ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rudenko** Jméno: **Vladislav** Osobní číslo: **434942**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové technologie a management**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Internetový obchod s adaptivními prvky

Název bakalářské práce anglicky:

E-shop with adaptive behavior support

Pokyny pro vypracování:

text (maximálně 1000 znaků) - vložte dle Pokynů pro psaní a zadávání BP/DP
1) Analyzujte podporu adaptace v existujících internetových obchodech.
2) Na základě existujícího základu aplikace implementujte vlastní řešení, stávající AppEngine nahraďte více standardním řešením (nasaditelným např. na aplikační server Heroku). Zaměřte se na implementaci nákupního košíku a správy objednávek, které v původní aplikaci chybí.
3) Navrhněte a implementujte prvky pro přizpůsobení aplikace uživateli a doporučení zboží. Implementujte funkce: naposledy navštívené produkty, na základě vlastností podobných zakoupenému zboží doporučené produkty, případně další.
4) Výslednou aplikaci otestujte z hlediska funkčnosti, použitelnosti a ověřte adaptivní chování.
Technologie: J2EE, Spring Boot, JPA, JSF, Primefaces

Seznam doporučené literatury:

[1] Brusilovsky P., Kobsa A., Nejdl W.: The Adaptive Web, Springer, 2007.
[2] Walls C.: Spring Boot in Action, Manning Publications, 2016.
[3] Varaksin O., Caliskan M.: PrimeFaces Cookbook - Second Edition, Packt Publishing Ltd, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Balík, Ph.D., Centrum znalostního managementu FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.02.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**

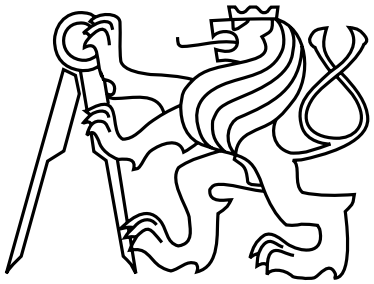
Ing. Martin Balík, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání Podpis studenta

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Internetový obchod s adaptivními prvky

Vladislav Rudenko

Vedoucí práce: Ing. Martin Balík, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

25. května 2018

Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé bakalářské práce Ing. Martinu Balíkovi, Ph.D. za odborné vedení, připomínky a čas, který mi věnoval při zpracování daného projektu. Vděčím též své manželce za trpělivost a podporu, kterou mi poskytovala během zpracování této práce.

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 25. 5. 2018 .....

Abstract

With the rising amount of information on the internet, the need for adaptation of web-shop content based on customers' preferences is increasing as well. The number of online shops and the variety of goods they offer is constantly growing. To ensure that the customer will be well-versed in the assortment offered and will be able to find goods he needs or is interested in, a shop has to adapt its content in some way. That is why the subject of this Bachelor's thesis is to create an online shop which will include adaptive elements, upload it to Heroku platform, and carry out its testing. The practical part of the thesis proposes, implements, and tests an internet shop itself, while the theoretical part of this thesis presents basic information about the nature and types of adaptation systems as well as providing examples of online shops using adaptive elements.

Key words: web-shop, adaptive web-shop, adaptive web systems, online shopping, online shop, e-shop

Abstrakt

S roustoucím množstvím informací roste i potřeba přizpůsobení obsahu internetového obchodu a nabízeného sortimentu potřebám zákazníka. Počet internetových obchodů a množství nabízeného zboží se neustále zvyšují. Aby se zákazník v nabízených položkách vyznal a našel právě to, co potřebuje, nebo to, co ho nejvíce zajímá, je nutné mu nabídnout personalizovaný obsah. Cílem této bakalářské práce je vytvoření vlastního internetového obchodu obsahujícího adaptivní prvky. Výsledný prototyp bude nasazen na platformu Heroku a bude provedeno jeho testování. Teoretická část uvádí základní informace o principech a typech adaptace a zároveň obsahuje několik příkladů internetových obchodů, které využívají adaptaci pro přizpůsobení svého obsahu potřebám zákazníka. Praktická část práce se zabývá samotným návrhem, implementací a testováním internetového obchodu.

Klíčová slova: internetový obchod, adaptivní internetový obchod, adaptivní webové systémy, adaptivní web, online obchod, online shopping, e-shop

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Nedá se popřít, že úspěch podniku záleží na tom, jak efektivně je tento podnik schopen nabídnout své služby zákazníkům a partnerům (ať už stávajícím nebo potenciálním), právě proto je internet v současné době jedním z hlavních prostorů pro provozování ekonomické činnosti. Na straně prodávajícího internetové obchody podstatně snižují náklady na provoz své činnosti, umožňují rozšířit odbytová schémata, a získat tak víc zákazníků. Na straně zákazníka pak tento typ obchodů umožňuje nakoupit skoro jakékoliv zboží bez ohledu na svou lokalitu, 24 hodin denně a celoročně, navíc je v tomto případě možnost získat o produktu podrobnější informace. Tyto skutečnosti dávají internetovým obchodům nespornou přednost v porovnání s kamennými prodejnami a jsou pro prodávajícího rozhodujícím faktorem pro zapojení internetových obchodů do svého hospodaření.

Avšak konkurence mezi internetovými obchody je v současné době tak obrovská, že se firmy musí zamyslet nejenom nad tím, jak přilákat zákazníka, ale i jak ho udržet a pobídnout k dalším nákupům. Z toho důvodu je pro zvýšení efektivity provozu internetového obchodu třeba zohlednit i jeho adaptivní chování vůči kupujícímu. Osobně se domnívám, že se v současnosti jedná o velice aktuální téma, neboť integrace adaptivních prvků do internetového obchodu může pomoct výrazně podpořit prodej, a tak pozitivně ovlivnit výši výnosů podniku.

Právě proto je cílem této bakalářské práce vytvoření internetového obchodu, do kterého budou integrovány prvky adaptivního chování.

V dané práci implementuji základní strukturu internetového obchodu, do kterého integruji prvky adaptivního chování pomocí Adaptive System Framework. Samotná aplikace je vyvinuta tak, aby mohla běžet na Heroku platformě.

První kapitola této práce se bude týkat teoretické části a uvede základní informace o tom, co je adaptivní webový systém a platforma jako služba (PaaS), dále pak popíše několik skutečných případů adaptivního chování. Druhá kapitola se bude zabývat návrhem aplikace a její adaptací, popíše funkční a nefunkční požadavky kladené na tuto aplikaci a uvede případy jejího použití. Do třetí kapitoly bude zahrnut popis implementace internetového obchodu včetně adaptace. Čtvrtá kapitola shrne průběh a výsledky provedeného testování aplikace provedeného pomocí uživatelů a pomocí jednotkových a integračních testů.

Kapitola 2

Teoretická část

Tato kapitola uvádí základní informací o podstatě adaptace, popisuje její základní techniky a pojmy s ní související. Navíc probere několik příkladů adaptace v existujících internetových obchodech.

2.1 Adaptivní webový systém

2.1.1 Základní informace o adaptaci

Otázka adaptivních webových systémů a jejich využití je velmi rozsáhlá, a proto v rámci dané práce uvedeme základní teoretické informace pro lepší představu o důležitosti adaptivních technologií v dnešní době.

2.1.1.1 User Model a User Profile

Adaptivní webové systémy si můžeme představit jako internetovou aplikaci založenou na principu odlišnosti jednotlivých uživatelů. Základem adaptivního webového systému je tzv. **User Model**, tj. každý uživatel aplikace má svůj model, podle jehož obsahu systém přizpůsobuje své reakce. Pro adaptaci je zásadní správně namodelovat uživatele. Mezi hlavní vlastnosti, které se v dnešní době modelují, patří: cíle, zájmy, znalosti, individuální schopnosti, pozadí a kontext práce. Pokud jde například o e-learning, systém se zaměří na znalosti uživatele, v případě informačních a doporučovacích systémů budou cílem spíše jeho zájmy. Je patrné, že cílem adaptivních systémů je zvýšení efektivity a kvality podávání informací. Modelování uživatele není možné uskutečnit bez vytvoření jeho **profilu (User Profile)**. Pro tvorbu profilu uživatele je třeba dostat od něj informace. Sběr informací o uživateli může probíhat dvěma způsoby: explicitně a implicitně[6].

- *Explicitní modelování*:
  - otevřené modelování,
  - komunikace s uživatelem,
  - probíhá na základě nastavení osobního profilu/účtu, vyplněním formuláře nebo dotazníku, tlačítka líbí/nelíbí,

  - zátěž na uživatele.

- *Implicitní modelování*:
  - skryté modelování,
  - pozorování chování uživatele (například historie prohlížení stránek, historie nákupů, hodnocení produktů),
  - zátěž na provozovatele systému kvůli instalaci softwaru, který bude uživatele pozorovat.

2.1.1.2 User Model a User Profile v ASF

Z výše uvedených informací se dá dojít k závěru, že pojmy User Model a User Profile jsou navzájem zaměnitelné v teorii. V rámci dané práce však budeme implementovat adaptaci pomocí Adaptive System Framework, který rozlišuje pojmy User Model a User Profile. Podle ASF toto rozlišení odpovídá rozdělení modelování na implicitní a explicitní[13]:

- **User Model** představuje implicitní personalizaci, kterou zajišťuje adaptivní systém, který sbírá potřebná data o uživateli.
- **User Profile** představuje explicitní personalizaci, kterou zajišťuje uživatel tím, že vkládá data o svých preferencích do systému.

Adaptace dle User Modelu je součásti dané práce a bude podrobněji popsána v bodech 3.4 a 4.4.4.

2.1.1.3 Content Unit Model v ASF

V rámci použití ASF pro adaptaci obsahu zmíním ještě Content Unit Model, který je doplňkem User Modelu a který se vztahuje ne k uživateli, ale k produktu. Kvůli tomu, že model je zaměřen na zpracování informací o produktech, může sloužit například k seřazení produktů do seznamů nejlépe hodnocených nebo nejvíce prodávaných[13]. Adaptace dle Content Unit Modelu je součásti dané práce a bude popsána podrobněji v bodech 3.4 a 4.4.4.

2.1.2 Doporučovací systémy

Doporučovací systémy se začaly ve větším měřítku využívat s aktivním růstem velkých internetových obchodů, jejichž nabídka produktů je obrovská, zatímco uživatel se nejspíš bude zajímat jenom o její malou část. Cílem doporučovacích systémů je tudíž vytvořit nějaký výběr položek z celkového sortimentu a nabídnout ho uživateli na základě jakéhosi algoritmu, který přizpůsobí obsah potřebám nebo zájmům uživatele. Dále se podíváme na několik typů doporučovacích systémů, které se aktivně využívají v dnešní době.

2.1.2.1 Doporučení založené na popularitě

Základní a pravděpodobně nejjednodušší typ doporučení, který je založen na „popularitě“ položky v systému. Patří sem například doporučení typů „Nejlépe hodnocené“ nebo „Nejlépe prodávané“. Jedná se však o nepersonalizovaný typ doporučení, tj. systém nebere v potaz individuální preference uživatele. Nicméně nemusí jít o seznam produktů vybraných z celého sortimentu, je možné omezit výběr na určitou kategorii zboží a nabízet relativně podobné položky. Dalšími možností omezení jsou například časové intervaly (nejprodávanější za den/týden/měsíc atd.), uživatelova poloha (nejprodávanější v ČR) apod. Dá se říct, že daný typ doporučení je skvělým doplněním k personalizovaným typům, které budou popsány v dalších bodech, avšak jako samostatný typ není vždy postačující pro kvalitní adaptaci obsahu. Příklad takového doporučení bude probrán v bodech 2.1.3.1 a 2.1.3.2, zároveň se budeme zabývat návrhem a implementací tohoto typu doporučení v následujících kapitolách.

2.1.2.2 Content-Based Recommendation Systems

Jedním z typů personalizovaných doporučovacích systémů jsou Content-based recommendation systems, neboli doporučovací systémy založené na obsahu. Daný typ doporučovacích systémů se orientuje na profil uživatele (jeho zájmy a hodnocení produktů) a podobnost produktů. Content-based doporučovací systémy fungují na principu “More of the same”, tj. pokud uživatel projeví zájem o určitou položku (nakoupí jí), budou mu nabídnuty podobné produkty (na základě porovnání atributů položek). Největší nevýhodou daných systémů je tedy požadavek, aby k položkám byla známa nějaká metadata, která se musí vyplňovat ručně, což je docela náročný proces. Další nevýhodou daného principu je tzv. “cold-start” problém, kdy zatím nejsou jasné preference nového uživatele a není co mu doporučit. Nejčastěji se tento nedostatek řeší doporučením založeným na popularitě[14].

2.1.2.3 Collaborative Filtering

Kolaborativní filtrování je v současné době jednou z nejpopulárnějších technik pro doporučení. Metody kolaborativního filtrování se dá rozdělit do dvou typů:

- **Memory-based Collaborative Filtering** je nejstarší metodou kolaborativního filtrování. Tyto metody předávají možné vztahy, které se počítají na základě známých vztahů. Používají statistické metody pro zjištění skupiny tzv. „sousedů“, kteří mají historii podobnou té, kterou má cílový uživatel (ve prospěch kterého se doporučení provádí). Jakmile takové sousedství je zjištěno, systém pomocí různých algoritmů kombinuje preference „sousedů“ do seznamu doporučení pro cílového uživatele. Daný typ kolaborativního filtrování je dále možné rozdělit do dvou směrů:
  - *User-based*: doporučení založené na podobnosti uživatelů. Základní myšlenkou daných systémů je předpoklad, že uživatelé, kteří mají podobnou historii nákupů, pravděpodobně budou mít obdobné zájmy, tj. je nutné vybrat podmnožinu uživatelů na základě podobnosti s aktivním uživatelem (historie nákupu nebo hodnocení) a doporučit danému aktivnímu uživateli chybějící položky (které si ještě nekoupil). Velkou nevýhodou dané metody je skutečnost, že se v současné době

zpracovávají obrovská množství uživatelů a položek, kvůli čemuž se v reálném čase nedá vypočítat předpověď kombinací preferencí „sousedů“ a aktivního uživatele.

  - *Item-based*: doporučení založené na podobnosti položek. Řeší problém user-based metody tím, že vypočítá předpověď použitím podobností mezi produkty a ne mezi uživateli. Základní myšlenkou dané metody je to, že pokud si uživatel koupil položku A, existuje určitá pravděpodobnost, že si v budoucnu bude chtít koupit podobnou položku B. Velmi zjednodušeně daný proces vypadá tak, že určitý algoritmus nejdříve přiřadí nákupům nebo hodnocením podobné položky, poté je zkombinuje do seznamu doporučených produktů.

- **Model-based Collaborative Filtering**, kam patří například neuronové sitě. Dané metody zajišt'ují doporučení tím, že nejdřív vytvářejí model, který bude předpovídat neznámé vztahy. K naučení tohoto modelu se využívají strojové techniky (machine-learning), známé vztahy se využívají jako cvičná data.

Kolaborativní filtrování se také setkává s problémem studeného startu, a to jak na straně user-based metod, tak i na straně item-based metod. Nový uživatel nemá historii nákupů a tudíž zatím žádný produkt nehodnotil, problém se dá vyřešit požádáním uživatele o vložení dat. Nově zařazený do obchodu produkt rovněž nemá žádné vztahy s jinými produkty ani hodnocení, což se řeší postupem času pomocí doporučení nového produktu využitím jiných technik[21] (například pomocí content-based doporučení dojde k nákupu a hodnocení nového produktu, a pak na základě vložených hodnocení bude zařazen do kolaborativního filtrování).

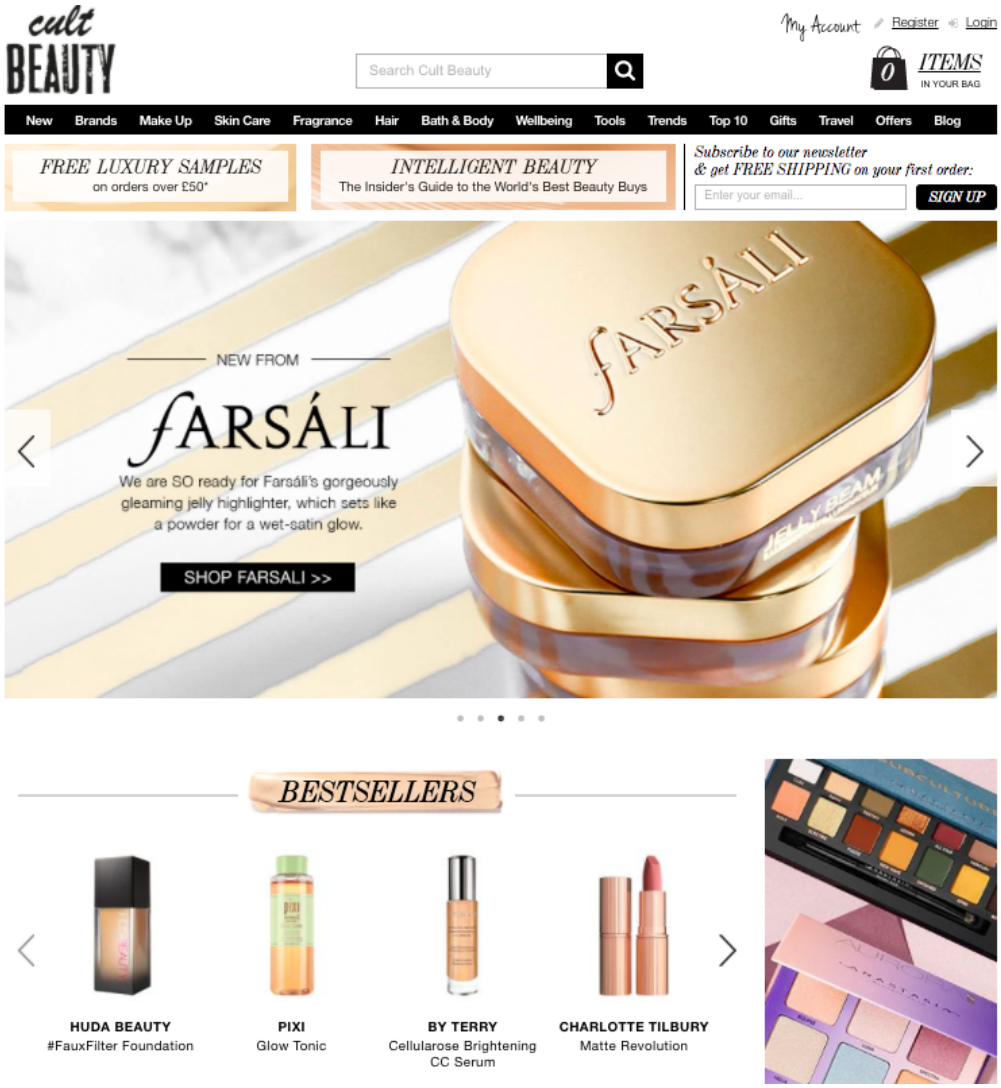
2.1.3 Příklady adaptace

Dále se podíváme na několik internetových obchodů, které využívají adaptaci obsahu svých stránek.

2.1.3.1 Cultbeauty.co.uk

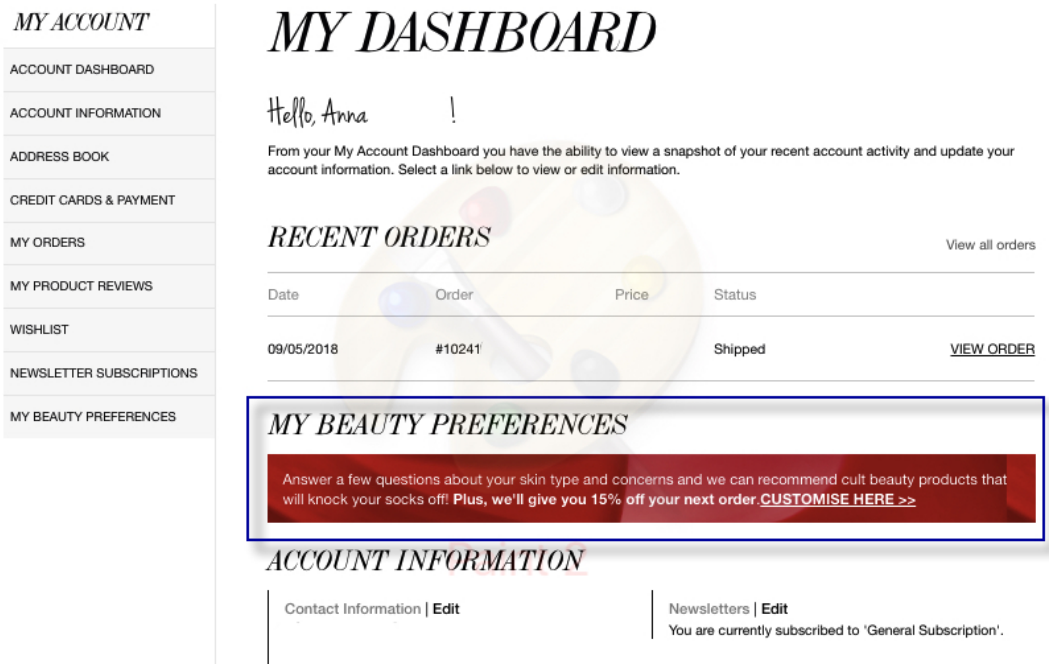
Pro první příklad se obrátíme na hodně populární britský obchod s kosmetikou a parfumerií cultbeauty.co.uk[7], který využívá adaptaci pro přizpůsobení obsahu svých stránek potřebám zákazníka a zvýšení prodeje.

Na obrázku 2.1 je znázorněna úvodní stránka obchodu pro nepřihlášeného uživatele, kterému je nabídnut seznam s best sellery daného obchodu.



Obrázek 2.1: Úvodní stránka cultbeauty.co.uk, nepřihlášený uživatel

Po přihlášení do uživatelského účtu nabízí obchod v sekci správy účtu „My Account“ uživateli doplnit podrobnější informací o svém věku, typu pleti, svých preferencích v kosmetice, oblíbených značkách apod., tj. obchod explicitně buduje model zákazníka cestou komunikace pomocí vyplnění dotazníku. Na základě odpovědí pak bude obchod schopen nabídnout zákazníkovi přesně to, co ho nejvíc zajímá. Hlavní motivací pro zákazníka v tomto případě je obdržení 15% slevového kupónu pro příští nákup (Obrázek 2.2).



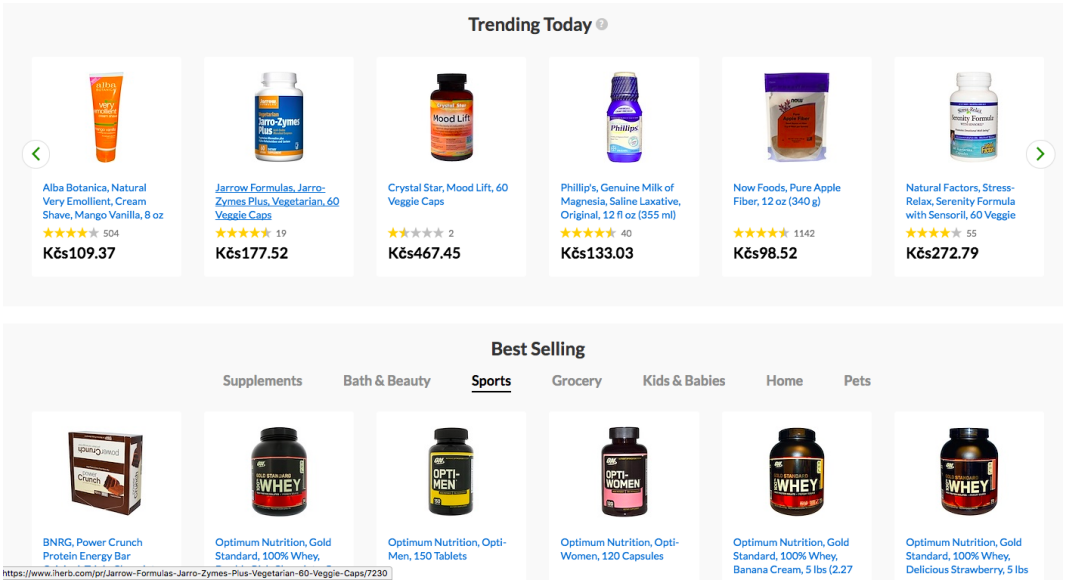
Obrázek 2.2: Nabídka doplnit osobní preference

Avšak po vyplnění předloženého rozsáhlého dotazníku jsme žádný slevový kupón neobdrželi, tudíž budeme muset kontaktovat zákaznický servis, abychom zjistili, jestli jsme všechno udělali správně nebo se jedná o technickou záležitost na straně obchodu.

