FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Automatizovaný metavyhledávač na internetových obchodech
Student:	Bc. Anna Vitmanová
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Obor / specializace:	Manažerská informatika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Se stále rostoucím počtem internetových obchodů je čím dál těžší hledat zboží od více prodejců. Vytvoření automatického metavyhledávače by mnoha obchodníkům ušetřilo čas i peníze při skupování zboží, a tak jim umožnilo svoje prostředky věnovat do rozšiřování svých služeb.

1) Analyzujte existující internetové obchody a metavyhledávače obsahu. Zaměřte se na možnosti strojového čtení obsahu a notifikací o dostupném zboží.
2) Analyzujte potřeby klienta co se týče vlastního metavyhledávače.
3) Navrhněte a naimplementujte prototyp vlastního metavyhledávače, který umožňuje jak jednoduché přidání sledovaných klíčových slov, tak přidání nových zdrojů obsahu. Metavyhledávač by měl ze začátku podporovat alespoň zdroje sBazar a Bazoš.
4) Naimplementujte automatické testy pro potvrzení korektního vyhledávání.
5) Proveďte ekonomicko-manažerské shrnutí projektu a porovnejte výhody a nevýhody vašeho přístupu se zaměřením na časové výhody.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 11. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Automatizovaný metavyhledávač na internetových obchodech

Bc. Anna Vitmanová

Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Nováček

3. května 2022

Poděkování

Děkuji všem blízkým za podporu, kterou mi poskytli nejenom při psaní diplomové práce, ale i během celého magisterského a bakalářského studia. Speciální díky potom patří mému vedoucímu práce, panu Ing. Tomáši Nováčkovi, a to za vstřícnost, trpělivost, ochotu a především všechnu pomoc, kterou mi během tvorby práce poskytl. Tomáši, tímto Ti tedy za vše ze srdce děkuji.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 3. května 2022 . . . . . .

Odkaz na tuto práci

Vitmanová, Anna. *Automatizovaný metavyhledávač na internetových obchodech.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Cílem této diplomové práce je implementace aplikace pro snadnější vyhledávání na internetových obchodech se zaměřením na některé internetové bazary. Hlavní myšlenkou je automatizace vyhledávání na více internetových bazarech najednou. Součástí práce je i celý proces vývoje software od analýzy současného řešení, přes sběr požadavků, návrh a implementaci, po nasazení a testování.

Klíčová slova webová aplikace, metavyhledávač, web scraping, MERN Stack

Abstract

The aim of this thesis is to implement an application for easier search on online stores with a focus on some online second-hand markets. The main idea is to automate searches on multiple online second-hand markets at once. Part of the work is the whole process of software development from the analysis of the current solution, through the collection of requirements, design and implementation, to deployment and testing.

Keywords web application, metasearch engine, web scraping, MERN Stack

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Internetové bazary se v posledních letech staly oblíbeným místem, kde se dennodenně uskutečňuje spousta obchodů. Jedná se o jednoduchý způsob, jak prodat či nakoupit použité nebo zánovní zboží. Těchto bazarů je ale na internetu mnoho. Pokud chce kupující najít to nejlepší zboží, může se doba hledání nepříjemně prodlužovat, protože neexistuje žádný systém, který by hledání na bazarech systematizoval.

Tato diplomová práce řeší primárně problém neexistence jednotného vyhledávání pro české internetové bazary. Téma práce vzniklo na základě specifických požadavků zadavatele, kterým je provozovatel opravny elektrotechniky. Zadavatel totiž tráví mnoho času vyhledáváním součástek či nefunkčních zařízení na bazarových serverech. Tento čas by zadavatel ale mohl využít i jinak, proto je potřeba tato vyhledávání automatizovat.

Hlavní myšlenkou je tedy implementace automatizovaného metavyhledávače pro více internetových bazarů s možným rozšířením do budoucna. Metavyhledávač funguje na principu rozesílání uživatelova dotazu do různých systémů, které nabízí svůj vlastní vyhledávač. Výsledky jsou potom sdružovány právě do výsledků metavyhledávače.

Kapitola 1

Cíl práce

Cílem této diplomové práce je implementace aplikace pro snadnější vyhledávání na internetových obchodech se zaměřením na některé internetové bazary. Hlavní myšlenkou je automatizace vyhledávání na více internetových bazarech najednou za účelem zefektivnění práce zadavatele. Hlavní funkcí celého systému je tedy konfigurace specifického vyhledávání, které jednou za určitý čas prochází určité servery a zobrazuje výsledky v aplikaci, přičemž uživatel by měl být na nově nalezené výsledky upozorněn.

Zadavatel totiž tráví vyhledáváním na internetových bazarech spoustu času, který by mohl být využit i jiným způsobem. Součástí dosažení cíle je i analýza současného řešení zadavatele spolu s dostupnými řešeními. Následuje sběr a analýza požadavků zadavatele, na jejichž základě bude postaven návrh a následná implementace celé aplikace.

Kapitola 2

Analýza

Tato kapitola se bude zabývat analýzou zadavatele a jeho současného řešení, které by mělo být co nejvíce automatizováno. Poté se zaměřím na prozkoumání dostupných řešení, která ale nejsou dle slov zadavatele dostatečná. Dále se tedy společně se zadavatelem zaměříme na jeho cíle a požadavky, které by měla výsledná aplikace naplňovat.

2.1 Analýza zadavatele a současného řešení

Zadavatelem je provozovatel opravny elektrotechniky, a to primárně Apple zařízení. Jedná se hlavně o chytré telefony (iPhone), chytré hodinky (Apple Watch), tablety (iPad) a notebooky (MacBook). Některé součástky nutné pro opravu nemůžou být zakoupeny v originální kvalitě, jelikož Apple nabízí opravy pouze u svých autorizovaných servisů[1].

Bazarový trh je i z tohoto důvodu těmito produkty poměrně nabitý, protože zařízení poškozená, opravená či z druhé ruky neztrácejí na hodnotě tolik jako zařízení jiných značek. Zadavatel si poškozené zařízení buď opraví sám za pomoci neoriginálních součástek, nebo využije nepoškozenou součástku na opravu jiného zařízení.

Zadavatel tedy musí několikrát denně procházet stránky různých internetových bazarů za účelem hledání elektrotechniky, která by byla vhodná na nebo pro opravu. Toto hledání zadavatelovi zabere hodně času, který by mohl věnovat samotným opravám, tudíž se pro něj jedná o prodělečnou činnost. Tuto činnost by ale bylo možné automatizovat.

Zadavatel primárně nefunkční či poškozené zboží vyhledává na webových stránkách `https://www.bazos.cz/` (dále Bazoš) a `https://www.sbazar.cz/` (dále SBazar). S těmito webovými stránkami budu dále pracovat jak v analýze podobných řešení, tak v implementační části aplikace.

Velký problém zadavatel vidí v tom, že na každém webu musí vyhledávat pomocí několika různých výrazů, aby našel všechny požadované výsledky. Ne-

existuje možnost nakonfigurovat si vyhledávání, které by zahrnovalo všechny možnosti, které zadavatel chce nalézt. Příkladem je vyhledávání rozbitých MacBooků. Aby zadavatel vyhledal všechny možné způsoby poškození notebooku, musí vyhledat výrazy *macbook rozbitý*, *macbook politý*, *macbook utopený*, *macbook nefunkční* a tak bych mohla pokračovat dále.

Na těchto serverech lidé často nabízí i opravu či výkup rozbitého zařízení. Tyto inzeráty zadavatele zdržují, protože jich je opravdu hodně. Navíc se často objevují v tzv. *topovaných* inzerátech. Inzerenti si tímto způsobem platí za možnost zobrazení jejich inzerátů na prvních příčkách vyhledávání. Zadavatel tedy na první pohled nevidí, které inzeráty přibyly jako poslední, a inzeráty, které již viděl, musí prohlížet znovu stále dokola.

Zároveň zadavatel nikde nemá přehled o tom, na jaké inzeráty už reagoval. Vše řeší pouze podle paměti, podle odeslaných e-mailových zpráv či podle přehledu telefonních hovorů.

2.2 Analýza dostupných řešení

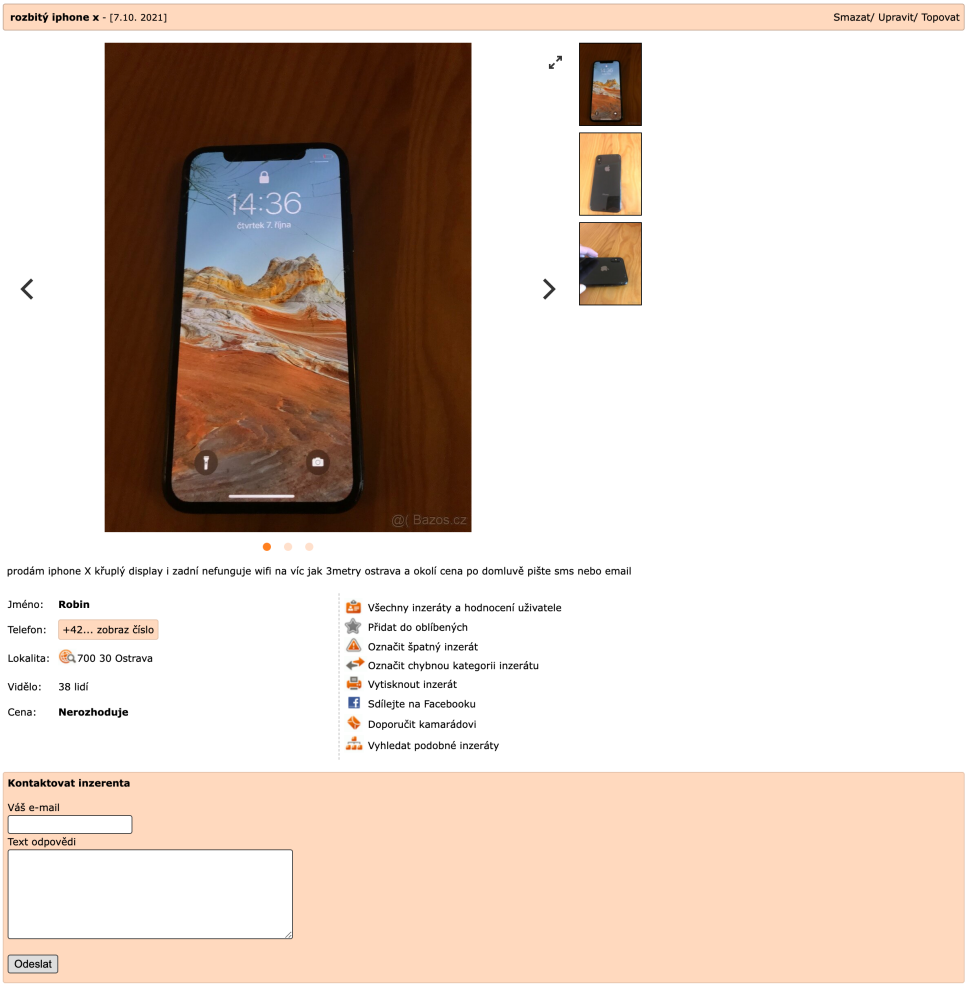
Pro analýzu dostupných řešení se vždy nejprve zaměřím na platformu jako takovou a poté na nějakou formu hlídacího psa, pokud jí server nabízí. Hlídací pes je služba, která většinou slouží k ohlídání dostupnosti konkrétního zboží a která může být nastavena uživatelem. Většinou je tato služba nabízena ve formě e-mailových upozornění.

2.2.1 Bazoš

Bazoš je jedním z populárních českých on-line bazarů. Dle statistik, které jsou dostupné na stránkách, se každý den přidá kolem šedesáti tisíc nových inzerátů, v kategorii „PC“ to je potom kolem dvou tisíc inzerátů za den a v kategorii „Mobily“ kolem tisíce a půl. Tyto údaje jsou platné platné k dubnu r. 2022.

Inzeráty na Bazoši obsahují následující informace (viz obr. 2.1):

- název,
- datum přidání,
- fotografie předmětu,
- popis,
- jméno inzerenta,
- telefonní číslo inzerenta,
- lokalitu a
- cenu.



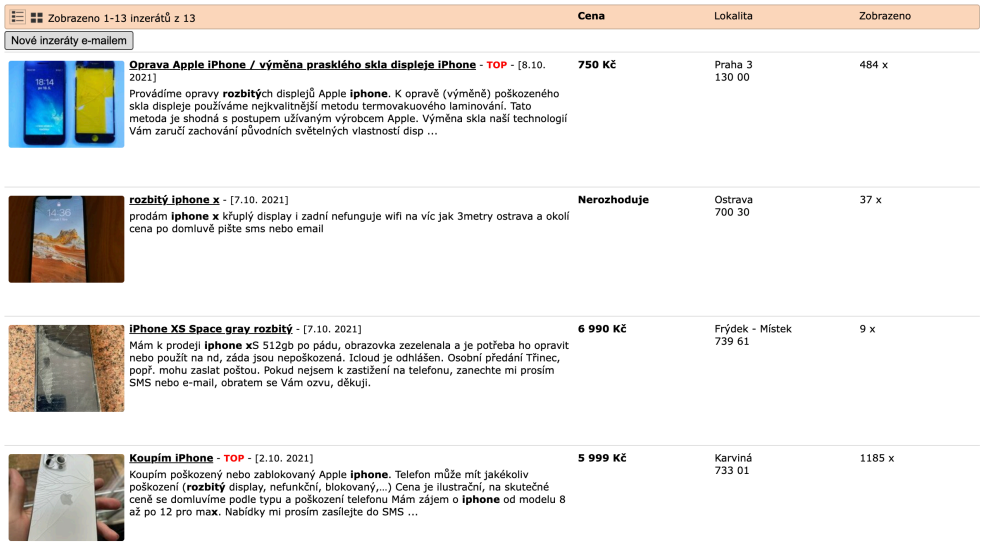
Obrázek 2.1: Detail inzerátu na Bazoši

Telefonní číslo inzerenta se zobrazí až po ověření uživatele. Uživatel musí nejprve zadat své vlastní telefonní číslo, na které se následně odešle SMS s ověřovacím kódem. Vložením tohoto kódu je poté uživatel ověřen a číslo zobrazeno.

Dále je zde možnost kontaktování inzerenta bez ověření telefonního čísla, a to pouze pokud má inzerent vyplněn e-mail (u inzerátu se tato informace nezobrazuje). Pod inzerátem lze potom vidět formulář pro odeslání zprávy s uživatelovou e-mailovou adresou.

2.2.1.1 Vyhledávání

Vyhledávání na Bazoši podporuje mimo klasický výčet výrazů i zpřesňující operátory. Pokud před výraz vložíme znak minus, inzeráty obsahující toto



Obrázek 2.2: Výsledky vyhledávání na Bazoši

slovo potom ve výsledcích vyhledávání nebudou figurovat. Pokud chceme vyhledat přesnou frázi, vložíme ji do uvozovek. Celý dotaz potom může vypadat například takto: *„iphone x“ display -praskly* (vyhledáváme inzeráty na display pro *iPhone X*, které neobsahují slovo prasklý).

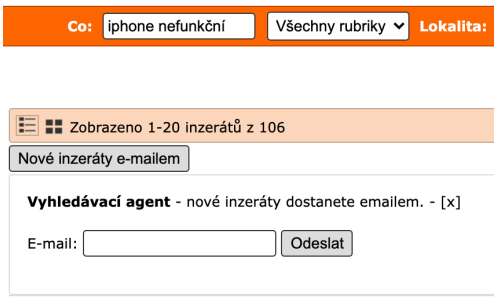
Jednotlivé výrazy jsou potom vyhledávány jak v názvech inzerátů, tak v jejich popisech. Můžeme říct, že se tedy jedná o formu *fulltextového* vyhledávání[2]. Zároveň se vyhledávání striktně nedrží zadaného výrazu, ale zahrnuje i inzeráty obsahující daný výraz v podřetězcích slov.

To má jak své výhody, tak nevýhody. Výhodou je třeba použití výrazu *praskl*, který bude vyhledávat inzeráty obsahující slova *praskl*, *prasklý*, *prasklé*, *prasklá*, *prasklina* atd. Nevýhodou je například vyhledávání výrazu *pro* (MacBook PRO), kdy výsledky budou obsahovat i inzeráty se slovy *prodávám*, *oproti*, *procesor* atd. Tato nevýhoda by se dala obejít využitím uvozovek (*„macbook pro“*), což se ale nedá použít pro všechny případy.

Vyhledávat se dá i s použitím různých parametrů, jako jsou cena či lokalita. Pokud se vyhledává podle lokality, berou se v potaz jen inzeráty v určené maximální vzdálenosti. Pokud chceme více specifikovat cenu, můžeme ji ohraničit jak shora, tak zdola.

Co se týče diakritických znamének, vyhledávání na ně nebere zřetel. Vyhledávač je tak schopen najít inzeráty jak v případě, kdy se znaménko v dotazu použije, ale v inzerátu není, tak v případě, že se v dotazu nepoužije, ale v inzerátu je obsaženo (výraz *prasklý* najde inzerát se slovem *praskly* a naopak).

Na obrázku 2.2 můžeme vidět, jak vypadají výsledky vyhledávání pro dotaz *iphone x rozbitý*.



Obrázek 2.3: Vyhledávací agent na Bazoši

2.2.1.2 Hlídací pes

Na Bazoši také můžeme nalézt možnost zavedení vyhledávacího agenta (viz obr. 2.3). Vyhledávací agent funguje na principu e-mailových zpráv s upozorněním na nově přidané inzeráty. Tyto zprávy chodí jednou za osm hodin a jdou nastavit pouze na jedno vyhledávání. Tudíž pokud chce člověk vyhledávat více předmětů, chodí e-mailů více v jeden čas. Zároveň nikde nelze dohledat, jaké hlídací agenty máme nastavené, protože Bazoš nepodporuje žádnou možnost registrace uživatele. To znamená, že zadavatel nemá žádný přehled, protože v e-mailových zprávách se nelze moc dobře orientovat. Zároveň vyhledávání neprobíhá tak často, jak by si zadavatel představoval.

2.2.2 SBazar

SBazar je dalším z českých on-line bazarů. Funguje jako přidružená služba k internetovému portálu Seznam.cz[1], který je druhým nejpoužívanějším vyhledávačem v České republice[3]. Ze statistik, které jsou dostupné na SBazaru, můžeme vyčíst, že je na těchto stránkách zveřejněno kolem dvou milionů nabídek, z toho na osmnáct tisíc inzerátů v sekci *Mobil bazar* a na sedmnáct tisíc v sekci *Počítače* (údaje platné k dubnu r. 2022).

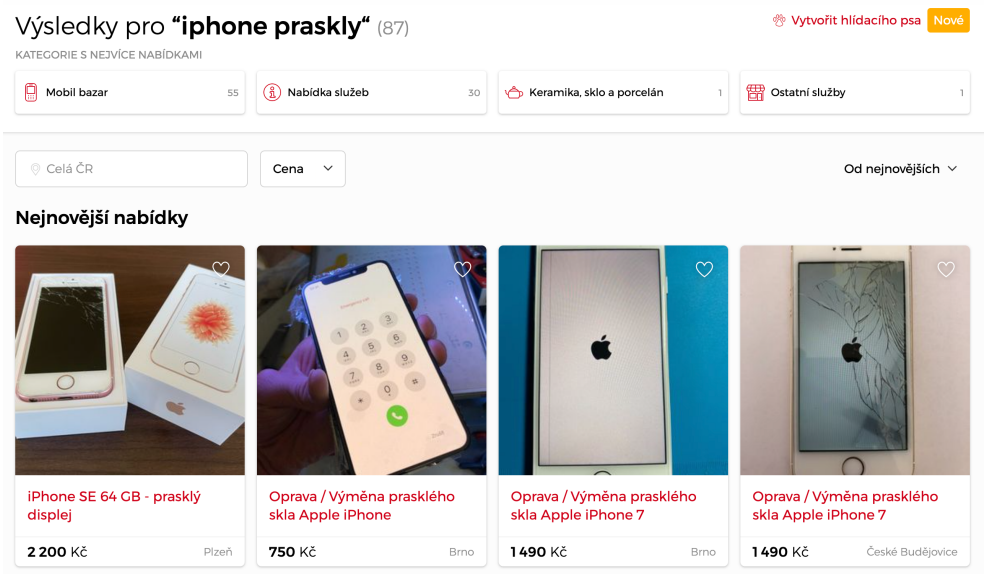
Inzeráty na SBazaru obsahují stejné (nebo velmi podobné) informace jako inzeráty na Bazoši (viz obr. 2.4). Nabízí i podobné funkce, co se týče odesílání zpráv inzerentovi s tím rozdílem, že není potřeba ověřovat uživatelovo telefonní číslo pro zobrazení inzerentova čísla.

