Masarykova univerzita
Fakulta informatiky

Email based recurrent reminder system

Bakalářská práce

Daniel Jurča

Brno, podzim 2019

Masarykova univerzita
Fakulta informatiky

Email based recurrent reminder system

Bakalářská práce

Daniel Jurča

Brno, podzim 2019

Na tomto místě se v tištěné práci nachází oficiální podepsané zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Daniel Jurča

Vedoucí práce: RNDr. Dušan Klinec

Poděkování

Tímto bych rád poděkoval svému vedoucímu RNDr. Dušanovi Klincovi za cenné rady, trpělivost a ochotu pří vedení práce. Chtěl bych také poděkovat rodině a přátelům za podporu.

Shrnutí

Cílem této bakalářské práce je navrhnout a implementovat webovou aplikaci na připomínání rekurentních událostí. O těchto událostech bude uživatel notifikován prostřednictvím e-mailových upozornění. Práce se v úvodu zabývá alternativními přístupy k řešení problematiky. Dále se věnuje popisu frameworku Laravel a ostatních použitých nástrojů. Následuje popis návrhu aplikace pomocí běžně používaných metod při vývoji softwaru a samotná implementace s ukázkami kódu a grafického rozhraní. Na závěr jsou uvedeny možnosti dalšího rozšíření aplikace.

Klíčová slova

Laravel, PHP, webová aplikace, notifikace, notifikační e-mail

Obsah

1 Úvod

V posledních letech zaznamenala výpočetní technika velký rozmach a setkáváme se s ní každodenně. Schopnost pracovat s počítačem nebo chytrým telefonem se stala v zaměstnání běžnou nutností. Nezbytným předpokladem k neopomenutí pracovních povinností je vytváření upomínek. Samolepící bločky tak postupně nahrazují notifikační aplikace hrající roli plánovače, které zpříjemní práci a připomenou důležité úkoly či schůzky.

Na internetu se nachází takových aplikací mnoho. Bohužel, jsou příliš komplikované nebo drahé. Posílání notifikací je u nich pouze vedlejší záležitostí. Mým cílem je vytvořit aplikaci, která se zaměřuje výhradně na tuto problematiku. Aplikace bude zdarma a bude dostupná pro všechny potenciální uživatele.

Cílem této práce je navrhnout a implementovat webovou aplikaci, která usnadní vytvářet události a následně je připomínat prostřednictvím e-mailové komunikace. Z důvodu využití mnoha open source technologií má být i výsledná aplikace volně dostupný open source projekt. Záměrem je, aby uživatelé nemuseli mít konkrétní typ účtu, instalovat speciální mobilní aplikaci nebo plugin do internetového prohlížeče. Mým úkolem je navrhnout jednoduché a intuitivní webové rozhraní. Velký důraz bude kladen na to, aby software mohl ovládat i méně zkušený uživatel.

Aplikace je vytvářena pro výzkumné centrum CRoCS (Centre for Research on Cryptography and Security) na Fakultě Informatiky, zaměřené na aplikovanou kryptografii a bezpečnost informačních technologií. Hlavním důvodem využití tohoto systému budou upomínky pro včasné dokončení deadlinů a organizaci schůzek zaměstnanců.

K implementaci notifikačního systému budu využívat open source framework Laravel [1] pro programovací jazyk PHP. K realizaci webdesignu budu používat technologie HTML, CSS a JavaScript. Případné další grafické elementy budou navrženy v grafickém editoru Adobe Photoshop.

První kapitola bude věnována notifikačním systémům, bude obsahovat představení aplikace a popisovat alternativní typy těchto systémů a již existujících nástrojů. V druhé kapitole se zaměřím na návrh systému a specifikaci požadavků spolu s diagramy případu užití

a ER diagramem[1]. Čtvrtá kapitola se bude zabývat nástroji, které byly použity při vyvíjení tohoto softwaru. Následující kapitola bude popisovat samotnou implementaci aplikace, která je rozdělena na serverovou část (backend [2]) a klientskou část (frontend [2]). V závěrečných kapitolách pak popíšu možnosti dalšího rozšíření aplikace a zhodnotím, zda byly naplněny cíle této práce.

1. ER diagram (ERD) - Entitně relační diagram

2 Notifikační systémy

Tato kapitola vysvětluje základní problematiku notifikačních systémů a popisuje, jak tyto aplikace pracují, jaké mají vlastnosti a využití. Zabývá se motivací k použití a nasazení nové aplikace. Dále zahrnuje výčet alternativních způsobů řešení upomínkového systému. Obsahuje ukázky a popis již implementovaných nástrojů pro tvorbu notifikací.

2.1 Problematika

V oblasti informačních technologií označujeme *notifikačním systémem* (NS) aplikaci, jejíž základním principem je upozornit cílovou skupinu na nadcházející událost. Cílovou skupinou může být osoba, skupina lidí, organizace nebo celý svět. Účelem NS je generovat a zasílat zprávy v nastavený čas. Tyto zprávy nazýváme notifikace z anglického *"notification"* (v překladu oznámení). Notifikace obsahuje především vhodně zvolený název popisující danou událost, stručný popis, datum a čas. Dnes je k dispozici řada webových i desktopových aplikací, které splňují aspekty notifikačního systému. Ve velké míře je schopnost zasílat upomínky pouze vedlejší funkcionalitou aplikace, která byla vyvinuta k jinému účelu. Ať už se jedná o aplikaci desktopovou či webovou.

2.2 Email based recurrent reminder system

Cílem bylo vytvořit webovou aplikaci, která bude nasazena na fakultním serveru FI. Webová aplikace je snadno udržitelná, nevyžaduje instalaci žádného softwaru a nebude omezena výběrem operačního systému. Systém je založen na rekurentním posílání notifikací prostřednictvím e-mailu. E-mail byl zvolen pro jednoduchost a rychlost komunikace. Pokud má uživatel synchronizovanou e-mailovou schránku na chytrém zařízení, je to pro něj velké plus. Notifikace se mu v tomto případě objeví na smartphonu či tabletu i bez nutnosti vyvíjení mobilní aplikace.

V laboratoři CRoCS bude systém sloužit k připomínání schůzek, deadlinů a dalších potřebných událostí. Pro ulehčení a zrychlení práce se zaměstnanci nemusí registrovat, aby notifikaci obdrželi. Pro vytvá-

ření těchto událostí je ale účet vyžadován. Aby bylo možné doručit e-maily i neregistrovaným osobám, musí být vytvořena role skupinového administrátora (Admin), který bude mít právo připojit osoby bez účtu k události. Admin má k dispozici logování odeslaných e-mailů i akcí pro odložení či dokončení rekurentních událostí. Může tak sledovat, kdo již úkol dokončil a kdo si jej neustále odkládá.

V notifikačním systému lze vytvářet události dvou typů:

Jednorázová událost (One-time) se odešle pouze jednou v nastavený den a čas. Taková událost má praktické využití v případech, kdy odkládání události nemá smysl, případně kdy odložení události není možné (termín je stanovený fixně, uživatel ho z vlastní vůle nemůže změnit). Pokud by mělo dojít k odložení události, fakticky se jedná o zrušení události a následně je třeba vytvořit novou událost v jiný termín. Tento typ události je možné použít například pro připomenutí důležité nezrušitelné schůzky nebo odevzdání práce do stanoveného deadlinu, který již nejde posunout. Jedná se o situace, kdy dostaneme připomenutí na poslední chvíli.

Rekurentní událost (Recurrent) naopak slouží pro vytváření upomínek, které chce uživatel dostávat opakovaně. Uživatel si nastaví čas prvního doručení notifikace i po jakém časovém intervalu se má notifikace znovu odeslat v případě nesplnění úkolu (uživatel odloží událost). Jakmile uživatel úkol splní (odsouhlasí dokončení), událost se považuje za hotovou a přestane se posílat. Rekurentní událost je možné využít při úkolech, které nemají daný pevný termín dokončení, ale je potřeba je splnit v rámci určitého časového intervalu (týden, měsíc, ...). Uživatel může mít v tomto rozhodném období takových úkolů více a díky rekurentní události na ně nezapomene a sám si určí termín jejich postupného splnění.

2.2.1 Motivace pro nasazení

Aplikace by měla být nasazena a testována v laboratoři CRoCS (Centre for Research on Cryptography and Security) [3]. Laboratoř se zabývá výzkumem bezpečnosti a soukromí v oblasti IT. Od prvopočátku sídlí CRoCS v budově Fakulty informatiky na Masarykově univerzitě v Brně.

Z pohledu využívání softwaru, se jedná o heterogenní prostředí. Pro zaměstnance zde nejsou kladeny nároky na použití speciálního operačního systému (dále OS) nebo webového prohlížeče. Aby mohla být aplikace nasazena do takového prostředí, je práce motivována následujícími požadavky:

- Aplikaci bude možno využít na kterémkoliv OS.
- Uživatelé, kteří budou notifikace pouze přijímat, si nemusí v systému vytvářet účet.

2.3 Alternativní přístupy

Existuje několik způsobů, jak přistoupit k vývoji notifikačního systému. Je tady hned několik zásadních rozhodnutí, která se musí stanovit před začátkem návrhu aplikace.

Typ aplikace - První krok je zvolit vhodný typ aplikace. Nabízí se několik možností.

Desktopová aplikace - Poskytuje řadu výhod. Programy se spouští přímo z pevného disku na našem počítači. Aplikace se načte velmi rychle a zároveň je k dispozici bez ohledu na internetové připojení. Ve většině případů však stejně vyžadujeme připojení k doručení notifikace uživateli. Desktopová aplikace vyžaduje instalaci a neustálé aktualizace, proto dnes vývoj takových aplikací značně ubývá. Aktualizace softwaru také mohou obsahovat škodlivý kód, skrze který může dojít k útoku na aplikaci. Dalším častým problémem této aplikace bývá meziplatformová kompatibilita. V porovnání z ostatními typy je náročnější na implementaci.

Webová aplikace - Jedná se o modernější a výhodnější řešení problému. Aplikace běží přímo v okně internetového prohlížeče. Uživatel tento software nemusí instalovat ani udržovat nejnovější verzi. Dnešní webové aplikace se vytváří s responzivním designem (responsive web design). Tento přístup zaručí, že zobrazení stránky bude optimalizováno pro všechny druhy chytrých zařízení. Nevýhodou webové aplikace je totální závislost na připojení k internetu.

Mobilní aplikace - Chytrý telefon dnes vlastní většina populace. Jelikož jde o zařízení, které nosíme neustále u sebe, může být aplikace velice efektivní. Software ovšem vyžaduje instalaci a často i připojení k internetu.

Typ notifikace - Vybrat vhodný způsob, jakým bude upomínka doručena. Notifikace je krátká zpráva, která informuje o určité události, tudíž její doručení by mělo proběhnout jednoduše a rychle. Na základě priority doručení můžeme notifikace rozdělit do dvou skupin.

1. **Zpráva bude doručena v rámci aplikačního rozhraní.**
   Taková informace se zobrazí až při samotné interakci s aplikací.

   Nejpoužívanější notifikací je *Pop-up* oznámení, které sdělí uživateli informace bez nutnosti na něj ihned reagovat. Obvykle se zobrazí na pár vteřin nebo má v rohu křížek na zrušení. Nejvíce se vyskytují na sociálních sítích.

   Dialogové okno se používá, pokud se požaduje okamžitá reakce od uživatele. Většinou zablokuje celou aplikaci, než dojde k interakci.

   Velmi často se vyskytuje také *Aplikační chat*, který je spravován týmem pro služby zákazníků. Jedná se o chatovacího klienta zasazeného přímo v aplikaci, skrze kterého lze v reálném čase komunikovat s uživatelem.

   Populární aplikace ukládají notifikace do seznamu, který si uživatel může kdykoliv otevřít a zkontrolovat (Facebook, YouTube, ...).

2. **Zpráva musí být k uživateli doručena rychlejší cestou prostřednictvím vnější komunikace.**
   Tato zpráva je závislá na čase, a tudíž má vysokou prioritu doručení.

   Nejrozšířenějším typem je *Notifikační e-mail*. Relativní většina webových aplikací využívá e-mail k doručení oznámení o události (např. reklamní sdělení).