2.1.3.2 Iherb.com

Dalším obchodem, který chci uvést jako příklad, je americký obchod iherb.com[12], kde se prodává zdravá výživa, doplňky stravy, vitamíny, kosmetika a kosmetické doplňky. Obchod má například následující adaptivní prvky, které jsou stejné pro přihlášeného a nepřihlášeného uživatele:

- Zobrazení seznamů „Best Sellers“ a „Trending Today“ na úvodní stránce obchodu. Podle počtu prodaných kusů obchod generuje seznamy nejvíc prodávaného zboží a nabízí jejich obsah zákazníkovi, přičemž seznam „Best Sellers“ je navíc rozdělen do podkategorií jako „Supplements“, „Bath&Beauty“, „Sports“, „Grocery“, „Kids&Babies“, „Home“ a „Pets“ (Obrázek 2.3).



Obrázek 2.3: Nabídka nejvíce prodávaného zboží v obchodě iherb.com

- Zobrazení seznamu „Recenly Viewed“ v detailech jakéhosi zboží (Obrázek 2.4). Zase jsme na stránce kokosového oleje a úplně dole můžeme najít seznam produktů, na které jsme se dívali, než se dostaly na tuto stránku s kokosovým olejem, což znamená, že obchod má zavedený prvek adaptace, který sleduje a ukládá chování zákazníka.

Obrázek 2.4: Seznam „Recently Viewed Products” iherb.com

2.1.3.3 Shrnutí

Tyto dva obchody jsem vybral jako příklady z důvodu, že moje rodina je oba docela často používá. V případě obchodu cultbeauty.co.uk se mi líbí myšlenka detailnější personalizace obsahu stránek na základě preferencí zákazníka. Tyto preference však musí označit sám (User Profile v ASF, není součástí návrhu a implemetace v rámci dané práce). Vyplnění dotazníku na stránkách cultbeauty.co.uk z naších zkušeností zabírá hodně času, museli jsme se vyznat

v obrovském množství značek, navíc pak nabídnout další značky, o které máme zájem, ale které nejsou zatím součástí nabídky obchodu. Nevýhoda této metody personalizace spočívá v tom, že se postupem času zájmy zákazníka mohou měnit, a tak vznikne potřeba v jejich aktualizaci, na kterou zákazník může zapomenout nebo vůbec se nebude jí zabývat kvůli časové náročnosti.

Obchod iherb.com hodně často využívám sám k nákupu sportovní výživy a doplňků stravy, proto tento obchod byl jedním ze zdrojů inspirace během návrhu adaptace ve vlastním internetovém obchodě. Osobně vnímám typ doporučení "Recently Viewed" (User Model v ASF 3.4) jako jeden z nejvíc užitečných, jelikož usnadňuje přehled a vrácení k produktům, na které jsem se díval před chvíli, tudíž nemusím je vyhledávat v historii prohlížeče.

Seznam "Best Sellers" (Content Unit Model v ASF 3.4) je v dnešní době samozřejmostí: skoro každý internetový obchod ho má hned na úvodní stránce. Dle mého názoru je však lépe implementován v obchodě iherb.com, protože je rozdělen podle kategorií pro usnadnění navigace, cultbeauty.co.uk takové rozdělení nemá.

Návrh adaptací "Best Sellers" a "Recently Viewed" je součástí bodu 3.4 této práce, jejich implementace je popsána v bodě 4.4.4.

2.2 Platform as a Service

Co je Platform as a Service (PaaS)[20]? S čím nám pomůže, pokud se rozhodneme založit internetový obchod? Platform as a Service je model cloud computingu, ve kterém uživatel získává přístup k využívání informačních a technologických prostředků (operační systémy, systémy pro správu databází, vývojové a testovací nástroje), které jsou umístěny u cloudového poskytovatele. V tomto modelu je veškerá informační a technická infrastruktura zcela spravována poskytovatelem, který definuje parametry přístupné uživatelům. Uživatel pak má možnost tyto platformy využít: vytvářet jejich virtuální kopie, instalovat, vyvíjet, testovat a používat aplikační software a zároveň dynamicky měnit jejich množství potřebné k provozu aplikace počítačových zdrojů.

Poskytovatel cloudové platformy může zpoplatnit své služby. Účtování poplatků je možné například v závislosti na délce doby provozu aplikace nebo na objemu přenesených dat a množství databázových transakcí. Poskytovatelé cloudových platforem mají ekonomický výnos za pomocí virtualizace a úspor z rozsahu, když z celého množství uživatelů pouze jejich část využívá výpočetní zdroje aktivně a najednou. Uživatelé mají výhodu v tom, že nemusí investovat do vlastní infrastruktury a platformy, která by odpovídala špičkovému výkonu, zároveň se pak nemusí starat o náklady na údržbu celého tohoto komplexu.

Jednoduše řečeno, PaaS umí skutečně zjednodušit provoz jakékoliv aplikace a velmi často se používá i pro vytvoření a zabezpečení provozu internetových obchodů. V současné době existuje několik společností, které poskytují služby PaaS. Mezi nejvýznamnější platformy patří například Google App Engine, Microsoft Azure Platform, OpenShift a Heroku. Dále bych rád stručně popsal každou z těchto platforem.

- **Google App Engine**[1] je služba, která zajišťuje hosting webových aplikací na serverech společnosti Google. Služba je určena pro jazyky Node.js, Java, Ruby, Go, Python a

[1] https://cloud.google.com/

PHP. Při registraci vyžaduje služba údaje platební karty. Po registraci se na uživatelský účet připíše kredit ve výši $300. Bezplatná zkušební verze končí, jakmile poplatky zákazníka za použití služeb přesáhnou $300 nebo po 12 měsících od data zahájení zkušební verze. Google App Engine pracuje se třemi druhy databází: Google Cloud SQL, Google Cloud Storage a Google Cloud Datastore. První dva druhy jsou určeny pro náročnější aplikace a jsou zpoplatněny na rozdíl od Google Cloud Datastore. Služeb této platformy využívají například Spotify, Netflix a obchod s oblečením Uniqlo[2].

- **Microsoft Azure Platform**[3] je cloudová platforma společnosti Microsoft. Používá se k vytváření, hostování a škálování webových aplikací přes datové centrum Microsoftu. Microsoft Azure je určen pro jazyky .NET, Java, Node.js, PHP nebo Python. SQL Azure nabízí MySQL a PostgreSQL databáze pro vývojáře. Platforma umožňuje vytvoření bezplatného účtu Azure, kam se po registraci připíše kredit 170 EUR, který je pak možné po dobu 30 dní používat na jakékoli produkty Azure. Pokud se zákazník po této době nerozhodne pro placený upgrade, může i nadále využívat služby Azure zdarma následujících 12 měsíců v rámci bezplatných produktů. Na stránkách Azure je možné nalézt detailnější popis veškerých bezplatných produktů. Při registraci vyžaduje údaje platební karty a může dojít k dočasné autorizační blokaci peněz na účtě. Služeb této platformy využívají například společnosti jako Adobe, Honeywell a GE Healthcare.

- **OpenShift**[4] je produkt společnosti Red Hat určený pro zavádění a správu softwaru. OpenShift podporuje jazyky Java, Node.js, .NET, PHP, Python, Ruby, Perl a databáze jako MariaDB, MongoDB, MySQL, PostgreSQL, Redis. Služba nabízí 2 typy předplatného: „Starter“ a „Pro“. První je určeno pro individuální použití, má limit 1 GiB RAM a 1 GiB úložiště, maximálně je možné pracovat s jedním projektem, služba je poskytovaná zcela zdarma. Při registraci nevyžaduje údaje platební karty. Předplatné „Pro“ se dá pořídit od $50 měsíčně, je určeno pro náročnější a profesionální projekty. Pro tuto platformu se rozhodli například Accenture, Dell a T-Systems[5].

- **Heroku**[6][10] je cloudová platforma, vznikla v roce 2007 a byla ve skutečnosti jednou z prvních cloudových platforem. Tato platforma původně podporovala pouze programovací jazyk Ruby, ale v současné době seznam podporovaných jazyků obsahuje také jazyky Java, Node.js, Scala, Clojure, Python, Go a PHP. V rámci této platformy se využívají databáze PostgreSQL, Redis a Kafka. S limitem 512 MB RAM, 1 web/1 worker je tato platforma bezplatná. Při registraci nevyžaduje údaje platební karty. Po registraci se hned uvádí detailní návod pro nahrání aplikace v Javě na server, jakož i pro všechny ostatní programovací jazyky, které Heroku podporuje. Služeb této platformy využívají společnosti jako Toyota, Macy's a CITRIX[7].

[2] https://cloud.google.com/why-google-cloud/
[3] https://azure.microsoft.com
[4] https://www.openshift.com/
[5] https://www.openshift.com/about/
[6] https://www.heroku.com/
[7] https://www.heroku.com/customers

PaaS	Java	PostgreSQL	Bezplatný	Platební údaje
Google Cloud Platform	ANO	ANO	NE	ANO
Microsoft Azure	ANO	ANO	NE	ANO
Red Hat OpenShift	ANO	ANO	ANO	NE
Heroku	ANO	ANO	ANO	NE

Tabulka 2.1: Přehled PaaS k tvorbě obchodu

Pro účely tohoto projektu jsem se rozhodl pro platformu Heroku, jelikož je bezplatná, relativně jednoduchá na použití a její technické limity vyhovují mým požadavkům.

Kapitola 3

Návrh aplikace

Tato kapitola se bude zabývat návrhem internetového obchodu. Začneme vymezením požadavků, dále probereme případy užití, strukturu obchodu a uživatelské rozhraní.

3.1 Požadavky

Ze zadání práce je patrné, že cílem projektu je návrh a implementace vlastního internetového obchodu s adaptivními prvky. Z toho vyplývají určité funkční a nefunkční požadavky.

3.1.1 Funkční

- Obchod se bude skládat z administrační a klientské části.
- Do administrační části budou mít přístup pouze přihlášení zaměstnanci obchodu.
- Přístup do klientské části bude omezen pro nepřihlášené uživatele na prohlédnutí seznamu nabízeného a detailů produktů bez možnosti přidání zboží do košíku.
- Přidat zboží do košíku budou moci pouze přihlášení uživatelé.
- Přihlášený uživatel bude moci spravovat svůj košík.
- Přihlášený uživatel bude moci z košíku udělat objednávku.
- V klientské části bude zboží rozděleno do kategorií.
- Detail zboží bude obsahovat i další doporučené zboží ze stejné kategorie.
- Zboží bude moci být hodnoceno pouze přihlášenými uživateli.
- Domovská stránka obchodu bude obsahovat seznam nejlépe se prodávajícího zboží.
- Domovská stránka obchodu bude obsahovat seznam naposledy zákazníkem prohlédnutého zboží.
- Domovská stránka obchodu bude obsahovat seznam zboží stejné kategorie, ze které si již zákazník něco kupoval.

3.1.2 Nefunkční

- Použití jazyku Java[17].
- Použití Apache Maven[29].
- Použití databáze PostgreSQL[32].
- Zpracování dat pomocí JPA[19].
- Použití frameworku Spring[22] a frameworku PrimeFaces[15].
- Zabezpečení aplikace pomocí Spring Security[1].
- Použití frameworku ASF[4] pro zavedení adaptivního chování.
- Použití knihovny React[9].
- Nasazení aplikace na PaaS Heroku[10].

3.2 Případy užití

Případy užití jsou kvůli přehlednosti rozděleny do dvou částí:

- administrační část,
- klientská část.

3.2.1 Případy užití klientské části

Případy užití klientské části jsou schematicky znázorněny na obrázku 3.1

1. Registrace uživatele
   Popis: Vytvoření účtu novému uživateli, kam se bude moct přihlašovat
   Aktéři: Návštěvník
   Scénář:
   (a) Uživatel klikne na pravé horní tlačítko „Register“.
   (b) Systém načte formulář pro vyplnění osobních údajů.
   (c) Uživatel zadá osobní údaje.
   (d) Uživatel uloží osobní údaje pomocí tlačítka „Save“.
   (e) Systém zkontroluje uvedené osobní údaje, zda již náhodou neexistuje účet se stejnými parametry.
2. Přihlášení uživatele
   Popis: Přihlášení existujícího uživatele do aplikace
   Aktéři: Návštěvník
   Scénář:

(a) Uživatel klikne na pravé horní tlačítko „Sign in“.

(b) Systém načte formulář pro přihlášení uživatele.

(c) Uživatel zadá svůj e-mail a heslo.

(d) Uživatel potvrdí údaje stisknutím tlačítka „Sign in“.

(e) Systém zkontroluje uvedené údaje, zda je má uložené v databázi.

(f) Pokud uvedené osobní údaje v databázi jsou, uživatel bude přesměrován na úvodní stránku obsahující adaptaci.

3. Změna osobních údajů
Popis: Změna osobních údajů přihlášeného uživatele
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na pravé horní tlačítko „My Account“.

(b) Systém načte formulář s osobními údaji uživatele.

(c) Uživatel změní potřebné osobní údaje.

(d) Uživatel potvrdí změny pomocí tlačítka „Save“.

(e) Systém uloží změny.

4. Zobrazení seznamu produktů
Popis: Zobrazení seznamu produktů na úvodní stránce aplikace
Aktéři: Návštěvník, přihlášený uživatel
Scénář:

(a) Uživatel otevře úvodní stránku obchodu.

(b) Systém načte seznamy produktů na základě adaptace a registrace.

5. Zobrazení detailů produktu
Popis: Zobrazení detailů produktu v aplikaci
Aktéři: Návštěvník, přihlášený uživatel
Scénář:

(a) Uživatel klikne na produkt.

(b) Systém načte detail vybraného produktu.

(c) Uživatel bude přesměrován na stránku s detailním popisem zvoleného produktu.

6. Přidání produktu do košíku
Popis: Přidání nového produktu do košíku přihlášeného uživatele
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne buď:

- na ikonu „Add to cart“ umístěnou pod každým produktem v seznamech produktů,
- nebo na tlačítko „Add to cart“ v detailu produktu.

(b) Systém přidá zvolený produkt do košíku uživatele.

(c) Systém upozorní uživatele na přidání zvoleného produktu do košíku.

7. Změna počtu produktů v košíku
Popis: Změna počtu produktů v košíku přihlášeného uživatele
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na pravé horní tlačítko „Cart“.

(b) Systém načte stránku košíku obsahující seznam produktů vybraných uživatelem.

(c) Uživatel změní počet potřebných produktů.

8. Odstranění produktu
Popis: Odstranění vybraného produktu z košíku přihlášeného uživatele
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na pravé horní tlačítko „Cart“.

(b) Systém načte stránku košíku obsahující seznam produktů vybraných uživatelem.

(c) Uživatel odstraní vybraný produkt z košíku.

9. Objednání produktů
Popis: Objednání vybraných produktů přihlášeným uživatelem
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na pravé horní tlačítko „Cart“.

(b) Systém načte stránku košíku obsahující seznam produktů vybraných uživatelem.

(c) Uživatel klikne na tlačítko „Check out“.

(d) Systém načte seznam produktů k objednání a způsoby dopravy.

(e) Uživatel zvolí vhodný způsob dopravy.

(f) Uživatel klikne na tlačítko „Order“, tím potvrdí objednávku.

(g) Po umístění objednávky bude uživatel přesměrován na stránku s poděkováním.

10. Hodnocení produktu
Popis: Hodnocení produktu v aplikaci přihlášeným uživatelem
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na produkt, který chce ohodnotit.

(b) Systém načte stránku s detaily produktu.

(c) Na stránce detailů produktu uživatel klikne na tlačítko s tužkou „Rate“.

(d) Systém načte formulář hodnocení produktu.

(e) Uživatel napíše komentář ke svému hodnocení.

(f) Uživatel zvolí vhodný počet hvězdiček.

(g) Uživatel potvrdí uložení svého hodnocení stisknutím tlačítka „Save“.

(h) Systém uloží hodnocení produktu.

11. Odhlášení uživatele
Popis: Odhlášení přihlášeného uživatele z aplikace
Aktéři: Přihlášený uživatel
Scénář:

(a) Uživatel klikne na pravé horní tlačítko „Sign out“.

(b) Systém odhlásí uživatele.

(c) Uživatel bude přesměrován na úvodní stránku s obsahem přizpůsobeným pro nepřihlášeného uživatele.

Obrázek 3.1: Případy užití klientské části

3.2.2 Případy užití administrační části

Administrační část funguje na základě existence rolí a privilegií s tím, že administrátor má možnost vytvořit nekonečný počet takových rolí a přiřadit jim různé kombinace privilegií. Pro popis případů užití v rámci této práci jsem použil dvě role: administrátor a zaměstnanec.

Případy užití administrační části jsou schematicky znázorněny na obrázku 3.2

1. Zobrazení seznamu produktů
   Popis: Zobrazení seznamu produktů v administrační části aplikace
   Aktéři: Administrátor, zaměstnanec
   Scénář:
   (a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
   (b) Systém načte a zobrazí uživateli seznam produktů.
2. Zobrazení detailů produktu
   Popis: Zobrazení detailů produktu v administrační části aplikace
   Aktéři: Administrátor, zaměstnanec
   Scénář:
   (a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
   (b) Systém načte a zobrazí seznam produktů.
   (c) Uživatel zvolí produkt v seznamu a klikne na něj.
   (d) Systém povolí tlačítko „Detail“ dole pod tabulkou se seznamem.
   (e) Uživatel klikne na tlačítko „Detail“.
   (f) Systém načte a zobrazí uživateli detail produktu.
3. Vytvoření nového produktu
   Popis: Vytvoření nového produktu v administrační části aplikace
   Aktéři: Administrátor, zaměstnanec
   Scénář:
   (a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
   (b) Systém načte a zobrazí seznam produktů.
   (c) Uživatel klikne na tlačítko „New“ pod tabulkou se seznamem produktů.
   (d) Systém načte a otevře formulář pro vytvoření nového produktu.
   (e) Uživatel zadá údaje o produktu.
   (f) Uživatel potvrdí uložení údajů pomocí tlačítka „Save“.
   (g) Systém zkontroluje, zda jsou všechny údaje v pořádku.
   (h) Systém vytvoří nový produkt.

(i) Systém přesměruje uživatele zpátky na seznam produktů.

4. Úprava produktu
Popis: Úprava údajů o produktu v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
(b) Systém načte a zobrazí seznam produktů.
(c) Uživatel zvolí produkt v seznamu a klikne na něj.
(d) Systém povolí tlačítko „Edit“ dole pod tabulkou se seznamem.
(e) Uživatel klikne na tlačítko „Edit“.
(f) Systém načte a otevře formulář s údaji o produktu.
(g) Uživatel změní potřebné údaje.
(h) Uživatel potvrdí změnu údajů kliknutím na tlačítko „Save“.
(i) Systém zkontroluje, jestli jsou všechny údaje v pořádku.
(j) Systém uloží změny produktu.
(k) Systém přesměruje uživatele zpátky na seznam produktů.

5. Přidání obrazovek
Popis: Přidání obrazovek k produktu v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
(b) Systém načte a zobrazí seznam produktů.
(c) Uživatel zvolí produkt v seznamu a klikne na něho.
(d) Systém povolí tlačítko „Pictures“ dole pod tabulkou se seznamem.
(e) Uživatel klikne na tlačítko „Pictures“.
(f) Systém načte a otevře stránku pro nahrávání obrazovek.
(g) Uživatel přidá příslušné obrazovky zvolenému produktu.

6. Odstranění produktu
Popis: Odstranění produktu v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Products“ v menu v levé straně úvodní stránky administrační části.
(b) Systém načte a zobrazí seznam produktů.
(c) Uživatel zvolí produkt v seznamu a klikne na něho.

(d) Systém povolí tlačítko „Delete“ dole pod tabulkou se seznamem.

(e) Uživatel klikne na tlačítko „Delete“.

(f) Systém zobrazí dialogové okno, aby uživatel potvrdil odstranění.

(g) Uživatel potvrdí odstranění kliknutím na tlačítko „Yes“.

(h) Systém odstraní vybraný produkt.

(i) Systém zavře dialogové okno.

7. Zobrazení seznamu kategorií
Popis: Zobrazení seznamu kategorií v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

   (a) Uživatel klikne na tlačítko „Categories“ v menu v levé straně úvodní stránky administrační části.

   (b) Systém načte a zobrazí seznam kategorií.

8. Zobrazení detailů kategorie
Popis: Zobrazení detailů kategorie v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

   (a) Uživatel klikne na tlačítko „Categories“ v menu v levé straně úvodní stránky administrační části.

   (b) Systém načte a zobrazí seznam kategorií.

   (c) Uživatel zvolí kategorii a klikne na ni.

   (d) Systém povolí tlačítko „Detail“ dole pod tabulkou se seznamem kategorií.

   (e) Uživatel klikne na tlačítko „Detail“.

   (f) Systém načte a otevře stránku s detaily zvolené kategorie.

9. Vytvoření nové kategorie
Popis: Vytvoření nové kategorie v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

   (a) Uživatel klikne na tlačítko „Categories“ v menu v levé straně úvodní stránky administrační části.

   (b) Systém načte a zobrazí seznam kategorií.

   (c) Uživatel klikne na tlačítko „New“ pod tabulkou se seznamem kategorií.

   (d) Systém načte a otevře formulář pro vytvoření nové kategorie.

   (e) Uživatel zadá potřebné údaje o kategorii.

   (f) Uživatel potvrdí uložení údajů pomocí tlačítka „Save“.

   (g) Systém zkontroluje, jestli jsou všechny údaje v pořádku.

   (h) Systém vytvoří novou kategorii.

(i) Systém přesměruje uživatele zpátky na seznam kategorií.

10. Úpravy kategorie
Popis: Úpravy údajů kategorie v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Categories“ v menu v levé straně úvodní stránky administrační části.
(b) Systém načte a zobrazí seznam kategorií.
(c) Uživatel zvolí kategorii v seznamu a klikne na ni.
(d) Systém povolí tlačítko „Edit“ dole pod tabulkou se seznamem kategorií.
(e) Uživatel klikne na tlačítko „Edit“.
(f) Systém načte a otevře formulář s údaji o kategorii.
(g) Uživatel změní potřebné údaje.
(h) Uživatel potvrdí změnu údajů kliknutím na tlačítko „Save“.
(i) Systém zkontroluje, jestli jsou všechny údaje v pořádku.
(j) Systém uloží úpravy kategorie.
(k) Systém přesměruje uživatele zpátky na seznam kategorií.

11. Odstranění kategorie
Popis: Odstranění kategorie v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Categories“ v menu v levé straně úvodní stránky administrační části.
(b) Systém načte a zobrazí seznam kategorií.
(c) Uživatel zvolí kategorii v seznamu a klikne na ni.
(d) Systém povolí tlačítko „Edit“ dole pod tabulkou se seznamem kategorií.
(e) Uživatel klikne na tlačítko „Edit“.
(f) Systém zobrazí dialogové okno, aby uživatel potvrdil odstranění.
(g) Uživatel klikne na tlačítko „Yes“ a potvrdí odstranění.
(h) Systém odstraní vybranou kategorii.
(i) Systém zavře dialogové okno.

12. Zobrazení seznamu objednávek
Popis: Zobrazení seznamu objednávek v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Orders“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam objednávek.

