Rád bych poděkoval vedoucímu mé diplomové práce, prof. RNDr. Petrovi Vojtášovi, DrSc., za cenné rady, konzultace a náměty, které jsem ve své práci použil.

Podobně patří moje poděkování Mgr. Alanovi Eckhardtovi, za konzultace a pomoc s řešením problémů implementační části diplomové práce.

V neposlední řadě bych se chtěl poděkovat všem, kteří mi poskytli cenné informace, popisující jejich zkušenosti a názory ohledně vytvořené aplikace.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 10. 12. 2009 Branislav Václav

Obsah

Název práce: Modely uživatelských preferencí v prostředí webovských obchodů
Autor: Branislav Václav
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: prof. RNDr. Peter Vojtáš, DrSc.
E-mail vedoucího: vojtas@ksi.mff.cuni.cz

Abstrakt

Cílem práce je získat přehled v široké oblasti modelů uživatelských preferencí v prostředí internetových obchodů. Z popsané množiny modelů bude vybrána část, pro kterou budou navrženy konkrétní metody výpočtu preferencí. Vybrané modely budou pak spolu s návrhem odpovídajícího webového prostředí implementované do formy ucelené fungující internetové aplikace.

Součástí vytvořené aplikace je i obsažení vhodné množiny testovacích dat. Nad těmito daty budou provedeny praktické experimenty, jejichž výsledky budou zohledněny v rámci posouzení dosáhnuté funkčnosti aplikace a jejího přínosu pro stávající uživatele internetových obchodů. Získaná uživatelská odezva bude pak využita pro stanovení dalších možností vývoje implementované aplikace.

Klíčová slova: internetový obchod, uživatelské preference, webová aplikace

Title: Models of user preferences in e-shop environment
Author: Branislav Václav
Department: Department of Software Engineering
Supervisor: prof. RNDr. Peter Vojtáš, DrSc.
Supervisor's e-mail address: vojtas@ksi.mff.cuni.cz

Abstract

The aim of this work is to gain insight into the broad range of models of user preferences inside an e-shop environment. A specific group of models will be selected from the overall described set, and an exact method of calculation for these models will be introduced. The selected models, together with a corresponding web environment design, will then be implemented into a comprehensive form of a working web application.

An integral part of the application is formed by the inclusion of an appropriate set of test data. Based on these data, practical experiments will be carried out, and consequent results will be considered in the assessment of the functionality of the provided application and its potential contribution for existing e-shop users. Acquired user feedback will then be used to identify further development opportunities of the implemented application.

Keywords: e-shop, user preferences, web application

1 Úvod

S neustálým zlepšováním dostupnosti Internetu mezi uživateli dochází i ke zvyšování poptávky po kvalitních internetových řešeních. Investice firem do vývoje stále sofistikovanějších webových aplikací každým dnem narůstají a s přihlédnutím na současné trendy, jakými jsou například nově vznikající multimediální standardy, zvyšování penetrace mobilního přístupu k Internetu, nebo příklon k technologiím cloud computingu, se zdá, že vývoj bude v následujících letech tuto prudce stoupající tendenci využívání Internetu následovat.

Jedním z problémů neustále se zvětšujícího množství dat obsažených v Internetu je ovšem schopnost jejich vhodné prezentace stále různorodější množině jeho uživatelů. Vyhledání požadované informace ve „změti“ informací různého druhu a kvality tak může pro uživatele představovat problém větší než kdykoli předtím.

Problematika sémantizace webu řešící popsaný problém hledáním vztahů mezi významem obsažených informací a specifickými požadavky jednotlivých uživatelů je rozsáhlou oblastí, do které se zařazují i modely uživatelských preferencí, tvořící ústřední téma obsahu této diplomové práce.

1.1 Vymezení cílů

Cílem práce je obeznámení čtenáře se širokou problematikou modelů uživatelských preferencí v prostředí systémů webových obchodů. Z popsané množiny modelů pak bude vybrána část, pro kterou budou navrženy konkrétní metody výpočtu preferencí. Vybrané modely pak budou spolu s návrhem odpovídajícího webového prostředí implementované do formy ucelené fungující internetové aplikace.

Součástí vytvořené aplikace bude i obsažení vhodné množiny testovacích dat. Nad těmito daty budou provedeny experimenty, jejichž výsledky budou zohledněny v rámci posouzení dosáhnuté funkčnosti aplikace a jejího přínosu pro stávající uživatele systémů webových obchodů. Získaná uživatelská odezva bude pak využita pro stanovení dalších možností vývoje, rozšíření a změn implementované aplikace.

1.2 Struktura práce

Obsah a struktura práce se řídí cíli vymezenými v kapitole 1.1. Další text této práce je strukturován následovně:

- Kapitola 2 popisuje problematiku modelů uživatelských preferencí v oblasti webových obchodů a zavádí příklady nejběžnějších modelů použitelných v této oblasti informačních systémů.
- Kapitola 3 popisuje vybrané modely preferencí a navržené metody jejich výpočtu, jako i celkovou architekturu jednotlivých vrstev implementované aplikace.
- Kapitola 4 se zaměřuje na popsání a vyhodnocení experimentů provedených nad finální verzí vytvořené aplikace.
- Kapitola 5 pak představuje další možnosti rozšíření nebo úprav vytvořené aplikace do budoucna.
- Kapitolu 6 tvoří závěr diplomové práce, kapitoly 7 a 8 pak popisují obsah dodaného CD média a seznam literatury použité během procesu tvorby této práce.

2 Modely uživatelských preferencí

Webový obchod obecně se dá charakterizovat jako virtuální webové prostředí, které ve své nejjednodušší podobě slouží majiteli obchodu k prezentaci nabízeného zboží a zákazníkovi k možnosti nabízené zboží zakoupit. Během let vývoje byly tyto systémy obohaceny o množství nových funkčností a prvků rozhraní, některé jsou dnes vnímány téměř jako standardní součást funkcionality každého nového eshopu. Jedná se zejména o:

- Jednoduché a přehledné uživatelské rozhraní, vycházející z ověřených principů rozvržení webových aplikací. Cílem je, aby křivka učení práce s aplikací byla pokud možno co nejstrmější.
- Možnost jednoduché navigace a hledání v množině nabízených produktů
    - Prohlížení nabídky
        - Logické a intuitivní zatřídění produktů do skupin (kategorií)
        - Možnost řazení produktů podle vybraných parametrů zájmu
        - Různé módy zobrazení nabídky
          (co se týče rozsahu i množství zobrazených informací o produktech)
    - Vyhledávání v nabídce
        - Vyhledávání a filtrování produktů v nabídce podle vybraných parametrů zájmu

Tyto základní vlastnosti se v dnešní době pokouší implementovat více či méně úspěšně většina současných eshop systémů. Cílem této kapitoly je popsat vybrané modely uživatelských preferencí, které by umožnily rozšíření stávající funkčnosti eshopů a posoudit jejich možný praktický přínos pro uživatele.

2.1 Globální prostředí aplikace

Webový obchod můžeme z konceptuálního hlediska rozdělit na globální prostředí tvořené prvky rozhraní zasazenými do pevného rozvržení (layoutu) stránek a dynamický obsah obchodu tvořen množinou nabízených produktů, obsluhován tímto globálním prostředím.

Personalizace globálního prostředí webových obchodů není v dnešní době příliš obvyklá. Možnost změny zobrazení nabídky na základě přímé uživatelské akce se dnes pokládá za standard, celkové přizpůsobení vzhledu a rozvržení uživatelského rozhraní se však obecně dá pokládat spíše za nežádoucí. Důvodem je především snaha o jednoduchost ovládání, která se nejjednodušeji dosahuje zaměřením se na typické uživatelské návyky. Jinými slovy, chceme, aby náš webový obchod vypadal a fungoval podobně jako ostatní obchody, s hlavním cílem neodradit uživatele přílišnou nestandardností anebo komplikovaností rozhraní.

2.2 Dynamický obsah aplikace

Prezentace nabídky webshopu je na druhé straně pro úspěšnost webového obchodu klíčová. Nabízí se paralela s reálným (kamenným) obchodem, kde taktéž jeden z hlavních komunikačních prostředků mezi majitelem a potenciálním zákazníkem obchodu tvoří výklad – prezentace vhodného výběru z nabídky daného obchodu. Některé současné systémy webových obchodů tuto analogii výkladu implementují – jedná se typicky o úvodní stránku, která může zobrazovat momentálně nejprodávanější zboží, zboží v akci, informace o slevách apod. Nevýhoda tohoto řešení, stejně jako v reálném světě, je pevná struktura obsahu stránky. Důsledkem je možná neefektivita ve schopnosti upoutat uživatele, která je v současném vysoce konkurenčním prostředí webových služeb velmi žádoucí. Tato neefektivita vychází ze zřejmé variability uživatelských preferencí. Nabídka postavena na míru jednomu uživateli nemusí zkrátka zaujmout uživatele jiného, protože se zajímá o jiný druh zboží.

Odstranění tohoto problému je základní myšlenkou úkolu, za kterým stojí snaha o implementaci vhodných modelů uživatelských preferencí v systémech webových obchodů [3].

2.3 Využití preferenčních modelů

Primárním cílem většiny modelů uživatelských preferencí je tedy dosažení schopnosti čím nejspolehlivěji upravit prezentaci nabídky obchodu tak, aby odpovídala aktuálním preferencím uživatele. Tuto skupinu preferencí můžeme klasifikovat jako uživatelský vkus (user taste information). Čím přesnějšími údaji popisujícími uživatelský vkus systém disponuje, tím spolehlivější je jeho výpočet pravděpodobností s jakou objekt dokáže zaujmout zvoleného uživatele (za předpokladu, že použitý algoritmus výpočtu je správný). Hovoříme o míře oblíbenosti objektu uživatelem reprezentované ohodnocením objektu (skóre) [1], a přesnost tohoto údaje vzhledem k realitě a k hodnocením ostatních objektů je jeden z klíčových ukazatelů kvality implementace zvoleného modelu preferencí.

2.4 Metody záznamu preferencí

Uživatelský vkus vzhledem ke zvolené oblasti zájmu je tedy definován souborem jednotlivých preferencí uvažovaných nad touto oblastí zájmu. Jak bylo zmíněno, čím detailnějšími a přesnějšími údaji o uživateli systém disponuje, tím větší je jeho potenciál produkovat kvalitnější výstupy. Je proto zřejmé, že většina systémů implementujících nějaký model preferencí se snaží o neustálé zpřesňování uživatelských údajů prostřednictvím různých metod sběru dat (data miningu) [2]. Metody získávání uživatelských preferencí rozdělujeme do následujících tří skupin.

2.4.1 Přímý způsob záznamu preferencí

Uživatel prostřednictvím dedikovaných prvků rozhraní přímo specifikuje preference, které definují jeho způsob ohodnocení objektů. Aby bylo toto možné, musí uživatel poznat model výpočtu ohodnocení objektů, nad kterým aplikace pracuje.

Výhodou této metody je vysoká přesnost, nevýhodou naopak pro uživatele potenciálně složitý způsob formulace požadavků v případě příliš komplikovaného modelu výpočtu.

Příklady

- Uživatel prostřednictvím vyhledávacího rozhraní specifikuje, že má zájem o produkty vyrobené jen zvoleným výrobcem a zároveň do maximální specifikované ceny.

2.4.2 Explicitní způsob záznamu preferencí

Explicitní způsob specifikace preferencí nevyžaduje od uživatele znalost hodnotícího vztahu. Probíhá na úrovni objektů, u kterých stačí specifikovat míru oblíbenosti uživatelem.

Tento způsob je pro uživatele jednoduchý a intuitivní, přesnost modelování uživatelského vkusu a tím i výstupních výsledků však klesá.

Příklady

- Uživatel hodnotí objekt kladnou anebo zápornou známkou.

2.4.3 Implicitní způsob záznamu preferencí

Tento způsob zahrnuje velké množství metod schopných úspěšně či méně úspěšně interpretovat činnost uživatele v aplikaci. Uživatel svoje preference přímo nevyjadřuje, sběr dat probíhá automaticky na pozadí [2]. Hlavní výhoda spočívá v nulové investici uživatele do procesu poskytování informací o svých preferencích.

Příklady

- Monitorování využité funkcionality, aktivit a času uživatele stráveného ve specifické oblasti aplikace (např. zobrazeni detailu produktu, označování textu, zakoupení produktu atd.)
- Sledování technických prostředků uživatele (např. přístup z mobilního zařízení)

Čím přesnější a sofistikovanější metoda sběru uživatelských preferencí se v aplikaci uplatňuje, tím lepší výsledky úspěšné personalizace je možné od systému očekávat. Toto platí hlavně v počátečných fázích nasazení aplikace, kdy systém nedisponuje dostatečným množstvím vstupních dat, anebo u aplikací, kde jsou možnosti vyjádření uživatelských preferencí např. z důvodu typicky krátkého času interakce uživatele s aplikací omezené.

2.5 Varianty modelů preferencí

2.5.1 Modely preferencí nad globálními daty uživatelských profilů

Modely preferencí založené na vhodné interpretaci globálních uživatelských dat patří ve své základní podobě k implementačně jednodušším. Preference se odvozují na základě pevného logického modelu, popisujícího vztahy mezi možnými hodnotami globálních údajů uživatelského profilu a pravděpodobností příklonu uživatele k zodpovídající uživatelské preferenci. Jednotlivé objekty (v našem případě nabízené produkty) a jejich vlastnosti nejsou v modelu výpočtu brány v potaz.

Výhodou aplikace tohoto modelu je možnost získání druhů preferencí, které by jinými metodami sběru dat šly získat velmi těžko. Nevýhodou je naopak relativně generická úroveň specifikace jednotlivých preferencí a relativně omezené možnosti využití v praxi.

Příklady

Osobní údaje uživatele jako např. pohlaví, datum narození, příjem, nejvyšší dosažené vzdělání, bydliště, rodinní stav, zájmy apod. lze na základě aktuálně platných společenských principů převést na zodpovídající typy uživatelských preferencí:

- Pohlaví uživatele zvyšuje pravděpodobnost zájmu o druh zboží zodpovídající zvolenému pohlaví (globálně, např. automobilové díly, elektronika; v rámci kategorie produktů např. kosmetika, šperky, náramkové hodinky)
- Věk, rodinný stav, zájmy uživatele taktéž vypovídají o preferované kategorii a vlastnostech zboží (např. mladá žena preferuje módu pro mladé, staršího svobodného muže nezajímají produkty starostlivosti o dítě, uživatel se zájmem o cestování a sport preferuje produkty výbavy pro tyto aktivity)
- Příjem typicky definuje požadovanou cenovou kategorii produktů (vysoký příjem např. předpokládá zájem o vyšší třídu automobilů)

Elementárním předpokladem úspěšného fungování tohoto typu modelu uživatelských preferencí je dostatečná dostupnost a kvalita požadovaných uživatelských dat. S nárůstem počtu uživatelů webových služeb se však zvyšuje i riziko zneužití těchto dat. Problematika ochrany soukromí a zabezpečení osobních údajů v prostředí Internetu každodenně nabývá na důležitosti, ochota poskytování těchto informací samotnými uživateli naopak klesá. V prostředí webových obchodů se navíc nabízí paralela s reálným světem, která taktéž přispívá k utváření „nežádoucích“ uživatelských návyků. Po zákazníkovi v kamenné prodejně nebývá typicky požadováno zveřejnění jakéhokoli osobního údaje za účelem možnosti zakoupení zboží. Požadování těchto údajů v prostředí webového obchodu může být proto uživatelem vnímáno jako mírně ofenzivní, bylo by tedy lepší se mu v rámci možností vyhnout.

2.5.2 Modely preferencí nad strukturou objektů obsažených v systému

Modely preferencí navržené nad množinou obsažených objektů patří k nejběžnějším typům implementovaných modelů. Objekty jsou v tomto případě vnímány jako nedělitelné celky bez jakékoli snahy výpočtu o vyhodnocování jejich vnitřních atributů.

Modely preferencí nad strukturou objektů bez propojení na uživatelský profil

Tyto modely výpočtu preferencí patří implementačně k nejjednodušším a velké množství webových obchodů je již v současnosti implementuje. Využívají implicitního sběru dat typicky na základě akcí zakoupení produktů a jejich výstupem bývá uspořádání objektů na základě úspěšnosti jejich prodeje. Tento model dokáže poskytnout informace o tom, které objekty celkově uživatele preferují, nevýhodou je však příliš všeobecné ponětí této problematiky a neschopnost přizpůsobení výsledkové sady individuálním rozdílům v jednotlivých aspektech uživatelského vkusu.

Modely preferencí nad strukturou objektů s propojením na aktuální uživatelský profil

Cílem je vyhodnocení oblíbenosti objektů pro daného uživatele na základě preferencí, poskytnutých výlučně tímto uživatelem.

Způsoby, kterými je možné získat od uživatele data sloužící jako základ pro výpočet zodpovídajících preferencí je více. Jedná se o explicitní a implicitní metody sběru dat. Mezi tyto metody může patřit:

- Explicitní vyjádření oblíbenosti objektu prostřednictvím přirazení kladní nebo záporní známky tomuto objektu, typicky pomocí prvků rozhraní jako jsou např. tlačítka „Promote“ / „Remove“, „Líbí se mi“ / „Nelíbí se mi“, palec nahoru / dolů.

  Tuto metodu využívá např. vyhledávač Google pro možnost ohodnocení výsledků vyhledávání, anebo množství hudebních doporučovacích systémů (recommender systems) pro možnost ohodnocení jednotlivých písní (viz obr. 1).

  Hlavní nevýhodou této metody je relativně malá hodnotící škála, nabízející jenom hrubou granularitu možných hodnocení, které důsledkem je nedostatečná přesnost výsledného ohodnocení objektů. Proto se tato metoda záznamu preferencí využívá typicky v kombinaci s jiným modelem výpočtu preferencí, který tyto údaje dokáže zpřesnit. Výhodou je naopak jednoduchost a nenáročnost použití pro uživatele, co v některých případech může paradoxně vést k získání přesnějších údajů ve srovnání s použitím sofistikovanějších metod sběru dat, jednoduše z důvodu, že uživatel je ochoten kliknout na tlačítko „se srdíčkem“, nechce se mu ale přemýšlet nad procentuálním ohodnocením objektu.

