ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace pro sdílené nákupní seznamy
Student:	Sára Juranková
Vedoucí:	Ing. Vojtěch Jirkovský
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je návrh a implementace webové aplikace pro vytváření a editaci nákupních seznamů a jejich sdílení mezi uživateli.

1) Analyzujte stávající řešení aplikací na sdílení nákupních seznamů.
2) Proveďte návrh vlastní aplikace.
3) Na základě analýzy a návrhu aplikaci implementujte.
- pro implementaci použijte PHP framework Symfony.
- aplikace by měla být navržena jako responzivní web, jehož uživatelské rozhraní se přizpůsobuje zobrazení na různých typech zařízení.
4) Proveďte testování aplikace a jeho výsledky vyhodnoťte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 5. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webová aplikace pro sdílené nákupní seznamy

Sára Juranková

Katedra softwarového inženýrství
Vedoucí práce: Ing. Vojtěch Jirkovský

10. května 2018

Poděkování

Děkuji Ing. Vojtěchu Jirkovskému za vedení mé práce. Dále bych chtěla poděkovat své rodině za podporu při psaní bakalářské práce a při studiu obecně.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2018 .....

Odkaz na tuto práci

Juranková, Sára. *Webová aplikace pro sdílené nákupní seznamy.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018. Dostupný také z WWW: ⟨`https://gitlab.fit.cvut.cz/juransar/bakalarka`⟩.

Abstrakt

Tato práce analyzuje současné řešení aplikací pro tvorbu a sdílení nákupních seznamů a navrhuje a implementuje vlastní aplikaci „Sdílené nákupy“. Aplikace nabízí tvorbu a sdílení nákupních seznamů mezi uživateli. Oproti ostatním aplikacím nabízí několik vlastních řešení, která zajistí její snadné využití a ovládání primárně českými uživateli. Přínosem této práce je alternativní řešení usnadňující organizaci jedné z nejčastějších aktivit moderního člověka – nakupování a spolupráci při něm.

Klíčová slova webová aplikace, sdílené seznamy, symfony, php

Abstract

This work analyzes existing web applications used for creating and sharing shopping lists and designs and implements a new one, Shared shopping lists. This applications offers a way to create and share shopping lists between users in order to simplify one of the most frequent activities of a modern person: shopping for family or friends. The application is designed in a way that makes it ease to use, primarily for czech users

Keywords web application, shared lists, symfony, php

Obsah

Seznam obrázků

Úvod

Dnešní společnost je bez debaty společností konzumní. Nakupování je součástí každodenního života všech, čím dál více se přesunuje z kamenných obchodů do sféry online nakupování a do online světa se přesunuje i jeho organizace.

Zároveň je jasné, že nakupování není vždy pouze záležitostí jednoho člověka. Téměř každý alespoň někdy nakupuje pro více lidí, například pro rodinu nebo přátele. Aplikace, kterou mám v plánu vybudovat, má za cíl zjednodušit významnou aktivitu života moderního člověka a nabízí nástroj tvorby a sdílení nákupních seznamů.

Usnadnění nákupů pro sdílenou domácnost nebo na společenské akce je hlavním cílem Sdílených nákupů.

Aplikace je určena všem, kteří chtějí koordinovat nákupy s ostatními členy rodiny, se svými spolubydlícími a ostatními známými. Vhodná je také při nakupování na společenskou akci nebo dovolenou. Ačkoliv mnoho aplikací pro tvorbu nákupních seznamů již existuje, neexistují, nebo nejsou ideální pro využití v českém prostředí, a chybí jim funkce pro koordinaci nákupů klíčové. Toto téma jsem si zvolila proto, že stávající řešení pokládám za neuspokojivá a věřím, že má implementace se s těmito nedostatky vypořádá.

Kapitola 1

Cíl práce

Cílem práce je vytvořit webovou aplikaci sloužící k vytváření nákupních seznamů a kooperativní sdílení seznamů s dalšími uživateli.

Samotnému vlastnímu řešení problematiky předchází analýza stávajících řešení, na jejímž základě jsou určeny funkční požadavky.

V implementační části jsem zvolila framework Symfony vyvinutý v jazyku PHP na základě kladných zkušeností s jeho předchozí verzí.

Zamýšlený přínos této práce spočívá v usnadnění organizace nákupů s participací a uplatněním požadavků více osob. Vytvářená aplikace má za cíl sloužit jako komplexní nástroj pro nakupování pro rodinu nebo přátele, případně na společenské akce. Aplikace by měla nabízet možnost vytváření nákupních seznamů, jejich sdílení s dalšími uživateli a také jejich úpravy, jako kopírování a spojování.

Kapitola 2

Analýza

2.1 Cílová skupina uživatelů

V rámci předmětu BI-TUR jsme spolu se spolužákem Kryštofem Novotným zkoumali, kdo by mohl být cílovou skupinou aplikace sloužící ke sdílení nákupních seznamů. Vytvořili jsme online dotazník, který měl za cíl zjistit, jak lidé aktuálně organizují své nákupy, zda používají k tvorbě seznamů aplikaci nebo webové rozhraní a jestli mají potřebu své nákupní seznamy sdílet. Celkový počet respondentů činil 88 lidí. Výsledky dotazníku najdete na přiloženém médiu.

Největší skupinu respondentů tvoří věková skupina 18–25 let (80,7 %) a od 26 do 35 let (18,2 %). Většina respondentů (95,5 %) žije v nějaké formě sdílené domácnosti, nejčastěji s rodiči. 65 % z nich alespoň někdy rozhoduje o tom, co se nakoupí.

Z 88 respondentů 33 % uvedlo, že nakupují nahodile, 29,5 % používá papírový nákupní seznam, 19,3 % respondentů využívá při nakupování poznámky v mobilu a 5,6 % respondentů využívá aplikaci určenou k tomuto účelu.

Přestože na základě výsledků dotazníkového šetření využívá aplikaci k tvorbě a sdílení nákupních seznamů poměrně malý počet respondentů, 51,1 % uvedlo, že nákupní seznamy sdílet potřebují. Na dotazník odpovídali převážně mladí lidé, kteří často využívají moderní technologie a aplikace usnadňující běžné činnosti jako je navigace, hledání spojů, vyhledávání restaurací a podobně. Lze předpokládat, že aplikaci ke sdílení nákupních seznamů nevyužívají, protože je takové řešení nenapadlo, nebo nenašli takovou, která by jim vyhovovala.

Za cílovou skupinu na základě průzkumu a na základě vlastních pozorování byli zvoleni lidé žijící ve sdílené domácnosti s rodinou, spolubydlícím nebo partnerem, kteří jsou zvyklí využívat webové nebo mobilní aplikace. Předpokládá se i situace, ve které mladší člen rodiny iniciuje starší rodinné příslušníky, se kterými sdílí domácnost, k využívání takovéto aplikace. Ti

už mohou mít s využíváním moderních technologií a zejména aplikací méně zkušeností. Při vývoji uživatelského prostředí bude tedy kladen důraz na jednoduchost a snadnou orientaci v aplikaci.

2.2 Funkční požadavky

2.2.1 F1 Kontrola přístupu

Uživatel se pro přístup do aplikace bude povinen registrovat. Při registraci bude nutné uvést e-mail, uživatelské jméno a heslo. Po registraci bude uživateli zaslán kontrolní e-mail, obsahující odkaz na smazání profilu pro případ, kdy se někdo pokusí registrovat pod cizím e-mailem. Na základě přihlašovacího jména a hesla se bude uživatel do aplikace přihlašovat.

2.2.2 F2 Správa profilu

Registrovanému uživateli bude umožněno svůj profil upravovat. Bude povolena změna e-mailu, zapnutí nebo vypnutí e-mailových notifikací a smazání profilu.

2.2.3 F3 Správa přátel

Uživatel si bude moci přidávat ostatní uživatele do seznamu přátel. Zároveň již přidané přátele bude moci ze svého seznamu přátel odstranit.

2.2.4 F4 Vytváření a odstraňování nákupních seznamů

Uživatel bude vytvářet nákupní seznamy. Při vytváření seznamu zadá jeho název. Vytvořený seznam bude mít možnost smazat, čímž se smaže i uživatelům, s nimiž ho případně sdílel.

2.2.5 F5 Práce s nákupním seznamem

Uživatel bude moci sdílet vytvořený nákupní seznam s uživateli, které má v přátelích. Zároveň přátele, kteří přijali pozvánku do seznamu, bude mít možnost ze sdíleného nákupního seznamu odstranit. Do seznamu bude moci přidávat položky. Celý seznam bude možné automaticky roztřídit podle regálů v obchodě. Uživateli bude také umožněno nechat si svůj seznam zaslat na e-mailovou adresu. Pokud budou položky v seznamu obsahovat cenu produktu, bude výsledná cena automaticky spočítána a zobrazena v detailu seznamu.

2.2.6 F6 Práce s nasdíleným nákupním seznamem

Pokud uživatel přijme pozvánku ke sdílení nákupního seznamu s jiným uživatelem, bude mít možnost s ním nakládat podobně jako s vlastním seznamem. To znamená, že bude schopný jej sdílet s vlastními přáteli, které majitel

seznamu v přátelích nemá. Nebude mít možnost z tohoto seznamu odstranit ostatní uživatele, ani seznam úplně smazat. Nicméně uživatel bude moci seznam opustit. Pokud tak učiní, položky, které do seznamu tento uživatel přidal, v něm zůstanou.

2.2.7 F7 Manipulace s položkami

V aplikaci budou existovat dva způsoby přidávání položek. Prvním je běžné přidání položky s případnými doplňujícími informacemi jako je odkaz, obrázek, cena nebo množství. Druhým způsobem je kopírování položek z jiného seznamu. Položky v seznamu bude možné označit odškrtnutím za nakoupené nebo je ze seznamu zcela odstranit. Dále bude možné položky v seznamu manuálně řadit, v případě roztříděného seznamu i přesunovat položky mezi regály, dojde-li k chybnému zatřídění.

