

Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství



Bakalářská práce

**Návrh a implementace webové aplikace pro
neziskovou organizaci**

Štěpán Pešout

© 2021 ČZU v Praze

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Štěpán Pešout

Systémové inženýrství a informatika
Informatika

Název práce

Návrh a implementace webové aplikace pro neziskovou organizaci

Název anglicky

Design and implementation of web application for non-profit organization

Cíle práce

Práce se zabývá analýzou požadavků, návrhem a implementací webové aplikace s databází v neziskové organizaci, která bude sloužit zejména pro evidenci údajů o fyzických a právnických osobách, jež mají určitý vztah k této organizaci. Další důležitou funkcí bude správa historie interakcí těchto subjektů, jakými jsou například účasti na akcích, sponzorské dary či e-mailová korespondence.

Metodika

Zpracování teoretické části práce spočívá ve studiu odborných informačních zdrojů. Na základě nich pak budou popsána teoretická východiska pro zpracování praktické části práce.

Po analýze požadavků na funkcionality bude následovat návrh včetně rozvržení uživatelského rozhraní a struktury databáze, poté implementace. Pro návrh a analýzu bude využito standardních metod interakčního designu a softwarového inženýrství, implementace bude provedena v jazyce PHP s šablonovacím systémem Twig za využití MySQL databáze. Klientská část bude vytvořena v JavaScriptu ES6, dále v CSS3 s preprocesorem Less a HTML5. Hotová aplikace bude nasazena a otestována. Budou shrnuty poznatky získané během jejího vývoje a navrženy případné další možnosti jejího rozvoje do budoucna.

Doporučený rozsah práce

43

Klíčová slova

nezisková organizace, webová aplikace, UI specifikace, PHP, MySQL, JavaScript


Doporučené zdroje informací

GRUBER, Martin. Mistrovství v SQL. Praha: Softpress, 2004. ISBN 80-864-9762-3.

JavaScript. MDN Web Docs. Dostupné on-line: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

KRUG, Steve. Nenuťte uživatele přemýšlet!: praktický průvodce testováním a opravou chyb použitelnosti webu. Brno: Computer Press, 2010. ISBN 978-80-251-2923-4.

PHP Documentation. Dostupné on-line: <https://www.php.net/manual>.



Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 12. 3. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 12. 3. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Návrh a implementace webové aplikace pro neziskovou organizaci“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2021

Poděkování

Rád bych touto cestou poděkoval Ing. Josefu Pavlíčkovi Ph.D. za vedení práce a vstřícnost při konzultacích; dále také všem zaměstnancům ZO ČSOP Vlašim, kteří se podíleli na návrhu a testování aplikace za jejich čas a ochotu.

Návrh a implementace webové aplikace pro neziskovou organizaci

Abstrakt

Tato bakalářská práce se zabývá analýzou požadavků, návrhem a následnou implementací webové aplikace s databází v neziskové organizaci.

Teoretická část práce obsahuje obecná pravidla návrhu uživatelského rozhraní a popis fungování webových aplikací. Dále jsou zde vymezeny pojmy týkající se relačních databází a vývoje softwaru za použití JavaScriptu, PHP, MySQL a dalších technických prostředků.

V praktické části je provedena analýza požadavků, po níž následuje návrh databáze a uživatelského rozhraní. Poté je popsána samotná implementace serverové a klientské části webové aplikace. Po nasazení je otestováno uživatelské rozhraní, shrnuty poznatky získané během vývoje a zhodnocena správnost počátečních předpokladů. Nakonec jsou navrženy možnosti dalšího rozvoje do budoucna.

Klíčová slova: nezisková organizace, webová aplikace, UI specifikace, PHP, MySQL, JavaScript

Design and implementation of web application for non-profit organization

Abstract

The main goal of this bachelor thesis is to analyse requirements, design and implement a web application with a database for a non-profit organization.

The theoretical part of the thesis contains general rules of user interface design and a description, how web applications work. Furthermore, the terms related to relational databases and software development using JavaScript, PHP, MySQL and other technologies are defined here.

In the practical part, there is the analysis of requirements, followed by the design of the database and user interface. Then the implementation of the server and client part of the web application is described. After deployment, the user interface is tested, the knowledge gained during development is summarized and the correctness of the initial assumptions is evaluated. Finally, the possibilities of further development in the future are proposed.

Keywords: non-profit organization, web application, UI specification, PHP, MySQL, JavaScript

Obsah

1 Úvod	13
2 Cíle práce a metodika	14
2.1 Cíle práce	14
2.2 Metodika	14
3 Teoretická východiska	15
3.1 Interakční design	15
3.1.1 Pět dimenzí interakčního designu	15
3.1.2 Design uživatelské zkušenosti (UX)	16
3.1.3 Design uživatelského rozhraní (UI)	16
3.1.4 UI specifikace	16
3.1.4.1 Persona	16
3.1.4.2 Případ užití	17
3.1.4.3 Scénář	17
3.1.4.4 Drátěný model	17
3.1.5 Prototyp	18
3.1.5.1 Low-fidelity prototyp	18
3.1.5.2 High-fidelity prototyp	18
3.1.6 Testování použitelnosti aplikace	19
3.1.6.1 Kvantitativní testování	19
3.1.6.2 Kvalitativní testování	19
3.2 Webová aplikace	20
3.2.1 Klientská část	21
3.2.2 Serverová část	21
3.2.3 Web API	22
3.2.3.1 Metody pro přístup ke zdrojům	22
3.2.3.2 Stavové kódy HTTP	23
3.2.4 Jazyk HTML	23
3.2.4.1 Tagy, elementy a atributy	23
3.2.4.2 Anatomie HTML dokumentu	24
3.2.5 Kaskádové styly (CSS)	25
3.2.5.1 Preprocesor Less	25
3.2.5.2 Framework Bootstrap	26
3.2.6 JavaScript	26
3.2.7 PHP	27
3.2.7.1 Šablonovací systém Twig	28
3.2.8 Relační databáze	28

3.2.8.1 Klíče	28
3.2.8.1 Normalizace databáze	29
3.2.8.2 Jazyk SQL	30
3.2.8.3 MySQL	30
4 Vlastní práce	31
4.1 Postup návrhu a vývoje	31
4.1.1 Návrhová část	31
4.1.2 Implementační část	32
4.1.3 Testovací část	32
4.2 Analýza požadavků	32
4.2.1 Seznam funkčních požadavků	33
4.2.1.1 Registrace	33
4.2.1.2 Přihlášení	33
4.2.1.3 Odhlášení	33
4.2.1.4 Evidence údajů o lidech	33
4.2.1.5 Evidence údajů o organizacích	33
4.2.1.6 Evidence údajů o sponzorských darech	33
4.2.1.7 Evidence údajů o akcích	33
4.2.1.8 Evidence údajů o typech akcí	34
4.2.1.9 Zadávání údajů do evidence	34
4.2.1.10 Editace údajů v evidenci	34
4.2.1.11 Smazání údajů v evidenci	34
4.2.1.12 Vyhledávání záznamů	34
4.2.1.13 Odeslání e-mailu	34
4.2.2 Seznam nefunkčních požadavků	34
4.2.2.1 Vzhled a uživatelské rozhraní	34
4.2.2.2 Rychlost	35
4.2.2.3 Technické prostředky na serveru	35
4.2.2.4 Jmenné konvence	35
4.3 Technické prostředky	35
4.3.1 Databáze	35
4.3.2 Serverová část	36
4.3.3 Klientská část	36
4.4 Návrh databáze	37
4.4.1 Evidované položky	37
4.4.1.1 Údaje o lidech	37
4.4.1.2 Údaje o organizacích	37
4.4.1.3 Údaje o sponzorských darech	38
4.4.1.4 Údaje o akcích	38

4.4.1.5 Údaje o typech akcí	38
4.4.1.6 Údaje o uživatelských aplikacích	38
4.4.1.7 Propojení relací	38
4.4.2 Návrh	40
4.4.2.1 Tabulka Lidé	41
4.4.2.2 Tabulka Organizace	41
4.4.2.3 Tabulka Dary	41
4.4.2.4 Tabulka Akce	41
4.4.2.5 Tabulka AkceTyp	41
4.4.2.6 Tabulka Interakce	41
4.4.2.7 Tabulka SysUsers	42
4.4.3 Splnění normálních forem	42
4.5 UI Specifikace	42
4.5.1 Motivace	42
4.5.2 Cílová skupina	42
4.5.3 Registrační stránka	43
4.5.3.1 Příklad užití	43
4.5.3.2 Scénář	43
4.5.3.3 Drátěný model	44
4.5.4 Přihlašovací stránka	44
4.5.4.1 Příklad užití	44
4.5.4.2 Scénář	44
4.5.4.3 Drátěný model	45
4.5.5 Horní lišta	45
4.5.5.1 Příklad užití	45
4.5.5.2 Scénář	45
4.5.5.3 Drátěný model	46
4.5.6 Dolní lišta	46
4.5.6.1 Příklad užití	46
4.5.6.2 Scénář	46
4.5.6.3 Drátěný model	46
4.5.7 Zobrazení záznamů	46
4.5.7.1 Příklad užití	46
4.5.7.2 Scénář	47
4.5.7.3 Drátěný model	48
4.5.8 Vyhledávací panel	48
4.5.8.1 Příklad užití	48
4.5.8.2 Scénář	48
4.5.8.3 Drátěný model	49

4.5.9 Panel pro přidání záznamu	50
4.5.9.1 Případ užití	50
4.5.9.2 Scénář	50
4.5.9.3 Drátěný model	50
4.5.10 Panel pro zobrazení detailu a editaci záznamu	51
4.5.10.1 Případ užití	51
4.5.10.2 Scénář	51
4.5.10.3 Drátěný model	52
4.5.11 Přidávání účastníků k akci	53
4.5.11.1 Případ užití	53
4.5.11.2 Scénář	54
4.5.11.3 Drátěný model	54
4.5.12 Vyskakovací hlášky a dialogy	55
4.5.12.1 Případ užití	55
4.5.12.2 Scénář	55
4.5.12.3 Drátěný model	55
4.6 Implementace	56
4.6.1 Struktura projektu	56
4.6.2 Konfigurační soubor na serveru	57
4.6.3 Renderování HTML kódu	57
4.6.4 Třídy na serveru a koncové body	61
4.6.4.1 Třídy	61
4.6.4.2 Koncové body (endpointy)	63
4.6.5 DOM funkce v JavaScriptu	64
4.6.6 Získávání dat z koncových bodů	65
4.7 Testování uživatelského rozhraní	67
5 Výsledky a diskuze	68
5.1 Výsledky testování uživatelského rozhraní	68
5.1.1 Problémy nalezené se čtyřmi účastníky	68
5.1.2 Problémy nalezené se třemi účastníky	68
5.1.3 Problémy nalezené se dvěma účastníky	68
5.1.4 Problémy nalezené s jedním účastníkem	69
5.2 Zhodnocení výsledků testování uživatelského rozhraní	69
5.3 Zhodnocení splnění funkčních požadavků	70
5.4 Zhodnocení volby technických prostředků	70
5.5 Návrhy na rozvoj do budoucna	70
6 Závěr	72
7 Seznam použitých zdrojů	73

Seznam obrázků

Obrázek 1 – Model databáze	40
Obrázek 2 – Registrační formulář (drátěný model)	44
Obrázek 3 – Přihlašovací formulář (drátěný model)	45
Obrázek 4 – Horní lišta (drátěný model)	46
Obrázek 5 – Dolní lišta (drátěný model)	46
Obrázek 6 – Zobrazení záznamů (drátěný model)	48
Obrázek 7 – Vyhledávací panel (drátěný model)	49
Obrázek 8 – Panel pro přidání záznamu (drátěný model)	50
Obrázek 9 – Panel pro zobrazení detailu a editaci záznamu (drátěný model)	52
Obrázek 10 – Panel Historie interakcí (drátěný model)	53
Obrázek 11 – Přidávání účastníků k akci (drátěný model)	54
Obrázek 12 – Vyskakovací hlášky a dialogy (drátěný model)	55

1 Úvod

V neziskových organizacích bývají mnohdy významným zdrojem příjmů příspěvky jejich členů či jiných sponzorů. Pokud organizace nevede dostatečnou evidenci těchto lidí nebo organizací, zpravidla přichází o mnoho potenciálních finančních prostředků. Evidování jednotlivých sponzorů bez aplikačního řešení lze považovat za velice náročnou činnost. Za předpokladu, že je charakter příspěvků takový, že jde spíše o velké množství v nižších částkách od mnoha drobných dárců, stává se takováto evidence téměř nemožnou.

V dnešní době existuje na trhu poměrně mnoho systémů pro tzv. řízení vztahů se zákazníky. Lze konstatovat, že nezanedbatelné množství z nich je pod takovým typem licence, že je možné je legálně používat zdarma. Dále také často jde o velice komplexní řešení, kdy se implementací jednoho velkého systému vyřeší hned několik problémů. I přes nesporné výhody těchto aplikací je nutné zmínit také jejich limity. Každá organizace je specifická a řeší své velmi konkrétní záležitosti. Může tedy snadno dojít k situaci, kdy je praktická použitelnost komplexního systému poměrně nízká, protože k danému problému přistupuje nevhodným způsobem. Přestože tyto aplikace často nabízí možnost další rozšiřitelnosti a přizpůsobitelnosti, pro neziskovou organizaci může být tato cesta finančně nákladná.

Pokud paleta problémů, které chce instituce řešit nasazením softwaru, čítá spíše menší množství položek, je namísto tvorba vlastního systému, na což je zaměřena tato bakalářská práce. Konkrétně půjde o webovou aplikaci pro Základní organizaci Českého svazu ochránců přírody Vlašim (dále v textu také jako ZO ČSOP). Očekávaným koncovým stavem je vyšší efektivita při shromažďování finančních zdrojů a zlepšení komunikace se členy i nečleny.

2 Cíle práce a metodika

2.1 Cíle práce

Práce se zabývá analýzou požadavků, návrhem a implementací webové aplikace s databází v neziskové organizaci, která bude sloužit zejména pro evidenci údajů o fyzických a právnických osobách, jež mají určitý vztah k této organizaci. Další důležitou funkcí bude správa historie interakcí těchto subjektů, jakými jsou například účasti na akcích, sponzorské dary či e-mailová korespondence.

2.2 Metodika

Zpracování teoretické části práce spočívá ve studiu odborných informačních zdrojů. Na základě nich pak budou popsána teoretická východiska pro zpracování praktické části práce.

Po analýze požadavků na funkcionality bude následovat návrh včetně rozvržení uživatelského rozhraní a struktury databáze, poté implementace. Pro návrh a analýzu bude využito standardních metod interakčního designu a softwarového inženýrství, implementace bude provedena v jazyce PHP s šablonovacím systémem Twig za využití MySQL databáze. Klientská část bude vytvořena v JavaScriptu ES6, dále v CSS3 s preprocesorem Less a HTML5. Hotová aplikace bude nasazena a otestována. Budou shrnuty poznatky získané během jejího vývoje a navrženy případné další možnosti jejího rozvoje do budoucna.