13. Zobrazení detailů objednávky
Popis: Zobrazení detailů objednávky v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Orders“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam objednávek.

(c) Uživatel zvolí objednávku v seznamu a klikne na ni.

(d) Systém povolí tlačítko „Detail“ dole pod tabulkou se seznamem kategorií.

(e) Uživatel klikne na tlačítko „Detail“.

(f) Systém načte a otevře stránku s detaily vybrané objednávky.

14. Úprava objednávky
Popis: Úprava údajů o objednávce v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Orders“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam objednávek.

(c) Uživatel zvolí objednávku v seznamu a klikne na ni.

(d) Uživatel klikne na tlačítko „Edit“.

(e) Systém načte a otevře formulář s údaji vybrané objednávky.

(f) Uživatel změní potřebné údaje.

(g) Uživatel potvrdí úpravy kliknutím na tlačítko „Save“.

(h) Systém zkontroluje, jestli jsou všechny údaje v pořádku.

(i) Systém uloží úpravy objednávky.

(j) Systém přesměruje uživatele zpátky na seznam objednávek.

15. Odstranění objednávky
Popis: Odstranění již vytvořené zákazníkem objednávky v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Orders“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam objednávek.

(c) Uživatel zvolí objednávku v seznamu a klikne na ni.

(d) Systém povolí tlačítko „Delete“ dole pod tabulkou se seznamem kategorií.

(e) Uživatel klikne na tlačítko „Delete“.

(f) Systém zobrazí dialogové okno, aby uživatel potvrdil odstranění.

(g) Uživatel klikne na tlačítko „Yes“ a potvrdí odstranění.

(h) Systém odstraní vybranou objednávku.

(i) Systém zavře dialogové okno.

16. Zobrazení seznamu uživatelských účtů
Popis: Zobrazení seznamu uživatelských účtů v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Accounts“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam uživatelských účtů.

17. Zobrazení detailů účtu
Popis: Zobrazení detailů účtu v administrační části aplikace
Aktéři: Administrátor, zaměstnanec
Scénář:

(a) Uživatel klikne na tlačítko „Accounts“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam uživatelských účtů.

(c) Uživatel zvolí účet v seznamu a klikne na něho.

(d) Systém povolí tlačítko „Detail“ dole pod tabulkou se seznamem účtů.

(e) Uživatel klikne na tlačítko „Detail“.

(f) Systém načte a otevře stránku s detaily zvoleného účtu.

18. Vytvoření nového uživatelského účtu
Popis: Vytvoření nového uživatelského účtu v administrační části aplikace
Aktéři: Administrátor
Scénář:

(a) Uživatel klikne na tlačítko „Accounts“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam uživatelských účtů.

(c) Uživatel klikne na tlačítko „New“ pod tabulkou se seznamem účtů.

(d) Systém načte a otevře formulář pro vytvoření nového účtu.

(e) Uživatel zadá potřebné údaje.

(f) Uživatel potvrdí uložení údajů pomocí tlačítka „Save“.

(g) Systém zkontroluje, jestli jsou všechny údaje v pořádku.

(h) Systém vytvoří nový účet.

(i) Systém přesměruje uživatele zpátky na seznam účtů.

19. Úprava uživatelského účtu
Popis: Úprava uživatelského účtu v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Accounts“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam uživatelských účtů.
    (c) Uživatel zvolí účet v seznamu a klikne na něho.
    (d) Systém povolí tlačítko „Edit“ dole pod tabulkou se seznamem účtů.
    (e) Uživatel klikne na tlačítko „Edit“.
    (f) Systém načte a otevře stránku formuláře pro úpravu účtu.
    (g) Uživatel změní potřebné údaje.
    (h) Uživatel potvrdí úpravy kliknutím na tlačítko „Save“.
    (i) Systém zkontroluje, jestli jsou všechny údaje v pořádku.
    (j) Systém uloží úpravy účtu.
    (k) Systém přesměruje uživatele zpátky na seznam uživatelských účtů.

20. Odstranění uživatelského účtu
Popis: Odstranění uživatelského účtu v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Accounts“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam uživatelských účtů.
    (c) Uživatel zvolí účet v seznamu a klikne na něho.
    (d) Systém povolí tlačítko „Delete“ dole pod tabulkou se seznamem účtů.
    (e) Uživatel klikne na tlačítko „Delete“.
    (f) Systém zobrazí dialogové okno, aby uživatel potvrdil odstranění.
    (g) Uživatel klikne na tlačítko „Yes“ a potvrdí odstranění.
    (h) Systém odstraní vybraný uživatelský účet.
    (i) Systém zavře dialogové okno.

21. Zobrazení seznamu rolí
Popis: Zobrazení seznamu rolí v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Roles“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam rolí.

22. Zobrazení detailů role
Popis: Zobrazení detailů role v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Roles“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam rolí.
    (c) Uživatel zvolí roli v seznamu a klikne na ni.
    (d) Systém povolí tlačítko „Detail“ dole pod tabulkou se seznamem rolí.
    (e) Uživatel klikne na tlačítko „Detail“.
    (f) Systém načte a otevře stránku s detaily zvolené role.

23. Vytvoření nové role
Popis: Vytvoření nové role v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Roles“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam rolí.
    (c) Uživatel klikne na tlačítko „New“ pod tabulkou se seznamem rolí.
    (d) Systém načte a otevře formulář pro vytvoření nové role.
    (e) Uživatel zvolí potřebná privilegia.
    (f) Uživatel potvrdí vytvoření a uložení nové role kliknutím na tlačítko „Save“.
    (g) Systém zkontroluje, jestli jsou všechny údaje v pořádku.
    (h) Systém vytvoří novou roli.
    (i) Systém přesměruje uživatele zpátky na seznam rolí.

24. Úprava role
Popis: Úprava role v administrační části aplikace
Aktéři: Administrátor
Scénář:

    (a) Uživatel klikne na tlačítko „Roles“ v menu v levé straně úvodní stránky administrační části.
    (b) Systém načte a zobrazí seznam rolí.
    (c) Uživatel zvolí roli v seznamu a klikne na ni.
    (d) Systém povolí tlačítko „Edit“ dole pod tabulkou se seznamem rolí.
    (e) Uživatel klikne na tlačítko „Edit“.
    (f) Systém načte a otevře stránku formuláře pro úpravu role.
    (g) Uživatel změní potřebné údaje.

(h) Uživatel potvrdí úpravy kliknutím na tlačítko „Save“.

(i) Systém zkontroluje, jestli jsou všechny údaje v pořádku.

(j) Systém uloží úpravy role.

(k) Systém přesměruje uživatele zpátky na seznam rolí.

25. Odstranění role
Popis: Odstranění role v administrační části aplikace
Aktéři: Administrátor
Scénář:

(a) Uživatel klikne na tlačítko „Roles“ v menu v levé straně úvodní stránky administrační části.

(b) Systém načte a zobrazí seznam rolí.

(c) Uživatel zvolí roli v seznamu a klikne na ni.

(d) Systém povolí tlačítko „Delete“ dole pod tabulkou se seznamem rolí.

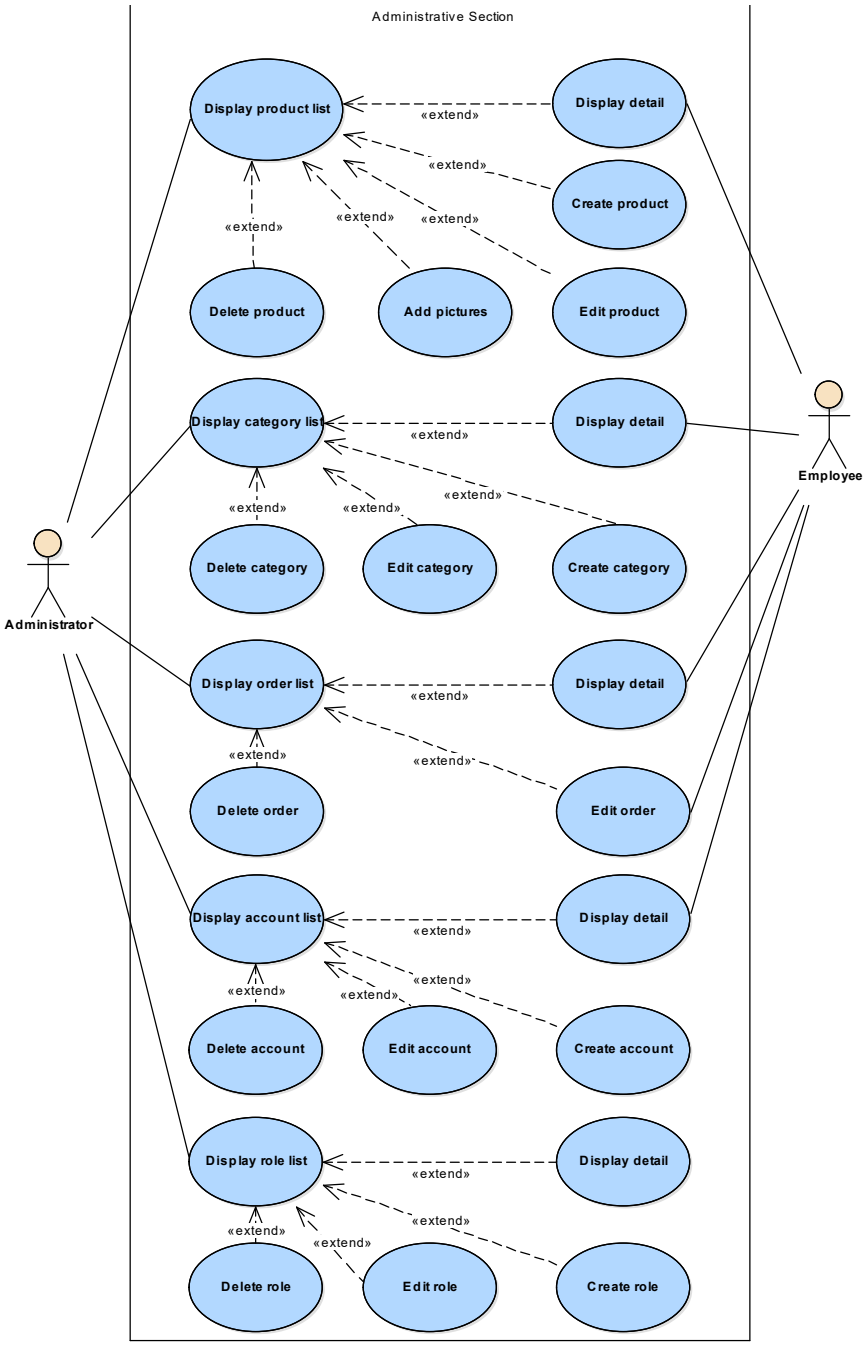
(e) Uživatel klikne na tlačítko „Delete“.

(f) Systém zobrazí dialogové okno, aby uživatel potvrdil odstranění.

(g) Uživatel klikne na tlačítko „Yes“ a potvrdí odstranění.

(h) Systém odstraní vybranou roli.

(i) Systém zavře dialogové okno.



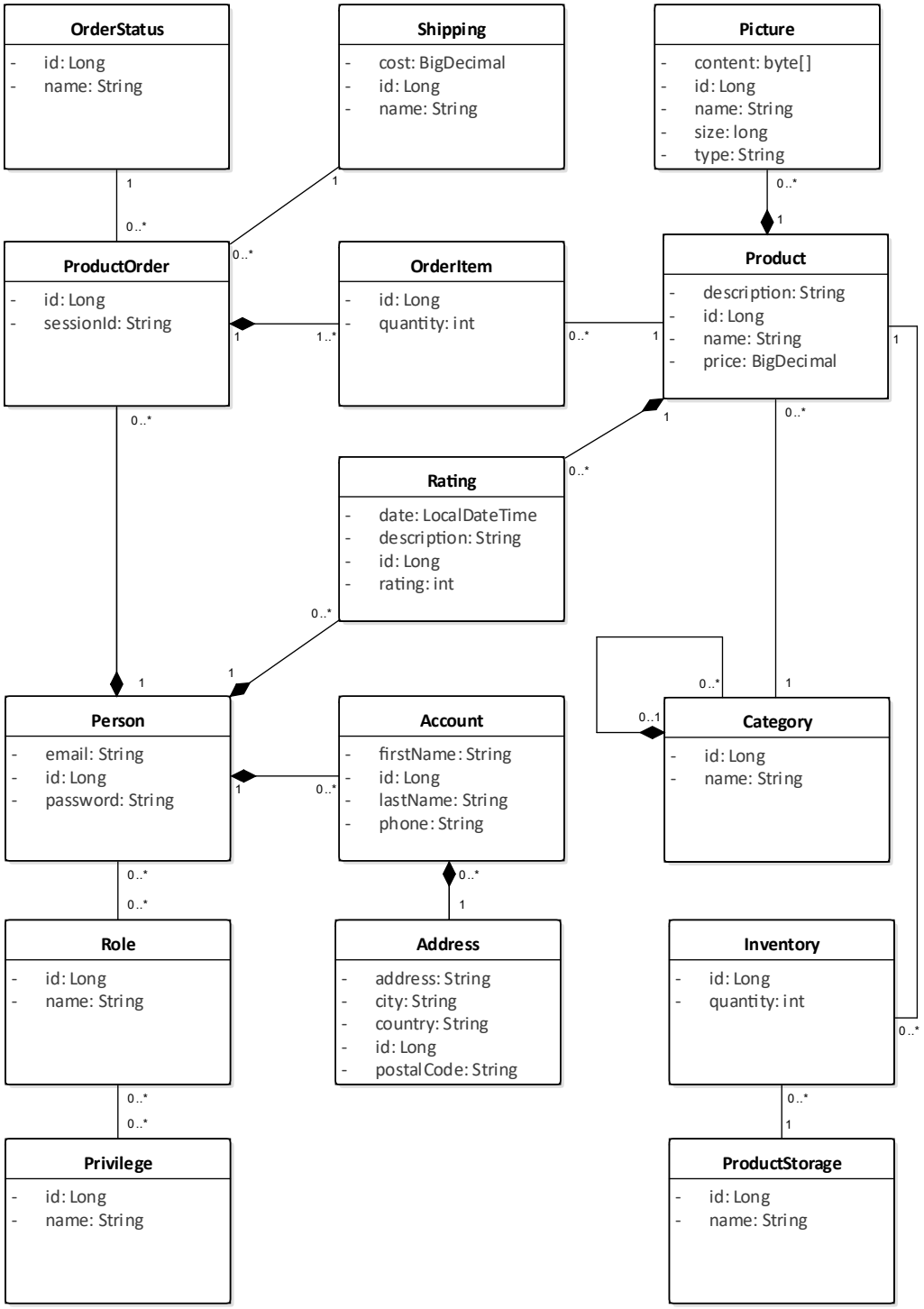
Obrázek 3.2: Případy užití administrační části

3.3 Návrh tříd

Důvodem pro založení jakéhokoliv obchodu je nakládání se zbožím. Aby se toto zboží pak dalo v obchodě jednoduše dohledat, rozdělíme ho do kategorií. Samozřejmě, že potřebujeme lidi, kteří si budou zboží vybírat a nakupovat. Zároveň obchod potřebuje mít systém objednávek, pomocí kterého zaměstnanci budou připravovat produkty k odeslání zákazníkovi. S ohledem na tyto požadavky jsem se rozhodl ve svém projektu pro použití následujících tříd (Obrázek 3.3):

- *Product* je nabízené zboží, je základní třídou obchodu. Mimo povinných parametrů, jako je název, cena, popis, bude obsahovat obrázky z třídy Picture a kategorie z třídy Category, do které bude patřit. Každý produkt se pak dá ohodnotit, k tomu nám poslouží třída Rating.
- *Category* je velmi důležitá třída, která slouží k rozdělení zboží do příslušných kategorií. Atributy této třídy jsou její název a rodič kategorie.
- *OrderStatus* představuje stav objednávky. Atributem je pouze název stavu.
- *Shipping* je typ doručení objednávky zákazníkovi. Obsahuje atributy, např. název typu doručení a jeho cena.
- *Picture* umožňuje nahrání obrázků k produktům. Obsahuje atributy, jako jsou název, obsah v bitech, typ a rozměr obrázku.
- *ProductOrder* je samotná objednávka, má atributy – unikátní číslo objednávky, ID uživatele, který tuto objednávku vytvořil, typ doručení zboží a stav této objednávky.
- *OrderItem* představuje položku objednávky. Vztahuje se ke třídám Product a ProductOrder a obsahuje atribut množství produktu v kusech.
- *Rating* je hodnocení produktu. Má atributy jako jsou ID uživatele, který daný produkt hodnotí, ID produktu, samotné hodnocení (1–5), komentář k hodnocení a datum, kdy toto hodnocení bylo vytvořeno nebo změněno.
- *Person* je uživatel dané aplikace. Jedná se jak o uživatele administrační, tak i klientské části. Třída má atributy: e-mail uživatele a heslo k uživatelskému účtu.
- *Account* je rozšířená informace o uživateli. Jedná se jak o uživatele administrační, tak i z klientské části. Třída má atributy, jako jsou jméno a příjmení uživatele, jeho telefonní číslo a adresa.
- *Role*: každý uživatel administrační části by měl mít svou roli v rámci aplikace, na základě které mu budou povolené funkce (Privilege). Role by měl vytvářet a přiřazovat administrátor. Jediným atributem této třídy je její název.
- *Address* je adresa uživatele. Jedná se jak o uživatele administrační, tak i klientské části. Třída má atributy jako jsou adresa (ulice a číslo domu), směrovací číslo, město a stát.

- *Inventory* jsou třídou, pomocí které je možné kontrolovat počet produktů na skladě a měnit jejich počet v případě dovozu produktů na sklad. Vztahuje se k třídám Product a ProductStorage a má atribut počtu produktů. V rámci daného projektu však není implementována vazba mezi danou třídou a prodaným zbožím.
- *ProductStorage* je seznam skladů, které má obchod. Jediným atributem této třídy je její název.
- *Privilege* je seznam funkcí pro uživatele administrační části, který je možné přiřadit k nějaké roli. Jediným atributem této třídy je její název.



Obrázek 3.3: Návrh tříd

3.4 Návrh adaptace

Adaptace se bude týkat pouze klientské části aplikace a pouze přihlášených uživatelů. Do obchodu budou zpracovány následující druhy adaptace:

- *Recently Viewed*: naposledy navštívené produkty. Budou se sledovat produkty, na které se zákazník naposledy díval. Pak budou tyto produkty seřazeny podle času návštěvy a zobrazeny zákazníkovi na úvodní stránce v seznamu s názvem „Recently Viewed".
- *Recommended for You*: zboží z kategorií, ze kterých si zákazník již kdysi něco kupoval. Pokud si zákazník nějaký produkt nakoupí, bude mu na úvodní stránce obchodu předložen seznam produktů ze stejné kategorie s názvem „Recommended for You". Adaptace bude probíhat za předpokladu, že se zákazníkovi líbí produkty z kategorie, ze které již nakupoval, a systém mu bude nabízet podobné zboží.
- *Best Sellers*: nejvíc prodávané zboží. Seznam, který bude předložen zákazníkovi na úvodní stránce obchodu a který bude generován na základě počtů prodaných kusů nějakého produktu. Adaptivní systém bude sledovat množství prodaných kusů nabízených produktů a pak budou nejvíce kupované produkty seřazeny v seznamu s názvem „Best Sellers".

3.5 Návrh uživatelského rozhraní

Jelikož se obchod skládá ze dvou částí, pro každou z nich navrhnu uživatelské rozhraní zvlášť pomocí Balsamiq Mockups[5]. Pro přehlednost jsou obrázky umístěny v příloze D k této práci.

3.5.1 Návrh administrační části

Administrační část je určena zaměstnancům internetového obchodu. Pro správu obchodu je třeba přiřadit nějakému zaměstnanci funkci administrátora obchodu. Administrátor pak může vytvořit libovolnou strategii, podle které budou pracovat ostatní zaměstnanci. Vytvořit strategii mu umožní tabulka rolí D.1, ve které administrátor bude schopen vytvářet nové role a přiřazovat k nim různé funkce z číselníku privilegií D.2. Role bude možné pojmenovat libovolným způsobem a přiřadit zaměstnancům určeným administrátorem D.3.

K řízení obsahu obchodu jsou určeny seznamy produktů D.4 a kategorií D.5. Zaměstnanci obchodu budou moci spravovat seznam produktů, přiřazovat je do různých kategorií se stromovou strukturou D.6. Zároveň budou moci nahrávat a mazat obrázky pro libovolné produkty D.7 a vyplňovat jejich detailnější popis. V záložce skladu bude možné spravovat různé existující sklady obchodu a přidávat do nich produkty. Také v záložce inventur bude možné kontrolovat počet produktů na skladu a měnit ho v případě dovozu produktů.

Poslední záložka v administrační části je seznam objednávek D.8. Zaměstnanci budou schopni řídit objednávky zákazníka, měnit stav objednávek D.9, měnit obsah objednávek, pokud to potřebuje zákazník D.10. Každá objednávka by měla obsahovat informace o uživateli, který tuto objednávku vytvořil, unikátní kód, způsob dopravy a položky produktů.

3.5.2 Návrh klientské části

Klientská část aplikace je určena pro potenciální zákazníky daného obchodu. Jedná se o širokou skupinu lidí a nejsou zde žádná omezení podle věku, pohlaví, místa bydliště apod. Zákazník má možnost založit si účet nebo se může přihlásit, pokud takový účet již má, avšak nemusí: pokud zůstane nepřihlášený, bude omezen v manipulaci se zbožím. Nepřihlášený zákazník nebude mít možnost přidat zboží do košíku, a tudíž nebude schopen vytvořit objednávku, ani přidat hodnocení produktům. Pro nepřihlášeného zákazníka neplatí ani adaptace. S perspektivou dalšího vývoje této aplikace bude daná omezení potřeba zrušit, protože odradí spoustu zákazníků.

Návštěvník obchodu má možnost na úvodní stránce zvolit kategorii zboží, která se mu nejvíc hodí. Pokud se jedná o přihlášeného zákazníka, na úvodní stránce se mu zobrazí následující seznamy:

- Best Sellers: nejvíc prodávané zboží.
- Recommended for You: zboží z kategorií, ze kterých si zákazník již kdysi něco kupoval.
- Recently Viewed: naposledy navštívené produkty.

Přihlášený uživatel pak může přidat produkt do košíku hned na úvodní stránce, musí k tomu kliknout na obrázek košíku umístěný pod každým produktem. Pokud si bude zákazník chtít prohlédnout detail produktu, klikne na obrázek vybraného produktu a bude přesměrován na stránku s detailním popisem. Na stránce detailů produktu si zákazník bude moci přečíst popis produktu, hodnocení produktu od jiných uživatelů a najít produkty ze stejné kategorie. Samozřejmě, že na stránce detailního popisu produktu bude přihlášený uživatel mít možnost přidat produkt do košíku nebo ohodnotit daný produkt, pokud s ním již má zkušenosti.

Pokud bude přihlášený uživatel chtít vytvořit objednávku, bude muset přejít na stránku košíku a zkontrolovat, jestli má všechny zvolené produkty v potřebném množství. Pokud to tak není, v košíku má možnost změnit počet produktů nebo smazat položku, kterou nepotřebuje. Dále zákazník pokračuje ke stránce dokončení objednávky. Na této stránce si bude zákazník muset vybrat způsob doručení a zkontrolovat celkovou částku objednávky. Uživatel má možnost vrátit se zpátky, pokud se mu něco nebude líbit nebo bude chtít svou objednávku pozměnit, a to dokud nezmáčkne tlačítko „Order“. Po zmáčknutí tlačítka „Order“ však zákazník také může zrušit objednávku nebo ji změnit, ale pouze za předpokladu, že v daném obchodě existuje zákaznický servis, kam může zavolat/napsat, a že se jedná o obchod, který umožňuje takové změny objednávek (v dnešní době skoro každý internetový obchod má svůj zákaznický servis a ve většině případů se dá objednávku měnit, dokud není expedována).

