VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2017 Nikolas Kantor

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ANALÝZA A PREZENTACE CENOVÝCH DAT VOZIDEL PRODÁVANÝCH PO INTERNETU

ANALYSIS AND PRESENTATION OF PRICE DATA OF VEHICLES SOLD OVER INTERNET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Nikolas Kantor

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Pavel Očenášek, Ph.D.

BRNO 2017

Zadání bakalářské práce/19767/2016/xkanto10

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Kantor Nikolas**
Obor: Informační technologie
Téma: **Analýza a prezentace cenových dat prodávaných vozidel prodávaných po Internetu**
Analysis and Presentation of Price Data of Vehicles Sold over Internet
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s webovými technologiemi a analyzujte informace dostupné z externích inzertních webů nabízejících automobily.
2. Analyzujte požadavky na webovou aplikaci, která bude umožňovat vyhledávání vozů podle různých parametrů, jako je třeba lokalita, výrobce, typ a podobné. Dále bude umožňovat porovnání konkrétního vozu u více autobazarů, zjištění hodnoty vozu na základě jeho parametrů (stáří, výrobce, typ, ...), zobrazení vývoje či odhadu ceny vozu v čase.
3. Vytvořte konkrétní návrh webové aplikace a architekturu uložení dat. Pokuste se v co největší míře při tvrorbě aplikace reflektovat také právní předpisy týkající se oceňování vozidel.
4. Implementujte aplikaci podle návrhu a proveďte testování na vhodně zvoleném vzorku dat.
5. Proveďte zkušební provoz aplikace s reálnými daty. Zhodnoťte dosažené výsledky a potenciál aplikace do budoucna.

Literatura:

- Liu Bing, Totty Brian. Web data mining: exploring hyperlinks, contents, and usage data. Springer, 2007. ISBN 978-3-540-37881-5.
- Gourley David, Totty Brian. HTTP: the definitive guide. O'Reilly, 2002. ISBN 15-659-2509-2.
- Mitchell Ryan, Holmes James. Instant web scraping with Java, Packt Publishing, 2013. ISBN 978-184-9696-883.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá procesem analýzy požadavků, návrhu a implementací webové aplikace, která má za úkol primárně poskytnout uživateli možnost vyhledávání nabídek vozů napříč více inzerčními servery s automobily. Dále má poskytovat uživateli další služby, jako odhad hodnoty vozu a zobrazení vývoje hodnoty vozu. Tato práce se podrobně zabývá každým krokem vývoje této aplikace od stanovení požadavků až po výslednou implementaci. Pro klientskou část aplikace byly použity technologie jako HTML, CSS, Javascript a JSP. Serverová část aplikace je poté napsané v jazyce Java v kombinaci s frameworkem Spring.

Abstract

This bachelor's thesis deals with the process of analyzing requirements, design and implementation of web application, which is primarily intended to provide the user with the possibility to search for offers of cars across multiple advertising websites with cars. It is also intended to provide the user with additional services such as vehicle value estimation and vehicle value development. This work deals in detail with each step of the development of this application, from the definition of requirements to the final implementation. For the client part of the application, technologies such as HTML, CSS, Javascript and JSP have been used. The server part of the application is then written in Java in combination with the Spring framework.

Klíčová slova

Webová aplikace, Webové technologie, Java, Spring Framework, Apache Tomcat, MySQL, Javascript, jQuery, Google Charts, HTML, CSS, vyhledávání vozů, oceňování vozů, vývoj ceny vozu.

Keywords

Web application, Web technologies, Java, Spring Framework, Apache Tomcat, MySQL, Javascript, jQuery, Google Charts, HTML, CSS, cars search, car valuation, car price development.

Citace

KANTOR, Nikolas. *Analýza a prezentace cenových dat prodávaných vozidel prodávaných po Internetu*. Brno, 2017. 34 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Očenášek Pavel, Ph.D.

Analýza a prezentace cenových dat prodávaných vozidel prodávaných po Internetu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Nikolas Kantor
17. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Pavlu Očenáškovi, Ph.D. za ochotu a vstřícnost při poskytnutých konzultacích a za cenné a užitečné rady při vývoji aplikace.

Obsah

1 Úvod

Tato práce popisuje celý proces vývoje webové aplikace, která umožňuje uživateli skrze grafické uživatelské rozhraní přístup k datům získávaným z různých inzerčních webů zaměřujících se na automobily, konkrétně osobní automobily. Aplikace také uživateli nabízí možnost využití dalších služeb, které jsou z většiny založeny na práci s již zmíněnými daty inzerčních webů. Důležitým aspektem aplikace je také intuitivní uživatelské rozhraní, umožňující uživateli efektivní práci s aplikací.

Základní funkčností aplikace je, stejně jako na každém inzerčním webu, vyhledávání vozů na základě uživatelem definovaných parametrů a následně zobrazení nalezených vozů. Díky velkému množství dat, se kterými aplikace pracuje, nabízí i další uživatelsky užitečné služby. Další funkcí aplikace je vyhledávání konkrétních vozů napříč autobazary na základě uživatelem definovaných parametrů tak, aby následně uživatel viděl nabídky konkrétních vozů na různých inzerčních serverech a mohl si je navzájem porovnat. Třetí službou, která aplikace uživateli nabízí je zjištění hodnoty vozu na základě jeho parametrů. Výpočet se provádí dvěma způsoby a uživateli jsou tedy zobrazeny dvě výsledné hodnoty. První hodnota je vypočítána pomocí váženého průměru na základě známých vozů z databáze. Druhá hodnota se vypočítává podle znaleckého standardu č.1/2005, který definuje metodiku oceňování vozů, kterou v obdobné variantě používají například pojišťovny pro oceňování vozidel. Poslední, a tedy čtvrtá služba uživateli umožňuje zobrazit si vývoj ceny vozu, a to v závislosti na stáří vozu a v závislosti na počtu najetých kilometrů, a to v podobě dvou grafů. Navíc díky proložení spojnicí trendu může uživatel vidět předpokládaný vývoj ceny do budoucna.

V této práci se detailně seznámíme s celých procesem tvorby této aplikace. Jako první se seznámíme s technologiemi, které aplikace používá. Dalším bodem je následně definice požadavků na tuto aplikaci, tak jak byly zadány zadavatelem. Následně se budeme podrobně zaobírat analýzou těchto požadavků, zvážením praktických a nepraktických aspektů zadání a podobně. Navazující kapitolou je návrh aplikace, který zachycuje model databáze, objektový model a další diagramy, použité pro následnou implementaci. Implementaci je tedy následující kapitolou, v níž je vysvětleno, jak je naimplementován model celé aplikace a také podrobněji rozebrána implementace jejich jednotlivých částí. Poslední kapitolou je testování, které popisuje, jak probíhalo testování samotné aplikace během vývoje i po jeho dokončení.

2 Použité technologie

Výběr technologií pro aplikaci byl omezen zadávajícím pouze z jednoho hlediska, a to, aby se jednalo o webovou aplikaci. Vzhledem k tomu, že by aplikace měla být provozována na portálu VUT a také vzhledem také k povaze služeb nabízených aplikací je vhodné, aby tato aplikace byla uživatelům přístupná rychle a jednoduše a pro tyto specifikace je webová aplikace nejlepším řešením.

Z dostupných a používaných technologií v oblasti webových služeb se pro zobrazování klientské části aplikace využívá kombinace HTML [1] a kaskádových stylů, tedy CSS [2]. Dále na klientské části aplikace se používá skriptovací jazyk Javascript [3] a jeho knihovna jQuery [4].

Pro serverovou straně aplikace je využit jazyk Java [5], který patří v oblasti webových aplikaci mezi velmi rozšířené jazyky a vyniká například svou bezpečností. Ve spolupráci s Javou se používá framework Spring [6], především jeho modul Spring MVC[1], který je v současnosti asi nejpoužívanějším MVC frameworkem. Spring patří v oblasti Javy momentálně mezi jeden z nejrozšířenějších frameworků vůbec. O samotný běh aplikace se potom stará aplikace Apache Tomcat [7], webový server stavěný pro jazyk Java. Apache Tomcat poté využívá technologii JSP (JavaServer Pages) [8] pro přenesení dat ze serverové části aplikace do klientské části. Jedná se o technologii pro vytváření dynamických stránek založená na jazyce Java. Pro uložení dat se bude používat databáze MySQL [9].

2.1 HTML

HTML je zkratka pro HyperText Markup Language, neboli Hypertextový značkovací jazyk. HTML vzniklo v roce 1990 a v dnešní době se vyskytuje především ve své poslední, nejnovější verzi HTML 5, která byla vydána po patnáctileté pauze. Mnoho webových stránek a aplikací však v dnešní době využívá verzi HTML 4.01, která byla vytvořena v roce 1999 komunitou W3C[2]. Původním plánem bylo po vydání jazyka HTML 4.01 přejít na jazyk XHTML, který měl být založen na univerzálním XML. Jedná se o nejpoužívanější jazyk pro tvorbu stránek v systému World Wide Web. A jeho struktura je podobná XML.

2.2 CSS

CSS je zkratkou pro Cascading Style Sheets, což znamená kaskádové stylopisy. Jazyk CSS se využívá v kombinaci s HTML a XHTML pro účely formátovaní elementů těchto jazyků. Jedná se tedy o

[1] MVC je jedním z modulů Spring frameworku, více na https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

[2] Konsorcium pracující na webových strandardech, více na https://www.w3.org/Consortium/

definování způsobu zobrazení těchto elementů. CSS bylo stejně jako XHTML navrženo organizací W3C a dnes se používá v nejnovější verzi 3. Hlavní myšlenkou jazyku CSS je oddělení vzhledu dokumentu od jeho struktury, potažmo samotného obsahu.

V počátcích jazyka HTML byla snaha o možnosti úpravy zobrazování elementů přímo pomocí samotného HTML, avšak nedostatek standardů a konkurenční boje mezi výrobci prohlížečů vedly k vzniku odděleného CSS. Většina těchto HTML tagů je v dnešních moderních prohlížečích již nepodporována, a naopak mnoho nových atributů, které přibyly v CSS3 již prohlížeče podporují. Příklad takového zastaralého tagu může být například tag `<align>`, který umožňoval zarovnáváni elementů na strany. Naopak příkladem nových vlastní CSS 3 může být například poměrně populární zaoblení rohů elementu, jedná se konkrétně o vlastnost `border-radius`. Dříve se takového zaoblení rohů muselo provádět pomocí obrázků, což pro vývojáře bylo značně neefektivní.

2.3 JavaScript

Javascript vznikl v roce 1995, avšak jeho první standardizovaná verze pochází z roku 1997, kdy byl standardizován asociací ECMA[3]. Jedná se o multiplatformní, objektově orientovaný jazyk, který je z hlediska syntaxe velmi podobný jazykům C++ a Java, avšak sémanticky se jedná o naprosto jiný jazyk. Díky své přenositelnosti, pro programátory přívětivé syntaxi a lehkosti se jazyk v dnešní době používá především součástí WWW stránek, kdy je Javascript využíván většinou na straně klienta, tedy v internetovém prohlížeči.