3 Teoretická východiska

3.1 Interakční design

Jedná se o poměrně nový obor výpočetní techniky, který se zabývá návrhem interaktivních produktů a služeb. Designér v podstatě navrhuje způsob, jakým s těmito produkty budou uživatelé komunikovat. Klíčové je zaměření na potřeby a omezení uživatelů v kontextu toho, co už znají; výsledek reflektuje jejich přesná očekávání a požadavky. [1] Uživatel je tedy při vytváření návrhu na prvním místě.

3.1.1 Pět dimenzí interakčního designu

Interakční design má pět základních principů, které jsou často nazývány pěti dimenzemi. [2]

První dimenzí jsou **slova**. Je důležité správně určit styl textu podle demografické skupiny. Slova by měla být výstižná a snadno srozumitelná. [2]

Vizuální reprezentace je druhou důležitou složkou. Zahrnuje všechno, co není slovo, tedy typografii, obrázky, ikony, diagramy a jakékoli další grafické prvky. Vizuály jsou první, co může upoutat uživatelskou pozornost. Pokud jsou tyto prvky správně zvolené, mají obdobnou vypovídající hodnotu, jako slova. Uživatelé by měli být schopni s nimi interagovat pouze pomocí své intuice. [2]

Neméně podstatná je také práce s **fyzickými objekty a prostorem** jako takovým. Vytváření dobrých vizuálů je důležité, ale podstatné je i jejich korektní umístění. Důležité je si uvědomit, že je třeba použít jiný design na mobilní zařízení a jiný na desktopy, neboť prostorové dispozice jsou rozdílné na každém rozlišení. [2]

Časová dimenze odkazuje na fakt, že uživatel stráví interakcí s produktem určitou dobu a také na to, že se některé typy obsahu mění v čase (animace, zvuk, video atd.). [2]

Pátým základním principem je **chování**. Zde je důležité počítat s tím, že každý uživatel se na stránce chová jinak a provádí jiné akce. Jinými slovy se jedná o souhrn toho, jak

předchozí čtyři elementy ovlivňují interakci uživatele s produktem. Klíčovou součástí je také zaměření na emocionální zpětnou vazbu a na nich postavená následná doporučení pro zlepšení uživatelské zkušenosti. [2]

3.1.2 Design uživatelské zkušenosti (UX)

UX je zkratka z anglického pojmu User eXperience a zahrnuje prožitky, emoce, přístup, postoje, styl používání a vývoj všech těchto aspektů. Jedná se tedy o vše, co uživatelé během používání produktu dělají, co cítí a prožívají, jaké to v nich budí nálady, co to v nich zanechává. UX design je komplexní činnost, která si klade za cíl navrhnout takový produkt, aby jeho používání bylo příjemné, budilo kladné emoce a aby si z něj uživatel odnesl pozitivní zážitek. Jedná se o nástroj, který by primárně měl co nejzajímavěji a nejpoutavěji dovést uživatele k cílům, které jsme si vytyčili předem, popřípadě které si stanovil zadavatel projektu. [3]

3.1.3 Design uživatelského rozhraní (UI)

Pojmem UI (User Interface) design rozumíme návrh samotného vzhledu softwarového produktu. Cílem je vytvořit očekávatelně se chovající rozhraní, které bude intuitivní a bude se uživatelům příjemně používat. [4] Uživatelské rozhraní by mělo vycházet z rámcových konceptů UX. Mezi tvůrci UI a UX je tedy velmi podstatná dobrá komunikace.

3.1.4 UI specifikace

Jedná se o strukturovaný dokument, který by měl obecně obsahovat motivaci, cíle a vzorové osoby, dále pak případy použití, ke kterým patří i požadavky na to, co systém zobrazí. Neměly by chybět drátěné modely logického designu. Dobrá UI specifikace a podle ní vytvořený prototyp může mnohdy předejít zbytečným chybám a finančním výdajům na následné opravy. [5]

3.1.4.1 Persona

Persona je fiktivní osoba, která je vymyšlena za účelem, aby reprezentovala reálné lidi, o kterých předpokládáme, že by mohli být uživateli. Jedná se tedy o zlidštěnou sumarizaci

charakteristik cílového publika. Tento model umožní návrhářům lépe porozumět uživatelům, jejich potřebám, zkušenostem, chování, očekávání a cílům, přičemž persony umožňují snazší simulaci toho, co budou dělat reální uživatelé během používání produktu. [6] Při konstruování fiktivního profilu se uvádí jméno, věk, gender, zájmy, typický den nebo krátká historie. Lze uvést i tzv. negativní personu, která představuje toho, pro koho produkt určený přímo není. Persony je možné nahradit také popisem cílové skupiny. [5]

3.1.4.2 Příklad užití

Jedná se o popis jednoho ze způsobů, jak může být systém použitý. Tento model popisuje interakci mezi uživatelem a systémem při dosahování konkrétního cíle. Je-li množina případů užití úplná, jedná se o kompletní specifikaci požadavků a očekávání uživatele. Případy užití se konstruují z důvodu, aby vývojový tým správně porozuměl potřebám uživatele. [7] Položek na seznamu očekávání uživatelů by mělo být z principu více, než přesných požadavků.

3.1.4.3 Scénář

Scénář je popis interakce mezi systémem a uživatelem který definuje, co systém zobrazí v reakci na akce uživatele. Může obsahovat více kroků, přičemž pro jednotlivé situace může existovat větší množství scénářů, které popisují kromě úspěšného provedení také další varianty, které mohou alternativně nastat. Jedná se tedy o informace, které jsou klíčové i pro následné testování. [7]

3.1.4.4 Drátěný model

V českém prostředí se také často označuje anglickým pojmem *wireframe*. Je vytvářen s cílem navrhnout rozložení prvků na stránce tak, aby uživatel co nejdříve dokončil akci, kterou chce provést. Zobrazuje rozložení stránky nebo uspořádání obsahu, včetně navigačních prvků. Tento model zpravidla postrádá stylizované písmo, barvy a grafické prvky, neboť hlavní důraz je kladen na účel, tedy na to, co obrazovka dělá, nikoli na vzhled. Drátěnými modely mohou být kresby tužkou nebo náčrty na tabuli; stejně tak je ale lze vytvářet v široké škále aplikací, z nichž mnoho je i bezplatných. [5]

3.1.5 Prototyp

Je-li implementován elektronickou formou, představuje v podstatě snímky obrazovky, mezi nimiž se lze pohybovat pomocí klikatelných navigačních prvků. Designér má tak možnost zjistit reálněji míru použitelnosti ještě předtím, než bude produkt vyvinut a vyhnout se tak rozsáhlým dodatečným úpravám ve chvíli, kdy bude již práce zdánlivě hotova. Rozlišují se dva typy prototypů: prototyp s nízkou mírou věrnosti označovaný spíše anglickým pojmem *low-fidelity* a prototyp s vysokou mírou věrnosti známý jako *high-fidelity*. [5]

3.1.5.1 Low-fidelity prototyp

Tento druh prototypu je abstraktnější a neobsahuje žádné graficky propracované prvky. Jeho vzhled připomíná drátěný model – výplňový text bez typografické stylizace, místo obrázků se umísťují prázdné obdélníky a tak dále. [5] Díky tomu se ale může tvůrce plně zaměřit na návrh a koncepty rozmístění prvků; nemusí se tak příliš zabývat technickou a grafickou stránkou věci. Zároveň ve chvíli, kdy dostane zpětnou vazbu od účastníka testu nebo zadavatele, je schopen v poměrně krátkém čase předělat kýženou část návrhu. Výhodou je i snadná přístupnost všem, protože díky jednoduchosti mohou na procesu návrhu participovat i lidé, kteří nejsou přímo designéři. [8]

3.1.5.2 High-fidelity prototyp

Prototyp s vysokou mírou věrnosti je velice podobný výslednému produktu a chová se téměř jako hotový software. Z toho vyplývá, že dochází k maximalizaci míry přirozeného chování účastníků testu. Když se označí konkrétní prvky, je možné získat hlubokou znalost vlastností těchto komponent (typicky například navigační ovládací prvky), což je u low-fidelity prototypu prakticky nemožné. Na druhou stranu, tato varianta prototypu je složitější nákladnější na přípravu, přičemž případné úpravy nejsou tak snadno a rychle proveditelné, jako u předchozího typu. [8]

3.1.6 Testování použitelnosti aplikace

Jde o proces, kdy se sledují lidé, kteří používají určitou aplikaci. Cílem je, aby se zjistilo, co je na námi vytvářeném produktu dobře či špatně navrženo a byly tak získány podklady pro další úpravy. Popřípadě je cílem prokázat, že se naše aplikace používá lehce. Testování použitelnosti se odlišuje od věcí jako jsou interview nebo průzkumy v tom, že nezjišťuje názory na minulou zkušenost s používáním produktů, nýbrž na něco, co je již navrženo. [9] Testování použitelnosti se rozlišuje na kvantitativní a kvalitativní.

3.1.6.1 Kvantitativní testování

Kvantitativní testování je prováděno na velkém množství uživatelů a výsledkem jsou statistická data. Je nutné definovat přesný protokol a striktně se jej držet v případě všech účastníků testu. Důležité je mít takovou skupinu, jež je reprezentativním vzorkem vzhledem ke všem uživatelům. Pokud je prováděno správným způsobem, autor aplikace získá například poměrně přesné údaje o tom, jak dlouho uživatelům trvá dokončit zadané úlohy či s jakým úkolem měli největší problémy. [9] Mnohdy je možné použít analytické nástroje, které se nasadí přímo na funkční web, čímž se výrazně sníží náklady.

3.1.6.2 Kvalitativní testování

Oproti tomu kvalitativní testování zahrnuje menší množství účastníků. Jedná se o vhodnou metodu pro pochopení, jaké konkrétní problémy mají uživatelé při plnění zadaných úkolů a proč k těmto problémům dochází. Je namístě uživatele také vyzvat k hlasitému přemýšlení, kdy komentuje, co dělá, proč to dělá a co očekává. Kvalitativní testování je možné provádět jak osobně, tak i vzdáleně. Mimo jiné můžeme vytvářet tzv. heatmapy, které vyjadřují, na které části obrazovky účastník testu upíral svou pozornost a kam klikal. Heatmapy lze získat buď pomocí monitorování činnosti myši, nebo sledováním pohybu očí. [10]

3.2 Webová aplikace

Webová aplikace je takový počítačový program, který běží na webovém serveru. Zásadním rozdílem oproti tradičním desktopovým aplikacím z uživatelského pohledu je, že nejsou spouštěny přímo operačním systémem, ale přistupuje se k nim za pomoci webového prohlížeče. [11]

Narozdíl od desktopových softwarových produktů se nemusí vývojáři webových aplikací zabývat kompatibilitou napříč různými operačními systémy. Při vydávání nových aktualizací také není zapotřebí řešit logistiku jejich distribuování, neboť k tomu, aby se k uživatelům dostala nová verze, stačí aktualizace na serveru. Z určitého pohledu může být výhodou také možnost vzdáleného ukládání dat. Výsledkem je, že tato data poté mohou být dostupná odkudkoli, tedy bez nutnosti je přesouvat mezi zařízeními. [11]

Tento typ aplikací se ale vyznačuje i určitými nevýhodami. Typická je nemožnost přistupovat k systémovým prostředkům, jako je procesor, paměť či souborový systém. To je důvodem, proč se někteří tvůrci softwaru drží desktopových verzí programů, zejména pak těch, které jsou výpočetně náročné. Často se jedná například o editory videa a další mediální software. [11] Mnohdy je zmiňovaná nevýhoda je také nutnost neustálého připojení k internetu. V dnešní době je ale v euroamerickém světě velice málo míst, kde by nebylo dostupné. Navíc existují určité aplikace, jako například Google Docs, které v omezeném režimu fungují i bez připojení k internetu.

Vzhledem k tomu, že různí uživatelé mají různé preference, mnoho společností (zejména větších) nabízí svoje softwarové produkty jak v desktopové, tak i ve webové verzi. Mezi takové se řadí například aplikace balíku Office od Microsoftu či některé produkty společností jako Apple nebo Adobe. [11] Dále také lze zmínit populární hudební aplikace typu Spotify a Tidal nebo komunikátory jako Skype či Discord.

Webové aplikace bývají postaveny na bázi modelu klient-server. Rozlišujeme tedy klientskou a serverovou část, přičemž tyto pojmy označují místa, kde běží kód. Servery mohou obsluhovat více uživatelů současně a klient může posílat více dotazů na různé servery. [12] Pro komunikaci mezi nimi se často používá tzv. API.

3.2.1 Klientská část

Tímto pojmem se rozumí všechno, co je zobrazeno nebo běží na straně klienta, tedy na zařízení koncového uživatele. Zahrnuje to, co uživatel vidí; tedy celé uživatelské rozhraní (text, obrázky, grafiku atd.) spolu se všemi akcemi, které aplikace provádí v rámci prohlížeče. [12]

Na straně klienta jsou prohlížečem interpretovány značkovací jazyky jako HTML a CSS. V současnosti existuje určitý trend odklonu od stavu, kdy je maximum procesů zpracovávaných na straně serveru směrem k tomu, že jich velká část běží u koncového uživatele. Pak bývají napsány téměř výhradně v JavaScriptu. [12] V praxi vývojář ale nemusí psát v čistém HTML, CSS nebo JavaScriptu, neboť existuje nespočet frameworků a knihoven, ve kterých lze vyvíjet a následně je kód automaticky přeložen. [13]

Mezi další charakteristiky kódu běžícího u klienta patří dynamické reakce na uživatelský vstup, dále také kompletní viditelnost a editovatelnost koncovým uživatelem a také nemožnost upravovat soubory na serveru či obsah databáze. [13]

3.2.2 Serverová část

Serverová část webové aplikace zahrnuje procesy, které se dějí na straně serveru a nikoli u koncového uživatele. V minulosti téměř veškerá aplikační logika běžela na serveru, což znamenalo velice časté stahování dat a také překreslování stránek. Bylo tedy nutné řešit problémy s výkonem serverového hardwaru a s tím spojený zhoršený uživatelský zážitek kvůli přílišné latenci. [12]

Možností jazyků, které lze použít pro vývoj serverové části aplikace je nepřeborné množství. Narozdíl od klientské části lze totiž použít jakýkoliv kód, který je možné spustit a který umí zpracovat HTTP požadavky. Mezi nejpobulárnější patří PHP, C#, Python (Django), JavaScript (Node.js) nebo Java. [13]

Serverová část se také stará o ukládání a načítání dat z databáze nebo také ze souborů. Charakteristické je, že si uživatel nemůže zobrazit kód běžící na serveru. Standardně také

reaguje na HTTP požadavky na předem definovaných URL, nikoli na jakýkoliv náhodně zvolený uživatelský vstup. [13]

3.2.3 Web API

API je zkratkou z anglického sousloví Application Programming Interface, což lze přeložit jako rozhraní pro programování aplikací. API jsou soubory funkcí a procedur, díky kterým lze vytvářet aplikace, které komunikují s jinými aplikacemi či službami. V minulosti byla velmi rozšířena procedurální SOAP (Simple Object Access Protocol) architektura, dnes je již ale mnohem běžnější datově orientovaný REST (Representational State Transfer). Zatímco SOAP používá XML formát, u RESTu to bývá nejčastěji JSON, díky kterému je po síti přenášeno menší množství dat. Formát JSON je oproti XML také výpočetně jednodušší na zpracovávání. [14]

Kromě veřejných API, které slouží pro komunikaci mezi na sobě nezávislými softwarovými produkty bývá API využíváno i pro propojení serverové a klientské části jedné webové aplikace. [14] Klient odešle požadavek (request) na tzv. API endpoint (koncový bod, reprezentován zpravidla pomocí URL). Server poté vrátí odpověď (response), která je v klientské části dále zpracována.