2.2.2.1 Vyhledávání

Vyhledávání na SBazaru nepodporuje tolik funkcí jako vyhledávání na Bazoši. Neexistuje (nebo minimálně jsem to nebyla schopna dohledat) zde žádná funkce vložení speciálních znaků k hledaným výrazům, po kterých by byly výsledky vyhledávání zpřesněny.

[1] `https://www.seznam.cz/`

Obrázek 2.4: Detail inzerátu na SBazaru



Obrázek 2.5: Vyhledávání na SBazaru

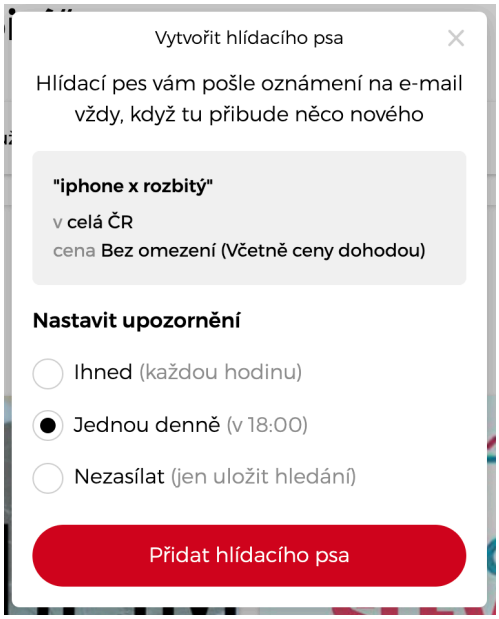
Výrazy z dotazu jsou potom vyhledávány jak v názvech, tak v popisech inzerátů. Nicméně ale nefunguje možnost vyhledávání podle podřetězců v jednotlivých slovech. Proto výraz, který by šel na Bazoši zahrnout pouze jedním slovem *praskl*, musí být vyhledán vždy zvlášť pro slova *prasklý*, *prasklá* atd.

Úplně stejně jako na Bazoši zde existuje možnost přidat do vyhledávání navíc parametr lokalita nebo cena. Stejně tak je tomu v případě použití diakritiky. Příklad výsledků vyhledávání na dotaz *iphone praskly* můžeme vidět na obrázku 2.5.

2.2.2.2 Hlídací pes

SBazar nabízí možnost nastavení hlídacího psa (viz 2.6). V porovnání s vyhledávacím agentem Bazoše se jedná o propracovanější funkcionalitu, a to už jen díky tomu, že se nastavuje pro právě přihlášeného uživatele a nemusí se zadávat e-mailová adresa. Také nabízí možnost nastavení upozornění, a to jak nastavení času, tedy kolikrát za den má přijít upozornění na e-mail, či jestli se má hlídací pes pouze uložit k účtu a hlídat bez žádných dalších upozornění.

Hlídací pes funguje dobře, nicméně je potřeba nastavit mnoho hlídacích psů pro skoro ten a samý výraz (to vyplývá ze způsobu, jakým SBazar vyhledává), což také není pro zadavatele přehledné.



Obrázek 2.6: Hlídací pes na SBazaru

2.3 Analýza požadavků

Cílem této kapitoly je analýza požadavků zadavatele a následné vymezení funkcí systému. Některé požadavky jsou nutné, ale některé jsou pro zadavatele pouze bonusem navíc či možným rozšířením do budoucna, pokud to bude potřebné.

Požadavky jsou obecně dvojího typu[4]:

Funkční požadavky (F) vyjadřují chování systému a jeho funkce. Typicky se systém od systému liší právě v těchto požadavcích.

Nefunkční požadavky (N) jsou všechny ostatní požadavky a typicky představují očekávané vlastnosti softwaru. Tyto požadavky mají potom jednotlivé systémy podobné. Mluvím například o snadném použití systému, přehlednosti, systém by měl být spolehlivý, dobře zabezpečený, měl by mít dobrý výkon a měl by být udržitelný. Dále se také například jedná o formu dodání software (mobilní, desktopová či webová aplikace).

2.3.1 Funkční požadavky

Funkční požadavky bylo jednoduché získat, jelikož měl zadavatel jasnou představu o tom, jak by měla aplikace fungovat. U některých požadavků jsme potom vedli další diskuzi, která vyústila v pár dalších požadavků, které zadavatel vnímá spíše jako benefity navíc.

2.3.1.1 F1 – Přihlášení

Zadavatel požaduje přihlášení do aplikace pod svým e-mailem a heslem. Do budoucna by bylo dobré, kdyby bylo možné přihlášení přes Facebook[2] či Google[3] účet, ale není to pro zadavatele podstatné.

2.3.1.2 F2 – Registrace nového uživatele

Systém bude sice primárně využívat pouze jeden uživatel, ale i tak je registrace nového uživatele pro zadavatele důležitá. Může se totiž stát, že systém nakonec bude využívat více lidí, takže více vytvořených uživatelů by bylo nutné pro přehlednost.

2.3.1.3 F3 – Přidání nové vyhledávací konfigurace

Vyhledávání by se neobešlo bez přidání vyhledávacích konfigurací. Jedna vyhledávací konfigurace se rovná jednomu specificky nastavenému vyhledávání pomocí určitých parametrů. V aplikaci by měl tedy existovat formulář pro přidání konfigurace, který by měl obsahovat:

- název konfigurace,
- vyhledávací *query*[4] (viz níže),
- možnost zvolení webových stránek, na kterých má vyhledávání probíhat (Bazoš, SBazar, případně Facebook Marketplace).

Vyhledávací *query* je seznam výrazů, které má vyhledávání zahrnovat. Výrazy by se měly dělit na dvě kategorie – povinné a volitelné. Povinné výrazy budou ve výsledcích obsaženy vždy. Volitelné výrazy se ještě člení do skupin. S povinnými výrazy musí být z každé volitelné skupiny obsažen jeden výraz. Zde je příklad, jak by vyhledávací *query* měla fungovat:

Povinné výrazy: apple, watch

Volitelné výrazy:

- rozbitý, prasklý
- display, displej

Výsledek:

- apple watch rozbitý display

[2] `https://www.facebook.com/`

[3] `https://www.google.com/`

[4] z anglického výrazu *search query* (jedná se o frázi, jež se předává vyhledávačům, které na jejím základě vrátí konkrétní výsledky[5])

- apple watch rozbitý displej
- apple watch prasklý display
- apple watch prasklý displej

2.3.1.4 F4 – Zobrazení přehledu vyhledávacích konfigurací

Po přihlášení do systému by na domovské obrazovce měl být vidět přehled již nastavených vyhledávacích konfigurací. Konfigurace od sebe budou jasně odlišené pomocí zobrazeného jména.

Zadavatel nepočítá s tím, že by měl konfigurací najednou mnoho, proto není potřeba řešit žádné řazení či filtrace. To bude ale možná potřeba do budoucna změnit a zapracovat nějaké řešení pro lepší přehlednost.

2.3.1.5 F5 – Správa vyhledávacích konfigurací

Na domovské stránce by v přehledu vyhledávacích konfigurací mělo být možné spravovat jednotlivé konfigurace. Hlavními body jsou:

- aktivace či deaktivace konfigurace (zda je aktuálně vyhledávání aktivní či nikoli),
- editace konfigurace (přechod na stránku s detailem konfigurace, kde je možná následná úprava),
- smazání.

2.3.1.6 F6 – Zobrazení výsledků vyhledávací konfigurace

Systém by měl samozřejmě nabízet možnost zobrazení výsledků dané vyhledávací konfigurace. Výsledek by měl obsahovat název inzerátu, detailní popis inzerovaného předmětu, datum přidání, cenu, případný kontakt na inzerenta a odkaz na samotný inzerát.

Tyto výsledky by měly mít možnost řazení, a to podle data přidání inzerátu nebo podle ceny. Výchozí řazení by mělo být od nejnovějších po nejstarší výsledky. Dále by měla být možnost filtrovat inzeráty podle stáří. Jako výchozí filtr se budou zobrazovat výsledky z posledních sedmi dní.

2.3.1.7 F7 – Správa výsledků vyhledávací konfigurace

Jednotlivé výsledky by potom uživatel měl mít možnost různě spravovat. Zde je ještě prostor pro budoucí změny podle toho, jak se bude zadavatelovi se systémem pracovat.

Zajímavé inzeráty, na které bude chtít zadavatel reagovat, bude možné přidat do oblíbených. Systém by měl podporovat možnost zobrazení pouze oblíbených inzerátů.

Inzeráty, které naopak neodpovídají uživatelovým požadavkům, by mělo být možné ignorovat, a dále v přehledu výsledků nezobrazovat. Zároveň ale zadavatel chce mít možnost takové inzeráty znovu zobrazit, kdyby si náhodou svůj předchozí čin rozmyslel. Proto je nutné, aby u výsledků byla možnost zobrazení i již ignorovaných inzerátů, které ale budou standardně skryty.

2.3.1.8 F8 – Pravidelné vyhledávání

Hlavní funkcí aplikace by mělo být automatické pravidelné vyhledávání aktivních konfigurací. Měl by tedy probíhat nějaký proces na pozadí, který by v nějakých pravidelných intervalech (například jednou za dvacet minut) spouštěl skript, který bude prohledávat pomocí jednotlivých konfigurací určené weby. Výsledky by se potom měly přidat do výsledků vyhledávání dané vyhledávací konfigurace.

Zároveň by měl systém kontrolovat, zda jsou inzeráty stále dostupné. Pokud ne, dále uživateli tyto inzeráty nezobrazovat. Zároveň by ale zadavatel chtěl mít přehled i o již vymazaných inzerátech, proto by se neměly inzeráty mazat z databáze, ale pouze označit jako nedostupné a v případě potřeby i ve výsledcích konfigurace přidat možnost na zobrazení již nedostupných inzerátů.

2.3.1.9 F9 – Upozornění na nově vyhledané zboží

Systém by měl umět odesílat upozornění s nově vyhledanými požadavky. Možnost zasílání upozornění by se měla dát měnit zvlášť u jednotlivých konfigurací, protože se může stát, že pro některou z vyhledávacích konfigurací uživatel nebude chtít dostávat upozornění, ale pro jinou konfiguraci to naopak chtít bude.

Upozornění by měla fungovat formou e-mailových zpráv na e-mailovou adresu uživatele, který si nastavil danou vyhledávací konfiguraci. Pokud bude nalezen nějaký nový inzerát odpovídající parametrům vyhledávání, měla by uživateli co nejdříve dorazit upozorňující zpráva.

Byla by samozřejmě vítána i jiná forma upozornění, například upozornění na mobilní telefon (viz sekce 2.3.2.2).

2.3.1.10 F10 – Konfigurace automatické zprávy pro inzerenty

Jedná se o možnost konfigurace zprávy, která se bude posílat inzerentům, pokud inzerát obsahuje email nebo telefonní číslo. V konfiguraci vyhledávání by se mělo dát nastavit, zda se tato zpráva bude odesílat automaticky, nebo po manuálně (například po stisku tlačítka u daného výsledku vyhledávání).

Tato možnost je trochu komplikovaná, protože by si každý uživatel musel ke svému účtu nakonfigurovat i nastavení svého vlastního SMTP (zkratka z anglického *Simple Mail Transfer Protocol*) serveru, aby odeslané e-maily byly zobrazeny v uživatelově e-mailové schránce. Nicméně pokud bude sys-

tém využívat pouze zadavatel, stačí mít jako jednotný SMTP server jak pro odesílání upozornění, tak pro odesílání zpráv jednotlivým inzerentům.

Tato funkcionalita by byla pro zadavatele přínosem, avšak není podstatná pro fungování aplikace, takže není nutné ji momentálně implementovat.

2.3.1.11 F11 – Real-time vyhledávání

Aplikace by mohla nabízet nejen možnost automatického vyhledávání, ale i *real-time* vyhledávání a zobrazení výsledků z požadovaných webů. Tato funkce je ale nicméně poměrně náročná na implementaci a zadavatelův požadavek je hlavně pravidelné automatické vyhledávání, takže je tato funkce spíše další možné rozšíření do budoucna.

Z části sice už bude implementace hotová, protože základ tohoto vyhledávání by byl stejný jako u toho automatického. Bylo by potřeba udělat několik změn na pozadí a také by se muselo lišit uživatelské rozhraní zobrazení výsledků. Kvůli množství nalezených výsledků by se totiž v tomto případě mělo řešit například stránkování.

2.3.2 Nefunkční požadavky

Nefunkční požadavky také byly hned po konzultaci se zadavatelem jasné a nejsou ani z povahy funkčních požadavků žádným překvapením.

2.3.2.1 N1 – Webová aplikace

Systém by měl být dostupný přes internet ve formě webové aplikace. Zároveň samotný systém z důvodu vyhledávání na jednotlivých stránkách potřebuje mít přístup k internetu. Bylo by tedy potřeba systém nasadit například na VPS (zkratka z anglického *Virtual Private Server*) server.

2.3.2.2 N2 – Mobilní aplikace

Mobilní aplikace není nutným požadavkem, ale byla by pro zadavatele velkým přínosem. Hlavním důvodem pro tvorbu mobilní aplikace by byla upozornění na nově přidané inzeráty, což zadavatel momentálně požaduje řešit pomocí e-mailových upozornění. Není nutná, ale bylo by dobré při implementaci myslet na to, aby bylo do budoucna možné snadno napsat i mobilní aplikaci.

2.3.2.3 N3 – Snadná rozšiřitelnost do budoucna

Mimo Bazoš a SBazar existuje na internetu spousta dalších serverů nabízející stejné či podobné služby. Jedná se například o již zmíněný Facebook Market-

place[5] nebo třeba Aukro[6]. Ze zahraničních stránek to je třeba eBay[7].

Systém by měl tedy být připraven na přidávání dalších možností vyhledávání na jiných internetových bazarech. Zároveň by se měla brát v úvahu i možnost, že by systém mohl hlídat zboží na klasických e-shopech, nejen těch bazarových. Zde by se jednalo spíše o možnost hlídání dostupnosti jednoho konkrétního typu zboží.

2.3.2.4 N4 – Zabezpečení

Webová aplikace by měla být nějakým způsobem zabezpečená proti úniku dat. Zároveň by měla být chráněna data jednotlivých uživatelů pomocí šifrování jejich hesel.

Dále by bylo dobré mít u nasazené aplikace přidružený SSL[8] certifikát. SSL certifikát zaručí identifikaci a šifrovanou komunikaci mezi prohlížečem a aplikací, čímž se stává více zabezpečenou proti odposlouchávání[6].

2.3.2.5 N5 – Využití technologií MERN

Pro vývoj systému je požadavek využít technologie z tzv. MERN Stacku, které budou popsány v sekci 3.1. Důvodem pro je jejich využití je zadavatelova orientace právě v těchto technologiích, což by mu mohlo usnadnit případný budoucí vývoj.

[5] `https://www.facebook.com/marketplace/`
[6] `https://aukro.cz/`
[7] `https://www.ebay.com/`
[8] zkratka z anglického *Secure Sockets Layer*

Kapitola 3

Návrh

Kapitola se bude zabývat navržením logiky a vzhledu aplikace. Nejprve se zaměřím na použité technologie a jejich výhody i nevýhody, poté zmíním i technologie použité při vývoji.

Dále se budu věnovat návrhu *back-endu*, kdy určím, jak bude například to, jak bude vypadat struktura databáze, nebo jaké budou probíhat procesy v logice aplikace.

V neposlední řadě bude v několika krocích navržen i *front-end* aplikace, tedy uživatelské rozhraní.

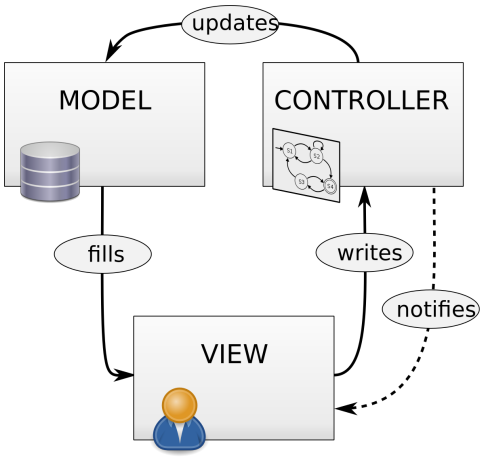
3.1 Technologie a architektura

Pro realizaci projektu jsem se rozhodla na základě požadavků klienta využít MERN Stack. Jmenovitě se jedná o technologie:

- MongoDB,
- Express(.js),
- React(.js),
- Node(.js).

Tyto technologie spolu dohromady perfektně fungují a jsou základem pro tradiční třívrstvou architekturu. Třívrstvá architektura je jeden z možných způsobů, jak strukturovat aplikaci. Hlavním principem je rozdělení aplikace do tří na sobě nezávislých komponent, neboli vrstev (viz obr. 3.1). Jedná se o prezenční (*view*), aplikační (*controller*) a databázovou (*model*) vrstvu[7].

Prezenční vrstva zajišťuje zobrazení informací uživateli a v tomto případě bude zajištěna Reactem. Aplikační vrstva se stará o logiku celé aplikace, o což se bude starat Express a Node. Databázová vrstva zpracovává a uchovává data, k čemuž bude sloužit MongoDB[8].



Obrázek 3.1: Třívrstvá architektura – diagram

Toto rozdělení má mnoho výhod, mezi kterými například jsou[9]:

- přehlednost,
- zjednodušuje se budoucí vývoj (například se může vyměnit typ databáze bez toho, aniž bychom hýbali s logikou aplikace),
- bezpečnost (klient nemá přímý přístup do databáze).

Nevýhody naopak můžou být:

- pomalejší komunikace, protože dotazy probíhají na dvou úrovních, nebo
- větší složitost celé aplikace, která může být u menších aplikací zbytečná.

3.1.1 Databázová vrstva – MongoDB

MongoDB je nerelační (nebo také NoSQL), dokumentově orientovaná databáze. Místo tabulek, na které jsme zvyklí v SQL databázích, se používají kolekce, místo řádků dokumenty (odtud dokumentová databáze) a místo sloupců pole.

Velkou výhodou je to, že se do sebe mohou dokumenty vnořovat, čímž nám místo několika propojených tabulek, jako by tomu bylo v SQL databázi, stačí pouze jediný dokument. Data jsou reprezentována v JSON formátu, díky čemuž je ukládání, manipulace a reprezentování dat v jednotlivých vrstvách aplikace velmi snadné[10].

Nevýhodou naopak může být limit na velikost dokumentu, větší potřeba paměti na uložení dat nebo nepodporování databázových transakcí[11].

3.1.2 Aplikační vrstva – Express a Node

Express je nejpopulárnější Node.js *framework* pro webové a mobilní aplikace. Jedná se o minimalistické, ale robustní řešení, které nabízí výkon, rozšiřitelnost a rychlost výsledné aplikace. Díky aktivní komunitě je spravován a stále vylepšován.

Slouží primárně pro manipulaci s HTTP[9] požadavky (GET, POST, DELETE atd.) na různých URL[10] cestách neboli routách. Také generuje odpovědi, které posílají informace zpět na *front-end*[12, 13, 14].

Node je JavaScriptový webový server. Jedná se o *open-source*[11] nástroj, který je multiplatformní a vhodný pro vývoj vysoce škálovatelných webových aplikací.

Typickým znakem Node.js je běh pouze v jednom vlákně (*single-thread*), z čehož plyne využívání principu tzv. asynchronního programování – jednotlivé požadavky nečekají na dokončení toho předchozího, čímž se maximalizuje potenciál výkonu. Pokud je ale potřeba počkat na dokončení, typicky třeba výsledek dotazu do databáze, jednoduše se před požadavek vloží operátor `await`, který vlákno zdrží na jednom místě[14, 16].