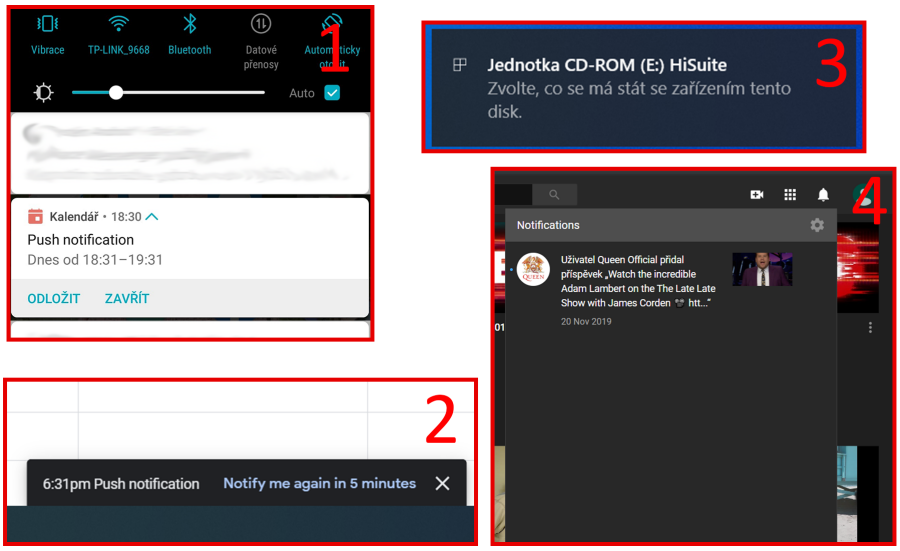
   Dále se běžně využívá takzvaný *Chatovací bot*, který skrze sociální síť kontaktuje uživatele. Obvykle zasílá zprávu pro-

střednictvím aplikace Facebook nebo Messenger. Do aplikací tohoto typu se osoba musí přihlásit přes účet na sociální síti. Chatovací bot není příliš běžný způsob obdržení notifikací. Ukázkovým příkladem může být aplikace Reminder Bot[1].

Notifikace *Web push* reprezentují oznámení, která jsou protlačena webovým prohlížečem. Aby se oznámení objevila na obrazovce, musí být v prohlížeči tato možnost povolena. *Mobile push* notifikace jsou jejich obdobou pro mobilní zařízení.

Textové zprávy na mobilní telefon by měli být až poslední volbou. Používají se pouze v krajních případech, kdy hrozí velká rizika.

Na obrázku 2.1 se nachází ukázky několika běžných notifikací. První notifikace reprezentuje *Mobile Push* notifikaci pro mobilní zařízení pomocí aplikace Google Kalendář. Druhá notifikace zobrazuje tutéž událost, prostřednictvím notifikace *Web push*.



Obrázek 2.1: Ukázka zobrazení několika základních notifikací

1. https://reminderbot.io/

Číslem 3 je označena notifikace *Pop-up*, která se zobrazí například při připojení zařízení k PC (ukázka z OS Microsoft Windows). Poslední obrázek ukazuje seznam notifikací v aplikaci YouTube.

2.4 Existující nástroje

Naprostá většina nástrojů, které mají funkci pro vytváření upomínek, jsou rozsáhlé aplikace s mnoha funkcemi. Schopnost notifikaci doručit je tedy pouze vedlejší (nicotná v porovnání se skutečnou podstatou aplikace). Následuje výběr několika nástrojů, které umožňují posílat upomínky skrze e-mail. Programy jsou stručně popsány a jsou vysvětleny principy jejich užívání. Jelikož takových aplikací existuje nespočet, uvádím zde pouze zlomek.

2.4.1 Volně dostupné nástroje

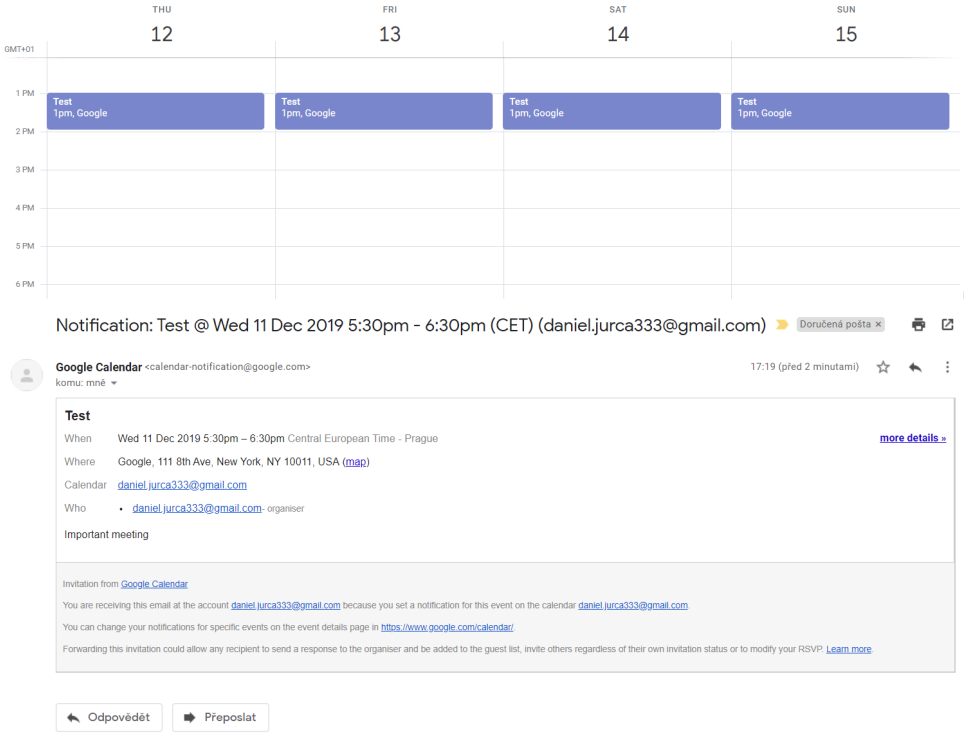
Používání volně dostupných nástrojů (*Free to use*) je zcela zdarma a bývají k dispozici online. Tyto programy mají tzv. freeware licenci. Některé jsou zároveň i open-source, což znamená, že mají dostupný zdrojový kód.

Google Kalendář [2]

Jedná se o jeden z nástrojů od společnosti Google, které jsou volně dostupné po vytvoření tzv. *Google Account* [4]. Kalendář je webová aplikace sloužící ke správě času. Mobilní zařízení s platformou Android obsahuje mobilní verzi Kalendáře defaultně.

Google poskytuje možnost instalace aplikace na PC. Umožňuje vytváření událostí s mnoha funkcemi. Vyjma základních vlastností lze také nastavit rekurentní zasílání upomínky i jakého typu upomínka bude (e-mail nebo oznámení v prohlížeči). Aplikace zároveň umožňuje posílat e-maily i mimo Google uživatele, proto je vhodná i pro celé skupiny lidí.

2. https://calendar.google.com/calendar/



Obrázek 2.2: Naplánovaná rekurentní událost a ukázka upomínky v aplikace Google Kalendář

Nudgemail [3]

Nudgemail je webová aplikace pro připomínání událostí prostřednictvím e-mailu. Pro využívání jeho služeb nevyžaduje vytváření žádného účtu. Uživatel pouze musí ověřit svoji e-mailovou adresu, na kterou mu budou notifikace přicházet. Je k dispozici zdarma [5].

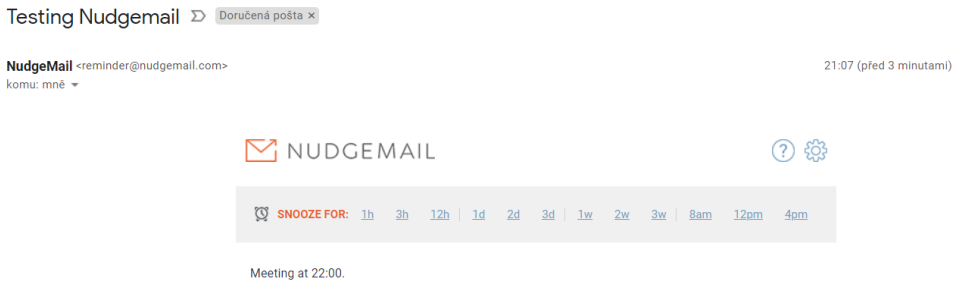
Aplikace neumožňuje vytvářet notifikace pro skupinu lidí (pouze osobní využití). Uživatel vytváří událost jen pro sebe přímo ze svého e-mailového klienta bez potřeby speciálních pluginů. Tvorba probíhá odesláním e-mailu ve speciálním formátu aplikaci Nudgemail. Předmět a zpráva se vyplní informacemi o události, které chce osoba obdržet. Adresa příjemce má speciální tvar *format@Nudgemail.com*. *Format* reprezentuje datum a čas odeslání upomínky, stejně tak může udávat,

3. https://www.nudgemail.com/

jak často se má notifikace posílat. Vytvořené upomínky nelze po zadání nijak upravovat, jedinou možností je notifikace zrušit a znovu vytvořit.

Ukázka:

Aby byla událost nastavena na **každého 24. prosince v 18:00** je třeba poslat e-mail na adresu – **yearly24dec600pm@Nudgemail.com**. Na obrázku 2.3 se nachází ukázka vygenerované upomínky.



Obrázek 2.3: Notifikační e-mail aplikace Nudgemail

2.4.2 Komerční nástroje

Za použití tohoto softwaru se musí tvůrci aplikace zaplatit. Většinou nabízí k vyzkoušení trial verzi na omezenou dobu.

Acuity Scheduling [4]

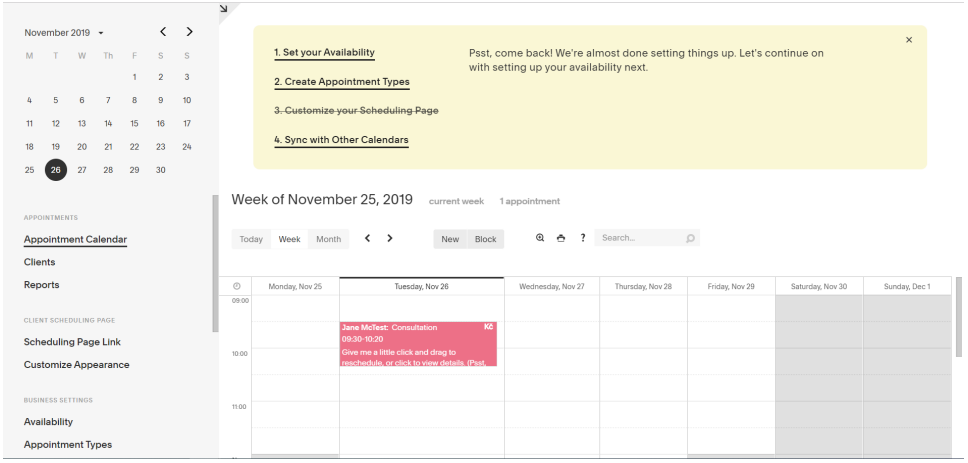
Webový nástroj pro vytváření schůzek a událostí. Poskytuje stejnojmennou aplikaci pro mobilní zařízení. Vyžaduje registraci uživatele. Software je doporučen používat v malých firmách.

Systém synchronizuje kalendáře v rámci aplikace, a tak plánování událostí nečiní žádné problémy. Díky tomu je snadné vytvářet setkání a úkoly pro jiné osoby. Kalendář Acuity Scheduling lze též synchronizovat s kalendářem Google. V trial verzi je funkce vytváření událostí omezena pouze na sebe. Notifikace probíhají skrze e-mail

4. https://acuityscheduling.com/

nebo textovou zprávu. Schopnost integrace se softwarem třetích stran a synchronizace kalendáře s Googlem i Outlookem.

Nejvyužívanější balíček aplikace Acuity Scheduling měsíčně vychází na 25 dolarů (cca 570 Kč) [6].



Obrázek 2.4: Webové rozhraní aplikace Acuity Scheduling

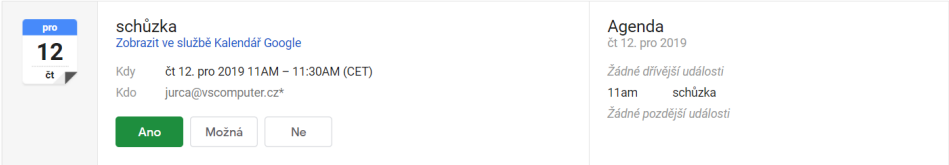
Microsoft Outlook [5]

E-mailový klient Outlook lze koupit samostatně nebo jako součást balíčku Microsoft Office. Uživatel si musí zaregistrovat novou e-mailovou adresu *@outlook.cz*. Outlook je k dispozici v desktop nebo online podobě.