V projektu se Javascript používá pro práci s HTML, s velkou měrou přispívá k dynamičnosti webu, kterou umocňuje použití knihovny jQuery. Dále se používá například k vytváření animací na webu a pro provádění asynchronních dotazů na serverovou část aplikace.

2.3.1 jQuery

Knihovna jQuery je nástavbou nad jazykem Javascript. Jedná se o nejrozšířenější knihovnu pro tento jazyk. Pochází z roku 2006, kdy ji na BarCampu[4] představil John Resig. Jedná se o objemově úspornou knihovnu, kterou nabízí uživatelsky velmi přívětivou syntaxi, avšak má obrovský potenciál. Hlavními klíčový vlastnostmi jQuery jsou manipulace s DOM[5], například dynamické změny CSS vlastností, zpracování a práce s událostmi, příkladem může být událost stisknutí tlačítka. Dále jQuery umožňuje provádění animací, kdy lze animovat například změna barvy elementu. Poslední z nejvýznamnějších vlastností je práce s Ajaxem [10], který umožňuje asynchronní vykonávání HTTP[6] požadavků.

[3] ECMA - https://www.ecma-international.org/
[4] BarCamp v ČR - http://www.barcamp.cz/
[5] DOM - https://www.w3schools.com/js/js_htmldom.asp
[6] Protokol na přenos hypertextových dokumentů – více https://www.w3.org/Protocols/rfc2616/rfc2616.html

Knihovna jQuery samozřejmě nabízí více funkcionality, o které si lze přečíst na jejich webových stránkách. Důkazem úspěchu jQuery může být seznam jeho oficiálních podporovatelů, mezi které patří například IBM, či WordPress.

2.4 Java

Java je objektově orientovaný programovací jazyk, který byl poprvé představen v květnu 1995 firmou Sun Microsystems. Od té doby až do května roku 2008 byl jazyk vyvíjen pouze touto společností. Od května 2008 byly zdrojové kódy Javy zveřejněny a nyní je vyvíjena jako open-source. Aktuálně je Java ve verzi 1.8, která byla vypadána v březnu 2014. Jedná se o obecně jeden z nejpoužívanějších programovacích jazyků na světě. V dubnu 2017 se Java podle TIOBE indexu[7], který zkoumá popularitu a programovacích jazyků drží na první místě s 15%, tedy dost velkým náskokem před jazykem C, který má pouze 7%. Jeho největší výhodou je přenositelnost, jelikož je tento jazyk nezávislý na platformě. Velkou oblibu si získal také díky jeho operačnímu systému Android, který je na Javě postaven.

Ve webových aplikacích se Java používá pro serverovou část aplikací. Její silné stránky spočívají především v robustnosti aplikace a bezpečnosti, nevýhodou mohou být větší nároky na zdroje. Používá se především pro velké projekty a také díky tomu je v dnešní době překonávána jazykem Python v oblasti webových aplikací, jelikož Python je naopak lehký jazyk s jednoduchou syntaxí a méně náročný na zdroje.

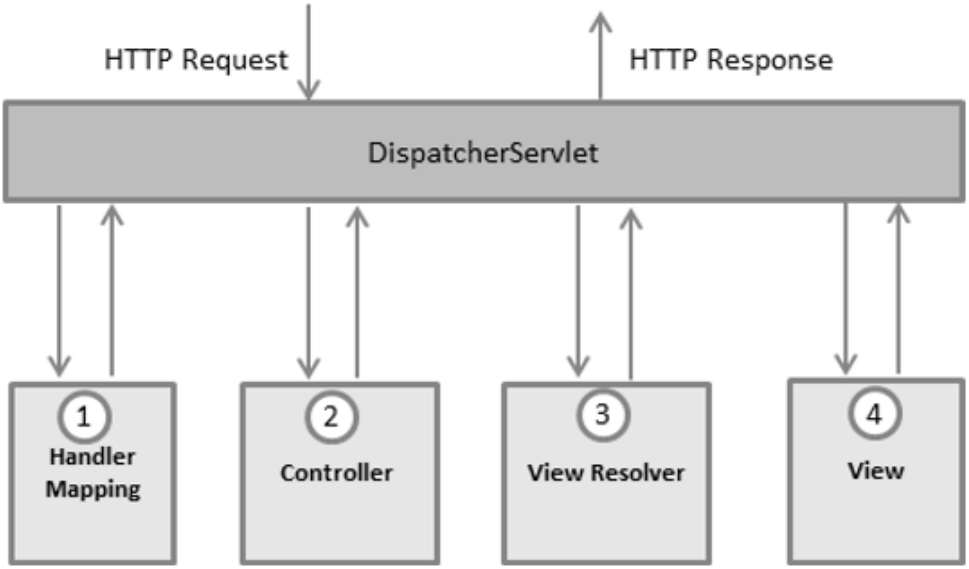
2.4.1 Spring

Spring je populární open-source[8] framework pro vývoj Java aplikací. Prvně spatřil světlo světa v říjnu 2002, však ne ještě pod jménem Spring, jeho autorem je Rod Johnson. Až později byl framework uvolněn pod jménem Spring jako open-source a jeho první verze vyšla v březnu 2004.

Důvodem pro vznik frameworku byla právě usnadnění vývoje enterprise aplikací[9]. Jednou z výhod springu je možnosti použití pouze některých jeho částí, bez nutností používat jiné části.

V tomto projektu je použit především Spring MVC pro oddělení datového modelu aplikace, uživatelského rozhraní a řídící logiky aplikace. Celý Spring MVC je postaven kolem tzv. DispatcherServletu, viz. [6], který řídí veškerý tok HTTP požadavků a deleguje požadavky na příslušné kontrolery na základě mapování (tzv. handler mappings), ty následně řídí aplikační logiku, obvykle získání či uložení dat, a poté vrátí DispatcherServletu jméno požadovaného zobrazení (anglicky View), který jej následně vyhodnotí a předá mu data a výsledné HTML vrátí uživateli do prohlížeče. Tato MVC architektura je dobře vysvětla na obrázku Obrázek 2.4.1.

[7] https://www.tiobe.com/tiobe-index/
[8] jedná se o software s otevřeným zdrojovým kódem
[9] podnikové aplikace



Obrázek 2.4.1: Spring MVC diagram [11]

2.4.2 Apache Tomcat

Apache Tomcat je Java Servlet[10] kontejner, který je vyvíjený jako open-source. Implementuje technologie jako Java Servlet, JavaServlet pages (JSP) a WebSockets[11] a poskytuje HTTP web server na kterém mohou běžet Java aplikace. Tomcat je vyvíjen softwarovou nadací Apache Software Foundation a uvolňován pod licencí Apache Licence 2.0[12]. Jedná se o open-source software.

2.4.3 JSP

JavaServer Pages, zkratkou JSP je technologie pro vývoj dynamických HTML stránek založená na jazyku Java. Byla vyvinuta a prvně vypuštěna v roce 1999 společností Sun Microsystems. Od roku 2010 se o vývoj stará společnost Oracle důsledkem akvizice společností. JSP se nejčastěji využívanou technologií svého druhu a je podporována většinou MVC frameworků. Používá pro propojení Javy a HTML, kdy se do HTML kódu pomocí speciálních tagů nazývaných skriptlety vkládá Java.

2.5 MySQL

MySQL je systém řízení báze dat uplatňující relační databázový model. Jedná se o multiplatformní databázi, kdy komunikace s ní probíhá pomocí jazyka SQL obohacených o specifický dialekt. Bylo vytvořeno společností MySQL AB, kterou později získal Sun Microsystems. Je dostupné jak pod

[10] Java Servlet - https://en.wikipedia.org/wiki/Java_servlet
[11] RFC 6455 - https://tools.ietf.org/html/rfc6455
[12] Apache licence 2.0 - https://www.apache.org/licenses/LICENSE-2.0

bezplatnou licencí, tak i pod komerční placenou licencí. MySQL je velmi oblíbené především pro svůj výkon a fakt, že se jedná o volně šiřitelný software.

3 Požadavky

Tato kapitola se zabývá specifikací požadavků na výslednou aplikaci. Jedná se společně s analýzou požadavků o nejdůležitější kroky při vývoji aplikace, jelikož špatná specifikace požadavků může celý vývoj aplikace odklonit špatným směrem.

Požadavky lze rozdělit do dvou hlavních kategorií, tedy požadavků na technickou stránku aplikace, což jsou především použité technologie a požadavků na funkcionalitu aplikace. Některé z požadavků byly zadány na přímo jako součást zadání a některé vyplývají z požadovaných vlastností aplikace. Na základě těchto požadavků je následně v následující kapitole 4 provedena jejich podrobná analýza.

3.1 Technické požadavky

Na technickou stránku aplikace, tedy na použité technologie nebyly ze strany zadání kladeny nijak výrazné požadavky. To, že se bude jednat o webovou aplikaci, vyplývá jak z plánového budoucího použití aplikace, tak i z povahy samotné aplikace. Následující požadavky na aplikaci hrály při výběru technologií důležitou roli. Byly požadovány následující vlastnosti aplikace:

- jednoduše dostupná aplikace
- jednoduše spustitelná aplikace
- intuitivní prostředí

Na základě prvních dvou požadavků vyplývá jednoznačně že webová aplikace je ideálním řešením. Aplikace, která je dostupná na veřejně přístupné adrese je nejlepším řešením pro dosažení co největší dostupnosti pro uživatele a zároveň jelikož v podstatě každý uživatel s přístupem na internet má internetový prohlížeč, tak se uživatel dostane do aplikace do několika vteřin, odpadá mu tím nutnost cokoliv instalovat, a může aplikaci v podstatě ihned začít používat.

3.2 Požadavky na funkcionalitu

Po funkcionální stránce byly požadavky mnohem podrobnější. Potenciál aplikace je díky práci s databází vozů o jedinečné velikost, tedy aspoň v rámci České Republiky, opravdu obrovský a díky tomu je tedy velké množství služeb, které mohou být nabízeny, navíc mohou být tyto služby díky velikosti této databáze opravdu spolehlivé, příkladem může být služba, která provádí ocenění vůz na základě jeho parametrů, tak že spočítá průměr ze všech známých cen vozů se stejnými parametry. Přesnost a spolehlivost této služby jsou úměrné počtu záznamů, se kterými tato služba pracuje.

Fakt, že aplikace pracuje s takovouto databází vozů poté do velké míry určuje povahu služeb, které poskytuje uživateli. Výsledná aplikace byla tedy po funkcionální stránce rozdělena na následující 4 služby pro uživatele.

