Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace pro evidenci odpracovaného času

2024

Jan Týče

Vedoucí práce:
RNDr. Arnošt Večerka

Studijní program: Aplikovaná informatika,
kombinovaná forma

Bibliografické údaje

Autor:	Jan Týče
Název práce:	Aplikace pro evidenci odpracovaného času
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	49
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Jan Týče
Title:	Software Application for Time Sheet Records
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Applied Computer Science, combined form
Supervisor:	RNDr. Arnošt Večerka
Page count:	49
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Zpracovávám návrh a implementaci webové aplikace pro evidenci, kontrolu a vyhodnocení odpracovaného času v advokátní kanceláři/inkasní agentuře. Účelem aplikace je minimalizovat čas strávený evidencí práce, umožnit přístup ze zařízení mimo podnikovou síť (zejména mobilních telefonů) a poskytnout přehled o odvedené práci jednotlivých pracovníků i přehled o práci odvedené pro jednotlivé klienty. Aplikace je vytvořena za použití jazyka PHP a frameworku Laravel.

Synopsis

I am designing and implementing a web application for recording, controlling and evaluating the time worked in a law firm/collection agency. The purpose of the application is to minimize the time spent recording work, to allow access from devices outside the company network (especially mobile phones) and to provide an overview of the work done by individual workers as well as an overview of the work done for individual clients. The application is created using PHP language and Laravel framework.

Klíčová slova: webová aplikace; PHP; Laravel

Keywords: web application; PHP; Laravel

Rád bych poděkoval vedoucímu práce RNDr. Arnoštu Večerkovi za citlivý přístup a cenné podněty k dopracování práce. Děkuji také řediteli naší společnosti, že mi umožnil aplikaci implementovat v praxi a kolegům z IT oddělení za nezbytnou úpravu na straně podnikové aplikace.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

Seznam tabulek

1 Cíle práce

Cílem práce je vytvořit webovou aplikaci pro evidenci odpracovaného času v konkrétní inkasní agentuře a advokátní kanceláří zabývající se hromadnou správou pohledávek. Každý pracovník totiž vytvoří několik desítek úkonů za den. Evidence slouží jako stěžejní podklad pro alokaci nákladů z hlediska manažerského účetnictví a také jako podklad pro kontrolu a poskytování zpětné vazby pracovníkům. Aplikace je vyvíjena přímo pro tuto konkrétní implementaci a bude tak přizpůsobena pro oboustranné propojení s podnikovou aplikací.

Webová aplikace bude sloužit jako alternativa evidence odvedené práce v dosavadní podnikové aplikaci, která není pro tuto činnost optimalizována, má pomalejší odezvu, a proto tato činnost zabere uživateli 20 až 30 minut denně. Z tohoto důvodu někteří pracovníci zápis do evidence odkládají, což má negativní vliv na její věrnost. Aplikace si tedy klade za cíl usnadněním této činnosti přispět jak ke snížení nákladů na samotnou evidenci, tak ke zvýšení její přesnosti.

Vyvíjená aplikace navíc umožní přístup z jiných zařízení než podnikových PC, a to zejména mobilních telefonů. Posledním cílem je umožnit uživateli filtrování a přednastavené statistické vyhodnocení jak jeho práce, tak práce jiných uživatelů, podle rozsahu přidělených oprávnění. Tato možnost v dosavadní aplikaci zcela chybí.

2 Existující řešení – aplikace ForMic

Podnik, pro který ke aplikace vyvíjena je tvořen spolupracující inkasní agenturou a advokátní kanceláří. Podnik poskytuje služby soudního a mimosoudního vymáhání pro vyšší stovky klientů převážně z řad větších věřitelů, jako jsou banky a jiné úvěrové instituce, pojišťovny, mobilní operátoři a další věřitelé s větším počtem pohledávek. Téměř veškerá činnost týkající se těchto služeb pak probíhá v podnikové aplikaci ForMic, která je produktem společností Redmond Consulting s. r. o. a GREENBERG Consulting s.r.o. [1] Jde o aplikaci s webovým rozhraním, kde kromě správy zákaznických dat, dokumentů, financí a externí i interní komunikace obsahuje také modul Marktime.

2.1 Evidence

Modul Marktime umožňuje uživateli zaevidovat odpracovaný čas s uvedením:

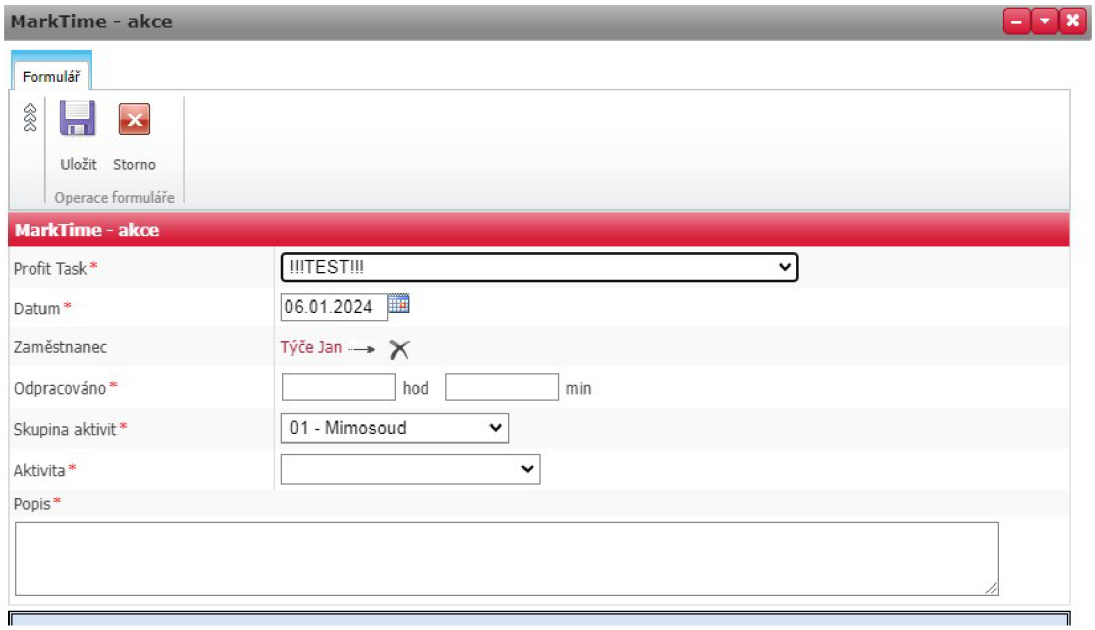
1. pro jakého klienta/na jakém projektu pracoval,
2. datum a kolik času prací strávil,
3. jakou činnost vykonával a to ve dvou úrovních obecnosti, když výběr obou úrovní je povinný,
4. krátkého textového popisu splněného úkolu.

Zápis probíhá dvěma způsoby. Především prostřednictvím standardního webového formuláře, kde uživatel musí vyplnit všechny shora uvedené údaje.

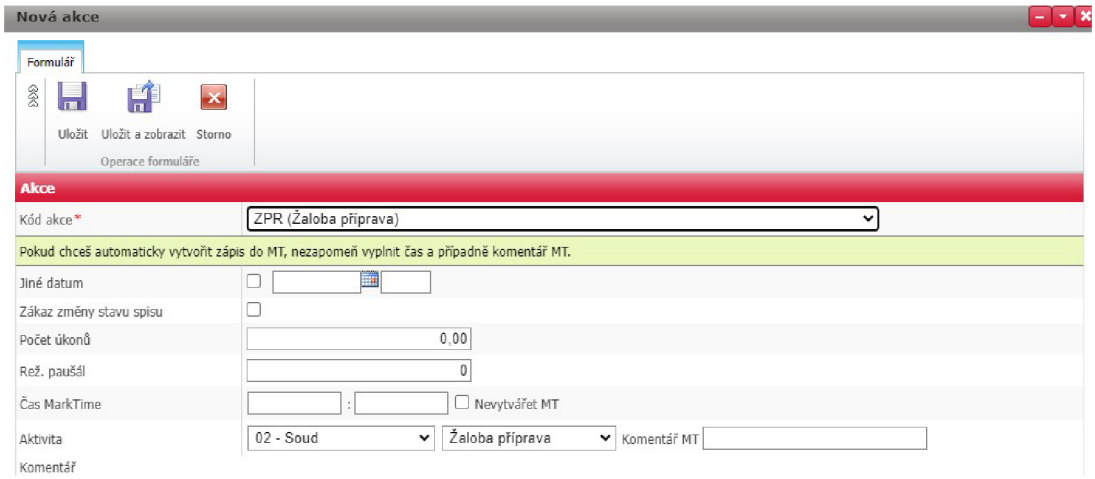
Další možností, která nabízí částečné zjednodušení je tzv. zápis při zadání „Akce“ (například příprava žaloby). Velká část úkolů je totiž spojena se zadáním Akce do příslušného „Spisu“. Uživatel může při zadání konkrétní akce odškrtnout checkbox, po čemž se mu zobrazí pole pro evidenci času připadajícího na práci odvedenou v souvislosti se zadáním Akce. Odpadá tak nutnost vyplnění data a také označení klienta, protože ten je navázán na Spis, do kterého zápis Akce probíhá. V případě některých akcí jsou také automaticky předvyplněny obě úrovně specifikace činnosti a zbývá tedy jen vyplnit čas a textový popis činnosti.

2.2 Zobrazení a přehled

Modul Marktime nabízí dva způsoby zobrazení záznamů o odvedené práci. Prvním je pohled „Marktime sumace“, která primárně zobrazí jednotlivé kalendářní měsíce, u kterých je odvedena celková doba, kterou v těchto měsících uživatel odpracoval. Každý řádek kalendářního měsíce lze rozbalit na seznam jednotlivých dní a u každého dne je opět odvedena celková odpracovaná doba uživatele. Rozbalením řádku dne lze pak zobrazit jednotlivé zápisy a s těmi následně manipulovat. Editace jakéhokoliv zápisu vrací uživatele na úvodní přehled v podobě nerozbalených kalendářních měsíců.



Obrázek 1: ForMic - zápis přes obecný formulář



Obrázek 2: ForMic - zápis při zadání akce

Druhou možností je zobrazení přes pohled „Kalendář“, který ke každému dni vypíše zápisy uživatele s datem tohoto dne.

2.3 Nedostatky existujícího řešení

V podniku pracuji déle než 3 roky a do září 2023, kdy jsem začal k evidenci odvedené práce používat pouze aplikaci vyvinutou v rámci této práce jsem zapsal přibližně 8400 položek záznamů. Považuji se tedy za vhodnou osobu k identifikaci nedostatků existujícího řešení z pozice uživatele. Svá níže uvedená zjištění jsem ověřil rozhovorem s dalšími uživateli, kteří mi mé závěry potvrdili.

Nedostatky se dají rozdělit do dvou zásadních skupin, a to nedostatky týkající se zápisu a nedostatky týkající zobrazení přehledů zapsaných dat.

2.3.1 Nutnost uvést trvání aktivity

Největším problémem pro mě představuje nutnost po skončení aktivity při její evidenci uvést, jak dlouho trvala. Uživatel přitom mívá standardně nižší desítky zápisů denně, pracuje pod časovým tlakem a významná část úkolů je urgentní povahy a nelze ji dopředu plánovat. V průběhu času jsem nepřišel na lepší řešení, než před započetím každé aktivity napsat na papír čas jejího zahájení a její popis. Tato manuální evidence mě (a i další kolegy) ale bohužel sváděla k tomu, že jsme zápisy přepisovali do elektronické evidence až ke konci měsíce, kdy už bohužel dost často nebylo z některých ručních zápisů patrné, co vlastně měly znamenat. Nechtěným důsledkem pak bylo také několik pokažených večerů či víkendových dnů.

V podniku máme zavedeno pravidlo, že zápis musí být proveden nejpozději následující pracovní den po ukončení aktivity. U jednotlivých pracovníků sledujeme i metriku jeho plnění, kde jsem právě z popsaného důvodu (nejen já) pravidelně dosahoval úspěšnosti okolo 20 %.

2.3.2 Způsob výběru klienta a obou typů aktivity

Existující formulář pro zápis záznamu obsahuje 3 html elemtenty <select>. Výběr možností pomocí těchto elementů je vždy na několik kliknutí. Navíc uživatel může z možností vybírat až poté, co se možnosti selectu zobrazí a nemůže další klik plánovat už v okamžiku, kdy vykonává kliky předchozí. Velmi problematickou hodnotím zvlášť volbu klienta/projektu z přibližně 300 možností, když se zde nacházejí skupiny možností se shodnými řetězci na začátku, takže ani pomocí klávesnice nelze hned dospět ke správné volbě. Praxe navíc ukázala, že dochází k chybám uživatelů, když tito omylem zvolí vedlejší možnost s obdobným názvem.

Tento nedostatek je do velké míry odstraněn při zápisu záznamu při zadání Akce, kdy jsou ve standardizovaných případech tyto možnosti předvyplněny. Činnost části pracovníků je ale taková, že zápis při zadání Akce lze využít jen u malé části jejich záznamů.

2.3.3 Další nedostatky

Aplikace Formic zpracovává odeslaný formulář minimálně několik vteřin. Pokud je vytížená databáze, může se tento čas ještě prodloužit. Prodlení s odezvou tak dále zvyšuje negativní vztah pracovníků k evidenci a vede k odkládání evidence nebo k vypuštění evidence některých aktivit a s tím souvisejícím snížením přesnosti evidence.

Během testování existujícího řešení při zpracování této práce jsem také narazil na to, že zapsat záznam za jiného uživatele. Záznam se normálně uloží do databáze, ale přes aplikaci jej nejsem schopen jej následně zobrazit ani smazat.

Takto jsem schopen záznam vytvořit za pracovníka, který již v podniku nepracuje. Platně se uložil dokonce záznam, kde jsem jako pracovníka, který práci odvedl, vybral soudního exekutora, který přirozeně v podniku nikdy nepracoval.

