEPLY. *Web App Development: Advantages and Disadvantages* [online]. Yeeply, 2017 [cit. 2020-05-09]. Dostupné z: `https://en.yeeply.com/blog/advantages-and-disadvantages-of-web-app-development`.

9. BECKER, Karen; ANTUAR, Nicholas; EVERETT, Cherie. Implementing an employee performance management system in a nonprofit organization. *Nonprofit Management and Leadership* [online]. 2011, roč. 21, č. 3, s. 255–271 [cit. 2020-05-06]. Dostupné z DOI: 10.1002/nml.20024.

10. DIAWATI, Prety; PARAMARTA, Vip; PITOYO, Djoko; FITRIO, Tomy; MAHRANI, Sri Wiyati. Challenges of implementing an employee management system for improving workplace management effectiveness. *Journal of Environmental Treatment Techniques* [online]. 2019, roč. 7, s. 1200–1203 [cit. 2020-05-08].

11. DEMČÁK, Marek. *Online dotazník - jak na to?* [online]. Webky.cz, 2020 [cit. 2020-06-04]. Dostupné z: https://www.online-dotaznik.cz/.

12. JONES, Steve; MURPHY, Fiona; EDWARDS, Mark; JAMES, Jane. Doing things differently: advantages and disadvantages of Web questionnaires. *Nurse researcher* [online]. 2008, roč. 15, s. 15–26 [cit. 2020-05-09]. Dostupné z DOI: 10.7748/nr2008.07.15.4.15.c6658.

13. CASTEN, Craig. *What is a slug?* [online]. Award Force, 2018 [cit. 2020-06-03]. Dostupné z: https://www.awardforce.com/blog/articles/what-is-a-slug/.

14. *PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains* [online]. JetBrains [cit. 2020-05-31]. Dostupné z: https://www.jetbrains.com/phpstorm/.

15. *WampServer* [online]. 2020 [cit. 2020-05-31]. Dostupné z: https://sourceforge.net/projects/wampserver/.

16. [online] [cit. 2020-05-31]. Dostupné z: https://git-scm.com/.

17. FOWLER, Martin. *Active Record* [online]. 2002 [cit. 2020-06-01]. Dostupné z: https://www.martinfowler.com/eaaCatalog/activeRecord.html.

18. OTWELL, Taylor. *Blade Templates* [online]. Laravel LLC, 2020 [cit. 2020-06-01]. Dostupné z: https://laravel.com/docs/7.x/blade.

19. MURPHY, Christopher. *A Comprehensive Guide To User Testing* [online]. 2018 [cit. 2020-06-03]. Dostupné z: https://www.smashingmagazine.com/2018/03/guide-user-testing/.

20. LAMPRECHT, Emil. *The Difference Between UX And UI Design - A Layman's Guide* [online]. CareerFoundry, 2019 [cit. 2020-06-03]. Dostupné z: https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/.

21. *What is CRUD?* [online]. Codecademy, 2020 [cit. 2020-06-04]. Dostupné z: https://www.codecademy.com/articles/what-is-crud.

22. *AJAX Introduction* [online]. Refsnes Data, 2020 [cit. 2020-06-04]. Dostupné z: https://www.w3schools.com/xml/ajax_intro.asp.

23. *Introduction to XML* [online]. Refsnes Data, 2020 [cit. 2020-06-04]. Dostupné z: `https://www.w3schools.com/xml/xml_whatis.asp`.

24. *JavaScript HTML DOM* [online]. Refsnes Data, 2020 [cit. 2020-06-04]. Dostupné z: `https://www.w3schools.com/js/js_htmldom.asp`.

Příloha **A**

Seznam použitých zkratek

UX User experience – Jak doslovný překlad napovídá, jedná se o interakci, nebo zkušenost uživatele s nějakým produktem. V softwarovém inženýrství se touto zkratkou také často označuje sousloví *user experience design*, čímž se označují prvky, které utvářejí interakci mezi uživatelem a nějakým systémem [20].

CRUD Create, Read, Update, Delete – Jedná se o akronym anglických slov pro vytváření, čtení, úpravu a mazání dat. Obvykle se používá pro definici metod na modelu/controlleru pro přístup k datům přes API [21].

AJAX Asynchronous JavaScript And XML – Není to programovací jazyk, používá kombinaci Http požadavků (pro vyžádání dat ze serveru), JavaScriptu a HTML DOMu (pro zobrazení dat), Ajax poskytuje možnost jak obnovovat data na webové stránce, bez nutnosti obnovit celou stránku [22].

XML eXtensible Markup Language – Velmi používaný značkovací jazyk pro zobrazování a přenos dat [23].

DOM Document Object Model – DOM definuje standard pro přístup k dokumentu, od toho vychází HTML DOM, který definuje standard pro přístup k HTML dokumentu [24].