2.2.8 F8 Upozornění

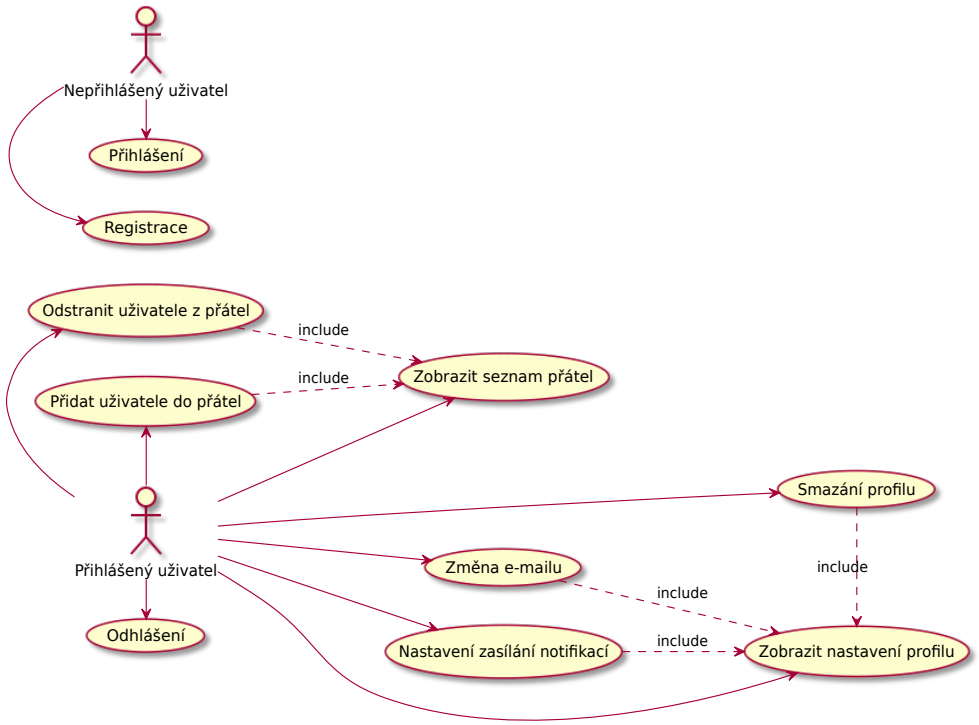
Aplikace bude generovat upozornění. Některá upozornění budou čistě informačního charakteru. Mezi tato upozornění se řadí informace o tom, že jiný uživatel přijal nebo odmítl žádost o přátelství, přijal či odmítl pozvánku do seznamu. Další upozornění budou obsahovat informace o odstranění daného uživatele ze seznamu přátel, o opuštění některého z jeho seznamů uživatelem, se kterým byl sdílen, nebo že byl uživatel ze sdíleného seznamu odstraněn. Další typ notifikací bude formulován jako dotazy. Tyto notifikace budou řešit otázky toho, zda chce uživatel přijmout nebo odmítnout pozvání do seznamu přátel jiného uživatele nebo do cizího nákupního seznamu. Notifikace si uživatel bude moci zobrazit, odstranit je a na notifikace ve formě dotazů bude moci zareagovat přijmutím nebo odmítnutím.

2.3 Nefunkční požadavky

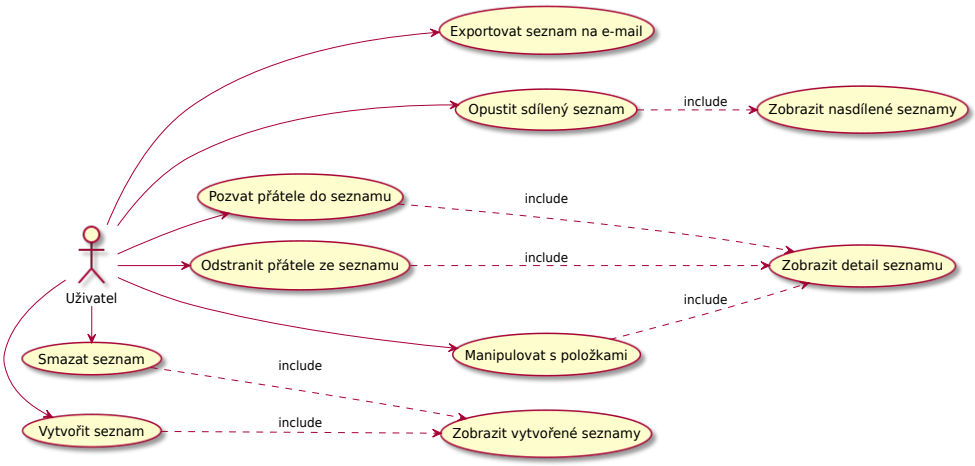
- Použití jazyka PHP a frameworku Symfony 3.4
- Použití ORM frameworku Doctrine 2
- Přehledné uživatelské rozhraní
- Responzivní uživatelské rozhraní, umožňující pohodlné zobrazení a obsluhu aplikace na různých typech zařízení
- Otevřené řešení – návrh umožňující jednoduché rozšíření a přidání funkcionalit

2.4 Případy užití

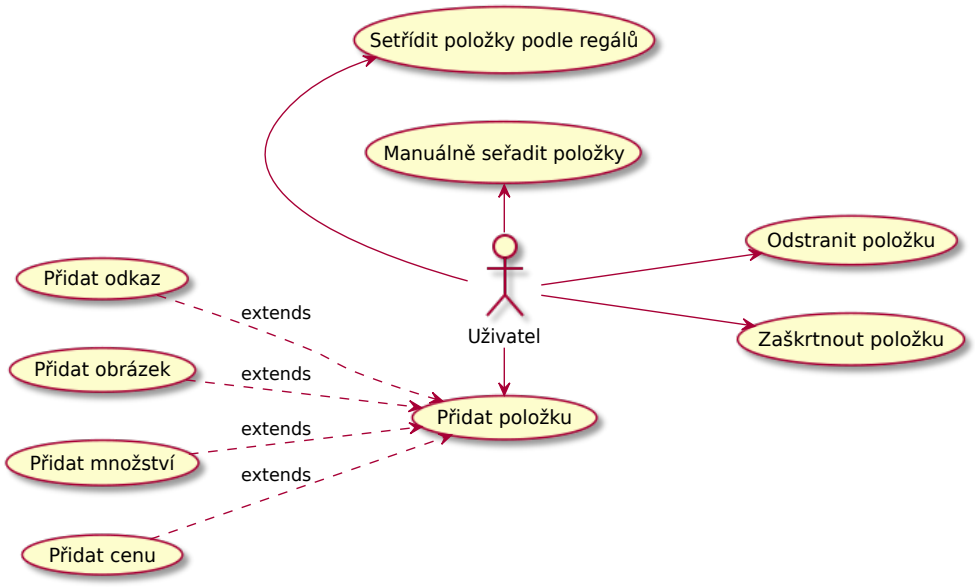
Diagramy 2.1-2.4 zobrazují případy užití. Celá aplikace obsahuje pouze dva účastníky, kterými jsou nepřihlášený/anonymní uživatel a přihlášený uživatel.



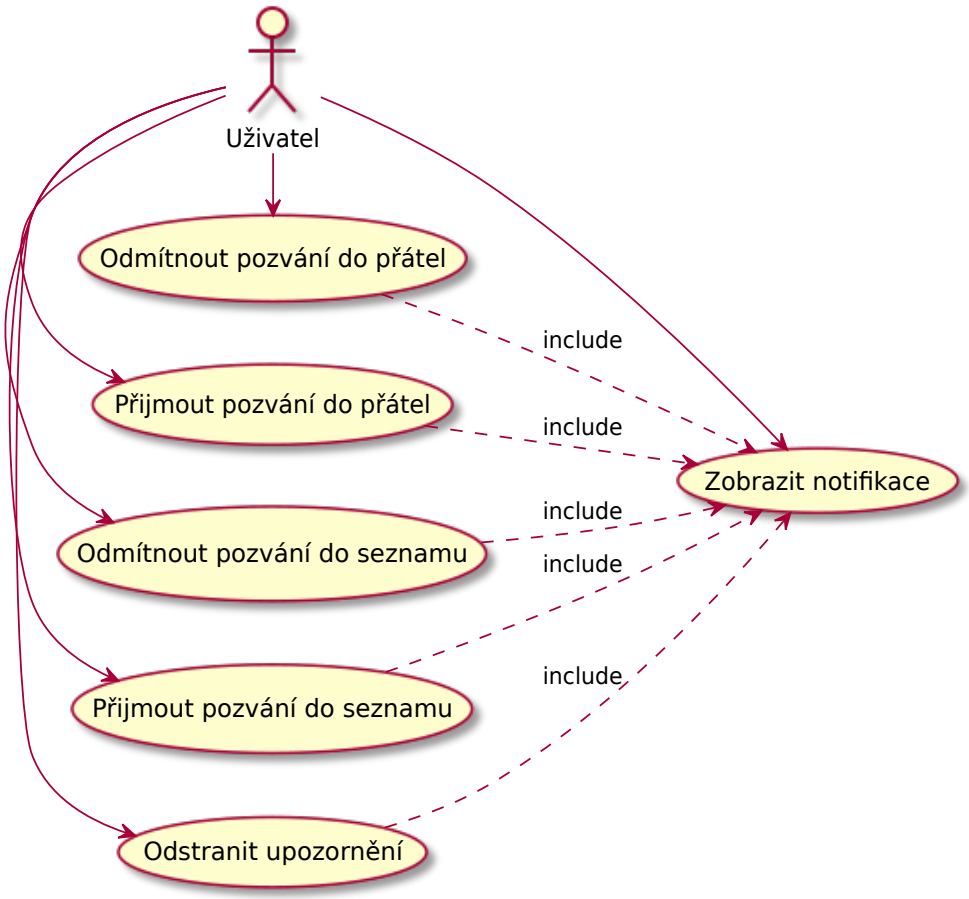
Obrázek 2.1: Případy užití – uživatelské účty



Obrázek 2.2: Případy užití – práce se seznamy



Obrázek 2.3: Případy užití – práce s položkami



Obrázek 2.4: Případy užití – notifikace

2.5 Současný stav řešení problematiky

V současnosti existuje více aplikací, které umožňují tvorbu a sdílení nákupních seznamů. Pro analýzu byly vybrány tři, které se jevily jako nejpodobnější aplikaci, jejíž tvorbou se tato práce zabývá, a které se nejvíce přibližují jejím funkčním požadavkům. Průzkum a zjištění technologií využívaných v těchto aplikacích byl proveden pomocí rozšíření do prohlížeče Wappalyzer [2].

