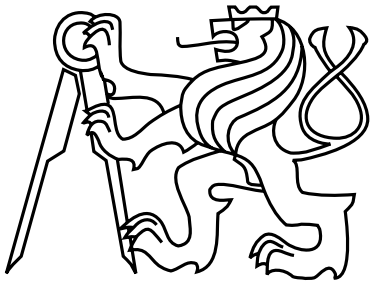
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

ShoppingShare - Aplikace pro sdílení nákupních seznamů

Marek Čech

Vedoucí práce: Ing. Ondřej Macek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

21. května 2014

Poděkování

Rád bych poděkoval vedoucímu práce za dobré rady a vstřícnost při konzultacích a také lidem, kteří byli ochotni aplikaci otestovat a sdělit cenné uživatelské zkušenosti.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 15. 5. 2014 .....

Abstract

The target of the thesis is to implement the system that enables to make shopping lists and to share them. The system consists of the server application with graphical user and REST interface and Android system application consuming the interface.
The main contribution of the thesis is based on the definition and solving problems that smartphone software developers could face. Above all, I want to highlight the problem of choosing technology, the method and security of the data transfer and implementation of modern elements of Android graphical user interface. I would also like to emphasize testing of smartphone applications.

Abstrakt

Cílem práce je implementovat systém, který umožní tvorbu a sdílení nákupů. Systém je složen ze serverové aplikace s webovým a REST rozhraním a aplikace pro systém Android toto rozhraní konzumující.
Hlavní přínos práce spočívá v definici a řešení problémů, které vývojáře mobilních aplikací mohou potkat. Především chci vyzdvihnout problém výběru technologií, dále způsob a zabezpečení přenosu dat, implementaci moderních prvků uživatelského rozhraní systému Android a nakonec testování mobilních aplikací.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

ShoppingShare je systém sloužící ke zjednodušení skupinových nákupů a nákupů obecně. Systém má odstranit tvoření obsahu nákupů seznamy v papírové podobě. Papírový seznam položek nákupu má několik nevýhod. Především má k němu přístup pouze člověk, který ho má u sebe. Systém ShoppingShare nabízí elektronickou alternativu s přístupem odkudkoli a kdykoli přes mobilní aplikaci a webové rozhraní.

V návrhu systému je kladen důraz na sociální kontakt. Nákupy jsou vytvářeny v rámci skupin, do kterých se uživatel dostane na základě pozvánky či schválené žádosti členem skupiny. Dění v rámci skupiny je bezpečné, mohou jej sledovat jen její členové.

Všichni členové mají stejná práva. Mohou vytvářet, editovat či mazat nákupy a jejich položky, vyřizovat žádosti o členství či uživatelům členství ve skupině nabízet.

Uživatel má možnost předpřipravit si nákupy pomocí šablon. Tyto šablony může vytvářet, editovat či mazat a jednoduše použít pro vytvoření nákupu.

1.1 Cíle

Cílem práce je vytvořit dobře otestovaný, robustní a do budoucna lehce rozšiřitelný systém, jehož implementovaná funkčnost odpovídá návržené množině popsané v tomto dokumentu.

Projekt se skládá z webové a mobilní aplikace. Webová aplikace představuje backend pro webové a REST rozhraní. Součástí systému je mobilní aplikace REST rozhraní konzumující. Vybranou platformou je z důvodu široké uživatelské základny systém Android.

Důraz je kladen na zabezpečení dat při skladování i přenosu a na synchronizační problémy vyplývající z možnosti přístupu do systému přes dvě rozhraní. Dalším důležitým faktorem je úspora dat při komunikaci mezi serverem a mobilní aplikací.

Součástí práce je vytvoření obou uživatelských rozhraní s identickou funkčností.

1.2 Motivace

Motivací k vytvoření systému ShoppingShare byl vypozorovaný volný prostor v oblasti aplikací sloužících ke zjednodušení nakupování. Další neméně podstatnou motivaci či výzvu

vidím ve složitosti systému. Webová a mobilní aplikace představují dvě rozhraní, které uživatel může využívat. Z tohoto důvodu lze očekávat širokou škálu použitých technologií.

1.3 Obsah dokumentu

Ve druhé kapitole je podrobně popsána funkčnost systému, a to včetně vypovídajících diagramů případů užití. Třetí kapitola je věnována implementaci, důležitou součástí je zhodnocení na projektu použitých technologií. Čtvrtá kapitola se zabývá testováním obou aplikací. Závěr obsahuje komparaci cílů naplánovaných a dosažených a náčrt budoucího směřování projektu.

Kapitola 2

Analýza

Kapitola podrobně popisuje analýzu systému ShoppingShare. Najdeme zde popis doménového modelu a funkčnosti systému.

2.1 Doménový model

V této části jsou podrobně popsány entity vyskytující se v systému. Entity společně tvoří doménový model. Entita představuje objekt z reálného světa a stejně jako v něm má vztahy na další objekty.

2.1.1 Uživatel

Uživatel je základní entitou systému. Uživatelem je myšlena osoba využívající služeb systému, má uživatelské jméno a heslo. Podobně jako osoba z reálného světa má uživatel jméno a příjmení. Uživatel patří do několika skupin (viz 2.1.2), má několik žádostí o členství (viz 2.1.3) a tvoří šablony nákupů (viz 2.1.6).

2.1.2 Skupina

Skupina sdružuje uživatele do celku. Má své jméno a datum vytvoření. Ke skupině je přidruženo několik nabídek či žádostí o členství (viz 2.1.3). V rámci skupiny jsou vytvářeny a prováděny nákupy (viz 2.1.4).

2.1.3 Žádost o členství

Žádost je členství je prostředkem, aby se uživatel stal členem skupiny. Žádost má dva typy (viz 2.1.10) a může se nacházet ve třech možných stavech (viz 2.1.9). K žádosti je vždy přidružen jeden uživatel a skupina.

2.1.4 Nákup

Entita nákup představuje nákup známý z reálného světa. Jako takový má svoje položky (viz 2.1.5) a jméno. Nákup má dále datum poslední úpravy a stav (viz 2.1.7), ve kterém se nachází. ShoppingShare je systém pro sdílení nákupů, každý nákup tedy patří k nějaké skupině.

2.1.5 Položka nákupu

Položka nákupu je nejatomárnější entitou systému. Každá položka patří buď nákupu, nebo šabloně nákupu (viz 2.1.6). Pokud položka patří k šabloně nákupu, jedná se vlastně o šablonu položky. I v této situaci je však považována za klasickou položku, a to z důvodu operací nad všemi položkami v systému. Položka nákupu má dále datum poslední změny a stav (viz 2.1.8), ve kterém se nachází.

2.1.6 Šablona nákupu

Uživatel má možnost tvořit a spravovat šablony nákupů. Motivací k tomu je urychlení tvorby často se opakujících nákupů. Šablona obsahuje několik položek, má svoje jméno, datum poslední změny a stav (viz 2.1.7), ve kterém se nachází.

2.1.7 Stav nákupu

Stav nákupu popisuje stav, ve kterém se nachází nákup nebo šablona nákupu. Nákup se může vyskytovat ve stavu „nový“, či „smazaný“.

