Mendelova univerzita v Brně
Provozně ekonomická fakulta

Implementace modulů informačního systému pro správu literárních kaváren

Diplomová práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Bc. Miroslav Škrobáček

Brno 2016

Na tomto místě bych rád poděkoval panu Mgr. et Ing. Janu Homolovi za možnost podílet se na vývoji této aplikace. Mé velké díky patří také panu Ing. Jiřímu Lýskovi, Ph.D. za vstřícný přístup a cenné rady při vedení této práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Implementace modulů informačního systému pro správu literárních kaváren**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou Směrnicí o zveřejňování vysokoškolských závěrečných prací.

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 16. května 2016 _____

Abstract

Škrobáček, M. Implementation of modules in the information system for literary cafes. Diploma thesis. Brno: Mendel University, 2016.
This thesis deals with design and implementation of modules in the information system for literary cafes. The theoretical part describes selected PHP MVC frameworks. Modules are created using several types of technology. The main part is created in the programming language PHP, using PHP framework Nette. As a data store is used relation database system MySQL.

Keywords

Thesis, information system, attendance, PHP, Nette, MySQL.

Abstrakt

Škrobáček, M. Implementace modulů informačního systému pro správu literárních kaváren. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.
Tato práce se zabývá návrhem a implementací modulů informačního systému pro správu literárních kaváren. V teoretické části se zabývá vybranými PHP MVC frameworky. Vlastní moduly jsou vytvořeny s využitím několika druhů technologií. Hlavní část je vytvořena v programovacím jazyce PHP a PHP frameworku Nette. Jako úložiště je použito relačního databázového systému MySQL.

Klíčová slova

Závěrečná práce, informační systém, docházka, PHP, Nette, MySQL.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Informační technologie si v dnešní době našly cestu téměř do všech oblastí lidské činnosti, kde zefektivnily tradiční způsoby řešení. Informační technologie nám tak poskytují cennou podporu v každodenním životě, ať už se jedná o zábavu nebo pracovní povinnosti.

V pracovním prostředí bezesporu patří mezi nejvýznamnější produkty elektronické informační systémy. Časy, kdy byly tyto systémy výhradně doménou velkých společností, jsou už dávno pryč. Elektronické informační systémy nás v dnešní době provází takřka na každém kroku. Můžeme se s nimi setkat například na vlakových nádražích v podobě příjezdových a odjezdových informačních tabulí nebo informační systémy veřejných knihoven poskytující informace o půjčovaných titulech.

Elektronické informační systémy jsou také čím dál více využívány v restauračních zařízeních, které ve větší či menší míře v sobě zahrnují samostatné systémy, mezi které patří například objednávkové systémy, rezervační systémy, účetnictví atd.

Řada těchto systémů je stále provozována lokálně na jednom zařízení bez možnosti komunikace s jiným zařízením. Obecně je nastaven trend tyto lokální systémy opatřovat rozhranním umožňující přístup z více zařízení. Nebo je nahrazovat webovými aplikacemi, které jsou díky Internetu dostupné téměř kdekoliv.

1.2 Cíl práce

Cílem této práce je vytvoření návrhu vybraných modulů do stávajícího informačního systému pro správu literárních kaváren a následně je do tohoto systému implementovat.

Úvodní část práce se zabývá programovacím jazykem PHP a vybranými MVC frameworky pro jazyk PHP.

Hlavní část práce je zaměřena na vlastní návrh a implementaci zvolených modulů. Těmto činnostem předchází analýza požadavků uživatelů stávajícího systému. Implementované moduly se zabývají problematikou související s každodenním provozem poboček. Do této problematiky spadá sdílení informací v kavárně, jak v rámci jedné, tak i mezi více pobočkami. Další modul obohatí stávající rezervační systém o novou funkcionalitu umožňující vytvářet blokace stolů na pobočkách. Zavedením komplexního docházkového modulu bude systém rozšířen o docházku zaměstnanců s možností vyplácení mezd. Posledním implementovaným modulem je správa událostí v kavárně.

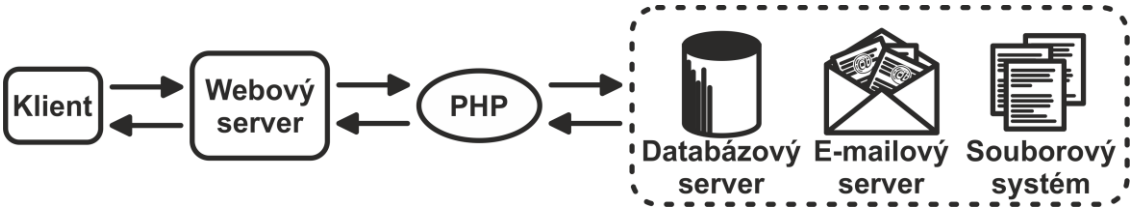
2 PHP a MVC frameworky

PHP: Hypertext Preprocessor více známý pod zkratkou PHP je open source (počítačový software s otevřeným zdrojovým kódem) skriptovací jazyk používaný převážně pro vývoj dynamických webových aplikací (The PHP Group, ©2001-2016). Zároveň se jedná o jazyk multiparadigmatický tzn., podporuje více než jedno programovací paradigma. V případě PHP se může jednat o níže uvedená paradigmata (PHP - Wikipedia, the free encyclopedia, 2016):

- Imperativní/Procedurální paradigma – daný problém je řešen krok pro kroku posloupností příkazů (algoritmus). Opakem imperativního paradigma je deklarativní paradigma, které klade důraz na definici výsledku. Způsob jakým je daný výsledek získán není vývojářem přímo definován a je v kompetenci konkrétního interpretu programovacího jazyka (Imperative programming - Wikipedia, the free encyclopedia, 2016).
- Funkcionální paradigma – toto paradigma spadá pod deklarativní programovací paradigma. Je založeno na vyhodnocování matematických funkcí, výrazů a jejich rekurzích. Základním kamenem je tedy funkce definovaná vstupními a jediným výstupním parametrem. Samotný program je pak složen funkcí, které se mohou mezi sebou navzájem provolávat (Functional programming - Wikipedia, the free encyclopedia, 2016).
- Objektově orientované paradigma – založeno na myšlence vzájemně komunikujících objektů, kde implementované objekty jsou abstrakcí objektů reálného světa. Každý z těchto objektů obsahuje vlastní množinu stavů, ke kterým lze přistupovat výhradně prostřednictvím definovaného rozhraní (vlastnost zapouzdření). Objekty jsou rozčleněny do tříd. Jinými slovy každý objekt je instancí dané třídy. Třídy mohou být hierarchicky strukturovány, kdy jedna třída je potomkem jedné či více jiných tříd (třída tedy dědí vlastnosti jiné třídy). Jazyk PHP nepodporuje vícenásobnou dědičnost, ta je nahrazena, obdobně jako v jazyce Java, možností implementovat variabilní množství rozhraní. Další z vlastností objektově orientovaného paradigmatu je polymorfismus. Tato vlastnost umožňuje pracovat s různými objekty stejným způsobem a to díky implementaci stejného rozhraní v těchto objektech (Objektově orientované programování – Wikipedie, 2016).
- Reflektivní paradigma – umožňuje programu zjistit svou syntaktickou strukturu a chování a za běhu ji upravovat. Toho se využívá například při vzdáleném zpracování. Tím je například umožněno zavolat metodu třídy pomocí textového řetězce, v němž je uveden název této metody. Dále lze vyhodnotit textový řetězec jako blok kódu a jiné obdobné funkce (Reflection (computer programming) - Wikipedia, the free encyclopedia, 2016).

I když jazyk PHP podporuje všechna výše uvedená paradigmata, není vždy nutné je všechna využívat v jediném projektu. Vývojář se může vydat cestou imperativního vývoje, aniž by využíval objektového návrhu. Tím lze dosáhnout snadnějšího pře-

sunu vývojáře z klasických strukturovaných jazyků na jazyk PHP. Stejně tak i vývojáři ovládající objektový návrh pod jinými jazyky mohou bez větších obtíží začít používat objektový návrh v jazyce PHP, který je v současné době de facto standardem a je podporován. Jak uvádí Procházka (2012), jazyk PHP je využíván pro skriptování na straně serveru (server-side) tzn., celá PHP aplikace běží na webovém serveru, kde se vygeneruje HTML stránka, která je následně poslána jako odpověď zpět klientovi. Základní princip fungování je znázorněn na Obr. 1.



Obr. 1 Princip komunikace klienta a webového serveru

Klient prostřednictvím nějaké akce (například kliknutím na odkaz nebo odesláním formuláře) v právě zobrazované HTML stránce vyšle HTTP požadavek na webový server. Webový server následně vyhodnotí tento požadavek. Pokud se jedná o požadavek nad PHP skriptem, je spuštěn interpret PHP, který požadavek zpracuje. Je-li to potřeba, může k tomu využívat dalších služeb, jako jsou například e-mailové servery, databázové servery, soubory a jiné služby. Celá stránka se tedy skládá dynamicky pomocí PHP skriptu. Na webovém serveru se tedy nenachází žádné statické stránky HTML, ale jsou generovány pomocí konkrétních PHP skriptů. Takto vytvořená HTML stránka je poslána jako odpověď zpět klientovi, který si ji může zobrazit například pomocí webového prohlížeče (Doyle, 2010).

Základní vlastnosti a schopnosti jazyka PHP jsou generování dynamických webových stránek, operace se soubory na straně serveru, získávání a zpracování formulářových dat, práce s cookies, práce s databázemi (MySQL, PostgreSQL, Oracle, atd.), řízení přístupu uživatelů, šifrování dat, výstupy v podobě HTML souborů, obrázků, PDF, XML a jiných. Dalšími výhodami je otevřený zdrojový kód a rozsáhlá komunita uživatelů (W3Schools Online Web Tutorials, 2016). Jedná se však o jazyk interpretovaný, tudíž postrádá rychlost a svižnost jazyků kompilovaných. Je také nezbytné zabránit přístupu nepovolaným uživatelům k serveru, na kterém se nachází PHP skripty, aby nebylo možné je číst nebo s nimi manipulovat (Linux Software, 2016).

Aktuálně je jazyk PHP podle webu w3techs (Historical trends in the usage of server-side programming languages, April 2016, 2016) nejméně dva roky na první příčce popularity programovacích jazyků na straně serveru pro webové aplikace s dominantními 82 %, za ním následuje ASP.NET se 16 %, přičemž je na těchto webech použito v 98,5 % PHP verze 5. Nejaktuálnější verze PHP 7 vydaná 3. 12. 2015 (PHP: Hypertext Preprocessor, ©2001-2016) je momentálně využívána v 0,4 % z celkových 82 %.

2.1 PHP frameworky

Pokud si programátor zvolí pro vývoj své aplikace jazyk PHP, může se vydat dvěma cestami. První možností je použití čistého jazyku PHP, druhou je použití některého z řady frameworků, které jsou na jazyce PHP postaveny.

Framework lze popsat jako strukturu nástrojů sloužící pro vývoj softwarových aplikací pro konkrétní platformu. Tím umožňuje vývojáři vytvářet aplikace rychleji, efektivněji, přehledněji a bezpečněji. Framework tak může vývojáři poskytnout sady tříd, funkcí, knihoven a aplikačních programových rozhraní (API) řešící časté problémy. Vývojář tak nemusí vymýšlet již vymyšlené a může se soustředit přímo na jádro problému (Framework Definition, 2016). Použitím PHP frameworku se uživatel může oprostit od lopotného vytváření vlastního nástroje, který by navíc musel sám řádně odladit a v průběhu času i udržovat, případně refaktorovat. Navíc za každým frameworkem stojí nemalá komunita uživatelů, která taktéž přispívá k jeho zlepšování vlastností, a tak jej pomáhají udržovat stále aktuálním a bezpečným. Většina PHP frameworků využívá architektury MVC (Model-View-Controler), který rozděluje aplikaci na tři vzájemně komunikující části (datový model, grafické uživatelské rozhraní a řídící logika). Nedílnou součástí bývají funkce pro práci s e-maily, soubory, validaci formulářů, práci s databází a abstrakce nad ní (Top 5 Best PHP Frameworks for 2016 To Become a Master Developer, 2016).

V dalším textu budou zmíněny některé z nejpopulárnějších PHP frameworků, jejichž seznamy jsou uvedeny na webech CodeandTuts (Top 6 PHP frameworks for 2016, 2016), Hongkiat (10 PHP Frameworks For Developers - Best Of – Hongkiat, 2016) a Easy Way of App and Website development!! (Top 5 Best PHP Frameworks for 2016 To Become a Master Developer, 2016).

2.1.1 Laravel

Laravel je webový open source PHP framework vydávaný pod licencí MIT. Tento framework je v porovnání s ostatními níže zmíněnými frameworky na trhu celkem krátce. Jeho první beta verze byla vydána 9. 6. 2011. První stabilní verze byla vydána v témže měsíci (History of Laravel PHP framework, Eloquence emerging, 2013). Poslední vydanou verzí je Laravel 5.2 (Laravel - The PHP Framework For Web Artisans, 2016).

Framework je postaven nad MVC architekturou, která umožňuje rychlý a přehledný vývoj aplikací, současně také využívá vlastností objektově orientovaného programování. Klíčovými vlastnostmi Laravelu jsou:

- Programová utilita Composer pro správu balíčků a závislostí mezi nimi.
- Objektově relační mapování (ORM) nazývaný Eloquent, který umožňuje konverzi v obou směrech mezi objekty objektového návrhu a relační databází. Každá z tabulek v relační databázi je reprezentována jedním konkrétním modelem.

- Nástroj pro vytváření SQL dotazů nad databází. Jedná se o sadu tříd, které umožňují zkonstruovat SQL dotaz pomocí návaznosti jednotlivých metod na místo jeho přímého zápisu ve zdrojovém kódu.
- Systém směrování umožňuje generování URI (Uniform Resource Identifier) ze symbolického odkazu a naopak.
- Architektura pro distribuované prostředí REST (Representational State Transfer) umožňující provádění CRUD (Create, Read, Update, Delete) operací pomocí HTTP požadavků. Na základě konkrétní HTTP metody je vyvolána patřičná akce, například pro metodu GET nad URI `http://seznamy.cz/uzivatel/2` jsou vráceny informace uživateli s identifikačním číslem 2. Metoda DELETE pak nad stejným URI provede smazání tohoto uživatele.
- Podporuje autoloading, který automaticky načítá požadované třídy až v případě jejich potřeby.
- Šablonový systém Blade sloužící k vytváření výsledných pohledů z jedné nebo více šablon. Tento šablonovací systém taktéž nabízí sadu vlastních příkazů, jako jsou cykly a podmínky, které jsou pak namapovány do PHP kódu.
- Jednotkové testování je přímo součástí Laravelu v podobě testů PHPUnit.
- Užitečným nástrojem může být i Artisan, což je označení rozhranní příkazové řádky, který mimo jiné nabízí i tvorbu vlastních příkazů.
- U projektů využívajících rozsáhlé databáze lze využít migrací. Jedná se o sadu nástrojů pro verzování databází, který se hodí ke správě databází, kde samotný dump databáze už není vyhovující.

Dalšími výhodami Laravelu je využití automatického stránkování, validace formulářů a stále se rozrůstající komunity uživatelů. Framework se hodí jak pro vývoj menších, tak i rozsáhlých aplikací (Laravel - The PHP Framework For Web Artisans, 2016; Laravel - Wikipedia, the free encyclopedia, 2016).

2.1.2 Symfony

Symfony je webový open source PHP framework vydávaný pod licencí MIT. Dne 18. 10. 2005 byla světu zpřístupněna první verze tohoto frameworku. Aktuálně poslední vydaná stabilní verze je Symfony 3.0. Tento framework je využíván jako součást některých významných projektů. Za zmínku především stojí systém pro správu obsahu Drupal, systém pro správu internetových fór phpBB, výše zmíněný PHP framework Laravel a spousta dalších aplikací.

Symfony je založeno na souboru samostatně znovupoužitelných PHP knihoven. Architektura je postavena na modelu MVC a objektově orientovaném modelu. Mezi základní vlastnosti a funkce frameworku Symfony lze zařadit následující:

- Jednoduchá a automatická instalace na standardních platformách s použitím Composeru.

- Jako ORM využívá externí framework pro objektově relační mapování Doctrine založené na objektově orientovaném přístupu.
- Soustředění se na routing, kdy URL adresa `index.php?uzivatel_id=2` nepřipadá v úvahu, místo toho je použito `/uzivatel/2`.
- Jako šablonovací systém využívá Twig, který oddělí aplikační logiku od pohledu. Pro úpravu výstupů využívá svých vnitřních funkcí a filtrů. Samotná Twig šablona je pak nativně překládána do PHP zdrojového kódu.
- Integrace nezávislé knihovny PHPUnit. Pro jednotkové testování využívá vlastních testovacích tříd.
- Implemetování REST API je možné instalací konkrétního balíčku nebo volbou distribuce Symfony Rest Edition.

U tohoto typu projektu není výjimkou velká komunita uživatelů. Stejně jako Laravel, má vlastní podporu validace formulářů. Autorizaci a autentizaci webové aplikace lze dosáhnout prostřednictvím konfiguračního souboru. Stejným způsobem je řešena i lokalizace. Framework bez problémů zvládá i rozsáhlejší projekty viz dříve zmíněné projekty (Symfony, High Performance PHP Framework for Web Development, 2016).

2.1.3 Zend framework

Zend framework je PHP framework vyvíjený stejnojmennou společností Zend Technologies Ltd. První verze byla poskytnuta veřejnosti v březnu roku 2006. Vydává se pod open source licencí BSD. Zatím poslední vydanou stabilní verzí je Zend Framework 2.5.3. Zend framework je používán pro web zpravodajské rozhlasové a televizní společnosti BBC nebo i jako webová prezentace pro WebEx společnosti Cisco a spoustu dalších projektů.

Zend framework je založen na objektovém návrhu, jehož struktura se skládá z jednotlivých komponent, kdy je každá vázána závislostmi na ostatní komponenty, umožňující použití pouze potřebných komponent typických pro konkrétní projekt. Sami vývojáři frameworku tento koncept nazývají use-at-will návrhem (užij-co-potřebuješ). Zend framework je postaven na MVC architektuře. Současně disponuje těmito vlastnostmi a funkcemi:

- Možnost správy a instalace jednotlivých komponent pomocí již zmíněného Composeru. Další možností je využití aplikace Pyrus (systém pro automatizaci a správu dokumentů).
- Součástí je balík pro testování PHPUnit a možnost využití služby Travis CI zajišťující testování a sestavování projektů na GitHubu.
- Možnost instalace a používání jen některých komponent nebo využití ověřených a odladěných komponent jiných uživatelů napsaných v Zend frameworku, které můžou značně zrychlit vývoj aplikace.
- Nabízí vlastní vývojové prostředí Zend Studio, které lze využít k vývoji nejen v tomto frameworku.

- Stejně jako u Symfony lze volitelně použít ORM framework Doctrine.

Dále framework nabízí šablonovací systém a vlastní Zend layouty, podporu moderních návrhových vzorů jako je správce událostí a dependency injection, podrobnou dokumentaci a rozsáhlou komunitu vývojářů, tvorbu formulářů s podporou HTML5 a jejich validace. Framework je vhodný spíše pro střední a rozsáhlé projekty (Zend Framework, ©2006-2016; Why Choose PHP Zend Framework Development?, 2016).

2.1.4 Phalcon

Zajímavostí Phalconu je, že se jedná o první PHP framework, který je implementovaný v programovacím jazyku C. To Phalconu umožňuje dosahovat mnohem vyšších rychlostí, než je tomu u výše uvedených frameworků, které jsou vyvíjeny hlavně v jazyku PHP. Pro názornost jsou na Obr. 42 uvedeny testy výkonností frameworků podle stránek sitepoint.com. Phalcon je vydáván pod open source BSD licencí. První verze Phalconu byla poskytnuta veřejnosti v listopadu 2012, zatím poslední stabilní verzí je Phalcon 2.1.0.

Phalcon je založen na objektovém návrhu a softwarové architektuře MVC. Základní vlastnosti a funkce jsou následující:

- Nejdůležitější vlastností frameworku je rychlost, která je způsobena tím, že je celý framework zkompilován. V porovnání s ostatními frameworky, které jsou interpretovány.
- K rychlosti přispívá též způsob cachování. Cachovat lze kompletní stránky i konkrétní proměnné a to jak do paměti, tak i do souborů.
- Implementace moderních postupů (Dependency injection, REST, autoloading, směrování, atd.).
- Přístup k databázi je umožněn pomocí implementovaného ORM nebo pomocí objektově orientovaného PSQL (Phalcon Query Language) používajícího standardní dotazy SQL. Pokud se uživatel rozhodne používat NoSQL databázi je mu k dispozici externí databáze MongoDB.
- Phalcon má vlastní šablonovací systém Volt psaný též v jazyku C.
- Pro testování aplikace lze integrovat testovací balíček UnitPHP.

Mimo jiné také podporuje nástroj pro generování formulářů a jejich validaci, vytváření CLI (Command Line Interface) aplikací. Dále může nabídnout aktivní komunitu a dokumentaci v podobě zdarma dostupného PDF souboru nebo webových stránek atd. (Phalcon - High Performance PHP Framework, 2016; Phalcon - nejrychlejší mezi PHP frameworky | SKOUMAL, 2016; PHP Master | PhalconPHP: Yet Another PHP Framework?, 2016).

2.1.5 Nette

Nette je nejpopulárnější český PHP framework, který však není mimo Českou republiku příliš známý. Byl zveřejněn roku 2007. Vydáván je jako open source ve

dvou možných licencích. Buď pod New BSD licencí, nebo GNU General Public Licence (GPL). Je pouze na uživateli jakou licenci si zvolí. Aktuálně poslední stabilní verzí je Nette 2.3.9. Framework je použit k vývoji webových stránek datového úložiště ulož.to, český lokalizace dokumentárně vzdělávacího kanálu National Geographic, webu Česko-Slovenské filmové databáze a spousty dalších webových aplikací.

Nette framework je založena na objektově orientovaném návrhu a MVC architektuře, která je zde nazývána MVP (Model, View, Presenter). Presenter zde nahrazuje Controler. Nette disponuje těmito vlastnostmi a funkcemi:

- Framework je možné získat pomocí aplikace Composer, nebo stažením přímo z webových stránek Nette. Stáhnout lze i předpřipravený projekt ve formě sandboxu.
- Komponentová architektura umožňuje využívat jak celý framework, tak i některé komponenty.
- K testování aplikace lze využít buď externího balíku PHPUnit, nebo Nette komponenty Nette\Test.
- Nette postrádá ORM. Místo toho lze využívat pro komunikaci s databází komponentu Nette\Database, případně implementovanou knihovnu NotORM a nebo databázovou knihovnu Dibi.
- Vlastní ladící nástroj Tracy nebo také známý pod názvem Laděnka.
- Pro produkční server lze stáhnout minimalizovanou verzi skládající se pouze z jediného souboru, což umožňuje rychlejší a snadnější nahrání Nette na server.
- Vlastní šablonovací systém Latte s filtry a automatickým ošetřením výstupů proti XSS (Cross-Site Scripting – metoda zneužívající neošetřených výstupů).

Výhodou může být aktivní česká komunita. Naopak nevýhodou může být, že téměř celý vývoj frameworku má na starosti jediná osoba. Disponuje možností rozlišení vývojového a produkčního serveru, API pro tvorbu formulářů včetně jejich validace, podporou routingu. Nevýhodou je občas nefungující cache, kterou je potřeba ručně promazat (Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework, 2016).

2.1.6 Porovnání PHP frameworků

Před vývojem vlastní aplikace je nutné zvážit, jaké nástroje se rozhodneme používat. Zda vyvíjet aplikaci pouze v jazyku PHP, nebo volit některý z mnoha frameworků. Vhodnější variantou je volba frameworku, protože uživateli nabízí sadu nástrojů na řešení typických problémů, které by uživatel volící čisté PHP pravděpodobně musel řešit sám.

Bude-li uživatel volit framework, se kterým ještě nepracoval, bude jej zajímat zpracování dokumentace případně aktivita a rozsáhlost komunity. Dále by neměl opomíjet ani další atributy a nástroje, které mohou frameworky nabídnout. Někte-

ré z těchto atributů a nástrojů jsou uvedeny v tabulce Tab. 1 porovnávající vybrané PHP frameworky.

- Přítomnost ORM – Nástroj umožňující automatický převod dat mezi databází a programovacím jazykem, kdy jsou data z tabulek chápaná jako objekty. Frameworky podporující ORM mají přehlednější strukturu dotazů.
- Šablonový systém – Podpora kvalitního šablonového systému, který navíc automaticky chrání před XSS útoky je velkou výhodou. Kvalitní šablonový systém nabízí framework Nette (Latte), který je na vyšší úrovni v porovnání s výše popisovanými frameworky.
- Bezpečnost – Bezpečnost je velmi důležitým faktorem k výběru frameworku. Základem je, aby aplikace byla schopna odolávat útokům, jako jsou CSRF, XSS a SQL injection. Výhodou je, pokud se o bezpečnost postará framework automaticky.
- Podpora databázových systémů – Většina webových aplikací pracuje s databází, proto je dobré si zjistit, jestli vybraný frameworku podporuje námi zvolenou databázi.
- Správa cache – Umožňuje ukládání vybraných dat, což vede k dalšímu zrychlení aplikace.

Zde zmíněné parametry jsou jen výčtem z mnoha parametrů, kterými moderní frameworky disponují. Dalším parametrem by mohla být rychlost frameworku, která je vždy horlivě diskutována mezi zastánci toho či onoho frameworku. Pro představu jak jsou na tom některé frameworky z pohledu rychlosti je na Obr. 42 zobrazen graf porovnávající frameworky.

K vývoji vlastních modulů jsem zvolil framework Nette. Důvodů k jeho volbě je hned několik. Jedná se o český, velmi populární framework s českou dokumentací a komunitou, proto by neměl být problém se sháněním vývojářů pro vývoj systému a modulů. Hlavním důvodem volby je, že stávající systém je právě v tomto frameworku implementován. Vývojem v jiném frameworku by vzniklo několik problémů, které by bylo potřeba řešit, například by se musel celý stávající systém přepsat v nově používaném frameworku nebo by musely být nově vzniklé moduly řešeny jako externí. Při variantě externích modulů by navíc musel vývojář buď ovládat oba frameworky, nebo by bylo potřeba více vývojářů, každý na jeden framework. Ani jedna z těchto variant by nebyla z časového ani finančního hlediska vhodná.

Tab. 1 Srovnání vybraných PHP frameworků

	Laravel	Symfony	Zend	Phalcon	Nette
ORM	Eloquent	Doctrine	Doctrine	ano	-
Šablonový systém	Blade	Twig	ano	Volt	Latte
Testování	PHPUnit	PHPUnit	PHPUnit	PHPUnit	PHPUnit
Ochrana před CSRF	ano	ano	ano	ano	ano
Ochrana před XSS	ano	ano	ano	ano	ano
Ochrana před SQL injection	ano	ano	ano	ano	ano
Podpora DBS	PostgreSQL MySQL SQLite MSSQL	PostgreSQL MySQL Oracle Memcache MongoDB MicrosoftBI DynamoDB GraphDB NoSQL CouchDB GemFire Membase	PostgreSQL MySQL SQLite Oracle MSSQL Derby IBM DB2 DBeaver	PostgreSQL MySQL SQLite Oracle Memcache MongoDB MariaDB	PostgreSQL MySQL SQLite Oracle MSSQL
Komunita	velká	velká	velká	rostoucí	střední česká
Dokumentace	rozsáhlá kvalitní	složitá kvalitní	kvalitní	kvalitní	slabší
Caching	ano	ano	ne	ano	ano
Správa session	ano	ano	ano	ano	ano
Určeno pro projekty	všechny	velké	střední velké	všechny	všechny
Rychlost	střední	střední	pomalý	velmi rychlý	pomalý
REST	ano	ano	ano	ano	ne
Rychlost učení	lehký	těžký	těžký	lehký	lehký
Migrace DB	ano	ano	ano	ano	ne
CLI	ano	ano	ano	ano	ne

Zdroj: Comparison of the top 6 PHP frameworks, 2015; Laravel - The PHP Framework For Web Artisans, 2016; Symfony, High Performance PHP Framework for Web Development, 2016; Zend Framework, ©2006-2016; Phalcon - High Performance PHP Framework, 2016; Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework, 2016

3 Stávající aplikace

Stávající informační systém slouží jako podpora zaměstnancům a zákazníkům kavárny k uspokojení jejich potřeb a požadavků. Systém je rozdělen do dvou funkčních částí, veřejné (frontendové) dostupné zákazníkům kavárny a administrační (backendové) sloužící výhradně potřebám zaměstnanců. Mnou navržené moduly se převážně týkají administrační části, proto bude více prostoru věnováno právě této části. Systém literární kavárny je vyvíjen jako webová aplikace. Při vývoji jsou využívány následující technologie a nástroje:

- Skriptovací jazyk PHP verze 5.3.3
- Webový server Apache http Server 2.0
- PHP framework Nette verze 2.0.10
- Šablonový systém pro PHP Latte (součástí Nette frameworku)
- Kaskádové styly (CSS)
- Skriptovací jazyk JavaScript a jeho knihovna jQuery verze 1.8.3
- Relační databázový server MySQL verze 5.1.73
- Nástroj pro správu databáze phpMyAdmin verze 3.5.3

Vzhledem k tomu, že navazuji na již existující projekt, je při vývoji modulů použito výše uvedených nástrojů. Pro vývoj systému je použito dvou serverů, vývojového a produkčního, s operačními systémy CentOS a výše zmíněnými nástroji. Adresářová struktura aplikace je znázorněna v přílohách tohoto dokumentu.

3.1 Uživatelské role a práva

Aplikaci může využívat několik typů uživatelů, které lze rozdělit podle toho, jakou část aplikace používají.

Veřejnou část aplikace využívají zákazníci, které lze rozdělit na neregistrované a registrované. Registrovaným zákazníkem se zákazník stane v případě vyplnění registračního formuláře na veřejném webu aplikace. Tento požadavek na registraci může být pak obsluhou potvrzen prostřednictvím administrační části webu. Po potvrzení získává uživatel svůj Café+ účet a přístup k dalším funkcím.

Administrační část je přístupná výhradně obsluze, která je autentizována prostřednictvím autentizačního formuláře vyžadujícího uživatelův e-mail a heslo do aplikace. Po úspěšném autentizování jsou uživateli přidělena práva odpovídající jeho roli v aplikaci. V závislosti na udělených právech jsou uživateli zpřístupněny odpovídající části aplikace.

