VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

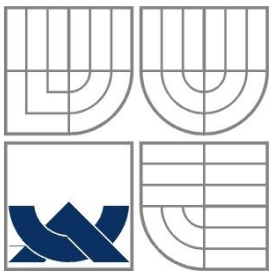
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INFORMAČNÍ SYSTÉM PRO VÝZKUMNOU SKUPINU DIAGNOSTIKA

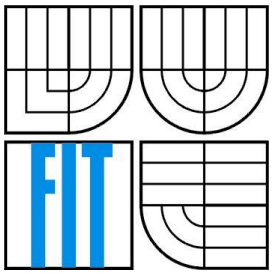
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PAVEL VAĎURA
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INFORMAČNÍ SYSTÉM PRO VÝZKUMNOU SKUPINU DIAGNOSTIKA

DIAGNOSTICS RESEARCH GROUP INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR‘S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL VAĎURA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ONDŘEJ ČEKAN

BRNO 2015

Abstrakt

Tato práce pojednává o návrhu a implementaci informačního sytému pro výzkumnou skupinu. V práci jsou popsány základní technologie pro tvorbu informačního systému, jako je PHP, PHP frameworky, MySQL a jQuery. Další část práce pojednává o specifikaci, analýze, návrhu a implementaci reflektující požadavky výzkumné skupiny na funkcionalitu informačního systému.

Abstract

This thesis discusses the design and implementation of an information system for a research group. The basic technologies for the creation of an information system such as PHP, PHP frameworks, MySQL and jQuery are described. Another part deals with the specification, analysis, design and implementation reflecting the requirements of the research group on the functionality of the information system.

Klíčová slova

Informační systém, databáze, PHP, MySQL, MyISAM, InnoDB, Nette Framework, jQuery, CakePHP, ZendFramework Git, Composer, Material design, sitemap, Latte, frontend Framework, Responzivní layot.

Keywords

Information system, database, PHP, MySQL ,MyISAM, InnoDB, Nette Framework, jQuery, CakePHP, ZendFramework Git, Composer, Material design, sitemap, Latte, frontend Framework, Responsive layot.

Citace

Pavel Vaďura: Informační systém pro výzkumnou skupinu diagnostika, bakalářská práce, Brno, FIT VUT v Brně, 2015

Informační systém pro Výzkumnou skupinu diagnostika

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ondřeje Čekana. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Vaďura
Datum (25. 4. 2015)

Poděkování

Děkuji vedoucímu práce panu Ing. Ondřejovi Čekanovi za odbornou pomoc.

Obsah

1 Úvod

Aktuálním celosvětovým trendem je snaha převádět většinu informací každodenního života do počítače, zejména na internet do databází. Dalo by se říct, že čím častější je možnost připojení k internetu, tím více člověk využívá přístupu k informacím pomocí webových aplikací. Příkladem může být poznámkový blok. Dříve postačoval notes a tužka. Nevýhodou bylo, že ztráta notesu znamenala ztrátu veškerých informací do něj zapsaných. Použití webové aplikace jako poznámkového bloku znamená, že informace se nemohou ztratit, pomineme-li fatální selhání zálohovacího systému serveru, na kterém aplikace běží. Další výhodou je multiplatformní přístup k dané aplikaci, požadavkem je jen přístroj, který se dokáže připojit k internetu. Takový přístroj může využít buď webovou aplikaci přístupnou přes prohlížeč internetových stránek, nebo nativní aplikaci připojující se přímo k datům.

Cílem práce je navrhnout a realizovat informační systém pro Výzkumnou skupinu diagnostika, ke kterému se bude přistupovat pomocí webového rozhraní přístupného přes internetový prohlížeč. Systém bude primárně umět spravovat události (schůzky) a plánovat termín uskutečnění dané schůzky. Předlohou pro plánování bude aplikace Doodle [1], kdy si každý pozvaný člen zvolí nabídnutý termín, který mu vyhovuje. Každý člen bude spravovat informace o své činnosti během roku. Tyto informace bude možno přehledně prezentovat v časové ose. Dále systém bude zaznamenávat výpůjčky věcí mezi jednotlivými členy. Navržený systém bude mít sekci určenou pro veřejnost, kterou správce systému bude moci upravovat pomocí webového rozhraní.

Správa webového rozhraní, vkládání informací o zápisu schůzek a činnosti uživatelů bude realizována prostřednictvím redakčního sytému. Momentálně se na trhu vyskytuje velké množství redakčních systémů. Jsou to systémy obecného použití, které obsahují velké množství funkcí a nástrojů, ale také úzce specializované systémy vytvořené pro přesně stanovené účely. Pro potřeby navrhované aplikace nebude vhodné použít systém obecného použití, jelikož velká část funkcí by nebyla využita a redakční systém by tak zbytečně svou velikostí zatěžoval informační systém. Vhodnějším řešením je vytvořit redakční systém navržený přesně pro potřeby konkrétní výzkumné skupiny.

Text této práce je rozdělen do několika kapitol. Postupně jsou popsány použité technologie, návrh a implementace aplikace. V závěru práce jsou uvedeny nástroje použité pro testování systému spolu s postupem instalace aplikace na webový server. Poslední část hodnotí dosažené výsledky.

Druhá kapitola pojednává o vybraných technologiích použitých pro tvorbu webového informačního systému. Tento oddíl je zaměřen na skriptovací jazyk PHP, databázový systém MYSQL, JavaScriptovou knihovnu jQuery a další použité technologie. Dále bude představen vybraný vzorek PHP frameworků, z nichž bude jeden použit pro implementaci navrhovaného systému.

Třetí kapitola se věnuje návrhu systému. Je zde popsán use case diagram pojednávající o činnostech, které bude systém vykonávat. Konceptuální diagram tříd informačního systému zobrazuje statické struktury a vztahy mezi těmito třídami. Dále je uveden vybraný reprezentativní vzorek případů užití navrženého systému. Závěr kapitoly pojednává o návrhu databáze pomocí konceptuálního diagramu tříd, který je následně převeden na schéma databáze.

Čtvrtá kapitola se zabývá implementací sytému a vybranými implementačními prostředky. Podoba adresářové struktury záleží na vybraném frameworku použitém pro realizaci. Následuje deskripce principů aplikace rozdělené na veřejnou a administrační část. Popsány jsou také vlastnosti Nette Frameworku spolu s tvorbou základních prvků systému. Závěrem je popsán layout systému a použitá technika pro vytvoření responsivního layoutu.

Kapitola pátá stručně popisuje způsoby testování redakčního systému, od grafického návrhu přes funkcionalitu systému až po rozesílání e-mailových zpráv uživatelům.

Šestá kapitola popisuje způsob instalace a přípravy demonstrační aplikace na webovém serveru.

Závěrečná sedmá kapitola zhodnocuje výsledky redakčního systému s možným rozšířením pro budoucí vývoj.

2 Použité technologie

Tato kapitola se zabývá technologiemi použitými na implementaci informačního systému. Dále jsou v kapitole 2.2 alternativní PHP frameworky, které je možno použít pro dosažení potřebného výsledku. V kapitolách 2.5 a 2.6 jsou popsány nástroje, které nejsou pro realizaci navrhovaného systému potřebné, jako např. nástroj Composer. Použitím těchto nástrojů dojde k zefektivnění práce a k předcházení možných problémů. Jelikož je potenciál pro reálné použití systému a jeho další budoucí rozvoj, je vhodné spravovat zdrojový kód pomocí verzovacího systému, který umožní úpravu systému pro více vývojářů.

2.1 PHP

PHP [2] (*Hypertext Preprocessor*) je interpretovaný skriptovací jazyk syntaxí podobný jazyku C. Nejčastěji se používá pro tvorbu internetových stránek a dynamických internetových aplikací, ale je možné jej využít také pro tvorbu desktopové aplikace použitím kompilované verze. Aplikace vytvořené v PHP skriptovacím jazyku jsou uloženy na serveru. Obvykle se při dotazu klienta na server spustí patřičný PHP skript, který po vykonání samotného skriptu na straně serveru uživateli zobrazí výsledek jeho činnosti, např. HTML stránku. Používáním PHP jazyka nedochází k vytváření žádných perzistentních proměnných, po vykonání požadavku si server neukládá žádné informace o proběhlé akci.

2.2 PHP framework

PHP framework je objektově orientovaný systém implementovaný v jazyce PHP. Framework pracuje s komponentami, které jsou realizovány jako abstraktní třídy a jsou integrovány ve frameworku. Interakce ve frameworku jsou definovány pomocí tříd, instancí, dědičnosti, polymorfismu a pravidel kompozice. Součástí frameworku jsou podpůrné programy, knihovny pro zjednodušení práce a řešení opakujících se činností, např. komunikace s databází a tvorbu transakčních dotazů.

Existuje mnoho PHP frameworků s různými vlastnostmi, zde bude uvedeno pouze několik vybraných příkladů. [3]

2.2.1 Nette Framework

Nette Framework byl vytvořen českým autorem Davidem Grudlem, který ho zveřejnil pod licencí GNU GPL. Nyní je Framework vyvíjen společností Nette Foundations. Nette Framework je českým projektem těšícím se největší české komunitě. Tento fakt dokazuje i dokumentace, která je psaná z větší části v češtině. Nicméně aktuálně je snaha projekt prosadit i mimo Českou republiku, s čímž souvisí transformace dokumentace a internetových stránek do angličtiny. Důležitým rysem Nette je softwarová struktura MVC, jež bude popsána v kapitole 3.4. Návrh frameworku využívá objektového návrhu postaveného na PHP 5 a komponentách. Jedním ze specifik tohoto frameworku je vlastní šablonovací systém, který používá pohledy (šablony). Vlastní řešení spočívá například v podobě zápisu odkazů nebo možnosti použít makra. Tato makra umožňují zapisovat cykly, podmínky, vypisovat proměnné atd. [4] Tento framework bude použit pro řešení projektu. Důvodem

pro použití právě Nette Frameworku je, že se jedná o jeden z nejvýkonnějších dostupných frameworků. [5] Dalším důvodem jsou pokročilé ladící nástroje popsané v kapitole 5 a v neposlední řadě již zmiňovaná největší česká komunita.

2.2.2 CakePHP

CakePHP je open source projekt zveřejněný pod MIT licencí, který vyvíjí Cake Software Foundation. Framework je postavený na PHP se softwarovou strukturou MVC a základy projektu jsou inspirované Ruby on Rails. Framework se vyznačuje krátkým a přehledným kódem, obsáhlou správou databáze a pokročilými nástroji pro práci s pohledy systému. Obsahuje mnoho připravených komponent pro použití ve vývoji, např. komponenta pro práci s e-mailem, autentifikaci apod. Pohledy jsou přehledně rozděleny na několik kategorií, a tak je umožněna snadná změna vzhledu prvků i celých stránek. CakePHP lze kombinovat se Zend Frameworkem, a tím využít předností obou frameworků. [6]

2.2.3 Zend Framework

Zend Framework je jeden z nejrozšířenějších open source PHP frameworků zveřejněný pod New BSD licencí. Zend framework je vyvíjen společností Zend Technologies Ltd., kterou založili Andi Gutmans a Zeev Suraski v roce 1997. Framework je objektově orientovaný, postavený na PHP 5.3, využívající modulárnosti a jmenných prostorů. Vývoj frameworku se zaměřuje na stabilitu a kompatibilitu knihovny, což má za následek, že se do finálních verzí dostanou jen změny, které jsou řádně otestovány. Nejen díky stabilitě frameworku má projekt velmi silnou uživatelskou základnu. Nevýhodou zvoleného způsobu vývoje je pomalá reakce na nové neosvědčené technologie. Této nevýhody využívají velké projekty, které potřebují, aby framework byl co nejstabilnější a zpětně kompatibilní. [7]

2.3 MySQL

MySQL (My Structured Query Language) je relační databázový systém typu DBMS (database managment system), který vychází z deklarativního programovacího jazyka SQL (Structured Query Language). Systém je zveřejněn pod dvěma licencemi - pod bezplatnou licencí GPL a pod komerční licencí. Jedná se o nejrozšířenější a nejoblíbenější systém, který klade důraz na rychlost a výkon, a to za cenu zjednodušení určitých částí systému, např. způsobů zálohování. Do MySQL je možné ukládat různá data (texty, obrázky atd.), s nimiž lze dále jednoduše pracovat (třídit, řadit, filtrovat apod.). Nejčastěji se MySQL používá ve spojení s jazykem PHP, který umožňuje přístup k uloženým datům. Každá databáze v MySQL ukládá data do předem definovaných tabulek. Pro jednodušší správy databáze se používají různé nástroje, např. phpMyAdmin. [8]

2.4 jQuery

Multiplatformní JavaScriptová knihovna jQuery [9] je navržená pro spouštění na straně klienta spolu s HTML kódem. Knihovnu představil v roce 2006 její autor Johh Resig pod svobodnou licencí MIT. Knihovna jQuery umožňuje:

- procházet a upravovat DOM (Document Object Model)

- vytvářet reakce na události
- manipulovat s CSS
- efekty a animace
- AJAX (Asynchronous JavaScript and XML) - načítání obsahu serveru bez nutnosti obnovení stránky
- podporuje mnoho plug-inu
- obsahuje mnoho utilit – např. informace o prohlížeči

2.5 Composer

Composer je nástroj, který poskytuje standartní formát pro správu závislostí a knihoven programovacího jazyka PHP. Tento nástroj vyvinuli vývojáři Nils Adermann a Jordi Boggiano v roce 2011. Composer umožňuje pomocí příkazové řádky nainstalovat do projektu závislost/knihovnu. Tímto způsobem je umožněno nainstalovat rozšíření třetích stran do vlastního projektu. Instalací je myšleno stažení kódu z databáze (Packaglist), umístění do zvoleného projektu a připravení na použití. Další výhodou přidání rozšíření do projektu pomocí Composeru je možnost aktualizace závislostí a rozšíření. [10] V bakalářské práci bude Composer využit pro získání aktuálního Nette frameworku a k následnému doplnění jeho rozšíření, jako je například rozšíření o Kdyby/Replicator. [11]

2.6 Git

Git je open source distribuovaný systém zveřejněný pod licencí GPL verze 2, sloužící ke správě jednotlivých verzí. Autorem systému je Linus Torvalds a původní použití bylo určeno pro vývoj jádra Linuxu. Hlavní výhodou systému Git je podpora pro nelineární distribuovaný vývoj, tzn. lze snadno vytvářet a slučovat větve projektu. Díky distribuovanému vývoji může mít každý programátor lokální kopii projektu na své pracovní stanici a po dokončení prací může změny zapsat buď do hlavní nebo případně do vedlejší větve s tím, že se později (po provedení testů a korekce) vedlejší větev sloučí s hlavní. Tento postup umožňuje mít hlavní větev projektu, kde se nachází hlavní verze programu, která je odladěná a funkční. Ve vedlejších větvích jsou umístěny úpravy a rozšíření projektu, jež lze po schválení jednoduše začlenit do hlavní větvě. Spojení úprav zde zajistí Git. [12]

3 Návrh informačního systému

Tato kapitola se zabývá návrhem informačního systému. Je zde uvedena neformální specifikace, ze které vychází diagram případů užití a jejich vybrané detaily. Závěrem je popsán diagram tříd a z něho vytvořený návrh databáze.

3.1 Neformální specifikace

Cílem práce je vytvořit webovou aplikaci, pomocí které bude možné spravovat plánování událostí. Přihlášený uživatel vytvoří událost, na niž pozve zvolené uživatele. Nejdůležitějším prvkem je plánování data uskutečnění vybrané události. Všichni pozvaní členové si pomocí webové aplikace zvolí, které termíny jim vyhovují, a poté zakladatel vybere jeden z termínů vyhovující všem účastníkům. Po stanovení závazného termínu akce budou všichni účastníci upozorněni na událost prostřednictvím e-mailu. Tyto e-maily bude systém automaticky generovat s ohledem na čas zbývající do začátku události, a to týden a den před jejím začátkem. Po skončení události je možné přidat do systému zápis z události.

Každý uživatel bude mít možnost reportovat o svojí činnosti v podobě roční zprávy a bude moci do systému vkládat úkoly. Úkolem je myšlen záznam o činnosti, který obsahuje informaci o datu začátku činnosti, popisu činnosti a předpokládaném datu ukončení prací na úkolu. K vytvořenému úkolu bude možné přidávat průběžné zprávy o splněných milnících úkolu a zprávu o jeho dokončení. Vložené zprávy se budou zobrazovat do časové osy, která bude přístupná i ostatním uživatelům systému. Systém bude zároveň kontrolovat termíny a podobně jako u schůzek bude upozorňovat na blížící se datum dokončení úkolu.

Systém bude zaznamenávat výpůjčky položek mezi členy skupiny. Uživatel zapůjčující položku vloží do systému záznam o datu půjčení, informaci o položce a uživateli, který si danou položku půjčuje.

Uživateli s oprávněním administrátora bude umožněn přístup do administrátorské sekce, kde bude moci vytvářet, upravovat a mazat obsah webové stránky ve veřejné části. Tato webová stránka slouží k prezentaci činnosti výzkumné skupiny pro širokou veřejnost. Dále zde bude mít možnost vytvořit přístup do systému pro nového člena a upravovat práva členů.

