VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE PRO SNADNOU SPRÁVU DEDLAJNŮ

WEB APPLICATION FOR SIMPLE MANAGEMENT OF DEADLINES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE KATEŘINA ŠMAJZROVÁ
AUTHOR

VEDOUCÍ PRÁCE prof. Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2019

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

21909

Studentka: **Šmajzrová Kateřina**
Program: Informační technologie
Název: **Webová aplikace pro snadnou správu dedlajnů**
Web Application for Simple Management of Deadlines
Kategorie: Uživatelská rozhraní
Zadání:

1. Seznamte se s problematikou tvorby webových aplikací; zaměřte se na problematiku velice jednoduchých a maximálně efektivních služeb.
2. Vyhledejte a analyzujte existující webové aplikace, řešící podobný problém.
3. Navrhněte funkčnost aplikace pro jednoduchou správu dedlajnů.
4. Prototypujte způsob interakce s aplikací a dílčí prvky uživatelského rozhraní a testujte je na uživatelích.
5. Navrhněte a implementujte řešenou aplikaci.
6. Testujte aplikaci na uživatelích a iterativně ji vylepšujte.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Jan Řezáč: Web ostrý jako břitva, Baroque Partners, 2014

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Tato práce řeší vytváření webové aplikace, která umožňuje jednomu správci snadnou správu jeho úkolů, které zadává svým spolupracovníkům či studentům. V následujícím textu je mimo jiné popsán návrh uživatelského rozhraní i databáze. Zvolený problém jsem vyřešila pomocí PHP frameworku Symfony, několik skriptů je napsáno v Javascriptu, k návrhu vzhledu byl využit CSS framework Bootstrap. Důležitou součástí této práce je uživatelské testování, data byla získávána především přes dotazníky, případně osobní konzultací, a výsledky jsou zde podrobně zpracovány. Po každé fázi testování došlo na základě získaných poznatků k vylepšení stávající verze, odstranění chyb a zpřehlednění uživatelského rozhraní. Hlavním přínosem aplikace je zjednodušení zadávání úkolů pro větší skupiny lidí, využijí ji třeba učitelé nebo vedoucí na táborech. Poskytne zadavateli úkolu rychlý přehled, kdo již dané zadání splnil, kdo ne a zda všichni dodrželi zadaný termín.

Abstract

This work solves the creation of a simple web application that allows one administrator to easily manage his tasks, which he enters to his colleagues or students. The following text describes the design of the user interface and the database. I had solved this problem using the PHP framework Symfony, several scripts are written in Javascript, CSS framework Bootstrap was used to design the website. An important part of this work is user testing, data was obtained mainly through questionnaires, or personal consultation, and the results are in this thesis. After each testing phase, the existing version was improved based on the obtained data, bugs were fixed and the user interface was improved. The main benefit of the application is the simplification of assigning task to larger groups of people, it can be used for example by teachers or camp leaders. It will provide a quick overview of who has already fulfilled the tasks.

Klíčová slova

webová aplikace, uživatelské rozhraní, použitelnost, webdesign, uživatelské testování, PHP framework Symfony, Bootstrap, správa dedlajnů, správa úkolů, CSS, Javascript, HTML/Twig, MySQL databáze

Keywords

web application, user interface, usability, webdesign, user testing, PHP framework Symfony, Bootstrap, management of deadlines, management of tasks, CSS, Javascript, HTML/Twig, MySQL database

Citace

ŠMAJZROVÁ, Kateřina. *Webová aplikace pro snadnou správu dedlajnů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Webová aplikace pro snadnou správu dedlajnů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana prof. Ing. Adama Herouta PhD. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

. . . . . .
Kateřina Šmajzrová
9. května 2019

Poděkování

Ráda bych podělovala prof. Ing. Adamu Heroutovi Ph.D. za cenné rady a vedení mé bakalářské práce. Také bych chtěla poděkovat všem uživatelům, kteří mi aplikaci pomohli prakticky testovat.

Obsah

Kapitola 1

Úvod

V této práci je podrobně popsán vývoj webové aplikace, která slouží ke správě úkolů s pevně daným termínem odevzdání. Zabývá se návrhem a implementací uživatelského rozhraní, databáze a uživatelským testováním.

Aplikace umožňuje uživateli zadat jeden úkol svým kolegům nebo studentům. Každému z nich přijde email, kde zaklikne, zda už úkol splnil. Pokud ano, tak se zadavateli úkolu okamžitě v přehledu objeví, že ten má hotovo. Celkově je aplikace navržena tak, aby se odlehčilo emailové schránce a uživatel nebyl zahlcený. Také mu poskytne rychlý přehled, jak to vypadá s daným zadáním, kolik lidí už na úkolu zapracovalo a komu je nutné poslat upomínku.

V první části práce jsou popsány použité technologie, je využit jazyk PHP konkrétně framework Symfony. Vzhled stránek využívá CSS framework Bootstrap, který je doplněn stylovacím souborem, kde jsou naprogramovány vytvořené prvky. Obě technologie jsou široce rozšířené. Obsahují podrobnou a přehlednou dokumentaci na webu a velké množství návodů. V této kapitole jsou také zmíněny aplikace podobné Deadlineru.

Druhá část popisuje návrh aplikace, uživatelského rozhraní i databáze. Ta je zobrazena v ER modelech, dále je v ní zmíněn popis jednotlivých podstránek a návrh uživatelského testování. V třetí části se práce věnuje implementaci databáze v MySQL, podrobněji jsou popsány důležité třídy a použité komponenty a knihovny dostupné ve frameworku Symfony. Předposlední kapitola je věnována testování, jsou v ní zpracovány výsledky dotazníků z průběhu testování a opravy na základě připomínek testerů. V závěru jsou popsána možná rozšíření a vylepšení tohoto projektu, která mohou být v budoucnu implementována.

Kapitola 2

Současné webové aplikace a jejich technologie

Webové aplikace [12] jsou poskytované uživatelům webových prohlížečů (klientů) přes Internet (server), oblíbené jsou především pro svoji rozšiřitelnost napříč všemi operačními systémy bez nutnosti instalace. Implementují se v nich různé informační systémy, eshopy, hry i chatovací aplikace.

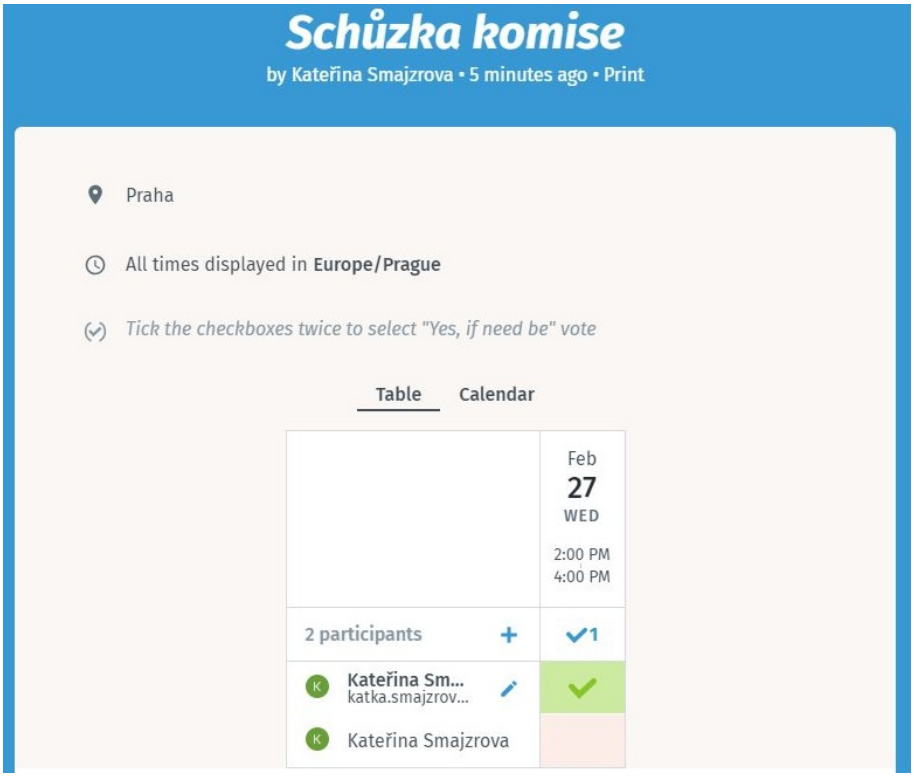
2.1 Příklady podobných aplikací a jejich řešení

Jedním z hlavních požadavků dnešní doby je mít webovou aplikaci, která je použitelná na počítači i na mobilu a usnadňuje práci při každodenních činnostech. Někteří uživatelé mají raději všechny služby a informace na jednom místě, jiní používají více aplikací, kde každá plní svůj **jedinečný** úkol stejně jako aplikace **Deadliner**.

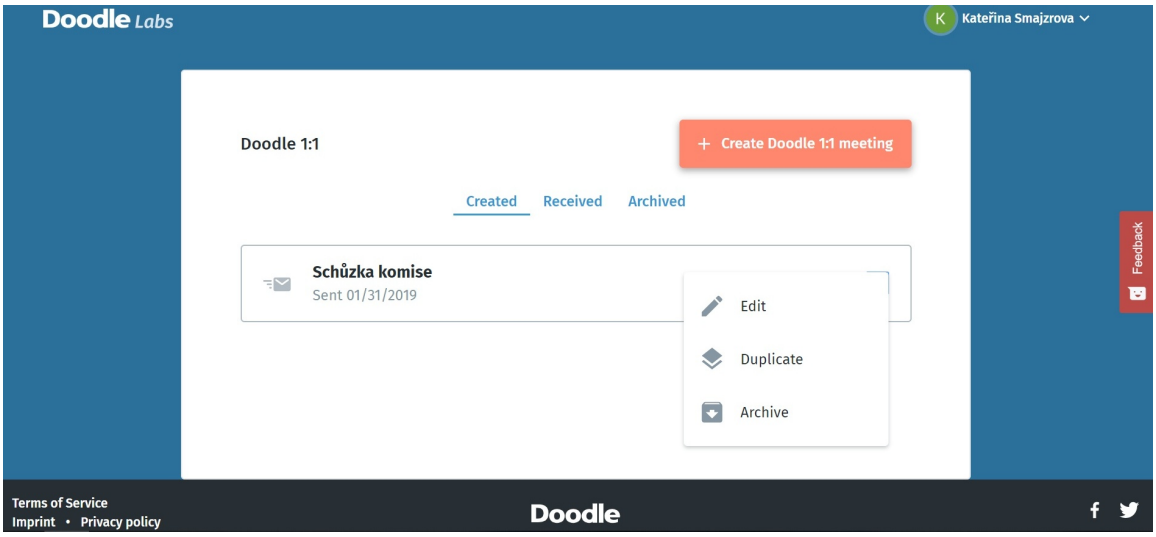
2.1.1 Doodle

Jednou z aplikací, které jsem zkoumala před návrhy uživatelského rozhraní, je Doodle[1], který slouží k plánování různých setkání s přáteli či kolegy. Je oblíbený hlavně díky jeho jednoduchému a intuitivnímu ovládání. Inspirovala jsem se v mnoha jeho funkcích, například v přihlášení. Když na stránky Doodle přijde návštěvník poprvé, nemusí se složitě registrovat a může se jednoduše propojit se svým Google či Facebook účtem. Samotné setkání je založeno během pár minut, nejdřív se přidají emailové adresy známých, kteří se mají účastnit setkání, poté je možné na své nástěnce sledovat, kdo přijde a komu se to nehodí. Všechny události jsou zobrazeny na přehledné nástěnce, viz obrázek 2.2. Pozvanému člověku přijde email s odkazem, kde může potvrdit či odmítnout svoji účast. Obrovskou výhodou je rychlost a přehlednost celé aplikace, není nutné posílat maily každému a poté složitě ve své emailové schránce hledat potvrzení účasti, to vše zajistí aplikace jako na obrázku 2.1. Důkazem, že podobné aplikace jsou oblíbené, je, že podle údajů v roce 2016 Doodle používalo více než 180 miliónů lidí po celém světě [1].

[1] https://doodle.com



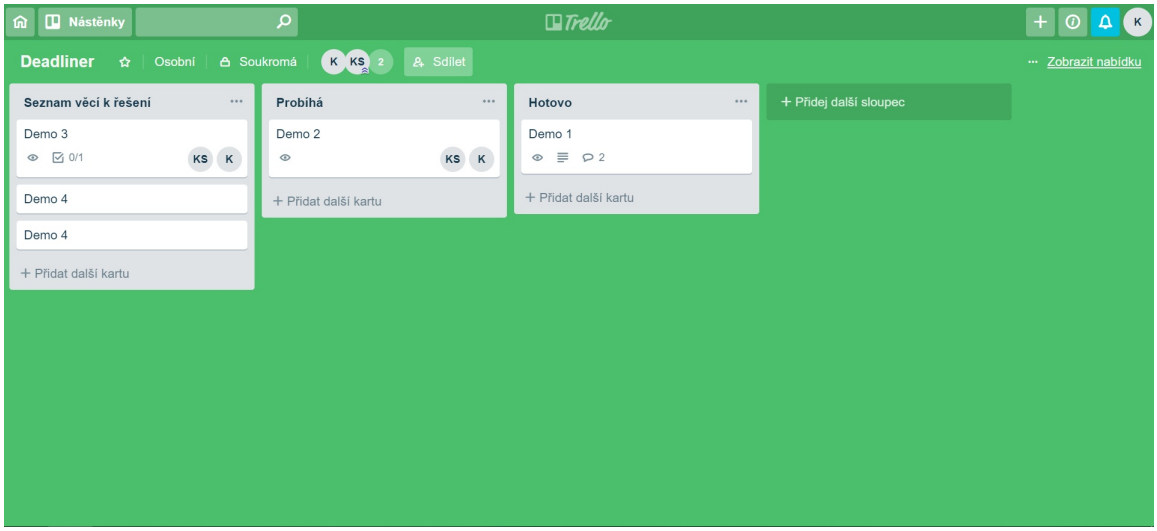
Obrázek 2.1: Ukázka nástěnky aplikace Doodle po odeslání úkolu, zatím setkání bylo potvrzeno jedním člověkem.



Obrázek 2.2: Ukázka seznamu setkání v aplikaci Doodle, po rozkliknutí je možné setkání upravovat, zkopírovat nebo smazat. Vše je navrženo, tak aby se uživatel zorientoval a přesně věděl, kam se jakou akcí dostane.

2.1.2 Aplikace na spolupráci týmů

Mezi podobné aplikace se dá zařadit aplikace Trello[2], která slouží spíše k pracovním účelům. Jeden uživatel vytvoří projekt, kam přidá své kolegy, postupně všichni mohou o daném úkolu diskutovat, přidávat další úkoly, viz obrázek 2.3. Důraz je opět kladen na rychlou a efektivní komunikaci, intuitivní ovládaní, ke kterému pomáhá třeba technologie Drag & Drop [5], v překladu táhni a pusť. Uživatel uchopí nějaký grafický prvek a pohybem myši ho přesune do jiné oblasti. Při úpravě úkolů se objeví vyskakovací okno, kam je možné přidat popis či komentář, seznam pod úkolů, přidat další členy nebo kopírovat úkol. Při návrhu aplikace jsem se snažila docílit podobného rozhraní jako v obrázku 2.4.



Obrázek 2.3: Obrázek zobrazuje jednotlivé slajdy s úkoly v aplikaci Trello, které je po rozkliknutí možné upravovat, případně komentovat.