2.5.1 OutofMilk.com

OutofMilk [3] je webovou aplikací, která se specializuje na vytváření a sdílení nákupních seznamů. Umožňuje přihlašování pomocí Facebooku, Google+ a e-mailu a její rozhraní je pouze v angličtině.

Tato aplikace umožňuje tvorbu seznamů poměrně pohodlným způsobem. OutofMilk umožňuje přesouvání položek mezi jednotlivými seznamy a tudíž si lze vytvořit několik soupisů základních potřeb z různých kategorií a pohodlně je přidat do libovolného vytvořeného nákupního seznamu.

Nevýhodou pro českého uživatele je již zmíněné anglické rozhraní a dále příliš mnoho možností a specifikací u položek, které běžný uživatel z České republiky pravděpodobně nevyužije. Mezi tyto možnosti patří například nastavení specifické ceny, která je pouze v dolarech, volitelné zahrnutí daně, nahrání kupónu a další. Naopak nelze doplnit informace o položce pomocí obrázku nebo odkazu.

OutofMilk běží na webovém serveru Apache a využívá frameworku Microsoft ASP.NET.

2.5.2 Tomikup.cz

Webová aplikace Tomikup je českou aplikací, prezentující se jako nástroj pro vytváření seznamů dárků, které chce uživatel dostat [4]. Umožňuje ovšem i vytváření sdílených nákupních seznamů, a proto byla do této analýzy zařazena.

Tomikup umožňuje registraci a přihlašování přes Google+, Facebook nebo přes e-mailovou adresu, pomocí které je zároveň možné pozvat někoho dalšího do aplikace.

Tvorba nákupních seznamů je v této aplikaci velmi snadná a rychlá. V průběhu zapisování položky se zobrazuje nápověda umožňující položku blíže specifikovat, například při zápisu cherry rajčat lze zadat pouze „raj“ a cherry rajčata se sama nabídnou.

Při zobrazení na velkých typech zařízení (stolní PC, notebook apod.) se zobrazují všechny seznamy na stejné stránce a nelze si zobrazit pouze jeden, což při větším počtu seznamů/položek snižuje přehlednost. Zobrazení na mobilních zařízeních tento problém nemá. Aplikace neumožňuje různé manipulace s již vytvořenými seznamy, jako jejich spojování a skládání. Aplikace

Tokmikup je optimalizovaná pro tvorbu tzv. wishlistů, tedy seznamů přání. Tvorba nákupních seznamů je až sekundární funkcí.

Tomikup běží na webovém serveru IIS. Využívá webového frameworku Microsoft ASP.NET a na frontendu používá mimo jiné JavaScript, což umožňuje například zmíněné našeptávání při tvorbě seznamů.

2.5.3 Wunderlist.com

Wunderlist je aplikace, která má webové rozhraní a je možné si ji stáhnout i jako aplikaci do mobilních zařízení [5]. Má podporu všech běžných operačních systémů mobilních zařízení. V této analýze se budeme zabývat její webovou verzí, kterou ovšem na mobilních zařízeních není možné běžným způsobem zobrazit v prohlížeči; po návštěvníkovi z mobilního zařízení totiž požaduje stáhnutí aplikace. Chceme-li tedy upravovat seznam a odškrtávat položky přímo v obchodě, je nutné nainstalovat aplikaci. Účet je možné si vytvořit pomocí Facebooku, Google+, Microsoft účtu a e-mailové adresy.

Aplikace umožňuje i jiné druhy seznamů, než pouze nákupní, například seznamy filmů, na které se chceme podívat, úkolníček a jiné. V rámci analýzy hodnotím pouze práci s nákupními seznamy.

Samotná tvorba i sdílení seznamů jsou velmi jednoduché. Stačí zadat položku do textového pole ve vybraném seznamu a stisknout klávesu enter. Položka se následně do seznamu přidá. Pro sdílení seznamu stačí zadat e-mail osoby, se kterou chceme seznam sdílet, ta se do aplikace přihlásí, odsouhlasí pozvánku do seznamu a seznam se jí automaticky přidá mezi ostatní.

Práce se seznamy je oproti jejich tvorbě vysoce neintuitivní. Pro úpravu položky je nutné na ni dvojkliknout, což je ve webových aplikacích poměrně nestandardní požadavek. Podobným způsobem lze k položce přidat detailní informace a vlastnosti, místo toho aby bylo vše nastaveno již při přidávání položky do seznamu. Je tedy nutné položku nejprve založit a až následně doplnit další detaily. Aplikace neumožňuje si někoho trvale přidat mezi přátele a je tedy nutné pokaždé znovu zadávat e-mail známého a o sdílení ho požádat.

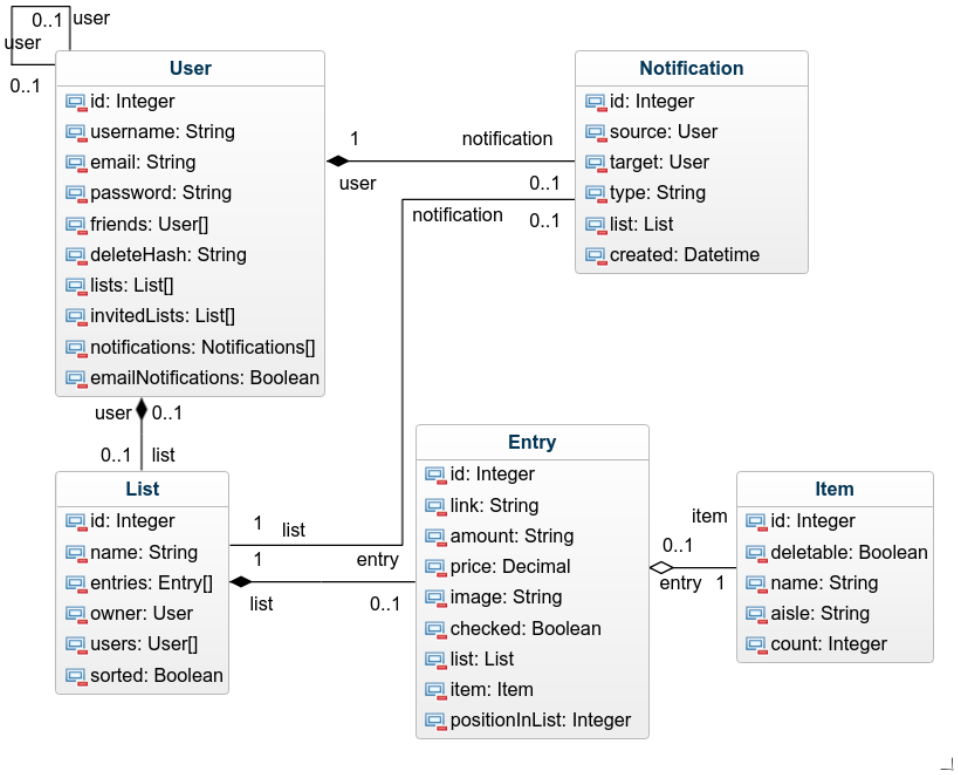
Wunderlist je aplikací založenou převážně na JavaScriptu.

Kapitola 3

Návrh

3.1 Model tříd

Tato sekce popisuje typy objektů v aplikaci, jejich vlastnosti a vztahy mezi třídami, pod které objekty spadají. Diagram tříd lze vidět na obrázku 3.1.



Obrázek 3.1: Návrh tříd

3.1.1 Třída User

Třída `User` reprezentuje uživatele aplikace. Jde se konkrétně o uživatele, kteří se do aplikace zaregistrovali.

- `id` - unikátní identifikátor uživatele
- `username` - unikátní jméno uživatele, pod kterým se přihlašuje do aplikace a které se zobrazuje ostatním uživatelům
- `email` - e-mail uživatele, na který mu chodí případná upozornění
- `password` - heslo uživatele
- `friends` - kolekce uživatelů, které má daný uživatel přidané mezi přátele
- `deleteHash` - unikátní hash každého uživatele, na jehož základě je generováno url zasílané e-mailem uživateli při registraci; slouží ke smazání uživatelského účtu bez přihlášení
- `lists` - kolekce nákupních seznamů, které uživatel vytvořil
- `invitedLists` - kolekce nákupních seznamů, do kterých byl uživatel pozván
- `notifications` - kolekce upozornění, mířených na daného uživatele
- `emailNotifications` - proměnná reprezentující, jestli má uživatel zapnuté/vypnuté zasílání upozornění na e-mail

3.1.2 Třída List

Třída `List` reprezentuje vytvořený nákupní seznam.

- `id` - unikátní identifikátor seznamu
- `name` - název seznamu
- `entries` - kolekce položek v seznamu
- `owner` - uživatel, který seznam vytvořil
- `users` - kolekce uživatelů, kteří byli do seznamu pozváni a pozvánku přijali
- `sorted` - proměnná reprezentující, jestli je seznam setříděný podle uliček/regálů

3.1.3 Třída Entry

Třída `Entry` reprezentuje položku v nákupním seznamu.

- `id` - unikátní identifikátor položky
- `link` - odkaz na doplňující informace k položce
- `amount` -množství, které chce uživatel zakoupit
- `price` - předpokládaná cena položky
- `image` - obrázek položky
- `checked` - proměnná určující, jestli je položka označena za nakoupenou
- `list` - nákupní seznam, pod který položka spadá
- `item` - objekt typu `Item` upřesňující některé vlastnosti položky
- `positionInList` - pořadí položky v nákupním seznamu (pořadí položek může uživatel měnit)

3.1.4 Třída Item

Třída `Item` obsahuje doplňující informace k položkám. Tyto informace nejsou rovnou obsaženy v třídě `Entry`, protože by docházelo k častým duplicitám v databázi.

- `id` - unikátní identifikátor položky
- `deletable` - slouží k rozeznání položek, které byly do databáze přidány uživateli a položek, které jsou předem připraveny
- `name` - jméno položky
- `aisle` - regál/ulička, pod kterou položka spadá
- `count` - proměnná, která drží informaci o tom, kolikrát se tato položka nachází v různých seznamech. Pokud je count 0 a deletable `true`, instance `Item` bude smazána z databáze.