3.2.3.1 Metody pro přístup ke zdrojům

REST architektura rozlišuje čtyři základní metody (Retrieve, Create, Delete, Update), které jsou implementovány pomocí odpovídajících metod HTTP protokolu. [15]

První z nich je idempotentní metoda **GET (Retrieve)**, která slouží k získání zdroje. Jde v podstatě o klasický požadavek na stránku. Pokud je nutné předat serveru nějaké parametry, lze tak učinit pouze pomocí URL, neboť požadavek odeslaný přes GET nemá tělo. [15]

Další velice často využívanou metodou je **POST (Create)**. Za pomoci této metody je možné posílat parametry v těle požadavku, proto se mnohdy používá jeden společný identifikátor (endpoint) pro různé požadavky. Tato metoda je také známá z klasických HTML formulářů. [15]

DELETE je idempotentní metoda, která se obecně využívá ke smazání zdroje. Volání je podobné, jako u GETu; také neobsahuje tělo. V praxi ale bývá někdy problematické vyvolat HTTP metodu DELETE, neboť ji nepodporuje řada HTTP nástrojů. V takovém případě bývá nahrazena například metodou POST, přičemž se pošle navíc ještě parametr, který určí že má být ve skutečnosti použita metoda DELETE. [15]

Poslední ze čtyř základních metod je také idempotentní **PUT (Update)**, tedy operace změny. Je podobná POSTu, s tím rozdílem, že je volána konkrétní URI zdroje, který se mění. U metody PUT platí totéž, co u DELETE – existují nástroje, které ji nepodporují, a proto se i zde někdy používá náhradní metoda. [15]

3.2.3.2 Stavové kódy HTTP

Jakmile server zpracuje požadavek a pošle odpověď, měl by zároveň odeslat i správný stavový kód. Jde o číslo v rozmezí 100 až 599.

- **100 až 199** jsou informační odpovědi,
- **200 až 299** úspěšné vyřízení požadavku,
- **300 až 399** přesměrování (klient musí udělat ještě další doplňkovou akci),
- **400 až 499** chyby na straně uživatele,
- **500 až 599** jsou chyby na straně serveru. [16]

3.2.4 Jazyk HTML

HTML je zkratka z anglického HyperText Markup Language, česky hypertextový značkovací jazyk. Jde o kód sloužící k zobrazování dat předem definovaným způsobem. HTML je tedy přesné a stručné vyjádření toho, co se zobrazí ve webovém prohlížeči. Konečnou podobu nové verze HTML schvaluje WWW Consortium (W3C), aby se dosáhlo jednotnosti každé z vydaných verzí. [17]

3.2.4.1 Tagy, elementy a atributy

Základ HTML syntaxe jsou tzv. tagy. Pomocí těch je možné vytvářet elementy. Hlavní části elementu jsou tyto:

1. **Otevírací tag**, který obsahuje název elementu a je obalen špičatými závorkami. Tím se označí místo, kde element začíná.
2. **Uzavírací tag**; stejný jako otevírací, ale obsahuje navíc ještě lomítko před názvem elementu.
3. **Obsah** – to je cokoli, co je mezi otevíracím a uzavíracím tagem, tedy například text a/nebo další elementy. [18]

Elementy mohou mít tzv. atributy, které obsahují dodatečné informace. Atribut se zapisuje za název elementu do otevíracího tagu a obsahuje jméno a hodnotu. Měl by vždy mít následující:

- mezeru mezi ním a názvem elementu (popřípadě předchozím atributem, pokud jich element má více);
- znaménko rovná se mezi atributem a hodnotou;
- hodnotu obalenou z každé strany uvozovkami (angloamerický styl uvozovek) či apostrofy. [18]

Speciální typ elementu je prázdný element. To je takový, který nemá žádný obsah a žádný uzavírací element. [18]

3.2.4.2 Anatomie HTML dokumentu

Na prvním místě by měl být tzv. **doctype**. Uvádí se zejména z historických důvodů, v dnešní době spíše jen proto, aby bylo jisté, že se dokument chová správně. [18]

Další nutností je **html element** (`<html>`), který obaluje celý obsah stránky (kromě doctype). Někdy je také nazýván jako kořenový. Uvnitř něj by měla být hlavička a tělo stránky. [18]

Hlavička (element `<head>`) obaluje prvky, které budou součástí HTML stránky, ale nebudou přímo viditelné. Jedná se například o klíčová slova, popis stránky, CSS styly, definici znakové sady aj. [18]

Tělo stránky (`<body>`) je element, který by měl obsahovat všechny prvky, které budou zobrazeny na stránce, tedy například text nebo jakýkoli mediální obsah. [18]

3.2.5 Kaskádové styly (CSS)

CSS (z anglického Cascading Style Sheets) je deklarativní jazyk, pomocí kterého lze definovat, jak vypadá webová stránka v prohlížeči. Prohlížeč aplikuje deklarace stylů na vybrané elementy, přičemž taková **deklarace** se skládá z **vlastnosti** a její **hodnoty**. Množina deklarací je seskupena v bloku, který je poté přiřazen jednomu či více tzv. **selektorům**. Tento zápis se nazývá **CSS pravidlo**. Obvykle jsou pomocí CSS stylovány HTML elementy, ale lze je teoreticky použít i s dalšími značkovacími jazyky, jako je například SVG či XML. V následujícím příkladu je ukázáno použití s jazykem HTML. [19]

Zápis CSS pravidla vypadá takto:

```
s1, .s2, #s3 {  
    v1: h1;  
    v2: h2;  
}
```

V tomto příkladu jsou selektory `s1` a `s2`. Každý selektor vybírá určité elementy, nejčastěji to bývá podle názvu elementu (`s1`) atributu třídy (`.s2`) či atributu id (`#s3`). Zde tedy bude blok stylů uplatněn na všechny elementy `<s1>`, dále pak na ty, které mají `class="s2"` nebo `id="s3"`. Na počtu elementů, které vybere selektor nezáleží; validní CSS pravidlo je tedy i takové, které se neuplatní na žádné elementy. V tomto příkladu budou modifikovány vlastnosti `v1`, resp. `v2` tak, že se jim přiřadí hodnoty `h1`, resp. `h2`. [19]

Pro zjednodušení nebo zkrácení zápisu je možné využít některé z mnoha různých preprocesorů.

3.2.5.1 Preprocesor Less

Less (Leaner Style Sheets - volně lze přeložit z angličtiny jako štíhlejší styly) je zpětně kompatibilní rozšíření jazyka CSS. Díky této vlastnosti je tedy jakýkoli validní CSS zápis automaticky také validní Less kód. Vzhledem k tomu, že webové prohlížeče nativně neumí

pracovat se styly ve formátu Less, je nutné je nejdříve převést do CSS. Toho lze docílit například pomocí speciální JavaScriptové knihovny. [20]

Mezi hlavní přednosti zápisu stylů ve formátu Less oproti tradičnímu CSS patří zejména úspornější zápis, možnost vnořování (nesting), díky kterému zápis lépe kopíruje reálnou strukturu HTML kódu, dostupnost aritmetických operací či také velké množství zabudovaných funkcí. [20]

3.2.5.2 Framework Bootstrap

Bootstrap je sada šablon a nástrojů pro zjednodušení tvorby responzivních webových stránek či aplikací, díky které lze jednoduše stylovat komponenty rozhraní. Byl vytvořen v roce 2010 vývojáři ze společnosti Twitter; první veřejné vydání bylo 2011. [21]

Bootstrap obsahuje knihovny v CSS a JavaScriptu. Mnoho komponent JavaScript nevyžaduje, a proto v určitých případech lze použít jen CSS bez JavaScriptového modulu. [21]

3.2.6 JavaScript

JavaScript je interpretovaný programovací jazyk s funkcemi první třídy (lze s nimi pracovat jako s jakoukoli jinou proměnnou). Jde o jazyk jednovláknový a dynamický, který je používá vícero paradigmat – objektově orientované, imperativní a deklarativní. Přestože je znám především jako skriptovací jazyk pro web, využívá jej i mnoho neprohlížečových prostředí jako Node.js, Apache CouchDB apod. Standardem pro JavaScript je ECMAScript. [22]

Při vývoji moderních webových aplikací se mnohdy používají různé frameworky, díky kterým je možné vytvářet škálovatelné a interaktivní aplikace jednoduchým způsobem. [22] Takovými jsou například React, Angular nebo Vue.js.

JavaScript tvoří společně s HTML a CSS třetí základní pilíř standardních webových technologií. Umožňuje vytvářet dynamicky se měnící obsah, ovládat multimédia, vytvářet animované prvky a mnoho dalšího. [23]

3.2.7 PHP

Zkratka PHP je rekurzivní akronym z PHP: Hypertext Preprocessor (česky PHP: Hypertextový preprocesor). Jedná se o široce používaný open-source skriptovací jazyk, který může být jednoduše vložen do HTML stránky a který je určen zejména pro vývoj webu. [24]

Základní odlišností PHP například od klientského JavaScriptu je to, že je kód spouštěn na serveru a poté zobrazen výstup, který je odeslán klientovi. Klient tedy obdrží výsledky, aniž by věděl, jaký byl původní kód. [24]

Vzhledem k tomu, že PHP je primárně zaměřené na skriptování serverové části webových aplikací, je možné například shromažďovat data z formulářů, generovat dynamický obsah stránek nebo odesílat a přijímat cookies. Skripty v tomto jazyce ale mohou být volány i z příkazové řádky; to je vhodné například pro cron (Unix) nebo Plánovač úloh (Windows), pokud se jedná o pravidelné spouštění. Teoreticky lze PHP využít i pro psaní desktopových aplikací disponujících uživatelským rozhraním (například za použití rozšiřujícího balíku PHP-GTK). Toto využití však není příliš časté a spíše je považováno za nevhodné. [25]

PHP skripty mohou běžet pod všemi hlavními operačními systémy, dokonce jsou podporovány i některé exotičtější unixové varianty (OpenBSD, HP-UX, Solaris apod.). Podobně to platí i o různých webových serverech, velké množství z nich PHP podporuje. Kromě svobody, kterou PHP disponuje při volbě operačního systému a webového serveru také nabízí možnost výběru programovacího paradigmatu – lze použít imperativní, objektově orientované či libovolnou kombinaci obou. [25]

U PHP neexistuje omezení na formát výstupu. Kromě klasického HTML je možné generovat například XML, obrázky, PDF soubory či dokonce Flash animace. Významná funkcionální je také podpora široké škály databázových systémů. [25]

Objevují se však i názory, které PHP kritizují; zejména pak jeho starší verze. Stěžují si například na nekonzistentnost (příbuzné věci si nejsou podobné a je nutné znát mnoho

výjimek), nedostatečnou stručnost zápisu či složité ladění kódu, když nastane nějaký problém. [26]

3.2.7.1 Šablonovací systém Twig

Twig je flexibilní a rychlý šablonovací systém pro PHP od tvůrců frameworku Symfony. Jde tedy o knihovnu navrženou ke slučování datového modelu s obecnými šablonami¹, přičemž výsledkem je hotový dokument. Logika je taková, že jsou vytvořeny šablony, ve kterých jsou definovány statické a dynamické části. Vykreslovací funkce později nahradí dynamické části daty, která jsou výstupem PHP skriptu. Rychlost Twigu je oproti čistému PHP kódu srovnatelná. [27] [28]

3.2.8 Relační databáze

Relační databáze je založená na relačním modelu – jedná se o velice rozšířený způsob ukládání dat složený z relací neboli tabulek, které obsahují uspořádané n-tice, tedy řádky. Tato struktura záznamů má pevně stanovené položky (sloupce). Sloupec je jednoznačně definován názvem, rozsahem (velikostí) a typem dat. Relace (tabulky) pak obsahují konkrétní data. [29]

Oproti ukládání dat do obyčejných souborů mají databáze několik podstatných výhod. Mezi ně patří zejména rychlejší a přímý přístup k datům, systém uživatelských práv nebo možnost pomocí dotazů přímo extrahovat množiny dat podle zadaných kritérií. [30]

3.2.8.1 Klíče

Je vhodné, aby každý řádek tabulky měl tzv. primární klíč. Tato hodnota musí být unikátní v rámci celé tabulky. Většinou jde o číslo, které je inkrementováno s každým dalším záznamem přidaným do tabulky, ale není to pravidlo, lze použít i kombinaci více údajů, které dohromady jednoznačně určí příslušný záznam. [29] [30]

¹ Šablony mohou být obecně použity kupříkladu pro vytváření dynamických HTML stránek, pro předzpracování zdrojového kódu nebo ke generování e-mailů.