2.1.8 Stav položky

Stav položky popisuje stav, ve kterém se nachází položka nákupu. Položka se může vyskytovat ve stavu „nová“, „nakoupená“, či „smazaná“.

2.1.9 Stav žádosti o členství

Stav žádosti o členství popisuje stav, ve kterém se nachází žádost, či nabídka ke členství. Žádost o členství se může nacházet ve stavu „nevyřízená“, „přijatá“, nebo „odmítnutá“.

2.1.10 Typ žádosti o členství

Typ žádosti podrobněji definuje žádost o členství. Zažádat o členství může sám uživatel nebo mu může být členství nabídnuto členem skupiny. Existují tedy typy „nabídka“ a „žádost“.

2.2 Funkčnost

ShoppingShare je systém sloužící ke zjednodušení skupinových nákupů a nákupů obecně. Uživatelé se sdružují do skupin definovaných v rámci doménového modelu. členové skupiny společně tvoří nákupy. Uživatel si má možnost předpřipravit nákupy pomocí šablon.

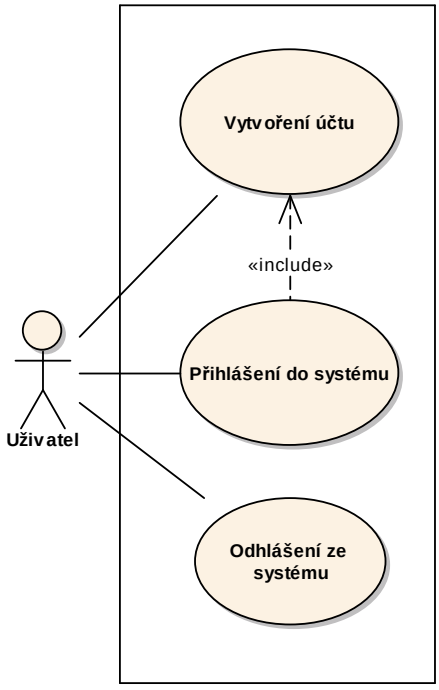
Webová i mobilní aplikace nabízejí dle zadání stejnou funkčnost. Rozdělil jsem ji na kategorie autentizace, sociální kontakt uživatelů, šablony nákupů a nakupování. K názornému popisu přispívají diagramy případů užití.

2.2.1 Uživatel systému

Uživatelem systému je myšlen člověk využívající systém ShoppingShare. Uživatelem se může stát jakýkoli člověk. K vytvoření účtu je vyžadováno připojení k internetu. Uživatel se může nacházet ve stavu příhlášený, nebo nepřihlášený.

2.2.2 Autentizace

Uživatel se může do systému přihlásit či z něj odhlásit. Pro přihlášení je nutné mít vytvořen účet.



Obrázek 2.1: Autentizace uživatele

2.2.3 Sociální kontakt uživatelů

Uživatel může mít členství ve více skupinách definovaných v rámci doménového modelu. Členem skupiny se stane, pokud skupinu sám vytvoří, stávající člen jej do ní pozve, nebo podá žádost o členství. Při vytváření skupiny má uživatel možnost hromadně odeslat nabídky ke členství jiným uživatelům systému. Pozvat uživatele může člen skupiny i kdykoli později.

Jakoukoli nabídku či žádost je možné potvrdit, nebo zamítnout. Žádosti o členství ve skupině může vyřídit jakýkoli člen skupiny, všichni členové skupiny mají stejná práva.

Obrázek 2.2: Sociální kontakt uživatelů

2.2.4 Šablony nákupů

Uživatel má možnost si předpřipravit často se opakující nákupy jako šablony definované v rámci doménového modelu . Šablonu lze, podobně jako nákup, vytvořit, upravovat a smazat. Stejné je to s položkami šablon.

Šablonu lze použít k vytvoření nového nákupu, což je popsáno v sekci věnované nakupování (viz 2.2.5).

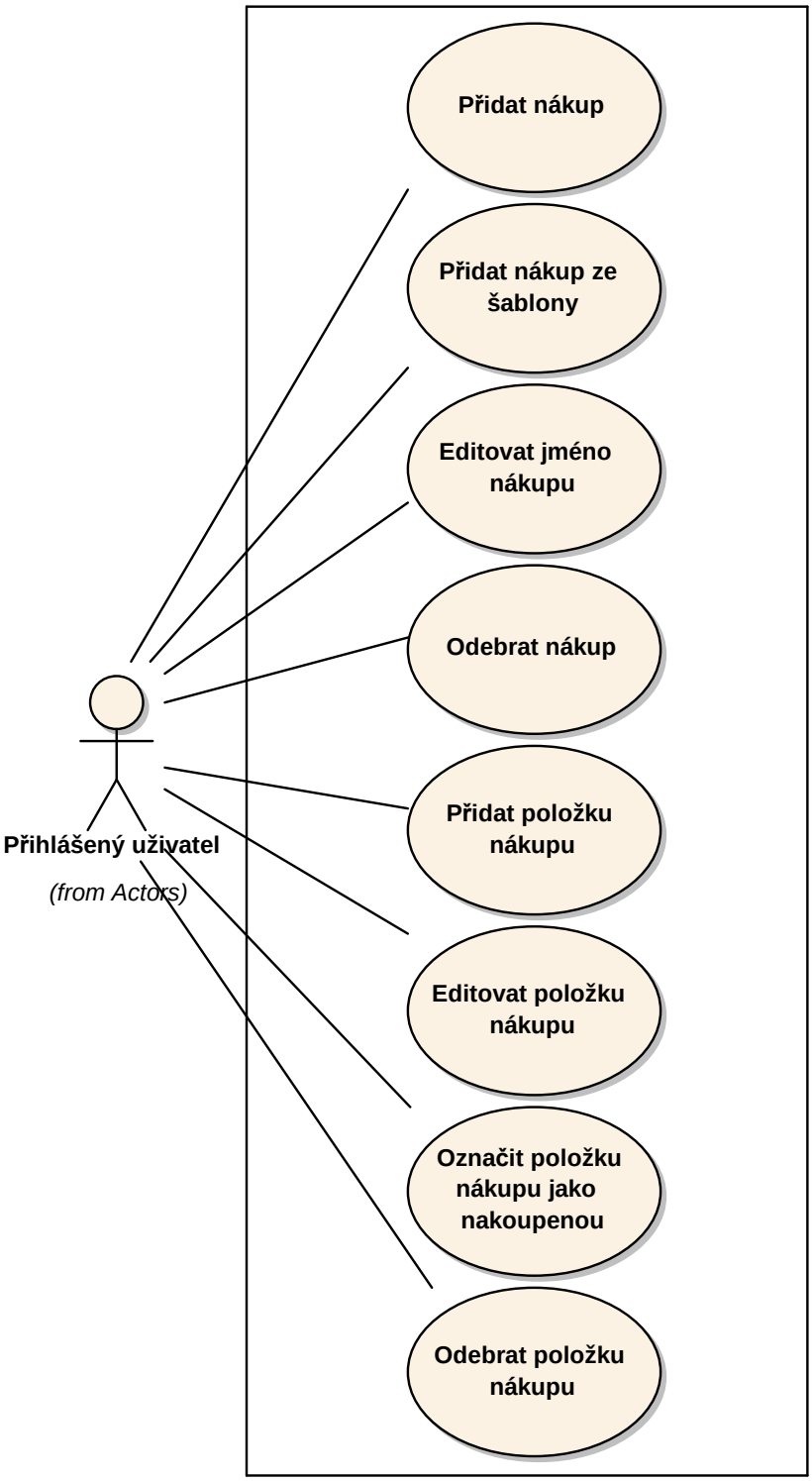
Diagram se nachází v příloze A.

