VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webová aplikace na plánování času

Time Tracking Web Application

2016 Martin Pospíšil

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Pospíšil**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webová aplikace na plánování času
Time Tracking Web Application

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit webovou aplikaci, která bude umožňovat registrovaným uživatelům plánování a organizaci času. Aplikace bude dostupná v prostředí Internetu.

Aplikace bude umožňovat:
1. Vkládání událostí.
2. Možnost uživatele vytvářet si své typy událostí.
3. Upozornění na začátek události.
4. Zobrazení harmonogramu.
5. Zobrazení statistik.

Práce bude obsahovat:
1. Implementaci výše popsané webové aplikace.
2. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.
3. Uživatelskou dokumentaci aplikace.

Seznam doporučené odborné literatury:

[1] LECKY-THOMPSON, Ed a Steven D NOWICKI. PHP 6: programujeme profesionálně. Vyd. 1. Brno: Computer Press. Programujeme profesionálně. ISBN 978-80-251-3127-5.
[2] LAVIN, Peter. PHP - objektově orientované: koncepty, techniky a kód. 1. vyd. Praha: Grada, 2009, 211 s. ISBN 978-80-247-2137-8.
[3] GREEN, Brad a Shyam SESHADRI. AngularJS. First edition. x, 183 pages. ISBN 978-144-9344-856.
[4] GILMORE, W. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radoslav Štrba**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 19. dubna 2016

Bospíšil .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 19. dubna 2016 Rospíšil .....

Rád bych na tomto místě poděkoval vedoucímu své práce panu Ing. Radoslavovi Štrbovi, především za čas strávený na konzultacích a poskytnuté odborné rady.

Abstrakt

Čas se v dnešní uspěchané době stává mnohem důležitější a vzácnější veličinou, než-li tomu bylo dříve a organizování času nedílnou součástí života každého z nás. Jednotlivci, ale i celé společnosti jsou nuceny vykonávat stále více činností při zachování stejného časového rozmezí. Právě tento aspekt spolu se současným trendem využívání chytrých zařízení, která máme vždy po ruce a jež jsou stále připojená k internetu, je hlavní příčinou rychle rostoucí poptávky po aplikacích, které plánování času umožňují. Cílem této práce je nastudování a provedení analýzy současného stavu v této oblasti spolu s popisem trendů budoucího vývoje, přičemž se zaměřím na ta nevyužívanější dostupná řešení. Součástí práce je také implementace webové aplikace sloužící ke správě času, která bude zahrnovat stěžejní funkce pro uživatele v přívětivé a jednoduché podobě.

Klíčová slova: webová aplikace, správa času, kalendář, AngularJS, PHP

Abstract

Time becomes much more important and precious nowadays in comparison with earlier times and time tracking is essential part of our lives. Individuals, but also whole companies are forced to do more and more activities in the same time range. Just this aspect together with currect trned of using smart devices, which we always have at hand and are continuously connected to the internet, are the main reasons of rapidly growing number of demand for appliacations, that time management allow. The goal of this work is study and analysis of current state in this area with a description of trends in the years to come, focusing on the most used and available solutions. The next part of work is implementation of a new web application for time tracking, which will include crucial functions in user friendly and simple form.

Key Words: web application, time tracking, calendar, AngularJS, PHP

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
DBMS	–	Database Management System
DOM	–	Document Object Model
HTML	–	Hyper Text Markup Language
HTTP	–	Hypertext Transfer Protocol
JSON	–	JavaScript Object Notation
MVC	–	Model View Controller
OOP	–	Object-oriented programming
ORM	–	Object-relational mapping
PDO	–	PHP Data Objects
PHP	–	PHP: Hypertext Preprocessor
SPA	–	Single Page Application
SQL	–	Structured Query Language
WYSIWYG	–	What You See Is What You Get

Seznam obrázků

1 Úvod

Určitou správu času provádí bez výjimky každý z nás. Ať už se jedná o čas volný nebo o čas pracovní, cílem lidí je vždy jej využít co možná nejefektivněji. Běžným dřívějším řešením se staly papírové deníky, bloky s poznámkami a diáře, které neodmyslitelně patřily k časově velmi zaneprázdněným lidem a také například ke studentům. Se vzrůstajícími časovými nároky však přestávaly stačit. Nástup moderních technologií pak znamenal takřka okamžitou revoluci a pro správu času začíná stále více lidí využívat softwarové systémy, přičemž svoji pozici v současnosti nadále upevňují.

1.1 Cíle bakalářské práce

Stručný popis cílů, které jsem si definoval před započetím samotné práce. Tyto cíle byly postupně splněny.

1. Nastudování problematiky softwarových systémů se zaměřením na správu času, jejich srovnání a směry udávání se v budoucnosti.
2. Analýza a návrh webové aplikace pro správu času s vybranými funkcemi.
3. Implementace webové aplikace.
4. Tvorba programátorské a uživatelské dokumentace.

1.2 Stručný obsah jednotlivých kapitol

Práce se věnuje rozboru aktuálního stavu na poli aplikací pro správu času a jeho plánování. Dále se zaobírá implementací nové webové aplikace, která uživatelům umožňuje definované funkce.

Obsahem 2. kapitoly práce je její teoretická část. Ta se věnuje především krátkému průzkumu napříč v dnešní době nejvyužívanějšími softwarovými systémy pro správu a plánování času. V kapitole jsou dále sumarizovány a porovnány jejich klady i zápory.

V 3. kapitole bude předveden návrh řešení implementace webové aplikace pro správu času, který zahrnuje specifikaci zadání, popis funkčních i nefunkčních požadavků a koncept databázového modelu.

Kapitola 4 obsahuje programátorskou dokumentaci. Obsahuje stručný popis vlastností netradičních technologií použitých při implementaci vyvíjené webové aplikace. Dále představuje zajímavá řešení různých částí celé aplikace a řešení problémů, jež se během implementace objevily.

V poslední kapitole s číslem 5 jsou představeny plány budoucího rozvoje aplikace vyvinuté v rámci práce. Také jsou zde shrnuty dosažené výsledky a osobní přínos.

2 Teoretická část

Současná situace podobných aplikací je velmi nesourodá. Ty se totiž, nehledě na kvalitu celkového zpracování, liší svým konkrétním využitím a cílí na jinou skupinu uživatelů. Typově největší zastoupení mají aplikace pro správu pracovního času zaměstnanců, kteří v rámci společnosti spolupracují na úkolech, kde je výhodné nebo nutné práci časově organizovat. Při tomto konkrétním využití je často vyžadována i funkce pro fakturace a různá jiná vyúčtování. Druhé, již podstatně menší zastoupení, pak mají aplikace spravující čas osobní. [1]

Zatímco na českém trhu najdeme softwarových systémů obdobného zaměření jen poskrovnu, na trzích zahraničních se již delší dobu těší stále větší oblibě, a proto jich můžeme najít poměrně velké množství.

2.1 Vybrané existující aplikace podobného zaměření

Jak již bylo zmíněno, aplikací pro správu času můžeme, především tedy zahraničního původu, najít velké množství. V rozumném časovém rozmezí nelze a ani nemá smysl všechny analyzovat a popsat. Bude vybráno několik zástupců pro jednotlivá využití, přičemž bude přihlédnuto mimo jiné k uživatelské oblíbenosti.

2.1.1 Google Calendar

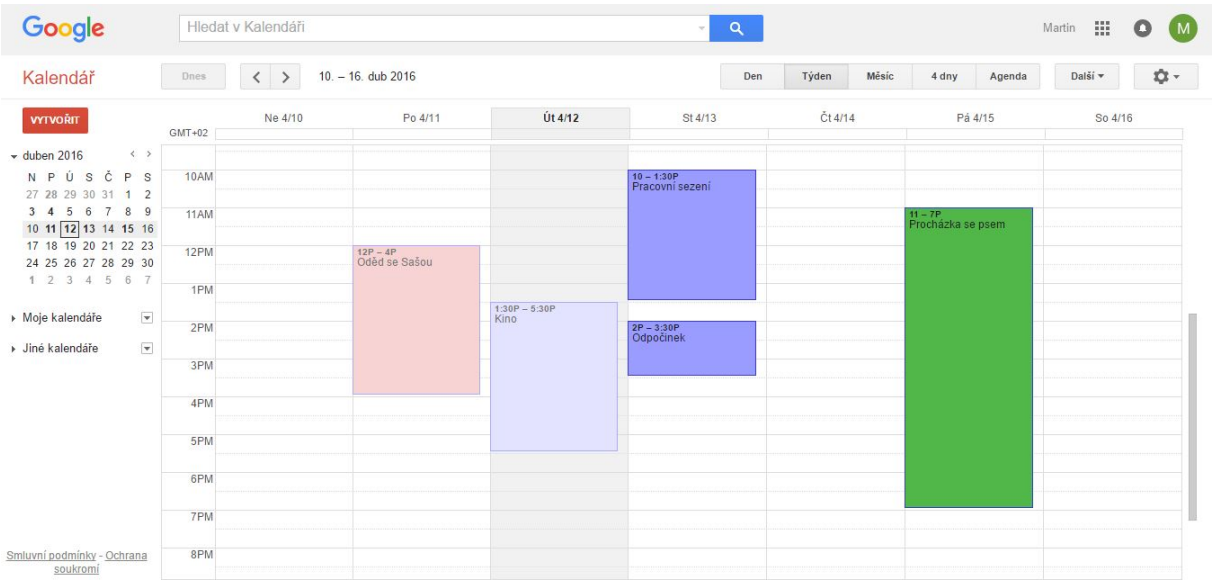
Google Calendar je jedna z nejznámějších služeb společnosti Google spuštěná v roce 2006. Přístup k ní mají všichni uživatelé, kteří vlastní Google účet, jež zpřístupňuje také ostatní služby společnosti. Využívání služby přes tento účet je zcela zdarma. Pro živnostníky a firmy se nabízí možnost provozování balíku služeb Google na vlastní doméně. Souhrnně se tomuto balíku říká Google Apps, který v základní verzi také není zpoplatněn. Za léta fungování služby se stala velmi hojně využívanou. Ačkoliv nabízí řadu pokročilých funkcí, v žádném konkrétním ohledu se nespecializuje a zaměřuje se na co možná největší skupinu potenciálních uživatelů. Služba je přeložena do mnoha jazyků včetně češtiny. [2]

Jednou z hlavních výhod služby je její dostupnost. Využívat ji lze nejen prostřednictvím webového prohlížeče, ale také prostřednictvím nativních aplikací pro různé operační systémy jako je např. Android OS, který je nejčastěji provozován na mobilních zařízeních. Mezi těmito zařízeními funguje automatická synchronizace, tedy uživatel pracuje vždy s aktuálními daty bez ohledu na způsob připojení ke službě. Data lze synchronizovat i s některými desktopovými klienty. Taktéž uživatelské rozhraní zůstává vždy stejné, což komfortu z užívání rovněž prospívá. K dalším z hlavních výhod patří integrace s jinými službami. Snadno lze do časového plánu přidat události přijaté v e-mailech služby Google Gmail, která je v nich umí sama rozpoznat. Další užitečná integrace je se službou Google Maps. Díky ní totiž uživatelé mohou připojit k události i její místo, které se jim spolu s událostí zobrazuje na mapě. Na toto místo se pak mohou nechat službou také navigovat. Užitečnou schopností je také sdílení kalendáře s více uživateli,

kteří jej pak mohou nezávisle na sobě upravovat. Toto sdílení je podpořeno obrovskou rozšířeností služby. V neposlední řadě služba umožňuje dostatečné možnosti drobnějších přizpůsobení, jako je nastavení způsobu oznamování začátků událostí, jejich barevná odlišení apod. [2]

Mezi nevýhody služby patří zejména její obecnost. Nehodí se pro specifická využití, která mohou být leckdy požadována, především pak při používání ve větších společnostech.