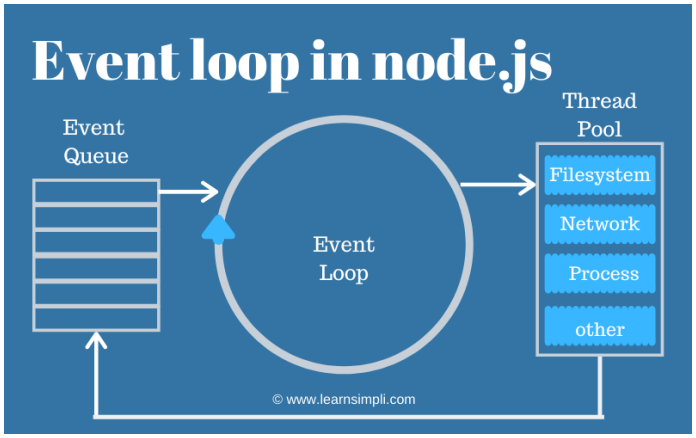
Architektura Node.js je založena na principu tzv. smyčky událostí (*Event Loop*, viz obr. 3.2), která zajišťuje neblokující I/O systém[12]. Je tak umožněno zpracování většího počtu požadavků současně. Požadavky jsou zařazovány do fronty, která je postupně smyčkou zpracovávána. Smyčka poté jednotlivé požadavky přiděluje odpovídajícím zdrojům. Dále smyčka také kontroluje, zda již nejsou nějaké požadavky zpracovány a vyhodnoceny. To funguje na principu tzv. *callbacku*. *Callback* je funkce, která se volá až poté, co doběhne daná úloha[17].

Pro uživatele nabízí spoustu benefitů, kterými jsou například[14, 16, 19]:

- výkon, který plyne z konceptu smyčky událostí a asynchronního programování, díky čemuž aplikace spotřebuje méně RAM[13] a dosáhne co možná největší škálovatelnosti,
- stále se rozvíjející jazyk JavaScript,
- NPM (Node Package Manager) umožňující snadný přístup k mnoha knihovnám a balíčkům,
- aktivní komunita, která Node.js stále vylepšuje.

Ačkoli při využití této technologie existuje mnoho výhod, můžeme nalézt i nevýhody, mezi které patří třeba:

[9] zkratka z anglického *Hypertext Transfer Protocol*
[10] zkratka z anglického *Uniform Resource Locator*
[11] označení pro programy, jejichž zdrojový kód byl poskytnut veřejnosti pro volné užití a další vývoj[15]
[12] zkratka z anglického *Input/Output*
[13] zkratka z anglického *Random Access Memory*



Obrázek 3.2: Node.js – smyčka událostí[18]

- nestabilní API[14], která se často mění a není zpětně kompatibilní,
- nevhodnost pro výpočetně náročné procesy, což je způsobeno nemožností využití více vláken (tyto procesy by potom blokovaly příchozí požadavky), a
- delší doba vývoje, protože si programátor musí skoro vše napsat od začátku.

3.1.3 Prezenční vrstva – React

React je JavaScriptová knihovna určená pro vývoj uživatelského rozhraní. Jedná se o deklarativní, efektivní a flexibilní řešení klientské části aplikace[20].

Hlavním stavebním prvkem jsou komponenty (*components*), což jsou znovupoužitelné HTML elementy. Skládáním komponent potom vzniká celé uživatelské rozhraní aplikace. Každá komponenta má svoje vlastnosti (*props*) a svůj stav (*state*). Vlastnosti vkládají komponentě nadřazené komponenty.

Stav si potom spravuje komponenta sama. Když se změní data, React aktualizuje a vykresluje pouze ty komponenty, které to potřebují. Tomuto přístupu se říká deklarativní (popisujeme finální uživatelského rozhraní pro každý stav). Díky tomu je chování komponent předvídatelnější a snadněji se celá aplikace ladí[21].

Mezi výhody řadíme hlavně svižnost celé aplikace, která je zaručena překreslováním jen potřebných komponent, množství knihoven, které můžeme využít při psaní aplikace, a dobrou dokumentaci společně s aktivní komunitou. Nevýhodou naopak může být nutnost využití nástrojů třetích stran či specifický syntax, který se liší od klasického JavaScriptu[22].

[14]zkratka z anglického Application Programming Interface, jedná se o rozhraní sloužící pro komunikaci se serverem

3.1.4 Web scraping

Server Bazoš podle nápovědy neposkytuje žádné API[23] a ani u SBazaru jsem nic podobného nedohledala. Alternativou pro vyhledávání zboží je tedy využití takzvaného *web scrapingu*.

Web scraping je automatizovaný proces shromažďování potřebného obsahu a dat z webové stránky. Pomocí robota se tyto stránky procházejí a pomocí jednotlivých HTML prvků a jejich názvů a atributů, které se předem získají z manuálního průchodu webových stránek, se poté vytáhnou informace, které si dále sami ukládáme do vlastní databáze[24, 25].

Výhodou použití technologie *web scrapingu* je vlastní práce s daty, kdy si sami získáme přesně ty informace, které využijeme, přesně v tom formátu, který potřebujeme. Oproti tomu využití API lehce zaostává v tom smyslu, že o tom, jaké informace nám přijdou a v jakém formátu budou, rozhoduje druhá strana (v našem případě by to byl Bazoš nebo SBazar).

Nevýhodou použití *web scrapingu* je nutnost kontroly jednotlivých použitých HTML atributů a tagů, tedy zda druhá strana nezměnila strukturu svých stránek. Na druhou stranu se může stát, že i obsah vracený API se může změnit a nemusí odpovídat formátu, se kterým aplikace počítá.

Největší problém ale nastává v případě, že jsou stránky kvůli častým dotazům *web sraperu* zablokovány pro IP adresu, ze které tyto dotazy na stránku chodí. To může být ošetřeno přidáním nuceného zpoždění do kódu *scraperu*. Toho se dá dosáhnout například pravidelným použitím funkce `sleep()`, která bude „uspávat“ běh aplikace po náhodnou dobu v řádu desetin sekund, aby simulovala používání běžným uživatelem, tedy člověkem[26].

Existuje mnoho programovacích jazyků a jejich knihoven, které jsou používány právě pro *web scraping*[27]. Dle vlastní zkušenosti jsem se rozhodla využít jazyk Python a jeho knihovnu Beautiful Soup[15], která je určena právě na parsování[16] dat, konkrétně na XML a HTML dokumenty[29].

Základem je získání stránky z nějaké URL, ze které se pomocí knihovny Beautiful Soup získá strukturovaný obsah, který dále můžeme prohledávat pomocí HTML atributů jako jsou třídy a ID (viz kód níže).

```
import requests
from bs4 import BeautifulSoup

url = "https://example.com"
page = requests.get(url)
soup = BeautifulSoup(page.content, "html.parser")
result = soup.find(id="someId", class_="someClass")
```

[15] `https://pypi.org/project/beautifulsoup4/`
[16] proces získávání strukturovaných dat z formátovaného textu[28]

3.1.5 Nástroje pro vývoj

V této sekci stručně popíši jednotlivé technologie, které budou využívány pro vývoj aplikace:

- Git – pro správu verzí aplikace,
- vývojové prostředí (IDE[17]) WebStorm (JetBrains[18]) – pro vývoj MERN aplikace,
- vývojové prostředí PyCharm (taktéž JetBrains) – pro vývoj Python skriptů pro web scraping,
- API klient Insomnia[19] a
- databázový klient Robo 3T[20].

3.2 UML diagramy

UML (zkratka z anglického *Unified Modeling Language*) je soubor grafických znázornění struktury a chování software. Jedná se o standard v oblasti analýzy a návrhu, kdy se programátoři v implementační části procesu vývoje řídí právě UML diagramy[30].

3.2.1 Doménový diagram

Doménový diagram je jedním z UML diagramů struktury a reprezentuje právě strukturu databáze. Jedná se o zjednodušenou verzi diagramu tříd, kdy nejsou potřeba definovat metody jednotlivých tříd (entit), ale pouze atributy. Entity jsou reprezentovány pomocí obdélníků, které jsou navzájem spojeny různými typy čar, jež reprezentují vztahy mezi nimi[31].

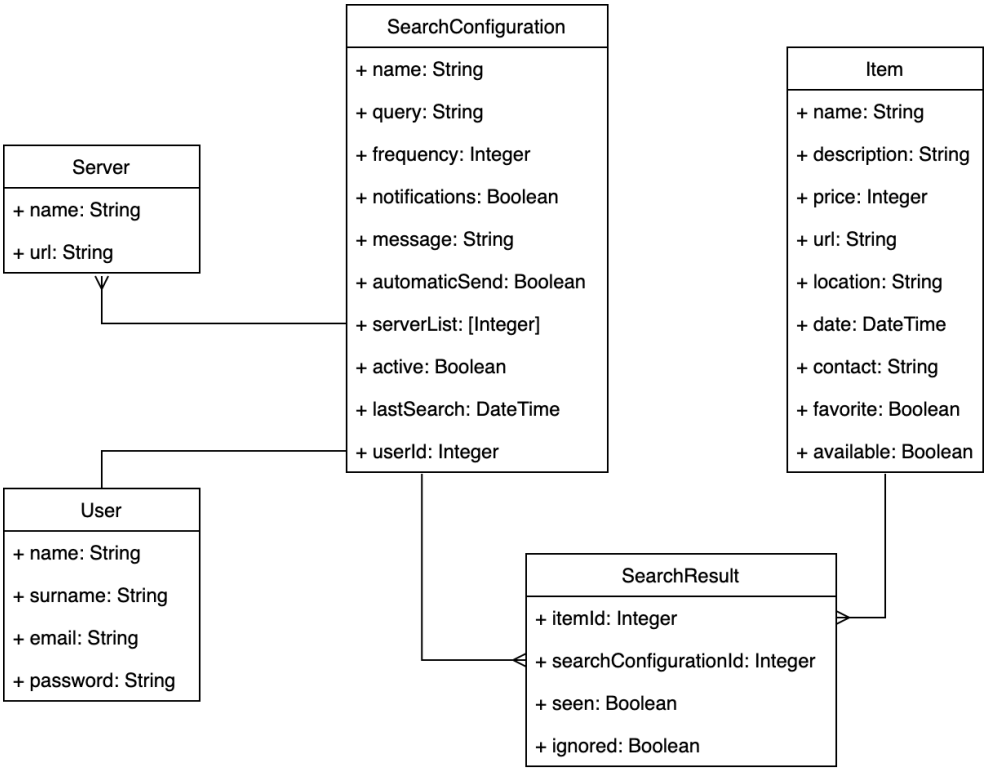
Struktura databáze by měla být poměrně jednoduchá (viz obr. 3.3). Aplikace bude obsahovat pouze pět tříd, kterými jsou *SearchConfiguration* (vyhledávací konfigurace), *Item* (inzerát), *SearchResult* (výsledek vyhledávání), *User* (uživatel) a *Server* (server, na kterém se vyhledává). Třídy pro inzerát a výsledek vyhledávání jsou od sebe odděleny hlavně z toho důvodu, že jeden inzerát může být výsledkem pro více vyhledávacích konfigurací. To potřebuje být odlišeno například kvůli možnosti označení inzerátu jako ignorovaný – je možné, že u jiné vyhledávací konfigurace naopak tento výsledek vidět chceme.

[17] zkratka z anglického *Integrated Development Environment*, jedná se o nástroj pro vývoj a jednodušší psaní kódu v daném programovacím jazyku

[18] `https://www.jetbrains.com/`

[19] `https://insomnia.rest/`

[20] `https://robomongo.org/`



Obrázek 3.3: Doménový diagram

3.2.2 Diagram případů užití

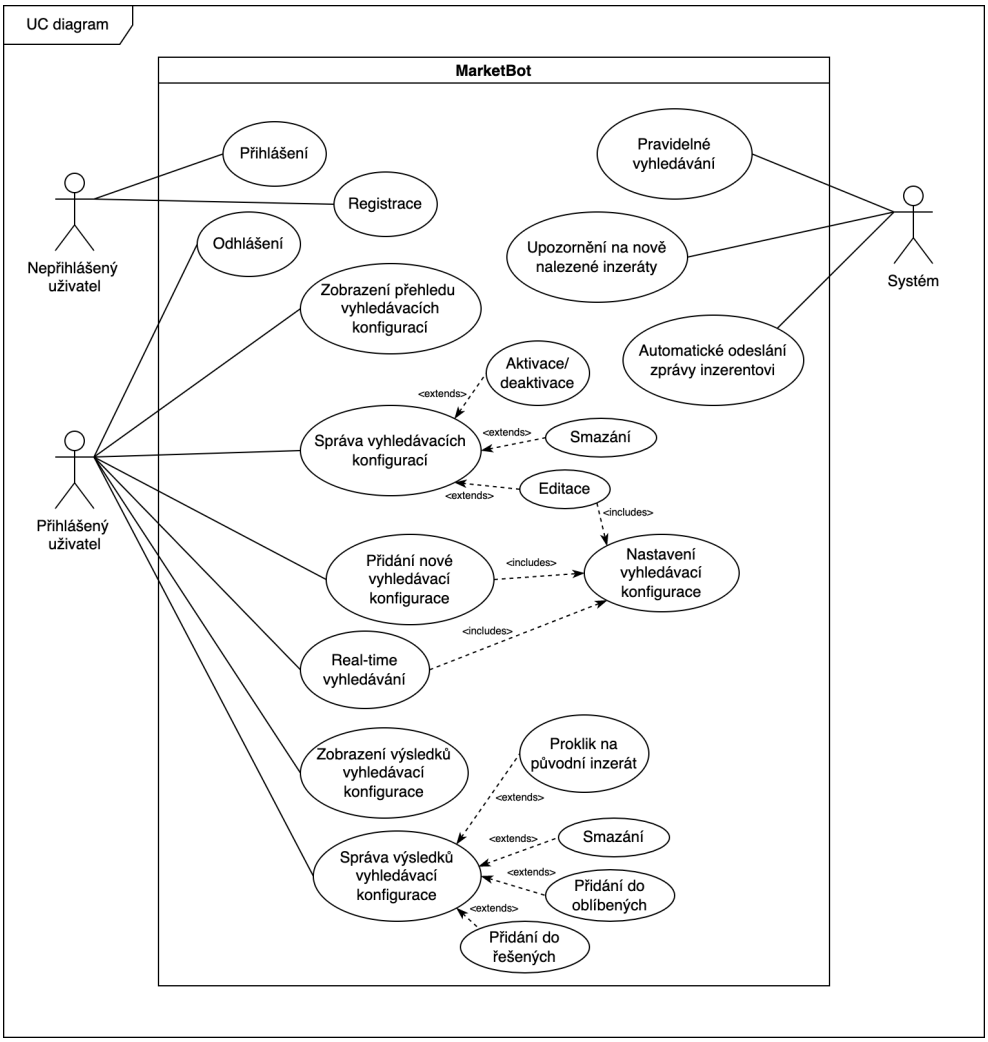
Diagram případů užití (anglicky *Use Case Diagram*) nám říká, jaké by měl software nabízet funkce. Jedná se tedy o jeden z UML diagramů chování. Skládá se z případů užití, aktérů a vztahů mezi nimi[32]. Základem pro jeho vytvoření jsou funkční požadavky zadavatele, které jsou zde graficky znázorněny (viz obr. 3.4).

V mém případě v diagramu figurují tři aktéři:

Nepřihlášený uživatel má k dispozici pouze případy užití přihlášení a registrace.

Přihlášený uživatel má přístup ke všem funkcím systému, které vyplývají z funkčních požadavků zadavatele. Některé případy užití jsou v těchto funkcích součástí jiných případů užití (příznak *includes*) a některé jsou rozšiřující funkcí (příznak *extends*).

Systém se potom stará o pravidelné či automatizované funkce.



Obrázek 3.4: Diagram případů užití

3.3 Uživatelské rozhraní

Uživatelské rozhraní představuje to, jakým způsobem uživatel interaguje s aplikací. Tento proces by měl být pro uživatele co možná nejintuitivnější a nejsnadnější. Při jeho návrhu by se mělo dodržovat několik zásad, jako jsou například[33]:

- konzistentní a oku příjemný design,
- snadno zapamatovatelné procesy,
- poznání cílového uživatele a navrhování systému pro něj,
- počítat s tím, že uživatel může chybovat, a přizpůsobit se tomu.

Pro lepší zapamatovatelnost aplikace pro uživatele jsem zvolila jednoduché pojmenování MarketBot (market od slova trh nebo bazar, bot od slova robot). Pro obecné pojmenování vyhledávacích konfigurací jsem zvolila výraz hlídací pes či chytrý hlídací pes, což vychází ze zajetých konvencí používání tohoto výrazu (viz sekce 2.2).

3.3.1 Persony

Persona je smyšlená postava, která představuje uživatele vyvíjeného softwaru. Má specifické vlastnosti, potřeby a názory a pochází z určitého sociálního zázemí. Jsou to archetypy popisující uživatele pomocí určitých vzorů v chování a jejich cílů.

Pro popis využívají techniku vyprávění příběhu, při které se snadněji vcítíme do role uživatelů, čímž si lépe dokážeme představit to, jak budou uživatelé opravdu systém používat. Persony tímto způsobem zastřešují informace o uživatelích tak, aby byly lépe pochopitelné jak pro experty, tak pro laiky.

Pro návrh jsou potřeba celkem tři druhy person – typický uživatel, příležitostný uživatel a tzv. negativní uživatel[34].

Persony byly navrženy ve spolupráci s Bc. Hanou Fukalovou a Bc. Václavem Králem při tvorbě semestrálního projektu do předmětu NI-NUR.

3.3.1.1 Typický uživatel (typical user)

Typický uživatel bude využívat všechny funkce systému. Systém a jeho uživatelské rozhraní tedy musí být navrženy tak, aby se dalo dosáhnout všech cílů tohoto uživatele. Z těchto cílů jsou potom vytvořeny jednotlivé případy užití (use cases, viz sekce 3.3.2).

Jméno: Daniel Slabý

Věk: 26

Pohlaví: Muž

Záliby: nové technologie, fitness, sledování filmů a seriálů

Typický den: Daniel je pánem svého času, protože pracuje jako podnikatel, ale většinou se snaží být v pracovní den v devět hodin ráno ve své kanceláři. Cestou tam se staví koupit si snídani v místní kavárně a hned poté se vrhá do práce vývojáře webových aplikací. Práci střídá s opravováním chytrých telefonů, hodinek nebo notebooků. Na oběd si zajde buď do restaurace, nebo si nechá jídlo doručit donáškovou službou. Navečer se vrací domů, kde věnuje čas rozvoji znalostí v podobě sledování videí na YouTube[21]. Když má náladu, navštíví blízké fitness centrum. Večer obvykle tráví čas hraním počítačových her, koukáním na nějaký film nebo seriál, nebo se svými přáteli v některém z oblíbených podniků.

Krátká historie: Daniel je mladý ambiciózní muž toužící po úspěšném životě. Stojí nohama na zemi a dává si realistické cíle, je ale velmi odhodlaný a v jejich plnění velmi svědomitý. Motivaci do budoucna vidí hlavně v úspěšném podnikání, založení rodiny a stavbě vlastního domu. Vystudoval elektrotechnickou střední školu se zaměřením na programování, vysoká škola ale poté nebyla nic pro něj. I proto se dal na dráhu podnikatele a věnuje se jen věcem, které ho opravdu zajímají. Co se týče servisu elektrotechniky, ve volném čase se vzdělává hlavně v oblasti opravy základních desek, což ho velmi baví a naplňuje a zároveň by na tom chtěl do budoucna své podnikání postavit.

3.3.1.2 Příležitostný uživatel (occasional user)

Příležitostný uživatel není primární uživatel, takže není tím, pro koho systém vytváříme. Nicméně bychom mu měli alespoň dát nějaké možnosti, jak svých cílů v systému dosáhne.