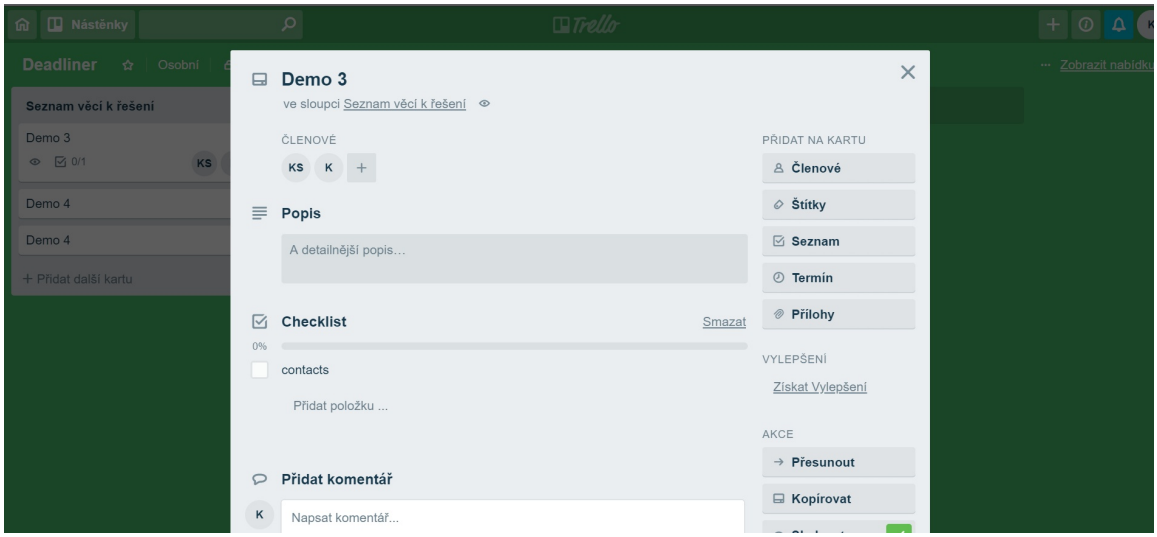
Objevila jsem ještě aplikaci ClickUp[3], která také slouží k vytváření úkolů a spolupráci týmů. Hned na začátku se objeví rychlý návod, jak aplikaci používat. Registrace také trvá pouze pár vteřin. Celkově mě aplikace zaujala právě kvůli použitelnosti, je přehledná a jednoduchá na ovládání. Nový úkol je vytvořen během momentu, poté se přidají lidé, kteří můžou začít diskutovat a pracovat. Je možné využít předgenerované často používané zadání, díky čemuž není nutné znovu vše vypisovat. Nástěnka, která zajišťuje přehledné a sjednocené zobrazení mnoha informací viz obrázek 2.5, je podobná třeba Trellu.

2.1.3 Další podobné aplikace

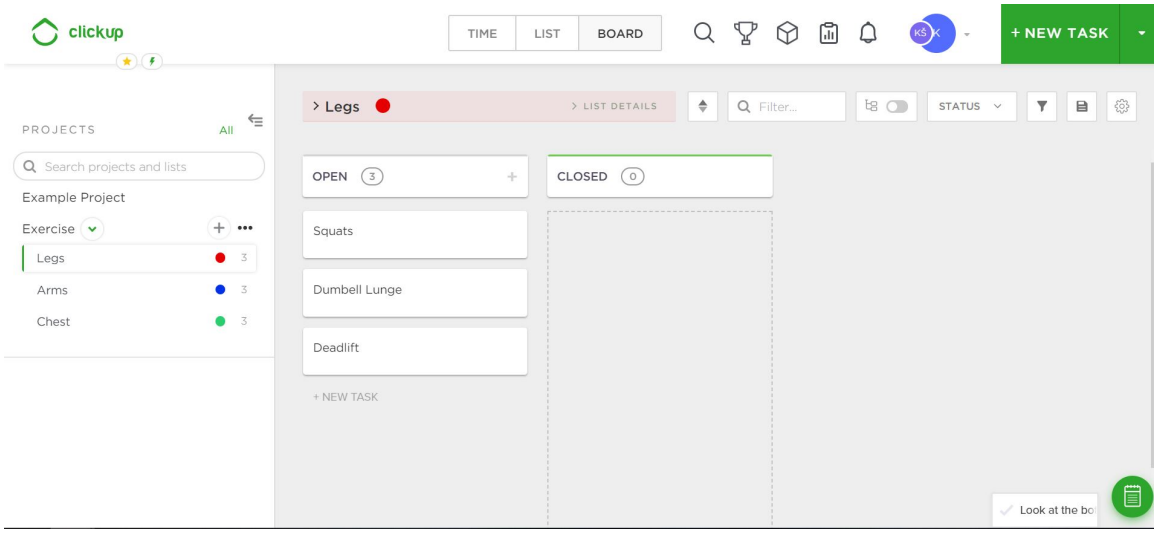
Vzhledem k tomu, že v Deadlineru je nutné také odesílat maily, tak jsem si prohlížela aplikace s tím související. Zmíním třeba Front[4], která slouží jako emailová schránka (Inbox) pro týmové projekty, kde se dá jednoduše diskutovat podobně jako v Trellu. Na běžné akce jako editaci či mazání se velmi často používají obecně známé ikonky jako v obrázku 2.6.

Aplikace Mailshake[5] umožňuje založit si kampaň pro velké množství lidí, v případě, že nedostane daný člověk odpověď, aplikace mu sama pošle další předvolený email. Jde

[2] https://trello.com
[3] https://clickup.com
[4] https://frontapp.com
[5] https://mailshake.com



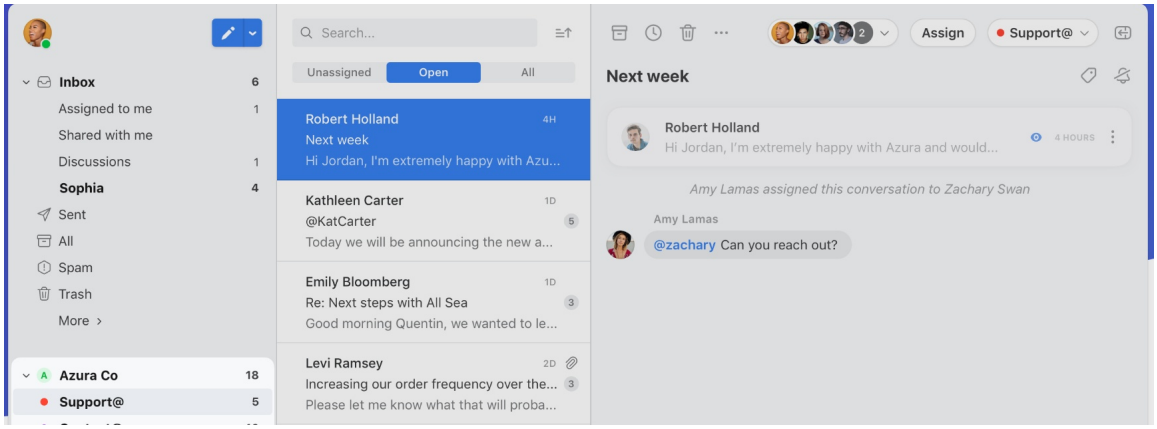
Obrázek 2.4: Ukázka úpravy jednoho z úkolů v aplikaci Trello, kde možné například přidat různé komentáře, nové úkoly nebo další členy.



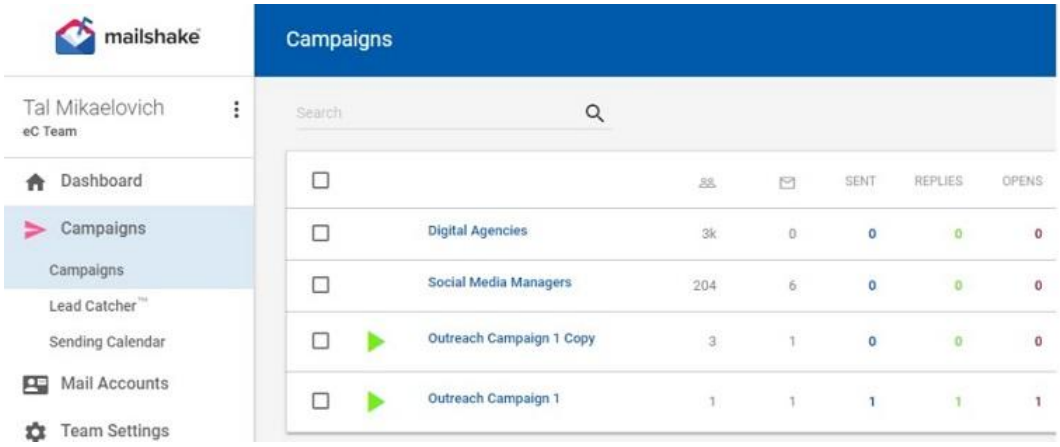
Obrázek 2.5: Ukázka nástěnky v aplikaci ClickUp, kde jsou všechny informace a úkoly přehledně na jednom místě, zvýrazněné je důležité tlačítko na přidání nového úkolu.

především o aplikaci k rozesílání newsletterů u velkých firem, takže stejně jako v Deadlineru se odesílá jeden email většímu množství lidí, zároveň tam také funguje posílání upomínek. V uživatelském rozhraní jsou zvýrazněná důležitá tlačítka jinou barvou, stejně tak některý text viz obrázek 2.7.

Velké množství z procházených aplikací má velmi podobný vzhled a rozmístění prvků především na úvodní stránce, což slouží pro lepší orientaci uživatelů, kteří si stále nemusí zvykat na něco nového. Při návrhu vzhledu hodně šetří barvami, výrazné se používají na tlačítka, případně zvýraznění klíčových elementů, pozadí jsou většinou světlá. Často se využívá technologie Drag & Drop. Pro nalákání uživatelů k použití aplikace je dobré mít údernou úvodní stránku, například se sloganem nebo ukázkou použití aplikace. Důležité



Obrázek 2.6: Nástěnka v aplikaci Front je opět podobná dříve zmíněným aplikacím, nejdůležitější informace jsou zvýrazněny. Připomíná emailovou schránku běžných emailových klientů, aby se uživatel rychle zorientoval.

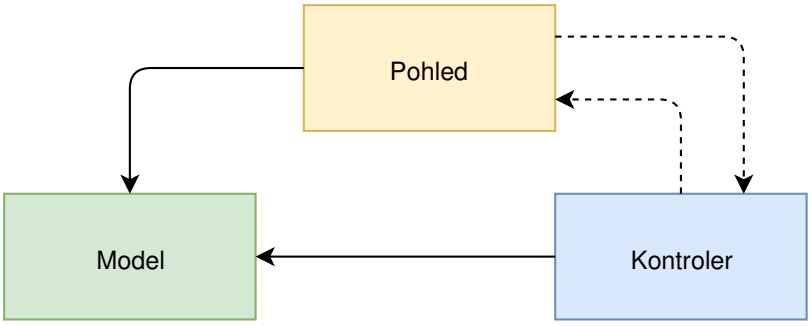


Obrázek 2.7: Ukázka nástěnky, která je opět podobná jako u předchozích aplikací, obsahuje seznam vytvořených kampaní, vlevo menu s možnostmi úprav.

je mít také co nejjednodušší a rychlé přihlášení, často využívané je přes účet Google či Facebook.

2.2 Nástroje na vytváření webových aplikací

Vzhledem k rostoucímu počtu uživatelů i webových stránek, je nutné stále vyvíjet a rozšiřovat technologie na tvorbu webových aplikací. V současné době existuje velké množství frameworků [5], které usnadňují práci tím, že obsahují balíčky a knihovny s velkým množstvím předprogramovaných funkcí, které se používají denně při vývoji, například klasické přihlášení nebo různé platební brány. Programátor pak nemusí programovat již vytvořené, ale může se zaměřit na specifické úkoly dané aplikace.



Obrázek 2.8: Ukázka, jak vypadají vazby v architektuře MVC, kromě dříve zmíněných jsou vazby mezi Kontrolerem a Pohledem. Pohled předává Kontroleru informace o kliknutí uživatele a Kontroler říká, jaký Pohled byl vybrán.

2.2.1 Architektura MVC

Architektura MVC [7] vytváří strukturu aplikace a zjednodušuje její chod. Dělí aplikaci na 3 logické celky, které je možné samostatně upravovat a při tom co nejméně ovlivňovat zbylé části. **Model** obstarává data a logiku aplikace, jako příklad se dá uvést odesílání mailů a správa uživatelů. **Pohled** (View) zobrazuje uživatelské rozhraní, většinou se jedná o HTML šablony s trochou PHP kódu. **Kontrolery**, neboli řadiče, se starají o to, co se děje v aplikaci, jedná se o propojení mezi moduly a pohledy. Existují dvě přímé vazby: Kontroler má přímý odkaz na Model, aby mohl upravit jeho data a Pohled má přímý odkaz na Model, aby mohl jeho data zobrazit viz obrázek 2.8.

2.2.2 SMTP

SMTP (Simple Mail Transfer Protocol) [8] je internetový protokol pro přenos elektronické pošty, funguje nad spolehlivým transportním protokolem TCP na portu 25. Zprávy přenáší přes agenta pro přenos zpráv MTA (Mail Transfer Agent). Odeslání začíná identifikací klienta příkazem *EHLO*, poté se přenese adresa odesílatele a příjemce, vlastní text je v příkazu *DATA*.

2.2.3 HTML

Jedná se o značkovací jazyk HyperText Markup Language [5] používaný na tvorbu webových stránek, které jsou propojené hypertextovými odkazy. Skládá se z elementů, které se stávají z párových (např. nadpisy) a nepárových tagů (např. obrázky). V současné době je již několik verzí HTML, poslední je HTML5, kde největším rozšířením je možnost přehrávat multimédia přímo v prohlížeči. Mezi tagy *head* se dávají metadata, která poskytují informace o jiných datech jako kódování, vykreslení obrazovky, jazyce, názvu, autoru, klíčových slovech nebo titulku stránky. Lze tam definovat i další dokumenty jako kaskádové styly nebo ikonku stránky. Do těla (body) se vkládá veškerý vlastní obsah daného dokumentu, existují různé velikosti nadpisů, seznamy číslované i odrážkové, odstavce a formuláře.

2.2.4 CSS

Kaskádové styly (Cascading Style Sheets) [10] popisují způsob zobrazení elementů jazyka HTML. Šetří práci, protože přes jeden styl lze zobrazit více HTML stránek stejným způ-

sobem. Nejčastěji je jeden nebo více souborů se styly odděleno od HTML a importováno v hlavičce, kde se hierarchicky řadí. Největší prioritu mají styly, které jsou přidány přímo do souboru HTML k elementům přes použití atributu *style*. Mezi nejrozšířenější volně stažitelé knihovny patří Bootstrap[6], který obsahuje šablony HTML, CSS i JavaScriptu. Díky tomu je možné velmi rychle a efektivně vytvořit webovou stránku, která podporuje responzivní design a je kompatibilní se současnými prohlížeči. Jsou v něm předprogramovány třeba **alerty** (upozornění, zprávy o úspěchu i o chybě) nebo tlačítka.

2.2.5 JavaScript

Velmi rozšířeným jazykem je JavaScript [10], který umožňuje psaní jednoduchých skriptů i celých aplikací. V něm je psaný i moderní framework Node.js, který obsahuje velké množství programů a knihoven, ze kterých je možné vytvořit kompletní webovou stránku. Kód běží na straně klienta, takže neumí pracovat se soubory, protože to by mohlo ohrozit bezpečnost uživatele. jQuery je knihovnou jazyka JavaScript, která především zjednodušuje práci s určitými elementy původního jazyka. Elementy se mohou psát stejně jako selektory v CSS, takže se pak lépe manipuluje a mění CSS styly, vytvářejí se jednodušeji efekty a animace.