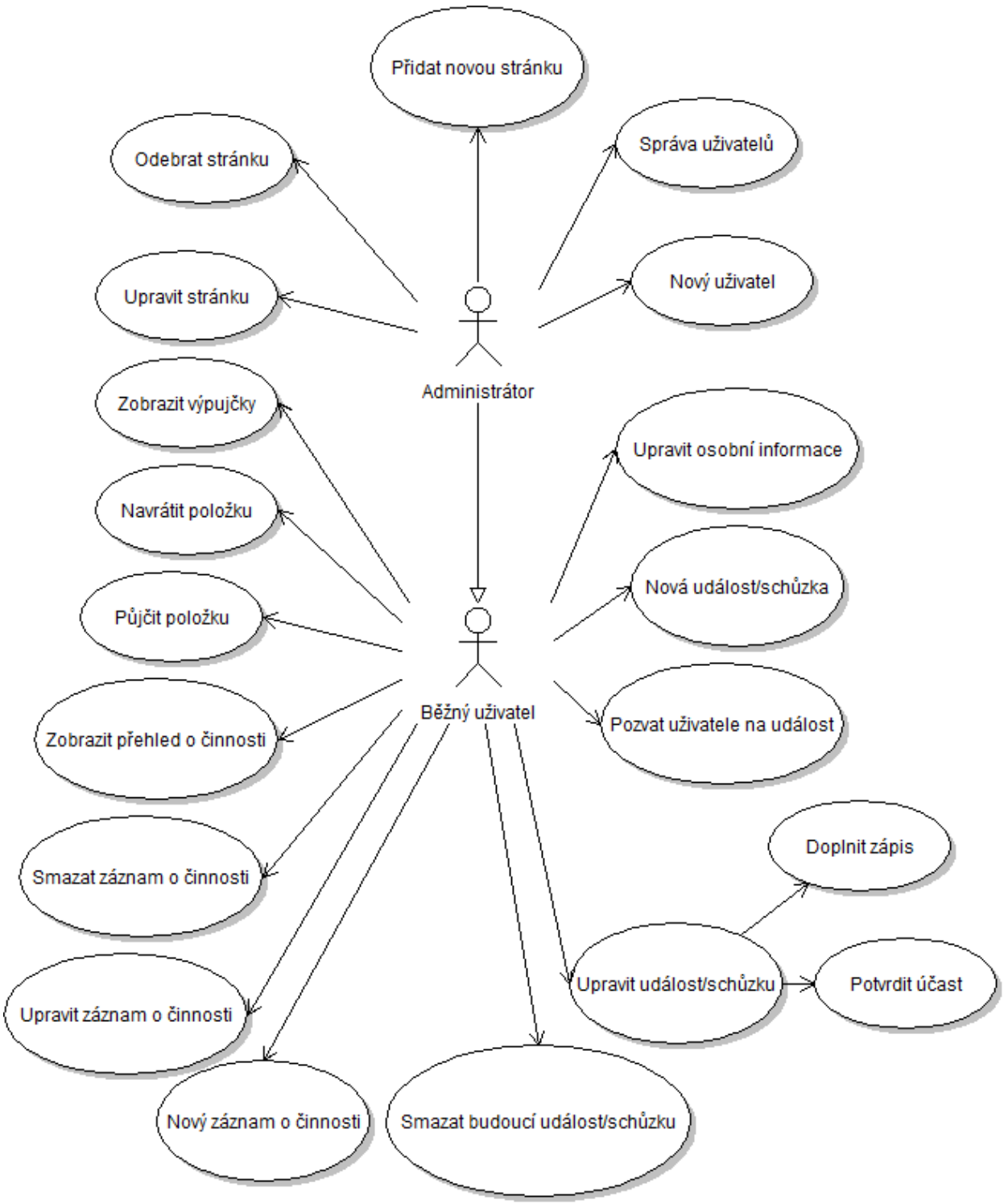
3.2 Analýza

Analýza vychází z neformální specifikace, podle níž by měl navržený systém rozlišovat dva typy rolí, respektive tři. Uživatel bez přístupu do systému může zobrazit pouze stránky pro veřejnost, nemá navrhnutou konkrétní roli, a proto není modelován.
Systém bude uvažovat tyto role:

1. Běžný uživatel – aktér, který je přihlášený do systému. Může vytvářet a spravovat vlastní schůzky, posílat pozvánky, vkládat zápis schůzky a potvrzovat účast na schůzkách. Běžný uživatel smí vkládat a upravovat zprávy o svojí činnosti, nahlížet na přehled činností ostatních členů a vkládat záznamy o výpůjčkách.

2. Administrátor – aktér, který rozšiřuje činnosti Běžného uživatele. Administrátor smí vytvářet nové Běžné uživatele, vytvořeným uživatelům upravovat práva a spravovat stránky určené veřejnosti, tzn. přidávat novou stránku, stávající stránky upravovat a odebírat již vytvořené stránky.

Navrhnutý diagram případů použití je zobrazen na obrázku 3.1.



Obrázek 3.1: Diagram případů použití

3.2.1 Detaily případu použití

V následujících tabulkách je uveden reprezentativní vzorek detailů případů použití.

Případ použití:	**Nový uživatel**	**ID:**	**1**
Stručný popis:	Vytvoření nového uživatele v systému.		
Primární aktéři:	Administrátor		
Předpoklady:	1. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém vytvoří nového uživatele spolu s přihlašovacími údaji.		
Hlavní tok:	1. Aktér přejde do sekce „Nový uživatel“. 2. Aktér vyplní vstupní formulář. 3. Aktér potvrdí vstupní data tlačítkem „Vytvořit“. 4. Pokud vstupní data nejsou validní, systém na dané chyby upozorní a vyzve uživatele k jejich napravení. V případě, že je vše v pořádku, přejde se na krok 5. 5. Systém vytvoří nového uživatele spolu s přihlašovacími údaji.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do roka.		

Případ použití:	**Nová událost/Schůzka**	**ID:**	**2**
Stručný popis:	Aktér vytvoří novou událost.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém vytvoří novou událost.		
Hlavní tok:	1. Aktér přejde do sekce „Nová událost“ 2. Aktér zadá informace k vytvářené události. 3. Pokud vstupní data nejsou validní, systém na dané chyby upozorní a vyzve uživatele k jejich napravení. V případě, že je vše v pořádku, přejde se na krok 4. 4. Systém vytvoří novou událost 5. Systém nabídne přechod na případ použití „Pozvat uživatele“ vztahující se k vytvořené události.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do měsíce.		

Případ použití:	**Pozvat uživatele na událost**	**ID:**	**3**
Stručný popis:	Aktér pozve ostatní členy na svoji událost.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je přihlášen do systému. 2. Aktér vytvořil událost, která se ještě neuskutečnila.		
Následné podmínky:	1. Systém pozve zvolené uživatele na událost.		
Hlavní tok:	1. Aktér zvolí, kteří uživatelé budou pozváni na akci. 2. Systém pozve zvolené uživatele na událost.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do měsíce.		

Případ použití:	**Potvrdit účast**	**ID:**	**4**
Stručný popis:	Aktér potvrdí svou účast na akci.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je pozván na událost. 2. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém potvrdí účast uživatele zaznamenáním vyhovujících časů konání události.		
Hlavní tok:	1. Aktér vstoupí na podrobnost dané akce 2. Aktér si vybere možné termíny akce, které mu vyhovují. 3. Systém vybrané termíny zaznamená.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do měsíce.		

Případ použití:	**Doplnit zápis**	**ID:**	**5**
Stručný popis:	Aktér doplní zápis z konané události/akce.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér vytvořil událost, která se již konala. 2. Aktér je přihlášen do systému.		
Následné podmínky:	2. Systém k již konané akci/události přidá zápis z události.		
Hlavní tok:	1. Aktér vyhledá svoje události/akce, které se již konaly. 3. Aktér upraví vybranou akci. 4. Aktér zadá zápis z konané události. 5. Systém uloží zápis.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do měsíce.		

Případ použití:	**Smaž budoucí událost**	**ID:**	**6**
Stručný popis:	Aktér smaže událost/akci, která se ještě nekonala.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér vytvořil událost, která se ještě nekonala. 2. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém smaže danou událost a případné pozvánky na událost.		
Hlavní tok:	1. Aktér vyhledá svoje události/akce, které se ještě nekonaly. 2. Aktér zvolí volbu „Smazat událost“ u vybrané akce. 3. Systém smaže danou událost a případné pozvánky na událost.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do roka.		

Případ použití:	**Zobrazit přehled o činnosti**	**ID:**	**7**
Stručný popis:	Zobrazí se přehled o činnosti daného uživatele.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém zobrazí přehled o činnostech vybraného uživatele.		
Hlavní tok:	1. Aktér zobrazí výpis uživatelů. 2. Aktér zvolí volbu „Přehled o činnosti“ u vybraného uživatele. 3. Systém zobrazí přehled o činnostech vybraného uživatele.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do týdne.		

Případ použití:	**Nový záznam o činnosti**	**ID:**	**8**
Stručný popis:	Do systému se vloží nový záznam o činnosti.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém zaznamená nový záznam o činnosti.		
Hlavní tok:	1. Aktér zobrazí výpis záznamů. 2. Aktér zvolí volbu „Nový záznam“. 3. Aktér vyplní formulář. 4. Systém zaznamená nový záznam o činnosti.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do měsíce.		

Případ použití:	**Zobrazit výpůjčky**	**ID:**	**9**
Stručný popis:	Zobrazí se aktuální výpůjčky knih a zařízení.		
Primární aktéři:	Běžný uživatel		
Předpoklady:	1. Aktér je přihlášen do systému.		
Následné podmínky:	1. Systém zobrazí aktuální výpůjčky knih a zařízení.		
Hlavní tok:	1. Aktér přejde do přehledu výpůjček. 2. Systém zobrazí aktuální výpůjčky knih a zařízení.		
Výjimky:	Selhání systému, Storno		
Frekvence:	Několikrát do týdne.		

Případ použití:	**Selhání systému**	**ID:**	**E.1**
Stručný popis:	V systému se vyskytne chyba a není možné dokončit danou operaci.		
Primární aktéři:	Dle daného případu použití.		
Předpoklady:	1. V systému se vyskytla chyba.		
Následné podmínky:	1. Systému nebyly provedeny změny.		
Tok:	1. Tok selhání systému je vyvolán během provádění hlavního toku. 2. Systém informuje uživatele o chybě.		
Frekvence:	Výjimečně		

Případ použití:	**Storno**	**ID:**	**E.2**
Stručný popis:	Přerušení operace.		
Primární aktéři:	Dle daného případu použití.		
Předpoklady:	1. Aktér provedl storno operaci.		
Následné podmínky:	1. V systému nebyly provedeny změny.		
Tok:	1. Tok Storno může být vyvolán během provádění hlavního toku.		
Frekvence:	Výjimečně		

3.3 Databáze

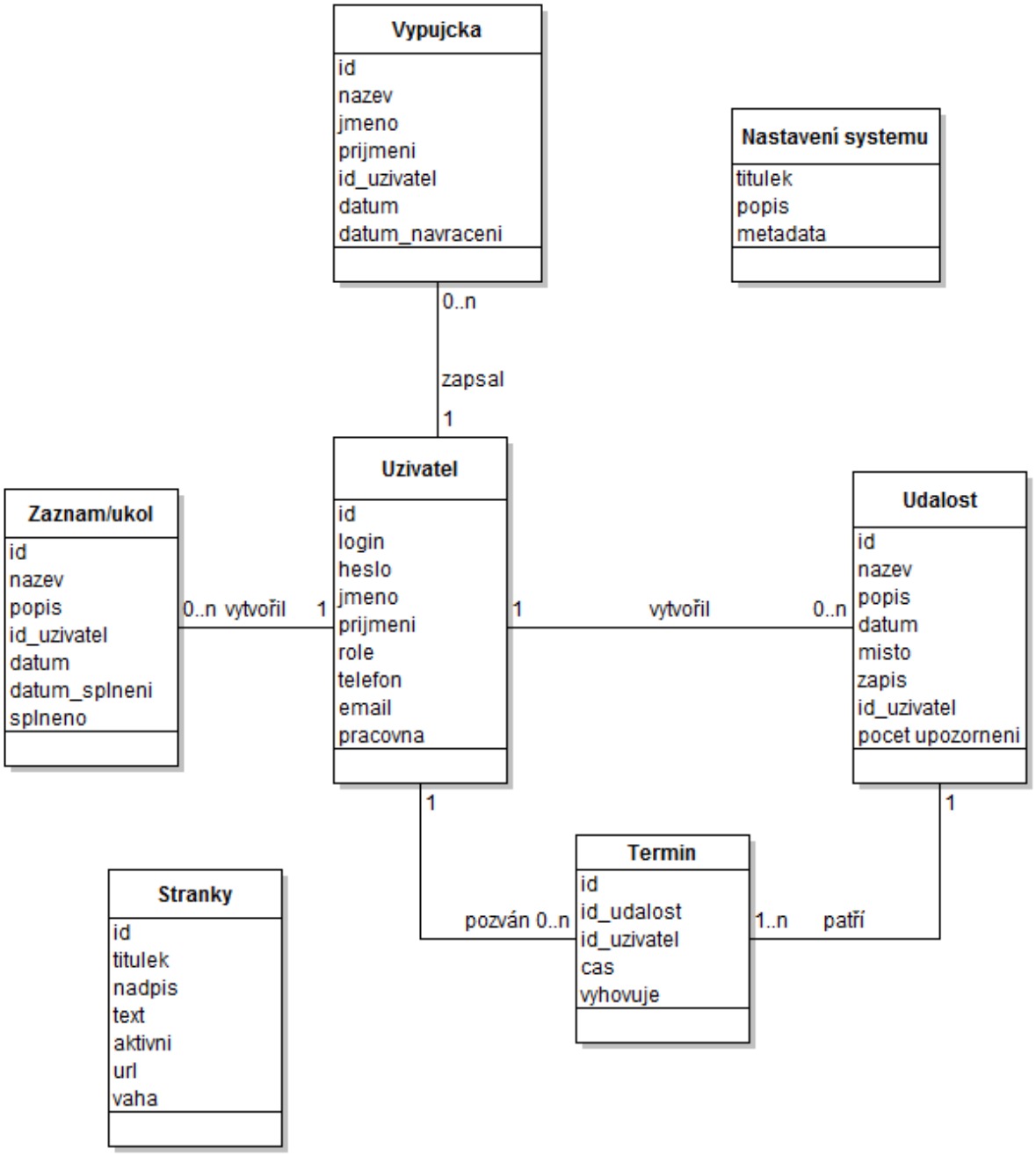
Návrh struktury databáze vychází z analýzy neformální specifikace. Z výsledků analýzy vyplývá nutnost ukládat informace o uživatelích, jejich záznamech (úkolech) a výpůjčkách. Dále bude nutno uchovávat data o událostech a termínech náležejících k daným událostem. Kromě zmíněných dat bude zapotřebí ukládat informace o webových stránkách určených pro veřejnost a informace týkající se základního nastavení parametrů systému.

Z výše uvedených požadavků bude navrženo 7 tříd, které zajišťují uchování všech potřebných informací. Tyto třídy jsou zobrazeny v konceptuálním diagramu tříd na obrázku č. 3.2.

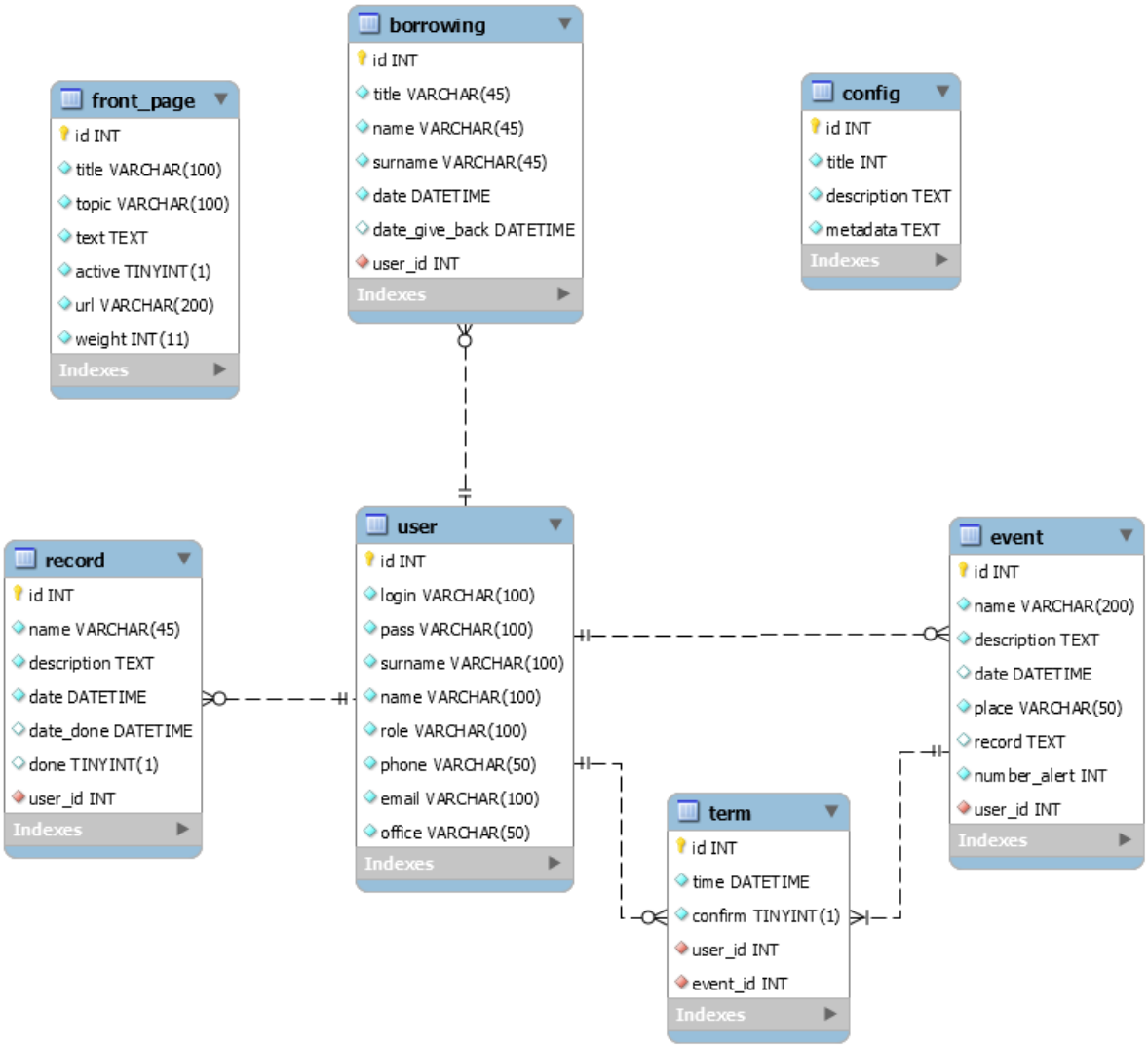
Přehled a význam tříd:

- Uzivatel – uchovává informace o uživatelích
- Zaznam – záznamy/úkoly, které se vztahují k uživateli
- Vypujcka – informace o půjčení položky

- Udalost – základní informace o události bez určení termínu konání
- Termin – termíny všech uživatelů ke všem událostem
- Stranky – HTML stránky, které se zobrazí veřejnosti
- Nastaveni serveru - parametry pro nastavení systému



Obrázek 3.2: Konceptuální diagram tříd



Obrázek 3.3: Návrh schématu databáze

Na obrázku 3.3 je zobrazena struktura databázových tabulek. Dále jsou popsány sloupce několika vybraných tabulek. Cizí klíče odkazující na navazující tabulku jsou tvořeny podle schématu „název navazující tabulky_id“. Například v tabulce `record` má cizí klíč na tabulku user název `user_id`. Tato konvence byla zvolena z důvodu přehlednosti, kdy z názvu sloupce je patrné, k jaké tabulce vedlejší klíč patří. Hlavním důvodem je ovšem získávání dat z databáze. Nette Framework dokáže díky názvu získat informace z vedlejší tabulky, aniž by byl explicitně použit v dotazu do databáze příkaz na spojení tabulek - více v kapitole 4.3.1.