Jméno: Alena Broumová

Věk: 34

Pohlaví: Žena

Záliby: ekologie, zahradničení, běhání, interiérový design

Typický den: Alena momentálně každé ráno vstává o půl sedmé, aby nachystala snídani své čtyřleté dceři, kterou následně odvádí do školky. Poté se vrátí domů a pracuje z domu. Dělá na poloviční úvazek účetnictví několika menším firmám z jejího města, aby si na mateřské přivydělala a nevyšla také z praxe ve svém oboru. Po práci si jde na chvíli zaběhat a

[21] https://www.youtube.com/

poté vyzvedává dceru ze školky. Doma věnuje nějaký čas uklízení, předělávkám bytu a vaření večeře. Večer po uložení dcery ke spánku ráda brouzdá po internetu, dívá se na obrázky interiérového designu v aplikaci Pinterest[22].

Krátká historie: Alena je svobodná matka. S bývalým manželem jsou tři roky rozvedení a péče o dceru je výhradně v její režii. Její manžel by měl platit na dceru alimenty, ale kvůli dluhům Alena málokdy peníze opravdu dostane. Alena pochází z velkého města. Když byla mladší, chtěla být interiérovou architektkou, ale rodiče ji přesvědčili ke zvolení perspektivnějšího oboru a tak Alena studovala ekonomii a účetnictví. Nejdůležitější věc pro Alenu v jejím životě je nyní její dcera. Chce pro ni vytvořit krásný domov. Ráda kupuje hezké vybavení do interiéru, ale kvůli nedostatku finančních prostředků musí často improvizovat. Pro dceru chce také vytvořit lepší budoucnost a myslí si, že nakupováním z druhé ruky alespoň trochu omezí konzum naší společnosti, který dnes tak ničí planetu.

3.3.1.3 Negativní uživatel (negative persona)

Jedná se o uživatele, který systém pravděpodobně nikdy nepoužije. Jakmile by se funkce systému začaly blížit k chování tohoto uživatele, tak je to znak toho, že je systém navržen špatně.

Jméno: Michal Dolák

Věk: 64

Pohlaví: muž

Záliby: fotbal, kutilství, televize

Typický den: To, kdy Michal vstane, zpravidla záleží na tom, kolik předešlý den vypil piv. Po snídani, kterou většinou tvoří tlačenka s chlebem, tráví den ve své milované dílně, kde sousedům za úplatu opravuje nábytek. Jelikož se do své práce vždy velmi zabere, tak zde i obědvá. Večery pak tráví buď u televize s pivem v ruce, anebo pokud se ten den hraje fotbal, tak s kamarády v hospodě (taky s pivem v ruce).

Krátká historie: Michal je většinou přátelský, ale občas až moc horlivý a tvrdohlavý člověk. Do všeho se žene po hlavě a když si něco myslí, tak vám to dozajista řekne. Tyto vlastnosti ho ale bohužel stály manželství a proto už delší dobu žije sám. Navštěvovat ho chodí jen občas jeho jediná dcera. Michal vystudoval jako lesník a po studiích se živil těžením dřeva a správou lesů na Křivoklátsku. Teď už je v důchodu a sem tam si načerno

[22] https://pinterest.com/

přidělává truhlařinou. S moderními technologiemi není moc kamarád, sám vlastní pouze tlačítkovou Nokii[23] a starý počítač, který dostal od dcery k narozeninám. Ten se ale do teď nenaučil pořádně používat.

3.3.2 Případy užití a scénáře

V této sekci se jedná o případy užití a scénáře z pohledu návrhu uživatelského rozhraní. Vychází z případů užití definovaných dříve, ale slouží spíše pro představu, co mají obsahovat jednotlivé obrazovky aplikace a jak na sebe mají navazovat.

Z případů užití nesmí vyplývat prvky grafického rozhraní, ale spíše co uživatel očekává, že mu systém umožní při plnění daného uživatelského cíle. Scénář je potom souhrn kroků, které vedou k jeho splnění[34]. Ke každému scénáři by měl být vytvořen i tzv. alternativní scénář pro případy, kdy nejde vše podle uživatelova očekávání.

Jednotlivé případy užití a scénáře jsou:

UC1: Přihlášení uživatele

Popis: Na hlavní stránce aplikace uživatel očekává přihlášení do svého účtu pomocí e-mailu a hesla. Po přihlášení uživatel očekává zobrazení jeho domovské stránky.

Scénář:

- vyplnění vstupů pro e-mail a heslo,
- kliknutí na tlačítko *Přihlásit se* a
- přesměrování na domovskou stránku.

Alternativní scénář:

- po kliknutí na tlačítko *Přihlásit se*:
  - některý z povinných vstupů není vyplněn → pole zčervená,
  - heslo není správné → uživatel je upozorněn a je navrhnuta možnost obnovení hesla,
  - pod daným e-mailem neexistuje žádný účet → upozornění uživatele a doporučení registrace.

UC2: Registrace nového uživatele

Popis: Na hlavní stránce uživatel očekává možnost registrovat se do aplikace. V registračním formuláři je předpokládáno vyplnění jména, příjmení, e-mailu a hesla. Po registraci by měl být uživatel rovnou přihlášen a zároveň schopen znovu se přihlásit do svého účtu pomocí UC1.

[23] mobilní telefon

Scénář:

1. přesměrování na stránku s registračním formulářem,
2. vyplnění vstupů pro jméno, příjmení, e-mail, heslo a ověření hesla,
3. kliknutí na tlačítko *Registrovat se* a
4. přesměrování na domovskou stránku.

Alternativní scénář:

- po kliknutí na tlačítko *Registrovat se*:
  - některý z povinných vstupů není vyplněn → pole zčervená,
  - neshoduje se heslo s ověřením hesla → uživatel je upozorněn,
  - pod daným emailem už existuje uživatelský účet → upozornění uživatele a doporučení přihlášení přes přihlašovací formulář.

UC3: Zobrazení a správa všech konfigurací hlídacích psů

Popis: Na domovské stránce uživatel očekává možnost zobrazení přehledu konfigurací (nastavených chytrých hlídacích psů). V přehledu konfigurací uživatel očekává možnost spravovat jednotlivé položky pomocí:

- aktivace/deaktivace (aktivovat nebo deaktivovat již nakonfigurovaného hlídacího psa),
- editace (editovat již nakonfigurovaného hlídacího psa),
- smazání (smazat již nakonfigurovaného hlídacího psa).

Scénář:

1. kliknutí na tlačítko zobrazit seznam hlídacích psů,
2. *scrollování* seznamem hlídacích psů,
3. kliknutí na konkrétní ikonku podle požadované akce:
   - přepínač (aktivace/deaktivace),
   - ozubené kolečko (editace),
   - křížek (smazání),
4. případné přesměrování na výsledky konkrétní konfigurace.

Alternativní scénář:

- není aktivní žádný hlídací pes → zobrazena pouze možnost přidat hlídacího psa,
- došlo k odhlášení uživatele → přesměrováno na přihlašovací stránku.

UC4: Konfigurace nového hlídacího psa

Popis: V přehledu již nastavených chytrých hlídacích psů uživatel očekává možnost přidat nového hlídacího psa. Uživatel by měl mít možnost nastavení následujících informací:

- název,
- hledané výrazy,
- seznam prohledávaných webů,
- frekvence vyhledávání,
- zpráva pro inzerenta,
- zda se má zpráva odesílat automaticky,
- zda se mají notifikace s novými nálezy zasílat na e-mail.

Po nastavení uživatel očekává aktivovaného hlídacího psa.

Scénář:

1. kliknutí na tlačítko *Přidat konfiguraci*,
2. vyplnění vstupů pro název, hledané výrazy, frekvenci vyhledávání, zprávu pro inzerenta,
3. zaškrtnutí *checkboxů* pro seznam webů, automatické odesílání zprávy inzerentovi, posílání notifikací na e-mail,
4. kliknutí na tlačítko *Přidat*.

Alternativní scénář:

- po kliknutí na tlačítko *Přidat* některý z povinných vstupů není vyplněn → pole zčervená a uživatel je upozorněn.

UC5: Zobrazení a správa výsledků konkrétního hlídacího psa

Popis: Na stránce s přehledem všech hlídacích psů uživatel očekává možnost zobrazení výsledků pro konkrétního hlídacího psa. Po zobrazení výsledků se očekává možnost správy jednotlivých položek:

- přidat výsledek do oblíbených,
- odstranit výsledek, který se nehodí,
- odškrtnout již vyřešený inzerát.

Scénář:

1. přesměrování na stránku se seznamem výsledků skrz položku v seznamu hlídacích psů,
2. *scrollování* seznamem výsledků (řazeno podle času vyhledání),
3. kliknutí na ikonku u položky podle požadované akce:
   - fajfka (vyřešený inzerát),
   - srdíčko (přidat do oblíbených),

– křížek (ignorovat inzerát).

Alternativní scénář:

– vybraný pes nenalezl žádné výsledky → zobrazení prázdného seznamu a nabídnuta případná editace hlídacího psa.

UC6: *Real-time* vyhledávání

Popis: Na domovské stránce uživatel předpokládá možnost zobrazení real-time vyhledávání. Po zobrazení real-time vyhledávání uživatel předpokládá možnost zadání požadovaného zboží a následné zobrazení výsledků, které jsou dostupné na bazarových webech.

Scénář:

1. přesměrování na stránku s *real-time* vyhledáváním,
2. vyplnění vstupu s názvem zboží,
3. kliknutí na ikonu lupa (vyhledat),
4. *scrollování* výsledky vyhledávání a
5. případné přesměrování na jednotlivé inzeráty.

Alternativní scénář:

– hledání nenalezlo žádné výsledky → zobrazen prázdný seznam a nabídnutí možnosti přidání hlídacího psa pro daný výraz.

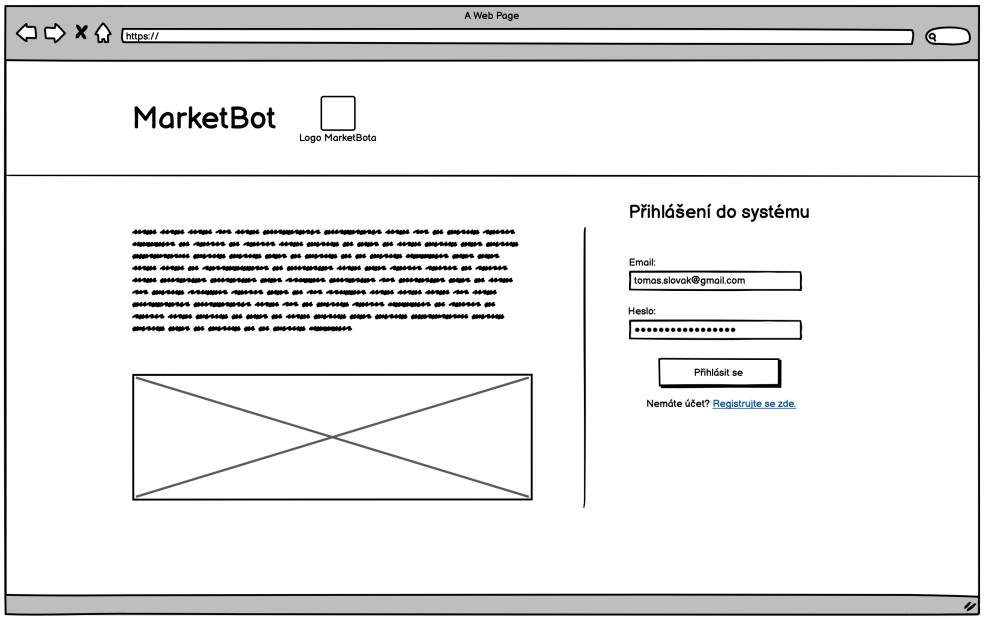
3.3.3 Prototypy

Prototypy aplikací se dělí na dva typy podle míry podobnosti s finálním vzhledem a funkcemi aplikace[34]:

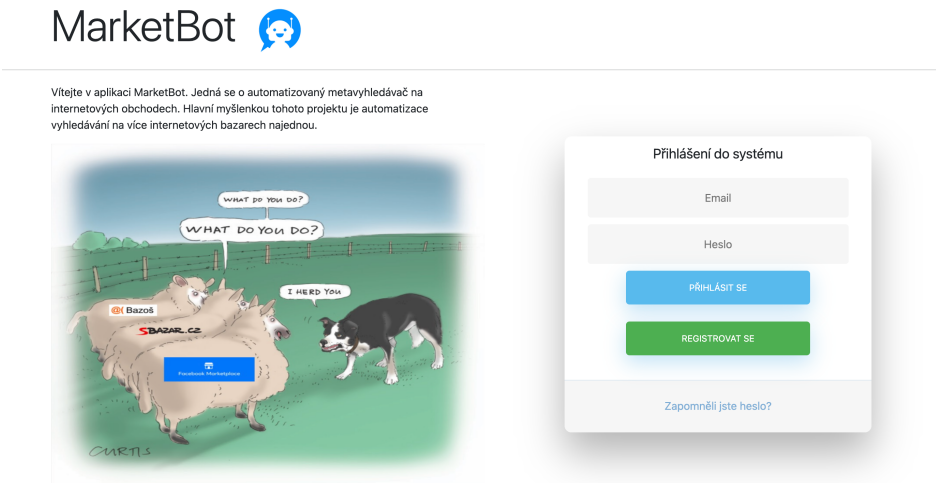
Low-Fidelity (Lo-Fi) je prvotní, méně propracovaný prototyp a bývá prvním krokem specifikující vzhled aplikace. Může se jednat například pouze o návrh pomocí papíru a tužky, v mém případě je však použita technika *wireframes*. Jedná se o černobílé náčrty jednotlivých oken aplikace s rozloženými prvky jako jsou třeba menu, hlavička, formuláře a tlačítka. Dle platformy, na které se *wireframes* vytváří, můžou navíc některé prvky simulovat i přesměrovávání na jiné stránky[35].

High-Fidelity (Hi-Fi) prototyp by měl reprezentovat konečný vzhled aplikace a pro jeho návrh by měla být použita finální technologie. Využívá se hlavně na uživatelské testování, tudíž by měl na uživatele působit dojmem již hotové aplikace a naplňovat všechny definované požadavky, aby se daly otestovat.

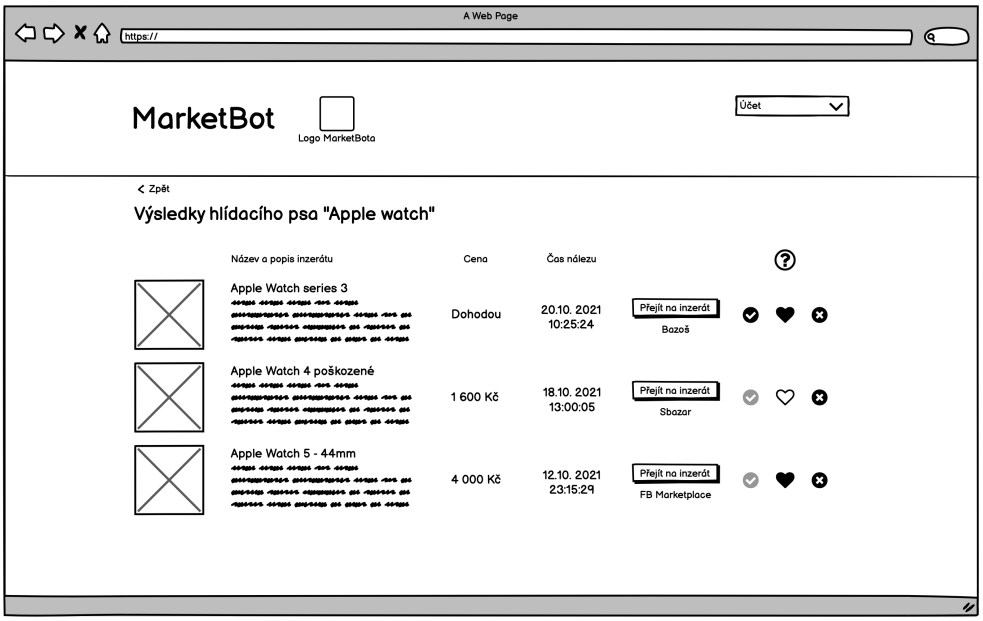
Na obrázcích (3.5, 3.7 a 3.9) jsou vyobrazeny ukázky prvotního náčrtu aplikace a na obrázcích (3.6, 3.8 a 3.10) pro porovnání propracovanější prototyp. Oba prototypy byly vytvořeny v semestrálním projektu předmětu NI-NUR ve spolupráci s Bc. Václavem Králem, Bc. Hanou Fukalovou a Bc. Janem Štefánikem. Kompletní *wireframes* se nachází na přiloženém paměťovém zařízení.



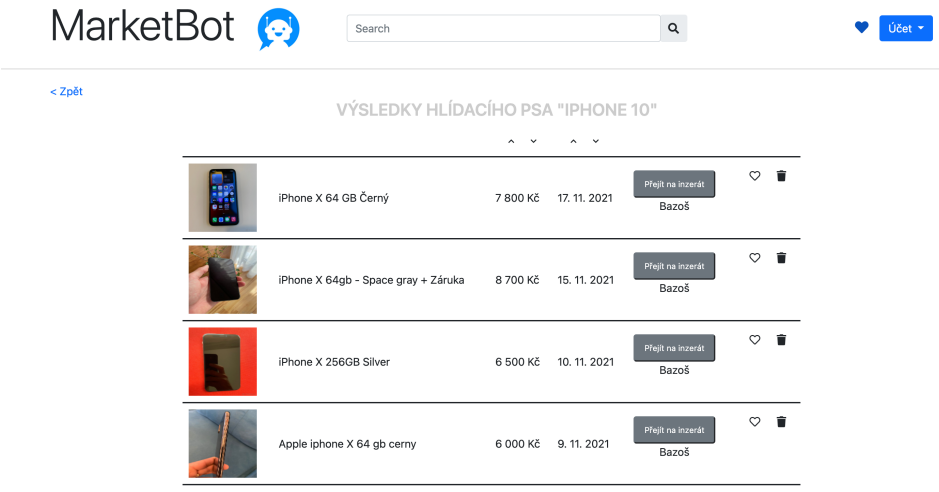
Obrázek 3.5: Wireframes – úvodní stránka



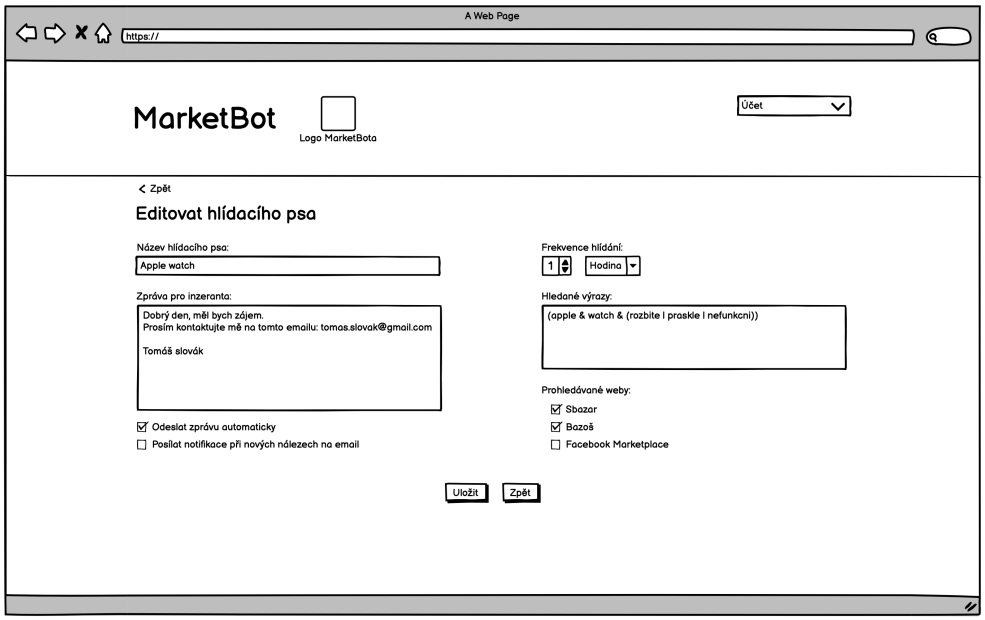
Obrázek 3.6: Hi-Fi prototyp – úvodní stránka



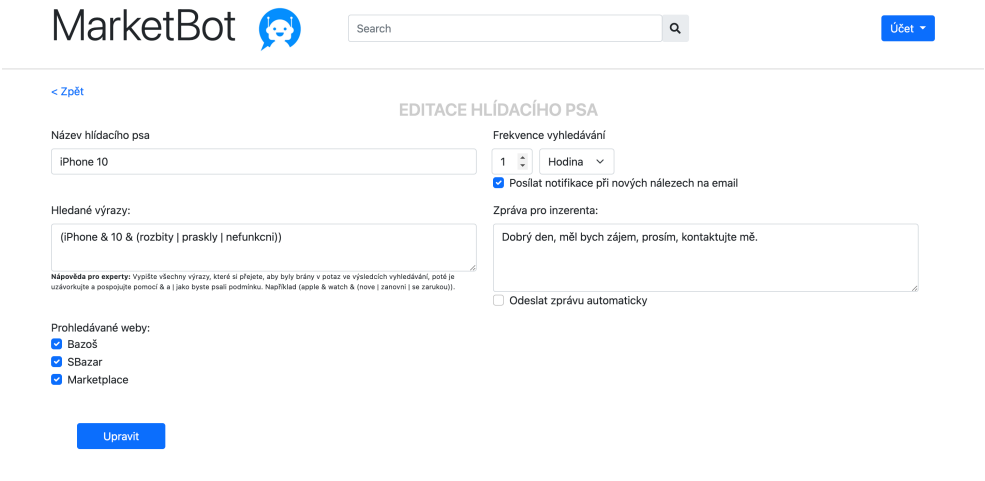
Obrázek 3.7: Wireframes – výsledky vyhledávací konfigurace



Obrázek 3.8: Hi-Fi prototyp – výsledky vyhledávací konfigurace



Obrázek 3.9: Wireframes – formulář s editací vyhledávací konfigurace



Obrázek 3.10: Hi-Fi prototyp – formulář s editací vyhledávací konfigurace

Kapitola 4

Realizace

V této kapitole se budu věnovat tomu, jak probíhala realizace požadavků a návrhu v procesu implementace a následného testování.