Z pohledu současného stavu je na poli aplikací pro plánování času služba Google Calendar první volbou pro většinu uživatelů. Při použití ve větších firmách však často nemusí vyhovovat pro svou obecnost, tedy chybějící specifické funkce, které právě firmy často potřebují.



Obrázek 1: Ukázka prostředí aplikace Google Calendar. [2]

2.1.2 PrimaERP Time Tracking

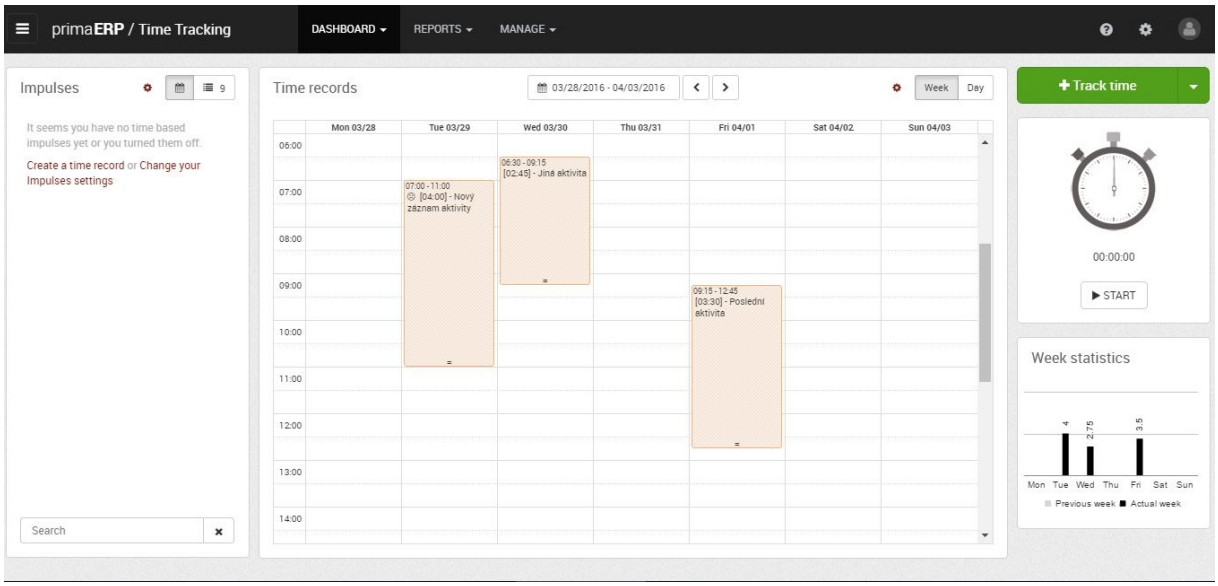
PrimaERP je projekt české společnosti ABRA Software. Společnost vznikla v roce 1991 s cílem produkovat software pro podnikatele. Poskytuje mimo jiné online cloudové účetnictví a webové i mobilní aplikace na zakázku. Produkty společnosti používá více než 20 tisíc zákazníků. [3]

Aplikace PrimaERP je primárně určena pro střední a velké firmy. Je zdarma pouze při minimalistickém použití, pro většinu zákazníků tak je ale zpoplatněna. Celková cena se odvíjí od počtu uživatelů, kteří aplikaci budou využívat. Samotný produkt je rozdělen do třech modulů. První modul se nazývá Attendance. Jedná se o správu docházky zaměstnanců, ve které se evidují pracovní časy a přestávky. Druhý modul nese jméno Time Tracking a má na starosti sledování času. Díky tomuto modulu je možné jednoduše zjistit efektivitu práce jednotlivců i týmů na projektech a jiných úkolech. Posledním modelem je Billing, který funkčně navazuje na modul předchozí. Ten umí na základě zaznamenaného času stráveného prací vytvořit vyúčtování. Aplikace je plně přeložena do sedmi jazyků včetně češtiny a pro každý jazyk zahrnuje i zákaznickou podporu. [3]

Mezi hlavní výhody aplikace patří především fakt, že jde o cloudový software, a tedy je dostupná vždy a kdekoliv, kde je internetové připojení. PrimaERP lze využívat i na mobilních zařízeních s nativní aplikací pro systém Android OS. Tato nativní aplikace sbírá data i bez aktuálního připojení k internetu a data automaticky synchronizuje ve chvíli, když je připojení dostupné. Další výhodou je velmi snadná integrace aplikace s externími programy či systémy přes API. [3]

Nevýhodou může být při dlouhodobějším využívání poměrně vysoké zpoplatnění, jež se menším firmám nemusí vyplatit. Platí se podle rozsahu využívání aplikace a to vždy za určité období, které si lze zvolit. Nelze si tedy zakoupit licenci za jednorázový poplatek.

Aplikace PrimaERP není příliš vhodná pro osobní použití a je možné tvrdit, že její volbou by takovýto uživatel spíše čas ztrácel, než-li nad ním získával kontrolu. Je určena pro využití většími firmami, pro které nabízí velké množství speciálních funkcí. Aplikace může být pro firmy velmi přínosná.



Obrázek 2: Ukázka prostředí aplikace PrimaERP Time Tracking. [3]

2.1.3 Paymo

Paymo je softwarová společnost původem z Rumunska. Její největším projektem je aplikace Paymo Application, kterou využívá více než 150000 uživatelů z celého světa. [4]

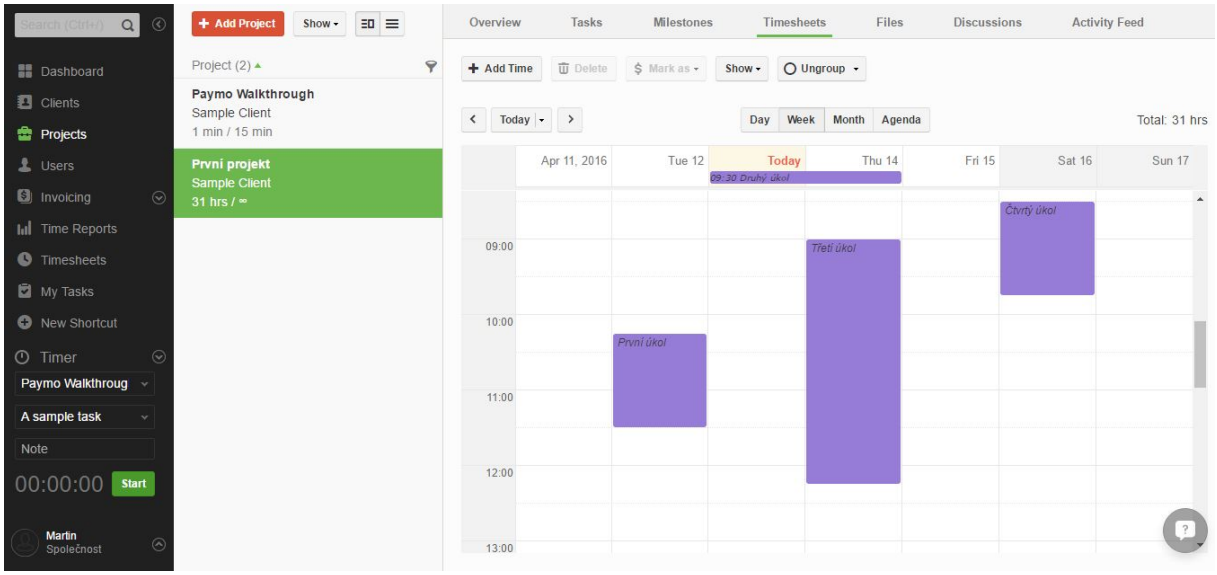
Produkt se zaměřuje na firemní využití, kdy primárně spravuje projekty a sleduje čas strávený jednotlivými úkoly. Kromě nástrojů pro řízení projektů a seldování času lze v aplikaci využívat také další funkce, jako je například modul řešící účetnictví. Využívání aplikace je zpoplatněno měsíčními poplatky. [4]

K hlavním výhodám aplikace lze řadit její dostupnost díky online účtu. Můžeme ji využívat s pomocí webového prohlížeče, počítače s nainstalovaným příslušným softwarem, ale také

lze využít nativní aplikace pro mobilní zařízení s operačním systémem Android OS nebo iOS. Všechny jmenované způsoby je možné kombinovat a využívat i bez internetového připojení, přičemž aplikace provede automaticky synchronizaci dat s online účtem později v době, kdy bude internetové připojení opět dostupné. [4]

Nevýhodou aplikace je zpoplatnění jejího využívání. Nevýhodou může být také chybějící lokalizace celého prostředí do některých jazyků včetně češtiny. [4]

Aplikace představuje moderní správu projektů, časového rozvrhu a vyúčtování. Svým potenciálním uživatelům slibuje úsporu času a celkové zlepšení ziskovosti, avšak většina funkcí byla navrhnuta pro malé a střední firmy, tedy tento produkt pro velké firmy není nejvhodnější volbou.



Obrázek 3: Ukázka prostředí aplikace Paymo. [4]

2.1.4 SprávceČasu.cz

Projekt Správce času je poměrně mladým počinem. Ačkoliv do současné doby nedosahuje takového využití jako již zmíněné nástroje, měl by být zmíněn. Hlavně protože jde o ryze český a ve své podstatě zajímavý projekt, který se navíc zaměřuje, na rozdíl od naprosté většiny aplikací na trhu, na správu času osobního. Autorem celého systému je David Kořínek. [5]