Příloha B

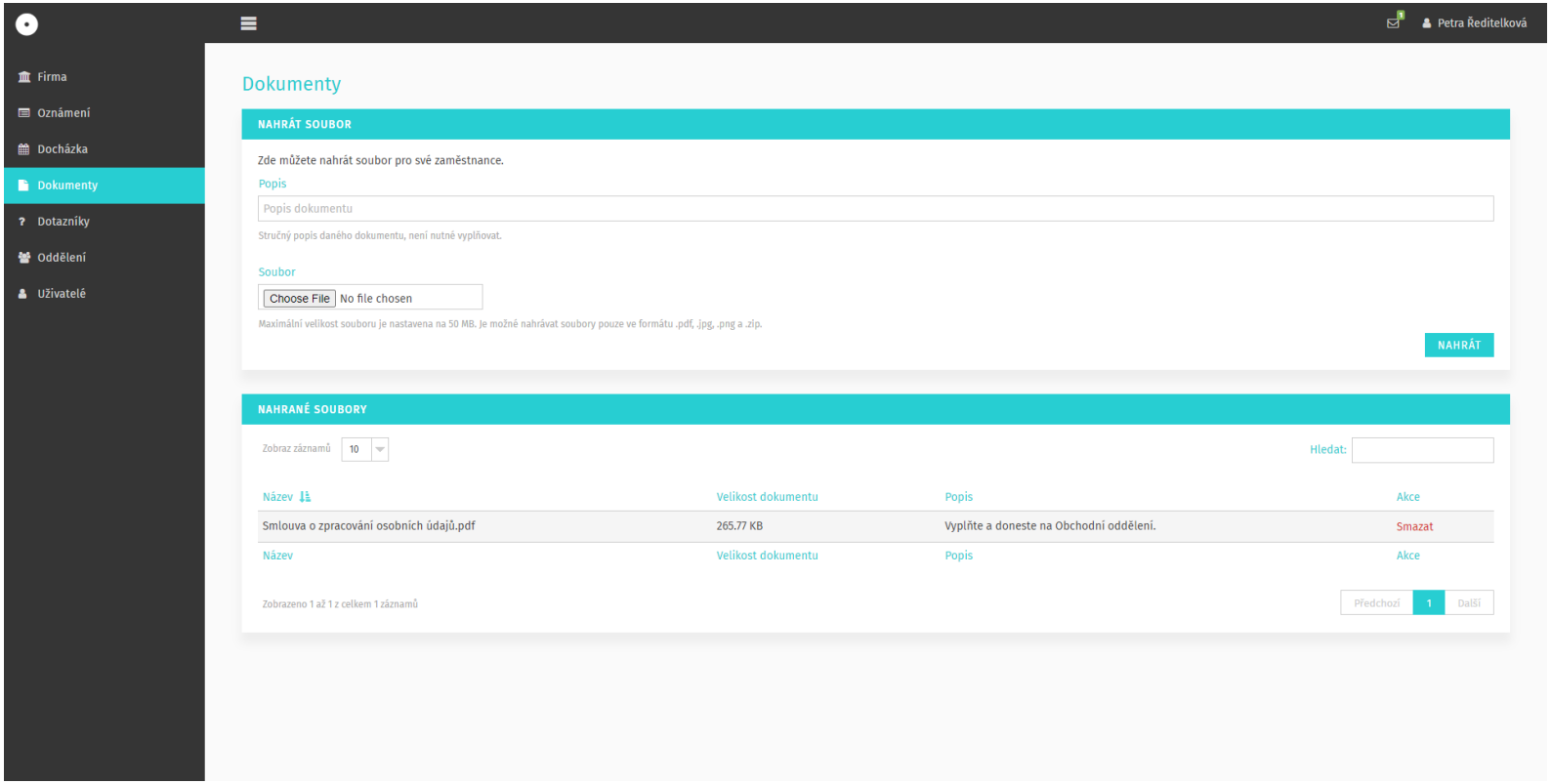
Obsah přiloženého disku

- `readme.txt` ..... stručný popis obsahu disku
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF

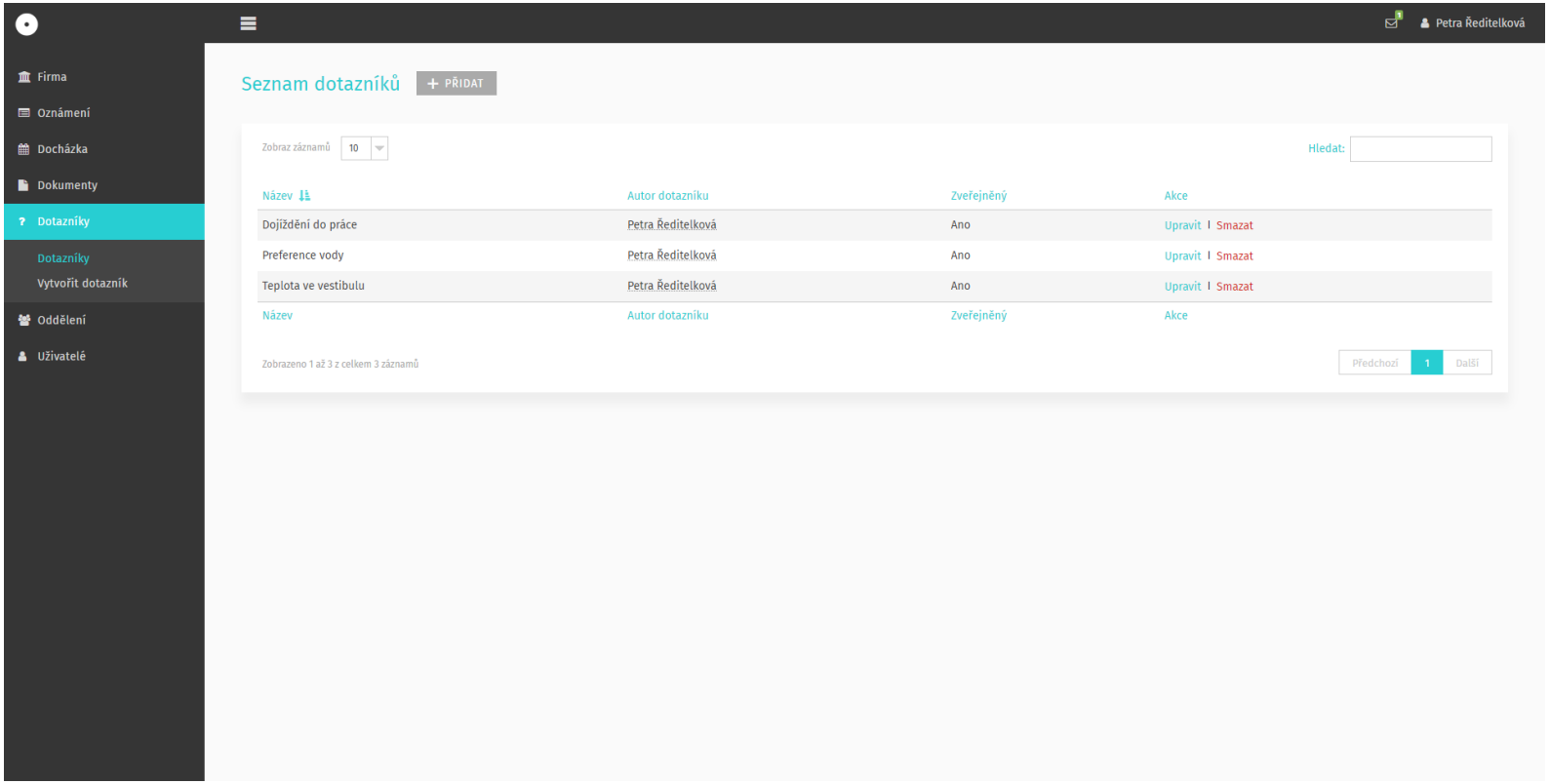
Příloha C

Ukázky vytvořeného prototypu aplikace

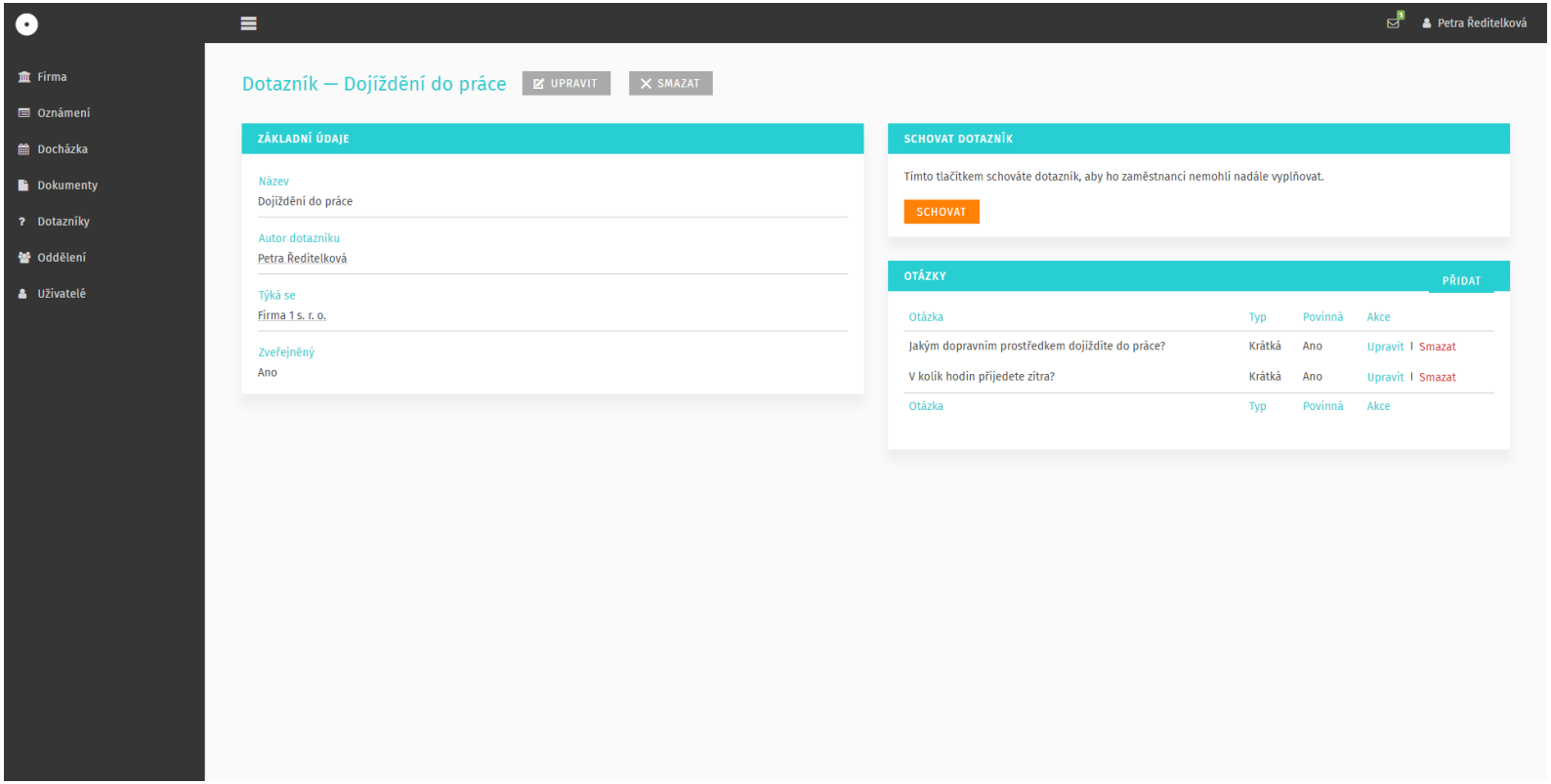
V této příloze najdete ukázky z vytvořeného prototypu aplikace.