- vyhledávání vozů
- vyhledávání konkrétních vozů
- odhad hodnoty vozu
- zobrazení vývoje hodnoty vozu

3.2.1 Vyhledávání vozů

Primární službou je vyhledávání nabízených vozů na základě zadaných parametrů. Uživatel tedy do formuláře zadá informace o vozu, o které má zájem a následně na základě známých záznamů v databázi, pokud samozřejmě nějaké existují, jsou mu zobrazeny výsledky tohoto hledání. Uživateli musí být přehledně zobrazeny jednotlivé vozy, tak aby jejich důležité parametry měl přístupné bez dalšího klikání. Uživatel také musí mít možnost projít si všechny výsledky hledání tak, aby se v nich mohl dobře orientovat. Obdobná funkcionalita je nabízena většinou inzerčních webů, jelikož se jedná o uživateli přívětivý přístup zobrazení dat z databáze, tedy nabídky vozů. Uživatel musí mít také možnost v případě zájmu o vůz možnost přejít na inzerát na inzerční server, ze kterého byl získán.

3.2.2 Vyhledávání konkrétních vozů

Další požadovanou službou je vyhledávání konkrétních vozů. Aplikace musí umožňovat vyhledávat jeden konkrétní vůz, nebo alespoň co nejpřesněji specifikovaný vůz mezi autobazary, tak, aby si uživatel mohl tyto vozy mezi sebou porovnat a vybrat, která nabídka tohoto vozu je pro něj nejvýhodnější a nejzajímavější. Uživatel by měl mít možnost vybrat si mezi vyhledáváním na základě unikátního čísla vozu, tedy VIN[13] kódu a mezi vyhledáváním pomocí formuláře podobnému u první služby. Nejpodstatnějším rozdílem mezi těmito dvěma službami by měl být způsob zobrazení vozů a to tak, aby pod jedním konkrétním vozem, u kterého budou zobrazeny jeho obecné parametry uživatel také viděl konkrétní nabídky na inzerčních serverech společně s jejich specifickými parametry, tedy například cenou, která je specifická pro každou nabídku.

3.2.3 Odhad hodnoty vozu

Třetí služba uživateli nabízí odhad hodnoty vozu na základě jím zadaných parametrů. Uživatel tedy musí do formuláře zadat přesné údaje o jeho voze tak, aby byl vůz dostatečně přesně specifikován. Některé parametry vozu jsou v tomto případě nedůležité, například barva vozu a z tohoto důvodu se formulář této služby bude lišit od formuláře používaného u prvních dvou služeb a to tak, že bude

[13] VIN kód je jednoznačný identifikátor vozidla

ochuzen o parametry, které jsou irelevantní vzhledem k hodnotě vozu. Následně by uživateli měla být vypočítána co nejpřesnější hodnota jeho specifikovaného vozu, a to na základě známých vozů v databázi.

3.2.4 Zobrazení vývoje hodnoty vozu

Poslední požadovaná, čtvrtá služba by měla uživateli dát možnost zobrazit si vývoj hodnoty vozu, opět podle jeho specifikací tak, aby to bylo srozumitelné, nejlépe v podobě grafu. Uživatel by měl mít možnost zobrazit si vývoj hodnoty vozu v závislosti na dvou parametrech, což jsou:

1. stáří vozu
2. počet najetých kilometrů

Tyto dva parametry jsou vždy pro vývoj ceny vozu klíčové. Součástí této funkcionality by měla být možnost vidět budoucí vývoj hodnoty vozu v závislosti na známých datech.

4 Analýza

Tato kapitola se důkladně zabývá zadáním práce, podrobně rozebírá a vyhodnocuje jednotlivé požadavky na aplikaci tak, jak jsou zadány v kapitole 3 a zvažuje jejich praktické a nepraktické stránky. Jedná o jednu nejpodstatnějších částí vývoje aplikace obecně, jelikož analýza požadavků udává směr, kterým se následně ubírá další vývoj aplikace a v případě že dojde ke špatné specifikaci požadavků, nebo se následně provede špatná analýza, tak to může být pro projekt fatální.

Konkrétně se tedy kapitola zabývá analýzou technických požadavků na aplikaci tak, aby byly všechny požadavky splněny. Následně také zkoumá požadavky na funkcionalitu aplikace, tedy na požadavky na všechny služby. Na základě této analýzy jsou následně v kapitolách 5 a 6 popsány návrh a následná implementace.

4.1 Analýza informací dostupných z inzerčních serverů

Tato analýza byla prováděna ve spolupráci s kolegou Kovářem Tadeášem, na jehož bakalářskou práci tato práce navazuje. Jelikož jeho prací je získávání informací z těchto serverů, tak prováděl tuto analýzu především on, načež se ukázalo, že by bylo vhodné, aby byly ukládány téměř všechny informace, které lze nalézt v inzerátech, jelikož jsou téměř všechny dostupné informace o vozech a jejich nabídkách podstatné.

4.2 Analýza technických požadavků

Jelikož z požadavků na technickou stránku aplikace bylo nejpodstatnější že se má jednat o webovou aplikaci, tak je nejdůležitějším bodem vybrání vhodného programovacího jazyku, který bude využíván na serverové straně aplikace. V současné době patří k nejrozšířenějším jazykům pro toto využití PHP, Python, Ruby, C# a Java. Co se týče rozšířenosti těchto jazyků v oblasti webových aplikací, tak PHP a C# patří v současné době ještě stále k nejrozšířenějším jazykům, a to především díky jejich rozšířenosti v minulosti. V současnosti jazyky Python a Ruby získávají více na popularitě, a to především díky jejich jednoduché syntaxi, rychlosti a dalším mnoho důvodům. Pro tuto práci byl však zvolen jazyk Java, a to především díky zkušenostem autora s tímto jazykem. Obecně je jazyk Java vhodnější pro vytváření větších aplikacích, díky jeho robustnosti a bezpečnosti, avšak nic nebrání jeho použití při tomto projektu. Více k tomuto jazyku je popsáno v kapitole 2.4

4.3 Analýza jednotlivých služeb

Tato kapitola a jejích podkapitoly se zabývají analýzou požadavků na jednotlivé služby.

4.3.1 Vyhledávání vozů

Služba pro vyhledávání vozů je provozována snad všemi inzerčními weby, a to především proto, že je to nejjednodušší a uživatelsky nejpříjemnější způsob, jak uživateli zobrazit data o vozech. Navíc tím i tato služba využívá plný potenciál databáze vozů, jelikož jde pomocí tohoto vyhledávání najít kterýkoliv vůz v databázi. I když každý inzerční web má tuto službu trošku jinak navrženou, tak v konečném důsledku se jedná o tu stejnou funkcionalitu. Je tedy na místě, aby byla součástí i této aplikace, tak jak je požadování v zadání. Jelikož se jedná o primární funkci této aplikace, tak při přístupu do aplikace je uživatel přímo přesměrován zde, pokud tedy samozřejmě nepřistupuje cíleně na jinou službu.

Nejlepším způsobem, jak tuto službu nabídnou uživatelům je vyhledávací formulář. Tedy formulář s několika výběrovými poli, které nabízí uživateli množinu hodnot, například vstupní pole pro výběr značky nabízející množinu značek vozů.

Bylo by vhodné, aby hodnoty v těchto polích byly aktuálně zobrazovány na základě ostatních zadaných parametrů. Například pokud uživatel vybere ve formuláři značku Škoda, tak se musí hodnoty v poli obsahující modely vozů aktualizovat tak, aby nabízelo správné modely. Nyní řekněme že uživatel vybere model Superb, tak se mu ve výběrovém poli s barvami načtou všechny barvy, ve kterých je vůz Škoda Superb dostupný a řekněme že mezi tyto barvy patří i hnědá, avšak hnědá barva se vyskytuje pouze u těchto modelů vozů starších než s rokem výroby 2002. (Stále se jedná pouze o příklad). Když v tento moment uživatel zadá rok výroby od 2003, tak se mu automaticky aktualizují barevné varianty vozu a v důsledku toho už zde nebude dostupná hnědá barva. Více o dynamicky doplňovaných formulářích naleznete v kapitole 6.

Je také vhodné, aby byla před započtením vyhledávání kontrolováno, zda uživatel zadal alespoň značku vozu, který chce hledat, aby nedocházelo k tam velkému náporu na databázi. Po vyplnění pole se značkou tedy může uživatel provést vyhledávání, kdy mu následně budou zobrazeny vozy nalezené v databázi. Výsledky vyhledávání by měly být rozděleny na stránky tak, aby uživatel mohl mezi těmito výsledky vyhledávání listovat. Uživatel také samozřejmě musí mít data vhodně zobrazeny, aby mu byly ihned dostupné klíčové informace o voze, přičemž další informace jako je například výbava mohou být zobrazeny na vyžádání.

Samotný vyhledávácí formulář by měl umožňovat vyhledávání na základě následujících parametrů vozu:

- značka vozu
- model vozu

- rok výroby (od a do)
- karoserie
- palivo
- tachometr (od a do)
- cena (od a do)
- barva
- objem motoru (od a do)
- výkon (od a do)

Obdobné služby

- Sauto.cz - jedna z největších stránek pro inzeráty, umožňuje na začátku vyhledávat podle několika parametrů, pro provedení vyhledávání nabízí ještě více možností specifikovat toto vyhledávání
- Cars.cz – podobné jako Sauto.cz, taky nabídne rozšířené vyhledávání až po prvním vyhledání, avšak co se týče obsahu dat tak Cars.cz nabízí více záznamů
- Anonce – obsahuje méně dat než Sauto.cz a Cars.cz, avšak nabízí podobnou službu pro vyhledávání, navíc přidává uživatelsky velmi přívětivou mapu pro vyhledávání podle lokality

4.3.2 Vyhledávání konkrétních vozů

Tato služba se může uživateli při zobrazení vyhledávácího formuláře zdát podobná, či totožná službě popsané v bodě 4.3.1, a to především díky stejnému vyhledávácímu formuláři, a díky funkci vyhledávání vozů, avšak není tomu tak. První znatelným rozdílem je zobrazení dvou vyhledávacích formulářů, z nichž první by měl být obdobný či totožný s formulářem v předchozí službě včetně dynamického načítání formulářových hodnot, avšak druhý by měl nabízet možnost vyhledávání podle VIN kódu vozidla. Největším rozdílem mezi těmito prvními dvěma službami je však účel této služby. Jedná se o službu, která nehledá obecně všechny konkrétní nabídky vozů na všech inzerčních serverech, avšak sdružuje nabídky vozů patřících k sobě, díky čemž uživatel může vidět více nabídek jednoho konkrétního vozu na různých inzerčních serverech. K tomu jsou mu samozřejmě zobrazeny informace o tomto voze, ale i zvláště informace ke každé nabídce tohoto vozu.

Vyhledávání pomocí VIN kódu tedy funguje obdobně, pouze s jediným rozdílem a tím je to, že je nalezen 1 či 0 vozů a k nim 1 a více nabídek. Nikdy se nestane že bude nalezeno více vozů, jelikož je VIN jednoznačný identifikátor.