K aplikaci ForMic lze také přistupovat pouze z podnikových PC s předinstalovanou VPN. Přístup z jiného PC není možný. Podnikové telefony mají v rámci MDM přístup pomocí VPN povolen, ale layout aplikace je tvořen pomocí tabulek s prakticky nulovou responzivitou, což přístup i přes podnikový telefon vylučuje. Pokud pracovník vykoná více aktivit bez přístup k podnikovému počítači, nemá je jak průběžně zapsat a je odkázán na zpětný zápis se všemi jeho popsanými nedostatky.

2.3.4 Nedostatky zobrazení

Ačkoliv jsem odpovědný za ekonomické výsledky mně svěřených projektů, existující aplikace mi neumožňuje sledovat nejdůležitější nákladovou složku, kterou je právě odpracovaný čas. Uživatel si může pomocí filtru zobrazit čas, který strávil prací pro konkrétního klienta, nemůže si ale zobrazit záznamy aktivit, které pro tohoto klienta zadali ostatní pracovníci. Ve filtru lze také zvolit vždy jen jeden parametr, nelze tedy například zobrazit sumu času strávenou pro dva vybrané klienty dohromady. Tyto potřeby částečně suplují přednastavené reporty prostřednictvím SQL Server Reporting Services, které jsou ale omezené a jejich vygenerování trvá desítky vteřin.

Výhodou zobrazení stávající aplikace naopak je, že dokáže jednoduše přes pohled Sumace zobrazit časovou řadu odpovídajícího zadaným parametrům za jednotlivé měsíce.

2.4 Aplikace třetích stran

Existuje větší množství aplikací třetích stran [2] [3] [4], které pokrývají i potřebnou funkcionalitu. Vždy ale obsahují další funkcionality zejména pro řízení projektů, které jsou v popisovaném případě nadbytečné. Podnik chce mít také svá data pod svoji výhradní kontrolou a aplikaci provozovat na své stávající infrastruktuře. Uzavření smlouvy s dalším zpracovatelem osobních údajů navíc podléhá předchozímu schválení některých klientů a je proto administrativně náročné. Vedení podniku se proto rozhodlo aplikaci třetí osoby v tomto případě nevyužít.

3 Uživatelské požadavky na vyvíjenou aplikaci

Je nutné, aby vyvíjená aplikace obsahovala základní funkcionality existujícího řešení. Základní uživatelské požadavky jsou proto následující:

1. Základním požadavkem je tedy vyvinout aplikaci, která stejně jako existující řešení umožní uživateli zápis, zobrazení, editaci a smazání záznamů odpracovaného času s uvedením data aktivity, vazby na projekt/klienta [1], vazby na skupinu aktivit a konkrétní aktivity a textového popisu záznamu.

2. Správa uživatelů bude vyhrazena pouze uživatelům s oprávněním Supervizor, kteří také mohou dalšímu uživateli udělit práva supervizora.

3. Z definovaných nedostatků existujícího řešení jsem pak formuloval tyto uživatelské požadavky:

4. Aplikace umožní zápis takovým způsobem, aby nebylo nutné zpětně dohledávat čas, kdy uživatel s aktivitou započal.

5. Proces zápisu bude maximálně zjednodušen tak, aby se minimalizoval počet operací nutných k vyplnění formuláře. Cílem je odstranit nutnost použití všech tří html atributů <select> pro většinu případů užití. Formulář bude v co nejmenší míře využívat pohyblivé a skryté prvky, aby si uživatel alespoň po část procesu jeho vyplňování mohl plánovat cestu myší dopředu.

6. Aplikace bude obsahovat systém oprávnění pro zobrazení záznamů zadaných jinými uživateli. Uživateli bude možné přiřadit 0 až N Portfolií. Uživatel pak získá právo k zobrazení, editaci i mazání záznamů, které jsou na tato Portfolia přiřazené. Tato funkce je určena převážně pro nižší management podniku.

7. Záznamy bude možné filtrovat podle data aktivity, uživatele a Portfolia. Filtr bude umožňovat zadání více hodnot od každého parametru. Spolu s vyfiltrovanými záznamy se zobrazí přehled odpovídající parametrům filtru. Přehled bude obsahovat sumu odpracovaného času s rozpadem na jednotlivá Portfolia, uživatele a skupiny aktivit.

8. Půjde o webovou aplikaci s responzivním designem. Referenčním pro desktopové zobrazení je operační systém Windows 10, prohlížeč Chrome, velikost displaye 1920 x 1080 a nastavení velikosti textu na 100 % v ideálním

[1] V dalším textu budu používat pro projekt/klienta termín „Portfolio“. Jde o termín běžně používaný v podniku, který vznikl z toho, že pro klienta je obvykle spravováno portfolio pohledávek. S tímto termínem pracuje uživatelské rozhraní aplikace a třídu `Portfolio` obsahuje i její zdrojový kód a tabulku `portfolios` obsahuje i databáze aplikace.

případně i s velikostí textu na 125 %. [2] Aplikace bude přizpůsobena mobilním o minimálním rozlišení 337 x 667. [3] Všechny funkce aplikace budou dostupné bez použití JavaScriptu[4]

9. Aplikace bude napsána v programovacím jazyce PHP a pro uložení dat bude použita databáze MySQL[5].

10. Aplikace bude připravena na propojení s podnikovou aplikací.

Mezi uživatelskými požadavky nakonec chybí požadavek obsažený v zadání práce: „Uživatel si bude moci nastavit upozornění při dosažení jím požadovaných parametrů.“ Po dohodě s managementem podniku jeho funkčnost zcela nahradí generované přehledy odpovídající parametrům filtru.

[2] Standardní parametry podnikových notebooků bez připojeného externího displaye

[3] Jde o rozlišení telefonu iPhone SE, který někteří pracovníci používají. Není používán telefon s menším rozlišením.

[4] Tento požadavek jsem přidal čistě z osobních důvodů, kdy na mobilním telefonu mám JavaScript ve webovém prohlížeči standardně vypnutý.

[5] Požadavek IT oddělení podniku. Jde o standardní parametry podnikových webových aplikací používaných mimo systém ForMic

4 Návrh řešení

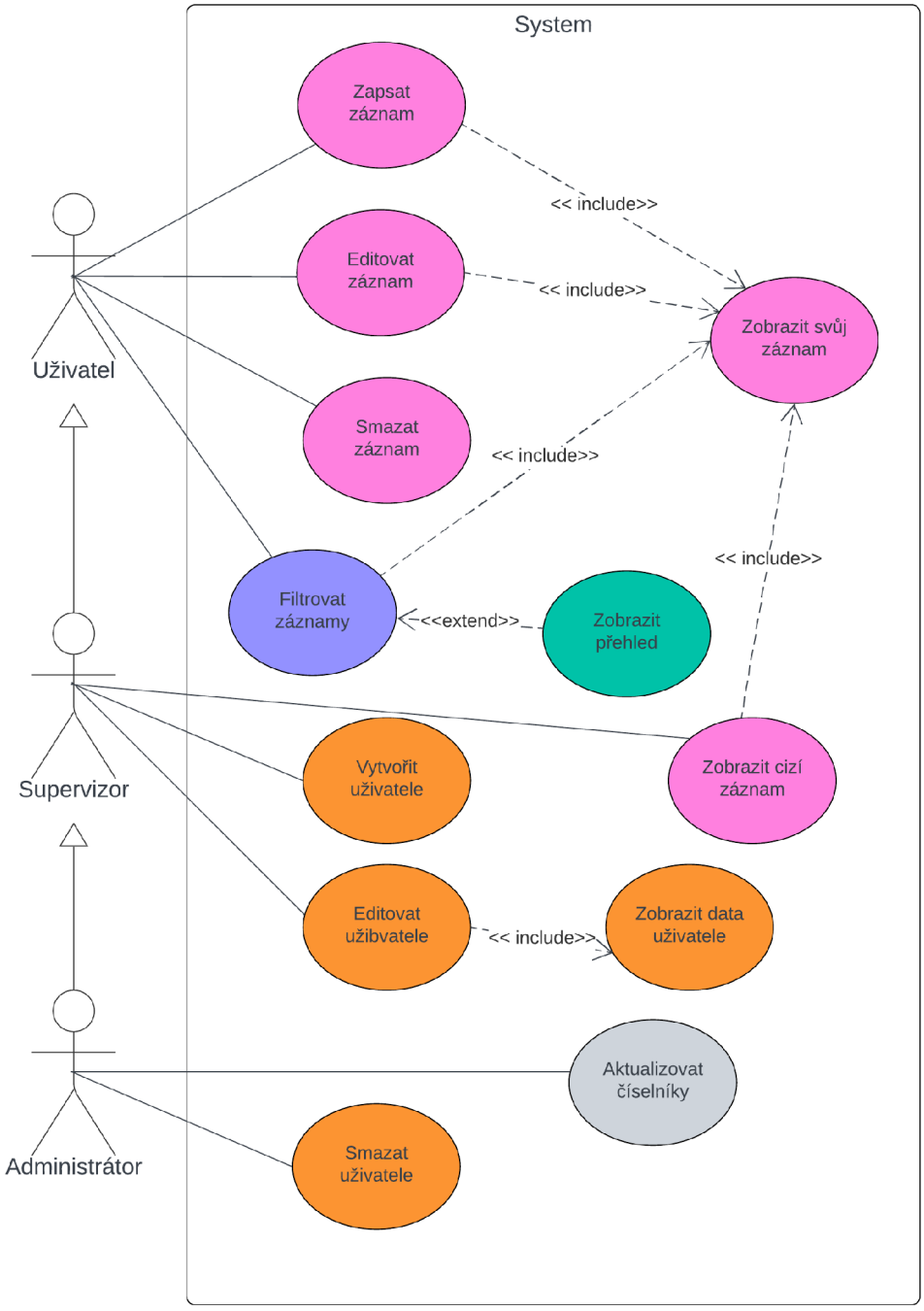
Tato část aplikace zobrazuje návrh řešení zobrazený prostřednictvím Diagramu případů užití a diagramu tříd.

4.1 Diagram případů užití

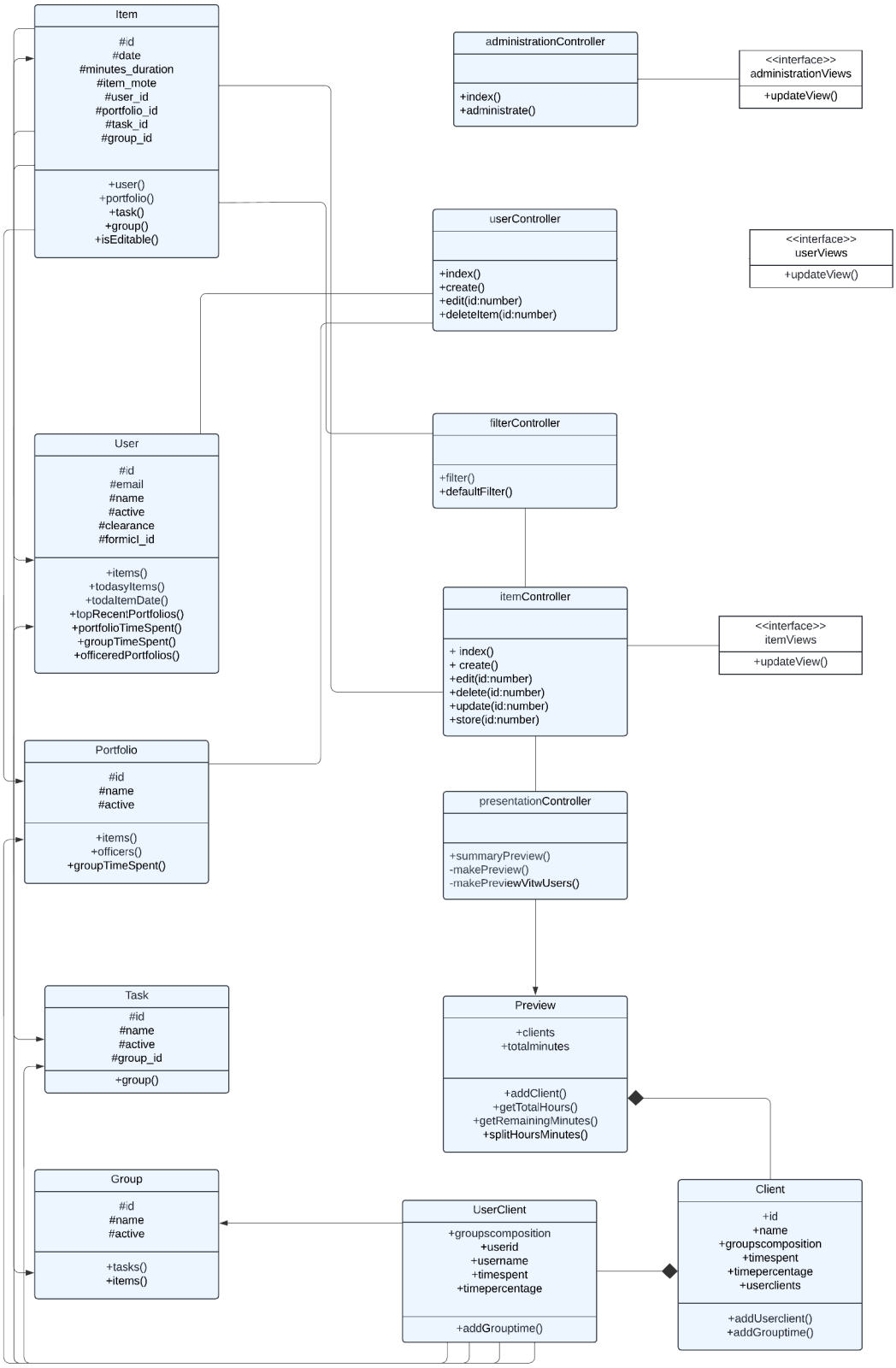
Základní funkce aplikace jsou znázorněny diagramem případů užití. Diagram přehledně znázorňuje navrhované tři úrovně oprávnění uživatelů aplikace spolu s činnostmi, které mohou uživatelé v těchto rolích vykonávat. Názvy případů užití jsou podle mě dostatečně určité na to, aby vyžadovaly další popis v této práci.