3.1.5 Třída Notification

Třída `Notification` reprezentuje upozornění.

- `id` - unikátní identifikátor upozornění
- `source` - uživatel, který notifikaci "vytvořil". Například při žádosti o přátelství se jedná o žadatele.

- `target` - uživatel, na kterého je upozornění mířeno a kterému se zobrazuje. Například při pozvání do seznamu je to uživatel, který byl pozván.
- `type` - typ upozornění. Existuje několik typů, konkrétně se jedná o:
  - `FRIEND_R` - žádost o přátelství
  - `FRIEND_A` - přijetí žádosti o přátelství
  - `FRIEND_D` - odmítnutí žádosti o přáteltství
  - `WISHLIST_R` - pozvánka do nákupního seznamu
  - `WISHLIST_A` - přijetí pozvání do seznamu
  - `WISHLIST_D` - odmítnutí pozvání do seznamu
  - `IWISHLIST_D` - odstranění ze sdíleného seznamu
  - `IWISHLIST_UD` - opuštění sdíleného seznamu
  - `UFRIEND_R` - odstranění z přátel
- `list` - pokud se upozornění týká nějakého nákupního seznamu, tato proměnná je instancí daného seznamu
- `created` - časový údaj o tom, kdy byla notifikace vytvořena

3.2 Možnosti řešení

Jak je vidět i z analýzy stávajících řešení, existuje celá řada způsobů, jak webovou aplikaci řešit. V této kapitole uvedu příklady různých takových možností. Kapitola se také věnuje výběru programovacího jazyka a frameworku.

3.2.1 Programovací jazyk

Serverové aplikace typicky píší například v jazycích Java, Python nebo PHP. S ohledem na požadavky na projekt, jeho rozsah a mé dosavadní zkušenosti jsem se ve finále rozhodovala mezi jazyky Python a PHP.

3.2.1.1 Python

Python je multiplatformní jazyk navržený v 90. letech Guido van Rossumem, který umožňuje programování v procedurálním, objektovém i funkcionálním stylu.

Jednou ze zajímavostí z historie Pythonu je nekompatibilita mezi verzemi 2.x a 3.x. Ze strany vývojářů šlo o záměrný krok, kdy verze 3.x odstraňuje množství nedostatků v předchozí verzi a přidává například podporu Unicode. V dnešní době jsou stále používány obě verze, podpora Pythonu 2.x však končí v roce 2020. [6]

3.2.1.2 PHP

PHP (rekurzivní zkratka pro PHP: Hypertextový preprocesor) je open-source programovací jazyk, primárně používaný pro vývoj dynamicky generovaných webových stránek. Jeho první verze byla vyvinuta v roce 1994 Rasmusem Lerdorfem a od té doby jazyk prošel velkým vývojem. Syntaxe jazyka PHP čerpá z jazyků C, Java a Pearl. [7]

Dnes je PHP jeden z nejpopulárnějších jazyků, používaných na serverové straně webových aplikací, kdy 83,2 % webů, u kterých jsou k dispozici informace o použitých technologiích, používá právě PHP. [8]

3.2.2 Framework

Důležitou součástí při výběru technologií je pro programátora i výběr frameworku. Ten nejen usnadňuje psaní aplikací v jazyku, pro který je framework určen, ale dodává aplikacím, které jej využívají, jednotnou strukturu.

Jedním ze základních rysů, kterým se framework liší od klasických knihoven funkcí, je princip „inversion of control“ (IoC, česky obrácené řízení). Běh aplikace v tomto návrhovém vzoru není dán tím, jak jsou volány příkazy, ale tím jak posloupnost příkazů a volání funkcí definuje framework.

Jak pro Python, tak pro PHP existuje celá řada frameworků. Já jsem se rozhodla pro porovnání frameworků Django pro Python a dále Nette a Symfony pro PHP.

3.2.2.1 Django

Django [9] je open-source webový framework napsaný v Pythonu. Je udržován nezávislou organizací Django Software Foundation a vznikl v roce 2003. Architektura aplikací napsaných v Djangu užívá tzv. konceptu MVC (model-view-controller). Ten dělí aplikaci na tři logické části, kde `model` reprezentuje business logiku aplikace, `view` zobrazuje uživatelské rozhraní a `controller` reaguje na události a zajišťuje změny v datovém modelu nebo uživatelském rozhraní.

Django nabízí vlastní systém mapování entit na databázi, šablonovací systém a při dodržování standardů programování podle dokumentace jde o bezpečný systém s ochranou proti CSRF, SQL injection a dalším známým útokům na chyby v zabezpečení webových stránek.

Příklady webových aplikací používajících Django framework jsou například Pinterest, National Geographic nebo Open Stack.

3.2.2.2 Nette

Podle ankety serveru Zdroják [10] byl framework Nette zvolen jako nejpopulárnější a nejpoužívanější framework v České republice .

Původním autorem tohoto frameworku je David Grudl, dnes je spravován organizací Nette Foundation.

Nette má vlastní šablonovací systém Latte, debbugovací nástroj Tracy, testovací systém Tester a je zabezpečený proti celé řadě útoků [11].

Za nevýhodu tohoto frameworku považuji jeho nedostačující dokumentaci a malou mezinárodní komunitu.

3.2.2.3 Symfony

Symfony je open-source framework, vydaný v první verzi v roce 2005. Jeho vývoj je sponzorován firmou Sensio Labs [12]. Vychází z návrhového vzoru MVC.

Pro mapování entit databáze využívá Doctrine ORM a jako šablonovací systém tomuto frameworku slouží Twig.

Jedná se o velmi rozšířený framework s obrovskou komunitou vývojářů a výbornou dokumentací, která je stavěna na reálně využitelných příkladech.

3.3 Zvolené řešení

3.3.1 Symfony 3.4

Vzhledem k tomu, že dokumentace Nette je dle mého názoru nedostatečná, zvažovala jsem, zda pro tuto práci využít Python+Django, nebo PHP+Symfony. Pro implementaci jsem nakonec zvolila framework Symfony ve verzi 3.4, protože jsem měla již předchozí kladnou zkušenost se Symfony 2. Existuje sice i verze Symfony 4, která obsahuje nové funkce a vylepšení, jeho podpora je ovšem o 2 roky kratší než verze 3.4 (do července roku 2018 oproti listopadu 2020). [13] Dalším důvodem je osobní preference jazyka PHP oproti Pythonu.

Symfony 3.4 vyžaduje verzi PHP 5.5.9 nebo vyšší, mnou zvolená verze je nejnovější PHP 7.2.

3.3.1.1 Doctrine

Doctrine [14] v Symfony zajišťuje mapování entit na databázi a následnou komunikaci s touto databází.

Informace o mapování je možné Doctrine poskytnout pomocí YAML/XML souborů nebo pomocí DocBlock anotací přímo v definici třídy. Já zvolila techniku anotací, ve kterých specifikujeme datové typy proměnných, jejich atributy (délka, jedinečnost, název sloupce ve výsledné tabulce atd.). Příklad takového mapování lze vidět v ukázce 3.1.

```
//...
use Doctrine\ORM\Mapping as ORM;
/**
 * @ORM\Entity(repositoryClass="App\Repository\
    EntryRepository")
 */
class Entry
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\ManyToOne(targetEntity="Wishlist",
        inversedBy="entries")
     * @ORM\JoinColumn(name="w_id", referencedColumnName
        ="id")
     * @var Wishlist
     */
    private $wishlist;
//...
```

Ukázka kódu 3.1: Ukázka mapování pomocí anotací

V příkladu lze vidět, že Doctrine mimo jiné podporuje definování proměnné jako `id` s autoinkrementem, nebo i vazby M-N mezi entitami.

Ukládání objektů do databáze a jejich opětování získávání zajišťuje v Symfony objekt zvaný `EntityManager`.

3.3.1.2 Twig

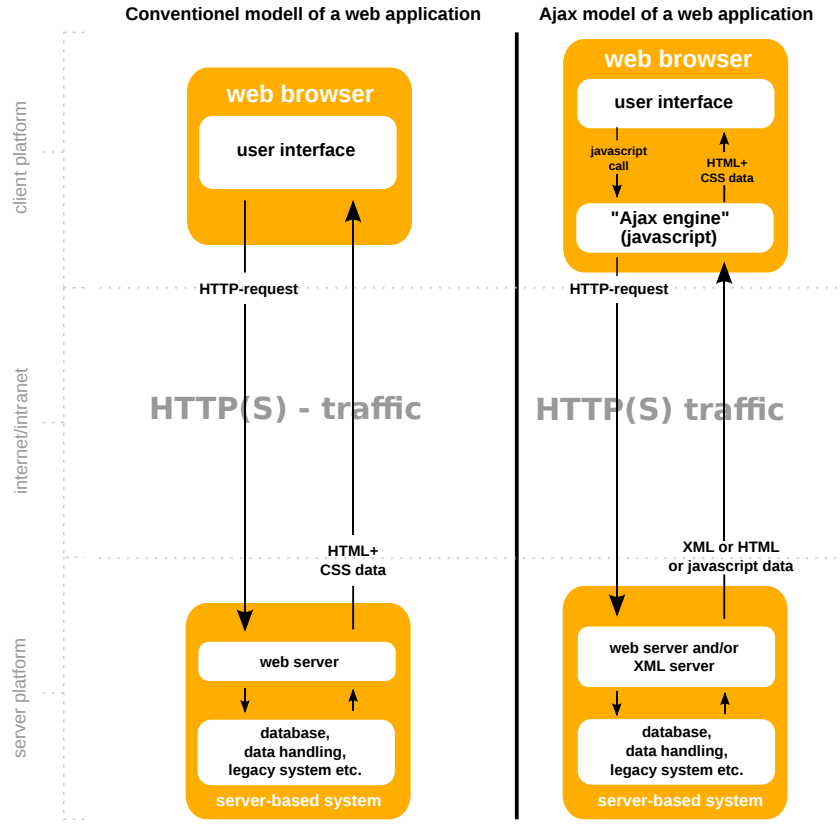
Twig [15] v Symfony slouží jako šablonovací systém. Díky automatickému využívání cache se jedná o systém s velmi rychlou odezvou. Dalšími vlastnostmi Twigu jsou automatické HTML escapování, možnost definice vlastních funkcí a filtrů a dědění šablon, díky kterému lze pro stránku vytvořit hlavní šablonu, od které ostatní šablony dědí a upravují jenom zvolené části.