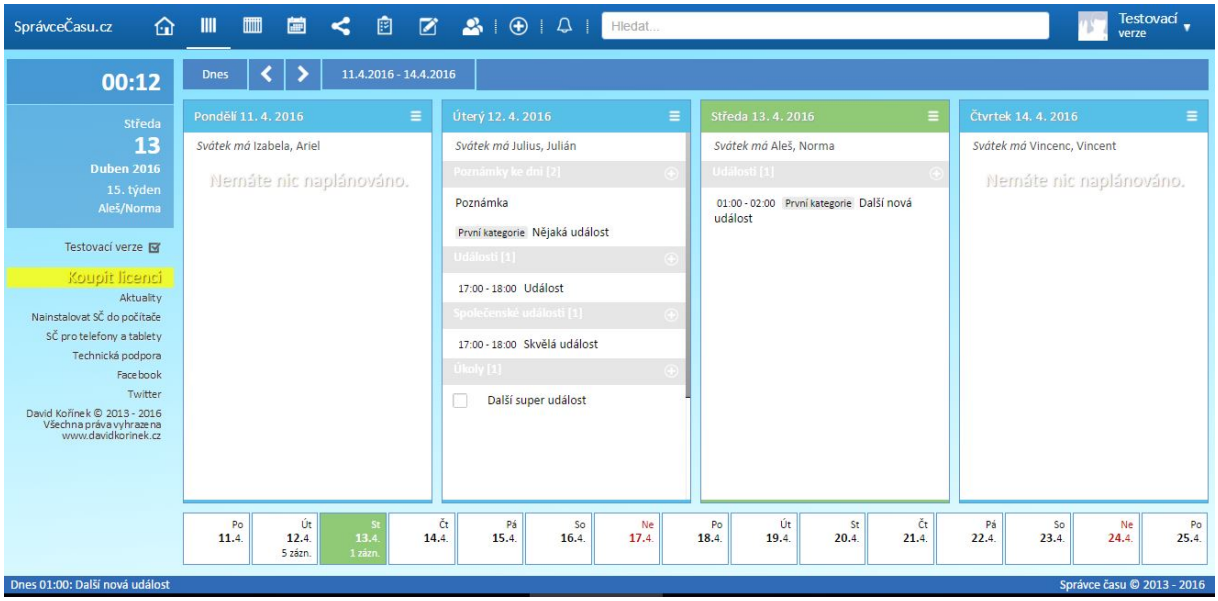
Jak již bylo zmíněno, tento nástroj je určen pro plánování osobního času, tedy uživatelské rozhraní i funkce nejsou zaměřeny na využití ve firmách. Jinými slovy lze vynechat spoustu funkcí, které běžný uživatel nebude potřebovat. Nicméně i přes tento fakt působí celý design nepřehledně a je zbytečně přesycen barvami. [5]

Co se funkcí týče, nabízí se jich poměrně dost. Kalendář umí zobrazit nejen základní události, ale také třeba schůzky, akce, poznámky nebo úkoly. U všech těchto možností lze nastavit důležitost a uživatelem definované štítky, jež slouží poté pro snadnější orientaci v kalendáři. Další vybranou funkcí je možnost sdílet vlastní nebo sledovat ostatní kalendáře, čímž dochází

k automatickému odběru událostí. Samotný uživatel si z tohoto důvodu může nastavit úrovně soukromí pro každou událost zvlášť a některé tak pro ostatní zneviditelnit. Projekt lze využívat z velké části zdarma. Zakoupením licence může uživatel v menší míře vylepšit funkčnost například možností psát delší poznámky k naplánovaným událostem, nicméně zpoplatnění by bylo správnější chápat spíše jako dobrovolnou podporu autora. [5]

Výhodou oproti jiným podobným aplikacím je již samotné konkrétní zaměření na osobní čas, jelikož takto zaměřených aplikací je značně méně než těch, které se zabývají správnou času pracovního v rámci firem. Další výhodou je možnost využívat aplikaci bezplatně. Jako přístup k aplikaci je primárně určen webový prohlížeč, ale je dostupná také jako nativní aplikace pro systém Windows a Android OS. Je však vždy vyžadováno neustálé připojení k internetu, což je oproti jiným aplikacím nevýhoda. Jistou nevýhodou může být absence specifičtějších funkcí. Tyto funkce v aplikaci chybí, z části určitě také proto, že nebyla vyvíjena větší firmou, nýbrž jednotlivcem.

Tato aplikace je vhodná pro běžného uživatele, který nehledá žádné speciální funkce. Umožňuje jednoduchou správu času, která však bude dostačující pro většinu uživatelů.



Obrázek 4: Ukázka prostředí aplikace Správce času. [5]

3 Návrh řešení

Před započetím samotné realizace práce je nutné přesněji specifikovat požadované vlastnosti a funkcionalitu, kterou by výsledná aplikace měla poskytovat. Dále je nutné, s ohledem na specifikované požadavky, navrhnout ucelené řešení zahrnující také návrh databázového modelu.

3.1 Požadovaná funkcionalita

Požadované funkce kladené na výslednou aplikaci vychází ze zadání práce. Představeny budou pouze stěžejní funkce, jež tvoří páteř funkcionality celé aplikace.

Aplikace bude umožňovat registraci, kterou dojde k vytvoření soukromého uživatelského profilu. Aby se zamezilo fiktivním registracím, bude nutné před přihlášením registraci nejdříve potvrdit a až poté bude uživatelský účet aktivní. Neregistrovaní uživatelé nebudou moci aplikaci využívat.

Funkce kalendář bude klíčovou v rámci aplikace. Uživatelům umožní pohodlně a intuitivně spravovat své události. Při práci s kalendářem bude možné jeho zobrazení přepínat v několika režimech, aby se docílilo co největší přehlednosti a přizpůsobitelnosti při práci, která může být různorodá. Výpis bude barevně odlišovat uživatelem definovanou kategorizaci jednotlivých událostí. Po vybrání události z výpisu se uživateli zobrazí detailní informace o dané události.

Uživatelé si také budou moci nechat vygenerovat přehledný časový harmonogram, který na základě zadaných kritérií zobrazí data v tabulkovém formátu. Dále bude umožněno filtrování těchto dat.

Další funkcí bude možnost kategorizace událostí. Kategorie budou tvořit základní třídicí mechanismus událostí, které uživatel vytvoří. Správa kategorií by měla umožnit základní operace nad kategoriemi, jako je jejich zobrazení, vytváření, editace a odstraňování. Aplikace nabídne kromě názvu i volbu barvy, jež bude kategorii následně reprezentovat v kalendáři i statistikách.

Uživatelé si jako doplňující třídicí nástroj ke kategoriím budou moct spravovat také své stavy pro události, které následně mohou vytvářeným událostem přiřazovat.

Dále si uživatelé budou moct nechat vygenerovat a zobrazit různé statistiky odrážející jejich aktivitu v aplikaci. Umožněno bude pracovat se základními souhrnnými statistickými daty, která zahrnou celkové využívání aplikace uživatelem. Dále bude možné nechat zobrazit data reprezentována v podobě přehledných grafů, a to v rámci různých časových období. Tato období budou nejen striktně daná, ale uživatel si bude moct sám nadefinovat období, ze kterého chce statistiku vytvořit.

S ohledem na analýzou zjištěné informace, týkající se existujících podobných řešení celé aplikace, bylo vyvozeno, že nejrozšířenější a zároveň nejpodobnější aplikací je Google kalendář. Také proto bude umožněno importování všech událostí z účtu u aplikace Google kalendář prostřednictvím přihlášení daného uživatele k aplikaci. Tato funkce zajistí nenásilný přechod uživatelů z aplikace Google kalendář na vyvíjenou aplikaci.

Funkce upozorňování na začátek události bude uživatelům umožňovat volbu mezi vytvořením události bez nastaveného upozornění a událostí s nastaveným upozorněním prostřednictvím e-mailové zprávy. V případě nastavení této funkce u nové události bude uživatel určitou dobu před začátkem události informován zprávou na e-mailovou adresu, jež při registraci zadal a ověřil.

Aplikace nabídne uživatelům i funkci oddělenou od funkčnosti kolem událostí. Uživatel bude moct spravovat své poznámky, které se nevztahují k dané události jako ty, které se vkládají při vytváření nové události a jež u nich plní funkci popisu. Ve výpisu těchto poznámek bude umožněno vyhledávat podle názvu a provádět hromadné operace nad výběrem, například odstranění několika vybraných poznámek najednou. Vkládání poznámek uživatelům zjednoduší WYSIWYG editor textu poznámky.

Uživatelský profil bude z části editovatelný. Konkrétně bude uživatelům umožněna změna hesla a e-mailové adresy. Ostatní údaje budou zobrazeny, avšak změnit je již nebude možné. Tato funkcionalita bude spadat pod nastavení profilu.

3.1.1 Use Case diagram

K funkční specifikaci softwarových systémů patří také vytvoření Use Case diagramu. Tento diagram zachycuje s jistou mírou abstrakce scénáře, které mohou být v aplikaci spuštěny.

Use Case diagram je v nejjednodušší formě reprezentací interakcí mezi uživatelem systému a systémem samotným, které rozumí vývojáři i zákazníci. Ukazuje vztah mezi uživatelem a různými případy užití, jež uživatel může vyvolávat. Use Case diagram může zobrazovat různé typy uživatelů systému. Tento diagram bývá často vytvářen spolu s dalšími typy diagramů. [6]

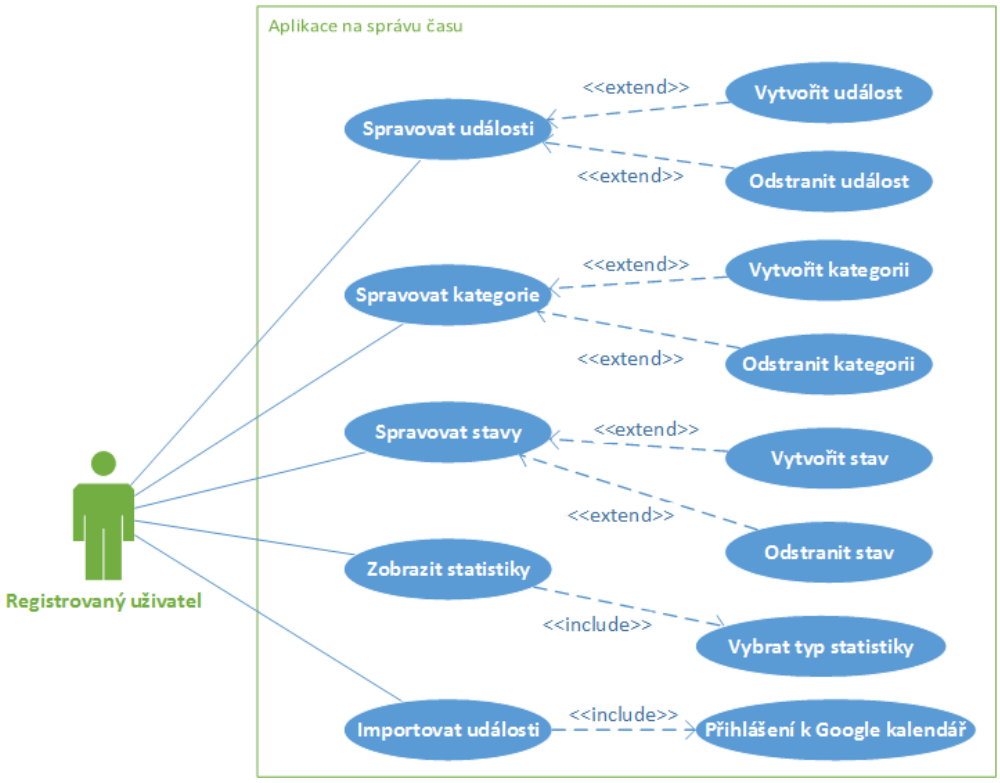
3.1.1.1 Aktéři vyskytující se v systému

- Neregistrovaný uživatel
  - Jde o uživatele, který se zatím do aplikace neregistroval a nemá tak možnost využívat všech jejích funkcionalit.
- Registrovaný uživatel
  - Jde o uživatele, který se již do aplikace registroval a má možnost ji plně využívat.
  - Může vyvolat stejné případy užití jako neregistrovaný uživatel a k těmto přidává další.

3.1.1.2 Vybrané skupiny Use Cases

- Registrace uživatele. Aktérem je neregistrovaný uživatel.
- Správa událostí, která obsahuje scénáře práce s událostmi. Aktérem je registrovaný uživatel a mezi scénáře patří například vytvoření události, její editace, odstranění.