2.2.6 PHP

PHP (Hypertext Preprocessor) [9] je serverový skriptovací programovací jazyk, používá se především pro tvorbu dynamických webových stránek, příkazy jsou prováděny na straně serveru (například Apache – multiplatformní server, na kterém běží většina současných webových aplikací) a vygenerovaný dokument se spustí na straně klienta. Ukázka skriptu, který vypíše „Ahoj, světe!“:

```
<?php echo "Ahoj, světe!"; ?>
```

Používá se také ke spojení s databází, je v něm napsán třeba phpMyAdmin nebo Adminer, obě aplikace slouží ke správě MySQL databáze. V PHP byl vytvořen třeba redakční systém WordPress či světoznámý Facebook. Najdeme v něm velké množství vestavěných funkcí pro práci s regulárními výrazy, časem, textem. Na jeho bázi je založeno také velké množství frameworků. V Česku je nejrozšířenější framework Nette vytvořený českými vývojáři. Ve světě jsou hodně podporované a používané frameworky Laravel a Symfony.

2.2.7 Symfony

Symfony se řadí mezi webové aplikační frameworky pro vývoj webových aplikací v jazyce PHP a využívá architekturu MVC (kapitola 2.2.1). První verze byla publikována už v roce 2005, kdy byly zároveň spuštěny webové stránky. Momentální nejnovější verze je 4.2, bakalářská práce je napsaná ve verzi 3.4, která vyšla v listopadu 2017 a bude oficiálně podporována tři roky. V době vývoje aplikace nabízela výrazně lepší podporu, jak samotného frameworku, tak doinstalovaných knihoven, proto byla práce vytvořena v této verzi.

Používá velké množství komponent a knihoven, které jsou buď přímo součástí při instalaci nebo je možné doinstalovat. Jedná se o jeden z nejrozšířenějších frameworků, který používá více než 600 000 lidí ve 120 zemích[7]. Díky tomu je možné najít obrovské množství návodů a je zajištěna kompatibilita s mnoha jinými aplikacemi. Díky Symfony [3] jsem

[6] https://getbootstrap.com
[7] https://symfony.com

objevila mnoho nových typů souborů, asi nejdůležitější je YML soubor, který je spojen s jazykem YAML (Ain't Markup Language určuje formát pro serializaci strukturovaných dat[8]). Používá se k organizaci souborů, aby se v nich dobře vyznal člověk.

Struktura složek a důležitých souborů v frameworku Symfony ve verzi 3.4 je následující, v další verzi už došlo k výrazné obměně:

- **app** – zde se nachází soubor AppKernel.php, do kterého se po instalaci knihoven přes nástroj Composer musí přidat reference na dané rozšíření, aby framework věděl, že ho má používat
  - **config** – v této složce se nachází všechny důležité konfigurační soubory
    * přístup do databáze se ukládá do **parameters.yml**
    * většina dalších údajů je pak v **config.yml**, do kterého uživatel většinou zasahuje nejčastěji při instalaci komponent, protože je obvykle nutné přidat odkazy nutné pro funkčnost nového rozšíření (například na šablony Twig nebo přihlašovací údaje)
    * v souboru **security.yml** se nastavuje rozlišení práv uživatelů (pomocí rolí) a na jakou stránku se lze dostat i bez přihlášení
    * v **services.yml** se definují cesty, většinou pro umístění kontrolerů
  - **Resources** – zde se ukládají všechny šablonovací soubory s příponou twig, které potom generují výsledné HTML stránky
- **bin** – soubory pro Symfony, pro běžného programátora nepodstatné
- **src** – jedna z nejdůležitější složek, do které se ukládají všechny kontrolery, entity (ve kterých jsou definovány tabulky v databázi), případně navržené často využívané formuláře nebo uživatelské funkce do šablonovacího systému Twig
- **translation** – zde jsou uživatelské překlady, které je vhodné ukládat do souboru s příponou `.yml`
- **var** – dočasná složka, která se vytvoří až po načtení stránky, jsou tam uložena data z paměti Cache, případně různé logovací soubory a jednotlivá sezení
- **vendor** – složka pro nástroj Composer, kam se během instalace ukládají všechny nové knihovny a rozšíření
- **web** – složka pro zobrazování na webu, nachází se v ní soubory kaskádových stylů i JavaScriptu, případně soubory nahrané uživatelem a obrázky do šablon (včetně ikonky stránky tzv. favicon), jedním z důležitých souborů je app.php, kde se vypíná případně zapíná debugovací mód, na ostrých stránkách by měl být určitě vypnut, aby se nepovolané osoby nedozvěděly, co nemají
- **composer.json**, **composer.lock** – dva soubory, které se nachází v složce root a jsou důležité pro Composer

[8] https://yaml.org

Composer

Composer je nástroj pro správu závislostí v PHP uložený v souboru *composer.json*. Lze díky němu deklarovat, instalovat a aktualizovat knihovny, které se po instalaci ukládají se do složky vendor. Composer používá většina frameworků.

Během instalace balíčků je nutné hlídat správnou verzi, která je v souboru uvedena za názvem, protože ne všechny jsou podporovány, takže je nutné postupovat podle návodu v oficiální dokumentaci každé knihovny. Nejčastěji se používá přes příkazovou řádku, bez které se při práci se Symfony neobejdeme, je třeba od instalace, práci s databází po přidávání nových knihoven. Také používá velké množství již existujících open-source projektů, které jsou součástí základního balíčku Symfony jako třeba Swift Mailer pro práci s maily, Doctrine či ORM, které slouží k práci s databází a mapování objektů.

Kontrolery

Jsou funkce PHP pro vytváření a čtení informací z objektů, které mají nějaký požadavek (Request). Poté vytvoří a vrátí objekt s odpovědí (Response) ve formě stránky HTML, souboru JSON, chyby či přesměrování. Jedná se o třídu, která již v názvu skrývá slovo *Controller* a svoje jméno, v ní jsou jednotlivé funkce, které ve svém názvu musí končit slovem *Action* (př. indexAction). V URL se přistupuje přes příkaz *Route*, který se skrývá v komentáři a je tam vepsaná relativní URL adresa, kde je určena jazyková verze stránek a zbytek adresy, zároveň je tam uveden název, přes který je možný volat funkci z jiných tříd i funkcí. Nejčastější návratovou hodnotou funkcí je vykreslení HTML stránky přes implicitní šablonovací systém Twig, případně přesměrovaní na již existující vykreslenou stránku.

Twig

Twig je šablonovací systém pro PHP, který kompiluje svůj specifický kód a vytvoří z něho HTML stránku. Je možné vkládat data, která pošle kontroler, případně ještě pracovat s proměnnými přes podmínky a cykly, dokonce si vytvářet vlastní proměnné a různá makra. Zároveň se tím zvýší i bezpečnost výsledného produktu, protože se v Twigu nedá míchat PHP a HTML. Syntaxe se vyznačuje složenými závorkami, které uzavírají proměnné či podmínky. Příklad vypsání jména kontaktu `{{contact}}` nebo uvození podmínky `{% if %}`

2.2.8 Databáze

Databáze je systém souborů, které mají pevnou strukturu a jsou jasně uspořádány a navzájem propojeny přes klíče. Systém řízení báze dat (dále SŘBD) je software, který zajišťuje komunikaci s databází a práci s daty. Tvoří rozhraní mezi aplikačními programy a uloženými daty. Mezi používaný software se řadí třeba Microsoft SQL Server, který je určený spíše pro velké systémy jako eshopy či řešení datových skladů, podporuje procedury, trigery i pohledy.

Asi nejrozšířenější SŘBD je MySQL [9], které je především optimalizováno pro rychlost. Je možné ho používat pod bezplatnou licencí i komerční. V projektu Deadliner je využívána databáze MySQL, formát úložiště je InnoDB, který je využíván pro plnění transakcí, podporuje cizí klíče. Úložiště dat je softwarová komponenta, kterou SŘBD používá k operacím nad databází jako vytvořit, přečíst, zeditovat a smazat, zároveň obsahuje engine, díky kterému je možné všechny tyto operace jednoduše vykonávat.

K vytváření tabulek a další manipulaci s databází se v Symfony používá knihovna Doctrine[9], která je volně dostupnou knihovnou, kterou Symfony pouze využívá. Doctrine obsahuje objektově orientovaný přístup a mapování, čímž se dá pracovat s databázovými prvky jako s objekty. Jedná se o ORM (Object Relational Mapping), což je nejvyšší vrstva, která zajišťuje mapování, ukládání a načítání objektů do relační databáze.

2.3 Uživatelské rozhraní

Uživatelské rozhraní (User Interface, UI) [2] je, to co uživatel vidí na obrazovce a jak komunikuje s daným zařízením, ovlivňuje chování různých systémů (třeba strojů nebo programů). Důraz se klade na jednoduchost a přehlednost, kromě vzhledu je důležitá také logická struktura programu, aby se uživatel snadno a rychle naučil rozhraní ovládat. Podstatným bodem při vytvoření UI je testování, tedy zkoumání chování a reakcí uživatelů při jeho používání.

Pojem uživatelská zkušenost (User Experience, UE) [11] také úzce souvisí s UI, jde o pocity a reakce, které mají uživatelé při práci s programem, co si pamatují při práci s rozhraním. Nejdřív je nutné zjistit očekávání potenciálních uživatelů, poté navrhnout funkce, o které má uživatel zájem, teprve dalším krokem bude návrh aplikace a uspořádání prvků. Hlavním cílem UI i UX je zjednodušit a zlepšit používání aplikace.

Příklady UI jsou webové a mobilní aplikace, operační a databázové systémy. Mezi faktory, které ovlivňují kvalitu rozhraní, patří přístupnost (responzivita [10], tedy jak se aplikace umí přizpůsobit na různě velkých zařízeních jako mobily, tablety nebo počítače), použitelnost [6], do které se řadí konzistentnost ovládacích prvků, intuitivita ovládání a zpětná vazba.

Základní dělení uživatelského rozhraní [4] na příkazový řádek (CLI = Command Line Interface), kdy uživatel zadá vstup textově do řádku, poté počítač vyhodnotí a vrátí výsledek. Další je grafické uživatelské rozhraní (GUI = Graphical User Interface), kdy základem je zobrazení WIMP (spuštěná okna, ikony – zkratky, přes které se spouští program, Nabídky – menu a Polohovací zařízení). Uživatel v GUI přímo ovládá objekty třeba pomocí myši, klávesnice nebo dotykové obrazovky. Mezi nejtypičtější používané prvky UI jsou zaškrtávací tlačítka, přepínače, seznamy či stránkování.

Použitelnost

Použitelnost [6] je soubor pravidel, které zlepšují interakci uživatele s webovou stránkou. Zvyšuje přehlednost a srozumitelnost, že se návštěvník na webu brzy zorientuje a pár kliknutími najde, co potřebuje. Pokud uživatel klikne na tlačítko, od kterého očekával něco jiného, hned se jeho zkušenost se stránkou zhorší. Jde například o to, že když bude nějaký odkaz podtržený, uživatel bude očekávat, že to je odkaz, zároveň je i barevně odlišený od dalšího textu. Negativní může být vymýšlet nový design a ovládání webu, které je neobvyklé, zároveň je důležité logicky stránky uspořádat nebo zajistit rychlé vygenerování stránky bez dlouhé odezvy.

[9] https://www.doctrine-project.org

Kapitola 3

Návrh webové aplikace

Aplikace **Deadliner** slouží k zadávání pracovních i osobních úkolů. Je zaměřena na lidi, kteří mají pod sebou své zaměstnance, kolegy, na vedoucí různých skupin, ať už skautských či sportovních, případně na učitele a jejich studenty. Například učitel zadá „*Do této schůzky mi vypracujte následující tabulku*“ nebo „*Do příště otestujte následující řešení*“. Úkol je zadán třeba 15 lidem. Zadavatel úkolu pak nemusí složitě hledat v mailu, kdo mu už úkol splnil a kdo ne, pouze se v aplikaci podívá na přehled. Pokud zjistí, že třeba dva studenti nemají splněno, tak jim pošle jedním kliknutím upomínkový email. Každý, komu je zadán úkol, dostane speciální email s jedním tlačítkem, které ho přesměruje na stránku se zadáním, kde může zaškrtnout splnění úkolu.

3.1 Případy užití

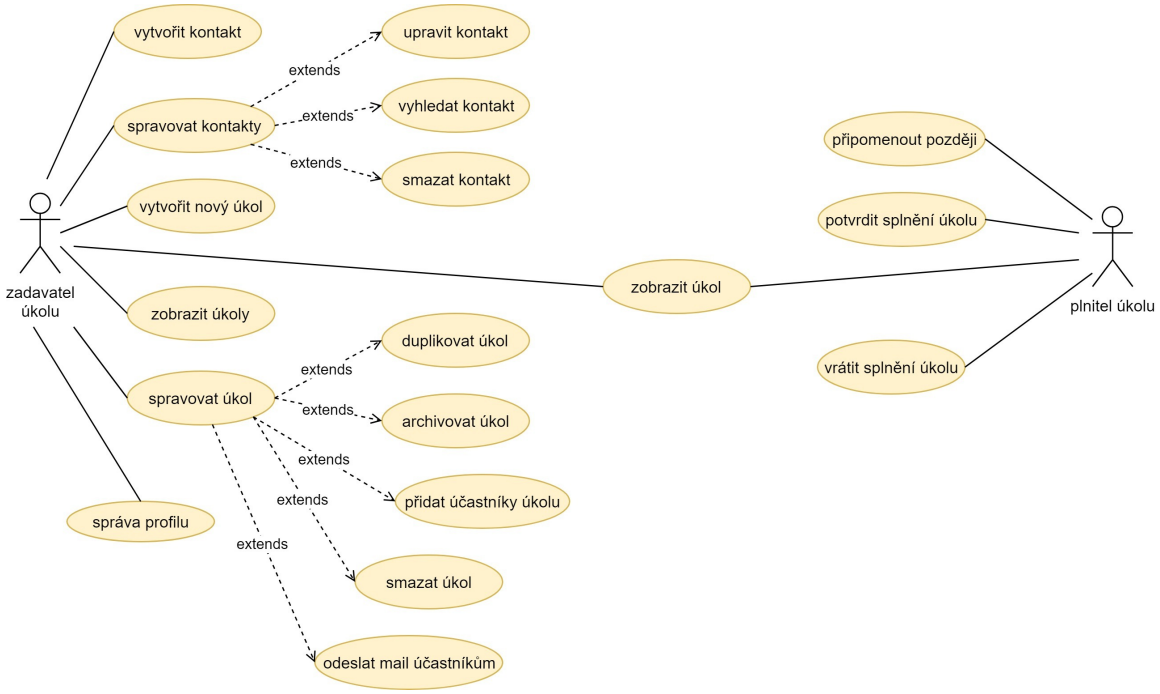
Diagram případu užití na obrázku 3.1 popisuje, jak se bude chovat systém pro konkrétního uživatele. Skupiny uživatelů jsou dvě, zadavatel úkolu a jeho plnitel.

Zadavatel úkolu se musí nejdříve zaregistrovat a přihlásit, poté bude jeho hlavním posláním přidávat úkoly, případně je upravovat. Do úkolů musí ale přidat nejdříve kontakty, které si nejdřív založí, zařadí je do skupin, případně je může upravit nebo smazat. Pro větší přehlednost bude všude vybráno výchozí řazení a maximální stránkování. Po splnění úkolu nebo vypršení konečného termínu je možné úkol archivovat. Pro případ, že se některé zadání opakuje častěji, tak je možné využít akci kopírování (duplikace) úkolu. Zadavatel taky musí mít možnost třeba si změnit heslo nebo upravit email nebo jméno, což je shrnuto do akce úpravy profilu. U každého úkolu je jednou z nejdůležitějších akcí odeslat email (jednotlivě nebo všem naráz) a poté mít přehled o úkolu.