4.2 Diagram tříd

Návrh struktury aplikace je znázorněn v níže uvedeném diagramu tříd.



Obrázek 3: Diagram případů užití



Obrázek 4: Diagram Tříd

5 Použité technologie

5.1 Použití frameworku

Jak vyplynulo z požadavku číslo 9 uvedeného v předchozí kapitole, aplikace musí být sepsána v programovacím jazyce PHP. Na počátku jsem tedy stál před volbou, zda využít pouze nástrojů tohoto programovacího jazyka, nebo zda využít nějakého frameworku, který PHP využívá.

Ztotožňuji se s názorem B. Campbella, který uvádí, že bez využití frameworku je sice možné dosáhnout výsledku, který bude přesněji odpovídat představám vývojáře, vyváženo je to ale tím, že vývoj jednotlivých prvků zabere větší množství času, přičemž u rozsáhlejších aplikací by mohlo jít i o jednotky let. Naopak využití frameworku sice vyžaduje kromě znalostí programovacího jazyka také znalost komponent vybraného frameworku, pozvolná učící křivka je ale vyvážena úsporou času vývoje (od určitého rozsahu aplikace). Z hlediska bezpečnosti předchází využití frameworku chybám způsobeným vývojářem, nicméně vývojář musí spoléhat na to, že samotný framework je z hlediska bezpečnosti koncipován správně [5].

Před započetím vývoje této aplikace jsem v PHP nikdy neprogramoval a oproti jazyku C#, mi přišel na první pohled zbytečně komplikovaný a vnitřně nesourodý. Byl jsem tedy v pozici, kdy jsem postrádal pokročilejší znalosti tohoto jazyka, tedy jeden ze základních důvodů pro nepoužití frameworku. Z minulosti jsem pak měl pozitivní zkušenost s programováním za použití frameworku .NET Core. Navíc jsem vyhodnotil svoje znalosti vývoje webových aplikací jako ne příliš pokročilé a riziko, že bych při vývoji aplikace zanedbal nějakou důležitou oblast, jako nezanedbatelné. Ze všech těchto důvodů jsem proto dospěl k jednoznačnému závěru, že při vývoji aplikace bude velmi vhodné, abych využil nějakého frameworku, který je postaven na jazyce PHP.

Z hlediska podniku nebyla tato volba podstatná, protože dosud používané aplikace jsou vytvořeny pouze v PHP. Kolegou z IT oddělení mi byl doporučen framework Nette. Provedl jsem rešerši dalších doporučovaných frameworků [6] a zjistil jsem, že z doporučovaných řešení jsou celosvětově nejpoužívanějšími frameworky založenými na PHP Laravel a CodeIgniter. Nette výrazně zaostává. [7][8][9].

Název	Počet firemních nasazení	Lokalizace
Nette	12756	62 % ČR, 6% SR
Laravel	127901	16 % USA, po 4% Velká Británie, Indie, Brazílie
CodeIgniter	138559	17 % USA, 14 % Indie

Tabulka 1: Přehled využití PHP frameworků

Uživatelská základna Nette je tak přibližně desetkrát menší, než dvou nej-

používanějších frameworků a je opravdu výrazně omezena na Českou republiku. Rozhodl jsem se proto využít jednu ze dvou zbývajících variant a osvojit si framework, který není závislý na situaci v konkrétním státě a pro své 10 násobně větší rozšíření má podle mého názoru mnohem lepší předpoklady pro svůj budoucí vývoj. Při rozhodování, jestli použít CodeIgniter nebo Laravel jsem se přiklonil k Laravelu, protože tento framework oproti CodeIgniteru nabízí předpřipravené nástroje pro autentizaci a autorizaci a také nástroje pro testování pomocí PHPUnit, a to i za cenu, že učící křivka tohoto frameworku je pozvolnější [10].

5.2 Laravel

Aplikace je vytvořena za využití framevorku Laravel verze 9.52.16. Započal jsem s vývojem, když byla aktuální verze 8.x, ale vzhledem k době, která uplynula od uzavření mého studia, do obhajoby této práce jsem byl nucen v průběhu upgradovat projekt na verzi 9. Tato verze má zajištěné bezpečnostní aktualizace do února 2024. Po uplynutí tohoto data tedy bude nutné aplikaci upgradovat na verzi 10.

5.2.1 Architektura MVC

Laravel je založen na architektuře Model – View – Controller, která umožňuje aplikaci rozdělit na tři základní skupiny komponent, které lze samostatně spravovat, aniž by došlo k ovlivnění komponent ostatních.

Model obsahuje data aplikace a aplikační logiku a uchovává v sobě data či zajišťuje jejich vyzvednutí a uložení v databázi a má na starosti veškeré výpočty nebo vyhodnocování pravdivosti [11].

Controller [6] přijímá žádosti od uživatelského rozhraní, přiřazuje zpracování dat příslušnému modelu, který tato data vrátí pohledu, aby je uživateli zobrazil.

Pohled potom představuje uživatelské rozhraní aplikace. Zobrazuje tedy uživateli data a přijímá od uživatele požadavky, které předává formou žádostí k vyřízení controlleru. Získaná data si pak od modelu převezme a zobrazí je uživateli

5.2.2 Komponenty Laravelu

V následujícím textu stručně přiblížím základní komponenty Laravelu v jednotlivých částech architektury MVC. Zabývám se pouze těmi komponentami které jsem při vývoji aplikace ve větší míře využil.

5.2.2.1 Eloquent ORM Základní komponentou Laravelu na modelové vrstvě je nástroj pro objektově relační mapování (object-realtionalship-mapper) nazvaný Eloquent, který umožňuje manipulaci s daty uloženými v databázi pomocí jednotlivých modelových tříd a jejich metod a atributů. Atributy modelu

[6] Tento termín záměrně nepřekládám

stačí definovat v třídě `Migration`, která slouží pro vytvoření nebo změnu související tabulky v databázi. Pokud programátor dodrží konvence předpokládané frameworkem, získá navíc automatická integritní omezení pro tvorbu id nebo cizí klíče [12]. [7]

Na modelové vrstvě pak jsou definovány metody pro vztahy jednotlivých modelů. [8] Pokud by programátor z nějakého důvodu nechtěl dodržet konvenci pojmenování modelů a jejich atributů a tabulek a sloupců v databázi, nemusí, ale je pak nuntné tyto metody přetěžovat.

5.2.2.2 Query Builder Query Builder představuje rozhraní pro vytváření a spouštění databázových dotazů. Lze jej použít k provádění většiny databázových operací v aplikaci a je nezávislý na databázových systémech, které Laravel podporuje. Využívá PHP Data Objects a předchází tak útokům typu SQL injection [13].

Instanci třídy `Builder` lze vytvořit a následně předávat mezi několika různými objekty a jejich metodami. Do databáze lze pak odeslat až dotaz z finální verze instance po všech úpravách. Nenahrávají se tak do paměti žádná nadbytečná data a zpracování dat je ponecháno databázovému systému, kde lze očekávat jeho větší efektivitu.

Obě uvedené komponenty umožňují aplikaci úplnou abstrakci od databázové vrstvy. V ideálním případě postačí změnit v konfiguračním souboru `.env`, který je umístěn v kořenovém adresáři aplikace, typ databáze, přístupové údaje do databáze a zadat příkaz k provedení migrace a případnému seedingu databáze a aplikace může být provozována s napojením na tuto novou databázi. Aplikaci pracující s databází je tak možné vytvořit a provozovat bez jediného řádku kódu v jazyce SQL.

5.2.2.3 Blade šablony V pohledové části aplikace je důležitou komponentou Blade Templating Engine. [14] Šablony Blade obsahují množství direktiv, které může programátor využít namísto psaní kódu v PHP. Souhlasím s názorem, že direktivy jsou spíše syntaktickým cukrem [15] [9], ale z mé osobní zkušenosti opravdu psaní kódu usnadňují. Ne vždy ve všech případech ale direktivy fungují jak mají. Měl jsem například problém v případě vnořených cyklů s přístupem k údajům z nadřazeného cyklu, pokud byl vnořený cyklus součástí html elementu a musel jsem pro tento případ nahradit direktivu kódem v čistém PHP, abych se k údajům dostal.

[7] Laravel například předpokládá, že každému modelu patří tabulka se stejným názvem, jen začínající malým písmenem. Sloupec s názvem jiné tabulky s přívlastkem „_id“ pak představuje cizí klíč pro záznamy v této tabulce.

[8] např. hasOne, belongsToMany a dokonce i manyToMany, předpokládající využití pivotové tabulky (opět se standardizovaným názvem model1_model2 obsahující 2 sloupce s id instancí obou modelů)

[9] Při generování pohledu se stejně překládají nejdříve do PHP a při nasazení aplikace na produkci lze tento překlad uložit do vyrovnávací paměti

Blade Templates obsahují také užitečný nástroj proti Cross Site Script (XSS) útokům. Jakýkoliv kód v PHP se totiž namísto tagu <?php...?> uvádí ve dvojitých složených závorkách. Před překladem do PHP projde takto označený kód sanitací a PHP funkce `htmlspecialchars` z něj odstraní potenciálně škodlivé znaky.

5.2.2.4 Směrování a Middleware

V rámci oblasti controllerů jsem využíval především směřování html žádostí k příslušným controllerům a autentizační a autorizační komponenty Laravelu.

Middleware poskytuje vhodný mechanismus pro kontrolu a filtrování požadavků HTTP vstupujících do aplikace. Laravel například obsahuje middleware, který ověřuje, zda je uživatel aplikace přihlášen. Pokud uživatel není přihlášen, middleware přesměruje uživatele na přihlašovací obrazovku aplikace. Pokud je však uživatel ověřen, middleware umožní požadavku pokračovat dále do aplikace [16].

Vlastní middleware jsem aplikoval pro kontrolu, jestli je uživatel aktivní. Dostal jsem se totiž do situace kdy jsem zjistil, že potřebuji okamžitě zamezit uživateli přístup do aplikace z důvodu ukončení spolupráce. Uživatel ale mohl být přímo přihlášen nebo mohl mít uložené dlouhodobé přihlášení na některém ve svých zařízeních a do jeho vypršení nebylo jednoduše možné mu ve vstupu do aplikace zamezit. Smazat jsem jej nemohl. Musel jsem tedy vytvořit middleware, který u každého požadavku ověřuje, jestli je uživatel, který jej odesílá aktivní, a pokud zjistí, že není, okamžitě jej odhlásí.

5.2.2.5 Brány a Politiky

Kromě vestavěných služeb ověřování poskytuje Laravel také jednoduchý způsob autorizace akcí uživatele vůči danému prostředku. Například i když je uživatel ověřen, nemusí být oprávněn aktualizovat nebo mazat určité modely nebo záznamy v databázi s nimi spojené. Laravel poskytuje dva základní způsoby autorizace akcí: brány a politiky. Brány poskytují jednoduchý přístup k autorizaci, zatímco politiky, podobně jako Controllery, seskupují logiku kolem určitého modelu [17].

Politiku jsem implementoval pro správu přístupů k instancím modelu `Item` a modelu `User`, protože bylo nutné u každé metody upravit rozsah oprávnění pro jednotlivé uživatele jak vzhledem k jejich roli, tak vzhledem k jejich atributu `officeredPortfolios`. Bránu jsem aplikoval, například když jsem při generování formuláře filtru záznamů v metodě `itemController->makeUsers()` musel posoudit, zda uživatel bude moci filtrovat položky podle ostatních uživatelů. Pokud by toto oprávnění neměl, neměl by se mu ani nabídnout jejich seznam. Jednak by neměl zjistit, kdo ostatní uživatelé jsou, a také by ani neměl mít možnost formulář chybně odeslat.

5.2.2.6 Validace

Laravel nabízí k využití předpřipravené validační nástroje, které umožňují kontrolu hodnot zadaných uživatelem podle určených parametrů. Nedojde tak k uložení hodnot, které by v budoucnu mohly způsobit chybu při

dalším běhu aplikace. Ve vyvíjené aplikaci jsou všechny vstupy uživatele validovány na straně serveru a není tedy možné je obejít například úpravou kódu zobrazené webové stránky.

5.2.3 Balíčky

Spolu ze základními komponentami Laravelu jsem použil následující balíčky:

5.2.3.1 Breeze 1.19.2 Laravel Breeze je minimalistická implementace všech ověřovacích funkcí Laravelu, včetně přihlášení, registrace, obnovení hesla, ověření e-mailu a potvrzení hesla [18]. Balíček obsahuje sadu Controllerů a Pohledů, ze kterých jsem vybral objekty, které v aplikaci při autentizaci potřebuji. Součástí balíčku je i soubor feature testů jeho základních funkcí pro testování pomocí PHPUnit.

5.2.3.2 bfinlay/laravel-excel-seeder 3.3.4 Balíček Laravel excel seeder [19] umožňuje naplnění [10] databáze na základě excelovských tabulek uložených do určeného adresáře aplikace. V aplikaci jsem jej použil proto, aby aktualizaci instancí modelu tříd `Portfolio, Group a Task` mohl provádět kvalifikovaný uživatel bez nutnosti přímého přístupu do databáze. Balíček po zavolání třídy odvozené od třídy `SpreadsheetSeeder` hledá soubory `*.xlsx` v zadaném umístění. V těchto souborech hledá listy, jejichž název odpovídá názvům tabulek v databázi a hodnotami ve sloupcích s názvem, které odpovídají názvu atributu příslušné tabulky, tyto atributy vyplní. Opakované spuštění s týmiž daty nezpůsobí komplikace. Pro tento postup jsem se rozhodl z důvodu, že přednastavené excelovské tabulky s daty z podnikové aplikace je možné získat kdykoliv nezávisle na součinnosti kolegů z IT přes SQL Server Reporting Services a není tak s aktualizací nutné čekat, až bude volná kapacita pro synchronizaci prostřednictvím databáze.