Tato služba uživateli může pomoci v situacích, kdy bude mít zájem o jeden konkrétní vůz, jehož zná přesná specifika, či vin kód, tak si pomocí této služby může tento vůz najít a následně okamžitě vidí porovnání cen v jednotlivých inzerátech na jednotlivých inzerčních serverech.

Tato služba je svým konceptem ojedinělá, minimálně tedy v rámci České Republiky a jednoznačně má své místo, jelikož dokáže uživateli ušetři spoustu času a zobrazit mu informace, které by jinak hledat na různých serverech na jednom místě. Existuje mnoho obdobných služeb, které se zaměřují především na získávání informací o voze především na základě jeho VIN kódu, avšak tyto služby se nedají brát jako přímá konkurence.

Obdobné služby

- VinCheck – nabízí možnost ověřit si informace o vozu na základě jeho VIN kódu, uživateli dokáže získat informace o voze jako jsou model, značka, datum registrace a výroby a další
- autoDNA – nabízí uživateli možnost ověřit více informací o autě podle zadaného VIN kódu, například zda není kradené, zda bylo kradené a další

4.3.3 Odhad hodnoty vozu

Mnoho uživatelů přichází na inzerční stránky za účelem koupě konkrétního vozu a mnoho z nich si často ani není jisto jakou přesně cenu, tento vůz, který chtějí má. Poté existují další uživatelé, kteří jednoduše jelikož se chtějí zbavit svého vozu, tak potřebí znát jeho cenu. Obě skupiny těchto uživatelů většinou postupují tak, že si vyhledají tento vůz dle jejich specifikací, následně je jim vráceno několik desítek, stovek, či tisíc záznamů, které samozřejmě mají velký cenový rozsah a pro uživatele je v takovéto situaci opravdu složité určit, jakou hodnotu asi jejich vůz opravdu má. Tento problém řeší právě služba provádějící odhad ceny vozu. Uživatel zde jednoduše zadá určité informace o vozu do vstupního formuláře a na základě těchto informací je mu vypočtena hodnota tohoto vozu. Formulář by se měl v tomto případě liší od formulářů použitých v předchozích službách, tento formulář totiž obsahuje pouze parametry, které jsou relevantní k hodnotě vozu, konkrétně jsou to tyto parametry:

- značka
- model
- karoserie
- palivo
- výkon
- objem motoru
- tachometr
- rok výroby
- cena nového vozu

Následně je cena vypočítávána dvěma různými způsoby. Prvním způsobem je výpočet pomocí váženého průměru. V tomto případě stačí, aby uživatel zadal pouze značku vozu a může následně provést výpočet ceny. Aby byl výpočet ceny co nejpřesnější, je vhodné, aby byl vypočítán s co nejvíce záznamů, a proto by některé ze vstupních polí mohli být při vyhledávání v databázi brány s tolerancí.

Pokud uživatel poté přidá do vyhledávání například rok výroby vozu, tak se spočítá výsledná cena na základě známých záznamů v databázi, které mají rok výroby v určité toleranci, kdy na základě toho, jak moc jsou si tento hledaný vůz a vůz v databázi podobné se určí váha tohoto vozu a následně je cena vypočítána jako vážený průměr. Ne všechny parametry formuláře by měly být udělány jako výběrové vstupní pole, ale některé by měly být pouze normálně textové vstupní pole z důvodu, aby zde uživatel mohl zadat i hodnotu která není známa v databázi, konkrétně se jedná o tyto parametry: výkon, objem motoru, tachometr a rok výroby. Pole pro cenu nového vozu není pro tento výpočet nijak důležité. Pokud je nalezeno malé množství výsledků vyhledávání měl by být uživatel upozorněn, že výsledek nemusí být naprosto přesný. Více k tomuto výpočtu naleznete v kapitole implementace.

Druhý způsob výpočtu ceny je vhodný udělat na základě teoretického výpočtu. Pro tyto účely, tedy konkrétně naceňování motorových vozidel existuje znalecký posudek č.1/2005 [12], který se právě touto problematikou naceňování zabývá. Pro využití tohoto výpočtu ceny musí být znám objem motoru vozu, počet najetých kilometrů, rok výroby a cenu nového vozu. Pokud uživatel některý z těchto parametrů nezadá a nechá si vypočítat cenu vozu je na to následně upozorněn a může doplnit chybějící informace a znovu si nechat cenu přepočítat. Prvním krokem při stanovení ceny vozu na základě znaleckého posudku je výpočet tzv. základní amortizace, která je vyjádřena v procentech. Její výpočet probíhá následovně:

> *„Základní amortizace se zpravidla stanoví jako aritmetický průměr dvou srážek, a to srážky za dobu provozu (ZAD) a srážky za počet ujetých km (ZAP). Způsob stanovení těchto srážek je podrobně popsán v kap. 2.2.1 ZS1.“ [13]*

Pro určení základní amortizace je tedy potřeba vypočítat tyto dvě srážky. Pro srážky za dobu provozu jsou relevantní informace objem motoru a také samozřejmě stáří vozu a pro srážku za počet ujetých kilometrů jsou důležité objem motoru a počet najetých kilometrů vozu. Následným vypočítáním průměru mezi těmito srážkami je stanovena základní amortizace. Následně je vypočítána redukovaná technická hodnota vozidla, a to tak že se z původní ceny vozidla odečte základní amortizace. Následně stačí na základě původní ceny vozidla, kterou zadal uživatel do formuláře a redukované technické hodnoty vozidla se už následně stanoví výchozí cena vozidla a tato cena vozidla je poté zobrazena uživateli.

V současné době existuje na internetu několik obdobných služeb, sloužících většinou za účelem následného odkupu vozu, z nichž některé jsou dokonce placené.

Obdobné služby

- http://www.ocenauto.cz/cs/ - nabízí zpoplatněné oceňování vozu, které je však prováděno lidmi, jedná se tedy o službu mnohem náročnější na zdroje
- https://www.autoesa.cz/vykup/oceneni-vozu - nabízí bezplatné ocenění vozu, které je však provozováno především za účelem výkupu vozu

4.3.4 Zobrazení vývoje ceny vozu v čase

Služby pro zobrazení vývoje ceny vozu není mezi inzerčními weby autobazarů vůbec rozšířená, podobně jako služba pro odhad ceny vozu. Mnoho uživatelů při koupi či prodeji vozu zvažuje, zda je na to právě vhodná doba, co by se stalo, kdyby uživatel s prodejem nějaký čas počkal, nebo jak moc klesne cena vozu za nějakou dobu. Uživatel by mohl také chtít vědět jakou hodnotu tento vůz měl před několika lety, jestli se mu tedy oplatí ho prodat. V tomto případě by uživateli mohla být výpomocná služba pro zobrazení vývoje ceny vozu

Uživatel by měl mít možnost prohlédnout si vývoj hodnoty vozu, způsobem, aby mu data byly co nejjednodušeji srozumitelní. Nejlepším způsobem, jak takto uživateli zobrazit data je graf. Hodnota vozu by měla být uživateli zobrazena v závislosti na minimálně jednom parametru, kterým je stáří vozu, které většinou hraje největší roli, stejně vhodné je však i zobrazení vývoje hodnoty na základě počtu najetých kilometrů. Důkazem důležitosti těchto dvou parametrů vozu může být způsob oceňování prováděný oceňovacím předpisem č.1/2005, o němž naleznete více v kapitole 4.3.3.

Formulář, který je v této službě uživateli zobrazen je podobný formuláři, který využívá služba pro oceňování vozů. Důležitým rozdílem mezi těmito dvěma formuláři je fakt, že u této služby by měly být její výsledky okamžitě po změně jakékoliv informace. Uživatel tedy musí opět zadat minimálně značku vozu, načež jsou mu následně zobrazeny grafy zobrazující vývoje hodnot dle stáří a dle počtu najetých kilometrů. Pokud uživatel následně změní například model vozu, jsou mu následně překresleny tyto grafy dle požadovaného vyhledávání bez jakéhokoliv dalšího potvrzování formuláře. Dalším rozdíl je to, že v tomto případě by všechny vstupní pole formuláře měly být výběrové vstupní pole a žádný z nich nebýt textovým vstupním polem a to proto, že tato aplikace nepracuje s tolerancemi ale pouze na základě přesně odpovídajících existujících dat.

Velkou výhodou této služby je také to, že by uživateli měla být schopna zobrazit budoucí vývoj hodnoty toho vozu, a to na základě křivky, kterou budou proloženy existující data. Na základě této křivky by poté měl uživatel vidět kam se pravděpodobně bude posouvat vývoj ceny jeho vozu.

Stejně jako v ostatních službách, tak i zde, v případě že není nalezen dostatečný počet záznamů pro vykreslení grafu, tak je na to uživatel upozorněn a v případě že změní požadavky vyhledávání, tak jsou opět vykresleny adekvátní grafy.

Obdobná služba není v současné době provozována snad žádným známým inzerčním webem v České Republice.

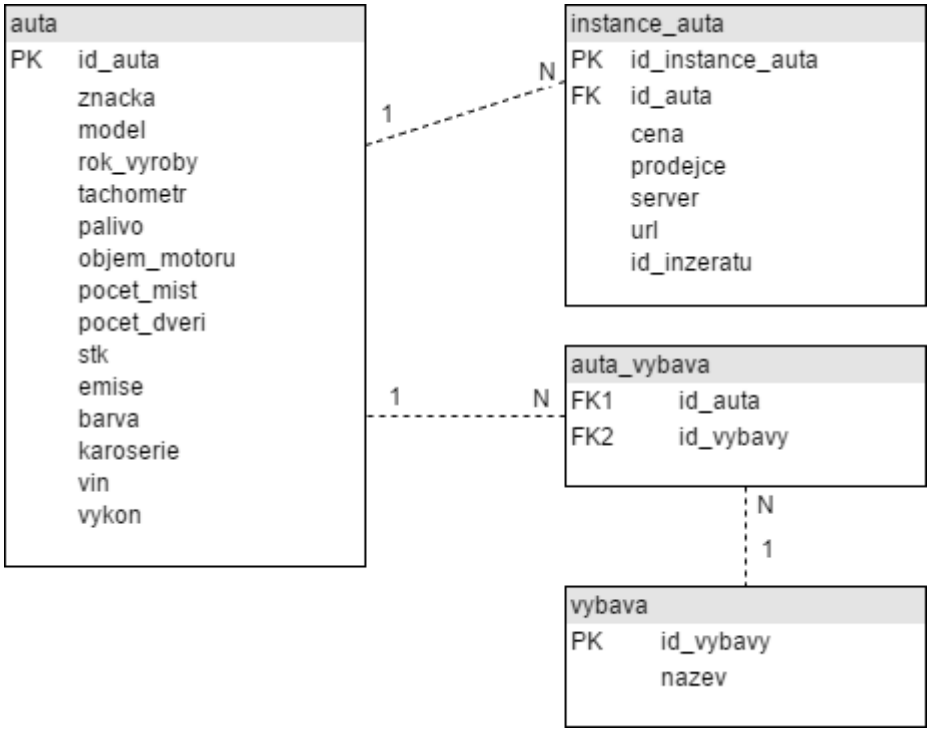
5 Návrh

Tato kapitola se zabývá návrhovou částí vývoje celé aplikace. Návrh aplikace obvykle v obecném procesu vývoje aplikace navazuje na analýzu požadavků na aplikaci. Jedná se tedy o třetí krok při vývoji aplikace, který je neméně důležitý, protože stejně jako v případě stanovení požadavků a analýzy požadavků je dobrý návrh dalším krokem na cestě k celkově dobrému výsledku. Na základě návrhu a jeho jednotlivých částí se totiž následně řídí celá implementace aplikace.