Slouží pro správu e-mailů, kontaktů a organizaci úkolů a času. Outlook je velmi často nasazován do velkých firem. Používá se jako stand-alone aplikace nebo spolu s Microsoft Exchange Server pro sdílení schránek a kalendářů. Součástí klienta je mimo elektronické pošty k dispozici kalendář pro plánování událostí. Událost lze nastavit opakovaně a přizvat další uživatele aplikace. Notifikace se zobrazí v dialogovém okně nebo pomocí upomínkového e-mailu.

Samostatná aplikace Microsoft Outlook je nyní (prosinec 2019) dostupná v přepočtu za necelé 3000 Kč [7].

5. https://outlook.live.com/



Obrázek 2.5: Notifikace aplikace Microsoft Outlook

FollowUpThen [6]

Zpoplatněný webový systém velmi podobný softwaru Nudgemail. Nabízí pouze měsíční zkušební verzi. Vyžaduje registraci uživatele prostřednictvím e-mailu.

Událost se vytvoří odesíláním e-mailu na adresu *@followupthen.com* (jako u Nudgemail. Předvolba adresy určuje termíny, kdy se upomínka odešle. Dovoluje posílat notifikace i jiným osobám, pokud je přidá do kopie. Mezi velmi přínosnou funkcionalitu patří možnost poslání notifikace ve formě textové zprávy.

FollowUpThen vychází v rozmezí na 2-9 dolarů měsíčně (cca 50-210 Kč)[8].

Poznámka

Komerční aplikace, které byly výše zmíněny, byly testovány ve zkušebních verzích. V případě Microsoft Outlook jsem využil studentskou licenci poskytovanou Masarykovou univerzitou.

6. https://www.followupthen.com/

3 Návrh aplikace

Následující kapitola se zabývá samotným návrhem aplikace. Na začátku charakterizuje specifikace požadavků na aplikaci, které byly dohodnuty po konzultaci s vedoucím práce. Dále zobrazuje možnosti uživatelů s diagramem případu užití. Na závěr popisuje návrh databáze s ER-diagramem.

3.1 Specifikace požadavků

Specifikace požadavků bývá zpravidla první fáze vývoje různých aplikací. Aplikační požadavky se dělí do dvou skupin:

Nefunkční požadavky - Definují omezující podmínky pro systém i jeho služby. Patří zde např. spolehlivost, bezpečnost, použití platformy, programovacího jazyka a další vlastnosti. Může být kritickým místem při vývoji softwaru.

Funkční požadavky - Popisují služby, které by měl systém poskytovat. Zachycují také chování systému v konkrétních situacích a na specifických vstupech.

3.1.1 Nefunkční požadavky

- *Technologie* - Webová aplikace je spustitelná na fakultním webovém serveru Apache s databází MySQL. Bylo mi doporučeno aplikaci implementovat v jazyce PHP v open-source frameworku Laravel.

- *Rozhraní* - Rozhraní aplikace musí být uživatelsky přívětivé a intuitivní, aby ji mohli používat i méně zdatní uživatelé.

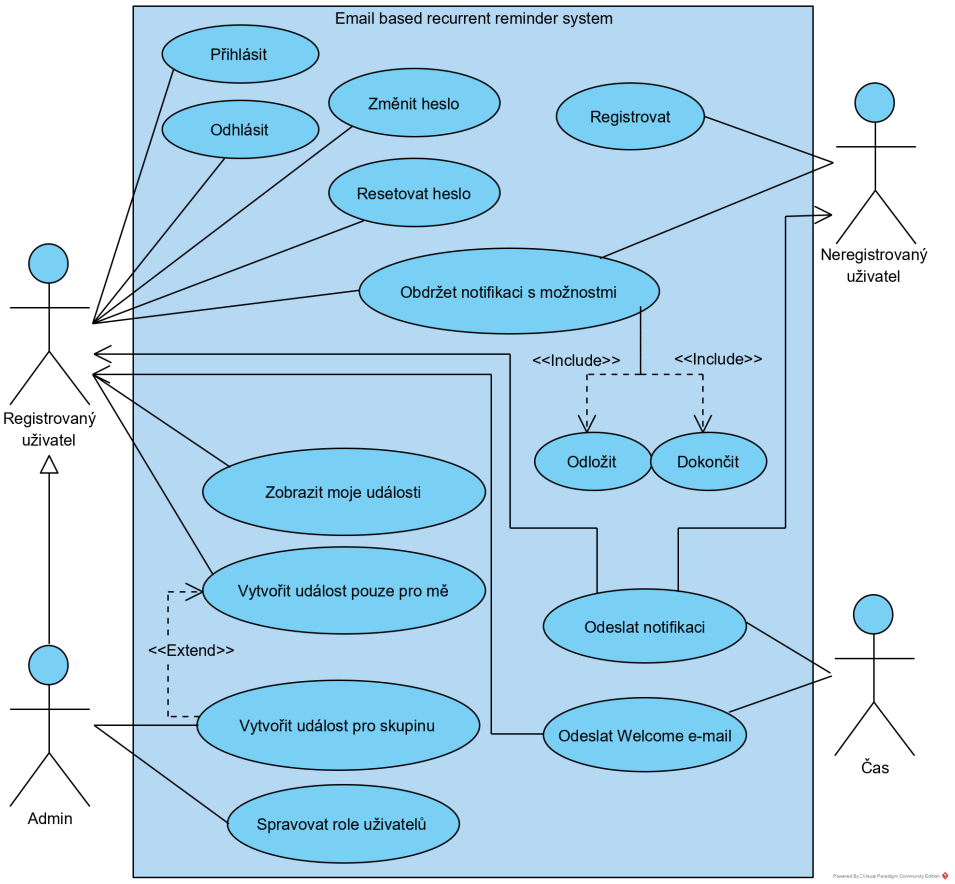
- *Jazyk* - Pro prostředí aplikace byl zvolen anglický jazyk, aby mohli systém plnohodnotně využívat i uživatelé mimo Českou a Slovenskou republiku.

3.1.2 Funkční požadavky

- *Registrace* - Systém umožňuje uživateli vytvořit účet v aplikaci.
- *Přihlášení a odhlášení* - Uživatel se se může přihlásit do systému prostřednictvím vytvořeného účtu. Zrovna tak je schopen se odhlásit ze systému.
- *Vytváření události* - Přihlášený uživatel je schopný vytvářet události, o kterých chce být informován. Autor události může zároveň událost upravit nebo smazat. U události může rozhodnout, zdali bude jednorázová nebo rekurentní.
- *Posílání notifikačních e-mailů* - Systém zašle upomínku se stručným popisem uživateli v podobě notifikačního e-mailu. Čas doručení notifikace se stanovuje při vytváření události.
- *Možnosti postpone a done* - Prostřednictvím doručené notifikace může uživatel událost odložit (*postpone*) nebo ji označí jako dokončenou (*done*).
- *Registrovaný uživatel* - Obyčejný uživatel má možnost vytvářet události pouze pro sebe. Může si zobrazit všechny události, u kterých je přihlášen.
- *Skupinový administrátor* - Role "Group Administrator"rozšiřuje funkci obyčejného registrovaného uživatele. Mimo tytéž vlastnosti, dokáže navíc vytvářet události pro skupinu uživatelů. Zároveň má právo spravovat role registrovaných uživatelů.
- *Neregistrovaný uživatel* - Uživatel může obdržet notifikaci i v případě, že nemá vytvořený účet. Disponuje pouze vlastnostmi odkládat a dokončit notifikace. Událost pro tento typ uživatelů je schopen vytvářet pouze group administrator.
- *Změna nebo resetování hesla* - V případě ztráty hesla, systém umožňuje registrovaným uživatelům jej resetovat skrze e-mail. Přihlášený uživatel si může své heslo změnit kdykoliv.
- *Welcome e-mail* - Jakmile se uživatel registruje, systém mu na adresu zašle uvítací e-mail.

3.2 Analýza případů užití

Diagram případů užití zachycuje funkční požadavky systému. Popisuje aktéry, kteří se systémem pracují a schopnosti každého z nich. Případy užití reprezentují jednotlivé elipsy. Základní funkcí notifikačního systému je vytváření událostí a posílání upomínkových e-mailů. Na následujícím obrázku se nachází již zmiňovaný diagram případů užití. Každý případ užití (use case) je podrobně popsán v předchozí kapitole.



Obrázek 3.1: Diagram případů užití (Use case diagram)

Na předchozím obrázku jsou zobrazeni **čtyři aktéři**. Registrovaný uživatel, neregistrovaný uživatel a admin (Group Administrator) byli

přesně definováni ve funkčních požadavcích. Čas představuje skrytého čtvrtého aktéra. Systém sám kontroluje, kdy se budou notifikace posílat na základě aktuálního datumu a času.

3.3 Návrh databáze

Jedna z nejdůležitějších součástí vývoje aplikace je řádně navržená databáze. Na obrázku 3.2 je zobrazena aktuální databáze systému prostřednictvím entitně relačního diagramu.

V průběhu implementace se databáze ještě hodně měnila. Tabulka ***migrations*** je jediná tabulka, která nesouvisí s během a fungováním samotné aplikace. Byla automaticky vygenerována frameworkem. Více informací o migracích viz kapitola 4.1.2.

Tabulka ***users*** reprezentuje uživatele, který se zaregistroval do aplikace. Obsahuje jméno, e-mailovou adresu a heslo. Tito uživatelé mají přístup do notifikačního systému pomocí přihlašovacích údajů.

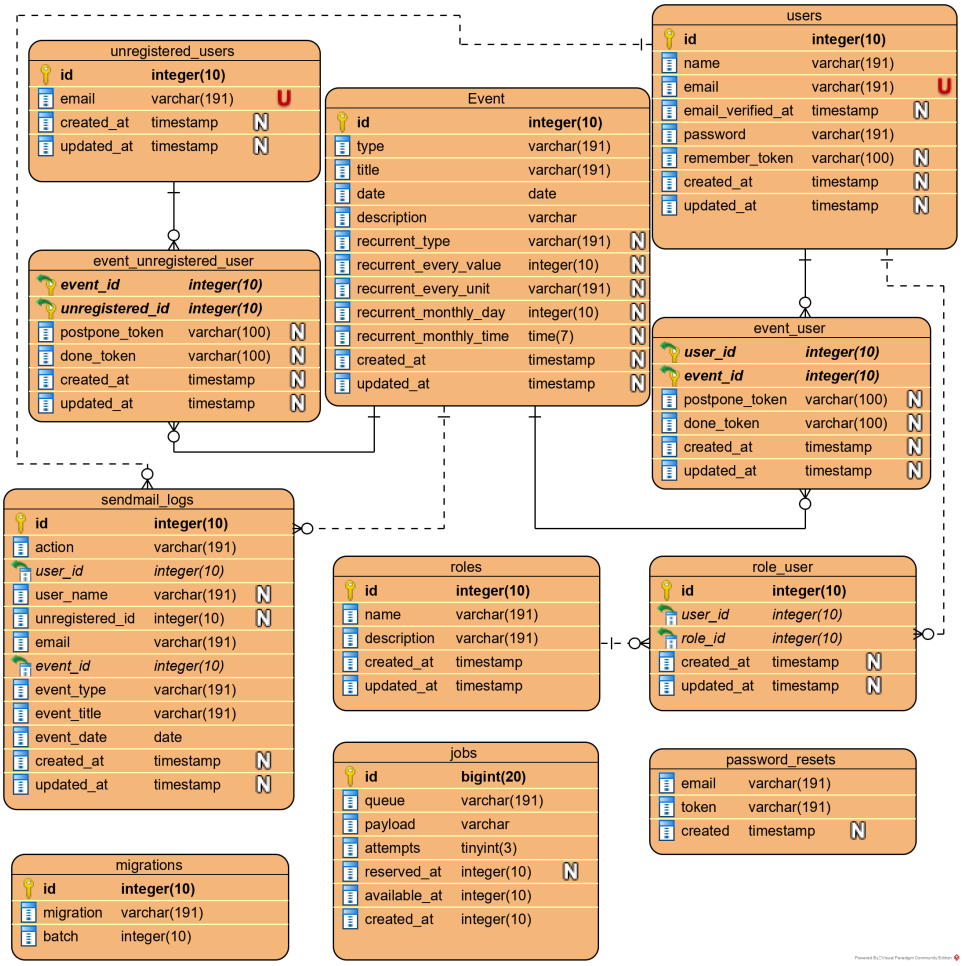
Tabulka ***unregistered_users*** představuje uživatele, kteří mohou pouze obdržet notifikační e-mail. Jsou zde uloženy pouze e-mailové adresy. Účelem této tabulky bylo předejít duplikaci e-mailu a pro přehlednost.