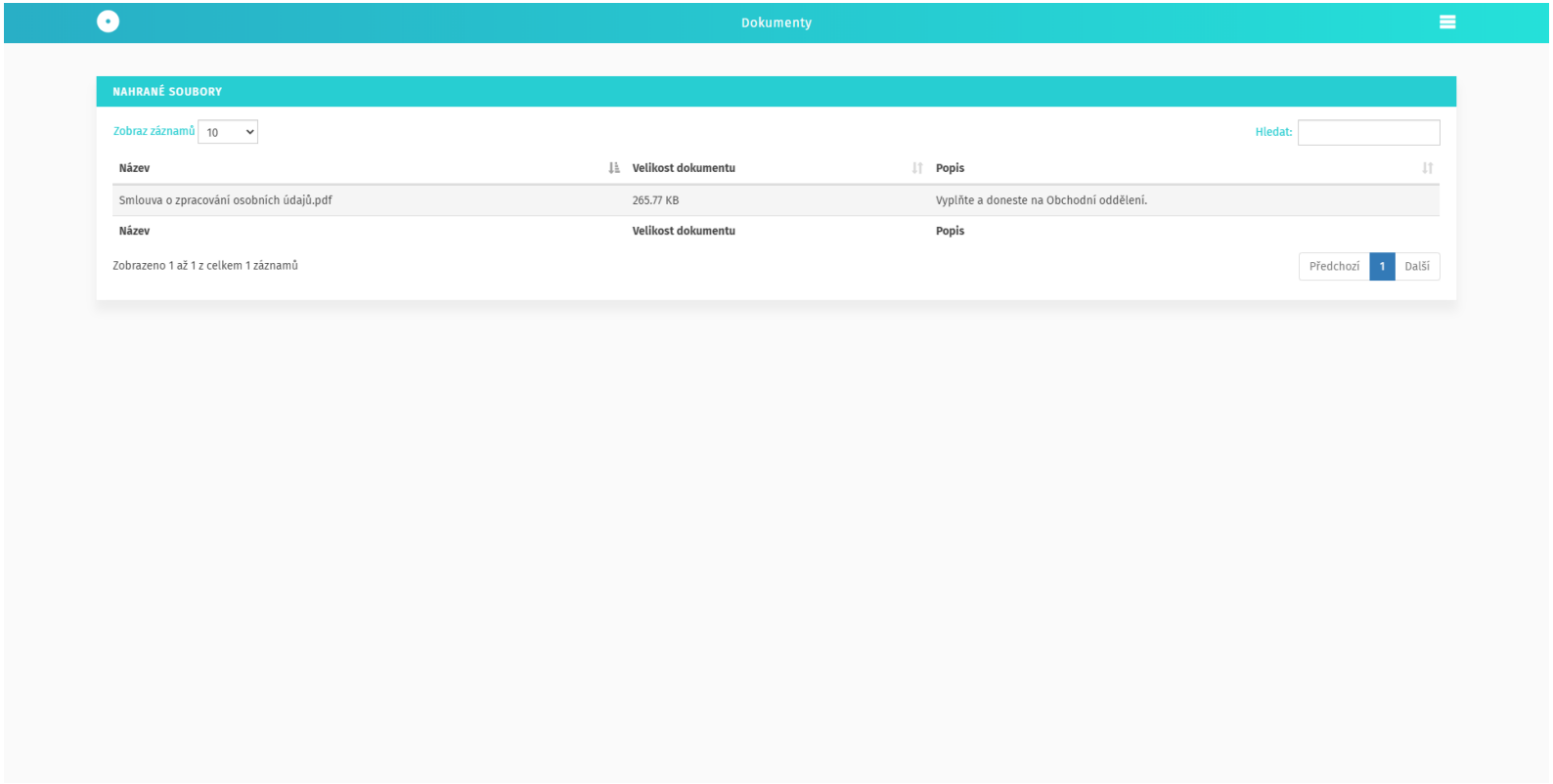
Obrázek C.1: Přidávání dokumentů v administrátorském rozhraní