Konkrétně tato kapitola zobrazuje a popisuje diagramy použité při vytváření návrhu aplikace. Mezi tyto diagramy zde patří entitně-vztahový diagram, který v tomto případě zobrazuje uložení jednotlivých tabulek v databázi a vztahy mezi těmito tabulkami. Dále se zde nachází diagram případů užití a jeho popis. Tento diagram zobrazuje skupiny uživatelů, kteří mohou s aplikací interagovat a také ukazuje jakým způsobem s ní mohou interagovat. Další podkapitolou je diagram tříd, zobrazující jednotlivé třídy, které budou následně v implementaci použity, a vztahy mezi těmito třídami. Poslední částí této kapitoly je návrh grafického uživatelského rozhraní, a to především proto, že intuitivní orientace v aplikaci a uživatelsky jednoduše srozumitelné zobrazení dát bylo jedním z požadavků na tuto aplikaci.

5.1 Entitně-vztahový diagram

Entitně-vztahové diagramy jsou založeny na Entitně-vztahovém modelu, který se zabývá abstraktním zobrazením dat a vztahy mezi nimi. V tomto případě se používá pro zobrazení modelu databáze, tedy zobrazení tabulek databáze, jejich sloupců a zobrazení vztahu mezi tabulkami. Model databáze vozů, se kterou pracuje tato aplikace je zobrazena na obrázku Obrázek 5.1.1: ER Diagram



Obrázek 5.1.1: ER Diagram

Jak jde vidět z obrázku, tak struktura databáze se skládá ze 4 tabulek. Základní tabulkou, na kterou jsou následně pomocí cizích klíčů navázány ostatní tabulky je tabulka `auta`. V této tabulce jsou uloženy informace o vozech jako takových, nikoliv o jejich konkrétních nabídkách, k tomu slouží tabulka `instance_auta`, Tímto způsobem se dobře docílí možnosti vyhledávání více nabídek konkrétních vozů, neboť pokud je inzerát na stejné auto na třech různých inzerčních serverech, budou pro něj existovat 3 záznamy v tabulce `instance_auta`, avšak pouze jeden záznam v tabulce `auta`. Podle obrázku si také jde udělat představu o tom, které parametry patří vozu jako takovému a které patří jeho konkrétní nabídce. Například parametr výkon se vždy váže přímo na specifický vůz, kdežto cena se svázána s konkrétním inzerátem. Další tabulkou v databázi je tabulka `vybava`. Tato tabulka obsahuje pouze dvojice identifikátor výbavy a jméno této výbavy. Je zde tedy uložen seznam všech možných výbavových prvků, které se ve vozech nacházejí. Jelikož se výbava váže přímo na konkrétní vozy, tedy tabulku `auta`, tak v tomto místě vzniká takzvaný vztah N ku N[14], který znamená že na jeden vůz může být navázáno více kusů výbavy, avšak stejně tak na jeden konkrétní kus výbavy může být navázáno více vozů. Proto je v tomto místě použita vazební tabulka `auta_vybava`, obsahující pouze identifikátory vozu a k němu patřící výbavy, která svým principem ruší vztah N ku N a rozbíjí jej na vztahy 1 ku N.

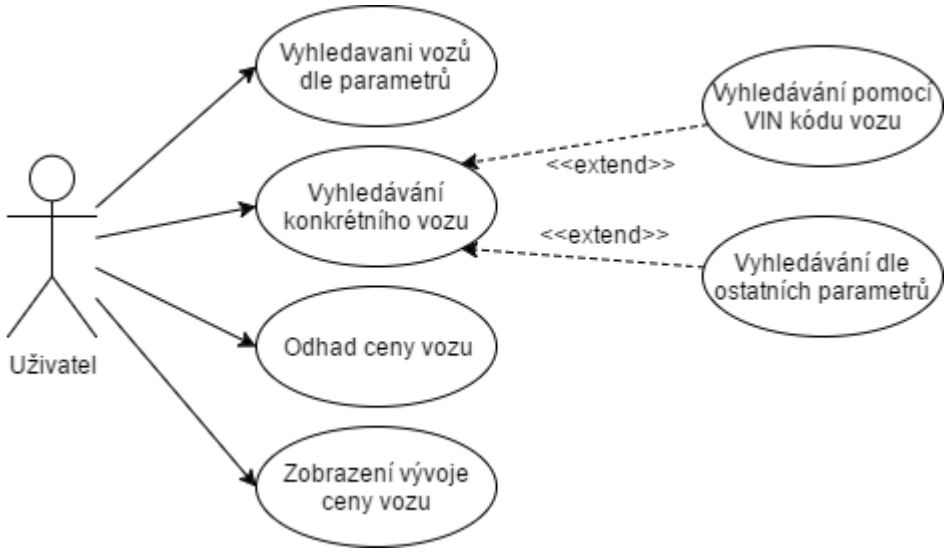
[14] Vztah N:N musí být rozdělen použitím vazební tabulky na vytahy 1 ku N, více naleznete na http://www.linuxsoft.cz/article.php?id_article=854

5.2 Digram případů užití

Diagram případů užití zachycuje chování systému z hlediska uživatele, používá především pro zobrazení různých rolí, které mohou uživatelé v rámci aplikace nabývat a následně ke zobrazení možností, jak mohou tito uživatelé podle jejich rolí se systémem interagovat. Zobrazuje také chování systému v závislosti na chování uživatele, tedy pokud některá z uživatelem požadovaných akcí potřebuje provést akci jinou, tak to může být v diagramu znázorněno.

V této aplikaci, jak můžete vidět na obrázku **Chyba! Nenalezen zdroj odkazů.** existuje pouze j ediná role, kterou může uživatel mít, a to je obyčejný uživatel služby, tedy návštěvník webové aplikace, v diagramu pod jménem `Uživatel`, který přistupuje do aplikace za účelem využít některé z jejích služeb. Kupříkladu v aplikacích, které obsahující administrátorskou správu je možné, aby dalším uživatelem v tomto diagramu byl administrátor, který může provádět jiné akce než obyčejný uživatel.

Tento uživatel má tedy v aplikaci možnost využití jedné ze čtyř služeb, které aplikace poskytuje, z nichž konkrétně druhá služba nabízí vyhledávání konkrétního vozu, které nabízí dvě možnosti, které jsou zde vyobrazeny vztahem `extend`, což znamená, že aby tedy uživatel mohl provést vyhledávání konkrétního vozu, tak musí buď použít vyhledávání pomocí VIN kódu vozu, či vyhledávání pomocí ostatních parametrů vozu.



Obrázek 5.2.1: Diagram případů užití

5.3 Diagram tříd

Diagram tříd je jedním ze 13 typů diagramů definovaných grafickým modelovacím jazykem UML. Zobrazuje strukturu objektových tříd a jejich vzájemné vztahy v celém systému. Obecně existují tři typy diagramů tříd, které jsou jazyk UML [14] podporovány a to konceptuální, specifikační a

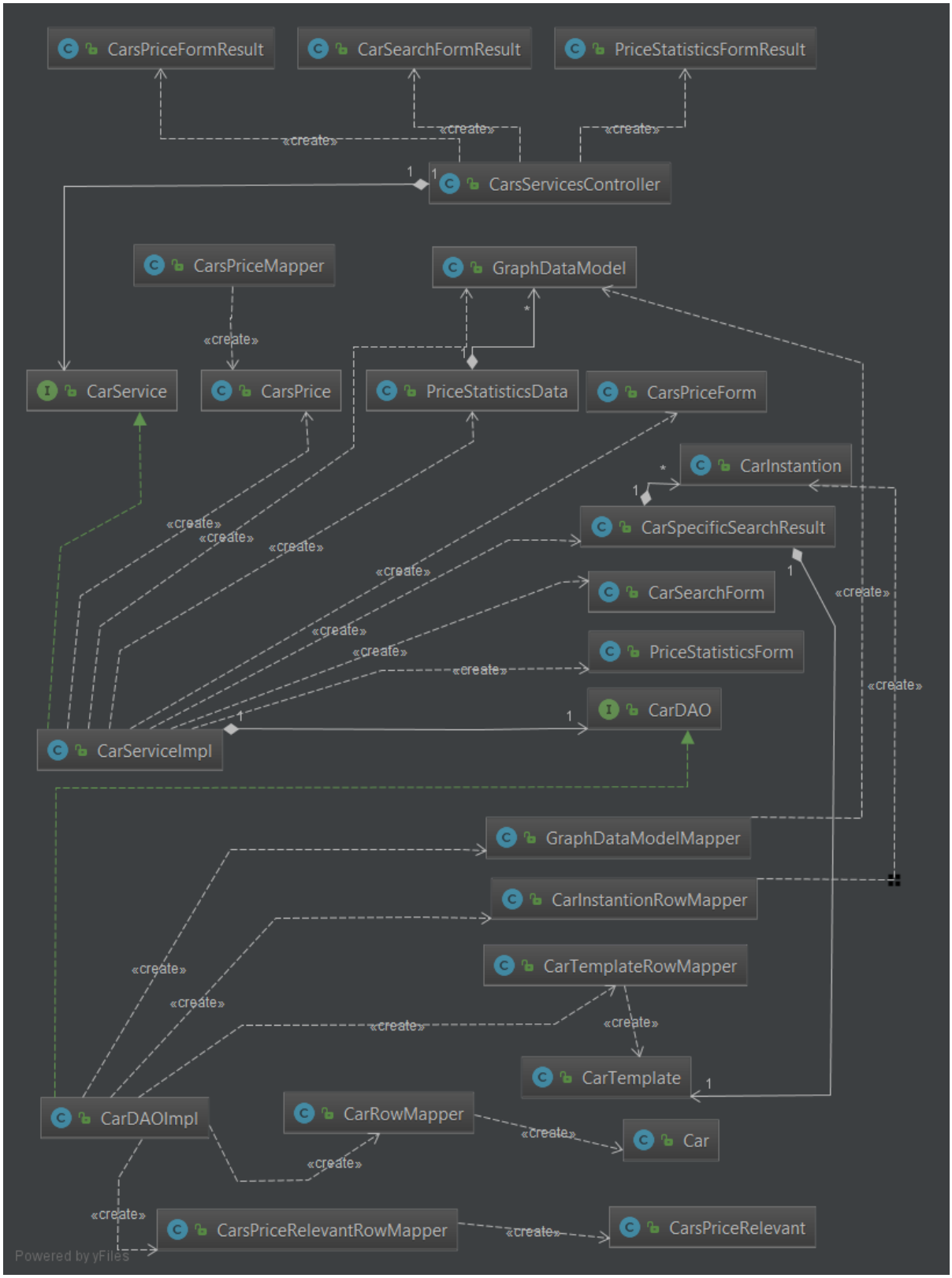
implementační. Poslední z nich, implementační, zobrazuje hotový model tříd, včetně proměnných a metod těchto tříd a využívá se pro zobrazení modelu hotové aplikace. Specifikační model specifikuje především rozhraní bez udání konkrétní implementace, používá se především pro vytvoření návrhu, kdy ještě neexistuje přesná představa o implementaci. Konceptuální pohled má za úkol zmapovat třídy zastupující objekty bez implementačních detailů. Jedná se o zcela implementačně nezávislý model.