2.2.5 Nakupování

Členové skupiny společně tvoří nákupy do nichž vkládají položky. Položky nákupu lze přidávat, upravovat a mazat stejně jako nákupy, do nichž patří. Každý člen skupiny je rovnocenný, má stejná práva k akcím ve skupině.

Položka nákupu je po vytvoření ve stavu „nová“. Po fyzickém nakoupení položky je vhodné položku označit, čímž se změní její stav na „nakoupená“.

Nákup lze navíc vytvořit ze šablony, definované v rámci doménového modelu a v rámci popisu funkčnosti (viz 2.2.4). Použitím šablony se do nově vytvořeného nákupu překopírují položky definované šablonou.

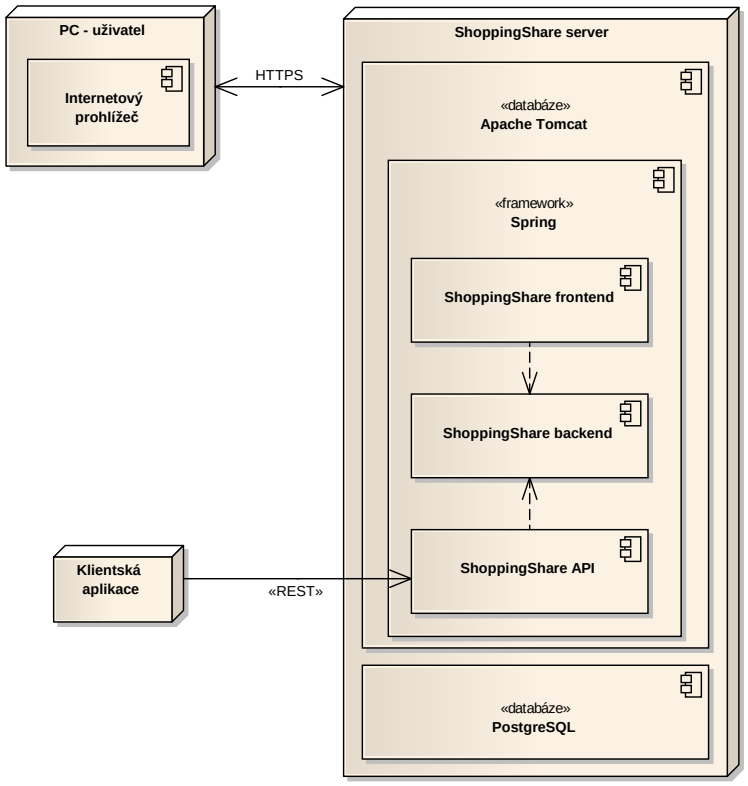


Obrázek 2.3: Funkčnost týkající se nakupování

Kapitola 3

Implementace

Implementace probíhá v oddělených projektech na serverovou část a mobilní aplikaci. Náhled na celý systém nabízí následující diagram nasazení. Hlavní částí systému je server, na kterém v kontejneru běží serverová aplikace a databáze. Serverová aplikace poskytuje rozhraní pro klientskou mobilní aplikaci a rozhraní webové. Veškerá komunikace je zabezpečena pomocí HTTPS.



Obrázek 3.1: Diagram nasazení popisující projekt ve vysokém náhledu

3.1 Webová aplikace

Následuje seznam použitých technologií při implementaci webové aplikace. Technologií je myšlen framework či knihovna.

3.1.1 Použité technologie

Výběr technologií pro serverovou aplikaci proběhl na základě předchozích zkušeností. Celá aplikace je zastřešena frameworkem Spring umožňujícím jednoduchou a rychlou tvorbu REST rozhraní, business logiky a přístupu k persistentní vrstvě.

J2EE framework

Spring[9] 3.2.3 – Populární aplikační framework pro vývoj J2EE aplikací. Tvoří robustní základ aplikací tvořených v jazyce Java. Největší konkurent, Enterprise Java Beans firmy Oracle, se frameworkem Spring inspiruje.

Databáze

Hibernate[8] 4.1.10 – Implementace Java Persistence API definované firmou Oracle. Slouží k objektově – relačnímu mapování. Hibernate umožňuje nezávislost tvořeného kódu na výrobci databáze.

PostgreSQL[7] 9.1 – Databáze využívaná pro vývoj na lokálním stroji i v produkčním prostředí. Pro svou jednoduchost a kvalitu je jednou z nejvíce používaných databází.

Testování

JUnit[6] 4.10 – Framework pro psaní unit testů v jazyce Java. Jedná se o nepsaný standart.

gmock[3] 0.8.3 – Mockovací framework pro psaní testů v jazyce Groovy.

Data

Jackson Data Mapper[5] 1.9.13 - Knihovna pro mapování objektů na JSON a naopak.

UI

Bootstrap[11] 3.0.0 – Framework pro rychlou tvorbu GUI.

3.1.2 Architektura

Spring společně se Spring MVC jsou frameworky navržené pro tvorbu aplikací s třívrstvou architekturou. Filozofie vychází z návrhového vzoru MVC[1], který definuje vrstvu představující rozhraní, vrstvu pracující s daty a vrstvu tvořící logiku, umístěnou mezi předešlými dvěmi vrstvami.

Jednotlivé vrstvy aplikace založené na frameworku Spring jsou mezi sebou propojeny přes rozhraní. Výhodou programování přes rozhraní je snadná výměna implementace vrstvy.

View vrsta

Vrstva tvořící rozhraní aplikace je implementována pomocí controllerů a souborů JSP[2]. Controllery jsou třídy přijímající dotazy na server. Na základě dat získaných od service vrstvy je vytvořen výstup, což je v systému ShoppingShare vygenerovaný HTML kód pomocí JSP, nebo JSON.

Service vrsta

Service vrstva tvoří logiku aplikace. Zpracovávaná data přicházejí ve webové aplikaci ShoppingShare z view vrstvy nebo z vrstvy databázové.

Databázová vrstva

Databázová vrstva je jediné místo aplikace s možností komunikace s databází. Je tvořena DAO[3] třídami a třídami reprezentujícími entity v rámci doménového modelu. Třídy doménového modelu jsou pomocí objektově relačního mapování namapovány na tabulky databáze. Objektově relační mapování je v režii frameworku Hibernate. Výhodou této technologie je nezávislost na použité databázi, která může být vyměněna dokonce za běhu aplikace.

[1] model-view-controller
[2] JavaServer Pages
[3] data access object

3.1.3 Rozhraní serveru

REST je datově orientovaná architektura sloužící ke vzdálenému přístupu k datům na základě jejich adresy. REST použitý v systému ShoppingShare je modifikovaný. REST definovaný tvůrcem používá ID jako součást adresy. V ShoppingShare jsou ID posílána zašifrována v těle dotazů. Rozhraní je využíváno mobilní klientskou aplikací i webovým rozhraním. REST rozhraní serverové aplikace popisuje následující tabulka.