Kromě primárního klíče je důležitou věcí také klíč cizí (či nevlastní). Díky nim je možné definovat vztah mezi více tabulkami, protože sloupec (nebo sloupce) s cizími klíči mohou obsahovat primární klíče z jiné tabulky. [30]

3.2.8.1 Normalizace databáze

Normalizace databáze je speciální postup pro úpravu struktury relační databáze tak, aby se zvýšila efektivita zpracovávání, ukládání, třídění a prohledávání. Během normalizace jsou měněny atributy jednotlivých tabulek a snižuje se jak redundance dat, tak i pravděpodobnost výskytu nekonzistentních záznamů. [31]

Míra normalizace se uvádí v tzv. normálních formách (NF). Standardně je rozlišováno šest NF; jsou očíslovány od jedné do pěti a poslední nese název Boyce-Coddova normální forma (BCNF). Pro formy s číslem platí, že čím vyšší toto číslo je, tím vyšší stupeň normalizace. BCNF do posloupnosti nezapadá, ale lze říci, že sahá výše, než do třetího stupně (databáze musí být alespoň v 3NF, aby byla v BCNF, ale nemusí být ve 4NF). Obecně lze říci, že při normalizaci dochází často k dekompozici, tedy zvyšování počtu tabulek. [31]

Kromě těchto standardních forem existují i další. V roce 2003 byla například představena šestá normální forma (6NF). Tu splní databáze tehdy, když každý řádek tabulky obsahuje primární klíč a nejvýše jeden další atribut. Reálné využití tyto NF naleznou zejména v datových skladech. [32] Při návrhu databáze pro běžné aplikace se však většinou končí s 3NF (či nejvýše s BCNF) a další normalizace se již neprovádí. [31]

Databáze je v **první normální formě (1NF)** tehdy, když pro každý řádek platí, že žádný sloupec neobsahuje více než jednu hodnotu. Pak se takový sloupec označuje jako atomický. [31]

Druhá normální forma (2NF) je dosažena, pokud všechny atributy vyjma logického klíče (množina atributů, které jednoznačně identifikují řádek) závisí pouze na tomto logickém klíči. Pokud takový logický klíč neexistuje, musí být vytvořen. Zároveň musí platit, že je databáze v 1NF. [31]

Pro **třetí normální formu (3NF)** je zásadní, aby žádný atribut neměl tranzitivní závislost na klíči. To znamená, že každý atribut v tabulce musí buď být částí klíče, nebo záviset pouze na hodnotě celého klíče. Opět tedy platí, že 3NF zahrnuje 2NF. [31]

Boyce-Coddova normální forma (BCNF) řeší situaci, kdy existuje několik klíčů, jejichž složky se vzájemně překrývají. BCNF tedy vede k tomu, že jsou odstraněny nadbytečné klíče. [31]

3.2.8.2 Jazyk SQL

SQL (Structured Query Language, česky strukturovaný dotazovací jazyk) podporují téměř všechny systémy řízení báze dat (SŘBD) a používá pro komunikaci s relačními databázemi. Dotaz (query) je požadavek, který se odesílá databázi; poté databáze vrátí určitou odpověď. SQL je deklarativní jazyk, což v praxi znamená, že je pouze uvedeno, jaké výsledky jsou požadovány, přičemž není zapotřebí definovat způsob, jak tyto výsledky získat. [33]

Existují firmy, které vyvíjí procedurální rozšíření tohoto jazyka, mezi ně patří kupříkladu PL/SQL a Transact-SQL. Samotné SQL ale je podmnožinou těchto nástrojů a zůstává neprocedurální. [33]

Při vývoji aplikací implementovaných pomocí architektury klient-server běží SŘBD zpravidla na serveru, kde jsou také uloženy fyzické databázové soubory. [33]

3.2.8.3 MySQL

MySQL je vícevláknový multiplatformní systém řízení báze dat vytvořený společností MySQL AB, ale v současné době vlastněný firmou Oracle. Je poskytován jak v open-source verzi, tak jako komerční software. Mezi vývojáři je oblíbená kombinace MySQL s operačním systémem GNU/Linux, webovým serverem Apache a PHP; tato sada se označuje zkratkou LAMP. [34]

Databáze MySQL klade důraz na rychlost, proto její tvůrci přistoupili na určitá zjednodušení. Dotazy na výběr jsou včetně odpovědí ukládány do mezipaměti. Server také provádí analýzu dotazů za účelem vytvoření interní stromové struktury a následné aplikace optimalizací. [34]

4 Vlastní práce

Cílem této části bakalářské práce je navrhnout a implementovat webovou aplikaci na základě požadavků a v rámci teoretických východisek. Při návrhu software mnohdy jeho tvůrci přesně neznají cílovou skupinu a její exaktní očekávání, preference a požadavky. V takové situaci je tedy nutné, aby využívali některé heuristické metody.

Návrh a vývoj aplikace, jež je hlavním předmětem této práce, však bude probíhat v jistém smyslu odlišně. Vzhledem k tomu, že půjde o interní systém v organizaci, ještě před započítím práce jsou známí téměř všichni lidé, kteří s ním budou pracovat. Díky tomu lze postavit aplikaci tak, aby jim lépe vyhovovala.

4.1 Postup návrhu a vývoje

4.1.1 Návrhová část

Po analýze požadavků bude ve spolupráci se zaměstnanci z organizace vytvořen návrh databáze. Správně vytvořený datový model je klíčový jak pro rychlost softwaru, tak pro efektivitu při následném programování aplikačního rozhraní nad databází. Nelze opomenout také portabilitu, která je značně závislá na kvalitě tohoto modelu. Jestliže bude míra portability na dobré úrovni, bude možné v budoucnu jednoduše přenést data na novou platformu, pokud toto řešení již nebude vyhovující vzhledem k technickému pokroku.

Poté přijde na řadu UI specifikace, potažmo návrh uživatelského rozhraní. V počátečních fázích designu bude hojně využívána metoda dlouhých rozhovorů průběžně doplňovaná náčrtý drátěných modelů. Specifikace bude během své tvorby odesílána ke konzultacím v pravidelných časových intervalech. Díky tomuto postupu nebude zapotřebí vytvářet tradiční prototyp.

4.1.2 Implementační část

Po kompletním návrhu databáze a UI specifikace je další v pořadí implementace. Cílem je připravit dvě verze. Tato práce se zabývá pouze první z nich; v praxi lze tuto verzi označit také jako beta.

Podobně jako u UI specifikace bude také realizováno mnoho průběžných konzultací se zaměstnanci organizace i během implementace. Díky tomu dojde k upřesnění nejasných položek mezi funkčními a nefunkčními požadavky tak, aby byla výsledná uživatelská zkušenost co nejlepší.

4.1.3 Testovací část

Jakmile bude beta dokončena, aplikace se nasadí na lokální server, který je fyzicky umístěn v budově sídla organizace. Uživatelské rozhraní bude otestováno kvalitativní metodou. Půjde o moderované testování, přičemž každý z účastníků testu bude používat počítač, se kterým jinak běžně pracuje. Bude zadáno několik úkolů, které se uživatelé pokusí splnit. Výsledky budou získávány zejména observační metodou, ale také z odpovědí na doplňující otázky.

Verdikt ohledně splnění funkčních požadavků bude znám spíše po dlouhodobějším používání nasazené beta verze k praktickým úkonům; v horizontu několika týdnů.

Na závěr bude vytvořena produkční verze, která zohlední poznatky k získané z testování beta verze. Její vývoj ale není předmětem této práce.

4.2 Analýza požadavků

Z rozhovorů se zaměstnanci organizace byl sestaven seznam požadavků. Softwarové inženýrství tyto požadavky dělí na funkční, které popisují, co by měl softwarový produkt reálně umět a na nefunkční, které říkají, jaké by měly být jeho obecné vlastnosti.

4.2.1 Seznam funkčních požadavků

4.2.1.1 Registrace

Registrace by měla být dostupná pouze pro členy organizace. Musí tedy být nějakým způsobem zajištěno, aby si člověk mimo tuto organizaci nemohl vytvořit účet v systému.

4.2.1.2 Přihlášení

Všechny funkce systému (kromě registrace) by měly být uživatelům dostupné až po přihlášení, aby byla zajištěna bezpečnost dat.

4.2.1.3 Odhlášení

Uživatelé by měli mít možnost se odhlásit.

4.2.1.4 Evidence údajů o lidech

Mělo by být možné evidovat základní údaje o lidech, dále pak sponzorské dary a interakce, jako jsou účasti na akcích, na e-mailových konverzacích a nálezy zraněných zvířat.

4.2.1.5 Evidence údajů o organizacích

U organizací je také důležité mít možnost evidovat základní údaje, kontaktní osoby, interakce a sponzorské dary.

4.2.1.6 Evidence údajů o sponzorských darech

Kromě údajů jako jsou datum, částka nebo účel apod. je požadována možnost zadávat a uchovávat informace, kdo byl dárcem (člověk nebo organizace).

4.2.1.7 Evidence údajů o akcích

Akcí se v tomto širším smyslu rozumí kromě hromadné akce i nález zraněného zvířete, účast na e-mailové konverzaci či různé schůzky apod. Jinými slovy cokoliv, co může

generovat určitou interakci se ZO ČSOP a zároveň to není dar. Proto je kromě názvu a data nutné uchovávat také informace o typu a účastnících.

4.2.1.8 Evidence údajů o typech akcí

Typy akcí se musí dotahovat z určitého seznamu, aby se předešlo nepřehlednosti záznamů a zvýšila se efektivita vyhledávání a práce se systémem.

4.2.1.9 Zadávání údajů do evidence

Měly by být připraveny formuláře pro přidávání údajů do evidence pro každý typ záznamu.

4.2.1.10 Editace údajů v evidenci

Je nutné připravit formuláře pro editaci údajů v evidenci pro každý typ záznamu.

4.2.1.11 Smazání údajů v evidenci

Je důležité mít možnost smazat záznamy z evidence.

4.2.1.12 Vyhledávání záznamů

Návrh a implementace řešení pro vyhledávání a zobrazování různých záznamů podle zadaných parametrů je také nezbytnou funkcionalitou. Kromě zadávání těchto parametrů by měly být k dispozici i možnosti řazení dat ve výsledcích vyhledávání.

4.2.1.13 Odeslání e-mailu

Pokud je akce typu e-mail, mělo by být k dispozici tlačítko, které bude přesměrovávat do e-mailového klienta a automaticky vyplní seznam příjemců a předmět.

4.2.2 Seznam nefunkčních požadavků

4.2.2.1 Vzhled a uživatelské rozhraní

Aplikace by měla mít dobře navržené UI. Je důležité, aby zaměstnanci organizace evidovali všechny interakce s lidmi a organizacemi, jinak nebude využit plný potenciál

systemu. Z toho důvodu je klíčové mít kvalitně řešené uživatelské rozhraní, protože jen tehdy budou chtít lidé aplikaci používat pro ukládání opravdu všech kontaktů a interakcí.

V některých situacích (zejména bezprostředně po nasazení) budou s největší pravděpodobností někteří uživatelé v aplikaci trávit mnoho času. Je tedy nutné, aby ani delší práce se systémem neunavovala oči. Proto bude zvoleno světlé rozhraní namísto tmavého a budou použity spíše pastelové barvy.

4.2.2.2 Rychlost

Požadavky na rychlost nejsou určeny přesnými hodnotami, ale je požadováno, aby byla co nejnižší odezva během používání systému, a to i při práci s většími kolekcemi záznamů. Pokud je aplikace pomalá, výrazně to snižuje míru její použitelnosti.

4.2.2.3 Technické prostředky na serveru

Na serveru, kam bude aplikace umístěna, běží Apache HTTP Server. Je tedy zapotřebí používat pouze takové technické prostředky, aby serverová část webové aplikace fungovala pod tímto softwarem.

4.2.2.4 Jmenné konvence

Pro pojmenovávání funkcí, proměnných, tříd apod. by měl být využit anglický jazyk. Součásti databáze budou pojmenovány česky s výjimkou systémových tabulek (například uživatelé), kde bude použita opět angličtina.

4.3 Technické prostředky

4.3.1 Databáze

Vzhledem k tomu, že je vyžadována kompatibilita s Apache, jako systém řízení báze dat bude zvoleno MySQL (verze 8.0) založené na relačním databázovém modelu. Nepředpokládá se, že by struktura databáze byla příliš složitá a ani nebudou zapotřebí složitější databázové funkce. Na druhou stranu, je nutné brát v potaz nefunkční požadavek na rychlost i při práci s větším množstvím záznamů. Proto je MySQL vhodnou volbou.

4.3.2 Serverová část

Serverová část bude napsána v PHP (verze 7.4.3) zejména s využitím objektově orientovaného programování. Tento jazyk byl zvolen, protože je vhodný pro webovou aplikaci této velikosti, velmi dobře podporuje MySQL i Apache a navíc je dostupný zdarma. Další výhodou jazyka PHP je, že je oblíbený mezi vývojáři. Proto pokud bude v budoucnu například potřeba doprogramovat určitý plugin, bude případně snadné sehnat vývojáře, který jej vytvoří.

HTML kostra stránky bude vygenerována také pomocí PHP a šablonovacího systému Twig. Díky tomu bude HTML kód rozdělen do jednotlivých komponent složených z dynamických a statických částí. Podle parametru v URL se určí jak to, které komponenty se zobrazí, tak to, čím se vyplní dynamické části šablon. Toto řešení skýtá oproti čistému HTML mnoho výhod. Mezi hlavní patří omezení redundance kódu nebo velice snadné přidávání dalších stránek stejného typu (jen s jinými daty). HTML jako takové sice patří spíše do klientské části, ale vzhledem ke způsobu generování je řazeno do části serverové.

Pro přidání šablonovacího systému Twig, jakožto knihovny třetí strany, bude využit nástroj pro správu závislostí Composer.

4.3.3 Klientská část

Stylování bude realizováno v CSS s preprocesorem Less. HTML kód na vyrenderované stránce bude značně složitý, a proto je preprocessor vhodná volba; zejména díky možnosti používat zanořování (nesting). Využit bude také framework Bootstrap, aby nebylo nutné ručně stylovat tabulky, formulářové prvky apod.

Implementace dynamických prvků stránky a asynchronního dotahování dat ze serveru (AJAX) bude provedena v čistém JavaScriptu s využitím standardů ES6. Tato volba je na první pohled do jisté míry neobvyklá; v praxi by byl v dnešní době nejspíše zvolen nějaký framework, například široce rozšířený React. Hlavním důvodem tohoto rozhodnutí je důraz na rychlost i za cenu náročnějšího vývoje. Předpokládá se totiž, že se běžně bude stávat, že uživatel bude pracovat se stovkami záznamů na jedné stránce. V takovém případě není

možné, aby byla stránka přerenderována po každé drobné změně; na rychlost a potažmo použitelnost by to mělo fatální dopad.

4.4 Návrh databáze

Jak již bylo zmíněno, bude využit relační databázový model. Po konzultaci se zaměstnanci ZO ČSOP je třeba sestavit konečný seznam evidovaných položek a propojení jednotlivých relací s jinými. Poté se přistoupí k návrhu modelu databáze, o které by mělo platit, že splňuje alespoň třetí normální formu.