Konečný vzhled a funkčnost aplikace na sto procent neodpovídají návrhu. Je to způsobeno přirozeným procesem, kdy si člověk uvědomí nové informace až v průběhu implementace. Zároveň zde jsou zapracovány některé z dodatečných změn, které byly konzultované se zadavatelem.

4.1 Implementace

Implementace je rozdělena do dvou aplikací, které spolu komunikují přes API rozhraní. Klientská aplikace reprezentuje prezenční vrstvu výsledné aplikace, serverová část zase vrstvu databázovou a aplikační.

Ukázky kódů v této sekci nemusí úplně přesně odpovídat realitě. Jedná se spíše o pseudokódy, které jsou zde pro lepší orientaci v jejich popisu. Jsou zkráceny či zjednodušeny hlavně pro lepší přehlednost.

4.1.1 Klient

Jedná se o React aplikaci. Struktura aplikace je poměrně jednoduchá (viz 4.1). Vstupní souborem pro prohlížeč je soubor `public/index.html`. V tomto souboru je v `<body>` jen jeden HTML element, a to `<div id='root'></div>`.

Zdrojové soubory aplikace se potom nachází ve složce `src`. Hlavním souborem aplikace je `index.js`, který plní již zmíněný `div` (viz kód níže).

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

```
client ..... kořenový adresář
├── public ..... adresář dostupný z prohlížeče
│   └── index.html
├── src
│   ├── components ..... znovupoužitelné komponenty
│   │   ├── Button.js
│   │   ├── Error.js
│   │   ├── Footer.js
│   │   ├── FrequencyForm.js
│   │   ├── Header.js
│   │   ├── LoginForm.js
│   │   ├── Logo.js
│   │   ├── RegisterForm.js
│   │   ├── SearchBar.js
│   │   ├── SearchConfigurationDetail.js
│   │   ├── SearchConfigurationForm.js
│   │   ├── SearchConfigurationList.js
│   │   ├── SearchConfigurationListItem.js
│   │   ├── SearchResultListItem.js
│   │   ├── SettingsForm.js
│   │   └── UserSettings.js
│   ├── images ..... adresář s použitými obrázky
│   ├── routes ..... jednotlivé stránky aplikace
│   │   ├── Home.js
│   │   ├── Login.js
│   │   ├── Register.js
│   │   ├── Search.js
│   │   ├── SearchConfigurationAdd.js
│   │   ├── SearchConfigurationEdit.js
│   │   ├── SearchConfigurationResults.js
│   │   └── Settings.js
│   ├── App.css
│   ├── App.js
│   ├── index.js ..... hlavní soubor, který zobrazuje aplikaci
│   ├── reportWebVitals.js ..... měření chodu aplikace
│   └── utils.js
├── .env.development ..... soubor s proměnnými pro vývojové prostředí
├── .env.production ..... soubor s proměnnými pro produkční prostředí
└── package.json ..... soubor se závislostmi projektu
```

Obrázek 4.1: MarketBot – struktura klientské aplikace

Komponenta `App` (soubor `App.js`) potom představuje strukturu celé aplikace. Strukturou myslím hlavně vytvoření cest neboli routování. Toho jsem dosáhla použitím komponenty `BrowserRouter` z balíčku `react-router-dom`.

Cesty použité v routování jsou k dispozici ve složce `routes`. Každá cesta představuje jinou obrazovku aplikace, celkem jich je osm, z toho jsou dvě veřejné a zbytek soukromých (dostupné pouze přihlášeným uživatelům). Tyto jednotlivé cesty jsou ve skutečnosti klasické React komponenty, které mám pouze oddělené v jiném adresáři od ostatních komponent kvůli lepší přehlednosti. Ostatní komponenty jsou potom ve složce `components`.

Mimo zdrojové soubory klientské aplikace se v kořenovém adresáři nachází také soubor `package.json`, který obsahuje všechny použité závislosti (balíčky). Také zde můžeme najít `.env` soubory, ve kterých jsou obsaženy proměnné pro dané prostředí (v našem případě vývojové nebo produkční).

HTML struktura všech stránek aplikace je stejná, každá obsahuje vždy hlavičku (komponenta `Header.js`), obsah a patičku (komponenta `Footer.js`). Hlavičku můžeme jako obvykle vidět nahoře na stránce, patičku zase dole. Obsah je ve středu a mění se v závislosti na tom, na které stránce se právě nacházíme.

Hlavička je dvojího typu – veřejná a soukromá. Pokud uživatel není přihlášen, tedy pokud se nacházíme na stránce s přihlášením nebo registrací, hlavička je veřejná. Odlišuje se od soukromé tím, že obsahuje pouze logo (komponenta `Logo.js`) aplikace. Soukromá hlavička obsahuje navíc vyhledávací lištu (komponenta `SearchBar.js`) a rozbalovací menu s odkazem na nastavení uživatele a s odkazem pro odhlášení (komponenta `UserSettings.js`).

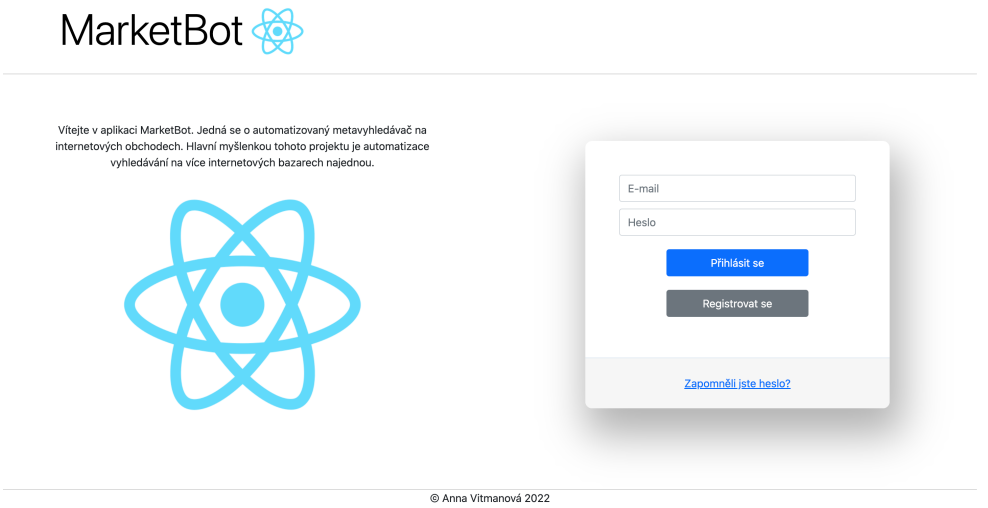
4.1.1.1 Přihlášení a registrace

Úvodní stránkou celé aplikace je stránka s přihlášením (viz obr. 4.2). Na této stránce můžeme vidět stručnou informaci o účelu aplikace a přihlašovací formulář (komponenta `LoginForm.js`). Nachází se zde i odkaz na stránku s registrací. Stránka s registrací (viz obr. 4.3) obsahuje pouze registrační formulář (komponenta `RegisterForm.js`).

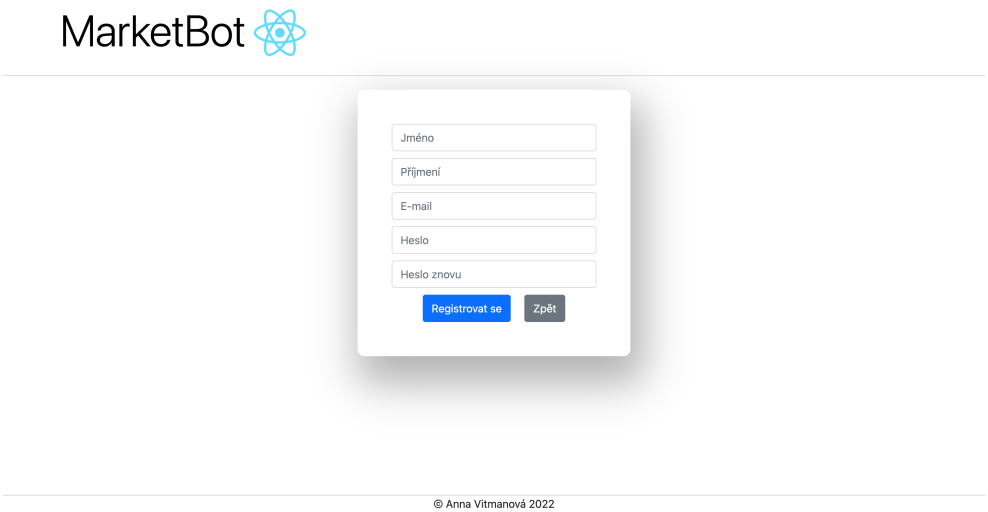
Po úspěšném přihlášení či registraci se do *local storage* prohlížeče vloží přístupový token, který v odpovědi odeslal server, a následně je uživatel přesměrován na domovskou stránku. *Local storage* je úložiště v prohlížeči (tedy na straně uživatele), které uchovává data, jež jsou či budou potřebná pro používání aplikace[36]. Tento token je poté vkládán do hlavičky každého dotazu z klienta na server (více v sekci 4.1.2.3).

4.1.1.2 Domovská stránka

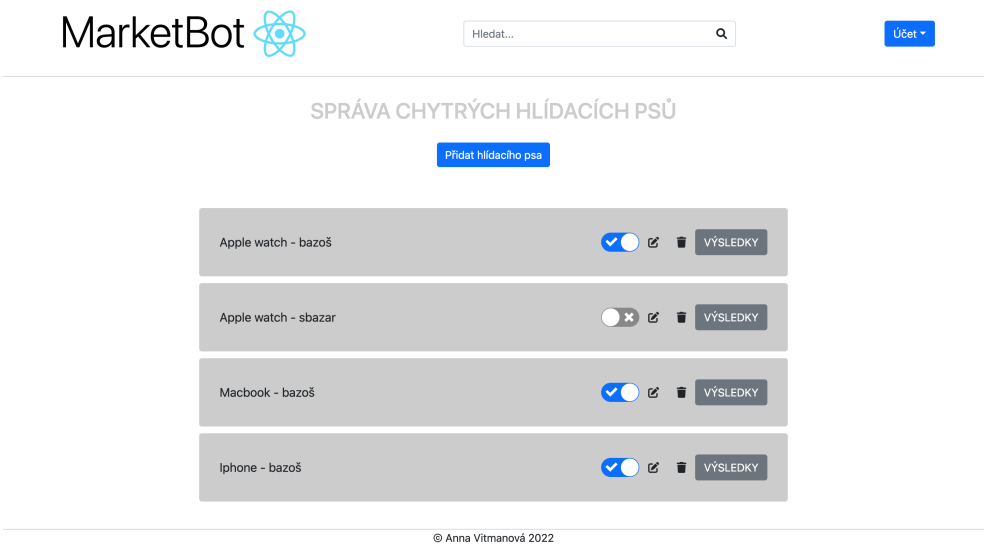
Domovská stránka (viz obr. 4.4) slouží hlavně jako přehled uživatelových vyhledávacích konfigurací. Mimo jiné se zde nachází i tlačítko *Přidat hlídacího psa*, které uživatele přesměruje na stránku s přidáním nové vyhledávací kon-



Obrázek 4.2: MarketBot – přihlášení



Obrázek 4.3: MarketBot – registrace



Obrázek 4.4: MarketBot – domovská stránka

figurace. Seznam všech konfigurací obsahuje všechny vyhledávací konfigurace vytvořené uživatelem (komponenta `SearchConfigurationList.js`). Momentálně pro tento seznam neexistuje žádné stránkování, se zadavatelem jsme se shodli, že pro jeho využití stránkování není potřeba, protože neplánuje mít mnoho konfigurací.

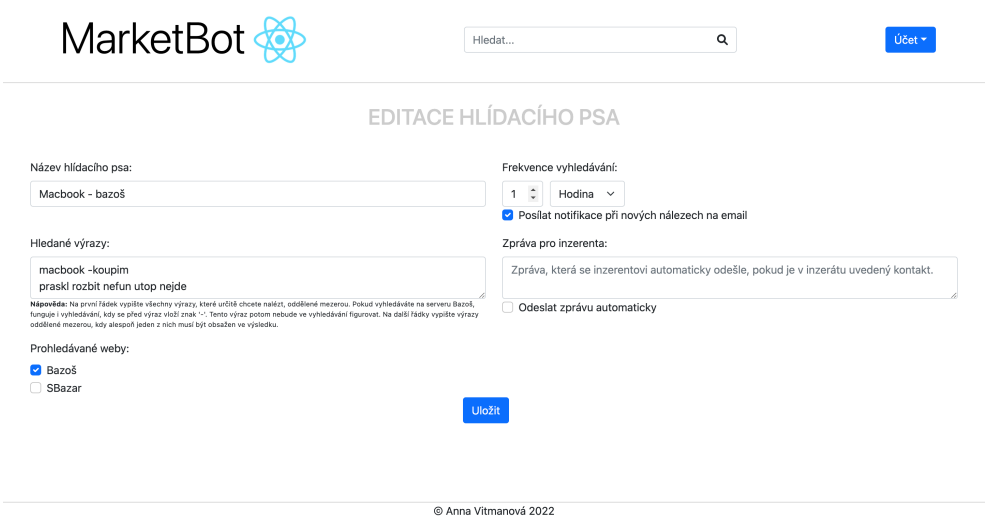
Každý prvek seznamu potom obsahuje název dané konfigurace na levé straně, na pravé straně můžeme vidět:

- přepínací tlačítko (*Switch Button*), které značí, zda je je vyhledávání aktivní či nikoli,
- ikonu editace, která přesměruje uživatele na stránku s formulářem pro editaci vyhledávací konfigurace,
- ikonu smazání, která danou vyhledávací konfiguraci smaže, a
- tlačítko pro přesměrování na stránku s výsledky vyhledávání konfigurace.

4.1.1.3 Přidání a editace vyhledávací konfigurace

Pro přidávání a editaci vyhledávací konfigurace je využíván stejný formulář. Jediný rozdíl je samozřejmě v tom, že pokud je konfigurace editována, jednotlivé vstupy už jsou předvyplněny hodnotami z databáze podle daného ID, které se nachází v URL cestě.

Jednotlivé vstupy potom jsou:



Obrázek 4.5: MarketBot – vyhledávací konfigurace

- *Název hlídacího psa*, který reprezentuje stručný popis toho, co vyhledávání konfigurace zahrnuje.
- *Hledané výrazy*
- *Prohledávané weby*
- *Frekvence vyhledávání*
- *Posílání upozornění na e-mail*
- *Zpráva inzerentovi*

Hledané výrazy jsou poměrně komplikovaným vstupem, proto pod tímto elementem vidíme celkem podrobnou nápovědu. Pro tento element (který je momentálně `<input type="text"/>`) by bylo do budoucna lepší vytvořit přehlednější komponentu, která by lépe reprezentovala hledané výrazy. Nejlepší by asi bylo reprezentovat jednotlivé řádky pomocí *React Multivalue Text Input*[24] komponenty (viz obr.4.6). Ve výchozím formuláři by vždy byla jedna tato komponenta pro základní hledané výrazy a tlačítko *Přidat volitelné výrazy*, po jehož stisknutí by se ve formuláři objevila další tato komponenta, která by reprezentovala volitelné výrazy (neboli další řádek v aktuálním vstupu).

Frekvence vyhledávání momentálně v aplikaci nehraje žádnou roli. Dle původního záměru je aplikace předpřipravena k jejímu užití, ale zadavatel se vyjádřil, že nakonec tato funkce asi nebude potřeba, protože chce stejně výsledky vyhledávat jak jen často to půjde.

[24] `https://www.npmjs.com/package/react-multivalue-text-input`

Obrázek 4.6: React Multivalue Text Input

Ani textový vstup *Zpráva pro inzerenta* a zaškrtávací pole *Odeslat zprávu automaticky* zatím nehrají žádnou roli, neb není na serveru zatím vyřešeno odesílání automatických zpráv pro inzerenta. Oproti frekvenci vyhledávání ale zadavatel počítá s tím, že do budoucna tato funkce implementována bude.

4.1.1.4 Výsledky vyhledávací konfigurace

Stránka (`SearchConfigurationResults.js`) slouží pro zobrazení a správu výsledků dané vyhledávací konfigurace. Jedná se o nejpropracovanější stránku aplikace, co se množství funkcí týče (viz obr. 4.7).