Events představuje událost, která obsahuje název, datum, popis a další vlastnosti v závislosti na typu události. Abychom nemuseli procházet více tabulek, ukládá události obou typů (jednorázové i rekurentní), tudíž některé vlastnosti mohou být nastaveny jako *nullable*. Na základě událostí uložených v této tabulce se vytváří notifikační e-maily.

Abychom byli schopni propojit uživatele a události, bylo potřeba vytvořit tabulky ***event_user*** a ***event_unregistered_user***. ***Event_user*** obsahuje odkazy na tabulky ***events*** a ***users***. Událost *event_id* je propojena s registrovaným uživatelem *user_id*. Tabulka ***event_unregistered_user*** funguje úplně stejně s tím, že se odkazuje na tabulku ***unregistered_users*** prostřednictvím *unregistered_id*. Zmíněné vlastnosti (sloupce) reprezentují cizí klíče. Tyto tabulky také obsahují sloupce *postpone_token* a *done_token*, které jsou nutné pro ověření bezpečnosti přístupu pro odkládání a dokončení událostí.

Roles definuje přístupové role pro registrované uživatele. Obsahuje sloupce pro název a popis. Prozatím existují pouze role uživatel

a administrátor. Tabulka ***role_user*** propojuje registrované uživatele *user_id* s jejich rolemi *role_id*.



Obrázek 3.2: ER diagram databáze

Tabulka ***jobs*** představuje práci, jejíž účel je poslat notifikační e-mail. Přidávání a odebírání do tabulky funguje na principu fronty.

Password_resets je vygenerovaná pro uživatele v tabulce ***users***. Slouží pro resetování hesla registrovaných uživatelů pomocí e-mailu a tokenu.

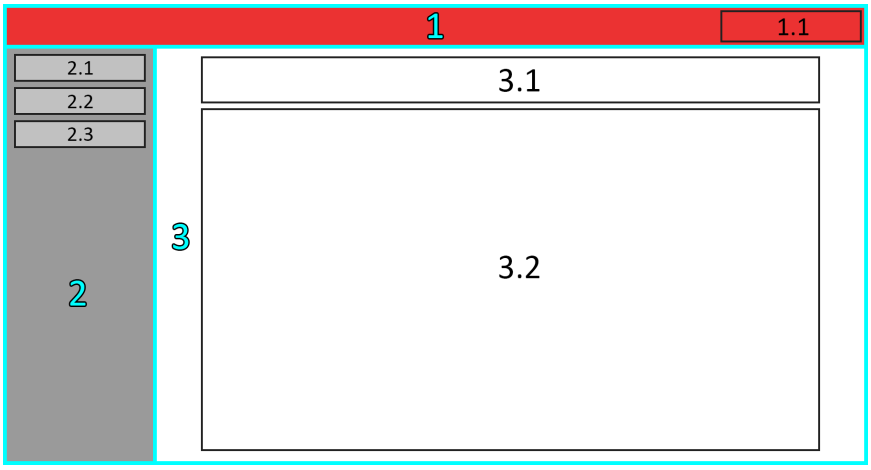
Tabulka ***sendmail_logs*** reprezentuje logovací záznam notifikačního systému. Ukládá informace týkající se upomínkových e-mailů. Obsahuje mnoho vlastností o uživateli i o události. Atribut *action* udává typ provedené akce ("SEND", "POSTPONE", "DONE").

3.4 Návrh grafického rozhraní

Kapitola se zabývá návrhem základních grafických komponent webové aplikace. Návrh je rozdělen na dvě části v závislosti na typu uživatele. Uživatelské prostředí je hlavním rozhraním pro manipulaci s událostmi a je dostupné pouze přihlášeným uživatelům. Notifikační e-mail je zpráva s informacemi o události a může ho obdržet kterýkoliv uživatel.

3.4.1 Uživatelské rozhraní

Návrh rozložení stránky webové aplikace lze rozdělit na 3 základní oblasti, viz obrázek 3.3. Každá oblast má svoji specifickou funkcionalitu a může obsahovat další podoblasti. Jednotlivé oblasti mají vyhrazené hranice pro snadnou a rychlou orientaci v aplikaci.



Obrázek 3.3: Koncept uživatelského rozhraní

Popis konceptu UI[1]

1. *Vrchní panel*. V tomto panelu se nachází jeden ovládací prvek, odhlásit uživatele z aplikace (viz panel 1.1). Zobrazuje jméno právě přihlášeného uživatele. Tento panel se vyskytuje v každé části aplikace, kromě stránek spojené s autentizací uživatele.

2. *Boční panel*. Reprezentuje hlavní navigační menu. Umožňuje navigaci mezi všemi okny aplikace. Panely s čísly 2.1, 2,2 a 2,3 reprezentují odkazy do ostatních oken webové aplikace. Boční panel je možné skrýt pro lepší přehlednost. Vyskytuje se ve všech případech spolu s vrchním panelem 1.

3. *Hlavní panel*. V této oblasti se nachází hlavní funkcionalita dané stránky. Obsahuje nejčastěji dva panely. Panel 3.1 identifikuje vlastnost stránky. V panelu 3.2 se nachází samotný obsah a funkcionalita. Tato oblast je jiná pro každou stránku aplikace.

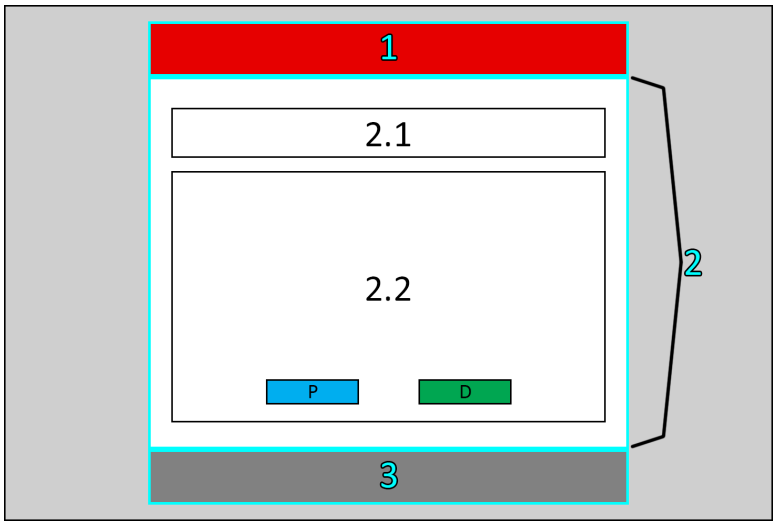
3.4.2 Notifikační e-mail

Obdržet notifikační e-mail je umožněno všem, nehledě na to, zda má osoba vytvořený účet. Nicméně, neregistrovaný uživatel dostane e-mail pouze tehdy, když pro něj vytvoří událost skupinový administrátor. Notifikační e-mail je navržený v podobném stylu jako UI webové aplikace.

E-mail obsahuje název události, datum a čas a popis události. Rozlišujeme dva typy notifikačního e-mailu, jednorázový a rekurentní. Pro rychlejší rozpoznání budou notifikace barevně rozlišeny. Rekurentní e-mail navíc obsahuje funkce pro vykonání akce (Postpone a Done).

Na obrázku 3.4 je vyobrazen návrh rekurentního notifikačního e-mailu, který je opět rozdělen na 3 základní oblasti. Následně jsou popsány jednotlivé oblasti a funkce. Šedá barva na obrázku reprezentuje pozadí e-mailu. Notifikace o události je velmi krátká zpráva, proto samotná informativní část e-mailu je kvůli estetice zúžena. Finální vzhled e-mailu lze vidět na obrázku 5.10.

1. UI (user interface) - uživatelské rozhraní



Obrázek 3.4: Koncept rekurentního notifikačního e-mailu

Popis konceptu notifikačního e-mailu

1. *Vrchní panel*. Obsahuje informace o jaký typ e-mailu se jedná. Tato oblast funguje jako rychlý rozpoznávací element. Objevuje se u obou typů.

2. *Hlavní panel*. Nejdůležitější oblast notifikačního e-mailu. Popisuje všechny informace o události. Tato oblast se mění v závislosti na typu notifikace. Oblast 2.1 určuje o jaký typ notifikace půjde spolu s barevným odlišením. Podrobné informace o události např. název události, datum s časem a popis jsou zahrnuty v oblasti s číslem 2.2. Jedná-li se o rekurentní událost, obsahuje ještě navíc 2 tlačítka pro vykonání akce. Modré tlačítko s nápisem `P` referuje k odložení události (Postpone) a zelené s písmenem `D` (Done) reprezentuje, že daný úkol byl dokončen.

3. *Spodní panel*. Tato oblast je stále stejná a vyskytuje se vždy. Informuje uživatele o automaticky vygenerovaném e-mailu. Často obsahuje odkazy na samotnou aplikaci nebo sociální sítě dané aplikace.

4 Použité technologie při vývoji

V této kapitole jsou popsány technologie použité při samotné implementaci aplikace. Po konzultaci s vedoucím práce bylo dohodnuto, že webová aplikace bude naprogramována v jazyce PHP. Používám jednu z nejnovějších verzí PHP 7.1.3.

Pro efektivní programování byl zvolen framework Laravel, který je určen pro vývoj webových aplikací. Tento framework popisuji v první podkapitole. Následující kapitoly se zabývají výběrem databáze a nástroji pro tvorbu frontendu.

4.1 Laravel Framework

Laravel Framework (dále Laravel) je open source PHP framework vytvořený v roce 2012 Taylorem Otwellem [1]. Framework je zaměřený na vývoj webových aplikací s architekturou Model-view-controller (MVC). V mé aplikaci využívám Laravel ve verzi 5.7. Framework je licencován pod licencí MIT [9].

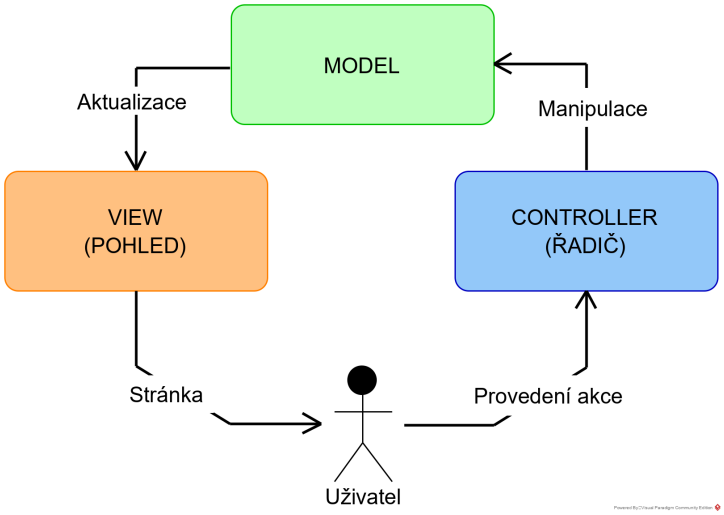
Laravel je dnes jeden z nejpopulárnějších a nejvyužívanějších PHP frameworků na světě. Ze statistiky pro nejlepší PHP framework v roce 2019 [10] obsadil první místo (43,7 %). Pomáhá vývojářům v tvorbě robustních webových aplikací. Tento framework využívá mnoho ověřených metod, tzv. best practices, pro řešení problémů.

4.1.1 Model-view-controller

MVC je architektura, která rozděluje aplikaci do 3 komponent. **Model** je základním prvkem celé aplikace. Reprezentuje data uložené v databázi, s nimiž aplikace manipuluje. **View** neboli pohled slouží jako prezentační vrstva. Ukazuje uživateli data ve vhodné podobě. **Controller** neboli řadič reaguje na vstup uživatele nebo událost. Upravuje předešlé 2 komponenty (Model, View).

Koncept MVC představuje cyklus mezi těmito komponentami. Uživatel provede akci v uživatelském rozhraní. Řadič zachytí informace o události (např. prostřednictvím http požadavku) a patřičně zaktualizuje model, který se týká vykonané akce. Model představuje doménovou vrstvu v tabulce databáze. Následně pohled zobrazí uživateli

zaktualizovaná data. Samotná data získává pohled přímo z modelu. Čekáním na další akci se cyklus znovu opakuje. Pro snadnější pochopení je MVC znázorněno diagramem na obrázku 4.1.