3.2 Veřejná část aplikace

Veřejná část slouží hlavně jako prezentace kavárny zákazníkům, proto je graficky příjemnější, než je tomu u administrační části. Lze ji používat jako anonymní nebo

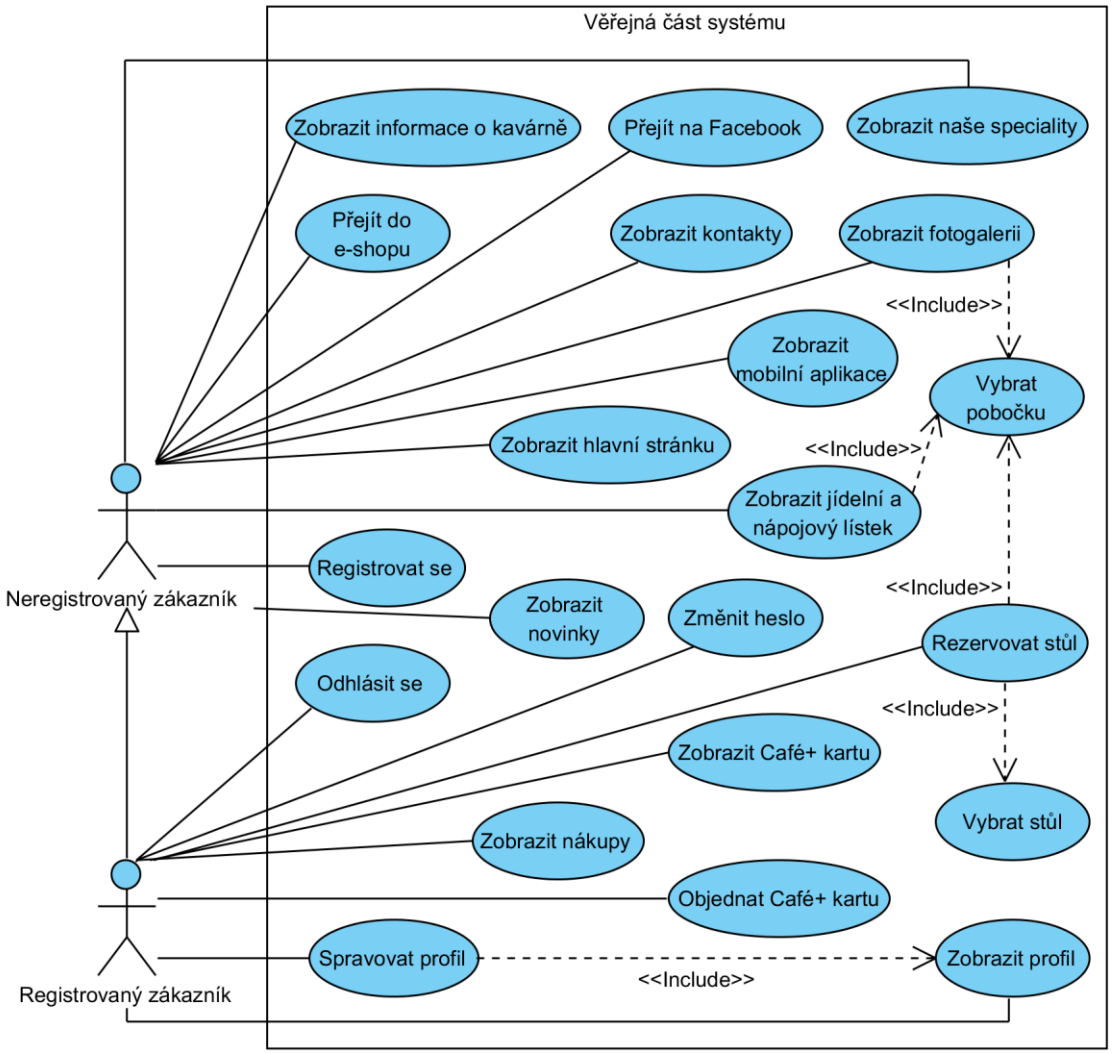
přihlášený uživatel (přihlášení je umožněno pouze registrovaným zákazníkům). Část přístupná pro všechny zákazníky (anonymní i přihlášené) se skládá z následujících modulů:

- Hlavní stránka – Základní informace o poskytovaných produktech rozdělených na stálou a sezónní nabídku, seznamu naposledy přidaných novinek a jednotlivých pobočkách kavárny. Dále obsahuje odkazy na stažení Android aplikace, e-shop a Facebook kavárny.
- Novinky – Podrobnější popis všech novinek dostupných pro jednotlivé pobočky.
- O kavárně, naše speciality – Tyto dvě stránky slouží jako prezentace kavárny a vlastních produktů.
- Jídelní lístek a nápojový lístek – Seznamy nabízených pokrmů a nápojů, včetně cen. Seznamy lze filtrovat podle poboček.
- Fotogalerie – Prezentace fotografií z jednotlivých poboček kavárny.
- Mobilní aplikace – Informace o aplikaci pro mobilní zařízení a možnosti jejího stažení.
- Kontakty – Kontaktní údaje na pobočky, včetně otevíracích dob.

Výše zmíněné moduly jsou dostupné jak pro přihlášené, tak i nepřihlášené zákazníky. Pokud je zákazník vlastníkem Café+ účtu a přihlásí se přes něj do aplikace, zpřístupní se mu navíc moduly Café+ účtu, mezi které patří:

- Nákupy – Seznam nákupů přes Café+ kartu, pokud ji zákazník vlastní nebo v minulosti vlastnil.
- Má Café+ karta – Informace o Café+ kartě zákazníka, pokud ji vlastní.
- Můj profil – Správa účtu zákazníka.
- Rezervace – Umožňuje provádět rezervace, případně je rušit, jménem přihlášeného zákazníka na konkrétní pobočce. Zákazník má také k dispozici přehled o již uskutečněných rezervacích na pobočkách.

Na Obr. 2 je zobrazen diagram případů užití (Use Case diagram) zachycující možná použití veřejné části systému pro konkrétní uživatelské role.



Obr. 2 Případy užití stávajícího systému (veřejná část)

3.3 Administrační část aplikace

Jak již bylo uvedeno v kapitole 3.1 Uživatelské role a práva je administrační část aplikace přístupná pouze registrovaným zaměstnancům obsluhující tuto aplikaci. Administrační část slouží jako podpora zaměstnancům kavárny k usnadnění jejich práce, proto je zde kladen důraz na praktickou stránku věci, aby byl zaměstnanec schopen s aplikací co nejefektivněji pracovat. V následujícím textu jsou zmíněny jednotlivé oblasti aplikace:

- Hlavní stránka – Na tuto stránku je zaměstnanec přesměrován po úspěšném přihlášení do aplikace. Stránka zaměstnanci umožňuje přejít do jedné z jeho pokladen, zároveň také poskytuje informace o výdělcích za aktuální den. Dále jsou zde zobrazeny informace týkající se pohybu peněz vzhledem k jednotlivým pokladnám zaměstnance. Jedna z tabulek zobrazuje čekající transakce mezi pokladnou zaměstnance a pokladnou jiného zaměstnance. Druhá tabulka

zobrazuje výběry z pokladen čekajících ke schválení vedoucím pracovníkem pobočky. Jsou zde také zobrazovány neschválené výběry z pokladny, ale pouze v případě má-li daný zaměstnanec neschválený výběr.

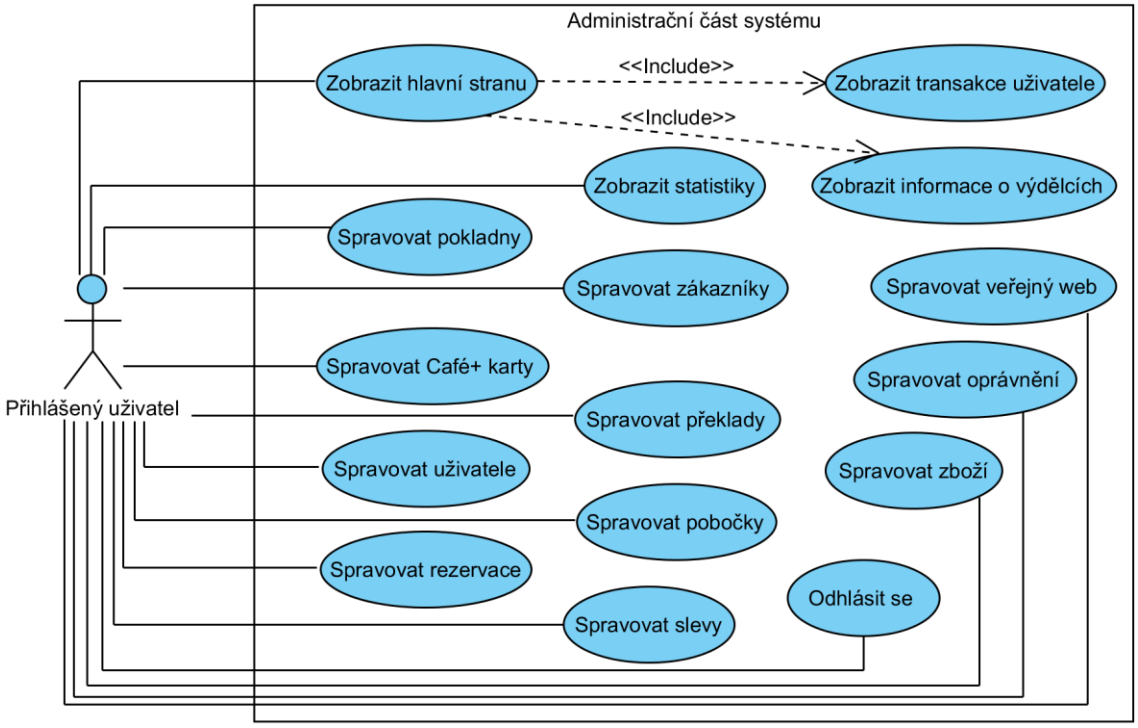
- Pokladny – Tato část zahrnuje správu pokladen. V aktuální verzi aplikace je dovoleno zaměstnanci vlastnit právě jednu pokladnu. Avšak při vývoji dalších modulů aplikace je nutno brát v úvahu i zaměstnance vlastnící více než jednu pokladnu. Tato skutečnost je zapříčiněna tím, že se v aplikaci stále nacházejí aktivní zaměstnanci vlastnící více pokladen. S pokladnami lze provádět základní operace jako je vytvoření pokladny a její přiřazení konkrétnímu zaměstnanci, přesun výdělku mezi pokladnami v rámci stejné pobočky a výběru hotovosti z pokladny nebo platby dodavateli za zboží. Příjmem do pokladny jsou pak platby zákazníků za poskytnuté služby. Veškeré akce v pokladně týkající se změny stavu výdělku jsou logovány i s patřičnými doprovodnými údaji, jako je stav pokladny, časový údaj transakce, množství přidaných/odebraných peněz, typ transakce (výběr za účelem nákupu zboží, výběr hotovosti, příjmy z objednávek) a uživatel provádějící transakci (transakci může provést, mimo uživatele vlastnící pokladnu, i vedoucí pobočky). Výběr hotovosti musí být vždy potvrzen vedoucím pobočky, jinak není možné peníze z pokladny vybrat.
- Statistiky – Soubor několika statistik zobrazující zisky filtrované podle různých kritérií (pobočka, časové období, pokladna, atd.), nejprodávanější zboží podle kusů a výdělků atd.
- Pobočky – Tato oblast je úzce spjata s veřejnou částí aplikace, například místem pobočky, otevírací dobou atd. U každé pobočky lze také nastavit, zda se bude zobrazovat ve veřejné části, administrační části nebo v obou částech. Volba zobrazení v administrační části je určena pouze pobočkám sloužící k testování a pobočkám, které svůj provoz ukončily a musely být uzavřeny, neboť z důvodu uchování údajů není možné tyto pobočky trvale odstranit z aplikace.
- Zboží – Tato sekce zahrnuje veškerou správu nakoupeného a prodávaného zboží na pobočkách. Existuje možnost, že některé pobočky mají oproti ostatním rozdílné zboží nebo jeho cenu, proto je zboží nastavováno pro každou pobočku tak, aby vyhovovalo její potřebě. Veškeré prodávané zboží je rozděleno do několika kategorií, například alkoholické nápoje, kávy, sladké dobroty. Nevýhodou tohoto zobrazení je neustále vyhledávání zboží v kategoriích. Pro usnadnění vyhledávání je k dispozici tzv. rychlá volba, ve které jsou zobrazeny nejprodávanější produkty.
- Uživatelé – Nedílnou součástí aplikace jsou i její uživatelé (zaměstnanci kavárny), kteří s aplikací komunikují. O každém zaměstnanci jsou vedeny základní informace včetně práv.
- Oprávnění – Oprávnění úzce souvisí s uživateli a zabraňuje neautorizovanému přístupu k různým částem aplikace. Oprávnění k přístupu získá uživatel pouze

s konkrétním přístupovým právem. To lze získat dvojím způsobem buď přímým přidělením, nebo přidělením role zahrnující toto právo. Budeme-li chtít vytvořit nového uživatele, který bude mít v rámci kavárny pracovní pozici Obsluha, pak tomuto uživateli přiřadíme následující práva:

  - Správa/vstup do adminu – přístup do administrační část
  - Pokladna – opravňuje k využívání pokladen
  - Správa zboží – evidence, aktualizace a správa zboží pobočky
  - Zkušební účtenka – kontrolní tisk účtenky

- Překlady – Aplikace je implementována tak, že umožňuje používání více jazykových mutací. V současné době to jsou čeština a angličtina. Každý uživatel si může zvolit, jakou jazykovou mutaci chce mít automaticky nastavenou. Toto nastavení je možné kdykoliv změnit. Současně lze využít i dočasného nastavení jazyka a přepínat si jej dle libosti. Při následujícím přihlášení do aplikace bude zase nastaven výchozí jazyk. Jazyková mutace se nezabývá pouze překladem samotného obsahu webu ale i jeho URI adres.
- Slevy – Čas od času kavárna nabízí slevy na vybrané zboží. Akční nabídka je rozdělena do několika typů, jako jsou pobočkové slevy, množstevní slevy a různé slevové kupóny.
- Zákazníci – Registrace zákazníků probíhá přes veřejnou část aplikace. Po odeslání žádosti o registraci zákazníkem je žádost uložena mezi neaktivované účty. Aktivaci účtu provádí vždy pověřená osoba prostřednictvím administrační části aplikace. Registrovaný zákazník získává pak přístup k novým funkcím (viz kapitola 3.2 Veřejná část aplikace) a možnost získat Café+ kartu.
- Café+ karty – Jedná se o karty opatřené EAN kódem umožňující zákazníkům uplatňovat slevy a jiné výhody. Používání karet je v aplikaci evidováno. Na základě toho lze zákazníkovi v budoucnu nabídnout slevy a zboží, které jsou pro něj relevantní.
- Správa webu – Umožňuje prostřednictvím administrační části spravovat veřejnou část. Lze spravovat novinky, fotogalerie poboček, prezentace zboží na úvodní stránce, nápojové a jídelní lístky.
- Rezervace – Poskytují informace o aktuálních rezervacích na pobočkách a možnost spravovat rezervace.

Na Obr. 3 je zobrazen obecný diagram případů užití zachycující vybraná použití administrační části systému. Actor reprezentující registrovaného uživatele je abstrahován, neboť role jednotlivých uživatelů nejsou zcela jednoznačně dány. Například dva uživatelé vykonávající roli obsluhy mohou mít rozličná přístupová práva. Registrovaný uživatel tedy obecně zastupuje uživatele se všemi dostupnými přístupovými právy.



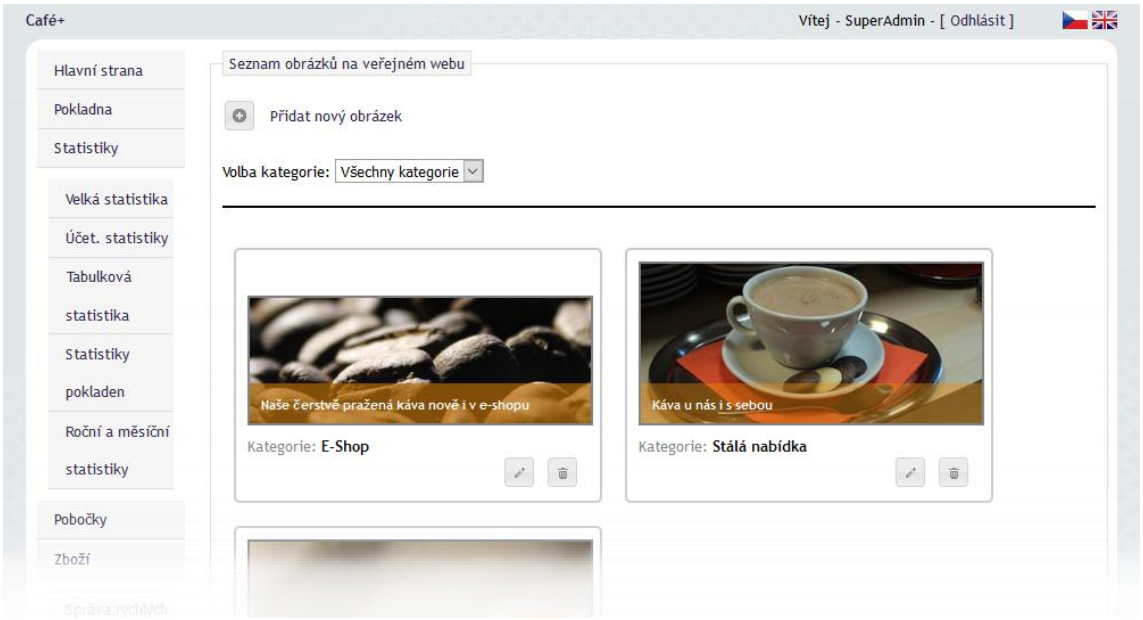
Obr. 3 Případy užití stávajícího systému (administrační část)

3.3.1 Obecný vzhled administrační části aplikace

K vytváření pohledů je použito šablonového nástroje Latte, balíček který je volitelnou součástí Nette frameworku. Základní šablona je pro všechny stránky stejná, mění se většinou jen titulek a obsah těla dokumentu. Aby byly výsledné pohledy pro uživatele příjemnější je použito Kaskádových stylů (CSS) a Javascriptu s knihovnou jQuery. Tyto nástroje umožní lépe strukturovat dokument. Javascript navíc poskytuje funkce podporující interaktivní prvky, které jsou obohaceny o další možnosti vizualizace nebo funkcionalitu.

Barevné schéma stránek je v odstínech šedi. Výraznější barvy jsou použity spíše pro zvýraznění důležitých částí, které mají uživatele upozornit na danou skutečnost. Samotný vzhled stránek je rozdělen do několika vrstev. Horní část stránky je rozdělena na dvě vertikální poloviny. V levé části se nachází odkaz na hlavní stránku aplikace skrývající se pod textem Café+. Pravá část informuje o právě přihlášeném zaměstnanci s možností odhlášení a tlačítky pro změnu jazyka. Další vrstvou je hlavní menu umístěné v levé části s hierarchicky koncipovaným seznamem položek. Hierarchie je znázorněna na Obr. 4, kde položka Statistiky je nadřazená seznamu položek Velká statistika, Účetní statistiky, Tabulková statistika, Statistiky pokladen, Roční a měsíční statistiky. Položky menu jsou přímo závislé na právech uživatele. Uživateli, který nemá právo spravovat zboží, nebude tato položka v menu zobrazena. Stejně tak tomu bude i u zbývajících položek menu. Vpravo od menu se nachází vlastní obsah stránky, který je dynamicky generován na základě zvoleného pohledu. Obr. 4 zachycuje pohled na seznam obrázků pro veřejnou

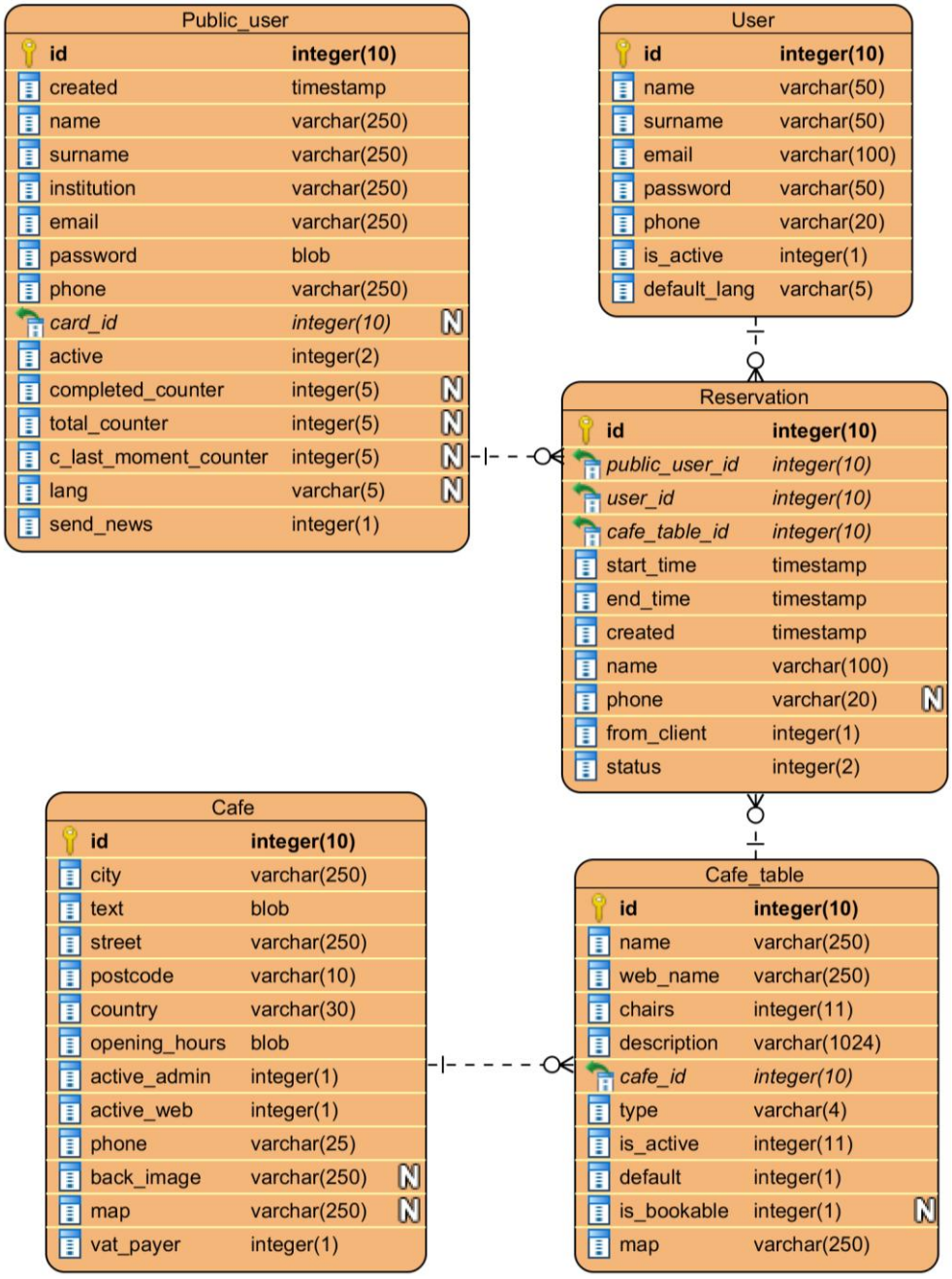
část aplikace, ke kterému se zaměstnanec dostane, pokud v menu zvolí položku Správa obrázků na webu. Stánku pak uzavírá patička, která obsahuje informace o provozovateli aplikace (patička není na Obr. 4 zobrazena).



Obr. 4 Administrativní část aplikace (Správa obrázků veřejné části)

3.4 Databáze

Data, která aplikace využívá, jsou primárně získávána z relační databáze běžící na databázovém systému MySQL. Databáze současného systému v sobě zahrnuje 69 vzájemně propojených tabulek. Vzhledem k takovému počtu tabulek by bylo složité rozumně zobrazit úplný ER diagram databáze. Proto je tento diagram součástí přiloženého média. V dokumentu jsou pouze popsány a znázorněny tabulky přímo související s nově vyvíjenými moduly (Obr. 5).



Obr. 5 ERD vybraných tabulek

Tabulka public_user v sobě uchovává údaje o registrovaných zákaznících používajících veřejnou část aplikace. V tabulce jsou uloženy osobní údaje zákazníka (jméno a příjmení), kontakty (e-mail, telefon), zašifrované heslo pomocí SHA1. Pokud zákazník vlastní Café+ kartu je zde také uvedena reference na tuto kartu do tabulky card. Současně si aplikace uchovává další údaje o zákaznícovi, mezi které patří například časový údaj o založení účtu, preferovaný jazyk, zasílání novinek na e-mail a údaje související s rezervacemi (celkový počet rezervací, uskutečněné rezervace,

neuskutečněné rezervace a rezervace zrušené na poslední chvíli). U účtu může být nastaven příznak aktivity poskytující informaci o tom, zda je účet aktivní nebo deaktivovaný.

Pro uchování informací o uživatelích (zaměstnancích) administrační části slouží tabulka user, ve které jsou obdobně jako v tabulce public_user opět uloženy osobní informace zaměstnance, jeho kontakty, zašifrované heslo, preferovaný jazyk a příznak aktivity účtu. Uživatelé s deaktivovaným účtem nemají přístup do administrační části.

Pro uložení informací o jednotlivých pobočkách kavárny slouží tabulka cafe obsahující údaje o adrese pobočky (stát, PSČ, město, ulice), krátký popis pobočky, telefon na pobočku, plán pobočky a otevírací dobu. Všechny tyto údaje jsou zobrazovány i ve veřejné části. Zbývající atributy jsou určeny pro administraci pobočky. Je-li pobočka plátcem DPH má nastaven pozitivní příznak vat_payer. Pobočka mající aktivní příznak active_admin je viditelná pouze v administrativní části. Naproti tomu pobočka mající aktivní oba výše zmíněné příznaky je viditelná jak ve veřejné, tak i v administrační části aplikace.

Každá pobočka je vybavena stoly a bary. Ty jsou evidovány v tabulce cafe_table. Aby bylo možné stůl identifikovat jak v rezervacích, tak i v prostředí pokladen je mu přiděleno jméno, případně i krátký popis. Dalšími údaji jsou počet míst k sezení, reference na pobočku, ve které se stůl nachází a umístění stolu na pobočce. Pro potřeby rezervace slouží příznaky is_active. Neaktivní stůl nemůže být rezervován a není ani zobrazován uživatelům, vyjma části týkající se správy stolů. Pozitivní příznak is_bookable značí možnost rezervovat stůl.

Pro potřeby rezervací je vedena tabulka reservation, která umožňuje zákazníkovi nebo zaměstnanci rezervovat konkrétní stůl na pobočce na konkrétní časový úsek. Stoly, které mohou být rezervovány, musí splňovat podmínky zmíněné výše v textu týkajícího se tabulky cafe_table. Rezervace obsahuje jméno osoby, na kterou má být rezervace provedena. Tato položka musí být vyplněna, aby bylo umožněno provádět rezervace uživatelů, kteří nejsou evidování v systému. V případě, že rezervaci provádí sám zákazník (skrze veřejnou část aplikace) je nastaven pozitivní příznak značící provedení rezervace zákazníkem. Každá rezervace prochází od svého vytvoření několika procesy, díky kterým může nabývat některého z dále uvedených stavů. Při vytvoření se rezervace nachází ve stavu vytvořená. Za absolvovanou rezervaci se považuje taková rezervace, která je zákazníkem uskutečněna. Pokud zákazník ví, že danou rezervaci z nějakého důvodu nemůže uskutečnit, může ji v aplikaci regulérně zrušit a to nejpozději dvě hodiny před jejím uskutečněním. Dále už je možné rezervaci zrušit pouze telefonicky, taková rezervace získá stav zrušeno na poslední chvíli. Posledním možným stavem rezervace je neabsolvováno. Tohoto stavu rezervace dosáhne, pokud zákazník rezervaci neabsolvuje a zároveň není rezervace zrušena jedním z výše uvedených způsobů. Pokud dojde k takové situaci je zákazníkovi navýšen počet neabsolvovaných rezervací. Zákazníkovi mající tři a více neabsolvovaných rezervací není umožněno si dále vytvářet rezervace.

4 Vývoj modulů

Stávající systém je plně nasazen v provozu a poskytuje zaměstnancům řadu nástrojů, které jim pomáhají řešit rutinní události. Mezi tyto nástroje lze zařadit například pokladní modul, modul správy zboží nebo rezervační modul. Informační systém je neustále ve vývoji a řada těchto nástrojů je postupem času obohacena o nové funkce. Případně je vytvořen nový modul, jedná-li se o soubor funkcí věnující se stejné oblasti. Mnou vyvíjené moduly jsou tedy reakcí na požadavky zákazníka vůči chybějící funkcionalitě v informačním systému.

Celý proces vývoje jednotlivých modulů je rozdělen do několika fází. V prvé řadě je potřeba zjistit, co zákazník od daného modulu očekává a jaké požadavky má splňovat. V této fázi probíhá konzultace se zákazníkem a uživatelem, která vede ke stanovení požadavků na konkrétní modul nebo vylepšení.

Po stanovení požadavků je vytvořen návrh modulu, který je zákazníkovi představen v podobě diagramu případů užití. Následná konzultace nad tímto návrhem umožní doplnění případných nově vzniklých požadavků, které nebyly doposud brány na zřetel, tak aby návrh odpovídal požadavkům a potřebám uživatelů.

Další fáze se už zabývá implementací modulu a navázání na stávající systém. K vývoji nových modulů jsou používány stejné nástroje, jako tomu bylo při vývoji stávajícího informačního systému. Důvodem jejich použití je zachování jednotnosti systému jako celku. Hlavní nástroj používaný při vývoji je PHP framework Nette a databázový systém MySQL. Pro zobrazení uživatelského rozhraní slouží šablonový systém Latte, který se stará o vygenerování HTML stránky. Ke zvýšení uživatelské přívětivosti modulů je v některých situacích využito skriptovacího jazyku Javascript a jeho knihovny jQuery. Implementace a navázání na systém probíhá v prostředí vývojového serveru. Veškeré vyvíjené moduly jsou zanášeny přímo do systému. Nejedná se tedy o samostatné moduly komunikující se stávajícím systémem, ale o moduly vnitřně spjaté a závislé na stávajícím systému. Prvním krokem při implementaci je vytvoření všech potřebných tabulek ve stávající databázi. U některých modulů je navíc potřeba provést i změny stávajících tabulek. Aby bylo možné tyto tabulky používat v aplikaci, je nutné pro každou z tabulek vytvořit odpovídající model. Po těchto krocích je databáze připravena na spolupráci s aplikací. Zbývá pouze vytvoření presenterů, aplikační logiky zpracovávající požadavky uživatelů a poskytování informací pohledům tvořených pomocí šablon systému Latte. Výsledný pohled poskytovaný uživateli je pak tvořen pomocí zmiňovaných šablon, kaskádových stylů (CSS) a případně funkcí Javascriptu.

Systém s nově implementovaným modulem je předložen uživateli, kterým je odzkoušen. Dojde-li k nejasnostem v použití modulu nebo vzniku nových požadavků ze strany zákazníka, provádí se reimplementace. Tento cyklus se provádí do doby, než jsou všechny tyto nedostatky vyřešeny.

V této chvíli je systém s novým modulem připraven k ostrému použití. Provede se tedy přenos do reálného provozu (na produkční server). Po nasazení je systém používán a jeho případné problémy jsou řešeny přímo během provozu systému.

5 Analýza a požadavky na moduly

5.1 Nástěnka

Nástěnka je nástroj sloužící k uchovávání a zveřejňování informací pro konkrétní skupinu uživatelů. Informace jsou na nástěnku přidávány prostřednictvím poznámek. Nástěnka může do jisté míry zároveň suplovat jakousi interní offline komunikaci mezi uživateli systému. V současném systému není taková funkcionalita vůbec implementována, ať už se jedná o komunikaci uživatelů nebo sdílení prostředků potřebných k výkonu práce.

5.1.1 Analýza a požadavky

Vhodnou formou pro uchování by tedy mohla být nástěnka s příspěvky od uživatelů. Do nástěnky by měl mít možnost přidávat příspěvky každý uživatel administrační části, který k tomu má patřičná oprávnění. Nástěnka by měla být rozumně členěna do kategorií, aby bylo možné příspěvky snáze zařadit a následně vyhledat. Příspěvky mohou být určeny pro všechny pobočky, nebo jen pro vybrané pobočky. Příspěvku je možno přiřadit přílohu v podobě soboru, například soubor PDF s návodem na obsluhu kávovaru, nebo dokumenty týkající se školení či recepty.

5.2 Rezervace a rezervační masky

Rezervace umožňují registrovaným zákazníkům zamluvení stolu na pobočkách literární kavárny skrze veřejnou část aplikace. Zákazníci se mohou registrovat pouze na pobočkách zobrazovaných na veřejném webu a stolech, které lze rezervovat (stoly s pozitivním příznakem is_bookable). Zároveň je možné rezervovat stoly i prostřednictvím administrační části aplikace, které smí provádět pouze zaměstnanci kavárny. Ti však mohou rezervovat stoly jak registrovaným tak i neregistrovaným zákazníkům.

5.2.1 Analýza a požadavky

Před vytvořením této práce byl rezervační systém implementován. Registrační systém disponoval operacemi pro správu registrací. Postupem času však vznikly nové požadavky, které buď systém neumožňoval, nebo operace byly zbytečně složité.

Do budoucna by bylo potřeba, kromě registrace jediného stolu, registrovat i celou kavárnu. V současném systému lze registrovat stoly během celé otevírací doby. Novým požadavkem je zamezení registrace v konkrétním období, například po dobu letních prázdnin, kdy jsou některé pobočky uzavřeny. I přesto se v systému čas od času objevují na toto období rezervace, kterým je nutno zabránit. Jedinou možností jak zabránit nechtěným rezervacím je zarezervování každý den stůl po stolu zaměstnancem. Zde se ovšem naráží na další problém, kterým je maximální počet neabsolvovaných rezervací a následným zablokováním rezervací pro

tohoto uživatele. Aby se zabránilo zablokování, musely by se všechny blokovací rezervace dodatečně potvrzovat jako absolvované. Tato nepříjemná situace by mohla být řešitelná jakousi maskou určující, kdy není možné rezervovat stoly.