Plniteli úkolu přijde email, po kliknutí na tlačítko s odkazem se dostane na stránku, kde uvidí přehled úkolu a bude moct zaškrtnout splnění úkolu, případně si nechá odeslat připomenutí později. Pokud uživatel omylem klikl na splnění, tak tuto akci musí mít možnost vrátit. Plnitel úkolu nepotřebuje žádnou úpravu profilu, protože se mu stránka zobrazí i bez registrace a přihlášení, nastavuje si pouze výchozí jazyk pro zobrazení stránky.

3.2 Návrh struktury databáze

Databáze se skládá celkem ze pěti tabulek, postupným testováním se dostaly tabulky do konečné verze více v obrázku 3.2. Uchovávají se v nich údaje o uživatelích, jejich úkolech a

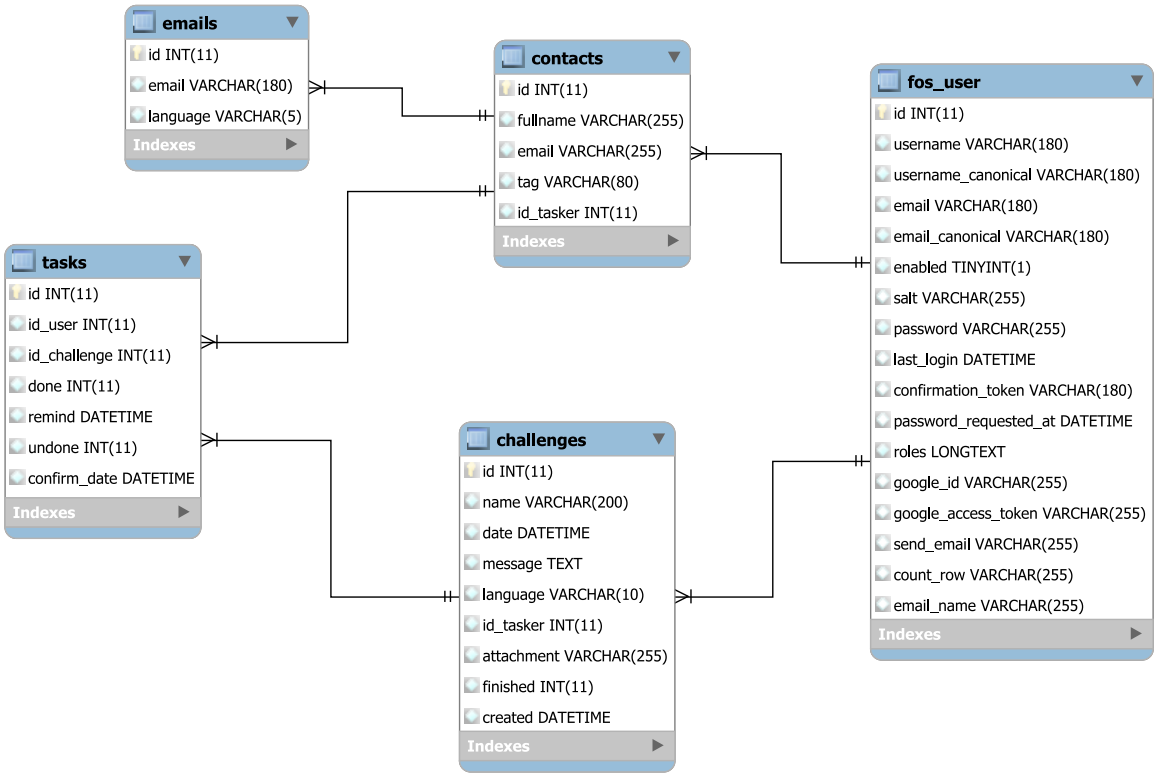


Obrázek 3.1: Případ užití ukazuje činnosti uživatelů, zadavatel úkolu se nejdříve musí přihlásit, vytvářet a spravovat úkoly i kontakty. Plnitel úkolu si pouze zobrazí úkol a potvrdí ho.

kontaktech. Další tabulka spojuje jednotlivé úkoly s přiřazenými kontakty a označuje, zda už byly splněny či ne. Poslední specifická tabulka uchovává výchozí jazyk každého kontaktu.

3.2.1 Tabulka `fos_user`

Tabulka bude obsahovat všechny registrované uživatele, případně přihlášené přes Google. Většina sloupců je automaticky definovaná a generovaná přes knihovnu pro práci s přihlášením. Celkově se jedná o 17 sloupců. Primárním klíčem je sloupec id, který je typu *int(11)*, poté následují čtyři sloupce typu *varchar(180)* username, username_canonical a email, email_canonical, do kterých se zapisuje přihlašovací jméno a email, jednou s diakritikou a podruhé bez ní. Sloupec speciálního typu *tiniint(1)* enabled ukazuje stav, zda uživatel potvrdil registraci a může se přihlásit. Také tam najdeme řetězec *varchar(255)* salt a password, kam se ukládají šifrované údaje. Stejný typ je confirmation_token, kam se uloží řetězec pro potvrzení účtu. Typu *datetime* jsou sloupce last_login a password_requested_at, které uchovávají informaci o posledním přihlášení a kdy uživatel naposledy žádal o změnu hesla. Během dvou hodin lze žádat o změnu hesla pouze jednou, poté se interval obnoví. Sloupec roles typu *longtext* slouží k uchování práv uživatele a je přímo navázán na logiku frameworku. Dva sloupce typu *varchar(255)* google_id a google_access_token byly přidány, aby bylo možné pamatovat si uživatele přihlášené přes jejich účet Google. Sloupce *varchar(255)* send_email a email_name byly přidány až dodatečně k nastavení emailové adresy a uživatelského jména, které se bude objevovat v emailu, pokud se liší od přihlašovacích údajů. Sloupec count_row *varchar(255)* slouží k nastavení výchozího počtu řádků v přehledu kontaktů nebo úkolů.



Obrázek 3.2: ER diagram s návrhem všech tabulek v databázi, které jsou popsány v kapitole 3.2, zobrazuje propojení tabulek a jejich sloupce. Vytvořen byl pomocí programu MySQL Workbench.

3.2.2 Tabulka `challenges`

Slouží k ukládání všech informací o úkolu. Obsahuje primární klíč id typu *int(11)*, který je nastaven, tak aby se sám inkrementoval. Dalším sloupcem je name typu *varchar(200)*, kam se ukládá název úkolu. Date je typu *datetime* a je v něm konečný termín. Text zprávy je obsažen ve sloupci message typu *text*. Atribut language *varchar(10)* ukládá jazyk uživatele, který má na stránkách v době psaní zprávy nastavený a než si každý kontakt nastaví vlastní jazyk zprávy, je používán tento výchozí. Důležitým cizím klíčem je id_tasker, kam se ukládá sloupec id z tabulky `fos_user`, tedy jedinečné číslo každého přihlášeného uživatele. Do sloupce attachment *varchar(255)* se ukládá název přílohy, která byla přiložena k danému úkolu. Do atributu finished *int(11)* se ukládá hodnota 1, pokud je úkol dokončen, jinak 0. Do sloupce created typu *datetime* se zaznamenává datum vytvoření úkolu, což je využíváno především k řazení.

3.2.3 Tabulka `contacts`

Nachází se v ní všechny informace o kontaktech a k jakému uživateli patří. Původně obsahovala šest sloupců, na základě testování byl sloupec photo vyškrtnut jako nepotřebný, měl obsahovat fotku každého kontaktu. Opět se zde nachází primární klíč id *int(11)*, který každý kontakt jednoznačně definuje. Ke každému uživateli jsou kontakty přiřazeny přes cizí klíč id_tasker *int(11)*. Poté obsahuje fullname, kam se vkládá celé jméno kontaktu, email

pro jeho emailovou adresu a tag, kam se ukládá skupina, do které patří. Všechny jsou typu *varchar*.

3.2.4 Tabulka `tasks`

`Tasks` spojuje všechny tři dosud zmíněné tabulky, přes jejich klíče, aby bylo možné jasně identifikovat, pro koho je daný úkol určený. Obsahuje tedy primární klíč id *int(11)*, který se sám inkrementuje. Cizí klíč id_user zobrazuje id kontaktu a sloupec id_challenge zobrazuje id daného úkolu z tabulky `Challenges`. V tabulce jsou ještě dva časové údaje typu *datetime*, kdy byl úkol potvrzen confirm_date a případně, na kdy je nastaveno emailové upozornění remind, oba údaje se pak zobrazují zadavateli úkolu, aby měl přehled o splnění. Ještě se v tabulce nacházejí dva sloupce typu *int(11)*, done a undone, které zobrazují, zda už je úkol splněn, respektive, jestli bylo potvrzení zrušeno.

3.2.5 Tabulka `emails`

Tabulka, která byla přidána až po testování, slouží k ukládání mailu každého kontaktu, ke kterému je přiřazen jazyk zvolený kontaktem. Důvod nové tabulky byl ten, aby si každý kontakt nemusel svůj jazyk vybírat několikrát, pokud by byl například přidán více uživateli. Tabulka obsahuje opět klasický primární klíč id *int(11)*, poté *varchar(255)* email a menší *varchar(5)* language, kam se ukládá zkratka vybraného jazyka.

3.3 Návrh uživatelského rozhraní

Při návrhu uživatelské rozhraní byly využity znalosti získané v kapitole 2.1 při zkoumání podobných aplikací. Nutné je, aby se uživatel při přístupu na stránky rychle zorientoval, proto je dobré držet se zažitých koncepcí a nevymýšlet novinky. Celkově se jedná o návrh pěti základních stránek.

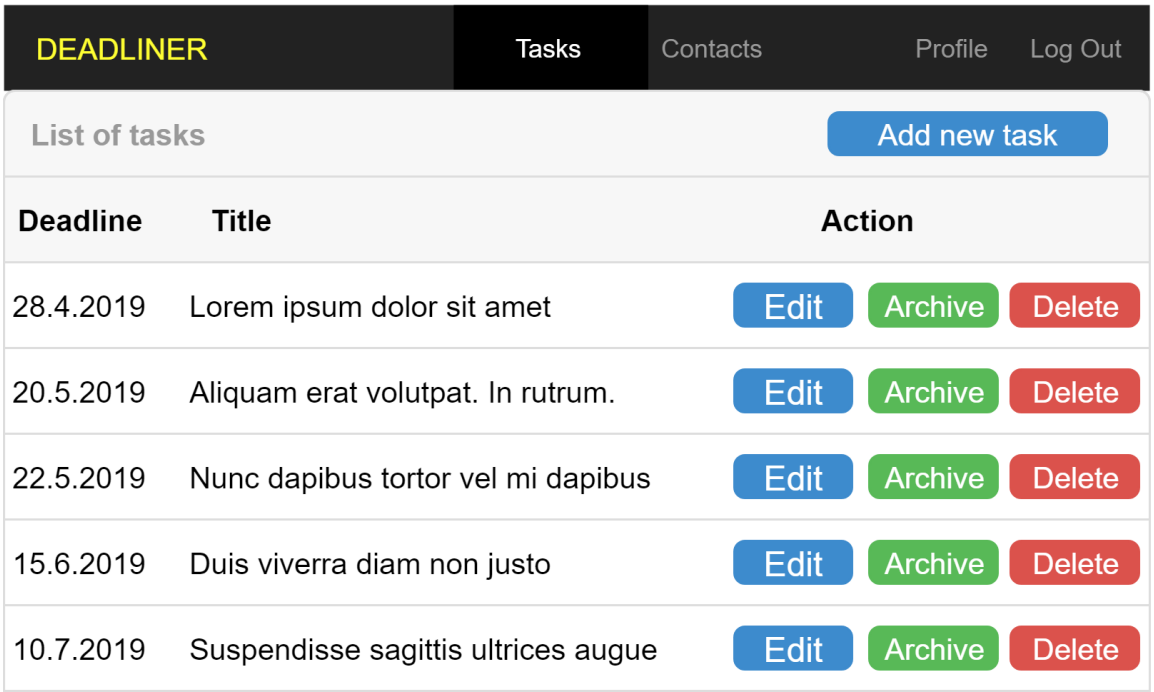
První stránka je s přehledem všech vytvořených úkolů jako v obrázku 3.3, která tvoří pouze seznam, kde je název úkolu, jeho konečný termín a nějaké dostupné akce jako upravit, smazat či archivovat úkol.

Další podstránka je stejná jak pro editaci, tak pro založení nového úkolu, viz obrázek 3.4. Obsahuje formulář, který se akorát v případě editace již předvyplní známými údaji. Lze tam zadat název úkolu (předmět emailu), zadání úkolu, což může být delší text, datum, do kdy má být splněn a dva boxy s kontakty, kde v jednom budou všechny kontakty a ve druhém pouze přidaní k danému úkolu. Dole bude tlačítko s odesláním úkolu emailem všem vybraným.

Kontakty je nutné nejprve přidat, poté se dají použít v úkolech. Každý bude mít vlastní fotku, jméno, email a skupinu, která je bude sjednocovat a jednodušeji se pak budou přidávat větší počty lidí do úkolu. Pro vložení bude formulář na stránce, pod ním bude seznam kontaktů, kde bude možné jednotlivé kontakty upravit či smazat jako v obrázku 3.5.

Poslední obrazovkou bude přehled daného úkolu, ten se bude zobrazovat jak zadavateli úkolu (obrázek 3.6), tak jeho plniteli. Budou tam všechny důležité informace jako konečný termín, název a text úkolu. Zadavatel úkolu bude mít pod tím přehled, kdo už úkol splnil, případně mu bude moct poslat připomínku. U toho, kdo úkol plní, nebude žádný přehled, kdo má splněno, ale dvě tlačítka **Splněno** a **Připomenout později**.

V menu stačí pouze položky Přidat úkol, Spravovat úkoly a Spravovat kontakty, případně úprava profilu nebo zvolení výchozího jazyka.