4.4.1 Evidované položky

4.4.1.1 Údaje o lidech

- Jméno
- Příjmení
- E-mail
- Pracovní e-mail
- Telefonní číslo
- Adresa
- Pracovní pozice
- Údaj o tom, zda si přeje být kontaktován (ano/ne)
- Datum poslední editace záznamu
- Poznámka

4.4.1.2 Údaje o organizacích

- Název
- Adresa
- Datum poslední editace záznamu
- Poznámka

4.4.1.3 Údaje o sponzorských darech

- Datum
- Částka
- Účel
- Údaj o tom, zda je dar hmotný (ano/ne)
- Datum poslední editace záznamu
- Poznámka

4.4.1.4 Údaje o akcích

- Název
- Datum
- Datum poslední editace záznamu
- Poznámka

4.4.1.5 Údaje o typech akcí

- Název
- Datum
- Datum poslední editace záznamu
- Poznámka

4.4.1.6 Údaje o uživateli aplikace

- Jméno uživatele
- Hash hesla

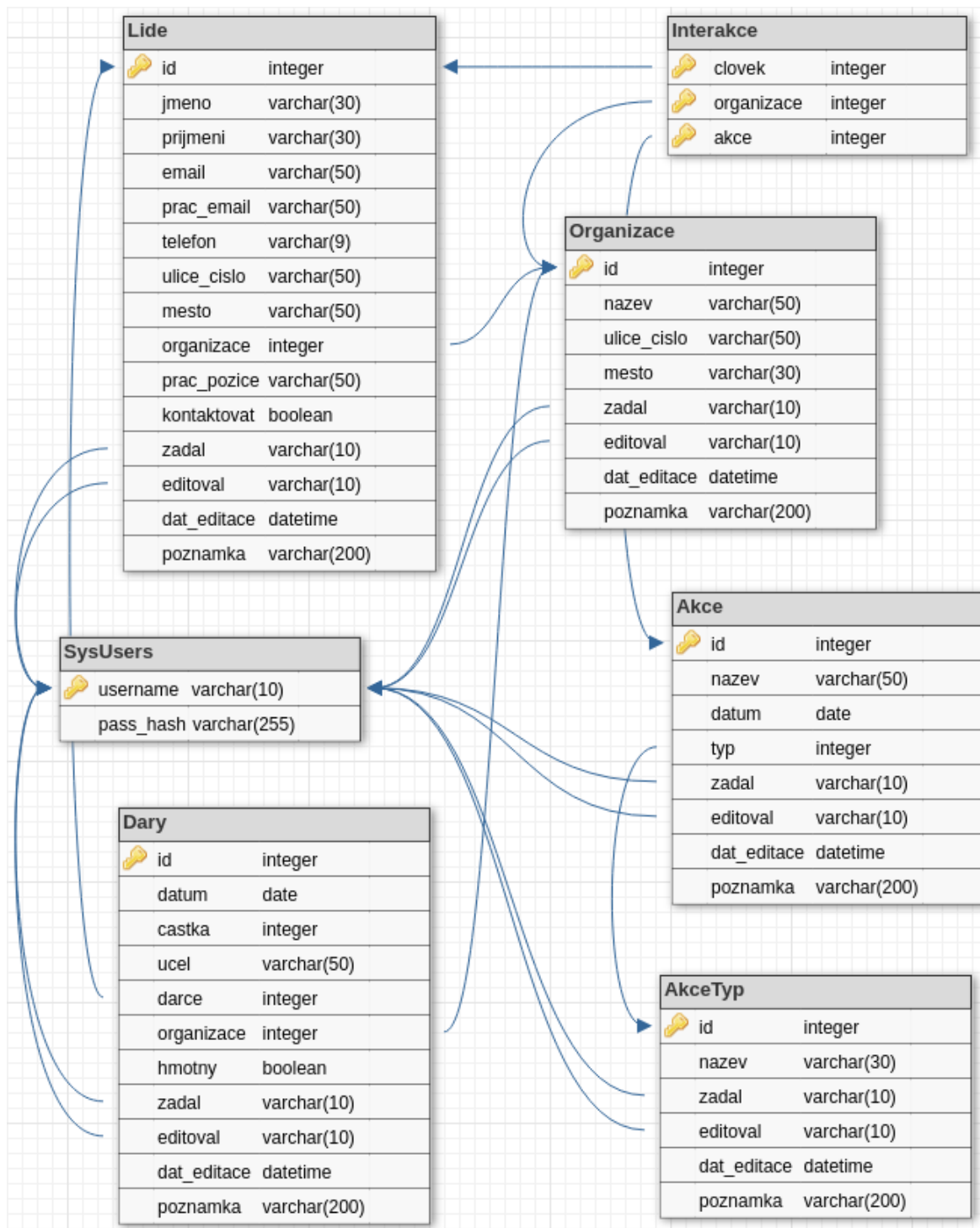
4.4.1.7 Propojení relací

- U všech relací kromě uživatelů a pomocných tabulek bude informace, jací uživatelé provedli vytvoření záznamu a poslední editaci
- Každý člověk může pracovat v nejvýše jedné organizaci a každá organizace má nula, jednoho, nebo více zaměstnanců

- Každý dar má právě jednoho dárce – člověka (ten může buď pro účely daru zastupovat organizaci, nebo dát dar sám za sebe) a každý člověk může dát nula, jeden, nebo více darů (stejně tak v rámci organizace)
- Každá akce má právě jeden typ a může existovat nula, jedna, nebo více akcí každého z typů
- Každá akce má nula, jednoho, nebo více účastníků a každý člověk se může účastnit nula, jedné, nebo více akcí

4.4.2 Návrh

Obrázek 1 – Model databáze



Zdroj: Vlastní dílo

Databáze bude obsahovat celkem sedm tabulek, jak je vidět na obrázku 1. Struktura šesti z nich (Lide, Organizace, Dary, Akce, AkceTyp, SysUsers) odpovídá skupinám položek popsaných výše doplněným o sloupce s primárními a cizími klíči, pokud je to třeba. Sedmá tabulka (Interakce) je pomocná. V každé z tabulek Lide, Organizace, Dary, Akce a AkceTyp vzniknou sloupce zadal a editoval obsahující cizí klíče z tabulky SysUsers. Další rozdíly oproti již popsané struktuře jsou uvedeny níže.

4.4.2.1 Tabulka Lidé

Byl přidán primární klíč (id), a organizace, tedy cizí klíč z tabulky Organizace.

4.4.2.2 Tabulka Organizace

Byl přidán pouze primární klíč (id).

4.4.2.3 Tabulka Dary

Byl přidán primární klíč (id), dále pak sloupec darce, což je cizí klíč z tabulky Lide a sloupec organizace, tedy cizí klíč z tabulky Organizace. Ten je vyplněn pouze tehdy, když dárcem vystupuje jménem nějaké organizace.

4.4.2.4 Tabulka Akce

Byl přidán primární klíč (id) a typ – cizí klíč z tabulky AkceTyp.

4.4.2.5 Tabulka AkceTyp

Byl přidán pouze primární klíč (id).

4.4.2.6 Tabulka Interakce

Tato pomocná tabulka vznikla dekompozicí a slouží pro evidenci účastníků na akcích. V této tabulce jsou tři sloupce: clovek (cizí klíč z tabulky Lide), organizace (cizí klíč z tabulky Organizace) a akce (cizí klíč z tabulky Akce). Všechny tyto atributy jsou součástí primárního klíče.

4.4.2.7 Tabulka SysUsers

Tabulka oproti sloupcům popsaným výše neobsahuje navíc žádné další. Primárním klíčem je zde username.

4.4.3 Splnění normálních forem

Tento návrh má ve všech sloupcích atomické hodnoty (se sloupci ulice_cislo se vždy pracuje tak, jako by byly atomické). Je tedy splněna 1NF. Platí také, že všechny atributy vyjma logického klíče závisí pouze na tomto logickém klíči, splněna je tedy i 2NF. Vzhledem k tomu, že žádný atribut nemá tranzitivní závislost na klíči, je možné prohlásit, že je databáze i ve 3NF, což byla původní podmínka.

4.5 UI Specifikace

4.5.1 Motivace

V současné době je evidováno v ZO ČSOP mnoho kontaktů na lidi a organizace ručně; včetně darů a interakcí. Problémem je také to, že každý ze zaměstnanců má svou vlastní evidenci a svůj vlastní systém. V záznamech každého z těchto lidí se také nezdá, že by se objevují jisté nekonzistence.

Z toho důvodu je třeba přinést nový způsob práce s evidovanými údaji, který budou všichni zúčastnění považovat za dostatečně uspokojivý. Kromě zřejmého faktu, že je nutné navrhnout vhodný datový model, je pro splnění této podmínky také velice důležité připravit kvalitní uživatelské rozhraní.

4.5.2 Cílová skupina

Budoucí uživatelé této webové aplikace jsou tedy někteří kancelářští zaměstnanci ZO ČSOP Vlašim. To jsou lidé se středoškolským či vysokoškolským vzděláním, kteří žijí na malém městě; popřípadě na vesnici poblíž. Denně ke své práci používají počítače, takže jsou běžně uživatelsky zdatní.

4.5.3 Registrační stránka

4.5.3.1 Příklad užití

Uživatel očekává

- možnost zadat uživatelské jméno;
- možnost zadat heslo včetně pole pro kontrolu;
- možnost odeslat formulář;
- odkaz na přihlašovací stránku, pokud už účet má.

Uživatel požaduje

- možnost vytvořit účet dostupnou pouze pro zaměstnance organizace.

4.5.3.2 Scénář

System zobrazí

- pole pro uživatelské jméno;
- pole pro heslo;
- pole pro kontrolu hesla;
- pole pro zvací kód, který bude známý jen pro lidi z organizace;
- tlačítko Registrovat se (pokud jsou údaje v pořádku, po kliknutí je uživatel přesměrován do systému);
- odkaz na přihlašovací stránku.

4.5.3.3 Drátěný model

Obrázek 2 – Registrační formulář (drátěný model)

ČSOP Vlašim
Registrace

Uživatelské jméno

Heslo

Heslo pro kontrolu

Zvací kód

Registrovat se

Již máte účet? [Přihlaste se.](#)

Zdroj: Vlastní dílo

4.5.4 Přihlašovací stránka

4.5.4.1 Příklad užití

Uživatel očekává

- možnost zadat své uživatelské jméno;
- možnost zadat své heslo;
- možnost odeslat formulář;
- odkaz na registrační stránku, pokud ještě účet nemá.

4.5.4.2 Scénář

System zobrazí

- pole pro uživatelské jméno;

- pole pro heslo;
- tlačítko Přihlásit se (pokud jsou údaje v pořádku, po kliknutí je uživatel přesměrován do systému);
- odkaz na registrační stránku.

4.5.4.3 Drátěný model

Obrázek 3 – Přihlašovací formulář (drátěný model)

Zaregistrujte se.'"/>

ČSOP Vlašim
Přihlášení

Uživatelské jméno

Heslo

Přihlásit se

Ještě nemáte účet? [Zaregistrujte se.](#)

Zdroj: Vlastní dílo

4.5.5 Horní lišta

4.5.5.1 Případ užití

Uživatel očekává

- menu, pomocí kterého bude možné přecházet mezi stránkami.

4.5.5.2 Scénář

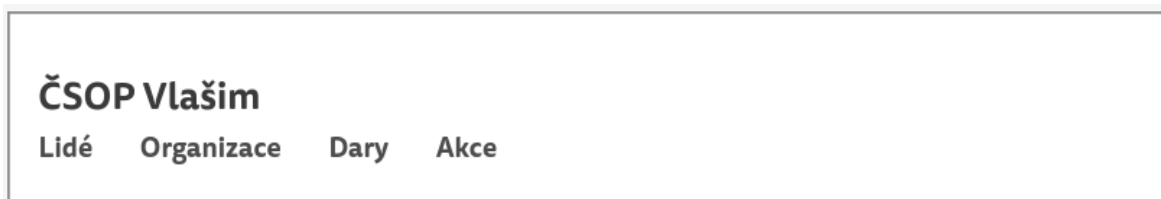
System zobrazí

- horní lištu a navigační menu s ikonami.

Odkaz na typy akcí (tabulka AkceTyp) systém zobrazí pouze při přidávání nebo úpravě akce.

4.5.5.3 Drátěný model

Obrázek 4 – Horní lišta (drátěný model)



Zdroj: Vlastní dílo

4.5.6 Dolní lišta

4.5.6.1 Příklad užití

Uživatel očekává

- zobrazení svého uživatelského jména (aktuální přihlášený uživatel);
- možnost se odhlásit.

4.5.6.2 Scénář

System zobrazí

- dolní lištu se jménem přihlášeného uživatele a klikatelným odkazem na odhlášení.

4.5.6.3 Drátěný model

Obrázek 5 – Dolní lišta (drátěný model)



Zdroj: Vlastní dílo

4.5.7 Zobrazení záznamů

4.5.7.1 Příklad užití

Uživatel očekává

- přehledné zobrazení záznamů na stránce;
- možnost vyhledávat mezi uloženými záznamy;
- možnost přidat záznam;

- možnost upravit záznam;
- možnost smazat záznam;
- možnost vybrat lidi ze záznamů k další případné práci s nimi.

4.5.7.2 Scénář

System zobrazí

- název stránky (tabulky);
- tlačítka na přidání záznamu a vyhledávání;
- tabulku se záznamy;
- frontu, tedy seznam vybraných lidí v horní části tabulky (pokud je v ní více než nula lidí); dále také křížek na odebrání jednotlivce z fronty a tlačítka na odebrání všech
- tlačítka Všichni do fronty (text se liší podle tabulky) pro přidání všech lidí ze záznamů do fronty v horní části tabulky (pokud záznamy obsahují nějaké lidi);
- tlačítka Poslat (pouze, pokud jde o akci, která je zároveň typu E-mail)
- u každého záznamu tlačítka Fronta (text se liší podle tabulky) pro přidání do fronty;
- u každého záznamu tlačítka Detail pro zobrazení detailu a možnost editace;
- u každého záznamu tlačítka Odstranit pro odstranění záznamu;

4.5.7.3 Drátěný model

Obrázek 6 – Zobrazení záznamů (drátěný model)

Fronta

Tabulka

Sloupec 1	Sloupec 2	Sloupec 3	Sloupec 4	Sloupec 5	Sloupec 6	<input type="button" value="Všichni do fronty"/>		
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>
Údaj 1	Údaj 2	Údaj 3	Údaj 4	Údaj 5	Údaj 6	<input type="button" value="Fronta"/>	<input type="button" value="Detail"/>	<input type="button" value="Odstranit"/>

Zdroj: Vlastní dílo

4.5.8 Vyhledávací panel

4.5.8.1 Příklad užití

Uživatel očekává

- možnost zadání vyhledávacích parametrů;
- možnost řazení.

4.5.8.2 Scénář

Po kliknutí na tlačítko Vyhledávání se zobrazí na pravé straně vysouvací panel. Systém v tomto panelu zobrazí

- název panelu, tedy nadpis Vyhledávání;
- tlačítko plus (+) pro přidání parametru (polí pro zadání vyhledávaného dotazu); u každého řádku s přidávanými poli systém zobrazí tlačítko mínus (-) pro odebrání tohoto parametru;

- výběrací nabídku se sloupci, podle vybraného se bude řadit;
- výběrací nabídku, jestli řadit vzestupně nebo sestupně;
- seznam sloupců se zaškrťovacími políčky (které sloupce zobrazit);
- tlačítko Vyhledat, které zavře panel a zobrazí výsledky vyhledávání.