5.3 Docházka

Docházka poskytuje zaměstnancům kavárny přehled o odpracovaných hodinách a evidenci jejich pracovní doby. Zaměstnavateli také nabízí přehled o odpracovaných hodinách zaměstnanců a z toho se odvíjející mzdové ohodnocení. S modulem docházky mohou pracovat pouze uživatelé registrovaní do administrační části aplikace. Stávající systém doposud postrádal jakoukoliv formu evidování docházky. Docházka byla evidována formou jiné aplikace, která nebyla s aplikací pro literární kavárny nijak spojena. Jednalo se o dva na sobě nezávislé systémy, které spolu nijak nekomunikovaly. V používané aplikaci na evidenci docházky byli evidováni jak zaměstnanci kavárny, tak i zaměstnanci, pro které je primárně aplikace určena. Tato skutečnost zaváděla do evidencí jistou míru nepřehlednosti, a proto bylo vhodné mít pro tyto dvě skupiny zaměstnanců separátní evidence docházky.

5.3.1 Analýza a požadavky

Nově vytvářená evidence docházky by měla využívat stávajících účtů uživatelů. Hlavním cílem by mělo být provádění a ukončování docházky. Zároveň by zde měla být i možnost ji editovat, pro případ že by došlo k nějakým nesrovnalostem. Na základě zisků by mělo být umožněno přidávat k běžné mzdě zaměstnanců i bonusy ze zisku. Aplikace by také měla disponovat funkcemi pro vyplácení mezd.

5.4 Události

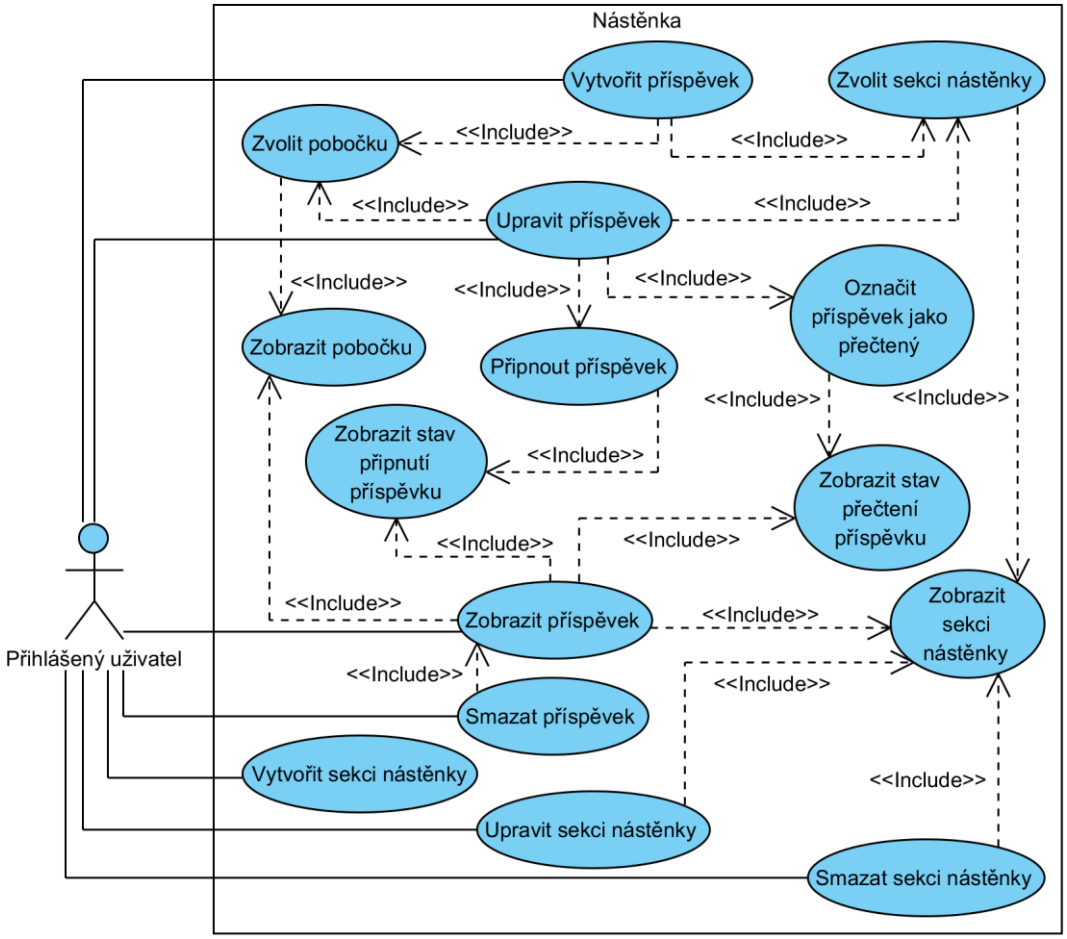
Události jsou modul sloužící pro evidenci a správu akcí na pobočkách. Poskytují zákazníkům informace o konaných akcích a rezervační systém na akce, které jsou kapacitně omezeny.

5.4.1 Analýza a požadavky

Rezervace místa na událost by měla být přístupná jak pro zaměstnance, tak pro zákazníky. Modul by měl vytvářet dva typy událostí, a to události nevyžadující registraci a události vyžadující registraci přes Café+ účet. Zaměstnanci by měli mít možnost smazat událost pouze za podmínky, že daná událost neprobíhá nebo neproběhla. V opačném případě by měla existovat možnost zrušení události, která přihlášené účastníky informuje o jejím zrušení.

6 Návrh modulů

6.1 Nástěnka



Obr. 6 Případy užití nástěnky

Obr. 6 znázorňuje případy užití modulu nástěnky v systému. Přihlášený uživatel musí mít patřičná přístupová práva k používání nástěnky. Uživateli je umožněno vytvářet příspěvky, u kterých si zvolí v jakých pobočkách a sekcích nástěnky se bude daný příspěvek zobrazovat. Vytvořený příspěvek je možné také editovat. Editaci může provést pouze vlastník příspěvku. Existující příspěvky může mazat pouze vlastník příspěvku, případně osoba mající právo na mazání cizích příspěvků.

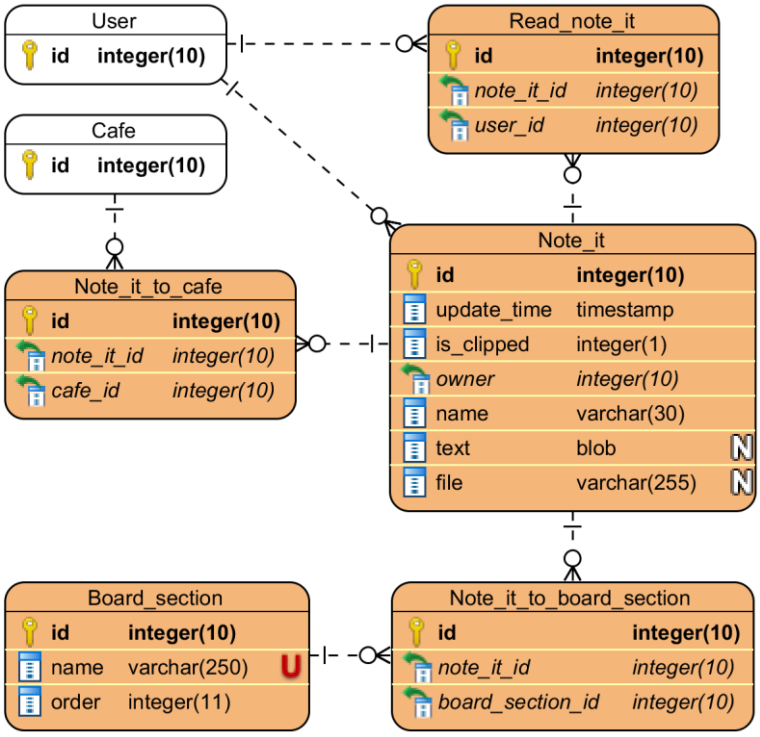
K trvalému uchování veškerých dat je potřeba zřídit v databázi patřičné tabulky. Ty jsou zachyceny v entitně relačním diagramu na Obr. 7. Tabulky user a cafe jsou součástí stávajícího systému, proto jsou zde uvedeny pouze s primárním klíčem a barevně odlišeny (bíle) od nových tabulek (barevně). Stejným způsobem budou zobrazovány i ostatní diagramy.

Za hlavní tabulku lze považovat tabulku příspěvků (note_it), která v sobě uchovává referenci do tabulky user na vlastníka příspěvku, čas vytvoření a popis. Volitelnými atributy jsou text příspěvku a cesta k souboru s přílohou. Pozitivní hodnota atributu is_clipped určuje, zda bude příspěvek zobrazen na nástěnce.

Pro evidenci uživatelů, kteří si přečetli konkrétní příspěvek, je vytvořena asociační tabulka read_note_it nahrazující vazbu M:N mezi tabulkami note_it a user. Těmto uživatelům již nebudou nadále zobrazovány konkrétní příspěvky na nástěnce.

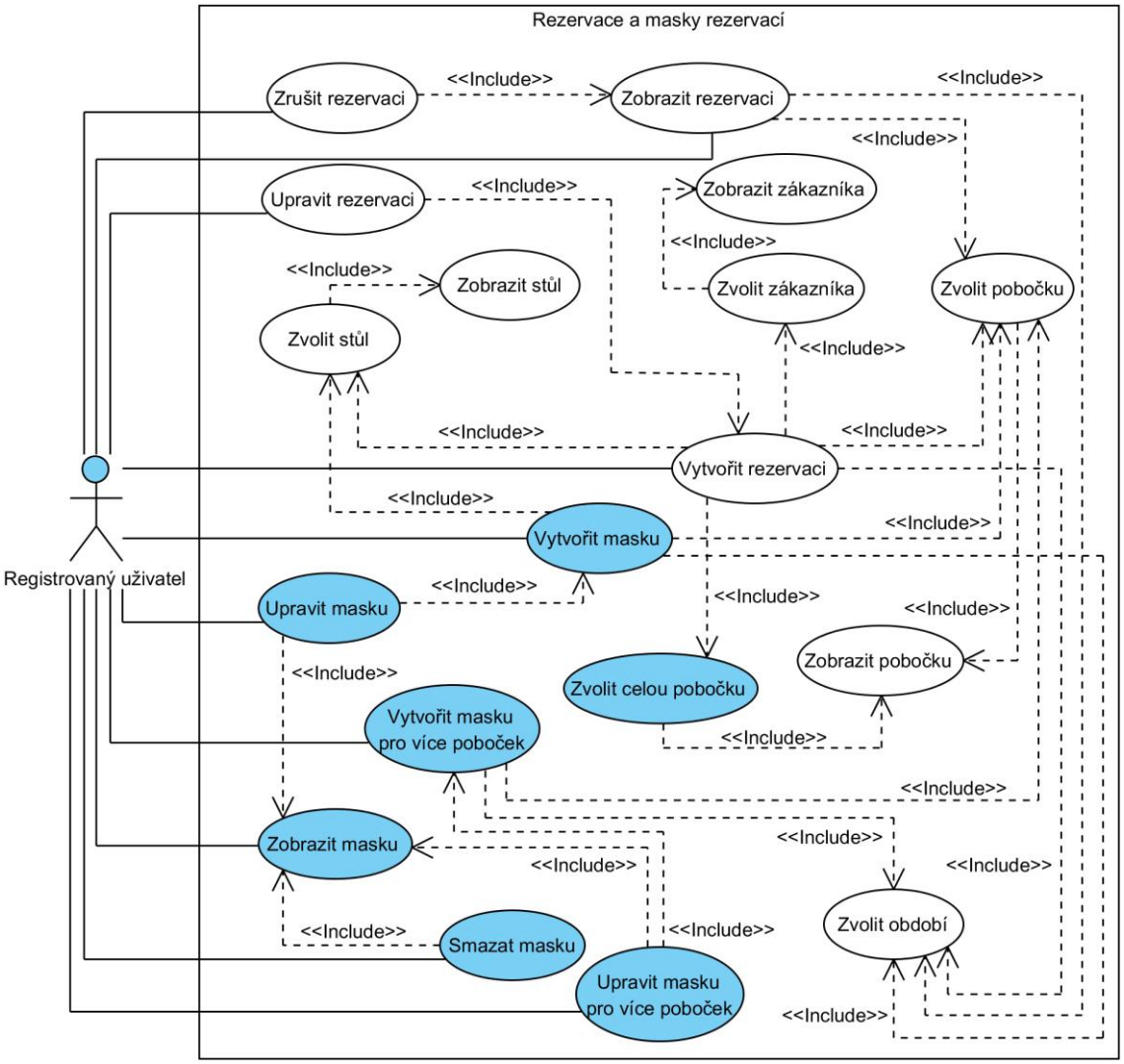
Sekce nástěnky jsou pak podchyceny tabulkou board_section, která v sobě uchovává unikátní název sekce a pořadí v sekcích zobrazovaných na nástěnce.

Provázání poznámky na sekci a pobočku jsou řešena asociačními tabulkami, note_it_to_board_section pro vazby mezi poznámkami a sekcemi a note_it_to_cafe pro vazby mezi poznámkami a pobočkami.



Obr. 7 ERD nástěnky

6.2 Rezervace a rezervační masky



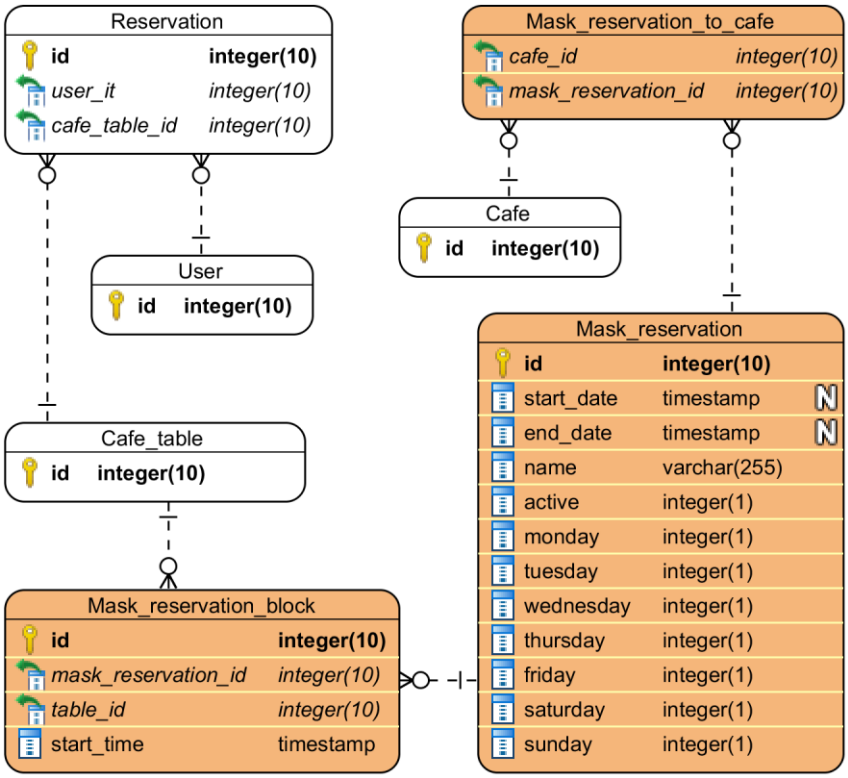
Obr. 8 Případy užití rezervací a masek rezervací

Obr. 8 znázorňuje případy užití modulů rezervací a masek rezervací v administrační části systému. Pro zjednodušení diagramu se předpokládá, že je registrovaný uživatel přihlášen do aplikace. Veřejná část aplikace umožňuje pouze provádění rezervací a jejich mazání. Veřejná část je téměř stejná jako administrační, případně je o některé funkce ochuzena (možnost rezervace celé pobočky) a proto nebude v práci popisována. Administrační část aplikace by měla být rozšířena o možnost rezervování celé pobočky v daném časovém intervalu. Zakázání rezervování se provede vytvořením masek, které budou překrývat rezervace a tím znemožní provedení rezervace v požadované časové úseky a to jak pro jednotlivé pobočky, tak i pro všechny pobočky kavárny.

Ke stávajícím tabulkám, které slouží pro správu rezervací, budou přidány nové tabulky sloužící pro správu masek rezervací. Entitně relační diagram těchto tabulek a na jejich navázání na stávající tabulky je znázorněno na Obr. 9. Stávající tabulka reservation zůstane nezměněna. Nové tabulky týkající se masek rezervací budou využívat referencí na stávající tabulky.

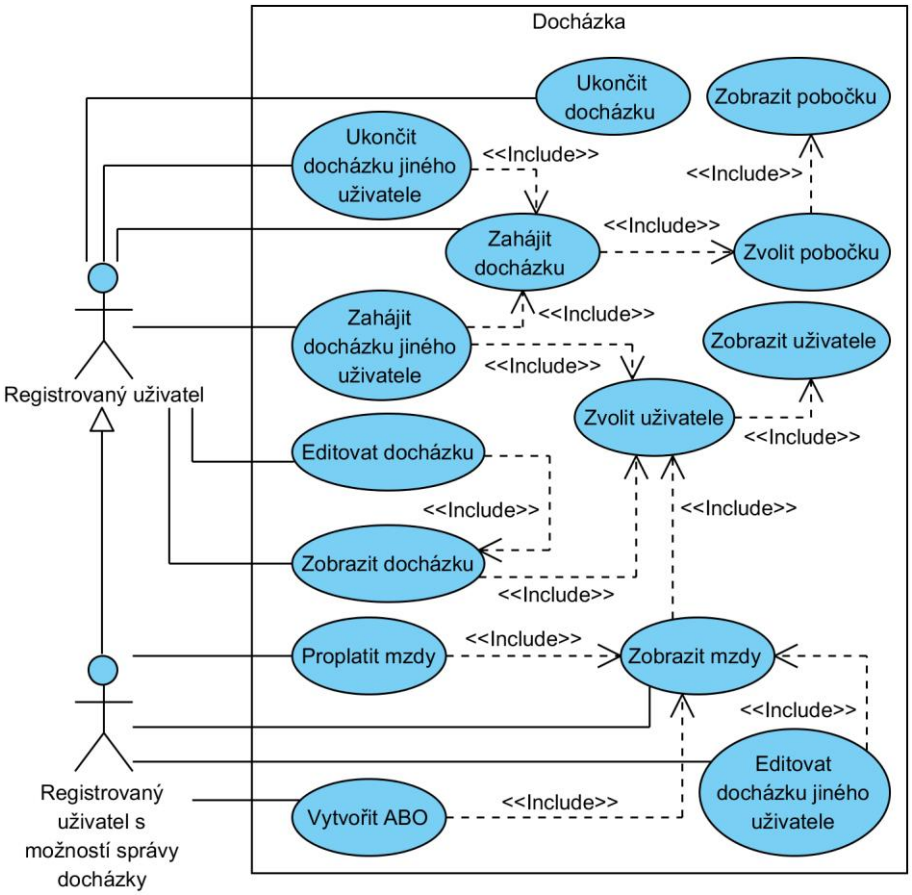
Tabulka masek rezervací (mask_reservation) poskytuje obecné informace o masce, jako jsou platnost masky (strat_date, end_date), pojmenování, příznak určující aktivaci masky a dny v týdnu, pro které je maska platná.

Zbývající dvě nové tabulky jsou asociační a propojují masku s pobočkami (mask_reservation_to_cafe) a se stoly (mask_reservation_block).



Obr. 9 ERD rezervací a masek rezervací

6.3 Docházka



Obr. 10 Případy užití docházky

Na Obr. 10 jsou znázorněny případy užití modulu docházky v administrační části systému. Pro jednoduchost je z digramu vynecháno přihlášení zaměstnance do systému (automaticky se předpokládá přihlášený zaměstnanec). Rozšíření stávající aplikace o modul docházky by měl přinést novou funkcionalitu v podobě evidence docházky a správy vyplácení mezd zaměstnancům. Docházka zaměstnance a s ní spojený výpočet mzdy bude postaven na údajích získaných z časů příchodů a odchodů zaměstnance z práce. Každý přihlášený zaměstnanec by měl mít možnost přihlásit/odhlásit jiné zaměstnance a to pouze na pobočce, na kterou je on sám přihlášen. Na konci období pak mohou zaměstnanci mající patřičná oprávnění provádět výpočty mezd zaměstnanců kavárny.

Modul docházky bude využívat pro svou činnost databázi, proto je potřeba vytvořit a případně pozměnit tabulky v databázi. Entitně relační diagram zachycující změny ve stávajících tabulkách a zavedení nových tabulek je znázorněn na Obr. 11. Ze stávajících tabulek bude modul využívat dříve zmiňované uživatelské účty a pobočky, na kterých mají tito uživatelé pokladny. Uživatelům je potřeba nejprve přidat atribut s číslem účtu, na který bude posílána mzda. Pro potřeby účetnictví je

přidám ještě příznak udávající, zda má uživatel podepsané prohlášení k dani z příjmu. Obdobně jako u uživatelů je i jednotlivým pobočkám potřeba přidělit čísla účtů. V současné době jsou sice mzdy zaměstnancům vypláceny pouze z jednoho účtu, ale pro pozdější účely by bylo vhodné, aby si každá pobočka mohla sama spravovat výplatu mezd.

Z nových tabulek je nejvýraznější tabulka docházky (attendance). V této tabulce se zaznamenávají veškeré příchody a odchody ze zaměstnání. Každý záznam si eviduje referenci na zaměstnance (vlastník záznamu) a pobočku, ve které byl záznam pořízen. Při založení záznamu se uživateli uloží čas a adresa IP, ze které bylo provedeno přihlášení. Při odhlášení uživatele z docházky se rovněž uloží čas a IP adresa. Příznak edited slouží pro evidenci, zda byl záznam pozměněn nebo ne. Pokud se provede změna, může uživatel měnící záznam uvést důvod změny. Poslední příznak (paid) uvádí, zda byla uvedená docházka již proplacena zaměstnanci v rámci mzdy.

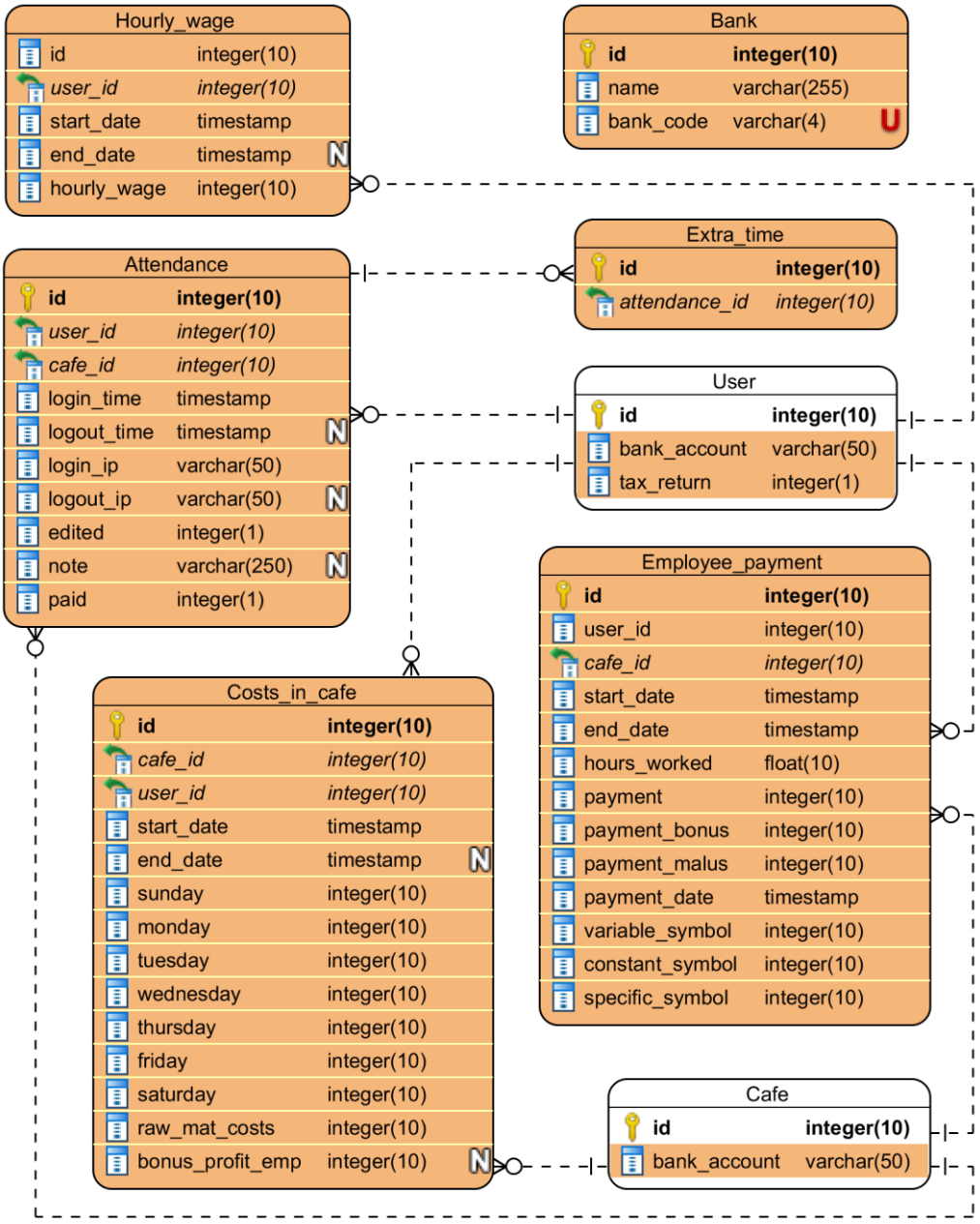
Aby bylo možné zaměstnanci přidávat odpracované hodiny, které si z nějakého důvodu zapomněl přihlásit, byla vytvořena tabulka extra_time. Tyto časy navíc jsou evidovány jako běžné záznamy v docházce, aby je bylo možné odlišit od opravdových běžných záznamů docházky je na tyto záznamy vytvářena reference pomocí zmiňované tabulky.

Pro výplatu mezd je potřeba znát hodinovou mzdu zaměstnance, ta je zaznamenána v tabulce hourly_wage. Vzhledem k tomu, že se hodinová mzda zaměstnance může časem měnit, proto je potřeba uchovávat i historické záznamy. Historie zároveň umožní udržovat správné údaje o výplatě mzdy i po změně hodinové mzdy. Bez historie by nebylo možné zpětně dohledat a přesně určit starší výplaty mzdy.

Zaměstnanci mohou také ke své základní mzdě dostávat bonusy ze zisku. Aby bylo možné vypočítat zisk za konkrétní období v kavárnách, je nutné evidovat i náklady na pobočkách. Náklady jsou několika druhů. Fixní náklady na provoz pobočky jsou rozpočítány jako pevně stanovená částka do jednotlivých dnů v týdnu. Suroviny použité na nabízené služby a produkty se do nákladů promítají procentuálně v atributu raw_mat_costs. Poslední údaj v tabulce se týká hodnoty, kterou dostanou zaměstnanci ze zisku (údaj je uváděn v procentech). Všechny záznamy nákladů se opět ukládají s historií, ze stejného důvodu jako u tabulky hourly_wage.

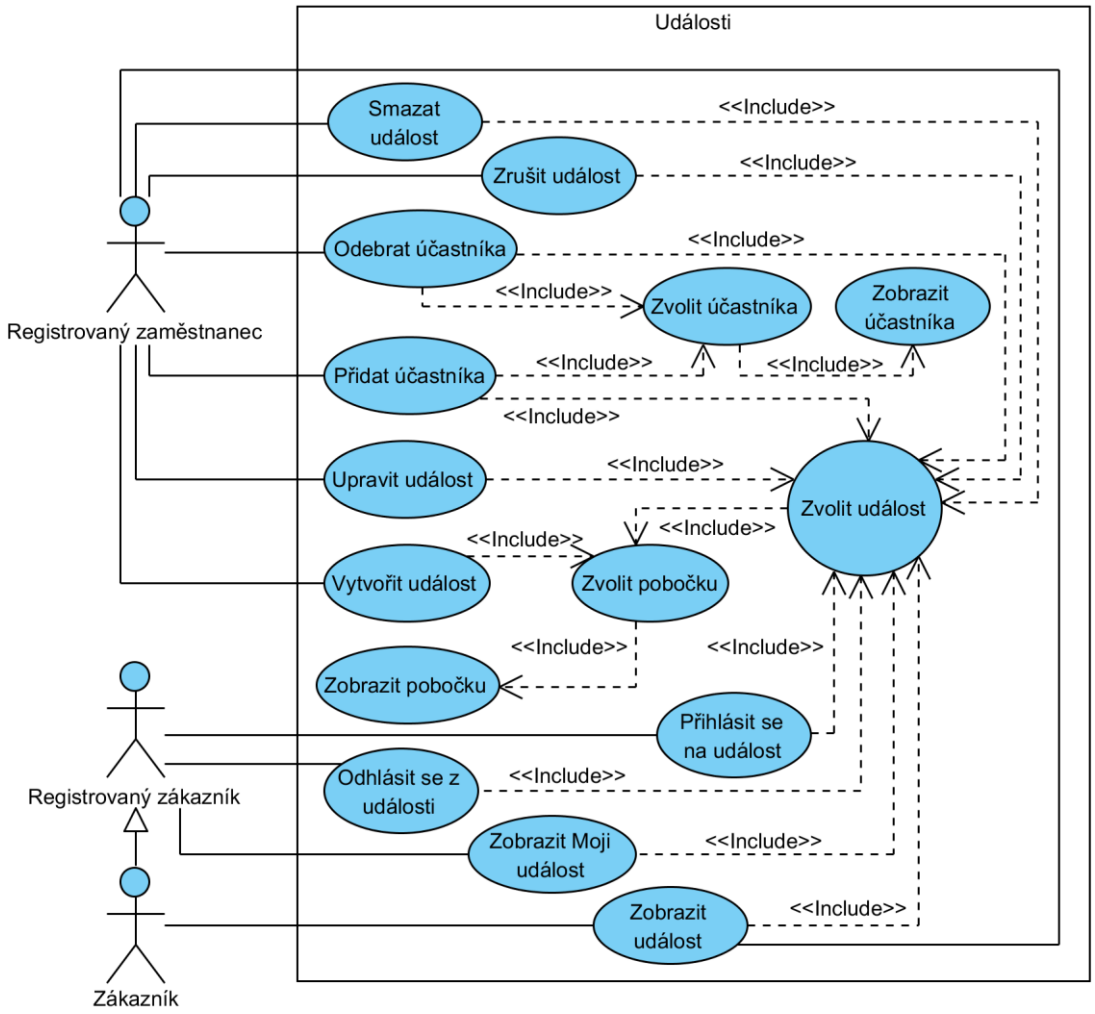
Jakmile je zaměstnanci proplacena mzda, provede se záznam do tabulky employee_paymend. Každý záznam obsahuje reference na zaměstnance a pobočku, z níž byla platba provedena, období, ve kterém byla platba provedena a celkový počet odpracovaných hodin. Uvedeny jsou i údaje týkající se výše mzdy (včetně bonusů a postihů) a platební údaje (variabilní, konstantní a specifický symbol).

Poslední tabulka související s docházkou je seznam vybraných poskytovatelů platebních styků v České republice (Česká národní banka – Česká národní banka, 2016).



Obr. 11 ERD docházky

6.4 Události



Obr. 12 Události: Případy užití

Obr. 12 znázorňuje případy užití modulu události v systému. Modul je používán v administrační i veřejné části aplikace. Ve veřejné části slouží hlavně pro zákazníky. Zde se mohou zákazníci dozvědět o plánovaných událostech. Po přihlášení do vlastního Café+ účtu si mohou zarezervovat místo. Administrační část modulu slouží výhradně zaměstnancům ke správě. Zaměstnanci zde mohou vytvářet události, měnit je, spravovat účastníky událostí nebo dokonce rušit události.

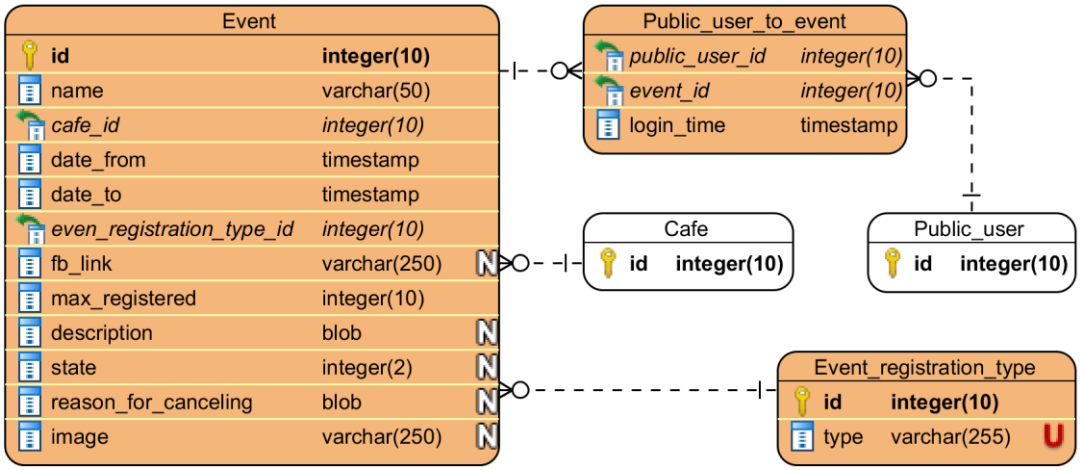
K trvalému uchování je opět použita databáze, do které se vytvoří potřebné tabulky. Tyto tabulky jsou zachyceny entitně relačním diagramem na Obr. 13. Na stávající tabulky cafe a public_user se nově vytvořené tabulky odkazují v podobě cizích klíčů.