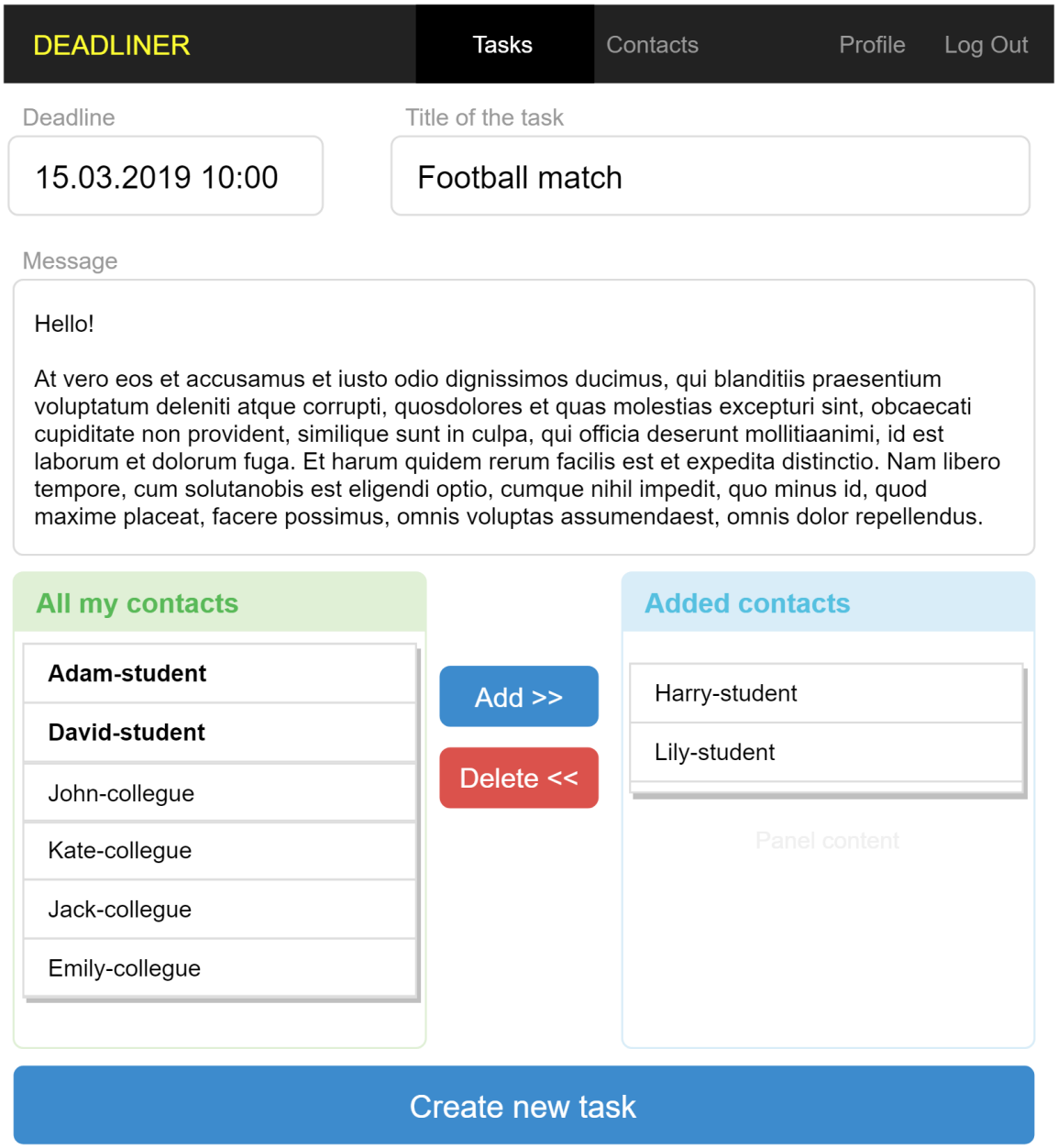
Obrázek 3.3: Návrh stránky s přehledem všech vytvořených úkolů a akcí ke každému úkolu. Návrh nástěnky byl inspirován aplikacemi zmíněnými v kapitole 2.1.

3.4 Návrh testování

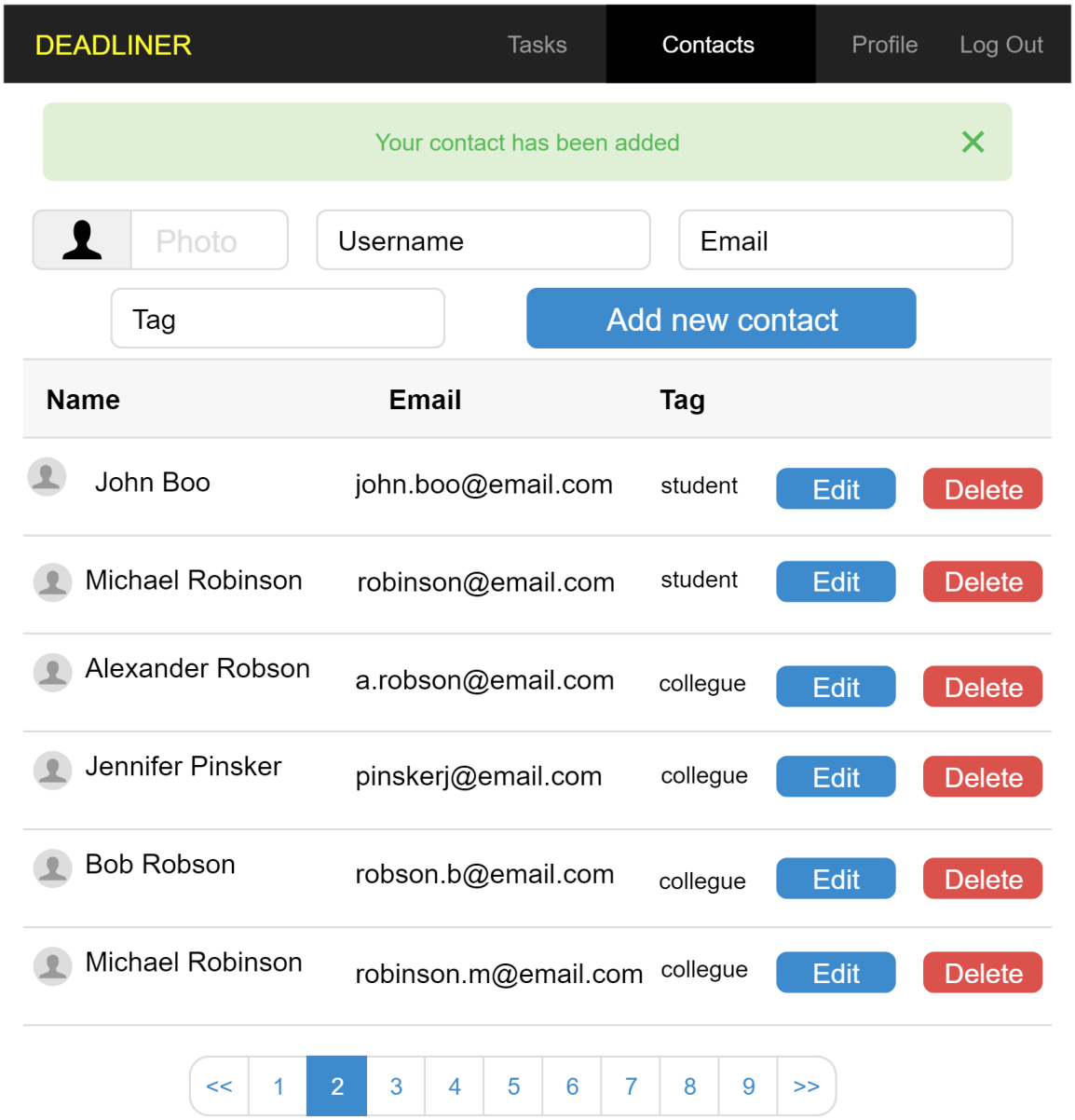
Budou prováděny dva druhy testování: programátorské a uživatelské. První typ provádí sám vývojář, který testuje chyby v programování, ošetření výjimek, případně testeři, kteří se trochu v programování vyznají. Druhý typ bude uživatelské testování, kdy široké veřejnosti budou zaslány různé dotazníky s cílenými otázkami na vzhled a dostupné funkce. Součástí tohoto testování bude testování použitelnosti, kdy bude aplikace testována uživateli, kteří budou pozorováni při používání aplikace.

3.4.1 Testování použitelnosti

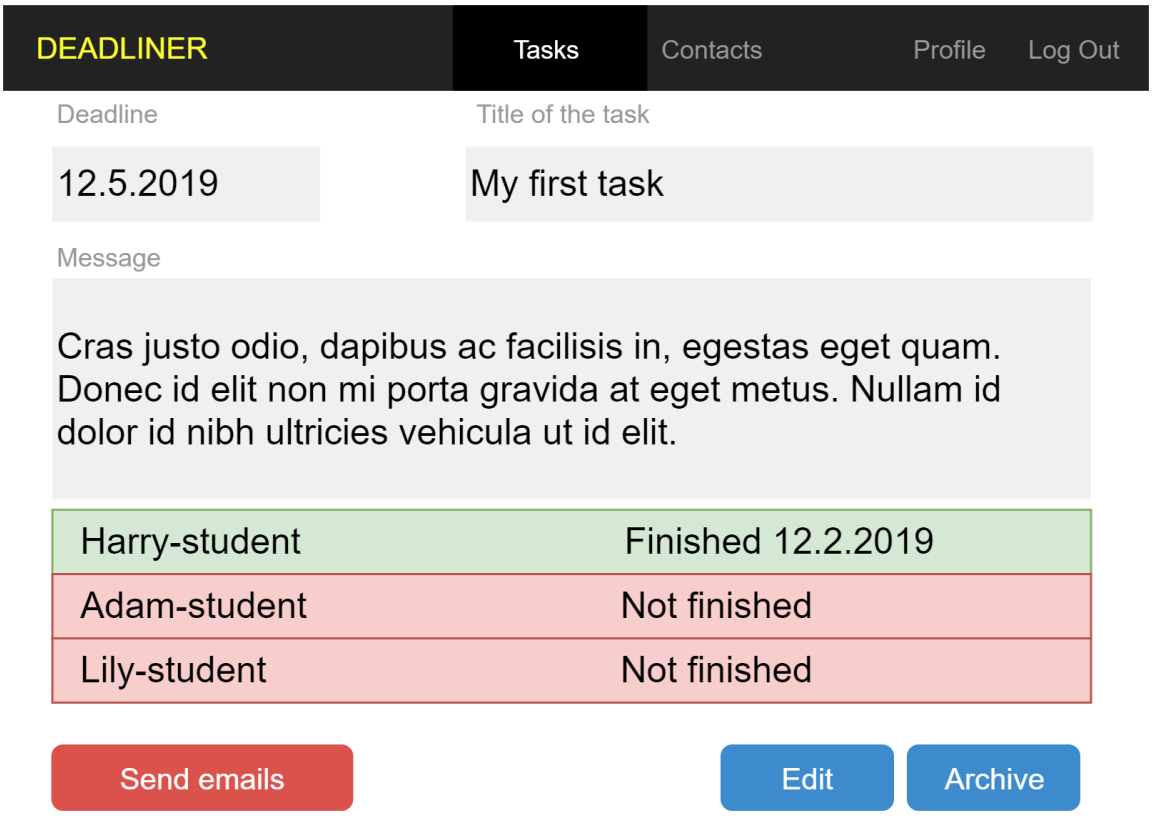
Cílem je otestovat jakým způsobem budou uživatelé produkt využívat. Během testování je nutné odhalit všechny chyby a stavy, které mohou nastat během používání. Většinou to probíhá tak, že přijde průměrný uživatel webu, dostává dotazy typu „*Rozumíte tomu?*“, „*Na co byste kliknul jako první*“, případně se pouze posadí k webu a sám s ním zkouší pracovat a je monitorován. Používají se různé metody [6], třeba eye-tracking zaznamenává kamerou pohyb očí, podle toho, kam uživatel kouká nejčastěji a co vyhledává se aplikace dá dále vylepšovat. Hallway testing je levná metoda, kdy u skupiny dobrovolníků se sleduje, jakým způsobem používají danou stránku.



Obrázek 3.4: Návrh stránky pro přidání případně editaci úkolu. Obsahuje jeden formulář, kde jsou pole na zadání názvu, zprávy a jedno velké výběrové pole pro přidání kontaktů k danému úkolu.



Obrázek 3.5: Návrh stránky pro přidání a editaci kontaktů každého uživatele. Obsahuje formulář pro přidání a poté seznam již přidaných lidí.



Obrázek 3.6: Návrh stránky pro zobrazení detailů vybraného úkolu, kromě zadání a konečného termínu, poskytuje přehled zadavateli, kdo jeho úkol už splnil a komu musí odeslat upomínku.

Kapitola 4

Implementace

Implementace proběhla především ve frameworku Symfony, který byl doplněn klasickými webovými technologiemi jako CSS a JavaScript. Databáze byla navržena v MySQL, pro její správu je používán phpMyAdmin[1].

4.1 Instalace Symfony

Přesný postup instalace frameworku Symfony přes příkazový řádek je na oficiálních stránkách[2]. V bakalářské práci byl použit příkaz *php symfony new deadliner 3.4*, který vytvoří přes Composer (kapitola 2.2.7) projekt. Všechny verze byly vytvářeny nejprve na lokálním serveru, který se spustí v příkazové řádce a stránky zobrazuje na adrese 127.0.0.1 s portem 8000. Pro běh lokálního serveru byl využíván program XAMPP[3], který poskytuje Apache server, na kterém běží většina webových aplikací na internetu. Podporuje jazyk PHP a dovolí vytvořit lokální databázi MySQL. Jako verzovací program je využíván klasický systém git přes stránku Bitbucket[4], jeho výhodou je rychlé obnovení jakékoliv předchozí verze.

4.2 Implementace databáze

Pro přístup do databáze je v Symfony nejprve nutné nakonfigurovat soubor *parameters.yml*, ve kterém se nastaví přihlašovací údaje k databázi. Jako host se zadává většinou lokální adresa 127.0.0.1, protože databáze i webové stránky běží na stejném serveru. Někdy se uvádí i port, nutné je přihlašovací jméno a heslo, které poskytne konkrétní webhosting.

Pokud připojení k databázi proběhne úspěšně, tak je možné začít vytvářet tabulky. K tomu se dá také využít opět Composer, což výrazně ulehčí práci, protože zajistí vytvoření příslušných Entit. **Entita** je speciální třída, kde jsou uloženy všechny sloupce z tabulek jako proměnné a k nim vytvořené speciální funkce pro vkládání a získávání dat z databáze. Tabulka patří do výchozí Bundle AppBundle, což je základní stavební jednotka, kde na základě doporučeného rozložení složek a správné konfiguraci Symfony přesně ví, kde hledat jakou část aplikace. První generování navržené tabulky tedy probíhá přes příkazový řádek, kde se postupně zadá název tabulky ve formátu `Bundle:Tabulka`, poté se postupně zadají jednotlivé sloupce a jejich typy. Často používaný příkaz `php bin/console`

[1] https://www.phpmyadmin.net
[2] https://symfony.com/doc/3.4/setup.html
[3] https://www.apachefriends.org/index.html
[4] https://bitbucket.org/

`doctrine:schema:update -force` založí tabulky přímo v MySQL databázi, případně aktualizují po změnách na základě obsahu Entit (například chci posléze přidat jeden sloupec, přidám ho do dané Entity v PHP kódu a pak zavolám funkci).

Kromě samotných tabulek je v každé třídě funkce Set a Get, které se dále používají při vkládání a získávání dat z databáze. Pokud je v každé tabulce klasické jedinečné identifikační číslo, tak se samo inkrementuje (nastaven AUTO_INCREMENT).

Níže zmíněné funkce je pak možné využívat v kontrolerech, například když je nutné najít emailovou adresu kontaktu, tak se využije knihovna Doctrine. Nejprve jsou získána konkrétní data z dané tabulky na základě identifikačního čísla kontaktu: `getRepository ('AppBundle: Contacts')->find(id)`, poté je zavolána `getEmail()`, která v získaných datech najde hledaný email. Vkládání do databáze probíhá stejně, jen je zavolána funkce `setEmail()`, poté se ještě použijí funkce *persist* a *flush*, po kterých se změny uložit.

Všechny tabulky byly implementovány podle navrženého ER-diagramu (obrázek 3.2).

4.3 Implementace důležitých tříd a komponent frameworku

Klíčovým souborem je *config.yml*, kde se nastavuje konfigurace používaných knihoven, například se přes něj získávají přihlašovací údaje ze souboru *parameters.yml* čtou. Najdou se tam veškerá důležitá nastavení Doctrine i Swift Maileru, který zajišťuje obsluhu mailu, také se tam nastavuje přihlašování uživatelů, cesta k databázovému souboru nebo přihlášení s jinými službami, jako Google. Téměř všechny doinstalované komponenty musí mít zde svoji část s nastavením. Na začátku je také důležité nastavit soubor *routing.yml*, kde původní nastavení říká, že na všechny funkce v kontrolerech v AppBundle bude možné směrovat přes anotace (annotation).

V kontrolerech existují dva způsoby, jak se pomocí příkazu *Route* přesměrovat na šablonu Twig. První `redirectToRoute` přesměruje na jinou již existující funkci, která většinou využívá metodu `render`, kde se volá přímo šablona Twig s argumenty proměnnými, které se v ní pak používají.