Obrázek 1
Příklady rozhraní existujících systémů pro kladné/záporné hodnocení objektů.
1 – Google.com, 2 – Engadget.com, 3 – Last.fm, 4 – Sourceforge.com

- Explicitní vyjádření oblíbenosti objektu prostřednictvím přirazení zodpovídající hodnoty na škále 0-5, případně 0-10.

  Tato metoda je kompromisem mezi výše popsanou metodou „palec nahoru / dolů“ a možností přidělení např. procentuálního hodnocení oblíbenosti objektu. Ve webových aplikacích je často využívána, nejběžnější prvek rozhraní je reprezentován skupinou prázdných a plných (případně poloplných) hvězdiček, vyjadřujících výsledné skóre. Tento postup zjemňuje škálu hodnocení, zavádí však i některé nové problémy. Jednak u příliš jemné škály může pro uživatele představovat přílišnou složitost zvolení správné hodnoty, co může vést až k nekonzistencím v hodnocení, na druhé straně, každý uživatel může mít vnímaný průměr svého hodnocení mírně posunutý, takže v případě kombinace různých modelů preferencí využívajících hodnocení více uživatelů je žádoucí tato hodnocení vhodným způsobem normalizovat.

Obrázek 2
Příklady rozhraní existujících systémů s detailnější škálou ohodnocení objektů.
1 – Cnet.com, 2 – Imdb.com, 3 – Mall.cz, 4 – Steepster.com

- Explicitní anebo implicitní vyjádření oblíbenosti objektu prostřednictvím upřednostnění jednoho objektu před druhým, případně před menší skupinou jiných objektů.

  Rozdíl typů sběru dat spočívá pouze v návrhu rozhraní, na základě kterého si uživatel je anebo není vědomý toho, že jeho preference jsou zaznamenávány. Tento způsob záznamu preferencí je pro uživatele jednoduchý, nakolik se od něj nevyžaduje schopnost posouzení globální míry oblíbenosti objektu, stačí jenom zvolit nejoblíbenější objekt z malé množiny. To však sebou přináší i problémy, nakolik systém nezná absolutní skóre objektů v rámci celé množiny, zaznamenává jenom komparativní preference mezi dvojicemi těchto objektů. Cílem výpočtu je pak najít takové uspořádání objektů, které co nejlépe zachovává původní hodnoty získaných preferencí, co může způsobovat problémy v případě, že zdrojová data jsou nekonzistentní a v částečném uspořádání daném vstupnými hodnotami získaných preferencí existují cykly.

  Pravděpodobnost vzniku nekonzistencí v uživatelském hodnocení je však možné výrazně snížit vhodným výběrem porovnávaných objektů. Čím jsou objekty různorodější, tím menší je pravděpodobnost chyby uživatele. Aby však bylo možné jednotlivé porovnávané objekty správně vybrat, musí model uživatelských preferencí pracovat kromě objektu i s úrovní jeho atributů. Proto je použití metody sběru komparativních preferencí vhodnější využít až v případě, kdy model výpočtu preferencí tuto úroveň implementuje.

- Implicitní vyjádření oblíbenosti objektu na základě monitorování vybraných uživatelských akcí.

  Sem patří jednoduché i pokročilejší metody monitorování činností uživatele, kterých cílem je úspěšná asociace příčin těchto činností s předpokládanou mírou oblíbenosti asociovaných objektů. Akce, které mohou ovlivňovat skóre jednotlivých objektů, můžou v prostředí webových obchodů zahrnovat:

  - Časté zobrazování detailu produktu, vyšší čas strávený na stránce detailu, otevíraní hypertextových odkazů v popisu produktu, prohlížení obrázků produktu, uložení odkazu na detail, označování textu, tisk stránky
  - Zakoupení produktu

  Výhoda této metody sběru dat je zřejmá, je jí nulová zatížitelnost uživatele. Nevýhodou může být variabilní přesnost získaných uživatelských preferencí a nemožnost jejich uživatelské korekce, jelikož uživatel typicky ani netuší, že jeho preference jsou zaznamenávány a navíc nezná ani model jejich výpočtu.

Modely preferencí nad strukturou objektů s propojením na více uživatelských profilů

Modely výpočtu využívající informace pocházející z více uživatelských profilů jsou specifické svou schopností doporučení vhodného objektu uživateli i v případě, že on sám žádnou kladnou preferenci nad tímto objektem nespecifikoval. Důvodem je fakt, že tuto preferenci nad vybraným objektem vyjádřil jiný uživatel a s její hodnotou se při výpočtu výsledních uživatelských preferencí uvažuje. Tyto metody výpočtu musí však fungovat v kombinaci s předešlými modely získávání a výpočtu preferencí, aby bylo možné vyhodnotit vztahy mezi jednotlivými objekty, nebo korektně stanovit charakteristiku posuzovaného uživatelského vkusu. Hovoříme o kolaborativních modelech výpočtu uživatelských preferencí a mezi základní způsoby aplikace těchto modelů mohou patřit:

- Kolaborativní filtrování objektů bez vyhodnocování vlastností zodpovídajících uživatelských profilů.

  Jednodušší metoda, při které se zkoumají vztahy mezi jednotlivými objekty a existence uživatelského profilu slouží jen len na vymezení prostoru, v rámci kterého jsou tyto vztahy zaznamenávány a analyzovány. Typickým výstupem takovéhoto modelu je např. množina vzájemně souvisejících objektů, získaná prostřednictvím implicitně zaznamenaných preferencí uživatelů systému. V případě získání kladné preference uživatele nad jedním z objektů je potom systém schopný na základě asociované množiny doporučit další objekty, u kterých je zvýšená pravděpodobnost, že by mohly uživatele zaujmout.

  Takovýto systém implementuje např. webový obchod Amazon, kde po vybrání konkrétního objektu je uživateli nabídnut seznam produktů, které zvyknou uživatele společně s tímto produktem zakoupit a je proto pravděpodobné, že by mohli zaujmout i aktuálního uživatele.

Obrázek 3
Příklad rozhraní webového obchodu Amazon.com, zobrazující množinu vzájemně souvisejících produktů

- Kolaborativní filtrování objektů s vyhodnocováním jednotlivých uživatelských profilů.

  Tento způsob zahrnuje známější formu implementace kolaborativního filtrování, v oblasti webových obchodů však, ke dnešnímu dni, nebývá příliš rozšířený. Myšlenka výpočtu spočívá v analýze uživatelského vkusu jednotlivých uživatelských profilů a stanovení míry shody tohoto vkusu se vkusem aktuálního uživatele. Preference uživatelů s podobným vkusem jsou potom použité pro výpočet predikce oblíbenosti jednotlivých objektů u aktuálního uživatele.

  Tento model výpočtu bývá poměrně přesný v případech, kdy systém disponuje dostatkem správných uživatelských dat. Jeho výhoda spočívá zejména ve faktu, že získané varianty uživatelského vkusu jsou založeny na reálných preferencích větší množiny uživatelů. Výstupní predikce uživatelského vkusu je potom přímo závislá na těchto reálních datech a míra její přesnosti může být v některých případech vyšší než u modelů využívajících pro výpočet uživatelského vkusu pouze vybraný matematický model.

2.5.3 Modely preferencí nad strukturou objektů a jejich atributů

Modely preferencí uvažující v algoritmu výpočtu s hodnotami atributů jednotlivých objektů poskytují oproti již zmíněným modelům několik zásadních výhod. V první řade je to možnost uživatele na základě vlastností atributů přesně specifikovat své preference pro zvolenou kategorii objektů. Výstupy tohoto modelu patří potom v případě, že vyjádřené preference exaktně kopírují reálné požadavky uživatele, k jedním z nejpřesnějších. Znalost atributů jednotlivých objektů taktéž udává předpoklad k možnosti vzniku pokročilejších algoritmů výpočtu, s pomocí kterých je možné na základě analýzy hodnot jednotlivých vlastností objektů a znalosti uživatelských preferencí nad některými z nich, odvodit míru oblíbenosti objektů, pro které systém nedisponuje dostatečným množstvím uživatelských preferenčních dat.

Modely preferencí nad strukturou objektů a jejich atributů s propojením na aktuální uživatelský profil

- Metody výpočtu nad preferencemi specifikovanými uživatelem přímo pomocí dedikovaných prvků rozhraní.

  Algoritmy výpočtu preferencí v závislosti od přímých požadavků uživatele na hodnoty atributů objektů v dnešní době implementuje velké množství webových obchodů. Jedná se o známou funkcionalitu vyhledávání produktů podle parametrů. Drtivá většina těchto implementací však používá relativně jednoduchý model výpočtu výsledkové sady, založený na pouhém ořezání vstupné množiny objektů tak, aby tyto vyhovovaly podmínkám rozsahových dotazů zadaných uživatelem. Nevýhoda této metody spočívá hlavně v možnosti eliminace potenciálně uživatelsky zajímavých objektů z výsledkové sady z důvodu, že tyto objekty nesplňují stoprocentně podmínky pro hodnoty vybraných atributů definovány uživatelem. V praxi tak může pozornosti

uživatele uniknout objekt, který má velmi dobré předpoklady zaujmout, nesplňuje ale jednu ze stanovených podmínek, přičemž priorita této podmínky vnímána uživatelem může být v konečném důsledku velmi nízká. Nežádoucnost tohoto chování pochopitelně stoupá úměrně s počtem objektů obsažených v databázi aplikace.

Model preferencí, který se snaží eliminovat výše popsané chování a přesněji aproximovat reálné preference uživatele, rozšiřuje výpočtový algoritmus o sofistikovanější metody stanovení výsledního skóre objektu v závislosti od hodnot jeho atributů a specifikovaných vstupních podmínek. Toto skóre vyjadřuje potenciální míru oblíbenosti objektu uživatelem a je určujícím údajem pro stanovení výsledního pořadí objektů. Výsledková sada tudíž nemusí být ořezána a uživateli jsou prezentované všechny objekty seřazené podle výsledního indexu doporučení vzhledem k zadaným vstupným podmínkám. Pokročilejší modely mohou potom implementovat i možnost přirazení priority k jednotlivým typům atributů a k nim asociovaným preferencím, co může pak zpřesnit výpočet a zlepšit tím uživatelský zážitek (user experience) z práce v aplikaci.

Metody výpočtu výsledního ohodnocení (skóre) objektu jsou různé. Jelikož objekt je v zásadě definován množinou svých atributů, výsledné skóre se počítá jako určitá forma agregace hodnocení všech jeho atributů vzhledem k zadaným podmínkám. Metoda výpočtu může taky zohlednit případné priority jednotlivých podmínek, pokud tuto možnost zvolený výpočtový model implementuje. Pro rozdílné typy atributů se jejich výslední ohodnocení může počítat různým způsobem, např.:

- Čísla mohou být ohodnocené maximálním skóre v případě, že patří do intervalu stanoveného podmínkou. V případě, že do něj nepatří, ale nachází se v jeho blízkosti, mohou stále získat nenulové skóre. Tato implementace umožní i objektům, které sice nesplňují specifikovanou podmínku, ale mají k ní velmi blízko, získat relativně vysoké hodnocení a dostat se do pozornosti uživatele.
- Řetězce mohou implementovat pokročilé metody vyhodnocování shody hledaného řetězce s porovnávanou hodnotou, při zohlednění např. překlepů apod. Systém může taktéž implementovat vybranou metriku pro jednotlivé textové hodnoty, umožňující další zpřesnění výsledného skóre pro posuzovanou hodnotu atributu. (Metrika může např. definovat vztahy mezi řetězci „Praha 1“, „Praha 2“, „Praha 10“, „Staré město“ anebo řetězci „HDMI 1.3“, „HDMI 1.1“, „S-VHS“.)
- Logické hodnoty mohou zřejmě nabývat pouze maximálního nebo minimálního skóre.

- Metody výpočtu nad preferencemi získanými explicitně nebo implicitně.

  Možnost přístupu k hodnotám atributů objektů dokážou využít i modely preferencí, které na vstupu nedisponují přímo poskytnutou množinou podmínek popisující přesné preferenční pravidla uživatele. Jde o modely výpočtu, které se naopak snaží prostřednictvím generalizace danou množinu pravidel algoritmicky zformulovat. Vycházejí tudíž z preferencí, získaných explicitním a implicitním způsobem, což typicky představuje omezenou množinu hodnocení některých objektů v kategorii, bez znalosti vztahů mezi hodnotami jejich atributů. Cílem výpočtu je pak na základě těchto hodnot atributů oblíbených objektů zkonstruovat pravidla, která s největší pravděpodobností definují reálné preference uživatele. Výhodou této metody je schopnost nabídnout uživateli na základě podobnosti hodnot jednotlivých atributů i takové objekty, pro které nemusí systém disponovat žádnými explicitně či implicitně poskytnutými preferenčními daty. Tato metoda je tudíž komplementární k metodě kolaborativního filtrování a v případě implementace dostatečně sofistikovaného algoritmu může u systémů, které disponují např. malým počtem uživatelů nebo nepřesnými daty většiny z nich, dosahovat kvalitnějších výsledků.

2.6 Časový aspekt v modelech uživatelských preferencí

Preference jednotlivých uživatelů v prostředí webových obchodů mají tendenci se s uplynulým časem měnit. Jedním z důvodů je přirozená nestabilita požadavků uživatele, kterou ovlivňuje mnoho proměnlivých životních faktorů, ať je to např. změna výšky příjmu, narození dítěte nebo zakoupení produktu vytvářející touhu po vhodném synergickém doplňku. Druhým z faktorů je dynamika samotného obsahu nabídky obchodu – starší zboží v nabídce morálně zastarává a nové produkty s lepšími vlastnostmi mají tendenci zaujmout popřední místa v míře uživatelské oblíbenosti.

Na modely výpočtu fungující výlučně nad krátkodobě vyjádřenými preferencemi (typicky např. vyhledávání podle parametrů) toto chování nemá velký vliv, u modelů, které svá vstupní data dlouhodobě zaznamenávají je ale nutné tento vliv možné proměnlivosti preferencí zohlednit. Nové produkty, pro které ještě neexistují žádná preferenční data, by měly mít stejnou šanci získat přízeň uživatele, jako měly již oblíbené produkty v momentě jejich přidání do nabídky. Prezentace obsahu by proto měla tyto zákonitosti zohledňovat.

Jednou z metod, jak se vypořádat s proměnlivostí uživatelských preferencí je používat pro výpočty různých modelů preferencí pouze aktuální preferenční data. To lze vyřešit zahazováním dat, která jsou starší než stanovená mez. Pro tento účel se ovšem musí udržovat historie všech sesbíraných dat spolu s časem jejich získání, co se může ukázat jako zbytečně komplikované řešení pro daný účel. Jednodušší metoda zahrnuje postupné průměrování hodnocení objektů, co vyžaduje neustálý přísun nových preferenčních dat uživatele v případě, že tento chce ovlivnit měnící se pořadí oblíbenosti produktů.

3 Design a implementace systému

3.1 Rozsah funkčnosti aplikace

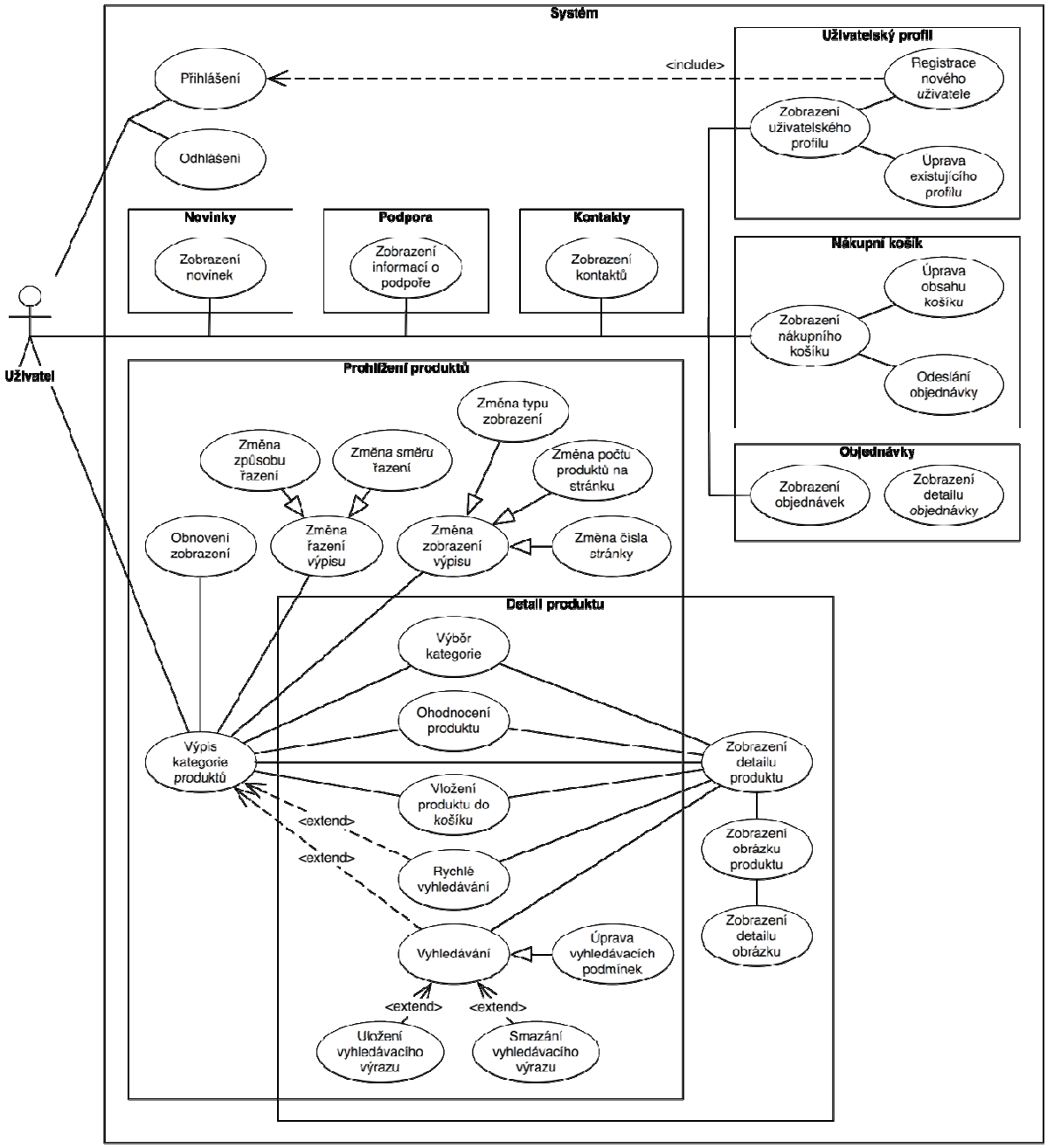


Diagram 1
Model případů užití [11]

Jedním z nutných předpokladů možnosti úspěšné implementace vybraných modelů uživatelských preferencí je existence webového prostředí, které umožňuje sběr preferenčních dat a prezentuje výsledky výpočtů vybraných modelů. Implementační část této diplomové práce proto zahrnuje návrh a realizaci takovéhoto prostředí, které je tvořeno vybranou

podmnožinou případů užití běžného webového obchodu, rozšířenou o funkcionalitu požadovanou vybranými preferenčními modely. Diagram č. 1 znázorňuje rozsah implementované aplikace co do vztahů mezi jednotlivými případy užití a jejich začleněním do různých obrazovek rozhraní aplikace.