Objektový model této aplikace z velké části vyplývá z použitých technologií pro její chod. Jelikož je v aplikaci použit modul frameworku Spring, Spring MVC, jsou tedy třídami, které jsou v hierarchii tříd nejvýše postavené, tzv. kontrolery. Tyto kontrolery se zabývají zpracováváním HTTP požadavků, kdy jsou tyto požadavky podle jejich adresy a podle typu delegovány na jednotlivé metody. Metoda `getCarsPrice`, tedy například odpovídá adrese v prohlížeči `/GetCarsPrice` a typu požadavku `post`.

V této aplikace se používá pouze jeden kontroler pro mapování všech požadavků na všechny služby, konkrétně se jedná o třídu `CarsServicesController`, která následně provádí veškerou aplikační logiku, která je na základě požadavku požadována. Pro provádění této aplikační logiky se následně využívá rozhraní služeb, konkrétně třída `CarService`, jejíž implementace `CarServicesImp` následně využívá pro přístup k datům z databáze rozhraní `CarDAO`. Tato třída má stejně jako u služby svojí implementaci jménem `CarDAOImpl`, která poskytuje rozhraní pro přístup k databázi. Pro práci s databází tato třída využívá třídu JdbcTemplate[15], která je součástí frameworku Spring. Zbylé třídy slouží buďto jako Java beany[16] pro uložení výsledků dotazů na databázi v Javě, nebo jako mapování pro výsledky zmíněných dotazů. Tento druhý typ tříd má obvykle příponu `Mapper`, či `RowMapper`.

Na následujícím obrázku Obrázek 5.3.1: Zjednodušený model tříd, jak už název napovídá lze nalézt zjednodušený model tříd, zobrazující pouze vztahy mezi třídami, nikoliv však jejich proměnné, či metody.

[15] JdbcTemplate - https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html
[16] Java beany - https://www.tutorialspoint.com/jsp/jsp_java_beans.htm



Obrázek 5.3.1: Zjednodušený model tříd

5.4 Návrh grafického uživatelského rozhraní

Návrh intuitivního grafického uživatelského rozhraní je důležitým bodem vývoje každé webové aplikaci, jejíž cílovou skupinou jsou normální uživatelé na internetu. Každá aplikace, kterou takovýto uživatel používá by na něj měla působit srozumitelně a měl by být schopen se v ní jednoduše orientovat. Aplikace používá tři základní rozložení. První z nich je zobrazeno na obrázku Obrázek 5.4.1: Rozložení č.1.

Obrázek 5.4.1: Rozložení č.1

Jak jde vidět na obrázku, tak se rozhraní skládá ze dvou základních oken, což jsou na levé straně menu, které umožňuje uživateli přepínání mezi jednotlivými službami aplikace. Toto menu zůstává stejné ve všech částech aplikace a je uživateli vždy zobrazeno, jedinou proměnnou je znázornění, na které službě se uživatel právě nachází. Na pravé straně se pak nachází blok s obsahem, konkrétně s vyhledávacím formulářem, či dvěma formuláři. Toto rozložení se používá ve všech 4 službách, kdy jsou formuláře pro každou službu trošku jiné. Toto rozložení zůstává aktivní, dokud se neprovede hledání, či neprovede první odeslání formuláře. Výjimkou je formulář ve službě pro vykreslování závislostí hodnoty vozu, kdy je formulář odeslán hned při změně některého z jeho parametrů.

Druhým rozložením používaným v aplikaci je rozložení obsahující další karty s výsledky, kdy je však stále viditelný i formulář služby. Oproti prvnímu rozložení se zde v pravé části rozhraní nachází další karta s výsledky. Toto rozložení je využíváno službami pro odhad hodnoty vozu a pro zobrazení vývoje hodnoty vozu, kdy se do již zmíněné nové karty zobrazují výsledky služby. Na následujícím obrázku Obrázek 5.4.2: Rozložení č.2 je toto rozložení ukázáno.

Obrázek 5.4.2: Rozložení č.2

Třetí rozložení je používání službami pro vyhledávání vozů a vyhledávání specifických vozů, kdy se po odeslání vstupního formuláře, tedy požadavku na vyhledání vozů, zobrazí nové okno, které má sice stále stejnou levou část, tedy nabídku služeb, avšak v pravé části jsou zobrazeny karty s výsledky hledání, přičemž je možnost mezi stránkami s těmito výsledky listovat. Toto rozložení je zobrazeno na následujícím obrázku.

Obrázek 5.4.3: Rozložení č.3

6 Implementace

Implementace aplikace vychází z velké části z návrhu, který je zpracován v předchozí kapitole 5. Většina věcí byla implementována tak bylo navrženo, a to včetně diagramu tříd, jehož plná verze lze nalézt v přílohách. Tato kapitola se tedy bude věnovat implementaci jednotlivých služeb a jejich uživatelského rozhraní a dále implementací dalších částí aplikace.

6.1 Implementace služeb

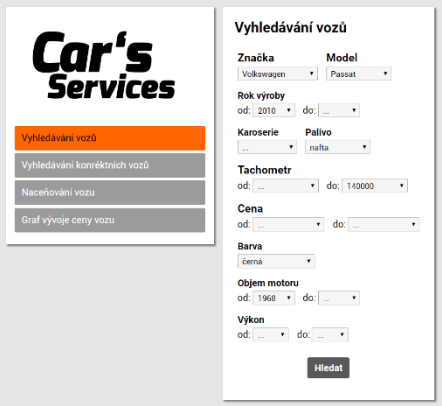
Tato podkapitola se zabývá detailněji implementací jednotlivých služeb. Veškerý tok HTTP požadavků aplikací obstarávají kontrolery aplikace. V případě této aplikace se jedná pouze o jeden jediný kontroler jménem `CarsServicesController`, kontroler obsahuje mapování jednotlivých požadavků, na jejichž základě poté provádí příslušnou aplikační logiku a následně navrací jméno příslušného pohledu, který má být vykreslen. Ukázku přesměrování nového požadavku na službu můžete vidět na ukázce kódu Ukázka kódu 6.1.1

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(ModelMap model) { return "redirect:/CarsSearch"; }
```

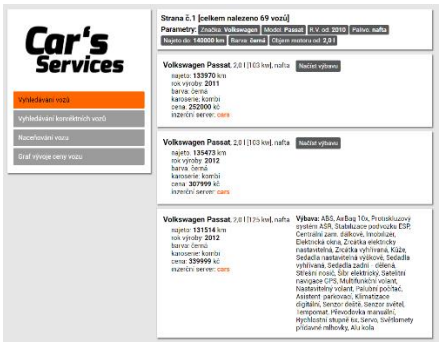
Ukázka kódu 6.1.1: Přesměrování nového požadavku v kontroleru

6.1.1 Vyhledávání vozů

Tato služba má na uživatelské straně implementován vyhledávací formulář, do kterého je uživatel nucen vyplnit alespoň jeden parametr, tím je značka vozu. Pokud má tento parametr vyplněn, tak je uživateli umožněno zahájit vyhledávání. Po kliknutí na tlačítko vyhledat je poslán na server požadavek s adresou obsahující cestu `/Search`, která je namapována na funkci `search`. Ta jako parametr obdrží onen vyhledávací formulář, který je poté následně předán do vrstvy se službami (anglicky Services), která společně s Dao vrstvou poskytuje přístup k databázi. Zde se provede získání dat z databáze za využití modelové třídy `Car` a za využití mapovací třídy `CarRowMapper`. Třída `JdbcTemplate`, která udržuje spojení s databází a obstarává samotnou komunikaci s databází provede dotaz na databázi, kdy se mí jako parametr pro dotaz předá instance mapovací třídy a výsledkem je seznam, v Javě list, objektů typu `Car`. Tento list je předán zpět z Dao vrstvy až do kontroleru, kdy je předán do příslušného pohledu a vykreslen uživateli tak, jak je ukázáno na obrázku Ukázka kódu 6.1.2Obrázek 5.4.3. Z tohoto obrázku je taky patrné, že je možné si pro každý vůz nechat dodatečně načíst výbavu. Na obrázku Ukázka kódu 6.1.1 je zobrazen vyhledávácí formulář.



Obrázek 6.1.1: Vyhledávací formulář



Obrázek 6.1.2: Výsledky vyhledávání

6.1.2 Vyhledávání konkrétních vozů

Tato služba je dostupná na adrese s cestou `/CarsSpecificSearch`. Služba pro vyhledávání konkrétních vozů nabízí uživateli dva odlišné formuláře, kterými může provádět vyhledávání.

První formulář, který umožňuje vyhledávání na základě parametrů vozu při odeslání vytvoří požadavek s cestou `/SearchSpecific`. Stejně jako u první služby pro vyhledávání je nalezena příslušná metoda v kontroleru, kterému je předán vyhledávací formulář. Tento formulář je opět předán do vrstvy služeb, kde se však data získávají trošku jinak. Konkrétní implementace je zobrazena ukázce kódu Ukázka kódu 6.1.2.