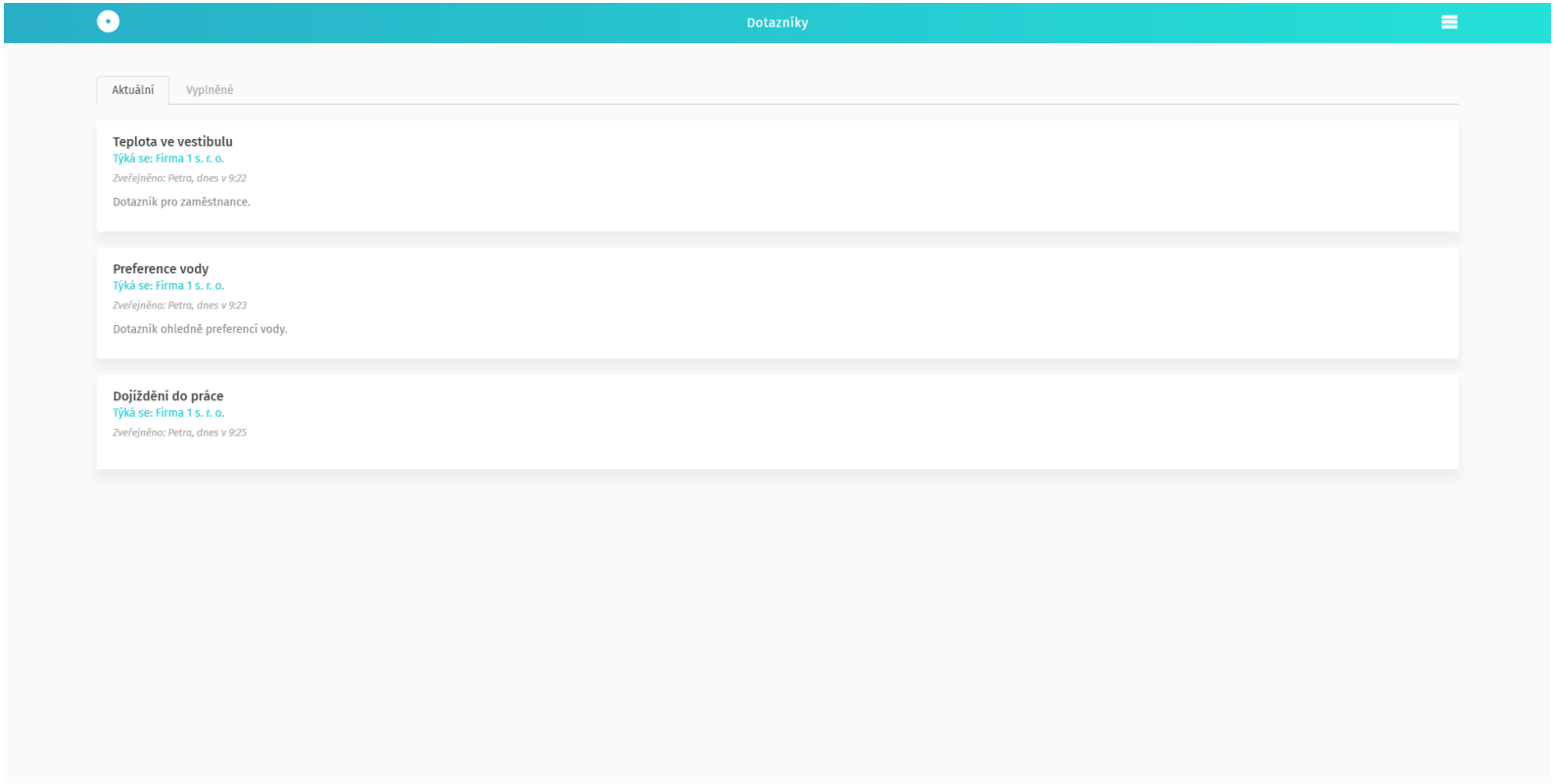
Obrázek C.2: Přehled dotazníků v administrátorském rozhraní



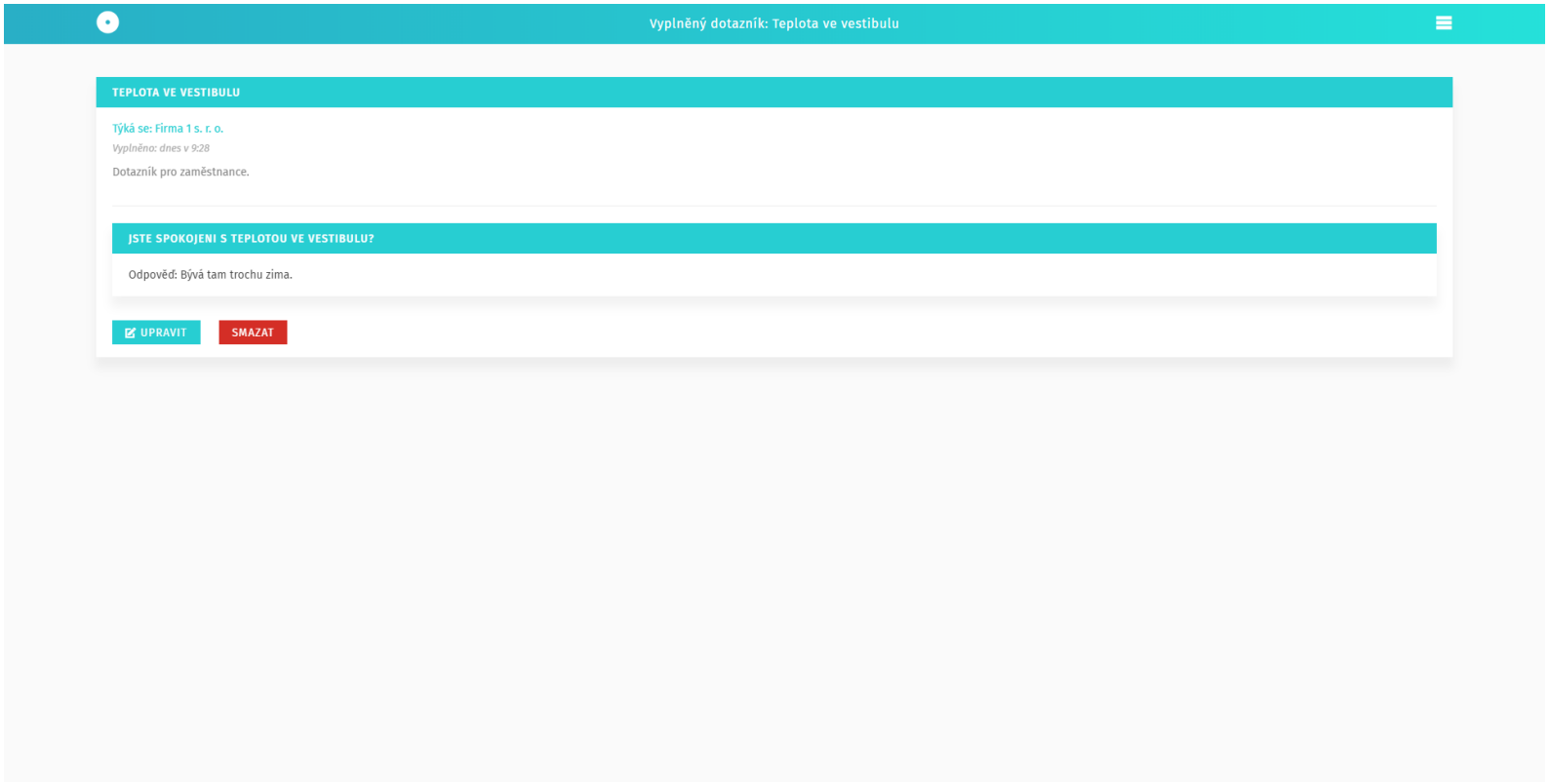
Obrázek C.3: Detail dotazníku v administrátorském rozhraní