3.2 Struktura dat

Pro snazší pochopení vybraných modelů preferencí a metod jejich výpočtu je dále vhodné stručně obeznámit čtenáře se zjednodušeným modelem datové struktury objektů, nad kterou je celková funkčnost aplikace vystavěna, implementaci modelů uživatelských preferencí nevynímaje. Reálně implementovaný objektový model je pak rozšířením prezentovaného konceptu, jeho celková znalost je ale pro zjednodušení představy fungování popsaných preferenčních modelů nepodstatná.

Objekty obsažené v datové struktuře aplikace můžeme v zásadě rozdělit na dvě skupiny. První jsou objekty nesoucí globální data aplikace, tvořené aktuální nabídkou eshopu s detailním popisem její vlastností. Tyto objekty jsou vytvořeny při prvním požadavku na zobrazení stránky a jejich existence trvá po dobu celého životního cyklu aplikace. Druhou skupinou jsou objekty, které se mění s aktuálně přihlášeným uživatelem. Sem patří například informace o uživatelském profilu nebo preferenčních datech uživatelů.

3.2.1 Globální obsah aplikace

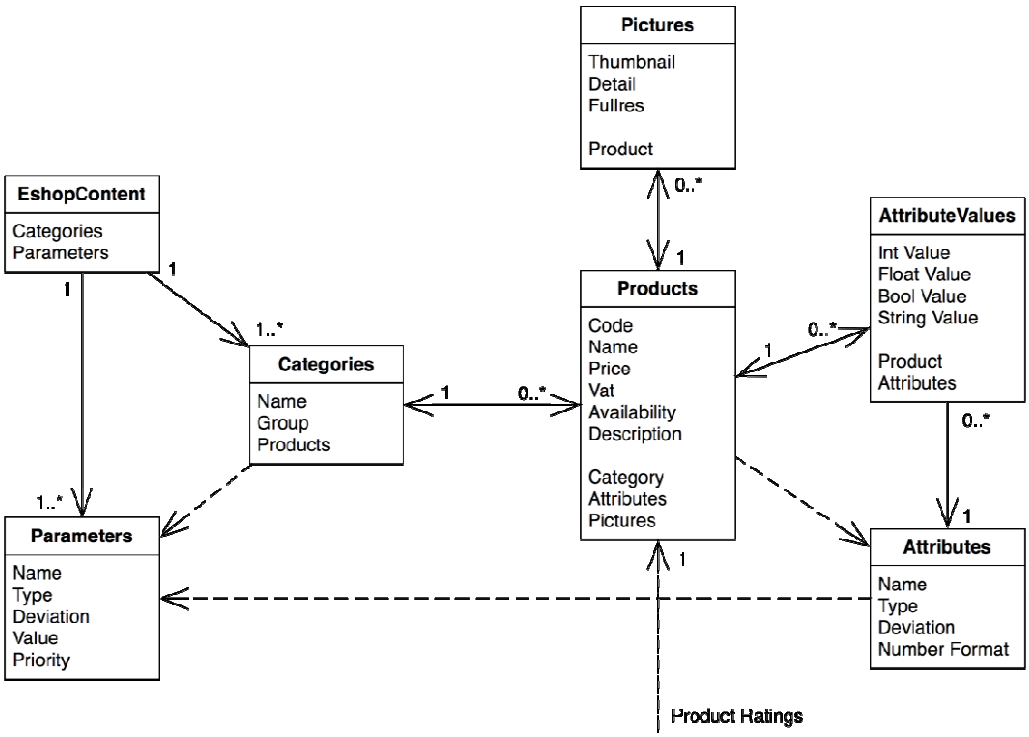


Diagram 2
Model datové struktury globálního obsahu aplikace

Obsah nabídky webového obchodu začíná rozdělením produktů do kategorií. Každá kategorie je asociována s množinou parametrů, které popisují vlastnosti všech objektů v ní obsažených. Dá se proto předpokládat, že tato množina atributů je pro každou kategorii jiná a každý produkt v nabídce pak patří vždycky právě do jedné z těchto kategorií. Kategorie může být z hlediska prezentace v rozhraní obsažena v nějaké pojmenované skupině, nebo se může zobrazovat přímo na úrovni skupin kategorií. Kategorie v současné implementaci nejsou mezi sebou nijakým způsobem propojeny (např. za účelem monitorování souvisejících objektů napříč různými kategoriemi).

Produkt, jakožto základní reprezentace objektu, nad kterým jsou modely výpočtu preferencí navrženy, je tvořen jak množinou vlastností společných pro všechny produkty v nabídce (např. název, cena, obrázek), tak množinou hodnot atributů, popisujících specifické vlastnosti vzhledem ke kategorii do které je produkt zařazen. Hodnoty jednotlivých atributů pro vybraný produkt ovšem nejsou povinné, a pokud některá z hodnot chybí, může to mít vliv na výpočet skóre objektů některých z implementovaných modelů uživatelských preferencí.

Typy atributů objektů, které aplikace implementuje, dělíme na následující dva podtypy:

- Jednoduchý atribut oplývající jedinou hodnotou zvoleného základního typu
- Seznam hodnot základního typu

V rámci podporovaných základních typů hodnot atributů pak rozlišujeme:

- Číselný atribut

  Sem patří buď celočíselné typy atributů, nebo atributy obsahující čísla s plovoucí desetinnou čárkou. U každého číselného atributu je navíc možné určit, jestli může obsahovat i záporné hodnoty, nebo vyžaduje čísla pouze rovné nebo větší 0. Součástí každé definice číselného typu atributu je pak formátovací řetězec, sloužící pro bližší upřesnění hodnoty číselného údaje, typicky např. prostřednictvím uvedení zkratky měrných jednotek, které číselný údaj vyjadřuje.

- Logický atribut

  Tento typ atributu může nabývat pouze jedné z logických hodnot „Ano“ nebo „Ne“.

- Textový atribut

  Jedná se o standardní textový řetězec.

Pozn.: Aplikace v současné podobě dokáže pracovat nad seznamy hodnot všech základních typů, v praktickém využití se ovšem prokázaly jako užitečné pouze seznamy řetězců, proto se bude v následujícím textu uvažovat jako s možnými zástupci složených typů atributů pouze s nimi.

3.2.2 Uživatelské data v aplikaci

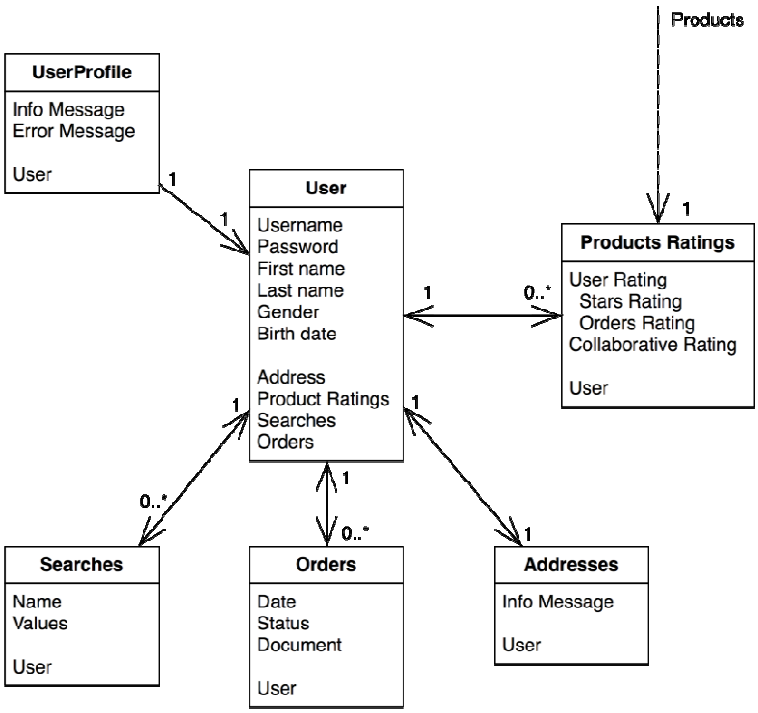


Diagram 3
Model datové struktury uživatelského profilu aplikace

Data aktuálního uživatelského profilu se s každým přihlášením a odhlášením uživatele do, resp. ze systému mění. Pro každého aktuálně přihlášeného uživatele existuje tudíž v paměti samostatní kopie datové struktury zodpovídajícího uživatelského profilu. Kromě pár stavových proměnných je tvořena především objektem aktuálně přihlášeného uživatele. Ten nese veškeré informace o osobních údajích, stejně jako informace o objednávkách, a hlavně hodnoty skóre pro všechny objekty nabídky vypočítány na základě implementovaných modelů preferencí. Tento model struktury dat umožňuje jednoduchou manipulaci s daty přihlášených uživatelů, aniž by bylo potřeba měnit obsah a strukturu dat celkové nabídky eshopu.

3.3 Vybrané modely preferencí

V rámci modelů preferencí zvolených pro implementaci se rozlišují dva základní uživatelské přístupy, které implementovaná aplikace využívá. První předpokládá, že uživatel přesně zná své preference v rámci zvolené kategorie objektů a je schopný je vyjádřit ve formě podmínek na hodnoty vybraných atributů. Preferenční data jsou tudíž získané přímou cestou a výstup výpočtu tak může dosáhnout vyšší přesnosti. Druhý přístup naopak předpokládá, že uživatel nemusí umět popsat nebo předat systému své preference přímým způsobem, nemusí dokonce znát ani důvody některých ze svých preferencí. V tomto případě je cílem systému nabídnout mu potenciálně zajímavé objekty na základě preferenčních dat, které je schopen uživatel systému sdělit.

3.3.1 Vybraný model uživatelských preferencí nad daty získanými přímým způsobem

Jak se dá předpokládat, tento model v zásadě implementuje metodu popsanou v kapitole 2.5.30. Jedná se tudíž o variantu vyhledávání objektů podle parametrů, která ovšem neořezává výsledkovou sadu, nýbrž pouze hledá takové uspořádání produktů, které co nejlépe vyhovuje zadaným podmínkám. Možnost ořezání výsledkové sady je ovšem zachována z důvodu zohlednění běžných uživatelských návyků, v případě že o ni uživatel přímo zažádá. Použitý model navíc implementuje možnost přiřazení priorit k jednotlivým podmínkám, co zvyšuje flexibilitu zaznamenání specifických uživatelských preferencí. Systém nabízí tři úrovně priorit, co považuji za vhodný kompromis vzhledem k želané úrovni složitosti rozhraní a potřeby většiny jeho uživatelů. Změny počtu podmínek, stejně jako hodnot jejich váhových koeficientů jde však docílit relativně snadnou úpravou zdrojového kódu aplikace.

Skóre objektu

Výsledné uspořádání objektů je stanoveno na základě hodnot celkového vypočteného skóre pro všechny objekty vstupní sady. Pro množinu podmínek c_i , $i = 1, \dots n$ specifikovaných uživatelem u se pak skóre $R(o,u)$ objektu o počítá jako

$$R(o,u) = \left(\sum_{i=1}^{n} p_{c_i} r(c_i, a_i) \right) \Big/ \sum_{i=1}^{n} p_{c_i}$$

- p_{c_i} je váhový koeficient priority podmínky c_i
- $r(c_i, a_i)$ je dílčí skóre reprezentující míru vyhovění hodnoty atributu a_i podmínce c_i

Pokud uživatel specifikuje požadavek na ořezání výsledkové sady podle jedné ze zadaných podmínek, množina vstupních objektů je zredukována o všechny objekty, pro které hodnota $r(c_j, a_j)$, $j \in \{1 \dots n\}$ pro zadanou podmínku c_j nenabývá maximálního možného skóre. Tato podmínka je pak z množiny podmínek c_i , $i = 1, \dots n$ odebrána a ve výpočtu celkového skóre objektu $R(o,u)$ se s ní dále neuvažuje.

Skóre atributů objektu

Metody výpočtu skóre pro jednotlivé hodnoty atributů objektů se liší v závislosti od aktuálního typu atributu.

Výpočet skóre číselných atributů

Podmínka pro atributy obsahující celá čísla nebo čísla s plovoucí desetinnou čárkou je vždy reprezentována intervalem $[c_{min}, c_{max}]$, případně intervaly $[c_{min}, +\infty)$ nebo $(-\infty, c_{max}]$, definujícími rozsah požadovaných hodnot. Podmínka na jedinou konkrétní hodnotu c_{val} se zřejmě převádí na interval $[c_{val}, c_{val}]$. Skóre $r(c,a)$ hodnoty atributu a pro podmínku c se pak počítá následovně:

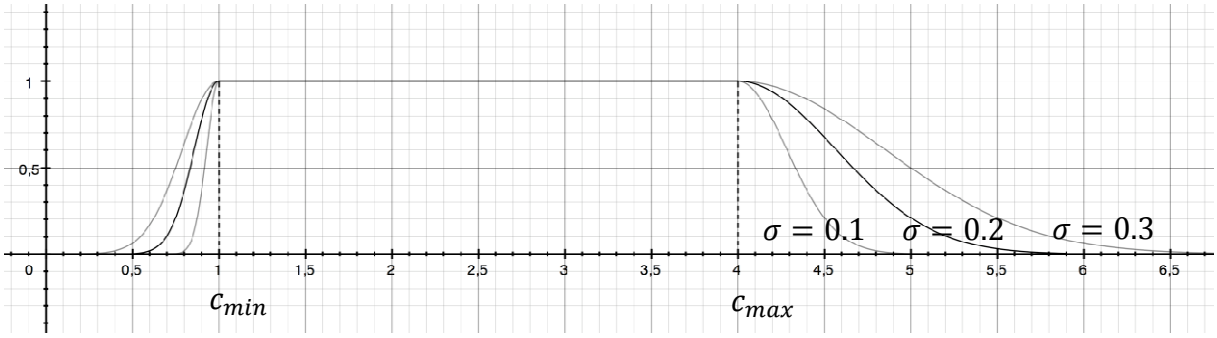
- Pro $a \in [c_{min}, c_{max}]$ nebo $a \in [c_{min}, +\infty)$ nebo $a \in (-\infty, c_{max}]$ platí $r(c,a) = 1$
- Pro $a \in (-\infty, c_{min})$ se $r(c,a)$ počítá jako

$$r(c,a) = exp\left(-\frac{(a - c_{min})^2}{(\sigma c_{min})^2}\right)$$

- Pro $a \in (c_{max}, +\infty)$ se $r(c,a)$ počítá jako

$$r(c,a) = exp\left(-\frac{(a - c_{max})^2}{(\sigma c_{max})^2}\right)$$

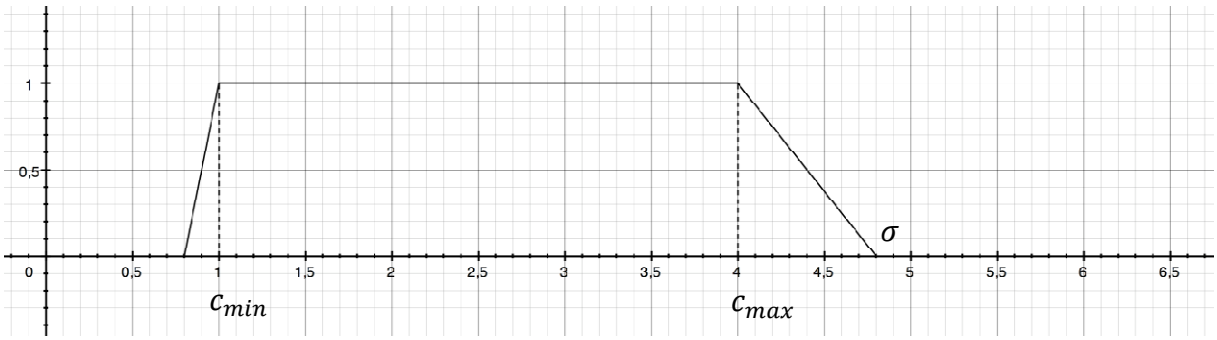
- hodnota σ reprezentuje povolenou míru odchylky a od požadovaného intervalu stanoveného podmínkou c a je součástí definice každého číselného atributu. Důvodem implementace σ je rozličné vnímání závažnosti odchylky hodnoty a od požadovaného intervalu a jejího možného dopadu na výpočet ohodnocení pro jednotlivé typy atributů.
  (Například míra povolené odchylky v počtu megabajtů u harddisků je zřejmě větší než míra vhodné odchylky pro rok výroby automobilů.)



Graf 1
Tvar hodnotící funkce pro vstupní podmínku danou intervalem $[1, 4]$

Pozn.: Tato metoda výpočtu skóre číselných atributů je mírně odlišná od původní verze návrhu, který byl založen na následujícím modelu výpočtu:

- Pro $a \in [c_{min}, c_{max}]$ nebo $a \in [c_{min}, +\infty)$ nebo $a \in (-\infty, c_{max}]$ platí $r(c,a) = 1$
- Jestli $a \in (-\infty, c_{min}(1 - \sigma)] \cup [c_{max}(1 + \sigma), +\infty)$, pak $r(c,a) = 0$
- Jestli $a \in (c_{min}(1 - \sigma), c_{min}) \cup \big(c_{max}, c_{max}(1 + \sigma)\big)$, pak $r(c,a) \in (0,1)$



Graf 2
Tvar původní hodnotící funkce pro vstupní podmínku $[1, 4]$

Jak se později ukázalo, tento model výpočtu skóre nesplňoval dostatečně požadavky na něj kladené, protože nedokázal vyhodnotit rozdíl mezi dvěma atributy, u kterých byly sice obě hodnoty mimo požadovaný interval $[c_{min}, c_{max}]$ i mimo interval jeho širšího okolí $\big(c_{min}(1 - \sigma), c_{max}(1 + \sigma)\big)$, v porovnání mezi sebou byly ale jedna od tohoto intervalu vzdálená v mnohem větší míře než druhá. Model výpočtu pak preferoval v některých případech výchozí pořadí objektů před pořadím, které by těmto hodnotám atributů logicky více nasvědčovalo.