Adresa	**HTTP metoda**	**Popis**
/group	PUT	Na základě formuláře vytvoří skupinu a vrátí její ID.
/group/all	POST	Vrátí seznam skupin, kterých je uživatel členem od data poslední synchronizace.
/group/web/all	GET	Vrátí seznam skupin, kterých je uživatel členem.
/group/startsWith	POST	Vrátí seznam skupin, kterým název začíná dle parametru.
/item	DELETE	Vymaže položku nákupu dle ID v parametru.
/item	PUT	Na základě formuláře vytvoří položku nákupu a vrátí její ID.
/item	POST	Na základě formuláře edituje položku.
/item/all	DELETE	Smaže položky dle seznamu ID v parametru.
/item/all	POST	Vrátí položky nákupu dle jeho ID v parametru.
/item/all/bought	POST	Označí položky jako nakoupené dle seznamu ID v parametru.
/item/web	POST	Na základě formuláře edituje položku.
/item/template	PUT	Na základě formuláře vytvoří položku šablony.
/item/template/all	POST	Vrátí položky šablony dle ID šablony v parametru.
/membershiprequest/accept	POST	Změní stav žádosti o členství na přijatá dle ID v parametru.
/membershiprequest/reject	POST	Změní stav žádosti o členství na odmítnutá dle ID v parametru.
/membershipoffer/all	POST	Vrátí nabídky ke členství uživatele vytvořené po datu v parametru.
/membershiprequest	PUT	Vytvoří novou žádost o členství dle ID skupiny v parametru.
/membershipoffer	GET	Vrátí nabídky ke členství uživatele.

Tabulka 3.1: Rozhraní serveru

Adresa	HTTP metoda	Popis
/membershipoffer	PUT	Na základě formuláře pozve uživatele do skupiny.
/membershiprequest/all	POST	Vrátí žádosti ke členství skupiny dle ID skupiny v parametru.
/membershipoffer/web/all	POST	Vrátí nabídky ke členství skupiny dle ID skupiny v parametru.
/offers	GET	Vrátí nabídky ke členství uživatele.
/purchase/all	POST	Vrátí nákupy skupiny dle ID v parametru.
/purchase/news	POST	Na základě formuláře vrátí novinky v nákupech.
/purchase	PUT	Na základě formuláře vytvoří nový nákup.
/purchase	POST	Na základě formuláře edituje nákup.
/purchase	DELETE	Smaže nákup na základě jeho ID v parametru.
/purchase/web	POST	Na základě formuláře edituje nákup.
/purchase/fromtemplate	PUT	Vytvoří nákup na základě ID šablony v parametru.
/purchasetemplate/news	POST	Vrátí novinky v šablonách na základě data poslední synchronizace v parametru.
/purchasetemplate	PUT	Vytvoří šablonu z existujícího nákupu/šablony.
/purchasetemplate/form	PUT	Na základě formuláře vytvoří šablonu.
/purchasetemplate/web/form	PUT	Na základě formuláře vytvoří šablonu.
/purchasetemplate	DELETE	Smaže šablonu na základě ID v parametru.
/purchasetemplate	POST	Na základě formuláře edituje šablonu.
/purchasetemplate/web	POST	Na základě formuláře edituje šablonu.
/purchasetemplate	GET	Vrátí všechny šablony uživatele.
/signup	POST	Na základě formuláře vytvoří účet a přihlásí uživatele.
/login	GET	Přihlásí uživatele webové aplikace.
/login/basic	GET	Přihlásí uživatele mobilní aplikace.
/logout	GET	Odhlásí uživatele.
/members	POST	Na základě formuláře vrátí uživatele, žádosti a nabídky ke členství skupiny.
/members/web	POST	Vrátí uživatele dle ID skupiny v parametru.
/user/startsWith	POST	Vrátí seznam uživatelů, kterým jméno začíná dle parametru.

Tabulka 3.2: Rozhraní serveru - pokračování

3.2 Mobilní aplikace

3.2.1 Použité technologie

Buildovací nástroj

Gradle[4] 0.7.1 – Nástroj pro správu, řízení a automatizaci buildů. Největším konkurentem je Maven, oproti němu však má kratší syntaxi. Zachovává jeho největší výhodu, využívá Maven repozitář.

UI

ActionBar-PullToRefresh[1] 0.9.3 - Knihovna od chrisbanes usnadňující implementaci pull-to-refresh návrhového vzoru. V době implementace se jednalo prakticky o jediné řešení.

Spojení se serverem

Spring for Android[10] 1.0.1 - Framework z rodiny Spring umožňující konzumovat REST API. I přes některé problémy popsané v následující kapitole se jedná o kvalitní řešení komunikace se serverovou aplikací.

Data

Jackson Databind[2] 2.1.2 - Knihovna pro mapování objektů na JSON a naopak.

3.2.2 Význačné prvky implementace

Problémy s technologiemi se týkaly pouze technologií pro mobilní aplikaci. Výběr technologií na platformě Android byl pro mne novinkou. Především je překvapující, že výrobce tohoto systému nenabízí knihovny k usnadnění implementace častých případů užití. Mám na mysli především knihovny pro spojení se serverem, například přes REST rozhraní, a knihovny pro implementaci pull-to-refresh GUI[4] návrhového vzoru.

Spojení se serverem

Pro spojení se serverovou aplikací jsem použil framework Spring for Android 3.2.1, dalšího člena obsáhlé rodiny Spring. O Spring for Android je možné mluvit jako o mladém frameworku, přesto jsem se rozhodl ho použít, a to z důvodu slibu kvality garantovaného značkou Spring.

Ve frameworku jsem objevil několik nedokonalostí. Nedokáže, aby HTTP dotaz typu DELETE obsahoval v těle data. ID entit jsem nechtěl používat jako součást URL, jsou tedy posílané zašifrované v těle dotazu.

Tento nedostatek jsem vyřešil[5] poděděním třídy HttpComponentsClientHttpRequestFactory[6], kdy ve chvíli vytváření dotazu typu DELETE vytvoří vlastní instanci reprezentující

[4] grafické uživatelské rozhraní

[5] Třída s řešením se jmenuje: com.cvut.shoppingshare.rest.ShoppingShareHttpComponentsClientHttpRequestFactory

[6] Celé jméno třídy je: org.springframework.http.client.HttpComponentsClientHttpRequestFactory

dotaz. Vlastní instance již nabízí možnost obsahu těla dotazu.

Dalším problémem Spring for Android je absence HTTP dotazu typu PATCH. Absence prvku množiny HTTP metod představuje problém s logickým namapováním funkčnosti serveru na HTTP metody. Metoda PATCH slouží k částečné aktualizaci záznamu v databázi, tedy například editace nákupu nebo položky. Problém jsem vyřešil přemapováním na metodu POST.

SSL certifikát

Na další problém jsem narazil při snaze použít vlastnoručně podepsaný SSL certifikát. Prezence certifikátu je nezbytná pro vytvoření šifrovaného spojení mezi klientem a serverem. Při startu projektu však nemá smysl pořizovat certifikát podepsaný autoritou, který spotřebovává finanční zdroje.

Vlastnoručně podepsaným certifikátům se Android brání z důvodu bezpečnosti. Systém při snaze komunikovat vyhodí výjimku[7]. Zdrojový kód řešení je v příloze B.1.