4.3.1 Přehled kontrolerů

Celkem tvoří systém 10 kontrolerů, většina z nich načítá data z formulářů a následně je zpracovává. Každý kontroler obsahuje nějaké akce, což jsou vlastně funkce. Každá z těchto speciálních funkcí musí vracet odkaz na nějakou šablonu, která vygeneruje HTML stránku. Data z formuláře se přenáší v požadavku (Request), takže například data ze vstupního pole formuláře *message* se získají přes funkci *get* zavolané na proměnou požadavek (*request*).

- **AddController** je třída, která zpracovává data z formuláře pro přidání úkolu, případně pro úpravu a zároveň zajišťuje kopírování dat při pokynu *Kopírovat úkol*. Poté data vkládá do tabulek Challenges a Tasks.
- **ConfirmController** generuje stránku s potvrzením úkolu, které přijde mailem na základě hash kódu, zároveň zajišťuje funkce na potvrzení úkolu a poslání upomínky.
- **ContactController** vytváří stránku s kontakty, zajišťuje jejich přidání, editaci i mazání, zároveň vyhledávání a řazení. Jednou ze zajímavých funkcí je *normalize*, která zajišťuje validitu emailu, takže odstraňuje veškerou diakritiku a speciální znaky nad písmeny.
- **DefaultController** zajišťuje akce z úvodní stránky, především odeslání připomínek.

- **FunctionController** obsahuje funkce na odesílání mailů, zároveň vytváří pomocí regulárního výrazu odkazy z adres.
- **GoogleController** rozšiřuje zpracování klasického přihlášení, pokud se někdo přihlásí přes účet Google. Jde především o speciální nastavení profilu, jako změna jména či emailu odesílatele nebo maximální počet řádků v seznamu kontaktů nebo úkolů na stránku.
- **OverviewController** poskytuje přehled vybraného úkolu a obsluhuje požadavky jako editace či smazání vybraného kontaktu u úkolu. Volá funkce na odesílání mailů.
- **RemindController** je kontroler, se kterým se běžný uživatel téměř nesetká, protože se spouští automaticky ve zvolenou ranní dobu pomocí tzv. cronů, které poskytuje webhosting. Jsou to události, které se pravidelně vykonávají. Řadí se tam akce jako připomenutí později, kdy najde odpovídající dny a pošle upozornění a zároveň pošle zadávajícím upozornění, že dnes je konečný termín jejich úkolu.
- **TaskController** pomocí funkcí v kontroleru se vykresluje přehled všech úkolů daného uživatele, zároveň se provádí archivace již dokončených úkolů, aby nepřekážely mezi těmi aktuálními.
- **PageNotFoundController** je kontroler, který pouze přesměrovává na stránku s úkoly, pokud podstránka zadaná uživatelem neexistuje (chyba 404). Zde je důležité i nastavení v souboru *routing.yml*, kde se nastaví, že pokud daná stránka není v aplikace dostupná, tak se zavolá tento kontroler. Využívá se regulární výraz `.*`, který značí jakýkoliv řetězec.

4.3.2 Přihlášení

Přihlášení je při vývoji webových stránek asi nejběžnější činnost na všech stránkách, proto je v Symfony možné doinstalovat a využívat FOSUserBundle. Instalace všech komponent probíhá přes `composer require friendsofsymfony/user-bundle " 2.0"`. Po fyzické instalaci jakéhokoliv Bundle je nutné ho umístit do souboru *app/AppKernel.php*, přesněji do funkce pro registraci pluginů.

Co přesně napsat do tohoto souboru se vždy najde v dokumentaci daného Bundle, například FOSUser, jako hodně využívaný má nejen stránky na GitHubu[5], ale i přímo v dokumentaci Symfony[6]. Po instalaci je nutné založit tabulku *fos_user* v databázi, všechny její nutné sloupce jsou součástí komponenty, takže stačí založit Entitu, která rozšiřuje třídu základního uživatele FOSUser a poté spustit příkaz na aktualizaci databáze v příkazovém řádku. Založí se třída, kam se napíše název dané tabulky a že *User* je rozšířen a obsahuje sloupce z třídy *BaseUser*, poté se vloží sloupec s identifikačním číslem a nastaví se automatické inkrementovaní.

Neméně podstatné je i nastavení v souboru *config.yml*, kde je nutné nastavit správně firewall, což je celkem individuální záležitost související s každým projektem, záleží nejen na nastavení frameworku, ale i serveru, na kterém program běží. Lze tam přidat třeba i jak dlouho si má cache paměť pamatovat přihlášení daného uživatelen, jaká Bundle je využívána a přesné cesty k přihlášení v souboru *security.yml*, například, že cesta pro přihlášení či cesta

[5] https://github.com/FriendsOfSymfony/FOSUserBundle
[6] https://symfony.com/doc/master/bundles/FOSUserBundle/index.html

k vytvořené tabulce. Poté už funguje přihlášení přes stránku *login* a další úpravy jsou již grafické.

Většina aplikací v dnešní době poskytuje kromě klasického přihlášení i možnost využít k přihlášení svůj účet Google. Existuje rozšíření HWIOAuthBundle[7], které kromě účtu Google umožňuje využít třeba účet Facebook nebo Twitter. Instalace je již výrazně složitější než u předchozího rozšíření. Liší se totiž verze od verze Symfony a návod z oficiální dokumentace nefungoval, jak by měl. Nutné je správně nastavit soubor *routing.yml*, aby framework přesně věděl, jaké stránky slouží k přihlášení. Oproti FOSUser, kde stačí jeden řádek s odkazem na speciální soubor se u HWI musí vysloveně uvést stránka pro přihlášení, spojení a přesměrování. Klíčová jsou opět nastavení v konfiguračních souborech jako správné nastavení firewallu, propojení s již existujícím Bundle FOSUser, protože i uživatelé, kteří se zaregistrují přes Google se zapisují do stejné tabulky.

Nejdůležitější částí nastavení je část, kterou je nutné vygenerovat na stránkách Google určených pro vývojáře[8], kam se zadá název projektu a poté doména, odkud se bude k tomu projektu možné přihlásit. Prvně je tam nutné zadat název domény https://deadliner.net a poté cestu, odkud probíhá přihlášení tedy https://deadliner.net/login/check-google (cesta je uvedená v souboru *security.yml*). Poté služba Google vygeneruje **Client ID** a **Client Secret**, což jsou dva jedinečné dlouhé kódy, které se zadají do odpovídajících položek v konfiguračním souboru *config.yml*. Je třeba opravdu přesně postupovat podle návodu v dokumentaci na správnou verzi, jinak se instalace může výrazně protáhnout, protože nastavení se vždy mírně liší. Nesmí se zapomenout například přidat `arguments: ['@fos_user.user_manager', google: googleID ]` tento řádek, který ještě upřesňuje, že ověřování přihlášení má být přes speciální id, které má každý uživatel služeb Google a je pak napojen na tabulku, kterou k přihlášení využívá FOSUser. Poté se ještě stejně jako předtím přidá kód do šablony Twig a je možné využívat přihlášení jako v obrázku 4.1.

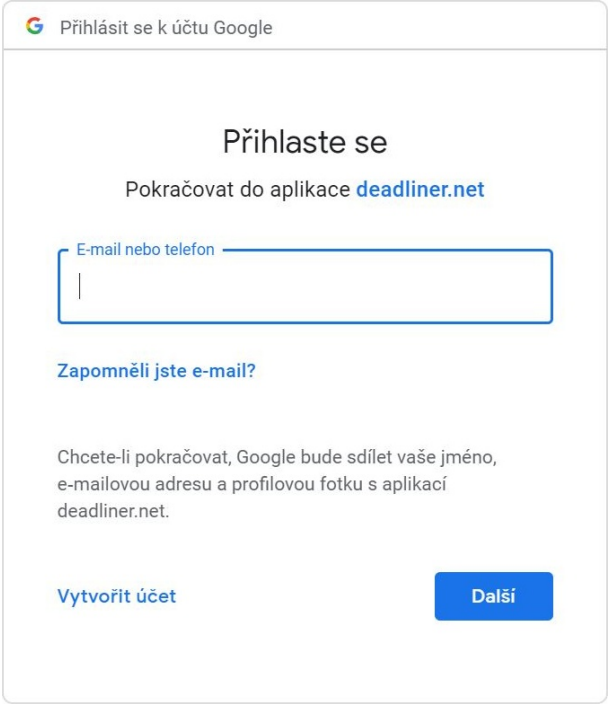
Pokud se někdo přihlásí, získá okamžitě přístup k částem systému, který nepřihlášený nevidí jako přidávání úkolů a kontaktů, jejich správa a úprava profilu. Existují tedy dvě role, první je Uživatel (USER), kterým se stane každý po přihlášení, nebo anonymní přístup (IS_AUTHENTICATED_ANONYMOUSLY) například pro stránku přihlášení nebo potvrzení (*confirm*).

4.3.3 Odesílání mailů

V souboru `parameters.yml` je nutné nastavit parametry pro SMTP přenos, který bude využíván. Je důležité vědět, jestli bude zabezpečený a odcházet přes port 587, s kódováním STARTLS nebo nezabezpečeně přes port 25. V projektu je odesílání mailů zásadní věcí, podle návrhu musí být vždy přesně znám odesílatel, který si sám nastaví jméno a email. Zabezpečený přenos v tomto případě nebude fungovat, protože není technicky možné zajistit, aby měl každý zaregistrovaný uživatel svůj účet u webhostingu, přes který by se přihlašoval.

V Symfony je možné pro odesílání využívat **Swift Mailer**[9], což je volně použitelná knihovna. Nejprve je vytvořena instance dané třídy, poté se zavolá jedna z funkcí například pro nastavení předmětu, odesílatele, příjemce, těla zprávy případně přílohy. Odeslání je pak provedlo přes funkci *send()* a instanci třídy tzv. *mailer*.

[7] https://github.com/hwi/HWIOAuthBundle
[8] https://console.cloud.google.com/apis/credentials
[9] https://swiftmailer.symfony.com



Obrázek 4.1: Po kliknutí na přihlášení Google se objeví následující stránka, která přesměrovává přímo na stránky Google, kde si uživatel zadá svoje jméno a heslo.

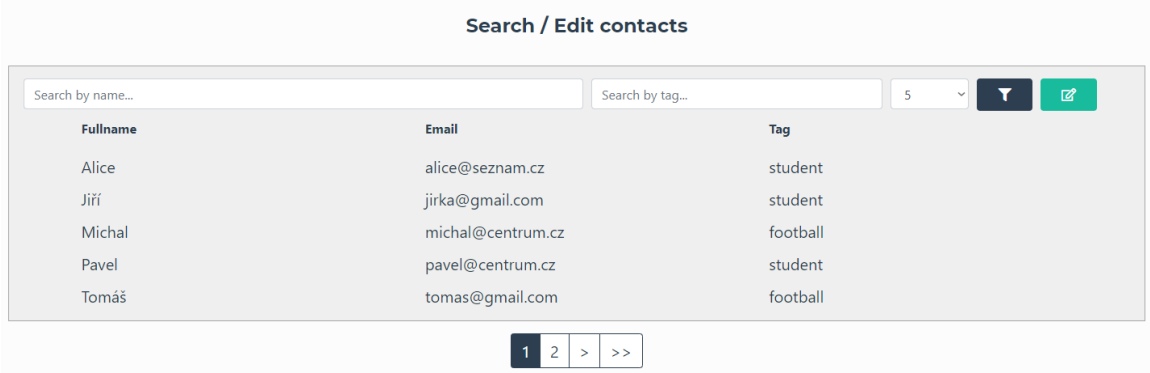
4.3.4 Hašovací klíče

V okamžiku, kdy uživatel zadá někomu úkol, tak je nutné vytvořit naprosto jedinečný odkaz, který bude obsahovat jednoznačné údaje k identifikaci zadavatele i plnitele úkolu a identifikační číslo úkolu. K tomu je připojeno speciální číslo z tabulky, která spojuje kontakt s úkolem a kde se zaznamenává splnění úkolu. K zašifrování se využívá knihovna Hashids-Bundle[10]. V nastavení se zvolí délka kódu i možné znaky abecedy, které se mohou k šifrování použít. Tentokrát se změny musí provést přímo v instalační složce *vendor* a v nastavení daného Bundle. V tomto projektu byla zvolena délka 64 znaků a je možné využívat velká i malá písmena a čísla. V Kontroleru se nejprve použije funkce *encode* s parametry, které mají být zakódovány (identifikační čísla z databáze, z toho je vytvořen hash kód `o1p567nqAdB10g3rwvLEOpxXzDZOOvumjHvIDeJlkM4QoGjbP9Y8Wm2KR5VNP6Rb`, na který se pro dekódování využije funkce *decode*, takže po načtení odkazu je možné přesně určit, pro koho je určen.

4.3.5 Další použité Bundle

Jednou ze základních věcí, které weby v dnešní době poskytují, je stránkování a řazení seznamů. Aby uživatel neměl všechno na jedné nekonečné stránce, zároveň, aby mohl najít nějaký starší záznam, případně si seznam seřadit. Rozšíření Symfony, které toto umožňuje, se jmenuje KNPPaginatorBundle[11], instalace probíhá standardně dle návodu v dokumentaci. Využívá se v projektu na stránce kontaktů a úkolů, jako na obrázku 4.2.

[10] https://github.com/lsv/hashids-bundle
[11] https://github.com/KnpLabs/KnpPaginatorBundle



Obrázek 4.2: Ukázka z finální verze aplikace, kde je stránka s přehledem kontaktů. Rozmístění prvků bylo ponecháno jako v návrhu (obrázek 3.5) a zmizel sloupec s fotkami. Dole je vidět ukázka stránkování s využitím KNPPaginator.

Nejprve je vytvořen speciální databázový dotaz, který data získá v požadovaném formátu, aby mohla být dále tříděná. Poté je zavolána integrovaná funkce *paginate* se třemi parametry – data z databáze, jaká stránka bude zobrazena jako první a limit, kolik je možné vypsat položek na jednu stránku. Přebývající hodnoty se přesunou na další stránku, navigace se objeví podle toho, kam je v šablonovacím systému Twig vložena funkce *knp_pagination_render*. V případě, že je třeba nějaké položky vyhledat, je opět použita databáze, ze které se data čtou přes *getAlnum* a jako parametr se použije název vstupního pole (input) z šablony. Poslední využitou funkčností je třídění sloupců. V šabloně Twig daný sloupec využije funkci *knp_pagination_sortable*. V ní je název hledaného sloupce v databázi, poté zda má být řazení vzestupné či sestupné přes databázovou funkci *ORDER BY*.

Zajímavostí je přidání vlastního kontroleru, aby se daly funkce jiného kontroleru volat z jakékoliv třídy. Přidá se nová služba (service), kde se zvolí vlastní název, který se používá při volání (get).

4.3.6 Vícejazyčnost

V Symfony je také možné relativně jednoduše implemenovat vícejazyčný web a to bez dalších instalací. V nastavení frameworku je nutné nastavit správně parametr *translator*, kde se nastaví výchozí jazyk. Přes parametr *locale* se pak v kontrolerech určuje cesta k souboru, tím i jazyk. Všude se používají relativní adresy, protože umožňují rychlou přenositelnost mezi hostingy bez měnění velkého množství absolutních odkazů. Překlady pak mohou vypadat následovně

```
language: 'cs'
flash.addcontact: 'Nový kontakt byl úspěšně přidán'
flash.editcontact: 'Kontakt byl upraven'
flash.deletecontact: 'Kontakt byl smazán'

language: 'en'
flash.addcontact: 'New contact has been added'
flash.editcontact: 'Contact has been changed'
flash.deletecontact: 'Contact has been deleted'
```

V Twig šabloně se k nim přistupuje přes zadání názvu překladu a poté zadání příkazu „trans“, podle jazykové verze se pak vybere, zda chci mít český (cs) nebo anglický (en) jazyk, na základě toho se zobrazí správná verze.