Tabulka `user` obsahuje identifikátor uživatele `id`, přihlašovací jméno `login`, heslo k přihlášení `pass`, jméno a příjmení uživatele `name` a `surname`, název oprávnění `role` a nepovinné sloupce telefonní číslo `phone`, e-mailová adresa `email` a označení pracovny `office`.

Tabulka `borrowing` obsahuje identifikátor `id`, název vypůjčené věci `title`, jméno `name` a příjmení `surname` toho, kdo si danou věc vypůjčil (osoba půjčující si věc nemusí být registrována v systému). Dále obsahuje datum zapůjčení předmětu `date`, datum navrácení `date_give_back` a cizí klíč na tabulku.

Tabulka `event` obsahuje identifikátor `id`, název události `name`, popis události `description`, místo konání události `place`, počet plánovaných e-mailových upozornění k odeslání

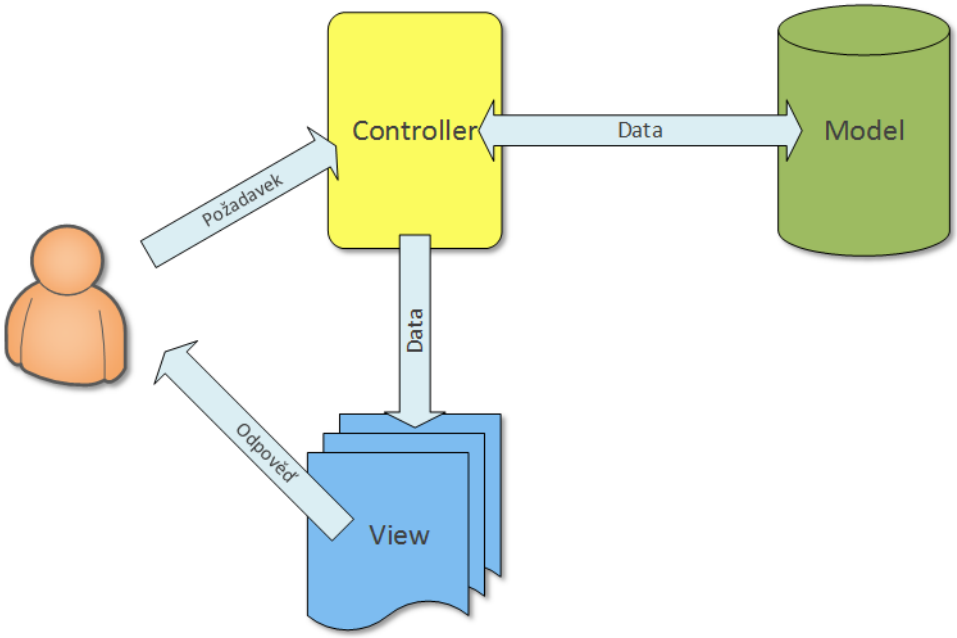
`number_alert`, cizí klíč na tabulku user `user_id` určující, kdo je organizátorem dané schůzky, domluvené datum konání `date` a zápis ze schůzky `record`.

Tabulka `term` obsahuje identifikátor `id`, možný čas konání události `time`, binární položku `confirm` rozhodující, zda daný čas vyhovuje, a cizí klíč na tabulku user `user_id` a na tabulku `event` `event_id`.

Tabulka `front_page` obsahuje identifikátor `id`, titulek stránky `title`, nadpis stránky `topic`, textový obsah stránky `text` a příznak, nastavující, zda stránka bude vykreslena `active`. Dále obsahuje url stránky pro lepší indexaci ve vyhledávačích `url` a váhu stránky `weight`, která určí, jak se stránky mají řadit za sebou.

3.4 Architektura systému

Architektura systému vychází z návrhového vzoru MVC (Model-View-Controller). [13] Jedná se o softwarovou architekturu vzniklou potřebou oddělit u aplikací s grafickým rozhraním kód aplikační logiky od kódu zobrazujícího data a kódu obsluhy. Kódem aplikační logiky je myšlen tzv. datový model (Model), který je datovým a funkčním základem aplikace. Tento model zastupuje databázi na aplikační úrovni, tzn. zajišťuje práci s databází, provádí přihlášení uživatele do systému atd. Model z principu neví o existenci view nebo controller. View, tzv. pohled, je vrstva aplikace zajišťující zobrazení výsledku požadavku. Tato vrstva zpravidla používá šablonovací systém, jehož šablony přesně ví, jak mají konkrétní data z databáze zobrazit. Poslední částí návrhového vzoru je Controller, tzv. řadič, který zpracovává požadavky uživatele. Obsluhou požadavků je myšleno volání patřičné aplikační logiky, která vrací odpovídající data a ta jsou následně předána pohledu k vykreslení, viz obrázek 3.4. Obdobou řadiče v Nette Frameworku jsou tzv. presentery.



Obrázek 3.4: Architektura systému vzoru MVC

4 Implementace

Tato kapitola se zabývá implementací informačního systému. Je zde podrobně popsána tvorba aplikace, práce s databází a její databázová vrstva na straně aplikace, propojení navržených modelů a částí systému. Zvláštní kapitola je věnována problematice vytváření url adres a adresování, kdy framework poskytuje nástroje pro snadnou tvorbu uživatelsky přívětivých adres. Na konci se kapitola zabývá grafickým návrhem systému a jeho responzivitou.

Základním kamenem systému je PHP Framework Nette. Při implementaci byla snaha využít co nejvíce prostředků, které nám nabízí framework nebo rozšíření určená přímo pro Nette. Konkrétně je převzata adresářová struktura, tzv. sandbox. Jedná se o adresářovou strukturu ukázkové aplikace, kde je předpřipraveno doporučené rozdělení adresářů podle datových vrstev a návrhového modelu.

K implementaci databáze je použita databáze MySQL. Důvodem použití MySQL je její rozšíření, viz kapitola 2.3. Spolu s výběrem databáze bylo potřeba určit úložiště, tzv. storage engine. Databáze nabízí k potenciálnímu výběru mnoho úložišť, nicméně prakticky připadají v úvahu pouze dvě, a to MyISAM a InnoDB.

MyISAM

MyISAM je nejpoužívanější úložiště, což dokládá i fakt, že bylo do verze 5.1 (aktuální veze 5.7.5) používáno jako výchozí úložiště systému MySQL. Úložiště je nástupcem formátu ISAM (Indexed Sequential Access Method) a obsahuje mnoho rozšíření. Systém ukládání dat je následující: každá tabulka je uložena pomocí tří souborů pojmenovaných stejně jako daná tabulka. Soubor s příponou `.frm` ukládá formát tabulky, soubor s příponou `.MYD` (MYData) ukládá data dané tabulky a v souboru s příponou `.MYT` (MYIndex) jsou uloženy indexy. [14]

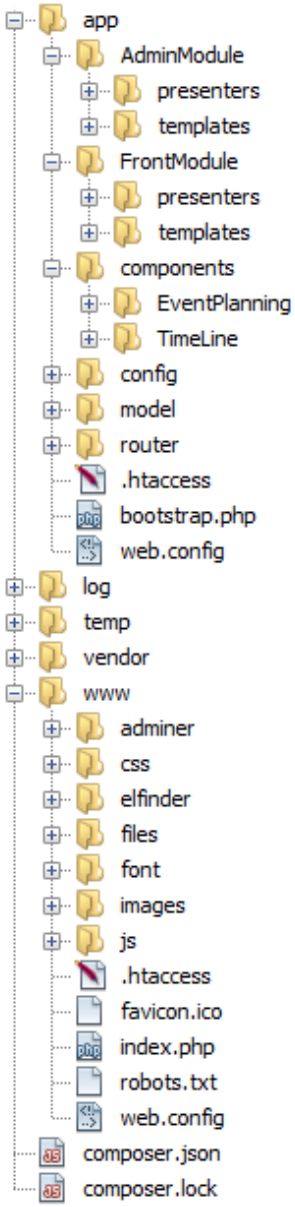
InnoDB

InnoDB byl navržen pro zpracování krátkodobých databázových transakcí, které se málokdy anulují. Od verze MySQL 5.5 se stává výchozím úložištěm databáze. Úložiště poskytuje standartní funkce transakcí kompatibilní s modelem ACID (atomicity, consistency, isolation, and durability). Pro optimalizaci dotazů databázi jsou implementovány primární a cizí klíče. Tím je umožněno vytvářet dotazy, které přistupují k úložišti minimálně, případně lze pomocí cizích klíčů vytvářet dotazy, které získávají data z více tabulek.

K implementaci projektu bylo zvoleno úložiště InnoDB, jelikož se jedná o aktuální výchozí úložiště databáze MySQL. Dalším důvodem pro danou volbu byla podpora v Nette frameworku (více rozepsáno v kapitole 4.3.1). [15]

4.1 Adresářová struktura

Jak již bylo zmíněno, adresářová struktura je postavena na struktuře tzv. sandboxu, viz obrázek 4.1. Použití doporučené adresářové struktury intuitivně vede ke vhodnému použití Nette frameworku a dodržování zásad objektového programování. Zároveň se snižuje prostor pro vytváření chyb.



Obrázek 4.1: Adresářová struktura aplikace

V kořenovém adresáři se nachází pět adresářů aplikace a dva soubory `composer.json` a `composer.lock`, ve kterých je zapsána konfigurace Composeru. Konkrétně jsou zde vypsány všechny použité závislosti včetně jejich verzí.

V adresáři `app` jsou uloženy zdrojové kódy aplikace. Projekt je rozdělený do dvou modulů adresáře `AdminModule` a `FrondModule`. Moduly obsahují šablony pro zobrazení a kódy prezenční vrstvy, které obsluhují dané šablony. Do zvláštního adresáře `components` jsou umístěny

komponenty `EventPlannig` a `TimeLine`. Tyto komponenty nejsou závislé na konkrétním modulu, respektive na presenteru, a dají se použít na libovolném místě. Adresář `config` obsahuje nastavení aplikace - nalezneme zde přihlašovací údaje do databáze, jak se má chovat `cache` a session aplikace, jaké služby (service) jsou v aplikaci dostupné. Dále jsou zde údaje, jak se má aplikace zachovat při vyskytnutí chyby, jakým způsobem má pracovat s adresami, atd. Společný adresář `model` obsahuje zdrojové kódy umožňující komunikaci s databází - jedná se o databázovou vrstvu, která je volána z presenterů. Posledním adresářem je `router`, jenž obsahuje třídu zajišťující překlad url adres na adresy určující použitý presenter a pohled. Ve složce `app` je uložen důležitý soubor `bootstrap.php`, který má na starosti načtení frameworku Nette společně s potřebnou konfigurací aplikace. Do adresáře `log` se zapisují veškeré události včetně chybových stavů. Dočasné soubory cache slouží ke zrychlení načtení stránek aplikace tím, že výsledný vygenerovaný zdrojový kód odeslaný do prohlížeče uloží a při stejném požadavku systém odpoví obsahem uloženého souboru cache. Tím odpadá režie běhu PHP skriptu a zároveň i režie dotazování dat na databázi. Do adresáře se ukládají nejen výstupy posílané klientům, ale také výsledky dotazů databáze a vygenerované nastavení aplikace. Správu obsahu dané složky plně zajišťuje framework, který určí, jaký obsah s jakou dobou expirace bude uchovávat.

Knihovny Nette frameworku jsou uloženy v adresáři `vendor`. Tento adresář obsahuje jak nativní knihovny frameworku, tak i rozšíření frameworku, které sem umístil již zmiňovaný správce závislostí Composer.

Posledním adresářem v kořenové složce je adresář `www`. Do tohoto adresáře mají přístup klienti, kteří se připojují na server a využívají aplikaci. Důležitým souborem je `index.php`, který je spouštěn při každém požadavku. Soubor obsahuje kód pro vyvolání souboru `app/bootstrap.php` a následné spuštění aplikace. Dalšími veřejně přístupnými kódy jsou kódy zajišťující vzhled aplikace uložené v adresářích `css`. Tyto kódy obsahují kaskádové styly a `font`, kde je uložený použitý font písma v projektu. Následující adresář `images` shromažďuje obrázky použité pro grafický návrh aplikace. Adresář `js` spolu s `elfinder` obsahuje JavaSriptové kódy a knihovny, které jsou interpretovány na straně uživatele. V projektu jsou použity knihovny třetích stran jQuery, textový editor CKEditor [16], správce souborů elFinder [17] a výběr času time picker [18]. Dále je využíván skript `nette.ajax.js` napsaný přímo pro Nette Framework, čímž lze snadno dosáhnout interaktivního chování aplikace pomocí ajaxových volání.

4.2 Veřejná část

Veřejnou částí projektu je myšlen prostor, kde se můžou volně pohybovat návštěvníci bez nutnosti přihlášení do systému – tj. prohlížení stránek, které jsou aktuálně zveřejněny, a zobrazení přihlašovacího formuláře. Pro indexovací roboty je připravena možnost načíst soubor sitemap.xml, který se vygeneruje při dotazu. Kódy pro veřejnou část jsou uloženy ve složce `app/FrontModule`. Dále budou popsány nejdůležitější konstrukce veřejné části, a to navigační menu projektu (stejný princip je využit i v administraci) a vykreslení obsahu stránky.

4.2.1 Navigační menu

Pro navigační menu aplikace byl použit plug-in Nette frameworku SmfMenu, který integruje knihovnu KnpMenu do frameworku. Definice menu se provádí v souboru `/app/FrontModule/presentres/BasePresenter.php`. Tento základní presenter rozšiřuje

všechny ostatní presentery ve veřejné části. Tím je zajištěno, že zde definované menu bude v celé veřejné části stejné.

Menu se vytvoří zavoláním funkce `createComponentMenu()`, v níž se nejdříve získá instance Smfmenu, která umožní práci s kořenovým prvkem menu. Následně je požádán model `\App\Model\FrontpageManager` o získání stránky určené k zobrazení ve veřejné části. Frontpage Manager při zavolání funkce `getActiveFrontPage()` provede dotaz do databáze a získá všechny stránky, které mají nastavenu hodnotu ve sloupci `active` na 1 a dané výsledy seřadí sestupně podle váhy určené ve sloupci `weight`.

Tvorbou obsahu navigačního menu rozumíme plnění datové struktury tree (strom). Ve veřejné části nebude nutné tvořit víceúrovňový strom, postačí strom jednoúrovňový vytvořený pomocí cyklu `foreach`. Přidání uzlu (položky v menu) znamená, že k hlavnímu uzlu je připojen potomek, který obsahuje název určený `$page->title` a atributy. Atributy jsou label, jehož obsah je stejný jako název uzlu, a link obsahující adresu. Adresa stránky se skládá z názvu presenteru Homepage, názvu pohledu default a interního čísla id. Výslednou adresu Framework interpretuje jako `/Homepage/default/id`. Naplnění instance komponenty smfMenu (viz ukázka) a následné vypsání v šabloně vytvoří odrážkový seznam.

```
foreach ($frontPage as $page)
{
      $root->addChild($page->title, array(
            'label' => $page->title,
            'link'  => array('Homepage:default', array('id' => $page->id))
      ));
}
```

4.2.2 Zobrazení stránky

Zobrazení stránky ve veřejné části zajišťuje presenter `/app/FromModul/presenter/Homepage` a pohled `default` volaný funkcí `actionDefault()`. Funkci lze předat nepovinný parametr `id` stránky. V případě, kdy je zadáno id stránky, požádá presenter model o data `frontpageManager->get($id)`. Po získání patřičných dat se zkontroluje, zda se daná stránka může zobrazit běžnému uživateli. Administrátor může přistupovat ke všem skrytým stránkám. Touto kontrolou se zamezí zobrazení stránky pomocí náhodného zadávání adres. Pokud není zadán nepovinný parametr, načítá se výchozí stránka, tj. stránka s nejvyšší váhou a povolením k zobrazení ve veřejné části.

4.2.3 Sitemap

Sitemap je zpravidla xml soubor uložený v kořenovém adresáři projektu, který slouží ke snazší indexaci stránek pro příchozí roboty internetových vyhledávačů. Sitemap obsahuje seznam stránek přístupných pro příchozí návštěvníky. Využívání sitemap souboru je jedním z nástrojů seo optimalizace, která vede k upřednostňování odkazů na web ve výsledcích hledání ve vyhledávačích.

V projektu se soubor generuje na vyžádání, není vygenerován předem, ale při požadavku na sitemap se vrátí aktuální data. O generování se stará soubor `/app/FromModul/presenter/Homepage` a volaný pohled `sitemap`. Do šablony jsou předány aktivní stránky (stejný seznam je použit i pro generování menu ve veřejné části) a šablona následně vygeneruje validní xml soubor s odkazy na dané stránky. Robot přicházející na stránky poté požádá

o soubor `sitemap.xml` a systém sám pomocí adresovacích pravidel zavolá obslužnou akci, jež vygeneruje soubor. [19]

4.2.4 Adresování

Adresování a tvorbu validních url adres včetně předávaných parametrů v projektu zajišťuje třída `/app/router/RouterFactory`. Zároveň dokáže podle zadaných pravidel url adresu přetransformovat do požadovaného tvaru. Základní adresování je rozděleno na dvě části podle navržených modulů administrace a veřejné části; tzn. adresy začínající `/admin/` (název domény se zde neuvádí, jelikož je irelevantní a pro zjednodušení adresy je / ekvivalentem http://www.nazev-domeny.cz/) směřují do administrační části aplikace. V administraci je použito základní adresovací pravidlo ve tvaru `/admin/<presenter>/<action>/<id>`, kde jednotlivé části znamenají:

- `<presenter>` - název presenteru, který se má zavolat při požadavku
- `<action>` - akce neboli pohled, který se má u daného presenteru použít
- `<id>` - vnitřní identifikátor, který odkazuje na konkrétní data

Příkladem takové adresy může být adresa úpravy vybraného záznamu `/admin/record/edit/?recordId=3`.