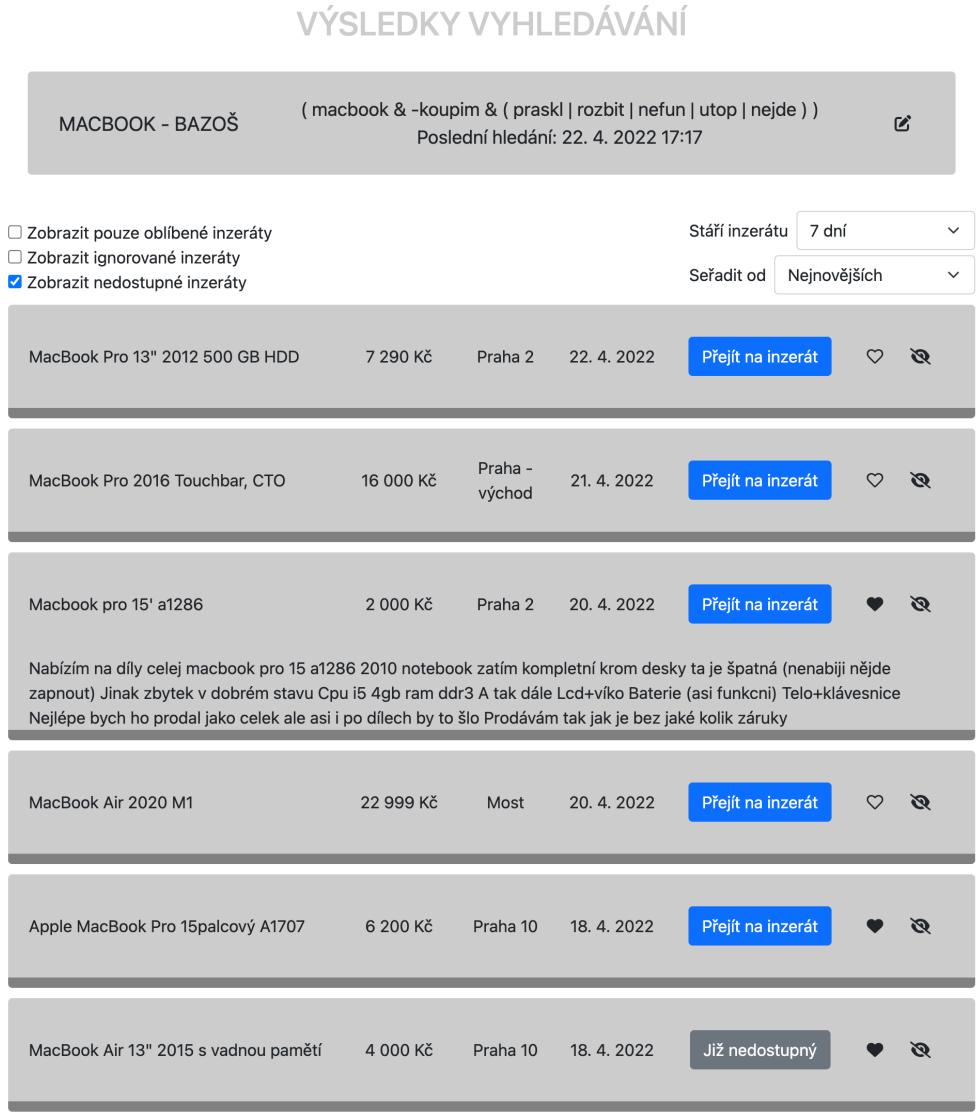
Nad výsledky se nachází detail konfigurace (komponenta `SearchConfigurationDetail.js`), aby měl uživatel přehled o tom, jaké výsledky mu jsou vlastně zobrazovány. Jedná se o název, vyhledávací *query* a čas, kdy doběhlo poslední vyhledávání. Zároveň je možné pomocí ikony vpravo přejít na stránku s editací konfigurace.

Pod tímto detailem se nachází samotný seznam s výsledky. Každý jeden výsledek je zobrazen pomocí samostatné `SearchResultListItem.js` komponenty. Je v ní uveden název inzerátu, cena, lokalita a datum přidání inzerátu. Dále je v komponentě tlačítko, po kterém je uživatel v novém okně prohlížeče přesměrován na URL adresu inzerátu. Dále je možné tuto komponentu rozbalit kliknutím na spodní tmavší lištu, čímž je zobrazen i popis daného inzerátu.

Vpravo se poté nachází klikatelné ikony:

Ikona srdce značí, zda je inzerát zařazen do oblíbených položek či nikoli. Pokud je vidět ikona obrysu srdce, znamená to, že inzerát není označen jako oblíbený. Po kliknutí je ikona změněna na vyplněné srdce a inzerát zařazen do oblíbených. Platí i opačný proces.

Ikona oko říká, zda chceme inzerát zobrazovat ve výchozím seznamu. Pokud je oko přeškrtnuté, položku v seznamu vidíme. Po kliknutí je položka



Obrázek 4.7: MarketBot – výsledky vyhledávání

skryta a inzerát označen jako ignorovaný. Pokud chceme položku znovu zobrazit, musíme si nejprve zobrazit ignorované inzeráty (viz další odstavec). Ikonu v tomto případě uvidíme nepřeškrtnutou a kliknutím se inzerát nastaví zpět na viditelný a ikona se změní.

Nad seznamem se nachází několik možností, jak si seřadit či vyfiltrovat výsledky. Na levé straně se nachází *checkboxy*, které jsou ve výchozím stavu vždy nezaškrtnuté. Je zde možnost zobrazit pouze oblíbené inzeráty, zobrazit i ignorované inzeráty a zobrazit i nedostupné inzeráty. Nedostupné inzeráty se od těch dostupných liší tím, že mají neaktivní tlačítko na přesměrování na inzerát. Na pravé straně se potom nachází dva *comboboxy*. Jeden filtruje výsledky podle stáří a nabízí možnost zobrazit inzeráty z posledního jednoho dne, z posledního týdne (výchozí filtr), z posledního měsíce či všechny nalezené inzeráty. Dále je zde možnost řazení inzerátů a to jak podle ceny, tak podle data přidání inzerátu, přičemž výchozí je řazení od nejnovějších inzerátů.

4.1.2 Server

Jedná se o Node.js aplikaci. Struktura aplikace (viz 4.8) je už o něco komplikovanější než klientská část. Ve zdrojových kódech se opakují následující názvy, které plynou z návrhu databáze:

- Item,
- SearchConfiguration,
- SearchResult,
- Server a
- User.

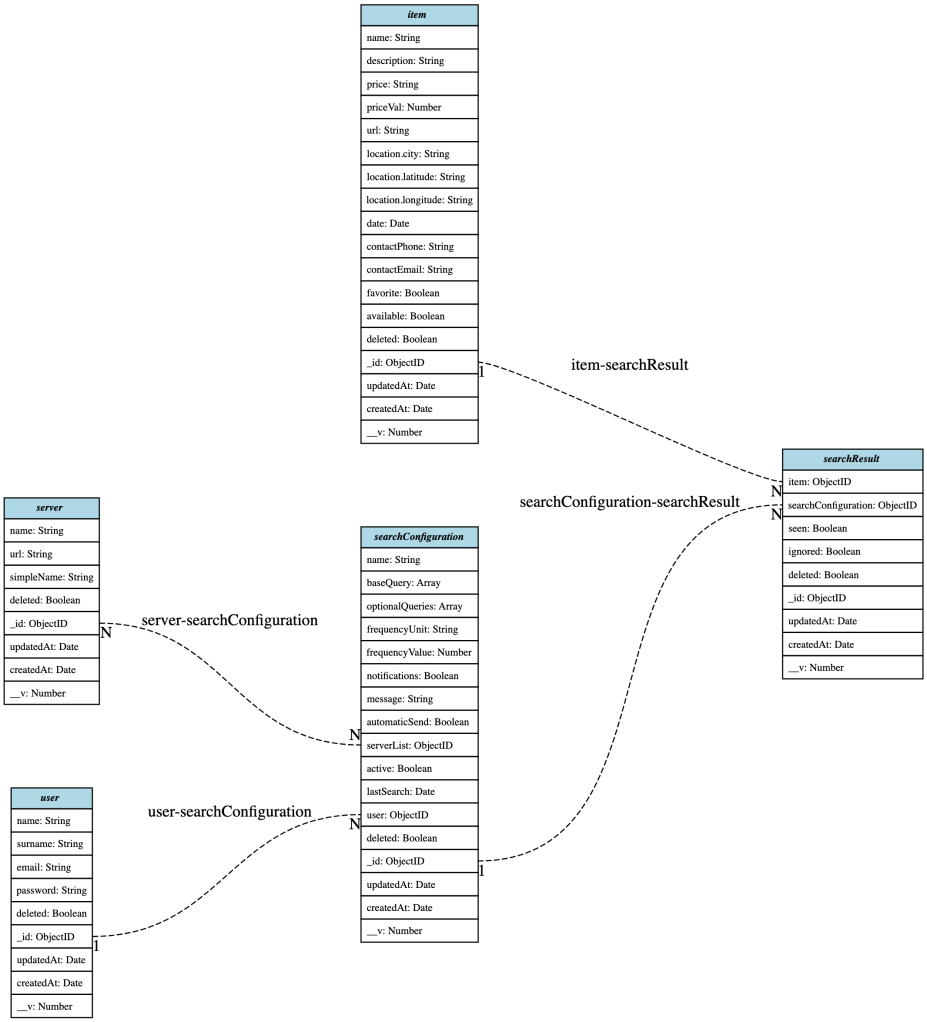
Pro těchto všech pět názvů je v každé části *back-endu* vytvořen pro lepší přehlednost samostatný soubor. Tyto části jsou:

Controller zajišťuje logiku celé aplikace. Mimo pět hlavních tříd vytvořených pro uvedené názvy obsahuje ještě `BaseController.js`, ze kterého ostatní třídy dědí, a který tedy obsahuje společné metody. Dále se zde nachází i `EmailController.js`, jež má na starosti odesílání e-mailů, a `CronController.js`, který bude dopodrobna popsán dále v textu.

DAO (zkratka z anglického *Database Access Object*) obsahuje rozhraní pro ovládání databáze. Jedná se o návrhový vzor, který odděluje kód přistupující do databáze od zpracování dat v logické části aplikace. Výhodou použití tohoto vzoru je například snadnější přechod na jinou databázi, kdy stačí přepsat pouze DAO a nezasahovat do jiných částí aplikace[37]. Také je zde obsažena třída `BaseDao.js`, ze které ostatní třídy dědí společné metody.

```
server ..... kořenový adresář
├── src
│   ├── controller
│   │   ├── BaseController.js
│   │   ├── CronController.js
│   │   ├── EmailController.js
│   │   ├── ItemController.js
│   │   ├── SearchConfigurationController.js
│   │   ├── SearchResultController.js
│   │   ├── ServerController.js
│   │   └── UserController.js
│   ├── dao
│   │   ├── BaseDao.js
│   │   ├── ItemDao.js
│   │   ├── SearchConfigurationDao.js
│   │   ├── SearchResultDao.js
│   │   ├── ServerDao.js
│   │   └── UserDao.js
│   ├── model
│   │   ├── ItemModel.js
│   │   ├── SearchConfigurationModel.js
│   │   ├── SearchResultModel.js
│   │   ├── ServerModel.js
│   │   └── UserModel.js
│   ├── router
│   │   ├── ItemRouter.js
│   │   ├── SearchConfigurationRouter.js
│   │   ├── SearchResultRouter.js
│   │   ├── ServerRouter.js
│   │   └── UserRouter.js
│   ├── scraper
│   │   ├── data
│   │   │   ├── bazos
│   │   │   └── sbazar
│   │   ├── resources
│   │   ├── bazos-scraper.py
│   │   ├── requirements.txt
│   │   └── sbazar-scraper.py
│   └── index.js
├── .babelrc ..... konfigurační soubor pro Babel (viz sekce 4.2)
├── .env.template ..... šablona souboru s proměnnými prostředí
├── logs.txt ..... soubor se záznamy z aplikace
└── package.json ..... soubor se závislostmi projektu
```

Obrázek 4.8: MarketBot – struktura server aplikace



Obrázek 4.9: MarketBot – struktura databáze

Model je část zodpovědná za správnou strukturu databáze a byla vytvořena právě na základě návrhu databáze. Konečná struktura databáze (viz obr. 4.9) se v průběhu implementace oproti návrhu v nějakých bodech liší, jedná se hlavně o automaticky přidané pole balíčkem `mongoose` (*createdAt* a *updatedAt*). Právě tento balíček zprostředkovává pro Node.js příkazy k ovládání Mongo databáze.

Router má na starost definování cest pro API, na které přistupuje *front-end* aplikace. To zajišťuje třída `Router` z knihovny `express`. Díky vytvoření routeru a definování cest jsou potom dotazy přesměrovávány do *controllerů*, ze kterých poté chodí zpět odpovědi.

4.1.2.1 Cron

Implementace tzv. *cronu* je velmi podstatnou částí celé aplikace. Jedná se o nástroj, díky kterému jsme schopni naplánovat rozvrh, ve kterém se budou pravidelně a opakovaně spouštět námi definované úlohy. Jednomu příkazu (každý příkaz je odpovědný za jednu úlohu) se potom říká *cron job*[38, 39].

V mém případě je je potřeba přidat *cron*, který bude mít na starosti pravidelné spouštění *scraper* skriptu (viz sekce 4.1.2.2) a následné uložení nově nalezených inzerátů. Implementace se nachází v souboru `CronController.js`.

Hlavní metodou, která je pravidelně každých dvacet minut spouštěna, je metoda `run()` (viz kód níže). Metoda nejprve zjistí, zda ještě neběží předchozí hledání, což je indikováno proměnnou `this.running`. Pokud běží, žádné další hledání neproběhne, protože by se mohlo stát, že se *cron* a skripty zacyklí (to plyne z mé implementace).

```
run() {
   if (!this.running) {
      let configurations = listActive();
      for (let conf of configurations) {
         for (let server of conf.serverList) {
            this.threadCnt++;
         }
      }
      if (this.threadCnt > 0) {
          this.running = true;
      }
      for (let conf of configurations) {
         if (shouldSearch(conf.id)) {
            createScraperResourcesCsv(conf);
            for (let server of conf.serverList) {
               execScript(conf.id, server, timeStarted);
            }
         }
      }
   }
}
```

Pokud *cron* neběží, nejprve z databáze zjistí, které jsou aktivní konfigurace. Tyto konfigurace postupně projde a zjistí, na jakých serverech by mělo vyhledávání probíhat. Podle toho se změní proměnná `this.threadCnt` (počet vláken), která značí, kolik vláken s jednotlivými Python skripty poběží. Každé vlákno se tedy rovná jednomu běžícímu skriptu. Pokud jsou nějaké aktivní konfigurace, proměnná `this.running` se nastaví na `true`.

Dalším krokem je projít všechny aktivní konfigurace a pro každý server, na kterém by měly vyhledávat, spustit skript (metoda `execScript()`). Počet

těchto spuštění by měl být roven proměnné `this.threadCnt`. Před spuštěním skriptu je ale potřeba *scraperu* nějakým způsobem předat informaci o tom, jaké výrazy by měl vyhledávat. To jsem vyřešila tím, že pro každou konfiguraci před spuštěním skriptu vytvořím ve *scraperu* soubor `/resources/id.csv`, ve kterém se nachází seznam hledaných výrazů (`createScraperResourcesCsv()` metoda). O tomto více v sekci 4.1.2.2.

Metoda `execScript()` (viz níže) spustí skript pomocí metody `exec()` z knihovny `child_process`. Poté se čeká na doběhnutí skriptu pomocí *callbacku.*, který se této metodě předává jako druhý parametr. Díky tomuto konceptu asynchronního programování (viz sekce 3.1.2) se nemusí čekat na doběhnutí jednoho skriptu, aby se mohl spustit skript další, ale mohou se spustit všechny skripty ihned po sobě. Dokončení operace nad daty ze *scraperu* potom řeší právě *callback* funkce.

```
execScript(id, server) {
   const scriptPath = path.resolve(...);
   exec(`python3 ${scriptPath} ${id}`, (err) => {
      if (err) {
         //handle error
      } else {
         this.saveResultsConfigs.push(id)
         this.threadCnt--;
         if (this.threadCnt === 0) {
            // 1. save results
            for (let conf of this.saveResultsConfigs) {
               this.saveResults(conf.id);
            }
            // 2. send notifying email
            this.generateEmails();
            this.newItems = {};
            this.running = false;
         }
      }
   })
}
```

Pokud skript nevrátí error, *callback* funkce nejprve vloží ID dané konfigurace do proměnné `this.saveResultsConfigs`. Tato proměnná slouží k tomu, aby byl po doběhnutí všech skriptů k dispozici seznam všech doběhlých konfigurací. Dále se proběhne `this.threadCnt--`, což značí, že doběhlo jedno „vlákno“. Pokud projde podmínka `(this.threadCnt === 0)`, znamená to, že už doběhla všechna tato „vlákna“, tedy skripty. Pokud doběhly všechny skripty, může začít ukládání výsledků do databáze. Postupně se pro všechny doběhlé konfigurace (což víme z proměnné `this.saveResultsConfigs`) zavolá metoda `saveResults()`.

```
saveResults(id) {
   let availableItems = this.getAvailableItems(id);
   const filePath = path.resolve(...);
   let newItems = [];
   const data = fs.readFileSync(filePath, 'utf8')
   const items = JSON.parse(data);
   for (let item of items) {
      delete availableItems[item.url];
      let newResult;
      let existingItem = ItemDao.find({url: item.url});
      if (existingItem) {
         const existingResult = SearchResultDao.find({
            item: existingItem.id,
            searchConfiguration: id
         });
         if (existingResult) {
            // result exists -> only update item
         } else {
            // create new search result
            newResult = SearchResultDao.create({
               item: existingItem.id,
               searchConfiguration: id
            })
         }
      } else {
         // create new item and new result
         const newItem = ItemDao.create(item);
         newResult = SearchResultDao.create({
            item: newItem.id,
            searchConfiguration: id
         })
      }
      if (newResult) {
         newItems.push(item);
      }
   }
   for (let item of availableItems) {
      ItemDao.update({_id: value}, {available: false})
   }
   if (!this.newItems[id]) {
      this.newItems[id] = [];
   }
   this.newItems[id].push(...newItems);
}
```

Metoda `saveResults()` (viz kód výše) jako první krok nalezne všechny již nalezené dostupné předměty `availableItems` dané konfigurace. Tato informace bude sloužit k tomu, abychom na konci určili, které předměty jsou již nedostupné. Poté získá nalezené předměty přečtením souboru, který byl vytvořen *scraperem*. Tyto předměty postupně prochází. Pokud je URL nově nalezeného předmětu rovna URL některého z dostupných předmětů, je tento předmět z dostupných předmětů odstraněn.

Poté se nalezený předmět musí uložit do databáze. Nejprve se s pomocí URL zkontroluje, zda je již předmět nalezen. URL jsou totiž vždy unikátní. Pokud je předmět nalezen, musíme zjistit, zda je tento předmět jedním z výsledků dané vyhledávací konfigurace. Pokud ne, tak je vytvořen nový výsledek vyhledávání. Pokud předmět v databázi vůbec neexistuje, je vytvořen jak nový předmět, tak nový výsledek vyhledávání.

Pokud je vytvořen nový výsledek vyhledávání (i v případě, že předmět v databázi už existoval), je tento předmět evidován do seznamu nově nalezených předmětů (`this.newItems`). Tento seznam bude později sloužit k odeslání e-mailu s upozorněním na nově nalezené inzeráty.

Po projití všech nalezených předmětu se nakonec ještě vracím k seznamu s dostupnými inzeráty. Z tohoto seznamu by v tomto bodě měly být odstraněny všechny inzeráty, které jsou momentálně dostupné. Všechny, které v seznamu zbyly, tedy v databázi označím jako nedostupné.

Jakmile se uloží všechny výsledky všech doběhlých vyhledávacích konfigurací (vracím se ke *callbacku* v metodě `execScript`), musí se ještě odeslat upozorňující e-maily na nově nalezené inzeráty. K tomu, jak již jsem zmínila dříve, slouží proměnná `this.newItems`. V této proměnné jsou uloženy ID konfigurací a k nim seznam nově nalezených inzerátů. Metoda `generateEmails()` postupně všechny tyto předměty projde a vytvoří jednoduchý HTML text. Poté se e-mail odešle na adresu uživatele, kterému patří daná konfigurace.

Tímto je proces *cronu* dokončen, proměnná `this.running` se znovu nastaví na `false` a nově nalezené inzeráty se vymažou. Tím jsou proměnné připraveny na další běh.

Toto řešení bude chtít do budoucna ještě upravit. Jednou z hlavních úprav bude odstranit proměnnou `this.running` a místo toho přidat tento indikátor k dané konfiguraci do databáze a pokaždé zkontrolovat, zda zrovna u dané konfigurace neprobíhá hledání, pokud ano, tak ji přeskočit.

4.1.2.2 Scraper

Pro každý server (nyní pouze Bazoš a SBazar) existuje vlastní Python *scraper* skript (`bazos-scraper.py` a `sbazar-scraper.py`). Tyto skripty jsou si velmi podobné, do budoucna se počítá s úpravou, kdy bude jeden *base* skript, ze kterého budou dědit metody ostatní skripty. Pro ukázky kódu zde budu využívat pouze soubor `bazos-scraper.py`.

Tyto skripty se spouští s argumentem, kterým je ID dané vyhledávací konfigurace a to například příkazem: `python3 bazos-scraper.py 1`.