Implementace pull-to-refresh návrhového vzoru

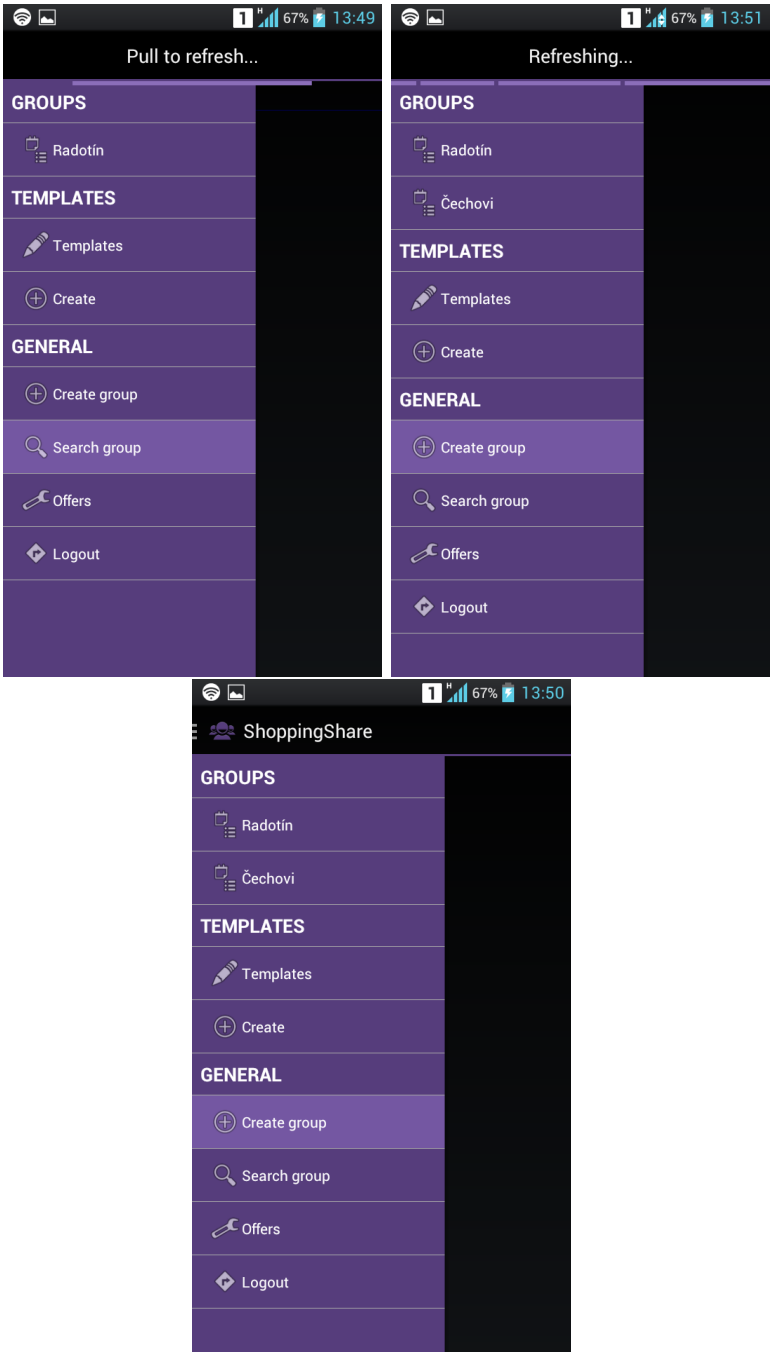
Pull-to-refresh je jedním ze základních gest používaných k obsluze zařízení využívajících systém Android. Tvůrce tohoto systému však nenabízí knihovny pro usnadnění jeho implementace. Vývojář tedy může funkčnost napsat sám, nebo použít knihovny napsané jiným vývojářem.

Knihovna ActionBar-PullToRefresh ve verzi 0.9.3 obsahuje několik zásadních problémů. Především se jedná o množství vyhazovaných chyb[8] zjištěných během testování.

Účel knihovny je spustit a vizualizovat aktualizaci dat, ukončení z důvodu vyhozené výjimky tedy není zásadním problémem. Uživatel v takovém případě znovu provede pull-to-refresh gesto. O nalezených chybách jsem informoval tvůrce knihovny.

Na následujícím obrázku je vidět implementace pull-to-refresh návrhového vzoru. První obrazovka ukazuje děj ve chvíli, kdy uživatel táhne prstem po obrazovce dolů. Na druhé obrazovce probíhá synchronizace dat se serverem. Třetí obrazovka ukazuje, že uživateli byla kladně vyřízena žádost o členství ve skupině a stal se jejím členem.

[7] Znění výjimky je: javax.net.ssl.SSLException: Not trusted server certificate exception
[8] Znění výjimky je: java.lang.NullPointerException



Obrázek 3.2: Implementovaný pull-to-refresh návrhový vzor

3.2.3 Aktivity a fragmenty

Aktivita je komponenta aplikace pro systém Android zajišťující obrazovku pro interakci s uživatelem. Fragment je součástí aktivity, přičemž aktivita může mít více fragmentů. Tím je zajištěno rozdělení implementace funkčnosti na ucelené celky. Příkladem může být obrazovka rozdělená na několik záložek. Každá záložka je reprezentována svým fragmentem, celou obrazovku zaštiťuje aktivita.

Aplikaci je možné vytvořit, a do verze Androidu 4.0 tomu tak bylo, jen pomocí aktivit. Nevýhodou tohoto přístupu je pomalejší vykreslování jednotlivých obrazovek a ztráta logického členění kódu aplikace.

Mobilní aplikace ShoppingShare je rozdělena na tři aktivity, které popisuje následující tabulka.

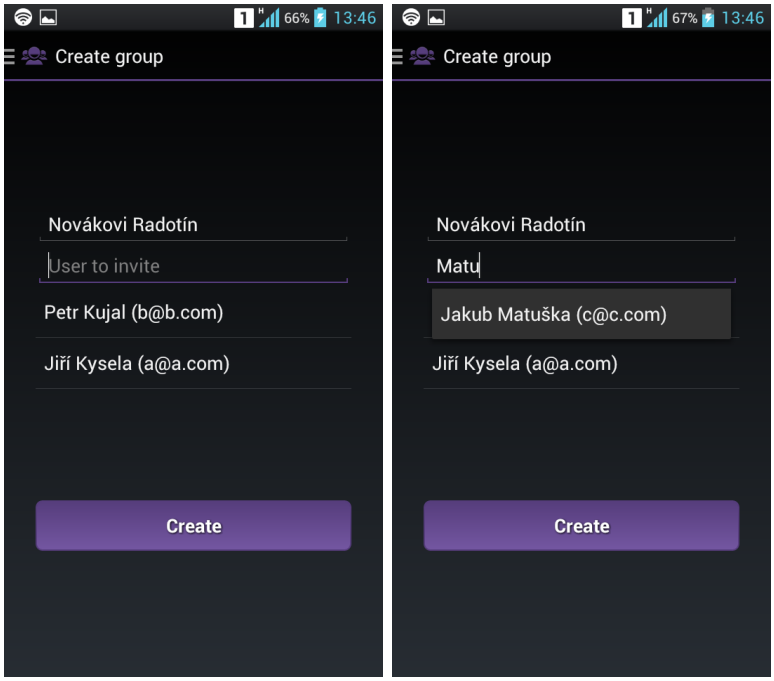
Název	**Popis**
LoginActivity	Aktivita sloužící k přihlášení.
SignUpActivity	Aktivita sloužící k vytvoření účtu.
ShoppingShareMainActivity	Hlavní aktitiva aplikace, obsahuje všechny fragmenty aplikace.