```
public List<CarSpecificSearchResult> searchSpecificCars(CarSearchFormResult form){
    List<CarSpecificSearchResult> searchResults = new ArrayList<>();
    List<CarTemplate> cars = carDAO.getCarTemplatesByParameters(form.getParametersMap());
    for(CarTemplate car : cars){
        List<CarInstantion> carInstantions = carDAO.getCarInstantionsByTemplateID(car.getIdOfCar());
        CarSpecificSearchResult result = new CarSpecificSearchResult(car, carInstantions);
        searchResults.add(result);
    }
    return searchResults;
}
```

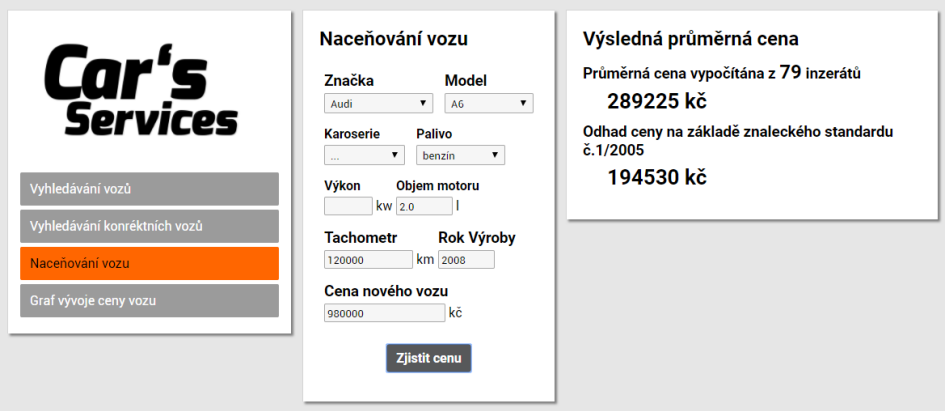
Ukázka kódu 6.1.2: implementace vyhledávání konkrétního vozu

Konkrétně můžeme vidět, že na základě vyhledávacího formuláře jsou prvně vyhledány vozy, ne instance vozů. Modelová třída pro uložení vozů se jmenuje `CarTemplate` a v Dao vrstvě je také použit i její mapovací třída `CarTemplateRowMapper`. Následně se prochází všechny tyto nalezené vozy a pro každý vůz jsou z tabulky obsahující nabídky vozů, ne vozy jako takové, vytáhnuty jejich veškeré známé nabídky. Pro uložení instancí vozů je použita modelová třída `CarInstantion`, ke které samozřejmě také existuje adekvátní mapovací třída `CarInstantionRowMapper`. Tyto nabídky jsou získány v listu a následně je vytvořena instance třídy `CarSpecificSearchResult`, která drží informace o jednom voze a jeho nabídkách. Tímto způsobem je vytvořen celá list těchto vozů a jejich nabídek, který je následně vrácen opět do kontroleru a kontroler tyto data promítne do odpovídajícího pohledu.

Obdobným způsobem a za využití stejných tříd je získáván výsledek pro vyhledávání pomocí formuláře pro VIN kód. Rozdíl mezi těmito dvěma formuláři je to, že při pokusu o vyhledání výsledků pomocí VIN kódu je prvně pomocí asynchronního dotazu zjištěno, zda takový vin v databázi existuje, pokud ano, tak je formulář normálně odeslán, v případě že neexistuje tak je na to uživatel upozorněn a formulář není odeslán.

6.1.3 Odhad hodnoty vozů

Tato služba je namapována na cestu `/CarsPricing`. Narozdíl od předchozích dvou službem je vyhledávací formulář ochuzen o pole, která jsou irelevantní pro zjišťování ceny vozu, jako jsou například barva, či samotná cena vozu. Navíc zde nejsou všechny vstupní pole řešeny jako výběrová pole, ale některé jako textové pole, a to za účelem zpřesnění výpočtu hodnoty vozu. Více o výpočtu naleznete v kapitole **Chyba! Nenalezen zdroj odkazů.**. Výsledkem této služby jsou dvě hodnoty, z nichž první je spočítána pomocí váženého průměru hodnot z databáze a druhá na základě znaleckého posudku. Výpočet na základě znaleckého posudku je blíže vysvětlen v kapitole 4.3.3.
V případě že má aplikace potřebné informace pro provedení obou výpočtů, tak může výsledná obrazovka uživatele vypadat nějak jako na obrázkuObrázek 6.1.3.



Obrázek 6.1.3: Ukázka úspěšného odhadu ceny

Další zajímavou vlastností této služby je, že formulář pro zjištění ceny je odesílán asynchronně, uživatel to tedy většinou ani nezaznamená, formulář zůstane vyplněný s původními daty pro případ, že by uživatel chtěl své vyhledávání upřesňovat a provádět zjištění ceny opakovaně.

6.1.4 Zobrazení vývoje hodnoty vozů

Pro vykreslování grafů zobrazující hodnoty ceny jsou v rámci této služby použity Google Charts[17]. Služba je mapována na cestu `/PriceStatistics`. Na vstupním formuláři této služby je zajímavý fakt, že při jakékoliv provedené změně ve formuláři nejsou pouze automaticky aktualizovány ostatní pole formuláře, ale zároveň je proveden asynchronní dotaz na server za účelem získaní aktuálních dat pro grafy a v případě že byl nalezen dostatek těchto dat, tak jsou tyto grafy překresleny.

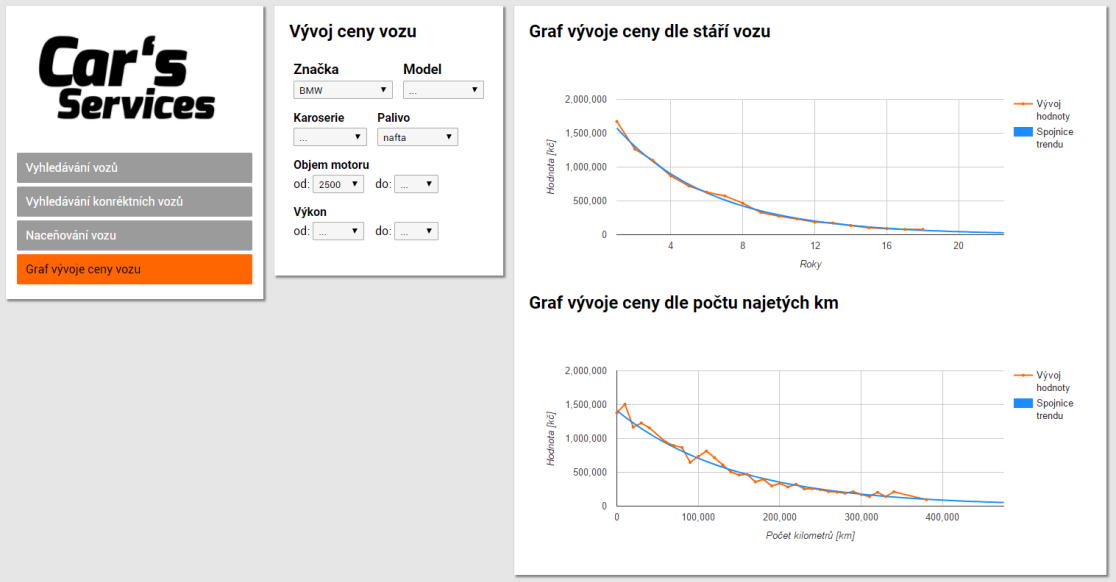
Pro vykreslení prvního grafu, který zobrazuje závilost ceny na stáří vozu, jsou jednoduše získány z databáze data podle zadaných parametrů, pro každý z roků je následně vypočítána průměrná cena a zároveň je uchována informace o počtu záznamů, ze kterých byla cena vypočítána.

Získání dat pro druhý graf už je o něco složitější, jelikož rozsah stáří automobilu může být v rozsahu 0 až například padesát let, tak u počtu najetých kilometrů může být rozsah 0 až milion kilometrů. Tímto způsobem by v případě počítání průměru na každý kilometr byl počet záznamů, ze kterých byl průměr vypočítán 1 a tím pádem by většina těchto dat byla považována za nedůvěryhodná a nevykreslená do grafu podle implementace v Javascriptu. Z tohoto důvodu je prvně provedené seskupení bodů do rozsahů po 10 000 km. Tímto se u většiny vozů dostaneme na rozsah hodnot x na 0 až 300.

Pro uložení jednoho bodu je použitá třída GraphDataModel, která v sobě drží informaci o x-ové souřadnici v grafu, průměrné ceně pro tuto souřadnici a o počtu záznamů ze kterých byla vypočítána. V Javě jsou všechny informace o bodech grafů uložené ve třídě PriceStatisticsData. Tato třída má dvě vnitřní proměnné. Jedná se o listy objektů typu GraphDataModel. Z kontroleru jsou tedy skrze vrstvu

[17] Knihovna pro vytváření grafů v Javascriptu, více na https://developers.google.com/chart/

služeb a Dao vrstvu získány data do obou grafů. Jelikož je stejně jako u služby pro výpočet hodnoty vozu volán tento HTTP požadavek asynchronně, tak jsou data vrácena do Javascriptu, který ještě provede jejich restruktuuralizaci tak, aby bylo možné body vykresli pomocí Google Charts. Javascript ještě zároveň provádí vyfiltrování bodů, a to na základě počtu záznamů, ze kterého byl tento bod vypočítán. Pokud je počet záznamů menší než 3, tak je bod považován za nedůvěryhodný a následně není vůbec vykreslen do grafu. Na obrázku Obrázek 6.1.4 je zobrazena služba při úspěšném načtení grafů.



Obrázek 6.1.4: Zobrazení vývoje hodnoty vozu

6.2 Implementace dalších částí aplikace

6.2.1 Dynamické formuláře

Dynamické formuláře jsou využívány všemi službami aplikace. Jedná se o formuláře tvořené výběrovými vstupními políčky a obsah těchto políček je dynamicky aktualizován na základě obsahu ostatních políček. Každá ze služeb má na tuto práci s formuláři vlastní třídy, ty se však navzájem liší pouze v proměnných, které tyto třídy obsahují. Příkladem může být služba pro vyhledávání automobilů, která pracuje s třídou `CarSearchFormResult`, která uchovává informace odeslané z formuláře.

Událost, která je vyvolána aktualizaci hodnot ve výběrových polích, způsobí provedení asynchronního volání na server, jemuž se zároveň předají data z formuláře tak, aby s nimi bylo možné dále pracovat. Zde se využije třída `CarSearchForm`, která obsahuje podobná pole jako `CarSearchFormResult` pouze s rozdílem v tom, že v `CarSearchForm` třídě jsou uchovány

listy parametrů. Provedením aplikační logiky se následně získají z nová data pro každé výběrové pole a ty se uloží do objektu třídy `CarSearchForm`.

Z kontroleru se tedy poté vrátí objekt třídy `CarSearchForm` obsahující nové hodnoty odpovídající omezením, které byly zadány v příchozím objektu třídy `CarSearchFormResult`. Javascript obdrží tedy objekt naplněný novými daty pro formuláře a následně dokáže updatovat hodnoty ve výběrových polích formuláře. Tímto tedy dojde k omezení ještě nezadaných parametrů již zadanými parametry.

Pro uživatele je tato akce, kdy se formuláře aktualizují, viditelná tak, že se na dobu trvání asynchronního požadavku na server zablokují vstupní pole a není možné s nimi manipulovat.

6.2.2 Odhad hodnoty vozu dle známých dat

Formulář využívaný ve službě pro ohodnocení vozu je specifický tím, že všechna jeho pole nejsou výběrové pole, ale některé z nich jsou obyčejné textové pole. Důvodem je potřeba dosáhnout co největší přesnosti při výpočtu hodnoty vozidla. A v případě že by uživatel prováděl vyhledávání formulářem, který má pouze výběrová pole, tak by počet výsledků byl v mnoha případech velmi omezený.

Služba tedy funguje tak, že opět provede asynchronní požadavek na server a zde jsou vybrány všechny záznamy pouze na základě polí značka, model, karoserie a palivo. Tyto informace předá Dao vrstva vrstvě služeb, kde se následně provede stanovení váhy každého záznamu, který byl z databáze získán. Tato váha se vypočítává na základě podobnosti informací, který byly zadány v textových vstupních polích a informací každého záznamu z databáze. Standartní váha každé ceny je 1.

Pro porovnání roků výroby vozu platí, že pokud se shodují, tak se váha vynásobí konstantou 100. V Případě, že se liší roky nanejvýše 2, tak je váha vynásobena konstantou 20. Stejné pravidla platí pro ostatní pole, takže pokud se výkon liší o nanejvýš 5 kw, tak je váha vynásobena konstantou 100, pokud se liší o nanejvýše o 15, tak je vynásobena hodnotou 20. V případě objemu motoru je první hranicí maximální rozdíl o 0.1 l a druhou hranicí 0.25 l. U počtu kilometrů je první hranicí 15 000 km a druhou 37 500 km. I když se tyto konstanty mohou zdát velké, tak je třeba mít na paměti, že výsledků hledání, které se neshodují třeba ani v jednom z parametrů je obvykle naprostá většina a postupné testování těchto konstant ukázal tyto hodnoty jako vyhovující.