Ve skriptu se zavolá funkce `run()`. Ta se podívá do adresáře `resources`, ve kterém najde `.csv` soubor, který svým názvem odpovídá argumentu, tedy ID konfigurace. Tento soubor byl, jak již jsem zmiňovala v sekci 4.1.2.1, vytvořen metodou `createScraperResourcesCsv()`. Obsah souboru víceméně odpovídá tomu, jak se zapisují hledané výrazy na *front-endu* (viz sekce 4.1.1.3), a může vypadat například takto:

```
apple;watch;-koupim
rozbit;praskl;nefun
```

Z tohoto souboru si potom skript vygeneruje všechny potřebné vyhledávací *query*. Ty se generují specificky pro každý server, neboť vyhledávací URL může vypadat rozdílně. Bazoš například pro spojení výrazů používá znak „+“, SBazar zase znak „%20“. Pro Bazoš jsou tedy jednotlivé vygenerované *query*:

- apple+wath+-koupim+rozbit
- apple+wath+-koupim+praskl
- apple+wath+-koupim+nefun

Pro každou z těchto *query* se potom složí celá URL, která je argumentem pro funkci `crawler()`. URL se podobně jako *query* skládá pro každý server odlišně. Viz:

```
BASE = "https://www.bazos.cz/"
SEARCH_QUERY_A = "search.php?hledat="
SEARCH_QUERY_B = "&rubriky=www&hlokalita=&humkreis=25
                  &cenaod=&cenado=&kitx=ano"

for query in queries:
    url = BASE + SEARCH_QUERY_A + query + SEARCH_QUERY_B
    crawler(url)
```

Funkce `crawler()` (viz kód níže) funguje na principu fronty, do které se nejprve přidá URL z parametru funkce. Fronta se postupně prochází cyklem `while`. Každá URL zařazená do fronty potom reprezentuje další stránku z výsledků vyhledávání. Pokud se tedy na stránce nachází HTML element s atributy, které jsou použity pro reprezentaci stránkování, přidá se do fronty další stránka.

Z každé této stránky se pomocí funkce `get_items()` získají všechny inzeráty zobrazené na stránce. Ty se potom `for` cyklem postupně prochází za použití funkce `sleep()`, která již byla zmíněná v sekci 3.1.4. K získávání dat ze stránky z již konkrétním inzerátem se používá funkce `parse_item_page()`.

```
def crawler(seed):
    frontier = [seed]
    while frontier:
        page = frontier.pop()
        try:
            req = requests.get(page)
            source = req.text
            soup = BeautifulSoup(source, "html5lib")
            items = get_items(soup)
            for item in items:
                random = randint(500, 1500) / 1000
                sleep(random)
                parse_item_page(item)

            pagination = soup.find(class_="strankovani")
            if pagination is not None:
                pagination = pagination.select("a")
                for p in pagination:
                if p.text == "Dalsi":
                    frontier.append(BASE_QUERY + p['href'])
```

Tato funkce sbírá data z jednotlivých inzerátů, která poté vkládá do globální proměnné `items`. Po prohledání všech výsledků pomocí vygenerovaných *query* se potom všechny výsledky uloží v JSON formátu do souboru `data/bazos/id.json`. S těmito daty dále pracuje *cron* zmíněný v předchozí sekci.

4.1.2.3 Zabezpečení

Zabezpečení aplikace stojí hlavně na principu použití tzv. *access tokenu* (přístupového tokenu). Jedná se o řetězec znaků, který je vygenerován pomocí soukromého tokenu, který se nachází na serveru. Součástí generování tokenu je i uložení jistých informací právě do něj. Tou informací je například ID uživatele nebo expirace, tedy doba platnosti tokenu.

Pro generování a ověřování tokenu je využita knihovna `jsonwebtoken`, konkrétně metody `sign()` a `verify()`. Tyto procesy potom probíhají v souboru `UserController.js`.

Přístupový token může klient získat přihlášením či registrací. Právě pokud úspěšně proběhne jedna z těchto dvou akcí, server vygeneruje přístupový token, který odešle v odpovědi klientovi, kde se uloží do *local storage* prohlížeče. Odtud je poté při každém dotazu na server přidáván do hlavičky dotazu.

Při každém dotazu na server je poté kontrolováno, zda přístupový token odpovídá generování právě soukromým tokenem (viz kód níže). Pokud ano, do hlavičky se přidá uživatel odpovídající příchozímu tokenu a komunikace může dál pokračovat. Pokud ne, server klientovi odešle zpět odpověď s informací,

že platnost tokenu vypršela, a uživatel je odhlášen. V některých případech (přihlášení a registrace) se může stát, že přístupový token v hlavičce vůbec není (protože nebyl ani v *local storage* prohlížeče), to není problém, avšak na ostatní cesty se klient nedostane.

```
authorize(req, res, next) {
   let token = req.headers["authorization"];
   if (token) {
      try {
         req.user = jwt.verify(token, TOKEN_SECRET);
         next();
      } catch (err) {
         res.status(401).send({
            errorMessage: "Token expired."
         });
      }
   } else {
      next();
   }
}
```

4.2 Nasazení

Nasazení aplikace probíhalo s pomocí zadavatele, který poskytl server, na kterém už bylo celé potřebné prostředí. Bohužel po prvním nasazení aplikace nefungovala tak, jak bylo očekávané. Nejspíše nastal někde nějaký problém a odesílalo se moc dotazů na stránku Bazoš, což zapříčinilo permanentní zablokování IP adresy serveru ze strany Bazoše. Na podporu jsem odeslala e-mail s žádostí o odblokování, ale to je zatím bez odezvy.

Jako druhý pokus jsem si koupila svůj vlastní server, který jsem si musela nakonfigurovat podle potřeb mé aplikace. Prvotní kroky pro přípravu serveru k nasazení byly:

1. Koupit VPS server (Wedos[25]) a přihlásit se na něj přes SSH pod uživatele `root` pomocí přidělených údajů.

2. Aktualizace systémových knihoven a celého systému (příkaz `apt update` a `apt ugrade`).

3. Vytvoření složky `.ssh` v domovském adresáři uživatele `root` se souborem `authorized_keys`, ve kterém se nachází veřejný SSH klíč zařízení, na kterém je systém vyvíjen.

[25] `https://www.wedos.cz/`

4. Vytvoření nového uživatele *deployer* (příkaz `adduser deployer`).
5. Přidání sudo[26] práv uživateli *deployer* (`usermod -aG sudo deployer`).
6. Vytvoření složky `.ssh` v domovském adresáři uživatele *deployer* se souborem `authorized_keys`, ve kterém se nachází veřejný SSH klíč zařízení, na kterém je systém vyvíjen.
7. Zrušení přihlášení heslem (úprava souboru `/etc/ssh/sshd_config`).
8. Vytvoření adresáře `/var/www/html/market-bot.cz` se složkami `client` a `server`, do kterého bude nasazena aplikace.
9. Instalace potřebných závislostí na server
   - MongoDB, Node, NPM, Python,
   - PM2[27] – drží při životě aplikaci,
   - knihovny, které jsou využity v serverové části aplikace – dostupné v souboru `package.json`.
10. Vytvoření souborů `.env` s proměnnými určenými pro produkční prostředí.

Nyní je server předpřipraven k nasazení. Byl vytvořen i uživatel *deployer*, pod kterým budeme aplikaci na server nasazovat.

Nasazení klientské části probíhá ze složky `client` na lokálním zařízení, a to zavoláním skriptu `npm run deploy` (definován v souboru `package.json`). Tento skript se skládá ze dvou po sobě jdoucích příkazů:

1. `npm run build`
   - používá příkaz `react-build` z knihovny `react`,
   - vytvoří složku `build`, ve které se nachází sestavená aplikace.
2. `scp -r ./build/* deployer@myserver:/.../market-bot.cz/client`
   - používá SSH připojení na server přes uživatele *deployer*,
   - zkopíruje složku `build` na server.

Serverová část aplikace se nasazuje obdobně skriptem `npm run deploy`, který se skládá ze třech příkazů:

1. `npm run build`

[26] zkratka ze Substitute User Do, jedná se o program, který na určitý čas přidá uživateli práva `root` uživatele[40]

[27] https://pm2.keymetrics.io/

- používá JavaScriptový kompilátor Babel[28] (překládá Node.js JavaScript do čistého JavaScriptu, čímž je zajištěno správné fungování aplikace ve všech prohlížečích[41]),
- taktéž vytvoří složku `build` se sestavenou aplikací.

2. `scp -r ./build/ deployer@myserver:/.../market-bot.cz/server/`
3. `ssh deployer@myserver pm2 reload MarketBot`
   - příkaz `pm2 reload` restartuje aplikaci.

Celá aplikace je dostupná na URL adrese `https://market-bot.cz/`. Zdrojové kódy jsou potom dostupné na přiloženém paměťovém zařízení.

4.3 Testování

Nedílnou součástí vývoje software je i fáze testování. Ta má za úkol odladění běhu aplikace pomocí odhalování chyb a ověřování správné funkčnosti systému. Chyby obvykle vznikají nejen ve fázi implementace, ale už během analýzy a návrhu, testování by tedy mělo být zahájeno co nejdříve[42].

4.3.1 Automatické testy

Automatické testování je založeno na principu napsání dalšího kódu, který bude kontrolovat správnost fungování aplikace. Hlavním rozdílem oproti manuálním testům je ten, že výstupem je rovnou informace, zda test prošel či nikoli, a nic se tedy nemusí ručně procházet[43].

Pro potřeby této práce byly napsány automatické testy pro ověření funkčnosti *scraperu*. Již dříve jsem totiž zmiňovala jednu z nevýhod použití *scraperu* oproti API, kterou je možná změna HTML struktury procházených stránek. Proto je potřeba jednou za čas ověřit, že struktura stránky a atributy jako ID nebo třídy zůstaly nezměněny.

```
#!/bin/sh
echo Running bazos html test
python3 test-html-bazos-scraper.py bazos
echo Running sbazar html test
python3 test-html-sbazar-scraper.py sbazar
```

Testy jsou k nalezení ve složce `server/src/scraper/test`. Pro každý server (tedy Bazoš a SBazar) je napsán samostatný test. Oba testy společně se dají spustit právě z této složky, a to pomocí skriptu `run_html_tests.sh`. Jedná se o jednoduchý skript (viz kód výše), který spustí Python programy, jež mají podobnou strukturu jako samotné *scraper* skripty. Ty ale se ale pouze

[28] https://babeljs.io/

snaží vytáhnout informace ze stránek, nikoli ukládat výsledky. Pokud se nepovede nějakou informaci získat, pomocí výstupu v konzoli jsme upozorněni na počet chyb a jaké chyby případně nastaly.

Dále jsou ve složce předpřipraveny testy pro ověření správnosti získaných informací. Ty jsou spuštěny skriptem `run_data_tests.sh`. Tyto testy zatím fungují pouze staticky, jedná se víceméně o kopii *scraper* skriptů a mají porovnávají výsledky nalezené na stránkách s výsledky, které by měly být nalezeny.

4.3.2 Uživatelské testy

Uživatelské testování neprobíhalo podle žádných scénářů a nebylo nikterak strukturované. Jedná se pouze o souhrn několika poznatků, které jsou buď chybou aplikace, nebo možným vylepšením do budoucna. Tyto poznatky byly nasbírány testováním se zadavatelem na již nasazené aplikaci a celý proces testování probíhal asi tři týdny. Testování bude probíhat i nadále, stejně jako oprava chyb či implementace nových funkcí.

Zde je v bodech vypsán souhrn poznatků:

- V e-mailu s upozorněním by mohly chodit statistiky týkající se počtu nově nalezených inzerátů. Stejně tak by mohly být tyto statistiky zobrazeny v přehledu vyhledávacích konfigurací.
- Společně s lokalitou ukládat i zeměpisné souřadnice, které by se mohly zanést na mapu zobrazovanou u výsledků.
- *Scrapery* jsou moc pomalé. Pokud je inzerátů na daný výraz hodně, může hledání trvat i v řádu hodin, je tedy momentálně potřeba ověřit si počet inzerátů a zvolit rozumné kombinace výrazů. Zadavatel zvažuje předělat *scrapery* tak, aby nevyhledávaly dál, pokud narazí na inzerát, který už je obsažen ve výsledcích dané vyhledávací konfigurace. Takto by ale nebylo možné označovat inzeráty jako nedostupné, tudíž je potřeba zvážit výhody a nevýhody těchto dvou způsobů, či vymyslet ještě nějaký jiný postup vyhledávání.
- K jednotlivým inzerátům přidat možnost napsání poznámky.
- Řadit výsledky podle pole *Item.createdAt* a ne podle data (na Bazoši ani SBazaru není dostupný čas přidání inzerátu).
- Přidat k výsledkům ikonu, která by signalizovala, že uživatel na inzerát již zareagoval.
- Pokud vyprší platnost přístupového tokenu, tak ho odstranit z *local storage*, aby byl člověk přesměrován na přihlašovací stránku.
- Integrovat do aplikace tlačítko zpět (viz prototyp 3.3.3).

- Přidat potvrzení registrace uživatelem, aby se nemohl jen tak někdo registrovat. Jednalo by se o přidání uživatele s extra právy (například *superadmin*), který bude moci spravovat registrované uživatele.
- Oblíbený inzerát se nyní objeví u všech uživatelů jako oblíbený. Nyní to nevadí, protože systém využívá primárně zadavatel, tedy jeden uživatel, ale pokud bude do budoucna více uživatelů, je nutné to opravit.
- Pokud se zobrazuje nějaká vyhledávací konfigurace, bylo by potřeba ošetřit, aby se nedala zobrazit jiným uživatelem. Uživatel se sice přímo sám od sebe z aplikace na cizí konfigurace nedostane, protože nemá k dispozici jejich ID, ale kdyby ID měl, tak mu momentálně aplikace povolí přístup na danou stránku (ověřuje se pouze to, jestli přístupový token odpovídá některému z registrovaných uživatelů).
- Přidat k vyhledávací konfiguraci možnost nastavit si pro každý server jiné hledané výrazy. Momentálně je to řešeno tak, že si uživatel musí nastavit dvě různé vyhledávací konfigurace, jednu pro Bazoš a druhou pro SBazar. Důvodem je odlišný princip vyhledávání na těchto dvou webech (viz sekce 2.2.1.1 a sekce 2.2.2.1).
- Získávat ze stránek i informaci, zda se jedná o „topovaný“ inzerát.

Kapitola 5

Ekonomicko-manažerské shrnutí

V této kapitole se budu věnovat zhodnocení, jak je aplikace ekonomicky výhodná pro zadavatele. Cílem projektu totiž bylo hlavně zjednodušení a zefektivnění jeho práce. Velkou roli v tomto zefektivnění hraje čas strávený nad vyhledáváním. Proto jsem se pro celkové zhodnocení výhod aplikace zaměřila právě na měření času, který zadavatel ušetří využíváním nového řešení.

Dále se chci zaměřit i na rozšiřitelnost aplikace nejen pro potřeby klienta, ale jako potenciální podnikatelský produkt. Popíši tedy, v jakých odvětvích by bylo použití automatizovaného vyhledávače vhodné. Pro případnou budoucnost projektu sestavím i podnikatelský plán.

5.1 Měření času

V následujících tabulkách (viz 5.1 a 5.2) můžeme vidět časy, které zadavatel stráví manuálním prohledáváním jednotlivých bazarů. Časy byly měřeny celkem třikrát pro každý server, přičemž měření probíhalo vždy po časovém úseku pěti hodin. Časy jsou uvedeny v minutách a sekundách, přičemž je počítán i čas strávený prohlížením samotných inzerátů a procházením obrázků. Výkyvy v měření souvisí tedy s tím, kolik inzerátů zaujalo zadavatele při daném měření.

Zároveň se při měření několikrát stalo i to, že zadavatel omylem procházel znovu ten stejný inzerát, což je způsobeno tím, že na těchto serverech uživatel nemá k dispozici žádnou informaci o tom, zda už inzerát viděl.

Z výsledků měření můžeme vypozorovat, že čas strávený jedním vyhledáváním na těchto dvou serverech je dohromady průměrně šestnáct minut. Pokud by tedy zadavatel vyhledával čtyřikrát denně, strávil by tak pouhým hledáním jednu hodinu a čtyři minuty denně.

Měření času stráveným hledáním pomocí aplikace MarketBot proběhlo celkem třikrát. Je zaznamenáno v tabulce 5.3. Měření probíhalo společně se zadavatelem, a to projitím e-mailových upozornění z celkem tří dnů, kdy se

Číslo měření	iPhone	Apple Watch	MacBook	iPad	Celkem
1	6:15	1:52	1:56	0:30	10:33
2	4:03	1:01	2:36	0:44	8:24
3	3:51	1:33	1:43	1:23	8:30
Průměr	4:43	1:29	2:05	0:52	9:09

Tabulka 5.1: Bazoš – měření času stráveného manuálním vyhledáváním

Číslo měření	iPhone	Apple Watch	MacBook	iPad	Celkem
1	1:52	1:28	1:30	2:06	6:56
2	2:34	0:56	1:13	1:45	6:28
3	3:03	1:23	1:46	0:58	7:10
Průměr	2:30	1:16	1:30	1:36	6:51

Tabulka 5.2: SBazar – měření času stráveného manuálním vyhledáváním

Číslo měření	Doba vyhledávání
1	4:35
2	2:54
3	3:10
Průměr	3:33

Tabulka 5.3: Měření doby vyhledávání pomocí aplikace MarketBot

každý den rovnal jednomu měření. Zadavatel vždy otevřel danou e-mailovou zprávu s upozorněním a pokud ho zaujal název a cena nalezeného inzerátu, věnoval čas i přesměrování na stránku s inzerátem a získání dalších informací z popisu nebo fotek. Takto byla postupně projita všechna upozornění, která v daný den dorazila na zadavatelovu e-mailovou adresu. Následně se z těchto tří měření vytvořil průměr, který činí tři minuty a třicet tři sekund na den.

Pro získání konečného zhodnocení a zjištění, zda je aplikace pro zadavatele ekonomicky výhodná, je potřeba porovnat rozdíl mezi manuálním a automatizovaným vyhledáváním. Ten činí celkem jednu hodinu denně (v případě, že by zadavatel manuálně vyhledával čtyřikrát denně). Pokud by zadavatel jednu hodinu své práce ohodnotil pět seti korunami českých, za týden by tedy prodělal tři a půl tisíce korun. Na měsíc to činí celkem patnáct tisíc korun.

Do měření se navíc nepočítá to, o kolik výhodných inzerátů a nabídek může zadavatel přijít. Dříve se mu pravidelně stávalo, že reagoval na inzerát, který již byl prodaný, ale pouze nebyl odstraněn inzerentem. Nyní je ve většině případů prvním zájemcem reagující na inzerát, což zadavateli zvyšuje celkový finanční obrat. Celkově tedy můžu zhodnotit, že systém je pro zadavatele velkým ekonomickým přínosem.

Pro měření manuálního vyhledávání byly využity následující výrazy:

iPhone

- Bazoš:
  - iphone -koupim nefun
  - iphone -koupim praskl
  - iphone -koupim rozbit
  - iphone -koupim utop
- Sbazar:
  - iphone nefunkcni
  - iphone nefunguje
  - iphone praskly
  - iphone prasklina
  - iphone utopeny

Apple Watch

- Bazoš:
  - apple watch -koupim rozbit
  - apple watch -koupim praskl
  - apple watch -koupim nefun
- Sbazar:
  - apple watch rozbity
  - apple watch praskly
  - apple watch prasklina
  - apple watch nefunkcni
  - apple watch nefunguje

MacBook

- Bazoš:
  - macbook -koupim praskl
  - macbook -koupim rozbit
  - macbook -koupim nefun
  - macbook -koupim utop
  - macbook -koupim nejde
- Sbazar:
  - macbook praskly
  - macbook rozbity

  - macbook nefunkcni
  - macbook nefunguje
  - macbook utopeny
  - macbook nejde

iPad

- Bazoš:
  - apple ipad praskl
  - apple ipad rozbit
  - apple ipad nefun
  - apple ipad utop
- Sbazar:
  - ipad praskly
  - ipad rozbity
  - ipad nefunkcni
  - ipad nefunguje
  - ipad utopeny

5.2 Budoucnost projektu

Aplikace má velký potenciál i v jiných oblastech, než je hledání elektrotechniky. Výsledek diplomové práce by měl sloužit hlavně pro potřeby zadavatele, ale při návrhu a implementaci jsem se snažila zohlednit právě možnost rozšiřitelnosti do budoucna i o jiné internetové obchody, ať už bazarové či nikoli.