Centrální tabulkou událostí je samotná tabulka událost (event), poskytující téměř veškeré informace týkající se událostí. Jsou zde uložena data platnosti události, popisy události, informace o maximálním počtu účastníků a pobočka, na které

se událost koná. K variabilním atributům patří obrázek události a odkaz na stránku Facebooku v případě, že se jedná o událost spravovanou prostřednictvím této stránky. Atribut stav udává stavovou hodnotu události, která může nabývat hodnot aktivní a zrušený. Pokud je nastaven stav zrušený, je vyplněn i důvod zrušení (reason_for_canceling).

Každá událost spadá do určité skupiny událostí, které jsou uvedeny v tabulce event_registration_type. Tyto skupiny jsou dány podle typu registrace. Do registrační skupiny café spadají události, které jsou přímo vázány na Café+ účty zákazníků. Události spravované prostřednictvím stránek Facebooku spadají do skupiny facebook. Do poslední skupiny patří události, u kterých není vyžadována žádná registrace.

Pro uchování údajů o přihlášených zákaznících událostí je zřízena asociační tabulka tvořící spojení mezi událostmi a zákazníky. Při registraci zákazníka na událost se vytvoří v atributu login_time záznam o čase, kdy byl zákazník na událost přihlášen.



Obr. 13 ERD událostí

7 Implementace modulů a navázání na stávající systém

7.1 Nástěnka

K nástěnce, sekcím nástěnky a všem operacím mají přístup všichni registrovaní uživatelé administrační části, proto není potřeba vytvářet žádná nová přístupová práva pro uživatele. Využívá se již existujícího přístupového práva (Přihlášení do adminu) umožňujícího vlastníkovi přístup do administrační části. Výjimkou jsou uživatelé, kteří mohou editovat a mazat cizí příspěvky. Těmto uživatelům je navíc přiděleno právo Mazání cizích příspěvků, které bylo v systému nově vytvořeno.

Aby bylo možné přistupovat k tabulkám popsaným ER diagramem na Obr. 7 bylo potřeba vytvořit modely reprezentující tyto tabulky. Proto byly vytvořeny následující modely v adresáři models (viz příloha B Adresářová struktura aplikace):

- NoteIt
- ReadNoteIt
- BoardSection
- NoteItToBoardSection
- NoteItToCafe

Pokud bychom chtěli zobrazit jednotlivé modely pomocí diagramu tříd, vypadaly by podobně jako na Obr. 44. Všechny zmíněné modely i modely zmíněné dále v textu jsou součástí jmenného prostoru Models a potomky modelu Base, který byl už součástí stávajícího systému. Z diagramu je patrné, že je model Base potomkem třídy Nette\Object. Model Base poskytuje všem svým potomkům rozhraní v podobě metod, z nichž nejdůležitější jsou:

- `save(array $data)` – Metoda vkládající data do tabulky prostřednictvím SQL příkazu INSERT. Návratovou hodnotou této metody je id právě vloženého záznamu.
- `update(array $data, $rowId)` – Metoda editující data v tabulce prostřednictvím SQL příkazu UPDATE. Parametr rowId značí id editovaného záznamu, které je také použito jako návratová hodnota metody.
- `saveOrUpdate(array $data)` – Metoda vkládá data do tabulky, pokud záznam s uvedeným id v datech existuje, provádí se editace tohoto záznamu.
- `get($value, $column)` – Metoda vrací záznam z tabulky, případně všechny záznamy, pokud jsou oba parametry nastaveny na defaultních hodnotách. Pokud je příznak column nastaven pozitivně, je výsledek ukládán do paměti cache.

- `delete($id, $column)` – Metoda vymaže záznam z tabulky, pokud je `column` nastaveno na NULL. Vymazání více záznamů se provede zadáním názvu sloupce do parametru column, parametr id v tomto případě reprezentuje hodnotu daného sloupce. Návratovou hodnotou této metody je počet vymazaných záznamů.

Všechny modelové třídy v sobě musí mít implementovanou metodu `getTableName()` vracející název tabulky, pro kterou je model vytvořen. Současně je vhodné, ne však nutné, vytvoření konstant, které v sobě nesou názvy sloupců dané tabulky (konstanty jsou uvedeny s prefixem C_) a název tabulky (konstanta TABLE_NAME). V každé modelové třídě mohou být ještě uživatelem definované atributy a metody. Tyto metody jsou v případě potřeby volány konkrétním presenterem v aplikaci, který data získaná z metody využívá pro svou potřebu, modifikuje je nebo případně předává do šablony, která je v podobě pohledu zobrazí uživateli. NoteIt poskytuje tyto veřejné metody:

- `getNoteItWithUsers($column, $asc, $fromUserId)` – Vrací seznam příspěvků včetně údajů o vlastníkovi příspěvku (fromUserId). Pokud je parametr fromUserId s hodnotou NULL, jsou vráceny všechny příspěvky nezávisle na uživateli. Atributy column a asc slouží pro seřazení výstupních dat.
- `getNotesIt($userId, $boardSectionId)` – Vrací všechny připnuté (zobrazitelné) příspěvky v dané sekci nástěnky. Parametr userId určuje, zda se jedná o příspěvky zobrazitelné jedním konkrétním uživatelem nebo všemi uživateli (pro volbu všech uživatelů je hodnota atributu nastavena na NULL).

BoardSection poskytuje tuto veřejnou metodu:

- `getNoteItCountInBSec($userId, $onlyClipped)` – Vrací sekce nástěnky včetně počtů připnutých/všech příspěvků. Volba varianty je dána hodnotou příznaku `onlyClipped`. Data jsou zobrazována pro konkrétního uživatele.

NoteItToBoardSection poskytuje tyto veřejné metody:

- `getBoardSectionForNoteIt($noteItId)` – Vrací všechny sekce nástěnky, na kterých se vyskytuje uvedený příspěvek.
- `deleteNoteItToBoardSectionRelation($noteItId)` – Smaže všechny záznamy v tabulce s definovanou poznámkou.

NoteItToCafe poskytuje tyto veřejné metody:

- `getCafeForNoteIt($noteItId)` – Vrací všechny pobočky, pro které je definovaný příspěvek určen.
- `deleteNoteItToCafeRelation($noteItId)` – Smaže všechny záznamy v tabulce s definovanou poznámkou.

Pro správu nástěnky a zpracování požadavků uživatelů týkajících se nástěnky bylo potřeba vytvořit presenter (BoardPresenter), který byl vytvořen v modulu administrace (adresář AdminModule) v adresáři presenters (viz příloha B Adresářová

struktura aplikace). Stejně jako modely, tak i všechny presentery jsou potomky základního presenteru. Presentery administrační části dědí své vlastnosti od presenteru AdminBasePresenter, který je uložen ve stejném adresáři jako jeho potomci. AdminBasePresenter vychází z BasePresenteru. BasePresenter je společně s ErrorPresenterem uložen v adresáři odděleném od ostatních (cafeplus.cz/app/presenters). Z tohoto základního presenteru dědí taky PublicBasePresenter, který je předkem všech presenterů ve veřejné části aplikace. BasePresenter je tedy pro všechny presentery aplikace předkem. Sám je potomkem Nette\Application\UI\Presenter.

Pro všechny společný BasePresenter poskytuje svým potomkům veřejné metody, jako jsou:

- `getLang()` – Vrací aktuálně používaný jazyk aplikace.
- `getModels()` – Vrací ModelLoader sloužící k automatickému načítání modelů aplikace.
- `flashUnknownError($log_msg)` – Flash zpráva o neznámých chybách. V GUI aplikace se zobrazuje ve středu aplikace v oranžově podbarveném poli.
- `flashWarning($message)` – Flash zpráva informující o známé chybě. V GUI aplikace se zobrazuje v horní části v červeně podbarveném poli.
- `flashSuccess($message)` – Flash zpráva informující o úspěšném provedení operace. V GUI aplikace se zobrazuje v horní části v zeleně podbarveném poli.
- `tr($message)` – Vrací překlad zprávy do aktuálně používaného jazyka.

Jak bylo zmíněno výše, bylo potřeba implementovat nový presenter starající se o správu nástěnky. Nástěnka také ovlivňuje stávající hlavní stránku, neboť je zde umístěna prezentační část nástěnky. Diagram tříd popisující nový BoardPresenter a editovaný HomepagePresenter je znázorněn na Obr. 48. V třídě HomepagePresenter jsou uvedeny pouze nové a editované metody týkající se nástěnky.

7.1.1 Přidání nového příspěvku

Pro přidání nového příspěvku bylo potřeba nejprve vytvořit formulář, který provede uživatele jeho vytvořením. Pomocí metody pro vytvoření komponenty `createComponentAddNoteItForm($name)` se vytvoří formulář, který si od uživatele vyžádá potřebné povinné (název příspěvku) a nepovinné údaje (obsah příspěvku, souborová příloha, pobočky a sekce nástěnky). Aby mohl uživatel přidat příspěvek do konkrétních poboček, musí zde mít aktivní pokladnu. Proto se nejprve z databáze zjistí, na kterých pobočkách má uživatel aktivní pokladny a ty mu jsou poté nabídnuty k volbě. Do sekcí nástěnek můžou přispívat všichni uživatelé administrační části. Ty není nutno nijak omezovat a jsou uživateli zpřístupněny všechny. Před odesláním formuláře se ověřují všechny vložené údaje. Například velikost vkládaného souboru do přílohy je limitována na hodnotu 5MB. Pokud je odesílaný formulář validní zavolá se metoda `addNoteItSubmitted($form)`,

zde se provede uložení dat do databáze a případné uložení souboru přílohy. Aby nedocházelo ke kolizím ukládaného souboru, opatří se prefixem. Tento prefix se skládá z aktuálního času v podobě časového razítka, ke kterému je pomocí zřetězení ještě připojena náhodná hodnota generovaná funkcí `rand()`. Takto spojené řetězec jsou ještě zašifrovány hašovací funkcí `md5()`. Po dokončení operací je uživatel o výsledku informován v podobě flash zprávy. Pokud dojde k úspěšnému uložení, je stránka přesměrována na seznam příspěvků.

Pro vykreslení stránky s formulářem je zavolána metoda `renderAdd($toBoardSection)`, která zprostředkuje vykreslení stránky pomocí šablony add.latte. Ukázka formuláře přidávající nový příspěvek je na Obr. 14.

Obr. 14 Nástěnka: Formulář pro přidání nového příspěvku

7.1.2 Editace příspěvku

Stejně jako u přidání nového příspěvku je i u editace nutné nejprve vytvořit formulář, skrze který bude možné vybraný příspěvek editovat. Formulář se vytvoří pomocí metody `createComponentEditNoteItForm($name)`. Oproti formuláři pro přidávání nového příspěvku přibyly ještě tři nové prvky. Prvním je skupina tří radio buttonů (přepínačů) umožňující volbu operace nad souborem přílohy. Pokud uživatel nemá v úmyslu ukládat přílohu, zvolí variantu Žádný soubor, pro pone-

chání stávajícího souboru Aktuální soubor. Volba Nový soubor zviditelní pole pro nahrání souboru, které bylo doteď skryto pomocí Javascriptu. Volbou jedné ze zbývajících variant se pole opět skryje. Zbývající dvě nové položky pak ovlivňují příznak připnutí příspěvku a označení příspěvku jako přečtený. Omezení na vkládané údaje jsou stejná jako u přidání nového příspěvku.

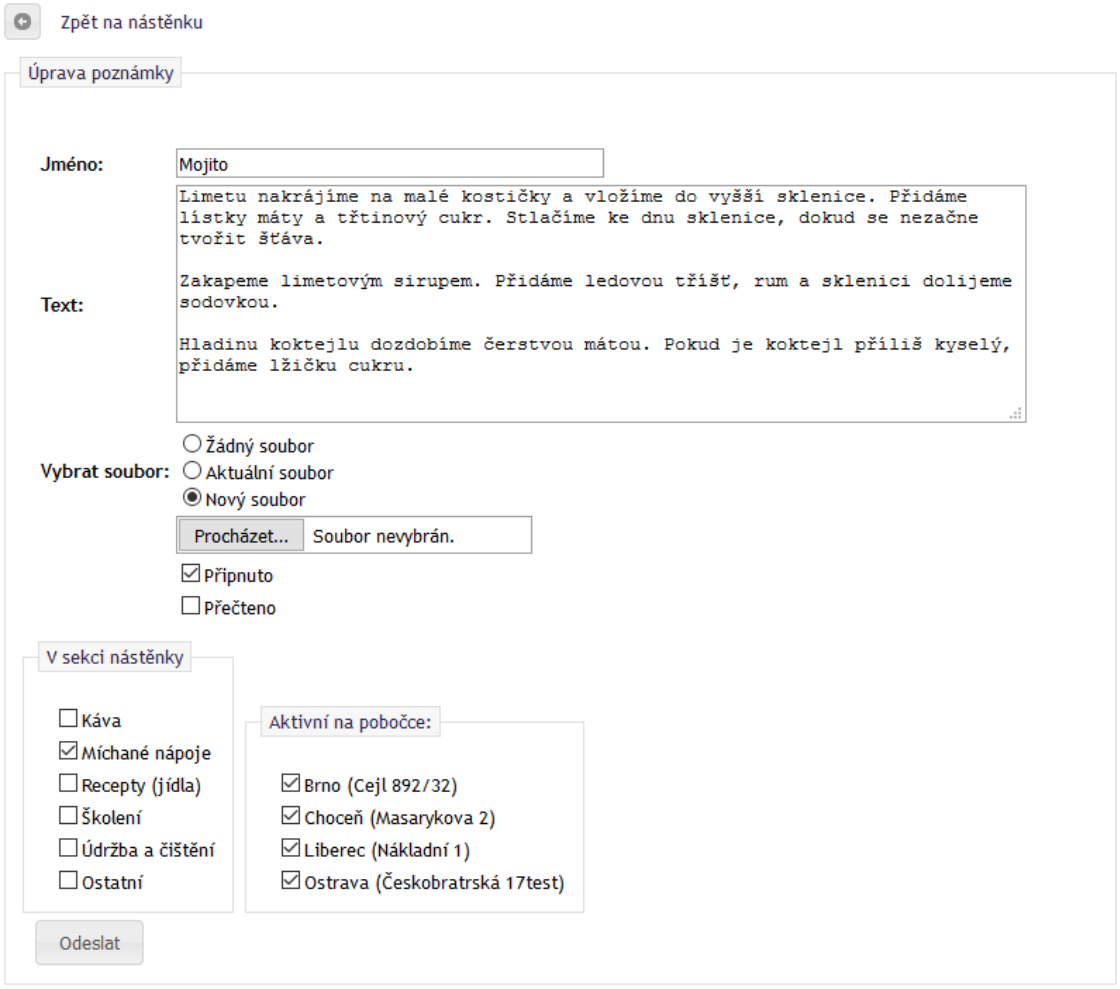
Formulář musí mít před zobrazením uživateli ještě vyplněny jednotlivé prvky. Toho lze dosáhnout získáním údajů z databáze skrze model NoteIt a jeho metodu `get()` s parametrem udávající hodnotu id editovaného příspěvku. Získaná data se vloží do jednotlivých prvků. Například vložení textového pole do formuláře včetně nastavení defaultní hodnoty vypadá následovně:

```
$form->addText(\Models\NoteIt::C_NAME, "Name:")
  ->addRule($form::MIN_LENGTH, 'Poznámka musí mít nejméně %d znak', 1)
  ->addRule($form::MAX_LENGTH, 'Max. počet znaků může být %d', 250)
  ->setAttribute("class", "noteItName")
  ->setDefaultValue($defaults[\Models\NoteIt::C_NAME]);
```

Na prvním řádku se do formuláře přidá textové pole, následující dva řádky přidávají validační pravidla. V tomto případě pravidla kontrolují minimální a maximální délku vkládaného textu. Další řádek se zabývá přidáním atributu class. A poslední řádek nastavuje textovému poli defaultní hodnotu, která bude zobrazena v textovém poli při zavolání formuláře.

Po odeslání formuláře se zavolá metoda `editNoteItSubmitted($form)` starající se o správné uložení údajů. Před samotným přepsáním údajů je potřeba provést několik kroků. Prvním krokem je získání aktuálních dat. Z tabulky note_it nás hlavně zajímá údaj týkající se přílohy (zbývající atributy se pouze aktualizují). V případě, že je v atributu přílohy uložena příloha (cesta k příloze) a nebyl označen radio button Aktuální soubor, je potřeba tento soubor ze serveru vymazat a nahradit jej novým souborem nebo prázdnou hodnotou. U tabulek note_it_to_cafe a note_it_to_board_section se provede vymazání všech záznamů a podle zvolených checkboxů se zadají nové záznamy. Tato varianta je mnohem jednodušší, než porovnávání starých a nových hodnot a na jejich výsledcích provádět aktualizace záznamů. Pozitivní hodnota příznaku udávající přečtení příspěvku uživatelem vytvoří nový záznam v tabulce read_note_it. Hodnota negativní odstraní odpovídající záznam. Po dokončení metody je uživatel informován o výsledku flash zprávou. Při úspěšné editaci dojde k přesměrování na seznam příspěvků.

O vykreslení formuláře se stará šablona editNoteIt.latte spolu s metodou `renderEditNoteIt($noteItId)`. Ukázka formuláře editující příspěvek je na Obr. 15.



Obr. 15 Nástěnka: Formulář pro editaci příspěvku

7.1.3 Smazání příspěvku

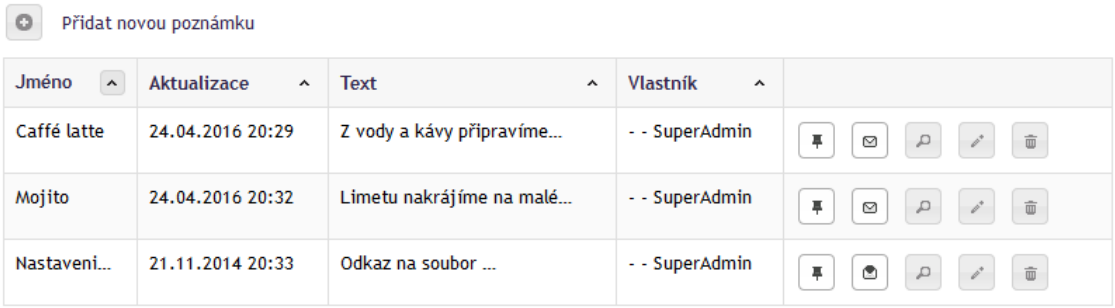
Pro smazání příspěvku slouží metoda, která se řadí mezi takzvané handlery, což jsou metody na zpracování signálů. Pro smazání příspěvku se spouští handler `handleDeleteNoteIt($id)` přijímající ve svém parametru id mazaného příspěvku. Před smazáním záznamu z databáze je nutné nejprve ověřit, zda příspěvek neobsahuje přílohu. Soubor přílohy je také nutno smazat ze serveru. V aplikaci se smazání nachází v seznamu příspěvků. Po provedení smazání se stránka přesměruje znovu sama na sebe.

7.1.4 Seznam příspěvků

Seznam příspěvků poskytuje rychlý přehled o příspěvcích na nástěnce. Hlavním prvkem na stránce je tabulka s příspěvky a operacemi dostupnými pro tyto příspěvky. Příspěvky jsou zobrazeny, jako řádky tabulky, ve sloupcích jsou pak zobrazeny jejich atributy (název, čas poslední aktualizace, ukázka z popisu, vlastník). V posledním sloupci se pak nachází pětice následujících ikon. První dvě pouze gra-

ficky zobrazují příznaky připnutí na nástěnce a přečtení příspěvku. Zbývající tři ikony jsou interaktivní. První z nich vede na stránku s podrobnějším přehledem o příspěvku. Druhá umožní přejít na stránku editující příspěvek. Poslední provede smazání příspěvku. Vzhledem k tomu, že se jedná o destruktivní operaci je po kliknutí na tlačítko zobrazeno dialogové okno požadující potvrzení ke smazání. Až poté je příspěvek smazán.

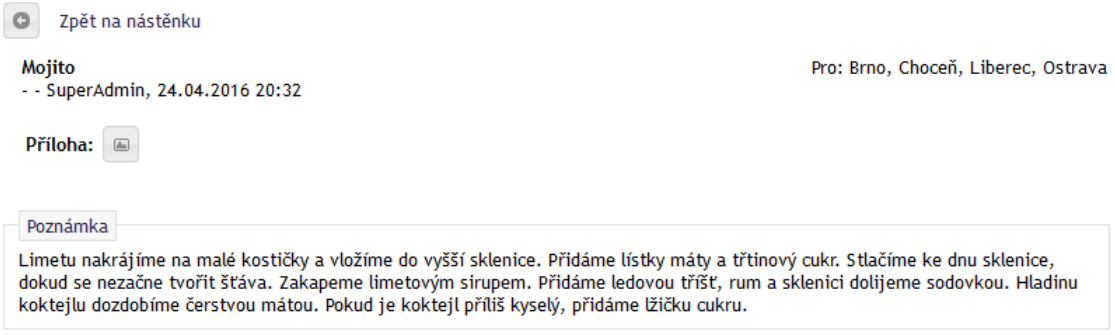
Zobrazení provádí metoda `renderList()`, která získá z databáze všechny potřebné údaje. Ty jsou následně zobrazeny šablonou list.latte. Ukázka tabulky s příspěvky je na Obr. 16.



Obr. 16 Nástěnka: Seznam příspěvků

7.1.5 Přehled o příspěvku

Přehled o příspěvku poskytuje uživateli údaje mnohem podrobněji, nežli je tomu u seznamu příspěvků, oproti tomu neposkytuje téměř žádné funkce. Jediné dvě funkce jsou stažení přílohy (pokud je dostupná) a návrat na stránku nástěnky (obdobné návraty se nacházejí na většině stránek aplikace). Vzhledem k tomu, že není potřeba vytvářet žádný formulář, tak se volá pouze metoda `renderPreviewNoteIt($noteItId)`, parametr noteItId udává, o jaký příspěvek se jedná. Před vykreslením šablony previewNoteIt.latte získá metoda potřebné informace o příspěvku z databáze. Ukázka stránky s přehledem o příspěvku je na Obr. 17.

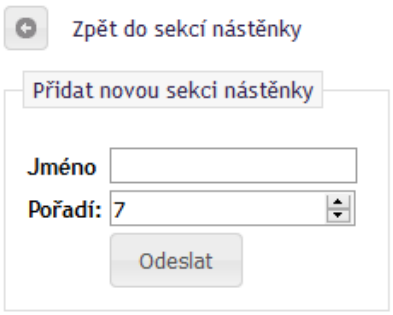


Obr. 17 Nástěnka: Přehled o příspěvku

7.1.6 Přidání sekce nástěnky

Přidání nové sekce nástěnky se provádí přes formulář vytvořený metodou `createComponentAddBoardSectionForm($name)`. Formulář se pokouší uživateli nabídnout následující číslo pořadí, po posledním, které se nachází v databázi, je pouze na uživateli, zda toto pořadí nechá na formuláři nebo jej změní. Toto číslo udává pořadí sekce na nástěnce zobrazené na hlavní stránce. Dvě položky mající duplicitní pořadí jsou pak řazeny abecedně podle názvu sekce. Díky tomu, že jsou názvy sekcí unikátní, lze zajistit vždy stejné pořadí na nástěnce. Další data získávaná z databáze jsou ohledně názvů již existujících sekcí. Ta jsou použita k validaci unikátnosti nově vytvářené sekce. Po odeslání formuláře je zavolána metoda `addBoardSectionSubmitted($form)`, která uloží novou sekci do databáze, po provedení uložení se přesměruje na seznam sekcí nástěnky.

Vykreslení stránky zajišťuje metoda `renderAddBoardSection()` a šablona addBoardSection.latte. Ukázka formuláře je na Obr. 18. Zde si lze povšimnout předvyplněného pořadí pro novou sekci.



Obr. 18 Nástěnka: Formulář pro přidání nové sekce

7.1.7 Editace sekce nástěnky

Editace sekce nástěnky funguje na stejném principu jako již zmíněná editace příspěvku. Formulář pro editaci sekce nástěnky se vytvoří metodou `createComponentAddBoardSectionForm($name)`. Tento formulář je podobný formuláři pro přidání sekce nástěnky jen s tím rozdílem, že se automaticky předvyplní údaje editované sekce. Také validace kontrolující unikátnost sekcí je pozměněna. Pokud by zůstala stejná jako u vytváření, nebylo by možné uložit sekci, u které jsme změnili pouze pořadí. Z databáze se tedy získají všechny názvy sekcí až na právě editovanou, tím se zachová unikátnost a zároveň je umožněno ukládání sekce, u které se provede pouze změna pořadí.

Vykreslení editačního formuláře provádí šablona editBoardSection.latte a metoda `renderEditBoardSection($boardSectionId)`.

7.1.8 Smazání sekce nástěnky

Smazání sekce nástěnky řeší metoda `handleDeleteBoardSection($id)`, která přijímá jako svůj jediný parametr id mazané sekce. Při smazání se kaskádovitě

odstraní záznam o sekci z tabulky board_section a všechna provázání na sekci v tabulce note_it_to_board_section. Pokud by došlo k situaci, že po smazání sekce bude existovat příspěvek, který nemá žádnou sekci, pak je tento příspěvek stále ponechán v databázi. Je pak na uživateli co s takovým příspěvkem udělá. Může jej nechat bez jakékoliv sekce, takový příspěvek nebude viditelný na hlavní stránce, avšak bude jej možné dohledat v seznamu příspěvků a zde s ním dále pracovat. Další možností je editace či ruční smazání příspěvku. Smazání sekce se v aplikaci nachází v seznamu sekcí nástěnky. Jakmile se provede smazání, dojde k přesměrování stránky sama na sebe a tím i k aktualizování seznamu sekcí.

7.1.9 Seznam sekcí nástěnky

Seznam sekcí nástěnky nabízí přehled o veškerých sekcích, do kterých je nástěnka členěna. Seznam je zobrazen v podobě tabulky, kde první část je tvořena informacemi o tabulce (název a pořadí sekce na nástěnce), druhou část tvoří operace nad jednotlivými sekcemi (editace sekce a smazání sekce). U operace smazání dochází k nevratnému odstranění sekce, proto je uživatel požádán o potvrzení akce dialogovým oknem.

Zobrazení stránky seznamu sekcí provádí `renderListBoardSection()`, která na základě řadící komponenty OrderLinks získá data z databáze a poskytne je šabloně listBoardSection.latte k vykreslení. Ukázka tabulky s několika sekcemi je na Obr. 19.

Obr. 19 Nástěnka: Seznam sekcí nástěnky

7.1.10 Nástěnka na hlavní stránce aplikace

Aby měl uživatel rychlejší přístup k příspěvkům a mohl sledovat nové příspěvky na nástěnce, byl zařazen blok s nástěnkou i na hlavní stránku. Nejprve bylo potřeba přidat a upravit některé metody v presenteru pro hlavní stránku (HomepagePresenter). Hlavní úpravy se týkaly metody pro vykreslování šablony hlavní stránky `renderDefault($boardSelectionId)` do jejíž hlavičky byl nově vložen parametr udávající aktuálně zvolenou sekci nástěnky. V případě, že není nastavena žádná sekce, pak bude považována za aktivní první sekce v pořadí udávající atribut order vedený v databázové tabulce u každé sekce. Mimo tento parametr byl vložen do metody ještě obslužný kód pro nástěnku, který lze rozdělit do následujících dvou částí:

- Získání všech sekcí a nepřečtených příspěvků s ohledem na aktuálně přihlášeného uživatele.
- Zobrazení příspěvků aktuálně zvolené sekce v závislosti na pokladnách uživatele. Uživateli jsou zobrazeny pouze příspěvky z poboček, na kterých má uživatel pokladnu.

Po získání všech údajů je možné vykreslit nástěnku, která je vykreslena pouze tehdy, pokud má uživatel alespoň jeden nepřečtený příspěvek. V opačném případě není blok s nástěnkou vůbec vykreslen.

Za předpokladu, že se uživateli nástěnka vykreslí, má k dispozici na hlavní stránce blok nástěnky podobný tomu na Obr. 20. Jednotlivé sekce jsou zobrazeny jako záložky, pod které spadají konkrétní příspěvky. Na záložce za textem si lze povšimnout čísla v červeném poli, které upozorňuje uživatele na nepřečtené příspěvky. Díky tomu získá uživatel přehled o nepřečtených příspěvcích, aniž by musel každou ze sekcí postupně otevírat a zjišťovat, zda je zde nějaký nepřečtený příspěvek. Příspěvky jsou stejně jako v seznamu příspěvků zobrazeny pouze se základními informacemi. Plné zobrazení příspěvku uživatel získá pomocí akce Přehled, která umožní otevření přehledu v dialogovém okně, nebo přesměruje na stránku přehledu známou z dřívějšího popisu. Druhé tlačítko (tlačítko s obálkou) umožňuje nastavit u příspěvku, že jej uživatel četl. Po uskutečnění této akce si systém vyžádá od uživatele potvrzení a v případě kladné odpovědi se příspěvek už nebude nadále zobrazovat na nástěnce hlavní stránky. Aby bylo možné přidávat příspěvky přímo z hlavní stránky aplikace, nachází se vždy pod seznamem příspěvků tlačítko umožňující přidat příspěvek do aktuálně zobrazované sekce nástěnky. V tomto případě se jedná o již uvedenou stránku pro přidávání nových příspěvků, která má navíc předvyplněnu onu aktuální sekci.

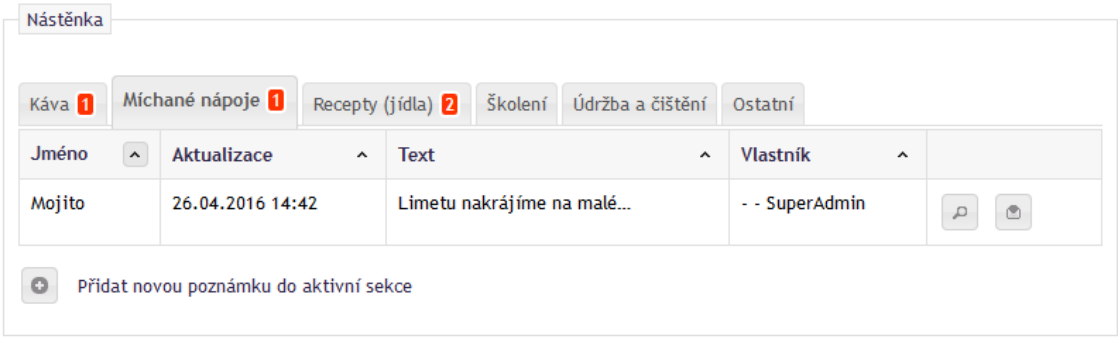
Aby se blok nástěnky zobrazil na stránce, bylo nutno upravit šablonu pro hlavní stránku default.latte. Pro zobrazení stránky v dialogových oknech bylo potřeba do bloku hlavičky v latte šabloně vložit následující Javascriptovou funkci (tato funkce byla už součástí stávající aplikace):

```
enableAjaxDialogWindow(
  'dialog', '.addNoteIt',
  {l}
    modal: true, disabled: false,
    width: 820, position: ['center', 10],
    title: this.title
  {r}
);
```

Po aktivaci odkazu s třídou addNoteIt se místo přesměrování na požadovanou stránku zobrazí dialogové okno s danou stránkou (v tomto případě se jedná o stránku přidávající nový příspěvek do nástěnky). Do dialogového okna není ale vložena celá stránka, vkládá se pouze blok stránky mezi latte značkami `{dialog}`

a `{/dialog}`. Zbývající údaje slouží pouze pro nastavení parametrů okna, jako je šířka okna, pozice okna a údaj, zda se jedná o modální či nemodální okno.

Zbývající úpravy šablony se týkají přidání latte a HTML kódů, které umožňují zobrazení vlastní nástěnky. Jednotlivé sekce jsou klasickým HTML seznamem odkazů, který svou podobu záložek získal pomocí kaskádových stylů. Seznamy položek jsou zobrazeny v tabulce.



Obr. 20 Nástěnka: Blok na hlavní stránce

7.2 Rezervace a rezervační masky

Pro přístup k maskám rezervací a jejich správě mají přístup všichni registrovaní uživatelé, kteří mají právo přístupu do administrační části. Současně k tomuto právu bylo potřeba ještě vytvořit právo Správa masek rezervací. Kombinace těchto dvou práv umožní uživateli vytvářet a spravovat masky rezervací.

Pro umožnění přístupu k tabulkám v databázi se vytvoří modely znázorněné na Obr. 45. Mezi tyto modely patří následující:

- MaskReservation
- MaskReservationBlock
- MaskReservationToCafe

MaskReservation poskytuje tyto veřejné metody:

- `getReservationMasksForCafe($cafeId, $column, $asc)` – Vrací všechny rezervační masky určené pro vybranou pobočku (cefeId), řazené podle zvoleného sloupce (column).
- `getAllReservationsMasksOnlyOneCafe($column, $asc)` – Vrací všechny rezervační masky určené pro právě jednu pobočku. Výstup je řazen (sestupně/vzestupně) podle sloupce (column).
- `getAllReservationsMasksMoreCafe($column, $asc)` – Vrací všechny rezervační masky určené pro více poboček než jednu. Výstup má stejné možnosti řazení záznamů jako metoda výše.
- `getCafesInReservationMask()` – Vrací seznam masek a pobočky, na které se masky používají.

- `getMaskReservation($maskId)` – Vrací rezervační masku včetně informace o pobočce, na kterou je použita. Tuto metodu lze použít pouze na masky s právě jednou pobočkou, na kterou jsou aplikovány.
- `deleteMask($maskId)` – Odstraní záznam s rezervační maskou (maskId) z databáze.
- `getSelectedCafesInReservationMask($maskId)` – Vrací seznam poboček, pro které je určena maska rezervací.
- `reservationNotAllowed($cafeId, $curDate)` – Vrací rozhodnutí, zda je možné na vybrané pobočce v daném dni provést rezervaci.

MaskReservationBlock poskytuje tyto veřejné metody:

- `getTimes($maskId)` – Vrací seznam časových úseků, u kterých není možné provést rezervace.
- `deleteMaskReservationBlocks($maskId)` – Odstraní všechny záznamy v tabulce týkající se vybrané masky.

MaskReservationToCafe poskytuje tyto veřejné metody:

- `deleteAllCafes($maskId)` – Odstraní všechny záznamy v tabulce týkající se vybrané masky.
- `getCafe($maskId)` – Vrací pobočku, na které je použita zvolená rezervační maska. Tuto metodu lze použít pouze na masky s právě jednou pobočkou, na kterou jsou aplikovány.

Pro správu masek rezervací bylo potřeba nově vytvořit presenter (MaskReservationPresenter), který je znázorněn na Obr. 49. Tento presenter se ve spolupráci s modelem MaskReservation stará o veškeré operace nad maskami rezervací jako je vytvoření/editaci masek jak pro jednu pobočku, tak i více poboček. V případě, že není už konkrétní maska potřebná, může být deaktivována nebo i trvale smazána.

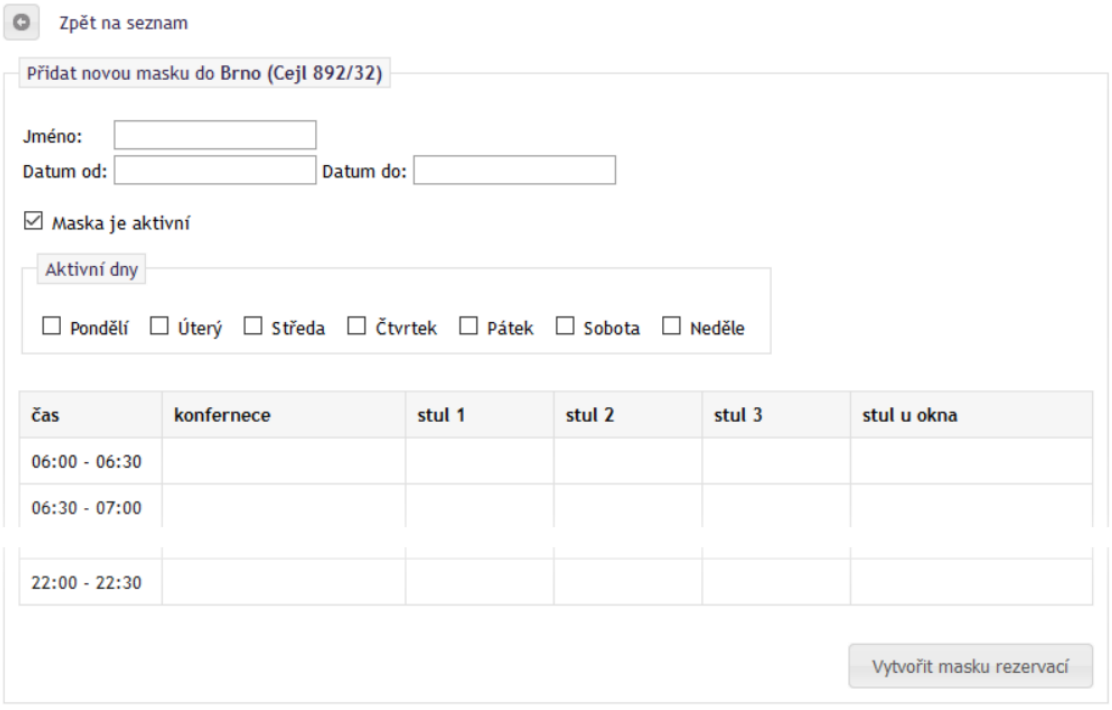
7.2.1 Přidání nové masky pro jednu pobočku

Pro přidání nové masky rezervace pro právě jednu pobočku bylo potřeba vytvořit formulář, který provede uživatele jejím vytvořením. Tentokrát jsou komponenty pro přidání a editaci masky rezervace založeny na stejné komponentě vytvořené metodou `maskReservationCommonForm($name)`. Základní komponenta nabízí společné prvky jako název masky, data platnosti, příznak aktivity masky a dny v týdnu, pro které bude maska platná. Na základě otevírací doby se vygenerují pro každý stůl časové bloky v intervalech třiceti minut. Tyto bloky jsou opatřeny skrytými poly, ve kterých je přenášena informace o zablokování časového úseku na konkrétním stole. Takto vytvořený formulář je předán metodě `createComponentAddMaskReservation($name)`, která předvyplní příznak aktivity masky na pozitivní hodnotu a opatří formulář potvrzovacím tlačítkem. Pro zpracování dat odeslaného formuláře se použije metoda `addMaskReservationSubmitted(\Nette\Forms\Form $form)`, díky níž

se uloží údaje o masce do tabulky mask_reservation. Na základě právě vložené masky je zjištěno její id, které je použito k vytvoření bloků zabraňujících rezervace v tabulce mask_reservation_block. Po provedení všech úkonů je uživatel informován o výsledku a přesměrován na stránku seznamů masek rezervací.

Hlavní částí masky je tabulka, ve které se vytváří vlastní maska rezervace. Ve sloupcích tabulky jsou znázorněny všechny stoly mající pozitivní příznak na rezervace. Řádky zobrazují časové intervaly po třiceti minutách od otevírací do zavírací doby pobočky. Vzhledem k tomu, že je možné, aby měla pobočka každý den různé provozní doby, je jako otvírací doba zvolena nejnižší hodina z celého týdne. U zavírací doby je naopak volena nejpozdější zavírací doba. Tím se maska více abstrahuje od nutnosti zadávat více masek pro různé dny a je vytvořena jedna univerzální pro všechny dny. Blokace v konkrétní časový interval se provede jednoduchým kliknutím do požadovaného bloku tabulky. Vybraný blok je barevně odlišen od standardního zobrazení tabulky. Opětovným klikem na označený blok se provede jeho odznačení.

Pro vykreslení stránky s formulářem je zavolána metoda `renderAdd($cafeId)` starající se o vykreslení šablony add.latte. Ukázka formuláře přidávající novou rezervační masku je na Obr. 21.



Obr. 21 Masky rezervací: Formulář pro přidání nové masky

7.2.2 Editace masky pro jednu pobočku

Při editaci se stejně jako při vytváření nové masky vychází z komponenty vytvořené metodou `maskReservationCommonForm($name)`. Tato komponenta je v metodě `createComponentEditMaskReservation($name)` doplněna o výchozí hodnoty související s maskou, které se získají pomocí modelů mask_reservation a mask_reservation_block. Takto vytvořený formulář je připraven pro editaci. Editace blokace probíhá opět nad tabulkou, ve které se klikem na požadovaný blok provede patřičná operace v závislosti na předcházejícím stavu bloku (označený blok se odznačí, neoznačený blok se označí). Po provedení editace a jejím potvrzení se vyplněný formulář zpracovává metodou `editMaskReservationSubmitted($form)`. Tato metoda provede aktualizace údajů v tabulce mask_reservation. K vytvoření záznamů o blocích slouží tabulka mask_reservation_block. Pro snadnější manipulaci se veškeré záznamy týkající se editované masky odstraní a vytvoří se nové. Na závěr se ještě provede přesměrování na stránku seznamů rezervačních masek a o úspěchu či neúspěchu editace je uživatel informován prostřednictvím flash zprávy.

K vykreslení formuláře pro editaci rezervačních masek je využito vykreslovací metody `renderEdit($maskId, $cafeId)` s použitím šablony edit.latte.

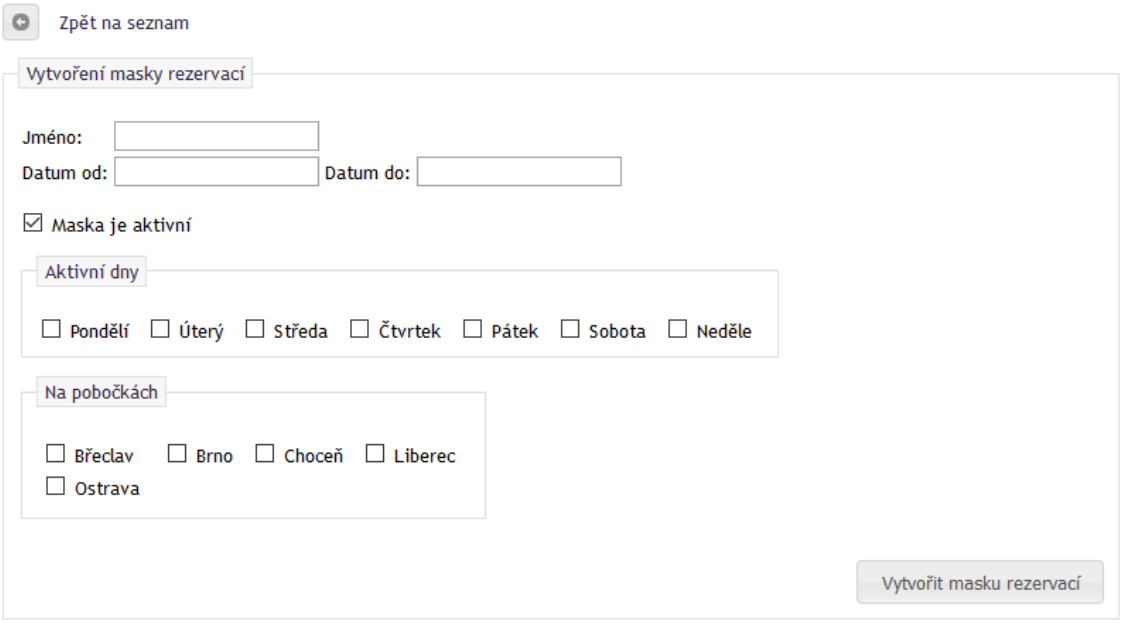
7.2.3 Smazání masky

Pro mazání masek slouží `handleDeleteMaskReservation($id)` přijímající ve svém jediném parametru id mazané rezervační masky. Metoda je použita pro mazání všech rezervačních masek, tedy jak pro mazání masek určených pro právě jednu pobočku, tak i masek určených pro více poboček. Po odstranění záznamu z tabulky mask_reservation se automaticky kaskádovitě odstraní i záznamy v jiných tabulkách mající referenci na mazaný záznam.

7.2.4 Přidání nové masky pro více poboček

Pro přidání a editaci masek rezervací pro více poboček je navržena komponenta obdobně, jako je tomu u masek rezervací pro jednu pobočku. Tato základní komponenta je vytvářena metodou `maskReservationMoreCommonForm($name)`. Vygenerovaný formulář obsahuje prvky pro název masky, časovou platnost, příznak aktivity masky a dny v týdnu. Tento formulář je pak využíván metodou `createComponentAddMaskReservationMoreCafe($name)` obohacující základní formulář o seznam poboček, na kterých může být maska nastavena a tlačítko pro odeslání formuláře. Současně je v této metodě vyplněn příznak aktivity masky na pozitivní hodnotu. Při práci s maskami pro více poboček se uvažuje zablokování celé pobočky, proto zde není potřebná žádná tabulka se stoly jednotlivých poboček. Pro zpracování dat odeslaného formuláře je použito metody `addMoreMaskReservationMoreCafeSubmitted($form)`. Princip uložení je obdobný uložení masek pro jednu pobočku.

K vykreslení formuláře pro přidání nové masky pro více poboček se využívá metody `renderAddMore($cafeId)`, která používá šablonu addMore.latte. Ukázka formuláře zobrazujícího přidání nové rezervační masky pro více poboček je na Obr. 22.



Obr. 22 Masky rezervací: Formulář pro přidání nové masky pro více poboček

7.2.5 Editace masky pro více poboček

Editace je prováděna s využitím základní komponenty vytvářené metodou `maskReservationCommonForm($name)`, kterou využívá formulářová metoda `createComponentEditMaskReservationMoreCafe($name)` pro vytvoření finálního formuláře. V této metodě se finální formulář obohatí o seznam poboček a tlačítko pro odeslání formuláře. Údaje o editované masce jsou získány z databáze. Ty jsou použity k vyplnění defaultních hodnot jednotlivých prvků finálního formuláře. Po odeslání finálního formuláře se zavolá a spustí metoda `editMaskReservationMoreCafeSubmitted($form)`, která se postará o uložení změn záznamu (ta se provede obdobně jako u editace záznamů masky pro jednu pobočku). Na závěr se provede přesměrování na stránku seznamů rezervačních masek. O provedení akce je uživatel informován prostřednictvím flash zprávy.

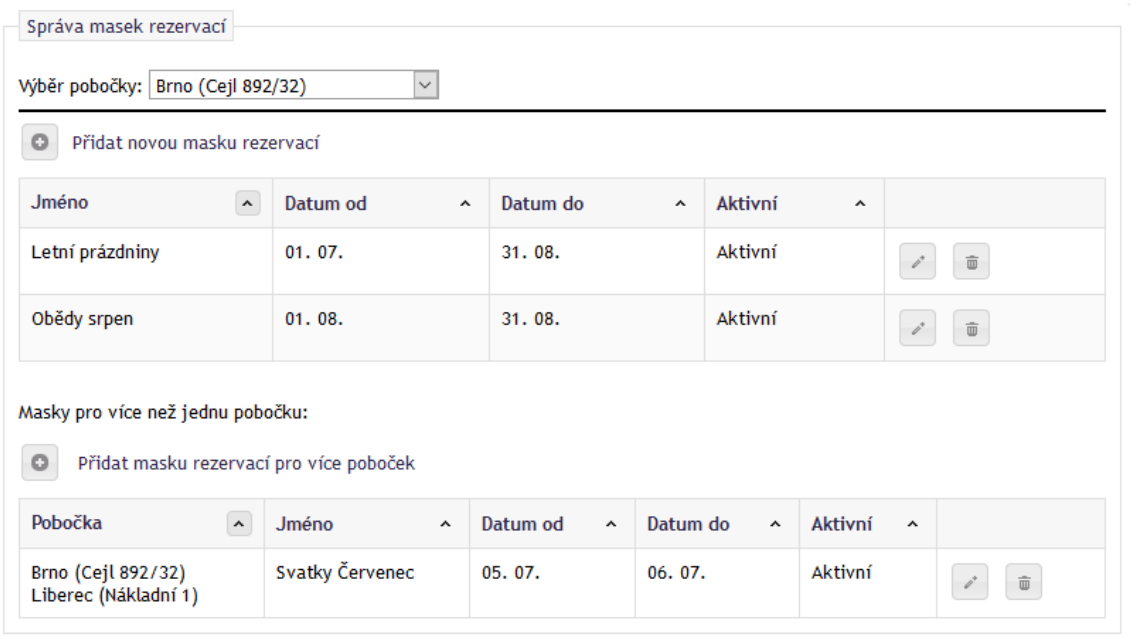
K vykreslení formuláře pro editaci masek rezervací pro více poboček se využívá metoda `renderEditMore($maskId, $cafeId)` využívající šablonu editMore.latte.

7.2.6 Seznam masek

Stránka zobrazující seznamy masek je rozdělena na několik částí. V první části si uživatel vybere pobočku v seznamu nabízených poboček nebo ponechá volbu pro výběr všech poboček. Tento výběr ovlivní vzhled následujícího bloku týkajícího se seznamu masek rezervací pro jednu pobočku. S výběrem konkrétní pobočky budou vykresleny pouze sloupce název, datum od, datum do, aktivní a sloupec operací nad maskami. Při výběru všech poboček se znepřístupní odkaz na přidání masky pro jednu pobočku a v tabulce se objeví sloupec pobočka, ve kterém bude vždy zobrazen název pobočky, pro kterou je maska určena. Zobrazení druhé části týkající se masek pro více poboček je po celou dobu volby poboček neměnné.

Operace přidávání masek jsou zobrazeny v podobě odkazů nad odpovídající tabulkou masek. Operace týkající se konkrétní masky jsou zobrazeny vždy na řádku odpovídajícím záznamu rezervační masky. Pokusí-li se uživatel odstranit masku, je nejprve vyzván o potvrzení prostřednictvím dialogového okna. Až po pozitivní odpovědi je záznam odstraněn z databáze.

K vykreslení stránky se seznamy masek se používá metoda `renderList($cafeId)` vykreslující do šablony list.latte.



Obr. 23 Masky rezervací: Seznam rezervačních masek

7.2.7 Aplikace masek na rezervace

Vytvoření masky samo o sobě nestačí na blokování rezervací. Vytvořené masky je potřeba navázat na stávající modul rezervací, aby plnily svou funkci blokování rezervací. Zároveň je nutné myslet na to, že se některé masky mohou v rámci času překrývat. Překrývání masek není v tomto případě nežádoucí. Masky jsou implementovány tak, aby mohlo být použito více masek ve stejný den. Máme-li například

dvě masky, z nichž první maska zakazuje vytváření rezervací v období obědů a druhá zakazuje provádět rezervace na jednom ze stolů a to po celý den, dochází na tomto stolu ke konfliktu masek v období obědů. U těchto masek se provede množinové sjednocení masek a obě masky se sloučí do jedné. Tato výsledná maska je namapována na rezervační tabulku, čímž zabrání vytvoření rezervace v daných blocích.

Mapování masek se provádí u všech přidávání a editování rezervací stejným způsobem. V odpovídající metodě (`renderAdd()`, `renderEdit()`) presenteru ReservationPresenter se přidá následující blok zdrojového kódu:

```
$isNotAllowed = $this->models->maskReservation
  ->reservationNotAllowed($cafe_id, $date);
if(!$isNotAllowed)
  $this->template->reservationMask = $this
    ->getReservationMask($cafe_id, $date);
$this->template->notAllowed = $isNotAllowed;
```

První příkaz se postará o získání příznaku určujícího, zda není nastavena blokace celé pobočky na celý den přes rezervační masku pro více poboček. Pokud ano, není třeba zbytečně provádět získání masek rezervací. Za předpokladu, že není aplikována žádná maska pro více kaváren, se získají blokace, které se následně promítnou skrze šablonu do pohledu.

7.3 Docházka

Docházka se z hlediska přístupu dělí na dvě části. První část je přístupná všem uživatelům administrační části. Tato část slouží pouze pro správu vlastní docházky. Druhá část umožňuje navíc spravovat cizí docházky a proplácet mzdy zaměstnancům. Pro přístup do této části je nutné mít navíc aktivované přístupové právo pro úpravu uživatelů.

Modely skrze které se přistupuje do databáze, jsou znázorněny na Obr. 46. Patří mezi ně:

- Attendance
- HourlyWage
- CostsInCafe
- ExtraTime
- EmployeePayment
- Bank

Attendance nabízí tyto veřejné metody:

- `isUserLogged($userId)` – Vrací informaci o tom, zda je uživatel přihlášen do docházky. Tato informace se určí z posledního záznamu docházky uživatele podle vyplněné hodnoty atributu logout_time.

- `lastLoggedDay($userId)` – Vrací poslední přihlášený den uživatele do docházky. Jako poslední den je chápán ten, který má hodnotu atributu `logout_time` rovnu hodnotě NULL.
- `lastLoggedId($userId)` – Vrací id poslední docházky uživatele. Určuje se stejným způsobem jako v předchozí metodě.
- `lastLoggedCafe($userId)` – Vrací id pobočky, ve které byl uživatel naposledy přihlášen. V případě, že uživatel ještě nebyl nikdy přihlášen, vrací hodnotu FALSE.
- `logoutFromWork($userId, $ip, $plusNMin, $attendanceId)` – Odhlásí zvoleného uživatele z docházky (uzavře jeho docházku). Odhlášení se provádí aktualizací odpovídajícího záznamu v tabulce.
- `getAttd($userId, $startDate, $endDate, $column, $asc, $onlyUnpaid)` – Vrací docházku uživatele ve zvoleném období. Ke každému záznamu ještě navíc spočítá počty minut a určí, zda přihlášení nebo odhlášení docházky bylo provedeno o víkendu. U časových údajů se provádí úprava podle vstupních parametrů (časové údaje z tabulky jsou přizpůsobeny vstupním časovým parametrům).
- `getAttdNoDatesTreatment($userId, $startDate, $endDate, $column, $asc, $onlyUnpaid)` – Obdoba výše uvedené metody, pouze se neprovádí úprava časových údajů, ale ponechávají se údaje z tabulky.
- `getAttdSum($userId, $cafeId, $startDate, $endDate, $onlyUnpaid)` – Vrací počet odpracovaných minut uživatele ve zvoleném období na definované pobočce.
- `getAttdSumForAll($cafeId, $startDate, $endDate)` – Vrací počet odpracovaných minut všech uživatelů na definované pobočce ve zvoleném období.
- `getAttdSumEachCafe($userId, $year, $month)` – Vrací počet odpracovaných minut uživatele v daném období. Odpracované minuty jsou rozděleny podle poboček.
- `getUsersAttdYears($userId)` – Vrací seznam let, ve kterých uživatel pracoval v kavárně.
- `getUnfinishedAttdInfo($cafeId, $userId, $startTime, $endTime)` – Vrací nedokončené docházky uživatele. Za nedokončenou docházku je považována ta, která má ve svém atributu logout_time hodnotu NULL.
- `countWeekDays($dateFrom, $dateTo)` – Vrací počet víkendových dnů mezi definovanými daty.
- `getProfit($cafeId, $dateFrom, $dateTo, $onlyUnpaid)` – Vrací zisk na pobočce mezi definovanými daty. Do této hodnoty jsou započte-

ny i náklady pobočky (nájem a energie), cena surovin a příjem z prodeje zboží a služeb.

HourlyWage nabízí tyto veřejné metody:

- `getCurrentHourlyWage($userId)` – Vrací aktuální hodinovou mzdu uživatele.
- `getPayment($userId, $dateFrom, $dateTo, $onlyUnpaid)` – Vrací mzdu uživatele v definovaném období. Metoda zohledňuje i historické údaje, spadají-li do definovaného časového intervalu. Mzda je vypočítávána pouze z odpracovaných hodin a hodinové mzdy. Bonusy a nedokončená docházka nejsou zahrnuty.
- `getPaymentInCafe($userId, $cafeId, $dateFrom, $dateTo, $onlyUnpaid)` – Metoda se chová stejně jako předchozí, údaje jsou však získávány pouze pro jednu pobočku.

CostsInCafe nabízí tyto veřejné metody:

- `saveAndUpdatePreview($cafeCosts)` – Slouží pro uložení nákladů pobočky. Existují tři možnosti uložení. Pokud neexistuje ještě žádný záznam, provede se vytvoření nového. Zbývající dvě operace se provedou, pokud záznam již existuje. V rámci stejného dne se bude stávající záznam aktualizovat. Pokud budou data odlišná, ukončí se stávající záznam a vytvoří se nový.
- `costsInCafe($cafeId, $dateFrom, $dateTo)` – Vrací náklady pobočky a náklady na suroviny.

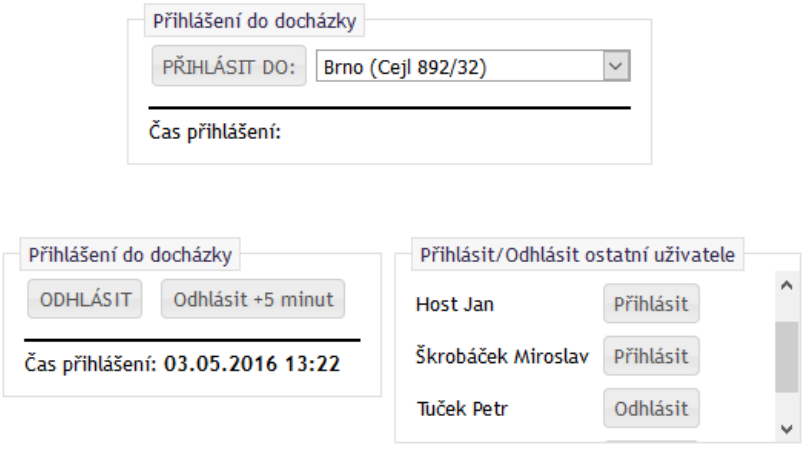
EmployeePayment nabízí tyto veřejné metody:

- `getUsersToPaydWages($cafeId, $isActive, $startTime, $endTime)` – Vypočítá mzdu (včetně bonusů ze zisku) všem zaměstnancům pobočky v daném období.
- `getPayedYears($order)` – Vrací roky, ve kterých byly vyplaceny mzdy.
- `getPaidInPaymentDate($paymentDate)` – Vrací podrobné údaje o vyplacených mzdách, které proběhly ve zvoleném dnu.
- `getAboData($paymentDate)` – Vrací údaje potřebné k vytvoření ABO souboru.
- `getAboDataForCafe($paymentDate, $cafeId)` – Vrací údaje potřebné k vytvoření ABO souboru pro konkrétní pobočku.
- `getCafesInPaymentDate($paymentDate)` – Vrací všechny pobočky, na kterých byly provedeny výplaty mezd v daném dnu.

Aby bylo možné docházku využívat ve stávající aplikaci bylo potřeba vytvořit patřičné presentery, které jsou zachyceny na Obr. 50. Celkově je modul docházky rozdělen na dvě části. První část se stará o docházku samotnou, druhá část pak na základě docházky vypočítává mzdy zaměstnancům.

7.3.1 Přihlášení a odhlášení z docházky

Aby bylo přihlašování do docházky co nejjednodušší a nejrychlejší, bylo vhodné vytvořit přihlašovací a odhlašovací formulář přímo na hlavní stránku aplikace. Na základě vyhodnocení metody `isUserLogged($userId)` z modelu Attendance se zvolí buď varianta pro přihlášení, nebo odhlášení z docházky. Při přihlašování si zaměstnanec volí, ve které pobočce se nachází. Zaměstnanci jsou nabídnuty pouze pobočky, ve kterých má aktivní pokladnu. Ze seznamu nabízených poboček je vždy označena pobočka, na níž byl zaměstnanec naposledy přihlášen, proto nemusí při každém přihlašování znovu vyhledávat pobočku. Je-li zaměstnanec přihlášen v docházce, nabídne se mu vedle odhlašovacího formuláře i seznam spolupracovníků ze stejné pobočky, na kterou se přihlásil. U každého zaměstnance v tomto seznamu je zobrazeno tlačítko přihlášení nebo odhlášení z docházky v závislosti na tom, zda je zaměstnanec právě přihlášen či nikoli. Tato funkcionalita umožní přihlašovat/odhlašovat spolupracovníky, aniž by se musel aktuální zaměstnanec odhlásit ze systému, aby umožnil svému spolupracovníkovi přihlášení/odhlášení se do docházky přes svůj účet. Toto je jeden ze způsobů jak může být zaměstnanec odhlášen z docházky. Dalším způsobem je, že se sám odhlásí prostřednictvím svého účtu. Zaměstnanec si v tomto případě volí mezi klasickým odhlášením a odhlášením, které je posunuto o určitý počet minut (aktuálně je to 5 minut). Druhou variantu využívají zaměstnanci, kteří na konci dne zavírají pobočku a provádí ještě další úkony i po tom, co už je počítač s aplikací vypnutý. Typicky se jedná o úklid a uzavření pobočky. Oba formuláře z hlavní stránky jsou zobrazeny na Obr. 24, kde formulář zobrazený výše je přihlašovací, formulář zobrazený níže v obrázku je odhlašovací s možností přihlášení/odhlášení spolupracovníků.



Obr. 24 Docházka: Přihlašovací a odhlašovací formuláře

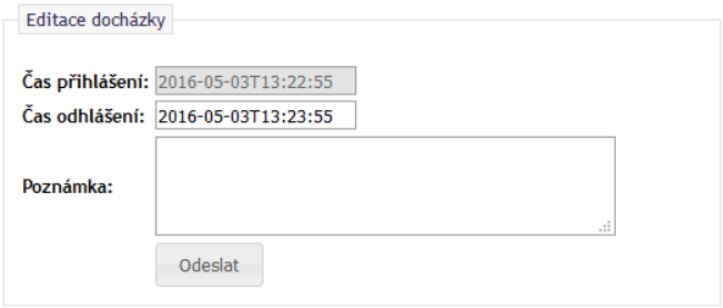
7.3.2 Editace a dokončení docházky

Všechny formuláře popsané v této kapitole jsou téměř identické, rozdíly jsou pouze v povolení editovat údaje týkající se časů přihlášení a odhlášení z docházky (údaje týkající se poznámky lze editovat ve všech variantách).

Formuláře týkající se editací docházek lze rozdělit na dvě skupiny podle toho, kdo editaci provádí. Provádí-li editaci docházky vlastník docházky, využívá se formuláře z metody `createComponentEditAttdDayForm($name)`. Tento formulář je vykreslován metodou `renderEditAttdDay($attendanceId, $editing)` pomocí šablony editAttdDay.latte. Editovat lze pouze čas odhlášení. Provádí-li editaci jiný zaměstnanec (často vedoucí pobočky) má k dispozici možnost editovat jak čas odhlášení, tak i čas přihlášení. Tento typ editace lze provádět skrze správu docházek.

Klasické dokončení docházky se provádí tlačítkem na hlavní stránce, jak bylo popsáno v kapitole výše. Pokud by však byl den odhlášení jiný, než den přihlášení, není možné provést tento postup odhlašování. Zaměstnanec, který se zapomněl předešlý den odhlásit je tedy upozorněn na tuto skutečnost a dokud neprovede dokončení docházky z předešlého dne, není mu umožněno se do docházky přihlásit. Dokončení docházky se řeší formulářem z metody `createComponentCompletitionAttdDayForm($name)`. Tento formulář nedovoluje měnit čas přihlášení do docházky. Vykreslován je pomocí `renderCompletitionAttendanceDay($attendanceId)` a šablony completitionAttendanceDay.latte.

Editace nad vlastní docházkou jsou časově omezeny. Změny lze provádět maximálně do týdne od data pořízení, pak už není zaměstnanec schopen docházku sám jakkoliv měnit. Změnu může provést jiná osoba s patřičnými přístupovými právy do správy docházky. Tato editace nabízí zaměstnanci měnit jak čas odhlášení, tak i čas přihlášení do docházky. Ukázka vzhledu editačního formuláře je na Obr. 25.



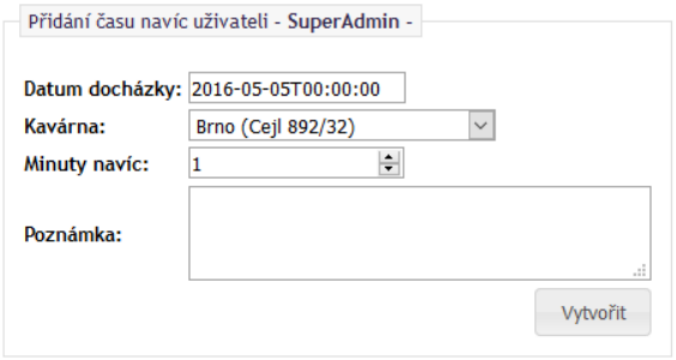
Obr. 25 Docházka: Formulář editace docházky

7.3.3 Přidávání bonusových hodin

V případě, že je potřeba přidat nějakému zaměstnanci hodiny navíc, může je zaměstnanec mající přístupová práva do správy docházky přidat pomocí funkce pro přidávání bonusových hodin. Pro generování formuláře se použije metoda `createComponentAddExtraTimeForm($name)`. Ve formuláři zaměstnanec zvolí časový údaj určující počátek časového období, pobočku, počet bonusových minut, případně poznámku, definující za co jsou bonusové minuty zaměstnanci uděleny.

Formulář zpracovaný metodou `addExtraTimeFormSubmitted($form)` vytvoří nový záznam v tabulce attendance. U tohoto záznamu není zapotřebí hlídat, zda není s jinou docházkou v kolizi. Po vytvoření záznamu se v tabulce extra_time vytvoří reference, aby bylo možné určit, že se jedná o bonusové hodiny.

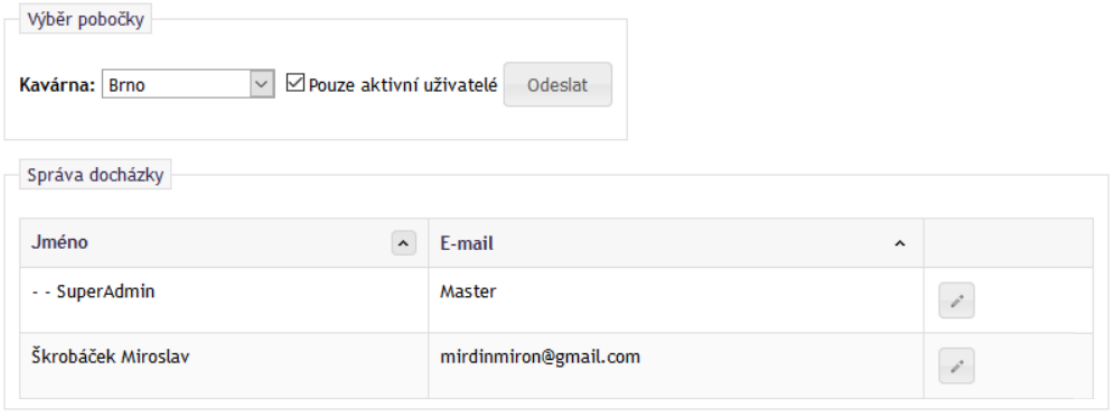
Formulář se zobrazí pomocí metody `renderAddExtraTime($userId, $date)` a šablony addExtraTime.latte. Ukázka formuláře je zobrazena na Obr. 26.



Obr. 26 Docházka: Formulář pro přidávání bonusových hodin

7.3.4 Správa docházky

Správa docházky nabízí zaměstnanci mající přístupová práva filtrovatelný seznam zaměstnanců a jejich docházek. Seznam je zobrazen v tabulce, ve které jsou základní údaje zaměstnanců (jméno, příjmení a e-mail) a odkaz do zaměstnancovy docházky. Filtr je založen na formuláři, který je generován metodou `createComponentSelectCafeForm($name)`. Vlastní stránka je pak zobrazena pomocí metody `renderManageAttd($cafeIi, $onlyActiveUsers)` a šablony manageAttd.latte. Ukázka stránky je na Obr. 27.



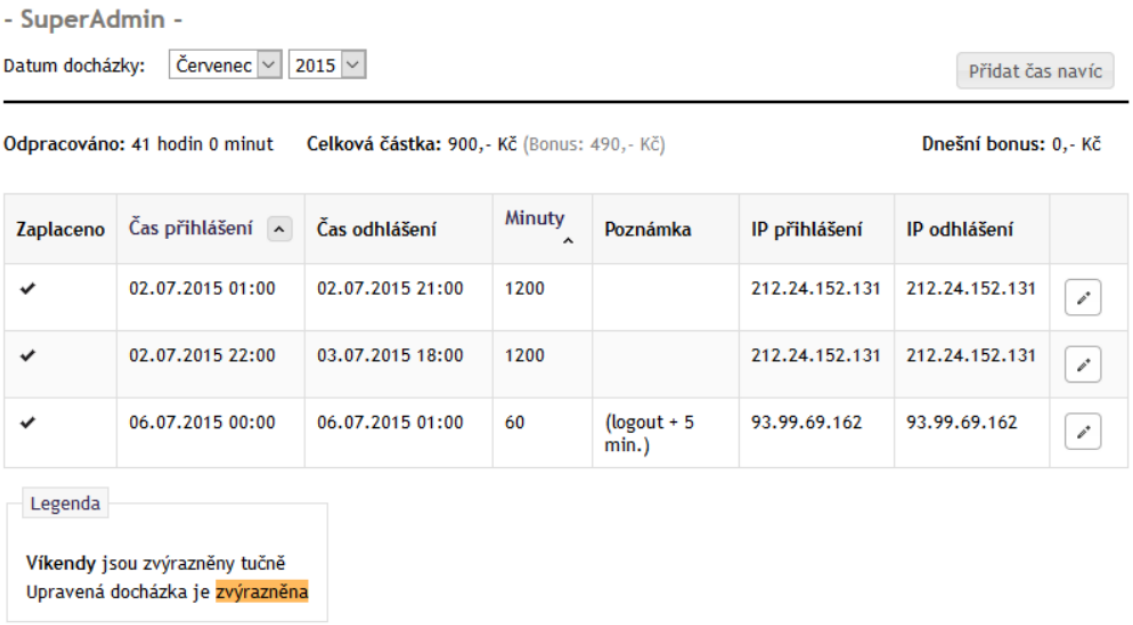
Obr. 27 Docházka: Správa docházky

7.3.5 Přehled docházky

Přehled docházky má dva režimy. První slouží vlastníkovi docházky k její správě, druhý slouží pověřeným osobám ke správě cizích docházek. Vzhledem k tomu, že se oba přehledy liší minimálně, bude zde popsán pouze přehled pro cizí správu docházky.

Nejdříve je potřeba zvolit období docházky, které se provádí pomocí formuláře generovaného metodou `createComponentUserOverviewForm($name)`. Na oba implementované selectboxy je navázána Javascriptová funkce, která při změně hodnoty jednoho z nich automaticky odešle formulář. Při vypnutí Javascriptu ve webovém prohlížeči se zobrazí klasické odesílací tlačítko.

Veškerá potřebná data pro zobrazení docházky se získávají prostřednictvím metody `renderManageUserOverview($userId, $attdDate)`, která získaná a vypočítaná data zprostředkovává šabloně renderManageUserOverview.latte. Stránka s přehledem docházky je zobrazena na Obr. 28.



Obr. 28 Docházka: Přehled docházky

Pomocí parametru userId se zjistí základní údaje o uživateli, které budou zobrazeny na stránce. Z parametru attdDate se získá měsíc a rok pro určení docházky.

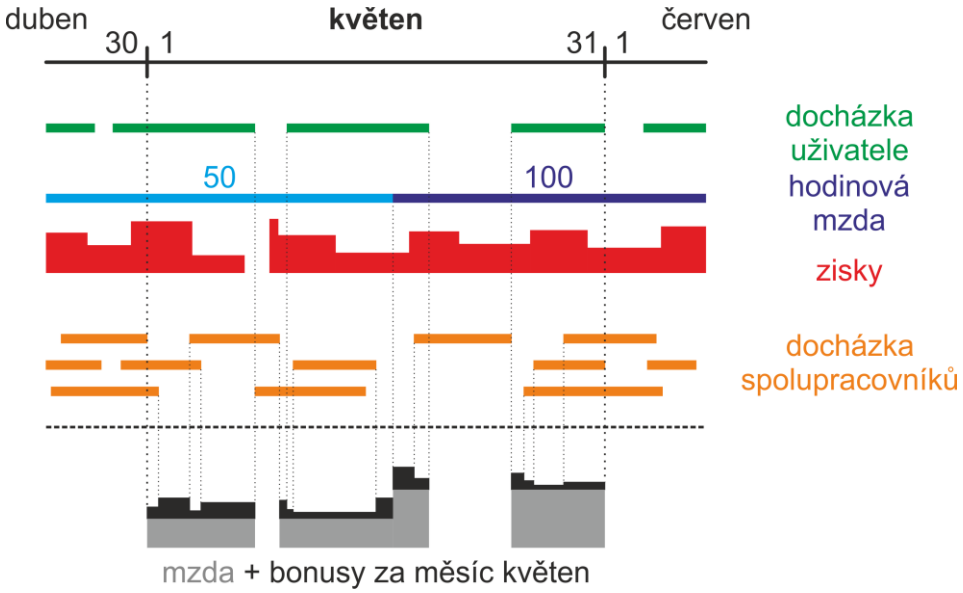
Následně se určí počáteční a koncové datum měsíce. Všechny tyto získané údaje pak poslouží jako vstupní parametry metodě `getAttendanceSum($userId, $cafeId, $startDate, $endDate, $onlyUnpaid)` z modelu attendace. Tato metoda získá z databáze všechny docházky zaměstnance, které jsou v kolizi s daty mezi prvním a posledním dnem měsíce. Data jednotlivých docházek, které jsou mimo měsíční interval, jsou nahrazena těmito daty. Získá se tak několik intervalů mající určitou časovou délku, která se

sečte. Výsledkem je potom počet odpracovaných minut zaměstnance v daném měsíci.

V dalším kroku je potřeba zjistit kolik si zaměstnanec vydělal na mzdě a bonusech. Vzhledem k tomu, že je potřeba uvažovat i historické údaje, nelze pouze vynásobit počet odpracovaných hodin a hodinovou mzdu. Výpočet mzdy a bonusů lze provádět současně a je potřeba získat údaje o nákladech na všech pobočkách, ke kterým se vztahuje docházka, zisky z prodejů a poměry odpracovaných hodin zaměstnance a všech zaměstnanců na zmíněných pobočkách. Na základě poměrů odpracovaných hodin a zisků se vypočítají bonusy. Znázornění výpočtu mzdy a bonusů nad jednou pobočkou je znázorněno na Obr. 29. Jednotlivé části určené pro výpočet jsou vzájemně porovnány mezi sebou a rozděleny na intervaly tak, aby každý interval obsahoval právě jednu hodnotu z každé části. Výsledné výpočty menších intervalů jsou průběžně sčítány, až je získána celková mzda zaměstnance a odpovídající bonusy ze zisku. Tato částka je pouze orientační a může být ještě změněna v rámci výplaty mzdy.

V dalších krocích se obdobným způsobem vypočítají i bonusy pro aktuální datum. Z databáze se ještě získá seznam odpovídajících docházek, které jsou zobrazeny do tabulky.

V tabulce jsou na jednotlivých řádcích zobrazeny docházky se základními údaji (stav proplacení docházky, časy přihlášení a odhlášení, počet odpracovaných minut, poznámka a adresy IP) a možností editace každé docházky. Řádky týkající se víkendových směn a editované docházky jsou zvýrazněny. Pokud je používán přehled v režimu editace cizí docházky, zobrazí se v pravém horním rohu tlačítko s možností přidat bonusové hodiny do docházky. V režimu editace vlastní docházky není tato funkce povolena.



Obr. 29 Ukázka provázání částí pro výpočet mzdy

7.3.6 Správa mezd zaměstnanců

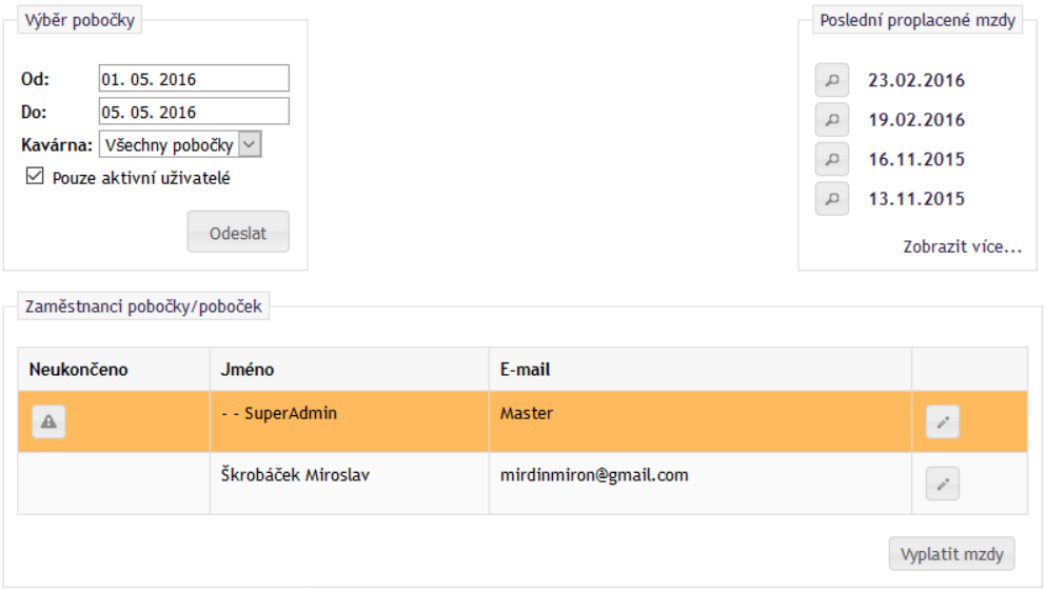
Správa mezd zaměstnanců slouží jako hlavní stránka pro správu mezd. Poskytuje odkazy na proplacené mzdy a slouží jako první krok pro vyplácení nových mezd.

V pravé horní části stránky se nachází odkazy na proplacené mzdy. V základu je zobrazeno pět posledních (řazení je podle data proplacení). Pokud by bylo potřeba vrátit se více do historie, je pod seznamem proplacených mezd odkaz na seznam všech proplacených mezd.

V levé horní části se nachází filtrovací formulář ovlivňující tabulku pod ním, která zobrazuje vybrané zaměstnance. Filtrovací formulář je vytvořen metodou `createComponentSelectCafeForm($name)`. Pomocí formuláře lze zvolit období, za které má být zaměstnancům vypočítána mzda. Případně se lze zaměřit pouze na konkrétní pobočku. Odeslaný filtrovací formulář se zpracuje metodou `selectCafeFormSubmitted($form)`, která provede přesměrování zpět na stránku se správou mezd s odpovídajícími parametry. V závislosti na těchto parametrech obslužná metoda `renderDefault($cafeId, $onlyActiveUsers, $startDate, $endDate)` získá požadované údaje a zobrazí je v tabulce.

Tabulka zobrazuje základní informace o zaměstnancích (jméno, příjmení a e-mail) a operaci pro editaci docházky, která je upravenou kopií editací docházky. První sloupec tabulky může zobrazovat upozornění na nedokončenou docházku zaměstnance. Upozornění je ve formě odkazu na stránku dokončení docházky, která již byla zmíněna výše v textu.

Na konci stránky se nachází tlačítko pro přechod na další krok vedoucí k vyplacení mezd. Po jeho aktivaci se provede přesměrování na stránku vyplacení mezd. Ukázka stránky se správou mezd zaměstnanců je znázorněna na Obr. 30.



Obr. 30 Docházka: Správa mezd zaměstnanců

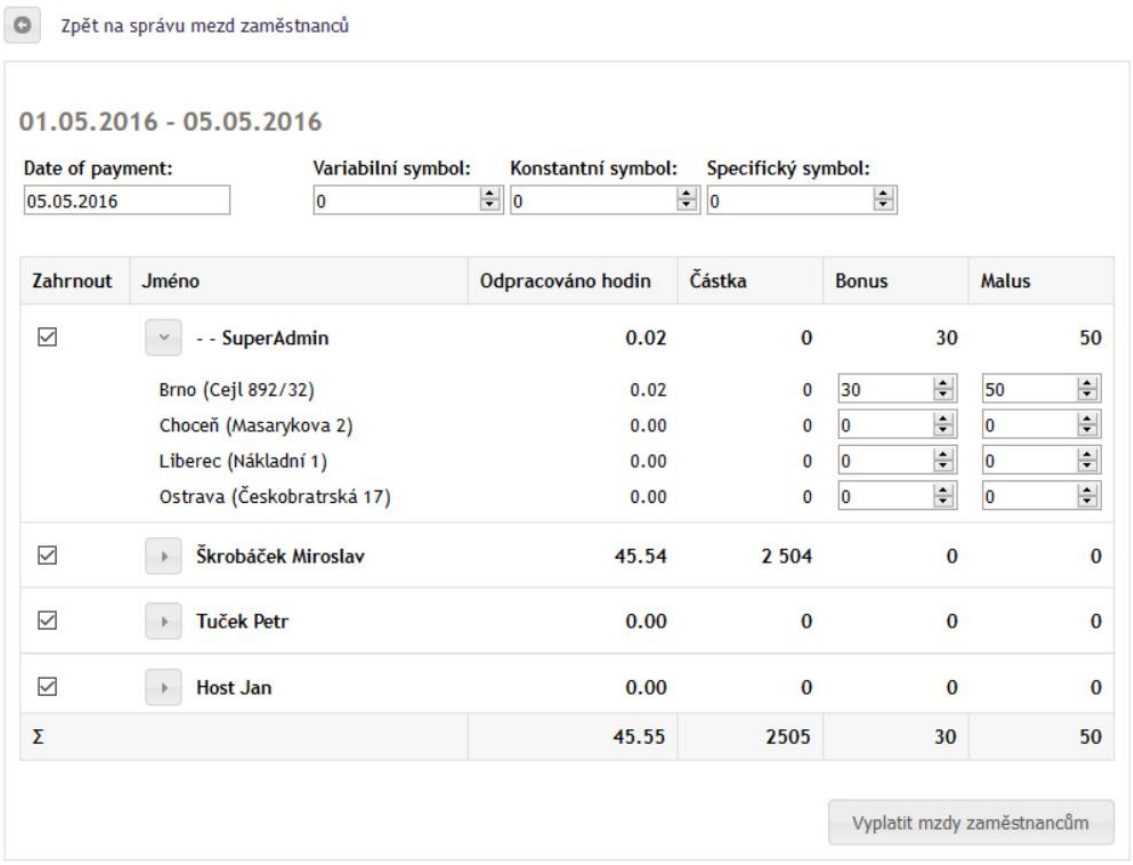
7.3.7 Vyplácení mezd

Formulář na stránce sloužící pro vyplácení mezd je generován metodou `createComponentPayWagesForm($name)`. Metoda si pomocí parametrů předaných metodě sloužící pro vykreslení stránky `renderPayWages($cafeId, $onlyActiveUsers, $startDate, $endDate)` získá seznam uživatelů s potřebnými údaji. Vykreslení se provádí s použitím šablony payWages.latte.

V horní části stránky je zobrazeno období, ke kterému se vztahuje vyplácení mezd. Aby bylo možné vytvořit ABO soubory, je potřeba si od zaměstnance vyžádat datum proplacení mezd. Aby bylo datum validní, je nutné jej nastavit nejdříve na následující datum po aktuálním datu. Následující tři položky slouží pro zadání variabilního, konstantního a specifického symbolu.

Pod těmito prvky se nachází tabulka se seznamem zaměstnanců, kterým bude vyplácena mzda. Každý zaměstnanec má kromě svého jména a příjmení uveden i počet odpracovaných hodin, vyplácenou mzdu, v níž jsou zahrnuty i bonusy ze zisku. Další dva údaje informují o přidělených variabilních bonusech a malusech. Pro získání více informací o vyplácené mzdě zaměstnanci lze pomocí tlačítka u jména zaměstnance rozbalit podrobnější informace. V tomto rozbaleném seznamu se nachází mzdy rozdělené podle jednotlivých poboček, na kterých má zaměstnanec pokladny. Je zde také možné přidělovat variabilní bonusy a malusy, jejichž změny se okamžitě pomocí javascriptových funkcí promítají jak do hlavního přehledu uživatele, tak i do sumačního řádku pod tabulkou. Zaměstnanec má tedy okamžitý přehled o tom, jaký finanční obnos bude vyplácen jednotlivým zaměstnancům.

Sloupec zahrnout určuje, zda se zaměstnanci bude mzda proplácet či nikoli. Proplacení mzdy v tomto kontextu znamená, že bude vytvořen záznam o proplacení mzdy v tabulce employee_paiment. U odpovídajících docházek bude nastaven pozitivní příznak v atributu proplaceno (payd). U docházek, u kterých dochází k přesahu, mimo vyplácené období se provede rozdělení, kdy je z přesahu vytvořena nová docházka. Druhé části docházky spadající do vypláceného období je nastaven pozitivní příznak proplaceno. Proplacení se také neprovede u zaměstnanců, kteří mají počet hodin, vyplacenou mzdu, variabilní bonus a malus rovny nule. Po vyhodnocení formuláře se stránka přesměruje zpět na správu mezd zaměstnanců. Ukázka stránky pro vyplácení mezd je na Obr. 31.

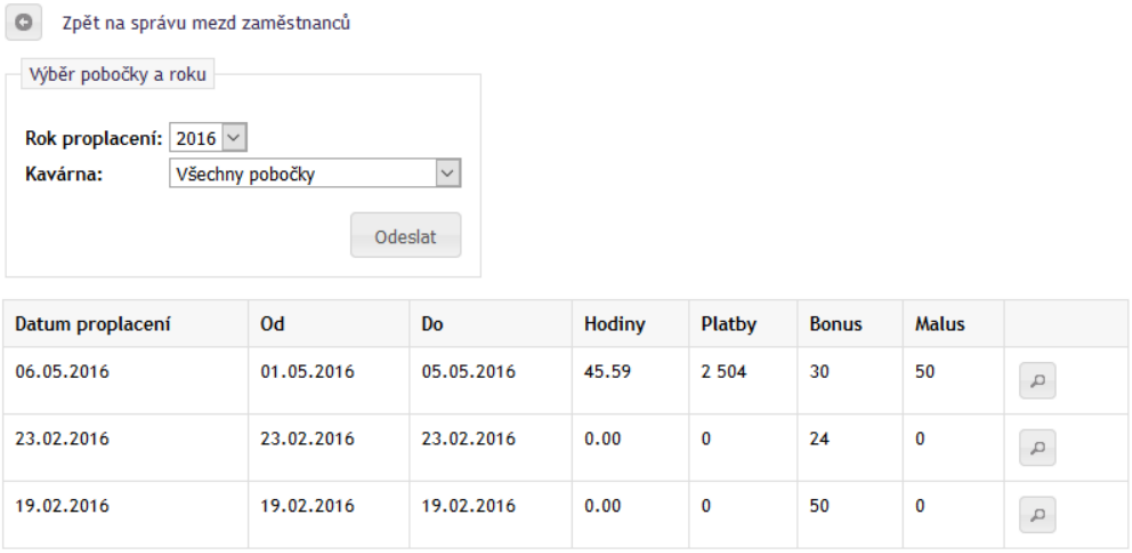


Obr. 31 Docházka: Stránky vyplácení mezd zaměstnancům

7.3.8 Seznam proplacených mezd

Pro filtrování seznamu proplacených mezd se používá formulář generovaný metodou `createComponentPaidPaymentSelectorForm($name)`, která umožňuje zaměstnanci si zvolit období a pobočku, pro kterou se má seznam zobrazit. Po odeslání formuláře se zobrazí tabulka se záznamy proplacených mezd seřazená sestupně podle data proplacení mezd. V tabulce se mimo toto datum nachází ještě časové období, pro které byly mzdy vypláceny a souhrny udávající počty odpracovaných hodin, vyplacené mzdy, případně bonusy a malusy. V posledním sloupci se nachází odkaz směrující na stránku s přehlednějšími informacemi o odpovídajících proplacených mzdách.

Vykreslení stránky zajišťuje metoda `renderPaidWagesList($cafeId, $year)` se šablonou paidWagesList.latte. Ukázka této stránky je na Obr. 32.



Obr. 32 Docházka: Seznam proplacených mezd

7.3.9 Přehled proplacených mezd

Stránka poskytuje podrobnější přehled o vyplacených mzdách. Na stránce je zobrazena tabulka, která je téměř identická s tabulkou vyplácení mezd, pouze variabilní bonusy a malusy jsou zde zobrazeny staticky. Novinkou je však seznam odkazů pro generování ABO souborů nacházející se pod touto tabulkou. Ukázka stránky zobrazující přehled proplacených mezd je na Obr. 33.

Automatizované bankovní operace (ABO) slouží k mezibankovnímu převodu peněz. ABO soubory mají jasně definovanou strukturu. Informace o struktuře těchto souborů jsou dostupné nejčastěji na stránkách bank v sekci pro vývojáře. I když je struktura souboru pevně dána, jednotlivé banky jsou různě benevolentní k vyplněným hodnotám v souboru. Je tedy vhodné mít pro každou konkrétní banku připravený vlastní ABO soubor, který odpovídá jejím požadavkům.

Nejprve jsem si vytvořil obecnou sadu tříd, která by měla splňovat všechny základní požadavky ABO souboru. Díky tomu, že jsou třídy navrženy jako abstraktní, nelze s nimi vytvářet objekty. Tyto třídy poskytují pouze rozhraní, které musí být implementováno v potomcích těchto tříd. Takto vytvořené třídy představují šablony už konkrétních bank. Vzhledem k tomu, že v současné době mají všechny pobočky zřízen účet u Fio banky, vytvořil jsem třídy pouze pro tuto banku. Bude-li potřeba někdy v budoucnu implementovat ABO soubory pro jiné banky, bude existovat tento soubor tříd, který návrh a implementaci nového ABO souboru urychlí.

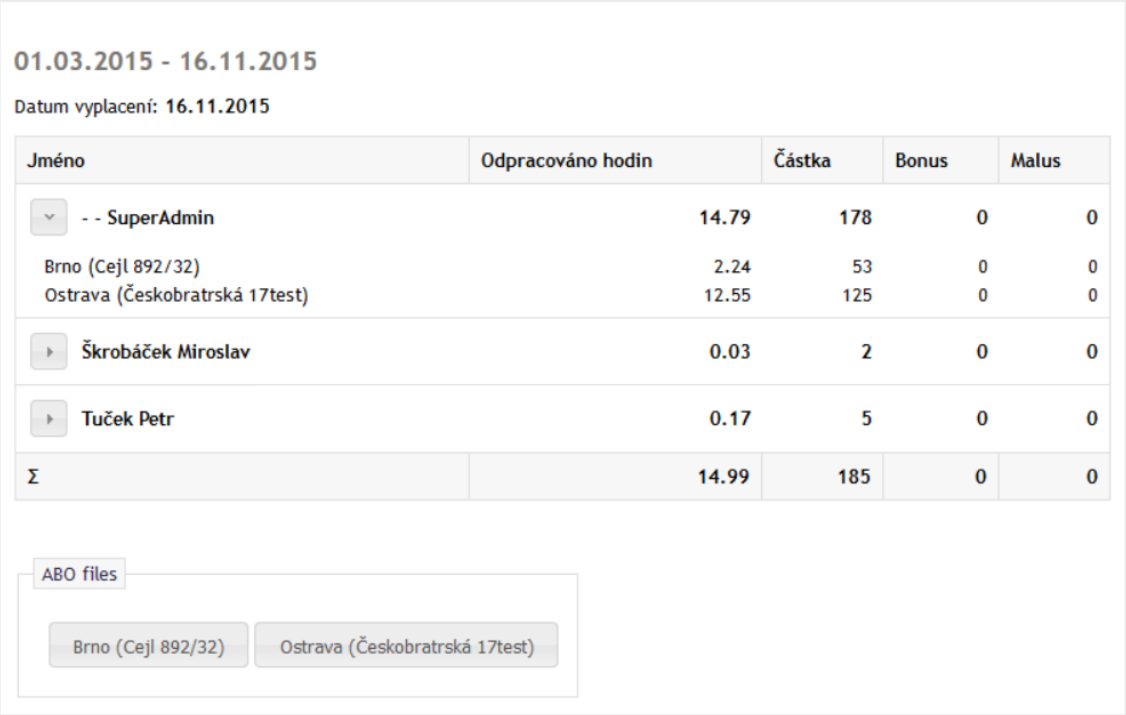
Jako referenční příručku na vytvoření ABO souboru jsem zvolil dokument, který je dostupný na stránkách Fio banky (Fio banka - česká banka pro váš účet nebo investice | Fio banka, 2016). Knihovna vytvořená z těchto souborů se v projektu nachází v adresáři libs/abo. Pro vytvoření ABO souboru je potřeba zavolat metodu `getAboDataForCafe($paymentDate, $cafeId)` z modelu EmployeePayment. Této metodě se předají dva parametry, datum vyplacení mzdy

a pobočka, pro kterou je vyplacení určeno. Tato metoda získá potřebná data z databáze, vytvoří objekt třídy AboRecord a naplní jej těmito daty. Zavoláním metody `getAboToString()` nad tímto objektem získáme zprávu ABO, která je pak uložena do souboru. Tento soubor se pak nabídne uživateli ke stažení. Ve staženém souboru se nachází údaje, které lze použít buď k přímému vyplacení mezd zaměstnancům, nebo jej lze použít jiným systémem sloužícím k účetnictví. Ukázka použití v aplikaci je zobrazena zdrojovým kódem:

```
$dateP = new \DateTime("2016-25-02 11:12:13"); // datum proplacení
$cafeId = 1;

$abo = $this->models->employeePayment->getAboDataForCafe($dateP, $cafeId);
$aboText = $abo->getAboToString();

$filename = "ABO" . $dateP->format("Ymd") . "_" . $cafeId . ".txt";

header("Content-Type: text/plain");
header('Content-Disposition: attachment; filename="'.$filename.'"');
header("Content-Length: " . strlen($aboText));
header("Connection: close");

echo $aboText;
```

Příkazem `echo $aboText;` se zaměstnanci nabídne textový soubor s ABO údaji ke stažení. Po otevření může obsah souboru vypadat následovně:

```
UHL1060516CAFEPLUS            0000000000001999
1 1501 001000 2010
2 123456-1234567890 5500 161115
123456-1234567890 5300 01000   AV:Vyplata 02/16 Cafeplus
666666-1111111111 200 44440   AV:Vyplata 02/16 Cafeplus
3 +
5 +
```



Obr. 33 Docházka: Přehled proplacených mezd

7.4 Události

Události jsou rozděleny na dvě části. Jedna je určena pro veřejnou část aplikace, druhá pro administrační část aplikace, přičemž přístup do některých částí ve veřejném webu je umožněn pouze zákazníkům majících založený Café+ účet. Administrační část slouží pro správu událostí a je přístupná pouze zaměstnancům kavárny. Přístup je tedy umožněn všem zaměstnancům kavárny.

Modely, přes které se přistupuje do databáze, jsou znázorněny na Obr. 47, mezi ně patří:

- Event
- EventRegistrationType
- PublicUserToEvent

Event nabízí tyto veřejné metody:

- `saveImage(\Nette\Http\FileUpload $img)` – Uloží obrázek na server a v návratové hodnotě vrací cestu tohoto uloženého obrázku.
- `regType($type)` – Podle zadaného parametru vrací odpovídající textový popis registračního typu události.
- `state($state)` – Podle zadaného parametru vrací odpovídající textový popis stavu události.

- `isPast($eventId)` – Vrací pozitivní hodnotu, pokud událost definovaná parametrem eventId proběhla.
- `isFull($eventId)` – Vrací pozitivní hodnotu, pokud na událost nelze přidat dalšího účastníka.
- `countRegistered($eventId)` – Vrací počet registrovaných účastníků události.
- `usersNotEvent($eventId)` – Vrací seznam registrovaných zákazníků, kteří mohou být na událost registrováni.
- `eventsWithRegistered($showOld, $order, $asc, $pageOffset, $itemPerPage)` – Vrací všechny události včetně počtů registrovaných účastníků. Parametry dále specifikují, zda se jedná jen o události, které proběhly, způsob řazení výsledných událostí a stránkování.
- `activePublicEventsWithRegistered()` – Vrací události, které jsou zobrazitelné ve veřejné části aplikace včetně počtů registrovaných účastníků.
- `eventsWithoutUser($publicUserId)` – Vrací události zobrazitelné ve veřejné části aplikace včetně počtů registrovaných účastníků, na které se může zákazník ještě registrovat.
- `usersEventsWithRegistered($publicUserId)` – Vrací události zobrazitelné ve veřejné části aplikace, na kterých je zákazník registrován.
- `sendMailToPublicUsers($eventId, $msg)` – Odešle e-mail všem účastníkům události.