Základní pravidlo pro veřejnou část je stejné jako pro část administrační jen s tím rozdílem, že v adrese odpadá část `/admin/`. Výsledné adresy jsou generovány ve tvaru `/homepage/default/?id=X`. I když popsané řešení adresování veřejné části je po technické stránce korektní, aktuálním standardům nevyhovuje, jak z pohledu seo optimalizace, tak z pohledu uživatelské přívětivosti. Z tohoto důvodu je v databázové tabulce `front_page` vytvořen sloupec url, který obsahuje nadpis stránky ve formátu vyhovujícím url adrese. Při vytváření nebo úpravě stránky se při zápisu do databáze pomocí funkce `\Nette\Utils\Strings::webalize()` získá url dané stránky, které se uloží do databáze. Zbytek režie obstará třída `RouterFactory`, která pomocí pravidla `/<id>` generuje potřebné adresy.

V případě vytváření url adresy třída načte z databáze hodnotu url sloupce pomocí zadaného identifikátoru a vygeneruje namísto adresy `/homepage/default/?id=2` adresu `/kontakt`.

V opačném případě, kdy uživatel přejde na odkaz `/kontakt`, dojde k prohledání databáze, zda se ve sloupci url nenachází hodnota `kontakt`. V případě kladného hledání se z nalezeného záznamu vezme informace o id stránky a dále se v aplikaci pracuje s adresou `/homepage/default/?id=2`. V případě nenalezení url se třída pokusí na danou adresu aplikovat nejbližší následující adresovací pravidlo, které by obvykle mělo mít obecnější charakter.

V případě, že na zadanou url adresu není možné aplikovat žádné adresovací pravidlo, dochází k vytvoření výjimky, kterou zpracuje třída `/app/FrontModule/ErrorPresenter`, jež vrátí chybovou stránku 404.

4.3 Administrační část

Administrační částí projektu je myšlen prostor, kde se můžou pohybovat pouze návštěvníci přihlášení do systému. V administraci se uživatelé dělí do dvou základních rolí, a to na běžného uživatele a administrátora. Uživateli s administrátorskými právy se zobrazí rozšířená administrátorská část. V této kapitole budou popsány nejdůležitější části administrace, tedy práce s událostmi a vytvořené komponenty pro výběr vyhovujících termínů a pro zobrazení chronologicky seřazených zápisů. Navigační menu a podobné výše popsané části systému budou vynechány z důvodu použití stejných postupů pro implementaci.

4.3.1 Databázová vrstva

Celá aplikace nepřistupuje k databázi přímo, tzn. v žádném presenteru se neprovádějí úkony, které by se připojovaly do databáze, případně posílaly dotazy pro získání dat z databáze. Tuto činnost zajišťuje databázová vrstva a její modely dostupné jako služby aplikace. Všechny modely rozšiřují základní model `BaseManager`, který se stará o připojení do databáze pomocí třídy `Nette\Database\Connection`, jenž tvoří obálku na PDO (PHP Data Objects).

PDO

PDO je objektová třída, která má na starosti práci s databází. Před vytvořením PDO byly snahy vytvořit nástroj, jenž by dokázal pracovat se všemi databázemi a využít jejich celý potenciál. Z důvodu velké složitosti bylo od této myšlenky upuštěno a vznikl kompromis PDO. Ovladač je charakterizován jako jednoduché objektové rozhraní, které pro společné vlastnosti všech databází nabízí jednotné ovládání. Kompromisem se zde stávají metody použitelné jen pro některé typy databází. [20]

Pro dotazování do databáze je použito třídy `Nette\Database\Context` umožňující pokládání databázových dotazů. Druhou použitou třídou je `Nette\Database\Table`, která zjednodušuje a optimalizuje výběr dat z databáze. Tato třída se snaží načítat data pouze z jedné tabulky do instance ActiveRow a případné vazby na další tabulky načíst druhým dotazem tak, aby pro některá data nebyly databázi položeny dva dotazy. Následující kód je příkladem práce s třídou `Nette\Database\Table`:

```
$books = $context->table('book');
foreach ($books as $book) {
        echo 'title:      ' . $book->title;
        echo 'written by: ' . $book->author->name;
}
```

V PHP kódu je možné získat obsah tabulky příkazem `$context->table('book')`, následně příkaz `foreach` iteruje získanými daty a vypisuje se název knihy a autorovo jméno, které je ale uložené v tabulce autor. Díky instanci `ActiveRow` není potřeba vytvářet další dotaz do databáze, instance se o něj postará sama. Výsledné dotazy položené databázi budou následující:

```
SELECT * FROM `book`
SELECT * FROM `author` WHERE (`author`.`id` IN (11, 12))
```

Díky automatickému použití cache navíc instance upraví SQL dotazy tak, aby se získaly pouze sloupce, které se používají (viz finální ukázka SQL dotazů). [21]

```
SELECT `id`, `title`, `author_id` FROM `book`
SELECT `id`, `name` FROM `author` WHERE (`author`.`id` IN (11, 12))
```

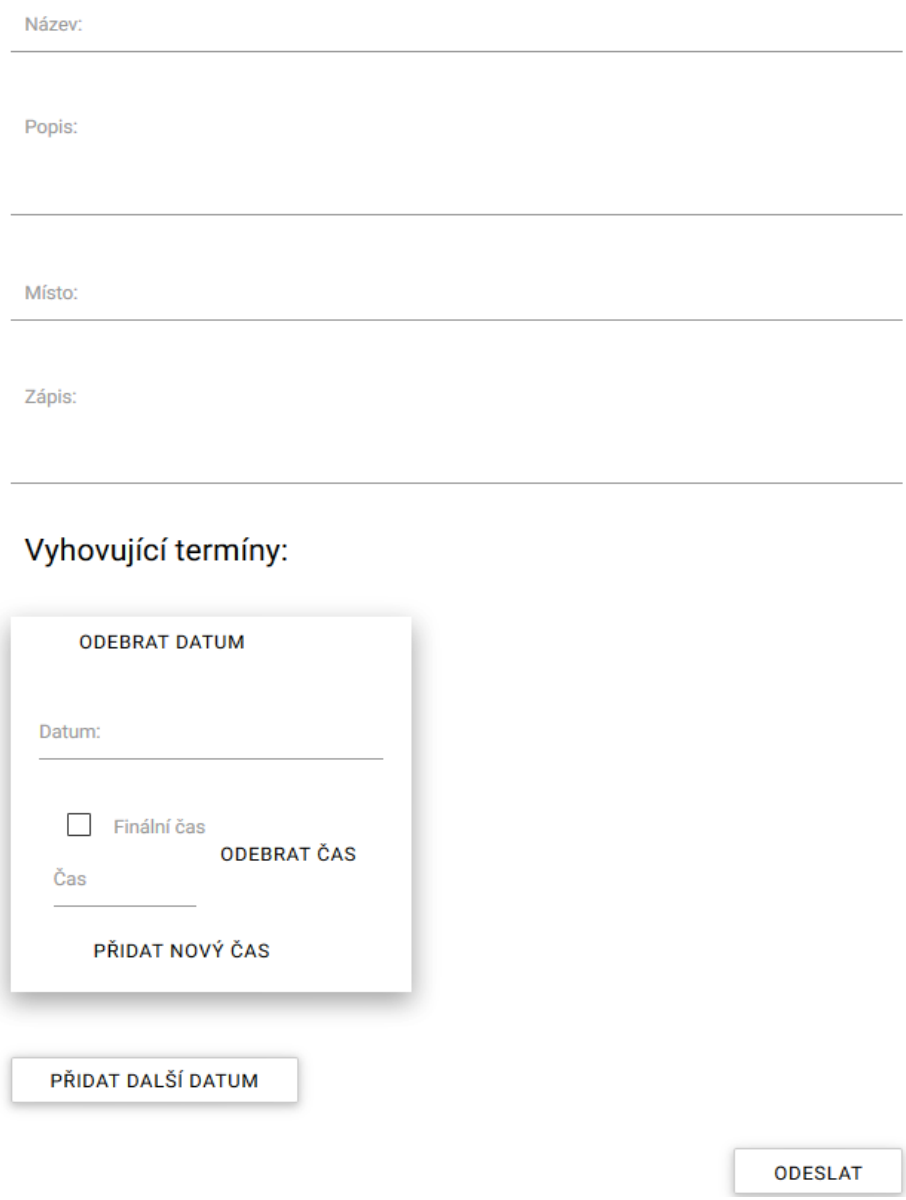
Všechny vytvořené modely tříd reprezentují konkrétní tabulku v databázi a mají sjednocené názvy společných metod. Společnými metodami jsou `getTable()` pro získání celé tabulky, `get($id)` pro získání konkrétního záznamu v databázi, `add($item)`, `edit($item)` a `delete($id)` pro vytváření, úpravu a odstranění dat z dané tabulky. Další metody se vážou na konkrétní tabulku a obsahují omezující podmínky pro práci s daty.

4.3.2 Přihlašování

Každý uživatel systému je reprezentován službou user, což je objekt třídy `Nette\Security\User`. Pro ověření postačí zavolat funkci objektu `isLoggedIn()`, podle které poznáme, zda je daný uživatel řádně přihlášen do systému. Všechny presentery v administrační části rozšiřují základní presenter `BasePresenter`, který při vytvoření objektu v metodě `startup()` zkontroluje, zda je uživatel přihlášen. U nepřihlášeného uživatele dojde k přesměrování na přihlašovací formulář ve veřejné části. Zároveň s přesměrováním se předává tzv. backlink, což je adresa, na kterou je po přihlášení uživatel přesměrován zpět. Po odeslání přihlašovacího formuláře implementovaného v presenteru SignPresenter dojde k předání vyplněných údajů modelu UserModel. Konkrétně se jedná o funkci `authenticate()`, která se dotazem do databáze pokusí najít uživatele na základě přihlašovacího jména. V případě nalezení záznamu provede verifikaci zadaného hesla pomocí `Passwords::verify()` a následně naplní službu user potřebnými daty. V případě negativní verifikace vygeneruje nový kontrolní součet hesla `Passwords::needsRehash`.

4.3.3 Formuláře

Nejdůležitější částí implementace systému je práce s formuláři, jež zajišťují plnění informací do systému. Nette Framework poskytuje třídu pro tvorbu formulářů `Nette\Forms`. Tato třída je zodpovědná za vykreslení formuláře v šabloně včetně kontroly na straně klienta. Samozřejmostí je kontrola i na straně serveru včetně ošetření zadaných dat před útoky. Dále bude popsán formulář pro vytvoření události - jedná se o nejsložitější formulář celé aplikace, který využívá rozšíření `Kdyby/Replikator` umožňující tvorbu dynamických formulářů.



Obrázek 4.2: Formulář pro vytvoření nové události

4.3.3.1 Vytvoření formuláře

Na obrázku Obrázek 4.3 je znázorněn zdrojový kód formuláře pro vytvoření nové události. Funkcí `$this->form()` získáme instanci třídy `\Nette\Application\UI\Form`. Stejný formulář se používá i při úpravě událostí, proto jsou zde uvedeny i hodnoty, které nejsou potřebné pro vytváření událostí. Tyto hodnoty jsou pro uživatele skryté a jsou definovány pomocí funkce `addHidden()` – jedná se o interní identifikátor události `id`, identifikátor uživatele vytvářejícího událost `user_id` a počet upozornění `number_alert`, jež má systém provést. Dalším prvkem formuláře je textový vstup pro Název a Místo konání události vytvořený funkcí `addText()` a vyplnění jejich hodnot je vyžadováno funkcí `setRequired()` - bez vyplněných hodnot nedojde k odeslání formuláře. Textová pole pro zadání popisu události a případného zápisu z jednání jsou určena funkcí `addTextArea()`. Následuje použití rozšíření `Kdyby/Replikator`, kdy je vytvořen dynamický prvek funkcí `addDynamic()`. Této funkci se předává jako parametr název vytvářeného

prvku, funkce obsahující prvky formuláře, které se mají dynamicky přidávat nebo odebírat, a posledním parametrem je počet vytvořených dynamických prvků při inicializaci formuláře.

Předávaná funkce jako parametr obsahuje formulářový prvek Datum a další dynamický prvek, který obsahuje vstupní pole pro zadání času a potvrzení finálního času `addCheckbox()`. To znamená, že formulář přijímá více možných dat konání akce a každé datum může obsahovat více časů konání. Zároveň dynamické prvky obsahují tlačítka pro přidání a odebrání dalšího dynamického prvku `addSubmit()`. Dané tlačítko způsobí odeslání formuláře, který je v presenteru odchycen a s patřičně upravenými daty vrácen zpět k vykreslení. Tlačítko pro odeslání formuláře k uložení vytváří funkce `addSubmit()`, které je předána adresa `onClick[] = callback($this, 'eventSucceeded')`.

```
protected function createComponentEventForm()
{
    $form = $this->form();
    $removeEvent = callback($this, 'EventFormRemoveElementClicked');
    $form->addHidden('id', null);
    $form->addHidden('user_id', null);
    $form->addHidden('number_alert', null);
    $form->addText('name', 'Název:')
        ->setRequired('Zadejte název.');
    $form->addTextArea('description', 'Popis:')->getControlPrototype()->setClass('materialize-textarea');
    $form->addText('place', 'Místo:')
        ->setRequired('Zadejte místo konání.');
    $form->addTextArea('record', 'Zápis:')->getControlPrototype()->setClass('materialize-textarea');
    $dates = $form->addDynamic('dates', function (Container $container) use ($removeEvent) {
        $container->addText('date', 'Datum:', 20, 20)->getControlPrototype()->setClass('datepicker');
         $times = $container->addDynamic(
            'times',
            function (Container $time) use ($removeEvent) {
                $time->addHidden('id', null);
                $time->addText('time', 'Čas')->getControlPrototype()->setClass('clockpicker');
                $time->addCheckbox('pick',"Finální čas");
                $removeBtn = $time->addSubmit('remove', 'Odebrat čas')
                    ->setValidationScope(false)
                    ->onClick[] = $removeEvent;
            },0);

        $times->addSubmit('add', 'Přidat nový čas')
            ->setValidationScope(FALSE)
            ->onClick[] = callback($this, 'EventFormAddElementClicked');

        $container->addSubmit('remove', 'Odebrat datum')
            ->setValidationScope(FALSE) # disables validation
            ->onClick[] = $removeEvent;
    },0);
    $form->addSubmit('send', 'Odeslat')
        ->onClick[] = callback($this, 'eventSucceeded');
    $dates->addSubmit('add', 'Přidat další datum')
        ->setValidationScope(FALSE)
        ->onClick[] = callback($this, 'EventFormAddElementClicked');
    $form->setDefaults(array("user_id" => $this->user->getId()));
    return $form;
}
```

Obrázek 4.3: Zdrojový kód formuláře pro vytvoření události

4.3.3.2 Zpracování formuláře

Po odeslání formuláře dochází k jeho kontrole na straně uživatele pomocí JavaScriptu a následně na straně serveru. Nette Framework umožňuje kromě použití předdefinovaných validačních pravidel

vytvořit validaci s vlastními pravidly. Dokud formulář neprojde v pořádku všemi kontrolami, není zavolána funkce na zpracování formuláře.

V případě, kdy se ve formuláři vyplňují data z jedné tabulky, postačí přijatá data předat modelu patřičné tabulky a dojde k zapsání informací do databáze. U formuláře s událostí se zadávají informace, které se ukládají do dvou tabulek, a to do tabulky `event` a `term`. Z tohoto důvodu musí funkce `eventSucceeded()` předávat modelu `eventManager` jen část získaných dat týkajících se události. U zadaných časů konání akce je potřeba pomocí dvou cyklů projít zadaná data a časy, a ty následně předat modelu `termManager`. Jak už bylo zmíněno výše, formulář pro událost se používá také pro úpravu události, a proto po uložení všech termínů konání akce dochází ke smazání termínů, které byly odstraněny ve formuláři, ale ne v databázi. Smazání termínů se provádí získáním všech termínů dané akce, které má přiřazené organizátor. Cyklus zjišťuje, zda byl daný termín odeslán formulářem, a pokud ne, dojde k jeho odstranění z databáze.

4.3.4 Šablony

Šablony se starají o vykreslení informací uživateli do uživatelsky přívětivé podoby. Vykreslením je zde myšleno vygenerování HTML kódu. Šablony jsou tvořeny šablonovacím jazykem, který umožňuje používání základních programovacích operací, jako je průchod cyklem, rozhodovací bloky a použití speciálních značek a funkcí, které můžou nahrazovat nebo měnit výsledný obsah. Hlavní výhodou použití šablon je jejich modulárnost, univerzálnost a opětovné použití. Díky zmíněným vlastnostem se výsledná HTML stránka zpravidla skládá z více šablon, to znamená, že se do základní šablony na závislosti zobrazovaných dat vkládají segmenty kódu, jejichž vygenerování zajišťují jiné šablony. Šablony by se v žádném případě neměly starat o získávání informací z databáze, ale měly by mít v připravených proměnných nachystána data v požadovaném formátu.

Nette Framework používá vlastní šablonovací systém Latte [22]. Tento systém je postaven na tzv. makrech sloužících k zápisu řídících příkazů pro práci s daty, obdobně jako zápis příkazů v PHP kódu. Makra jsou při provádění scriptu nahrazena ekvivalentními kusy PHP kódu, které provedou požadovanou činnost. Zápis maker se provádí do složených závorek `{}`. Dalším nástrojem, který systém používá, jsou tzv. Filtry (dříve helper). Filtry [23] jsou funkce, které pomáhají upravit nebo přeformátovat data do výsledné podoby. Zapisují se do makra za uvedenou proměnnou pomocí svislítka |, např. vypsání určitého počtu znaků z řetězce se zapíše jako {$title|truncate:5}.

Systém používá dvě základní šablony `@layout.latte` a `@layoutAdmin.latte`, které slouží pro dané části systému. Tyto šablony tvoří základní HTML kostru layuotu, jež obsahuje odkaz na soubory s kaskádovými styly CSS a kód pro vykreslení navigačního menu. Toto menu je součástí implementovaného layotu, který bude popsán níže. Nakonec jsou uvedeny odkazy na JavaScriptové soubory a kódy. Důležitou součástí šablon jsou tzv. bloky - makro, které je nahrazováno obsahem konkrétní stránky `{include #content}`.

Bloky slouží k nahrazení, případně rozšíření obsahu. Konktrétně blok `{block scripts}` obsahuje všechny vazby na JavaScriptové soubory a JavaScriptový kód. Například ve volané šabloně `/app/AdminModule/templates/Record/add.latte`, jež vykresluje formulář pro přidání záznamu do systému, je opět definován blok scripts. Makro `{include #parent}` určuje, že se má obsah bloku připojit k obsahu bloku definované v základní šabloně. Tato konstrukce zařídí, aby JavaScriptový kód pro textový editor a správce souborů používaný v případě plnění daného formuláře nezpomaloval načítání ostatních částí systému tam, kde není potřebný.