Obrázek C.4: Přehled dokumentů v zaměstnaneckém rozhraní



Obrázek C.5: Přehled dotazníků v zaměstnaneckém rozhraní

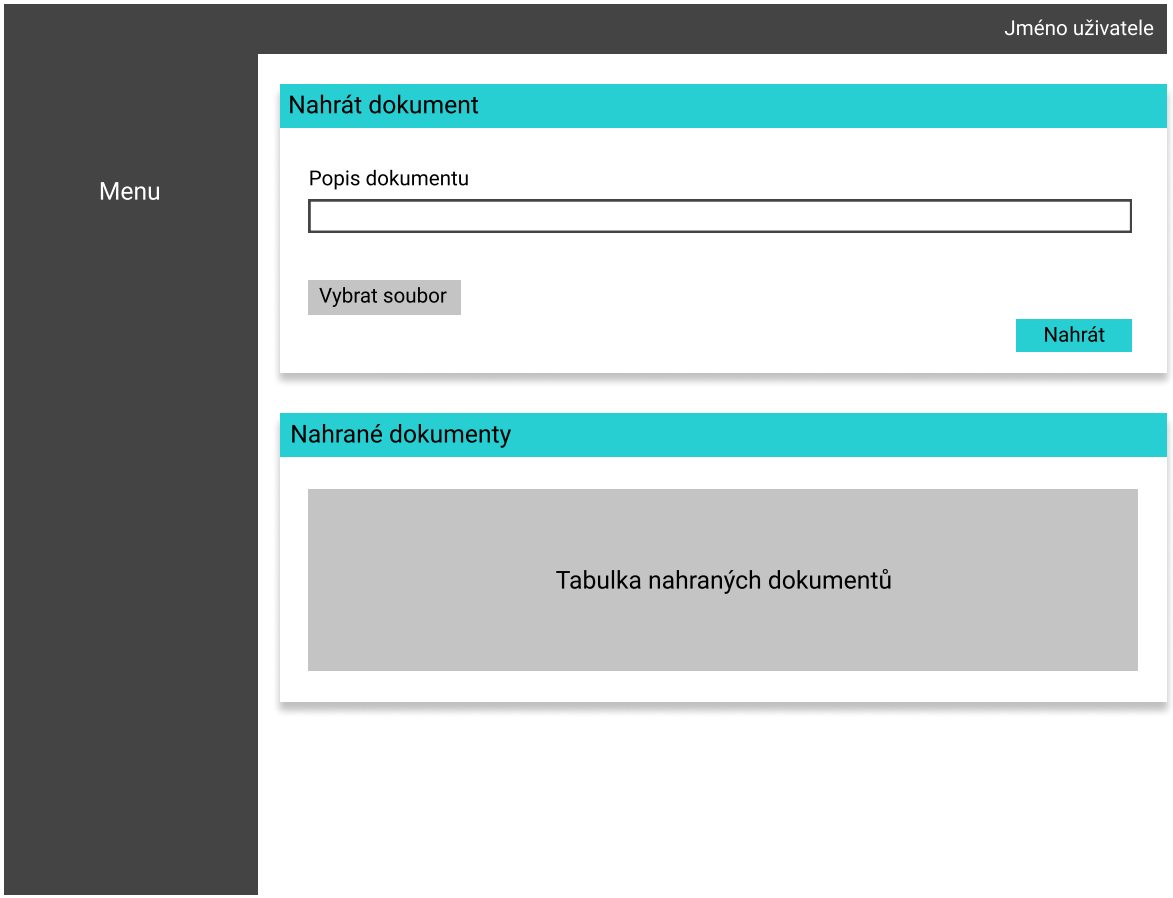


Obrázek C.6: Detail vyplněného dotazníku v zaměstnaneckém rozhraní