Celá tato myšlenka v sobě nese velký potenciál výdělku. Ten by mohl proudit z placení za konfiguraci vyhledávání (tedy za jednoho chytrého hlídacího psa), nebo také za založení prémiového účtu, který by oproti běžnému měl nějaké výhody. Prémiový účet by mohl být nabízen na týden zdarma a byl by navázán na číslo kreditní karty, aby se nestávalo, že si lidé zakládají prémiové účty stále dokola.

V aplikaci by ale bylo potřeba udělat několik změn, které by se týkaly hlavně přidání dalších skriptů pro prohledávání na jiných serverech. Pro výdělek by bylo zase potřeba udělat několik změn v oblasti uživatelských účtů, tedy přidat možnost registrace prémiového uživatele. Protože by se za použití služby platilo, musela by se i přidat platební brána. Změn by bylo určitě potřeba udělat mnohem více, proto by bylo potřeba provést novou analýzu a návrh změn, které by musely být implementovány do současného řešení.

Další změnou by mohlo být přidání možnosti nastavení typu či kategorie vyhledávaného zboží při tvorbě nové konfigurace. To by rozšířilo vyhledávání

o určité parametry. Hledání elektroniky, ke kterému je aplikace současně určena, by měla například typ různé či jiné, protože se jedná o typ zboží, pro které není lehké zavést specifické parametry.

Zde uvedu příklady konkrétních odvětví nebo typů zboží, které bych do budoucna chtěla brát v potaz. S nimi uvedu o případné nové parametry, které by lépe specifikovaly vyhledávání v daných oblastech. Pro tyto odvětví je ale potřeba udělat větší rešerši, zda již neexistují řešení na podobné bázi.

Nemovitosti jsou odvětvím, ve kterém vidím velmi velký potenciál, protože v současné době poptávka převyšuje nabídku a ceny nemovitostí rostou nahoru[44]. Lidé shánějí dům v rozpětí několika let, takže si myslím, že by lidé rádi zaplatili za možnost nastavení chytrého hlídacího psa. Zároveň by bylo zajímavé zacílit se na realitní makléře. V implementační části by se jednalo přidání serverů pro vyhledávání nemovitostí, jako jsou `www.sreality.cz`,`www.bezrealitky.cz`, `www.reality.cz` a podobně. Zároveň by se zohlednily parametry:

- zda vyhledáváme byt, dům či třeba kancelářské prostory,
- velikost užitné plochy,
- rozvržení místností (2kk, 3+1,…),
- zda je součástí zahrada, terasa nebo balkon či
- jestli je nemovitost po rekonstrukci nebo před rekonstrukcí.

Ojetá vozidla jsou dalším odvětvím, ve kterém je momentálně velmi zvýšená poptávka kvůli nedostatku nových vozů. Rostou tím pádem i ceny ojetých aut, kterých na trhu začíná být také nedostatek[45]. V implementační části by se jednalo o přidání serverů pro vyhledávání ojetých vozidel, jako jsou `https://www.sbazar.cz/`, `https://www.autobazar.cz/` a mnoho dalších (třeba i těch německých). Parametry pro vyhledávání by byly:

- značka,
- model (případné další specifikace jako řada),
- nájezd kilometrů,
- typ paliva,
- rok výroby,
- vybavení vozidla (klimatizace, vyhřívání sedaček atd.) nebo
- zda má auto manuální či automatické řazení,

Obrázek 5.1: Lean Canvas

5.2.1 Podnikatelský plán

Podnikatelský plán shrnuje a lépe formuluje podnikatelský záměr. Jedná se o dokument, který by se měl vytvářet před počátkem samotného podnikání a měl by poskytovat ucelený obraz. Součástí je například i zamyšlení nad konkurenční výhodou, kdo bude zákazníkem či jaká bude struktura výdajů a příjmů. Na jeho základě by se mělo rozhodnout, zda má podnikání smysl a zda se bude projekt dále realizovat[46].

Tento podnikatelský plán je založen na plánu, který vznikl při tvorbě semestrálního projektu do předmětu NI-TSW ve spolupráci s Bc. Hanou Fukalovou, Bc. Rostislavem Babáčkem, Bc. Martinem Kupkou a Bc. Janem Štefánikem.

5.2.1.1 Podnikatelský model

Pro sestavení podnikatelského modelu byla využita technika *Lean Canvas*. Jedná se plánovací techniku, jejímž cílem je rychlé a přehledné sestavení plánu. Plánuje se pomocí modelu, který je reprezentován maticí s devíti různými oblastmi (viz obr. 5.1). Jednotlivé oblasti jsou[47]:

Problém představuje důvody, proč by mohli být zákazníci nespokojení s dosavadním řešením a jaké alternativy jsou pro dosažení cílů využívání. V našem případě existují následující problémy:

- nutnost prohledávat různé bazary,

- nemožnost nastavení chytrého hlídacího psa a
- neexistence automatizovaného vyhledávače pro internetové bazary (Bazoš, SBazar, Aukro atd.).

Řešení je způsob, jak může být vypořádáno s uvedenými problémy pomocí našeho produktu. Řešením je tedy návrh a realizace webové aplikace, která uživateli umožní lépe specifikovat, jaké zboží hledá. Zároveň je schopna zákazníka notifikovat, pokud se dané zboží objeví na jednom z hlídaných bazarů, případně i kontaktuje prodejce a zboží zarezervuje.

Unikátní přidaná hodnota říká, co zákazník získá používáním našeho produktu oproti použití alternativ. Výhody pro zákazníka tedy můžou být:

- zákazník nemusí ručně vyhledávat na různých internetových bazarech,
- informování zákazníka téměř ihned, jakmile je vyhledávaný produkt přidán na některý z hlídaných serverů, a
- eliminace času stráveného vyhledáváním zboží.

Segmenty zákazníků definují, kdo jsou budoucí uživatelé produktu, tedy kdo si náš produkt či službu potenciálně koupí. Také je potřeba rovnou specifikovat, kteří uživatelé budou využiti jako tzv. první vlaštovky (jedná se o segment, díky kterému bychom na trhu prorazili). Konkrétními segmenty můžou být:

- provozovatelé opraven elektroniky,
- realitní agentury a makléři,
- autoservisy či autobazary (ideální první vlaštovky, jelikož je tento segment zákazníků velmi velký),
- starožitníci, restaurátoři a sběratelé.

Cesty k zákazníkům specifikují způsoby, jakými je možné oslovit jednotlivé segmenty zákazníků. Mezi tyto cesty můžeme zahrnout třeba:

- sociální sítě,
- internetovou reklamu,
- reklamu v tisku nebo
- přímý kontakt s autoservisy, se starožitnictvími atd.

Klíčové metriky jsou způsob, kterým se dá hodnotit úspěšnost či neúspěšnost produktu. Náš produkt můžeme hodnotit například:

- počtem reálných uživatelů, zadaných hlídacích psů a jejich výsledků,
- kolik se realizovalo obchodů pomocí platformy,

- kolik zákazníků se rozhodlo přejít z bezplatné verze na placenou službu či
- jaké jsou realizované zisky a zda se plní finanční plán.

Struktura nákladů popisuje počáteční náklady na zavedení produktu a dále náklady potřebné na provoz a údržbu:

- analýza zákazníků a návrh změn v aplikaci na základě jejich požadavků,
- implementace změn a celkové testování aplikace,
- nasazení aplikace, její chod a údržba (hlavně nákup dostatečně výkonného serveru, který bude zvládat nápor zákazníků),
- marketing a
- zákaznická podpora.

Proud příjmů naopak určuje, jaké by měly být zisky ze zavedení produktu. Mělo by se specifikovat, z čeho budou příjmy plynout:

- určitá částka za jednoho hlídacího psa,
- využití prémiového účtu za měsíční poplatek a
- reklama.

5.2.1.2 Harmonogram

Předpokládám, že aplikace by měla být nasazena do devíti měsíců. Přínosné pro vývoj aplikace by bylo spuštění neplacené beta verze která by mohla pomoci opravit některé zásadní chyby ještě předtím, než budou zákazníci za službu platit. Poté počítám s pravidelnou údržbou a případně dalším rozšiřováním o jiné internetové bazary či další oblasti jako jsou třeba již zmíněné nemovitosti. Zde je v bodech uveden harmonogram:

1. průzkum trhu (1. měsíc),
2. analýza a návrh (2. a 3. měsíc),
3. implementace a testování (4. a 5. měsíc)
4. nasazení beta verze,
5. oprava na základě chyb hlášených z beta verze (6. až 9. měsíc),
6. nasazení plné verze,
7. servis a údržba.

5.2.1.3 Rizika

Pro podnikání je důležité stanovit, jaká jsou případná rizika. Projekt musí být na tyto rizikové situace připraven, ale hlavním cílem je se těmto rizikům vyvarovat, pokud je to možné. Ke každému riziku by tedy měly být uvedeny způsoby, kterými se dá danému riziku předejít (tzv. mitigace). Pokud by byla rizika příliš velká, podnikatel se na jejich základě může rozhodnout nepokračovat v projektu a předejít tak případným finančním ztrátám.

Mezi hlavní rizika se řadí následující situace:

- Uživatelé nejsou ochotni platit za nabízenou službu.
  - **Mitigace:** Provedení důkladného průzkumu trhu a následné zvolení té nejvhodnější ceny.
- Může nastat blokace ze strany bazarových serverů.
  - **Mitigace:** Diskuze s bazary o výhodách plynoucích z platformy pro obě strany.
- Hlídací pes nebude vracet dostatečně validní výsledky, a uživatelé nemusí být tím pádem spokojeni se službou.
  - **Mitigace:** Zvolení vhodné implementace vyhledávaných parametrů, optimalizovaná rychlost a časový interval prohledávání bazarových serverů. Nasazení beta verze.
- Nárůst zákazníků je moc pomalý.
  - **Mitigace:** Propracovaná marketingová kampaň zaměřená především na cílové skupiny.

Závěr

Cílem diplomové práce byla implementace aplikace pro automatizaci vyhledávání na více internetových bazarech najednou. Účelem bylo zjednodušení a zefektivnění práce zadavatele. Cíl práce byl splněn za pomoci celého procesu vývoje od analýzy současného řešení a sběr požadavků, přes návrh a implementaci, až po nasazení a testování konečné aplikace.

Zadavatel aplikaci aktivně využívá ke své práci a je velmi spokojen s výsledkem. Nicméně je potřeba doladit ještě několik funkcí či opravit pár chyb, které byly odhaleny v průběhu testování aplikace.

Aplikace má zároveň i velký potenciál pro budoucí vývoj a je potřeba zvážit, zda bude nebo nebude realizován podnikatelský plán, který byl sestaven v poslední kapitole práce. Tento plán by bylo určitě potřeba rozšířit a ještě více se zamyslet nad samotným podnikatelským záměrem a finančním plánem. To ale nebylo cílem této práce, tudíž této části není věnován tak velký prostor, jaký by mohl být.

Závěrem musím zhodnotit, že pro mě samotnou byla tato práce velkým přínosem. Během celého procesu jsem se naučila spousty novým věcem. Hlavně praktická část dostupná na přiloženém paměťovém zařízení mi prohloubila znalosti v oboru informačních technologií a posunula mě v tomto odvětví dál.

Literatura

[1] Apple: Informace o originálních displejích iPhonu. [online], březen 2022. Dostupné z: https://support.apple.com/cs-cz/HT210321

[2] TopRanker: Co je to fulltext / fulltextové vyhledávání? [online]. Dostupné z: https://topranker.cz/slovnik/fulltext-fulltextove-vyhledavani/

[3] Václavík, L.: Seznam už po Googlu sbírá jen drobky. Statistiky vyhledávání ukazují, že každý rok klesá. [online], duben 2021. Dostupné z: https://www.zive.cz/clanky/seznam-uz-po-googlu-sbira-jen-drobky-statistiky-vyhledavani-ukazuji-ze-kazdy-rok-klesa/sc-3-a-209811/default.aspx

[4] StrephonSays: Functional and non-functional requirements. [online]. Dostupné z: https://cs.strephonsays.com/functional-and-non-functional-requirements-3325

[5] GrowHackScale: Search Queries. [online]. Dostupné z: https://growhackscale.com/glossary/search-queries

[6] Hanák, J.: Vysvětlení SSL certifikátů: Co jsou, jak fungují a proč je používat. [online], březen 2016. Dostupné z: https://www.master.cz/blog/co-jsou-ssl-certifikaty-a-ssl-protokoly-jak-funguji-vysvetleni-navod/

[7] ManagementMania: Třívrstvá architektura (Three-tier architecture). [online], prosinec 2015. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

[8] MongoDB: MERN Stack Explained. [online]. Dostupné z: https://www.mongodb.com/mern-stack

[9] GeeksForGeeks: Advantages and Disadvantages of Three-Tier Architecture in DBMS. [online], prosinec 2021. Dostupné z: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-three-tier-architecture-in-dbms/

[10] Sedláček, P.: Lekce 5 - Úvod do MongoDB. [online]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-mongodb

[11] ACodes: ALL ABOUT MONGODB: THE NOSQL DATABASE. [online], červenec 2019. Dostupné z: https://acodez.in/mongodb-nosql-database/Advantages_of_MongoDB

[12] ExpressJS. [online]. Dostupné z: https://expressjs.com/

[13] Volodymyr, T.: Express.js Mobile App Development: Pros and Cons for Developers. [online], červenec 2020. Dostupné z: https://apiko.com/blog/express-mobile-app-development/

[14] Community: Express/Node introduction. [online], duben 2022. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[15] Štráfelda, J.: Open source. [online]. Dostupné z: https://www.strafelda.cz/open-source

[16] Koďousková, B.: PROČ K VÝVOJI WEBOVÝCH APLIKACÍ POUŽÍT TECHNOLOGII NODEJS? [online], duben 2021. Dostupné z: https://www.rascasone.com/cs/blog/node-js-architektura-moduly-npm

[17] @chaitanyashah707: Node.js Callback Concept. [online], říjen 2021. Dostupné z: https://www.geeksforgeeks.org/node-js-callback-concept

[18] Simpli, L.: What is the event loop in node.js? [online], září 2019. Dostupné z: https://www.learnsimpli.com/what-is-the-event-loop-in-node-js/

[19] VIPTrust: Node JS. [online]. Dostupné z: https://viptrust.com/technologie/ostatni/node-js

[20] React: Getting Started. [online]. Dostupné z: https://reactjs.org/docs/getting-started.html

[21] Sidorenko, A.: React is Declarative - What Does it Mean? [online], září 2021. Dostupné z: https://alexsidorenko.com/blog/react-is-declarative-what-does-it-mean/

[22] Koďousková, B.: VÝVOJ WEBOVÝCH APLIKACÍ S REACTJS, NEBO VUEJS? [online], červen 2021. Dostupné z: https://www.rascasone.com/cs/blog/vyvoj-webovych-aplikaci-reactjs-vuejs

[23] Bazoš: Nápověda. [online]. Dostupné z: https://www.bazos.cz/napoveda.php

[24] Impreva: Web Scraping. [online]. Dostupné z: https://www.imperva.com/learn/application-security/web-scraping-attack/

[25] Breuss, M.: Beautiful Soup: Build a Web Scraper With Python. [online], červen 2021. Dostupné z: https://realpython.com/beautiful-soup-web-scraper-python/what-is-web-scraping

[26] Pai, S.: Web Scraping vs API: What's the best way to extract data. [online], srpen 2021. Dostupné z: https://www.blog.datahut.co/post/web-scraping-vs-api

[27] TechnologyHQ: What Programming Language Is Best for Your Web Scraping Project? [online], leden 2022. Dostupné z: https://www.technologyhq.org/what-programming-language-is-best-for-your-web-scraping-project/

[28] TheMightyProgrammer: What is Parsing? [online]. Dostupné z: https://themightyprogrammer.dev/article/parsing

[29] Soup, B.: Beautiful Soup Documentation. [online]. Dostupné z: https://beautiful-soup-4.readthedocs.io/en/latest/

[30] Čápka, D.: Lekce 1 – Úvod do UML. [online]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[31] Čápka, D.: Lekce 4 – UML – Doménový model. [online]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram

[32] Čápka, D.: Lekce 2 – UML – Use Case Diagram. [online]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[33] Indeed: What Is a User Interface? (Definition, Types and Examples). [online], září 2021. Dostupné z: https://www.indeed.com/career-advice/career-development/user-interface

[34] Pavlíček, J.: Lecture 3: UI design steps. [online]. Dostupné z: https://docs.google.com/presentation/d/1CldSHyC8D8cN2LGrqUVJ2U5eEKn5z9MpkFPmzwZX4Zc

[35] Dubinská, L.: CO JE WIREFRAME WEBU, PROČ HO POTŘEBUJETE A JAK HO VYTVOŘIT? [online], září 2021. Dostupné z: https://www.rascasone.com/cs/blog/co-je-wireframe-predstavujeme-5-duvodu-proc-je-pro-klienty-drateny-model-dulezity

[36] Raj, M.: How to Use Local Storage with JavaScript. [online], leden 2021. Dostupné z: https://www.section.io/engineering-education/how-to-use-localstorage-with-javascript/

[37] @vita: Lekce 7 - Wicket - Databáze. [online]. Dostupné z: https://www.itnetwork.cz/java/wicket/java-wicket-databaze

[38] Pilný, V.: Co je to cron a k čemu se používá? [online]. Dostupné z: https://www.hostingy.net/co-je-to-cron-a-k-cemu-se-pouziva/

[39] Pair: What is Cron and How do I Use It? [online], únor 2020. Dostupné z: https://www.pair.com/support/kb/configuring-cron/

[40] Sletton, E.: How to Set Up and Manage Sudo Permissions. [online], květen 2020. Dostupné z: https://www.liquidweb.com/kb/how-to-set-up-and-manage-sudo-permissions/

[41] Johnson, N.: What is Babel, and how will it help you write JavaScript? [online]. Dostupné z: http://nicholasjohnson.com/blog/what-is-babel/

[42] kITner: Co je testování softwaru? [online]. Dostupné z: https://kitner.cz/testovani_softwaru/co-je-testovani-softwaru/

[43] Pekař, L.: AUTOMATICKÉ TESTOVÁNÍ SOFTWARU – NEBOLI KÓD, CO KONTROLUJE KÓD. [online], říjen 2017. Dostupné z: https://bonsai-development.cz/clanek/automaticke-testovani-softwaru

[44] Zpasti: Jaký bude vývoj cen nemovitostí v roce 2022 v ČR. [online], leden 2022. Dostupné z: https://zpasti.cz/blog/vyvoj-cen-nemovitosti

[45] ČTK: Ceny ojetých aut letos mohou vzrůst až o patnáct procent. Čechy trápí jejich nedostatek. [online], leden 2022. Dostupné z: https://www.e15.cz/byznys/prumysl-a-energetika/ceny-ojetych-aut-letos-mohou-vzrust-az-o-patnact-procent-cechy-trapi-jejich-nedostatek-1386662

[46] Shoptet: Business plán. [online]. Dostupné z: https://www.shoptet.cz/slovnik-pojmu/business-plan/

[47] BezvaVejška: Lean Canvas a Business model Canvas – Inovace a podnikání. [online], červenec 2017. Dostupné z: https://bezvavejska.cz/lean-canvas-business-model-canvas-inovace/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

DAO Database Access Object

HTTP Hypertext Transfer Protocol

ID Identity

IDE Integrated Development Environment

IP Internet Protocol

JSON JavaScript Object Notation

MERN Mongo Express React Node

NPM Node Package Manager

PC Personal Computer

RAM Random Access Memory

SMS Short Message Service

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

SSL Secure Sockets Layer

SQL Structured Query Language

UC Use Case

UML Unified Modeling Language

URL Uniform Resource Locator

VPS Virtual Private Server

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého paměťového zařízení

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `wireframes.pdf` ..... wireframes v klikatelném PDF