Nyní když má každá hodnota určenou svou váhu, tak se provede mezi těmito cenami vážený průměr. Výsledná hodnota udává odhadovanou cenu vozu.

7 Testování

Testování aplikace je nedílnou součástí vývoje každé aplikace. Jelikož má tato aplikace jako zdroj dat databázi, jež je výsledkem bakalářské práce kolegy Kováře Tadeáše, tak jsme při vývoji museli poměrně úzce spolupracovat. Navíc díky faktu, že jsme každý měli zvolený jiný programovací jazyk pro aplikace nebylo ani možné vytvořit si společné rozhraní pro přístup k této databázi. Data jsme si tedy při vývoji tedy předávali postupně, tak, jak jsme oba na práci postupovali. Obvykle to probíhalo tak, že když se kolega dostal za nějaký bod ve své práci, kdy provedl stahování, tak mi tuto databázi vyexportoval, předal, a já jsem následně databázi naimportoval. Samozřejmě, že tento způsob předávání dat není zcela ideální, avšak postačil pro účely testování, a především umožnil vývoj aplikace, jelikož testování aplikace bylo nedílnou součástí jejího vývoje. Testování lze rozdělit na následující tři etapy:

- Testování během vývoje aplikace
- Testování na vhodně zvoleném vzorku dat
- Testování v reálném provozu

7.1 Testování během vývoje aplikace

Toto testování tvořilo nedílnou součást vývoje aplikace, jelikož v případě, kdy by nebyly dostupné žádné testovací data, nebylo by možné ani pokračovat ve vývoji. První vzorek dat se kterým se vyvíjelo obsahoval pouze několik tisíc záznamů, které nebyly často ani kompletní. Nic z těchto faktorů nijak výrazně nepozastavilo vývoj aplikace. Jediným výraznějším problémem bylo u druhého vzorku dat špatné uložení diakritiky, což způsobovalo jisté problémy při testování. Následující vzorek dat, který byl používán při dokončování aplikace již tyto problémy neměl.

7.2 Testování na vhodně zvoleném vzorku dat

Toto testování bylo prováděno při dokončování a po dokončení implementace základní funkcionality aplikace, tedy v moment, kdy byly zprovozněny všechny služby. Vzorek dat, který pro toto testování byl použit již neobsahoval chybu s diakritikou, vyskytly se pouze drobné potíže s importem dat do databáze, které však byly poměrně jednoduše řešitelné. Toto testování pomohlo poodhalit několik neošetřených chyb při práci s daty, avšak služby pro výpočet ceny a pro zobrazování vývoje hodnoty vozu byly důkladněji testovány až po obdržení kompletního vzorku dat, který už simuloval reálná data.

7.3 Testování v reálném provozu

Testování v reálném provozu bylo prováděno až po dokončení implementace všech služeb. Jelikož zde přibyly data z dalšího inzerčního serveru, která byly zpracovávány odlišným způsobem nastalo zde několik nekonzistencí a jelikož byl tento vzorek dat dodán poměrně pozdě před dokončením práce, tak nebylo provedeno testování tak důkladné jako u předchozího vzorku dat. Je tedy možné, že některé nekonzistence přetrvávají, nejedná se však o chyby, které by zapříčinily konec chodu celé služby, nebo podobné zásadní problémy.

Více nad těmito daty byly testovány služby pro odhad hodnoty vozu a pro zobrazení vývoje hodnoty na základě různých parametrů.

7.4 Dosažené výsledky testování

Výsledné testování všech služeb poskytovalo přesvědčivé výsledky, kdy především díky velkému množství záznamů v databázi dochází k dosti přesnému výpočtu skutečné ceny vozu. Samozřejmě, že cena, která je vypočítána na základě reálných dat se často liší oproti ceně, která je získána na základě znaleckého posudku. Jedním z důvodů může být i to, že poslední aktualizace tohoto znaleckého posudku pochází z roku 2005 a za tuto dobu se již trend ve vývoji cen vozu zajisté dost změnil.

Uspokojivé výsledky přineslo i testování aplikace pro zobrazení vývoje hodnoty vozu, kdy data zobrazené grafem většinou odpovídají skutečnému vývoji ceny. Téměř každý výsledný graf ukazuje klesající tendenci, tak jak je očekáváno. Hodnoty tohoto grafu jsou navíc proloženy spojnicí trendů, jde tedy vidět i předpokládaný vývoj do budoucna.

Jak již bylo zmíněno, tak služby, které umožňují vyhledávání obsahují drobné nekonzistence způsobené právě nedostatkem času na testování. Ve většina případů použití však tyto služby fungují, tak jak je od nich očekáváno.

8 Závěr

Cílem této práce bylo vytvoření webové aplikace, která bude uživateli nabízet primárně možnost vyhledávání vozů napříč nabídkou na různých inzerčních serverech. Na základě velkého potenciálu databáze s ojedinělou velikostí v rámci České Republiky, kterou tato služba používá, poté vyplývají další služby, kterými jsou oceňování vozů a také zobrazení vývoje hodnoty vozů.

Požadavky na aplikaci byly již od začátku dobře specifikovány. Ve výběru požadovaných technologií aplikace byla autorovi svěřena poměrně velká volnost, nicméně požadavky na funkčnost aplikace, tedy jednotlivé služby, byly specifikovány přesněji. Po stanovení požadavků byla provedena analýza těchto požadavků a zároveň analýza existujících služeb, které jsou stejné, či podobné službám v aplikaci. Na základě těchto požadavků byl následně vytvořen návrh aplikace, podle nějž následně probíhala implementace. Poslední etapou vývoje této aplikace bylo testování.

Potenciál této aplikace leží právě ve velikosti databáze vozů a v tom, že sdružuje služby, které jinak uživatelé musí hledat na různých místech. Navíc díky velikosti databáze jsou tyto služby provozovány s výjimečnou přesností. Některé ze služeb poskytovaných aplikací dokonce v současnosti nemají obdobu.

Tato práce navazuje na práci Tadeáše Kováře, jehož bakalářská práce se zabývá právě získáváním dat z inzerčních serverů. Do budoucna by tyto aplikace měla být spuštěny v rámci VUT portálu a rozhodně nabízejí prostor na další rozvoj.

Nesporným přínosem pro mne jako autora bylo lepší seznámení s celým procesem vývoje webové aplikace. Také jsem se toho mnoho naučil v oblasti webových technologií a podstatně si rozšířil své znalosti o jazyku Java. Vyzkoušel a naučil jsem se pracovat s částmi Spring frameworku, který je dneska při vývoji webových aplikací v Javě opravdu praktický. Zajímavou a novou věcí byla také práce s grafy v Javascriptu a v neposlední řadě jsem se toho mnoho dozvěděl o technikách naceňování automobilů a celkově o naceňování movitého majetku.

Literatura

[1] SCHAFER, Steven M. HTML, XHTML a CSS: bible [pro tvorbu WWW stránek] : 4. vydání. Praha: Grada, 2009. Průvodce (Grada). ISBN 978-80-247-2850-6.

[2] DOMES, Martin. 333 tipů a triků pro CSS. 2., aktualiz. vyd. Brno: Computer Press, 2011. ISBN 978-80-251-3366-8.

[3] ZAKAS, Nicholas C. JavaScript pro webové vývojáře. Brno: Computer Press, 2009. Programujeme profesionálně. ISBN 978-80-251-2509-0.

[4] CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery: [kompletní průvodce vývojáře]. Brno: Computer Press, 2013. Mistrovství. ISBN 978-80-251-4103-8. [5]Java

[6] DESSÌ, Massimiliano. Spring 2.5 aspect-oriented programming: create dynamic, feature-rich, and robust enterprise applications using the Spring framework. Birmingham, U.K.: Packt Pub., 2009.

[7] Apache Tomcat 8: Version 8.5.15, May 5 2017. Apache Tomcat 8 [online]. [cit. 2017-05-18]. Dostupné z: http://tomcat.apache.org/tomcat-8.5-doc/index.html

[8] BURD, Barry A. JSP: JavaServer Pages: podrobný průvodce. Praha: Computer Press, 2003. ISBN 80-7226-804-X.

[9] GILMORE, W. J. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[10] ZAKAS, Nicholas C., Jeremy MCPEAK a Joe FAWCETT. Ajax profesionálně. Brno: Zoner Press, 2007. Encyklopedie webdesignera. ISBN 978-80-86815-77-0.

[11] Spring MVC: diagram. In: Tutorialspoint [online]. [cit. 2017-05-18]. Dostupné z: https://www.tutorialspoint.com/spring/images/spring_dispatcherservlet.png

[12] KREJČÍŘ, Pavel a Albert BRADÁČ. Znalecký standard č. I/2005: oceňování motorových vozidel. Brno: Akademické nakladatelství CERM, 2004. ISBN 80-7204-370-6.

[13] Kledus, R., Oceňování movitého majetku, VUT v Brně 2014, ISBN: 978-80-214-5040-0 (cs) - druhé, aktualizované vydání

[14] BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. Příklady modelů analýzy a návrhu aplikace v UML. Praha: Oeconomica, 2013. ISBN 978-80-245-1922-7.

Seznam příloh

Příloha A: Obsah CD

Příloha A: Obsah CD

CD obsahuje:

1. kompletní diagram tříd – soubor DiagramTrid.png
2. návod na instalaci aplikace v souboru installation.txt
3. databázi vozů – skript vozidla.sql, který je zabalený v balíku vozidla.zip
4. bakalářskou práci ve formátech PDF a DOCX.
5. samotnou aplikaci zabalenou v balíku application.zip, kterou je následně možné zprovoznit dle instalačního návodu