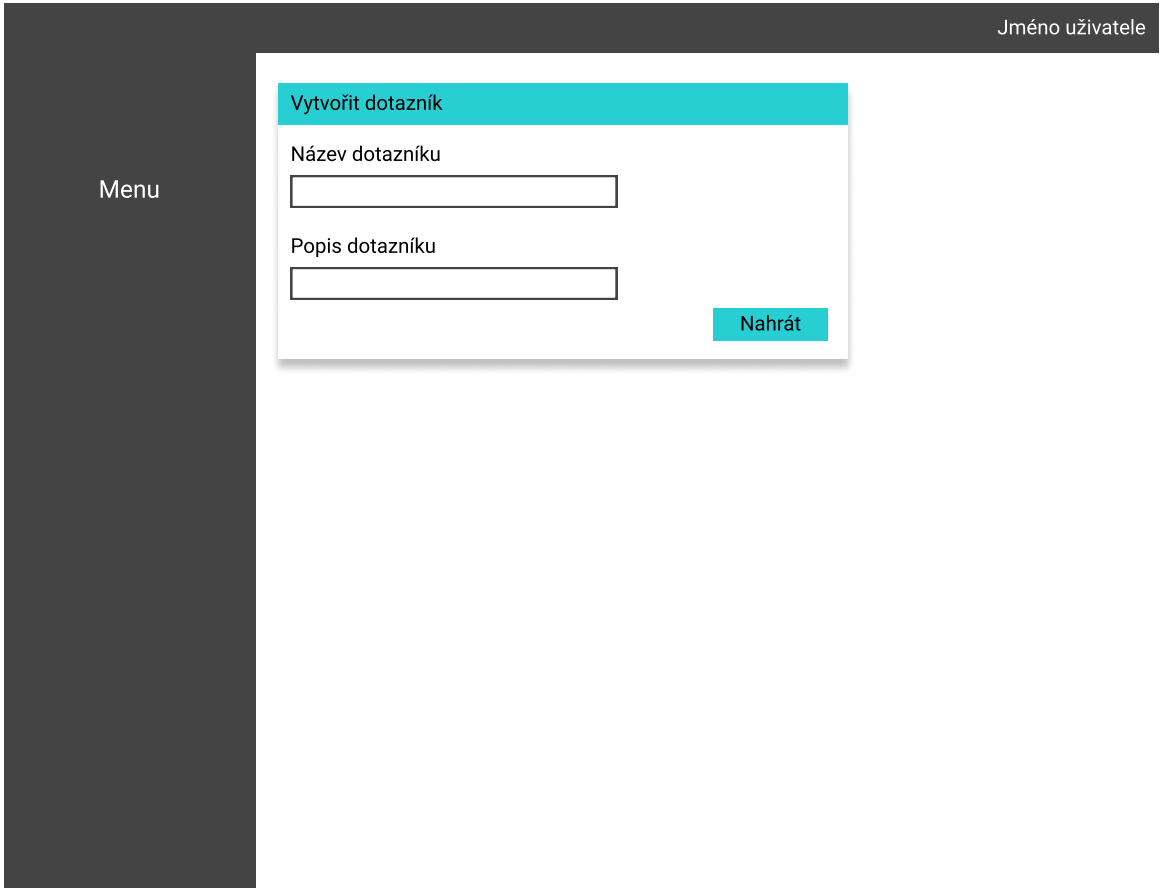
Příloha D

Návrhy uživatelského rozhraní dokumentů a dotazníků

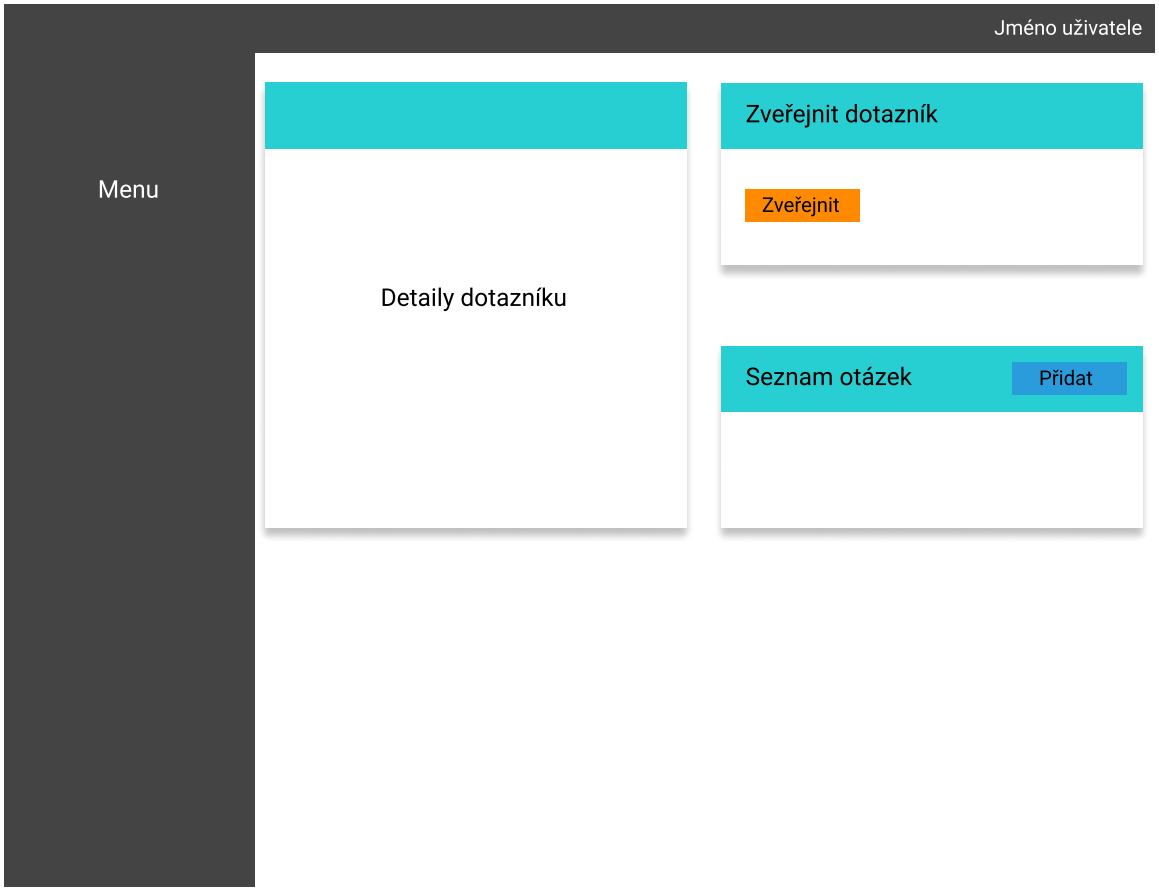
V této příloze najdete návrh uživatelského rozhraní pro dotazníky a dokumenty.