4.5.8.3 Drátěný model

Obrázek 7 – Vyhledávací panel (drátěný model)

Vyhledávání X

Parametry

Jméno	▼	je přesně	▼		-
E-mail	▼	obsahuje	▼		-

+

Řadit podle

Příjmení	▼	vzestupně	▼
----------	---	-----------	---

Zobrazit sloupce

<input checked="" type="checkbox"/> Jméno	<input checked="" type="checkbox"/> Příjmení
<input checked="" type="checkbox"/> E-mail	<input type="checkbox"/> Pracovní e-mail
<input type="checkbox"/> Ulice a č. p.	<input checked="" type="checkbox"/> Město

Vyhledat

Zdroj: Vlastní dílo

4.5.9 Panel pro přidání záznamu

4.5.9.1 Příklad užití

Uživatel očekává

- zobrazení formuláře pro přidání záznamu.

4.5.9.2 Scénář

Po kliknutí na tlačítko Přidat se zobrazí na pravé straně vysouvací panel. Systém v tomto panelu zobrazí

- název panelu, tedy nadpis Přidat;
- pole pro všechny editovatelné atributy;
- tlačítka Uložit (po kliknutí uloží záznam a zavře panel) a Zavřít panel.

4.5.9.3 Drátěný model

Obrázek 8 – Panel pro přidání záznamu (drátěný model)



The diagram shows a rectangular panel with a light gray border. At the top left, the word "Přidat" is written in a bold, sans-serif font. At the top right, there is a small "X" icon. The main area of the panel is divided into two columns and four rows. Each of the eight cells in this grid contains the word "Atribut" above a rectangular input field. At the bottom of the panel, there are two rounded rectangular buttons. The left button is labeled "Uložit" and the right button is labeled "Zavřít panel".

Zdroj: Vlastní dílo

4.5.10 Panel pro zobrazení detailu a editaci záznamu

4.5.10.1 Příklad užití

Uživatel očekává

- zobrazení formuláře pro editaci záznamu;
- zobrazení informací o vytvoření záznamu a poslední editaci;
- zobrazení historie interakcí (pokud jde o organizaci nebo člověka).

4.5.10.2 Scénář

Po kliknutí na tlačítko Detail se zobrazí na pravé straně vysouvací panel. Systém v tomto panelu zobrazí

- název panelu, tedy nadpis Detail;
- pole pro všechny editovatelné atributy s hodnotami, které jsou nyní vyplněny;
- tabulku, která obsahuje informace, kdo záznam zadal a editoval, dále datum editace;
- tlačítko, které otevře historii interakcí (vysouvací panel s tabulkami se záznamy o darech a účastech na akcích), ale pouze v případě, že jde o detail organizace nebo člověka;
- tlačítka Uložit (po kliknutí uloží záznam a zavře panel), Zavřít panel a Odstranit (po kliknutí smaže záznam a zavře panel).

4.5.10.3 Drátěný model

Obrázek 9 – Panel pro zobrazení detailu a editaci záznamu (drátěný model)

Detail X

Atribut	Atribut
<input type="text" value="Hodnota"/>	<input type="text" value="Hodnota"/>
Atribut	Atribut
<input type="text" value="Hodnota"/>	<input type="text"/>
Atribut	Atribut
<input type="text"/>	<input type="text" value="Hodnota"/>
Atribut	Atribut
<input type="text"/>	<input type="text"/>

Zadal	Uživatel
Editoval	Uživatel
Datum editace	1. 1. 1970

[Historie interakcí](#)

Zdroj: Vlastní dílo

Obrázek 10 – Panel Historie interakcí (drátěný model)

Historie interakcí (Jméno Příjmení) X

Dary

Datum	Částka	Účel	Organizace
1. 1. 1970	1000	Péče o zeleň	
2. 1. 1970	2000		MěstoVlašim

Účasti na akcích

Datum	Název	Typ	Organizace
1. 1. 1970	Akce 1	Přednáška	
2. 1. 1970	Informace o akcích	E-mail	

Zdroj: Vlastní dílo

4.5.11 Přidávání účastníků k akci

4.5.11.1 Případ užití

Uživatel očekává

- možnost přidat k akci nula, jednoho, nebo více účastníků;
- přidat účastníky uložené ve frontě.

4.5.11.2 Scénář

Systém zobrazí pole s účastníky. Po kliknutí na toto pole se zobrazí na pravé straně vysouvací panel s těmito komponenty:

- seznam již přidanych lidí;
- tlačítko Přidat, které zobrazí textové pole, kam lze vyplnit jméno člověka (možné i včetně organizace);
- tlačítko Vložit z fronty, které přidá všechny záznamy, které jsou momentálně ve frontě;
- tlačítko Odstranit všechny;
- tlačítko Uložit (uloží účastníky a vrátí uživatele k editaci akce), tlačítko Zavřít panel (vrátí uživatele k editaci akce).

4.5.11.3 Drátěný model

Obrázek 11 – Přidávání účastníků k akci (drátěný model)

The image shows a modal window titled "Účastníci" with a close button "X" in the top right corner. Inside the modal, there are three buttons: "Přidat", "Vložit z fronty", and "Odstranit všechny". Below these buttons are four text input fields for names, each with a blue "X" icon for clearing the field. The fields are labeled "Jméno Příjmení", "Jméno Příjmení (Firma)", "Jméno Příjmení", and "Jméno Příjmení (Firma)". At the bottom of the modal are two buttons: "Uložit" and "Zavřít panel".

Zdroj: Vlastní dílo

4.5.12 Vyskakovací hlášky a dialogy

4.5.12.1 Příklad užití

Uživatel očekává

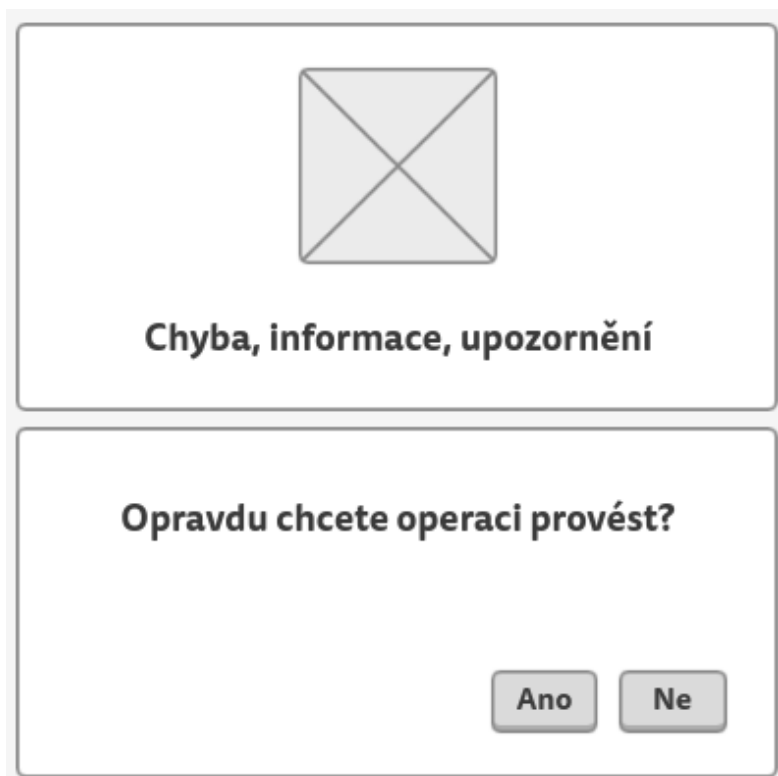
- informační hlášky v případě úspěchu;
- chybové hlášky;
- upozornění;
- dotazovací dialogy v případě, že systém bude potřebovat další informace;
- potvrzovací hlášky, zda skutečně určitou operaci provést.

4.5.12.2 Scénář

System zobrazí příslušné vyskakovací hlášky a dialogy.

4.5.12.3 Drátěný model

Obrázek 12 – Vyskakovací hlášky a dialogy (drátěný model)



Zdroj: Vlastní dílo

4.6 Implementace

Tato část práce si klade za cíl pouze popsat koncepty, jak aplikace funguje a jak je navržena; kompletní zdrojové kódy beta verze jsou přiloženy k této práci na kompaktním disku. Vzhledem k tomu, že se jedná o kopii verze v současné době nasazené na produkci, nejsou obsaženy soubory verzovacího systému GIT ani soubory Composeru (composer.json a composer.lock).

V ukázkách kódu v této části práce bude mnohdy něco vynecháno. Taková místa budou označena třemi tečkami. Pro zkrácení bude také odstraněna většina komentářů.

4.6.1 Struktura projektu

V kořenovém adresáři aplikace se tedy nachází pět složek a tři soubory.

První z nich, **app**, obsahuje PHP skripty, tedy soubory s třídami, které jsou nepostradatelné pro správný chod projektu. Dále jsou v ní umístěny soubory, jež slouží jako samotné koncové body (endpointy) při volání z klientské části.

Další složka se jmenuje **css**. Uvnitř ní je soubor se styly ve formátu Less a také knihovny třetích stran (Bootstrap, Font Awesome).

V adresáři **js** je soubor s klientským JavaScriptem zajišťujícím dynamiku stránky a asynchronní získávání dat ze serveru. Je zde umístěna knihovna SweetAlert, která se stará o vyskakovací dialogy a hlášky a také minifikovaný parser less.js pro zpracování stylůpisů.

Složka **templates** obsahuje Twig soubory představující komponenty (šablony), ze kterých se poskládá výsledný render stránky.

Adresář **vendor** je vytvořen za použití Composeru a obsahuje knihovny třetích stran. V případě této aplikace se jedná pouze o již zmíněný šablonovací systém.

V kořenové složce se nachází ještě soubory `index.php` (spouští render všechny stránek kromě chybových); dále `error.php` (spouští render chybových stránek) a `.htaccess` (směrování požadavků a konfigurace serveru).

4.6.2 Konfigurační soubor na serveru

Konfigurační soubor se nachází v adresáři `inc` (pod `app`) a slouží k nastavování základních parametrů, jako jsou přístupy do databáze, zvací kód do registrace či rychlost algoritmu `bcrypt`.

```
/* ... */

// DB credentials
define("DB", array(
    "server" => "",
    "db_name" => "",
    "user" => "",
    "password" => ""
));

// Home URL
define("HOME", "");

// Registration code
define("REG_CODE", "");

// Developer mode
define("DEVELOPER", false);

// Password hash hardware cost
define("BCRYPT_COST", 12);

/* ... */
```

4.6.3 Renderování HTML kódu

Vykreslovací funkce pro všechny stránky kromě chybových jsou spouštěny ze souboru `index.php`. Ve složce se šablonami (templates) existuje pouze jedna sada šablon s dynamicky doplňovanými částmi. Podle parametru v URL a podle toho, zda je uživatel přihlášen, se vytvoří instance příslušné třídy s daty, která se následně propíše do šablony. To je vidět na následující ukázce části kódu zmíněného souboru.

```

/* ... */
$current_page = $_GET["page"] ?? "";
$redirect = false;

if (Login::checkLogin()) {
    switch($current_page) {
        case "lide":
            $page = new Lide();
            break;
        case "organizace":
            $page = new Organizace();
            break;
        case "dary":
            $page = new Dary();
            break;
        case "akce":
            $page = new Akce();
            break;
        case "typy-akci":
            $page = new AkceTyp();
            break;
        case "":
        case "registrace":
        case "prihlaseni":
            $redirect = "lide"; break;
        default:
            AppError::throw("Page not found", 404);
    }
    if (!$redirect) {
        echo $twig->render("main.twig", $page->getData());
        die();
    }
} else {
    /* ... */
}
header("Location: " . HOME . "?page=" . $redirect);

```

Metody tříd uvedené v ukázce tedy pouze vracejí určitou kolekci dat (vícezměrné pole). Největší výhodou tohoto řešení je, že pokud bude zapotřebí kupříkladu přidat pole do určitého formuláře, stačí specifikaci tohoto prvku přidat do pole dat v příslušné metodě. Obdobným způsobem lze v krátkém čase přidávat i celé stránky. Stačí jen vytvořit třídu a její instanci. Na následujícím příkladu je takováto třída ukázána.

```

/* ... */
class Lide extends Page {

    private function getColumns() {
        $p = $this->getPatterns();
        return array(
            array(
                "value" => "jmeno", "name" => "Jméno",
                "checkbox" => true, "checked" => true,
                "search" => true, "sort" => true,
                "editable" => true, "type" => "text",
                "required" => true, "pattern" => $p["str"],
                "maxlength" => 30, "flex" => 1,
                "autocomp" => false
            ),
            array(
                "value" => "prijmeni", "name" => "Příjmení",
                "checkbox" => true, "checked" => true,
                "search" => true, "sort" => true,
                "editable" => true, "type" => "text",
                "required" => true, "pattern" => $p["str"],
                "maxlength" => 30, "flex" => 1,
                "autocomp" => false
            ),
        );
        /* ... */
    }

    /* ... */

    public function getData() {
        return array(
            "columns" => $this->getColumns(),
            "menu" => $this->getMenu(),
            "pagename" => "Lidé",
            /* ... */
        );
    }
}

```

Na dalším příkladu se nachází kód šablony, která generuje pole na formulářích pro vytvoření nebo editaci záznamu.

```
{% for column in columns %}
  {# ... #}
  <label for="{{column.value}}_{{page}}">
    {# ... #}
  </label>
  {% if column.type == "boolean" %}
    <select
      class="form-control"
      name="{{column.value}}"
      id="{{column.value}}_{{page}}"
    >
      <option value="1">ano</option>
      <option value="0">ne</option>
    </select>
  {% else %}
    <input
      class="form-control"
      name="{{column.value}}"
      id="{{column.value}}_{{page}}"
      {% if column.type == "queue" %}
        type="text"
        {% if page == "create" %}
          onfocus="editParticipants('create')"
        {% else %}
          onfocus="editParticipants('edit')"
        {% endif %}
      {% else %}
        type="{{column.type}}"
      {% endif %}
      {% if column.type != "email" %}
        pattern="{{column.pattern}}"
      {% endif %}
      maxlength="{{column.maxlength}}"
      {{column.required ? "required" : ""}}
    >
    {% if column.autocomp %}
      {# ... #}
    {% endif %}
  {% endif %}
{% endfor %}
```

4.6.4 Třídy na serveru a koncové body

4.6.4.1 Třídy

Třídy, jejichž metody vrací kolekce dat určených pro doplnění do šablon byly již popsány výše. Kromě nich ale v aplikaci existuje dalších šest tříd.

O chyby se stará třída **AppError**. Pokud aplikace běží ve vývojářském módu (nastavuje se v konfiguračním souboru), chyby se vypisují s přesným popisem. V opačném případě se vyrenderuje stránka s kódem chyby (pomocí `error.php` a příslušné Twig šablony).