Obrázek 4.1: Diagram architektury Model-view-controller

4.1.2 Nejlepší vlastnosti Laravelu

Laravel poskytuje řadu funkcionalit, které samotnému vývojáři usnadní práci. Následně jsou popsány jeho nejlepší vlastnosti. [11]

Template Engine umožňuje vytvářet skvělé webové rozložení pomocí vestavěných šablon. Tento nástroj se nazývá **Blade**. Dovoluje používat čistý PHP kód v aplikačních pohledech. Primárními výhodami jsou dědičnost a direktivy pro rozšíření šablon (section a yield).

Artisan je nástroj pro příkazový řádek, který umožňuje generovat části programu s připraveným kódem, které běžný vývojář musí sám vytvářet. Dovoluje generovat strukturu databáze, migrace, základní MVC soubory a mnoho dalších.

Eloquent ORM[1] je programové API[2], které poskytuje aktivní manipulaci s databází. Každá tabulka v databázi má specifický model,

1. Object Relational Mapping - překládání dat mezi relační databází a objektově orientovaným programovacím jazykem
2. Application Programming Interface - rozhraní pro programování aplikace

skrze kterého s ní komunikuje. Komunikace probíhá použitím samotného jazyka PHP, aby nebylo nutné využít čisté SQL dotazy. Eloquent ORM také zaručuje bezpečnost těchto dotazů (ochrana před útoky *sql injection*). Prostřednictvím modelu získáváme data uložené v jeho korespondující tabulce. Tabulkové záznamy lze stejně tak přidávat nebo mazat.

Libraries nabízí předinstalované knihovny, které se neobjevují v ostatních PHP frameworcích. Nabízí mimo jiné knihovny pro autentizaci, CSRF[3] ochranu a resetování hesla.

Migration system má vlastnost rozšiřování již existujících databázových tabulek, bez potřeby tabulku znovu vytvářet. Díky tomu pak nedochází ke ztrátě dat. Migrace povolují i samotné vytváření nové tabulky.

Zabezpečení je jeden z klíčových prvků webové aplikace. Laravel sám poskytuje velmi kvalitní zabezpečení aplikace. Využívá například Bcrypt hašovací algoritmus pro lepší ochranu hesla a předpřipravené SQL výrazy.

Laravel využívá programovou utilitu *Composer*, která spravuje v projektu závislosti. Instaluje do projektu závislé knihovny, které jsou nutné ke správnému běhu aplikace. Samotná instalace Laravelu probíhá přes Composer.

Vue.js je JavaScriptový framework, který nabízí Laravel ve výchozím stavu. Používá se pro tvorbu dynamických responzivních webů.

4.1.3 Porovnání s jinými frameworky

Laravel v roce 2018 obsadil první příčku v kategorii Top 10 PHP frameworků [12], jako ve většině výzkumů. Mezi další kvalitní PHP frameworky můžeme zařadit např. Symfony, CodeIgniter, CakePHP nebo Yii. Následuje porovnání s frameworky Symfony a CodeIgniter.

Laravel vs. Symfony

Framework **Symfony** používá stejně jako Laravel MVC architekturu. Zatímco Laravel využívá svůj vlastní template engine (Blade), Symfony využívá nástroj Twig. Blade umožňuje znovupoužití kódu narozdíl

3. Cross-site Request Forgery

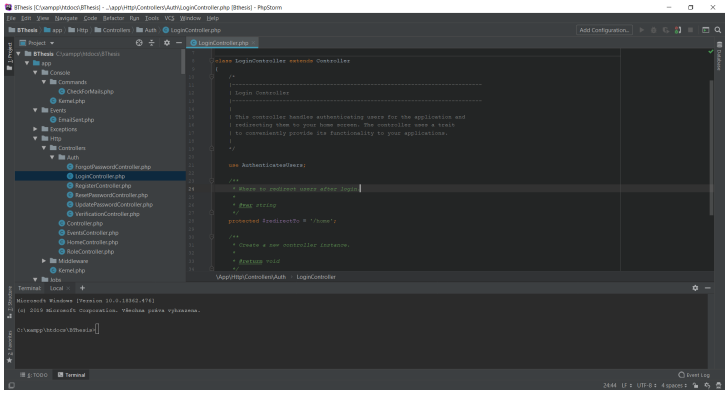
od nástroje Twig. Framework Symfony je při načítání stránek pomalejší a také obsahuje daleko méně naučných materiálů než Laravel. Na druhou stranu, pro vytváření rozsáhlých projektů je lepší použít framework Symfony, který je na to speciálně navržen. [13]

Laravel vs. CodeIgniter

Framework **CodeIgniter** nepodporuje objektově relační mapování (ORM). Povoluje však využití více databází než Laravel (např. ORACLE, Microsoft SQL Server). Nicméně samotná tvorba databáze je v Laravelu velice ulehčena pomocí migrací, kdežto v CodeIgniteru vyžaduje mnoho práce. Dále vyžaduje integraci frameworku pro vytváření šablon, protože sám žádný neposkytuje. [14]

4.2 Vývojové prostředí (IDE)

Pro efektivní implementaci aplikace jsem zvolil vývojové prostředí **PHPStorm**[4] od společnosti JetBrains. Se softwarem od JetBrains již mám zkušenosti z jiných předmětů (CLion, IntelliJ IDEA, PyCharm). IDE je velmi přívětivé a přehledné, zároveň povoluje integraci mnoha frameworků, včetně Laravelu. Díky studentské licenci [15] lze využít kterýkoliv software od JetBrains. Na obrázku 4.2 lze vidět vývojové prostředí PHPStorm se strukturou frameworku Laravel.



Obrázek 4.2: Vývojové prostředí PHPStorm

4. https://www.jetbrains.com/phpstorm/

4.3 MySQL

Databázi MySQL jsem zvolil na základě databáze, kterou Fakulta informatiky Masarykovy univerzity poskytuje. Pro samotný vývoj webové aplikace jsem použil open source multiplatformní softwarový balíček **XAMPP**[5]. Obsahuje v sobě nástroje Apache HTTP Server, databázi MySQL a jazyky PHP a Perl. Nástroj je licencován pod licencí GNU General Public License [16].

4.4 Nástroje pro tvorbu grafického rozhraní

Pro tvorbu webových stránek jsem použil značkovací jazyk HTML a jazyk CSS[6] pro vytváření stylů. Konkrétní verze jazyků jsou HTML5 a CSS3. Pro kvalitnější vzhled aplikace jsem využil komplexnější grafické nástroje viz následující podkapitoly.

4.4.1 Bootstrap 3

Boostrap je nejznámější open source frontendový framework pro vytváření responzivního webu. Tento web je vhodný i pro mobilní zřízení. Bootstrap je licencovaný pod MIT licencí [17].

Poskytuje šablony webových komponent založené na jazycích HTML, CSS, případně také JavaScript. Mezi komponenty patří např. tabulky, tlačítka, formuláře, webové navigace, ale také samotná typografie.

4.4.2 AdminLTE 2

AdminLTE je šablona pro celkové komponenty postavená na technologiích Bootstrap 3 a jQuery 1.11+. Jedná se opět o open source software s MIT licencí [18].

Nabízí předdefinované komponenty od jednoduchých boxů a tlačítek až po navigační panely, formuláře pro registraci a přihlášení uživatele.

5. https://www.apachefriends.org/index.html
6. Cascading Style Sheets - jazyk pro stylování HTML elementů

4.4.3 Vue.js

Jedná se o open-source JavaScriptový framework pro tvorbu uživatelských rozhraní s MIT licencí [19]. Zaměřuje se pouze na komponentu view u architektury MVC. Vue je především určeno pro vytváření malých, jednoduchých komponent.

V aplikaci jsem použil rozšíření **vue-datetime picker**[7]. Toto rozšíření slouží ke snadnému a responzivnímu vkládání data a času. Nástroj je taktéž licencován pod licencí MIT [20].

4.5 Ostatní

Při navrhování grafických prvků webové aplikace jsem použil grafický editor **Adobe Photoshop** [21] od firmy Adobe Systems. Tento software je určený pro tvorbu bitmapové grafiky. Photoshop byl využit ve zkušební verzi programu. UML diagramy v této práci byly vytvořeny pomocí nástroje **Visual Paradigm** [22]. Software je licencován pod studentskou licencí Masarykovy univerzity.

7. https://github.com/mariomka/vue-datetime

5 Implementace

V následující kapitole popisuji proces implementace webové aplikace, který je doplněn o krátké ukázky kódu. Implementace je rozdělena na serverovou a klientskou část. Zabývá se přiblížením základních implementačních problémů a vlastností systému.

Webová aplikace prozatím nebyla spuštěna na žádném serveru, tudíž se implementace nezabývá samotným nasazením systému. Ovšem do budoucna by mohla být spuštěna na fakultním serveru Masarykovy univerzity.

5.1 Serverová část aplikace

Serverovou část můžeme taky nazvat backend. Obsahuje veškerou skrytou logiku webové aplikace.

5.1.1 Struktura projektu

Základní adresářová struktura frameworku Laravel je velmi složitá a obsahuje nespočet souborů. Kvůli větší přehlednosti v projektu zde uvádím seznam základních adresářů s jejich cestami.

- Klíčová část naší aplikace se nachází v adresáři `\app`. V této složce se nachází třídy reprezentující jednotlivé modely (např. `\app\User.php`).
- Složka `\app\Http` obsahuje řadiče aplikace (`\app\Http\Controllers`) a middleware (`\app\Http\Middleware`) použitý při http požadavcích.
- Adresář `\app\Mail` obsahuje třídy reprezentující e-maily.
- V adresáři `\app\Jobs` se nachází třídy úloh, které se ukládají do tabulky **jobs**.
- Adresář `\config` obsahuje konfigurační soubory pro celý framework. Nachází se zde například soubory pro nastavení databáze nebo posílání e-mailů.

- Ve složce `\database` se nachází soubory pro vytváření a inicializaci databázových tabulek. Zejména jsou zde soubory všech migrací (`\database\migrations`).
- Adresář `\routes` obsahuje soubory potřebné ke směrování aplikace. Nachází se zde soubor `web.php`, který definuje všechny cesty pro naše webové rozhraní.

5.1.2 Směrování a Middleware

Směrování (**Routing**) nám překládá adresy URI[1] na řadiče s konkrétním zavoláním metod. Na základě vstupu uživatele určuje vyvolání reakce.

Všechny přístupové cesty naší webové aplikace jsou definovány v souboru `web.php`. Vytvořené cesty v tomto souboru jsou připojeny k middleware skupině `web`.

Při samotné registraci určujeme HTTP požadavek cesty, její URI a příslušný řadič s metodou, která příkaz zpracuje. Následující tabulka uvádí příklad definice struktury registrování jednotlivých cest.

Tabulka 5.1: Princip vytváření cest

URI	HTTP	řadič s funkcí	popis
/create	GET	EventsController-create()	formulář události
/events	GET	EventsController-index()	seznam událostí
/events	POST	EventsController-store()	uloží novou událost
/events	DELETE	EventsController-destroy()	odstraní událost

Frameworkem byly samostatně vygenerovány cesty pro registraci a přihlášení uživatelů, které jsem rozšířil o změnu hesla a e-mailu. Dalšími cestami jsou aktualizace (úpravy) již existujících událostí nebo zobrazení uživatelských nastavení. Velmi speciálními případy jsou pak cesty pro odkládání a dokončení události. Při zadání nedefinované cesty se uživateli zobrazí stránka s chybou.

1. URI - jednotný identifikátor zdroje

Následující ukázka kódu reprezentuje cesty pro zobrazení stránky s formulářem události (metoda GET) a její samotné vytvoření (metoda POST).

Kód 5.1: Ukázka cest v souboru web.php

```
//struktura volani metody radice
//controller@method

Route::get('/create', 'EventsController@create');
Route::post('/events', 'EventsController@store');
```

Middleware poskytuje filtrování HTTP požadavků, které chtějí vstoupit do aplikace.

Laravel již od základu nabízí middleware pro ověření uživatele (*Authentication* middleware). HTTP požadavek se dokončí, pokud je uživatel úspěšně ověřen. V opačném případě je uživatel přesměrován na obrazovku pro přihlášení.