Twig zároveň může běžet v takzvaném sandbox módu, který vyhodnocuje nedůvěryhodný šablonový kód a zvyšuje tak bezpečnost v případech, kdy uživatelé mohou modifikovat design šablony.

3.3.2 JavaScript

Pro některé funkcinality vykonávané na straně klienta jsem použila javascriptovou knihovnu JQuery [16]. Jedná se o open-source software a nejpoužívanější javascriptovou knihovnu [17].

JQuery mi primárně umožní realizovat asynchronní požadavky na server, při kterých bude upraven datový model a u klienta se překleslí pouze relevantní část stránky, bez nutného obnovení celého dokumentu v prohlížeči. Porovnání běžného HTTP požadavku s asynchroním požadavkem lze vidět na obrázku 3.2.



Obrázek 3.2: Synchronní a asynchronní požadavek na server [1]

Pro realizaci některých dalších funkcionalit použiji rozšíření jQuery nazvané jQuery UI [18]. Jde o nadstavbu jQuery zaměřenou na uživatelské rozhraní. Nabízí nástroje pro interakci, implementaci efektů a použití různých widgetů. Primárně tento framework využiji pro manuální řazení nákupních seznamů uživatelem použitím komponenty jQuery UI Sortable.

Pro našeptávání položek při tvorbě nákupního seznamu nebo pro přidávání uživatelů do přátel využiji javascriptovou knihovnu typeahead.js [19].

3.3.3 Bootstrap

Bootstrap [20] je frontendový framework, který umožňuje pohodlný vývoj uživatelského rozhraní a je kompatibilní se všemi moderními prohlížeči.

Při použití Bootstrapu se stránka chápe jako mřížka o dvanácti sloupcích, které lze používat individuálně, nebo je spojovat do širších sloupců. Zároveň lze pomocí `class` tagů definovat chování těchto sloupců na různých zařízeních (počítač, tablet, smartphone apod.), díky čemuž se snadněji vytváří responzivní webové rozhraní, tzn. rozhraní, které se přizpůsobuje všem typům zařízení, z nichž je web dostupný.

Kapitola 4

Realizace

4.1 Fungování a struktura aplikace

4.1.1 Adresářová struktura

Takto vypadá adresářová struktura mé aplikace. Následovala jsem tradiční strukturu Symfony aplikací.

- `config` ..... obsahuje konfigurační soubory
- `src` ..... obsahuje PHP zdrojové kódy
  - `Controller` ..... třídy obsluhující požadavky
  - `Entity` ..... definice entit
  - `Form` ..... třídy pro budování formulářů
  - `Functionality` ... třídy zajišťující komplexnější manipulaci s datovým modelem
  - `Repository` ..... třídy komunikující s databází
  - `Security` ..... security voters, zajišťující hlasování o přístupu
  - ..
- `templates` ..... obsahuje šablony Twig
- `public` ..... obsahuje veřejně dostupné soubory
- `vendor` ..... knihovny třetích stran nainstalované přes Composer
- `tests` ..... PHPUnit testy
- `.env` ..... soubor obsahující proměnné prostředí
- ..

4.1.2 Typická obsluha požadavku

Typická obsluha požadavku vypadá takto: pokud klient odešle požadavek na stránku, front controller s názvem `index.php` nacházející se ve složce `public` požadavek zpracuje a předá ho mnou definovaným controller třídám, na základě cest definovaných anotacemi.

Metoda controlleru, která obsluhuje požadavky pro danou cestu, poté obvykle provede nějakou manipulaci s datovým modelem. Nejčastěji zavolá některou z metod obsažených v třídách `src\Functionality`, které definují komplexní práci s datovým modelem, díky čemuž zpřehledňují metody controllerů.

Tyto třídy jsou takzvané služby (services), obsažené v objektu zvaném service container. Aby byla třída definovaná jako služba, je to nutné Symfony sdělit v konfiguračním souboru `config\services.yaml`, kde lze zároveň nadefinovat argumenty konstruktoru této třídy.

Metody tříd `Functionality` pak volají metody tříd `src\Repository`. Tyto třídy mají za úkol komunikovat s databází, získávat z ní objekty, upravovat je a opětovně ukládat nebo mazat.

Poté co došlo k případným změnám a jejich uložení do databáze se opět vracíme do controlleru. Nakonec metoda controlleru vrací odpověď. Obvykle se volá metoda `render`, ve které se specifikuje Twig šablona k vykreslení a data, která šablona vykresluje. Tato metoda vytvoří kompletní HTTP odpověď i s hlavičkami a vykreslenou šablonou.

V některých případech v aplikaci je v controlleru vracen objekt `JSONResponse`; jde především o asynchronní požadavky, například pro nastavení našeptávaných řetězců pro textové pole formuláře.

4.2 Kontrola přístupu

Ke kontrole přístupu lze přistupovat primárně ze dvou úhlů. Prvním je autentizace, to znamená určení identity uživatele a druhou je autorizace, jejímž hlavním úkolem je zamezení přístupu k určitým částem aplikace na základě identity poskytnuté procesem autentizace. V této části popíšu svojí implementaci těchto dvou konceptů.

4.2.1 Autentizace

Pro zajištění autentizace jsem zvolila tradiční metodu registračního formuláře, který vyžaduje unikátní uživatelské jméno, e-mail a heslo. Registrovaný uživatel se následně do aplikace přihlásí pomocí jména a hesla, které si určil. K tomu jsem tedy definovala vlastní třídu reprezentující uživatele, která implementuje `UserInterface`.

Dále bylo v konfiguračním souboru `security.yaml` nutné nadefinovat pro aplikaci firewall. Hlavním úkolem firewallu je určení způsobu autentizace uživatelů (v mém případě se tedy jedná o formulář), odkud má aplikace získávat informace o uživatelích, kam má přesměrovat nepřihlášené uživatele (snaží-li se dostat k informacím vyžadujícím přihlášení) nebo jaká je url adresa pro odhlašování. V tomto konfiguračním souboru jsem definovala třídu `User` jako poskytovatele autentizace pro firewall a nastavila metodu hashování

hesel pomocí bcrypt. Informace o uživateli s hashem jeho hesla se díky tomu načítají z mé databáze a zároveň se do ní ukládají.

4.2.2 Autorizace

Symfony umožňuje zamezit přístup k částem aplikace na základě uživatelských rolí. Pomocí regulárních výrazů se v konfiguračním souboru nastaví, ke které podskupině stránek podle url má uživatel dle své role povoleno přistupovat. Toho jsem využila pro určení hlavní stránky a stránek užívaných k registraci a přihlašování jako dostupných bez autentizace. Pro všechny ostatní stránky stačí `ROLE_USER`, což je kterýkoliv registrovaný a přihlášený uživatel.

Dalším nástrojem k autorizaci jsou v Symfony `Security voters`. Jde o nejpodrobnější přístup ke kontrole uživatelských oprávnění, které Symfony nabízí. Pro správné zabezpečení mé aplikace bylo nutné, aby například uživatel neměl přístup k jiným nákupním seznamům než k těm, které vytvořil, případně k těm, které s ním byly sdíleny. Podobně nesmí mazat uživatele ze seznamů, které nevlastní, nesmí manipulovat s upozorněními, která nejsou mířena na jeho osobu, mazat profily jiným uživatelům a podobně. Toho nelze docílit pouze pomocí již zmíněných rolí. Definovala jsem tedy vlastní třídy, implementující `VoterInterface`, která kontrolují, zda uživatel má potřebné oprávnění.

Potřebná kontrola přístupu se pak provádí v příslušné metodě controlleru, kde se zavolá například $this−>denyAccessUnlessGranted('edit', $list);, což zavolá příslušný `ListVoter`, který zkontroluje, že aktuálně přihlášený uživatel má pravomoci k provádění úprav zadaného seznamu.

4.3 Práce s formuláři

4.3.1 Vytváření formulářů

Formuláře se v Symfony tvoří pomocí komponenty `The Form Component`. Nejobvyklejším řešením je vytvořit pro každý formulář třídu, ve které se definují jeho náležitosti, díky čemuž lze formulář použít opakovaně kdekoliv v aplikaci. Tyto třídy musí dědit od `AbstractType`. Formulářům lze přiřadit třídu, na kterou se mají jeho hodnoty mapovat. Při vytváření formuláře pomocí metody `createForm` se mu tedy dá do parametrů přidat objekt a po odeslání vyplněného formuláře Symfony automaticky využije setterů (metod nastavujících vlastnosti objektu) dané třídy pro propojení hodnot z formuláře s atributy objektu.

Formulářová komponenta zároveň umožňuje dynamické generování položek formuláře, například na základě dat konkrétního uživatele nebo podle toho, jak uživatel vyplnil jinou část téhož formuláře. Této funkcionality se docílí pomocí napojení naslouchání událostem (Event Listener) na formulář pomocí funkce `addEventListener`, v jejíchž argumentech specifikujeme, o který typ události se jedná a jak se má událost zpracovat.

Této funkcionality jsem využila například při vytváření formuláře na přesun položek mezi seznamy. Požadavkem bylo, aby si uživatel vybral, ze kterého seznamu si přeje položky přesunout. Následně by se mu zobrazily položky tohoto seznamu. Z těch si vybere ty, které jsou určeny k přesunutí a nakonec formulář odešle. Abych docílila tohoto chování formuláře, bylo nutné na výběr nákupního seznamu připojit naslouchání na událost typu `POST_SUBMIT`, díky čemuž formulář zareaguje na vybraný seznam a přidá sadu checkboxů s položkami z tohoto seznamu.

Dále jsem využila naslouchání události typu `PRE_SET_DATA` při generování formuláře umožňujícího uživateli do seznamu pozvat své přátele. Zde bylo nutné, aby se již při vytváření formuláře přidaly jako možnosti pouze ti uživatelé, které má aktuální uživatel v přátelích.