Třída **Connection** umožňuje vytvořit nové připojení do databáze (resource) a pak jej uzavřít. Její konstruktor je na následující ukázce.

```
public function __construct() {
    $this->conn = new mysqli(
        DB["server"],
        DB["user"],
        DB["password"],
        DB["db_name"]
    );
    if ($this->conn->connect_error) AppError::throw(
        "Unable to reach database; {$this->conn->connect_error}"
    );
    $this->conn->set_charset("utf8");
}
```

Database je třída, která implementuje vykonávání databázových dotazů. Obsahuje metody pro výběr z tabulky (včetně případného řazení), editaci záznamu, vytvoření záznamu a odstranění záznamu a dalších několik pomocných metod. Je kladen velký důraz na zamezení SQL injection i u složitějších dotazů. Následující kód ukazuje metodu na odstranění záznamu (`$table` je vždy validní jméno tabulky; ošetřeno dříve v kódu).

```

private function deleteRecord($table, $id) {
    if (!$id) AppError::throw("SQL query error - corrupted \"id\" param");
    $query_str = "DELETE FROM {$table} WHERE id = ?";
    $query = $this->conn->prepare($query_str);
    $query->bind_param("i", $id);
    $query->execute();
    if ($query->error) AppError::throw("SQL query error: {$query->error}");
    /* ... */
}

```

Vykonávání pokročilých databázových dotazů implementuje třída **SpecialQueries**. Jedná se zejména o takové, které jsou nad více tabulkami. Na ukázce je část metody pro získání historie interakcí určitého člověka.

```

public function getPersonHistory($id) {
    if (!$id || !is_int($id)) AppError::throw("Corrupted id param");
    $query = $this->conn->prepare(
        "SELECT
            Akce.*,
            AkceTyp.nazev AS typ_akce,
            Interakce.organizace,
            Organizace.nazev AS 'organizace_jmeno'
        FROM Interakce
        LEFT JOIN Organizace ON Organizace.id = Interakce.organizace
        LEFT JOIN Akce ON Akce.id = Interakce.akce
        JOIN AkceTyp ON AkceTyp.id = Akce.typ
        WHERE Interakce.clovek = ? ORDER BY Akce.datum DESC"
    );
    $query->bind_param("i", $id);
    $query->execute();
    if ($query->error) AppError::throw("SQL query error: {$query->error}");

    $result = $query->get_result();
    $result_array_akce = array();
    while ($row = $result->fetch_assoc()) array_push(
        $result_array_akce, $row
    );
    /* ... */
    return array(
        "akce" => $result_array_akce,
        /* ... */
    );
}

```

Třída **Login** obsahuje metody pro registraci, přihlášení, odhlášení a pro kontrolu, zda je někdo přihlášen. Při registraci je heslo zašifrováno pomocí algoritmu Blowfish a hash hesla uložen do databáze. Přihlášení uživatele je řešeno přes `$_SESSION`, kam se uloží jméno přihlášeného uživatele. Toto řešení není z hlediska bezpečnosti příliš dokonalé, ale vzhledem k tomu, že půjde o interní aplikaci nedostupnou mimo vnitřní síť, je dostačující. Na ukázce je volání funkce, která generuje hash z hesla. Konstanta `BCRYPT_COST` je definována v konfiguračním souboru.

```
$pass_hash = password_hash(
    $password,
    PASSWORD_DEFAULT,
    array("cost" => BCRYPT_COST)
);
```

Poslední třídou je **Queue**, tedy fronta. Ukládá obsah fronty do `$_SESSION`, aby byl dostupný i při přechodu mezi stránkami. Při přidávání do fronty se posílá jen parametr ID; pomocí dotazu do databáze se získá celý záznam o člověku.

4.6.4.2 Koncové body (endpoints)

V aplikaci existují čtyři soubory implementující koncové body API. Konkrétně to jsou:

- `history.php` – vrací historii interakcí člověka či organizace;
- `logreg.php` – umožňuje spouštět metody třídy `Login`;
- `query.php` – rozhraní pro posílání dotazů do databáze;
- `queue.php` – spouští operace nad frontou (metody třídy `Queue`).

Všechny tyto skripty přijímají data pouze prostřednictvím metod `POST` nebo `GET`, které také suplují `DELETE` a `PUT`, pokud je to třeba. Přijímaná data jsou zpracována a odeslána jako parametry metod tříd popsaných výše. Výstup je poté vypsán. Na následující ukázce je endpoint pro frontu (`queue.php`).

```

$action = $_GET["action"] ?? "";
$id = $_POST["id"] ?? "";
$params = $_POST["params"] ?? array();

$q = new Queue();

switch ($action) {
    case "add":
        if (empty($params)) AppError::throw("Params array is not defined");
        echo json_encode($q->add(json_decode($params)));
        die();
    case "remove":
        if (!$id) AppError::throw("ID is not defined");
        echo json_encode($q->remove($id));
        die();
    case "get":
        echo json_encode($q->get());
        die();
    case "clear":
        $q->clear();
        echo json_encode(array("state" => "success"));
        die();
    default:
        AppError::throw("Param \"action\" is invalid");
}

```

4.6.5 DOM funkce v JavaScriptu

Vzhledem k tomu, že není využit žádný framework jako React nebo Angular, je nutné pracovat s objektovým modelem HTML stránky napřímo. Pro zkrácení zápisu jsou připraveny funkce, které ve většině případů vrací referenci na DOM funkci nebo jinak manipulují s prvky tohoto objektového modelu. Některé z nich jsou ukázány v následujícím příkladu. Logika tohoto řešení je částečně podobná dříve velice populárnímu jQuery.


```

const _ = id => document.getElementById(id);

const getClass = name => document.getElementsByClassName(name);

const creElem = name => document.createElement(name);

const idListByTag = (tag, parent_id) =>
  Array.prototype.slice.call(
    _(parent_id)
      .getElementsByTagName(tag)
      .map(id => id.getAttribute("id"))
  );

const disable = id => _(id).disabled = true;

const enable = id => _(id).disabled = false;

const hide = id => typeof id === "string"
  ? _(id).style.display = "none"
  : id.style.display = "none";

const show = (id, style = "block") => typeof id === "string"
  ? _(id).style.display = style
  : id.style.display = style;

```

4.6.6 Získávání dat z koncových bodů

Na klientské části je k tomuto účelu využívána výhradně funkce `fetch()`. Jsou definovány obecné funkce pro získávání dat z databáze, frontu apod. V následující ukázce je část funkce, která odesílá požadavek na editaci záznamu a zpracovává výsledek. Dále je demonstrováno použití funkcí `_set` a `_get`, které spolu s `_del` pracují s vlastnostmi objektu `window`, což jsou v jistém smyslu globální proměnné.

```

const editQuery = form => {
  loading();
  const formData = new FormData(form);
  /* ... */
  fetch(
    getUrl(_get("page"), "edit"), {
      ...postData(form),
      body: formData
    }
  ).then(
    response => response.json()
  ).then(
    data => {
      const query = _get("last_query");
      const index = _get("edited_index");
      query[index] = {
        ...query[index],
        ...data,
        id: +_get("last_id").substring(3)
      }
      _set("last_query", query);
      fillSearchTable(query);
      closePanel("sidepanel_edit");
      /* ... */
      loading(false);
      _("edit").reset();
      dialog("success", "Záznam editován úspěšně");
    }
  ).catch (
    error => {
      console.error(error);
      loading(false);
      dialog("error", "Došlo k chybě při odesílání požadavku");
    }
  );
};

```

4.7 Testování uživatelského rozhraní

Po implementaci a nasazení beta verze na lokální server byla aplikace otestována. Jak již bylo zmíněno, šlo o testy uživatelského rozhraní kvalitativní metodou. Každý z pěti účastníků používal svůj počítač, na kterém vykonává běžné pracovní povinnosti.

Během tohoto moderovaného testování dostali uživatelé celkem jedenáct úkolů, které na sebe navazovaly. Jako první bylo přečteno zadání úkolu a nechán dostatečný čas na splnění. Nepodařilo-li se to uživateli (nebo váhal-li dlouho), dostal nápovědu. Pokud nebyl problém zjevný, byla zahájena diskuze, aby byly bezpečně identifikovány problematické komponenty uživatelského rozhraní. Na konci testování měl každý účastník možnost se vyjádřit k UI a předat své pocity, které ještě nezazněly.

Účastníci dostali tyto úkoly:

1. Zaregistrujte se a přihlašte se
2. Vytvořte organizaci se sídlem ve Vlašimi
3. Vyhledejte všechny organizace se sídlem ve Vlašimi
4. Vytvořte člověka, přiřadte mu tuto organizaci a určete, které položky jsou na formuláři povinné
5. Přidejte nový dar, jako dárců vyberte člověka (s organizací) vytvořeného v předchozím kroku
6. Vložte do fronty libovolné dva lidi
7. Vytvořte novou akci; jako účastníky vložte lidi z fronty, k nimž při vytváření akce přidejte ještě libovolného jednoho dalšího
8. Vytvořte e-mail pro vámi vybrané lidi a odešlete jej
9. Najděte historii interakcí libovolného člověka
10. Odstraňte tuto novou organizaci a člověka
11. Odhlašte se

5 Výsledky a diskuze

5.1 Výsledky testování uživatelského rozhraní

Jak již bylo zmíněno, testu se zúčastnilo celkem pět lidí. Následující seznam je utříděn podle toho, s kolika účastníky byl tento problém nalezen.

5.1.1 Problémy nalezené se čtyřmi účastníky

- U vyhledávání není zjevné, jak se zadávají parametry, je zobrazeno pouze tlačítko na přidání. Uživatel očekává, že systém zobrazí pole pro první parametr už při otevření postranního vyhledávacího panelu.

5.1.2 Problémy nalezené se třemi účastníky

- Tlačítko Poslat u akce typu E-mail by spíše mělo mít popis Napsat.
- Pokud je do pole s automatickým doplňováním možné zadat jen malé množství hodnot, je lepší použít statický select box.

5.1.3 Problémy nalezené se dvěma účastníky

- Při zadání příliš dlouhého uživatelského jména při registraci systém nezobrazí chybovou hlášku.
- Jakmile je potvrzeno vyhledávání a panel se zavře, po znovuotevření již nejsou zobrazeny původní parametry. Pokud není nic nalezeno, panel by se neměl ani zavřít.
- U polí s automatickým doplňováním (autocomplete) se nic neděje, pokud do něj uživatel klikne a není jasné, že má začít psát. Uživatel očekává nápovědu.
- Postranní panel nezobrazí posuvník, pokud se nevejde na výšku na obrazovku.
- U pole s názvem Dárce není uvedeno, že lze zadávat člověka i organizaci.

- Pokud se uživatel pokusí odeslat e-mail člověku, který tento kontaktní údaj nemá uvedený, tak by měl systém zobrazit kromě chyby také odkaz na editaci tohoto člověka.
- Otevírání tabulek se seznamem darů a účastí na akcích by mělo být možné i kliknutím na nadpis, nejen na šipku.
- Tlačítko Odstranit je v tabulce zbytečné, stačí v panelu Detail, jde o zřídka používanou akci.
- Tlačítko Odhlásit se by mělo být v pravém horním rohu, nikoli dole.
- Šedé tlačítko Zavřít panel vypadá neaktivní, barva je nevhodně zvolena.

5.1.4 Problémy nalezené s jedním účastníkem

- U vyhledávacích parametrů by bylo lepší přidat filtr pro nevyplněné namísto filtru pro vyplněné sloupce.
- Tlačítko Fronta v tabulce není dostatečně výrazné. Toto tlačítko by mělo zmizet, pokud již účastník ve frontě je.
- Odkaz na historii interakcí je málo výrazný.
- Tlačítko Fronta by mělo být přidáno i na panelech Detail a Historie interakcí.
- V panelu Detail jsou pole nepřehledně utříděna. Uživatel například očekává jméno a příjmení vedle sebe, nikoli pod sebou.

5.2 Zhodnocení výsledků testování uživatelského rozhraní

Přestože účastníci testu hodnotili celkový dojem z uživatelského rozhraní spíše jako kladný, bylo nalezeno velké množství nedokonalostí. Pravděpodobným důvodem výskytu těchto problémů je chybný počáteční předpoklad, že není zapotřebí vytvářet prototyp. Ukázalo se, že ani intenzivní konzultace a kreslení drátěných modelů nemůže nahradit autentický zážitek z vyzkoušení prototypu. Vzhledem k tomu, že opravy těchto chyb budou

prováděny na již hotové betaverzi, zabere tento proces násobně více času, než kdyby byly tyto problémy odladěny ještě na úrovni prototypu.

5.3 Zhodnocení splnění funkčních požadavků

Výsledek ohledně splnění funkčních požadavků není zatím znám. Za prvních deset dní testovacího provozu se ale kromě nápadů na vylepšení nad rámec původního zadání nic neobjevilo. Lze tedy předpokládat, že funkční požadavky jsou splněny dostatečně, ale nelze to v tuto chvíli ještě říci jistě, neboť testovací provoz trvá zatím příliš krátkou dobu.

5.4 Zhodnocení volby technických prostředků

PHP a MySQL lze považovat za již tradiční volbu; jsou to technické prostředky, které spolu v součinnosti až na drobné výjimky fungují rychle a dobře. V dnešní době ale není příliš standardní programovat klientskou část webové aplikace v čistém JavaScriptu. Tento výběr znamenal vyšší časové náklady na vývoj, ale během testovacího provozu betaverze bylo možné pozorovat latence pouze v řádu jednotek milisekund i u tisíce záznamů.

5.5 Návrhy na rozvoj do budoucna

Kromě zmíněných problémů s uživatelským rozhraním, které musí být vylepšeno lze navrhnout několik funkcionalit a vylepšení, které by mohly být přidány.

První takovou funkcionalitou jsou automatické importy a exporty dat. Pro efektivitu práce by určitě bylo přínosné mít možnost automaticky přidávat dary z účetního systému. Zpočátku ostrého provozu produkční verze bude také nutné přidat mnoho dat z různých zdrojů; zaměstnanci organizace by tedy možná uvítali funkci importu XLSX souborů.

Dalším vylepšením by mohly být automatické e-maily posílané v reakci na určité akce (například poděkování dárci po přijetí daru) a s tím související předdefinované e-mailové šablony.

Pokud by v budoucnu docházelo často k editacím formulářů a jiných prvků na stránce, bylo by možné přemístit data o nich z PHP metod do databáze. To by znamenalo

jednodušší a rychlejší úpravy a přehlednější zobrazování parametrů. Také by bylo možné připravit funkcionalitu, která by umožňovala editaci formulářů přes API z klientské části aplikace bez nutnosti měnit data přímo v databázi.

Vhodnou úpravou by bylo také nahradit stylopis ve formátu Less již předem zkompilovaným CSS kódem. V tom případě by již nebylo nutné používat skript less.js a tak by pravděpodobně došlo ještě k dalšímu zrychlení aplikace.

6 Závěr

Cílem této práce byl návrh, implementace a otestování webové aplikace v rámci neziskové organizace, která je určena pro evidenci údajů o fyzických a právnických osobách, které mají určitý vztah k této organizaci. Další důležitou funkcí této aplikace je možnost sledovat historii interakcí s těmito subjekty, tedy sponzorské dary, proběhlá e-mailová korespondence či účasti na akcích.

V teoretické části, která spočívala ve studiu informačních zdrojů byl vymezen rámec praktické části, tedy teoretická východiska.