Výpočet skóre logických atributů

Pro atributy nabývající pouze logických hodnot „Ano“ nebo „Ne“ specifikuje podmínka vždy pouze jednu z těchto hodnot c_{bool} a metoda výpočtu skóre $r(c, a)$ se řídí relativně předvídatelnými instrukcemi:

- Jestli $a = c_{bool}$, pak $r(c, a) = 1$
- Jestli $a \neq c_{bool}$, pak $r(c, a) = 0$

Výpočet skóre textových atributů

Pro textové typy atributů je podmínka vyjadřující zvolenou uživatelskou preferenci vždy reprezentována množinou textových hodnot $c_{s_1}, \dots c_{s_n}$. Metoda výpočtu výsledného skóre pak závisí od typu atributu, jestli tento nabývá pouze jediné textové hodnoty, nebo je tvořen seznamem více hodnot.

Pro jednoduchý textový atribut se výsledné skóre $r(c, a)$ stanoví následovně:

- Pokud $\exists i, i \in \{1, \dots n\}$ takové, že $c_{s_i} = a$, pak $r(c, a) = 1$
- Pokud $\forall j, j = 1, \dots n$ platí, že $c_{s_j} \neq a$, pak $r(c, a) = 0$

V případě atributu typu seznamu řetězců, kde jeden atribut může nabývat více textových hodnot $a_1, \dots a_m$, se hodnota $r(c, a)$ pro zmíněný atribut vypočítá podle

$$r(c, a) = \left(\sum_{i=1}^{m} r(c, a_i) \right) / n$$

- n je počet prvků množiny textových hodnot $\{c_{s_1}, \dots c_{s_n}\}$ reprezentujících podmínku c popisující vybranou uživatelskou preferenci nad atributem a

Výpočet dílčích skóre $r(c, a_i)$ se pak řídí pravidly pro výpočet $r(c, a)$ v případě jednoduchého typu textového atributu, popsanými výše.

Výpočet skóre chybějících atributů

Pokud hodnota atributu $a_j, j \in \{1 \dots k\}$ aktuálně vyhodnocovaného objektu pro podmínku c_j chybí, pak vždycky platí $r(c_j, a_j) = 0$.

3.3.2 Vybraný model uživatelských preferencí nad daty získanými explicitně a implicitně

Pokud uživatel neví nebo není ochoten specifikovat podmínky popisující jeho preference nad zvolenou kategorií objektů, využívá aplikace pro výpočet doporučení kombinaci modelů preferencí popsaných v kapitole 2.5.2. Musí proto disponovat určitou množinou vstupních preferenčních dat, které zaznamenává v průběhu práce uživatele se systémem prostřednictvím některých explicitních a implicitních metod popsaných v zmíněné kapitole.

Explicitní preference uživatele

Explicitní metoda záznamu preferencí je tvořena prvkem rozhraní, umožňujícím přidělit vybranému objektu skóre pomocí volby zodpovídajícího počtu hvězdiček z intervalu 0 – 5. Rozsah tohoto intervalu jde opět relativně jednoduchým zásahem do zdrojového kódu aplikace změnit, pět hvězdiček bylo ale stanoveno jako vhodný kompromis mezi flexibilitou hodnocení a množstvím, při kterém je pravděpodobnost vzniku nekonzistence jednotlivých hodnocení ještě relativně nízká. Zvolený počet hvězdiček je taky v souladu s existujícími návyky uživatelů, protože se nejčastěji používá i v jiných, nejen webových, systémech. Přidělený počet hvězdiček c_{stars} se pak převádí na hodnotu odpovídajícího skóre $S^E(o,u)$ objektu o pro uživatele u z intervalu $[0,1]$ pomocí vztahu

$$S^E(o,u) = c_{stars}\left(\frac{1}{n}\right)$$

- n je maximální možný počet přidělených hvězdiček

Aplikace v současné podobě neimplementuje normalizaci hodnocení jednotlivých uživatelů, spočívající v analýze průměrného hodnocení každého z nich a ve snaze o sjednocení absolutní hodnoty skóre těchto průměrů; pro maximálně pět hvězdiček však zahrnutí této funkcionality není až tak důležité, protože odchylky v hodnocení jednotlivých uživatelů nebývají velké. V případě malé množiny získaných preferencí pro konkrétního uživatele může mít navíc tato normalizace spíše negativní dopad na přesnost výpočtu výsledného skóre objektů.

Implicitní preference uživatele

Implicitní metoda, jak z jejího názvu vyplývá, naopak uživatele do žádné činnosti vědomého ohodnocování oblíbenosti objektů nenutí a v prostředí aplikace proto taky nedisponuje žádným dedikovaným prvkem rozhraní. Metod vyhodnocování uživatelských preferencí na základě monitorování aktivity uživatele je mnoho, aktuální implementace se ovšem omezuje pouze na funkcionalitu zakoupení produktu, která výsledné skóre produktu navýší na maximum. Formálně tedy můžeme napsat, že pro hodnocení $S^I(o,u)$ objektu o pro uživatele u platí, že

- pokud byl objekt o někdy v průběhu existence uživatelského profilu uživatele u zakoupen tímto uživatelem, pak $S^I(o,u) = 1$,
- v opačném případě $S^I(o,u) = 0$

Celkové skóre objektu

Jak již bylo zmíněno, pro výpočet výsledného skóre objektu pro aktuálního uživatele se využívá kombinace několika základních modelů uživatelských preferencí, zohledňujících jak preference vyjádřené explicitně nebo implicitně pouze aktuálním uživatelem, tak preference jiných uživatelů, kteří s aktuálním uživatelem sdílí podobný uživatelský vkus. Model výpočtu tudíž implementuje metodu kolaborativního filtrování, popsanou obecněji v kapitole 2.5.2.

Přesný výpočet celkového ohodnocení $S(o,u)$ objektu o pro uživatele u se pak řídí vztahem

$$S(o,u) = p_{S^U}\left(S^U(o,u)\right) + p_{S^C}\left(S^C(o,u)\right)$$

- $S^U(o,u)$ je skóre objektu získané výpočtem nad lokálními preferencemi uživatele
- $S^C(o,u)$ je skóre objektu získané metodou kolaborativního filtrování
- p_{S^U}, p_{S^C} jsou hodnoty váhových koeficientů pro jednotlivé složky výpočtu, přičemž vzhledem k absenci normalizační úpravy výsledného skóre musí platit $p_{S^U} + p_{S^C} = 1$

Pozn.: Hodnoty p_{S^U} , p_{S^C} jde v implementaci snadno změnit úpravou třídy konstant využívaných aplikací, jejich výchozí hodnoty byly stanoveny na 0.7 a 0.3.

Výpočet skóre objektu nad lokálními preferencemi uživatele

Lokální preference uživatele jsou tvořeny daty získanými explicitním a implicitním způsobem, výslední skóre nad těmito preferencemi je tudíž tvořeno kombinací vypočtených skóre pro oba způsoby získávání preferencí, tak jak jsou popsány v kapitolách 2.4.2 a 2.4.3. Výsledná hodnota skóre $S^U(o,u)$ objektu o vzhledem k lokálně specifikovaným preferencím uživatele u se pak počítá jako

$$S^U(o,u) = p_{S^E}\left(S^E(o,u)\right) + p_{S^I}\left(S^I(o,u)\right)$$

- $S^E(o,u)$ je skóre objektu vypočteno na základě explicitních preferencí uživatele (tvořených v současné podobě implementace počtem přidělených hvězdiček)
- $S^I(o,u)$ je skóre objektu získané na základě implicitních preferencí uživatele (reprezentovaných v současné podobě aplikace informací o zakoupení objektu)
- p_{S^E}, p_{S^I} jsou hodnoty váhových koeficientů pro jednotlivé složky výpočtu, přičemž vzhledem k absenci normalizace výsledného skóre musí platit, že $p_{S^E} + p_{S^I} = 1$

Pozn.: Hodnoty p_{S^E} , p_{S^I} jde v implementaci snadno změnit úpravou třídy konstant využívaných aplikací, jejich výchozí hodnoty byly stanoveny na $0.5/0.7$ a $0.2/0.7$, což má za následek nárůst celkového skóre objektu $S(o,u)$ přesně o 10% pro každou přidělenou hvězdičku vyjadřující explicitní preferenci uživatele. Důvodem je možnost intuitivnějšího vnímání vztahu mezi počtem přidělených hvězdiček a výsledném skóre objektu uživatelem.

Výpočet kolaborativní složky celkového skóre objektu

Zvolená metoda kolaborativního filtrování zohledňující nejen preference aktuálního uživatele, ale i preference ostatních uživatelů v systému, pracuje ve dvou krocích. V prvním kroku stanoví pro všechny uživatele v systému, s výjimkou aktuálního uživatele, míru shody

jejich uživatelského vkusu se vkusem aktuálního uživatele. Z množiny těchto uživatelů je pak vybraná podmnožina, obsahující jenom uživatele se vkusem co nejvíce podobným vkusu uživatele aktuálního. Preference těchto uživatelů jsou pak použity pro výpočet kolaborativní složky hodnocení objektů pro aktuálního uživatele.

Míra shody preferencí $M(u, u^j)$ aktuálního uživatele u s preferencemi uživatele u^j, $j \in \{1, \dots m\}$, $u \neq u^j$, pro množinu objektů o_i, $i = 1, \dots n$ se vypočítá pomocí

$$M(u, u^j) = 1 - \left(\sum_{i=1}^{n} \left| S^U(o_i, u) - S^U(o_i, u^j) \right| / n \right)$$

- $S^U(o, u)$ je skóre objektu o uživatele u zohledňující lokální preference tohoto uživatele, jehož postup výpočtu byl popsán výše

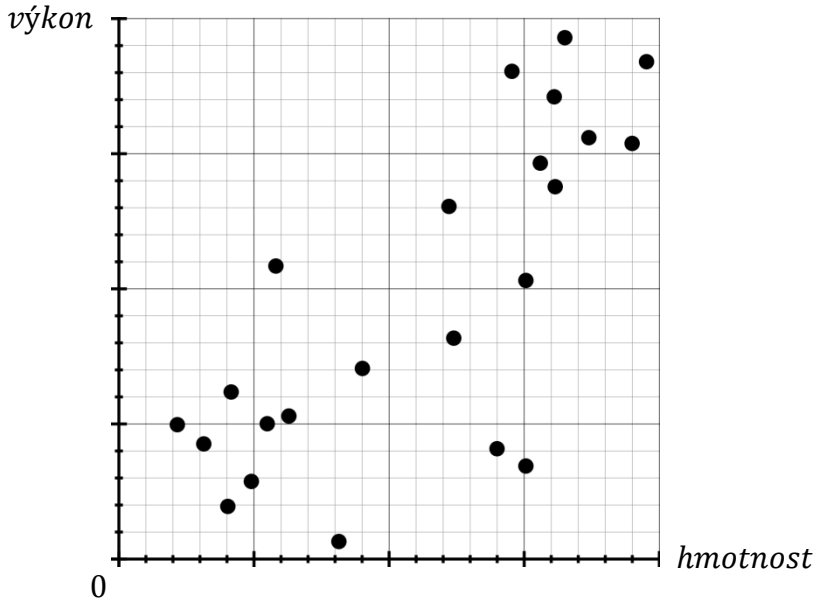
Analyzuje se tudíž míra rozdílnosti jednotlivých skóre vypočtených nad lokálními preferencemi obou uživatelů pro všechny objekty o_i, typicky představující obsah jedné z kategorií produktů nabídky eshopu.

Výsledná hodnota kolaborativního skóre $S^C(o, u)$ pro objekt o uživatele u se pak vypočítá pomocí

$$S^C(o, u) = \sum_{i=1}^{p} \left(\frac{M(u, u^i)}{\sum_{j=1}^{p} M(u, u^j)} \right) S^U(o, u^i)$$

- u^i, u^j jsou uživatelé podobní aktuálnímu uživateli u co do vlastností uživatelského vkusu, kteří byli vybráni na základě hodnot $M(u, u^j)$ vypočtených v předcházejícím kroku
- p je počet uživatelů, s hodnotami kterých skóre nad lokálními preferencemi se ve výpočtu kolaborativního skóre počítá, přičemž zřejmě $p < m$, kde m reprezentuje počet všech uživatelů systému

Hodnota p jde jednoduchou úpravou zdrojového kódu aplikace změnit, její výchozí hodnota byla stanovena na 5. Její implementace do modelu výpočtu kolaborativního skóre objektů má za následek, že výsledná hodnota skóre není pouhým průměrem skóre nad lokálními preferencemi všech uživatelů systému, váženým mírou jejich shody se vkusem aktuálního uživatele, nýbrž se tato výsledná hodnota víc přiklání k aktuálním preferencím menší množiny uživatelů, co může vyústit k jednoznačnějšímu výslednému pořadí produktů, přesněji popisujícímu aktuální preference uživatele. Předpokladem pro vyšší úspěšnost tohoto modelu výpočtu je fakt, že rozdíly v uživatelských preferencích nejsou mezi uživateli distribuovány rovnoměrně po celé možné škále, nýbrž jsou situovány do shluků. Existuje tak množina uživatelů s velmi podobnými preferencemi, které jsou zase odlišné od preferencí jiné množiny podobných uživatelů. Hodnota proměnné p pak dokáže stanovit míru, s jakou bude výsledné skóre asociováno s hodnotami preferencí jednoho z těchto preferenčních shluků.



Graf 3
Příklad preferencí uživatelů pro množinu notebooků vzhledem ke dvěma vybraným atributům objektů této množiny – hmotnosti a výkonu.

Vzhledem ke zřejmé časové náročnosti výpočtu kolaborativní složky hodnocení objektů, stoupající s každým nově vzniklým uživatelem v systému, implementuje aplikace výpočet kolaborativního skóre pouze v momentě přihlášení aktuálního uživatele do systému. Pro aktualizaci kolaborativní složky celkového skóre objektů je proto nutné se do zodpovídajícího uživatelského profilu opětovně přihlásit.

3.4 Požadavky na systém

Požadavky na funkčnost systému vyplývají zejména z typu implementované aplikace a rozsahu řešené úlohy:

- Webová aplikace nezávislá od platformy a použitého prohlížeče
- Implementace vybraných modelů uživatelských preferencí popsaných v kapitole 3.3
- Uživatelské rozhraní dodržující základní uživatelské návyky získané z existujících prostředí webových obchodů
- Dostatečně velká a různorodá množina testovacích dat
- Schopnost záznamu všech uživatelských akcí z důvodu možnosti jejich následného statistického vyhodnocení

Součástí aplikace není administrátorské rozhraní sloužící ke změně globálních parametrů běhu aplikace nebo k editaci struktury a obsahu nabídky webového obchodu.

3.5 Použité technologie

Jako programovací jazyk serverové části aplikace byl zvolen jazyk Java [5], který je implementačně nezávislý od operačního systému, čímž splňuje první požadavek platformní nezávislosti nejen pro webové rozhraní, ale taky pro serverovou část aplikace. Volba aplikačního serveru padla na volně dostupný server Apache Tomcat [8] ve verzi 6.

Generování prezentační vrstvy je zajištěno technologií JSP (JavaServer Pages) [6] a JSTL (JavaServer Pages Standard Tag Library), pro vyšší přehlednost a možnost opětovného využití segmentů kódu (typicky některých komplexnějších prvků rozhraní) se využívá možnosti tvorby vlastních Custom Tagů.

Architektura aplikace se řídí modelem MVC (Model-View-Controller), který odděluje prezentační vrstvu od logiky aplikace a jejího datového modelu, což dělá aplikaci modulární a zjednodušuje možnosti rozšíření do budoucna.

Volba relační databáze uchovávající veškerá data aplikace (kromě logů zaznamenávajících činnost uživatele) padla na databázi MySQL, která splňuje podmínku platformní nezávislosti, je volně dostupná a pro oblast webových obchodů také dostatečně výkonná. Pro přístup do databáze se využívá standardní aplikační programové rozhraní Java Database Connectivity (JDBC) [7].

Prezentační vrstva je tvořena XHTML 1.0 Strict kódem a funguje ve většině moderních internetových prohlížečů, jako jsou Microsoft Internet Explorer 8, Opera, prohlížeče založené na jádře Gecko (např. Mozilla Firefox) i na jádře Webkit (Apple Safari, Google Chrome). Dynamické chování na straně klienta je zabezpečeno skripty v jazyce Javascript s použitím frameworku Prototype [9] s nadstavbou Script.aculo.us [10], styly zobrazení jsou definováni pomocí CSS. Pro zobrazení detailu obrázku produktu se používá volně šiřitelná komponenta LightBox.