Nette Framework zvládá velkou část vykreslování automaticky sám, což dokládá způsob zápisu vykreslení formuláře. Například šablona pro přidání nové výpůjčky do systému vypadá následovně:

```
{block headerH1}Nová výpůjčka{/block}
{block content}
{control borrowingForm}
```

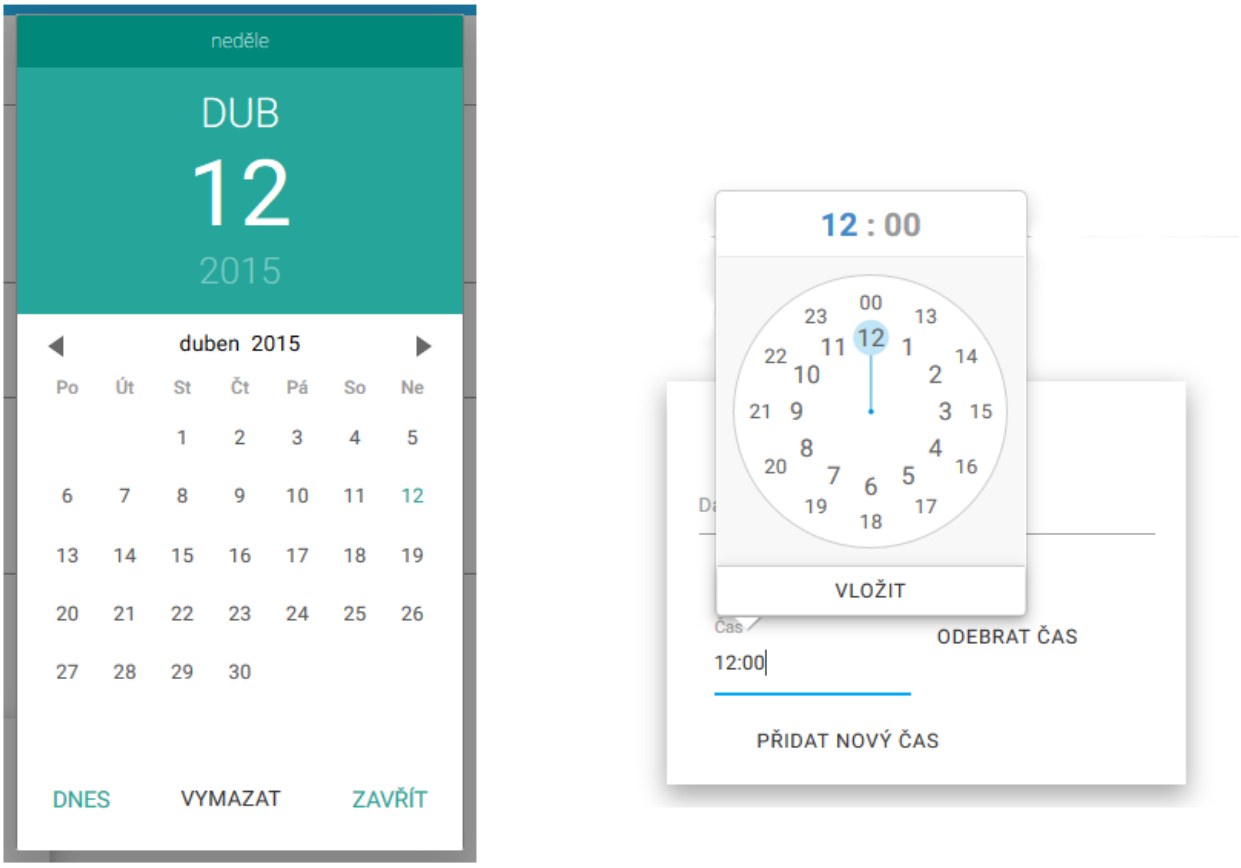
Blok headerH1 nastaví nadpis, jenž je zakomponován v layotu základní šablony. Blokem content je definován obsah stránky v základní šabloně. V konkrétním příkladě je zde pouze makro {control borrowingForm}, které je nahrazováno vygenerovaným formulářem definovaným v presenteru BorrowingPresenter funkcí createComponentBorrowingForm(). Framework sám zajistí korektní vygenerování HTML kódu a s ním spojený Javascriptový kód zajišťující kontrolu na straně uživatele.

```
{form eventForm}
<div class="row">
    <div class="input-field"> {label name}{/label}{input name} </div>
</div>
<div class="row">
    <div class="input-field"> {label description}{/label} {input description} </div>
</div>
<div class="row">
    <div class="s12 input-field"> {input place}{label place/} </div>
</div>
<div class="row">
    <div class="input-field">{label record}{/label}{input record}</div>
</div>
{snippet itemsList}
    <h5>Vyhovující termíny:</h5>
    <div class="row">
    {foreach $form['dates']->containers as $dateid => $user}
        <div class="col s12 m3 l3 z-depth-2 eventcard" >
            <div class="col s12">
                {input dates-$dateid-remove class=>'ajax btn btn-flat eventbutton'}
            </div>
            <div class="input-field col s12">
                {$user['date']->label}{$user['date']->control}
            </div>
            <div class="col s12">
                {foreach $user['times']->containers as $timeid => $time}
                    <div class="col input-field s12 checkBox">
                        {input $time['pick']: }{label $time['pick']: /}
                    </div>
                    <div class="col input-field s6 time-input">
                        {$time['time']->label}{$time['time']->control}{$time['id']->label}{$time['id']->control}
                    </div>
                    <div class="col s6">
                        {input dates-$dateid-times-$timeid-remove class=>'ajax btn btn-flat eventRemoveTime'}
                    </div>
                {/foreach}
                <div class="col s12">
                    {input dates-$dateid-times-add class=>'ajax btn btn-flat eventbutton'}
                </div>
            </div>
        </div>
    {/foreach}
    </div>
{/snippet}
<div class='row'>
    {input dates-add class=>'ajax btn btn-default'}
</div>
<div class='row right-align'>
    {label send /} {input send class=>'btn btn-default'}
</div>
{/form}
```

Obrázek 4.4: Zdrojový kód šablony formuláře pro vytvoření události

Samozřejmě ne vždy musí automatické generování formuláře znamenat získání požadovaného výsledku. Z tohoto důvodu umožňuje Framework provést ruční generování formuláře tak, aby bylo dosaženo požadovaného výsledku; viz Obrázek 4.2.

Na obrázku 4.4 se nachází zdrojový kód ručně vykresleného formuláře pro vytvoření nové události v systému. Celý formulář je zaobalen makrem `{form eventForm}` určujícím, o jaký formulář se jedná. Dále následuje vypsání tří řádků, na kterých se nacházejí vstupy pro název události, popis a zápis z události. Vstupy jsou definovány makry label a input, kdy label je nahrazen popisem daného prvku a makro input je nahrazeno samotným prvkem, do kterého je možné vepsat konkrétní hodnotu. Další část formuláře zajišťuje vypsání možnosti zadat termíny konání, které lze dynamicky přidávat a odebírat. Makro `{snippet itemsList}` určuje, jaká část formuláře se má překreslit při ajaxovém požadavku na přidání nového termínu. Termíny formuláře se zobrazují pomocí dvou vnořených cyklů foreach, kdy první cyklus iteruje nad možnými dny a druhý nad možnými časy daného dne. Průchod prvního cyklu nejdříve vykreslí tlačítko pro odstranění daného data a následně vstup pro zadání data. Datum není potřeba vepsat do prvku, nýbrž postačí na příslušný vstup kliknout a zobrazí se kalendář pro výběr požadovaného data; viz Obrázek 4.5. Po vypsání data následuje druhý cyklus foreach, který prochází zadané časy ke konkrétnímu datu. Podobně jako u data se zde vykreslí tlačítko pro smazání času a vstupní pole pro zadání času, které se opět zadává pomocí JavaScriptové komponenty, jež je součástí front-end responzivního frameworku Materialize uvedeného v kapitole 4.4. Navíc se zde zobrazuje vstup pro určení termínu konání akce. Posledním prvkem v cyklu dat je tlačítko pro přidání dalšího vstupu pro zadání času. Čas se zadává pomocí komponenty pickadate.js [18] znázorněné na obrázkuObrázek 4.5. Celý formulář ukončují tlačítka pro přidání vstupu pro datum a pro odeslání formuláře.



Obrázek 4.5: Javascriptové komponenty pro volbu data (vlevo) a času (vpravo)

4.3.5 Komponenta EventPlanning

Navržená komponenta EventPlanning zajišťuje vykreslení tabulky možných termínů akce. Konkrétní řádky zastupují pozvané uživatele na danou akci, sloupce reprezentují možné termíny konání události. Na průniku řádků a sloupců se nachází informace, zda danému uživateli vyhovuje konkrétní termín. Tabulka je navržena tak, aby bylo ihned zřejmé, který termín vyhovuje nejvíce uživatelům. Zároveň je součástí tabulky i formulář, který vypadá jako další řádek tabulky. Na průniku sloupců s řádkem formuláře se zobrazí formulářové prvky checkbox button. Formulář může nabývat dvou podob. První podoba je pro uživatele, který je na událost pozván. V takovém případě uživatel může zvolit nebo změnit vyhovující termíny. V druhé podobě se formulář vykresluje pro uživatele, který je zároveň organizátorem konkrétní události. Jelikož tento uživatel je organizátorem, vychází se z předpokladu, že se může dostavit na všechny jím navržené termíny, a proto nemá smysl určovat, zda se na nějaký termín může či nemůže dostavit. Z toho důvodu takový uživatel pomocí formuláře určuje závazný termín konání akce.

Komponenta se skládá ze tří souborů, a to šablony, třídy a rozhraní. Použití komponenty je následující: v presenteru se přichystá funkce `createComponentEventPlaning()`, jíž úkolem je vytvořit instanci komponenty, která předá potřebné závislosti na třídy pracující s databází. Dále komponentě určí, o jakou událost se jedná a jaký uživatel daný přehled zobrazuje. Nakonec vrací zdrojový kód formuláře určený k vykreslení a poskytovaný komponentou.

Vygenerování patřičné tabulky a formuláře zajišťuje funkce `render()` třídy komponenty `/app/components/EventPlanning/EventPlanningControl`. Z databáze se získají všechny dostupné termíny k vybrané akci, které se následně podle data setřídí do pomocné struktury, jež je následně předána do šablony, kde slouží k vygenerování tabulky a formuláře.

Zpracování formuláře po odeslání obstarává funkce třídy komponenty `eventPlanningSucceeded($form)`, kdy cyklus projde všechny termíny formuláře, které jsou následně upraveny v databázi. Pokud formulář odešle již zmíněný organizátor události, dojde k vyhledání potvrzeného termínu, který je následně zapsán do databáze k patřičné události.

4.3.6 Komponenta TimeLine

Vytvořená komponenta TimeLine má na starosti vykreslení informací do takové struktury, aby mohl prohlížeč po aplikaci kaskádových stylů zobrazit informace chronologicky seřazené a provázané s časovou osou. Záměrně se informace blíže nespecifikují, protože tato komponenta je schopna vykreslit chronologicky jakékoliv události nebo záznamy. Použití i struktura komponenty je obdobná jako u komponenty EventPlanning, tj. skládá se ze čtyř souborů, dvou šablon pro vykreslení událostí a záznamů obsluhujících třídy a rozhraní. V presenteru je umístěna funkce `createComponentTimeLine()` volaná ze šablony. Tato funkce vytvoří instanci, které předá data (odpověď na dotaz databázi) k vykreslení a určení, zda se jedná o událost nebo záznam. Následně funkce v presenteru vrátí vygenerovaný obsah určený k vykreslení. O vzhled vykreslených dat se stará CSS knihovna Timeline. [24]

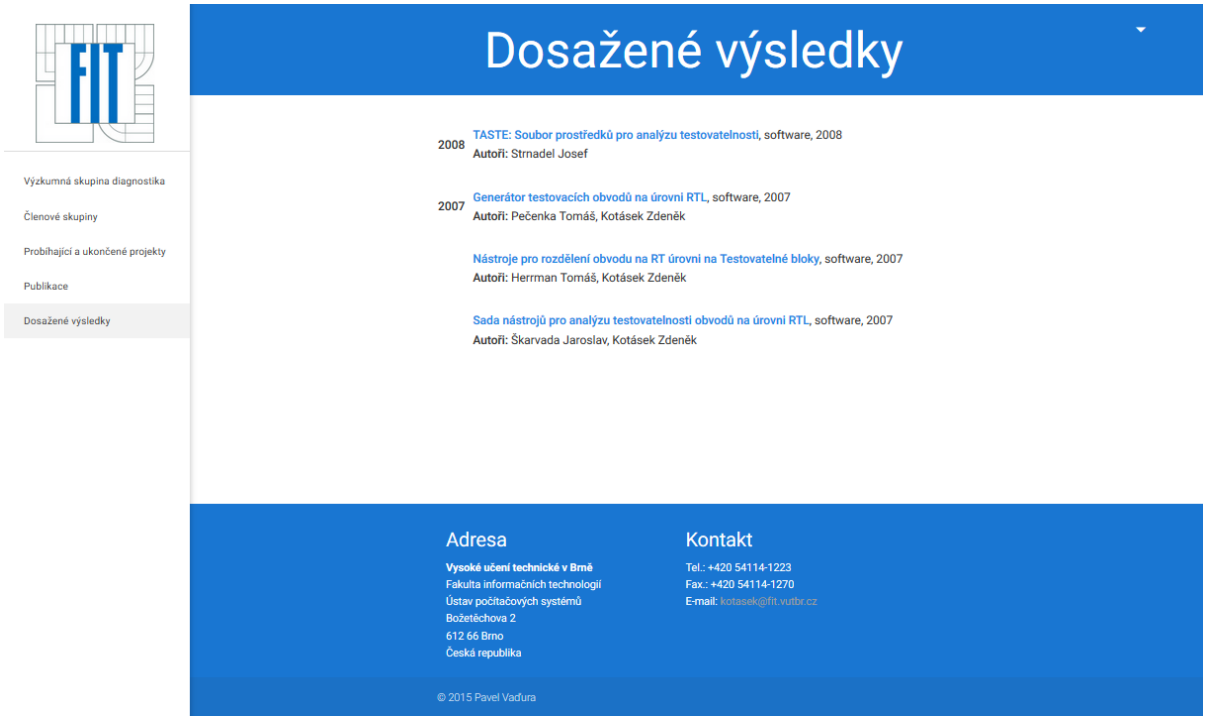
4.4 Layout systému

Pro vytvoření layoutu systému byl použit front-end responzivní framework Materialize [25] založený na Material Designu. Materialize framework je vyvíjen týmem studentů z Univerzity Carnegie

Mellon pod licencí MIT. Jedná se o soubor kaskádových stylů kooperujících s JavaScriptovými knihovnami, které zajišťují funkcionalitu virtuálních komponent, např. kalendář na obrázku 4.5.

Material design je vizuální jazyk vytvořený společností Google, jenž byl představen v půli roku 2014 na konferenci Google I/O. Jedná se o moderní design určený pro efektivnější využití dostupného prostoru přinášející konzistenci uživatelského prostředí napříč zařízeními, jako je stolní počítač, tablet a chytrý telefon. Součástí charakteristického designu je použití animací, které usilují o co největší vypovídací hodnotu s důrazem na plynulost. Dalšími charakteristickými prvky je font písma, tzv. Roboto, použité stínování prvků a paleta použitých barev. [26]

Layout se skládá ze čtyř hlavních částí (viz Obrázek 4.6). Horní okraj stránky je lemován modrým pruhem, ve kterém se nachází hlavní nadpis stránky. V pravé části pruhu je umístěn rozbalovací seznam, kde se pro nepřihlášeného uživatele po kliknutí zobrazí možnost přihlásit se do systému. Záměrně zde není uveden žádný popisek, aby to náhodného návštěvníka neodpoutávalo od prohlížení stránek a zároveň zbytečně nenavádělo na přihlašovací formulář, kde by se mohl pokoušet odhalit přihlašovací údaje. V případě přihlášeného uživatele je zobrazeno přihlašovací jméno uživatele a seznam nabízející možnosti přepnutí z administrační části do veřejné a opačně, možnost přejít na správu osobních informací a odhlášení ze systému. Po celé výšce levé strany je umístěno navigační menu systému, které zdánlivě překrývá horní pruh. První položkou navigačního menu je vždy logo vedoucí na výchozí stránku systému ve veřejné části a na přehledovou stránku v části administrační. Třetí částí je prostor pro obsah stránek, jenž je umístěn na bílém podkladu a zarovnán na střed tak, aby byl postranními bílými pruhy oddělen od navigačního menu a konce obrazovky. Poslední částí je patička obsahující kontaktní údaje, která se vzhledem podobá hornímu pruhu stránky.