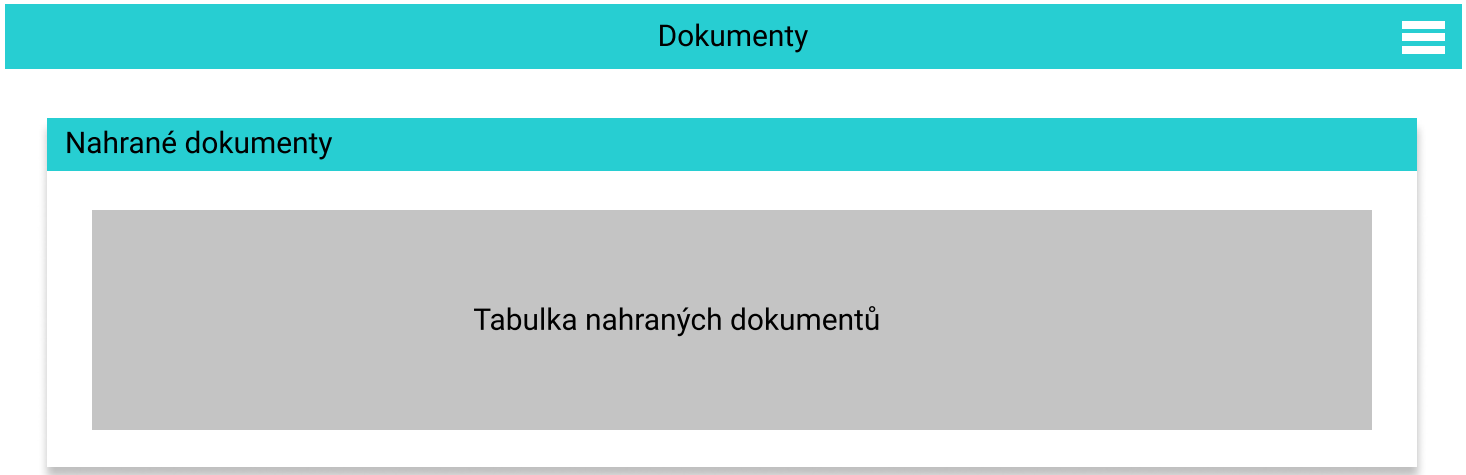
Obrázek D.1: Přidávání dokumentů v administrátorském rozhraní



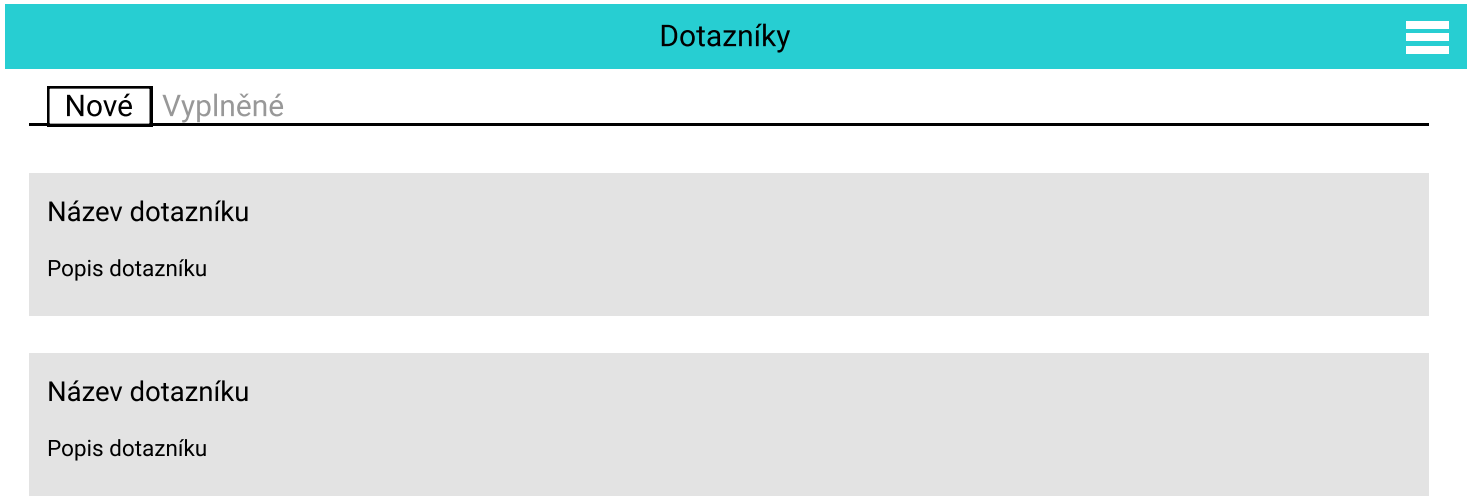
Obrázek D.2: Vytváření dotazníků v administrátorském rozhraní



Obrázek D.3: Detail dotazníku v administrátorském rozhraní



Obrázek D.4: Přehled dokumentů v zaměstnaneckém rozhraní



Obrázek D.5: Přehled dotazníků v zaměstnaneckém rozhraní

Obrázek D.6: Stránka vyplňování dotazníku v zaměstnaneckém rozhraní