3.6 Architektura aplikace

3.6.1 Datový model aplikace

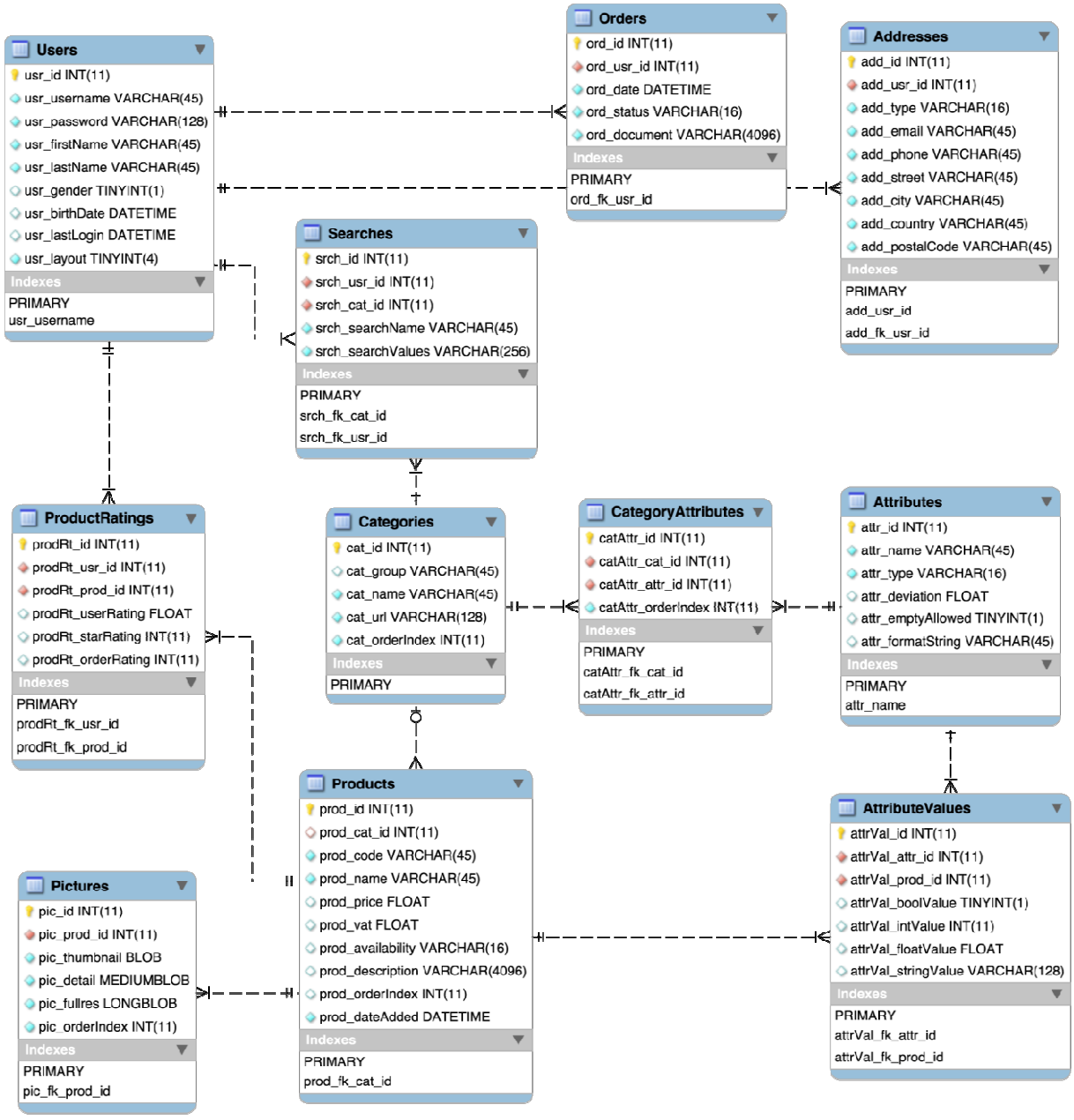


Diagram 4
Datový model aplikace.

Jak je patrné, ER model (Entity Relationship) použité databáze přibližně odpovídá modelu datové struktury popsané v kapitole 3.2. Vzhledem k tomu, že účel většiny polí se dá odhadnout z jejich názvu nebo datového typu, obsahuje tato kapitola pouze popis využití jednotlivých tabulek, případně některých jejich vlastností, které nemusí být od pohledu na uvedený ER diagram zřejmé.

Globální obsah

- *Categories* – kategorie produktů nabídky webového obchodu
- *Attributes* – množina atributů produktů, asociována také s jednotlivými kategoriemi eshopu
    - *attr_deviation* – míra povolené odchylky σ popsaná v kapitole 3.3.1
    - *attr_formatString* – formátovací řetězec, nesoucí doplňující údaj pro číselné typy atributů (viz kapitola 3.2.1)
- *CategoryAttributes* – spojovací tabulka pro atributy a kategorie
- *AttributeValues* – hodnoty atributů produktů

  V závislosti od aktuálního typu atributu se vždy využívá pouze jeden ze sloupců *attrVal_intValue*, *attrVal_floatValue*, *attrVal_boolValue* nebo *attrVal_stringValue*. Pokud typ atributu definuje jeho hodnotu jako seznam hodnot základního typu, je tento seznam reprezentován jedním záznamem v tabulce *AttributeValues* pro každou hodnotu seznamu.

  Existence samostatných tabulek *Attributes* a *AttributeValues* umožňuje sdílení definovaných atributů přes více kategorií, co může být výhodné, pokud chceme ten samý atribut využít ve více kategoriích, nebo pokud chceme např. globálně změnit jeho název nebo hodnotu povolené odchylky σ.

- *Products* – produkty nabídky webového obchodu
    - *prod_vat* – výška daně z přidané hodnoty, vyjádřena koeficientem pro výpočet výsledné ceny
    - *prod_description* – popis produktu

      Popis produktu je tvořen segmentem XHTML kódu, povolené značky (tagy) jsou `<p>`,`<h1>`,`<ul>`,`<li>`,`<a>`. V případě detailního módu zobrazení všech produktů se pro popis použije obsah prvního elementu odstavce `<p></p>`.

- *Pictures* – obrázky produktů nabídky

  Obsažení obrázků v databázi umožňuje např. jednoduchou zálohu obsahu, protože všechny relevantní data jsou v databázi přítomná a jakékoliv zálohování souborů je možné vynechat. Obrázky se zkopírují do souborového systému při prvním požadavku na vykreslení stránky aplikace.

Uživatelské data

- *Users* – uživatelé registrováni v systému
- *Addresses* – adresy registrovaných uživatelů
- *Orders* – objednávky uživatelů

  Objednávky jsou v současné implementaci reprezentovány vygenerovaným dokumentem, obsahujícím všechny důležité informace. Vazba na původní produkty se nezachovává.

- *Searches* – přímé specifikace preferencí, které se uživatel rozhodl uložit pro budoucí použití

  Bližší popis této funkcionality viz kapitola 3.7.2

- *ProductRatings* – skóre produktů uživatele vypočítané implementovanými modely preferencí
  - *prodRt_userRating* – skóre objektů nad lokálními preferencemi uživatele Tato hodnota zodpovídá $S^U(o,u)$ popsané v kapitole 3.3.2.

    Skóre se aktualizuje při každé změně hodnot její dílčích složek. Důvodem je zvýšení efektivity výpočtu kolaborativního skóre, které pak již nemusí počítat dílčí složky uživatelského skóre pro všechny produkty všech uživatelů. Pokud se ovšem změní hodnoty váhových koeficientů p_{S^E}, p_{S^I}, je nutné tyto hodnoty pro všechny objekty všech uživatelů aktualizovat.

  - *prodRt_starRating* – počet hvězdiček přidělených objektu uživatelem

    V aktuální implementaci je tato hodnota reprezentována celým číslem, jedná se tudíž o dílčí hodnotu $S^E(o,u)$ z kapitoly 3.3.2, ze které se výsledná hodnota skóre počítá

  - *prodRt_orderRating* – skóre nad implicitními preferencemi uživatele, tvořenými informací o zakoupení produktu

    Jedná se o hodnotu $S^I(o,u)$ popsanou v kapitole 3.3.2

  Kolaborativní složka výsledného skóre objektů se do databáze neukládá, vyhodnocuje se v momentě přihlášení uživatele do systému.

3.6.2 Model architektury aplikační vrstvy

Struktura balíčků aplikace

Diagram 5 znázorňuje organizaci a propojení Java balíčků ve zdrojovém kódu serverové části aplikace. Jednotlivé Java balíčky obsahují názvy obsažených tříd, které budou detailněji popsány v následující kapitole.

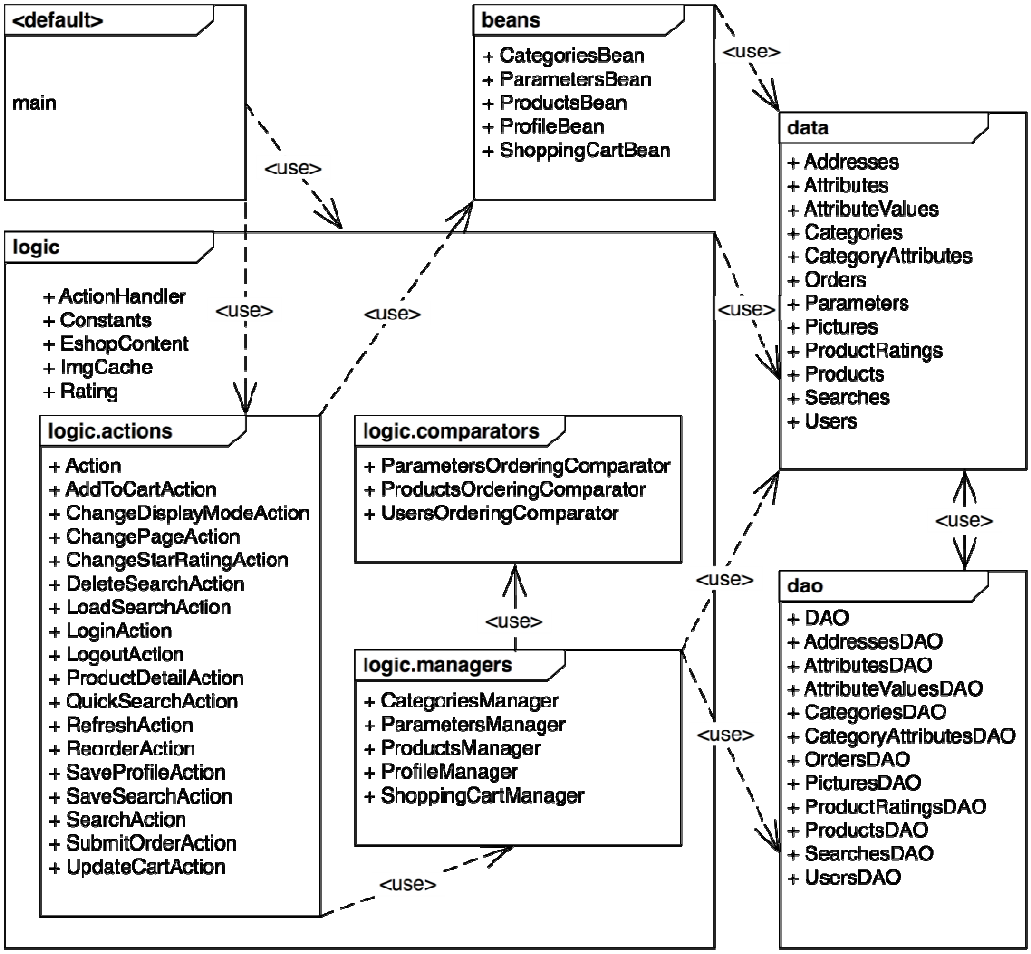


Diagram 5

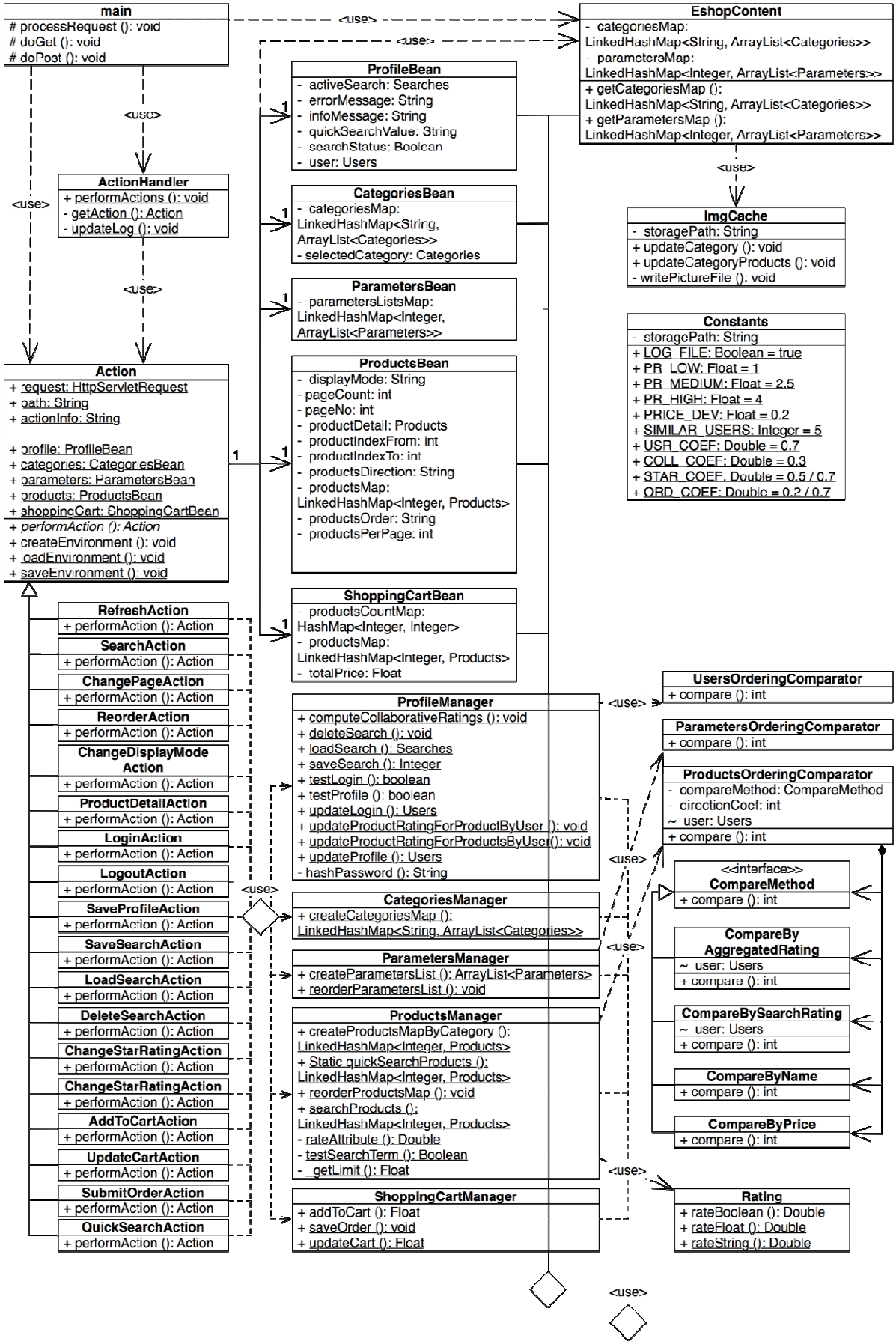
Model Java balíčků aplikace

Aplikaci obsluhuje hlavní servlet *main*, nacházející se v nejvyšší úrovni její struktury. Provádění všech uživatelských akcí, výpočtů a manipulací s daty zabezpečuje balíček *logic*, implementující veškerou výpočetní logiku aplikace. V závislosti od charakteru obsažených tříd se tento balíček dále člení na balíčky *logic.actions*, *logic.managers* a *logic.comparators*. Data jsou uchovávané v instancích tříd balíčku *data*, načítání dat z databáze zabezpečují metody obsažené v balíčku *dao*. Balíček *beans* pak slouží jako kontejner pro data, které jsou prostřednictvím technologie JSP vkládány do prezentační vrstvy aplikace.

Model tříd aplikace

Následující diagram znázorňuje strukturu aplikace, co se týče obsažených tříd a jejich vzájemných vztahů. Vzhledem k prostorovému omezení jsou některé údaje z diagramu vynechány, jedná se zejména o:

- Konstruktory tříd
- Vstupní parametry všech obsažených metod (pro diferenciaci atributů a metod objektů se používá konstrukce „()“ uvedené za jménem příslušné metody)
- *Get* a *Set* metody obsažených atributů
- Některé méně důležité nebo v aktuální verzi nepoužívané atributy nebo metody tříd


main
processRequest (): void
doGet (): void
doPost (): void
EshopContent
- categoriesMap: LinkedHashMap<String, ArrayList<Categories>>
- parametersMap: LinkedHashMap<Integer, ArrayList<Parameters>>
+ getCategoriesMap (): LinkedHashMap<String, ArrayList<Categories>>
+ getParametersMap (): LinkedHashMap<Integer, ArrayList<Parameters>>
ActionHandler
+ performActions (): void
- getAction (): Action
- updateLog (): void
ImgCache
- storagePath: String
+ updateCategory (): void
+ updateCategoryProducts (): void
- writePictureFile (): void
Constants
- storagePath: String
+ LOG_FILE: Boolean = true
+ PR_LOW: Float = 1
+ PR_MEDIUM: Float = 2.5
+ PR_HIGH: Float = 4
+ PRICE_DEV: Float = 0.2
+ SIMILAR_USERS: Integer = 5
+ USR_COEF: Double = 0.7
+ COLL_COEF: Double = 0.3
+ STAR_COEF: Double = 0.5 / 0.7
+ ORD_COEF: Double = 0.2 / 0.7
ProfileBean
- activeSearch: Searches
- errorMessage: String
- infoMessage: String
- quickSearchValue: String
- searchStatus: Boolean
- user: Users
CategoriesBean
- categoriesMap: LinkedHashMap<String, ArrayList<Categories>>
- selectedCategory: Categories
ParametersBean
- parametersListsMap: LinkedHashMap<Integer, ArrayList<Parameters>>
ProductsBean
- displayMode: String
- pageCount: int
- pageNo: int
- productDetail: Products
- productIndexFrom: Int
- productIndexTo: int
- productsDirection: String
- productsMap: LinkedHashMap<Integer, Products>
- productsOrder: String
- productsPerPage: int
ShoppingCartBean
- productsCountMap: HashMap<Integer, Integer>
- productsMap: LinkedHashMap<Integer, Products>
- totalPrice: Float
Action
+ request: HttpServletRequest
+ path: String
+ actionInfo: String
+ profile: ProfileBean
+ categories: CategoriesBean
+ parameters: ParametersBean
+ products: ProductsBean
+ shoppingCart: ShoppingCartBean
+ performAction (): Action
+ createEnvironment (): void
+ loadEnvironment (): void
+ saveEnvironment (): void
RefreshAction
+ performAction (): Action
SearchAction
+ performAction (): Action
ChangePageAction
+ performAction (): Action
ReorderAction
+ performAction (): Action
ChangeDisplayMode Action
+ performAction (): Action
ProductDetailAction
+ performAction (): Action
LoginAction
+ performAction (): Action
LogoutAction
+ performAction (): Action
SaveProfileAction
+ performAction (): Action
SaveSearchAction
+ performAction (): Action
LoadSearchAction
+ performAction (): Action
DeleteSearchAction
+ performAction (): Action
ChangeStarRatingAction
+ performAction (): Action
ChangeStarRatingAction
+ performAction (): Action
AddToCartAction
+ performAction (): Action
UpdateCartAction
+ performAction (): Action
SubmitOrderAction
+ performAction (): Action
QuickSearchAction
+ performAction (): Action
ProfileManager
+ computeCollaborativeRatings (): void
+ deleteSearch (): void
+ loadSearch (): Searches
+ saveSearch (): Integer
+ testLogin (): boolean
+ testProfile (): boolean
+ updateLogin (): Users
+ updateProductRatingForProductByUser (): void
+ updateProductRatingForProductsByUser(): void
+ updateProfile (): Users
- hashPassword (): String
CategoriesManager
+ createCategoriesMap (): LinkedHashMap<String, ArrayList<Categories>>
ParametersManager
+ createParametersList (): ArrayList<Parameters>
+ reorderParametersList (): void
ProductsManager
+ createProductsMapByCategory (): LinkedHashMap<Integer, Products>
+ Static quickSearchProducts (): LinkedHashMap<Integer, Products>
+ reorderProductsMap (): void
+ searchProducts (): LinkedHashMap<Integer, Products>
- rateAttribute (): Double
- testSearchTerm (): Boolean
- getLimit (): Float
ShoppingCartManager
+ addToCart (): Float
+ saveOrder (): void
+ updateCart (): Float
UsersOrderingComparator
+ compare (): int
ParametersOrderingComparator
+ compare (): int
ProductsOrderingComparator
- compareMethod: CompareMethod
- directionCoef: int
~ user: Users
+ compare (): int
<<interface>> CompareMethod
+ compare (): int
CompareBy AggregatedRating
~ user: Users
+ compare (): int
CompareBySearchRating
~ user: Users
+ compare (): int
CompareByName
+ compare (): int
CompareByPrice
+ compare (): int
Rating
+ rateBoolean (): Double
+ rateFloat (): Double
+ rateString (): Double
<use>