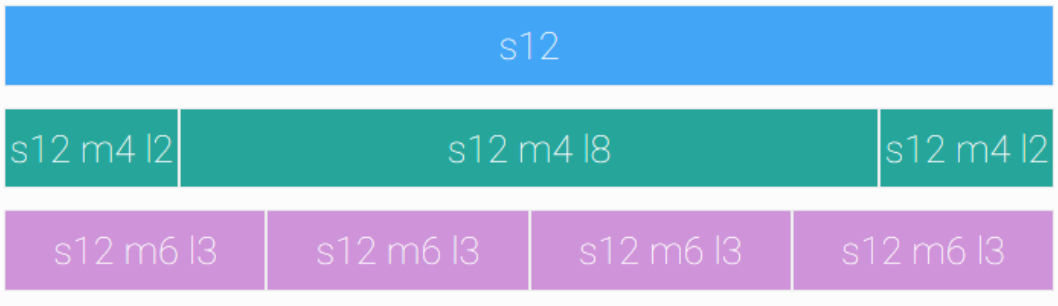


Obrázek 4.6: Layout aplikace

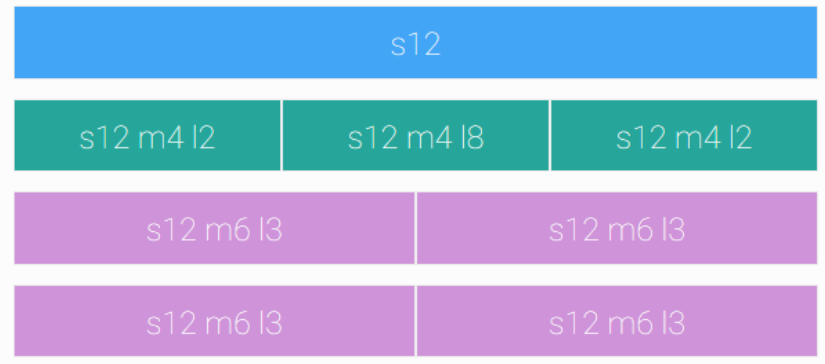
4.4.1 Responzivní layout

Jak již bylo zmíněno výše, použitý front-end framework Materialize je responzivní, takže dokáže přizpůsobit obsah stránky zařízení, na kterém se má výsledek zobrazit. Možností rozpoznání zařízení se nabízí několik. Pomocí kaskádových stylů je možné stanovit chování stránky pro několik různých rozměrů obrazovky. Také lze rozpoznávat zařízení pomocí http hlaviček, které odesílá prohlížeč a podle daných hlaviček uživateli poskytnout patřičný zdrojový kód. Další možností je použití JavaScriptové knihovny, např. jQuery Mobile Framework, kterou se nadefinuje chování částí stránky. [27]

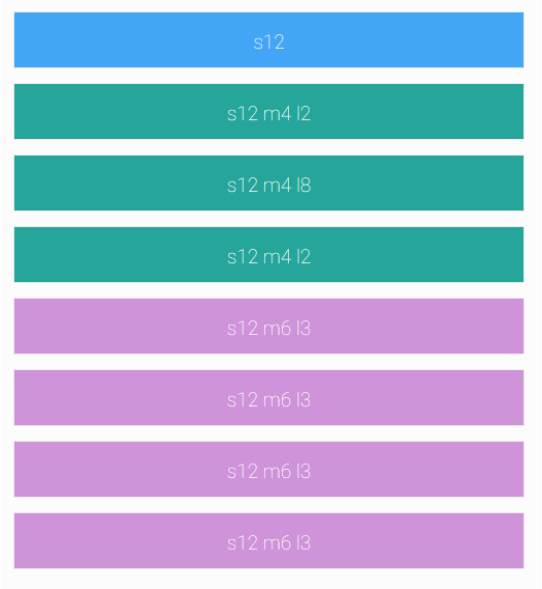
Použitý framework v systému pracuje s kaskádovými styly, které jsou rozděleny do tří kroků, tzv. breakpoint. Konkrétně jsou tyto kroky rozděleny podle šířky na mobilní zařízení (do 600 px), tabletová zařízení (do 992 px) a stolní zařízení (nad 992px). Nastavování šíře jakéhokoliv HTML prvku se provádí pomocí určení třídy. Každý krok je rozdělen na dvanáct částí, tzn. nastavení vlastnosti je dáno tím, o jaký krok se jedná - s (small), m (medium), l (large), a počtem částí. Plné zobrazení ve všech krocích se zapíše jako `s12` (viz modrý obdélník na obrázcích 4.7, 4.8 a 4.9). Obrázky znázorňují chování prvků s nastavenými vlastnostmi pro zobrazení na stolním, tabletovém a mobilním zařízení. Zde můžeme vidět chování prvků při změně šíře zobrazované plochy. Vždy je dodržován počet dvanácti částí na řádek, jak je patrné u růžových obdélníků, které se v tabletovém zobrazení řadí pod sebe (viz Obrázek 4.8). Nastavené vlastnosti zároveň respektují vlastnosti rodičovských prvků, tzn., že pokud budeme mít prvek s vlastností `s6` a tento prvek bude obsahovat další prvek taktéž s vlastností `s6`, bude výsledná šíře potomka čtvrtina zobrazované plochy. Toto chování zajistí, že nebude docházet k nevhodnému překrývání prvků nebo k rozbití vzhledu aplikace jiným způsobem.



Obrázek 4.7: Ukázka layotu při zobrazení na stolním zařízení



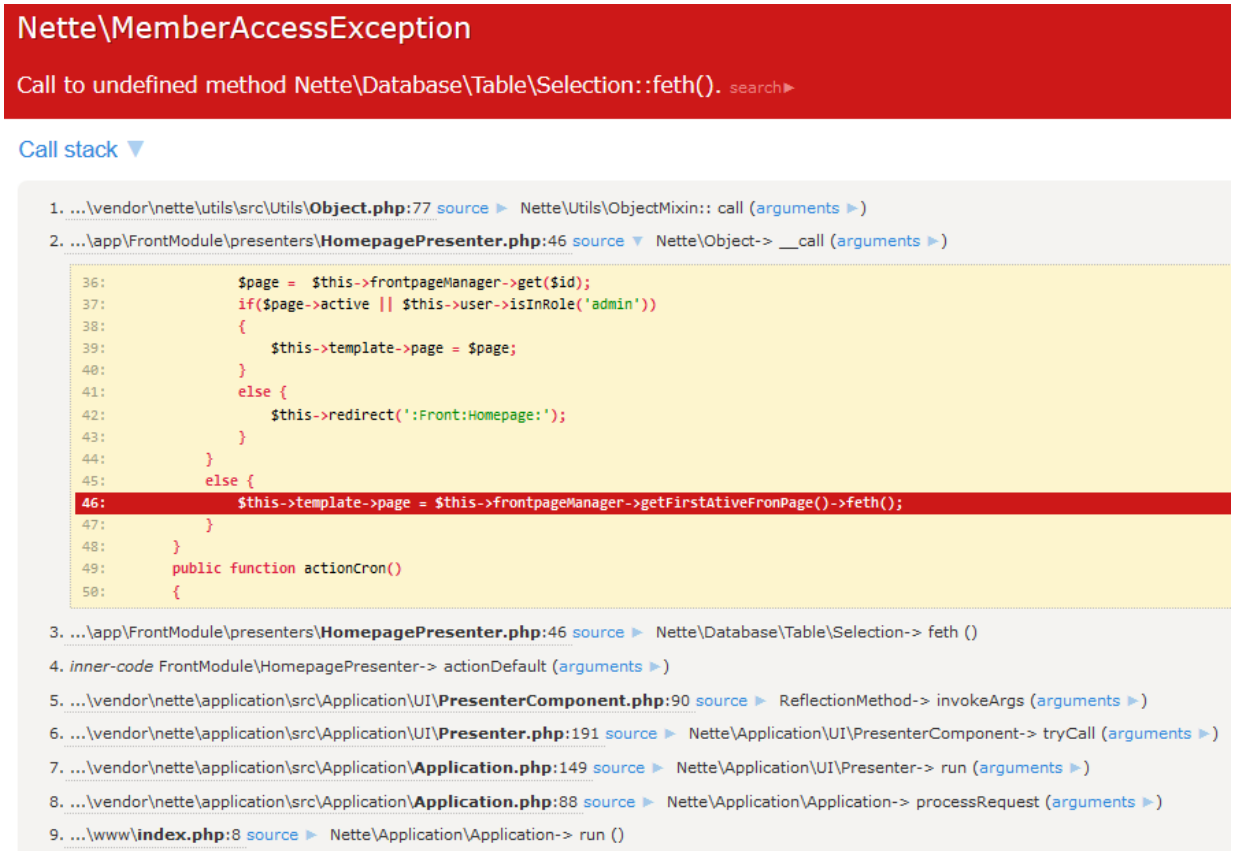
Obrázek 4.8: Ukázka layotu při zobrazení na tabletovém zařízení



Obrázek 4.9: Ukázka layotu při zobrazení na mobilním zařízení

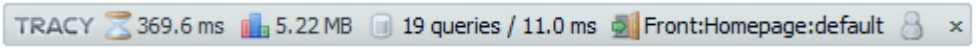
5 Testování

Testování a ladění systému probíhalo po celou dobu implementace webové aplikace. Největší část chyb, převážně syntaktických, se podařilo odhalit již v editoru. Editor NetBeans provádí syntaktickou analýzu psaného kódu a dokáže upozornit na chybné konstrukce. Chyby, které neodhalil editor, hlásil interpret jazyka PHP, respektive knihovna Nette frameworku `Tracy\Debugger` nazývaná také jako laděnka. [28] Při výskytu chyby knihovna informuje o dané chybě a ukáže část zdrojového kódu, kde chyba vznikla. Zároveň zobrazí definované proměnné spolu s jejich obsahem, viz Obrázek 5.1.



Obrázek 5.1: Laděnka Nette frameworku

Dalším užitečným nástrojem laděnky je tzv. debugger bar. Tato plovoucí lišta je umístěna na všech stránkách v pravém dolním rohu. Lišta zobrazuje informace spojené s vykonáváním skriptu, jako je čas trvání běhu skriptu, paměťová náročnost skriptu a zobrazení provedených SQL dotazů do databáze. Debugger bar zobrazí nejen SQL dotazy, ale také časovou náročnost každého dotazu a samozřejmě i data, kterými databáze odpovídá. Do Debugger Baru lze přidávat další rozšíření. Ukázka podoby lišty je na obrázku 5.2.



Obrázek 5.2: Debugger bar

Popsané ladící nástroje lze samozřejmě provozovat jen na vývojové verzi aplikace. Je nepřípustné, aby měl návštěvník v případě výskytu nějaké chyby možnost zjistit části programu,

případně jaké SQL dotazy se provádějí při vykreslení stránky, a to hlavně z bezpečnostního důvodu. Na tuto skutečnost dokáže ladící knihovna zareagovat. Dochází totiž ke kontrole, zda se aplikace spouští z veřejné IP adresy, a všechny chybové zprávy se zapíší do souboru `log/error.log`.

Pro testování vizuálního vzhledu aplikace byly použity tři nejpoužívanější prohlížeče. Stránky byly testovány v prohlížeči Mozilla Firefox verze 35.0.1, Internet Explorer verze 11 a Google Chrome verze 42.0.2311.90. V těchto prohlížečích se stránky zobrazily korektně dle definovaného grafického návrhu. Zároveň byla kontrola provedena i na smartfonu a tabletu, kde se projevily vlastnosti responzivního layoutu.

Při ladění aplikace bylo zapotřebí vypisovat obsahy polí a proměnných. K této činnosti byla použita funkce `Debugger::dump()` vycházející z funkce `var_dump()`. Obě tyto funkce dokáží virtualizovat vypsané pole včetně indexů pole. Důvodem pro použití funkce `dump()` je její podpora ze strany frameworku a hlavně fakt, že bez dalšího zásahu dokáže zobrazit naformátované informace v HTML stránce, zatímco výstup funkce `var_dump()` se slije do nepřehledného bloku textu.

6 Demonstrační aplikace

V této kapitole je popsán postup pro instalaci systému na webový server. Pro chod webové aplikace je vyžadován interpret jazyka PHP minimálně verze 5.3.1. Pro databázový server MySQL je vyžadována minimálně verze 5.6. Nette Framework klade zároveň jisté požadavky na webový server a nastavení PHP. Kontrolu všech požadovaných vlastností lze provést spuštěním PHP skriptu `checker.php` na daném serveru, který je součástí Nette Frameworku. [29]

6.1 Instalace aplikace

Na přiloženém CD jsou dostupné všechny zdrojové kódy aplikace. Instalace aplikace se provede zkopírováním adresáře `/bpis`, který je umístěn v hlavní složce, na webový server. V případě, že zdrojové kódy umístíme na server, jenž běží na linuxovém operačním systému, je potřeba zkontrolovat přístupová práva souboru. Dále je potřeba vytvořit MySQL databázi a její přístupové údaje, do které se pomocí souboru `bpis_database.sql` importují tabulky a pomocí souboru `bpis_storage_database.sql` data pro základní nastavení aplikace. Oba výše jmenované soubory nalezneme na přiloženém CD. Po vytvoření a naplnění databáze je potřeba v aplikaci nastavit připojení do této databáze. Připojení se nastavuje v konfiguračním souboru `/bpis/app/config/config.neon`. Po těchto krocích je aplikace připravena na adrese `http://adresa_serveru/bpis/www/`. Aby systém automaticky rozesílal pozvánky na události, je potřeba nastavit opakované spouštění skriptu (dvakrát denně), který nalezneme na adrese `http://adresa_serveru/bpis/www/homepage/cron/`.

6.2 Popis aplikace

Webová aplikace se skládá z veřejné a administrační části. Při navštívení webové aplikace se uživatel dostane do veřejné části, kde jsou zobrazeny informace o Výzkumné skupině diagnostika. V této části se nalezne na levé straně navigační menu, které umožňuje procházet veřejně přístupné informace. Prvním prvkem navigačního menu je odkaz v podobě loga fakulty vedoucí na hlavní stránku. Zbytek obsahu navigačního menu je definován nastavením veřejných stránek v administrační části. V pravém horním rohu se nachází rozbalovací nabídka, která nabývá dvou podob. Pro nepřihlášeného uživatele nabídka zobrazí možnost přihlášení do systému. Přihlášenému uživateli se zobrazí možnost přechodu do administrační části, na správu osobních údajů a odhlášení ze systému.

Při přechodu do administrace se kontroluje přihlášení uživatele, pokud uživatel není přihlášen do systému, je přesměrován na přihlašovací formulář. V administrační části nalezneme opět po levé straně navigační menu obsahující odkazy na dostupné sekce administrační části. Úvodní stránka zobrazuje rychlý přehled aktivit systému. Nalezneme zde výpis tří nejnovějších a tří nejbližších událostí. U každé události nalezneme základní informace (název události, datum a čas konání) a po kliknutí na danou událost můžeme zobrazit bližší informace o pořadateli, popis akce a odkaz na detailní popis. Obdobně jako události jsou zobrazeny i tři aktuální výpůjčky. Pod popsanou nabídkou je vykreslena časová osa, ve které jsou uvedeny záznamy aktivit přihlášeného uživatele. Zde můžeme nalézt základní informace o jednotlivých aktivitách a ikony pro práci se záznamem (smazat, upravit, zobrazit detail). Následující část je zaměřena na popis jednotlivých sekcí.

Do sekce `Veřejná část` určené pro zprávu stránek veřejné části může vstoupit pouze uživatel s rolí administrátora. Při vstupu do této sekce se v podobě tabulky zobrazí výpis všech stránek určených pro veřejnou část. Stránky jsou řazeny podle váhy, kterou vidíme v prvním sloupci dané tabulky. Následují sloupce s titulkem, nadpisem a textem. V předposledním sloupci je vykreslena ikona, která informuje uživatele, zda je povoleno zobrazování dané stránky ve veřejné části. V posledním sloupci se nacházejí ikony pro zobrazení dané stránky ve veřejné části, úpravu a smazání stránky. Novou stránku můžeme přidat kliknutím na ikonu nad tabulkou.

Další sekcí přístupnou pouze administrátorovi je `Nastavení systému`. V této sekci můžeme nastavit doplňující informace sloužící k popisu stránky zobrazené ve veřejné části, a to titulek, popis a metadata systému.

Sekce `Výpůjčky`, která se zobrazuje všem přihlášeným uživatelům, obsahuje tabulku s informacemi o výpůjčkách přihlášeného uživatele s možností sledovat stav výpůjčky (navráceno/nenavráceno), upravovat ji nebo případně smazat. V případě, že vypůjčená věc nebyla navrácena, zobrazuje se další ikona pro rychlé navrácení. Nad tabulkou je opět umístěn odkaz na vložení nové výpůjčky.

V sekci `Záznamy` nalezneme časovou osu s výpisem záznamů, podobně jako na titulní straně administrační části. Na časové ose jsou záznamy uživatele seřazeny podle stáří od nejnovějšího po nejstarší. Pro lepší přehlednost zobrazovaných dat má každá událost na ose vyznačen aktuální stav záznamu. Možnými stavy jsou: záznam je neukončený, záznam je neukončený po době předpokládaného ukončení, záznam je uzavřen. U každého záznamu se nacházejí ikony pro rychlé dokončení záznamu, smazání záznamu, úpravu záznamu a přechod na detailní stránku se záznamem. Dalšími prvky v sekci jsou tlačítka pro přidání nového záznamu a tlačítko, které změní výpis záznamů z časové osy na tabulku.

Sekce `Události` je podobná sekci Záznamy - nachází se zde časová osa s událostmi, na které je uživatel pozván nebo je jejich organizátorem, možnost přepnutí výpisu událostí na výpis pomocí tabulky a možnost vytvoření nové události. Pro vytvoření události je nutné vyplnit do formuláře informace o dané události, tj. název, popis a místo konání. Volba možných termínů konání se provede přidáním možného data kliknutím na tlačítko „PŘIDAT DALŠÍ DATUM“. Po přidání data se do formuláře vloží box, ve kterém se určí den konání události. Dále box obsahuje tlačítka „ODEBRAT DATUM“ pro zrušení data a tlačítko „PŘIDAT ČAS“, jež přidává do boxu vstup pro zadání možného času konání události, checkbox pro učení skutečného termínu konání a tlačítko pro smazání. Po nastavení vhodných termínů konání události a odeslání formuláře dojde k přesměrování na formulář, kde organizátor zvolí uživatele, kteří budou pozváni na událost. Výběrem uživatelů se dokončí proces vytvoření události.

V sekci `Uživatelé` se nachází přehled všech uživatelů v systému včetně informací o nich (příjmení, jméno, telefon, e-mail, pracovna). Tento přehled je znázorněn pomocí tabulky, ve které nalezneme uživatele seřazené abecedně podle příjmení. U každého uživatele se nachází odkaz na detailní popis uživatele a u vlastního záznamu má uživatel možnost přejít na formulář, kde upraví své osobní informace. V detailním popisu uživatele nalezneme jeho osobní informace, aktuální výpůjčky a časové osy zobrazující záznamy a události, kterých se daný uživatel zúčastnil nebo zúčastní. Uživateli s administrátorskými právy se navíc v sekci `Uživatelé` zobrazí ikony pro vytvoření, editaci a smazání uživatele. Proces vytvoření nového uživatele znamená vyplnění povinných údajů. Poté bude administrátor přesměrován na formulář s výpisem všech stávajících událostí, na které může nového uživatele pozvat.

7 Závěr

Tato práce se zabývala návrhem a implementací informačního systému pro Výzkumnou skupinu diagnostika. Systém reflektuje specifické požadavky skupiny, proto nebylo možné použít open-source redakční systém. Hlavní dovedností systému je možnost spravovat události (schůzky) a plánovat termín konání dané schůzky. Jako doplňující funkce reflektující požadavky výzkumné skupiny jsou implementované záznamy výpůjček, správa záznamů o činnosti uživatele a správa stránek pro veřejnou sekci.