Tabulka 3.3: Seznam aktivit mobilní aplikace

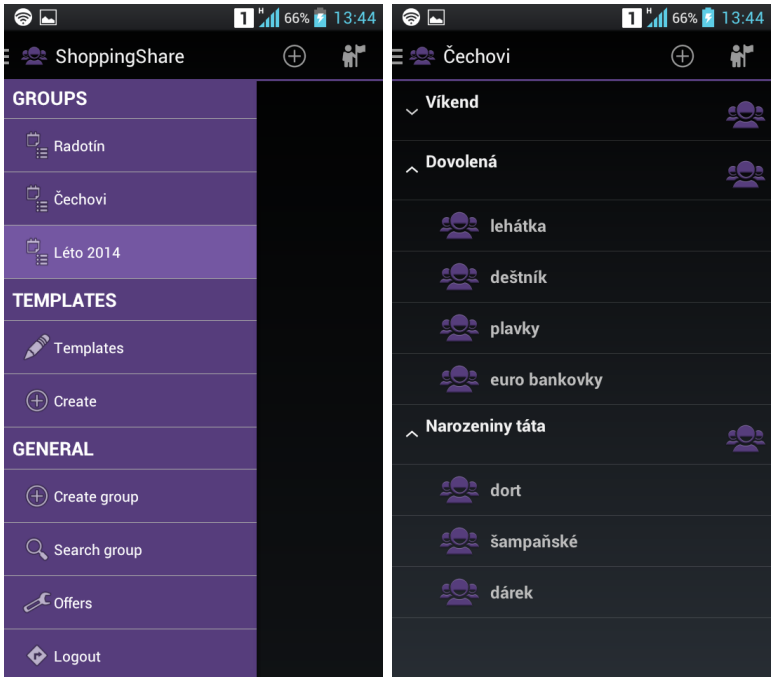
Aktivita ShoppingShareMainActivity obsahuje všechny fragmenty aplikace. Jejich seznam a popis funkčnosti obsahuje následující tabulka.

Název	**Popis**
AddGroupFragment	Fragment sloužící k vytvoření skupiny. Zajímavostí tohoto fragmentu je našeptávač jmen uživatelů, které chceme do tvořené skupiny pozvat.
AddTemplateFragment	Fragment sloužící k vytvoření šablony nákupu.
GroupMembersFragment	Fragment obsahující 3 záložky. První obsahuje členy skupiny, druhá žádosti o členství a třetí odeslané nabídky ke členství skupiny. Jsou zde možnosti jako přijímat/odmítat žádosti nebo vytvářet nové.
GroupPurchasesFragment	Fragment obsahující nákupy a položky vybrané skupiny. Jsou zde možnosti jako přidávat/odebírat nákupy/položky.
MembershipOffersFragment	Fragment obsahující příchozí nabídky ke členství od skupin.
SearchGroupFragment	Fragment sloužící k vyhledání skupiny, aby mohla být vytvořena žádost o členství.
TemplatesFragment	Fragment sloužící ke správě šablon nákupů.

Tabulka 3.4: Seznam fragmentů aktivity ShoppingShareMainActivity



Obrázek 3.3: AddGroupFragment s našeptávačem uživatelů k pozvání do skupiny

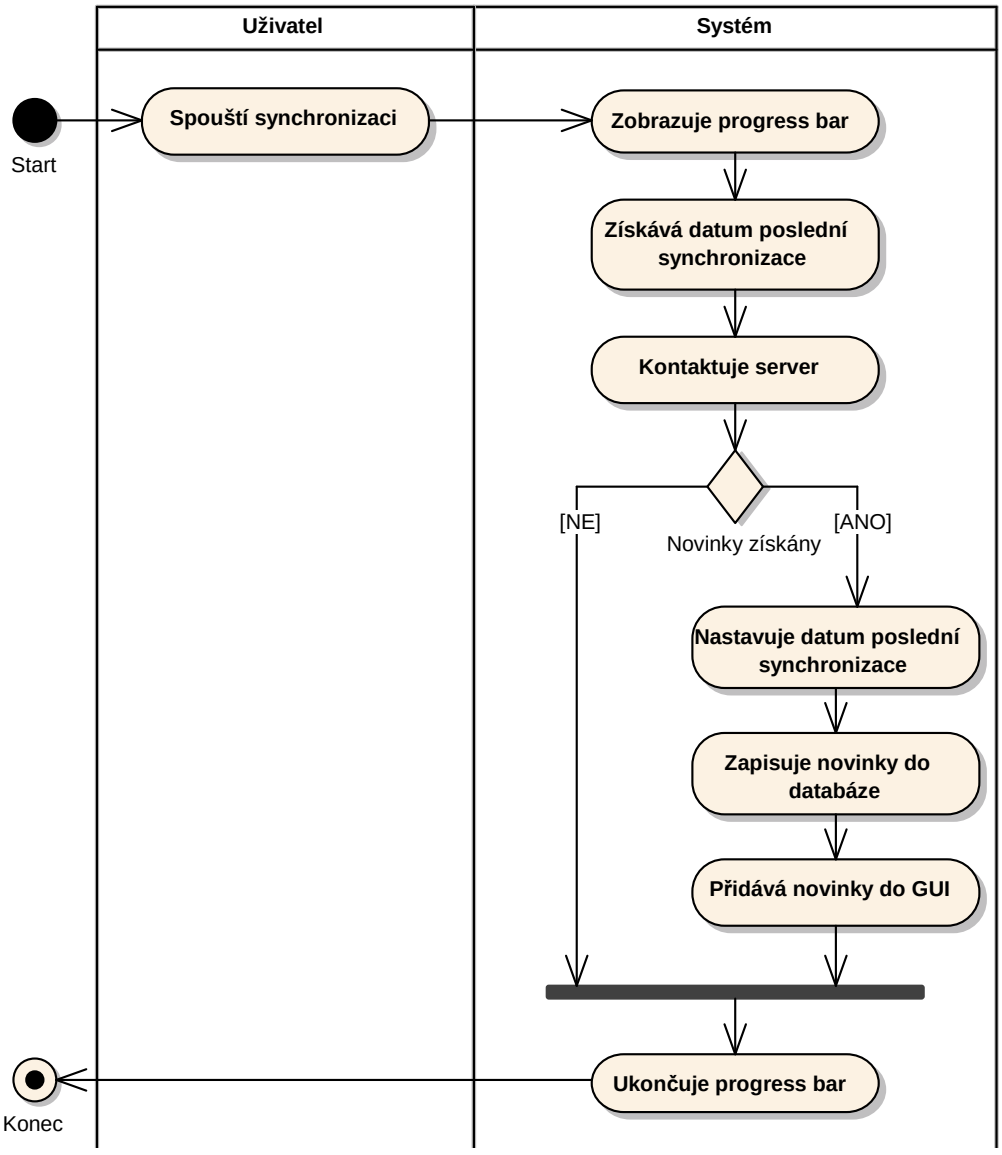


Obrázek 3.4: Vysouvací menu aplikace a GroupPurchasesFragment s nákupy

3.2.4 Synchronizace dat se serverem

Primární úložiště dat je v systému ShoppingShare realizováno databází na serveru. Mobilní aplikace se serverem synchronizuje pouze data týkající se skupin, kterých je uživatel členem. Synchronizovaná data jsou uložena v interní databázi systému Android. Sekundární úložiště slouží k zobrazení dat i bez internetového připojení telefonu.

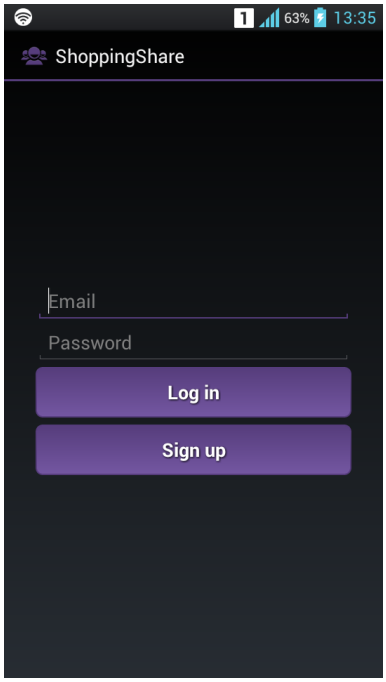
Synchronizace je spuštěna automaticky při přechodu mezi obrazovkami aplikace. Manuální spuštění je možné pomocí gesta pull-to-refresh. Princip synchronizace je naznačen na následujícím diagramu.



Obrázek 3.5: Diagram popisující synchronizaci dat mezi mobilní aplikací a serverem

3.3 Zabezpečení

Účet uživatele je chráněn pomocí kombinace e-mailu a hesla, což je vidět na následujícím obrázku.



Obrázek 3.6: Přihlašovací obrazovka mobilní aplikace

Webová aplikace

Webová aplikace je zabezpečena pomocí Spring Security. Jedná se o modul frameworku Spring. Spring Security umožňuje jednoduše nastavit úrovně zabezpečení pro přístup na jednotlivé adresy či k metodám. Adresy sloužící k přihlášení, vytvoření účtu nebo přístupu k prvkům uživatelského rozhraní jsou přístupné i bez přihlášení. K přístupu na ostatní adresy je nutno mít roli uživatele získatelnou po přihlášení do systému. Uživatel se přihlašuje pomocí emailu a hesla.

Spring Security je zcela nezávislý na použitém aplikačním serveru. Nastavení je realizováno v rámci nastavení Spring, v samostatném konfiguračním souboru.

Přenos dat

Přenos dat je zabezpečen šifrováním, což je zajištěno protokolem HTTPS. SSL certifikát je vlastnoručně podepsaný, což přináší problémy popsané v části o použitých technologiích 3.2.2.

Mobilní aplikace

Zabezpečení mobilní aplikace je zajištěno v rámci zabezpečení systému Android, kdy žádná aplikace nemá přístup k běhu aplikace jiné.

Další důležitou součástí je zabezpečení interní mobilní databáze. To je realizováno nastavením, aby k dané databázi měla přístup pouze aplikace ShoppingShare.

Kapitola 4

Testování

Testování je nedílnou součástí vývoje software. Ověřuje se, zda funkcionalita odpovídá specifikaci a jestli software neobsahuje chyby. ShoppingShare je systém, jehož vývoj je řízen testy. Před implementací funkčnosti je napsán test tuto funkčnost testující. Funkčnost je správně naimplementována ve chvíli neselhání testu. Vývoj řízený testy zaručuje menší chybovost softwaru. Zároveň je k dispozici sada testů ověřujících velkou část funkčnosti, čímž lze prověřit zásahy do již existujícího kódu.

4.1 Unit testování

Unit testování je testování nejmenších jednotek. V aplikaci psané v jazyce Java se jedná o metody. Pro unit testování serverové aplikace jsem použil frameworky JUnit a gmock, jednotlivé testy jsou tedy psané v jazyce Groovy. Výhody oproti testům psaným v jazyce Java vidím v kratší syntaxi a přístupu k privátním členům třídy.

Testy nevyužívají produkční databázi, ale vlastní embedovanou, čímž je zaručeno oddělení produkčního a testovacího prostředí.

V současné době serverová aplikace obsahuje několik desítek automatizovaných unit testů, díky nimž se podařilo objevit chyby většinou nízké závažnosti.

4.2 Testování aplikace pro systém Android

Automatické testování aplikací pro systém Android je vhodné pouze pro několik specifických případů užití. Místo něj se v praxi používá pseudonáhodný generátor uživatelské interakce Monkey, schopný běhu v emulovaném systému či na reálném zařízení. Výhoda nástroje Monkey je objevení krizových situací, o kterých při psaní automatizovaných testů nemusíme vědět. Díky tomuto nástroji byla zjištěna závažná chyba projevující se v uzavření spojení do lokální databáze systému Android při dlouhodobější minimalizaci aplikace. Dále bylo objeveno několik chyb použité knihovny ActionBar-PullToRefresh, o čemž je více napsáno v části o význačných prvcích implementace (viz 3.2.2).

Aplikace pro systém Android byla dále testována s uživateli.

4.3 Testování s uživateli

Pro testování s uživateli jsem vytvořil 3 kategorie lidí dle zkušeností s informačními technologiemi. Testy probíhali s jedním velmi zkušeným, dvěmi středně zkušenými a dvěmi málo zkušenými. Cílem testů bylo zvládnout použít veškerou funkcionalitu definovanou případy užití.

4.3.1 Velmi zkušený uživatel

Velmi zkušený uživatel neměl víceméně žádný problém použít veškerou funkcionalitu systému. Přínosem tohoto testování byly návrhy ke zlepšení vzhledu a hlavně rozložení elementů webové aplikace. K aplikaci pro systém Android žádné připomínky či návrhy uživatel neměl. Aplikace se mu líbila.

4.3.2 Středně zkušený uživatel

Středně zkušený uživatel měl v průměru problém s dvaceti procenty funkcionality u webové aplikace. Zjištěnou příčinou problémů bylo špatné rozložení elementů, čímž se potvrdily návrhy velmi zkušeného uživatele. K aplikaci pro systém Android žádné připomínky či návrhy uživatel neměl. Středně zkušený uživatel pomohl upravit případ užití „Přidat nákup ze šablony“, čímž usnadnil práci se systémem.

4.3.3 Málo zkušený uživatel

Málo zkušený uživatel měl velké problémy systém používat. Nejprve nepochopil účel systému, poté nevěděl jak případy užití provést. U aplikace pro systém Android neznal základní uživatelská gesta definovaná výrobcem systému. Málo zkušený uživatel probudil myšlenku o vytvoření manuálu, který by se objevil při prvním spuštění aplikace a později výběrem z nabídky.