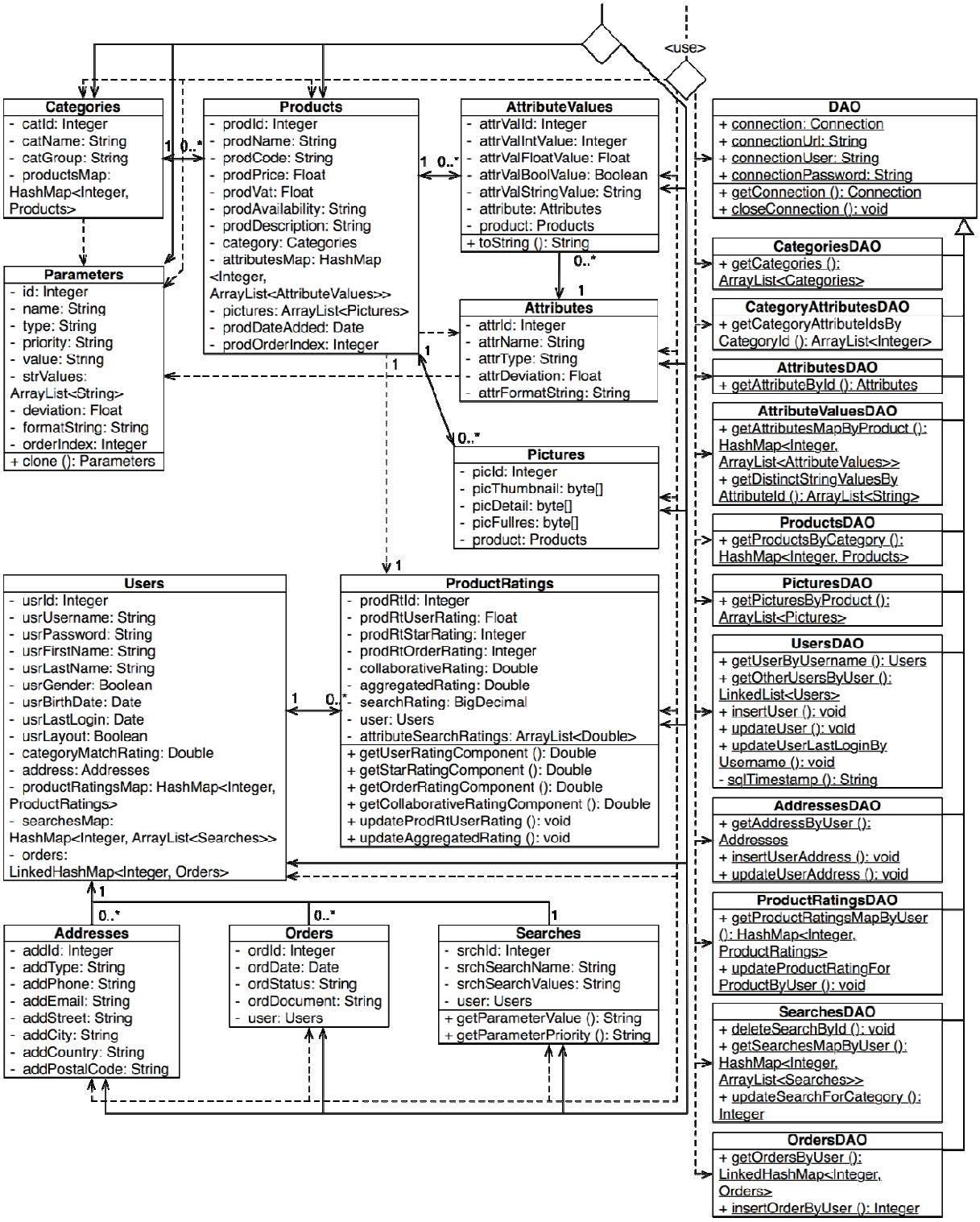


Diagram 6

Model Java tříd aplikace

Horní část diagramu se zaměřuje zejména na vizualizaci struktury business logiky aplikace, ve spodní části je pak obsažen hlavně datový model aplikace, který odpovídá konceptu představenému v kapitole 3.2.

Výpočetní logika aplikace

- *main* – hlavní servlet, obsluhující veškeré uživatelské požadavky přicházející z prohlížeče
- *EshopContent* – globální obsah aplikace, tvořený strukturou všech kategorií nabídky obchodu a strukturou jim odpovídajících parametrů
- *ActionHandler* – třída sloužící k určení a spouštění požadovaných uživatelských akcí
- *Action* – společný předek všech implementovaných akcí, poskytuje přístup k aktuálně využívaným datům relace, definuje abstraktní metodu *performAction()*
  - … *Action* – specifická akce aplikace, implementující metodu *performAction()*
- … *Bean* – objekty obsahující veškerá data potřebná pro správné vykreslení stránky
  - *ProfileBean* – objekt sdružující data ohledně aktuálního uživatelského profilu
  - *CategoriesBean* – data kategorií produktů
  - *ParametersBean* – data parametrů kategorií
  - *ProductsBean* – data obsahující informace ohledně aktuálně zobrazované množiny produktů
  - *ShoppingCartBean* – data nákupního košíku
- … *Manager* – metody definované v těchto třídách patří ke stěžejní součásti výpočetní logiky aplikace
  - *ProfileManager* – třída zastřešující operace nad daty aktuálního uživatelského profilu
  - *CategoriesManager* – třída operací nad daty kategorií produktů
  - *ParametersManager* – třída obsahující metody nad daty parametrů kategorií a produktů v nich obsažených
  - *ProductsManager* – třída operací nad daty aktuálně zobrazované množiny produktů
  - *ShoppingCartManager* – operace nad daty nákupního košíku
- *Rating* – třída definující metody výpočtu hodnocení pro různé typy atributů produktů
- … *Comparator* – metody definující pravidla pro požadované řazení objektů
  - *UsersOrderingComparator* – třída umožňující řazení uživatelů podle vypočtené hodnoty shody uživatelského vkusu s aktuálně přihlášeným uživatelem
  - *ParametersOrderingComparator* – komparátor řazení parametrů podle hodnot a priorit specifikovaných uživatelských podmínek
  - *ProductsOrderingComparator* – komparátor řazení produktů podle vybraného parametru zájmu
- *ImgCache* – třída umožňující přenos dat obrázků produktů z databáze do souborového systému serveru provozujícího aplikaci
- *Constants* – třída definující konstanty použité ve výpočtech

Data aplikace

- *Categories* – kategorie nabídky obchodu
- *Parameters* – parametry kategorií a všech v nich obsažených produktů
- *Products* – produkty nabídky obchodu
- *Attributes* – atributy produktů
- *AttributeValues* – hodnoty atributů jednotlivých produktů
- *Pictures* – obrázky produktů v různých velikostech; po zapsání do souborového systému existence objektů z důvodu uvolnění paměti zaniká
- *Users* – uživatelé registrovaní v systému
- *ProductRatings* – uživatelské hodnocení produktů
- *Addresses* – adresy uživatelů
- *Orders* – objednávky uživatelů
- *Searches* – uložená hledání uživatelů nad vybranými kategoriemi produktů
- *Dao* – společný předek všech implementovaných DAO tříd, poskytuje přístup k připojení k databázi
    - … *Dao* – třídy zabezpečující komunikaci a přenos dat mezi databázovou a aplikační vrstvou aplikace

Proces zpracování požadavku

Třída *main* je výchozím bodem veškeré výpočetní činnosti aplikace. Provedení jejího kódu je vyvoláno zodpovídajícím požadavkem prohlížeče, výsledná data jsou předána technologii zajišťující vytvoření výstupních HTML stránek. Obecný postup zpracování požadavků aplikace je stručně znázorněn na následujícím diagramu:

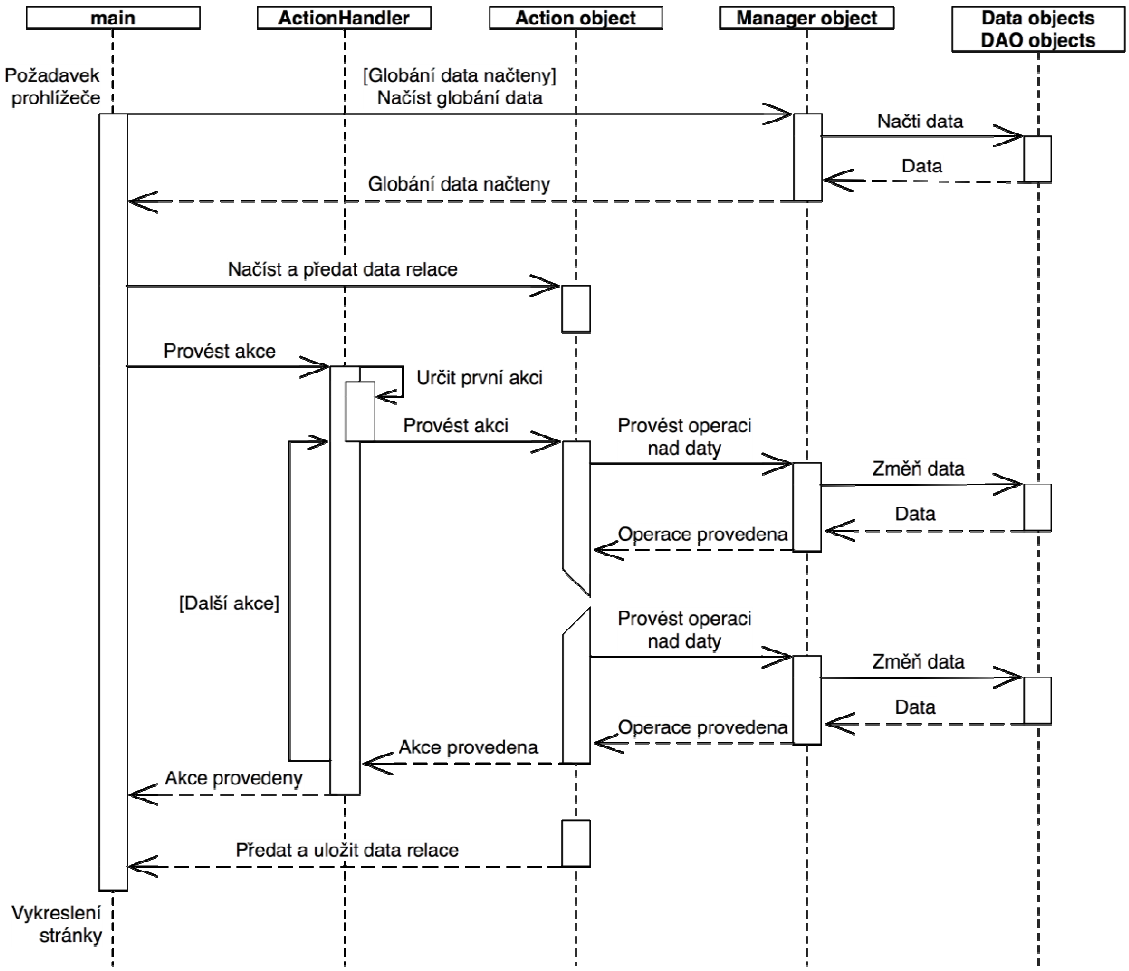


Diagram 7

Zpracování požadavku prohlížeče

- Aplikace zkontroluje strukturu globálních dat, pokud neexistují, zařídí jejich načtení z databáze
- Aplikace načte data aktuální relace, pokud neexistují vytvoří novou relaci a z důvodu bezpečnosti nastaví pro provedení výchozí akci aplikace
- Pomocí třídy *ActionHandler* je spuštěna sada akcí odpovídající aktuálnímu požadavku uživatele
- Aplikace uloží data aktuální relace a předá výstupní data technologii JSP

3.6.3 Model architektury prezentační vrstvy

Prezentační vrstva aplikace se skládá z JSP/XHTML kódu jednotlivých obrazovek, CSS souboru definujícímu vzhled a rozvržení jednotlivých prvků rozhraní a souborů v jazyce Javascript, obsahující definice objektů zajišťujících dynamické chování aplikace na straně klienta (v prohlížeči). Pro každou z obrazovek existuje odpovídající Javascriptový soubor, implementující funkci *pageLoad()*, která zajišťuje aktivaci veškerého dynamického chování pro tuto obrazovku. Zdrojový kód všech skriptů je oddělen od kódu stránek, co zvyšuje modularitu aplikace a zjednodušuje případnou implementaci změn a dalších rozšíření.

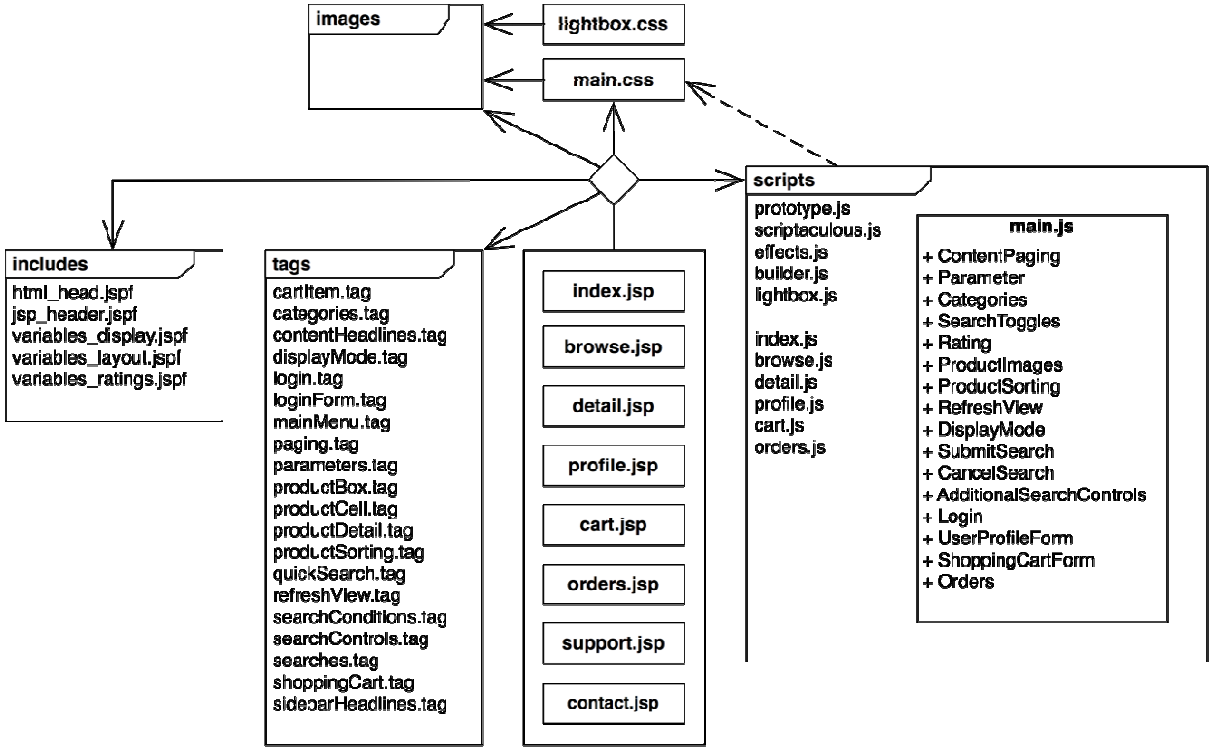


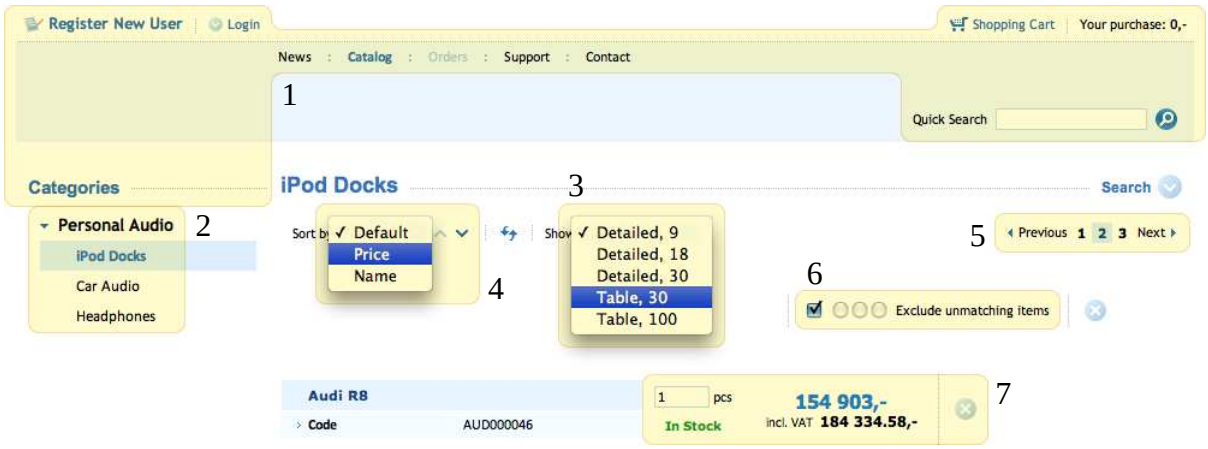
Diagram 8
Struktura prezentační vrstvy

3.7 Uživatelské rozhraní

Navrhnuté a implementované uživatelské rozhraní se snaží o zachování většiny prvků ovládání, na které jsou uživatelé zvyklí z prostředí již existujících webových obchodů [4]. Představuje ovšem také pár nových myšlenek z oblasti prezentace a zadávání uživatelských údajů. Cílem této kapitoly je popsat obě z těchto skupin vlastností rozhraní a v případě nových řešení také přiblížit jejich očekávaný přínos pro uživatele.