- Správa kategorií, která obsahuje scénáře pro práci s kategoriemi. Aktérem je registrovaný uživatel a mezi scénáře patří například vytvoření kategorie nebo její odstranění.
- Správa stavů, která obsahuje scénáře práce se stavy. Aktérem je registrovaný uživatel a mezi scénáře patří například vytvoření stavu nebo jeho odstranění.



Obrázek 5: Zjednodušený Use Case diagram: Práce s aplikací v sekci správy času.

3.2 Požadavky uživatelského rozhraní a designu

Grafické zpracování by mělo být co možná nejjednodušší, aby se zachovala přehlednost. Barevný profil aplikace bude vybrán v kontrastních barvách zajišťujících dobrou čitelnost. V aplikaci bude převládat modrá barva.

Rozložení prvků na stránce bude především uživatelsky intuitivní. Taktéž všechny formuláře a okna, jež budou v aplikaci použita, se budou řídit pravidly správného návrhu uživatelských rozhraní.

3.3 Návrh architektury aplikace

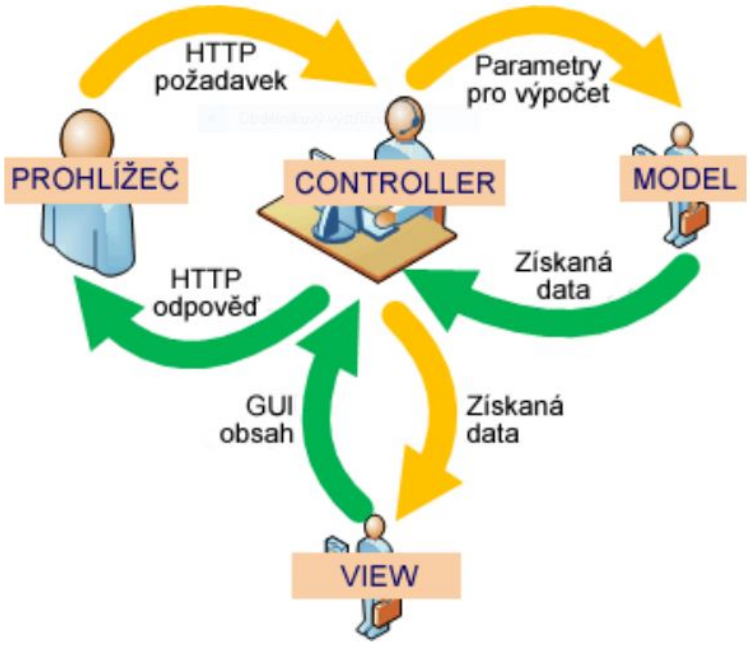
Aplikace má být dostupná v prostředí internetu. Úkolem návrhu architektury aplikace je zvolit správnou webovou architekturu a najít takové řešení, které zabezpečí případné budoucí změny aplikace, ať už v podobě přechodu na jiný databázový systém nebo změny v uživatelském rozhraní aplikace.

Architektura aplikace bude postavena na návrhovém vzoru MVC, který je při vývoji právě webových aplikací velmi často využívaný. Díky této architektuře docílíme rozdělení systému v zásadě do tří na sobě aplikačně nezávislých částí a výrazně tímto usnadníme správu i budoucí možné úpravy aplikace.

3.3.1 MVC architektura

Tato architektura řeší problém kódu, který v rámci stejného místa mísí logiku aplikace i svůj výstup uživatelům, čímž znemožňuje snadnou údržbu a rozšiřitelnost aplikací. MVC architektura prakticky rozděluje aplikaci na komponenty třech typů. [7]

- Model (Model) - Model spravuje data a logiku aplikace. Tato část aplikace nic neví o pohledech ani řadičích.
- View (Pohled) - Pohled má na starosti výstup aplikace a obsahuje jen nezbytné minimum logiky, která je potřebná pro správnou reprezentaci výstupu. Uživatel prostřednictvím pohledu interaguje s aplikací.
- Controller (Řadič) - Řadič je, zjednodušeně řečeno, spojovník mezi modelem a pohledem. Reaguje na uživatelem vyvolané události a zajišťuje patřičné změny v modelu i pohledu.



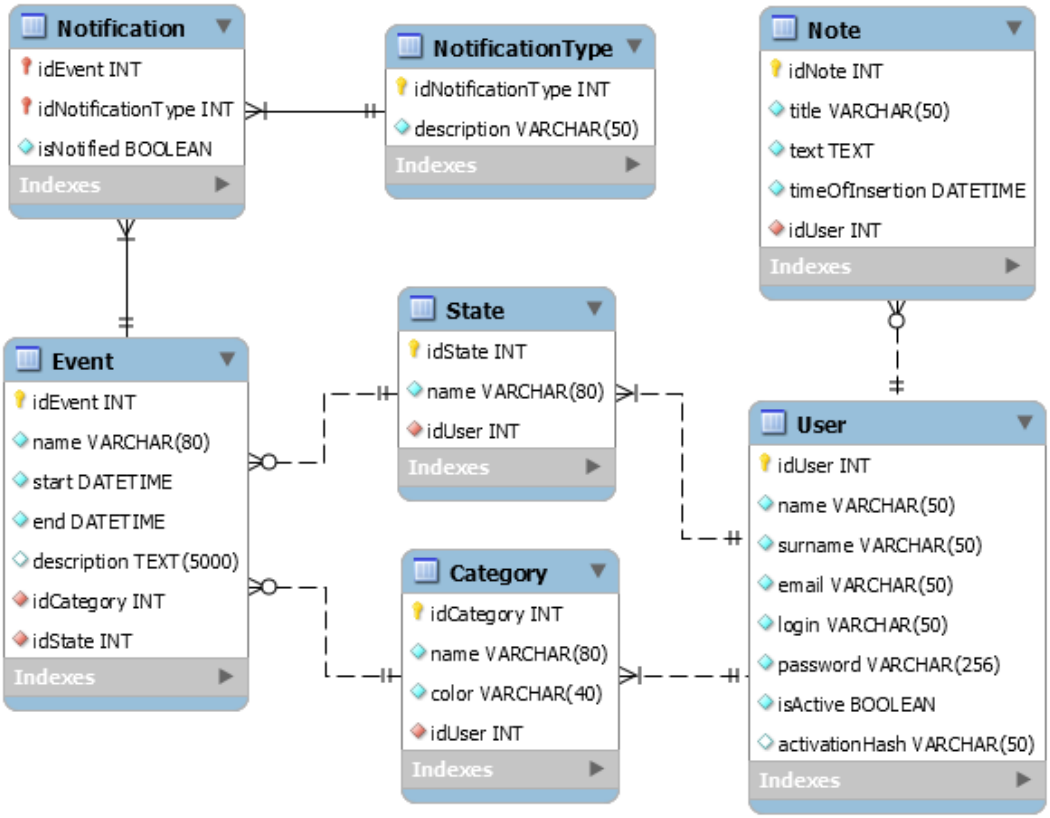
Obrázek 6: Schéma návrhového vzoru MVC při využití ve webových aplikacích. [8]

3.3.2 SPA

V dnešní době je pro webové aplikace stále více moderní vyvíjet je jako SPA. Také proto byl tento koncept využit pro nově vyvíjenou aplikaci. Myšlenkou SPA je zvýšit komfort uživatele při prohlížení webové aplikace a přiblížit se tak například uživatelskému komfortu aplikací pro desktop. Ve své podstatě jsou SPA aplikace navrženy tak, aby byl veškerý potřebný kód ze serveru stažen pouze jednou při prvním načtení. Všechny další interakce uživatele s aplikací jsou řešeny asynchronně klientským skriptovacím jazykem. SPA aplikace, pokud je vše správně, stránku nikdy neobnovuje. Obnovení je však stále možné, pokud jej uživatel provede ručně sám. [9]

3.4 Návrh databázového modelu

Většina aplikací ke svému provozu potřebuje trvalé úložiště svých dat. Existuje mnoho databázových systémů, které se mezi sebou často velice liší, například způsobem reprezentace dat nebo svými výkonnostními limity. Pro vyvíjenou aplikaci je vhodné navrhnout koncept modelu databáze takový, jež nebude závislý na konkrétně zvoleném systému řízení báze dat, přestože otázka správného zvolení databázového systému spolu s návrhem modelu je pro správně a efektivně fungující aplikaci klíčové.



Obrázek 7: Návrh modelu databáze.

3.4.1 Návrh komunikace datové vrstvy a domény aplikace

Spolu s návrhem databázového modelu je žádoucí navrhnout také způsob komunikace datové vrstvy s doménou aplikace. Námi volené databázové systémy jsou relační a vyvíjená aplikace objektově orientovaná, tedy data jsou v trvalém úložišti uložena jiným způsobem než dočasně v aplikaci. Také některé z vlastností, jež najdeme v objektově orientované aplikaci, v relačních databázích nenajdeme. Jde například o dědičnost nebo možnost ukládat kolekce. Kvůli těmto faktům je nutné implementovat určité ORM.

Nejvhodnějším řešením je využití některého z návrhových vzorů, které patří do skupiny, která řeší architekturu mezi zdrojem dat a aplikací. Do této skupiny mimo jiné řadíme návrhové vzory Active Record, Row Data Gateway, Table Data Gateway nebo Data Mapper. Při návrhu aplikace bylo nakonec rozhodnuto pro implementaci návrhového vzoru Data Mapper.

4 Programátorská dokumentace

Implementaci celé webové aplikace lze rozdělit na dvě základní části. První část je klientská, které se někdy říká frontend. Tato část je zobrazována prohlížečem, a proto jsou pro ni využity technologie, které jsou pro prohlížeče standardizované a rozumí jim. Konkrétně byl využit jazyk HTML, kterým byl strukturován obsah jednotlivých stránek. Pro definici vzhledu stránek byl použit jazyk CSS. Posledním vybraným jazykem v klientské části aplikace, který byl použit pro dodání funkčnosti, je jazyk JavaScript. Jelikož webová aplikace byla navržena jako SPA, na straně klienta se odehrává velká část funkčnosti celé aplikace. Proto bylo využito moderního JavaScriptového MVC frameworku od společnosti Google, který se nazývá AngularJS.

Druhá část aplikace je serverová, která se někdy nazývá backend. Tato část není uživatelům aplikace viditelná a běží na serveru. Pro implementaci funkcionality byl pro tuto část vybrán jazyk PHP ve verzi 5, která již ve velké míře umožňuje OOP.

Dalším technologickým rozhodnutím bylo určení technologií pro komunikaci mezi klientskou a serverovou částí aplikace, jelikož je navrhnuta jako SPA. Využívá se technologie AJAX, díky které lze asynchronně zasílat od klienta na server HTTP požadavky a zpracovávat odpovědi. To vše na pozadí aplikace, aniž by došlo k znovunačtení webu. Zasílaná data jsou v aplikaci reprezentována ve formátu JSON, který je klientským i serverovým skriptovacím jazykem přímo podporován.