Laravel dále nabízí middleware k ochraně proti *CSRF* útokům. Jedná se o útok pomocí nezamýšleného požadavku pro vyvolání akce ve webové aplikaci.

Obě middleware spadají pod middleware skupinu *web*. Tato skupina se pak volá na ověření HTTP požadavků v naší aplikaci.

5.1.3 Registrace a přihlášení

Uživatel může pracovat s událostmi až po vytvoření účtu a přihlášení do systému. Pokud není přihlášen a pokusí se vstoupit do aplikace (pomocí adresy URL), je přesměrován zpět na obrazovku pro přihlášení (middleware 5.1.2).

Framework tuto problematiku vývojářům velice ulehčuje. Pomocí příkazu `php artisan make:auth` vygeneruje do projektu všechny potřebné soubory pro registraci a přihlášení uživatelů. Vytvoří složku `..\Controllers\Auth` s potřebnými řadiči, definuje potřebné cesty v souboru `web.php` a middleware pro ověření. Zároveň vytvoří model uživatele (třída `User`) s migrací a všechny potřebné pohledy pro jednotlivé situace. Nastavení pro autentizaci se nachází v souboru `config/auth.php`.

Řadič `RegisterController` se zabývá vytvořením nového uživatele. `LoginController` definuje samotné přihlášení a ověření správnosti zadaných přihlašovacích údajů. Resetování hesla probíhá odesláním linku na e-mail uživatele. Tento odkaz přesměruje uživatele do rozhraní, ve kterém lze heslo změnit bez ověření. Celou logiku resetování hesla řeší řadiče `ResetPasswordController` a `ForgotPasswordController`.

Laravel pro proces autentizace používá strážce (*guard*) a poskytovatele (*provider*). Strážce ověřuje uživatele při provedení požadavků pomocí tzv. *session guard* a ukládá jeho stav pomocí uložiště relací a souborů cookies. Poskytovatelé určují, jakým způsobem jsou získávána data uživatele z databáze aplikace. V naší aplikaci se data získávají pomocí Eloquentu ORM.

Ověření uživatele při požadavku probíhá následovně. Strážce (guard) zkontroluje vlastnosti uživatele získané z relace (session) a porovnává je s daty uložené v databázi, které mu předá poskytovatel (provider). Při úspěšném ověření dojde ke zpracování požadavku.

Z pohledu databáze výše zmíněná funkcionalita vyžaduje tabulku pro ukládání uživatelů se všemi vlastnostmi a tabulku pro resetování hesel, kde se vytváří záznamy s ověřovacími tokeny.

Změna e-mailu a hesla v API

Přihlášený uživatel je schopen v rámci svého uživatelského rozhraní změnit svůj e-mail i heslo. Tato funkcionalita je zcela běžná v jiných aplikacích, proto jsem se rozhodl ji implementovat.

V případě absence těchto možností by uživatel vůbec nebyl schopen nastavit novou e-mailovou adresu a pro změnu hesla by musel použít odkaz pro jeho resetování.

O změnu hesla se stará řadič `UpdatePasswordController`, který heslo zaktualizuje, pokud jsou splněny dva následující požadavky. Nové heslo musí splňovat základní parametry pro vytvoření a uživatel musí prokázat svoji totožnost zadáním svého současného hesla.

Kód 5.2: Ověření současného hesla a uložení nového

```
if(Hash::check($request->old, $hashedPassword)){
```

```
    $user->fill([
      'password' => Hash::make($request->password)
    ])->save();
}
```

Řadič `UpdateEmailController` pak provádí podobným způsobem aktualizaci e-mailu. Pro změnu e-mailu pouze stačí ověřit totožnost heslem.

5.1.4 Databáze

Databázové tabulky byli vytvářeny prostřednictví migrací a příkazem `php artisan migrate`. Každá migrace obsahuje metody `up` a `down`, které slouží pro vytvoření a stažení tabulky. Při vytváření migrace definujeme tabulku pro model, který se zde bude ukládat. Z tohoto důvodu migrace obsahují vlastnosti modelu a jejich omezení (typy, primární klíč, nulovatelné typy, ...). Všechny tabulky byly popsány při návrhu databáze 3.2.

Kód 5.3: Schéma pro vytvoření neregistrovaného uživatele

```
Schema::create('unregistered_users',
                function (Blueprint $table)
    {
        $table->increments('id');
        $table->string('email')->unique();
        $table->timestamps();
    });
```

Pro získání dat z databáze a jejich úpravu jsem využil nástroj Eloquent ORM. Prostřednictvím modelů lze vytvářet intuitivní a bezpečné dotazy nad tabulkami.

Kód 5.4: Ukázka ORM přístupu

```
//ziskani uzivatele podle e-mailu
$user = User::where('email', $mail)->first();
//vytvoreni noveho uzivatele
$newUser = new User();
$newUser->name = 'Daniel␣Jurca';
$newUser->mail = 'daniel.jurca@seznam.cz';
$newUser->save();
```

5.1.5 Role

Pro nutnost rozlišení uživatelů bylo potřeba vytvořit role, které definují jejich funkce v aplikačním rozhraní. Rozlišujeme role běžného uživatele a skupinového administrátora. Na základě role se uživateli zobrazují příslušné pohledy.

Implementace logiky se nachází v řadiči `RoleController`. Řadič obsahuje metody pro správné zobrazení pohledů a upravování uživatelských rolí.

Určování rolí

Při registraci nového uživatele je jeho role nastavena na běžného uživatele (**User**). Nabude tak všech funkcí, které příslušná role povoluje.

Na samotném začátku aplikace existuje jeden uživatel, který má roli skupinového administrátora (**Group Administrator**). Tento administrátor může běžným uživatelům nastavovat role, a tak jim poskytnou stejnou funkcionalitu, kterou má on sám. Zároveň mu nikdo nemá právo změnit jeho roli.

5.1.6 Události

Události můžou být dvou typů, jednorázová a rekurentní. Systém povoluje vytvářet události pouze registrovaným uživatelům. Na základě uživatelské role je možné vytvářet události pouze sám pro sebe nebo skupinu uživatelů.

Řadič `EventsController` se zabývá celkovou manipulací s událostmi. Mimo jejich vytváření a úpravu, obsahuje dále metody k propojení uživatelů s událostmi a generování nových rekurentních událostí na základě časové periody.

Jednorázová a rekurentní událost

Jednorázová událost (**One-time**) se pošle pouze jednou. Běžný uživatel nastavuje u této události pouze název, datum a čas doručení a její stručný popis. V případě, že událost vytváří admin, respektive uživatel s rolí *Group Administrator*, může navíc nastavit i další příjemce události. Tito příjemci se definují prostřednictvím e-mailových adres.

Rekurentní událost (**Recurrent**) obsahuje všechny vlastnosti jednorázové události. Jediným rozdílem je možnost nastavit opětovné zasílání události, pokud ji uživatel odloží. Opakování události lze nastavit dvěma způsoby. Opakování pouze určitý den v měsíci nebo za uplynulou jednotku času.

Kód 5.5: Vytvoření události One-time

```
$event = new Event();
$event->title = request('title');
$event->type = request('type');
$event->date = request('date');
$event->description = request('description');
$event->save();
```

Vytvoření události a propojení s uživatelem

Proces vytvoření události začíná vyplněním formuláře a jeho potvrzením. Následně se spustí požadavek POST, který zpracuje metoda `store()` v řadiči `EventsController`. Tato funkce ověřuje předaná data a ukládá událost do databáze.

Až po úspěšném uložení události do databáze se vytváří vztahy k příjemcům. Ti jsou předáni jako jeden řetězec e-mailových adres. Pokud žádný registrovaný uživatel se zadaným e-mailem neexistuje, prochází se tabulka pro neregistrované uživatele. První uživatel se shodnou e-mailovou adresou je připojen k události. Pokud se žádný uživatel nenalezne, vytvoří se jako nový neregistrovaný uživatel v tabulce ***unregistered_users*** a připojí se k události.

Kód 5.6: Ukázka připojení události k uživateli

```
$event; //Ulozena udalost
$user;  //Registrovany uzivatel
$event->users()->attach($user);
```

Metody potřebné pro úpravu a odstranění událostí se také nachází v řadiči `EventsController`. Aktualizace události probíhá téměř identicky jako samotné vytváření. Při odstraňování události se musí nejprve přerušit všechny vazby s připojenými uživateli, teprve poté je možné událost bezpečně odstranit.

5.1.7 Plánování úloh

Pro odesílání notifikačních e-mailů ve správný čas bylo potřeba vytvořit úlohu, která bude události v databázi vyhledávat a následně zpracovávat. Laravel nabízí vytvoření vlastního konzolového příkazu, který se bude v rámci naší úlohy opakovaně volat.

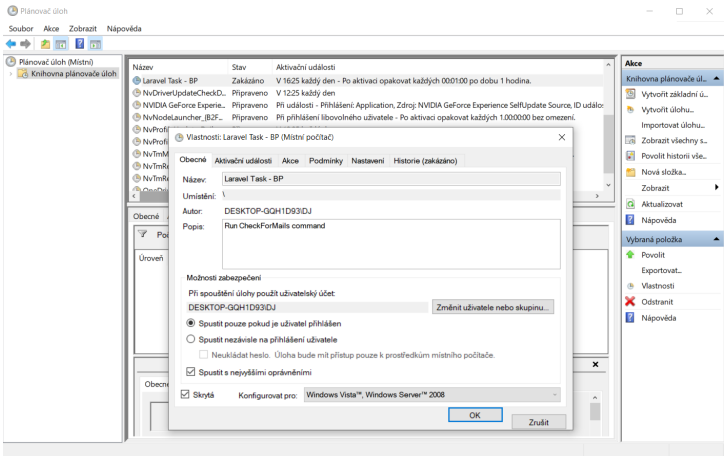
Implementoval jsem příkaz `CheckForMails`, jehož úkolem je vyhledávat události v tabulce ***events***, které se musí v daný okamžik odeslat. Události jsou vybírány na základě porovnání s aktuálním časem (s přesností na minuty).

Kód 5.7: Kód pro vyhledávání událostí

```
$events = Event::where('date', '=',
                Carbon::now('Europe/Prague')
                ->format('Y-m-d␣H:i:00'))
                ->get();
```

V momentě nalezení události se vytvoří úloha (`SendMailJob` viz další kapitola) pro odeslání e-mailu. Jedná-li se o rekurentní událost, vytvoří se její nová kopie s posunutým datem a časem. Nové datum a čas se určí podle rekurentního nastavení události.

Součástí příkazu také probíhá kontrola starších událostí. Při výpadku serveru může dojít k přeskočení událostí, které měly být v daném čase odeslány. Jestliže se takové události najdou, zpracují se a odešlou opožděně.



Obrázek 5.1: Plánovač s úlohou Laravel Task

Pomocí aplikace Plánovač úloh v OS Windows (obrázek 5.1), jsem vytvořil úlohu **Laravel Task**, která každou minutu spustí tento příkaz (`php artisan schedule:run`).

5.1.8 Generování e-mailu

Tato podkapitola vysvětluje princip odeslání e-mailu od spuštění příkazu `CheckForMails`.

Laravel Queues API nám poskytuje využít fronty pro zpracování úloh. Tuto úlohu nelze zaměnit za plánovanou úlohu (vytvořenou aplikací Plánovač úloh). Framework ji nazývá **job**.

Implementoval jsem úlohu (job) `SendEmailJob`, která odesílá událost zadanému příjemci a jejíž instance se vytváří v průběhu příkazu `CheckForMails`. Pro vytvoření úlohy `SendMailJob` je potřeba dvou parametrů, uživatele (příjemce) a událost.

Následujícím krokem je úlohu spustit, čímž se dostane do fronty (do již vytvořené tabulky ***jobs***). Nastavení fronty je možné upravit v souboru `config\queue.php`.

Kód 5.8: Odeslání úlohy do fronty

```
$job = (new SendEmailJob($event, $user))
dispatch($job);
```

Úlohy ve frontě se začnou vykonávat až po vytvoření a spuštění pracovníka (`php artisan queue:work`). Pracovník (**worker**) je proces, který se stará o obsluhu úloh zařazených ve frontě. Vybere z fronty událost principem FIFO (z anglického First In, First Out) a zpracuje ji. Zpracování úlohy probíhá zavoláním její metody `handle()`, která e-mail odešle.