3.7.1 Zavedené prvky rozhraní

Uživatelé dnešních webových služeb již většinou mají konkrétní představu ohledně požadavků na funkčnost a způsobů ovládání jim známých typů webových aplikací. Většinu z těchto požadavků se snaží implementovat i navrhnuté prostředí, jedná se zejména o:

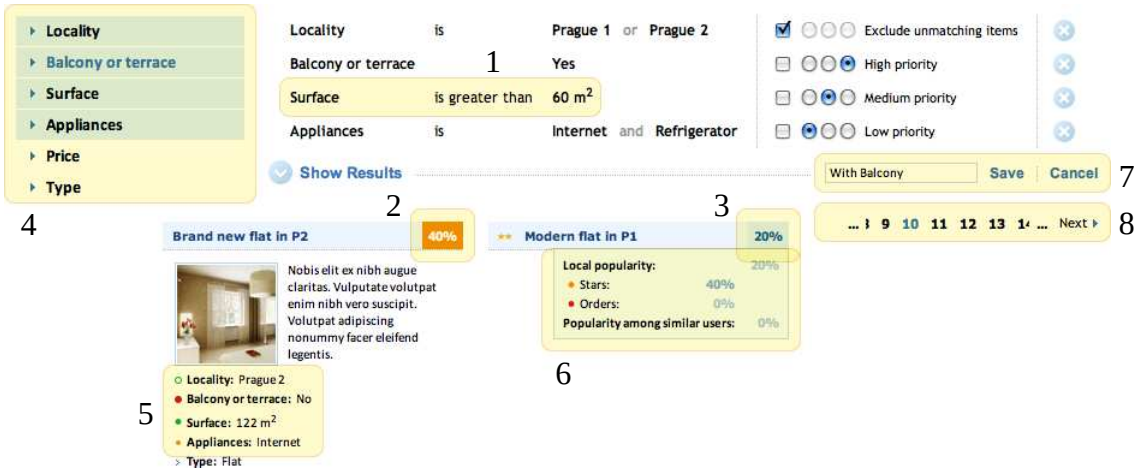


Obrázek 4
Prvky uživatelského rozhraní

- Standardní celkové rozvržení stránek (layout), logické rozmístění hlavních navigačních prvků do záhlaví, levého bočního pruhu stránky (hlavní menu, výpis kategorií, přihlášení, nákupní košík, rychlé hledání) (obr. 4/1)
- Členění kategorií do skupin (obr. 4/2)
- Různé možnosti zobrazení nabízených produktů, lišící se co do množství obsažených informací i počtu najednou zobrazených produktů (přepínaní mezi detailním pohledem, tabulkou nebo detailem produktu) (obr. 4/3)
- Řazení seznamu produktů dle vybraného parametru zájmu, změna pořadí produktů (vzestupně, sestupně) (obr. 4/4)
- Stránkování seznamu produktů, změna počtu zobrazených produktů na stránku (obr. 4/3, 5)
- Možnost ořezání výsledkové sady produktů vzhledem k zadanému parametru zájmu (viz kapitola 3.3.1) (obr. 4/6)
- Možnost editace obsahu košíku před odesláním objednávky (obr. 4/7)
- Použití ověřených piktogramů pro ikony (nákupní košík, hvězdičky vyjadřující explicitní preference uživatele)

3.7.2 Nově navržené prvky rozhraní

Vzhledem k tomu, že aplikace implementuje některé z modelů uživatelských preferencí, které se v prostředí webových obchodů běžně nevyskytují, byla potřeba k těmto modelům navrhnout taky adekvátní prvky uživatelského rozhraní. Ty jsou popsané v následujícím seznamu. Některé z nově navržených součástí se pak snaží hlavně o rozšíření funkčnosti a zpříjemnění obsluhy aplikace ve srovnání s běžnými implementacemi těchto ovládacích prvků.



Obrázek 5
Prvky uživatelského rozhraní

- Specifikace přímých preferencí uživatele pomocí stanovení jednotlivých podmínek na atributy produktů probíhá kompletně na straně klienta, není tudíž nutné čekat na odpověď serveru pro každou ze specifikovaných podmínek.
- Hodnoty stanovených podmínek se převádějí do slovního vyjádření v běžném jazyce, co může některým uživatelům ulehčit vnímání vlivu jednotlivých podmínek na výslednou sadu doporučených produktů (obr. 5/1).
- Rozhraní poskytuje informaci o vypočteném skóre pro aktuální model uživatelských preferencí, co umožňuje uživateli znát a zužitkovat tuto hodnotu skóre i v případě, že využívá jiný typ řazení seznamu produktů (obr. 5/2, 3).
- Pro výpočet nad přímo poskytnutými preferencemi se navíc používá kromě procentuálního vyjádření výsledného skóre také intuitivnější metoda reprezentace pomocí barev semaforu (obr. 5/2), která se řídí následujícími pravidly:
    - Hodnota, která patří intervalu $(^2/_3, 1]$ je reprezentována zelenou barvou
    - Hodnota, která patří intervalu $[^1/_3, ^2/_3]$ je reprezentována oranžovou barvou
    - Hodnota, která patří intervalu $[0, ^1/_3)$ je reprezentována barvou červenou
- Parametry produktů kategorie, nad kterými uživatel specifikuje podmínky a priority vyjadřující jeho aktuální preference jsou v rámci výpočtu přeskupeny podle existence těchto podmínek a hodnot jim asociovaných priorit. Na základě výsledného pořadí

parametrů je pak možné snadno rozpoznat aktuální priority preferencí uživatele (obr. 5/4).

- Funkcionalita přeskupení hodnot se ovšem uplatňuje také u zobrazení atributů jednotlivých produktů, čeho je možno využít v detailním pohledu na výsledkovou sadu „přímého“ modelu preferencí, kde se vždycky zobrazují nejprve ty atributy, které uživatele nejvíce zajímají. Před názvy atributů se pak zobrazují barevná kolečka, reprezentující hodnoty dílčích složek celkového skóre pro jednotlivé atributy. Jejich barva se řídí pravidly pro využití barev semaforu popsanými výše, jejich velikost pak reprezentuje úroveň zvolené priority. Tato funkcionalita zvyšuje orientaci uživatele ve výsledném seznamu produktů a umožňuje uživateli jednodušeji porozumět výslednému skóre objektu (obr. 5/5).
- V případě, že se zobrazují hodnoty skóre založené na explicitních a implicitních preferencích, má uživatel možnost podívat se na dílčí hodnoty tohoto skóre, které mu můžou poskytnout vysvětlení pro výslední ohodnocení vybraného produktu. V tomto prvku rozhraní se taky jako doplňující informace využívají barvy semaforu (obr. 5/6).
- Uživateli je nabídnuta možnost uložení přímého dotazu nad kategorií objektů pro jeho pozdější využití. Tato funkcionalita umožňuje zaznamenání aktuálních uživatelských preferencí a může být pro uživatele užitečná, jelikož eliminuje nutnost opětovné specifikace oblíbených preferenčních podmínek (obr.5/7).
- Ovládací prvek stránkování produktů obsahuje čísla všech stránek aktuálního seznamu, čímž umožňuje přechod na libovolnou stránku bez nutnosti dodatečného mezikroku přepnutí na jinou, neželanou stránku seznamu (obr. 5/8).

3.8 Problémy vzniklé během implementace a metody jejich řešení

Jedním z problémů, odhaleným během testů provázejících proces vývoje aplikace, se ukázala být nedostatečná přesnost základních datových typů jazyka Java pro číselné hodnoty s plovoucí desetinnou čárkou – zejména typu Double. Tento datový typ umožňuje nést relativně přesné hodnoty jak velkých, tak i velmi malých čísel pomocí nastavení zodpovídající hodnoty exponentu, nedokáže však s požadovanou přesností zachytit hodnoty jejich součtu. Tento požadavek vychází z metody výpočtu preferencí podle uživatelem specifikovaných podmínek na hodnoty jednotlivých atributů, zejména pak z metody výpočtu ohodnocení číselných typů atributů popsané v kapitole 3.3.1. Funkce výpočtu dílčího skóre se stoupající vzdáleností posuzované hodnoty od požadovaného intervalu stanoveného počáteční podmínkou totiž relativně rychle konverguje k nule, což může pro některé velmi malé hodnoty vypočteného skóre způsobit, že jsou v okamžiku jejich zakomponování do výpočtu výsledného skóre „překryty“ nepřesnostmi reprezentace řádově vyšších hodnot dílčích skóre jiných atributů hodnoceného objektu.

Tento problém byl vyřešen zavedením datového typu BigDecimal pro reprezentaci hodnoty celkového skóre objektu, který umožňuje variabilně specifikovat míru přesnosti reprezentované hodnoty požadované implementovaným modelem výpočtu preferencí. Daní za získanou přesnost je ovšem vyšší časová náročnost výpočtu, i když v aktuální implementaci nad poskytnutými daty bude tato spíše velmi těžko postřehnutelná.

Je třeba připomenout, že datový typ Double, reprezentující hodnoty dílčích skóre atributů objektů zůstal zachován, co může způsobit, že velmi malé hodnoty vypočteného skóre nebudou tímto datovým typem reprezentovatelné. Takovéto hodnoty atributů jsou už ale hodně mimo aktuální zájem uživatele, proto tato vlastnost nemá v zásadě žádný negativní vliv na přesnost výstupu zmíněného modelu preferencí.

4 Praktické experimenty

Hlavní úlohou praktických experimentů bylo obeznámit uživatele s vytvořenou aplikací, představit jim možné výhody implementovaných modelů preferencí a dát jim možnost si celkovou funkčnost aplikace a dopad implementovaných modelů vyzkoušet v praxi.

Předané uživatelské poznatky z používání aplikace byly pak zohledněné v procesu vyhodnocení celkové použitelnosti aplikace a stanovení jejích silnějších a slabších stránek. Na základě získaných informací byly pak definovány některé z možných variant dalšího rozvoje aplikace, které jsou popsány v kapitole 5.

4.1 Uživatelská odezva

Data uživatelské odezvy byly získány především přímým způsobem prostřednictvím dotazníku, který uživatelé v rámci konečné fáze experimentů dobrovolně vyplnili. Dotazník se zaměřil především na následující zkoumané oblasti:

- Ochota poskytování osobních údajů v prostředí internetových obchodů
- Ochota uživatelů specifikovat oblíbenost pro jednotlivé skupiny nabízených produktů
- Míra ovlivnění konečného výběru zboží zvoleným pořadím zobrazovaných produktů (tj. začleněním vybraných modelů preferencí do způsobu prezentace nabídky obchodu)
- Míra zjednodušení procesu výběru zboží zvoleným pořadím zobrazovaných produktů
- Správnost výpočtů implementovaných modelů preferencí
- Užitečnost možnosti vyhledávání produktů na základě přímé specifikace vyhledávacích podmínek
- Určení priorit pro lokální a kolaborativní složku výsledného skóre objektů
- Přehlednost uživatelského prostředí aplikace
- Stabilita aplikace

V rámci vyhodnocení výsledků experimentů byly ovšem kromě údajů z vyplněných uživatelských dotazníků využity také informace z log souborů aplikace, zaznamenávající veškeré uživatelské akce. Funkcionalita, která byla naimplementována za tímto účelem, umožnila jednak získání informací, které v dotazníku nebyly přímo obsaženy, kromě toho pak posloužila jako kontrolní prvek, umožňující ověřit věrohodnost informací obsažených v odpovídajících dotaznících.

Pozn.: Kompletní šablona původního dotazníku, jako i všechny vyplněné varianty dotazníku s uvedenými poznámkami uživatelů a zaznamenané log soubory jsou k dispozici na CD přiloženém k práci.

4.2 Vyhodnocení experimentů

Osobní informace	**Hodnocení uživatelů**					
Ochota poskytnout osobní informace						
Pohlaví	4	-	-	5	2	2
Datum narození	5	-	-	5	2	3
Příjem	5	-	-	5	5	5
Nejvyšší dosažené vzdělání	5	-	-	5	4	3
Bydliště	3	-	-	1	1	1
Rodinní stav	5	-	-	5	4	4
Zájmy	4	-	-	5	5	3
Standardní prohlížení nabídky						
Správnost výsledního pořadí produktů						
Nabídl systém správné produkty?	2	-	1	3	1	2
Míra ovlivnění výsledným pořadím a hodnocením produktů						
Ovlivní pořadí a ohodnocení zobrazených produktů váš výběr?	3	3	2	2	4	2
Zjednoduší pořadí a ohodnocení zobrazených produktů váš výběr?	2	1	1	2	2	2
Ochota specifikovat hodnocení oblíbených produktů pomocí hvězdiček						
Ochota specifikovat nejoblíbenější produkty, u kterých existuje šance případného zakoupení	1	4	2	5	1	2
Ochota specifikovat méně oblíbené produkty, které nejspíš zakoupeny nebudou	4	5	5	5	4	5
Důležitost složek hodnocení (v %, součet obou by měl být 100%)						
Produkty, které jsem sám zakoupil nebo ohodnotil	20	80	-	50	60	40
Produkty, které ohodnotili mě podobní uživatelé	80	20	-	50	40	60
Vyhledávání v nabídce						
Správnost výsledního pořadí produktů						
Nabídl systém správné produkty?	1	-	1	2	1	1
Míra ovlivnění výsledním pořadím a hodnocením produktů						
Ovlivní pořadí a ohodnocení zobrazených produktů váš výběr?	1	-	2	1	2	1
Zjednoduší pořadí a ohodnocení zobrazených produktů váš výběr?	1	1	1	1	1	1
Užitečnost funkcionality vyhledávání						
Užitečnost možnosti vyhledávání	1	-	1	1	1	1
Užitečnost možnosti ořezání výsledkové sady produktů	1	-	1	1	1	1
Užitečnost možnosti uložení vyhledávání pro pozdější využití	2	-	3	2	1	2
Uživatelské rozhraní						
Přehlednost / složitost / intuitivnost ovládání	2	1	1	3	2	2
Stabilita aplikace	1	1	1	1	1	1

Tabulka 1

Vyhodnocení uživatelských dotazníků. Hodnocení 1- 5, 1 – nejlepší, 5 - nejhorší

- Ochota poskytování osobních údajů v prostředí internetových obchodů

  V oblasti poskytování osobních informací se u většiny uživatelů z posuzované množiny potvrdil předpoklad popsaný v kapitole 2.5.1. Uživatelé neradi poskytují informace o sobě, pokud tyto nejsou nezbytně nutné pro vybraný účel, jako je např. znalost adresy pro možnost dodání zakoupeného zboží. Možnou variantou, která může

změnit rozhodnutí uživatelů je obsažení informace, že jimi poskytnuté data budou zohledněny v rámci posuzování pro ně vhodných objektů nabídky.

- Ochota uživatelů specifikovat míru oblíbenosti nabízených produktů

  V tomto případě vyšla najevo přirozená „lenost uživatelů“ – pokud nabízená funkcionalita nenabízí okamžitě prokazatelný přínos, nemá uživatel motivaci této funkčnosti využívat. Uživatelé vyjádřili pouze průměrnou ochotu ohodnocování nejoblíbenějších produktů. U produktů, které uživatele zajímaly méně, se pak tendence možného hodnocení snížila na minimum. S ohledem na tuto zjištěnou skutečnost se v rámci možných variant rozšíření aplikace nabízí myšlenka zavedení pouze dvoustupňové škály hodnocení produktů, která může na mnohé uživatele působit jako méně náročná varianta způsobu možného zadávání preferencí.

- Míra ovlivnění konečného výběru zboží implementovanými modely preferencí

  U této otázky, jak ukázaly i sesbírané informace veškerých uživatelských akcí, se přístup jednotlivých uživatelů liší. Někteří se spoléhají výlučně na hledání zboží pomocí přesné specifikace požadovaných parametrů, jiní naopak nezkoumají žádné z pokročilých možností aplikací a spoléhají se výlučně na funkcionalitu, která jim je standardním způsobem předkládána.

  U první skupiny uživatelů se dá předpokládat, že pečlivost, s jakou přistupují k procesu nákupu požadovaného zboží, se projeví i ve finálním výběru, který bude ovlivněn spíš výsledky vyhledávání, než výchozím pořadím objektů vypočteným na základě obecnějších metod výpočtu preferencí. Pro druhou skupinu uživatelů může mít výchozí zobrazení nabídky naopak vyšší prioritu, neboť tato skupina zkrátka nemá dostatek informací, času nebo vůle trávit procesem výběru zboží v obchodě delší čas a ráda si proto nechá od systému poradit.

  Implementovaná aplikace umožňuje využití obou zmíněných přístupů, čímž poskytuje oběma skupinám uživatelů možnost využití vybraného modelu preferencí.