Kapitola 4

Implementace obchodu

4.1 Nastavení Spring Boot

Spring Boot[33] je framework, který umožňuje vytvářet Spring aplikace vysoké kvality s minimálním úsilím. Daný projekt se vytvářel v prostředí Intellij Idea, které umožňuje vytvořit Spring Boot projekt. Pokud však někdo bude chtít vyvíjet svůj projekt v jiném prostředí, Spring Boot má oficiální stránky[23], kde si každý může vytvořit projekt podle svých potřeb, jenž pak může být použit v jiném IDE (Integrated Development Environment). Spring Boot všechna základní nastavení přidá automaticky do projektu.

Pro automatizaci buildingu daného projektu jsem zvolil nástroj Maven a Java jazyk. Do Mavenu jsem přidal všechny knihovny potřebné pro daný projekt.

Pro vývoj webové aplikace bude potřeba přidat do souboru pom.xml následující „Web“ nastavení:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

„Web“ nastavení zároveň obsahuje potřebné RESTful service, Spring Web MVC framework a aplikační server Tomcat.

Můj projekt je založen na principu „model view controller“, proto byl použit Spring Web MVC framework (Spring MVC)[25], který tento princip realizuje. Daný framework nabízí speciální anotace, jako jsou @Controller a *@RestController*, které umožňují zpracovávat HTTP požadavky. Spring Boot nabízí automatické nastavení Spring MVC ve vytvořeném projektu, ale vždy existuje možnost tato nastavení pak změnit pomocí nově vytvořené třídy s anotací *@Configuration*.

Pro usnadnění práce s ukládáním objektů do databáze a naopak (objektově relační mapování) v Java jazyce byla použitá knihovna Java Persistence API[19], což je standardní technologie, která umožňuje propojit Java třídy s tabulkami v databázi.

Zapojení Java Persistence API do projektu vypadá následovně:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Závislost *spring-boot-starter-data-jpa* v sobě obsahuje následující knihovny:

- **Hibernate**[27]: knihovna, která nejen řeší problém komunikace Java tříd s tabulkami v databázi, ale také automatizuje SQL dotazy a umožňuje připojení do několika různých databází najednou (MySQL, PostgreSQL, Oracle atd.).
- **Spring Data JPA**[24]: knihovna, která umožňuje automatickou implementaci jednoduchých metod pro vyhledávání v databázi.
- **Spring ORM**[26]: knihovna, která umožňuje komunikaci Java tříd s tabulkami v databázi pomocí Dependency Injection.

Obvykle je třeba JPA entity sepisovat v persistence.xml. V Spring Boot projektech však není potřeba tento soubor vytvářet. Anotace *@SpringBootApplication*, která se nachází v hlavní třídě AdaptShopApplication, zajistí vyhledání Java tříd v celém projektu. Třídy s anotacemi *@Entity*, *@Embeddable* nebo *@MappedSuperclass* budou vyhledány pomocí „Entity Scanning“ a přidány do Dependency Injection.

Jako hlavní databáze projektu byla zvolena PostgreSQL, jelikož bezplatná verze Heroku podporuje pouze tuto databázi. Zapojení databáze do projektu vypadá následovně:

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

Pro testování projektu je však potřeba použít jinou databázi, aby se testovací data neukládala do hlavní databáze projektu. Pro účely testování projektu se bude hodit H2 embedded databáze. Její zapojení do projektu vypadá následovně:

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
</dependency>
```

Spring Boot automaticky nastaví veškeré knihovny potřebné pro testování projektu, což vypadá takhle:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

Spring-boot-starter-test obsahuje následující knihovny použité v tomto projektu:

- **JUnit**[31]: knihovna pro unit testy v jazyce Java.
- **Mockito**[28]: testovací knihovna pro Javu, která umožňuje vytvářet falešné objekty a metody v unit testech.

4.2 Nastavení JavaServer Faces

JavaServer Faces (JSF)[3] je specifikací jazyka Java, která podporuje vývoj uživatelského rozhraní pro webové aplikace. Vzhled webových stránek v JSF se vytváří pomocí XML souborů, které komunikují se serverovou stranou projektu pro ukládání stavu komponent.

4.2.1 Nastavení závislostí

Závislosti, které je potřeba přidat do pom.xml souboru jsou:

```
<dependency>
    <groupId>org.apache.myfaces.core</groupId>
    <artifactId>myfaces-impl</artifactId>
    <version>2.2.12</version>
</dependency>
<dependency>
    <groupId>org.apache.myfaces.core</groupId>
    <artifactId>myfaces-api</artifactId>
    <version>2.2.12</version>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
<dependency>
    <groupId>org.ocpsoft.rewrite</groupId>
    <artifactId>rewrite-servlet</artifactId>
    <version>3.4.1.Final</version>
</dependency>
<dependency>
    <groupId>org.ocpsoft.rewrite</groupId>
    <artifactId>rewrite-integration-faces</artifactId>
    <version>3.4.1.Final</version>
</dependency>
<dependency>
    <groupId>org.ocpsoft.rewrite</groupId>
    <artifactId>rewrite-config-prettyfaces</artifactId>
    <version>3.4.1.Final</version>
</dependency>
```

```
<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>6.1</version>
</dependency>
```

kde:

- myfaces-api je specifikace rozhraní JSF.
- myfaces-impl je implementace rozhraní JSF.
- tomcat-embed-jasper je nutná k tomu, aby Java Virtual Machine (JVM) mohla analyzovat a zobrazovat JSF v režimu runtime.
- rewrite-servlet, rewrite-integration-faces, rewrite-config-prettyfaces: tyto tři závislosti řeší routing v projektu a přepisují URL adresy pro Servlet a Java Web Framework. Pomocí Rewrite knihovny budou URL adresy intuitivní a odpovídající RESTful konvencím.
- primefaces: open source knihovna, která obsahuje spoustu komponent pro uživatelské rozhraní, jako jsou tabulky, inputy, labely atd.

Jakmile jsou všechny JSF závislosti na místě, je potřeba změnit proces buildingu projektu a přidat následující nastavení:

```
<build>
    <outputDirectory>src/main/webapp/WEB-INF/classes</outputDirectory>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Dané nastavení pomůže knihovně Rewrite dohledat všechny potřebné Java třídy a přepsat URL adresy správně.

4.2.2 Nastavení JSF

Pro nastavení JSF v Spring Boot projektu je potřeba vytvořit **web.xml** soubor ve složce src/main/webapp/WEB-INF/. Obvykle se v Spring Boot projektech web.xml soubor nevytváří, ale kvůli tomu, že se v daném projektu používá JSF, bude potřeba web.xml soubor vytvořit a nastavit FacesServlet, ContextLoaderListener a RequestContextListener:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
    <listener>
        <listener-class>
         org.springframework.web.context.ContextLoaderListener
</listener-class>
    </listener>
    <listener>
        <listener-class>
         org.springframework.web.context.request.RequestContextListener
</listener-class>
    </listener>
</web-app>
```

Tag servlet-mapping sděluje FacesServletu, aby zachycoval požadavky obsahující URL s příponou .jsf a zpracovával je v kontextu JSF. Ostatní dvě nastavení ContextLoaderListener a RequestContextListener jsou listenery, které integrují JSF do kontextu knihoven Spring.

Druhý XML soubor, který je potřeba vytvořit ve složce src/main/webapp/WEB-INF/, je **faces-config.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns="http://xmlns.jcp.org/xml/ns/javaee"
              xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
              version="2.2">
    <application>
        <el-resolver>
         org.springframework.web.jsf.el.SpringBeanFacesELResolver
</el-resolver>
    </application>
</faces-config>
```

V souboru faces-config.xml se registruje Expression Language resolver (ELResolver), který deleguje odpovědnost za názvy referencí na WebApplicationContext, tj. odpovědnost za názvy referencí v projektu má Spring.

Nakonec je potřeba přidat dvě beany do hlavní Java třídy AdaptShopApplication:

```
@Bean
public ServletRegistrationBean servletRegistrationBean() {
    FacesServlet servlet = new FacesServlet();
    return new ServletRegistrationBean(servlet, "*.jsf");
}

@Bean
public FilterRegistrationBean rewriteFilter() {
    FilterRegistrationBean rwFilter = new FilterRegistrationBean(
        new RewriteFilter()
    );
    rwFilter.setDispatcherTypes(EnumSet.of(
        DispatcherType.FORWARD, DispatcherType.REQUEST,
        DispatcherType.ASYNC, DispatcherType.ERROR
    ));
    rwFilter.addUrlPatterns("/*");
    return rwFilter;
}
```

Tím se nastavení JSF v Spring Boot projektu ukončí a Java třída, která používá JSF, může vypadat následovně:

```
@Controller("productList")
@Scope(value = "session")
@ELBeanName(value = "productList")
@Join(path = "/admin/products", to = "/shop/product-list.jsf")
public class ProductListController extends BaseController {
    // TODO...
}
```

- *@Controller*: Spring anotace, která definuje třídu jako controller. Název v závorkách pak bude použit v .xhtml souborech.
- *@Scope*: Spring anotace, která definuje život jedné instance.
- *@ELBeanName*: anotace z knihovny Rewrite, která přidává název beany v daném scope.
- *@Join*: anotace z knihovny Rewrite, která mění veřejný URL na URL v kontextu JSF.

4.3 Nastavení bezpečnosti

V tomto bodě pojednáme o nastaveních bezpečnosti administrační a klientské částí aplikace.

4.3.1 Administrační část

Pro zajištění bezpečnosti aplikace je nutno přidat funkci přihlášení uživatele. V tomto projektu byly jako přihlašovací údaje použity e-mail a heslo uživatele. Možnost registrace v administrační části není, protože by to nedávalo smysl. Každý uživatelský účet má být vytvořen administrátorem, který pak přihlašovací údaje posílá uživatelům (zaměstnancům obchodu). Pro zajištění dané funkce byla použita knihovna Spring Security, která podporuje přihlášení a ověření uživatelů.

Pro zapojení Spring Security do projektu je potřeba přidat následující závislost do souboru pom.xml:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

ID položek v login.xhtml mají být „username“ a „password“, protože defaultně Spring Security hledá parametry právě s těmito názvy. PrependId musí mít hodnotu „false“, jinak JSF knihovna přidá „ID“ na konci názvu parametrů.

Dále by se měla upravit výchozí nastavení bezpečnosti. K tomu je třeba vytvořit novou třídu s názvem AdministrationSecurity, která zdědí a přepíše metody z třídy WebSecurityConfigurerAdapter. WebSecurityConfigurerAdapter je třída z knihovny Spring Security, zajišťuje defaultní nastavení bezpečnosti.

Anotace, které je potřeba přidat do této třídy, jsou:

- *@Order(1)* je anotace, která definuje pořadí anotovaných tříd. Je nezbytná pro rozdělení nastavení bezpečnosti v administrační a klientské části aplikace.
- *@Configuration* je anotace, která definuje třídu AdministrationSecurity jako globální nastavení aplikace.
- *@EnableWebSecurity* je anotace, která aktivuje funkci bezpečnosti pomocí Spring Security.
- *@EnableGlobalMethodSecurity*(prePostEnabled = true) je anotace, která povoluje globální metody z knihovny Spring Security jako hasRole, hasAuthority atd.

První metoda třídy WebSecurityConfigurerAdapter, kterou přepíšeme, je **configure (HttpSecurity http)**:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable()
            .antMatcher("/admin/**").authorizeRequests()
            .anyRequest()
            .hasAnyAuthority(privilegeService.getAllNames())
            .and().formLogin()
            .loginPage("/admin/login").permitAll()
```

```
                .failureUrl("/admin/login?error=true")
                .loginProcessingUrl("/admin/login")
                .defaultSuccessUrl("/admin/products")
                .and().logout()
                .logoutUrl("/admin/logout")
                .logoutSuccessUrl("/admin/login");
}
```

Tato metoda definuje nastavení ověření uživatelských údajů (e-mail a heslo). Daná metoda obsahuje:

- http.csrf().disable(): vypíná zabezpečení od CSRF útoků v knihovně Spring Security, protože JSF knihovna již toto zabezpečení obsahuje.
- authorizeRequests().anyRequest().authenticated(): zajišťuje, aby každá akce uživatele v aplikaci vyžadovala ověření tohoto uživatele pomocí přihlašovacích údajů.
- anyRequest().hasAnyAuthority(privilegeService.getAllNames()): definuje, že uživatel, který se přihlašuje do systému, musí mít alespoň jedno privilegium.
- formLogin().loginPage("/admin/login").permitAll(): definuje stránku login.xhtml jako přihlašovací a zpřístupní ji pro libovolného uživatele.
- failureUrl("/admin/login?error=true"): nastavení, které definuje, že v případě zadání nesprávných přihlašovacích údajů bude uživatel přesměrován zpátky na stránku login.xhtml, ale s parametrem „error=true“, který umožní zobrazení chybové hlášky uživateli.
- logout().logoutSuccessUrl("/login.xhtml"): nastavení, které defaultně používá URL „/logout“ z třídy WebSecurityConfigurerAdapter a které odhlásí uživatele z aplikace. LogoutSuccessUrl pouze ukazuje, kam bude přesměrován uživatel po úspěšném odhlášení (stránka login.xhtml).

Metoda **configure(AuthenticationManagerBuilder auth)** definuje nastavení AuthenticationManager v aplikace:

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws
    Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(
        bCryptPasswordEncoder
    );
    auth.inMemoryAuthentication()
        .withUser("rudenvla@fel.cvut.cz").password("admin123")
        .roles("ADMIN").authorities(
            privilegeService.getAllAuthorities()
        );
}
```

Daná metoda obsahuje:

- **auth.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordEncoder)**: umožňuje nastavit způsob šifrování hesla. BCryptPasswordEncoder je knihovna Spring Security, pro kterou je potřeba vytvořit beanu v hlavní třídě projektu AdaptShopApplication:

```
@Bean
public BCryptPasswordEncoder bCryptPasswordEncoder() {
    return new BCryptPasswordEncoder();
}
```

- **auth.inMemoryAuthentication().withUser("rudenvla@fel.cvut.cz") .password("admin123") .roles("ADMIN").authorities(privilegeService.getAllAuthorities())**: defaultní nastavení přihlašovacích údajů pro administrátora.

Metoda **configure(WebSecurity web)** definuje nastavení WebSecurity:

```
@Override
public void configure(WebSecurity web) {
    web.ignoring().antMatchers("/javax.faces.resource/**");
}
```

Daná metoda obsahuje:

- **web.ignoring().antMatchers("/javax.faces.resource/**")**: umožňuje přístup statickým zdrojům z knihovny JSF bez přihlášení.

Metoda **corsConfigurationSource()** definuje nastavení CORS:

```
@Bean
CorsConfigurationSource corsConfigurationSource() {
    final UrlBasedCorsConfigurationSource source =
       new UrlBasedCorsConfigurationSource();
    CorsConfiguration config =
       new CorsConfiguration().applyPermitDefaultValues();
    config.addAllowedOrigin("*");
    config.addExposedHeader("Authorization");
    config.addExposedHeader("Content-Type");
    config.addExposedHeader("Accept");
    config.addAllowedMethod("OPTIONS");
    config.addAllowedMethod("POST");
    config.addAllowedMethod("PUT");
    config.addAllowedMethod("PATCH");
    config.addAllowedMethod("DELETE");
    source.registerCorsConfiguration("/**", config);
    return source;
}
```

Daná metoda obsahuje:

- **config.addAllowedOrigin("*")**: umožní zavolat backend z jakékoliv domény.
- **config.addExposedHeader("Authorization")**: v hlavičce odpovědi na požadavek povolí unikátní token uživatele, který pak bude nutný v klientské části obchodu.
- **config.addAllowedMethod("OPTIONS"/"POST"/"PUT"/"PATCH"/"DELETE")**: seznam povolených http požadavků na backend.

4.3.2 Klientská část

Hlavním aspektem bezpečnosti v klientské části aplikace je zase zajištění funkce přihlášení. Stejně jako v administrační části, byl i zde zvolen e-mail a heslo uživatele jako přihlašovací údaje.

Pro implementaci dané funkce je v první řadě potřeba přidat do souboru pom.xml knihovnu JWT (JSON Web Token)[2]:

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.0</version>
</dependency>
```

Dále je potřeba vytvořit třídu, která bude ověřovat přihlašovací údaje uživatele a vytvářet unikátní token (JwtAuthenticationFilter):

```
public class JwtAuthenticationFilter extends
    UsernamePasswordAuthenticationFilter {

    private final AuthenticationManager authenticationManager;

    public JwtAuthenticationFilter(
        AuthenticationManager authenticationManager
    ) {
        this.authenticationManager = authenticationManager;
    }

    @Override
    public Authentication attemptAuthentication(
        HttpServletRequest request, HttpServletResponse response
        ) throws AuthenticationException {
        try {
            LoginDto credentials = new ObjectMapper().readValue(
                request.getInputStream(), LoginDto.class
            );

            return authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(
```

```
                    credentials.getEmail(),
                    credentials.getPassword(),
                    Collections.emptyList()
                )
            );
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected void successfulAuthentication(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain chain,
        Authentication authResult
    ) {
        String token = Jwts.builder()
            .setSubject(
                ((User) authResult.getPrincipal()).getUsername()
            )
            .setExpiration(
                new Date(
                    System.currentTimeMillis() + EXPIRATION_TIME
                )
            )
            .signWith(SignatureAlgorithm.HS512, SECRET.getBytes())
            .compact();
        response.addHeader(HEADER_STRING, TOKEN_PREFIX + token);
    }
}
```

Třída JwtAuthenticationFilter dědí z UsernamePasswordAuthenticationFilter třídy a přepisuje z ní dvě následující metody:

- **attemptAuthentication**: v této metodě je potřeba vytáhnout přihlašovací údaje z JSONu a přidat je do AuthenticationManager.
- **successfulAuthentication**: po úspěšném přihlášení uživatele se vytváří unikátní token.

Dále je pro ověření tokenu, který bude obsažen v hlavičce HTTP požadavku, nutné vytvořit ještě jednu třídu (JwtAuthorizationFilter):

```
public class JwtAuthorizationFilter extends
    BasicAuthenticationFilter {

    public JwtAuthorizationFilter(
```

```
        AuthenticationManager authManager
    ) {
        super(authManager);
    }

    @Override
    protected void doFilterInternal(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain chain
    ) throws IOException, ServletException {
        String header = request.getHeader(HEADER_STRING);

        if (header == null || !header.startsWith(TOKEN_PREFIX)) {
            chain.doFilter(request, response);
        } else {
            UsernamePasswordAuthenticationToken authentication =
                getAuthentication(request);
            SecurityContextHolder.getContext().setAuthentication(
                authentication
            );
            chain.doFilter(request, response);
        }
    }

    private UsernamePasswordAuthenticationToken getAuthentication(
        HttpServletRequest request
    ) {
        String token = request.getHeader(HEADER_STRING);
        if (token != null) {
            String user = Jwts.parser()
                    .setSigningKey(SECRET.getBytes())
                    .parseClaimsJws(token.replace(TOKEN_PREFIX, ""))
                    .getBody()
                    .getSubject();

            if (user != null) {
                return new UsernamePasswordAuthenticationToken(
                    user, null, new ArrayList<>()
                );
            }
            return null;
        }
        return null;
    }
}
```

Třída JwtAuthorizationFilter dědí z BasicAuthenticationFilter třídy a přepisuje z ní metodu getAuthentication, která parsuje unikátní token z hlavičky HTTP požadavku a validuje přihlašovací údaje uživatele v daném tokenu.

Dále je potřeba integrovat filtry JwtAuthorizationFilter a JwtAuthenticationFilter do Spring Boot. K tomu vytvoříme novou třídu UserSecurity, která bude dědit z třídy WebSecurityConfigurerAdapter:

```
@Order(2)
@Configuration
@EnableWebSecurity
public class UserSecurity extends WebSecurityConfigurerAdapter {

    private final UserDetailsService userDetailsService;
    private final BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired
    public UserSecurity(
        UserDetailsService userDetailsService,
        BCryptPasswordEncoder bCryptPasswordEncoder
    ) {
        this.userDetailsService = userDetailsService;
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
    }

    @Override
    protected void configure(HttpSecurity http) throws
        Exception {
        http.cors().and().csrf().disable().authorizeRequests()
                .antMatchers(SIGN_UP_URL).permitAll()
                .antMatchers(PRODUCTS_URL).permitAll()
                .antMatchers(CATEGORIES_URL).permitAll()
                .antMatchers(PRODUCT_URL).permitAll()
                .anyRequest().authenticated()
                .and()
                .addFilter(
                    new JwtAuthenticationFilter(
                        authenticationManager()
                    )
                )
                .addFilter(
                    new JwtAuthorizationFilter(
                        authenticationManager()
                    )
                )
                .sessionManagement().sessionCreationPolicy(
                    SessionCreationPolicy.STATELESS
```

```
                );
    }

    @Override
    protected void configure(
            AuthenticationManagerBuilder auth
    ) throws Exception {
        auth.userDetailsService(userDetailsService).
        passwordEncoder(bCryptPasswordEncoder);
        auth.inMemoryAuthentication()
                .withUser("rudenvla@fel.cvut.cz")
                .password("user111")
                .roles("USER");
    }
}
```

Třída UserSecurity přepíše z třídy WebSecurityConfigurerAdapter dvě následující metody:

- **configure(HttpSecurity http)**: definuje nastavení dvou skupin endpointů: veřejně přístupných a zabezpečených. Tj. registrace, seznam produktů, seznam kategorií a detail produktu jsou veřejné endpointy, ostatní jsou zabezpečené a vyžadují přihlášení uživatele.
- **configure(AuthenticationManagerBuilder auth)**: umožní nastavit způsob šifrování hesla. K šifrování hesla použijeme BCryptPasswordEncoder z knihovny Spring Security. BCryptPasswordEncoder je knihovna, pro kterou je třeba vytvořit beanu v hlavní třídě projektu AdaptShopApplication:

  ```
  @Bean
  public BCryptPasswordEncoder bCryptPasswordEncoder() {
      return new BCryptPasswordEncoder();
  }
  ```

4.4 Nastavení knihovny Adaptive System Framework (ASF)

4.4.1 Model

Každý model v projektu dědí abstraktní generickou třídu AbstractPersistable z balíčku **cz.cvut.fel.asf.persistence.jpa**, která poskytuje standardní metody pro entity, jako jsou:

- mapování ID, gettery a settrery,
- toString, equals, hashCode.

4.4.2 Repository

Každé rozhraní repositáře v projektu dědí generické rozhraní IDAO z balíčku **cz.cvut.fel.asf.persistence**, které poskytuje nejčastěji používané metody pro komunikaci s databází:

- findById,
- findAll,
- update,
- delete.

Každý repositář dědí abstraktní generickou třídu AbstractDAO z balíčku **cz.cvut.fel.asf.persistence.jpa**, která implementuje standardní a nejčastěji používané metody z rozhraní IDAO.

Zároveň každá třída implementuje generické rozhraní ICanDelete z balíčku **cz.cvut.fel.asf.persistence**, které obsahuje metodu checkCanDelete. Pomocí metody checkCanDelete a abstraktní třídy BusinessCheckResult z balíčku **cz.cvut.fel.asf.businesschecks** je možné nastavit, za kterých podmínek lze smazat entitu z databáze. Pokud dojde k chybě během odstranění entity, třída BusinessCheckResult tuto chybu chytne a může pak vypsat chybovou hlášku.

4.4.3 Controller

Každý controller dědí abstraktní třídu AbstractController z balíčku **cz.cvut.fel.asf.web.primefaces**, která dědí abstraktní třídu AbstractSuperController ze stejného balíčku. Tyto třídy zpracovávají chyby z BusinessCheckResult třídy a přidávají chybové hlášky do FacesContext.

4.4.4 Adaptace

Hlavní třída, která byla použita pro zavedení adaptace, je AdaptationManager z balíčku **cz.cvut.fel.asf.adapt.gomawe**. AdaptationManager je továrna, která obsahuje 3 typy modelu adaptace:

- User Model: adaptace obsahu v závislosti na chování uživatele, např. na počtu kliků na stejný link atd.
- User Profile: adaptace obsahu stránek podle nastavených preferencí uživatele.
- Content Unit Model: adaptace podle kontentu, např. nejlépe prodávané produkty atd.

Aby AdaptationManager začal fungovat, je potřeba vytvořit beany v hlavní třídě projektu AdaptShopApplication:

```
@Bean
public IUserModelAttributeDAO userModelAttributeDAO() {
    return new TableUserModelAttributeDAOImpl();
}

@Bean
public IUserProfileAttributeDAO userProfileAttributeDAO() {
    return new TableUserProfileAttributeDAOImpl();
}

@Bean
public IContentUnitModelAttributeDAO
        contentUnitModelAttributeDAO() {
    return new TableContentUnitModelAttributeDAOImpl();
}

@Bean
public AccountRepository accountRepository() {
    return new AccountRepositoryImpl();
}

@Bean
public IDAO userDAO() {
    return new PersonRepositoryImpl(accountRepository());
}

@Bean
public String userModelImplementationClass() {
    return AdaptUserModel.class.getName();
}

@Bean
public String contentUnitModelImplementationClass() {
    return AdaptContentModel.class.getName();
}

@Bean
public AdaptationManager adaptationManager() throws
    InstantiationException {
    AdaptationManager adaptationManager = new AdaptationManager();
    adaptationManager.setUserModelAttributeDAO(
        userModelAttributeDAO()
    );
    adaptationManager.setUserProfileAttributeDAO(
        userProfileAttributeDAO()
    );
```

```
        adaptationManager.setContentUnitModelAttributeDAO(
            contentUnitModelAttributeDAO()
        );
        adaptationManager.setUserDAO(userDAO());
        adaptationManager.setUserModelImplementationClass(
            userModelImplementationClass()
        );
        adaptationManager.setContentUnitModelImplementationClass(
            contentUnitModelImplementationClass()
        );
        return adaptationManager;
    }
```

Dále je třeba vytvořit tabulky v databázi podle entit nacházejících se v knihovně ASF. První adaptace, která bude přidána, je **„Nedávno prohlédnuté produkty“**, tj. první tabulka, kterou je potřeba vytvořit, bude mít název „user_model_attribute“ z třídy TableUserModelAttribute z balíčku **cz.cvut.fel.asf.adapt.gomawe.storage.jpa**:

```
CREATE TABLE IF NOT EXISTS user_model_attribute (
  id_user_model_attribute BIGSERIAL PRIMARY KEY,
  id_user VARCHAR(255),
  domain_model_class_name VARCHAR(255),
  domain_model_instance_id VARCHAR(255),
  attribute_name VARCHAR(255),
  attribute_value VARCHAR(255)
);
```

Další typ adaptace je **„Doporučeno přímo pro vás“**. Entita zůstává stejná jako v předchozím typu adaptace a tabulka pro ni již existuje.

Poslední typ adaptace je **„Nejvíc prodávané produkty“**, pro kterou bude potřeba vytvořit novou tabulku podle třídy TableContentUnitModelAttribute z balíčku **cz.cvut.fel.asf.adapt.gomawe.storage.jpa**:

```
CREATE TABLE IF NOT EXISTS content_unit_model_attribute (
  id_content_unit_model_attribute BIGSERIAL PRIMARY KEY,
  domain_model_class_name VARCHAR(255),
  domain_model_instance_id VARCHAR(255),
  attribute_name VARCHAR(255),
  attribute_value VARCHAR(255)
);
```

Poslední část nastavení adaptace v projektu je vytvoření tříd, které budou dědit ze tříd TableUserModel a TableContentUnitModel z balíčku **cz.cvut.fel.asf.adapt.gomawe.storage.jpa** a které zajistí komunikaci s tabulkami přidanými dříve.

- Třída AdaptUserModel:

```
public class AdaptUserModel extends TableUserModel {

    public AdaptUserModel(IUser user) {
        super(user);
    }

    public void setProductVisitTime(IPersistable object) {
        putAttributeValue(
            object,
            PRODUCT_VISIT_TIME.getName(),
            LocalDateTime.now().toString()
        );
    }

    public void setPurchaseTime(IPersistable object) {
        putAttributeValue(
            object,
            PURCHASE_TIME.getName(),
            LocalDateTime.now().toString()
        );
    }
}
```

- AdaptContentModel:

```
public class AdaptContentModel extends TableContentUnitModel {

    public AdaptContentModel(IPersistable domainModelInstance) {
        super(domainModelInstance);
    }

    public void incrementPurchaseCount() {
        String count = getPurchaseCount();
        if (count == null || count.isEmpty()) {
            setPurchaseCount(1);
        } else {
            setPurchaseCount(Integer.valueOf(count) + 1);
        }
    }

    private String getPurchaseCount() {
        return getAttributeValue(PURCHASE_COUNT.getName());
    }

    private void setPurchaseCount(int count) {
        putAttributeValue(
```

```
                PURCHASE_COUNT.getName(),
                String.valueOf(count)
            );
        }
    }
```

Kapitola 5

Testování

5.1 Jednotkové testy a integrační testy

Pro testování přizpůsobení obsahu obchodu byly použity dva typy testů:

- **Jednotkové testy** jsou testy jednotlivých service metod a neměly by komunikovat s databází. Tyto testy kontrolují, jestli každá metoda správně zpracovává data a vrací očekávaný výsledek. V našem případě se testovaly 3 service metody, které vracely seřazený seznam produktů vytvořený na základě interakce uživatele se systémem. Každá metoda se testovala dle počtu produktů v seznamu, tj. jestli metoda vrací stejný počet produktů, který přichází z databáze, a dle pořadí produktů v seznamu, tj. jestli produkty budou seřazeny ve stejném pořadí, ve kterém přicházejí z databáze. Všechny metody, které komunikují s databází, byly namokovány pomocí knihovny Mockito a tím pádem izolovaly jednotkové testy od databáze.
- **Integrační testy** jsou testy metod repositáře, které komunikují s databází, posílají data do databáze nebo z ní data stahují. Tyto testy kontrolují, jestli každá metoda správně ukládá data do databáze a stahuje data podle očekávaného výsledku. V našem případě se testovalo šest metod: 3 metody ukládající data do databáze a 3 metody stahující data z databáze. První tři metody se testovaly podle toho, jestli nespadnou během ukládání dat do databáze. Další tři metody se testovaly podle toho, jestli budou vracet data ve správném pořadí a počtu. Pro tyto účely byla zapojena testovací in-memory databáze H2, aby se testovací data nepletla se skutečnými daty aplikace.

Po napsání testů byla zajištěna kontrola na chybu NullPointerException a správné pořadí produktů vracených z databáze. Testování celé aplikace zabírá hodně času, ale je dobrým příspěvkem do budoucna pro zajištění správného fungování metod a pro vyhnutí se velkým chybám v implementaci.

5.2 Testování adaptivního chování

Pro testování adaptivního chování probereme kroky, pomocí kterých tester může ověřit, jestli aplikace doporučuje správně. Pro zjednodušení a unifikaci vezmeme za předpoklad, že

do obchodu jsou zařazeny produkty A, B, C, D, E, F, G, H, I a J. Výběr produktu není omezen jednou kategorií, tester si může zvolit libovolné produkty ze skutečného sortimentu a přiřadit je uvedeným písmenům. Vizualizaci testování adaptivního chování je možné najít v příloze E.

5.2.1 Recently Viewed

Popis: Zobrazení nedávno prohlédnutých produktů.
Aktéři: Tester
Scénář:

1. Tester si založí nový uživatelský účet v klientské části.
2. Tester se přihlásí do klientské části aplikace pomocí přihlašovacích údajů.
3. Tester postupně prohlédne zvolené produkty v následujícím pořadí: A, B, C, D, E, F, G, H, I a J.
4. Tester se vrátí na úvodní stránku obchodu.
5. Systém by měl zobrazit seznam "Recenlty Viewed" s nedávno prohlédnutými produkty v následujícím pořadí: J, I, H, G, F, E, D, C, B, A E.2.

5.2.2 Recommended for You

Předpokladem uskutečnění testování doporučení "Recommended for You" je již založený uživatelský účet testera a přihlášení do klientské části aplikace.
Popis: Zobrazení produktů z kategorií, ze kterých si uživatel něco nakupoval
Aktéři: Tester
Scénář:

1. Tester si vloží do košíku po 1 kusu produktů B, E a F.
2. Tester přejde do košíku, kde bude mít pouze zmíněné produkty B, E a F.
3. Tester dokončí objednávku a nakoupí vybrané produkty.
4. Tester se vrátí na úvodní stránku obchodu.
5. Systém zobrazí seznam "Recommened for You" obsahující produkty z kategorií, ze kterých si tester před chvíli nakoupil produkty B, E a F E.4.

5.2.3 Best Sellers

Předpokladem uskutečnění testování doporučení "Best Sellers" je již založený uživatelský účet testera, přihlášení do klientské části aplikace a uskutečněný nákup produktů B, E a F (víz bod 5.2.2).
Popis: Zobrazení nejvíc prodávaných produktů.
Aktéři: Tester
Scénář:

1. Tester si vloží do košíku produkty A, B, C, D, E, F, G, H, I a J v následujících množstvích:
   - A - 12 kusů
   - B - 8 kusů
   - C - 14 kusů
   - D - 10 kusů
   - E - 9 kusů
   - F - 18 kusů
   - G - 5 kusů
   - H - 9 kusů
   - I - 15 kusů
   - J - 8 kusů
2. Tester přejde do košíku, kde bude mít pouze zmíněné v tomto bodě produkty v uvedených množstvích.
3. Tester dokončí objednávku a nakoupí vybrané produkty.
4. Tester se vrátí na úvodní stránku obchodu.
5. Systém zobrazí seznam "Best Sellers" s nejvíc prodávanými produkty, a to na základě obou předchozích nákupů (tj. nákup z bodu 5.2.2 se započte také). Pořadí položek v seznamu "Best Sellers" může mít podobu jedné z následujících variant:
   (a) F, I, C, A, D, E, B, H, J, G
   (b) F, I, C, A, E, D, B, H, J, G
   (c) F, I, C, A, E, D, H, B, J, G E.6
   (d) F, I, C, A, D, E, H, B, J, G

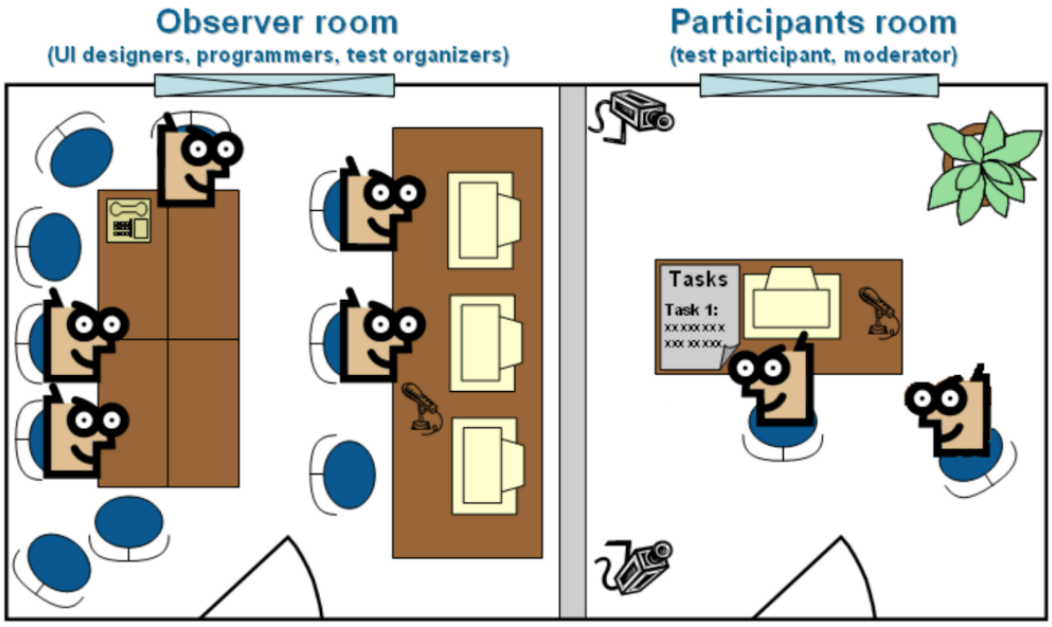
K tomu došlo na základě toho, že se počítáme s oba nákupy, tj. produktů B, E a F je o 1 kus víc, a tak nám vznikla skutečnost, že produktů D a E je stejné množství (10 kusů), produktů B a H je také stejné množství (9 kusů). Proto systém vygeneruje jednu z uvedených posloupností s tím, že položky D - E a B - H budou seřazeny náhodným způsobem, protože implementovaný systém nebere v úvahu další možné faktory mimo počtu prodaných kusů.

5.3 Testování pomocí uživatelů

Testování probíhalo dne 4. 5. 2018 od 9:00 do 15:00 v Usability Labu (ULAB), celkem se testování zúčastnilo 5 lidí, přičemž na každého participanta bylo vyhrazeno 60 minut času. Podrobnější informace o participantech a testování se dá najít v příloze C dané bakalářské práce.

5.3.1 Usability Lab

Laboratoř 5.1 se nachází ve 4. patře budovy E Fakulty elektrotechnické na Karlově náměstí v Praze. Tvoří ji 2 oddělené místnosti – testovací místnost pro participanty a sledovací místnost pro pozorovatele. Testovací místnost je vybavena 2 kamerami, 2 mikrofony, reproduktory a počítačem. Sledovací místnost je pak vybavena několika počítači, monitory, reproduktory, mikrofonem a DVD rekordérem. Na všech počítačích je nainstalován pozorovací program Morae, který umožňuje pozorovatelům sledovat a nahrávat obrazovku testovacího počítače spolu se zvukem nahrávaným mikrofonem a zároveň provádět logování v reálném čase. Ve sledovací místnosti mohou pozorovatelé zobrazit obraz z kamery na monitoru a nahrávat jej pomocí DVD rekordéru. Mikrofon ve sledovací místnosti lze zapnout/vypnout pomocí tlačítka a slouží ke komunikaci s moderátorem v testovací místnosti.



Obrázek 5.1: Usability Lab[8]

5.3.2 Výběr účastníků

Výběr účastníků probíhal mezi studenty, protože jsou na jedné straně potenciálními zaměstnanci internetových obchodů a na druhé straně jsou běžnými uživateli internetu a online obchodů.

5.3.3 Výsledky testování

Dále se podíváme na seznam nálezů, které vyplynuly z testování, ať již ze zpozorované interakce uživatele se systémem, nebo byly uživatelem zmíněny jako připomínky v post-testovém formuláři. Jednotlivým nálezům byla také přiřazena závažnost, na stupnici od 1 do 3 následujícím způsobem:

Číslo	Priorita	Popis
1	Vysoká	Problém je nutné neprodleně opravit, jelikož zabraňuje správné funkci aplikace
2	Střední	Problém může zpomalit či znesnadnit práci s aplikací
3	Nízká	Kosmetický problém, který nenarušuje práci s aplikací

Tabulka 5.1: Priority

Priorita 1:

- **Potvrzení filtrů**: Jakmile uživatel v administrační části potvrdí filtrování pomoci tlačítka „Enter“, proběhne odhlášení uživatele.
- **Povinná pole při registraci nejsou označena**: Uživatelé si stěžovali, že neví, která registrační pole jsou povinná.
- **Objednávku se dá dokončit bez výběru formy doručení zboží**: Při testování bylo zjištěno, že uživatel není povinen zadat způsob dopravy zboží před potvrzením objednávky.
- **Administrační část, záložka s objednávkami – chyba**: Nedá se otevřít detail objednávky, chyba 403.
- **Libovolný zaměstnanec v administrační části má přístup do záložky „Role“**: Každý uživatel má přístup do záložky „Role“, a může tak role měnit.

Priorita 2:

- **Nedostatečná zpětná vazba při výběru primárního obrázku zboží**: Chybí tlačítko potvrdit (pro návrat zpět). Volba primárního obrázku není popsaná.
- **Nedostatek možnosti volby množství kusů produktu přidávaného do košíku**: V klientské části při přidávání zboží do košíku není možnost rovnou zvolit potřebný počet kusů.
- **Obarvení rozkliknutých kategorií v klientské části**: Filtruje se podle poslední vybrané, ale zvýrazněné jsou všechny.
- **Vyhledávání mezi produkty v klientské části**: Uživatelé nemohli najít žádný způsob vyhledávání produktů, což vedlo k pokusům vyhledávat v seznamu pomocí funkcionality prohlížeče (ctrl+F).

Priorita 3:

- **Číslo objednávky v administrační části označeno jako „session“**: Někteří uživatelé měli zpočátku problém lokalizovat formulář pro zadání čísla objednávky – ten je totiž označen jako „session“.

- **Matoucí označení hodnocení**: Hodnocení produktů v e-shopu se uživatelé snažili hledat pod ikonkou s hvězdami, správně je ale nutné stisknout tlačítko editace
- **Vyhledávání v seznamu produktů**: Vyhledávání v seznamu produktů v e-shopu pomocí kombinace ctrl+f nefunguje i přesto, že uživatelé zadávali správné názvy
- **Zpětná vazba při vkládání zboží do košíku**: Jeden z uživatelů si při přidávání zboží do košíku stěžoval, že zpětné vazbě „něco chybí“ – zpětná vazba je realizována pouze pomocí notifikací v levém horním rohu obrazovky
- **V klientské části je tlačítko přihlášení matoucí**: Přidat popis do tlačítka „Přihlásit se“
- **Není vidět hodnocení produktů hned na stránce seznamu produktů**: Přidat hodnocení produktů pod obrázek

5.3.4 Závěr

Je patrné, že testování aplikace pomocí uživatelů je hodně přínosné a pomáhá odhalit spoustu chyb a nedostatků. Část z těchto nedostatků se mi podařilo odstranit:

- Číslo objednávky v administrační části označeno jako „session“: název “session” byl změněn na “order number”.
- Libovolný zaměstnanec v administrační části má přístup do záložky „Role“: každá záložka v administrační části je ted’ přístupná zaměstnancům pouze na základě určených administrátorem privilegií.
- Nedá se otevřít detail objednávky, chyba 403: chyba byla opravena, detail se dá otevřít.
- Po potvrzení filtrování pomoci tlačítka „Enter“, proběhne odhlášení uživatele: chyb opravena, po potvrzení filtrování pomocí tlačítka “Enter” již nedochází k odhlášení uživatele.

Zbývající část nalezených nedostatků zatím slouží jako základ pro zlepšení aplikace do budoucna.

Kapitola 6

Závěr a návrhy na zlepšení

Tato kapitola stručně shrne výsledky této bakalářské práce a předloží několik nápadů na budoucí vývoj a zlepšení implemenované aplikace.

6.1 Závěr

V této práci jsem stručně popsal podstatu adaptivních systémů a principy, na kterých může být založené jejích fungování. Zároveň jsem rozebral několik příkladů současných internetových obchodů, které podporují adaptaci obsahu svých webových stránek. Cílem práce bylo vytvoření internetového obchodu s adaptivními prvky. Pro adaptaci jsem využil sledování interakcí zákazníka se systémem a nabízení mu seznamu s nedávno prohlédnutými produkty a seznamu s produkty z kategorií, ze kterých si již zákazník kdysi nějaké zboží kupoval. Zároveň se nabízí seznam s best sellery obchodu, který se generuje v závislosti na počtu prodaných kusů. V projektu byla naimplementována funkce nákupního košíku v klientské části a správy objednávek v administrační části. Výsledná aplikace pak byla otestována několika způsoby: pomocí uživatelů a jednotkových a integračních testů. Pro testování adaptivního chování byl předložen postup kroků, které se mají udělat k ověření správnosti doporučení. Hodně přínosným bylo testování aplikace pomocí uživatelů, na základě jehož se mi pak podařilo odstranit určité nedostatky, kterých jsem si nevšiml během zpracování projektu.

Osobní přínos dané práce vidím hlavně v tom, že se v průběhu vytvoření daného projektu jsem se potkal s velkým množstvím aspektů vývoje aplikace: od komunikace se "zákazníkem" až po nasazení aplikace na vzdálený server. Dozvěděl jsem se spoustu věcí týkajících se adaptace webových stránek.

6.2 Návrhy na zlepšení

Možností pro zlepšení a rozšíření aplikace vidím velké množství. V administrační části by bylo vhodné zavedení skladu a inventur, platebního systému, sledování a přehled určitých statistik nákupu produktů apod. Do klientské části by bylo vhodné integrovat další typy doporučení. Návrhy na zlepšení existujících metod doporučení vidím v zohlednění následujících aspektů:

- Omezení počtu nabízených produktů v seznamech doporučení. Omezení je možné provést dle vlastních představ (nejčastěji cca. 10-15 produktů).
  - V případě vytvoření seznamu s best sellery by bylo vhodné se zaměřit nejenom na počet prodaných kusů, ale i na jejích hodnocení. Usnadnilo by to generování seznamů s omezeným počtem míst v případě, že některé produkty budou mít stejný počet prodaných kusů a nebude jasné, který produkt se má nabídnout jako poslední, systém dá přednost produktu, který má lepší hodnocení a doporučí právě ho.
  - Pokud hovoříme o doporučení, kdy systém nabízí zboží z kategorií, ze kterých si zákazník již nějaké produkty kupoval, omezení počtu míst v seznamu bude vhodné, aby nedocházelo k případům, kdy zákazník koupí položky ze veškerých možných kategorií a tak dostane seznam obsahující celý sortiment obchodu mimo již zakoupeného zboží. Doporučení by bylo vhodné provádět nejenom na základě zájmu zákazníka o určitou kategorii zboží, ale i s ohledem na to, z jaké kategorie si zákazník kupuje nejvíce/nejčastěji.

Literatura

[1] ALEX, B. Spring Security Reference. https://docs.spring.io/spring-security/site/docs/5.0.3.RELEASE/reference/htmlsingle/, stav z 25. 5. 2018.

[2] Auth0. *JSON Web Tokens* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://jwt.io/`>.

[3] Auth0 Inc. *Developing JSF applications with Spring Boot* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://auth0.com/blog/developing-jsf-applications-with-spring-boot/`>.

[4] BALíK, M. – JELíNEK., I. Adaptive system frameworks. In *A way to a simple development of adaptive hypermedia systems.*, 2013.

[5] Balsamiq Studios, LLC. *Balsamiq Mockups* [online]. 2008. [cit. 25. 5. 2018]. Dostupné z: <`https://balsamiq.com`>.

[6] BRUSILOVSKY, P. International Conference, AH 2000 : Trento, Italy, August 28-30, 2000 : proceedings. In *A way to a simple development of adaptive hypermedia systems.*, s. 58–59. Berlin: Springer, c2000. Lecture notes in computer science, 1892.

[7] Cult Beauty Ltd. *Cult Beauty* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://www.cultbeauty.co.uk/`>.

[8] doc. Ing. Adam Sporka, Ph.D. *Testování uživatelského rozhraní 2017/2018* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://cent.felk.cvut.cz/predmety/Y39TUR/?page=slides`>.

[9] Facebook Inc. *React* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://reactjs.org/`>.

[10] Heroku. *Heroku* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://www.heroku.com/`>.

[11] Heroku. *The Heroku CLI* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://devcenter.heroku.com/articles/heroku-cli`>.

[12] iHerb Inc. *iHerb* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://www.iherb.com/`>.

[13] MARTIN, B. – IVAN, J. Generic Ontology-based Model for Adaptive Web Environments. In *A Revised Formal Description Explained within the Context of its Implementation.* IEEE 16th International Conference on Computational Science and Engineering, 2013.

[14] Matěj Jakimov. *Multimodální doporučovací systém* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://is.muni.cz/th/s97i8/jakimov.pdf>.

[15] MERT ÇALışKAN, O. V. PrimeFaces Cookbook. In *Over 90 practical recipes to learn PrimeFaces - the rapidly evolving, leading JSF component suite.* Packt Publishing, BIRMINGHAM - MUMBAI, 2015.

[16] Node.js Foundation. *Downloads* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://nodejs.org/en/download/>.

[17] Oracla. *Java + You, Download Today* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://java.com>.

[18] Oracle. *Java SE Downloads* [online]. [cit. 25. 5. 2018]. Dostupné z: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

[19] Oracle. *Java Persistence API* [online]. [cit. 25. 5. 2018]. Dostupné z: <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>.

[20] Oracle. *What Is Platform as a Service (PaaS)?* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://www.oracle.com/cloud/platform/what-is-paas/index.html>.

[21] Petr Cvengroš. *Universal Recommender System* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://is.cuni.cz/webapps/zzp/detail/85227/?lang=en>.

[22] Pivotal Software, Inc. *Spring by Pivotal* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://spring.io/>.

[23] Pivotal Software, Inc. *SPRING INITIALIZR* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://start.spring.io/>.

[24] Pivotal Software, Inc. *Spring Data JPA* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://projects.spring.io/spring-data-jpa/>.

[25] Pivotal Software, Inc. *Spring Web MVC framework* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html>.

[26] Pivotal Software, Inc. *Object Relational Mapping (ORM) Data Access* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/orm.html>.

[27] RedHat. *Hibernate* [online]. [cit. 25. 5. 2018]. Dostupné z: <http://hibernate.org/>.

[28] Szczepan Faber and friends. *Mockito* [online]. [cit. 25. 5. 2018]. Dostupné z: <http://site.mockito.org/>.

[29] The Apache Software Foundation. *Apache Maven* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://maven.apache.org/>.

[30] The Apache Software Foundation. *Apache Maven* [online]. [cit. 25. 5. 2018]. Dostupné z: <https://maven.apache.org/download.cgi>.

[31] The JUnit Team. *JUnit 5* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://junit.org/junit5/`>.

[32] The PostgreSQL Global Development Group. *PostgreSQL* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://www.postgresql.org/`>.

[33] WALLS, C. Spring Boot in Action. Shelter Island, NY: Manning Publications, 2016.

[34] Yarn. *Installation* [online]. [cit. 25. 5. 2018]. Dostupné z: <`https://yarnpkg.com/lang/en/docs/install/#windows-stable`>.

Příloha A

Spuštění projektu lokálně

A.1 Administrační část

A.1.1 Nastavení Java na vlastním počítači

1. Nainstalovat Javu z oficiálních webových stránek Oraclu[18]
2. Otevřít "Vlastnosti počítače" pomoci pravého tlačítka myší.
3. Vybrat "Upřesnit nastavení systému".
4. V nově otevřeném okně zvolit "Proměnné prostředí".
5. Do "Systémové proměnné" přidat novou proměnnou JAVA_HOME s cestou do složky jdk.
6. Do uživatelské proměnné PATH přidat %JAVA_HOME%\bin.

Úspěšné nastavení Javy se dá zkontrolovat zadáním příkazu:

```
java -version
```

Výpis z konzole by měl vypadat následovně:

```
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```

A.1.2 Nastavení Maven na vlastním počítači

1. Maven je možné nainstalovat z oficiálních webových stránek[30]
2. Po úspěšné instalaci rozbalit archiv do libovolné složky.
3. Otevřít "Vlastnosti počítače" pomoci pravého tlačítka myší.