4.3.2 Odeslání formulářů

4.3.2.1 Standardní odesílání

Standardně formuláře odesílají požadavek typu `POST` do stejného controlleru, ve kterém byl formulář vytvořen. Toto chování lze změnit parametrem `action` při vytváření formuláře, kterému předáme název controlleru určeného k přijetí odeslaného formuláře a jeho zpracování. Této možnosti jsem využila především v controlleru obsluhujícím zobrazení specifického nákupního seznamu. Tato stránka obsahuje celkem tři různé formuláře (přidání položky, pozvání přátel a přidání položek z jiného seznamu). Jejich obsluha v jednom controlleru by vedla k dlouhému nečitelnému kódu.

Komponenta `The Form Component` obsahuje metody pro zpracování formulářů. První takovou metodou je `handleRequest`, která rozeznává, jestli byl formulář odeslán, a pokud ano, zapíše odeslaná data do atributů objektu, který je s formulářem spjatý (například při registraci uživatele nastaví zadané uživatelské jméno, heslo apod. nově vytvořenému objektu `User`).

Za metodou `handleRequest` následuje tradičně podmínka **if** ($form−> isSubmitted()&& $form−>isValid()), kde `isSubmitted` vrací `false` pokud formulář nebyl odeslán a `isValid` kontroluje správnost dat v odeslaném formuláři.

4.3.2.2 Odesílání pomocí jQuery AJAX

V některých případech v mé aplikaci nebylo vítané, aby po odeslání formuláře došlo k překreslení celé stránky. Například při přidávání položek k seznamu nebo odškrtávání nakoupených položek bylo vhodné, aby se překreslila pouze část stránky obsahující výčet položek spadajících do daného seznamu.

K tomu bylo potřeba využít odeslání a načtení dat na pozadí. Tuto funkcionalitu nabízí například jQuery metoda `ajax()`. V této metodě se specifikuje adresa url, na kterou se má požadavek odeslat, jeho metoda (`POST\GET\PUT\DELETE`), data a případně jejich typ (JSON atd.).

V metodě controlleru, pod kterou příslušné url spadá, poté proběhlo již zmíněné zpracování hodnot, a pokud to bylo potřeba, jako například u přidávání nové položky, byla vrácena data určená k vykreslení. Pro konverzi dat do formátu vhodného k předávání dat byla použita komponenta Symfony zvaná `Serializer`. Pomocí této komponenty byla data serializována do formátu JSON. Tato data byla následně zpracována v callback funkci ajaxového požadavku `success` pomocí metody `JSON.parse()` a následně se tato data podle potřeby vložila do HTML. Po dobu zpracování formuláře bylo tlačítko k přidání položky vypnuté, aby bylo uživateli jasné, že probíhá akce na pozadí. Zároveň bylo potlačeno standardní odeslání formuláře javascriptovou metodou `preventDefault`

4.3.3 Validace formulářů

4.3.3.1 Validace odeslaných dat

Ve chvíli, kdy se na odeslaném formuláři volá metoda `isValid`, se kontroluje, zda je objekt spojený s tímto formulářem po aplikaci odeslaných dat validní, to znamená, zda data, která byla na atributy namapována, splňují pravidla, která byla pro tyto atributy deklarovaná.

Možností, jak definovat pravidla pro validitu, se nabízí několik. Mohou se definovat v konfiguračních souborech typu YAML nebo XML, pomocí statické PHP funkce v definici třídy nebo pomocí anotací. Já jsem podobně jako u práce s Doctrine zvolila metodu anotací. Ověřování, zda je soubor nahraný k položce nákupního seznamu skutečně obrázek pak probíhá takto:

```
//...
use Symfony\Component\Validator\Constraints as Assert;
//...
class Entry
{
    //...

    /**
     * @ORM\Column(type="string", nullable=true)
     *
     * @Assert\Image (
     *      maxSize="8Mi"
     * )
     */
    private $image;
    //...
```

Zde definuji že nahraný soubor musí být obrázek, jehož maximální velikost je osm mebibytů. Pokud se uživatel pokusí nahrát jiný typ souboru, nebo bude soubor příliš velký, metoda `isValid` vrátí `false`.

4.3.3.2 HTML5 validace

Většina moderních prohlížečů si umí vyžádat splnění jistých validačních požadavků přímo na straně klienta. Obvykle jde například o povinné pole, u kterého prohlížeč informuje uživatele, že před odesláním formuláře toto pole musí vyplnit.

V Symfony se určuje typ pole v definici tohoto pole ve formulářové třídě. Framework obsahuje celé množství těchto typů jako `TextType`, `EmailType` a další. Podle těchto typů se při renderování formuláře v šabloně samy určí příslušné HTML tagy. Podobně je možné nastavit zmíněnou povinnost vyplnění pole pomocí atributu `required`.

Pokud tedy specifikuje typ pole jako `UrlType`, Symfony vykreslí toto pole s atributem `type="url"`, díky čemuž prohlížeč sám validuje data na straně uživatele.

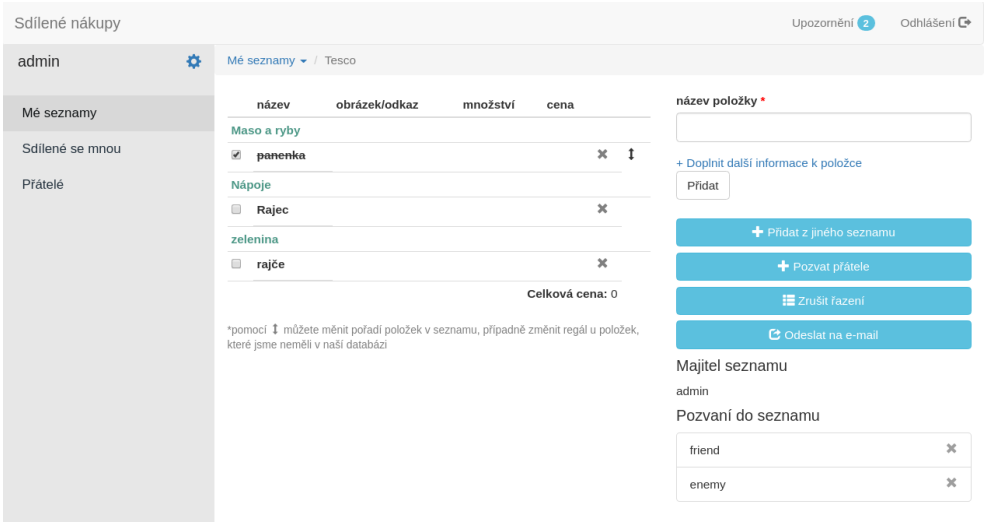
4.4 Řešení uživatelského prostředí

Při tvorbě uživatelského prostředí se musí klást důraz na jednoduchost, přívětivost a intuitivnost. Tyto vlastnosti samozřejmě každý chápe jinak. Co je pro někoho samozřejmé a pochopitelné na první pohled, může někomu jinému připadat zcela neintuitivní. Při realizaci uživatelského rozhraní jsem se tedy inspirovala trendy a zvyklostmi v uživatelských rozhraních moderních aplikací a webových stránek.

Například pro ikony jsem zvolila knihovnu Glyphicon Halflings. Tyto ikony jsou inspirovány AIGA piktogramy a symboly vytvořenými společností Apple jako součást operačního systému OS X. Díky tomu a i proto, že tyto ikony jsou součástí Bootstrapu, s nimi (nebo podobnými ikonami) má zkušenost velká část běžných uživatelů. Většina ikon na stránce je navíc doplněna o text vysvětlující jejich význam.

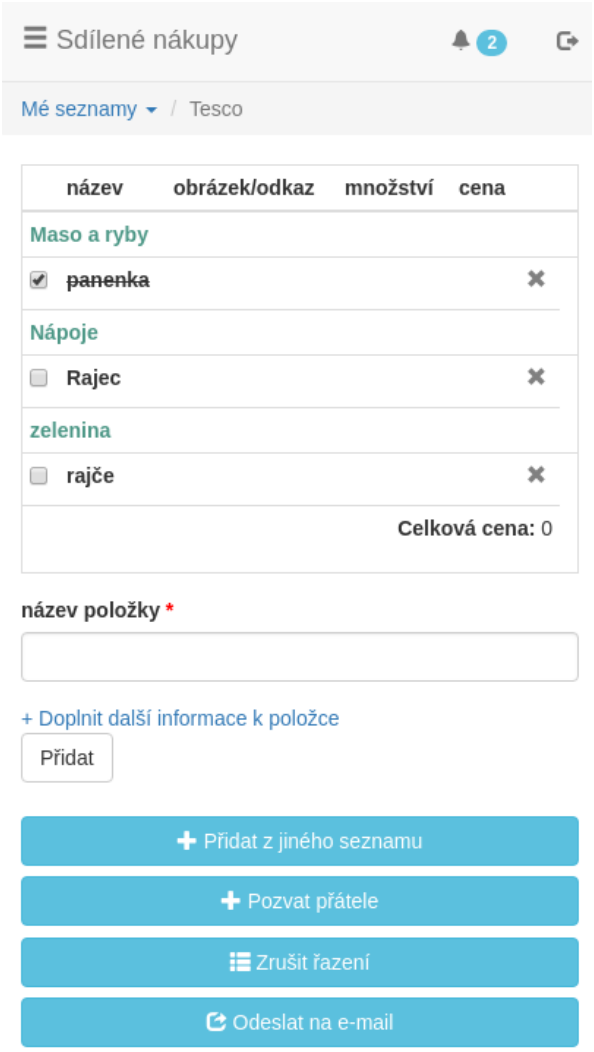
Všechny stránky na webu dědí od `Twig` šablony nazvané `base.html.twig`. Ta definuje základní kostru uživatelského rozhraní, jako horní a postranní menu. Pomocí tagů `block` se pak do této kostry vkládají data z jiných šablon.