- Míra zjednodušení procesu výběru zboží implementovanými modely preferencí

  V této otázce se všichni dotazovaní uživatelé shodli, že použité modely preferencí jednoznačně ulehčují proces výběru správného produktu a jejich začlenění do aplikace je proto vysoce opodstatněné.

- Správnost výpočtů implementovaných modelů preferencí

  Uživatelé neshledali žádné chyby ve výpočtu použitých preferenčních modelů. Tady je třeba připomenout, že pro hlubší analýzu správnosti modelů, zejména těch pracujících nad daty více uživatelů, je potřeba většího množství sesbíraných dat, které v čase realizace praktických experimentů nebyli k dispozici.

- Užitečnost možnosti vyhledávání produktů na základě přímé specifikace vyhledávacích podmínek

Tato funkcionalita byla všemi uživateli označena za velmi žádoucí, čemuž jistě odpovídá i zahrnutí této možnosti do většiny nově vznikajících systémů internetových obchodů. Jako méně důležitá, stále však ne zbytečná, pak byla označena možnost uložení vyhledávacího výrazu pro budoucí použití uživatelem.

- Určení priorit pro lokální a kolaborativní složku výsledného skóre objektů

  V této oblasti byly informace poskytnuté uživateli velmi různorodé, někteří preferovali spíše vlastní hodnocení produktů, jiných pak zajímalo hlavně hodnocení ostatních uživatelů. Aktuální implementace používá rozdělení 70% pro lokální preference a 30% pro kolaborativní filtrování, bližší popis je pak možné najít v kapitole 3.3.2.

- Přehlednost a intuitivnost uživatelského prostředí aplikace

  V této oblasti byli někteří uživatelé zcela bez připomínek, jiní pak uvedli množství návrhů, možných změn a vylepšení. Některé z těchto podnětů byly zapracovány do kapitoly 5.4.

- Stabilita aplikace

  Uživatelé nezaznamenali žádný problém.

5 Možné varianty vylepšení a rozšíření aplikace

5.1 Rozsah aplikace

Jako jedna z prvních možností rozšíření aplikace se nabízí implementace všech funkčností, které jsou potřeba, aby mohl být systém v praxi použitelný. Aktuální prostředí nabízí pouze podmnožinu běžných případů užití, chybí např. možnost platby kartou, možnost volby jiné fakturační a doručovací adresy, zrušení již odeslané objednávky, kontrola stavu odeslané objednávky, atd.

Nutným předpokladem pro možnost reálného využití aplikace je taktéž existence administračního prostředí, umožňujícího spravovat veškerý obsah aplikace, ať už se jedná o nabízené produkty, registrované uživatele s jejich preferenčními daty, nebo vytvořené objednávky. Důraz by měl být kladen především na vysokou produktivitu takovéhoto prostředí.

5.2 Technologie

Z hlediska použitých technologií se můžeme zaměřit na následující oblasti zájmu:

- Aplikační a integrační vrstva aplikace

  V současné podobě aplikace využívá standardní webové technologie jazyka Java v jejich základní podobě. Během let vznikly ovšem různé nadstavby pro tento jazyk, které standardizují architekturu rozsáhlejších webových aplikací anebo zjednodušují propojení s jejich databázovou vrstvou. Jedná se o frameworky jako např. Spring, Struts, Hibernate nebo JavaServer Faces. Využití některých z těchto frameworků může zjednodušit implementaci v rámci dalšího rozvoje aplikace.

- Prezentační vrstva

  U webových aplikací bývá často řešeným problémem kompatibilita zobrazení a fungování stránek v různých internetových prohlížečích. V tomto směru se aplikace snaží o relativně širokou podporu, možnosti rozšíření ovšem existují – jedná se zejména o implementaci podpory pro starší verze prohlížečů (Internet Explorer 6, 7), nebo o vytvoření rozvržení optimalizovaného pro mobilní verze prohlížečů.

  V nejmodernějších webových aplikacích se rovněž využívá technologie AJAX, která umožňuje načítání pouze vybraných částí stránek. Toto chování jednak zvyšuje výkon aplikace, protože server generuje pouze potřebné data, která navíc méně zatěžují síťové připojení, na druhé straně se pak chování aplikace jeví jako svižnější, protože nedochází k překreslování celého obsahu stránky a uživatel čeká kratší dobu na odpověď ze strany serveru.

  Dalším posuzovaným aspektem prezentační vrstvy je bezpečnost. V této oblasti je vhodné u některých akcí aplikace změnit způsob předávání parametrů z metody GET

na POST a zabezpečit aplikaci vůči speciálně zkonstruovaným URL požadavkům, ohrožujícím stabilitu aplikace. V případě potřeby vysoké míry bezpečnosti je pak možné využít zabezpečeného HTTPS protokolu, toto ale v oblasti webových obchodů nebývá příliš obvyklé.

Možnosti vylepšení se vztahují také na použitý Javascript framework Script.aculo.us, kde může jeden z použitých efektů „rozbalení“ bloku vyvolat chybu v případě rychlého opětovného kliknutí na prvek spouštějící tento efekt.

5.3 Použité modely uživatelských preferencí

Jelikož implementované modely jsou v předchozím textu rozděleny podle metody sběru vstupních preferencí, budou možnosti rozšíření těchto modelů popsány postupně vzhledem k tomuto rozdělení.

- Vybraný model uživatelských preferencí nad daty získanými přímým způsobem

  Jak bylo pro tento model výpočtu zmíněno v kapitole 3.8, velmi malé hodnoty dílčího skóre atributů nemusí být reprezentovatelné pomocí datového typu Double. Jedno z řešení tohoto problému spočívá v úpravě výpočetního vztahu pro stanovení skóre číselných atributů popsaného v kapitole 3.3.1 takovým způsobem, aby bylo zajištěno, že i nejmenší možné skóre pro vybraný číselný atribut bude s dostatečnou přesností reprezentovatelné datovým typem Double. Aby bylo možné garantovat tuto vlastnost, musí systém znát minimum a maximum všech posuzovaných hodnot, tj. všech hodnot vybraného číselného atributu pro všechny hodnocené produkty v kategorii.

  Další z možných rozšíření aplikace spočívá v obsažení schopnosti vyhodnocování „vzdálenosti“ mezi jednotlivými hodnotami typu textového řetězce, tak jak je to popsáno v kapitole 2.5.3. Systém by tak věděl určit vztahy mezi jednotlivými hodnotami a objekty obsahující např. atribut „Manuální klimatizace“ by mohli být ohodnoceny pro podmínku „Automatická klimatizace“ vyšším skóre, než objekty, které touto hodnotou pro zvolený atribut nedisponují.

- Vybraný model uživatelských preferencí nad daty získanými explicitně a implicitně

  U tohoto modelu výpočtu preferencí se možnosti rozšíření vážou zejména na přesnější schopnost záznamu implicitních preferencí uživatele, zahrnující monitorování většího množství uživatelských akcí. Některé příklady těchto akcí je možné najít v kapitole 2.5.2.

  Monitorováno může být ovšem i vyhledávání produktů za pomoci prvního z uvedených modelů, pracujícího nad přímo vyjádřenými preferencemi uživatele. Možnost ukládání oblíbených vyhledávacích výrazů pro zvolenou kategorii k tomu navíc přímo vybízí. Bylo by tak možné propojit oba z uvedených modelů, a výslední hodnocení produktů v módu běžného prohlížení by pak kromě aktuálně používaných hodnot záviselo i od hodnot předchozích hledání uživatele. Toto propojení

implementovaných modelů by pak jistě šlo rozšířit i pro opačný směr, kde by výsledky hledání zohledňovali dlouhodobé preference aktuálního uživatele. Je ale potřeba zvážit, do jaké míry by toto chování bylo užitečné a jestli by uživatelům nezpůsobovalo spíše potíže ohledně pochopení výsledního pořadí vyhledávaných produktů.

Další z možných rozšíření implementovaných modelů preferencí zahrnuje analýzu explicitně preferovaných objektů vzhledem k hodnotám jejich atributů a následné stanovení predikce oblíbenosti pro nehodnocené objekty, jakož i zohlednění délky existence jednotlivých produktů v obchodě. Obě tyto problematiky jsou popsány v kapitolách 2.5.3 a 2.6.

Nutnou součástí procesu optimalizace výpočtu zvolených modelů preferencí je pak dlouhodobé testování a ladění použitých modelů na základě vyhodnocování sesbíraných preferenčních dat a poskytnuté uživatelské odezvy.

5.4 Uživatelské rozhraní

V oblasti uživatelského rozhraní aplikace jsou možné varianty vylepšení rozděleny do následujících kategorií:

- Intuitivnost rozhraní

  V oblasti intuitivnosti rozhraní bylo na základě uživatelské odezvy stanoveno pár možných problémů, na které může být snaha o vylepšení rozhraní zaměřena:

  - Pozice, popisky a grafické ztvárnění tlačítek pro zapnutí a spuštění vyhledávání

    Tlačítka pro někoho nejsou dostatečně výrazná, nutnost potvrzení vyhledávacích podmínek nemusí být na první pohled zřejmá.

  - Nemožnost vybrání zaškrtávacího pole (checkboxu) během zadávání číselné podmínky, pokud tato není předem specifikována

    Tato vlastnost může být v rozporu s běžnými návyky uživatelů.

  - Ztvárnění prvku pro specifikaci priority zvolené podmínky

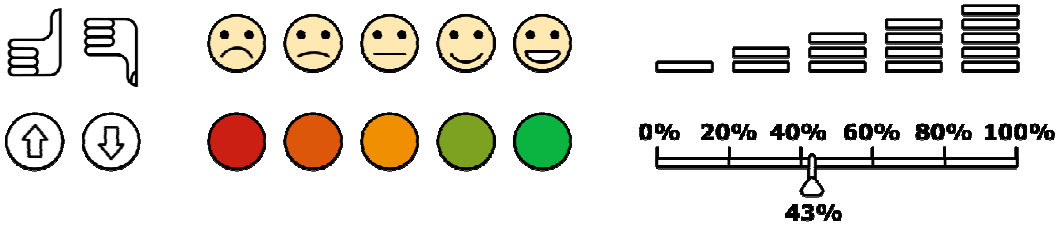
    Funkcionalita prvku nemusí být na první pohled zřejmá.

- Kompatibilita rozhraní

  Kromě problematiky zobrazování a fungování aplikace v různých prohlížečích se možnosti optimalizace týká také např. schopnost ovládání aplikace jenom za pomocí klávesnice.

- Prvky rozhraní pro explicitní hodnocení objektů

  V aktuální implementaci se používají hvězdičky, je ovšem možné použití i jiných prvků rozhraní. Některé z možných příkladů jsou zobrazeny na následujícím obrázku:



Obrázek 6
Varianty prvků rozhraní pro zadávání hodnocení

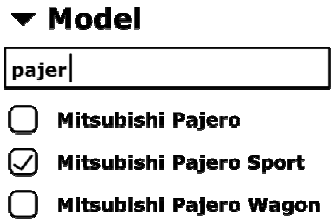
- Prvky rozhraní pro specifikaci vstupných podmínek vyhledávání

  Na základě testů práce s aplikací byly stanoveny následující možnosti vylepšení vstupních prvků:

  - Implementace závislostí mezi vybranými hodnotami atributů
    Tato funkcionalita umožní např. filtrování zobrazení textových hodnot vybraných atributů na základě zvolených hodnot jiných atributů. Příkladem může být skrytí názvů modelů aut, které neodpovídají vybrané značce.

  - Možnost uživatelského filtrování textových hodnot atributů, pokud je jejich počet vyšší.

    Možná varianta ovládacího prvku je zobrazena na následujícím obrázku:



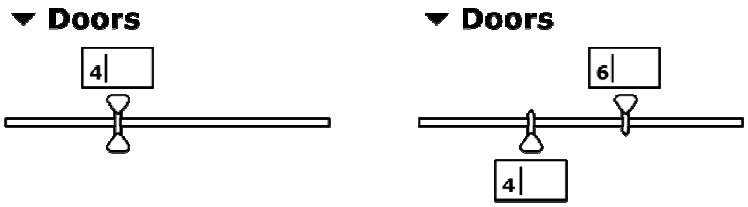
Obrázek 7
Návrh rozhraní pro filtrování textových hodnot

    Množina zobrazených hodnot je omezena podmínkou obsažení řetězce specifikovaného ve vstupním poli.

  - Vizuální specifikace požadovaného číselného intervalu

    Častá specifikace požadovaného číselného intervalu pomocí vpisování jeho hraničních hodnot do vstupních polí se může stát pro uživatele časem frustrující.

Navržený ovládací prvek umožňuje jednodušší specifikaci číselného intervalu:



Obrázek 8
Návrh rozhraní pro specifikaci číselné podmínky

Tažením za prostřední část ukazatele se mění konkrétní požadovaná hodnota, tažením za spodní, resp. horní část ukazatele se mění spodní, resp. horní hranice požadovaného intervalu. Umístění horní nebo spodní části ukazatele na zodpovídající okraj lišty způsobí nastavení této hranice požadovaného intervalu na maximum/minimum všech posuzovaných hodnot, resp. na hodnoty $-\infty/\infty$.

6 Závěr

Prvotním cílem práce bylo obeznámení se s problematikou uživatelských preferencí v oblasti webových služeb, s hlavním zaměřením na systémy internetových obchodů. Byly vybrané některé z popsaných modelů preferencí a prezentovány navržené způsoby jejich výpočtu. Tyto modely byly pak implementované do prostředí, navrženého a vytvořeného primárně pro tento účel. Finální aplikace byla rozšířena o množinu testovacích dat a její funkčnost byla otestována na vybrané skupině uživatelů. Na základě jejich odezvy byl posouzen přínos aplikace a byly stanoveny některé z možností dalšího rozvoje.

Výsledky experimentů uskutečněných nad finální verzí implementované aplikace prokázaly, že navržená aplikace vyhověla většině stanovených požadavků i vzniklých očekávání. Uživatelé uvítali přítomnost a způsob fungování použitých modelů uživatelských preferencí a poukázali tím na užitečnost a opodstatněnost dalšího rozvoje v této oblasti. Jako každý systém, i navržená aplikace musí však projít fázemi dlouhodobějšího testování a reálného nasazení do praxe. Důsledkem bývá vznik nových požadavků na funkčnost, stabilitu, případně výkon aplikace, které mohou být v počátečních fázích nasazení hůře odhalitelné. Ve zkoumané oblasti toto platí dvojnásob, neboť dlouhodobější provoz přispívá také k možnosti analýzy a ladění výpočtů implementovaných modelů preferencí. Některé z modelů navíc pro správné a užitečné fungování vyžadují velkou množinu sesbíraných uživatelských dat, čeho je možno dosáhnout pouze delší dobou provozu aplikace v internetovém prostředí.

Tato práce se zaměřila zejména na návrh, implementaci a otestování aplikace v rámci počátečních fází životního cyklu internetových informačních systémů. Nepokrývá tudíž problematiku dlouhodobého testování, ladění a rozšiřování aplikace vzhledem k nově-vzniklým požadavkům její uživatelů. Některé z možných směrů dalšího rozvoje aplikace byly však v práci nastíněné, a jelikož je implementace postavena výhradně na otevřených technologiích nezávislých od použité platformy, nemělo by dalšímu rozvoji aplikace v případě zájmu potenciálních tvůrců nic bránit.

7 Obsah CD

K této práci je přiloženo CD, které obsahuje zdrojové kódy implementované aplikace, popis databáze obsahující testovací data a text této práce v elektronické podobě. Součástí jsou také zdrojové soubory řešení, použitých během procesu návrhu a implementace vytvořené aplikace.

Obsah CD se řídí následující strukturou:

- *1 eshop* – zdrojový kód aplikace a použité knižnice
  Údaje pro připojení k databázi se specifikují ve třídě *main.java*, knižnice použité ve zdrojovém kódu jsou umístěny v adresáři *web/WEB-INF/lib*. Pro možnost rychlého přihlášení byl v systému vytvořen účet s přihlašovacími údaji *guest*/*guest*.
- *2 db* – obsah databáze
  Dump databáze ve formátu SQL.
- *3 experiments* – soubory sesbírané uživatelské odezvy v rámci uskutečněných praktických experimentů
  - *logs* – log soubory zaznamenané činnosti uživatelů
  - *surveys* – vyplněné dotazníky uživatelů
- *4 text* – text této práce ve formátu PDF
- *5 resources* – soubory použité během procesu tvorby aplikace a textu práce
  - *db* – návrhy databázového modelu, zdrojové soubory pro dodaná testovací data ve formátech Microsoft Excel 2007 (s makry) a Adobe Photoshop
  - *gui* – návrhy uživatelského rozhraní ve formátu Macromedia Fireworks
  - *html* – realizace grafického návrhu do jazyka XHTML a implementace dynamického chování rozhraní v jazyku Javascript
  - *text* – zdrojové soubory všech obrázků, grafů, diagramů a tabulek použitých v textu práce ve formátech Macromedia Fireworks, OmniGraffle a jiných
- *6 install* – instalační soubory aplikací pro platformu Win32

8 Seznam použité literatury

[1] S.-J. Ko, J.-H. Lee: *User Preference Mining through Collaborative Filtering and Content Based Filtering in Recommender System*
In Proceedings of the 3rd International Conference on E-Commerce and Web Technologies, 244-253, 2002

[2] B. Mobasher, R. Cooley and J. Srivastava: *Automatic Personalization Based on Web Usage Mining*
In Communications of the ACM, 142-151, 2000

[3] G. Rossi, D. Schwabe and R. Guimaraes: *Designing Personalized Web Applications*
In Proceedings of the 10th World Wide Web Conference, 275-284, 2001

[4] Andrew Sears, Julie A. Jacko: *Handbook for Human Computer Interaction, 2nd Edition*, CRC Press, 2007

[5] Java Reference documentation
http://java.sun.com/reference/docs/

[6] JavaServer Pages Technology
https://java.sun.com/products/jsp/

[7] Java SE Technologies
http://java.sun.com/javase/technologies/database/

[8] Apache Tomcat 6.x
http://tomcat.apache.org/tomcat-6.0-doc/index.html

[9] Prototype Javascript framework
http://api.prototypejs.org/

[10] Script.aculo.us
http://wiki.github.com/madrobby/scriptaculous

[11] UML Specifications
http://www.jeckle.de/uml_spec.htm