4. Vybrat "Upřesnit nastaveni systému".
5. V novém okně zvolit "Proměnné prostředí".
6. Do "Systémové proměnné" přidat následující tři proměnné:
   (a) M2_HOME s cestou do složky
   (b) M2 s cestou %M2_HOME%\bin
   (c) MAVEN_OPTS s hodnotami Xms256m -Xmx512m
7. Do uživatelské proměnné PATH přidat %M2%.

Správné nastavení Maven se dá zkontrolovat pomocí příkazu:

```
mvn --version
```

Výpis z konzole by měl vypadat následovně:

```
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T09
Maven home: C:\Users\GC\apache-maven-3.5.2\bin\..
Java version: 1.8.0_161, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jre1.8.0_161
Default locale: cs_CZ, platform encoding: Cp1250
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

A.1.3 Nastavení ASF lokálně

1. V projektu vytvořit složku, do které umístit .jar soubory ASF knihovny.
2. V této složce vytvořit .bat soubor, do kterého zadat následující skript:

```
call mvn install:install-file -Dfile=ASF.Core-0.4-SNAPSHOT.jar
-DgroupId=cz.cvut.fel.asf -DartifactId=asf-core -Dversion=0.4.RELEASE
-Dpackaging=jar -Djavadoc=ASF.Core-0.4-SNAPSHOT-javadoc.jar
call mvn install:install-file -Dfile=ASF.Persistence.JPA-0.4-SNAPSHOT.jar
-DgroupId=cz.cvut.fel.asf -DartifactId=asf-persistence-jpa -Dversion=0.4.RELEASE
-Dpackaging=jar -Djavadoc=ASF.Persistence.JPA-0.4-SNAPSHOT-javadoc.jar
call mvn install:install-file -Dfile=ASF.Web.Primefaces-0.4-SNAPSHOT.jar
-DgroupId=cz.cvut.fel.asf -DartifactId=asf-web-primefaces -Dversion=0.4.RELEASE
-Dpackaging=jar -Djavadoc=ASF.Web.Primefaces-0.4-SNAPSHOT-javadoc.jar
```

Daný script nainstaluje lokální knihovny ze složky s .jar soubory do Maven repositáře.

A.1.4 Závěrečná nastavení

Dále je potřeba:

1. Vytvořit novou PostgreSQL databázi.
2. Spustit v databázi database-schema.sql (je uvnitř projektu).
3. Otevřít application.properties a změnit:
   (a) spring.datasource.url
   (b) spring.datasource.username
   (c) spring.datasource.password
4. Vytvořit .jar soubor:

   ```
   mvn clean test package
   ```

5. Spustit .jar soubor:

   ```
   java -jar <path-to.jar>
   ```

A.2 Klientská část

1. Nainstalovat Node.js[16]
2. Nainstalovat Yarn[34]
3. V projektu test-projects/adapt-shop-fe spustit následující příkazy:

   ```
   yarn # nainstaluje všechny knihovny z package.json
   yarn start # spustí projekt
   ```

Příloha B

Nastavení Heroku

B.1 Administrační část

B.1.1 Nastavení knihovny ASF na Heroku

1. V hlavním projektu na úrovni složky "src" a souboru "pom.xml" vytvoříme složku s názvem "repo".

2. Přes konzoli zavoláme následující tři příkazy:

```
call mvn deploy:deploy-file -Durl=file:///path/to/yourproject/repo/
-Dfile=ASF.Core-0.4-SNAPSHOT.jar -DgroupId=cz.cvut.fel.asf
-DartifactId=asf-core -Dversion=0.4.SNAPSHOT -Dpackaging=jar
-Djavadoc=ASF.Core-0.4-SNAPSHOT-javadoc.jar
call mvn deploy:deploy-file -Durl= file:///path/to/yourproject/repo/
-Dfile=ASF.Persistence.JPA-0.4-SNAPSHOT.jar -DgroupId=cz.cvut.fel.asf
-DartifactId=asf-persistence-jpa -Dversion=0.4.SNAPSHOT -Dpackaging=jar
-Djavadoc=ASF.Persistence.JPA-0.4-SNAPSHOT-javadoc.jar
call mvn deploy:deploy-file -Durl=file:///path/to/yourproject/repo/
-Dfile=ASF.Web.Primefaces-0.4-SNAPSHOT.jar -DgroupId=cz.cvut.fel.asf
-DartifactId=asf-web-primefaces -Dversion=0.4.SNAPSHOT -Dpackaging=jar
-Djavadoc=ASF.Web.Primefaces-0.4-SNAPSHOT-javadoc.jar
```

   kde do atributu -Durl přidáme cestu do složky "repo".

3. Do pom.xml přidáme závislosti:

```
<dependency>
   <groupId>cz.cvut.fel.asf</groupId>
   <artifactId>asf-core</artifactId>
   <version>0.4.SNAPSHOT</version>
</dependency>
<dependency>
   <groupId>cz.cvut.fel.asf</groupId>
```

```
    <artifactId>asf-persistence-jpa</artifactId>
    <version>0.4.SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>cz.cvut.fel.asf</groupId>
    <artifactId>asf-web-primefaces</artifactId>
    <version>0.4.SNAPSHOT</version>
</dependency>
```

4. A custom repository:

```
<repositories>
    <repository>
        <id>project.local</id>
        <name>project</name>
        <url>file:${project.basedir}/repo</url>
    </repository>
</repositories>
```

B.1.2 Nastavení PostgreSQL databáze

1. Než uděláme deploy projektu na Heroku, založíme bezplatný účet a nainstalujeme Heroku CLI[11]

2. Po instalaci Heroku CLI otevřeme konzoli a přihlásíme se:

```
$ heroku login
Enter your Heroku credentials.
Email: java@example.com
Password: ****
```

3. Po přihlášení vytvoříme novou PostgreSQL databázi pro naši aplikaci:

```
$ heroku addons:create heroku-postgresql
```

4. Aby aplikace byla připojená ke vzdálené databázi, je potřeba změnit následující údaje v application.properties souboru:

```
spring.datasource.url=SPRING_DATASOURCE_URL
spring.datasource.username=SPRING_DATASOURCE_USERNAME
spring.datasource.password=SPRING_DATASOURCE_PASSWORD
```

   SPRING_DATASOURCE_URL, SPRING_DATASOURCE_USERNAME, SPRING_DATASOURCE_PASSWORD jsou globální proměnné na serveru Heroku, které poskytují URL a přihlašovací údaje k nově vytvořené databázi.

5. Po úspěšném vytvoření databáze pomoci příkazu:

```
$ heroku pg:psql
```

otevřeme psql konzoli a nahrajeme do vytvořené databáze všechno, co je obsazeno v souboru database-schema.sql.

B.1.3 Nasazení na Heroku

1. Před tím, než uděláme nasazení projektu na Heroku, vytvoříme bezplatný účet a nainstalujeme Heroku CLI[11]

2. Po instalaci Heroku CLI otevřeme konzoli a přihlásíme se:

```
$ heroku login
Enter your Heroku credentials.
Email: java@example.com
Password: ****
```

3. Jakmile se přihlásíme, vytvoříme Git repositář a přidáme do něj všechny soubory obsazené v projektu:

```
$ git init
$ git add .
$ git commit -m "first commit"
```

4. Než uděláme „push“ projektu, vytvoříme nový projekt na Heroku:

```
$ heroku create
```

5. Uděláme deploy projektu na Heroku:

```
$ git push heroku master
```

6. Otevřeme stránku aplikace pomocí příkazu:

```
$ heroku open
```

B.2 Klientská část

B.2.1 Nasazení na Heroku

Předpokladem nasazení klientské části na Heroku je existující bezplatný účet a nainstalovaný Heroku CLI. B.1.3

1. Jakmile se přihlásíme, vytvoříme Git repositář a přidáme do něj všechny soubory obsazené v projektu:

```
$ git init
$ git add .
$ git commit -m "first commit"
```

2. Než uděláme „push“ projektu, vytvoříme nový projekt na Heroku:

```
$ heroku create
```

3. Uděláme deploy projektu na Heroku:

```
$ git push heroku master
```

4. Otevřeme stránku aplikace pomocí příkazu:

```
$ heroku open
```

Příloha C

Testování

C.1 Pre-test dotazník

Před samotným testováním jsme nechali participanty vyplnit krátký dotazník, jehož účelem bylo zjištění informací o participantovi, relevantních pro testování.

Pre-test:

1. Jaký je Váš věk?
2. V jakém oboru studujete/pracujete?
3. Jaké jazyky umíte?
   - Anglicky
   - Česky
   - Německy
   - Španělsky
   - Rusky
4. Jak často pracujete s počítačem?
   - Několikrát denně
   - Několikrát týdne
   - Několikrát měsíčně
   - Nikdy
5. Jaký operační systém používáte?
   - Mac OS
   - Microsoft Windows
   - Linux
   - Jiné

6. Jak často nakupujete přes Internet?
   - Několikrát do týdne
   - Několikrát do měsíce
   - Několikrát do roku
   - Nenakupují
7. Máte zkušenost s provozem internetového obchodu?
   - Ano
   - Ne

C.2 Úkoly

Administrační část:

1. Přihlásit se do administrační části.
2. Vytvořit novou kategorii "Category 10.<participant number>"a zařadit ji do kategorie "Category 1.5".
3. Vytvořit nový produkt "Produkt 10.<participant number>"s cenou 5000. Zařadit do "Category 4.1".
4. Přesunout produkt "Product 5.1"do kategorie "Category 2.1".
5. Přidat 3 obrázky: "1", "2", "3"k produktu "Product 1.1.1"a vybrat obrázek "2"jako primární.
6. Vytvořit novou roli "TESTER"s privilegií "Vytvořit produkt"a "Smazat produkt".
7. Vyhledat objednávku číslo 1037208396, navýšit počet kusů "Product 1"o 3 a změnit stav objednávky na "SHIPPED".

Klientská část:

1. Založit nový účet.
2. Nakoupit produkty: "Product 4"- 2 kusy, "Product 9.1"- 3 kusy, "Product 1"- 1 kus.
3. Ohodnotit produkt "Product 4".
4. Odhlásit se.

Bonus
V administrační části dohledat vlastní účet a změnit role na "TESTER".

C.3 Post-test dotazník

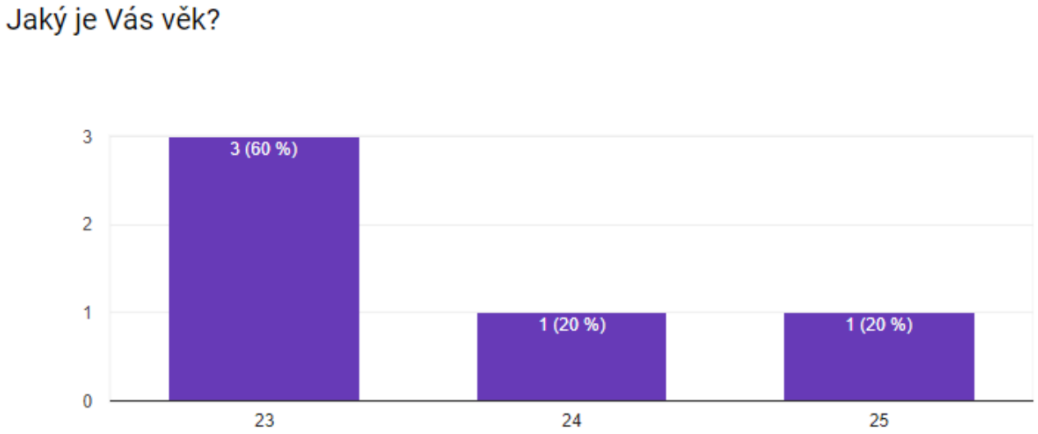
Po dokončení všech úkolů jsme nechali participanty vyplnit závěrečný dotazník, ve kterém měli vlastními slovy testovanou aplikaci zhodnotit.

Post-test:

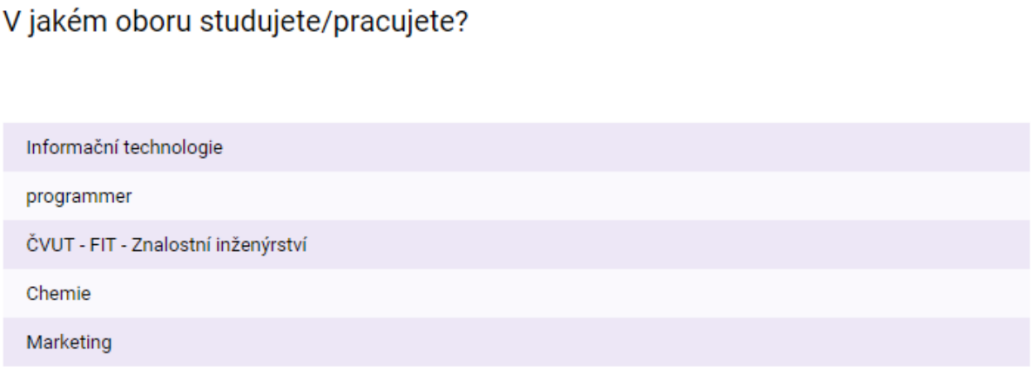
1. Měli jste nějaké potíže s vypracováním úkolů
2. Co se Vám líbilo?
3. Co byste zlepšili?

C.4 Výsledky Pre-testu

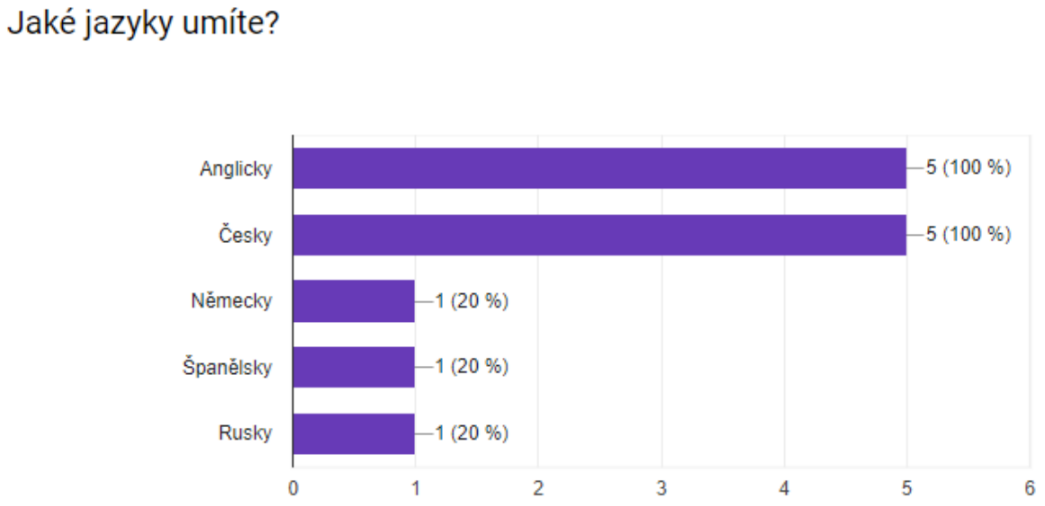
Testování se zúčastnilo celkem 5 lidí, jejichž výsledky Pre-testu vypadají následovně:



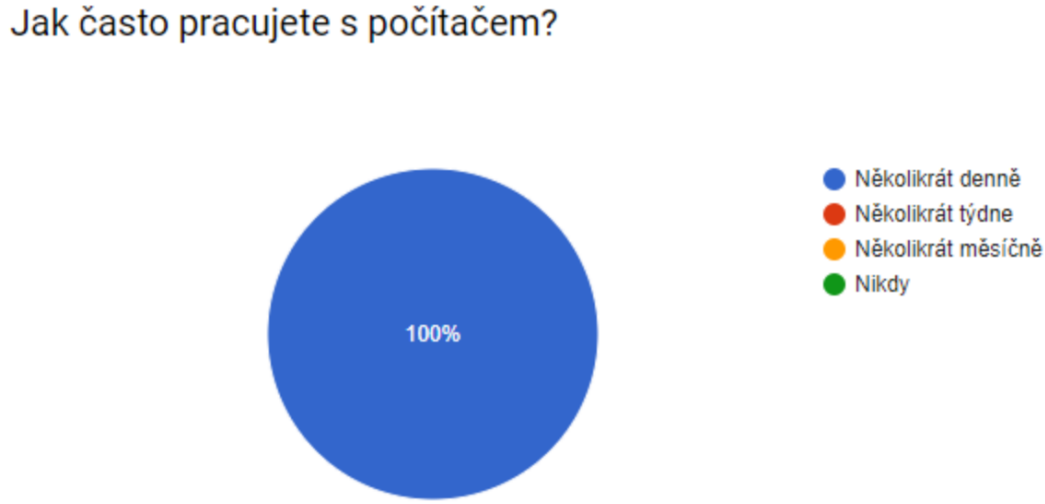
Obrázek C.1: Věk participatnů



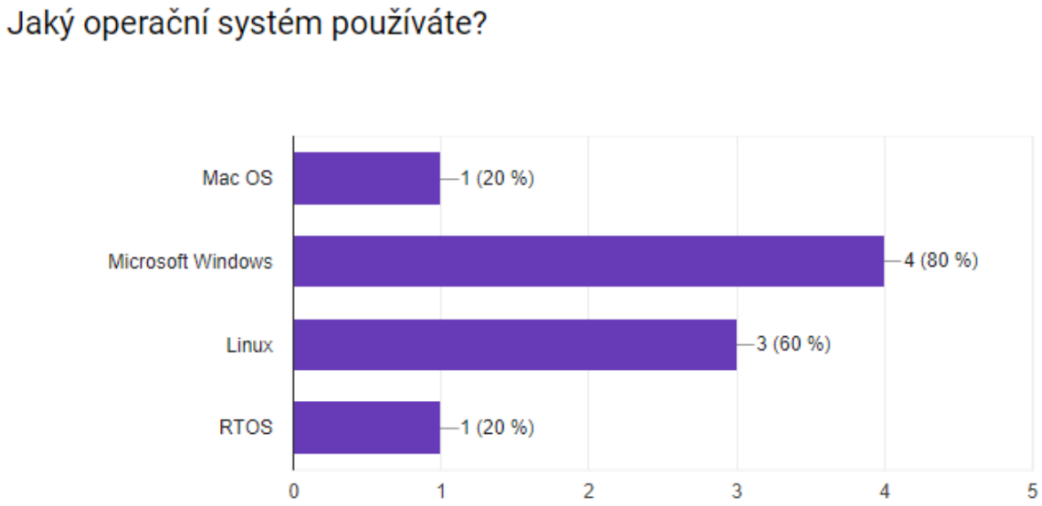
Obrázek C.2: Studijní obor



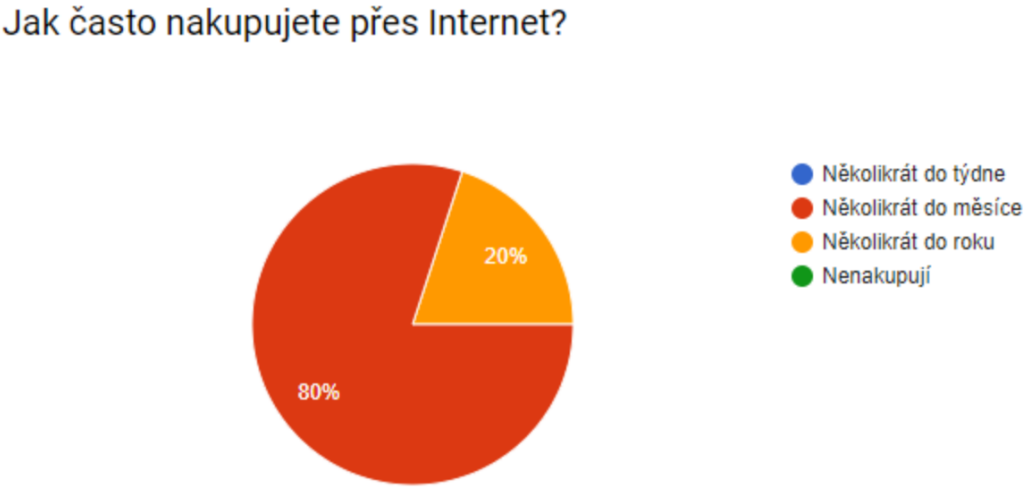