Aby se daly překlady používat i v kontrolerech, je nutné použít knihovnu TranslatorInterface, přes kterou je možné využívat třídu i v PHP kódu, která implementuje nezbytnou funkci *trans*.

4.4 Implementace skriptů

Pro některé úkony je dobré využít skriptovací jazyk JavaScript, který umožňuje práci přímo na straně prohlížeče. Jde především o přenos kontaktů z dostupných do přidaných, kde se do HTML prvku *select* přidávají a odebírají prvky podle toho, na které uživatel klikne. K tomu je třeba více funkcí, aby mohlo být implicitně určeno, co se má kdy kam přesunout.

Využívá se třeba při najetí (onhover) na nějakou položku, kdy je jednoduché dočasně změnit stylování daného prvku jako v obrázku 5.4. Případně pro automatické obnovování stránky po určitém časovém úseku, aby uživatel měl v aktuální záložce přesné údaje, kdo již potvrdil.

Volání funkcí probíhá po specifikované činnosti uživatele, například po kliknutí (onclick). Užitečným příkazem je **confirm**, který vrací *true* pouze v případě, že uživatel danou činnost potvrdí po vyskočení okna; jedná se o nevratné změny jako třeba smazání položky.

4.4.1 Implementace kaskádových stylů pomocí šablon Twig

Vzhled je implementován knihovnou Bootstrap 3, která obsahuje soubory CSS a Javascriptu, celá struktura musí být v hlavičce HTML souboru.

Výhody knihovny Bootstrap jsou, že často používané prvky jsou nastylované tak, aby se pouze malou úpravou daly hned používat a při tom vypadaly dobře. Jedná se především o tlačítka, klasická menu, kde poté funguje i responzivní design. Výchozí šablona obsahuje pouze bílé pozadí, proto existuje velké množství šablon[12], kde si člověk najde přímo styl, který mu na danou stránku vyhovuje. Jedná se o open-source, takže je možné kód libovolně upravovat. Vlastní prvky samozřejmě předprogramovány nejsou, takže se používá soubor *style.css*, kam byl implementován vzhled seznamů úkolů i kontaktů nebo přehled jednotlivých úkolů a responzivní design těchto prvků manuálně přes tag `@media`.

Vzhled byl nejprve implementován podle původních návrhů, postupným testováním byly zjištěny nedostatky, především původní barvy se testerům vůbec nelíbily. Nakonec byla zvolena šablona Freelancer[13], která byla zmodifikována. Dlouho se řešilo i rozmístění jednotlivých prvků na různě velkých obrazovkách, k ladění responzivity byla využita stránka Quirktools[14]. Každému vyhovuje něco jiného, ale většina testerů ocenila větší mezery mezi prvky, dostatek místa pod nadpisy a jasné oddělení nesouvisejících údajů, například v původním návrhu byla matoucí možnost nastavení data mezi stejně velkými tlačítky *Upozornit později* a *Splněno*, protože někdo nevěděl, k čemu patří.

Užitečnou věcí je v šabloně u práce s daty příkaz *ago*, který si sám přepočítá datum vůči dnešnímu dni a poté zobrazuje relativní časové údaje, které často člověku řeknou výrazně víc než pouhé datum. Jako například, že konečný termín úkolu je **zítra**. Toto rozšíření vyžaduje, aby všechny datumy v databázi byly typu *datetime*.

[12] https://startbootstrap.com/themes/
[13] https://startbootstrap.com/themes/freelancer/
[14] http://quirktools.com/screenfly/

Flash message je rozšíření, které uživateli oznamuje, zda se jím vykonaná činnost povedla. Je předprogramováno v šablonovacím systému Twig s využitím frameworku Bootstrap. Používá se většinou po přidání dat do databáze nebo odeslání mailu, uživateli se v horní části stránky zobrazí informativní zpráva, která při dalším načtení stránky zmizí.

Kapitola 5

Nasazení do provozu a testování

5.1 Nasazení systému do online provozu

Nejdříve byla celá aplikace vyvíjena pouze na lokálním počítači s vlastním serverem, pro účely testování byly různé verze postupně nasazovány na doménu druhého řádu. Po několika měsících testování se vše spustilo na ostré doméně https://deadliner.net/. Jako první po registraci domény byl vyzkoušen hosting Wedos, který má bohužel velká omezení při posílání zpráv přes protokol SMTP. Na hostingu je vyžadováno speciální nastavení, které není nikde pořádně popsáno a ani po komunikaci s podporou se nepodařilo zprovoznit, aby odesílatel zprávy mohl být kdokoliv a nemuselo se používat šifrované spojení přes port 587 s přihlášením konkrétního uživatele. Tento problém nakonec musel být vyřešen přechodem na webhosting C4, kde odesílání přes port 25 funguje.

Pokud jsou správně nastaveny relativní adresy a nepoužívají se ty absolutní, přechod na nový hosting vyžaduje pouze změnu databázových přihlašovacích údajů a vypnutí debug módu ve složce web, která je klíčová pro všechny soubory JavaScriptu i CSS, ukládají se tam také přílohy mailů. Na webhostingu je nutné, aby složku *web* bral jako výchozí (root). Buď poskytuje přímo rozhraní, kde se nastaví, která složka je kořenová (jako C4) nebo je nutné nastavit soubor *.htaccess*, kde se nastavují různá pravidla týkající se přesměrování kdekoliv na stránkách, případně třeba ze staré domény na novou.

5.2 Testování

Velmi rozsáhlou částí bakalářské práce bylo testování, kromě ladění typických programovacích chyb jako neošetření toho, že žádné úkoly či kontakty neexistují, na což obvykle stačí podmínka. Bylo také provedeno velké množství uživatelského testování na spuštěné aplikaci, k tomu byly vyhodnocovány různé dotazníky a prováděno testování použitelnosti na dobrovolnících.

V první fázi byla navržena především databáze a nejdůležitější stránky, které byly konzultovány pouze s pár lidmi. Postupně se testovala důležitost jednotlivých položek v databázi a zároveň také, co má jaká stránka obsahovat a co uživatel ocení nejvíc.

Poté úspěšném spuštění přihlašování a odladění největších chyb, byla aplikace prakticky využita několika prvními uživateli, kteří rozeslali své úkoly asi čtyřiceti lidem. K tomu byl přiložen krátký dotazník ohledně uživatelské přívětivosti stránky, co by se dalo vylepšit.

Postupně se přicházelo i na chyby v implementaci, které nebyly při prvním testováním odhaleny a především se přidávala nová funkčnost, která testerům chyběla.

Jako první se opravovala databáze, která celkově prošla mnoha změnami, než se dostala do verze v návrhu na obrázku 3.2. Oproti původnímu návrhu byla v tabulce uživatelů odstraněna položka fotky, kde měl mít každý kontakt svůj portrét, který měl ulehčovat orientaci v seznamu kontaktů, ale nikdo to nevyužíval. Jinak byly tabulky výrazně rozšířeny, protože při návrhu ještě nebylo úplně jasné, jak se bude aplikovat tabulka zadavatelů úkolů, která se nakonec využila z knihovny FOSUserBundle.

Z dotazníků vyplynulo, že by pár uživatelů ocenilo prioritu nebo možnost více příloh. Postupně se ukázalo, že jsou obě rozšíření nepotřebné. V případě, že je nutné odeslat více souborů, tak se dá použít nějaké úložiště a do emailu přidat odkaz. Priorita zajímala pouze malý okruh uživatelů a spíše jen zabírala místo ve formuláři pro přidání úkolu.

V dotaznících také padl návrh na možnost přidání vzkazu odesílateli úkolu. To nakonec po konzultaci s dalšími testery implementováno nebylo, protože email se odesílá přímo z adresy zadavatele. Takže pokud je nutná odpověď, je možné ji poslat přímo z emailového klienta.

Jednoho z testerů napadla možnost odmítnutí úkolu, která zatím implementována nebyla, protože pokud již zadavatel úkol pošle, tak chce, aby ho dotyčný splnil. Pokud by více uživatelů projevilo o danou funkci zájem, tak může být v budoucnu přidána.

Dotazníky také posloužili k ladění detailů v aplikaci. Jedná se například o tzv. „našeptávání“ ve formuláři přidání skupiny ke kontaktu. Pokud už v databázi existuje daný název skupiny, tak pole formuláře typu *datalist* nabízí možnosti, které se už v databázi u daného uživatele nachází.

Postupně bylo laděno i řádkování, aby si uživatel mohl nastavit, kolik řádků chce podle velikosti monitoru. Ze dvou možností nastavení přes Cookies a nastavení profilu, byla vybrána ta druhá, aby v případě, že se přihlásí z jiného zařízení, mu tam číslo zůstalo. Užitečnou věcí se ukázalo také, pokud se například zadavatel úkolů přihlásí přes Google, tak by si rád změnil email, ze kterého se budou úkoly odesílat a nastavil jméno.

Velkou především grafickou proměnou procházely postupně všechny stránky. Ukázalo se jako nutné, aby se správně objevovaly nové řádky a z odkazů byly opravdu klikatelné odkazy otevírány v novém okně kvůli přehlednosti zpráv. Důležité údaje se odlišily barevně jako na stránce potvrzení úkolů, která postupně prošla proměnou oproti původnímu obrázku 5.1, například se zpřehlednilo, kdo posílá úkol komu a kdy. Zvýraznil se konečný termín červenou barvou, pokud už je po dedlajnu, jinak je zelený jako v obrázku 5.2. Zároveň tam bylo přidáno tlačítko, aby se plnitel úkolu mohl vrátit zpátky k akci, například pokud na *Splněno* klikl omylem. Nadpisy byly vybarveny do šeda, aby jim uživatel nepřikládal velkou váhu.

Postupnými změnami prošla i grafika emailu, který se odesílá. Z původního odkazu bylo uděláno tlačítko, které vypadá výrazně důvěryhodněji jako v obrázku 5.3. To bylo nutné vyladit v nejčastěji používaných klientech, tedy Thunderbirdu a Outlooku, kde právě produkt Microsoftu potřebuje speciální označení a nastavení, aby HTML zpráva vypadala, jak má. Do předmětu také byl přidán název aplikace, aby uživatel hned věděl, že mu přišel úkol. Úvodní text byl také laděn dlouho, aby přesně vystihoval myšlenku aplikace a uživatel neměl pocit, že se jedná o spam.

Celkově byly texty a jejich překlady stále přepisovány, protože prvotní verze nevystihovaly správně a srozumitelně, co přesně znamenají, takže to uživatele mátlo. Občas se chybou v programování objevovaly dokonce úplně špatné texty, což bylo postupně vyladěno. V tlačítkách byly texty nahrazeny standardními ikonkami, na které jsou uživatelé zvyklí, takže například archivace úkolu byla nahrazena hvězdičkou jako v obrázku 5.4, která, pokud je vybarvená, tak je úkol aktivní. Uživatel je na něco podobného zvyklý z jiných aplikací,

Obrázek 5.1: Ukázka jedné z prvních verzí návrhu stránky, na které uživatelé potvrzují splnění úkolu. Stránka byla navržena podle návrhu, ale během testování se zjistilo, že není přehledná. Nebylo na první pohled jasné, jaké údaje jsou nejdůležitější, ani nebyly klikatelné odkazy.

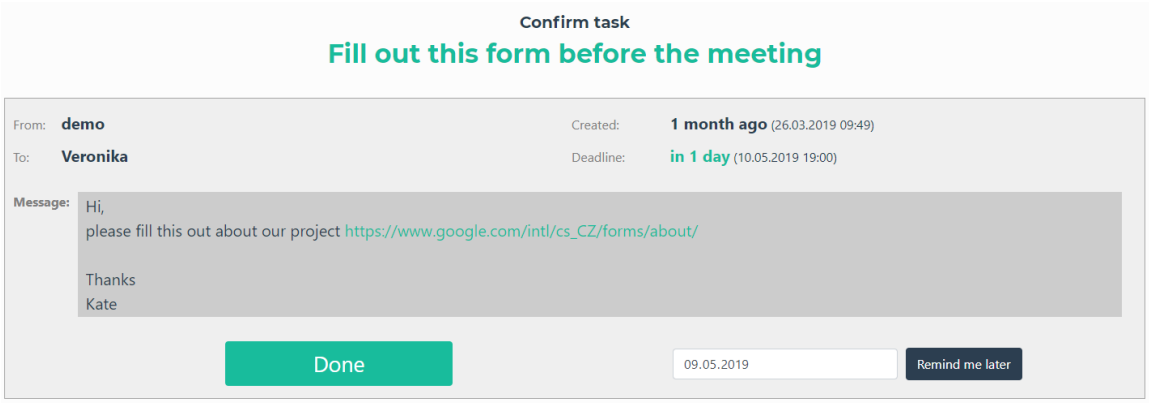
takže přesně ví, co od toho čekat, neaktivní úkoly se navíc řadí dolů a jsou vybarveny do šeda.

Testování proběhlo i ve výběru výchozího řazení, u úkolů se vybíralo mezi datem vytvoření a konečným termínem, více lidem vyhovovalo řazení podle data vytvoření a u kontaktů podle abecedy.

V seznamu všech úkolů byl největší změnou rozložení informací do dvou řádků, čímž došlo k výraznému zpřehlednění a také odstranění většiny tlačítek a jejich přesunutí do přehledu konkrétního úkolu. Byla ponechána pouze archivace, také rozkliknutí úkolu bylo umožněno kdekoliv na zvýrazněné ploše přes CSS třídu *:hover*.

Všechny změny je nutné kontrolovat v často používaných prohlížečích, ve Firefoxu na rozdíl od jiných prohlížeů při lichých hodnotách *border* dělí z neznámého důvodu dané hodnoty číslem tři. To pak dělá problémy při dopočítávání ostatních hodnot (jako *padding*), což mělo za následek „uskakování“ prvků. Bylo nutné nastavit všechny požadované hodnoty na sudá čísla. Na většině stránkách také byly odstraněny nic neříkající nadpisy jako například „Seznam úkolů“, kdy je uživateli naprosto jasné, že je na seznamu úkolů a zbytečně zabírají místo. Jinde byly ponechány, ale výrazně zmenšeny a zesvětlena barva, aby je lidský mozek řadil jako méně podstatnou informaci.

Přehled daného úkolu také prošel v čase výraznými obměnami, postupně byly přidány a zvýrazněny různé údaje, především se jedná o přehled lidí, kteří mají úkol plnit. Ke každému byla přidána dvě tlačítka jako v obrázku 5.5, jedno na smazání daného člověka a druhé na poslání upozornění přímo jemu. Ze seznamu úkolů se sem přesunula tlačítka na poslání emailu všem, editace či smazání úkolu. Z testování vyplynulo, že by se hodila i duplikace daného zadání. Zásadní změna se odehrála i v odesílání mailů, v původní verzi okamžitě po přidání úkolu se odeslaly všechny maily. Ale zjistilo se, že je lepší nejdříve



Obrázek 5.2: Vylepšená verze stránky na potvrzování splnění úkolu po testování. Bylo přidáno více barev pro rozlišení důležitosti, mírně upraveno rozmístění prvků a přidáno více mezer, aby prvky měly více vzdušnosti.

přidat úkol, všechno zkontrolovat a až poté odeslat upozornění. Také zadavatel vidí, kdy naposledy bylo zasláno upozornění danému člověku, případně kdy úkol splnil.