Praktická část spočívala v analýze požadavků, ve volbě vhodného postupu vývoje a technických prostředků. Byl proveden návrh databáze a uživatelského rozhraní, z čehož bylo možné vycházet v implementační části.

V části nazvané Výsledky a diskuze byly shrnuty poznatky získané během testování aplikace; dále bylo zhodnoceno, které předpoklady byly během návrhu a vývoje správné a které spíše ne. Byly navrženy možnosti dalšího rozvoje do budoucna, z nichž některé budou přidány do produkční verze aplikace, která ale není předmětem této bakalářské práce.

7 Seznam použitých zdrojů

1. Interaction Design. Interaction Design Foundation [online]. [cit. 2021-02-02].
Dostupné z: <https://www.interaction-design.org/literature/topics/interaction-design>
2. SMITH, Alan. A Brief Introduction To Interaction Design. Usability Geek [online].
[cit. 2021-02-05]. Dostupné z:
<https://usabilitygeek.com/introduction-interaction-design/>
3. STANÍČEK, Petr. Dobrý designér to všechno ví!. I. vydání. Kamenné Žehrovice:
vydáno vlastním nákladem autora, 2016. ISBN 978-80-260-9427-2.
4. User Interface Design. Interaction Design Foundation [online]. [cit. 2021-02-05].
Dostupné z: <https://www.interaction-design.org/literature/topics/ui-design>
5. PAVLÍČEK, Josef. Interakce člověk a počítač: předmět v rámci studia. Praha:
Česká zemědělská univerzita v Praze, 2019
6. SNOW, Hu. What Is & How to Create Perfect Personas Step by Step. Mockplus
[online]. 2017 [cit. 2021-02-08]. Dostupné z:
<https://www.mockplus.com/blog/post/how-to-create-personas>
7. ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce
testováním. I. vydání. Brno: Computer Press, 2013. ISBN 978-80-251-3816-8.
8. ESPOSITO, Emily. Low-fidelity vs. high-fidelity prototyping. Inside Design
[online]. 2018 [cit. 2021-02-09]. Dostupné z:
<https://www.invisionapp.com/inside-design/low-fi-vs-hi-fi-prototyping/>
9. KRUG, Steve. Nenuťte uživatele přemýšlet!: praktický průvodce testováním a
opravou chyb použitelnosti webu. Brno: Computer Press, 2010. ISBN
978-80-251-2923-4.

10. Qualitative User Testing Methods. Usability Testing [online]. [cit. 2021-02-02].
Dostupné z:
<http://usabilitytesting.sg/user-experience-course/lesson-5-qualitative-user-testing-methods/>
11. CHRISTENSSON, Per. Webová aplikace. TechLib [online]. 2014 [cit. 2021-03-03]. Dostupné z: https://tech-lib.eu/definition/web_application.html
12. What do client side and server side mean? Client side vs. server side. Cloudflare [online]. [cit. 2021-03-04]. Dostupné z:
<https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/>
13. KLIMUSHYN, Mel. Web Application Architecture from 10,000 Feet, Part 1 – Client-Side vs. Server-Side. Atomic Object [online]. 2015 [cit. 2021-03-04].
Dostupné z:
<https://spin.atomicobject.com/2015/04/06/web-app-client-side-server-side/>
14. SIEGEL, Camille a Arun DORAIRAJAN. What is an API? API Friends [online]. 2020 [cit. 2021-03-04]. Dostupné z:
<https://apifriends.com/api-management/what-is-an-api/>
15. MALÝ, Martin. REST: architektura pro webové API. Zdroják [online]. 2009 [cit. 2021-03-04]. Dostupné z:
<https://zdrojak.cz/clanky/rest-architektura-pro-webove-api/>
16. HTTP response status codes. MDN Web Docs [online]. [cit. 2021-03-05]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
17. BROŽA, Petr. Tvorba www stránek pro úplné začátečníky. V. vydání. Brno: Computer Press, 2004. Bestseller (Computer Press). ISBN 80-251-1300-0.
18. HTML basics. MDN Web Docs [online]. [cit. 2021-03-05]. Dostupné z:
https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

19. Syntax - CSS: Cascading Style Sheets. MDN Web Docs [online]. [cit. 2021-03-05].
Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/CSS/Syntax>
20. Features In-Depth. Less.js Documentation [online]. [cit. 2021-03-06]. Dostupné z:
<http://lesscss.org/features/>
21. Introduction - Bootstrap v5.0. Bootstrap Docs [online]. [cit. 2021-03-06]. Dostupné z:
<https://getbootstrap.com/docs/5.0>
22. JavaScript. MDN Web Docs [online]. [cit. 2021-03-06]. Dostupné z:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
23. What is JavaScript? MDN Web Docs [online]. [cit. 2021-03-06]. Dostupné z:
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
24. What is PHP? PHP Documentation [online]. [cit. 2021-03-06]. Dostupné z:
<https://www.php.net/manual/en/intro-what-is.php>
25. What can PHP do? PHP Documentation [online]. [cit. 2021-03-06]. Dostupné z:
<https://www.php.net/manual/en/intro-whatcando.php>
26. Proč je PHP paskvil a proč se přesto tak hojně používá? IMPnet [online]. 2018 [cit. 2021-03-06]. Dostupné z:
<https://www.impnet.cz/blog/proc-je-php-paskvil-a-roc-se-presto-tak-hojne-pouziva/>
27. Introduction. Twig Documentation [online]. [cit. 2021-03-06]. Dostupné z:
<https://twig.symfony.com/doc/3.x/intro.html>
28. BODNAR, Jan. Twig tutorial. ZetCode [online]. 2020 [cit. 2021-03-06]. Dostupné z:
<https://zetcode.com/php/twig/>
29. HERNANDEZ, Michael J. Návrh databází. II. vydání. Praha: Grada, 2006.
Profesionál. ISBN 80-247-0900-7.

30. HORVÁTH, Tomáš. Teoretický úvod do relačních databází. Programujte.com [online]. 2007 [cit. 2021-03-07]. Dostupné z:
<http://programujte.com/clanek/2007110801-teoreticky-uvod-do-relacnich-databazi/>
31. GRUBER, Martin. Mistrovství v SQL. I. vydání. Praha: Softpress, 2004. ISBN 80-864-9762-3.
32. KNOWLES, Curtis. 6NF Conceptual Models and Data Warehousing 2.0. 2012. SAIS 2012 Proceedings. Georgia Southern University.
33. OPPEL, Andrew J. SQL bez předchozích znalostí: průvodce pro samouky. I. vydání. Brno: Computer Press, 2008. ISBN 978-80-251-1707-1.
34. MySQL Documentation. MySQL [online]. [cit. 2021-03-08]. Dostupné z:
<https://dev.mysql.com/doc/>

8 Přílohy

Příloha 1 – Snímky obrazovky z aplikace

Příloha 2 – CD se zdrojovým kódem aplikace a s databází

Příloha 1 – Snímky obrazovky z aplikace

ČSOP Vlašim
Přihlášení

Uživatelské jméno

Heslo

[➔ Přihlásit se](#)

Ještě nemáte účet? [Zaregistrujte se.](#)

ČSOP Vlašim
Registrace

Uživatelské jméno

Heslo

Heslo znovu pro kontrolu

Zvací kód

[➔ Registrovat se](#)

Již máte účet? [Přihlaste se.](#)

ČSOP Vlašim

[Lidé](#)
[Organizace](#)
[Dary](#)
[Akce](#)

Lidé

JMÉNO	PŘÍJMENÍ	E-MAIL	TELEFON	ULICE
Jiří	Ota	jiří@seznam.cz	77766555	Blanická 123
Michal	Šustek	m.sustek@gmail.com	602334556	Alžirská
Pavel	Pěkný	pavel.pekny@seznam.cz	728779900	Žižkov
Petr	Vesecký	petr.vesecky@email.cz		Janáč
Štěpán	Novák	stepannovak@centrum.cz	603111222	Antala
Kateřina	Sochorová	sochorova.k@seznam.cz		

Přihlášený uživatel: **stepan**

Detail

Jméno * **Ulice a č. p.**

Příjmení * **Město**

E-mail **Organizace**

Pracovní e-mail **Pracovní pozice**

Telefon **Kontaktovat**

Poznámka

Zadal admin
Editoval xxxx
Datum editace 1. 3. 2021 (8:44)
[Historie interakcí](#)

ČSOP Vlašim

[Lidé](#)
[Organizace](#)
[Dary](#)
[Akce](#)

Lidé

JMÉNO	PŘÍJMENÍ	E-MAIL	TELEFON	ULICE
Jiří	Ota	jiří@seznam.cz	77766555	Blanická 123
Michal	Šustek	m.sustek@gmail.com	602334556	Alžirská
Pavel	Pěkný	pavel.pekny@seznam.cz	728779900	Žižkov
Petr	Vesecký	petr.vesecky@email.cz		Janáč
Štěpán	Novák	stepannovak@centrum.cz	603111222	Antala
Kateřina	Sochorová	sochorova.k@seznam.cz		

Přihlášený uživatel: **stepan**

Přidat

Jméno *

Příjmení *

E-mail

Pracovní e-mail

Telefon

Poznámka

ČSOP Vlašim

[Lidé](#)
[Organizace](#)
[Dary](#)
[Akce](#)

Vyhledávání

Parametry

Jméno -

Organizace -

Město Vlašim

+

Řadit podle

Vyberte...

Zobrazit sloupce

Jméno
 Příjmení
 E-mail
 Pracovní e-mail

Telefon
 Ulice a č. p.
 Město
 Organizace

Pracovní pozice
 Kontaktovat
 Zadal
 Editoval

Datum editace

JMÉNO	PŘÍJMENÍ	E-MAIL	TELEFON	ULICE
Jiří	Ota	jriri@seznam.cz	777666555	Blanice
Michal	Šustek	m.sustek@gmail.com	602334556	Alžírská
Pavel	Pěkný	pavel.pekny@seznam.cz	728779900	Žižkov
Petr	Vesecký	petr.vesecky@email.cz		Janač
Štěpán	Novák	stepannovak@centrum.cz	603111222	Antala
Kateřina	Sochorová	sochorova.k@seznam.cz		

Přihlášený uživatel: **stepan**

ČSOP Vlašim

[Lidé](#)
[Organizace](#)
[Dary](#)
[Akce](#)

Fronta

Michal Šustek (Město Vlašim) X
Pavel Pěkný X
Petr Vesecký X
Štěpán Novák X

Lidé

JMÉNO	PŘÍJMENÍ	E-MAIL	TELEFON	ULICE A Č. P.	MĚSTO	↑↑ Všechny do fronty
Jiří	Ota	jriri@seznam.cz	777666555	Blanická 123	Vlašim	<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>
Michal	Šustek	m.sustek@gmail.com	602334556	Alžírská 7	Praha 6	<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>
Pavel	Pěkný	pavel.pekny@seznam.cz	728779900	Žižkovo nám. 8	Vlašim	<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>
Petr	Vesecký	petr.vesecky@email.cz		Janačkovo nábřeží 13	Praha 5	<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>
Štěpán	Novák	stepannovak@centrum.cz	603111222	Antala Staška 1028/69	Praha 4	<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>
Kateřina	Sochorová	sochorova.k@seznam.cz				<input type="button" value="Fronta"/> <input type="button" value="Detail"/> <input type="button" value="Odstranit"/>

Přihlášený uživatel: **stepan** Odhlásit se

ČSOP Vlašim

Lidé Organizace Dary Akce

Fronta

Vyčistit

Pavel Pěkný Petr Vesecký Štěpán Novák

Akce

Vyhledávání

Přidat

NÁZEV	DATUM	TYP	ÚČASTNÍCI			
Informace o nadcházejících akcích	1. 3. 2021	Email	Jiří Ota, Michal Šustek, Pavel Pěkný, Petr Vesecký, Štěpán Novák (Město Vlašim)	Postat	Účastníci do fronty	Detail Odstranit
Zajímavá přednáška	6. 3. 2021	Přednáška	Štěpán Novák, Kateřina Sochorová		Účastníci do fronty	Detail Odstranit
Meeting	1. 3. 2021	Schůzka	Jiří Ota (Druhá Firma), Michal Šustek (Město Vlašim)		Účastníci do fronty	Detail Odstranit
Výzva k zaplacení poplatku	1. 3. 2021	Email	Petr Vesecký	Postat	Účastníci do fronty	Detail Odstranit
Pozvání na zahradní slavnost	31. 3. 2021	Ostatní	Michal Šustek, Kateřina Sochorová		Účastníci do fronty	Detail Odstranit

Přihlášený uživatel: **stepan**

Odhlásit se

ČSOP Vlašim

Lidé Organizace Dary Akce

Fronta

Vyčistit

Pavel Pěkný Petr Vesecký Štěpán Novák

Organizace

Vyhledávání

Přidat

NÁZEV	ULICE A Č. P.
Město Vlašim	Jana Masaryka 123
První Firma	
Druhá Firma	Tylov 456

Přihlášený uživatel: **stepan**

Historie interakcí (Město Vlašim)

Dary

DATUM	ČÁSTKA	ÚČEL	JMÉNO	PŘÍJMENÍ
1. 3. 2021	1234		Michal	Šustek
15. 10. 2020	1500	Zeleň	Štěpán	Novák

Účasti na akcích

DATUM	NÁZEV	TYP	JMÉNO	PŘÍJMENÍ
1. 3. 2021	Meeting	Schůzka	Michal	Šustek
1. 3. 2021	Informace o nadcházejících akcích	Email	Štěpán	Novák

Zavřít panel

ČSOP Vlašim

Lidé Organizace Dary Akce

Fronta Vyčistit

Jiří Ota Michal Šustek Pavel Pěkný Petr Vesecký

Akce Vyhledávání Přidat

NÁZEV	DATUM	TYP	ÚČASTNÍCI
Informace o nadcházejících akcích	1. 3. 2021	Email	Jiří Ota, Michal Šustek, Pavel Pěkný, Petr Vesecký
Zajímavá přednáška	6. 3. 2021	Přednáška	Štěpán Novák, Kateřina Sochorová
Meeting	1. 3. 2021	Schůzka	Jiří Ota (Druhá Firma), Michal Šustek (Město Vlašim)
Výzva k zaplacení poplatku	1. 3. 2021	Email	Petr Vesecký
Pozvání na zahradní slavnost	31. 3. 2021	Ostatní	Michal Šustek, Kateřina Sochorová

Přihlášený uživatel: **stepan**

Účastníci

+ Přidat + Vložit z fronty Odstranit všechny


Jiří Ota Michal Šustek Pavel Pěkný Petr Vesecký

Štěpán Novák (Město Vlašim)

Uložit Zavřít panel

Chcete tohoto člověka vložit do fronty jako zástupce organizace?

Ano
Ne



Záznam editován úspěšně

Příloha 2 – CD s databází a zdrojovým kódem aplikace

CSOP_Vlasim.sql

CSOP_Vlasim.zip