Kapitola 5

Závěr

5.1 Reálné nasazení

Referenční verze systému je nasazena na školou poskytnutém serveru a doméně <www.shoppingshare.net>. Podle míry vzrůstání počtu uživatelů bude rozhodnuto o pronajmutí vlastního výkonnějšího řešení.

5.2 Budoucí rozvoj

V projektu je naplánováno pokračovat i po skončení této práce. V první fázi budou shromažďovány a analyzovány reakce uživatelů na momentální funkčnost, vzhled i celkový přínos systému.

Nápady na rozšíření funkčnosti se objevily už při testování s uživateli. Jako příklad bych uvedl myšlenku přidělování nákupu položek členům skupiny. Přidělený člen, viditelný v seznamu položek, by byl za nákup položky odpovědný.

Další plánovanou funkčností je napovídání položek nákupu na základě průběžně vytvářené analýzy často se pospolu vyskytujících položek.

5.3 Zhodnocení projektu

Úkolem bylo implementovat systém, který umožní snadnou tvorbu a sdílení nákupních seznamů mezi uživateli. Systém se měl skládat z webové aplikace a aplikace pro Android zařízení. Veškerá zadaná funkčnost byla zanalyzována, naimplementována a otestována, a to pro obě kompomenty systému.

Systém je postaven na moderních technologiích fungujících na principu vícevrtstvé architektury. Z toho důvodu je možné jakoukoli vrstvu navázáním na stávající rozhraní jednoduše nahradit. Webová aplikace je schopna díky abstrakci nad databázovou vrstvou pracovat s jakoukoli fyzickou databází. Změnu konkrétní databáze lze provést i za běhu webové aplikace.

Vytvořené rozhraní REST webové aplikace je možné využít ke komunikaci s jakýmkoli mobilním systémem, desktopovou aplikací nebo čímkoli jiným schopným konzumovat REST rozhraní s daty ve formátu JSON.

Vývoj systému ShoppingShare byl pro mě výzvou s vidinou velkého množství získaných zkušeností. Toto očekávání se splnilo i díky množství použitých technologií.

Literatura

[1] chrisbanes. *ActionBar-PullToRefresh* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <https://github.com/chrisbanes/ActionBar-PullToRefresh>.

[2] FasterXML. *Jackson Databind* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <https://github.com/FasterXML/jackson-databind>.

[3] gmock. *gmock* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://code.google.com/p/gmock>.

[4] Gradleware. *Gradle* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://www.gradle.org>.

[5] Jackson. *Jackson Data Mapper* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://jackson.codehaus.org>.

[6] JUnit. *JUnit* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://junit.org>.

[7] PostgreSQL. *PostgreSQL* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://www.postgresql.org>.

[8] Red Hat. *Hibernate* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://hibernate.org>.

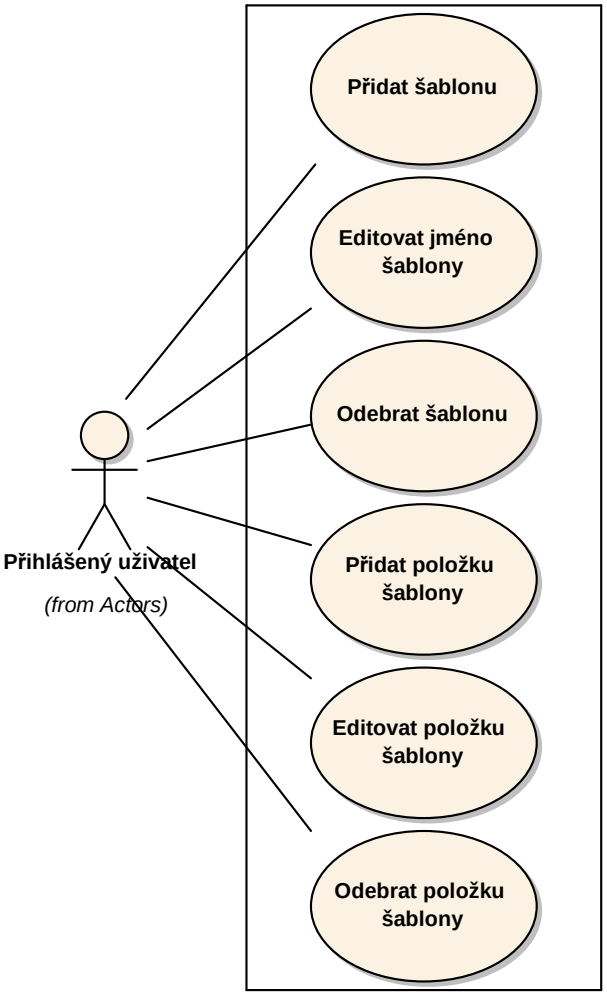
[9] Spring. *Spring Framework* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://http://projects.spring.io/spring-framework>.

[10] Spring. *Spring for Android* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://projects.spring.io/spring-android>.

[11] Twitter. *Twitter Bootstrap* [online]. 2014. [cit. 21. 5. 2014]. Dostupné z: <http://getbootstrap.com>.

Příloha A

UML diagramy



Obrázek A.1: Práce se šablonami nákupů

Příloha B

Ukázky zdrojových kódů

B.1 Vlastní SSL certifikát v aplikaci pro systém Android

Řešením pro používání vlastnoručně podepsaného SSL certifikátu je přepsat funkčnost třídy SSLSocketFactory[1]. Vlastní implementace přepisuje funkčnost metod kontrolujících certifikát a vyhazujících výjimku při zjištění problému.

Řešení se použije při vytváření objektu HttpClient[2] jako verifikátor certifikátů. Objekt HttpClient je potřebný pro vytvoření každého dotazu na server.

```
package com.cvut.shoppingshare.rest;

import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.apache.http.conn.ssl.SSLSocketFactory;

public class EasySSLSocketFactory extends SSLSocketFactory {

    SSLContext sslContext = SSLContext.getInstance("TLS");
```

[1] Celé jméno třídy je: org.apache.http.conn.ssl.SSLSocketFactory
[2] Celé jméno třídy je: org.apache.http.client.HttpClient

```
    public EasySSLSocketFactory(KeyStore truststore) throws
        NoSuchAlgorithmException, KeyManagementException,
        KeyStoreException, UnrecoverableKeyException {
         super(truststore);
         TrustManager tm = new X509TrustManager() {
             public void checkClientTrusted(X509Certificate[] chain
                 , String authType) throws CertificateException { }

             public void checkServerTrusted(X509Certificate[] chain
                 ,  String authType) throws CertificateException {}

             public X509Certificate[] getAcceptedIssuers() {
                 return null;
             }
         };

         sslContext.init(null, new TrustManager[] { tm }, null);
    }

    @Override
    public Socket createSocket(Socket socket, String host, int
        port, boolean autoClose) throws IOException,
        UnknownHostException {
         return sslContext.getSocketFactory().createSocket(socket,
            host, port, autoClose);
    }

    @Override
    public Socket createSocket() throws IOException {
         return sslContext.getSocketFactory().createSocket();
    }

}
```

Příloha C

Obsah přiloženého CD

Přiložené CD obsahuje složky:

- **text:** text práce ve formátu PDF
- **latex:** text práce v LaTeXu
- **zadani:** oskenované zadání práce
- **source:** dva zip soubory se zdrojovými kódy
- **packed:** apk a war soubory