Začátek práce se zabýval popisem technologií vhodných k použití. Kromě technologií, které vyžadovalo zadání práce, byly popsány i alternativní možnosti nástrojů, jako jsou PHP frameworky. Zároveň zde byly uvedeny nástroje Git a Composer, které přímo nesouvisely s implementací systému, ale napomáhaly k efektivnější práci při vývoji aplikace.

Po definování nástrojů vhodných pro implementaci systému následovala část, kde se pojednávalo o návrhu systému. Na základě požadavků výzkumné skupiny se vytvořila neformální specifikace systému, která posloužila jako základ pro analýzu systému a návrhu diagramu případů použití. Na základě tohoto diagramu byli navrženi jednotliví aktéři a jejich práva. V poslední části návrhu systému se stanovila podoba schématu relační databáze.

Pro implementaci aplikace byl zvolen objektový PHP Nette Framework, jehož návrhová architektura vychází z návrhového vzoru MVC. Volba frameworku ovlivnila i popsané vlastnosti aplikace, např. adresářovou strukturu vycházející z doporučené podoby, tzv. sandbox, nebo úložiště InnoDB použité v databázi MySQL. Následoval popis tvorby veřejné části aplikace. Konkrétně zde byly uvedeny postupy a principy pro vytvoření navigačního menu a způsob zobrazování webové stránky určené pro širokou veřejnost. Zvláštní část se věnovala způsobu vytváření a zpracování url adres pomocí adresovacích pravidel. Posledním bodem ve věřené části byla tvorba podpory pro indexovací roboty v podobě generování souboru `sitemap.txt`. Následovalo vytvoření administrační části, kde byla popsána práce s databází, tzv. databázová vrstva. Pro přehlednost nebyla v práci věnována pozornost implementaci všech částí administrace, nýbrž pouze vybraných částí, které názorně ukázaly princip implementace aplikace. Formuláře zastupoval dynamický formulář pro vytvoření události, ve kterém bylo použito i rozšíření `Kdyby\Replikator`. V práci bylo popsáno vytvoření formuláře, způsob jeho zpracování a uložení do databáze. Zároveň formulář posloužil pro vysvětlení implementace pohledů sytému, tzv. view. Do aplikace byly implementovány i dvě komponenty, z nichž první komponenta EventPlanning zobrazuje termíny k dané události s možností volby vyhovujících termínů. Druhá komponenta TimeLine zajišťuje vykreslení událostí nebo záznamů do časové osy. V poslední části implementace se práce zabývala navrženým layoutem aplikace. Zároveň byly uvedeny způsoby postupu pro implementaci responzivního layoutu.

Poslední část práce se věnovala nástrojům a způsobům testování realizovaného systému. Spolu s návodem na instalaci aplikace na webový server byl popsán způsob použití všech částí vytvořeného systému.

V důsledku vytvoření specifického systému podle požadavků výzkumné skupiny vznikla specializovaná aplikace, jejíž výhody jsou rychlost, nízké požadavky na úložiště a možnost snadné implementace nových funkcí nejen díky objektovému návrhu. Vzhled aplikace působí celistvým a moderním dojmem díky použití front-end frameworku Materialize založeném na Material designu.

Možnosti dalšího rozvoje systému jsou rozsáhlé a z velké části budou reflektovat požadavky vývojové skupiny, které vyplynou z dalšího užívání aplikace. Jako možnost rozšíření se zde jeví

napojení na produkt kalendář společností Google a Microsoft. Takové rozšíření by umožnilo ihned po stanovení závazného termínu události tuto informaci delegovat do kalendářů uživatelů, kteří by tak měli větší přehled o nadcházejících událostech. Zároveň by tím odpadla nutnost přepisovat událost do kalendáře pro ty uživatele, kteří tyto kalendáře využívají jako diář napříč různými zařízeními. Dalším směrem, kam by se mohly ubírat možnosti vývoje systému, je zavedení podpory práce se soubory. Nyní aplikace umožňuje pomocí WYSIWYG („What you see is what you get“) editoru rozšířeného o správce souborů elFinder vkládat do textu soubory, převážně obrázky. Vhodným místem pro podporu vkládání souborů jako příloh jsou bezesporu zápisy výsledku události a zápisy o činnosti.

Literatura

[1] *Doodle zjednodušuje plánování* [online]. 2015. [cit. 2015-05-05]. Dostupné z: http://doodle.com/cs/

[2] KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora*. Vyd. 1. Brno: Computer Press, 2007, 607 s. ISBN 978-80-251-1813-9.

[3] BC. OLIŠAR, Petr. *Porovnání Nette a Zend Frameworku*. Praha, 2012. Dostupné z: https://isis.vse.cz/zp/portal_zp.pl?podrobnosti=89656. Bakalářská práce. Vysoká škola ekonomická v Praze. Vedoucí práce Ing. Jan Mittner.

[4] Nette Framework: Rychlý a pohodlný vývoj webových aplikací v PHP. [online]. [cit. 2013-11-22]. Dostupné z: http://nette.org/cs/#toc-features

[5] Seriál: ORM test PHP frameworků (9 dílů). *Zdroják - o tvorbě webových stránek a aplikací* [online]. 2013 [cit. 2015-04-22]. Dostupné z: http://www.zdrojak.cz/serialy/test-php-frameworku/

[6] CakePHP: the rapid development php framework. [online]. [cit. 2013-11-22]. Dostupné z: http://cakephp.org/

[7] Zend Framework. [online]. [cit. 2013-11-22]. Dostupné z: http://framework.zend.com/

[8] KOFLER, Michael. *Mistrovství v MySQL 5*. Vyd. 1. Překlad Jan Svoboda, Ondřej Baše, Jaroslav Černý. Brno: Computer Press, 2007, 805 s. ISBN 978-80-251-1502-2.

[9] jQuery: The Write Less, Do More, JavaScript Library. [online]. [cit. 2014-01-16]. Dostupné z: http://jquery.com/

[10] Composer. *Composer* [online]. [cit. 2015-03-25]. Dostupné z: https://getcomposer.org

[11] Kdyby/Replicator. *GitHub* [online]. 2014 [cit. 2015-04-22]. Dostupné z: https://github.com/Kdyby/Replicator

[12] CHACON, Scott. *Pro Git*. Praha: CZ.NIC, c2009, 263 s. Edice CZ.NIC. ISBN 978-80-904248-1-4. Dostupné z: https://knihy.nic.cz/files/nic/edice/scott_chacon_pro_git.pdf

[13] MVC Architecture. *Mobile DevTools: Remote Debugging for Android with Screencast - Google Chrome* [online]. 2013 [cit. 2015-04-22]. Dostupné z: https://developer.chrome.com/apps/app_frameworks

[14] 15.2 The MyISAM Storage Engine. *MySql* [online]. [cit. 2015-03-24]. Dostupné z: http://dev.mysql.com/doc/refman/5.6/en/myisam-storage-engine.html

[15] 14.2 The InnoDB Storage Engine. *MySQL* [online]. 2015 [cit. 2015-04-20]. Dostupné z: http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html

[16] CKEditor. *CKEditor.com | The best web text editor for everyone* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: http://ckeditor.com/

[17] ElFinder. *ElFinder - file manager for web* [online]. 2012 [cit. 2015-04-25]. Dostupné z: http://elfinder.org/

[18] The time picker. *Pickadate.js* [online]. 2014 [cit. 2015-04-20]. Dostupné z: http://amsul.ca/pickadate.js/time/

[19] What are Sitemaps? *Sitemaps.org* [online]. 2008 [cit. 2015-04-20]. Dostupné z: http://www.sitemaps.org/index.html

[20] PHP: PDO - Manual. *PHP: Hypertext Preprocessor* [online]. 2001-2015 [cit. 2015-04-20]. Dostupné z: http://php.net/manual/en/book.pdo.php

[21] Database\Table. *Nette Framework* [online]. 2008, 21. 1. 2015 [cit. 2015-04-20]. Dostupné z: http://doc.nette.org/cs/2.3/database-table

[22] Šablony. *Nette Framework* [online]. 2008, 12. 3. 2015 [cit. 2015-04-22]. Dostupné z: http://doc.nette.org/cs/2.3/templating

[23] Výchozí Latte filtry. *Nette Framework* [online]. 2008, 21. 1. 2015 [cit. 2015-04-20]. Dostupné z: http://doc.nette.org/cs/2.3/default-filters

[24] Timeline. *Bootsnipp* [online]. 2013 [cit. 2015-04-22]. Dostupné z: http://bootsnipp.com/sergiors/snippets/yGbV

[25] Materialize. *Documentation - Materialize* [online]. 2014 [cit. 2015-04-20]. Dostupné z: http://materializecss.com/

[26] Introduction - Material design - Google design guidelines. *Material design* [online]. 2014 [cit. 2015-04-22]. Dostupné z: http://www.google.com/design/spec/material-design/introduction.html

[27] KADLEC, Tim. *Responzivní design profesionálně*. Vyd. 1. Brno: Zoner Press, 2014, 246 s. Encyklopedie Zoner Press. ISBN 978-80-7413-280-3.

[28] Debugování a zpracování chyb. *Nette Framework* [online]. 2008 [cit. 2015-04-22]. Dostupné z: http://doc.nette.org/cs/2.3/debugging

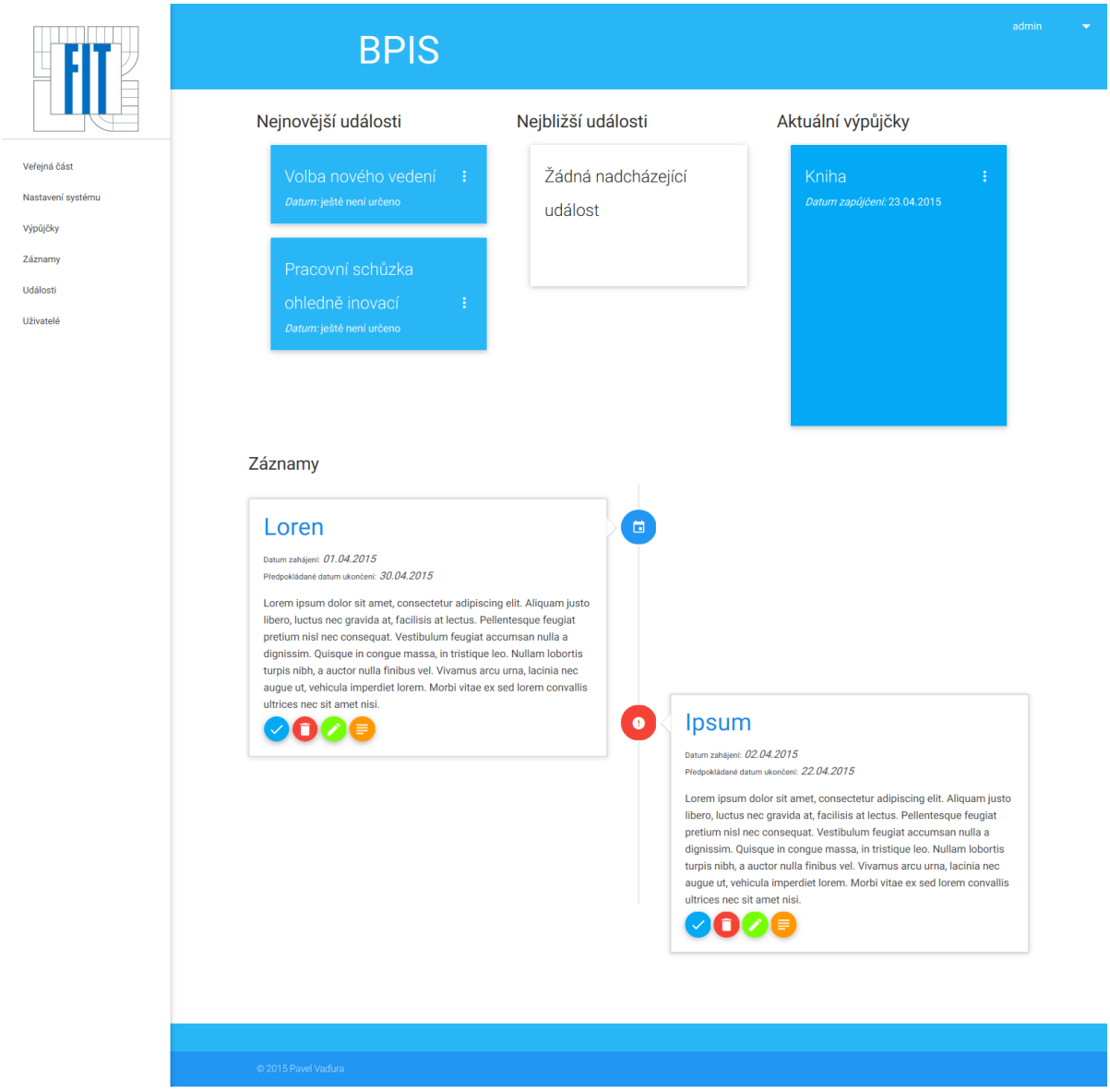
[29] Požadavky Nette Framework. *Nette Framework* [online]. 2008, [cit. 2015-04-25]. Dostupné z: http://doc.nette.org/cs/2.3/requirements