5.2.3.3 laravelcollective/html 6.4.1 Laravel Collective HTML [20] je balíček pro zjednodušené generování webových formulářů. Osobně opravdu raději vytvářím základní formuláře a jejich prvky za jeho použití. Balíček obsahuje vestavěnou ochranu proti Cross site request forgery útokům ve formě vkládání CSRF tokenu. Na straně frameworku pak dochází k jeho ověřování. Tento balíček bude muset být do budoucna v aplikaci nahrazen jinou alternativou, protože dle sdělení jeho autora, byl ukončen jeho vývoj a nebude možné jej od verze Laravel 11 nainstalovat. Tuto informaci jsem při zahájení vývoje aplikace neměl, jinak bych zvolil jiné řešení. Laravel již nyní jako alternativu propaguje balíček spatie/laravel-html [21]. Již na první pohled je ale zřejmé, že, ačkoliv je balíček na první pohled postaven na podobném základě, bude nutné kód minimálně odladit a zrevidovat. Nestandardní komplikovanější prvky formulářů to neovlivní, protože jsou stejně vytvořeny bez použití nástrojů tohoto balíčku.

[10] seeding

5.3 Tailwind

Podstatnou součástí každé aplikace je samozřejmě také způsob prezentace dat uživateli. U webových aplikací k tomuto účelu slouží HTML, CSS a také JavaScript. Vytváření HTML webových stránek ve vyvíjené aplikaci je technologicky řešeno pomocí šablon Blade, které jsou součástí Laravelu. JavaScript, v souladu s uživatelskými požadavky, se v kódu webových stránek nevyskytuje.

Pokud jde o rozhodnutí o způsobu práce s CSS, stál jsem před obdobným rozhodnutím, jako při volbě, zda při vývoji aplikace využít framework, nebo jen samotný programovací jazyk, přičemž i v tomto případě přiměřeně platí argumenty, kterými jsem se zabýval výše. Vzhledem ke skutečnosti, že je pro vyvíjenou aplikace zásadní, aby zadávání záznamů bylo maximálně optimalizované rozhodl jsem se nejít cestou použití CSS frameworku Bootstrap a jeho předpřipravených komponent, ale pro grafickou úpravu webových stránek použít framework Tailwind.

Tailwind je vhodnější pro projekty, které vyžadují větší individualizaci jednotlivých komponent [22] [23], čehož je využito zejména v prvcích formuláře, které nahrazují původní elementy <select> a také při koncipování responzivního designu aplikace.

Ačkoliv na první pohled může zdrojový kód webové stránky s Tailwind CCS vypadat jako by byly použity vložené styly CSS, není tomu tak [24]. Při použití vložených stylů je každá hodnota magickým číslem, kdežto při použití Tailwind CSS jde o výběr z nízkoúrovňových předpřipravených tříd z přednastaveného designového systému, které zajišťují že uživatelské rozhraní bude vypadat jednotně. Další velkou výhodou je, že úpravou jedné části kódu obvykle nedojde k narušení na jiném místě, což při úpravě globálních CSS tříd často vyloučit nelze. S rostoucí velikostí projektu toto riziko rychle roste [25].

6 Datová základna aplikace

Aplikace potřebuje ke svému fungování trvale ukládat data. Pro jejich uložení byl uživatelskými požadavky určeno uložení do databáze MySQL. Jak jsem již uvedl v sekci Použité technologie, objektově relační mapování Laravelu umožňuje přenechat odpovědnost za vytvoření struktury databáze frameworku. Programátor pouze vytvoří instance třídy `Migration` do kterých uvede své požadavky na změny v databázi. Při spuštění první migrace dojde i k vytvoření databáze aplikace, pokud není databáze tohoto jména na serveru již vytvořena.

Framework zvolí správný datový typ jednotlivých atributů a při dodržení konvencí pojmenování sloupců vytvoří defaultně veškerá potřební omezení pro cizí klíče a vytvoří také indexy na primární a sekundární klíče. Manuálně jsem přidával pouze indexy na `date` v tabulce `items` a `name` v tabulkách `users` a `portfolios`, podle kterých probíhá v aplikaci řazení podle abecedy před zobrazením dat uživateli. Vytvoření tabulek v databázi při nasazení aplikace je tak otázkou zadání jednoho pokynu, což jsem velmi ocenil, když jsem aplikaci několikrát nasazoval na různé servery.

Pro přehledné zobrazení struktury datové základny aplikace uvádím Model vztahu mezi entitami [11] vytvořený aplikací MySQL Workbench. Z modelu jsem vypustil tabulky vytvořené frameworkem pro autentizační část aplikace, které nemají s datovým modelem žádný vztah a nejsou mým autorským dílem.

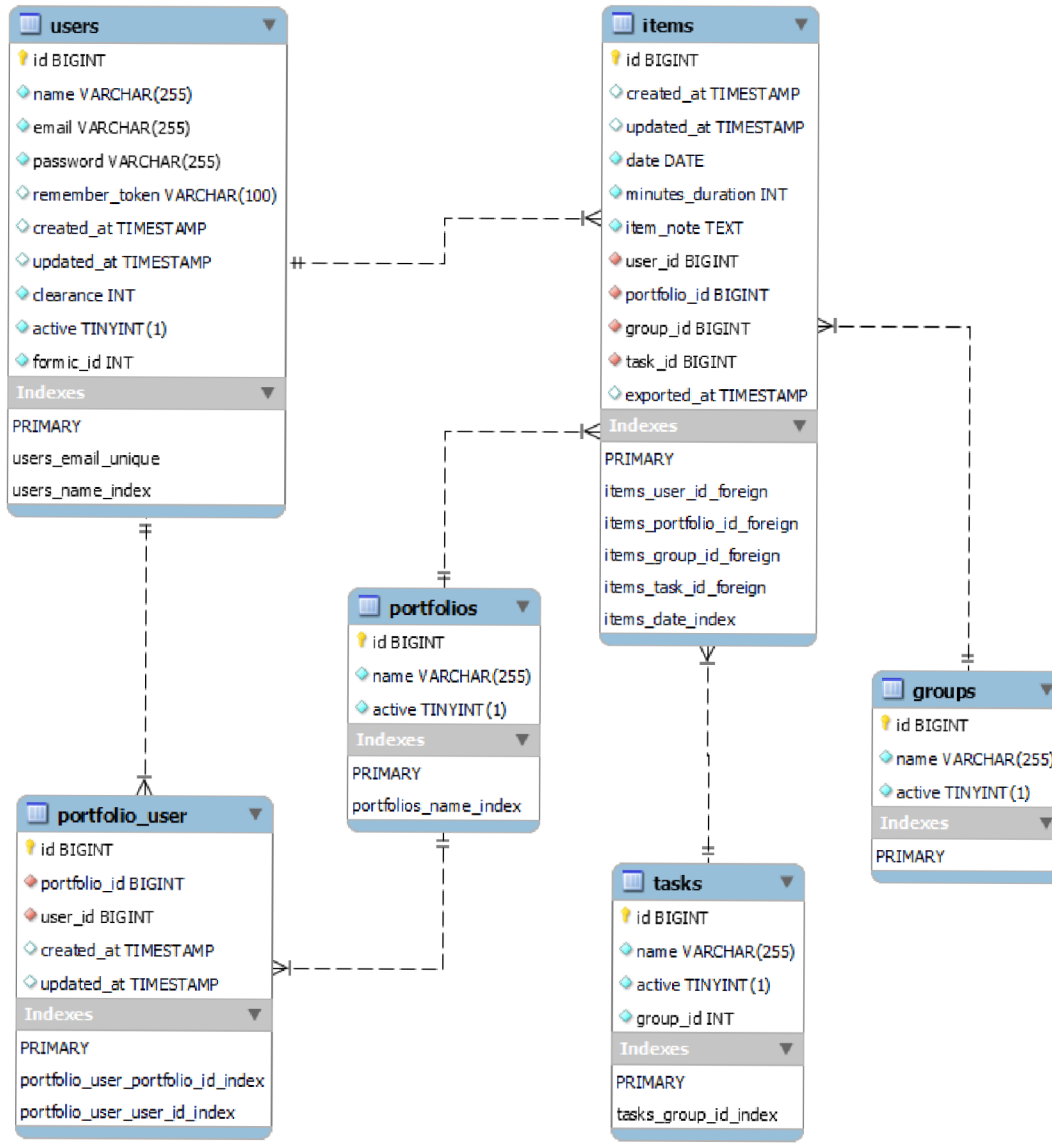
6.1 Ukládání dotazů do vyrovnávací paměti

Při testování databáze popsaném v části Testování jsem zjistil, že pokud uživatel nechá vytvořit `Preview` z milionu položek, je dotaz velmi náročný. [12].

Identifikoval jsem proto tyto nejnáročnější dotazy, u kterých může docházet k opakování a nastavil jsem ukládání jejich výsledku do vyrovnávací paměti. Předně jde o generování `Preview` a seznamu uživatelů a portfolií do filtru dodatečného filtrování již vyfiltrovaných hodnot. Tyto hodnoty se ukládají do vyrovnávací paměti na 10 minut. Dále jde o seznam nejčastějších portfolií uživatele generovaných do formuláře pro zápis záznamu, který je ve vyrovnávací paměti uložen po dobu 1 hodiny. Tento prvek formuláře vychází z dlouhodobého vyhodnocení zápisů uživatele a zpoždění jeho aktualizace nepřináší uživateli žádnou újmu. Naopak na straně databáze může docházet k úspoře 10 až 20 strukturovaných požadavků za každého uživatele denně.

[11] Entity-relationship Diagram

[12] Pomiňme nyní skutečnost, že přehled za takto dlouhou dobu bude bude uživateli k ničemu, protože může odpovídat záznamům všech pracovníků za velmi dlouhou dobu v horizontu 10 nebo 15 let. Zastávám názor, že pokud bude aplikace alespoň částečně připravena na nesmyslné požadavky, může to předejít komplikacím, které jsem při jejím vývoji nepředvídal



Obrázek 5: Model vztahu mezi entitami

7 Testování

Testování je nezbytnou součástí procesu vývoje software. Bez jeho provedení nelze vyloučit riziko, že aplikace při nasazení na produkci obsahuje chyby a nesplňuje tak uživatelské požadavky.

Dokumentace Laravelu přímo uvádí, že jde o framework, který je zaměřený na testování [26]. Podpora pro testování pomocí nástroje PHPUnit je zahrnuta již v základní instalaci frameworku. Framework také obsahuje pomocné nástroje a metody, které testování usnadňují.

Laravel podporuje oba způsoby testování, tedy jak jednotkové (unit tests), tak testy funkcí (feature tests). Jednotkové testy se zaměřují na malou izolovanou část kódu. Obvykle přímo na jednu metodu. Při provádění těchto testů nespouští Laravel aplikaci, a proto nemají přístup k databázi aplikace ani k dalším službám frameworku.

Testy funkcí naopak mohou testovat větší a ucelenější část kódu, včetně toho, jak spolu několik objektů interaguje, nebo dokonce vyřízení celého HTTP požadavku. Dokumentace Laravelu doporučuje, aby převážná většina testů byla funkčními testy. Tyto typy testů poskytují největší jistotu, že aplikace jako celek funguje tak, jak má [26]. V případě funkčních testů dochází ke spuštění aplikace i k zápisům do databáze. Lze tedy testovat chování aplikace v podmínkách blízkých jejímu nasazení.

Ve vyvíjené aplikaci jsem se rozhodl využít pouze funkční testy a to především z důvodu, že metod, které by nějakým způsobem nepracovaly s HTTP požadavky nebo s databází se v kódu mnoho nenachází. Primárním úkolem aplikace ostatně je ukládat data do databáze a zase z databáze data uživateli zobrazovat.

Dokumentace doporučuje započít každý funkční test s prázdnou databází, která je před spuštěním vlastního testu naplněna pouze daty podle parametrů testovací metody. Nedojde tak k ovlivnění výsledků prováděného testu průběhem testů předchozích. Objektově relační mapování Eloquent, umožňuje pro každý model vytvořit objekt odvozený od třídy `Factory` který obsahuje parametry, ve kterých se mají pohybovat hodnoty atributů modelu. Následně lze v rámci testování vytvořit libovolný počet individualizovaných instancí modelu a s nimi testování provádět. V případě své aplikace jsem `Factory` vytvořil pro modely `Item` a `User`.

Aplikace obsahuje 27 testů, jejichž seznam uvádím v příloze A. Sedm z těchto testů bylo vytvořeny automaticky při instalaci autentizačního balíčku Laravel Breeze a testují jeho funkce. Názvy všech vytvořených testů přitom podle doporučení dokumentace Laravelu odpovídají co nejpřesněji funkcím, které testují.

Rozsah testování bývá určován podílem kódu, jehož funkčnost testování ověřuje. Nejde o dokonalé měřítko kvality testů, ale nabízí rozumnou, objektivní, standardní metriku s použitelnými údaji univerzálně pro všechny produkty. Pokrytí kódu v rozsahu 60 % bývá označováno jako akceptovatelné, ve výši 75 % jako výborné a v rozsahu 90 % jako výjimečné. [27]

Rozsah testování pro aplikaci vyvíjenou v Laravelu lze zjistit pomocí pří-

kazu `php artisan test --coverage`. Laravel do výpočtu pokrytí zahrnuje všechny Controllery, Modely, Middleware, Politiky, Requests a Service Providers a také případně dodatečné třídy vytvořené vývojářem. Takto vypočtená hodnota u vyvíjené aplikace dosahuje 88 %[13], což je s ohledem na výše uvedené doporučení velmi dobé pokrytí. U většiny tříd s menším než 100% pokrytím kódu jde přitom o třídy, které byly vytvořeny při instalaci frameworku a do jejich funkčnosti jsem nijak nezasáhl. Do výpočtu vstupují také jen schémata Pohledů a nikoliv již samostatné Pohledy, i když fakticky jejich testování probíhá, protože jsou vraceny testovanými metodami Controllerů. Hodnota pokrytí testy mnou vytvořeného kódu by tak byla při jejich zahrnutí ještě vyšší.