PublicUserToEvent nabízí tyto veřejné metody:

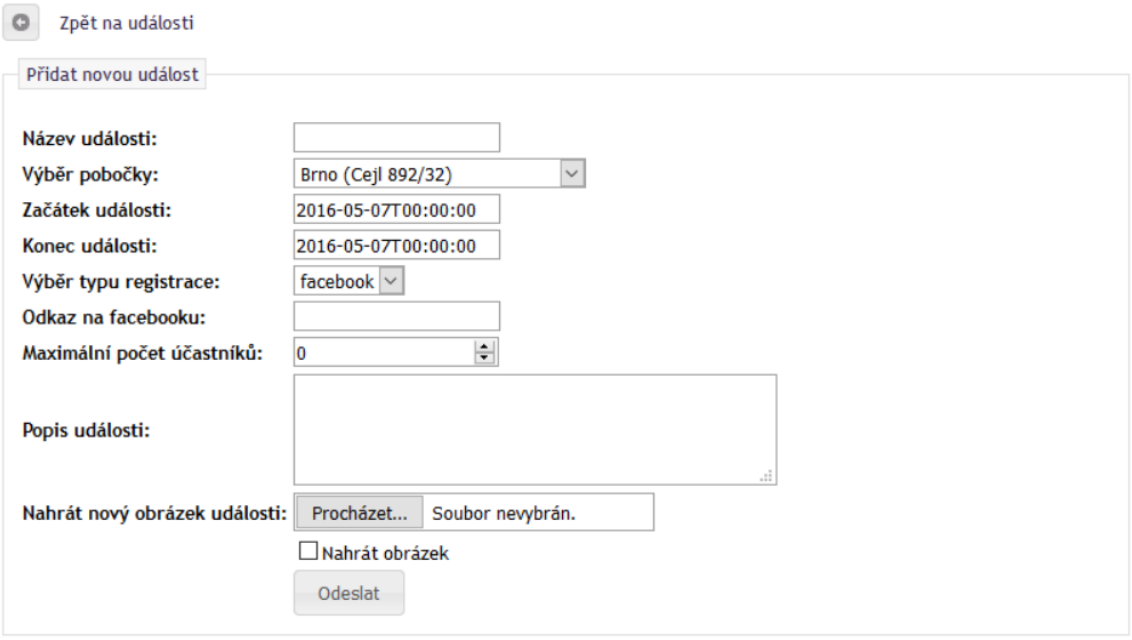
- `removePublicUserFromEvent($userId, $eventId)` – Odebere vybraného účastníka z události.
- `removeAllPublicUsersFromEvent($eventId)` – Odebere všechny účastníky z vybrané události.
- `lastLogged($eventId, $count)` – Vrací seznam naposledy přihlášených zákazníků o maximálním počtu udávající parametr count z vybrané události.

Aby bylo možné události využívat ve stávající aplikaci bylo potřeba vytvořit odpovídající presentery jak pro administrační, tak i veřejnou část aplikace. Tyto presentery jsou zachyceny na Obr. 51. Nejprve jsou popsány funkce událostí v administrační části a poté ve veřejné části aplikace.

7.4.1 Přidání nové události

Formulář pro přidání nové události se vytvoří komponentovou metodou `createComponentAddEventForm($name)`, která rozšiřuje formulář z metody `createComponentEventForm($name)`. Tento základní formulář je společný pro přidání a editaci události. Formulář obsahuje prvky sloužící k získání názvu,

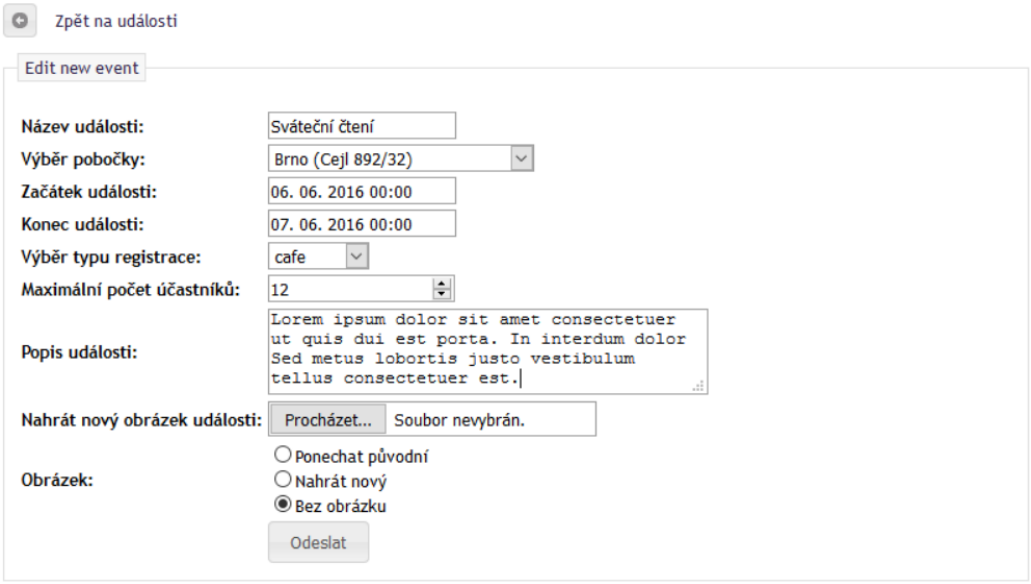
časového období, typu registrace, maximálního počtu účastníků, případně popis a uložení obrázku k události. Pokud je u typu registrace vybrán typ facebook, zobrazí se na stránce ještě pole pro vložení odkazu profilu na událost ze stránek Facebooku. Metoda `createComponentAddEventForm($name)` do formuláře nově přidá checkbox na potvrzení k uložení obrázku události a odesílací tlačítko přeposílající vyplněný formulář metodě `addEventSubmitted($form)`. Ta provede validaci, a pokud je vše v pořádku, uloží novou událost do databáze. Vykreslení formuláře se provede metodou `renderAdd()` s šablonou add.latte. Ukázka vzhledu formuláře pro přidání události je na Obr. 34.



Obr. 34 Událost: Formulář pro přidání nové události

7.4.2 Editace události

K vytvoření formuláře pro editaci exitující události je použito metody `createComponentEditEventForm($name)`. Jak bylo již zmíněno v kapitole výše, je i zde využito společného formuláře, který je rozšířen touto metodou o nové prvky a vyplnění všech prvků formuláře. K nově vzniklým prvkům patří skupina radio buttonů, kterými lze určit akci prováděnou s obrázkem události. Přidáno je také tlačítko pro odeslání formuláře, které po své aktivaci odešle formulář metodě `editEventSubmitted($form)`. V této metodě se provede aktualizace údajů a zpracování obrázku. Pokud se sníží maximální počet účastníků a přihlášených účastníků je více než povolený počet, provede se odebrání naposledy přihlášených účastníků. Těmto účastníkům je následně odeslán informační e-mail. Po zpracování editace události se provede přesměrování na správu událostí. Ukázka vzhledu formuláře pro přidání události je na Obr. 35.



Obr. 35 Události: Formulář editace události

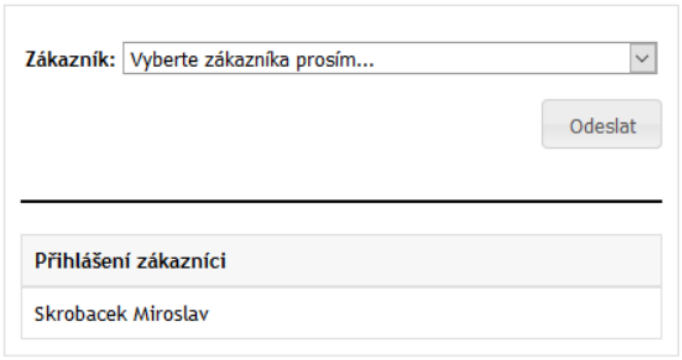
7.4.3 Profil události

Profil události poskytuje zaměstnanci informace o události a seznam přihlášených účastníků. V levé části profilu se nachází formulář generovaný metodou `createComponentEventProfileForm($name)`. Tento formulář má na všech prvcích zakázánu editaci (slouží zaměstnanci pouze pro čtení). Jestliže existuje pro událost obrázek, pak je zobrazen pod popisem události. Na Obr. 36 je zobrazena varianta bez obrázku. V pravé části stránky se nachází seznam účastníků s možností odebrání z události. Pro odebrání slouží metoda `handleDelete($eventId)`, která při odebírání záznamu pošle zákazníkovi informační e-mail.

Obr. 36 Události: Profil události

7.4.4 Přidání účastníka

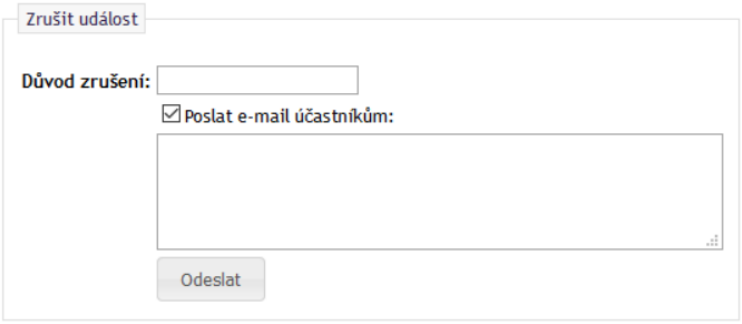
Pro přidání účastníka slouží jednoduchý formulář generovaný metodou `createComponentPublicUserSelectionForm($name)` nabízející zaměstnanci seznam zákazníků, kteří ještě nejsou registrováni na událost. Pro snadnější orientaci a určení, zda se zákazník už registroval na událost, slouží tabulka s vypsanými účastníky události. Formulář i se seznamem účastníků je zobrazen na Obr. 37.



Obr. 37 Událost: Formulář pro přidání účastníka

7.4.5 Zrušení události

Zrušení události se provádí prostřednictvím metody `createComponentCancelEventForm($name)`, která po zaměstnanci vyžaduje uvedení důvodu zrušení. Volitelně pak nabízí poslání informačního e-mailu všem účastníkům. Zrušit lze pouze události, které ještě neproběhly nebo neprobíhají. K vykreslení dochází prostřednictvím šablony cancelEvent.latte a metody `renderCancelEvent($eventId)`. Ukázka formuláře je na Obr. 38.

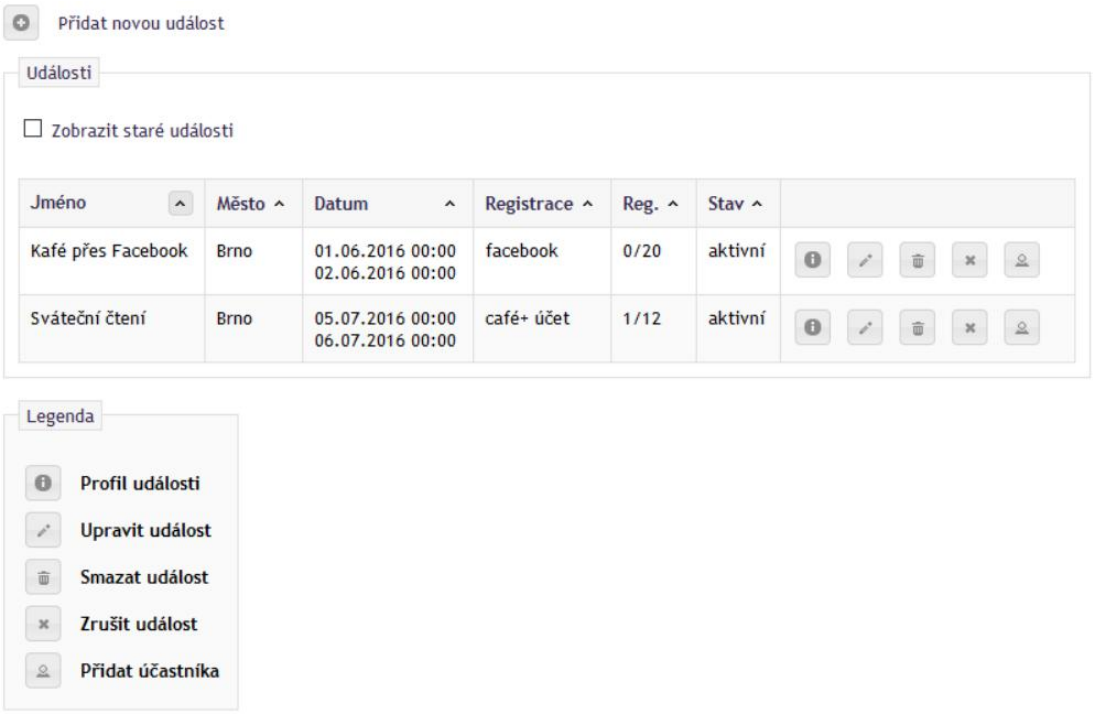


Obr. 38 Události: Formulář pro zrušení události

7.4.6 Správa událostí

Stránka nabízí přehled o jednotlivých událostech. Pokud by počet zobrazených událostí měl překročit hodnotu 15, bude se vykreslovaná tabulka stránkovat pomocí komponenty `createComponentVisualPaginator($name)`. Každá udá-

lost je navíc opatřena skupinou funkčních prvků, které umožňují provádět s událostí sadu operací. Půjdeme-li postupně, tak jak jsou zobrazeny v ukázce na Obr. 39, bude první operací přechod na profil události. Toto je jediná operace prováděná i na událostech, které proběhly. Zbývající události jsou přístupné pouze do začátku události, poté se znepřístupní. K těmto operacím patří editace události, smazání události, zrušení události a přidání nového účastníka. Stránka je vykreslována pomocí metody `renderList($showOld)` a šablony list.latte.



Obr. 39 Události: Stránka pro správu událostí

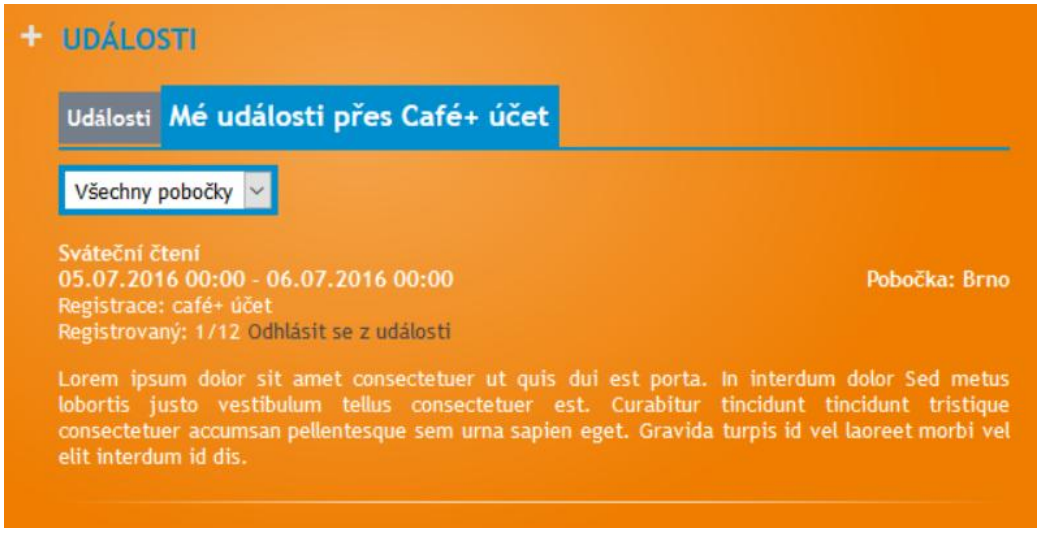
7.4.7 Události ve veřejné části

Seznam událostí je dostupný ve veřejné části aplikace a je přístupný všem zákazníkům kavárny. V této části jsou zobrazovány pouze události, které buď aktuálně probíhají, nebo ještě neproběhly. Události je možné filtrovat podle poboček filtrem generovaným metodou `createComponentCafesDefaultForm($name)`. Filtr se skládá ze seznamu poboček a odesílacího tlačítka, které je při zapnutém Javascriptu skryto. Zobrazování událostí se provádí vždy při změně hodnoty selectboxu, kterou odchytává funkce Javaskriptu. Podle nové hodnoty se zviditelní pouze odpovídající události. Pokud je Javaskript v prohlížeči zakázán, provádí se filtrování až po odeslání formuláře stiskem odesílacího tlačítka. Ukázka veřejného seznamu událostí je na Obr. 40.

Pokud se zákazník přihlásí do svého Café+ účtu bude mít k dispozici rozšířený přehled událostí, ve kterém se nachází obdobný seznam jako pro nepřihlášené zákazníky. Přihlášený zákazník se na rozdíl od nepřihlášeného může na tyto události

přihlásit. Zároveň je zákazníkovi k dispozici filtrovatelný seznam událostí, na kterých je registrován. Filtrování seznamu funguje na stejném principu jako seznam pro nepřihlášené zákazníky. Tento seznam nabízí zrušení rezervací na událostech, na kterých je zákazník rezervován. Ukázka správy událostí je na Obr. 41.

Obr. 40 Události: Seznam událostí (veřejná část aplikace)



Obr. 41 Události: Správa událostí (veřejná část aplikace)

8 Závěr a diskuze

Cílem diplomové práce bylo navrhnout a implementovat vybrané moduly do informačního systému pro správu literárních kaváren.

Kapitola PHP a MVC frameworky se zabývá základním popisem skriptovacího programovacího jazyka PHP a několika vybraných PHP MVC frameworků, které patří v současné době k jedněm z nejpopulárnějších. Ve světě to jsou Laravel, Symfony, Zend framework a Phalcon. Z českých frameworků je to Nette, v němž probíhal také vývoj samotných modulů.

V kapitole Stávající aplikace jsem se pokusil shrnout nejdůležitější informace týkající se stávajícího informačního systému, mezi které patří používané technologie a struktura aplikace. Aplikaci je rozdělena na dvě části. První veřejně přístupnou zákazníkům a druhou sloužící výhradně pro potřeby zaměstnanců kavárny. Obě části aplikace jsou pak stručně popsány z několika pohledů, které poskytují obecný náhled, jak lze aplikaci používat a jakým způsobem pracuje.

Kapitola Vývoj modulů se věnuje návrhu a vývoji vybraných modulů (nástěnka, rezervace, docházka a události). Každému modulu předchází specifikace požadavků, následovaná návrhem modulu a vlastní implementací. Implementace je provedena pomocí skriptovacího jazyka PHP a frameworku Nette.

S ohledem na uvedený předpokládaný cíl práce lze již konstatovat, že byl předpokládaný cíl splněn a navržené moduly byly úspěšně implementovány do aplikace. Vzhledem k tomu, že jsou moduly částečně závislé na některých částech stávající aplikace, není možné je používat nezávisle na aplikaci, aniž by bylo třeba je pozměnit. Ačkoliv je systém již aktivně využíván, stále se jedná o systém ve vývoji.

Vhodným navázáním na tuto práci by byla implementace automatického testování, které tento systém postrádá. V administrační části by také bylo možné navrhnout a implementovat modul týkající se elektronické evidence tržeb, který by byl navázán na pokladní část systému.

9 Literatura

10 PHP Frameworks For Developers - Best Of - Hongkiat. *Design Tips, Tutorials and Inspiration* [online]. 2016 [cit. 2016-04-03]. Dostupné z: http://www.hongkiat.com/blog/best-php-frameworks/

Comparison of the top 6 PHP frameworks. *CRM & CMS Development Illinois | Suyati Technologies* [online]. 2015 [cit. 2016-05-08]. Dostupné z: http://suyati.com/comparison-of-the-top-6-php-frameworks/

Česká národní banka - Česká národní banka [online]. 2016 [cit. 2016-04-25]. Dostupné z: http://www.cnb.cz/cs/index.html

DOYLE, Matt. *Beginning PHP 5.3*. Indianapolis, IN: Wiley Pub., 2010. Wrox beginning guides. ISBN 04-704-1396-4.

Fio banka - česká banka pro váš účet nebo investice | Fio banka [online]. 2016 [cit. 2016-01-06]. Dostupné z: http://www.fio.cz/

Framework Definition. *The Tech Terms Computer Dictionary* [online]. 2016 [cit. 2016-04-02]. Dostupné z: http://techterms.com/definition/framework

Functional programming - Wikipedia, the free encyclopedia. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-25]. Dostupné z: https://en.wikipedia.org/wiki/Functional_programming

History of Laravel PHP framework, Eloquence emerging. *Maxoffsky - Maks Surguy's blog on PHP and Laravel* [online]. 2013 [cit. 2016-04-02]. Dostupné z: http://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

Historical trends in the usage of server-side programming languages, April 2016 [online]. 2016 [cit. 2016-04-01]. Dostupné z: http://w3techs.com/technologies/history_overview/programming_language

Imperative programming - Wikipedia, the free encyclopedia. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-25]. Dostupné z: https://en.wikipedia.org/wiki/Imperative_programming

Laravel - The PHP Framework For Web Artisans [online]. 2016 [cit. 2016-04-02]. Dostupné z: https://laravel.com/

Laravel - Wikipedia, the free encyclopedia. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-02]. Dostupné z: https://en.wikipedia.org/wiki/Laravel

Linux Software [online]. 2016 [cit. 2016-02-26]. Dostupné z: http://www.linuxsoft.cz/

Objektově orientované programování – Wikipedie. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-25]. Dostupné z: htt-

ps://cs.wikipedia.org/wiki/Objektov%C4%9B_orientovan%C3%A9_programov%C3%A1n%C3%AD

Phalcon - High Performance PHP Framework [online]. 2016 [cit. 2016-04-04]. Dostupné z: https://phalconphp.com/en/

Phalcon - nejrychlejší mezi PHP frameworky | SKOUMAL. *SKOUMAL* [online]. Praha, 2016 [cit. 2016-04-04]. Dostupné z: http://www.skoumal.net/cs/phalcon-nejrychlejsi-mezi-php-frameworky/

PHP Master | PhalconPHP: Yet Another PHP Framework? *SitePoint – Learn HTML, CSS, JavaScript, PHP, Ruby & Responsive Design* [online]. 2016 [cit. 2016-04-04]. Dostupné z: http://www.sitepoint.com/phalconphp-yet-another-php-framework/

PHP - Wikipedia, the free encyclopedia. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-25]. Dostupné z: https://en.wikipedia.org/wiki/PHP

PHP: Hypertext Preprocessor [online]. The PHP Group, ©2001-2016 [cit. 2016-02-25]. Dostupné z: http://php.net/

PROCHÁZKA, David. *PHP 6: začínáme programovat*. 1. vyd. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-3899-4.

Reflection (computer programming) - Wikipedia, the free encyclopedia. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-02-25]. Dostupné z: https://en.wikipedia.org/wiki/Reflection_%28computer_programming%29

Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework [online]. 2016 [cit. 2016-04-04]. Dostupné z: view-source: https://nette.org/cs/

Symfony, High Performance PHP Framework for Web Development [online]. 2016 [cit. 2016-04-02]. Dostupné z: http://symfony.com/

Top 5 Best PHP Frameworks for 2016 To Become a Master Developer. *Easy Way To create your Mobile and Website Development - Build a Mobile App and Website Today!!* [online]. 2016 [cit. 2016-04-02]. Dostupné z: https://webandmobiletech.wordpress.com/2016/01/01/top-5-best-php-frameworks-for-2016-to-become-a-master-developer/

Top 6 PHP frameworks for 2016. *CodeandTuts* [online]. 2016 [cit. 2016-02-05]. Dostupné z: http://www.codeandtuts.com/top-6-php-frameworks-for-2016/

W3Schools Online Web Tutorials [online]. 2016 [cit. 2016-02-26]. Dostupné z: http://www.w3schools.com/default.asp

Why Choose PHP Zend Framework Development? *Easy Way To create your Mobile and Website Development - Build a Mobile App and Website Today!!* [online]. 2016 [cit. 2016-04-03]. Dostupné z: https://webandmobiletech.wordpress.com/2016/03/18/why-choose-php-zend-framework-development/

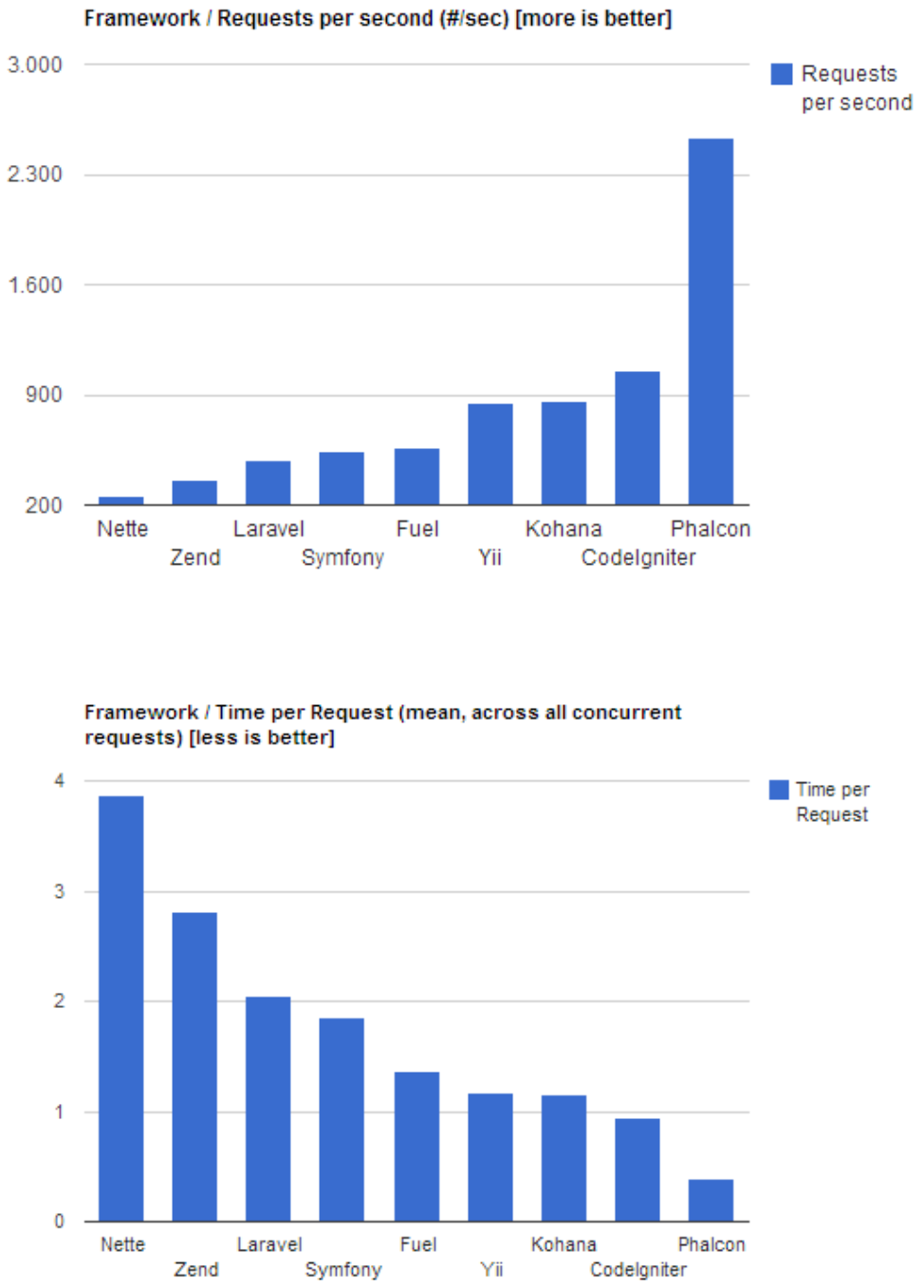
Zend Framework [online]. ©2006-2016 [cit. 2016-04-03]. Dostupné z: http://framework.zend.com/

10Seznam obrázků

11 Seznam tabulek

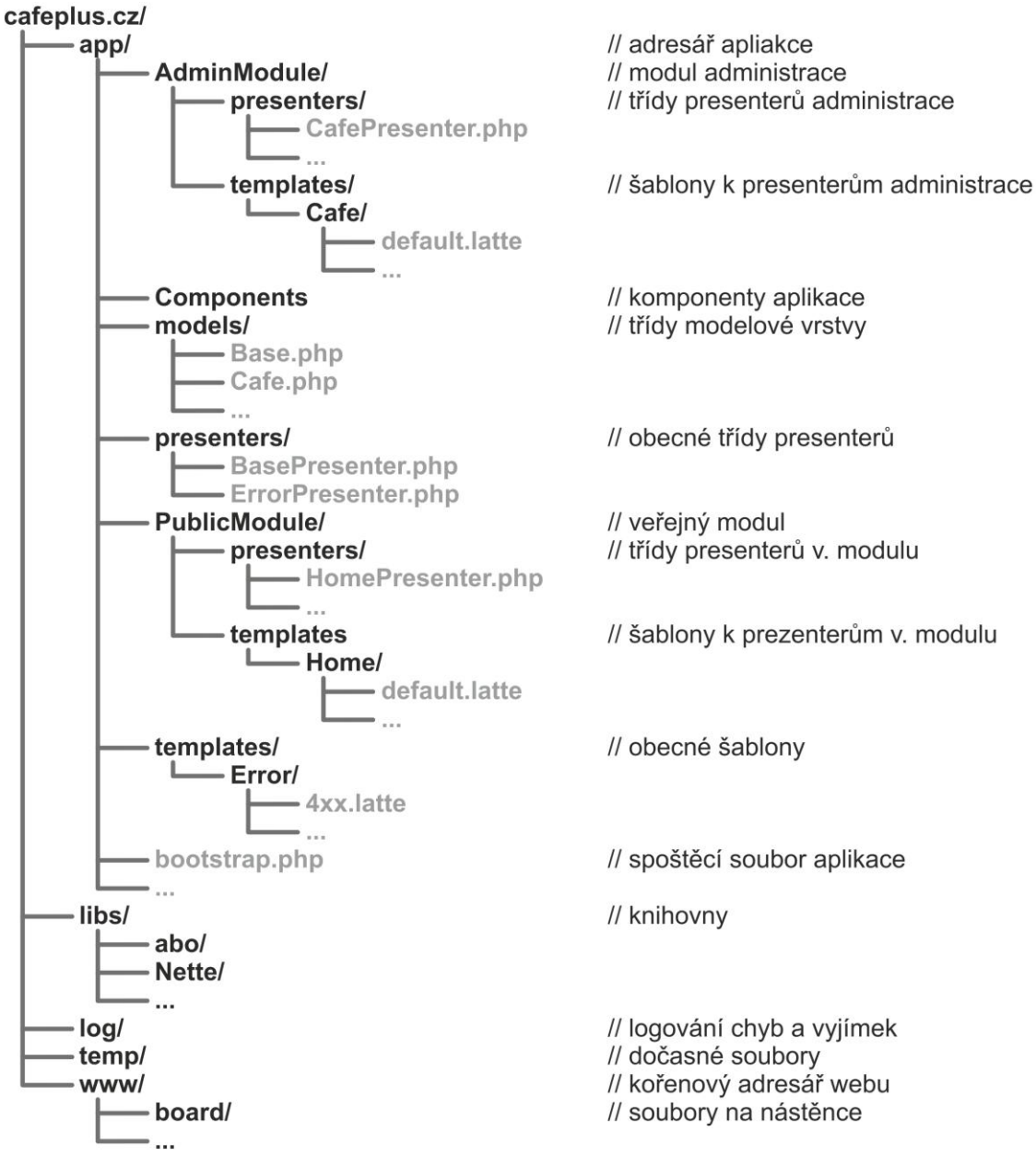
Přílohy

A Testy výkonností PHP frameworků



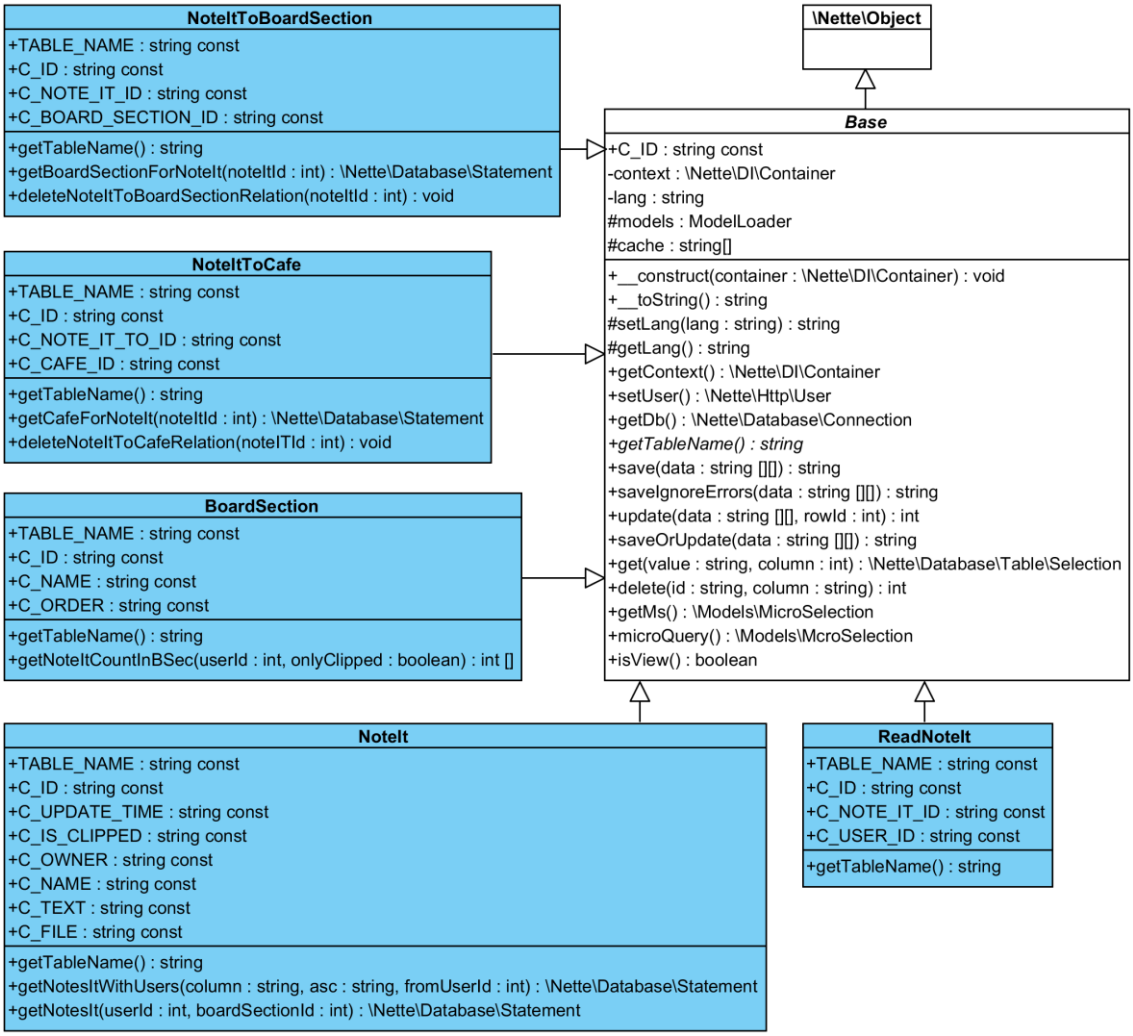
Obr. 42 Testy výkonností PHP frameworků
zdroj: PHP Master | PhalconPHP: Yet Another PHP Framework?, 2016