Otázka databáze a zvolení vhodného DBMS byla nastíněna již v návrhu aplikace. Nároky na databázi nebyly příliš vysoké, požadovaná totiž byla základní funkcionalita i rychlost. Otázku pomohly vyřešit mimo jiné zvolené technologie, které jsou s některými konkrétními DBMS velmi dobře provázány. Rozhodovaly ovšem i licenční náklady, které musely být nulové. Nakonec byl zvolen databázový systém MySQL, který všechna stanovená kritéria splňuje.

4.1 Architektura aplikace

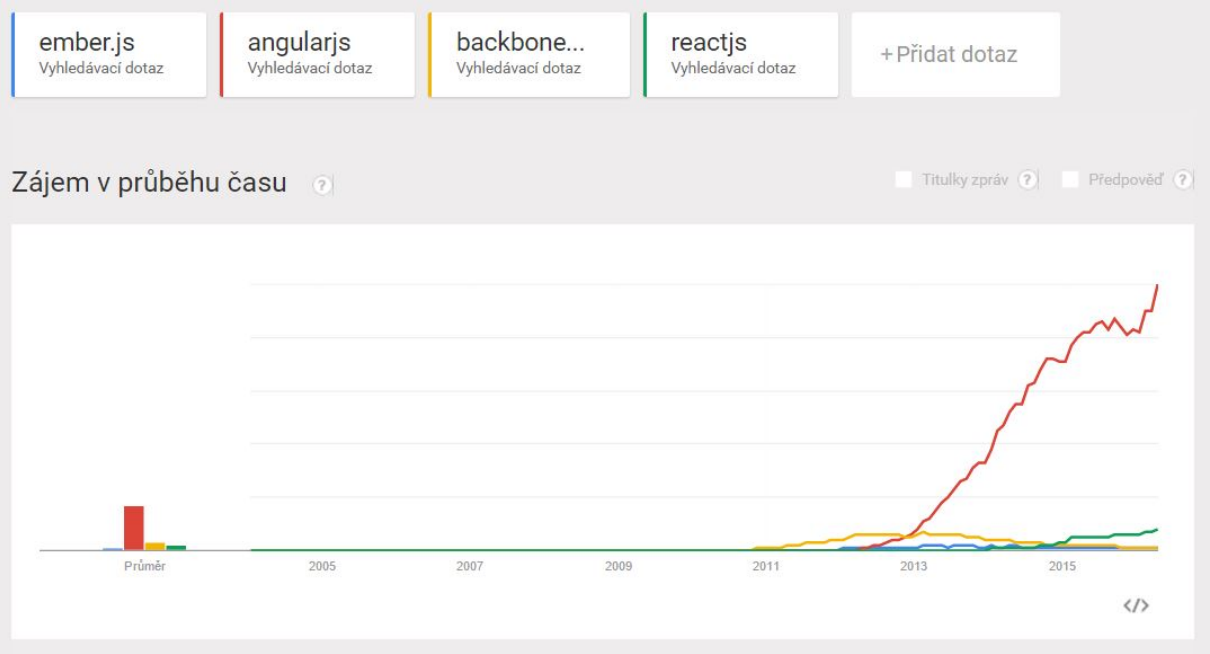
Vyvíjená aplikace je webovou aplikací, která využívá architekturu klient-server. Byla implementována jako SPA aplikace. Pro klientskou část je využit architektonický návrhový vzor MVC.

4.2 Vybrané použité technologie

V aplikaci bylo nutné zvolit a využít hned několik technologií. V této podkapitole budou použité technologie, které nejsou obecně tolik obvyklé, stručně představeny.

4.2.1 AngularJS

Trendy ve vývoji webových aplikací se poměrně výrazně začaly měnit s nástupem HTML5 technologií. Stále větší oblibě se těší jazyk JavaScript, pomocí kterého se, s využitím patřičného frameworku, již začíná programovat dokonce i serverová část aplikací. [10]

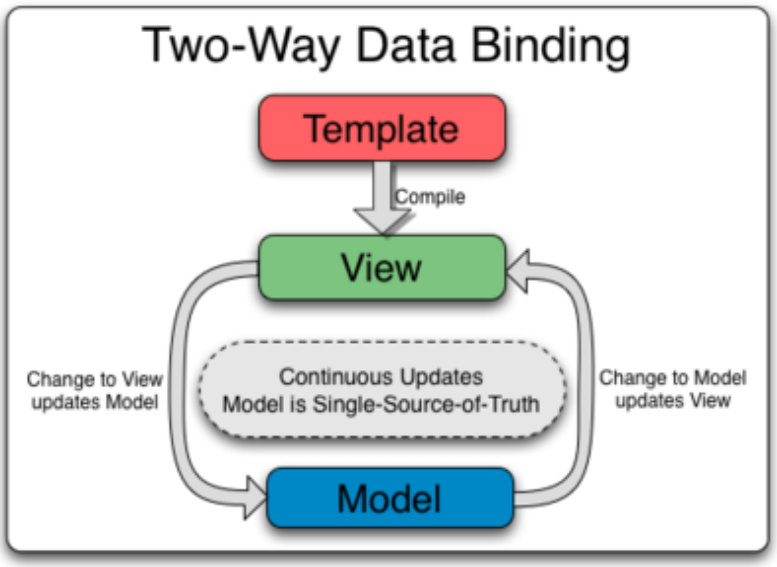


Obrázek 8: Zájem o AngularJS framework v kontrastu s jinými současnými frameworky. Data získána ze služby Google Trends. [11]

Pro implementaci funkčnosti na straně klienta ve vyvíjené aplikaci byl využit v poslední době velice oblíbený JavaScriptový MVC framework od společnosti Google, za jehož vznikem stojí Miško Hevery. [12]

AngularJS je užitečný zejména pro komplexnější aplikace, u kterých se díky jeho MVC architektuře a možnosti rozdělovat kód do modulů stále tvoří udržovatelný kód. Je také velmi vhodný pro tvorbu webových aplikací typu SPA, protože framework svými myšlenkami tento koncept plně podporuje. [12]

Pravděpodobně nejužitečnější vlastností AngularJS frameworku je vlastnost zvaná two-way data binding, což lze do češtiny přeložit jako dvoucestná datová synchronizace. Zjednodušeně řečeno jde o automatický způsob aktualizace výstupu, tedy pohledu, když dojde ke změně dat v modelu aplikace a naopak, dojde-li ke změně dat uživatelem v pohledu aplikace, automaticky se aktualizují data modelu. Tato vlastnost tak osvobozuje od nutnosti manipulovat s DOM webu nebo registrace tzv. callback funkcí, jež by se po daných změnách sami volaly. [12]



Obrázek 9: Schéma funkce dvoucestné datové synchronizace, která je mimo jiné využita i v AngularJS. [13]

Velmi užitečnou vlastností je také zabudovaná podpora návrhového vzoru Dependency Injection. Dovoluje deklarativně zapsat, jak jsou jednotlivé komponenty aplikace propojeny a jaké jsou mezi nimi závislosti. [12]

Právě zmíněná zabudovaná podpora Dependency Injection napomáhá snadnému testování aplikací vytvořených za pomoci frameworku AngularJS. Framework byl od počátku navržen tak, aby jednoduché testování umožňoval a výhodně k tomu využíval právě i implementovaný Dependency Injection. [12]

AngularJS přichází s řadou nových značek a rozšiřuje tak jazyk HTML. Přichází však také s možností vytvořit vlastní elementy nebo atributy k jazyku HTML, které budou specifické pro danou aplikaci. V názvosloví AngularJS se jim říká direktivi. Lze díky nim vytvářet znovupoužitelné komponenty a přitom skrývat jejich mnohdy složitou vnitřní strukturu, definovaný vzhled i funkčnost. Užitečné jsou také při lokalizaci aplikace do různých jazyků. [12]

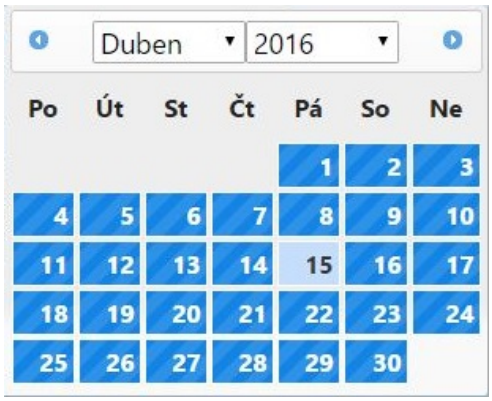
Framework také usnadňuje i běžnější záležitosti, jako je třeba validace vstupu, tedy formulářů. Jedná se o validaci zadaných dat na straně klienta v prohlížeči, takže je nutná druhotná validace dat také na straně serveru. AngularJS umožňuje jednoduše definovat pravidla, přičemž není nutné psát žádný validační kód v JavaScriptu. [12]

4.2.2 Použité knihovny a komponenty třetích stran

Při implementaci vyvíjené aplikace bylo využito několik technologií, knihoven a komponent třetích stran, které byly do aplikace integrovány. Komponenty představují ověřené řešení určité specializované problematiky a usnadnily vývoj příslušných částí aplikace. Tato sekce se zaměří na ty nejdůležitější.

Pro animace bez využití AngularJS frameworku, tedy na úvodní stránce aplikace před přihlášením uživatele, byla využita knihovna jQuery. [14] Využito bylo také jQuery UI, což je sada prvků uživatelského rozhraní, která je s knihovnou jQuery implementována. Uplatnění ve vyvíjené aplikaci našla kvůli potřebě malého rozbalovacího kalendáře, přes který lze pohodlně zvolit datum namísto jeho vepisování do vstupních polí. V rámci jQuery UI se tato komponenta nazývá datepicker. [15] Komponenta CKEditor představuje WYSIWYG editor a byla využita při vkládání nových poznámek jako uživatelsky přívětivý editor textu poznámky. [16]

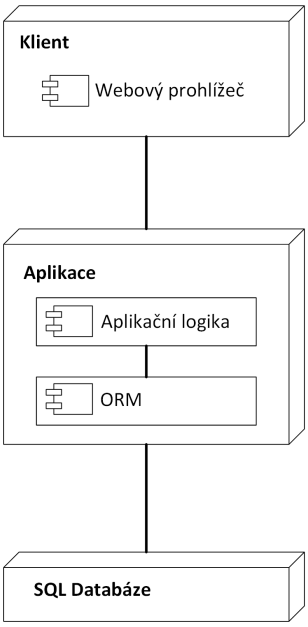
Dále bylo využito několik modulů frameworku AngularJS. Pro usnadnění práce s modálními okny to byl například modul ngDialog. [17] Pro možnost tzv. routování, tedy práce s URL adresou a tvorbu jejich přehlednějších forem, byl využit modul angular-route. [18] Animace v rámci frameworku byly řešeny za pomocí modulu angular-animate. [22] Stránkování jakéhokoliv pole dat dokáže řešit modul dirPagination. Tento modul byl využit pro vyřešení stránkování výpisu seznamu poznámek. [20] Funkcionalitu statistik, přesněji vykreslení patřičných grafů, pomohl vyřešit modul angular-charts. [21]



Obrázek 10: Ukázka komponenty datepicker, která byla využita pro usnadnění vyplňování vstupních polí typu datum. [15]

4.3 Diagram nasazení

Uživatel přistupuje k aplikaci, která je umístěna na serveru, přes internet pomocí webové prohlížeče. Aplikace pak komunikuje s SQL databází prostřednictvím ORM, se kterým uvnitř aplikace komunikuje aplikační logika. Znázornění diagramu nasazení je na obrázku 11.



Obrázek 11: Ukázka komponenty datepicker, která byla využita pro usnadnění vyplňování vstupních polí typu datum.

4.4 Routování

Jelikož je aplikace implementována jako SPA, neobnovuje se ani při změně URL adresy. V aplikace bylo provedeno tzv. routování. V AngularJS frameworku se routování nastavuje pomocí služby `$routeProvider`. Každé nastavené routě můžeme přidělit jiný controller. Nastavení routování v aplikaci osvětlí výpis 1, na kterém je vidět zdrojový kód použitý pro routování ve vyvíjené aplikaci.

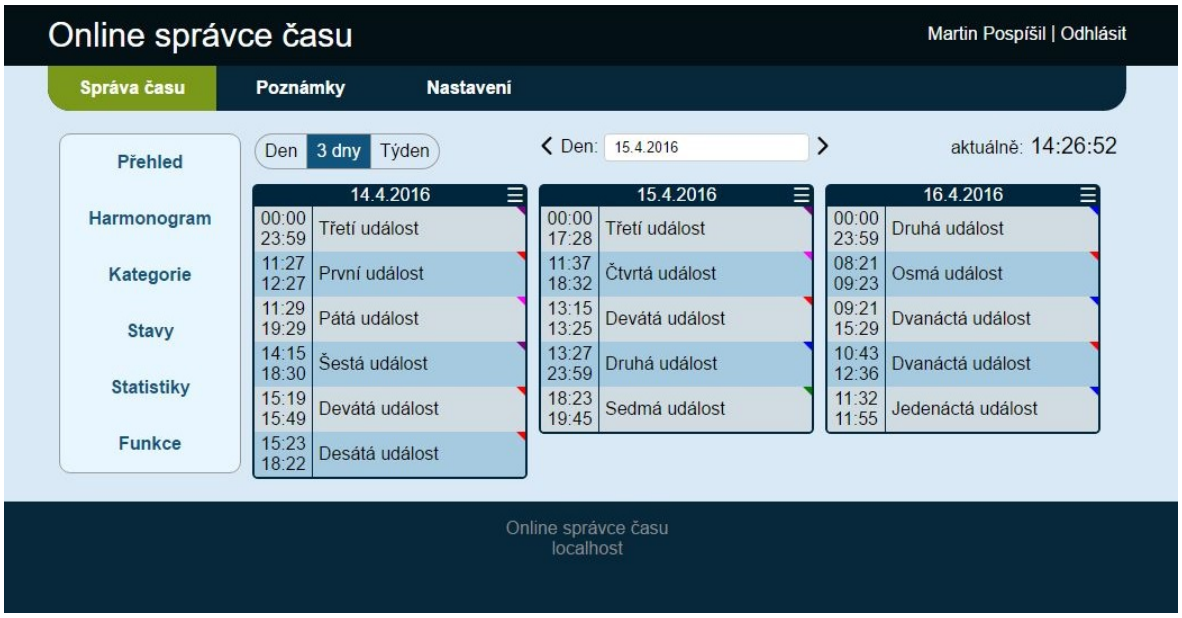
```
myApp.config(function($routeProvider, ngDialogProvider) {
   $routeProvider.
      when('/timeManagement', {
         templateUrl: 'partial/admContentTimeManagement.php',
         controller: 'TimeManagementController',
         activeMenuItem: 'timeManagement'
      }).
      when('/notes', {
         templateUrl: 'partial/admContentNotes.php',
         controller: 'NotesController',
         activeMenuItem: 'notes'
      }).
      when('/settings', {
         templateUrl: 'partial/admContentSettings.php',
         controller: 'SettingsController',
```

```
        activeMenuItem: 'settings'
    }).
    otherwise({
        redirectTo: '/timeManagement'
    });
});
```

Výpis 1: Ukázka řešení routování ve vyvíjené aplikaci.

4.5 Uživatelské rozhraní

Na základě deklarovaného návrhu bylo implementováno uživatelské rozhraní. Implementace byla realizována s ohledem na plánovaný budoucí rozvoj aplikace, který bude zahrnovat vývoj nativní aplikace pro systém AndroidOS. Tento systém lze nejčastěji nalézt u mobilních zařízení s obvykle menším rozlišením obrazovky. Samotná implementace webové aplikace tak mohla být plně soustředěna pro zobrazení na monitorech. Ideální zobrazovací zařízení by mělo ve své šířce zobrazovat 1024 a nebo více pixelů.



Obrázek 12: Ukázka hlavní stránky aplikace po přihlášení uživatelem.

Webové rozhraní aplikace po přihlášení uživatele je intuitivně rozděleno na statickou část v malém horizontálním pruhu a dynamicky se měnící část, která se nachází ve zbytku prostoru stránky pod částí statickou. Statická část obsahuje odkazy směřující na úvodní stránku aplikace, do uživatelské části a odkaz sloužící k odhlášení uživatele. Dále obsahuje horizontálně orientované hlavní menu, které přestavuje jednotlivé logické celky aplikace, přes které uživatel ovládá dynamický obsah pod tímto menu. Aktuálně zvolená možnost je v pak v aplikaci přehledně

zviditelněna podbarvením v dostatečném kontrastu ke zbytku menu. Dynamická část obsahuje vždy patřičné podmenu k právě zvolené položce a pak samotný obsah.

4.5.1 Kalendář

Nejdůležitější částí implementace v rámci uživatelského rozhraní byla implementace kalendáře. Kvůli zvýšení přehlednosti bylo upuštěno od tradičnějšího tabulkového zobrazení, přičemž místo něj byl implementován jako sloupcový. Kalendář pro zajištění ještě většího uživatelského komfortu lze přepínat v různých zobrazeních. Navigace v kalendáři je pak umožněna dvojím způsobem. Lze zvolit konkrétní datum z malého kalendáře, jež se zobrazí po kliknutí do pole s aktuálně vybraným datem, ale také se lze s pomocí přináležících šipek navigovat kalendářem o jednotlivé dny vpřed i vzad. Uživatel se z výpisu, aniž by prováděl jakoukoliv další akci, dozví v rámci zobrazovaných dnů, jaké události jsou pro něj naplánovány, jejich časové vymezení i pro událost zvolenou kategorii, respektive barvu kategorie jako malý trojúhelníček v rohu řádku s danou událostí. Pro získání detailnějších informací o události je potřebné událost zvolit prostřednictvím jejího názvu. Následně je zobrazeno modální okno s podrobnějšími informacemi a také prvky obstarávající operace nad danou událostí. Tyto operace jsou úprava a odstranění události. U kalendáře je zobrazen i aktuální přesný čas.

Obrázek 13: Ukázka modálního okna s detailnějšími informacemi o události.

4.5.2 Poznámky

Zajímavé prvky uživatelského rozhraní najdeme také v implementaci části aplikace, která má na starost správu poznámek. Již z návrhu bylo dáno, že výpis poznámek včetně jejích editačních prvků, s ohledem na maximální přehlednost, bude tabulkového formátu. V poznámkách lze jednoduše vyhledávat za pomoci vyhledávacího pole nad výpisem poznámek. Toto pole filtruje výpis poznámek v reálném čase s každou změnou řetězce, podle něhož se vyhledává. Dále má

každá poznámka ve výpisu na začátku svého řádku zaškrtávací políčko. Využití těchto políček je při aplikaci hromadných operací, jež uživateli například umožní odstranit několik vybraných poznámek najednou. Ovládací prvky pro hromadné operace jsou umístěny pod výpisem na levé straně.

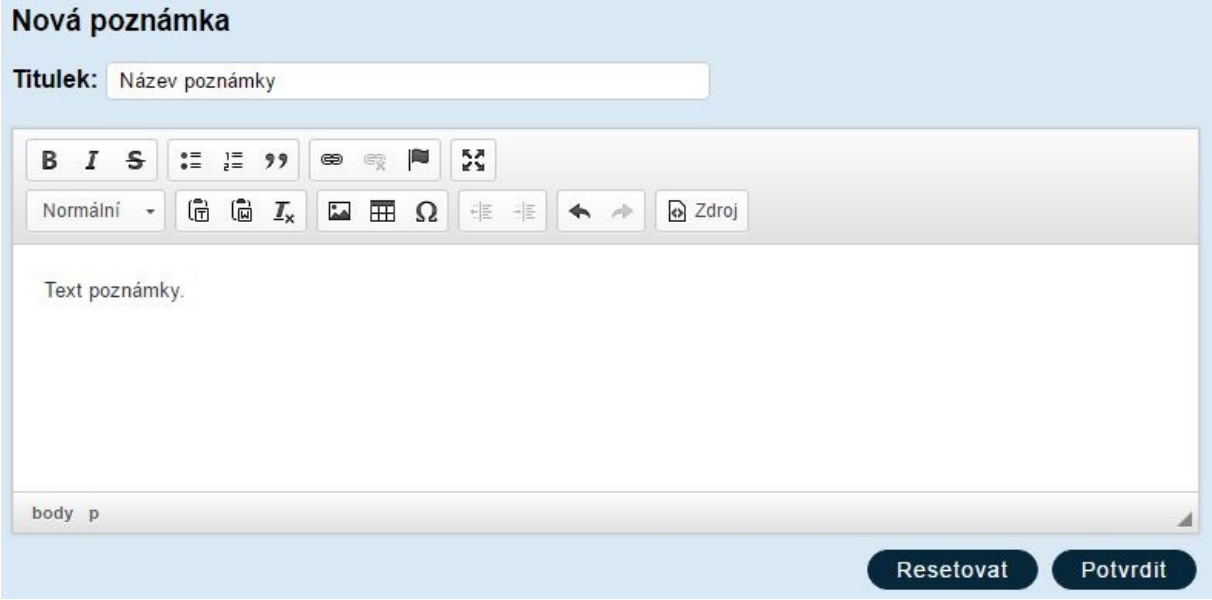
Jelikož počet uživatelem vytvořených poznámek není aplikací nikterak omezen, bylo třeba řešit, jak nejvhodněji zobrazit výpis s velkým počtem poznámek. Zřetel byl brán na moderní trendy a opět na intuitivní provedení pro uživatele. Při implementaci byla nad výpisem vpravo vytvořena rozbalovací nabídka, která umožňuje uživateli zvolit počet poznámek, jež se budou na stránce vypisovat. Změna se provádí dynamicky bez nutnosti obnovy celé stránky. V případě, že počet celkově vyhovujících událostí pro výpis je vyšší než počet, jež uživatel v nabídce vybral, v uživatelském rozhraní přibude další prvek, který se obecně nazývá stránkování. Tento prvek umožňuje procházení všech poznámek při zachování stejného počtu současně zobrazených poznámek, jenž si uživatel zvolil.

Obrázek 14: Ukázka správy poznámek.

Velmi vhodným prvkem uživatelského rozhraní, vycházejícího opět z návrhu aplikace, je WYSIWYG editor. WYSIWYG editor se využívá při vkládání nové poznámky jako editor textu.

Hlavní výhoda takových editorů oproti obyčejným vstupním polím je formátování zadaného textu. Uživatel nemusí znát žádný značkovací ani jiný jazyk, kterým se texty webových stránek formátují, a přesto je schopen vytvořit text, který bude naformátován. Uživatel pomocí editoru může navíc do textu, samozřejmě podle nastavení a možností editoru, vkládat také obrázky, odkazy či tvořit tabulky. Podporovány jsou také známé funkce z mnoha desktopových editorů

textu jako třeba tlačítka pro vrácení či opakování poslední akce. Pro vyvíjenou aplikaci byla velmi výhodná i funkce, jež umožňuje editor roztáhnout přes celou obrazovku uživatele, a tím ještě více usnadnit psaní textu poznámky.



Obrázek 15: Ukázka využití WYSIWYG editoru při vkládání nové poznámky.

4.5.3 Implementace formulářů

K nedílné součástí webových aplikací, které interagují s uživatelem, jsou možnosti pro uživatelské vstupy. Ty ve webovém prostředí nejčastěji představují formuláře.

V implementaci bylo nutné zvolit nástroje k validování zadaných dat a také nástroje pro odeslání dat ke zpracování na server. Ve vyvíjené aplikaci byl, jak již bylo zmíněno, k programování na straně klienta v prohlížeči využit javascriptový framework AngularJS. Tento framework umí efektivně řešit oba tyto úkoly.

Bez využití AngularJS by bylo nutné řešit validaci formuláře samostatně. Bylo by potřeba namapovat na událost odeslání formuláře funkci, která by data validovala a vrátila výsledek validace s informací o chybách. Dále by bylo nutné mapovat validační funkce i k dalším událostem jako jsou stisk klávesy uživatelem v určitém vstupním poli, aby bylo možné okamžitě uživatele upozorňovat na data, která neodpovídají předepsanému formátu, a tedy jsou nesprávně zadaná.

AngularJS však validaci formuláře velmi usnadňuje. Přichází s několika speciálními direktivami, které se zapisují do kódu a následně díky nim AngularJS sám vyhodnocuje informace o celém formuláři a stavu vyplnění všech jeho částí. Programátor tedy musí specifikovat pouze pravidla danými direktivami a reagovat na stavy, které AngularJS pro každou část formuláře automaticky přiřazuje. Mezi tyto stavy patří `pristine`, který je zapnutý, pokud daná část formuláře ještě nebyla vůbec aktivována, tedy se k ní uživatel zatím stále nedostal. Tento stav se dá považovat za výchozí. Protikladem k tomuto stavu je stav `dirty`, který bude aktivní od doby,

kdy uživatel pole ve formuláři poprvé využije. Dalším stavem je `touched`, který se aktivuje, když uživatel s danou částí formuláře aktuálně pracuje. Poslední dva stavy jsou `valid` a `invalid`. Tyto stavy jsou si protikladné a dávají programátorovi informaci o správnosti či nesprávnosti dat v příslušné části formuláře. AngularJS také podle stavu, ve kterém se nachází dané prvky formuláře, nastavuje odpovídající třídy. Díky tomuto lze definovat kaskádové styly těmto třídám a určovat tak vzhled formulářových prvků v daných stavech. [23]

Chybové zprávy pro uživatele pak AngularJS řeší další speciální direktivou, konkrétně se tato direktiva nazývá `ng-messages`. [23]

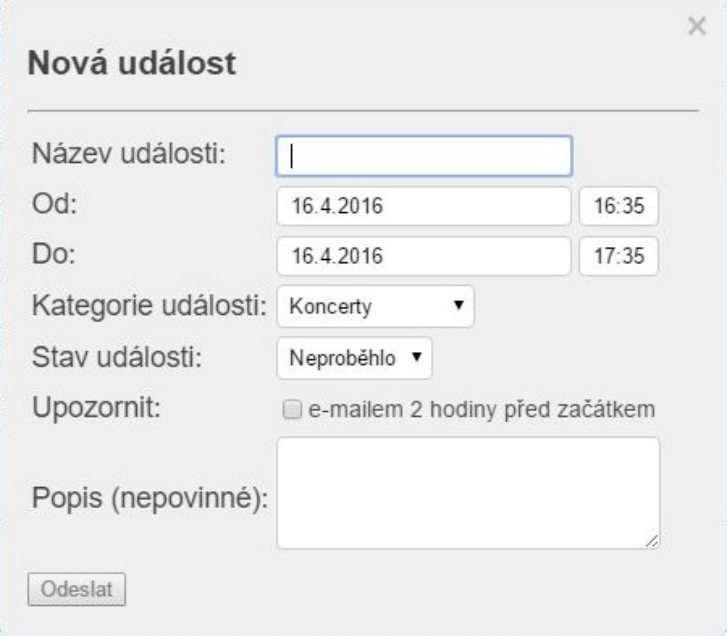
```
<form name="formular">
   <input type="text"
          name="vstup"
          ng-model="data.vstup"
          ng-minlength="5"
          ng-maxlength="10"
          ng-required="required" />
</form>

<div ng-messages="formular.vstup.$error">
   <span ng-message="minlength">Vstup je prilis kratky.</span>
   <span ng-message="maxlength">Vstup je prilis dlouhy.</span>
   <span ng-message="required">Vstup je povinne pole.</span>
</div>
```

Výpis 2: Ukázka validace vstupního pole v AngularJS včetně řešení zobrazení chybových zpráv.

Jelikož je vyvíjená aplikace navrhnuta i implementována jako SPA, je nevyhovující, aby byla webová stránka s formulářem znovu načtena po odeslání dat na server a přijetí odpovědi, což je standardní chování. Je nutné data z formuláře odeslat asynchronně, tedy využít technologie AJAX.

V jádru AngularJS je již zabudována podpora technologie AJAX. Konkrétněji ke komunikaci s vzdáleným serverem slouží služba $http, která dokáže vytvářet požadavky i přijímat ze serveru odpověď. Pro usnadnění práce a zjednodušení zdrojového kódu nabízí AngularJS několik zkracujících metod této služby pro jednotlivé typy požadavků. Tyto metody jsou `get()`, `head()`, `post()`, `put()`, `delete()`, `jsonp()` a `patch()`. Při zpracovávání formulářů v aplikaci byla převážně využívána metoda `post()`, jejímž prostřednictvím lze na server zaslat i data. [24]



Obrázek 16: Ukázka jednoho z formulářů využitých v aplikaci.

4.5.4 Statistiky

Zajímavé řešení uživatelského rozhraní si vyžádala i část vyvíjené aplikace, která uživateli umožňuje zobrazit základní souhrnnou statistiku i generovat statistiky, jež jsou reprezentované grafy, v jím zvoleném časovém období.

Aplikace uživateli nabízí na výběr hned z několika možností, z nichž každá vytvoří jinou statistiku nebo pro jiné období. Uživatel si také může definovat období své. Po výběru z nabídky je statistika aplikací okamžitě vypočítána a zobrazena. Pokud uživatel dodatečně zadává parametry pro výpočet, statistika se opět automaticky překresluje. Toto chování aplikace bylo implementováno z důvodu vytvoření většího komfortu pro uživatele při ovládání statistik.

Při implementaci grafů byl pro vyšší uživatelskou přehlednost brán ohled na barevné odlišení kategorií, jež si pro ně uživatel volí. Při vykreslování tzv. koláčových grafů pro kategorie je vždy daný výsek v grafu vybarven dle příslušné barvy.

4.6 Serverová část

Pro implementaci serverové části byl při návrhu zvolen jazyk PHP. Serverová část je velmi důležitá, protože ačkoliv je aplikace implementována jako SPA, z bezpečnostních důvodů není možné některé funkcionality řešit na straně klienta. Serverová část obstarává logiku celé neveřejné části aplikace, do které je možný přístup i bez autentizace. Dále umožňuje registraci a proces přihlášení včetně zasílání ověřovacích e-mailových zpráv prostřednictvím e-mailového serveru. Na starosti musí mít i funkci notifikování událostí, jejichž začátek se blíží. Také obsluhuje veškeré asynchronní požadavky z SPA aplikace po přihlášení uživatele a komunikuje s databázovou

vrstvou prostřednictvím svých tříd, které jsou k tomuto účelu speciálně vyhrazeny. Tyto třídy tvoří souhrnně ORM.

4.6.1 Vyřešení nezávislosti na datové vrstvě

Již v návrhu aplikace bylo stanoveno, že závislost na konkrétní implementaci datové vrstvy by měla být minimální, nejlépe pak žádná. Pro splnění tohoto požadavku bylo využito nástroje, který je interně zabudován v PHP. Tento nástroj se nazývá PDO.

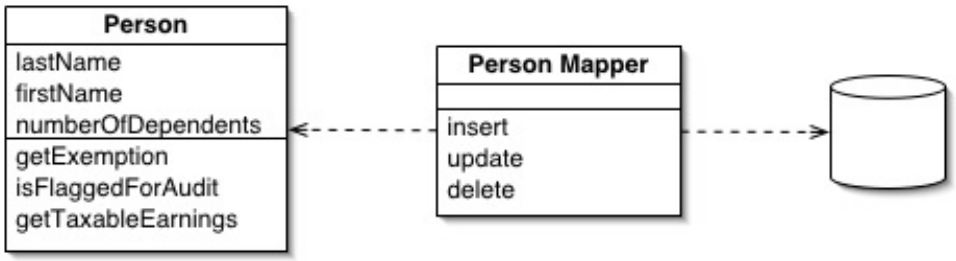
PDO najdeme v PHP verze 5 nebo vyšší. Prakticky představuje objektově orientovaný nástroj, jež poskytuje jednotné rozhraní pro práci s dvanácti různými databázovými systémy. Díky stejnému rozhraní je přechod na využívání jiného databázového systému jednoduchý a nevyžaduje v podstatě žádné změny ve zdrojovém kódu, snad až na tzv. connection string a několik dotazů, jež jsou speciálně psány pro konkrétní databázový systém. [25]

4.6.2 Implementace ORM

Pro implementování ORM byl využit návrhový vzor Data Mapper.

Návrhový vzor Data Mapper

Návrhový vzor Data Mapper implementuje aplikační vrstvu, jež se stará o oddělení dat uložených v databázi a doménových objektů v paměti, se kterou aplikace pracuje. Vrstva je zodpovědná za přenos těchto dat oběma směry a zároveň se stará o nezávislost. Doménové objekty neobsahují žádné základní ani jiné operace související s perzistencí, která je přesunuta na oddělený mapovací objekt. Ten má přístup jak k databázovému systému, tak i k doménovému objektu. [26]



Obrázek 17: Schéma příkladu využití návrhového vzoru Data Mapper. [26]

4.6.3 Autentizace

Autentizace je z důvodu bezpečnosti implementována na straně serveru. Proces vyžaduje práci s nejcitlivějšími uloženými daty, které představují hesla uživatelů. Ta jsou proto dodatečně chráněna také kryptograficky.

Při registraci uživatel dochází k šifrování jím zvoleného hesla silným jednosměrným kryptografickým algoritmem. Teprve tento obraz hesla, jež může být pro jeden vzor různý, je uložen.

Tato funkcionalita je implementována funkcemi z jádra PHP, díky čemuž aplikace může začít kdykoliv používat novější a silnější šifrovací algoritmy spolu s přechody na nové verze PHP, jež je instalována na serveru.

```
$cmd->bindParam(
                ':password',
                password_hash($password, PASSWORD_DEFAULT),
                PDO::PARAM_STR,
                256
               );
```

Výpis 3: Ukázka možnosti využití kryptografie při mapování hesla na atribut SQL dotazu funkcí z jádra PHP.

4.6.3.1 Obecně využívané hashovací funkce

Hashovací funkce jsou jednosměrné a mapují předaný řetězec libovolné délky na řetězec délky pevně dané. Kromě využití pro skrývání informací se využívají také například k porovnání dvou zpráv. [27]

Mezi nejznámější hashovací funkce patří:

- MD4
- MD5
- SHA-1
- SHA-2

4.6.4 Implementace ověření platné registrace uživatele

Aplikace během svého testování narazila na problém s podvodnými registracemi, který je spolu s řešením podrobněji popsán v podkapitole 4.7.1.

Řešením tohoto problému bylo ověřování nových registrací pomocí aktivačního odkazu, který je zasílán e-mailovou zprávou. Implementačně je toto řešeno vygenerováním náhodného řetězce znaků a čísel, který je v databázi dočasně uložen a přiřazen k jednotlivým, v té době neaktivním, uživatelům. Pro aktivaci uživatelského učtu je nutné přejít na speciálně vytvořenou stránku v rámci aplikace a předat jí aktivační řetězec spolu s jedinečným identifikátorem uživatele, kterému aktivační řetězec přináleží, prostřednictvím parametrů. Správný aktivační odkaz s potřebnými parametry je sestaven pouze v ověřovacím e-mailu, který přijde na uživatelem zadanou e-mailovou adresu.

Při správně zadaných parametrech je aktivační řetězec z databáze odstraněn a danému uživateli je účet aktivován.

```php
/**
 * Vrati nahodne vygenerovany aktivacni retezec pro potvrzeni registrace.
 * @param $activationHashLength int pozadovana delka vygenerovaneho aktivacniho
     retezce, defaultne 50 znaku
 * @return string vraci vygenerovany aktivacni retezec
 */
private function getActivationHash($activationHashLength = 50)
{
    $chars = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charsLength = strlen($chars);

    $activationHash = '';

    for ($i = 0; $i < $activationHashLength; $i++)
    {
        $activationHash .= $chars[ rand(0, $charsLength - 1) ];
    }

    return $activationHash;
}
```

Výpis 4: Ukázka metody s úkolem vygenerovat náhodný aktivační řetězec.

4.7 Problémy a jejich řešení

Během implementace aplikace bylo nutné vyřešit několik problémů, které se vyskytly. V této podkapitole bude nastíněn jeden vybraný problém a představeno bude také jeho vyřešení.

4.7.1 Podvodné registrace

Aplikace byla kvůli testování během implementace několikrát nasazena do ostrého provozu, jinými slovy byla již dostupná v prostředí internetu. V dané době byla funkcionalita registrace řešena implementací registračního formuláře. Ten při registrování uživatelé vyplnili a následně byla data zkontrolována v prohlížeči i na serveru. V případě, že byla ve správném formátu, byla registrace dokončena a uživatel se mohl přihlásit.

V době těchto testování se však ukázal problém podvodných registrací, které byly způsobeny automatickým vyplněním formuláře tzv. spamovými roboty. Bylo tedy nutné najít řešení, jak při registraci rozlišit mezi skutečnými uživateli a těmito roboty.

Jedním z možných řešení bylo zakomponovat do registračního procesu tzv. captcha kód. Ve své podstatě jde o obrázek s zdeformovaným textem, který je nutné opsat do příslušného

textového pole. Principiálně tato ochrana vychází z předpokladu, že lidský mozek je schopen, narozdíl od spamového robota, přečíst právě i zdeformovaný text. Bohužel však obecná uživatelská zkušenost vypovídá, že míra deformace je někdy tak velká, že kontrolní text z obrázku nedokáže přečíst ani člověk, což kazí uživatelský komfort. V případě vyvíjené aplikace by v těchto případech byl uživatel nucen registraci opakovat. Jelikož se přidal i požadavek na možnost zamezení přístupu uživatele do aplikaci, nebylo toto řešení zcela vhodné. [28]

Pro zamezení podvodných registrací bylo nakonec zvoleno řešení jiné. To pro ověření skutečného uživatele vyžaduje potvrzení celé registrace odkazem, jež po odeslání vyplněného registračního formuláře přijde uživateli e-mailovou zprávou na adresu zadanou při registraci. Do doby, než takto uživatel svoji registraci potvrdí, je z pohledu aplikace brán jako neaktivovaný, nemá možnost se do aplikace přihlásit a využívat ji. Toto řešení přináší dodatečnou funkcionalitu, kdy administrátor může uživatele deaktivovat a zamezit mu tak, například dočasně, přístup k aplikaci.

5 Závěr

Tato práce měla za cíl implementovat webovou aplikaci pro správu a organizaci času. Jejím úkolem bylo také analyzovat současný stav v oblasti softwarových produktů s podobným zaměřením.

Návrh a vývoj aplikace byl zaměřen na čas osobní a byl přizpůsoben dnešním trendům v oblasti webových aplikací i nejvyužívanějším webovým prohlížečům. Již od samotného návrhu přetrvávala snaha o aplikaci, která bude nezávislá na konkrétně zvoleném databázovém systému a jež bude co nejsnadněji udržovatelná. Všechny stanovené cíle práce byly postupně splněny v plném rozsahu.

Práci na návrhu a implementaci aplikace považuji za velmi zajímavou a přínosnou. Získal jsem mnoho cenných zkušeností a seznámil se s novými technologiemi, které pro mne byly do této doby zcela neznámé.

5.1 Plány dalšího rozvoje

Vyvinutá aplikace se bude v blízké budoucnosti i nadále rozvíjet. Předpokládá se zdokonalení některých funkcí spolu s implementací nových. Některé z plánů jsou uvedeny v následujícím seznamu:

- Propojení funkcionality poznámek na jednu z nejznámějších aplikací, které slouží ke spravování poznámek, na aplikaci Evernote.
- Rozšíření funkce importu událostí z aplikace Google kalendář v podobě přidání možnosti zapnutí automatické synchronizace, při které se automaticky stahují i odesílají všechny události tak, aby stav v obou těchto aplikacích byl vždy stejný.
- Možnost vytvářet více kalendářů.
- Implementace funkčnosti sdílení kalendáře, do kterého by bylo možné nahlížet i veřejně.
- Přidání drag and drop funkcionality do kalendáře pro zvýšení uživatelského komfortu.

Dále je již v plánu vytvoření nativní aplikace pro mobilní zařízení s operačním systémem Android OS. S tímto bylo počítáno od samého počátku návrhu webové aplikace, a proto nebyla navrhnuta jako responzivní či nějak jinak speciálně přizpůsobena mobilním zařízením.

Literatura

[1] Best Time Tracking Software [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.capterra.com/time-tracking-software>

[2] Kalendář Google [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.google.com/intl/cs/calendar/about>

[3] Sledování času, vyúčtování a evidence docházky [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.primaerp.com>

[4] Paymo: Online Project Management App [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.paymoapp.com>

[5] Správce času [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.spravcecasu.cz>

[6] UML 2 Use Case Diagrams: An Agile Introduction [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.agilemodeling.com/artifacts/useCaseDiagram.htm>

[7] Úvod do architektury MVC [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.zdrojak.cz/clanky/uvod-do-architektury-mvc>

[8] Igloonet Blog [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.igloonet.cz/blog/zprijemnete-si-vyvoj-webovych-stranek-s-frameworkem-ruby-on-rails-dil-2>

[9] Single page apps in depth [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.singlepageappbook.com/goal.html>

[10] Začínáme s AngularJS [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.zdrojak.cz/clanky/zaciname-s-angularjs>

[11] Google Trends [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.google.cz/trends>

[12] AngularJS - Superheroic JavaScript MVW Framework [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.angularjs.org>

[13] AngularJS Documentation [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <docs.angularjs.org/guide/databinding>

[14] jQuery [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.jquery.com>

[15] jQuery UI [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.jqueryui.com>

[16] CKEditor - The best web text editor for everyone [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.ckeditor.com>

[17] ngDialog.js by likeastore [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.likeastore.github.io/ngDialog>

[18] AngularJS: Documentation for route [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.docs.angularjs.org/api/ngRoute/service/$route>

[19] AngularJS: API: ngAnimate [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.docs.angularjs.org/api/ngAnimate>

[20] GitHub: Pagination Directive [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.github.com/michaelbromley/angularUtils>

[21] Angular-charts [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.chinmaymk.github.io/angular-charts>

[22] AngularJS: API: ngAnimate [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.docs.angularjs.org/api/ngAnimate>

[23] AngularJS: Developer Guide: Forms [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.docs.angularjs.org/guide/forms>

[24] AngularJS: API: $http [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.docs.angularjs.org/api/ng/service/$http>

[25] PHP: PDO - Manual [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.php.net/manual/en/book.pdo.php>

[26] P of EAA: Data Mapper [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.martinfowler.com/eaaCatalog/dataMapper.html>

[27] Hashovací funkce [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7029>

[28] CAPTCHA: Jak se stát otrokem podivného obrázku [online]. 2016 [cit. 2016-04-06].
Dostupné z WWW: <www.zive.cz/Clanky/CAPTCHA-Jak-se-stat-otrokem-podivneho-obrazku/sc-3-a-144482/default.aspx>

A Instalace aplikace

Aby mohla být aplikace spuštěna, je nutné mít na serveru nainstalované PHP ve verzi 5 a vyšší. Dále je vyžadován databázový systém. Doporučeným je systém MySQL ve verzi 5 a novější, jelikož pro používání tohoto systému je aplikace nastavena ve výchozím stavu. Na zvoleném databázovém enginu pak nezáleží. Při využití jiného databázového systému je třeba upravit připojovací řetězec ve třídě `Database`.

Pro podporu zasílání ověřovacích registračních e-mailů a možnost využívání funkcionality notifikací, jež zasílá e-mailové zprávy uživatelům před začátkem jejich událostí, je nutné mít na serveru k dispozici také mail server a přidán záznam v plánovači úloh pro pravidelné spouštění v půlhodinových intervalech scriptu `scheduled/upcomingEventsNotification.php`.

Ve zvolené databázi je dále před začátkem používání aplikace nutné vytvořit strukturu všech tabulek. To lze jednoduše provést vykonáním SQL scriptu s názvem `initDB.sql` v adresáři `sql`. Přístupové údaje k databázi je zapotřebí vložit v souboru `orm/Database.php` jako implicitní parametry pro konstruktor.

Při testování aplikace bylo využito PHP ve verzi 5.6.18, MySQL databáze ve verzi 5.6.28 a programu CRON, jenž na operačním systému Linux umožňuje jako systémový nástroj pravidelné spouštění scriptu, který zajišťuje funkcionalitu notifikací.

Pro rychlé spuštění aplikace lze využít také balík instalací, který se nazývá XAMPP. Ten obsahuje server Apache, PHP i MySQL.

B Adresářová struktura k práci přiloženého CD

Součástí bakalářské práce je CD. Kořenový adresář tohoto CD je strukturován následovně:

- `text` - obsahuje text bakalářské práce v elektronické podobě
- `documentation` - obsahuje uživatelskou dokumentaci aplikace
- `source` - obsahuje veškeré zdrojové kódy aplikace