Přínos takto vytvořených testů hodnotím jako velmi vysoký. Po každé změně kódu nebo při aktualizaci na vyšší verzi frameworku nebo některé jeho komponenty je možné během 30 vteřin, které provedení testů na mém starším počítači trvá, ověřit, zda zůstala funkčnost testy pokrytého kódu zachována.

Ačkoliv vytvořené testy na produkční prostředí nepatří, protože dojde ke ztrátě již uložených dat v databázi, jde o důležitou součást vývoje aplikace, a proto aplikaci pro účely této práce nasazuji ve vývojovém nastavení a je možné testy spustit pokynem `php artisan test` zadaném v adresáři aplikace.

Následně je nutné provést novou migraci databáze pokynem `php artisan migrate:fresh` a `php artisan db:seed`, aby došlo opětovně k vytvoření uživatelských účtů zřízených pro testování.

7.0.1 Testování databáze

Popsané funkční testy se omezují pouze vytvoření několik instancí modelu `Item` a jednoho nebo dva uživatele a jejich uložení do databáze. Je to především z důvodu, aby testy proběhly v rozumné době. Vzhledem k tomu, že rychlá odezva aplikace je důležitým uživatelským požadavkem, testoval jsem navíc manuálně funkčnost aplikace s databází naplněnou v řádově větším rozsahu.

Pomocí Factories popsaných výše jsem do databáze uložil milion instancí [14] `Item` a rozdělil jsem je mezi 100 uživatelů. Následně jsem zkoušel vygenerovat pohled `items.index`, který obsahuje data z třídy `Preview`. Jde o přehledovou část vypočítanou z rozsahu odpracovaného času určeného podle parametrů filtru. Vytvoření `Preview` s rozdělením podle uživatelů obsahuje 3 vnořené cykly, proto je výpočetně náročné. Při vytvoření `Preview` pro 5 nejvíce časově náročných portfolií podle výše uvedených parametrů dochází k odeslání 6468 dotazů do databáze a jejich vyřízení prvotně zabralo na mém počítači více než 2 minuty. V databázi přitom byly obsaženy již všechny potřebné indexy.

Prošel jsem tedy všechny časově náročnější dotazy a odbouráním duplicitních požadavků se mi podařilo uspořit přibližně 500 dotazů do databáze a 15 % času potřebného na vyřízení dotazu. Dalších 15 % času se mi podařilo odstranit když

[13] Po dodatečném doplnění administrationControlleru, na který chybí testy jde o 85 %

[14] Jde o několikanásobek řádků, než kolik obsahuje tabulka zápisů za 8 let fungování systému evidence v podniku. Jen vytvoření těchto dat trvalo několik hodin.

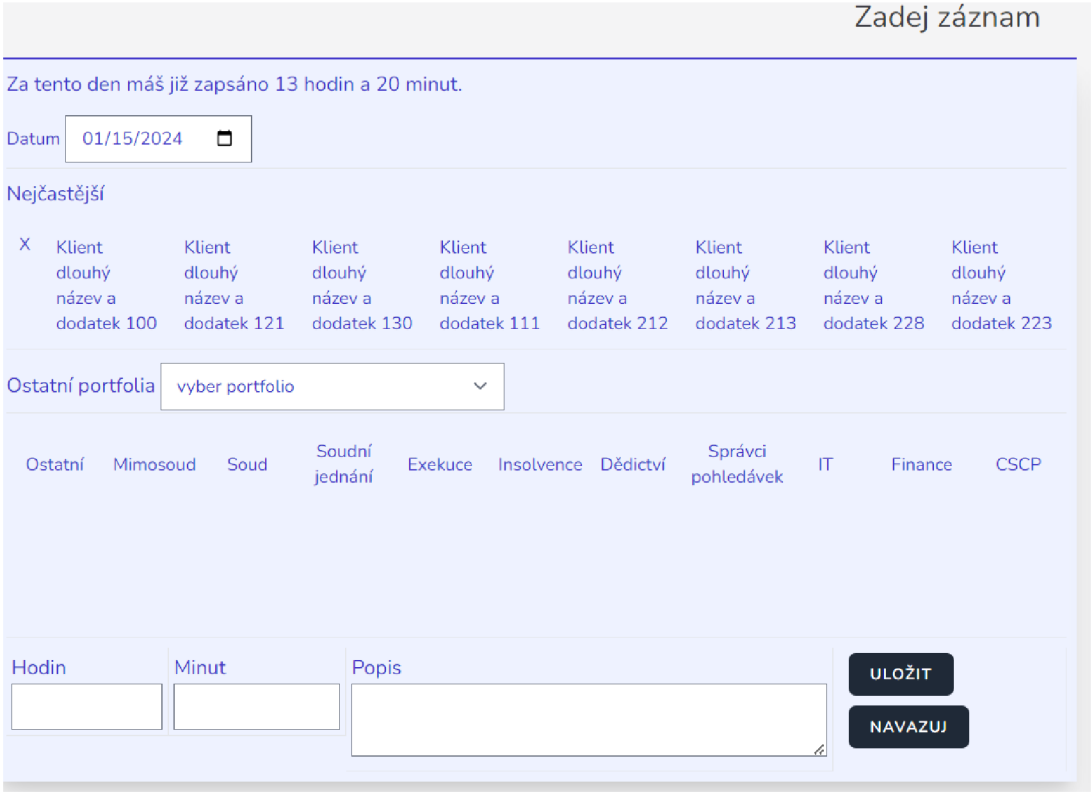
jsem dotaz na distinct `Portfolios` a distinct `Users` z vyfiltrovaných `Items` nahradil dotazem na where exists. Dále se mi již dotaz optimalizovat nepodařilo a na základě těchto výsledků jsem se proto rozhodl nejnáročnější dotazy a jejich výsledky ukládat do vyrovnávací paměti aplikace.

8 Uživatelská příručka

8.1 Zápis položky Marktime

Základní funkcí, která je dostupná každému uživateli je zápis položky Marktime. Vstupní formulář se zobrazí po kliknutí na odkaz „Zadej záznam“ v menu aplikace.

Po kliknutí se zobrazí formulář přehled položek uživatele zapsaných s datem poslední zapsané položky, pokud její zápis proběhl dnes, jinak k dnešnímu datu. Cílem je, aby uživatel při typickém scénáři, kdy dnes postupně zapisuje řadu položek s datem předchozího dne dostal po zadání každé položky opět přehled položek se včerejším a nikoliv dnešním datem. Datum pro které jsou již zapsané položky zobrazeny, lze také změnit odpovídajícím přepsáním zvýrazněného řetězce v adresním řádku .../**2023-11-04**/todayItems.



Obrázek 6: Formulář pro zadání Položky

Formulář obsahuje následující funkcionality (seznam odpovídá jejich umístění od shora dolů):

1. Suma odpracovaného času, který uživatel již zadal v položkách s datem pro které je formulář vygenerován.

2. Datum, kdy uživatel práci odvedl. Pokud je poslední položka uživatele jiného data než dnešního, předvyplní se datum této položky, jinak se vyplní

dnešní datum. Předvyplnění záměrně probíhá nezávisle na případné změně datového řetězce v adresním řádku prohlížeče.

Nelze zadat datum v budoucnosti a zároveň nelze zadat datum o více než 30 dní předcházející dnešku.

3. Nejčastější: Seznam nejčastěji zadávaných portfolií vytvořený pro přihlášeného uživatele, jehož použitím se lze v převážné většině případů vyhnout nutnosti výběru s rozsáhlého seznamu „Ostatní portfolia“. Portfolia jsou řazena podle počtu zápisů, která na ně uživatel provedl za aktuální a předcházející měsíc. Tento seznam je uložen v mezipaměti a je podle skutečných dat aktualizován až při následujícím požadavku uživatele, který je učiněn alespoň po jedné hodině od předchozí aktualizace.

   Volba portfolia v tomto seznamu má přednost před volbou v poli „Ostatní portfolia“. Zrušit jednou provedenou volbu lze kliknutím na „X“ na začátku formuláře. Poté aplikace zohlední hodnotu „Ostatní portfolia“.

4. Ostatní portfolia: Seznam všech aktivních portfolií pro výběr portfolia, které není obsaženo v „Nejčastější“. Zvlášť na mobilním telefonu není jeho použití uživatelsky přívětivé.

   Pokud není vybrána hodnota „Nejčastější“ akceptuje aplikace volbu z tohoto seznamu.

5. Skupina úkolů: Po kliknutí na zvolenou skupinu úkolů se níže vypíše seznam úkolů náležejících do této skupiny. Při změně skupiny se zobrazí seznam úkolů patřících k nově vybrané skupině. Pokud byl před změnou skupiny vybrán úkol z jiné skupiny, dojde změnou skupiny k jeho „odvybrání“, takže vždy je možné zvolit jen úkol náležející do vybrané skupiny úkolů.

   Musí být vybrána jedna skupina úkolů.

6. Úkol: Seznam úkolů, které náleží do uživatelem vybrané skupiny. Je nutné kliknutím označit jednu možnost.

   Musí být vybrán jeden úkol.

7. Hodin a Minut: Čas strávený prací na zapisované položce.

   Pokud není formulář odeslán tlačítkem Navazuj, je nutné vyplnit alespoň jednu z hodnot. Nelze vyplnit méně než 1 minutu a více než 10 hodin a 600 minut[15].

8. Popis: Upřesňující popis položky. Nemůže být prázdný.

9. Tlačítko Uložit: Uloží záznam do databáze standardním způsobem.

[15]Cílem omezení je zabránit překlepům v zadání minut do hodin anebo chybného navazování na včerejší položku. Více, než 10 hodin je možné zadat kombinací obou polí - např. 10 hodin a 60 minut.

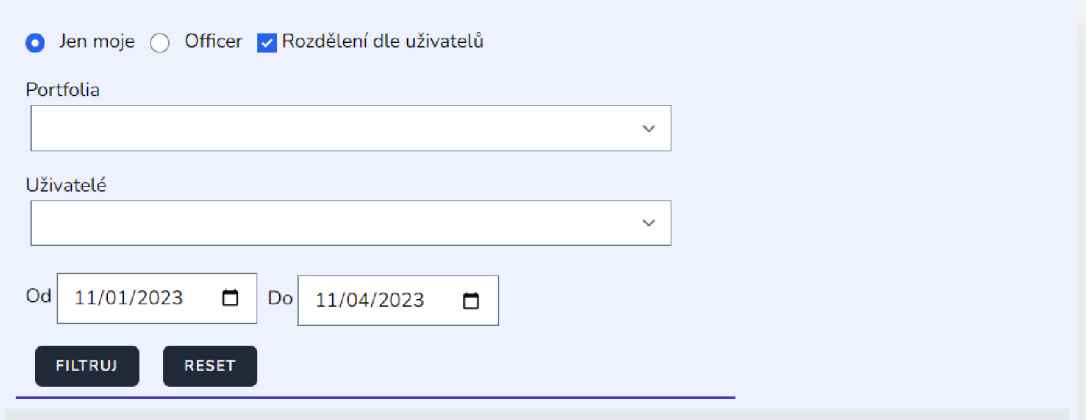
10. Tlačítko Navazuj: Při uložení vynuluje hodnoty Hodin a Minut a namísto toho položce přiřadí trvání v počtu celých minut uplynulých od okamžiku zadání předchozí položky uživatele.[16]

8.2 Zobrazení přehledu odvedené práce

8.2.1 Zobrazení přehledu práce přihlášeného uživatele

Každý uživatel si může zobrazit jím zadané položky za libovolný časový úsek a pro libovolné z Portfolií, na která zápis v daném časovém úseku provedl. Formulář pro zadání parametrů se zobrazí po kliknutí na odkaz „Přehled záznamů“ v menu aplikace. Defaultní nastavení filtruje všechny položky uživatele s datem od prvního dne aktuálního měsíce.

Ve výchozím filtru lze vybrat jedno Portfolio a jednoho Uživatele.



Obrázek 7: Základní filtr Položek

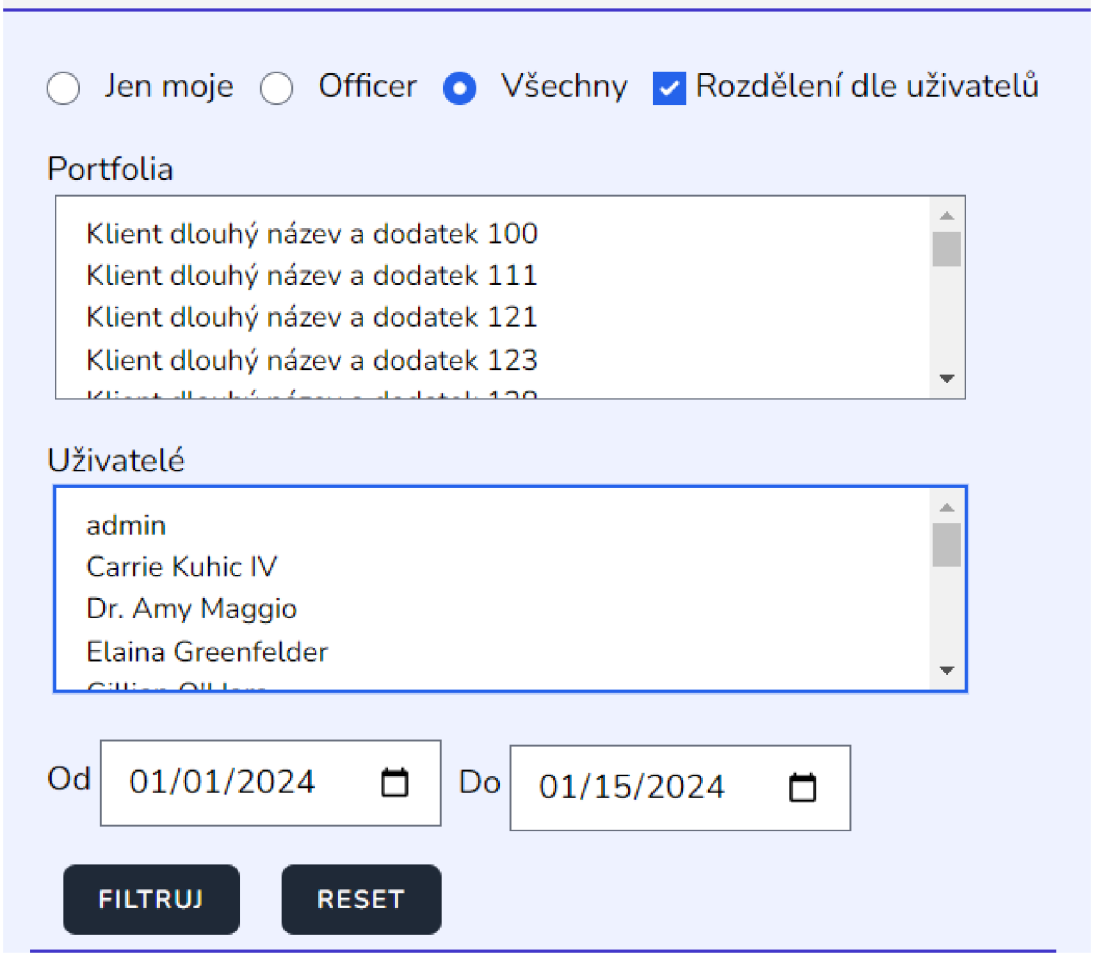
Při dalším odeslání filtru lze výběr dále zpřesňovat.