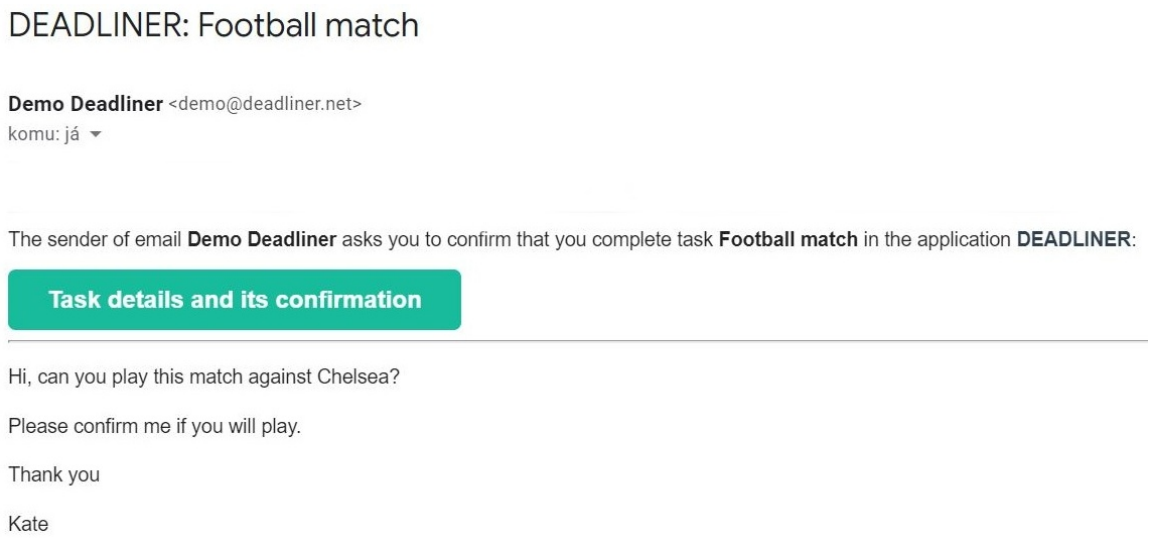
V kontaktech se také nahradil text tlačítek ikonkami a zpřehlednil se i formulář na přidávání nových záznamů. Zároveň výpis je pouhý přehled oproti původním formulářovým prvkům, viz obrázek 5.6, které je možné otevřít po přepnutí do editačního módu, kde se kontakt upraví. Zvyšuje to především přehlednost stránky. Při přidávání byla také ošetřena validita emailu. Pár testerů navrhovalo možnost importu kontaktů například z programu Microsoft Outlook, což nakonec nebylo implementováno kvůli bezpečnosti, aplikace by mohla být jednoduše využita k odesílání spamů.

Stránka potvrzení úkolů byla snad nejvíce testována i laděna, protože na tuto stránku se dostalo nejvíce lidí, v dotaznících byl i návrh na přímé potvrzení úkolu v mailu. To bylo nevýhodné hlavně při prvním setkání s aplikací, protože když někomu přišel email poprvé, tak nevěděl, co od toho čekat, klikl na tlačítko a měl označeno jako splněno, aniž by cokoliv udělal. Vzhled prošel pár úpravami, především došlo ke zmenšení tlačítka splnit úkol a výrazné odlišení od data s potvrzením později, protože ze začátku to pár lidem připadalo, že tam nastavují datum splnění.

Úvodní stránka byla navržena až výrazně později, původní obsahovala pár obrázků z používání aplikace a přihlášení. Obrázky byly ale celkem velké, takže příchozí uživatel v případě, že měl menší monitor, formulář na přihlášení hned neviděl, což bylo celkem matoucí. Proto se úvodní nápis výrazně zmenšil a video s návodem a další informace byly přesunuty dolů. Také byl do zápatí přidán formulář, kde je možné nahlásit chyby nebo připomínky.

Nejvíce připomínek přišlo v úvodních dotaznících na špatný vzhled, především nevhodné kombinace barev, které byly nahrazeny stávajícími z šablony Freelancer. Na základě testování se upravovaly i velikosti nadpisů a šířky tlačítek. Velmi často se objevil prvek, který nebyl správně vycentrován, takže bylo nutné na základě návrhů upravit zarovnání například u tlačítka *Potvrdit splnění úkolu*.

Bylo zjištěno, že vyskakovací okna, kdy se po uživateli chce, aby potvrdil danou akci, nejsou třeba například u archivace úkolů, ale stačí to jen u smazání, kde se jedná o nevratnou činnost.



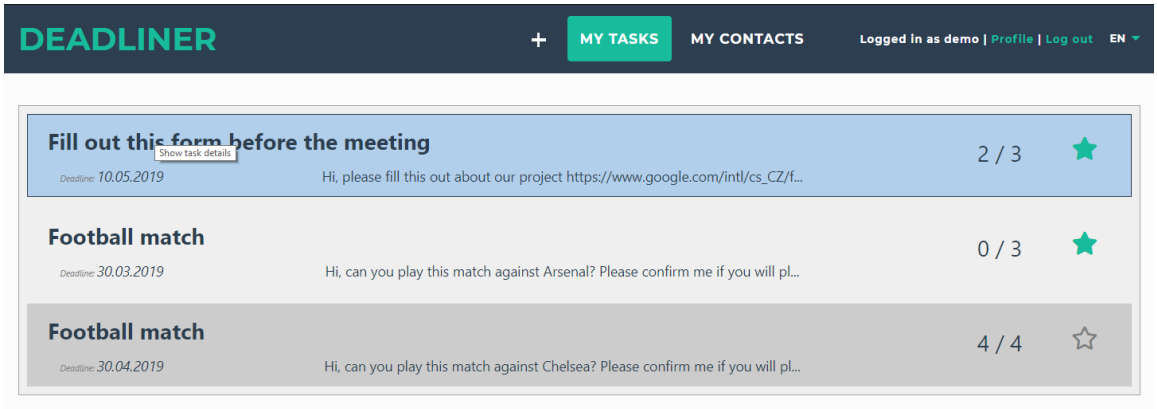
Obrázek 5.3: Konečná verze odesílaného mailu, který je zobrazen v Google Mailu. Odkaz byl schován do tlačítka, už v předmětu je jasně rozlišeno, z jaké aplikace daný mail přišel. Úvodní text je jasně odlišen čarou.

V mobilní verzi jsou prvky dle zvyklostí řazeny pod sebe a některé na mobilu špatně použitelné ani nejsou vidět. Předpokládá se, že uživatel se na mobilu chce spíš jen podívat, jak to vypadá s jeho úkolem než zakládat nový.

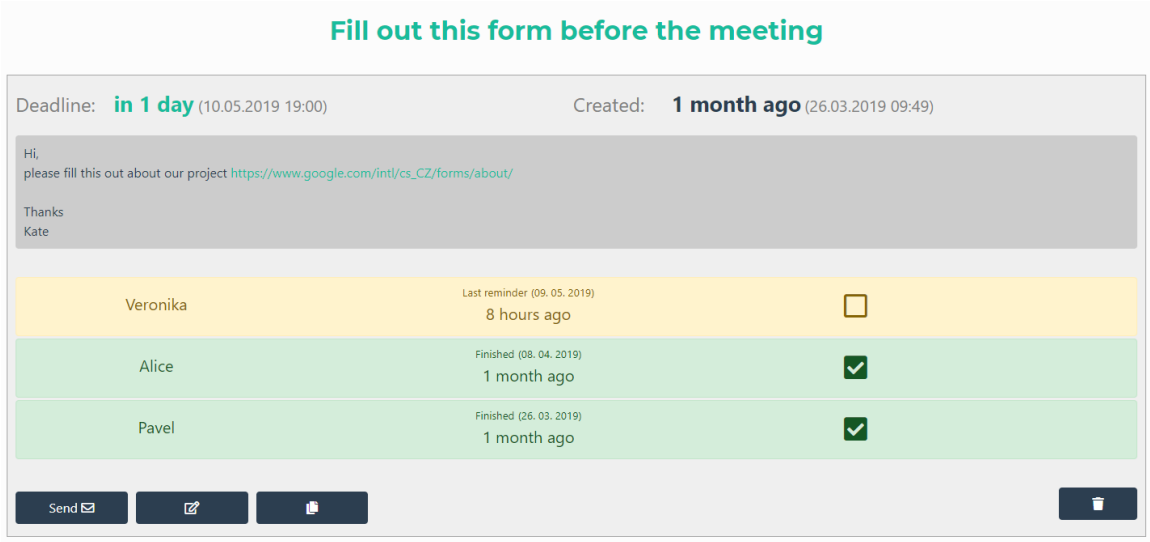
Z dotazníků například vyšlo, že většině lidí, kteří potvrzují splnění úkolu, nový způsob v aplikaci vyhovuje výrazně víc než psaní klasického mailu. Z celkem 32 odpovědí (ze 2 dotazníků) se 22 lidem Deadliner líbí, 6 lidem je to jedno a 4 raději napíší klasický email.

Jednou z dalších testovaných oblastí v dotaznících bylo přihlášení, kde 15 z 34 testovaných dalo přednost právě přihlášení pomocí Google účtu, 10 by si vytvořilo nový účet a 9 lidí by ocenilo možnost přihlášení přes účet na Facebooku.

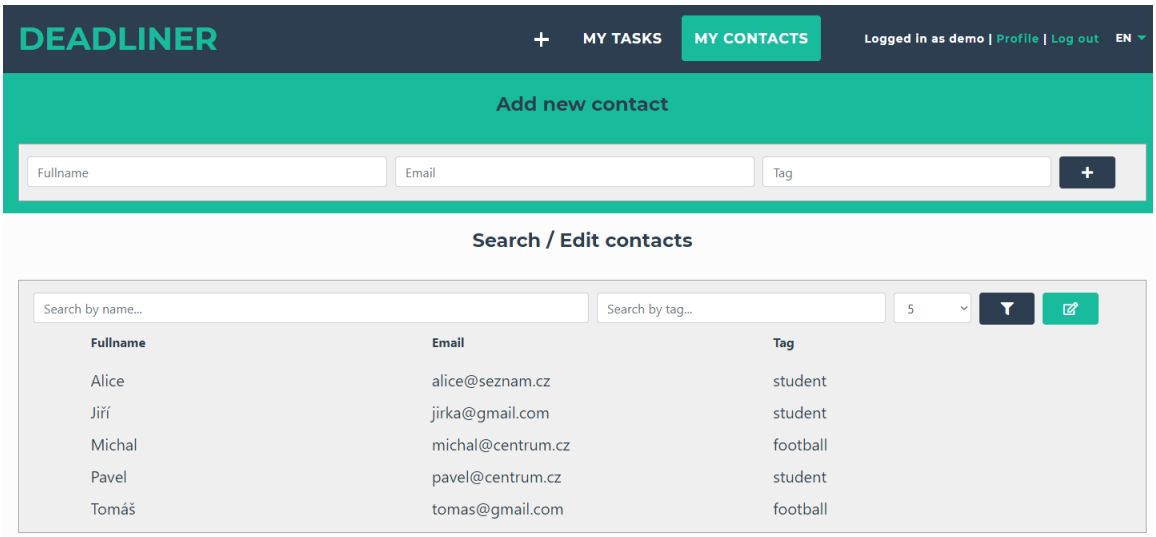
Další otázky se týkaly především uživatelského přívětivosti jednotlivých stránek, většina lidí zaškrtla možnost, že se jim celkový vzhled líbí a ohodnotili ho známkou 2. U konkrétních stránek se většinou rychle zorientovali, případně měli problém s barevnými kombinacemi. Někteří vůbec nepochopili účel některých stránek. Jednalo se ještě o vzhled před přidáním ikonek do tlačítek, občas nebylo pochopeno, k jakému účelu je dané tlačítko. Především byla matoucí archivace úkolu.



Obrázek 5.4: Současná podoba po testování stránky s úkoly, po najetí na jeden z úkolů je možné ho rozkliknout případně při kliknutí na hvězdičku vpravo archivovat.



Obrázek 5.5: Konečná verze stránky s přehledem vybraného úkolu. Je možné jasně vidět zadání a kdo už úkol splnil, kdo ne a datum, kdy ho splnil, případně kdy si nechal naposled poslat upozornění. Tlačítka, kde místo textu jsou ikonky slouží k editaci, smazání či odeslání emailového upozornění všem. U jednotlivých kontaktů je možné je smazat či poslat upozornění pouze jednomu.



Obrázek 5.6: Ukázka z finální verze aplikace, kde je stránka s přehledem kontaktů. Rozmístění prvků bylo ponecháno jako v návrhu (obrázek 3.5), pouze přidání je v jednom řádku a zmizel sloupec s fotkami.

Kapitola 6

Závěr

Na základě zadaných požadavků a dat získaných testováním jsem navrhla a implementovala webovou aplikaci pro správu dedlajnů. Víc jak půl roku bylo věnováno uživatelskému testování a iterativnímu vylepšování aplikace Deadliner, postupně se měnilo třeba rozestavení prvků.

Na začátku byly prozkoumány současné webové technologie, jakým způsobem probíhá vývoj aplikací. Kromě toho bylo vyhledáno několik podobných aplikací a důkladně prostudováno jakým způsobem fungují. Nejprve byl testován návrh rozhraní i databáze a analyzovány požadavky. Poté byla aplikace implementována pomocí vybraného frameworku PHP a Bootstrap. Pak uživatelé aplikace po dobu přibližně půl roku v praxi testovali uživatelské rozhraní, co jim chybí za funkce, které jsou zbytečné a ladil se také celkový vzhled aplikace. Zadání práce bylo splněno.

V nejbližší budoucnosti bude hlavní cíl pokusit se aplikaci rozšířit mezi více lidí, v současné době ji aktivně používá pouze pár jedinců. K tomu budou využity sociální sítě jako Facebook nebo Twitter. Aplikace má výhodu, že jakmile ji začne využívat pár lidí, tak tím, že budou posílat úkoly dalším, ji vlastně sami budou dělat reklamu. V současné době je aplikace ve finální fázi zveřejněná na https://deadliner.net, další vylepšení budou záviset na podnětech uživatelů. Vzhledem k tomu, že se jedná o aplikaci, která pracuje s termíny, dalo by se do budoucna uvažovat o rozšíření pomocí napojení na kalendáře uživatelů.

Literatura

[1] Doodle (website). [Online; navštíveno 05.02.2019].
URL https://en.wikipedia.org/wiki/Doodle_(website)

[2] Rozdíl mezi UI a UX: definice pojmů. [Online; navštíveno 06.02.2019].
URL http://blog.iquest.cz/2017/11/rozdil-mezi-ui-ux-definice-pojmu.html

[3] Symfony. [Online; navštíveno 06.02.2019].
URL https://en.wikipedia.org/wiki/Symfony

[4] Typy uživatelských rozhraní a jejich specifika. [Online; navštíveno 08.02.2019].
URL https://wikisofia.cz/wiki/Typy_uživatelských_rozhraní_a_jejich_specifika

[5] Gasston, P.: *Moderní web*. Brno: Computer Press, 2015, ISBN 978-80-251-4345-2.

[6] Krug, S.: *Don't make me think*. New Riders, 2014, ISBN 978-0-321-96551-6.

[7] Kumar, A.: *Sencha MVC architecture*. Birmingham : Packt Publishing, 2012, ISBN 978-1-84951-888-8.

[8] Matoušek, P.: *Síťové aplikace a jejich architektura*. Brno: VUTIUM, 2014, ISBN 978-80-214-3766-1.

[9] Ponkrác, M.: *PHP a MySQL: bez předchozích znalostí*. Brno: Computer Press, 2007, ISBN 978-80-251-1758-3.

[10] Sharkie, C.; Fisher, A.: *Responzivní webdesign: okamžitě*. Brno: Computer Press, 2015, ISBN 978-80-251-4384-1.

[11] Unger, R.; Chandler, C.: *A project guide to UX design: for user experience designers in the field or in the making*. Berkeley: New Riders, 2012, ISBN 978-0-321-81538-5.

[12] Řezáč, J.: *Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů*. Jihlava: Baroque Partners, 2014, ISBN 978-80-87923-01-6.