Obrázek C.3: Jazyky



Obrázek C.4: Práce s počítačem



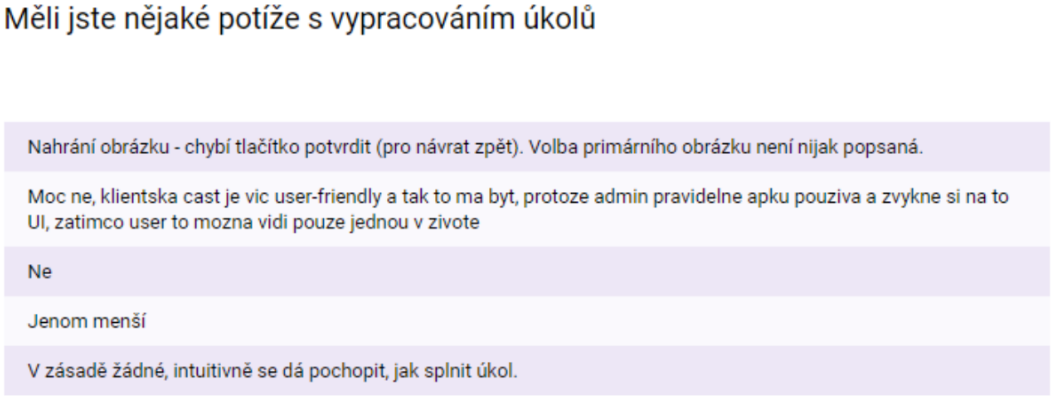
Obrázek C.5: Operační systém



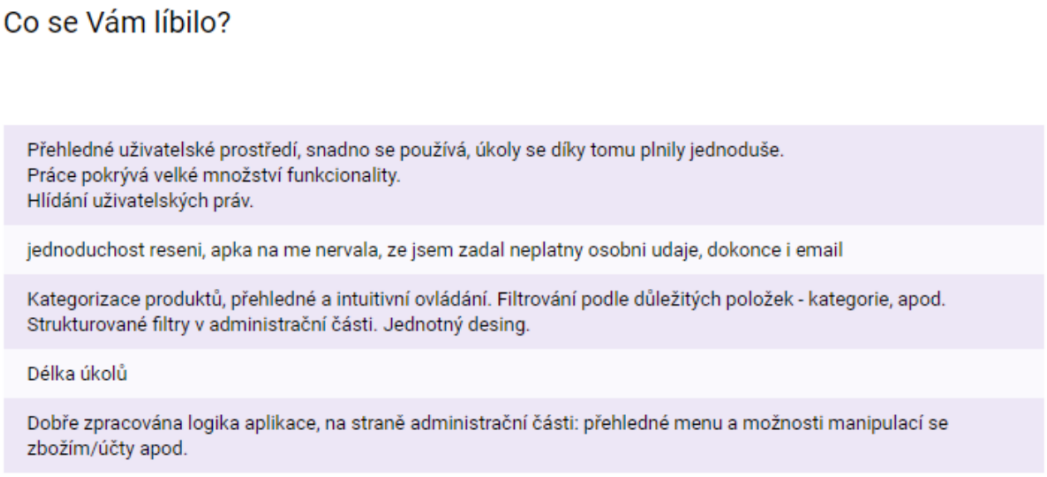
Obrázek C.6: Nákupy přes Internet

Obrázek C.7: Zkušenost s provozem internetového obchodu

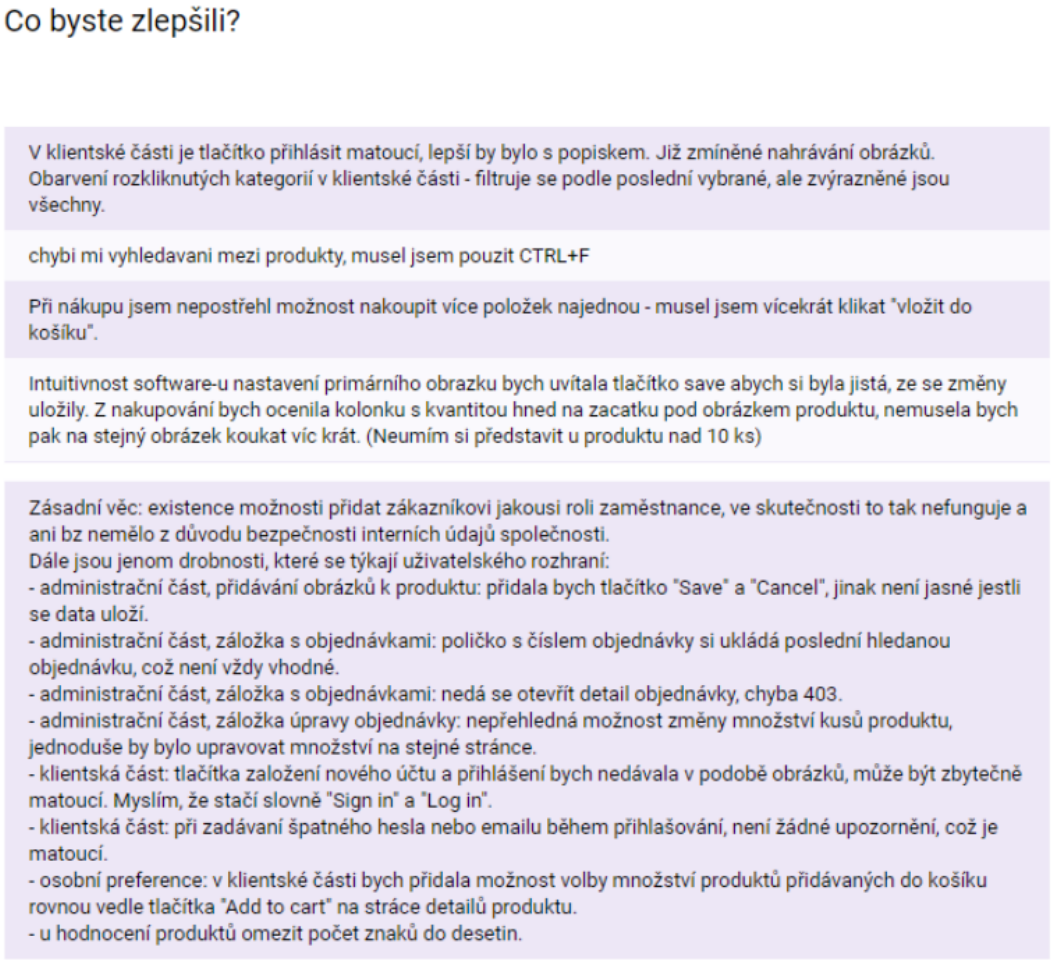
C.5 Výsledky Post-testu



Obrázek C.8: Potíže



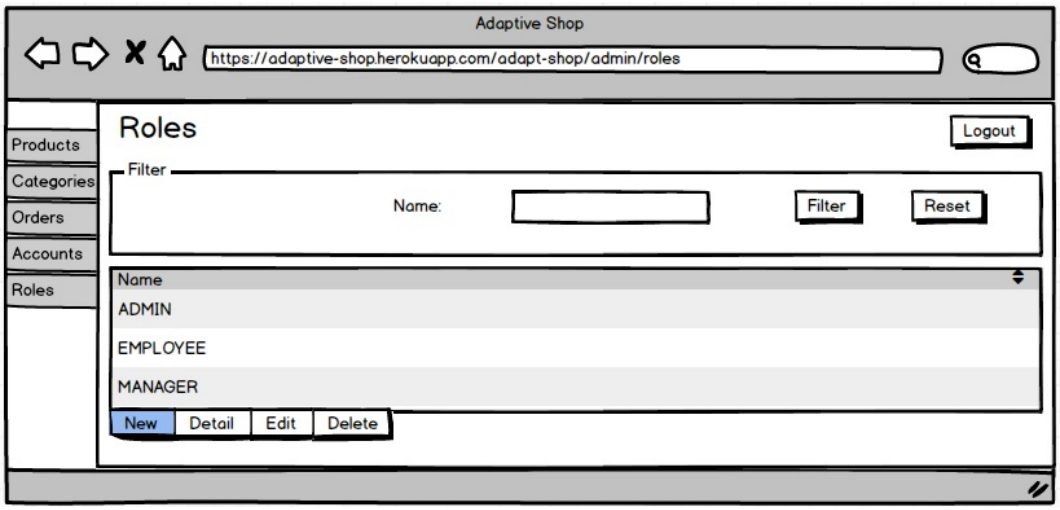
Obrázek C.9: Líbilo se



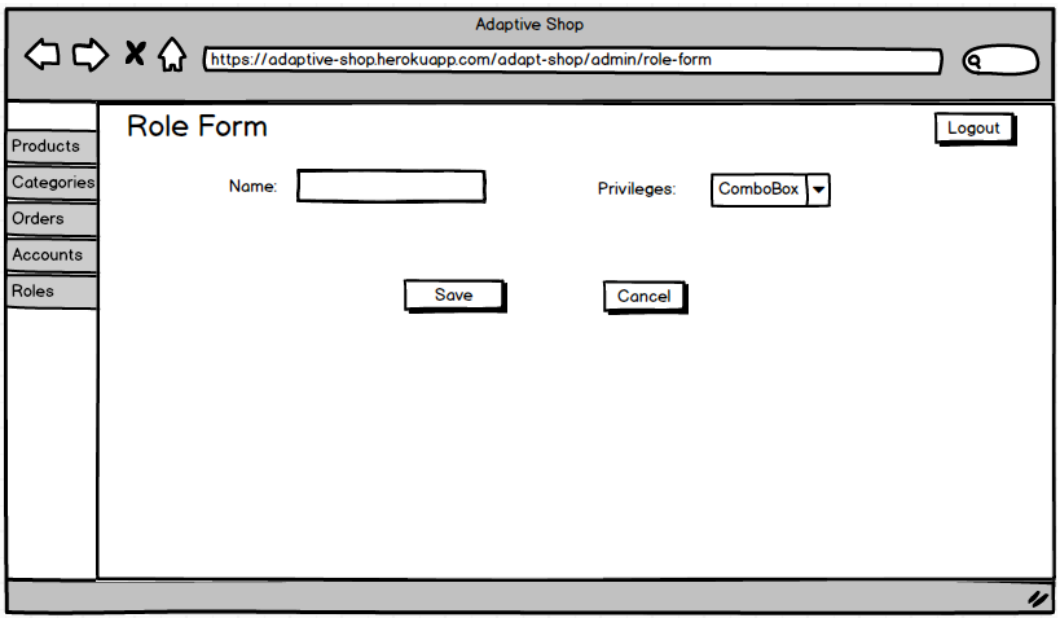
Obrázek C.10: Zlepšit

Příloha D

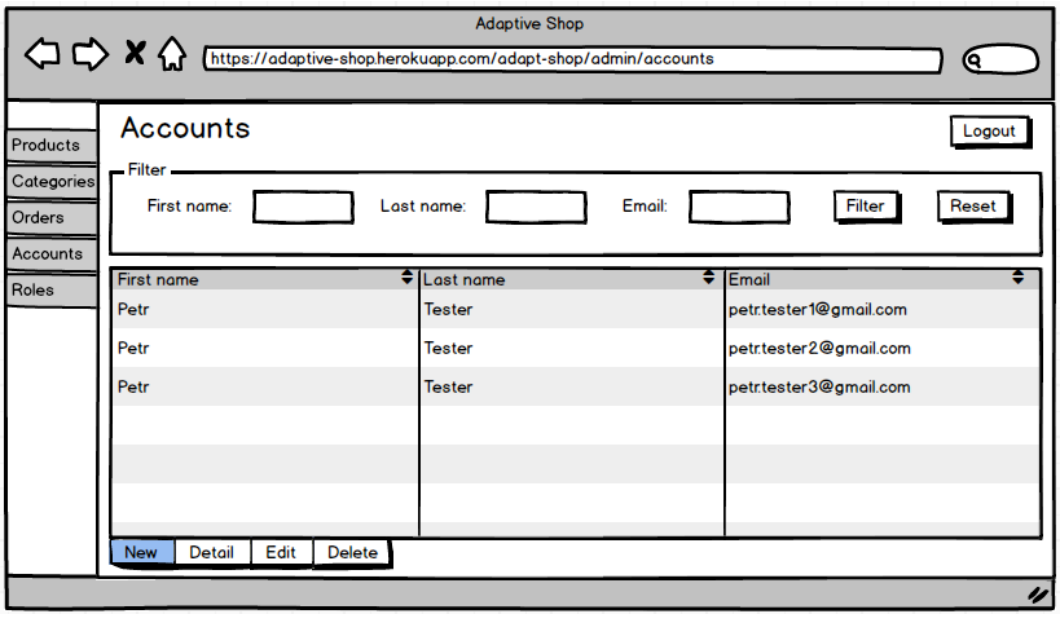
Obrázky



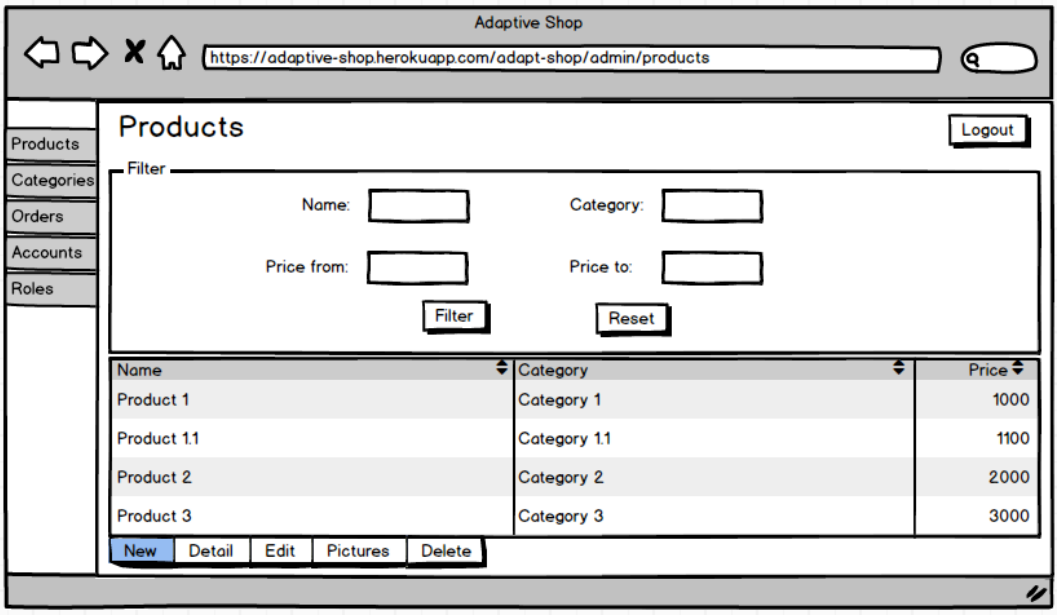
Obrázek D.1: Seznam rolí



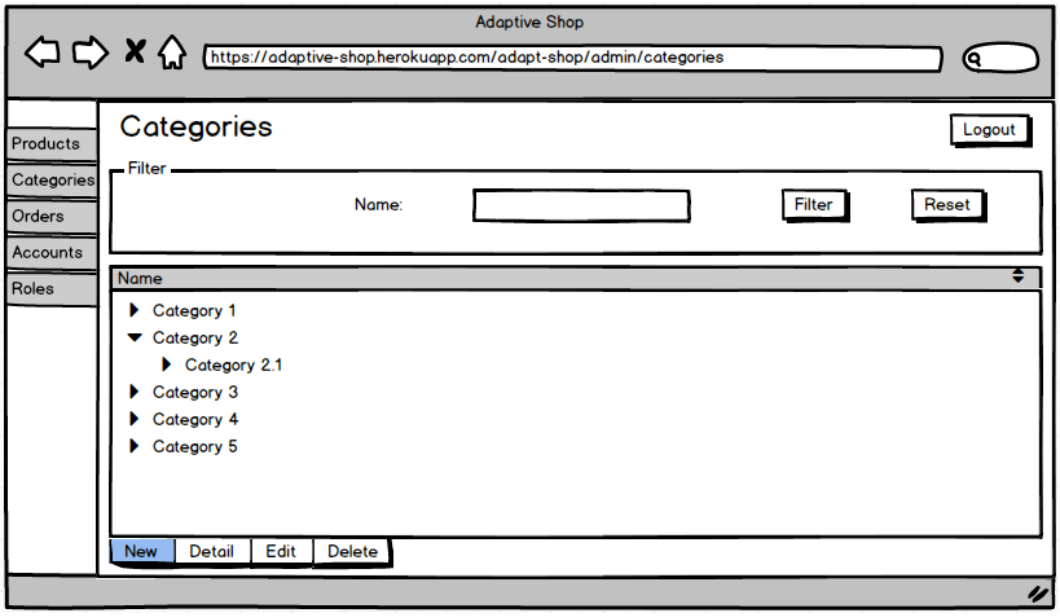
Obrázek D.2: Detail role



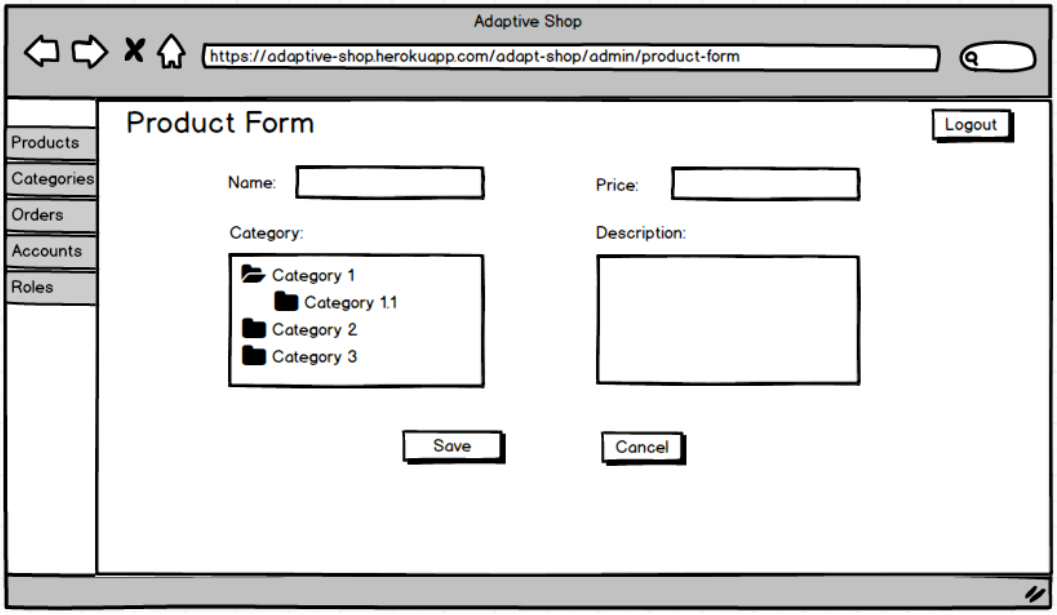
Obrázek D.3: Seznam uživatelů



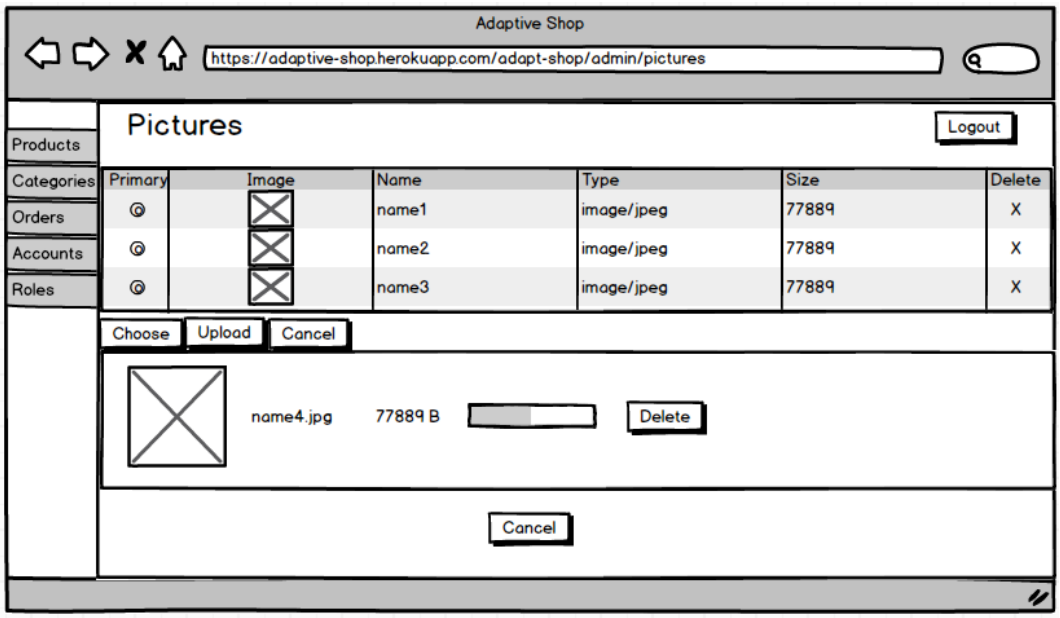
Obrázek D.4: Seznam produktů



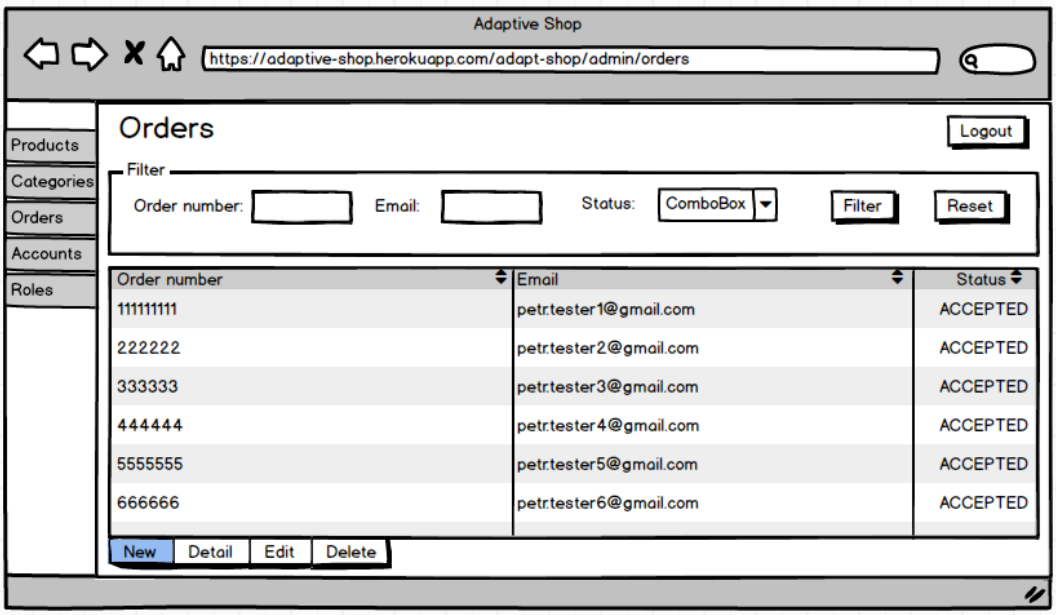
Obrázek D.5: Seznam kategorií



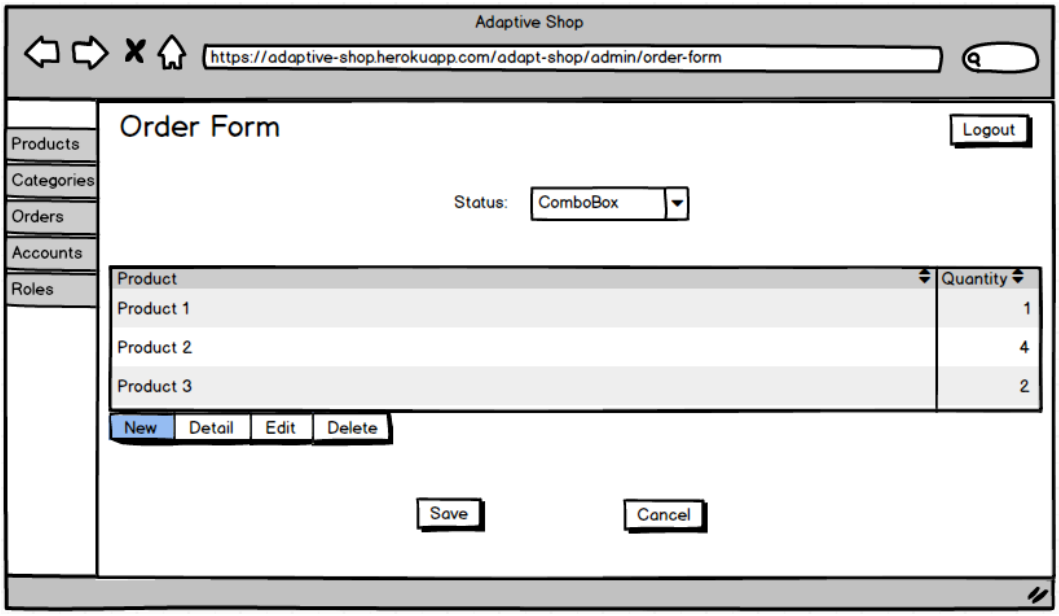
Obrázek D.6: Detail produktu



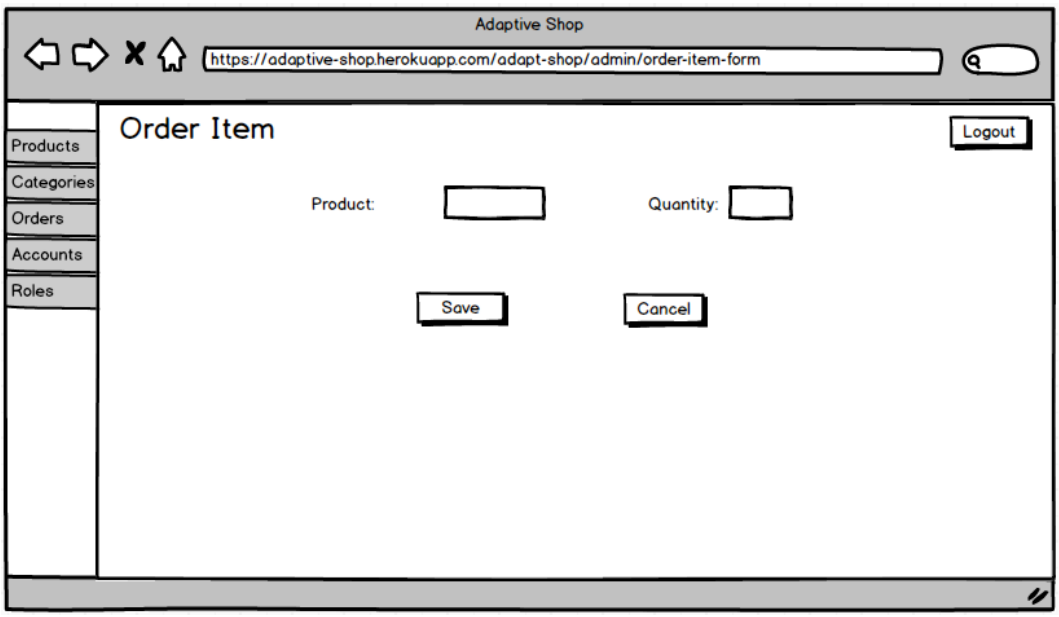
Obrázek D.7: Seznam obrázků



Obrázek D.8: Seznam objednávek



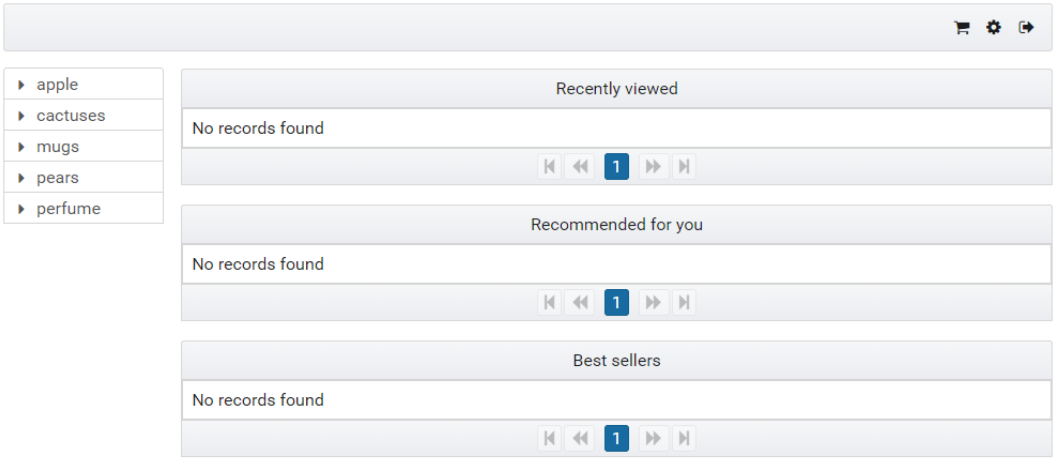
Obrázek D.9: Detail objednávky



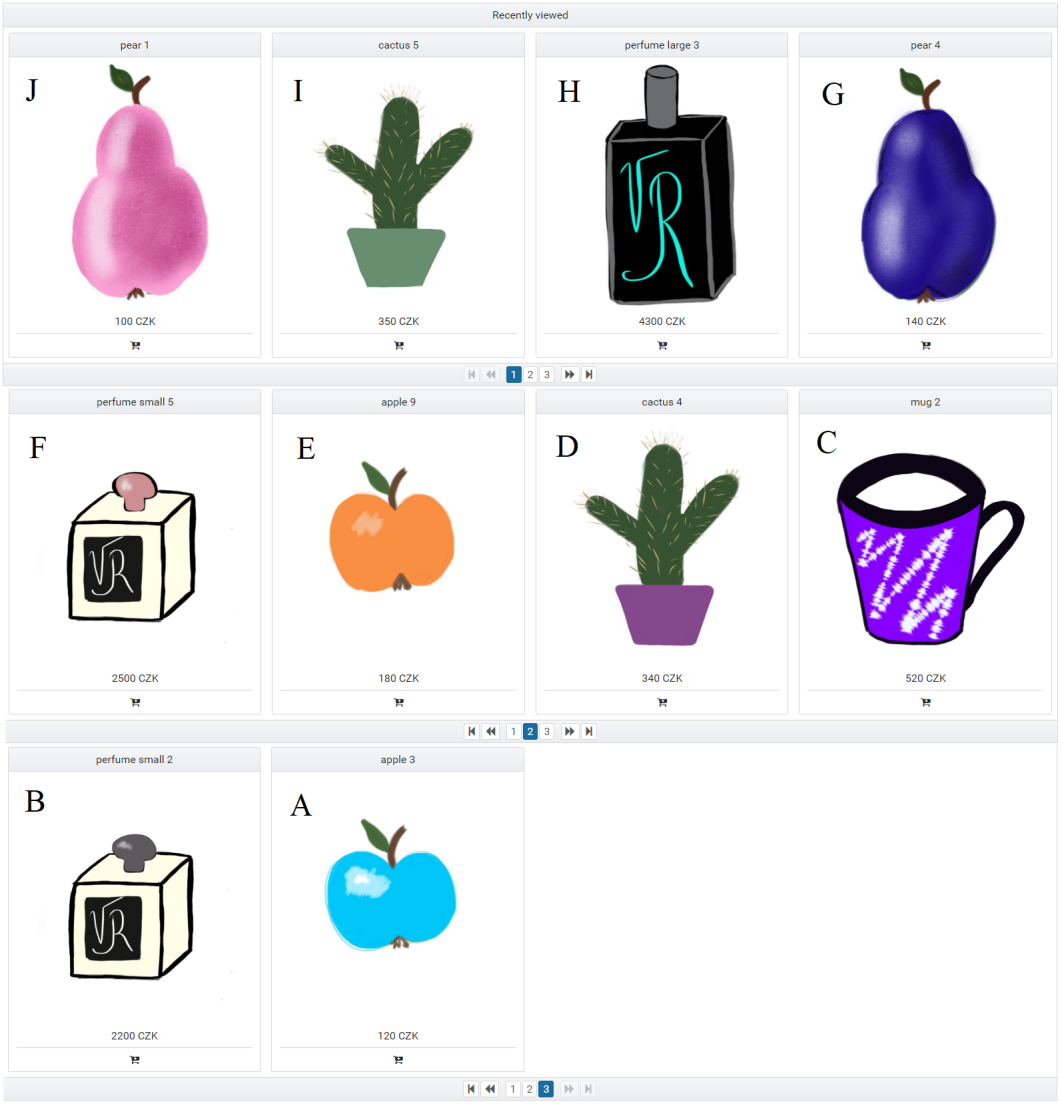
Obrázek D.10: Detail objednaného produktu

Příloha E

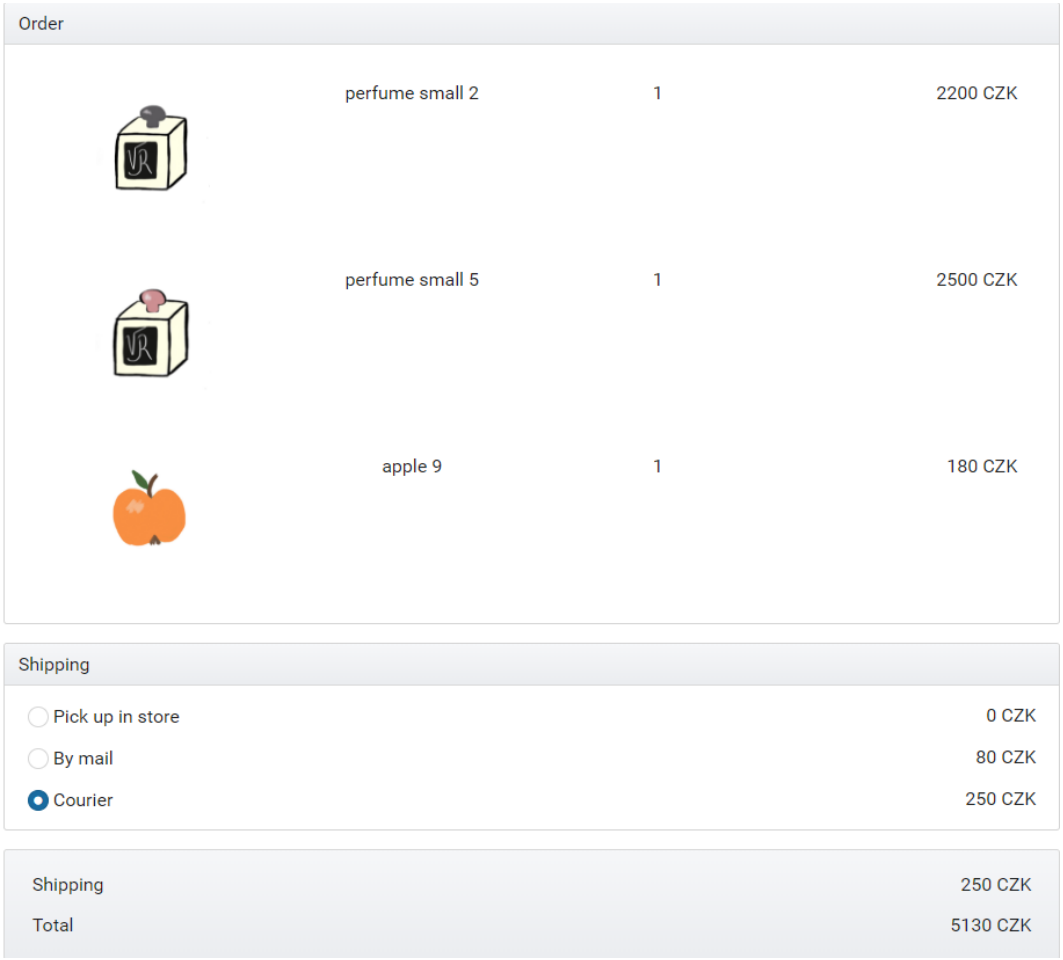
Vizualizace testování adaptivního chování



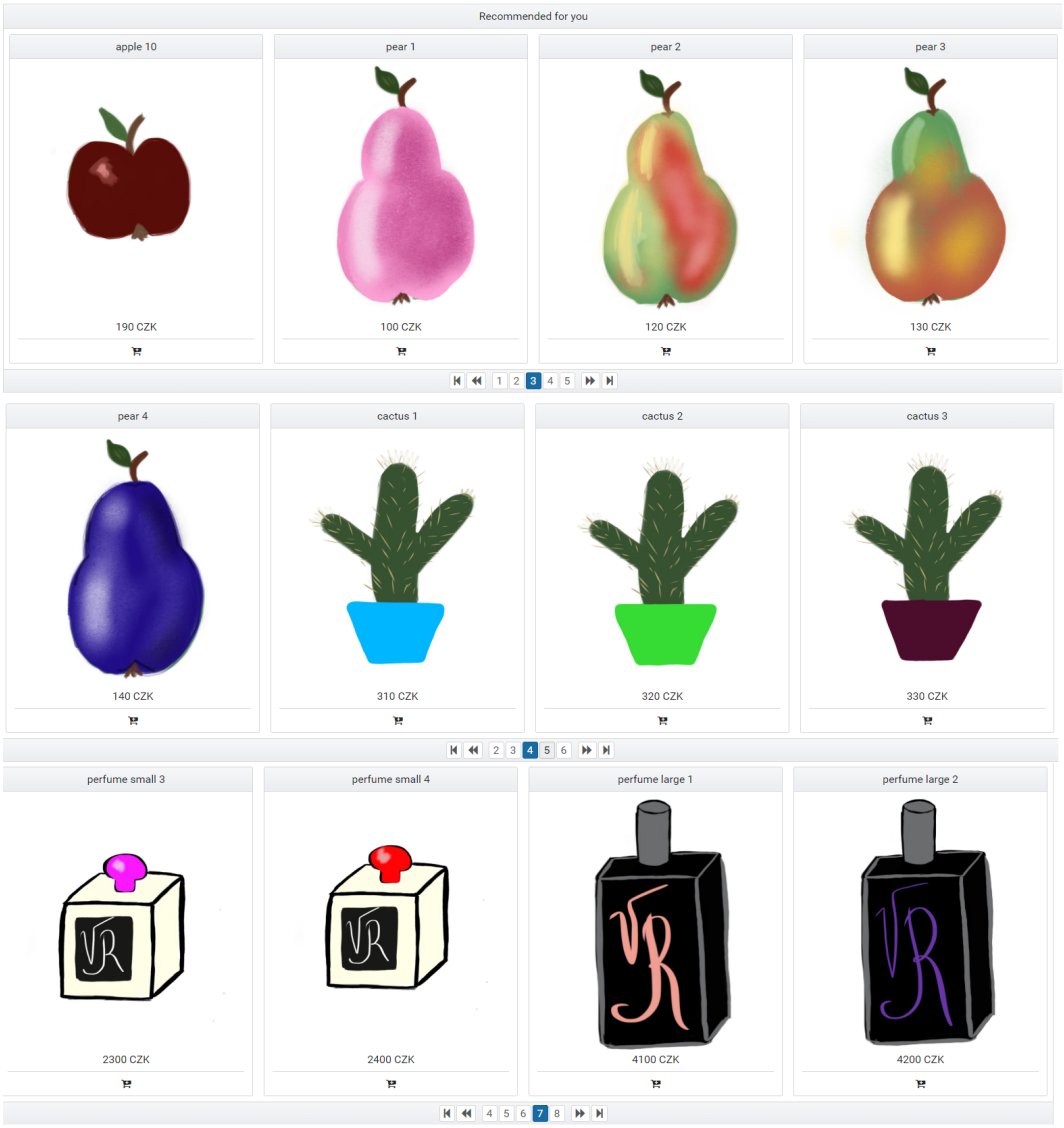
Obrázek E.1: Úvodní stránka pro nového uživatele



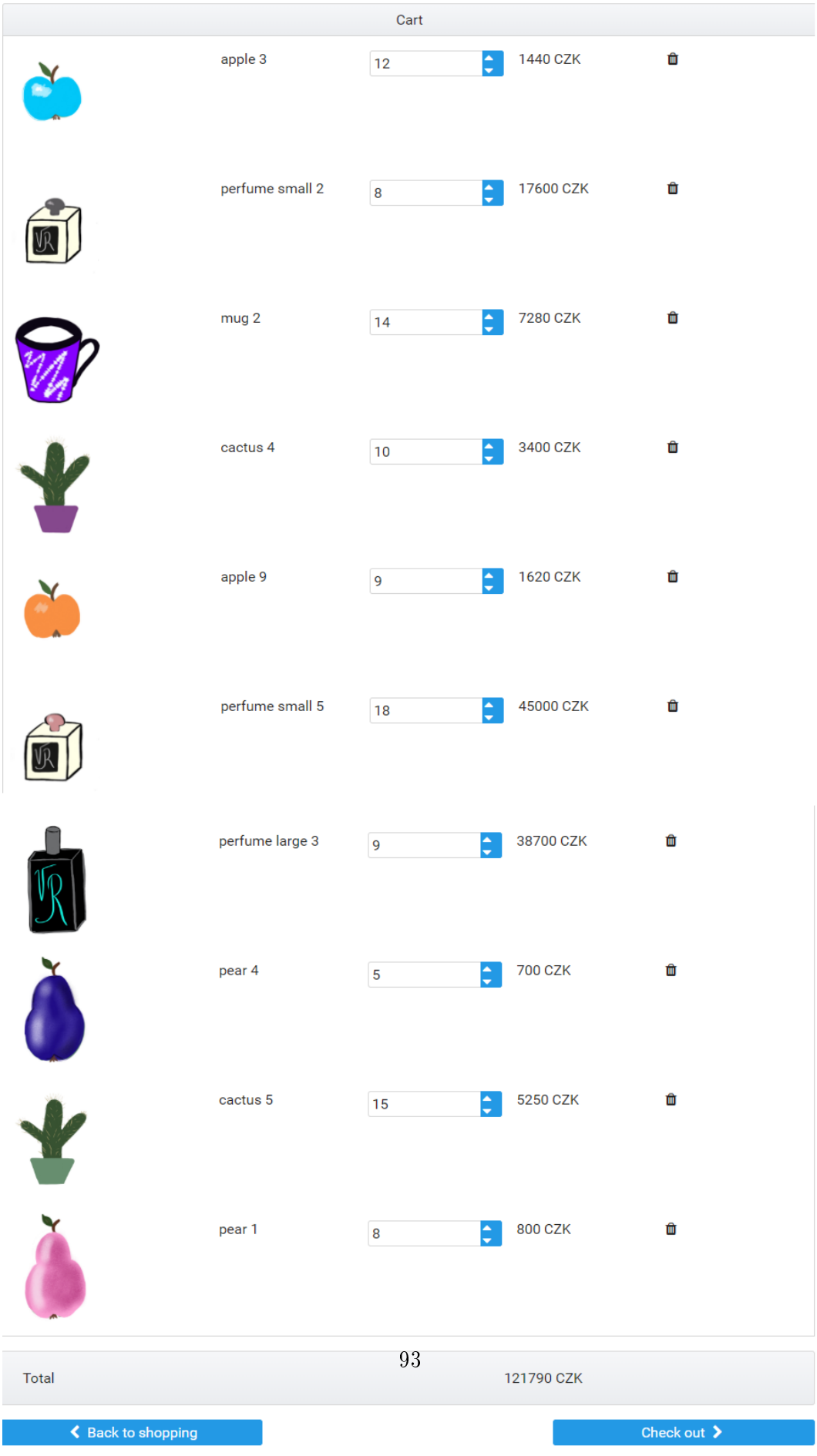
Obrázek E.2: Seznam s nedávno prohlédnutými produkty na úvodní stránce obchodu



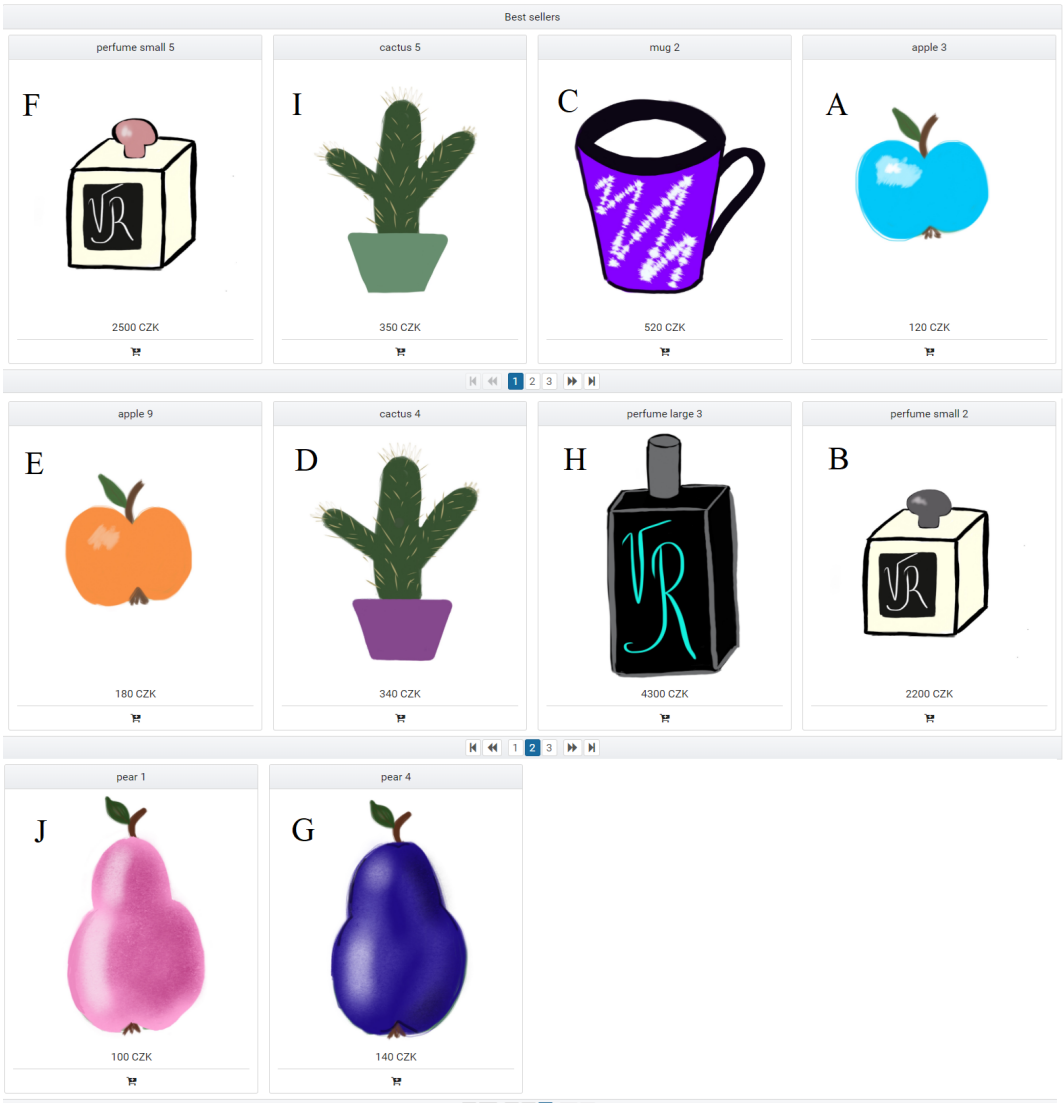
Obrázek E.3: Rekapitulace objednávky z bodu 5.2.1



Obrázek E.4: Seznam „Recommended for you“ s doporučením na základě předchozích nákupů



Obrázek E.5: Košík s objednávkou z bodu 5.2.2



Obrázek E.6: Seznam „Best Sellers“ vytvořený na základě počtů prodaných kusů

Příloha F

Obsah přiloženého CD

- adapt-shop - hlavní projekt
- asf-lib - ASF knihovna
- documents - složka pro veškeré dokumenty
  - diagrams - UML diagramy včetně zdrojových souborů pro Enterprise Architect
  - mockups - návrh uživatelského rozhraní
  - text-pdf-tags - pdf dokumenty textu práce s číslem verze v názvu
  - text-source - zdrojové soubory pro latex
- related-projects - složka pro pomocné projekty
- test-projects - složka pro prototypy a testovací projekty