A to volbou jednoho až všech Portfolií nebo jednoho až všech uživatelů. Pokud uživatel nevybere žádné z Portfolií, aplikace tento krok vyhodnotí tak, jakoby vybral všechna.

Kromě stránkovaného seznamu těchto položek se uživatel zobrazí Preview, které obsahuje Portfolia seřazená podle sumy odpracovaného času v provedeném výběru. Ke každému portfoliu je uvedena:

1. Suma odpracovaného času
2. Relativní podíl vůči ostatním portfoliím
3. Ke každému portfoliu rozdělení na Skupinu úkolů včetně hodnot ad 1. a ad 2. pro tuto Skupinu úkolů.

[16] Dochází tedy ke „ztrátě“ průměrně půl minuty na takový zápis. Při 20 zápisech za den tak půjde o přibližně 10 minut, což pravděpodobně nedosahuje času, který uživatel na pracovišti stráví jinak, než prací.



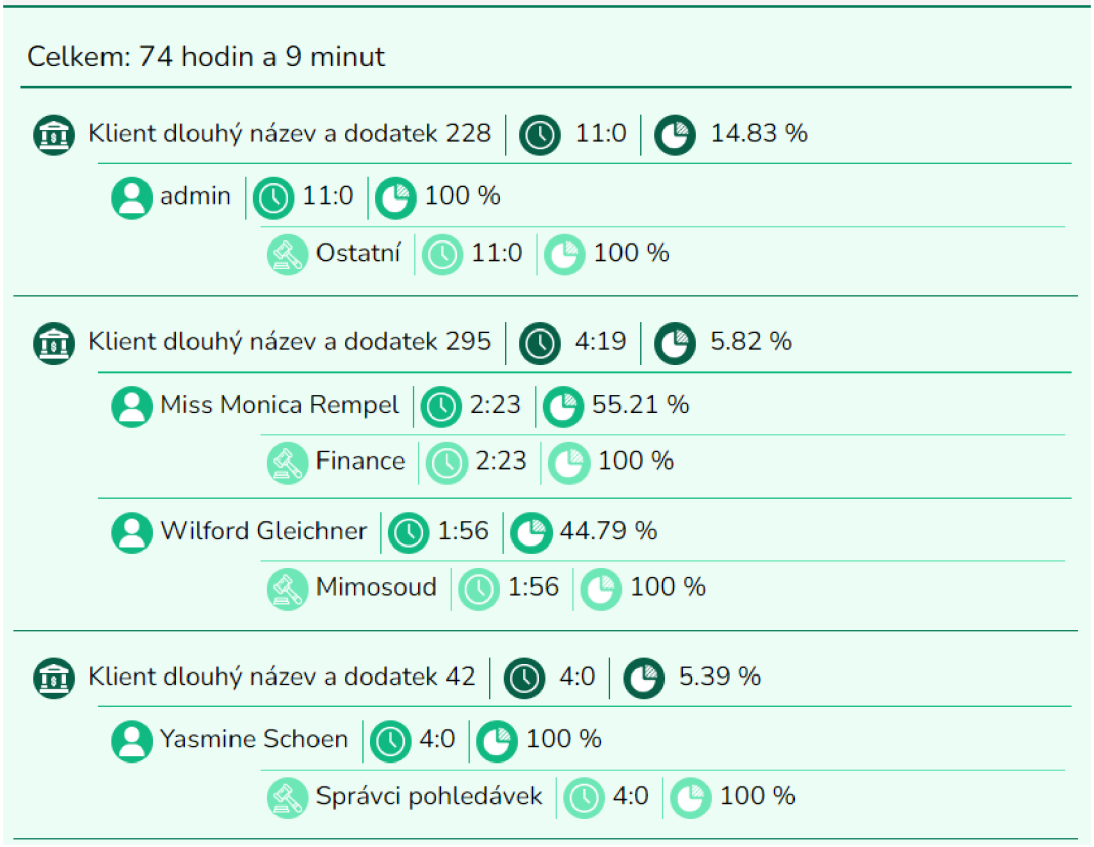
Obrázek 8: Dodatečné filtrování z nalezených výsledků

4. Zaškrtnutím pole „Rozdělení dle uživatelů“ před odesláním formuláře je do Preview vloženo ještě rozdělní odpracovaného času na jednotlivé uživatele. V případě generování přehledu vlastních položek je ale přirozeně obsažen pouze přihlášený uživatel.

Při změně data dojde vždy ke zrušení omezení Portfolií a Uživatelů.

Vždy se lze pokynem Zpět v prohlížeči vrátit k výsledkům předchozího filtrování. Parametry filtru jsou totiž vždy uvedeny v adresním řádku webového prohlížeče.

Pokud uživatel v posledních 10 minutách provedl filtraci se stejnými parametry, zobrazí mu aplikace seznam Položek i Preview z dat vyrovnávací paměti a nebudou do uplynutí této lhůty zohledněny případné změny v Položkách za tuto dobu. Generování Preview již z řádově milionů Položek je výpočetně náročné.



Obrázek 9: Preview

8.2.2 Zobrazení přehledu práce na portfoliích officera

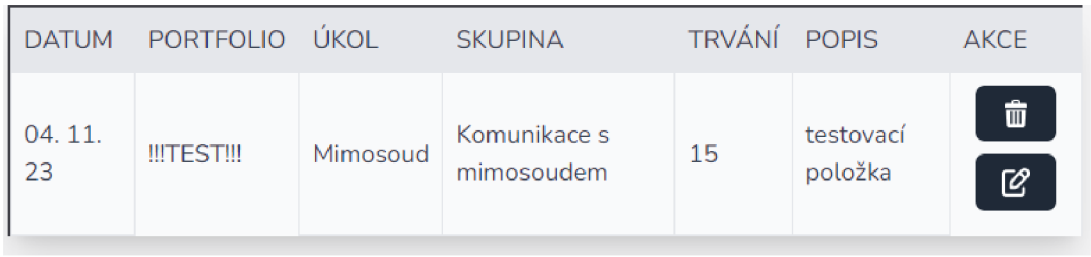
Uživatel s oprávněním alespoň Supervisor může uživateli přiřadit 1 až všechna portfolia, jako Officerovi. Officerů může být jedno Portfolio i více. Officerovi se při zakliknutí volby „Officer“ zobrazí všechny Položky zapsané na Portfolia, jejichž je officerem. Následně může filtrování upřesňovat zúžením výběru Portfolií anebo Uživatelů. V prohlížeči Chrome se volba provádí kliknutím se stisknutou klávesou Ctrl.

8.2.3 Zobrazení přehledu práce jiného uživatele

Uživatel s oprávněním alespoň Supervisor si může zobrazit Položky všech ostatních uživatelů a provádět filtrování.

8.2.4 Editace a smazání Položky

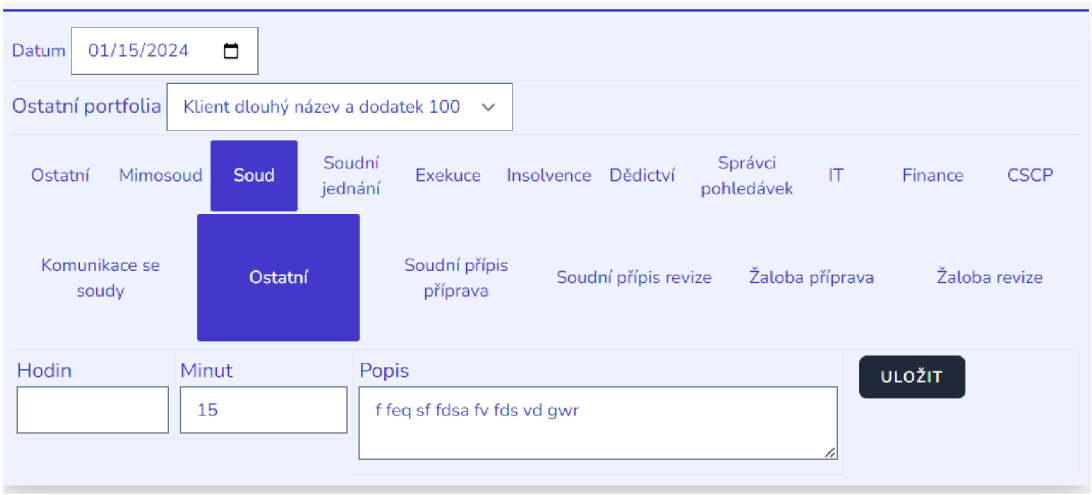
Položku lze smazat a editovat prostřednictvím ikony koše nebo ikony papíru s tužkou zobrazené u této položky. Jakmile je Položka exportována do Formicu, je pro editaci i smazání uzamčena a tyto ikony se ani nezobrazí. K exportu obvykle dochází každý den v noci.



Obrázek 10: Umístění ikon editace a smazání

Před uzamčením položky se uplatňují následující pravidla. Každý uživatel může mazat a editovat svoje Položky. Uživatel s oprávněním alespoň Supervisor může mazat a editovat i neuzamčené položky jiných uživatelů.

Při volbě editace se zobrazí editační formulář s předvyplněnými stávajícími údaji Položky. Uživatel může následně některé údaje upravit a aktualizovaný záznam uložit.



Obrázek 11: Editační formulář

Ke smazání položky dochází okamžitě po kliknutí na ikonu odpadkového koše bez nutnosti dalšího potvrzování. Smazání položky nelze vzít zpět.

8.3 Oprávnění uživatelů

8.3.1 Běžný uživatel

Běžný uživatel může zapisovat Položky a dokud nejsou uzamčené, provádět jejich editaci nebo mazání. Má právo k zobrazení svých položek a položek Portfolií, u kterých je Officer.

8.3.2 Supervisor

Supervisor má všechna oprávnění, jako Běžný uživatel. Kromě toho může zobrazovat, editovat a mazat Položky libovolného uživatele.

Supervisor má také oprávnění provádět správu uživatelů do úrovně Supervisor, kromě mazání uživatelů a kromě přidělení práv Administrátora. Supervisor může provést deaktivaci uživatele. Nemůže deaktivovat sám sebe, protože by došlo k okamžitému odhlášení z aplikace bez možnosti se přihlásit zpět.

8.3.3 Administrátor

Administrátor má všechna oprávnění jako Supervisor. Kromě toho může smazat uživatele, který nemá zadanou žádnou Položku. Uživatel, který má zadanou alespoň jednu Položku je vždy místo smazání deaktivován. Nelze smazat sám sebe.

8.4 Správa uživatelů

8.4.1 Vytvoření uživatele

Uživatel s oprávněním alespoň Supervisor může vytvořit nového uživatele. Formulář pro vytvoření uživatele je přístupný z menu aplikace pod odkazem „Přidej uživatele“. Doména emailu všech uživatelů je jednotná pro všechny uživatele aplikace kromě emailu Administrátora vytvořeného při instalaci aplikace a je obsažena v konfiguraci aplikace. Žádný z uživatelů aplikace ji nemá právo změnit.

Uživatele je vhodné vytvořit jako aktivního, jinak mu aplikace neumožní přihlášení. Formic ID slouží pro export Položek do Formicu a musí odpovídat ID uživatele v aplikaci Formic.