Seznam příloh

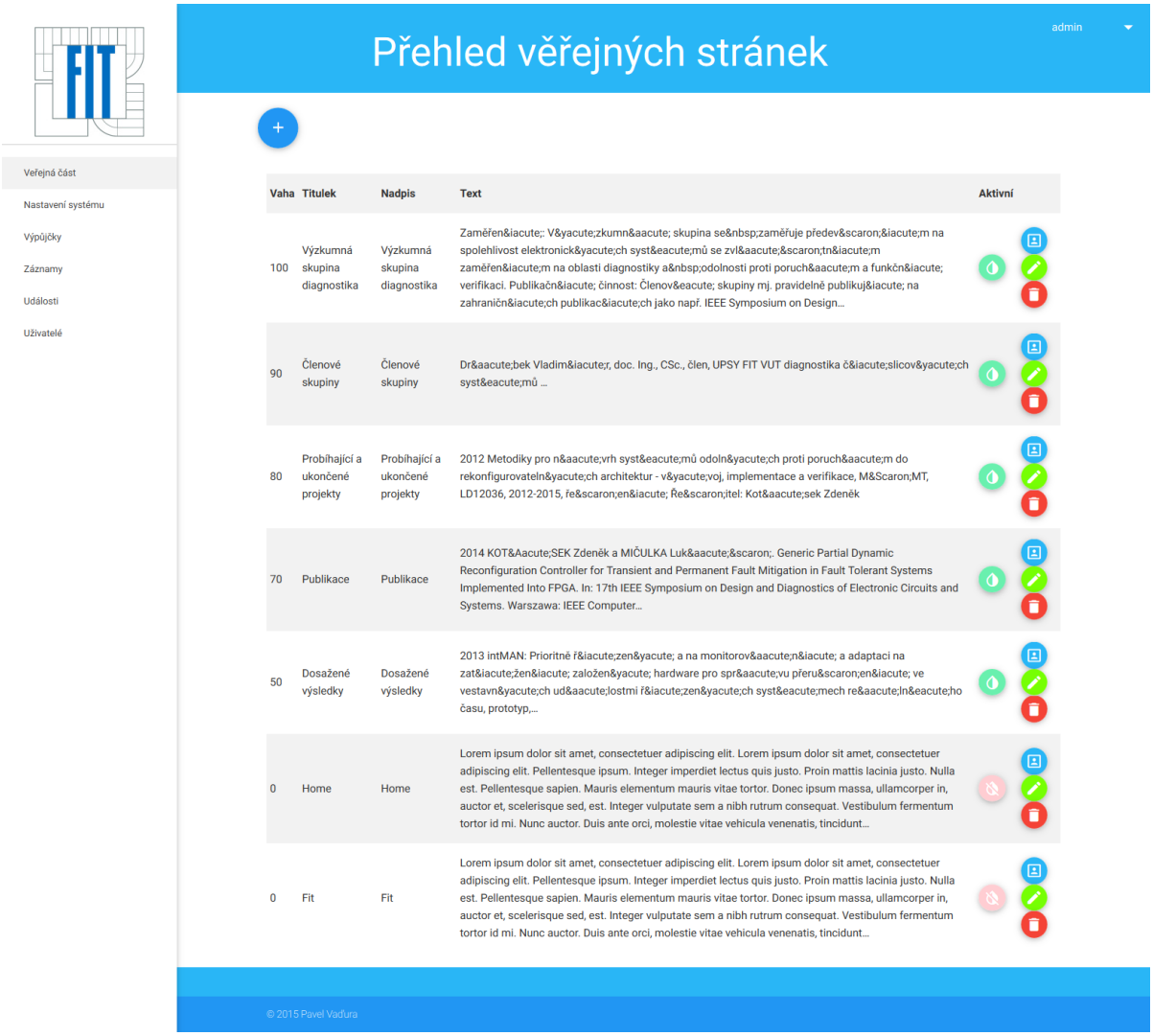
Příloha 1

Ukázky vzhledu aplikace

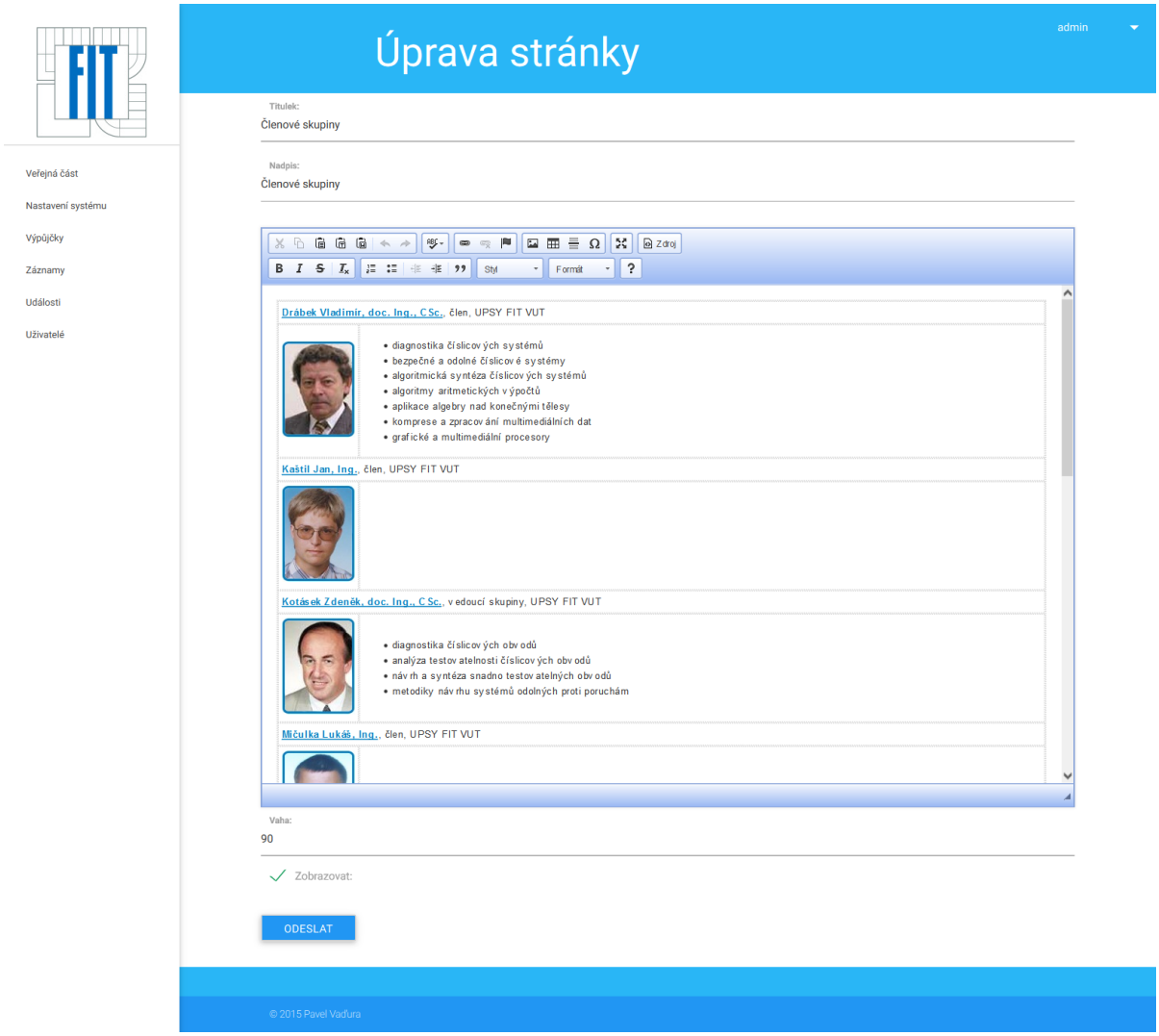
Úvodní stránka administrace



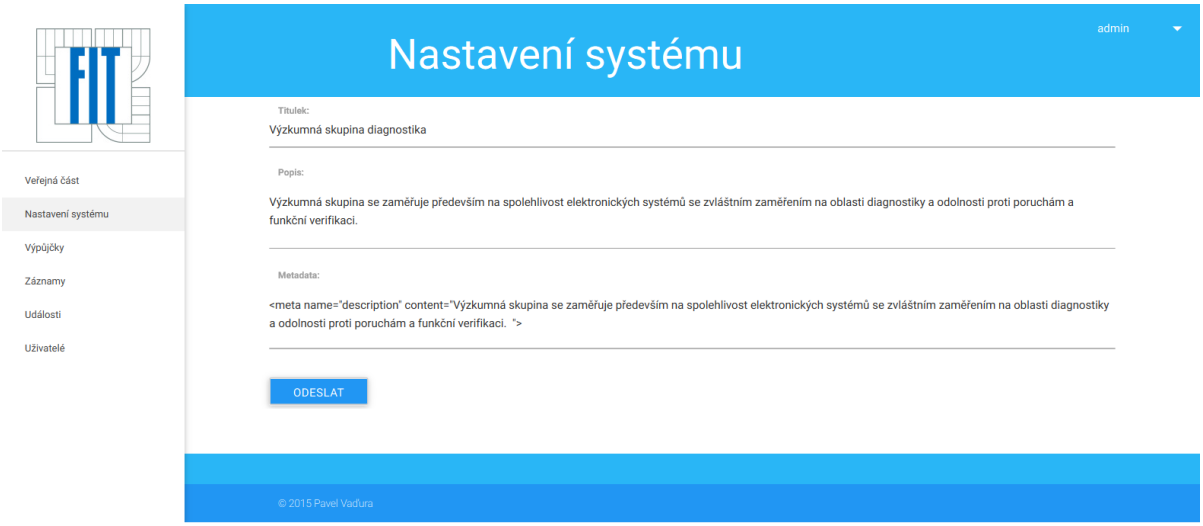
Stránka `Přehled veřejných stránek`



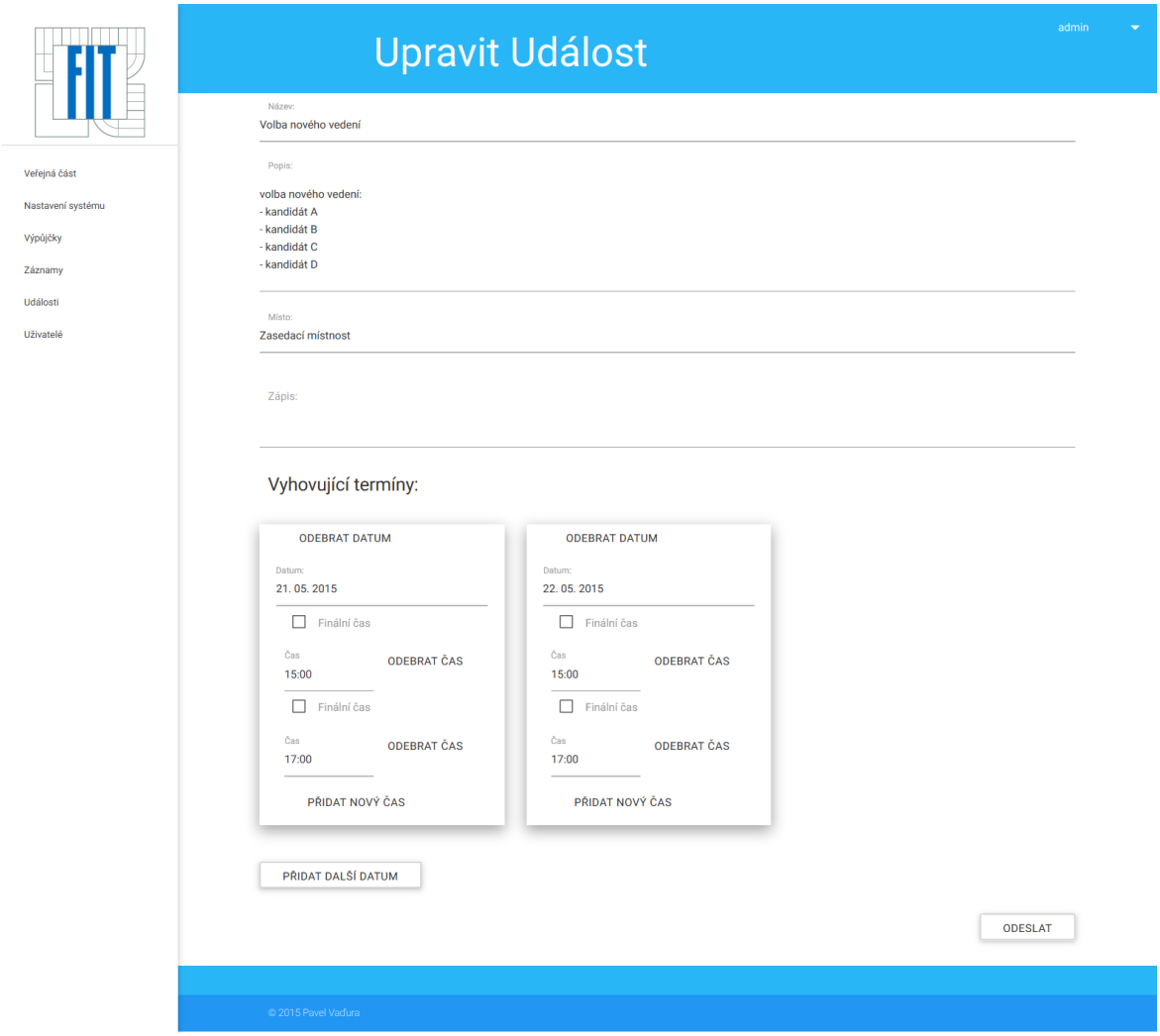
Stránka `Úprava stránky`



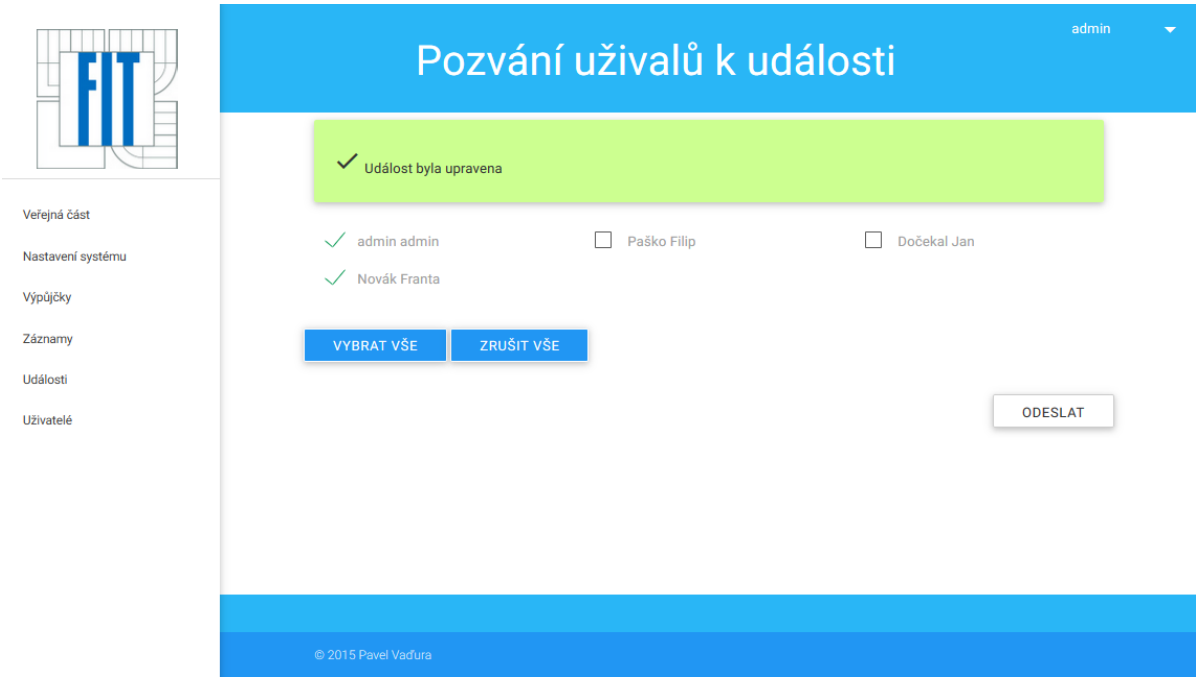
Stránka `Nastavení systému`



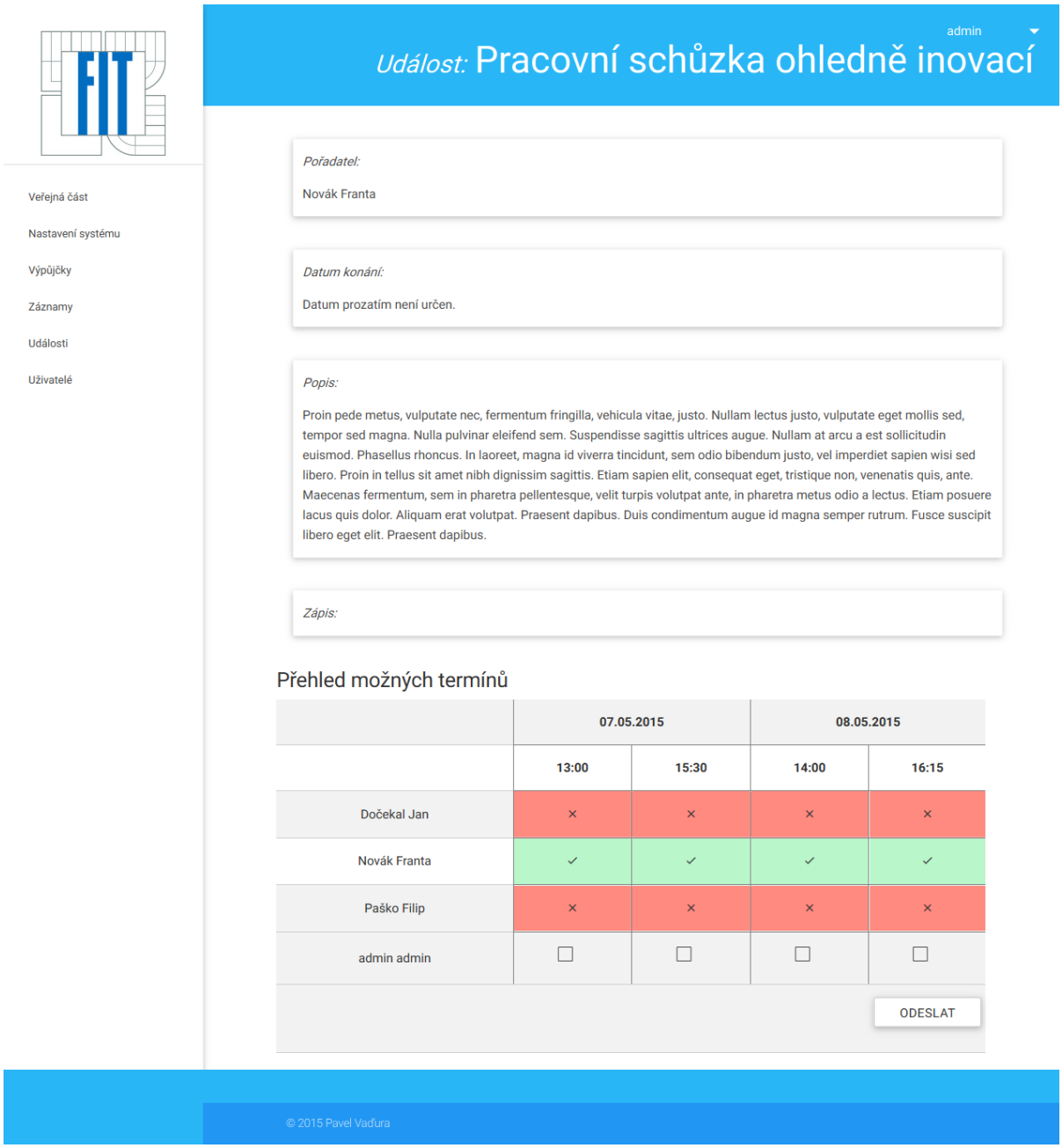
Stránka `Upravit Událost`



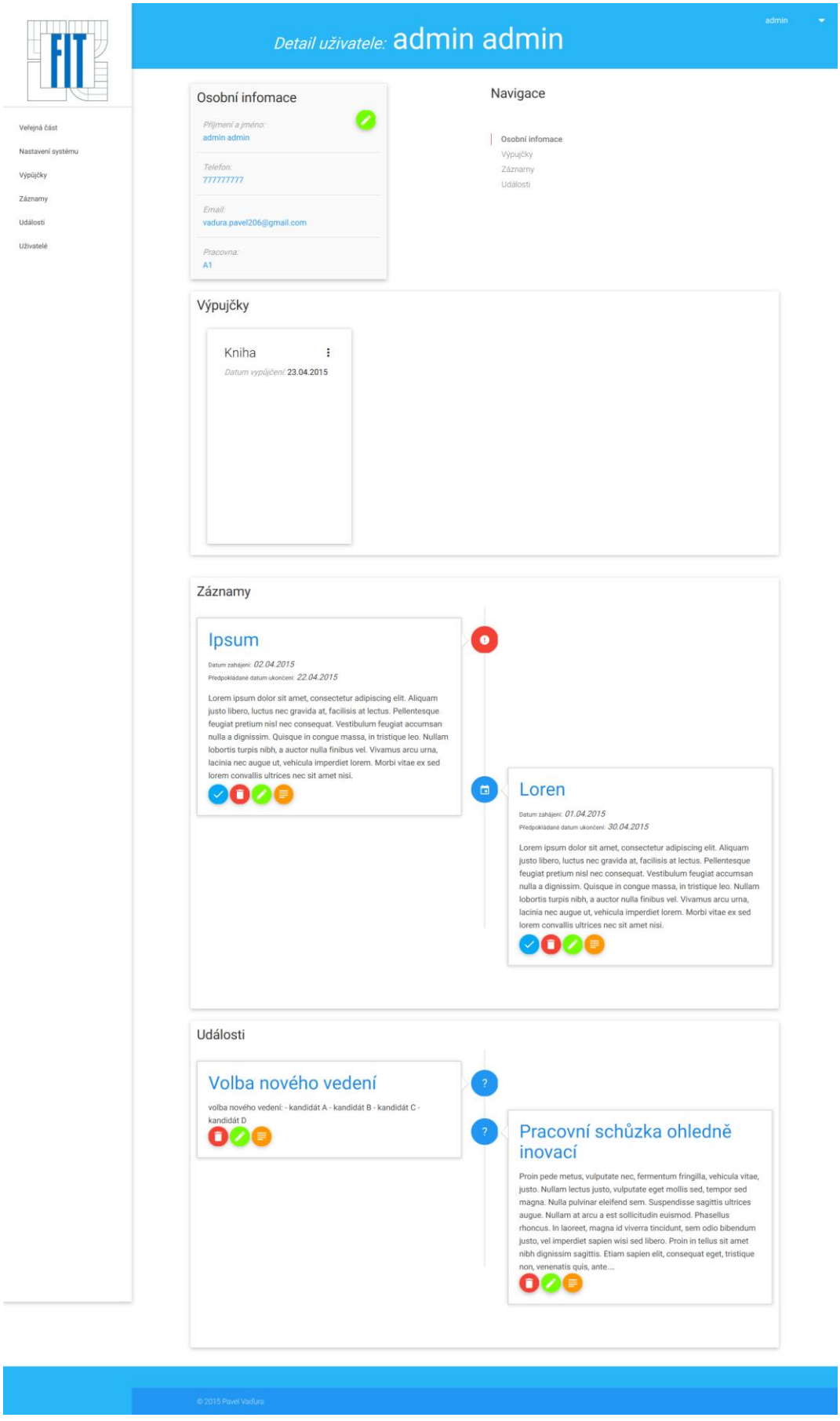
Stránka `Pozvání uživatelů k události`



Stránka s detailem události



Stránka s podrobnostmi o uživateli



Stránka `Přehled výpůjček`