B Adresářová struktura aplikace

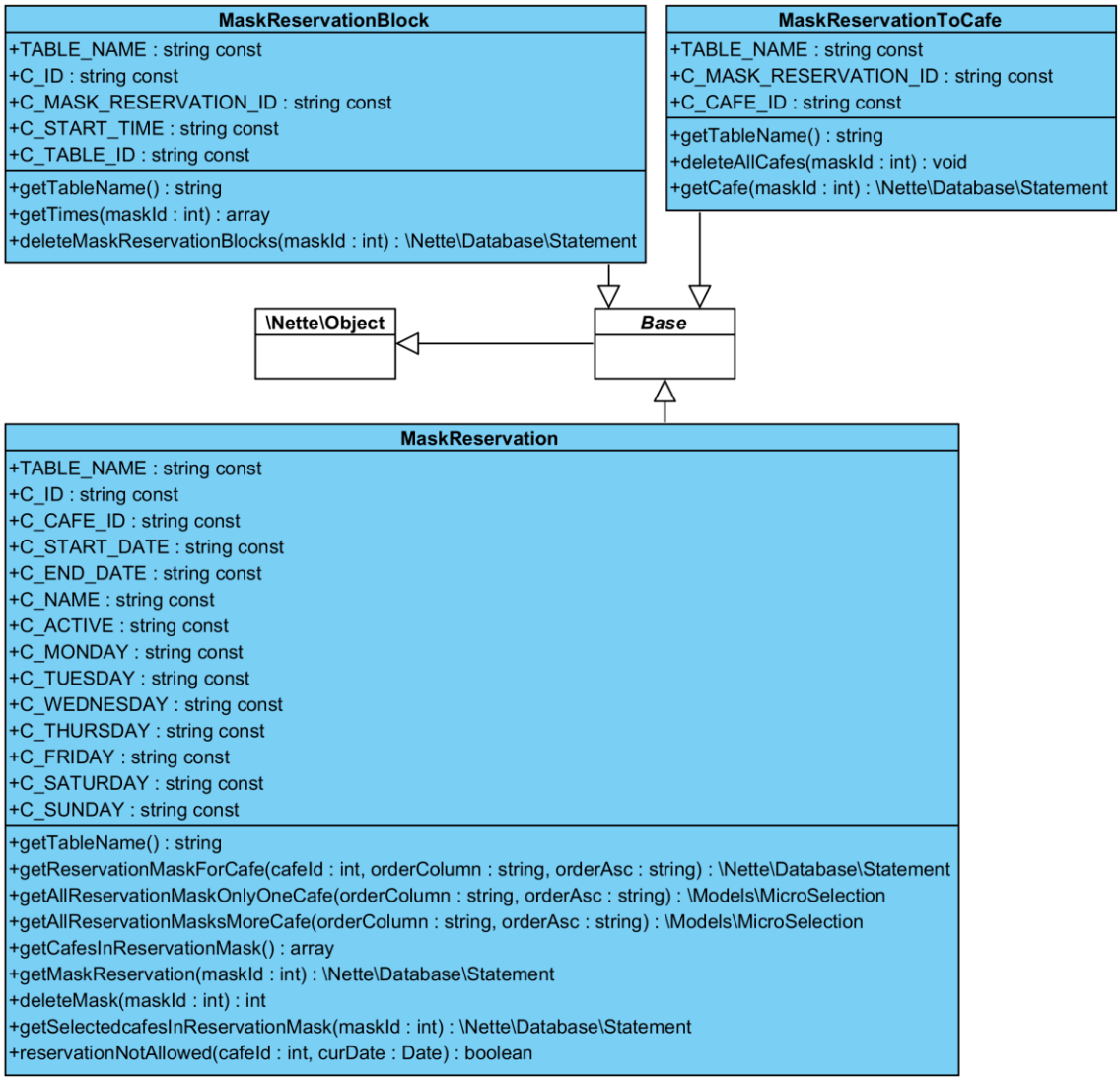


Obr. 43 Adresářová struktura aplikace

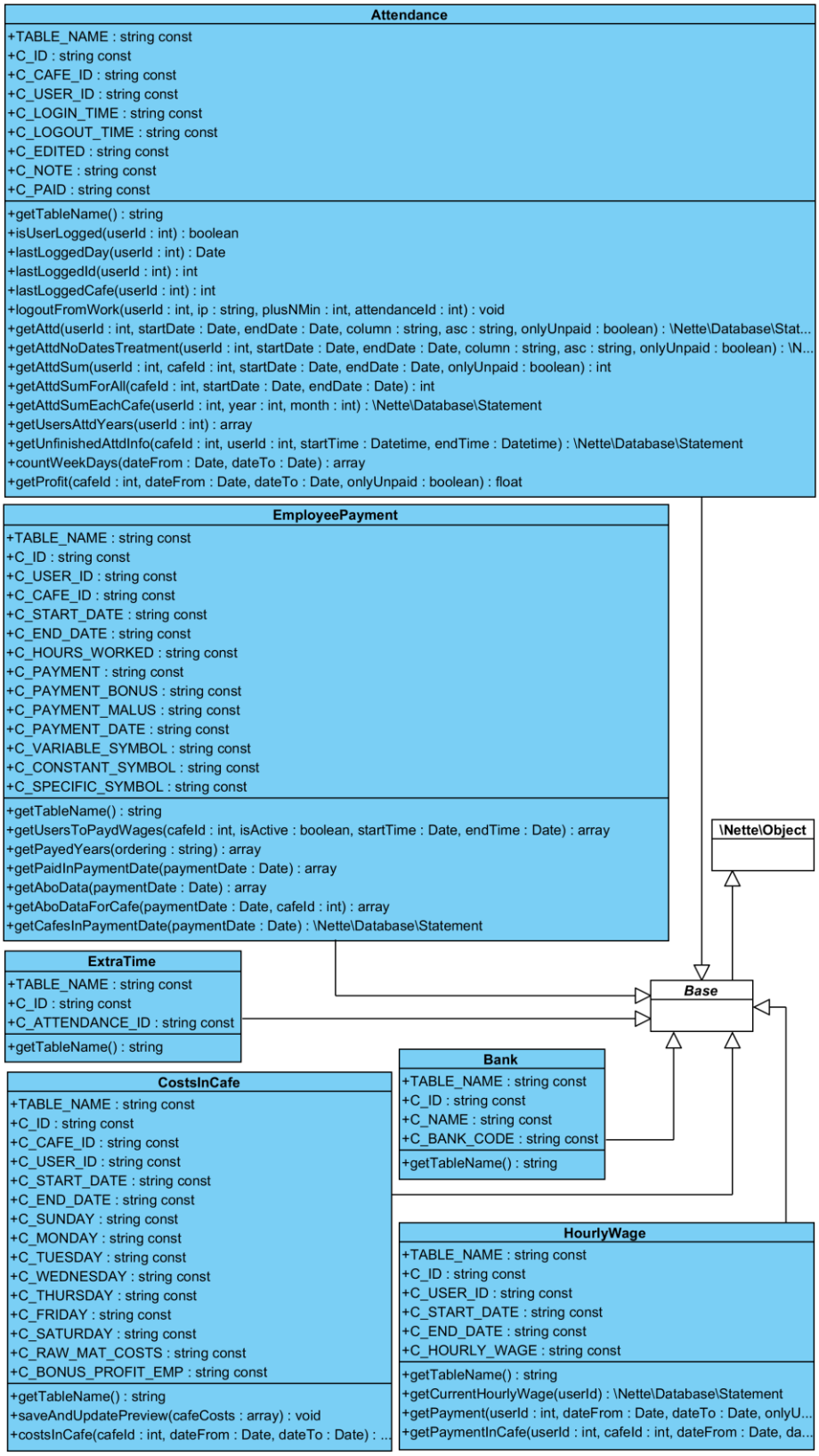
C Diagramy tříd modelů



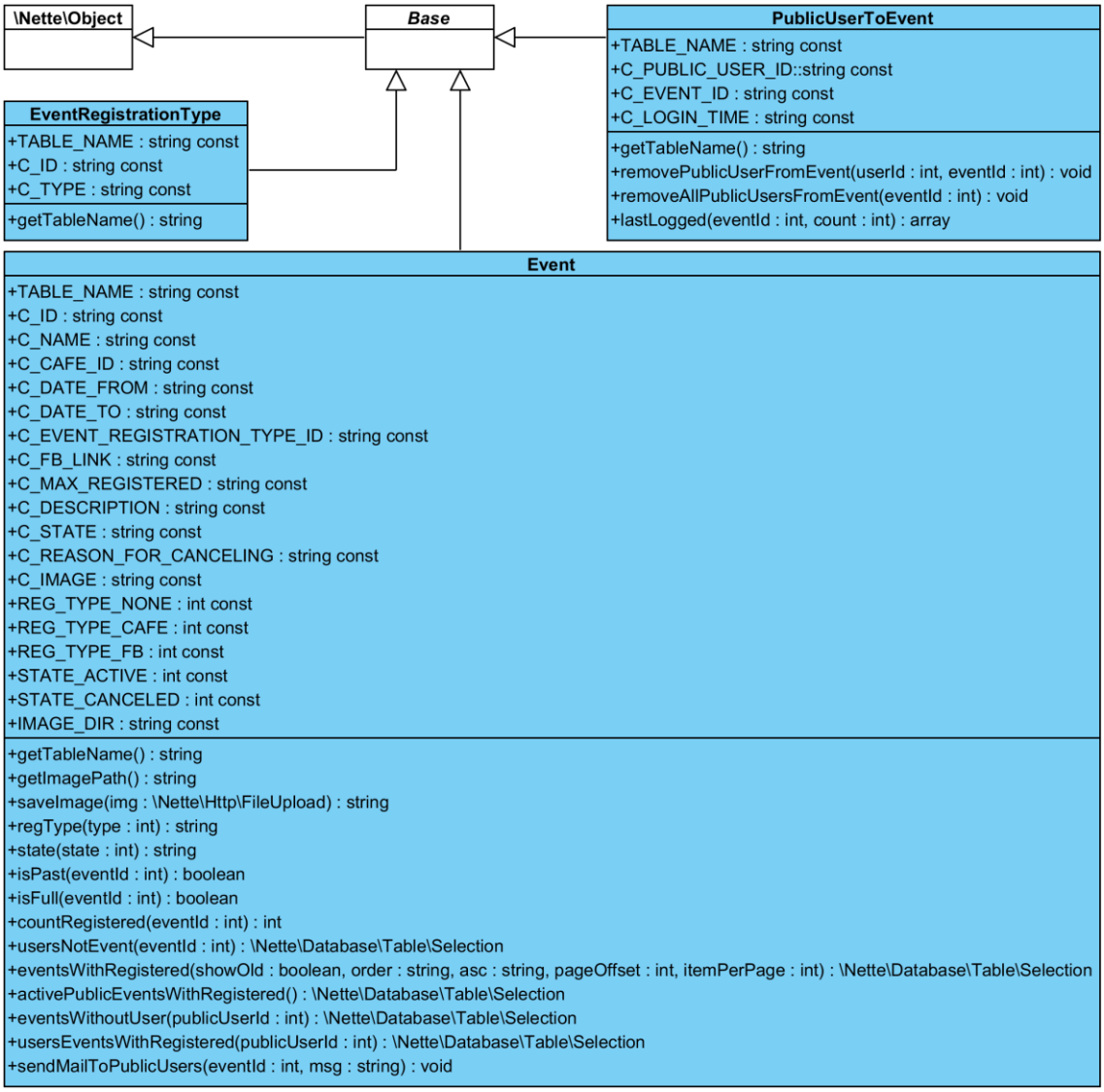
Obr. 44 Diagram tříd modelů pro nástěnku



Obr. 45 Diagram tříd modelů pro rezervační masky

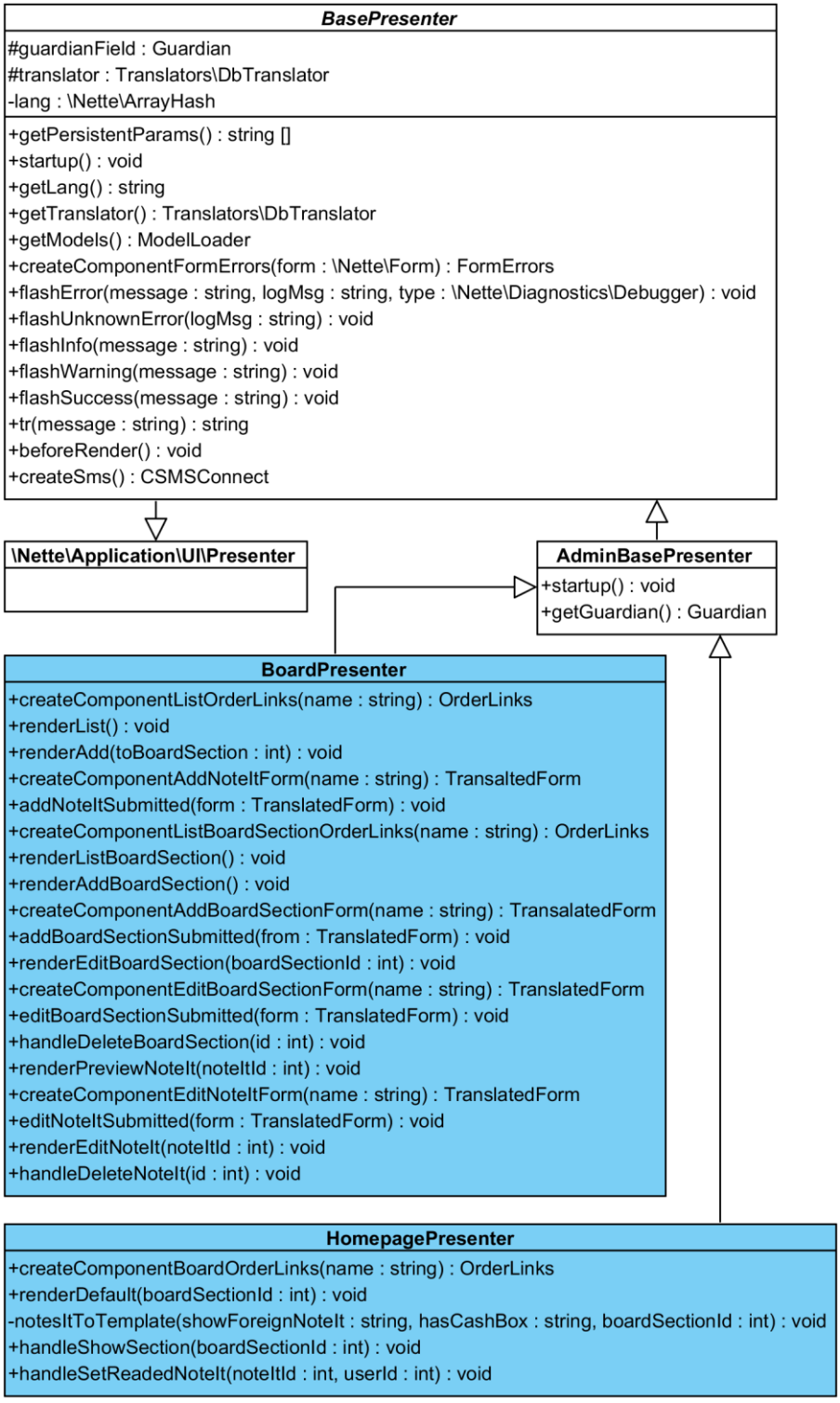


Obr. 46 Diagram tříd modelů pro docházku

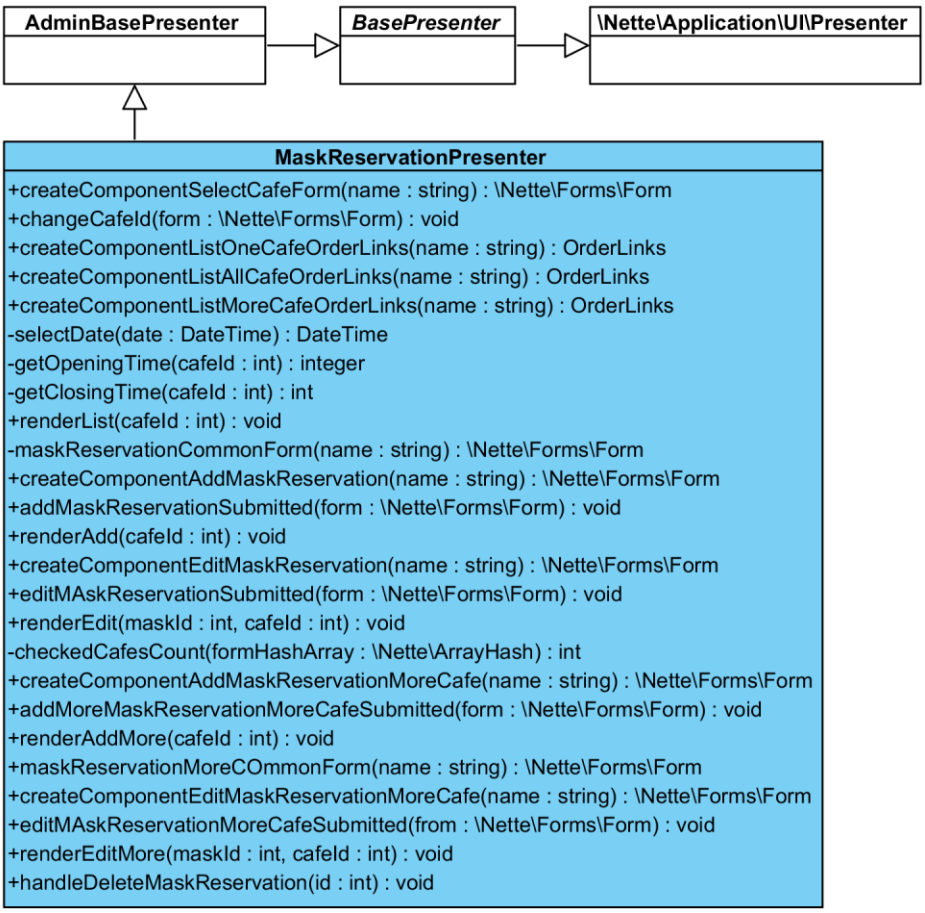


Obr. 47 Diagram tříd modelů pro události

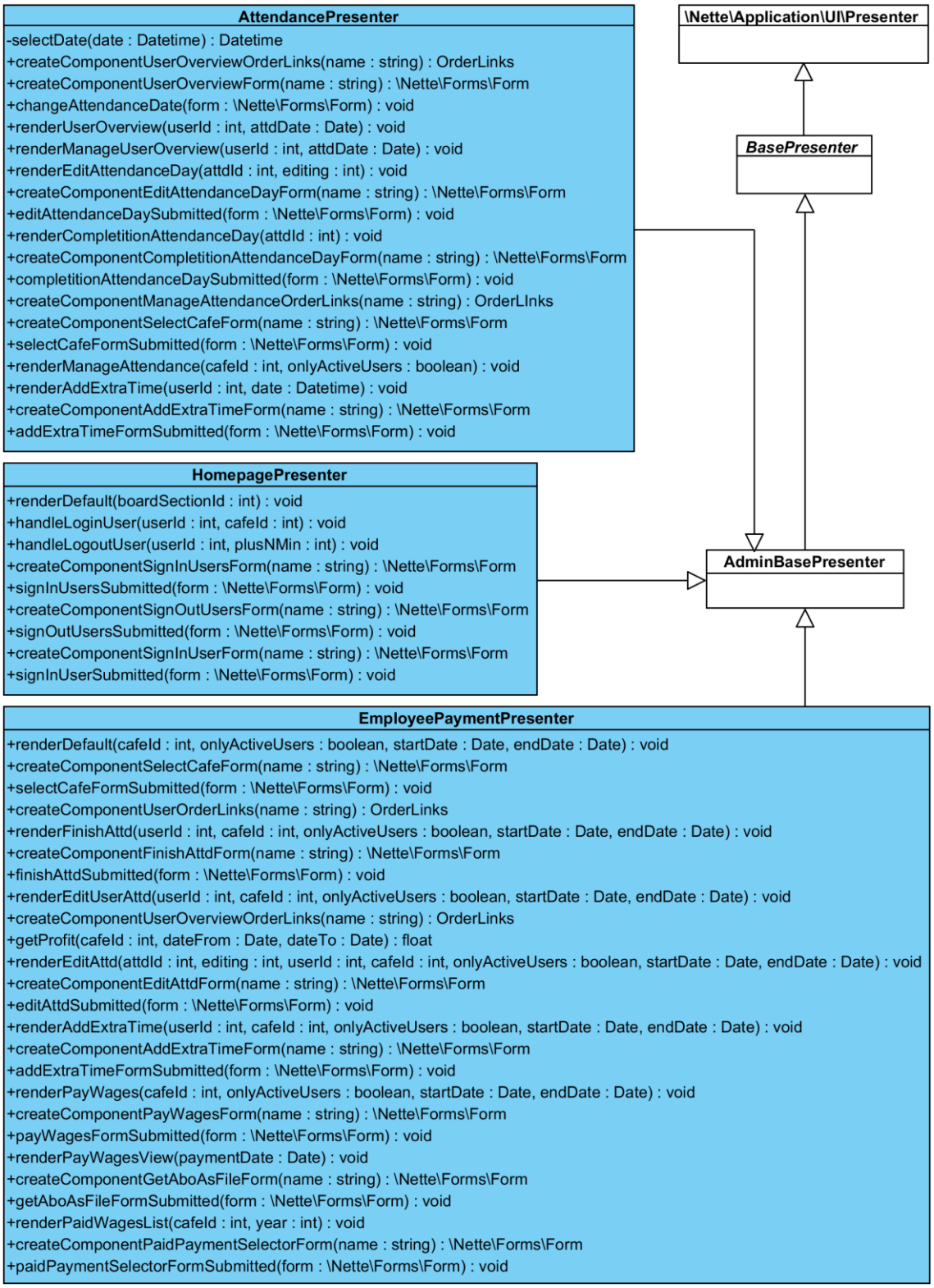
D Diagramy tříd pro presentery



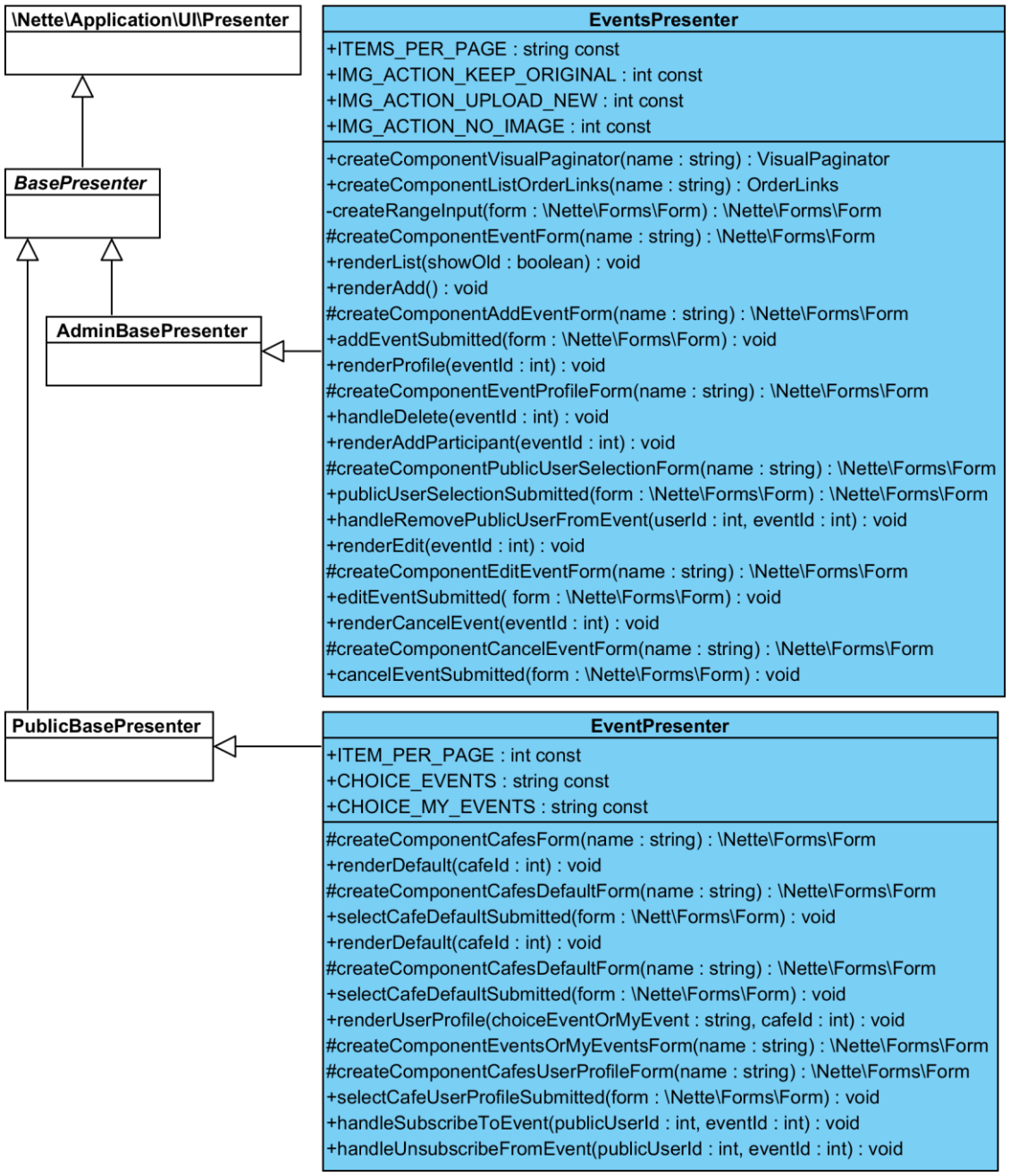
Obr. 48 Diagram tříd pro presentery nástěnky



Obr. 49 Diagram tříd pro presenter masky rezervací

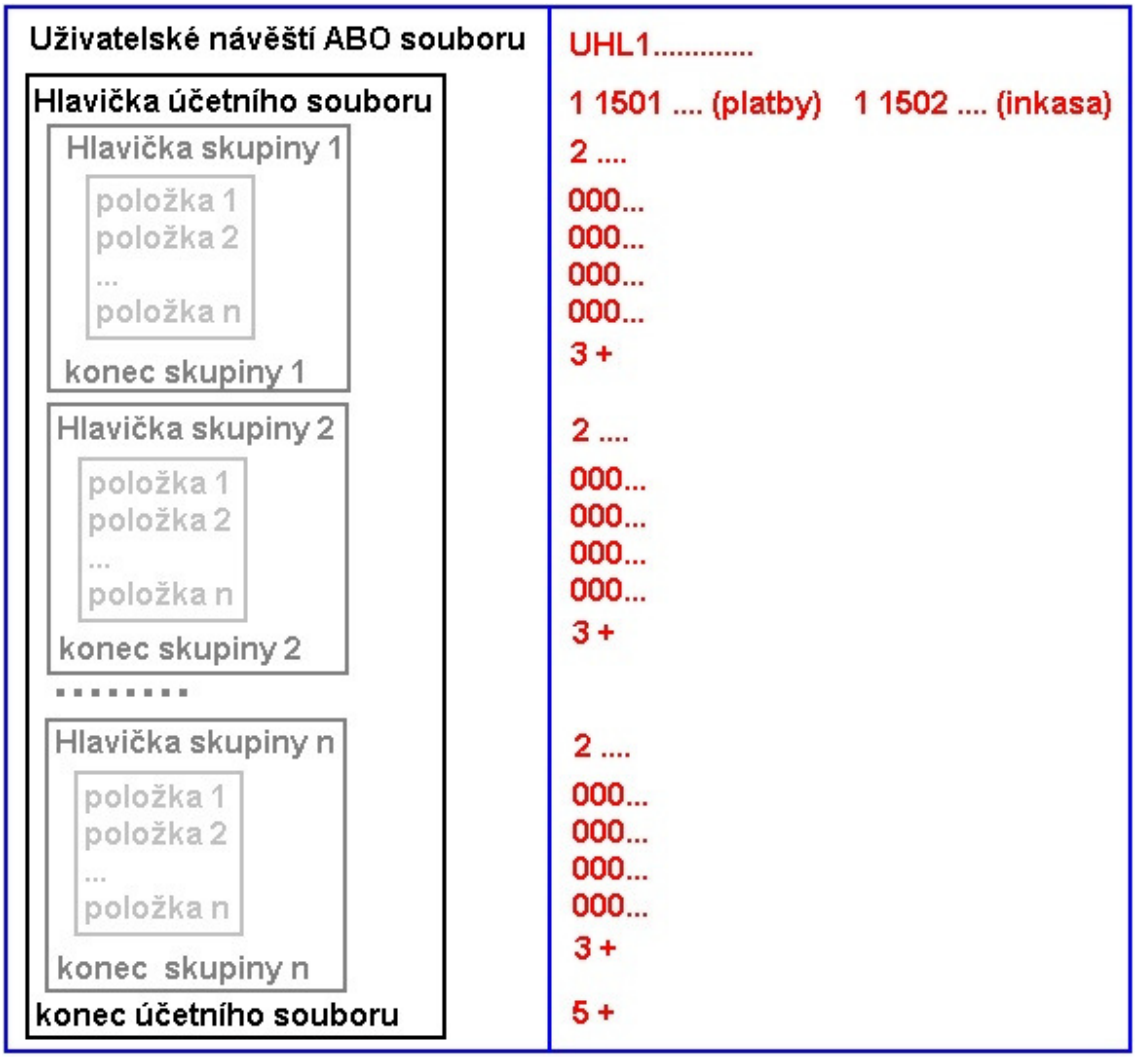


Obr. 50 Diagram tříd pro presentery docházky



Obr. 51 Diagram tříd pro presentery událostí

E Ukázka struktury ABO souboru (Fio banka)



Obr. 52 Ukázka struktury ABO souboru (Fio banka)
Zdroj: Fio banka - česká banka pro váš účet nebo investice | Fio banka, 2016