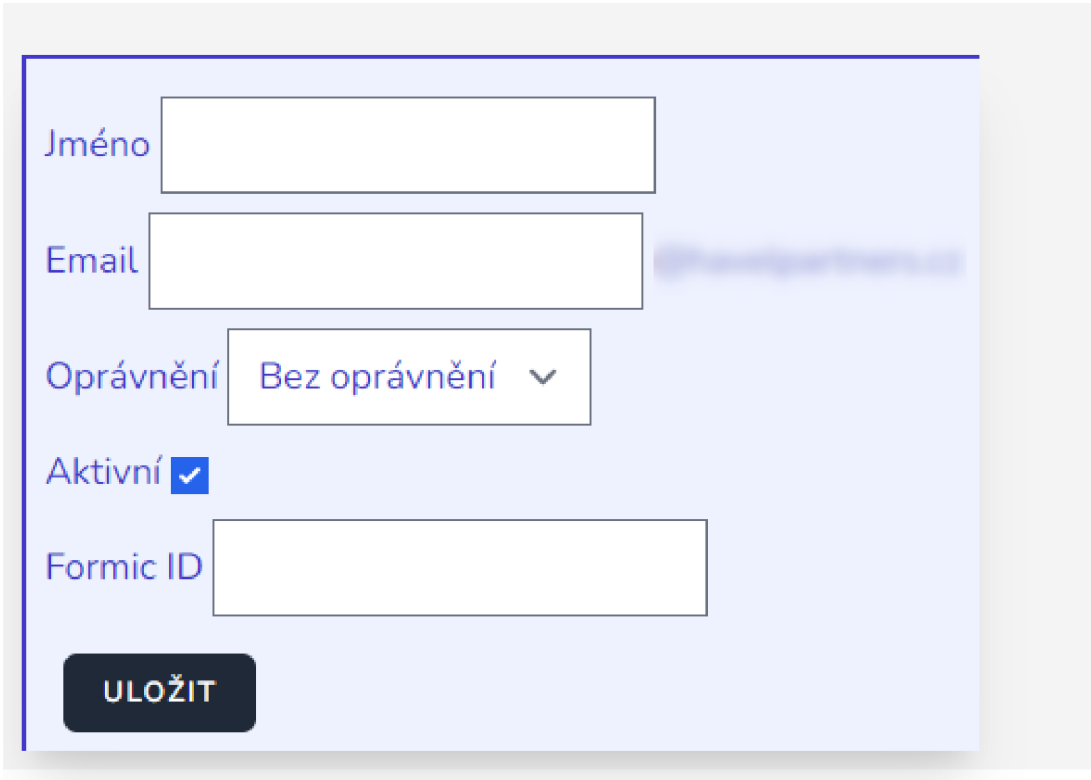
Nově vytvořenému uživateli je vytvořeno náhodné heslo, které nelze zjistit. Uživatel následně musí přes odkaz „Zapomněli jste heslo?“ požádat o zaslání odkazu pro obnovení hesla. Odkaz mu aplikace zašle na email zadaný při vytvoření uživatele.

8.4.2 Zobrazení přehledu uživatelů

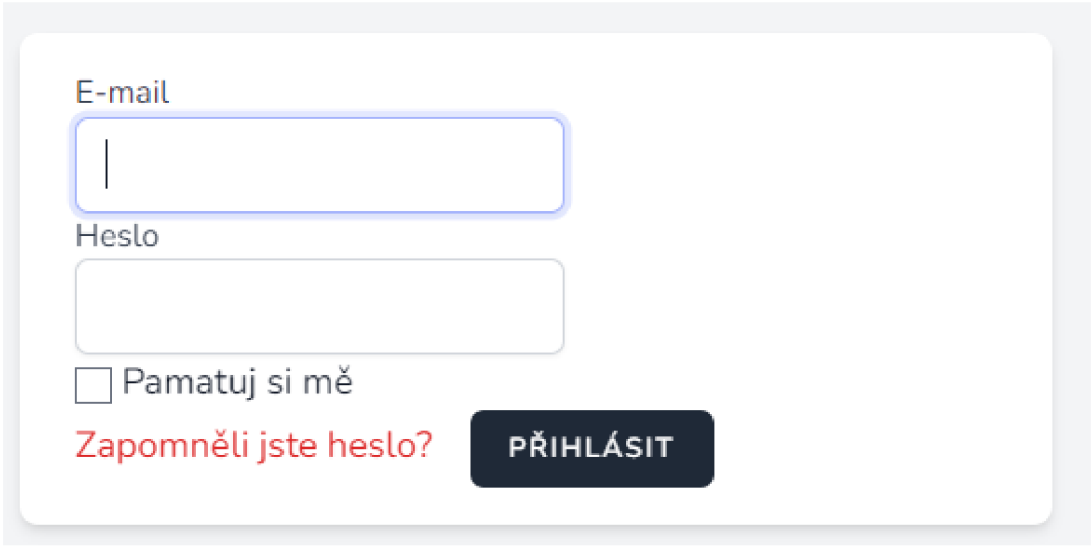
Uživatel s oprávněním alespoň Supervisor je oprávněn k zobrazení přehledu všech uživatelů. U každého z nich je potom uveden odkaz na jeho editaci nebo smazání, jejichž rozsahu je omezen oprávněním uživatele.

8.4.3 Editace uživatele

Editace uživatele je přístupná přes ikonu „papír s tužkou“ u každého uživatele. Uživatel nemůže editovat uživatele s vyšším oprávněním, než které má on sám. Může editovat sám sebe, kromě provedení deaktivace. Snížení úrovně oprávnění sám sobě není zakázáno.



Obrázek 12: Formulář vytvoření uživatele



Obrázek 13: Přihlašovací formulář a odkaz na obnovení hesla

Formulář pro editaci uživatele obsahuje předvyplněná dosavadní data editovaného uživatele. Přidáním Portfolia do správy získá uživatel oprávnění Officera Portfolia a bude oprávněn k zobrazení Položek ostatních uživatelů zapsaných na toto portfolio. Přidání Portfolií probíhá po jednom z rozvinovacího menu Při-

ID	JMÉNO	EMAIL	OPRÁVNĚNÍ	AKTIVNÍ	FORMIC ID	AKCE
1	admin	[redacted]	Administrátor	Ano	135	
2	vedouci	vedouci@upol.cz	Administrátor	Ano	136	
3	oponent	oponent@upol.cz	Administrátor	Ano	137	
6	Mariah Bashirian	jaquelin26@example.org	Bez oprávnění	Ano	533	

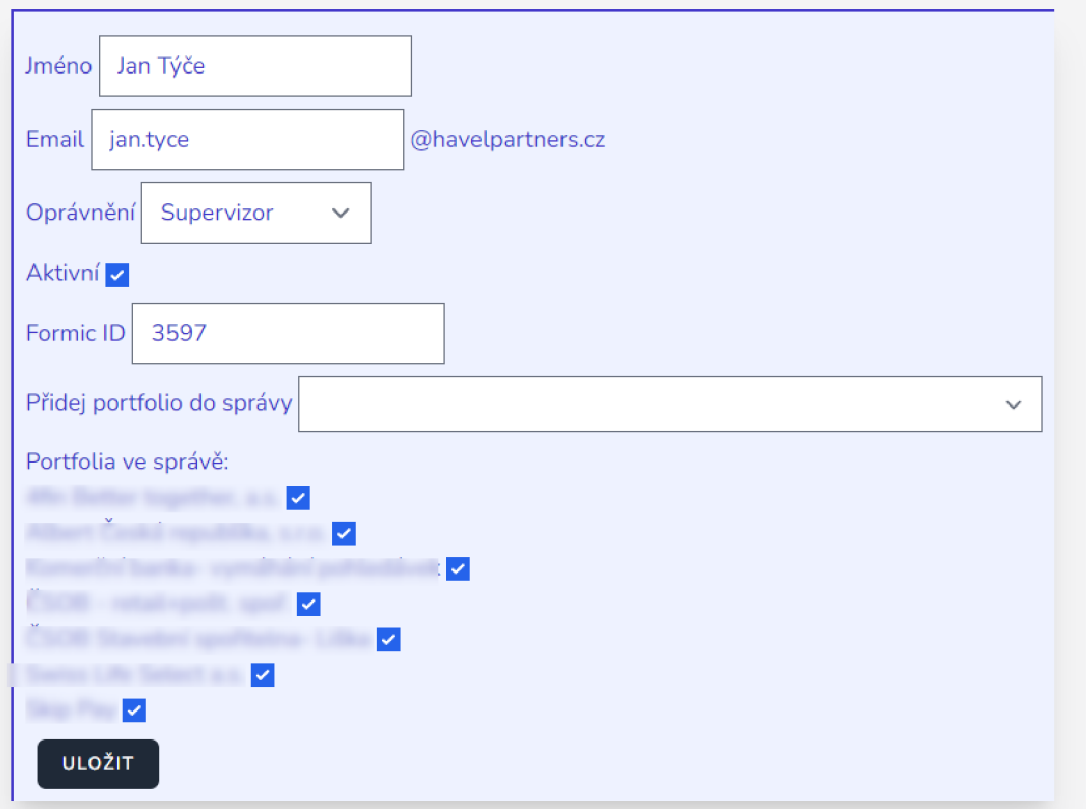
Obrázek 14: Přehled uživatelů

dej portfolio do správy. Po uložení změn se načte opětovně editační formulář uživatele, se zohledněním těchto změn. Takto lze tedy poměrně jednoduše uživateli přiřadit více portfolií. U každého z přiřazených portfolií je následně uveden checkbox. Jeho vykliknutím dojde po uložení formuláře k odebrání Portfolia ze správy.

8.4.4 Správa aplikace

Administrátor je oprávněn provést aktualizaci seznamu Portofilí, Skupin úkolů a Úkolů v databázi. Nové seznam lze vložit v sekci Správa aplikace v jedné tabulce formátu Microsoft Excel. Tabulka musí obsahovat 3 listy s názvy `portfolios`, `groups`, `tasks`. Na každém z těchto listů musí být vyplněny všechny sloupce uvedené v aktualizačním formuláři. Pokud jsou vložena chybná data, dojde pravděpodobně k chybnému stavu aplikace, který nelze odstranit bez přístupu do databáze. Nehrozí ztráta uživatelských dat, ale nově vytvořená uživatelská data mohou být nepoužitelná.

V této sekci lze také vymazat všechny vyrovnávací paměti aplikace. Volba této možnosti nevede k ohrožení chodu aplikace ani ke ztrátě uživatelských dat.



Obrázek 15: Formulář pro editaci uživatele včetně checkboxů již přiřazených Portfolií

9 Programátorská příručka

9.1 Struktura zdrojového adresáře

Adresář aplikace obsahuje větší množství adresářů a souborů. Převážná část těchto souborů je součástí frameworku Laravel a pro běžnou funkčnost aplikace do nich není nutné zasahovat. Cílem této části práce je poskytnout vodítko k tomu, k jakému účelu jednotlivé soubory slouží a v jakém souboru má programátor potřebnou funkčnost hledat. Dále se zabývám pouze soubory, které jsem při vývoji aplikace vytvořil nebo funkčně upravoval. Ostatní adresáře a soubory, které nejsou mým autorským dílem, ponechávám bez komentáře.

app/

Classes/
Zde je umístěna samostatně vytvořená třída `Preview`, sloužící jako datová struktura pro uspořádání dat pro vytvoření přehledu z vyfiltrovaných hodnot a třídy `Client` a `UserClient`.

Http/

Controllers/
Obsahuje třídy Controllerů. V podadresáři `Auth` jsou autentizační Controllery balíčku Breeze.

Middleware/
Obsahuje Middleware využívaný frameworkem a mnou doplněný Middleware `BlockDeactivatedUsers` sloužící kontrole každého požadavku, zda jej zasílá uživatel, který nebyl deaktivován. Pokud byl deaktivován, provede se jeho odhlášení.

Models/
Obsahuje třídy Modelů. Atributy modelů a struktura vztahů mezi modely jsou popsány v části Datová základna aplikace.

Policies/
Obsahuje politiky `ItemPolicy` a `UserPolicy` upravující oprávnění uživatelů k nakládání s Modely `Item` a `User`.

config/
Adresář config kromě nastavení různých oblastí frameworku obsahuje také mnou vytvořený soubor `constants.php` definující globální konstanty jako počet Portoflií v Preview nebo počet Portfolií ve zkrácené volbě. Zde se také nachází emailová adresa uživatele, který je při prvním spuštění aplikace vytvořen jako administrátor. Zde je také nastaveno, že se mu vytvoří náhodné heslo [17], počet prvků na stránce při stránkování a firemní emailová doména, se kterou se automaticky zapisují emaily uživatelů [18]. Úpravou těchto hodnot může programátor jednoduše změnit chování aplikace.

public/
Adresář public obsahuje soubor `index.php`, který je vstupním bodem do aplikace. V tomto adresáři jsou také uloženy veřejné soubory pro vytvoření webových stránek, jako jsou obrázky a CSS.

database/

factories/
Obsahuje `ItemFactory` a `UserFactory` sloužící k vytváření testovacích záznamů do databáze pro provedení testů.

migrations/
Obsahuje migrace vytvářející a měnící databázi.

seeders/
Obsahuje `DatabaseSeeder` ve kterém je nakonfigurováno vytvoření účtu administrátora, pokud databáze neobsahuje žádného uživatele a

[17] Pro účely testování aplikace vedoucím a oponentem se novému uživateli vytváří heslo password. Heslo lze změnit pomocí formuláře pro obnovu zapomenutého hesla. Na serveru katedry na adrese 158.194.92.79 je aplikace nasazena s funkčním připojením k mailserveru

[18] Úmyslně nechceme, aby bylo možné zadat email uživatele s jinou než firemní doménou

plnění a aktualizace databáze z vložených tabulek ve formátu Microsoft Excel.

resources/

css/
Obsahuje soubor `app.css`, který následně vstupuje do kompilace „veřejného“ css v adresáři public. Zde je upraveno chování css skupiny tříd prvku `radio-hack`, jejíž funkčnost nešlo upravit pomocí Tailwind.

lang/
Obsahuje soubory pro překlad zpráv frameworku o vyřízení žádosti uživatele do jazyka lokalizace aplikace. Aplikace obsahuje tyto soubory pro český a anglický jazyk v adresářích s příslušným názvem. Obsahuje také soubor `cs.json`, pomocí něhož se na jazyk lokalizace překládají také zprávy systému a text v Blade šablonách, který je uvozený direktivou dvou podtržítek.

views/
Obsahuje soubory Blade šablon pro generování Pohledů. Soubory jsou rozděleny do složek podle příslušnosti ke Controlleru, který je generuje (`auth/` pro autentizační Controllery, `items/` pro `itemController` a `users/` pro `userController`). Adresář `components/` obsahuje komponenty Pohledů, které jsou použity, aby byla eliminována duplikace kódu. Takže například ve formuláři pro vytvoření i pro editaci záznamu je použita stejná komponenta `create-form.blade.php`, které mění chování v závislosti na vyplnění parametru isEdit.

routes/
Adresář routes obsahuje soubory určující směrování příchozích požadavků ke Controllerům. V aplikaci využívám pouze soubor `web.php`.

storage/app/database/seeders Zde aplikace ukládá poslední vloženou tabulku s údaji k aktualizaci Portfolií, Skupin úkolů a Úkolů. Zdrojová data aplikace zde již tuto tabulku obsahují a z této tabulky se provádí plnění databáze při nasazení aplikace.

tests/
Adresář test obsahuje soubor tříd testů popsaných v části Testování.

Dále jsou v kořenovém adresáři aplikace uloženy samostatné soubory, z nichž zásadní je soubor **`.env`**, do kterého je pro samotnou funkčnost aplikace nutné vyplnit údaje nutné pro přihlášení a pro přístup do databáze a také údaje pro přihlášení a přístup k mailserveru.

9.2 Spuštění aplikace

Proces nasazení aplikace je popsán v souboru `README.txt`, který se nachází v kořenovém adresáři dat bakalářské práce.

Po dobu obhajoby práce je aplikace zprovozněna na serveru Katedry Informatiky. Virtuální server je dostupný na adrese http://158.194.92.79.

Závěr

Výsledkem této práce je webová aplikace pro zápis odpracovaného času. Povedlo se mi podle mého názoru splnit uživatelské požadavky a tuto činnost učinit pro uživatele o něco méně nepříjemnější. Povedlo se mi zejména zjednodušit manipulaci s formulářem pro zápis záznamů, zpřístupnit data ostatních uživatelů podle zvoleného systému oprávnění a zpřístupnit aplikaci mimo podnikovou síť.

Aplikace je v ostrém testovacím provozu od září 2023. Zápisy jejím prostřednictvím provádím já a další dva kolegové. Prakticky ihned od nasazení vzrostla v mém případě metrika včasnosti zápisu ze zmiňovaných 20 % na 100 % a na této hodnotě setrvává a zbavil jsem se úplně zbytečné pomocné papírové evidence. Zároveň mi vzrostl průměrný počet zápisů z 7,6 na 13,3 denně včetně svátků dovolených a víkendů. Rozvržení času je tak výrazně přesnější, když dříve jsem zápis kratších aktivit poněkud opomíjel.

Aktuálně je zprovozněno napojení aplikace na podnikovou databázi, která si z databáze aplikace každý den stahuje nové záznamy. U exportovaných záznamů následně dochází k blokaci editace a blokaci smazání. Aktuálně čekám na nasazení importu záznamů z podnikové databáze do databáze aplikace. Uživatel tak bude mít v aplikaci vždy všechny svoje zápisy a pokud dojde k editaci již exportovaného záznamu, promítne se tato editace také do aplikace. Jakmile bude tento problém vyřešen, nabídneme aplikaci k použití dalším kolegům.

V průběhu vývoje došlo k docela zásadnímu rozšíření počtu Skupin úkolů. Vzhledem k responzivnímu designu i u desktopového zobrazení nedošlo ani po aktualizaci databáze k zásadnějšímu narušení vzhledu formuláře pro vytváření záznamů. Formulář nenarušilo ani zvýšení počtu nejčastějších Portfolií z 5 na 8, které si vyžádala kolegyně z důvodu snadnějšího používání mobilní verze zobrazení. Nicméně je nutné dodat, že možnosti formuláře nejsou neomezené a při zobrazení 8 nejčastějších Portfolií a zobrazení 15 úkolů ve Skupině úkolů Finance dojde při šířce 5/12 obrazovky a velikosti textu 125 % k mírnému posunu následujících prvků formuláře, kterému jsem se chtěl vyhnout. Všechny požadované vlastnosti formuláře jsou ale při nastavené velikosti textu na 100 % plně zachovány.

Zpětně hodnotím jako dobrou volbu rozhodnutí vyvíjet aplikaci za použití frameworku Laravel. Při studiu jeho součástí jsem se dozvěděl mnoho o fungování webové aplikace a podrobná dokumentace i široká uživatelský komunita byly velmi užitečné k vyřešení problémů na které jsem narazil. Zdaleka jsem přitom nevyužil všech možností, které framework nabízí, a určitě bych se v něm nebál započít s vývojem další aplikace většího rozsahu. Užitečné a poněkud těžce nabyté poznatky jsem také získal při konfiguraci virtuálního linuxového serveru pro účely odevzdání zdrojového kódu této práce, když s touto oblastní jsem dosud neměl vůbec žádné zkušenosti. Aplikaci jsem doposud nasazoval vždy na hotový hosting a prováděl jsem tak prakticky jen kopírování jejích souborů.

Pokud jde o budoucí vývoj plánuji přidat do formuláře pro zápis tlačítko začátek/konec práce. Nyní musí uživatel pořád vyplnit čas první aktivity po

příchodu do práce nebo po přestávce. Stisknutím tlačítka ale mohu jednoduše provést zápis do databáze zvláštního typu a od něj pak existující metodou spočítat čas první aktivity a tento pomocný záznam následně vymazat. Budu muset v dohledné době nahradit balíček Laravel Collective a provést upgrade na Laravel 10.

Conclusions

The result of the thesis is a web application for recording the time worked. In my opinion, I managed to meet the user requirements and make this activity a little less uncomfortable for the user. I have managed to simplify the handling of the entry form, to make the data of other users accessible according to the authorization system chosen, and to make the application accessible outside the corporate network.

The application has been in live test operation since September 2023. Immediately after deployment, the promptness metric in my case increased from the above-mentioned 20 % to 100 % and has remained at this value up to now. I got rid of the useless paper records too. At the same time, my average number of entries has increased from 7.6 to 13.3 per day, including holidays and weekends. The time allocation is thus significantly more accurate. Previously, I had neglected to enter shorter activities.

Currently, the application's connection to the company database is enabled, New entries are downloaded from the application's database every day. Editing and deletion is disabled for exported records. I am currently waiting for the procedure for importing records from the enterprise database to the application database to be deployed. This way, the user will always have all of their records in the application and if an already exported record is edited or deleted, this action will also be reflected in the application. Once this issue is resolved, the Application will be fully operational and available for use by other colleagues.

During the development process, the number of Task Groups has been expanded quite substantially. Due to the responsive design, even in the desktop view, there was no major distortion in the appearance of the record creation form even after the database update. The form was also not distorted by the increase in the number of the recent Portfolios from 5 to 8, which was requested by a colleague to make the mobile version easier to use. However, it should be added that the form's capabilities are not unlimited and displaying the 8 most common Portfolios and displaying 15 tasks in the Finance Task Group will cause a slight shift of the following form elements at 5/12 screen width and 125 % text size, which I wished to avoid. However, all the desired form features are fully maintained with the text size set to 100 %.

With hindsight, I consider the decision to develop the application using the Laravel framework a great choice. I learned a lot about the functionality of a web application while studying its components, and the detailed documentation and the broad user community were very helpful in solving the problems I encountered. In doing so, I have by no means used all the possibilities the framework offers, and I certainly wouldn't be afraid to start developing another larger scale application in it. I also gained useful and somewhat hard-won knowledge when configuring a virtual Linux server for the purpose of submitting the source code for this thesis, having had absolutely no experience in this area before. Up to now I have always deployed the application on a ready-made hosting server and

have done so by simply copying its files.

As for future development I plan to add a start/end button to the entry form. Now the user still has to fill in the time of the first activity after coming to work or after a break. However, by pressing the button I can simply make an entry into a database of a special type and then use an existing method to calculate the time of the first activity from there and then delete this auxiliary record. I will also need to replace the Laravel Collective package and upgrade to Laravel 10 shortly.

A Seznam funkčních testů aplikace

- AuthenticationTest
  - login screen can be rendered
  - users can authenticate using the login screen
  - users can not authenticate with invalid password
- PasswordResetTest
  - reset password link screen can be rendered
  - reset password link can be requested
  - reset password screen can be rendered
  - password can be reset with valid token
- FilterControllerTest
  - custom filter can be processed
- ItemControllerTest
  - index view can be rendered
  - index view not split by users can be rendered
  - todays items view can be rendered
  - correct item can be created
  - subsequent item can be created
  - incorrect item cannot be created
  - edit item form can be rendered
  - edit item form cannot be rendered for unauthorized item
  - item can be updated only by authorized user
  - item can be delted only by authorized user
- UserControllerTest
  - user index view can be rendered only for authorized user
  - create user form can be rendered only by authorized user
  - correct user can be created only by authorized user
  - incorrect user cannot be created
  - user edit form can be rendered only by authorized user
  - user can be updated only by authorized user
  - user can be deleted only by active admin

B Použité ikony

V aplikaci byly použity tyto ikony ze serveru Flaticon.com vytvořené autory s níže uvedenými pseudonymy. Autorům děkuji za zpřístupnění jejich práce k dalšímu využití.

- Koš - bqlqn
- Edit - dmitri13
- Banka - Freepik
- Uživatel - Freepik
- Hodiny - Those Icons
- Kladívko - Freepik
- Podíl - juicy_fish

C Obsah elektronických dat

`text/`
: Adresář obsahuje text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce. Všechny (textové) přílohy, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu jsou uloženy v podadresáři `/kidiplom`.

`README.TXT`
: Textový soubor s postupem pro zprovoznění aplikace na referenční čisté instalaci Ubuntu Server 22.04 LTS včetně pokynů pro nastavení nástroje VirtualBox.

`src/`
: Kompletní zdrojové texty aplikace včetně zdrojových textů frameworku Laravel v adresářové struktuře pro zkopírování na úložiště webového serveru.

Literatura

[1] Redmond Consulting, s.r.o. *Prezentace systému ForMic* [online]. 2022 [cit. 2024-1-7]. Dostupný z: ⟨http://www.gbnl.cz/⟩.

[2] QUITEC s.r.o. *Prezentace aplikace Lamael* [online]. 2024 [cit. 2024-1-7]. Dostupný z: ⟨https://www.lamael.cz/⟩.

[3] Toggl OÜ. *Powerful and frictionless time tracking for the entire team* [online]. 2024 [cit. 2024-1-7]. Dostupný z: ⟨https://toggl.com/⟩.

[4] Easy Software Ltd. *Ovládněte své projekty, pracujte v klidu,doručte více* [online]. 2024 [cit. 2024-1-7]. Dostupný z: ⟨https://www.easyproject.cz/⟩.

[5] Campbell, Brooke. *Coding from scratch vs. using a framework* [online]. 2017 [cit. 2024-1-8]. Dostupný z: ⟨https://devm.io/programming/coding-scratch-vs-using-framework-134319⟩.

[6] Codiant Software Technologies Pvt. Ltd. *Top 5 PHP Frameworks for Web Development in 2023* [online]. 2023 [cit. 2024-1-8]. Dostupný z: ⟨https://codiant.com/blog/top-5-php-frameworks-for-web-development-in-2023/⟩.

[7] ENLYFT NETWORKS PRIVATE LIMITED. *Companies using Laravel* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://enlyft.com/tech/products/laravel⟩.

[8] ENLYFT NETWORKS PRIVATE LIMITED. *Companies using Nette* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://enlyft.com/tech/products/nette⟩.

[9] ENLYFT NETWORKS PRIVATE LIMITED. *Companies using CodeIgniter* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://enlyft.com/tech/products/codeigniter⟩.

[10] Jackson, Paul. *CodeIgniter vs Laravel – Difference Between Them* [online]. 2023 [cit. 2024-1-8]. Dostupný z: ⟨https://www.guru99.com/laravel-vs-codeigniter.html⟩.

[11] Koďousková, Barbora. *ARCHITEKTURA MVC: DEFINICE, STRUKTURA, FRAMEWORKY* [online]. 2021 [cit. 2024-1-8]. Dostupný z: ⟨https://www.rascasone.com/blog/architektura-mvc-struktura-frameworky⟩.

[12] Laravel LLC. *Eloquent ORM* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/eloquent⟩.

[13] Laravel LLC. *Database: Query Builder* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/queries⟩.

[14] Laravel LLC. *Blade Templates* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/blade⟩.

[15] Grunwell, Steve. *Writing Custom Laravel Blade Directives* [online]. 2019 [cit. 2024-1-8]. Dostupný z: ⟨https://stevegrunwell.com/blog/custom-laravel-blade-directives/⟩.

[16] Laravel LLC. *Middleware* [online]. [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/middleware⟩.

[17] Laravel LLC. *Authorization* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/authorization⟩.

[18] Laravel LLC. *Starter Kits* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://laravel.com/docs/9.x/starter-kits#laravel-breeze⟩.

[19] Finlay, Brion. *laravel-excel-seeder* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://github.com/bfinlay/laravel-excel-seeder/blob/master/LICENSE⟩.

[20] Engebretson, Adam. *LaravelCollective/html* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://github.com/LaravelCollective/html⟩.

[21] SPATIE BVBA. *Laravel-html* [online]. 2024 [cit. 2024-1-8]. Dostupný z: ⟨https://spatie.be/docs/laravel-html/v3/introduction⟩.

[22] Codeheart, Irelia. *Tailwind vs. Bootstrap* [online]. 2023 [cit. 2024-1-11]. Dostupný z: ⟨https://caisy.io/blog/tailwind-vs-bootstrap⟩.

[23] Kumari, Riya. *Tailwind CSS Vs Bootstrap* [online]. 2023 [cit. 2024-1-11]. Dostupný z: ⟨https://www.tutorialspoint.com/tailwind-css-vs-bootstrap⟩.

[24] Dayan, Sarah. *No, Utility Classes Aren't the Same As Inline Styles* [online]. 2024 [cit. 2024-1-11]. Dostupný z: ⟨https://frontstuff.io/no-utility-classes-arent-the-same-as-inline-styles⟩.

[25] Tailwind Labs Inc. *Utility-First Fundamentals* [online]. 2024 [cit. 2024-1-11]. Dostupný z: ⟨https://tailwindcss.com/docs/utility-first⟩.

[26] Laravel LLC. *Testing* [online]. 2024 [cit. 2024-1-11]. Dostupný z: ⟨https://laravel.com/docs/9.x/testing⟩.

[27] Carlos Arguelles Marko Invanković, Adam Bender. *Code Coverage Best Practices* [online]. 2020 [cit. 2024-1-11]. Dostupný z: ⟨https://testing.googleblog.com/2020/08/code-coverage-best-practices.html⟩.