Uživatelské rozhraní je zároveň implementováno tak, aby ho bylo možné pohodlně zobrazit na různých typech zařízení bez ohledu na velikost obrazovky. Aplikace je tedy použitelná jak na širokoúhlých monitorech, tak na menších displejích chytrých telefonů. Zde je využit zmiňovaný framework Bootstrap 3. Ten umožňuje rozdělit stránku na několik řádků, které mohou obsahovat různý počet sloupců, které vyplňují tuto řádku v definovaném poměru. Tento poměr lze definovat různě pro různé šířky obrazovky. Zároveň se díky bootstrapovým `hidden` třídám dají jednotlivé prvky na stránce skrýt na různých velikostech okna prohlížeče. Díky tomuto takzvanému grid-systému se prvky na stránce přesunují podle potřeby a dovolují smysluplné zobrazení aplikace nehledě na velikost okna nebo typu zařízení.



Obrázek 4.1: Snímek obrazovky – ukázka uživatelského rozhraní na běžném počítači

Ukázka chování uživatelského rozhraní na zařízeních s různou velikostí obrazovky je znázorněna na obrázcích 4.1 a 4.2.



Obrázek 4.2: Snímek obrazovky – ukázka uživatelského rozhraní na chytrém telefonu

Kapitola 5

Testování

5.1 Jednotkové testování

Jednotkové testy ověřují funkčnost dílčích částí systému. Symfony nabízí k jednotkovému testování knihovnu PHPUnit.

Pro každý test je vytvořena PHP třída, která dědí od `PHPUnit\Framework\TestCase`. Testy se tradičně nacházejí ve složce `\Tests`, jejíž podsložky kopírují adresářovou strukturu testovaných tříd. Například testujeme-li třídu `App\Entity\User`, bude se její test nacházet ve složce `App\Tests\Entity`.

Pomocí PHPUnit jsem testovala třídy reprezentující mnou definované entity v aplikaci a následně repository třídy, které mají za úkol komunikaci s databází.

Aby nedošlo k narušení databáze používané aplikací v případě testů proběhlých s chybou, je nutné vytvořit kopii této databáze a doplnit informace o této databázi do konfiguračního souboru `phpunit.xml.dist`.

Jednotkovými testy bylo odhaleno několik chyb, hlavně v počátku vývoje, které jsem následně odstranila. Například při odstraňování nákupních seznamů z kolekce seznamů uživatele byla původně chybně volána metoda `remove` místo správné metody `removeElement`. V některých případech, kdy mezi sebou měly objekty vztah ManyToMany docházelo při inicializaci těchto vztahů k zacyklení.

V repository třídách byl pak nalezen problém s metodou `getOneOrNullResult`, způsobený nedostatečným nastudováním dokumentace Doctrine. Tato metoda vyhodí výjimku, pokud bude v databázi nalezeno více řádků odpovídajících dotazu, místo toho, aby (jak jsem předpokládala) vrátila první nalezený řádek, případně `null`, nebude-li žádný odpovídající řádek nalezen.

5.2 Testování uživatelského rozhraní v prohlížečích

Aplikace byla, jak jsem již zmínila, navržena pro pohodlné zobrazení na různých typech zařízení. Je ovšem nutné vzít v potaz i správnou funkčnost při zobrazování v různých prohlížečích.

Aplikaci jsem otestovala na následujících prohlížečích:

- Google Chrome 65.0.3325.181
- Chromium 65.0.3325.181
- Mozilla Firefox 59.0.2
- Microsoft Edge 41.16299.15.0

Chromium bylo používáno při vývoji, tudíž na něm aplikace byla testována průběžně a finální testování neodhalilo žádné nedostatky.

V prohlížeči Firefox, docházelo k chybnému odesílání ajaxového požadavku při přidávání položky. Byl způsoben tím, že jQuery funkci `.submit()` nebyla dodávána `event` v argumentech, následující volání `event.preventDefault()` tedy nemělo žádný účinek.

V prohlížeči MS Edge docházelo k chybě při mazání položek v seznamu. Důvodem bylo, že má metoda byla pojmenovaná remove, což je ovšem už existující metoda jQuery. Místo smazání položky byla tedy smazána pouze ikonka ke smazání položky. Tento problém byl vyřešen přejmenováním metody na jméno `removeEntry`.

5.3 Uživatelské testování

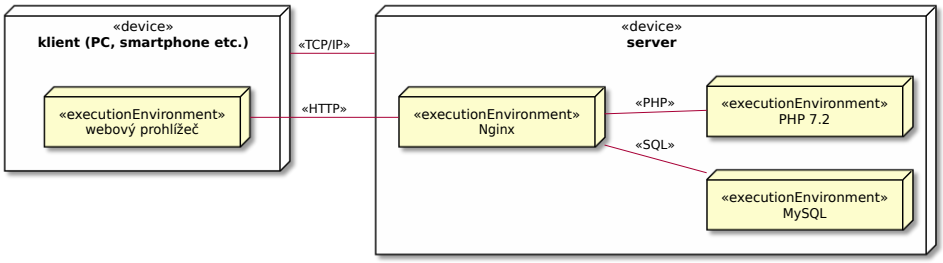
Uživatelské testování proběhlo v rámci již zmíněné semestrální práce v předmětu BI-TUR ve spolupráci s Kryštofem Novotným, který při testování hrál roli pozorovatele.

5.3.1 Nasazení aplikace

Pro demonstrativní účely a uživatelské testování byla aplikace nasazena na webový server s adresou `http://159.89.11.237`. Diagram nasazení najdete na obrázku 5.1.

5.3.2 Testovací scénáře

Testovací scénáře jsme vytvořili tak, aby co nejlépe simulovaly reálné situace, které mohou nastat, a přirozený způsob učení se práci s novou aplikací. Nedávali jsme tedy uživatelům scénář, který by striktně diktoval posloupnost kroků potřebných ke splnění, ale spíše jsme pracovali se scénáři ve formě příběhů „ze života“.



Obrázek 5.1: Diagram nasazení aplikace

Vzhledem k tomu, že aplikace je určena k střídavému používání na počítači a chytrém telefonu, naše scénáře toto zohledňují. Některé scénáře jsou tedy rozděleny na části Doma a Obchod, kde část Doma probíhala na počítači a část Obchod na chytrém telefonu.

5.3.2.1 Scénář 1

Místo 1: Doma

Pořádáte grilování a potřebujete sdílet položky, které je potřeba nakoupit s přáteli. Nakupovat budete v Tescu.

Potřebujete nakoupit tyto položky: chleba, klobása, pivo a následně je nasdílet s uživatelem **sára**.

Místo 2: Obchod

Povedlo se vám nakoupit chleba a pivo. Klobásu jste nesehnali. Označte položky v seznamu jako nakoupené

5.3.2.2 Scénář 2

Potřebujete nakoupit cestou na chatu a vytváříte si nový nákupní seznam. Chleba a klobása se budou hodit, zkopírujte tyto položky z vašeho seznamu, který byl určen na grilování. Nevíte, jestli vám náhodou nedojdou v blízké době mobilní data, radši si tedy, dokud jste na wi-fi, nechte exportovat seznam na e-mail, který se vám automaticky stáhne do telefonu.

5.3.2.3 Scénář 3

Místo 1: Doma

Přátelé potřebují, abyste jim nakoupili některé položky. Přijměte pozvánku k seznamu Albert. Aby byl váš nákup pohodlnější, nechte si tento seznam automaticky setřídit podle regálů. Pokud bylo něco setříděno špatně, nebo se regál nepovedlo určit, zařaďte to správně.

Místo 2: Obchod

Jste v oddělení zeleniny – nakupte všechnu zeleninu, pak jděte do oddělení masa a nakupte všechno maso. V Albertu se vám líbí ještě láhev s alkoholem. Nemáte ale už dost peněz na její nákup. Nasdílejte ji přátelům i s fotkou. Obrázek u položky si pro kontrolu zobrazte.

5.3.3 Testeři

Pro testování jsme měli pětici testerů.

tester 1 Bc. Aneta Peychlová, 23 let, studentka práv na Univerzitě Karlově

tester 2 Ing. Nadia Goulliová, 54 let, účetní

tester 3 Ing. Kateřina Bakošová, 53 let, daňový poradce

tester 4 Ing. Tomáš Sušánka, 26 let, kryptograf

tester 5 Bc. et Bc. Miriam Juranková, 26 let, studentka Pedagogické fakulty Univerzity Karlovy

5.3.4 Výsledky testování

5.3.4.1 Tester 1

Podle očekávání si nejmladší uživatelka s aplikací poradila nejlépe. Déle jí trvalo najít tlačítko sloužící k přesunu položek z jiného seznamu a po registraci si nebyla jista, jestli je nutné registraci potvrdit ještě e-mailem.

5.3.4.2 Tester 2

S druhou testerkou se odhalilo podstatně víc problémů. V prvním scénáři měla potíže s tím, že při pozvání přátel do seznamu nedošlo k žádné odezvě systému, která by jí sdělila, že pozvánka byla odeslána.

Při snaze rozkliknout vytvořený seznam ho omylem smazala, protože křížek určený ke smazání seznamu příliš sváděl ke kliknutí.

Obecně měla problém s tlačítky na doplňující práci se seznamy, jako jsou export na e-mail, sdílení seznamu, přesun položek z jiného seznamu a podobně. Tato tlačítka na stránce přehlížela. Dále jí činilo problémy doplnit specifičtější informace k položce, jako je například obrázek ve třetím scénáři.

Zároveň by jí více vyhovovalo, kdyby se formulář k přidávání položek zobrazoval vlevo a samotný nákupní seznam vpravo. Měla problém s konceptem „přátel“, který jí příliš připomínal sociální sítě, ve které nemá důvěru.

5.3.4.3 Tester 3

Třetí testerka měla hlavní problém s přehlížením `flash` zpráv, které slouží jako odezva po vykonání akce. Tyto zprávy pro ni nebyly dostatečně výrazné, nevěděla tedy, jestli akce jako přidání uživatele do přátel nebo export na e-mail skutečně proběhly.

Další problém opět nastal při rozklikávání vytvořeného seznamu. V tomto případě bylo potřeba kliknout na název seznamu, ale vzhledem k tomu, že výčet seznamů vypadá jako seznam tlačítek, klikala vedle textu na prázdné místo ve zdánlivém tlačítku.

Stejně jako druhá testerka měla problém objevit, kde se k položce přidávají detailní informace.