Kód 5.9: Odeslání e-mailu

```
Mail::to($this->user)
->send(new EventNotification($this->event,$this->user));
```

Třída `EventNotification` nám umožňuje předat informace o příjemci a události koncovému pohledu. Ten vygeneruje e-mail do uživatelské podoby.

Laravel poskytuje několik možností, jak e-maily posílat. Vlastnosti e-mailu lze nastavit v příslušném konfiguračním souboru `mail.php`. Pro přenos elektronické pošty jsem zvolil protokol **SMTP** (Simple Mail Transfer Protocol).

SMTP [23] je komunikační protokol pro přenos zpráv elektronické pošty. Zajišťuje přímé spojení mezi odesílatelem a příjemcem. Server, který posílá e-mail pomocí smtp, používá pro navázání spojení protokol TCP[2] (port 25). Velmi často se využívá spolu s protokoly POP nebo IMAP.

5.1.9 Odložení a dokončení události

Funkcionalita pro akce prováděné prostřednictvím e-mailu se nachází v řadiči `BacktrackingController`.

Při propojování událostí s jejich příjemci (tabulky ***event_user*** a ***event_unregistered_user***), dochází k vytváření ověřovacích tokenů pro akce *Postpone* a *Done*. Jedná-li se o jednorázovou událost, akce *Postpone* a *Done* nejsou k dispozici, a tudíž se tokeny nenastavují (jsou nulové).

Při vytváření e-mailu se nastaví odkazy s ověřovacími tokeny na příslušná tlačítka. Stiskem tlačítka (Postpone nebo Done) se vyvolá GET požadavek s tokenem. Předaný token se porovná s vygenerovaným tokenem v databázi. V případě, že se tokeny shodují, zavolají se metody `postponeWithToken` (odložení události) nebo `doneWithToken` (dokončení události). Při dokončení události se přeruší vztah mezi uživatelem a událostí.

Kód 5.10: Získání ověřovacího tokenu uživatele pomocí id

```
$actualDoneToken = $event->users()
                        ->where('user_id', $user->id)
                        ->firstOrFail()
                        ->pivot->done_token;
```

5.2 Klientská část

Tato kapitola se zabývá frontendem webové aplikace. Obsahuje charakterizaci pohledů se systémem šablon (**Blade**) a grafické ukázky z aplikace. Dále popisuje prezentaci dat v uživatelském rozhraní.

2. Transmission Control Protocol

5.2.1 Pohledy

Celé rozhraní webové aplikace bylo vytvářeno v pohledech (dále také view) za podpory Blade šablon. Blade podporuje používání čistého jazyka PHP. Zároveň disponuje například speciální deklarací direktiv pro rozšiřování pohledů, vytváření podmínek a cyklů. Všechny pohledy pro aplikaci se nachází v souboru `\resource\views`.

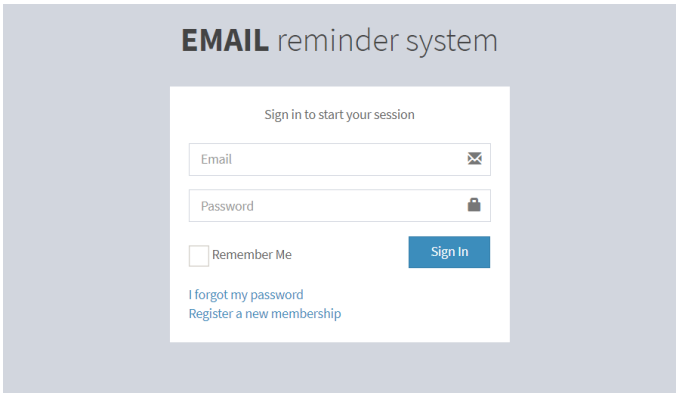
Kód 5.11: Ukázka cyklu foreach

```
@foreach($my_one as $event)
    <tr><td>{{ $event->title }}</td>
        <td>{{ $event->date }}</td>
        <td>{{ $event->description }}</td></tr>
@endforeach
```

5.2.2 Aplikace z pohledu uživatele

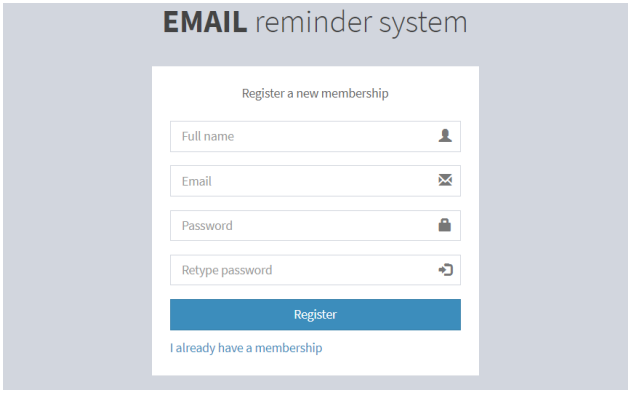
Přístup do systému

Pro přístup do naší aplikace musí být uživatel zaregistrován a přihlášen. Pokud není uživatel přihlášen, je přesměrován zpátky na stránku s pohledem `login.blade.php` (obrázek 5.2).



Obrázek 5.2: Stránka pro přihlášení uživatele

Pokud uživatel nemá vytvořený účet, po kliknutí na odkaz *Register a new membership* jej aplikace přesměruje na stránku pro registraci nového uživatele (obrázek 5.3). Oba tyto pohledy byly vytvořeny pomocí šablony AdminLTE.

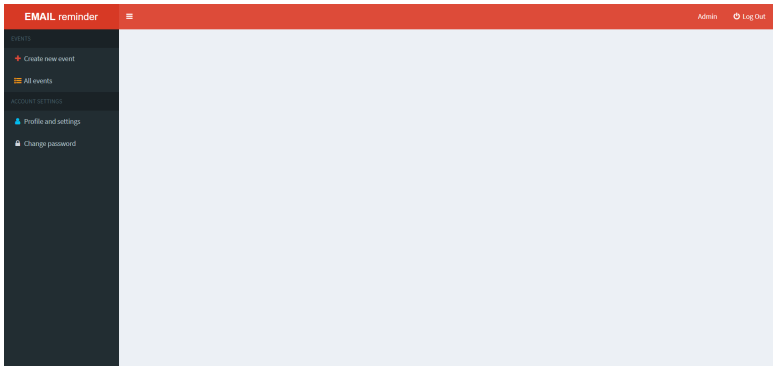


Obrázek 5.3: Registrace uživatele

Na stránce pro přihlášení je také možné resetovat heslo uživatele. Lze proto využít odkaz *Forgot my password*, kde můžeme heslo resetovat.

Uživatelské rozhraní

Po úspěšném přihlášení aplikace uživatele přesměruje na domovskou stránku (`\home`). V průběhu práce se na tuto stránku aplikace přesměruje i kliknutím na logo v levém horním rohu. Nachází se zde navigační panely šablony AdminLTE, které se objevují na každé stránce uvnitř aplikace. Stránka je zobrazena na obrázku 5.4.

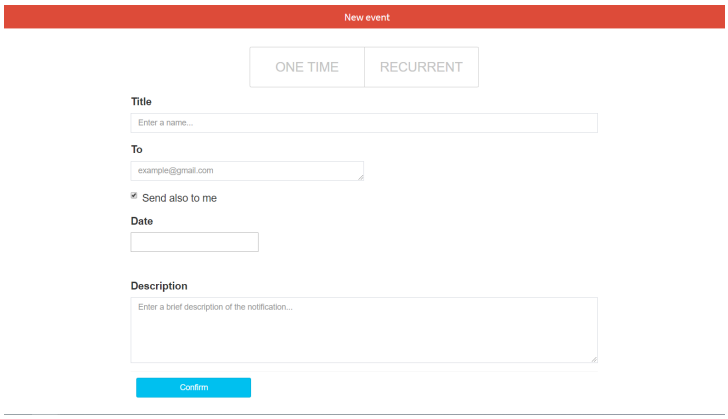


Obrázek 5.4: Navigační panely aplikace

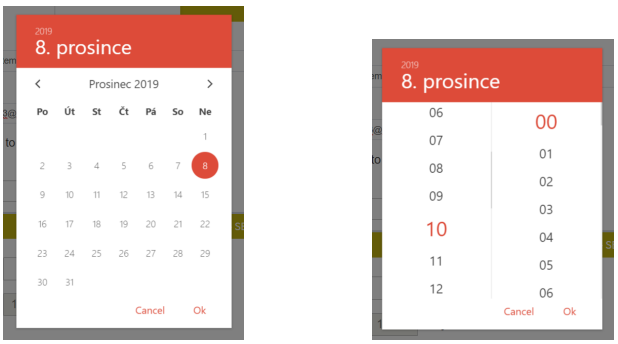
Horní navigační lišta nabízí pouze jméno přihlášeného uživatele a tlačítko pro odhlášení. Když se uživatel z aplikace odhlásí, je přesměrován zpět na stránku pro přihlášení.

Boční panel je možné skrýt. Obsahuje navigace do ostatních částí aplikace (Create new event, All events, Profile and settings, Change password).

Create new event přesměruje uživatele na stránku s pohledem `createEvent`. Nachází se zde formulář pro vytvoření jednorázové i rekurentní události. Uživateli s rolí *Group Administrator* se navíc zobrazí pole pro definování více příjemců (viz obrázek 5.5). Pro výběr datumu a času jsem použil rozšíření **Vue-datetime picker** (viz obrázek 5.6).

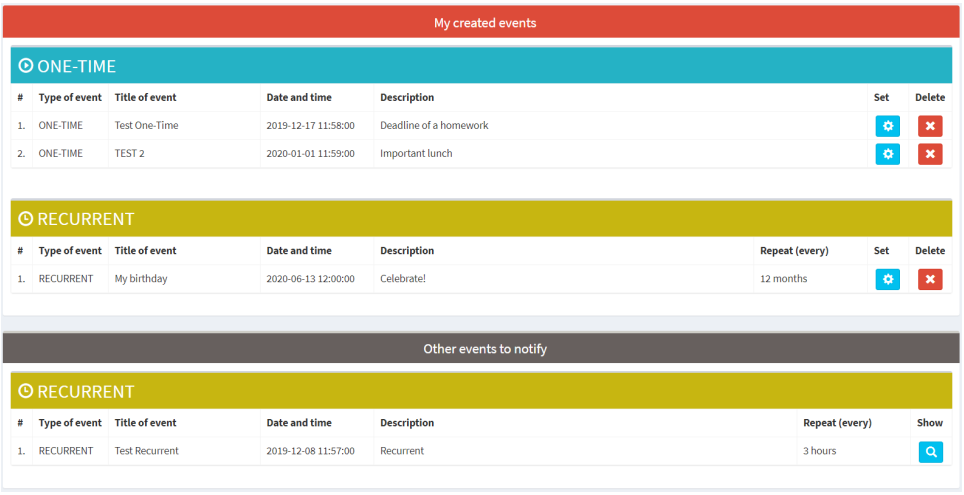


Obrázek 5.5: Formulář pro vytvoření události (Group Administrator)



Obrázek 5.6: Vue-datetime picker

All events reprezentuje stránku pro zobrazení všech událostí. Obsah stránky je rozdělen na seznam vlastních událostí a událostí, které byly pro uživatele vytvořeny jinou osobou. Vlastní události se dají upravovat nebo odstranit. Cizí události lze pouze rozkliknout pro podrobnější zobrazení.



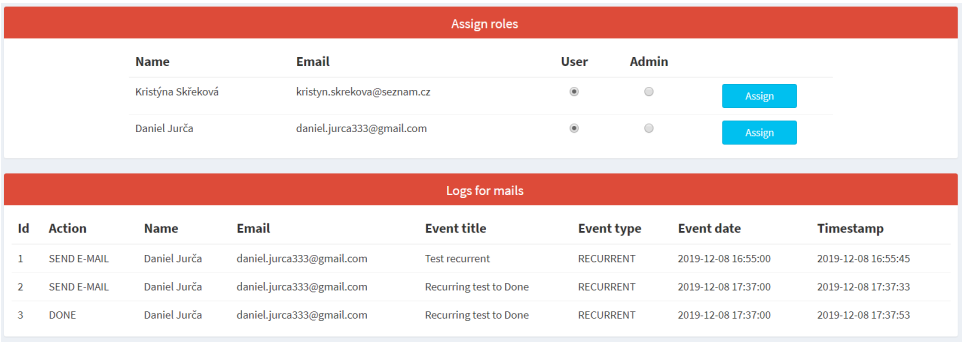
Obrázek 5.7: Události připojené k uživateli

Ukázka kódu 5.12 zobrazuje formulář pro odstranění události metodou DELETE v pohledu `allEvents.blade.php`. Po úspěšném odstranění se stránka zaktualizuje.

Kód 5.12: Formulář pro odstranění události

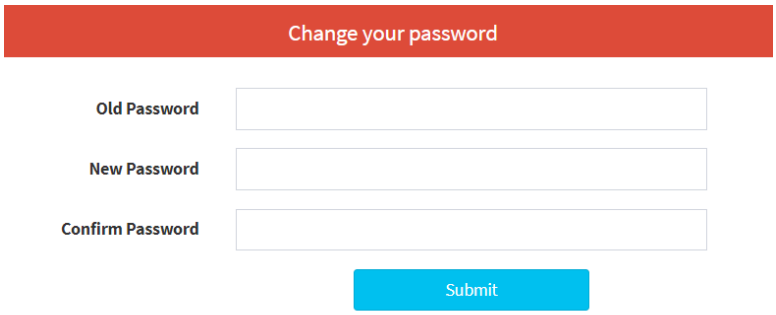
```
<form action="{{route('event.destroy',$event)}}" method="post">
    @method('DELETE')
    {{ csrf_field() }}
    <button class="..." type="submit"></button>
</form>
```

Profile and Settings zobrazí pro standardního uživatele možnost měnit si svoji e-mailovou adresu. Administrátorovi navíc zobrazí rozhraní pro přidělování rolí registrovaným uživatelům a záznam logování odeslaných, odložených a dokončených e-mailů (viz obrázek 5.8).



Obrázek 5.8: Možnosti administrátora (logování a nastavení rolí)

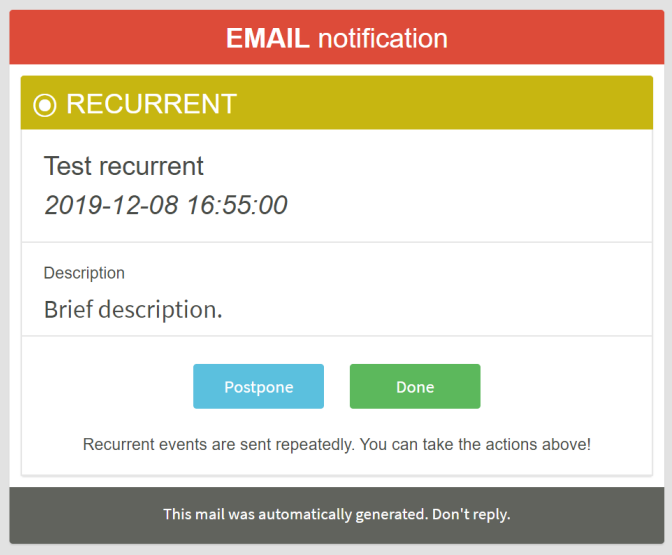
Change password přesměruje uživatele na stránku pro změnu hesla (obrázek 5.9). Heslo musí mít alespoň 6 znaků.



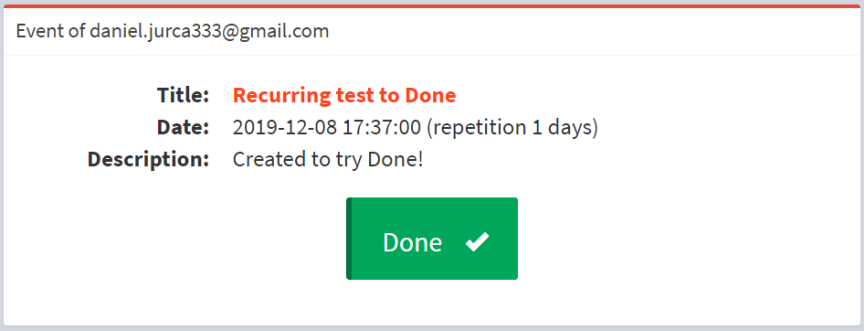
Obrázek 5.9: Formulář pro změnu hesla

5.2.3 Notifikační e-maily a backtracking

Upomínkové e-maily byly vytvořeny ve stejném stylu jako webová aplikace. Obsahují tlačítka pro odložení a dokončení události. Po stisknutí je uživatel přesměrován na obrazovku s informacemi o provedené akci. V případě, že se někdo pokusí uhádnout ověřovací token, je přesměrován na chybovou stránku pro zakázaný přístup.



Obrázek 5.10: Notifikační e-mail pro rekurentní událost



Obrázek 5.11: Obrazovka po dokončení události

6 Budoucí záměry

Do budoucna plánuji do aplikace ještě přidávat další funkcionalitu a upravovat tu stávající.

Prvním záměrem bude doladění detailů, aby bylo možné spustit webovou aplikaci na HTTPS serveru pro větší zabezpečení. Zřejmě k tomu poslouží server na Fakultě informatiky.

Další plánovanou změnou je vytvoření rozšíření pro manipulaci se skupinami uživatelů. Skupinový administrátor bude moci separátně vytvářet skupiny příjemců, které se budou s událostmi propojovat. Administrátor v samotném formuláři bude schopen interaktivně upravovat tuto cílovou skupinu. Responzivní prvky tohoto rozšíření budou implementovány v JavaScriptu (JavaScriptovém frameworku).

Plánuji také lépe optimalizovat zobrazení webové aplikace pro zařízení s menším rozlišením. Zejména se jedná o tablety a mobilní telefony.

7 Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci v jazyce PHP pro vytváření notifikací. Hlavní myšlenkou této notifikační aplikace je vytváření událostí, které se přenáší formou elektronické pošty. Systém nabízí manipulaci s událostmi a zároveň poskytuje správu uživatelských rolí. Administrátor má povoleno vytvářet události pro skupinu uživatelů bez ohledu na to, zda jsou nebo nejsou v aplikaci zaregistrovaní.

Struktura práce odpovídá záměru, který jsem popsal v úvodu této práce. Na začátku se tedy práce zabývá uvedením do problematiky notifikačních systémů, včetně popisu již existujících notifikačních nástrojů. Dále popisuji mou motivaci pro vytvoření práce a jakým způsobem jsem k vytváření aplikace přistupoval. Poté se zaměření práce již přesouvá přímo k popisu procesu vyvíjení aplikace se všemi náležitostmi.

Následující část se tak věnuje samotnému návrhu softwaru, kde jsou charakterizovány aplikační požadavky, návrh databáze a koncept uživatelského rozhraní. Databáze a požadavky jsou doplněny o příslušné diagramy. Ve čtvrté kapitole popisuji jednotlivé technologie použité při implementaci aplikace.

V poslední části se nachází proces samotné implementace, která se zaměřuje na základní implementační problémy takových systémů s ukázkami kódu a uživatelského rozhraní. Na závěr jsou v kapitole šesté popsány možná rozšíření a vylepšení aplikace do budoucnosti.

Snažil jsem se systém navrhnout tak, aby jej bylo možné dále rozšiřovat a zlepšovat, proto se počítá s jejím dalším vývojem a nasazením na server, kde bude k dispozici pro vyzkoušení.

Mým cílem, který jsem si v úvodu stanovil, bylo vytvoření jednoduché open source aplikace, jejíž hlavním posláním bude upomínání úkolů prostřednictvím notifikačních emailů. Tento cíl byl dle mého názoru v podstatné části splněn. Nicméně, jak už jsem uvedl, systém bych rád dále vylepšoval a více ho přibližoval k dokonalosti a naprosté uživatelské přívětivosti.

Jsem přesvědčen, že software je navržen tak, aby ho i méně zdatný uživatel mohl bez větších problémů využívat. Software tak může sloužit k uplatnění pro širokou veřejnost.

Bibliografie

1. *Laravel Manual* [online]. Taylor Otwell [cit. 2019-04-26]. Dostupné z: https://laravel.com/docs/.
2. DOSTALOVÁ, Zuzana. *Frontend vs. Backend* [online]. Czechitas z.s., 2014 [cit. 2019-04-26]. Dostupné z: https://www.czechitas.cz/cs/blog/zaciname-s-it/frontend-vs-backend.
3. *Centre for Research on Cryptography and Security* [online]. CRoCS [cit. 2019-11-21]. Dostupné z: https://crocs.fi.muni.cz/.
4. *Google Calendar API Terms of Service* [online]. Google, Inc., 2013 [cit. 2019-12-10]. Dostupné z: https://developers.google.com/calendar/terms.
5. *Terms of Use* [online]. Nudgemail [cit. 2019-12-10]. Dostupné z: https://www.nudgemail.com/terms-of-use/.
6. *Pricing & Sign up* [online]. Acuity Scheduling [cit. 2019-12-11]. Dostupné z: https://acuityscheduling.com/signup.php.
7. *Outlook* [online]. Microsoft Corporation [cit. 2019-12-11]. Dostupné z: https://www.microsoft.com/en-us/p/outlook/cfq7ttc0k7c4?activetab=pivot%5C%3aoverviewtab.
8. *Features (Pricing)* [online]. FollowUpThen, Inc. [cit. 2019-12-10]. Dostupné z: https://www.followupthen.com/features.
9. *Laravel MIT license* [online]. Laravel [cit. 2019-11-05]. Dostupné z: https://laravel-guide.readthedocs.io/en/latest/license/.
10. *Top 8 PHP Frameworks in 2019* [online]. Medium Corporation [cit. 2019-12-11]. Dostupné z: https://medium.com/hackernoon/top-8-php-frameworks-in-2019-b6be163605c8.
11. *Why To Go With PHP Laravel Framework?* [online]. Techcronus Business Solutions, 2018 [cit. 2019-12-10]. Dostupné z: https://www.techcronus.com/blog/go-php-laravel-framework/.
12. *Top 10 Must-Have PHP Frameworks in 2019* [online]. Dot Com Infoway, 2018 [cit. 2019-12-02]. Dostupné z: https://www.dotcominfoway.com/blog/top-10-must-have-php-frameworks-in-2019/#gref.

13. *Laravel vs Symfony* [online]. EDUCBA [cit. 2019-12-08]. Dostupné z: https://www.educba.com/laravel-vs-symfony/.

14. *Laravel vs Codeigniter* [online]. EDUCBA [cit. 2019-12-08]. Dostupné z: https://www.educba.com/laravel-vs-codeigniter/.

15. *TOOLBOX SUBSCRIPTION AGREEMENT FOR EDUCATION* [online]. JetBrains s.r.o., 2019 [cit. 2019-12-08]. Dostupné z: https://www.jetbrains.com/student/license_educational.html.

16. *About: The Licence* [online]. Apache Friends [cit. 2019-12-08]. Dostupné z: https://www.apachefriends.org/about.html.

17. OTTO, Mark; THORNTON, Jacob. *Bootstrap* [online]. GitHub, Inc, 2019 [cit. 2019-12-10]. Dostupné z: https://github.com/twbs/bootstrap/blob/master/LICENSE.

18. *AdminLTE* [online]. GitHub, Inc, 2018 [cit. 2019-12-10]. Dostupné z: https://github.com/ColorlibHQ/AdminLTE/blob/master/LICENSE.

19. YOU, Evan. *vuejs.org* [online]. GitHub, Inc, 2018 [cit. 2019-12-10]. Dostupné z: https://github.com/vuejs/vuejs.org/blob/master/LICENSE.

20. JUÁREZ, Mario. *vue-datetime* [online]. GitHub, Inc, 2017 [cit. 2019-12-10]. Dostupné z: https://github.com/mariomka/vue-datetime/blob/v1.x/LICENSE.

21. *Adobe Photoshop* [online]. Adobe Systems [cit. 2019-11-28]. Dostupné z: https://www.adobe.com/cz/products/photoshop.html.

22. *Visual Paradigm* [online]. Visual Paradigm International Ltd. [cit. 2019-11-29]. Dostupné z: https://www.visual-paradigm.com/.

23. *Simple Mail Transfer Protocol* [online]. Wikipedia, the free encyclopedia [cit. 2019-12-07]. Dostupné z: https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol.

A Elektronická příloha

Součástí této bakalářské práce je i elektronická příloha, která je veřejně dostupná v informačním systému MU (na https://is.muni.cz/th/bth3l/). Součástí archivu této přílohy je:

- Projekt se zdrojovým kódem aplikace.
- Přihlašovací údaje pro testování.
- Postup pro lokální instalaci webové aplikace.