Nakonec měla potíže se zobrazením seznamů, které s ní byly sdíleny, protože text odkazu v postranním menu zněl „Sdíleno se mnou“ a z tohoto názvu nebylo jasné, že odkazuje na další seznamy.

5.3.4.4 Tester 4

Čtvrtý tester měl opět problém s přehlížením tlačítek pro doplňující práci se seznamy. Najít tlačítko k přesunutí položek z jiného seznamu tak trvalo déle, než by bylo ideální, a nejprve si myslel, že je nutné otevřít starý seznam a odtamtud položky přesunout.

Jinak testování proběhlo bez problémů.

5.3.4.5 Tester 5

Pátá testerka měla problém pouze s pozváním přátel do seznamu. Ve chvíli, kdy uživatel ještě v přátelích nikoho nemá a snaží se do seznamu někoho pozvat, nebylo zjevné, že je nutné nejdřív přátele „získat“.

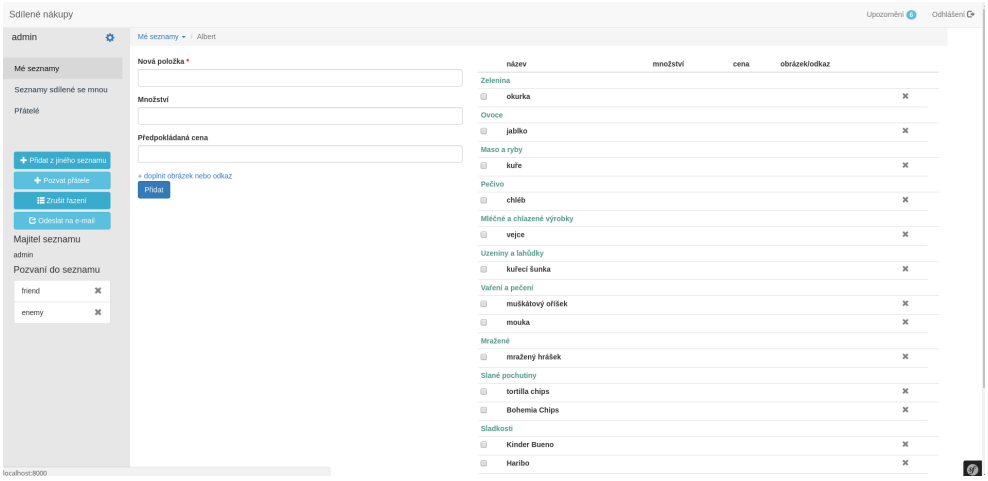
5.3.5 Změny provedené na základě výsledků testování

Na základě zjištěných problematických míst jsem provedla v uživatelském rozhraní několik změn.

Pro lepší odezvu aplikace byly přidány `flash` zprávy tam, kde chyběly, a na jejich zobrazení byly použity výraznější barvy.

Byly prohozeny sloupce pro vytváření nové položky a pro seznam položek tak, aby byl nákupní seznam zobrazován vpravo. Informace o uživatelích v seznamu a ovládací tlačítka pro práci se seznamem byly přesunuty do postranního menu, aby hlavní okno bylo lépe uspořádané. Tlačítka pro práci se seznamem byla střídavě obarvena na různé odstíny modré, aby nepůsobila jako jedna uniformní plocha a uživatel si jich spíše všiml a rozlišil je.

Při vkládání nové položky byly původně formulářové prvky pro doplňující informace skryté a zobrazitelné byly jen při kliknutí na odkaz „+ Přidat další informace k položce“. Po debatě s testery jsme usoudili, že by bylo lepší,



Obrázek 5.2: Snímek obrazovky – ukázka uživatelského rozhraní po změnách na základě uživatelského testování

kdyby se minimálně vložení ceny a množství zobrazovalo rovnou, protože jde o často vkládané informace. Přidání odkazu a obrázku zůstalo skryté pod rozkliknutelným odkazem, ale jeho text byl změněn na „doplnit obrázek nebo odkaz“, aby bylo uživateli jasnější, co se pod odkazem skrývá.

Problém s rozklikáváním vytvořeného seznamu byl vyřešen tím, že na aktivní odkaz bylo převedeno celé tlačítko a nejen jeho text. Dále byl odkaz s křížkem pro smazání seznamu obarven na červeno, aby bylo lépe rozpoznatelné, že jde o potenciálně nechtěnou akci. Před smazáním seznamu se také nyní zobrazí vyskakovací okno s dotazem, jestli chce uživatel seznam opravdu smazat.

V postranním menu byl nahrazen text „Sdíleno se mnou“ na text „Seznamy sdílené se mnou“, aby bylo jasnější, k čemu tento odkaz slouží.

Do formuláře k pozvání přátel do seznamů bylo přidáno sdělení, že je nutné si nejdříve někoho přidat do přátel, než se je snažíte pozvat do seznamu.

Poslední změnou bylo prohození pořadí sloupců v seznamu, kde se nyní zobrazují obrázek a odkaz až po ceně a množství, vzhledem k jejich menšímu využití i důležitosti.

Na obrázku 5.2 lze vidět, jak bylo uživatelské rozhraní z obrázku 4.1 upraveno.

Závěr

Cílem práce bylo vytvořit webovou aplikaci, která by sloužila k vytváření a sdílení nákupních seznamů s dalšími uživateli. Funkční požadavky takové aplikace byly sestaveny na základě analýzy již existujících řešení. Samotnému vytvoření aplikace předcházelo dotazníkové šetření, které stanovilo předpokládanou cílovou skupinu, která bude aplikaci využívat a její potřeby a požadavky.

Aplikace Sdílené nákupy byla vytvořena v souladu s cílem práce tak, aby obsahovala všechny stanovené funkční i nefunkční požadavky. Pro implementaci byl využit framework Symfony ve verzi 3.4 spolu s ORM Doctrine. Webová aplikace Sdílené nákupy byla implementována a následně otestována. Na základě výsledků uživatelského testování byla implementace upravena v závislosti na nalezených nedostatcích.

Při tvorbě a testování vlastní aplikace byly identifikovány další funkce, které by bylo vhodné doplnit. Tyto další možnosti přesahují rámec této práce a mohou být implementovány v budoucnosti. Mimo jiné se objevily tyto nejvýraznější návrhy nových funkcí:

- nečekaná funkce reverzního seznamu věcí, které jsou k dispozici a není třeba je opatřovat
- možnost připojení k cenovým databázím a porovnání celkové ceny nákupu v různých obchodech
- jednorázové sdílení nákupního seznamu s neregistrovanými uživateli např. pomocí QR kódu

Literatura

[1] Haischt, D. S.: A traditional, stateless web application and it's ajaxified counterpart on the right site [online]. [cit. 18.4.2018]. Dostupné z: `https://commons.wikimedia.org/wiki/File:Ajax-vergleich-en.svg`

[2] Alias Elbert aj.: Identify technology on websites [online]. [cit. 11.12.2017]. Dostupné z: `https://www.wappalyzer.com`

[3] GmbH, B. I.: Out of Milk [online]. [cit. 11.12.2017]. Dostupné z: `https://outofmilk.com`

[4] TOMIKUP.CZ: FAQ / TOMIKUP otázky a odpovědi [online]. [cit. 11.12.2017]. Dostupné z: `https://www.tomikup.cz/stat/faq`

[5] Wunderlist [online]. [cit. 11.12.2017]. Dostupné z: `https://www.wunderlist.com/`

[6] Summerfield, M.: *Python 3*. CPress, první vydání, 2013, ISBN 978-80-251-2737-7.

[7] The PHP Group: PHP: Preface - Manual [online]. [cit. 25.3.2018]. Dostupné z: `http://php.net/manual/en/preface.php`

[8] Q-Success: Usage of server-side programming languages for websites [online]. [cit. 25.3.2018]. Dostupné z: `https://w3techs.com/technologies/overview/programming_language/all`

[9] Django Software Foundation: Django - The web framework for perfectionists with deadlines. [online]. [cit. 11.12.2017]. Dostupné z: `https://www.djangoproject.com/`

[10] Zeman, M.: Výsledky: Technologie na českém webu [online]. [cit. 11.12.2017]. Dostupné z: `https://www.zdrojak.cz/clanky/vysledky-technologie-na-ceskem-webu/`

[11] Nette Foundation: Nette Framework [online]. [cit. 11.12.2017]. Dostupné z: https://nette.org/cs/

[12] SensioLabs: About Symfony Project [online]. [cit. 11.12.2017]. Dostupné z: https://symfony.com/about

[13] SensioLabs: Creating Symfony applications [online]. [cit. 11.12.2017]. Dostupné z: https://symfony.com/download

[14] Doctrine [online]. [cit. 18.4.2018]. Dostupné z: https://www.doctrine-project.org/about/

[15] SensioLabs: Twig - The flexible, fast, and secure PHP template engine [online]. [cit. 18.4.2018]. Dostupné z: https://twig.symfony.com/

[16] The jQuery Foundation: jQuery [online]. [cit. 18.4.2018]. Dostupné z: https://jquery.com/

[17] Q-Success: Usage of JavaScript libraries for websites [online]. [cit. 18.4.2018]. Dostupné z: https://w3techs.com/technologies/overview/javascript_library/all

[18] The jQuery Foundation: jQuery UI [online]. [cit. 18.4.2018]. Dostupné z: http://jqueryui.com/

[19] Harding, J.; Skarich, V.; Trueman, T.: typeahead.js [online]. [cit. 18.4.2018]. Dostupné z: https://twitter.github.io/typeahead.js/

[20] Otto, M.; fat: Bootstrap [online]. [cit. 18.4.2018]. Dostupné z: https://getbootstrap.com/

Příloha **A**

Seznam použitých zkratek

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

XML Extensible Markup Language

YAML YAML Ain't Markup Language

DocBlock formátovaný komentář zdrojového kódu, dokumentující specifický segment kódu

ORM Objektově relační mapování

MVC Model-view-controller

AJAX asynchronní JavaScript a XML

Příloha B

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `screenshots` ..... screenshoty výsledné aplikace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `install.txt` ..... instalační příručka
  - `TUR.csv` ..... výsledky dotazníku z předmětu BI-TUR
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF