VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PORTÁL PRO AGREGACI DAT Z WEBOVÝCH ZDROJŮ

PORTAL FOR AGGREGATION OF DATA FROM WEB SOURCES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TIBOR MIKITA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Mikita Tibor, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Portál pro agregaci dat z webových zdrojů**
Portal for Aggregation of Data from Web Sources
Kategorie: Web

Zadání:

1. Seznamte se s používanými způsoby prezentace informací ve veřejných webových portálech a s možnostmi extrakce dat z těchto portálů.
2. Prostudujte datové modely umožňující flexibilní reprezentaci dat s variabilní strukturou, chybějícími údaji a podobně. Zaměřte se i na reprezentaci sémantiky dat.
3. Po dohodě s vedoucím zvolte vhodnou aplikační doménu a navrhněte architekturu portálu pro shromažďování dat z heterogenních webových zdrojů a jejich jednotnou prezentaci.
4. Navržené řešení implementujte pomocí vhodných technologií.
5. Proveďte testování vytvořeného portálu s alespoň třemi různými zdroji dat.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Juneau, J.: Java EE 7 Recipes, Apress, 2013
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015
- Burget, R.: Information Extraction from HTML Documents Based on Logical Document Structure, Brno University of Technology, 2004

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Abstrakt

Táto práca sa zaoberá extrakciou a následnou agregáciou dát z heterogénnych webových zdrojov. Cieľom je vytvorenie portálu, funkčnej webovej aplikácie, využívajúcej vhodné technológie. Hlavné zameranie práce je na architektúru a samotnú implementáciu aplikácie. Za aplikačnú doménu bolo zvolené ubytovanie, resp. hľadanie prenájmu. Pre extrakciu dát sa využíva API portálu alebo wrapper. Získané dáta sa ukladajú do dokumentovej databázy. V tejto práci sa podarilo navrhnúť a implementovať systém, pomocou ktorého je možné získavať inzeráty s prenájmami bytov z viacerých webových zdrojov zároveň a tie jednotnou formou prezentovať používateľovi na jednom mieste.

Abstract

This thesis deals with data extraction and data aggregation from heterogeneous web sources. The goal is to create a platform and a functional web application using appropriate technologies. The main focus of the thesis is on the application design and implementation. The application domain is accommodation or lease of apartments. For the data extraction, we use the portal API or a wrapper. Obtained data is stored in a document database. In this thesis, we managed to design and implement a system that allows to obtain rental ads from multiple web sources at the same time and to present them in a uniform way.

Kľúčové slová

web, extrakcia dát, flexibilné dátové modely, heterogénne webové zdroje, zhromažďovanie dát, jednotná prezentácia dát, webový portál

Keywords

web, data extraction, flexible data models, heterogeneous web sources, data aggregation, uniform data presentation, web portal

Citácia

MIKITA, Tibor. *Portál pro agregaci dat z webových zdrojů*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Portál pro agregaci dat z webových zdrojů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána doktora Burgeta. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tibor Mikita
19. mája 2019

Poďakovanie

Týmto by som rád poďakoval pánu doktorovi Burgetovi za jeho odborné rady, cenné konzultácie a ľudský prístup, ktoré mi pri tvorbe tejto práce venoval.

Obsah

Kapitola 1

Úvod

Človek niekoľkokrát za život hľadá bývanie pre seba, prípadne pre celú svoju rodinu. Dnes existujú webové stránky, ktoré prostredníctvom inzerátov ponúkajú byty na prenájom či kúpu. Bývanie sa dá hľadať aj na sociálnych sieťach. Problémom je, že existuje veľa možností, kde hľadať bývanie. Človek tak musí sledovať niekoľko stránok naraz. Každá stránka navyše zobrazuje informácie iným spôsobom. To môže spôsobovať problémy, hlavne pri porovnávaní viacerých ponúk. Zároveň je potrebné rozlíšiť už videné ponuky od nových, prípadne vyfiltrovať tie, o ktoré už nie je záujem. Niekedy je užitočné ukladať si históriu komunikácie s majiteľom, resp. prenajímateľom.

Koncept realitných serverov pripomína trh s ponukou a dopytom. Podobne fungujú internetové obchody s rôznym tovarom. Mnoho ľudí dnes pri vyberaní elektroniky prv navštívi portál, ktorý agreguje viacero ponúk z rôznych zdrojov, resp. internetových obchodov. Tam si vyberú produkt na základe ceny, hodnotení či recenzií. Vedia si tak vybrať produkt, ktorý bude pre nich vhodný, a za dobré peniaze. Až potom, čo si na takejto stránke produkt vyberú, navštívia konkrétny internetový obchod, kde si tento produkt kúpia. Výhodou je, že zákazník má všetky potrebné informácie, dokonca aj cenové ponuky z rôznych internetových obchodov o danom produkte, na jednom mieste, a podľa toho vie nakúpiť najvýhodnejšie.

Niečo také by malo existovať aj pre trh s prenájmami bytov. Potenciálni nájomníci by videli ponuky na jednom mieste, mohli by ich medzi sebou porovnávať, filtrovať na základe parametrov a podobne. Mohli by si tak nájsť ponuku, ktorá by im vyhovovala, a nemuseli by tak niekoľko hodín denne stráviť sledovaním viacerých webových portálov zároveň. Všetky potrebné informácie, vrátane kontaktu na prenajímateľa, by si našli v jednotlivých inzerátoch. Nie je nutné navštíviť webovú stránku, kde bol inzerát uverejnený.

Táto práca je zameraná na návrh a implementáciu vyššie spomínanej aplikácie. Výstupom je funkčný webový portál agregujúci inzeráty s prenájmami bytov z rôznych webových zdrojov. Táto práca sa zaoberá aj praktickými problémami pri riešení tejto úlohy. Hlavné zameranie je na návrh, flexibilný dátový model a samotné prevedenie aplikácie.

V kapitole 2 je popísaný spôsob akým dnes fungujú webové aplikácie a aké sú trendy v tejto oblasti. Kapitola 3 sa zaoberá extrakciou dát z webu. Sú v nej popísané metódy, ktoré je možné použiť pre získavanie dát zo štruktúrovaných dokumentov. Kapitola 4 porovnáva SQL a NoSQL databázy, a venuje sa problematike dátového modelu. Kapitola 5 obsahuje analýzy existujúcich riešení. Rozoberá nielen portály agregujúce dáta z rôznych webových zdrojov, ale aj služby, ktoré uverejňujú inzeráty s prenájmami bytov a v akom formáte tieto služby prenášajú dáta o inzerátoch. Kapitola 6 sa venuje návrhu aplikácie. Predstavuje architektúru a popisuje jej komponenty. Implementačné detaily sú spomenuté v kapitole 7. Kapitola 8 rozoberá spôsob testovania a praktické problémy, ktoré počas testovania nastali.

Kapitola 2

Architektúra webových aplikácií

2.1 Model klient-server

Webové aplikácie používajú model *klient-server*. Tento model sa radí k architektúram distribuovaného výpočtu. Pracovné úlohy alebo výpočty sa rozdeľujú medzi poskytovateľov a žiadateľov. Poskytovateľ, inak nazývaný server, poskytuje nejaké zdroje alebo služby. Naopak žiadateľ, inak nazývaný klient, tieto zdroje alebo služby požaduje. [34]

Server je centralizované úložisko dát. Klientov môže byť viac, a každému je poskytnutá rovnaká prezentačná vrstva.

V prípade webových aplikácií je serverom tzv. webový server, ktorý poskytuje dokumenty. Klientom je webový prehliadač, prostredníctvom ktorého používateľ komunikuje so serverom a jednotlivé dokumenty získava.

Kód, ktorý je na strane klienta, beží vo webovom prehliadači a reaguje na vstupy od používateľa. Kód, ktorý je na strane servera, odpovedá na HTTP požiadavky.

2.2 Protokoly HTTP a HTTPS

Hypertextový prenosový protokol (angl. *Hypertext Transfer Protocol*, HTTP), je aplikačný protokol používaný na prenos *hypermédia*[1] medzi serverom a klientami WWW služby [13].

HTTP má niekoľko verzií. Väčšina webových stránok dnes používa verziu HTTP/1.1. Táto verzia bola predstavená už v roku 1997, a neskôr definovaná v roku 1999. Najnovšia verzia HTTP/2 bola predstavená v roku 2015. Tá je používaná už takmer tretinou všetkých webových stránok [49].

Pre protokol HTTP existuje rozšírenie, ktoré prináša bezpečnú komunikáciu medzi používateľovým počítačom a webovou stránkou. Jeho názov je zabezpečený hypertextový prenosový protokol (angl. *Hypertext Transfer Protocol Secure*, HTTPS). Troma hlavnými výhodami sú šifrovanie, integrita dát a autentifikácia. Pre šifrovanie používa protokol TLS (*Transport Layer Security*), a preto je často nazývaným protokolom *HTTP over TLS*. [15, 35]

[1] *Hypermédium* označuje nelineárne médium alebo dokument, ktorý obsahuje okrem textu a odkazov na iné texty aj videozáznamy, zvuky a grafiku

Medzi najpoužívanejšie metódy HTTP požiadaviek patria:

`GET` získanie zdroja,

`POST` vytvorenie nového zdroja,

`PUT` upravenie existujúceho zdroja,

`DELETE` vymazanie zdroja.

Existuje štandard, podľa ktorého je možné zistiť, či bola HTTP požiadavka úspešná, alebo nie, prípadne ďalšie informácie. Používajú sa na to tzv. stavové kódy, ktoré pozostávajú z troch číslic. Sú roztriedené do piatich hlavných kategórií. Ich prehľad je možné vidieť v tabuľke 2.1.

Prefix	**Význam**	**Príklady**
`1xx`	informácia	`100` – server dostal hlavičku a čaká na telo správy
`2xx`	úspech	`200` – všeobecná odpoveď pre úspešnú požiadavku `201` – nový zdroj bol úspešne vytvorený `202` – požiadavka bola prijatá, spracovanie ešte neskončilo `204` – podobne ako 200, ale server nevrátil žiadne dáta
`3xx`	presmerovanie	`301` – zdroj bol presunutý `304` – od poslednej požiadavky sa zdroj nezmenil
`4xx`	chyba na strane klienta	`400` – zlá požiadavka `401` – problém s overením identity `403` – zamietnutý prístup `404` – požadovaný zdroj neexistuje
`5xx`	chyba na strane servera	`500` – všeobecná chyba `503` – služba dočasne nedostupná napr. preťaženie servera

Tabuľka 2.1: Prehľad HTTP stavových kódov

2.3 Trendy

Jedným z dnešných trendov je architektúra orientovaná na služby (angl. *Service Oriented Architecture*, SOA). Služba je často opakujúca sa funkčná časť informačného systému, ktorá má špecifikovaný výstup. Môže sa skladať z ďalších menších služieb. Detaily implementácie služby sú jej používateľom neznáme a služba tak navonok pôsobí ako tzv. *čierna skrinka*. Používateľom je poskytnuté len štandardné rozhranie API (*Application Programming Interface*), pomocou ktorého prebieha komunikácia. [46]

Ďalším trendom je tzv. *single-page aplikácia*. Táto webová stránka sa skladá z individuálnych komponentov, ktoré môžu dynamicky a nezávisle meniť svoj obsah alebo podobu na základe interakcie s používateľom, a to takým spôsobom, že nedochádza k znovu-načítaniu celého dokumentu zo servera. [27]

Používateľské rozhranie pri takejto webovej stránke bohato využíva programovací jazyk JavaScript. Pre odosielanie asynchrónnych ale aj synchrónnych požiadaviek sa používa technológia AJAX (*Asynchronous JavaScript and XML*) [14] alebo protokol *Websocket* [12].

2.4 Porovnanie statických a dynamických webových stránok

Webové stránky môžu byť statické alebo dynamické. V začiatkoch internetu boli webové stránky len statické. Statické webové stránky sú jednoduché webové stránky, ktoré majú za úlohu niečo prezentovať. Ich obsah sa dynamicky nemení. Je to jednoduchý HTML dokument, ktorý nie je generovaný jazykom PHP. Takto vytvorená webová stránka je doručená používateľovi v rovnakej podobe, ako je uložená na serveri. Ak sa už obsah má zmeniť, mení sa priamo HTML dokument na serveri. [45]

Väčšina dnešných populárnych webových stránok je dynamická. Dynamické webové stránky menia svoj obsah dynamicky na základe údajov o používateľovi, času a podobne.

Rozlišujeme dva hlavné typy dynamických webových stránok podľa toho, na ktorej strane dochádza k skriptovaniu a vytváraniu obsahu.

Generovanie obsahu webovej stránky môže prebiehať na strane servera. Odpovede servera na požiadavky klienta sa líšia na základe rôznych parametrov a dát, ktoré klient posiela. Typ webového prehliadača, časové pásmo, stav databázy alebo servera vedia taktiež ovplyvniť štruktúru alebo formu výslednej webovej stránky. [26]

Tieto webové stránky sú často vytvorené za pomoci skriptovacích jazykov na strane servera, akými sú napríklad PHP, JavaScript, Go či Python.

Druhá možnosť je vložiť skripty priamo do HTML dokumentu a spúšťať ich na strane klienta pri načítaní webovej stránky. Týmto sa zabezpečí lepšia interakcia používateľa s webovou stránkou. Vďaka technológii DOM (*Document Object Model*) je možné meniť štruktúru, obsah, ale aj prezentačné vlastnosti všetkých elementov HTML dokumentu. [11]

2.5 Možnosti implementácie WEB API

Server často poskytuje WEB API pre klientov. WEB API je rozhranie, pomocou ktorého klienti komunikujú so službou, ktorú poskytuje daný server. Existuje viac možností vytvorenia tohto rozhrania. Medzi prvými technológiami v tejto oblasti bolo RPC (*Remote Procedure Call*) [6]. Kód alebo procedúra sa vykonáva na vzdialenom počítači. Syntax volania tejto procedúry pripomína volanie lokálnej funkcie – stačí poznať názov vzdialenej procedúry a jej argumenty. Prenášané dáta sa serializujú pomocou formátu XML alebo JSON.[2, 43]

Nástupcom RPC je SOAP (*Simple Object Access Protocol*), protokol pre výmenu správ medzi klientom a serverom. Využíva formát XML na serializáciu a protokol HTTP pre prenos dát. [24]

Medzi najpoužívanejšie technológie pre implementáciu webových rozhraní v súčasnosti patria REST API a GraphQL.

2.5.1 REST API

Web zažíval na konci 20. storočia rýchly vývoj. Počet používateľov a počet dokumentov rástol exponenciálne. Podľa niektorých odborníkov hrozil kolaps. Bolo potrebné vyriešiť problémy rozšíriteľnosti webu. Stanovili sa podmienky, resp. zásady, ktoré bolo potrebné začať dodržiavať pre zvrátenie vtedajšej situácie.

Nasledujúci text je spracovaný na základe informácií zo zdroja [25].

Zásady

Web je v prvom rade systém založený na architektúre klient-server (pozri sekciu 2.1). Bolo potrebné vymedziť, ktoré úlohy má vykonávať klient a ktoré server. Klient a server sa môžu vyvíjať samostatne, nezávisle, pomocou rôznych technológií, musia avšak dodržiavať jednotné rozhranie.

Komunikácia vo webovom priestore je založená na dodržiavaní jednotného rozhrania. Rozhranie je definované na základe niekoľkých princípov. Všetko, čo sa nachádza na webe, je zdroj alebo prostriedok, ktorý má svoju unikátnu adresu URI (*Uniform Resource Identifier*). Klienti nemanipulujú priamo s týmito zdrojmi, ale s ich reprezentáciami. Jeden zdroj môže mať viacero reprezentácií. Zmenu stavu určitého zdroja je možné docieliť pomocou požiadavky na server, a server sám rozhodne, či takejto žiadosti vyhovie alebo nie. Aktuálny stav zdroja je možné získať z odpovede servera taktiež pomocou požiadavky. Jednotlivé požiadavky už vo svojich správach, prípadne v ďalších metadátach, obsahujú všetky potrebné informácie a detaily, ktoré server potrebuje na posúdenie konkrétnej požiadavky klienta. Stav aplikácie je daný hypermédiom. Hypermédium je súčasťou odpovede servera na požiadavku klienta. Obsahuje odkazy k relevantným, resp. súvisiacim zdrojom. Predpokladá sa, že každý odkaz je implementovaný pomocou REST a klient sa teda dokáže ľahko orientovať v danej službe.

Požiadavka na vrstvený systém umožňuje sieťovým zariadeniam transparentné nasadenie medzi klientom a serverom použitím jednotného rozhrania. Medzi takéto sieťové zariadenia môžu patriť proxy, brány alebo iné služby, ktoré zaisťujú bezpečnosť, použitie vyrovnávacej pamäte a vyrovnávanie záťaže (angl. *load balancing*).

Dôležitou súčasťou webu je používanie vyrovnávacej pamäte. Zrýchľuje prenos dát medzi klientom a serverom, a tým zvyšuje dostupnosť a spoľahlivosť služieb. Vyrovnávacia pamäť môže byť dostupná na niekoľkých úrovniach.

Stav aplikácie je na strane klienta. Server nemá povinnosť udržiavať stav aplikácie, pamätať si minulé požiadavky. Pri posielaní požiadaviek je úlohou klienta poskytnúť serveru všetky relevantné informácie, celý kontext.

Posledná zásada je voliteľná. Ide o tzv. kód na požiadanie. Server môže dočasne poskytnúť klientovi skripty, ktoré môže spúšťať. Ak klient chce takýto skript spustiť, musí rozumieť kódu, ktorý mu server zaslal. Tým medzi nimi nastáva závislosť, čo môže spôsobiť narušenie zásady oddeliteľnosti klienta od servera.

Definícia

REST (*Representational State Transfer*) je webová architektúra, ktorá spĺňa vyššie uvedené zásady. Oproti RPC či SOAP je REST orientovaný dátovo, nie procedurálne. Dátami myslíme zdroje, ktoré môžeme zo servera získať, a s ktorými môžeme manipulovať. Vyjadrujeme nimi stav aplikácie. Webové API, ktoré odpovedá architektúre REST sa nazýva REST API. Služba, ktorá poskytuje REST API ako svoje rozhranie, má označenie *RESTful*.

Typy zdrojov

Existuje niekoľko typov zdrojov: dokument (angl. *document*), kolekcia (angl. *collection*), sklad (angl. *store*) a radič (angl. *controller*).

Dokument predstavuje nejaký objekt alebo databázový záznam. Kolekcia je spravovaná na strane servera a obsahuje množinu dokumentov, resp. iných zdrojov. Sklad je na druhú stranu spravovaný na strane klienta. Do skladu je možné vložiť nejaké zdroje, prípadne

ich odtiaľ vybrať. Sklad žiadne zdroje priamo nevytvára, teda ani negeneruje nové URI. Avšak každý zdroj v sklade má svoju URI, ktorá je definovaná klientom pri vkladaní do skladu. Radič obsluhuje všetko ostatné, napríklad spúšťanie procedúr na strane servera, ktoré priamo nesúvisia s nijakým iným zdrojom.

Odporúčané pravidlá

Pri vytváraní REST API sa odporúča dodržiavať niekoľko pravidiel.

Pre URI platí, že lomka „/“ musí indikovať hierarchiu vzťahov medzi zdrojmi a nesmie byť posledným znakom. Je zakázané používať podčiarkovník „_“, namiesto toho sa odporúča používať pomlčku „-“ pre zlepšenie čitateľnosti.

Ak URI odkazuje na nejaký dokument, ktorý je súčasťou kolekcie, mala by byť táto skutočnosť zakódovaná do adresy. Pre kolekcie a sklad sa používa množné číslo, zatiaľ čo pre dokumenty jednotné číslo. Pre URI radiča by sa malo používať sloveso.

Jednotný prístup pre prácu so zdrojmi je zabezpečený vďaka CRUD operáciám:

- vytvorenie dát (angl. *Create*),
- získanie dát (angl. *Retrieve*),
- modifikovanie dát (angl. *Update*),
- vymazanie dát (angl. *Delete*).

Názvy CRUD funkcií by nemali byť súčasťou URI. Operácie by mali byť implementované pomocou odpovedajúcich HTTP metód (pozri sekciu 2.2). Súčasťou výsledku klientovej požiadavky má byť aj správne použitie HTTP stavového kódu v odpovedi (pozri tabuľku 2.1).

2.5.2 GraphQL

GraphQL je open source vyhľadávací jazyk (angl. *query language*). Je považovaný za efektívneho a flexibilného nástupcu REST API. Nasledujúci text vznikol na základe zdroja [9].

Používateľ pri vytváraní požiadavky na GraphQL server definuje štruktúru, podľa ktorej chce dáta filtrovať. Dáta dostane v rovnakej podobe zo servera. Existuje teda vzťah medzi požiadavkou a odpoveďou. Klient žiada len tie položky, ktoré využije, a vždy dostane to, čo očakáva. Server presne vie, ktoré položky klient potrebuje. Veľkosť prenášaných dát sa tak rapídne zníži.

GraphQL zjednodušuje prácu s viacerými zdrojmi naraz. Kým v REST API na získanie určitých informácií je potrebné zaslať viac požiadaviek, v GraphQL je možné špecifikovať v jednej požiadavke všetky potrebné informácie o všetkých potrebných zdrojoch.

Pre prenos GraphQL komunikácie sa využíva protokol HTTP, ale jazyk na tomto protokole nie je závislý. Neexistuje štandard, ktorý by popisoval, aké HTTP metódy a stavové kódy sa majú na čo použiť, ako v prípade REST API. GraphQL server si vystačí s jedným koncovým bodom, pomocou ktorého dokáže spracovať všetky požiadavky klienta. Takýto prístup bol zvolený najmä kvôli mobilným aplikáciám, ktoré majú obmedzené zdroje a kde je potrebné rýchlo dodať požadované dáta.

Ak chce klient komunikovať so serverom, pošle mu dokument obsahujúci operácie. Existujú dva typy operácií: dopyt (angl. *query*) a mutácia (angl. *mutation*). Dopyt sa používa na čítanie dát, mutácia na modifikáciu dát. Každá operácia má svoje meno. Môže to byť ľubovoľný reťazec, ale mal by byť výstižný.

V dopyte je možné špecifikovať položky (angl. *field*), ktoré je potrebné získať. Každá položka má určený dátový typ. Existujú jednoduché dátové typy, zoznamy a objekty. Ak je požiadavka na zdroj typu objekt, je možné navyše v požiadavke špecifikovať, ktoré položky z daného objektu sú vyžadované (tzv. *sub-selection*).

Každej položke môžu byť priradené argumenty, ktoré slúžia na filtrovanie objektov, resp. zdrojov.

Požiadavky sa môžu skladať z fragmentov. Fragmentom sa označuje skupina položiek, ktorá sa opakovane vyskytuje v požiadavke. Takýmto spôsobom je možné text požiadavky zredukovať a odstrániť duplicitný text.

V požiadavkách je možné použiť aj premenné, ktoré uľahčujú dynamické vkladanie argumentov do položiek. Požiadavka je tak znovu použiteľná. Klient zároveň nemusí požiadavku generovať znova pre iné parametre.

Ak klient potrebuje iné názvy položiek ako mu server dokáže vrátiť, môže v požiadavke použiť aliasy. Nimi klient premenuje položky, ktoré od servera dostane.

Pomocou mutácií je možné modifikovať dáta na strane servera. V mutácii je taktiež možné, rovnako ako v dopyte, definovať podobu odpovede. Rovnako ako v prípade dopytu, aj mutácia môže obsahovať viacero položiek naraz. Rozdiel je v tom, že dopyt spracováva tieto položky paralelne, zatiaľ čo mutácia ich spracováva sekvenčne.

Jadrom GraphQL servera je schéma. Popisuje funkcionalitu, ktorá je dostupná pre klienta, teda požiadavky, ktoré môže klient posielať na server. Je zrozumiteľne napísaná v jazyku SDL (*Schema Definition Language*). V schéme sa nachádzajú definície typov objektov a vzťahov medzi nimi.

Jazyk GraphQL vyvinul Facebook. Prvá verzia vyšla v roku 2015. Tento jazyk je používaný veľkými spoločnosťami ako Facebook, GitHub, Pinterest alebo Yelp. [16]

Kapitola 3

Extrakcia dát z webu

Webové stránky sú napísané v jazyku HTML. Pre formátovanie textu na webových stránkach sa spočiatku používali štýly definované pomocou jazyka HTML. V roku 1996 vznikol jazyk CSS (*Cascading Style Sheet*), ktorý umožňuje pridávať rôzne štýly do HTML dokumentov. Môžeme tak meniť písmo, farby, veľkosť medzier a podobne. Jazyk je čitateľný a zrozumiteľný. Dôležitou vlastnosťou tohto jazyka sú kaskády. Jednotlivé definície štýlov sa na seba vrstvia, čím je možné predefinovať už existujúce štýly. Dnes už vyše 95% webových stránok používa pre formátovanie textu jazyk CSS [33]. [23]

Jazyk HTML sa vyvíjal, a v roku 2014 vyšlo odporúčanie konzorcia W3C pre novú verziu HTML5. Pribudli nové sémantické značky. Sú nimi napríklad `article` (článok), `header` (hlavička), `footer` (päta), `nav` (navigácia) alebo `section` (sekcia). Jazyk CSS sa začal používať pre formátovanie textu, HTML výhradne pre definíciu štruktúry a obsahu dokumentu. [17]

Aj keď HTML5 poskytuje niekoľko sémantických značiek, rozhodne ho nemôžeme považovať za jazyk popisujúci sémantiku textu všeobecne. Použitím atribútu `class` je možné definovať vlastnú sémantiku. Neexistuje však žiaden štandard, ktorý by sa používal globálne. Nie je možné jednoznačne nájsť element v HTML dokumente, ktorý reprezentuje napr. inzerát bývania bez toho, aby bol známy kontext. Neexistujú žiadne striktné pravidlá pre reprezentáciu informácií. HTML poskytuje definície len pre prezentáciu informácií, nie pre ich sémantiku. Z tohto vyplýva, že informácie o dokumente nedokážeme spracovať priamo.

Sémantiku je možné popísať pomocou sémantického webu. Je to nová forma webového obsahu, ktorému má rozumieť nielen človek, ale aj počítač. Toto rozšírenie existujúceho webu je dnes ešte len v počiatočných fázach vývoja. Má priniesť informácie štruktúrované podľa štandardizovaných pravidiel, ktoré umožnia priradiť jednotlivým informáciám ich význam. Vyhľadávanie a spracovanie informácií sa uľahčí. Počítač bude schopný riešiť sofistikovanejšie úlohy. Dnešný web je plný dokumentov, ktoré sú písané tak, aby im rozumeli ľudia. Dokumenty nie sú definované v takej forme, aby mohli byť automaticky spracované s cieľom získania potrebných informácií. Naopak, sémantický web k tomuto mieri. Základy stavia na technológiách RDF (*Resource Description Framework*) [22] a OWL (*Ontology Web Language*) [48]. Keďže sémantický web je ešte len vo vývoji a zďaleka nie je globálne používaný, nebudeme ho ďalej uvažovať. [4]

Súčasný web obsahuje HTML dokumenty bez sémantiky. Aby tieto dokumenty mohli byť spracované automaticky, je potrebné z týchto dokumentov extrahovať dáta, a to najlepšie taktiež automaticky.

Cieľom extrakcie dát je nájsť informáciu v texte dokumentu a uložiť ju v štruktúrovanej podobe. Všeobecne existujú dva prístupy. Buď je dopredu známe, aké dáta sa majú extraho-

vať, alebo sa extrahujú všetky dostupné dáta, a tak je potrebné analyzovať celý dokument. Extrakcia dát má na najvyššej úrovni tri fázy [10]:

1. lokalizácia relevantných dokumentov,
2. identifikácia dát v dokumentoch,
3. uloženie dát do databázy.

V tejto práci je využitý prvý prístup a extrakcia dát prebieha vo všetkých troch fázach. Extrahujú sa inzeráty s prenájmami a s nimi súvisiace údaje (adresa, cena, dispozícia, výmera a podobne).

Existujú rôzne techniky extrakcie dát a v tejto kapitole sú bližšie špecifikované niektoré z nich.

3.1 Extrakcia dát z HTML

HTML je pološtruktúrovaný dokument. Môže obsahovať veľké bloky čistého textu (angl. *plain text*), ale aj štruktúry, napr. tabuľky, zoznamy, odstavce. Používateľ vníma obsah webovej stránky ako formátovaný text umiestnený v rôznych sekciách. Počítač vidí namiesto formátovaného textu čistý text ohraničený značkami. Pre automatické získanie dát z takého dokumentu preto nestačia algoritmy používané na extrakciu dát z čistého textu.

Informácie spomenuté v tejto podkapitole sú čerpané zo zdrojov [10, 21].

3.1.1 Wrapper

Wrapper je základným nástrojom pre extrakciu informácií z pološtruktúrovaných dokumentov. Text s hľadanou informáciou je väčšinou obklopený značkami alebo iným textom, kľúčovými slovami a podobne. Ak sú tieto hranice identifikovateľné, je možné získať požadované informácie vytvorením procedúry, ktorá ich vie presne lokalizovať.

Hranice sa definujú na základe *extrakčných pravidiel*. Nevýhodou je, že tieto pravidlá sú pevne viazané, resp. závislé na konkrétnom dokumente a jeho štruktúre. Webové stránky, ktoré poskytujú významovo rovnaký obsah, majú všeobecne inú štruktúru, formát, inak používajú značky. Každý dokument tak musí mať svoje vlastné pravidlá. Pre získanie nových informácií z nových dokumentov je potrebné vytvoriť nové pravidlá. Ak sa dokument zmení, wrapper prestane spoľahlivo fungovať a extrakčné pravidlá sa budú musieť prispôsobiť.

Triedy wrapperov

Extrakčné pravidlá pre wrappery môžeme rozdeliť do šiestich skupín.

LR (*left-right*) je základnou triedou. Wrappery patriace do tejto triedy analyzujú dokument zľava doprava a pritom hľadajú oddeľovače, ktoré tvoria pár. Pár oddeľovačov predstavuje jeden hľadaný atribút, jednu hľadanú informáciu. Pár oddeľovačov je tvorený ľavou (otváracou) a pravou (zatváracou) značkou. Procedúra najprv hľadá prvý výskyt ľavej značky a zapamätá si jej pozíciu. Potom hľadá prvý výskyt pravej značky. Ak ju nájde, všetko to, čo sa nachádza medzi týmito dvoma značkami, považuje za nájdený atribút. Rovnako pokračuje v hľadaní ďalších atribútov.

Všeobecnejšou triedou je **HLRT** (*head-left-right-tail*). Táto trieda rozdeľuje dokument na tri hlavné časti: hlavička, telo a päta. Algoritmus je podobný ako v triede LR, avšak nepodstatné a zavádzajúce časti v hlavičke a v päte dokumentu sú ignorované. Relevantné

informácie sa hľadajú v tele dokumentu použitím rovnakého algoritmu ako v prípade LR triedy.

OCLR (*open-close-left-right*) je alternatívou k HLRT. Taktiež rieši problém nesprávnej extrakcie informácií z irelevantného textu, ale používa na to iný spôsob, viac sofistikovanejší. Dokument nerozdeľuje na tri časti. Namiesto toho vidí dokument ako sekvenciu dvojíc, ktoré sú oddelené nepodstatným textom. Každá dvojica začína otváracou a končí zatváracou značkou. Informácie sa hľadajú v texte medzi týmito dvojicami, opäť pomocou LR metódy.

Kombináciou HLRT a OCLR vznikla trieda **HOCLRT** (*head-open-close-left-right-tail*). Najprv sa vyberie telo dokumentu, v ňom sa hľadajú dvojice ohraničené otváracou a zatváracou značkou a v týchto dvojiciach sú pomocou základnej LR metódy vyhľadané potrebné informácie.

Všetky vyššie spomenuté triedy počítajú s pravidelnou tabuľkovou štruktúrou dokumentu. Algoritmy sú na to prispôsobené. Niekedy je potrebné extrahovať informácie z nepravidelnej štruktúry, akými sú napr. zanorené viacúrovňové zoznamy. Pre riešenie tohto problému existujú modifikované wrappery z tried **N-LR** alebo **N-HLRT**.

Tvorba wrapperov

Konštrukcia wrappera prebieha buď manuálne alebo automaticky. Ako už bolo spomenuté vyššie, wrapper je úzko spojený so štruktúrou dokumentu. Aj pri menšej zmene môže dôjsť k znefunkčneniu wrappera. V takom prípade musia byť extrakčné pravidlá wrappera znova prispôsobené modifikovanému dokumentu. Je vhodné použiť automatickú konštrukciu wrappera, aby nedochádzalo k zbytočným komplikáciám.

Manuálna konštrukcia je časovo náročnejšia, viac náchylná k chybám a nedá sa efektívne škálovať. Napriek tomu je využívaná viac ako automatická.

Pri automatickej konštrukcii wrapperov sa využíva *wrapper induction* založená na strojovom učení s učiteľom (angl. *supervised learning*). Aj tento prístup je časovo náročný, keďže je potrebné manuálne vytvoriť kvalitnú trénovaciu sadu. Touto metódou môže vzniknúť wrapper, ktorý nebude presný v detekcii požadovaných informácií. Pri zmenách dokumentu bude možno potrebné celý proces učenia zopakovať.

Kapitola 4

Dátový model

Dáta po získaní z webu, ešte pred ich spracovaním, je potrebné niekam uložiť. Tento účel spĺňajú databázy. Existuje však veľa druhov databáz a je veľmi dôležité zvoliť tú správnu, aby v neskoršej fáze vývoja softvéru nedošlo ku komplikáciám. Databázu je potrebné prispôsobiť dátam, ktoré v nej budú uložené – čo tie dáta predstavujú, aký majú účel, celkový objem dát a podobne. Návrh dátového modelu má veľký význam.

Na najvyššej úrovni sa databázy rozdeľujú na relačné a NoSQL (*Not only SQL*). Relačná databáza vznikla už v roku 1969 a navrhol ju Edgar Codd z IBM. Dodnes dominuje medzi všetkými databázovými modelmi.

Relačná databáza organizuje dáta do tabuliek. Vychádza z matematického pojmu *relácia*. Relácia je tabuľka zložená z atribútov (stĺpce) a usporiadaných n-tíc (riadky). Každý stĺpec má definovaný jednoznačný názov a dátový typ (doménu). Jeden riadok tabuľky reprezentuje jeden záznam.

V rámci relačnej databázy je možné vytvárať vzťahy medzi jednotlivými tabuľkami. Záznam v tabuľke sa môže odkazovať na záznam v inej tabuľke, a to prostredníctvom tzv. primárnych a cudzích kľúčov. Tým dokáže dáta ukladať efektívne, bez redundancie.

Relačná databáza sama zabezpečuje integritu dát. Pre dotazy a úpravy nad dátami v relačnej databáze sa používa jazyk SQL (*Structured Query Language*). [18]

Neskôr vznikla potreba vyvinúť inú databázu, databázu bez tabuliek. Aplikácie sa začínajú skladať z niekoľkých menších častí tzv. služieb, ktoré musia byť dostupné vždy a pre milióny rôznorodých zariadení. Kladú sa vysoké požiadavky na škálovanie. Organizácie začínajú používať *cloud computing*, aby výrazne zvýšili svoju výpočtovú silu. Distribuované databázy sú čím ďalej, tým viac populárne. Úložisko dát je decentralizované. Dáta sa uchovávajú v niekoľkých kópiách (úmyselná redundancia) pre zvýšenú odolnosť proti výpadkom a zvýšenie rýchlosti. K dispozícii sú veľké objemy dát, nad ktorými je potrebné vykonávať množstvo operácií. Dáta už nie sú len štruktúrované, ale aj pološtruktúrované, neštruktúrované a polymorfické. Vznikajú nové dátové typy, ktoré komplikujú prácu s databázou. Dlhé niekoľkomesačné vývojové cykly vystriedali krátke šprinty s rýchlo meniacim sa kódom. Tento nový iteratívny vývoj vyžaduje časté zmeny databázovej schémy.

NoSQL databázy prichádzajú s riešením. Oproti relačným ponúkajú väčšie škálovanie, vyššiu výkonnosť a dátový model, ktorý rieši problémy spomenuté vyššie. [29]

4.1 Porovnanie SQL a NoSQL databáz

V tejto podkapitole sú porovnané SQL a NoSQL databázy. Väčšina informácií je prebraná z príspevku od Craiga Bucklera [8].

Spôsob uloženia dát

SQL databázy používajú na ukladanie dát tabuľky. Tabuľky nie sú flexibilné.

Každá tabuľka má určité podmienky, ktoré musia byť splnené pre úspešné vloženie dát. Nie je možné vložiť do tabuľky číslo na miesto, kde sa očakáva reťazec. Zmena štruktúry databázy vyžaduje vykonať časovo náročnú operáciu, *migráciu databázy*.

NoSQL databázy ukladajú dáta iným spôsobom. Využívajú dvojice kľúč-hodnota, ktoré môžu byť zaobalené do dokumentov a tie do kolekcií, ktoré sú obdobou tabuliek. Výhodou je voľná štruktúra dokumentov. Neexistujú striktné pravidlá, ktoré určujú, čo do dokumentu môže a čo nemôže byť vložené. Zároveň je to nevýhoda, keďže do databázy môžu byť vložené aj neplatné dáta, čo vedie k problémom s konzistenciou.

Schéma

Do SQL databázy nie je možné pridať dáta bez vytvorenia tabuliek a definovania ich schém. Schéma definuje typy polí (angl. *fields*), primárne a cudzie kľúče, indexy a podobne. Predtým, než sa začne implementovať aplikačná vrstva (angl. *business logic*), ktorá manipuluje s dátami, je potrebné dobre navrhnúť a definovať schému. Schému je možné zmeniť aj v neskoršej fáze vývoja softvéru, ale v prípade väčších zmien to môže byť komplikované.

Do NoSQL databázy môžu byť dáta vložené kedykoľvek a kdekoľvek. V niektorých prípadoch dokonca nemusí existovať ani databáza, či kolekcia pri vkladaní nového dokumentu. Stačí ho jednoducho vložiť, a ak daná databáza alebo kolekcia neexistuje, vytvorí sa za behu. Schéma síce v NoSQL databázach neexistuje, ale niektoré typy podporujú validačné pravidlá. Pomocou nich je možné obmedziť podobu vstupných dát.

Normalizácia a denormalizácia

SQL databáza má byť normalizovaná. Normalizáciou sa odstraňujú redundantné dáta použitím primárnych a cudzích kľúčov. Ak sú dáta redundantné, ich zmena je náročná, pretože je potrebné nájsť všetky výskyty. Cudzie kľúče odkazujú na ďalšie tabuľky, resp. primárne kľúče v týchto tabuľkách. Pomocou príkazu `JOIN` je možné jednotlivé tabuľky spojiť a získať všetky súvisiace informácie.

V NoSQL databázach je normalizácia možná (pomocou referencií), ale nevyužíva sa. Hlavným dôvodom je spájanie súvisiacich dokumentov, ktoré musí používateľ riešiť sám. Tieto operácie sú časovo náročné. Naopak, vyhľadávanie v rámci denormalizovaných dát je efektívnejšie. Nevýhodou sú duplicity, s tým spojená pomalá aktualizácia dát a nezaručená konzistencia dát.

Integrita dát

SQL databázy kontrolujú integritu dát pomocou obmedzení (angl. *constraints*), napr. v podobe cudzích kľúčov. Položka tabuľky sa nemôže odkazovať na neexistujúcu položku. Zároveň je možné nastaviť akciu, ktorá sa vykoná po vymazaní položky odkazovanej cudzím kľúčom. Tým sa používatelia dokážu vyhnúť nezmyselným alebo neplatným dátam.

V NoSQL databáze je možné vymazať dokumenty aj keď sú odkazované z iných dokumentov, databáza to povoľuje. Ideálne by mal návrh NoSQL databázy vyzerať tak, že všetky informácie, ktoré spolu úzko súvisia, sú uložené v jednom spoločnom dokumente.

Výkonnosť

NoSQL databázy dokážu rýchlejšie odpovedať na požiadavky vďaka denormalizácii. V SQL databázach je často využívaná operácia `JOIN` časovo náročná, najmä pri spájaní mnohých alebo veľkých tabuliek. Záleží aj od návrhu databázy a na dátach, s ktorými sa pracuje. Dobre navrhnutá SQL databáza môže byť rýchlejšia ako zle navrhnutá NoSQL databáza.

Transakcie

Databázová transakcia je skupina operácií, ktorá má tieto vlastnosti: atomičnosť, konzistencia, izolovanosť a trvalosť. Pre túto skupinu vlastností sa používa skratka ACID podľa anglických variant ich názvov. V tejto skupine musia byť úspešné všetky operácie alebo žiadna. Ak niektoré uspejú a niektoré zlyhajú, nastane *rollback*, teda návrat už vykonaných zmien. V systéme teda nemôže zostať žiaden rozpracovaný stav, musí prebehnúť celá transakcia. Vykonanie takejto transakcie nemôže vyvolať nekonzistentný stav v databáze. Transakcia sa neuskutoční, pokiaľ nie je schopná dodržať pravidlá konzistencie databázy. Transakcia je vykonávaná v izolácii. Počas nej sa nesmie vykonať žiadna iná transakcia, ktorá by ju nejakým spôsobom ovplyvnila. Súbežné transakcie sú možné, ale nesmú sa ovplyvňovať. Po vykonaní transakcie musí byť zaručené, že zmeny sú skutočne uložené v databáze, a už nemôžu byť stratené. [20]

Existuje teorém z oblasti teoretickej informatiky, ktorý stanovuje nevyhnutné kompromisy pri požiadavkách na vlastnosti databázového systému. Eric Brewer ho popísal v roku 1999 a nazýva sa teorém CAP (*Consistency, Availability, Partition tolerance*). Tvrdí, že v distribuovaných systémoch nie je možné poskytovať viac ako dve záruky z týchto troch: konzistencia, dostupnosť, odolnosť k prerušeniu. Konzistencia zaručuje konzistentné dáta nezávisle na bežiacich operáciách či ich umiestnení. Dostupnosť zaisťuje nepretržitú prevádzku a možnosť vždy zapísať alebo čítať dáta. Odolnosť k prerušeniu znamená, že systém bude pokračovať v činnosti, až kým nedôjde k úplnému zlyhaniu siete alebo výpadkom všetkých uzlov (serverov). Teorém CAP inými slovami hovorí, že v prípade výpadku časti siete je potrebné zvoliť si medzi konzistenciou a dostupnosťou. [42]

Vlastnosti ACID sa zdajú byť nepostrádateľné v relačných databázach, avšak podľa teorému CAP poskytujú konzistenciu pre databázy rozdelené na viacerých serveroch (*partitioning*) na úkor dostupnosti. NoSQL databázy riešia teorém CAP obmedzením konzistencie dát použitím modelu BASE (*Basically Available, Soft state, Eventually consistent*). Táto skratka vyjadruje to, že aplikácia beží takmer neustále, nemusí byť vždy konzistentná, ale skôr či neskôr sa do konzistentného stavu dostane. Prípadné nekonzistencie môžu byť riešené pri čítaní (verziovanie, invalidácia cache), pri zápise (distribúcia zmien) alebo asynchrónne (replikácia dát). Táto „slabá“ konzistencia môže mať za následok to, že v jeden okamih dvaja používatelia databázy vidia rôzne verzie tých istých dát. Dáta sú ale vždy dostupné. Zároveň to prináša aj zrýchlenie celého systému. [32, 37]

Škálovanie

Je dobré myslieť na škálovanie už od začiatku vývoja softvéru, aby sa v neskorších fázach predišlo preťaženiu systému. Záťaž sa dá rozložiť na viac serverov. V prípade SQL databáz

to môže byť veľkou výzvou kvôli zložitému dátovému modelu, kde je mnoho vzťahov medzi tabuľkami. Je potrebné riešiť otázky ako tabuľky od seba oddeliť a ako v nich potom hľadať informácie.

Niektoré NoSQL databázy v sebe majú zabudovanú podporu pre replikovanie a horizontálne delenie dát (angl. *sharding*). Dnes už aj SQL databázy podporujú podobné princípy, ktoré sú však zložité na realizáciu.

4.2 Typy NoSQL databáz

Poznáme niekoľko typov NoSQL databáz. Je zrejmé, že každý typ je vhodný pre iné prípady použitia.

4.2.1 Databázy založené na princípe kľúč-hodnota

Prvým z typov NoSQL databáz je databáza zložená z dvojíc kľúč-hodnota. Základnou dátovou štruktúrou je slovník (angl. *dictionary*). Do tejto databázy môžeme uložiť hodnotu typu číslo, reťazec, JSON alebo pole. Spolu s hodnotou musíme uložiť aj kľúč k tejto hodnote, ten slúži ako referencia k danej hodnote. V databáze neexistujú duplicitné kľúče. K žiadaným hodnotám je možné dostať sa len pomocou kľúča. Používajú sa na to hashovacie tabuľky, ktoré sú efektívne. Hodnoty sú uložené binárne, a teda databázový systém ich neanalyzuje, neskúma ich typy a podobne. Vykonáva sa to až na strane aplikácie. [44]

Najznámejším zástupcom týchto databáz je Redis. Používa sa najmä na cachovanie dát. Ďalším zástupcom môže byť napríklad Amazon S3[1] alebo Amazon DynamoDB[2]. Patria medzi cloudové úložiská.

4.2.2 Dokumentové databázy

Dokumentové databázy predstavujú rozšírený koncept databáz založených na princípe dvojíc kľúč-hodnota. Hodnota je štruktúrovaná a analyzovaná databázovým systémom. Využíva sa formát XML a najmä JSON a jeho binárna podoba. Existuje možnosť odkazovania sa na iný dokument, ale vzhľadom na denormalizáciu popísanú v kapitole 4.1 sa namiesto toho používa vnáranie štruktúr. Dokumentové databázy prichádzajú s možnosťou organizovať dokumenty do kolekcií. Kolekcie sú obdobou tabuliek v relačných databázach. Naproti tomu databázy založené na princípe dvojíc kľúč-hodnota uchovávajú všetky dáta v jednom mennom priestore. Hlavným zástupcom je MongoDB[3].

MongoDB má využitie najmä v analýze v reálnom čase, internete vecí (angl. *Internet of Things*), veľkých dátach (angl. *Big Data*), produktových katalógoch, bezpečnosti a detekcii podvodov, mobilných aplikáciách, správe obsahu a v sociálnych aplikáciách [47].

[1] https://aws.amazon.com/s3/
[2] https://aws.amazon.com/dynamodb/
[3] https://www.mongodb.com/

4.2.3 Grafové databázy

Grafové databázy používajú teóriu grafov pre ukladanie, mapovanie a zisťovanie vzťahov medzi jednotlivými entitami. Grafová databáza pozostáva z kolekcie uzlov a hrán. Každý uzol reprezentuje entitu v modelovanom svete. Entitou sa rozumie napr. osoba, miesto alebo vec. Každá hrana reprezentuje vzťah medzi dvoma uzlami. Hrana definuje ako spolu dva konkrétne uzly súvisia. Každý uzol v grafovej databáze je definovaný unikátnym identifikátorom, množinou vychádzajúcich hrán, množinou vchádzajúcich hrán a množinou vlastností vyjadrených ako dvojice kľúč-hodnota. Každá hrana v grafovej databáze je definovaná unikátnym identifikátorom, začínajúcim uzlom, končiacim uzlom a množinou vlastností. [36]

Grafové databázy sú orientované na vzťahy a nie na diskrétne dáta. Vzťahy sú perzistentné, nemusia sa teda stále prepočítavať a odvodzovať z iných vecí, ako to je v prípade relačných databáz, ktoré na to využívajú cudzie kľúče. Výsledkom je jednoduchší dátový model, ale s väčšou vyjadrovacou schopnosťou.

Grafové databázy sú vhodné na analýzu vzťahov ako takých. Využitie našli napríklad aj v spracovaní dát zo sociálnych médií. Grafové databázy sa dnes považujú za veľmi nádejnú technológiu využiteľnú najmä v oblasti veľkých dát, sémantického webu a ontológie. Množstvo dát na webe rastie a vzťahy medzi týmito dátami sú stále komplikovanejšie. Relačné databázy na to nie sú pripravené a požiadavky na získanie dát so zložitými väzbami budú čoraz náročnejšie na výpočet. Schéma grafického dátového modelu vie byť veľmi flexibilná. Nie je potrebné po čase zahodiť aktuálny model a navrhnúť nový, je možné aktualizovať už existujúci. Neplatia tu striktné pravidlá ako v prípade relačných databáz. [38]

Najznámejším zástupcom grafových databáz je Neo4j. Prichádza s kompletnými riešeniami v oblasti detekcii podvodov, grafe znalostí, monitorovania sietí a databázovej infraštruktúry, sociálnych médií a analýzy sociálnych sietí, správy identity a prístupu, umelej inteligencie a ďalších. [31]

4.2.4 Stĺpcové databázy

Dáta v stĺpcových databázach sú organizované do stĺpcov. Používa sa koncept priestoru kľúčov (angl. *keyspace*). Pripomína schému v relačných databázach. V priestore kľúčov sa nachádzajú rodiny stĺpcov (angl. *column families*). Tie pripomínajú tabuľky v relačných databázach, keďže sa taktiež skladajú z riadkov. Rozdiel je v tom, že každý riadok môže mať rôzny počet stĺpcov rôzneho typu. Každý stĺpec má názov, hodnotu a časovú značku. [19]

Výhodou stĺpcových databáz je efektívna kompresia a rozdeľovanie dát. Veľkú rolu zohrávajú najmä v analýze veľkých dát. Stĺpcové databázy vedia efektívne počítať agregačné funkcie (suma, počet, priemer a podobne). Sú škálovateľné. Dizajnovo sú stavané k masívnemu paralelnému spracovaniu dát. Stĺpcové databázy vedia veľmi rýchlo načítať dáta. Tabuľku s biliónom riadkov je možné načítať behom niekoľkých sekúnd, a tak spracovanie a analýza dát môže začať prakticky okamžite. [7]

Najznámejším riešením je Cassandra[4], ktorú vyvinul Facebook. Jej kód neskôr zverejnil, stal sa z nej open source softvér. Momentálne spadá pod správu firmy Apache. Cassandru je možné použiť napríklad na sledovanie transakcií, ukladanie časovej rady, sledovanie stavu alebo ukladanie dát s históriou [39].

[4] http://cassandra.apache.org/

Kapitola 5

Analýza existujúcich riešení

5.1 Portály pre agregáciu dát

V tejto kapitole sú popísané tri portály, ktoré agregujú dáta rôzneho druhu z rôznych webových zdrojov. Sú predstavené služby, ktoré ponúkajú svojim zákazníkom. Tieto informácie poslúžia pri vytváraní vlastného portálu ako inšpirácia na zlepšenie.

Heureka

Heureka[1] je bezpochyby najväčším nákupným poradcom na českom trhu. Táto platforma elektronického obchodovania, ktorá sa radí medzi tie najväčšie, bola spustená v roku 2007. Má vysokú návštevnosť, mesačne ich stránky navštívi okolo štyroch miliónov ľudí. V roku 2018 bola ich webstránka deviatou najnavštevovanejšou v rámci Českej republiky [41].

Zákazníci si pomocou ich nástrojov môžu na jednom mieste porovnať výrobky z desiatok tisíc internetových obchodov. Samozrejmosťou je prehľad cien, ale ponúkajú aj možnosť písania a prezerania recenzií, či už na konkrétny produkt alebo na samotný obchod. Dávajú priestor na zdieľanie skúseností a hodnotení ostatných nakupujúcich.

Heureka ponúka internetovým obchodom službu *Overené zákazníkmi*. V rámci nej reálni zákazníci nezávisle hodnotia obchody prostredníctvom dotazníka o zhodnotení spokojnosti s nákupom. Ak má daný obchod pozitívnu spätnú väzbu, dostane od Heureky certifikát. Jeho platnosť sa môže skončiť v prípade zvýšenia počtu nespokojných zákazníkov. Predpoklad je taký, že zákazníci radšej nakupujú v obchodoch, ktoré sú overené. Certifikát je tak veľkou motiváciou pre internetové obchody na to, aby skvalitnili svoje služby.

Zákazníci majú taktiež možnosť nakúpiť produkty priamo prostredníctvom Heureky, nemusia navštíviť konkrétny internetový obchod. Výhodou je spojenie viacerých objednávok z rôznych obchodov do jednej. Heureka zároveň poskytuje garanciu nákupu[2]. V prípade nedoručenia tovaru zákazníkom vrátia peniaze. V prípade doručenia zlého tovaru alebo poškodenia tovaru vybavujú potrebnú administratívu, a dokonca je možné vrátiť nerozbalený a nepoškodený tovar do troch mesiacov.

Heureka ponúka desiatky miliónov produktov. Nechýba filtrovanie podľa parametrov, napr. podľa ceny alebo značky. Parametre môžu byť viac špecifické, mení sa to podľa kategórie produktu. Heureka ponúka taktiež funkcie *Nákupný poradca* a *Poradňa*. Nákupný poradca predstavuje návod ako si vybrať daný produkt. Skladá sa z viacerých odporúčaní,

[1] https://www.heureka.cz/
[2] https://www.garancenakupu.cz/

napr. na čo si dávať pozor alebo podľa akých parametrov vyberať. Poradňa ponúka osobný kontakt s reálnymi osobami, odborníkmi na danú problematiku.

Veľmi využívaným nástrojom je porovnávanie produktov. Ak sa používatelia nevedia rozhodnúť medzi niekoľkými produktmi, môžu si ich zobraziť naraz spolu s ich parametrami a sledovať rozdiely. Ďalšou vhodnou funkciou je *Strážiť cenu*. Používateľ si nastaví určitú hranicu, a keď cena klesne pod túto hranicu, Heureka ho informuje prostredníctvom emailu.

Zboží.cz

Alternatívou k Heureke je Zboží.cz[3] od Seznamu. Je to služba, ktorá ponúka informácie o produktoch a o jeho cenách v registrovaných internetových obchodoch. Poskytujú sprostredkovaný predaj, na rozdiel od Heureky neprevádzkujú samotný predaj. V prehľadnej forme ponúkajú informácie o produktoch naprieč niekoľkými internetovými obchodmi.

Rovnako ako Heureka poskytujú pomoc pri vyberaní produktov. Súčasťou je vysvetlenie základných pojmov, skratiek, či vymenovanie výhod a nevýhod. V niektorých prípadoch je dostupný videozáznam, kde je produkt otestovaný a bližšie predstavený.

Produkty je možné označiť ako obľúbené a vrátiť sa k nim neskôr. Je možné ich taktiež porovnávať medzi sebou. Zaujímavým doplnkom sú návody k produktom a príbalové letáky k liekom.

Súčasťou webovej stránky sú informácie o predajných miestach. Každé miesto uvádza popis ponuky, spôsob dopravy a platby, verejné sú taktiež hodnotenia ich zákazníkov.

Glami

Glami[4] je vyhľadávač, ktorý sa zameriava na oblečenie a módu všeobecne. Vznikol v roku 2013. Podobne ako Zboží.cz predaj len sprostredkováva. Zákazníci nakupujú až priamo v internetových obchodoch a cez Glami si konkrétnu časť oblečenia len vyhľadajú. Cieľom Glami je zhromažďovať a kategorizovať oblečenie na jednom mieste, a tak nakupovanie zjednodušiť a priblížiť módu naozaj každému.

Služba ponúka filtrovanie, oblečenie je navyše delené do rôznych kategórií. Nezvyčajné je rozdelenie samotných módnych značiek do tried ako prémiové, športové, „eco-friendly“, česká móda a podobne. V ponuke je okolo dvoch miliónov produktov od tisícok značiek.

Keďže Glami jednotlivé produkty nepredáva, nezaoberá sa sťažnosťami súvisiacimi s nákupom. Svojich zákazníkov priamo vyzýva, aby sa s prípadnými otázkami obrátili na konkrétny internetový obchod.

[3] https://www.zbozi.cz/
[4] https://www.glami.cz/

5.2 Portály poskytujúce inzeráty s prenájmami bytov

Táto kapitola sa venuje portálom, ktoré ponúkajú predaj alebo prenájom nehnuteľností prostredníctvom inzerátov. Spomínajú sa tu všeobecné informácie o niekoľkých portáloch a o službách, ktoré ponúkajú. Kapitola obsahuje analýzu spôsobu komunikácie medzi klientom a serverom. Súčasťou toho je rozbor posielaných dát a ich forma, čo je využité pri získavaní informácií o prenájmoch z týchto portálov.

UlovDomov.cz

Najznámejšou databázou prenájmov bytov a spolubývania v Českej republike je UlovDomov.cz[5]. Všetky inzeráty zobrazuje na jednej mape. Portál neposkytuje inzerciu na predaj nehnuteľností.

Počet aktívnych inzerátov je dnes približne 9 000, a z toho až polovica má adresu v Prahe a jej okolí. Väčšinu inzerátov tvoria prenájmy, spolubývanie má menšie zastúpenie.

UlovDomov.cz vykonáva analýzy nad dostupnými dátami a zobrazuje zaujímavé štatistiky. Jedným z príkladov je priemerný čas v minútach, za ktorý inzerát získa svojho prvého záujemcu. Vedľa tejto informácie sa nachádza odkaz na funkciu *Strážny pes*. Je to nástroj, pomocou ktorého je možné nastaviť si upozornenie na nové inzeráty vyhovujúce daným požiadavkám. Tieto upozornenia je možné dostávať denne, týždenne, alebo po 8 hodinách od zverejnenia inzerátu. Ak si zákazník zaplatí *Premium účet*, môže tieto upozornenia dostávať ihneď po zverejnení inzerátu, a zvýšiť tak šance na nový domov.

UlovDomov.cz otvorene poskytuje dokument s názvom *Ťahák nájomníka*. Ten je plný nevyhnutných otázok, ktoré by mali potenciálni nájomcovia položiť prenajímateľovi pri prehliadke bytu. Slúži ako pomôcka pri zhodnotení stavu bytu. Okrem základných informácií o prenájme, akými sú adresa, dispozícia alebo výmera, dokument obsahuje niekoľko ďalších častí. V nich rozoberá cenu nájomného so všetkými jeho položkami, zariadenie bytu, dôležité termíny spojené s dobou nasťahovania, dĺžkou zmluvy a podobne. Dokument sa taktiež zaoberá aj lokalitou a pripomína nájomcom, aby nezabudli skontrolovať dôležité veci týkajúce sa vybavenia bytu.

Klient portálu získava informácie o inzerátoch prostredníctvom REST API. Všetky nasledujúce informácie sú získané experimentálne. Autor tejto práce nemá žiadnu znalosť o verejne dostupnej dokumentácií k API, ktoré portál využíva. Niektoré informácie môžu byť nepravdivé, zavádzajúce alebo nekompletné.

Zoznam inzerátov je možné získať prostredníctvom zadania požiadavky `POST` na adresu `https://www.ulovdomov.cz/fe-api/find?offers_only=true`. Telo požiadavky pozostáva z niekoľkých parametrov. Tie sú popísané v tabuľke 5.1.

[5] `https://www.ulovdomov.cz/`

Názov	Typ	Popis	Poznámka	Povinný
`query`	textový reťazec	mesto, adresa alebo PSČ		nie
`offer_type_id`	textový reťazec	typ ponuky	`1` - prenájom `2` - spolubývanie	áno
`dispositions`	pole textových reťazcov	dispozícia	`1` - garsónka `2` - 1+kk `3` - 1+1 `4` - 2+kk ... `14` - 7+kk `15` - 7+1 `16` - atypický `29` - dom	nie
`price_from`	textový reťazec	min. cena v Kč		nie
`price_to`	textový reťazec	max. cena v Kč		nie
`acreage_from`	textový reťazec	min. rozloha v km^2		nie
`acreage_to`	textový reťazec	max. rozloha v km^2		nie
`added_before`	textový reťazec	počet dní od pridania inzerátu		nie
`furnishing`	pole textových reťazcov	vybavenie - všeobecne	`FULL` - úplne `MEDIUM` - čiastočne `NONE` - vôbec	nie
`conveniences`	pole textových reťazcov	vybavenie - konkrétne položky	`washing_machine` - pračka `dishwasher` - umývačka riadu `fridge` - chladnička `cellar` - pivnica `balcony` - balkón	nie
`is_price_commision_free`	logická hodnota	s/bez provízie realitnej kancelárie[6]		nie
`sort_by`	textový reťazec	podľa čoho sa majú inzeráty zoradiť	`" "` - najlepšie `date:desc` - najnovšie `price:asc` - najlacnejšie `date:desc` - najstaršie `price:asc` - najdrahšie[7]	nie
`bounds`	objekt	ohraničený priestor na mape, v ktorom sa má hľadať	`north_east` a `south_west` - povinné, musia obsahovať reálne čísla `lng` a `lat`	áno

Tabuľka 5.1: Parametre tela požiadavky na získanie zoznamu inzerátov z UlovDomov.cz

Odpoveď servera na túto požiadavku je vo formáte JSON. Podstatnou súčasťou odpovede je pole objektov, ktoré reprezentujú inzeráty. Niektoré dátové položky patriace k objektu inzerátu sú popísané v tabuľke 5.2.

[6]Hľadať inzeráty bez provízie realitnej kancelárie je možné na ich webovej stránke len pri zakúpení Premium účtu. Ich verejné neplatené API však dovoľuje filtrovať inzeráty na základe tohto parametra a to úplne zadarmo.

[7]Najstaršie a najdrahšie inzeráty nie je možné vyhľadať priamo pomocou ich webového nástroja. Ich API to dovoľuje.

Názov	Typ	Popis
`id`	celé číslo	unikátny identifikátor inzerátu
`offer_type_id`	celé číslo	typ ponuky (rovnako ako v tabuľke 5.1)
`disposition_id`	celé číslo	dispozícia (rovnako ako v tabuľke 5.1)
`acreage`	celé číslo	rozloha v km^2
`price_rental`	celé číslo	cena za prenájom v Kč
`photos`	pole objektov	fotky nehnuteľnosti
`street`	objekt	ulica
`village_part`	objekt	časť obce
`village`	objekt	obec
`description`	textový reťazec	popis inzerátu
`absolute_url`	textový reťazec	absolútna cesta k inzerátu
`published_at`	textový reťazec	dátum pridania inzerátu vo formáte ISO8601
`lng`	reálne číslo	zemepisná dĺžka
`lat`	reálne číslo	zemepisná šírka

Tabuľka 5.2: Položky odpovede servera na požiadavku zoznamu inzerátov z UlovDomov.cz

Vyššie popísaná odpoveď nezobrazuje všetky informácie o inzeráte, ale len tie základné. Pre získanie detailných informácií o inzeráte, je potrebné vytvoriť novú požiadavku na adresu `https://www.ulovdomov.cz/fe-api/offer/<id>`. Využíva sa metóda `GET`. Premenná `<id>` v adrese predstavuje unikátny identifikátor inzerátu. Odpoveď zo servera je vo formáte JSON a obsahuje všetky dostupné informácie o inzeráte. V tabuľke 5.3 sú popísané položky, ktoré pribudli. Tie, ktoré už sú súčasťou tabuľky 5.2, nie sú znova spomenuté.

Názov	Typ	Popis	Poznámka
`furnishing_id`	textový reťazec	vybavenie – všeobecne	rovnako ako v tabuľke 5.1
`conveniences`	pole textových reťazcov	vybavenie – konkrétne položky	rovnako ako v tabuľke 5.1
`price_rental`	celé číslo	cena za nájom	
`floor_level`	textový reťazec	číslo podlažia	reťazec začínajúci písmenom „p“ doplnený o číslo
`energy_efficiency_rating_id`	celé číslo	energetická náročnosť budovy	`1` - A – mimoriadne úsporná `2` - B – veľmi úsporná `3` - C – úsporná `4` - D – menej úsporná `5` - E – nehospodárna `6` - F – veľmi nehospodárna `10` - G – neuvedená
`nearby`	pole objektov	obchody, zastávky MHD v okolí	názov, vzdialenosť a typ
`owner`	objekt	informácie o majiteľovi alebo maklérovi, resp. prenajímateľovi	id, meno, priezvisko, fotka
`status`	textový reťazec	stav inzerátu	`ACTIVE` - aktívny inzerát
`median`	celé číslo	priemerná cena za byt v rovnakej lokalite s rovnakou dispozíciou	

Tabuľka 5.3: Položky odpovede servera na požiadavku detailu inzerátu z UlovDomov.cz

Súčasťou odpovede sú aj položky, ktoré by mali niesť informáciu o mesačných poplatkoch, výške vratnej kaucie, provízii realitnej kancelárie a aj informácie o dostupnosti a konkrétny dátum možného nasťahovania. To sa avšak nevyužíva a tieto položky sú prázdne reťazce alebo majú nulové hodnoty. Jednotlivé položky ceny je potrebné zistiť z popisu inzerátu.

Bezrealitky.cz

Bezrealitky.cz[8] predstavuje ďalší internetový trh s nehnuteľnosťami, avšak na rozdiel od UlovDomov.cz ponúka inzeráty bez asistencie realitných kancelárií. Už od roku 2007 je možné prostredníctvom ich webovej stránky predať alebo prenajať nehnuteľnosť, čo prebieha výhradne medzi priamymi vlastníkmi a ich záujemcami. Po tejto možnosti siahajú ľudia, ktorí nechcú utrácať peniaze za provízie realitných kancelárií, a dokážu tak ušetriť nemalú finančnú čiastku. Aktuálne sa na stránke nachádza okolo 3 000 aktívnych inzerátov s prenájmami nehnuteľností, 3 000 inzerátov s predajom, a okolo 100 inzerátov so spolubývaním. Návštevnosť stránok je okolo pol milióna mesačne [5].

Bezrealitky.cz ponúka aj doplnkové služby, prostredníctvom ktorých je napríklad možné zaplatiť si vyhotovenie nájomnej či kúpnej zmluvy. Vykonávajú taktiež fotenie nehnuteľnosti, odhad ceny nehnuteľnosti a vedia zabezpečiť technickú inšpekciu nehnuteľnosti.

Po zakúpení Premium účtu zákazníci získajú prednostné zobrazenie ich správ majiteľovi inzerátu, upozornenia na nové inzeráty každú hodinu (bez Premium účtu to je raz za deň), či online podporu a pomoc s kúpou alebo prenájmom.

Webová stránka portálu Bezrealitky.cz získava informácie o inzerátoch z GraphQL servera. Na získanie potrebných informácií o všetkých inzerátoch stačí použiť jedno API volanie. Detailné informácie nie je nutné získavať prostredníctvom samostatného koncového bodu ako to je v prípade UlovDomov.cz.

Pre zistenie všetkých dostupných parametrov a položiek GraphQL schémy bol využitý nástroj Apollo CLI[9]. V dokumentácii tohto nástroja[10] je popísané akým spôsobom je možné stiahnuť celú GraphQL schému. Jediným povinným paremetrom je adresa GraphQL servera.

Po naštudovaní stiahnutej schémy boli zistené nasledujúce skutočnosti. V dopyte je možné požadovať zoznam inzerátov pomocou položky `advertList`. Zoznam parametrov, ktoré je možne priradiť tejto položke je znázornený v tabuľke 5.4.

Názov	**Typ**	**Popis**
`ids`	pole celých čísel	zoznam unikátnych identifikátorov inzerátov
`offerType`	pole textových reťazcov	typ ponuky: `SPOLUBYDLENI` `PRODEJ` `PRONAJEM` `UNDEFINED`

`estateType`	pole textových reťazcov	typ nehnuteľnosti:
		`NEBYTOVY_PROSTOR` `BYT` `POZEMEK` `KANCELAR`
		`REKREACNI_OBJEKT` `DUM` `GARAZ` `UNDEFINED`

`disposition`	pole textových reťazcov	dispozícia:
		`UNDEFINED` `DISP_1_KK` `DISP_2_KK` `DISP_7_1`
		`GARSONIERA` `DISP_1_1` `...` `OSTATNI`

`priceFrom`	celé číslo	minimálna cena
`priceTo`	celé číslo	maximálna cena
`surfaceFrom`	celé číslo	minimálna rozloha
`surfaceTo`	celé číslo	maximálna rozloha
`balcony`	logická hodnota	prítomnosť balkóna
`terrace`	logická hodnota	prítomnosť terasy

Tabuľka 5.4: Parametre požiadavky na zoznam inzerátov z Bezrealitky.cz

[8] https://www.bezrealitky.cz/
[9] https://github.com/apollographql/apollo-tooling
[10] https://www.apollographql.com/docs/ios/downloading-schema/

Odpoveďou na túto požiadavku je zoznam inzerátov. Inzerát je objekt, ktorý má niekoľko položiek. Ich skrátený zoznam spolu s ich významom je možné vidieť v tabuľke 5.5.

Názov	Typ	Popis
`id`	celé číslo	unikátny identifikátor inzerátu
`absoluteUrl`	reťazec	absolútna adresa inzerátu
`advertObject`	objekt	položky popísane v tabuľke 5.6
`images`	pole objektov	odkazy na fotky nehnuteľnosti

Tabuľka 5.5: Položky objektu reprezentujúci inzerát z Bezrealitky.cz

Názov	Typ	Popis
`street`	reťazec	ulica
`houseNumber`	reťazec	číslo domu
`city`	reťazec	mesto
`zip`	reťazec	poštové smerovacie číslo
`offerType`	pole textových reťazcov	rovnako ako v tabuľke 5.4
`estateType`	pole textových reťazcov	rovnako ako v tabuľke 5.4
`disposition`	pole textových reťazcov	rovnako ako v tabuľke 5.4
`surface`	celé číslo	výmera
`description`	reťazec	popis inzerátu od autora
`etage`	celé číslo	podlažie
`price`	celé číslo	cena nájmu
`charges`	celé číslo	mesačné poplatky
`deposit`	celé číslo	vratná kaucia
`balcony`	logická hodnota	prítomnosť balkóna
`terrace`	logická hodnota	prítomnosť terasy

Tabuľka 5.6: Položky objektu obsahujúceho špecifické informácie o inzeráte z Bezrealitky.cz

Ďalším pozorovaním bolo zistené, že v odpovedi na požiadavku zaslanú na GraphQL server sa nenachádza relevantná informácia o dostupnosti inzerátu. Položka na to síce existuje, ale nepoužíva sa. Pre získanie tejto informácie je preto potrebné stiahnuť HTML dokument z adresy `https://www.bezrealitky.cz/nemovitosti-byty-domy/<id>`. Tento naformátovaný dokument obsahuje všetky informácie o inzeráte. Keďže sa informácia nachádza v HTML dokumente, je potrebné použiť niektorú z techník pre extrakciu dát z HTML dokumentov, ktoré sú spomenuté v kapitole 3.1.

Sreality.cz

Sreality.cz[11] je jedným z mnohých projektov firmy Seznam. Súčasne patrí medzi najväčšie české realitné servery s 80 000 aktívnymi inzerátmi a priemernou návštevnosťou 1 300 000 reálnych používateľov [40]. Inzerátov s prenájmami je len okolo 8 000 a z toho väčšina nehnuteľností sa nachádza na území Českej republiky. Tento projekt sa nezameriava len na český trh. V širokom spektre inzerátov je možné nájsť okrem domácich ponúk aj nehnuteľnosti z 21 rôznych krajín sveta.

Okrem štandardných služieb ponúka podrobný prehľad o návštevnosti a zobrazení publikovaných inzerátov. Inzerát môžu majitelia zvýhodniť pomocou rôznych techník, napr. *topovaním* či prostredníctvom tipov, bannerových plôch a podobne. Prenajímateľom garantuje dlhodobú vysokú návštevnosť a potencionálnym nájomcom najaktuálnejšie informácie. Podobne ako predchádzajúce portály, aj tento ponúka funkciu Strážny pes, pomocou ktorého je možné dostávať upozornenia na nové inzeráty.

Pre získanie zoznamu inzerátov z tohto portálu je možné použiť požiadavku `GET` na adresu `https://www.sreality.cz/api/cs/v2/estates`. Všetky parametre sú nepovinné a majú číselné hodnoty. Niektoré parametre môžu vo svojej hodnote obsahovať viac čísel, potom sú oddelené znakom „|“ a zakódované pomocou URL kódovania. Prehľad použiteľných parametrov je možné vidieť v tabuľke 5.7.

[11] `https://www.sreality.cz/`

Názov	Popis	Poznámka
`category_main_cb`	typ nehnuteľnosti	`1` - byty `3` - projekty `5` - komerčné `2` - domy `4` - pozemky `6` - ostatné
`category_type_cb`	typ inzerátu	`1` - predaj `2` - prenájom `3` - dražby
`category_sub_cb`	dispozícia bytu	`2` - 1+kk `9` - 4+1 `12` - 6 a viac `3` - 1+1 `10` - 5+kk `16` - atypický `4` - 2+kk `11` - 5+1 `47` - izba
	typ domu	`33` - chata `39` - vila `35` - pamiatka/iné `40` - na kľúč `37` - rodinný `43` - chalupa `44` - poľnohospodárska usadlosť
`room_count_cb`	veľkosť domu (počet izieb)	`1`, `2`, `3`, `4`, `5` (5 a viac), `6` (atypický)
`building_condition`	stav objektu	`1` - veľmi dobrý `6` - novostavba `2` - dobrý `7` - k demolácii `3` - zlý `8` - pred rekonštrukciou `4` - vo výstavbe `9` - po rekonštrukcii `5` - developerské projekty
`building_type_search`	stavba	`1` - panel `2` - tehla `3` - ostatné
`ownership`	vlastníctvo	`1` - osobné `2` - družstevné `3` - štátne
`furnished`	vybavenie	`1` - áno `2` - nie `3` - čiastočne
`floor_number`	min. a max. poschodie	dve čísla oddelené „`|`“
`usable_area`	min. a max. úžitková plocha	dve čísla oddelené „`|`“
`estate_age`	vek inzerátu	`2` - deň `8` - posledných 7 dní `31` - posledných 30 dní
`locality_region_id`	kraj	napr. `1` - Jihočeský
`locality_district_id`	obec	napr. `1` - České Budějovice
`locality_country_id`	štát	`null` - Česká republika (predvolené) `10000` - zahraničie `10001` - všade `191` - Chorvátsko ...
`distance`	okolie v km	
`region`	upresnenie lokality	automatické dopĺňanie požiadavkou `GET` na `https://www.sreality.cz/api/cs/v2/suggest` s viacerými parametrami
`czk_price_summary_order2`	min. a max. cena	dve čísla oddelené „`|`“
`something_more1`	extra vybavenie	`3090` - balkón `3110` - terasa `3100` - lodžia `3120` - pivnica
`something_more2`	extra vybavenie	`3130` - bazén (len pre domy) `3140` - parkovanie `3150` - garáž
`something_more3`	extra vybavenie	`1820` - bezbariérový prístup `3310` - výťah
`something_more4`	extra informácie pre domy	`100111` - drevostavba `2123` - nízkoenergetický
`object_kind_search`	extra informácie pre domy	`1` - samostatný `2` - v bloku/radový
`object_type`	extra informácie pre domy	`1` - prízemný `2` - poschodový

Tabuľka 5.7: Parametre požiadavky zoznamu inzerátov z Sreality.cz

Odpoveď zo servera je vo formáte JSON. Najdôležitejšou položkou je pole objektov `estates`, ktoré obsahuje jednotlivé inzeráty. Inzerát obsahuje viacero údajov, z toho tie podstatné sú popísané v tabuľke 5.8.

Názov	**Popis**
`labelsAll`	pole značiek inzerátu, napr. `new_building`, `personal`, `brick`, `cellar`, `garage`, `furnished`
`locality`	adresa nehnuteľnosti
`seo`	adresa inzerátu skladajúca sa zo štyroch častí: `category_main_cb`, `category_type_cb`, `category_sub_cb` a adresa nehnuteľnosti
`hash_id`	unikátny identifikátor inzerátu
`price`	cena za mesačný nájom
`gps`	zemepisná šírka a dĺžka

Tabuľka 5.8: Položky odpovede servera na požiadavku zoznamu inzerátov z Sreality.cz

Pomocou tejto požiadavky nie je možné dostať všetky informácie. Pre detaily je potrebné vytvoriť požiadavku na adresu `https://www.sreality.cz/api/cs/v2/estates/<hash-id>` pomocou metódy `GET`. Jediná premenná v odkaze predstavuje unikátny identifikátor inzerátu, ktorý je možné získať z odpovede na predošlú požiadavku (položka `hash_id` v tabuľke 5.8). Najdôležitejšie položky odpovede sú popísane v tabuľke 5.9.

Názov	**Popis**
`codeItems`	informácie o inzeráte prostredníctvom kódových slov a číselných hodnôt (v rovnakej podobe ako sa vyskytujú v tabuľke 5.7, napr. `building_type_search`, `ownership`, `something_more1`, `something_more2`, `something_more3`)
`items`	štruktúrované informácie o inzeráte (pole objektov s položkami typ, názov, hodnota)
`locality`	adresa nehnuteľnosti
`meta_description`	popis inzerátu
`price_czk`	cena nájomného za mesiac

Tabuľka 5.9: Položky odpovede servera na požiadavku detailu inzerátu z Sreality.cz

Kapitola 6

Návrh aplikácie

Pre návrh aplikácie je potrebné urobiť analýzu existujúcich riešení a webových zdrojov, z ktorých sa budú čerpať informácie. Analýza jednotlivých portálov je popísaná v kapitole 5.2. Ďalším krokom je porovnanie jednotlivých zdrojov dát medzi sebou. Z toho vychádza návrh dátového modelu, ktorý je popísaný v kapitole 6.1. Súčasťou návrhu je aj definovanie funkcií, ktoré bude aplikácie spĺňať, čo je popísané v kapitole 6.2. Navrhnutá aplikácia agreguje len inzeráty s prenájmami bytov, nezameriava sa na inzeráty s predajom či spolubývaním.

6.1 Dátový model

Cieľom návrhu dátového modelu bolo zistiť, aké dáta portály poskytujú, v akom formáte, čo je a čo nie je možné spoľahlivo získať. Výsledok týchto zistení je možné vidieť v porovnávajúcej tabuľke 6.1.

Portál	Základné údaje							Ceny					Vybavenosť									Užitočné informácie				Dátumy		Typ nehnuteľnosti								
	Adresa	Dispozícia	Výmera	Podlažie	Popis	Fotky	Kontakt	Cena za mesiac	Poznámka k cene	Poplatky za mesiac	Vratná kaucia	Provízia realitnej kancelárie	Vybavenosť všeobecne	Práčka, umývačka riadu, chladnička	Balkón	Terasa	Lodžia	Pivnica	Internet	Parkovanie, garáž, výťah, bezbariér.	Energetická náročnosť budovy	Typ vlastníctva	Typ budovy	Stav objektu	Dátum pridania inzerátu	Dátum nasťahovania sa	Izba	Byt	Dom	Pozemok	Garáž	Kancelária	Nebytový priestor	Rekreačný objekt	Atypický
U	●	●	●	●	●	●	●	●	●	-	-	●	●	●	●	-	-	●	-	-	●	-	-	-	●	◐	-	-	◐	-	-	-	-	-	◐
B	●	●	●	●	●	●	●	●	-	●	●	◐	●	-	●	●	-	-	-	-	◐	●	●	-	-	●	-	●	●	●	●	●	●	●	-
S	●	●	●	●	●	●	●	◐	●	-	-	-	●	-	●	●	●	●	●	●	●	●	●	●	◐	-	◐	-	-	-	-	-	-	-	●

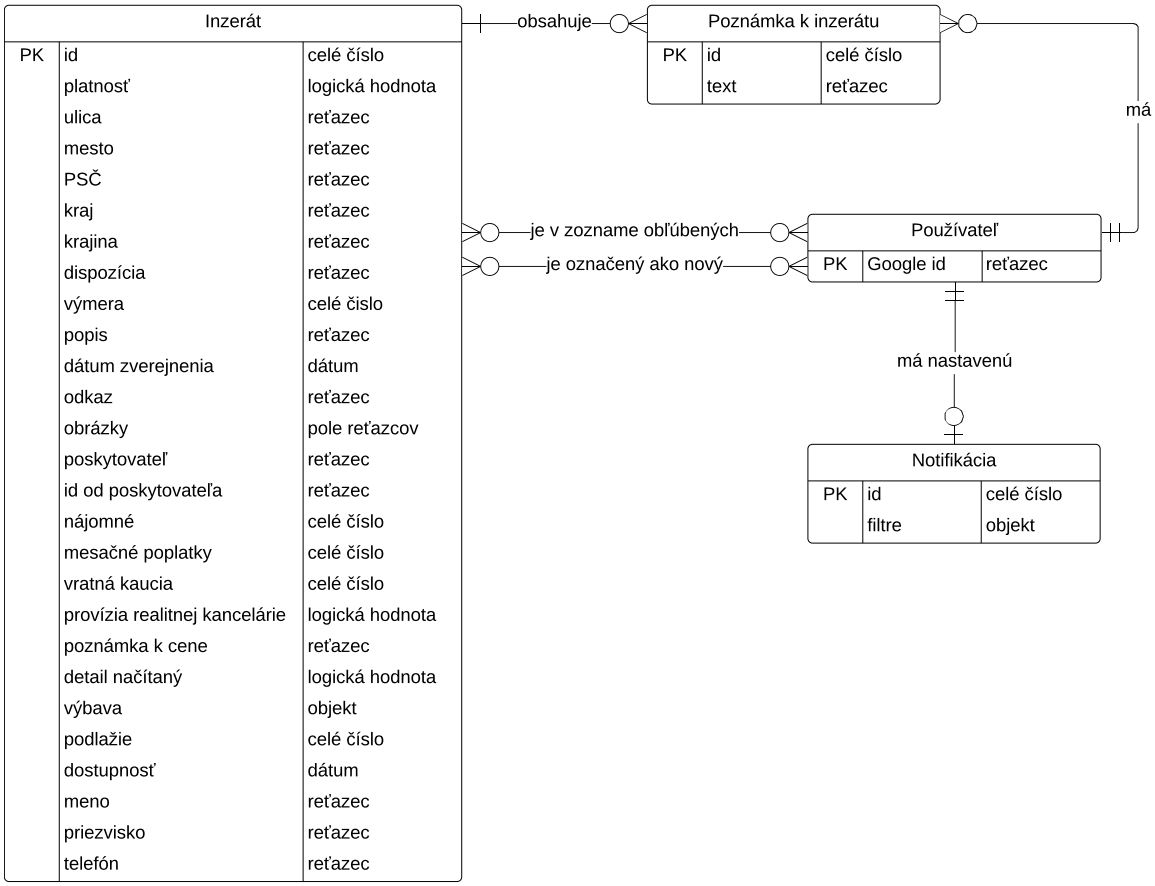
● = podporuje; ◐ = čiastočne podporuje; - = nepodporuje;
U = UlovDomov.cz; **B** = Bezrealitky.cz; **S** = Sreality.cz

Tabuľka 6.1: Porovnanie portálov podľa dostupnosti údajov v inzeráte

V tejto tabuľke je možné vidieť existenciu spoločných informácií, ktoré sú dostupné zo všetkých portálov. Každý portál poskytuje v inzeráte základné údaje o nehnuteľnosti, napr. adresu, dispozíciu či výmeru bytu. Avšak existujú aj také informácie, ktoré je možné získať len z určitého portálu. Niektoré portály napr. uvádzajú len ceny mesačného nájmu, iné aj poplatky za vratnú kauciu, províziu realitnej kancelárie, či iné mesačné poplatky (za energie a podobne). Niektoré portály napr. jednoznačne neuvádzajú, či daná nehnuteľnosť je, alebo nie je vybavená internetovým pripojením. Takéto informácie sú často len súčasťou popisu inzerátu. Z takého neštruktúrovaného textu sa informácie ťažko extrahujú, keďže neexistuje štandard, ktorý popisuje ich podobu.

Po tejto analýze sa zistilo, že dátový model, ktorý má aplikácia využívať, musí byť flexibilný. Niektoré údaje budú dostupné v každom inzeráte, niektoré budú zastúpené len v časti inzerátov. Vhodným kandidátom je tak dokumentová databáza (pozri sekciu 4.2.2).

Súčasťou návrhu dátového modelu je ER (*Entity-Relationship*) diagram znázornený na obrázku 6.1.



Obr. 6.1: ER diagram

Diagram obsahuje veľkú entitu *Inzerát*. V nasledujúcom texte budú postupne popísané niektoré atribúty tejto entity.

Platnosť inzerátu označuje stav inzerátu. Inzerát môže byť aktívny, teda platný, alebo neaktívny, teda neplatný. Nový inzerát je prirodzene aktívny a po čase sa stane neaktívnym, a to v prípade, že autor inzerátu ponuku stiahne, alebo sa nájde záujemca o túto ponuku.

Popis inzerátu predstavuje popis nehnuteľnosti, ktorý zadáva používateľ pri zverejňovaní inzerátu. Obsahuje veľa dôležitých informácií, avšak jedná sa o neštruktúrovaný text a tak extrakcia dát z tohto atribútu nie je vhodná.

Portál môže explicitne udávať *dátum zverejnenia* inzerátu, avšak nie je to pravidlom. V prípade, že tento údaj nie je dostupný, dátumom zverejnenia budeme rozumieť dátum „nájdenia“ inzerátu pri extrakcií dát z portálov.

Každý inzerát bude obsahovať *odkaz*, ktorý vedie na stránku portálu, kde bol inzerát zverejnený.

Každý správny inzerát obsahuje *fotky* nehnuteľnosti, čím dokáže prilákať viac záujemcov. Navrhovaný dátový model počíta len s odkazmi na obrázky, databáza neukladá média ako také.

Poskytovateľom rozumieme portál, kde bol inzerát zverejnený, napríklad UlovDomov.cz. Atribút *id od poskytovateľa* predstavuje unikátny identifikátor inzerátu priradený interne poskytovateľom. V rámci jedného poskytovateľa je tento údaj unikátny a môže slúžiť na zistenie staršieho výskytu v navrhovanej databáze. Inzerát v takom prípade nebude znova vložený do databázy.

Provízia realitnej kancelárie sa v portáloch často uvádza v štruktúrovaných dátach len ako logická hodnota. Jednoznačne je teda možné povedať, či prenajímateľ požaduje províziu alebo nie. Výška danej provízie je už potom vopred neznáma, prípadne je obsiahnutá v popise inzerátu, ktorý je, ako už bolo spomínané, neštruktúrovaný.

Poznámka k cene je atribút, do ktorého je možné zahrnúť všetky ostatné neštruktúrované informácie o cene inzerátu. Cena prenájmu sa takmer všade uvádza ako celé číslo, avšak ostatné poplatky, akými sú mesačné poplatky za energie, internet, vratná kaucia, či provízia realitnej kancelárie, sa uvádzajú niekedy ako neštruktúrovaný text, čím nie je možné jednoznačne určiť hodnoty týchto atribútov pomocou celých čísel.

Informácie, ktoré je možné získať o inzeráte, môžeme rozdeliť na najvyššej úrovni do dvoch kategórií: základne a detailné. Základné informácie je možné získať naraz jednou požiadavkou na API pre všetky inzeráty. Medzi základné informácie patrí napríklad adresa, dispozícia, fotky alebo výmera. Ďalšie informácie, nazvime ich detailné, je možné získať zväčša ďalším volaním API, a to pre každý inzerát zvlášť. Atribút *detail načítaný* teda indikuje, či inzerát obsahuje všetky informácie vrátane detailných, alebo je ešte potrebné tieto informácie získať. Viac informácií k tejto problematike je popísaných v sekcii 6.3.

Atribút *výbava*, podobne ako atribút *filtre* v prípade entity *Notifikácia* má dátový typ objekt. Takýto dátový typ je bežný v prípade dokumentových databáz. Môže obsahovať atribúty, podobne ako samotný dokument. Objekt *výbava* obsahuje atribúty, ktoré predstavujú jednotlivé položky výbavy, napr. chladnička, práčka, garáž a podobne. Hodnoty týchto atribútov sú logické hodnoty podľa toho, či konkrétna nehnuteľnosť týmito položkami disponuje alebo nie. Ak tento objekt neobsahuje konkrétnu položku výbavy, znamená to, že nevieme s istotou povedať, či je táto položka súčasťou výbavy nehnuteľnosti alebo nie. Naopak, ak objekt obsahuje takúto položku, vieme podľa jej logickej hodnoty určiť, či nehnuteľnosť má alebo nemá danú položku zahrnutú vo výbave. Podobne je to aj v prípade filtrov. Pre zapnutie notifikácie je potrebné nastaviť filtre, ktoré sú súčasťou objektu atribútu *filtre* entity *Notifikácia*.

Dostupnosť je dátum možného nasťahovania nájomcov do bytu. Tento údaj prenajímatelia neudávajú vždy. Zároveň je potrebné brať do úvahy aj situáciu, kedy je tento dátum starší než aktuálny dátum, a vhodne sa s tým vysporiadať na prezentačnej vrstve aplikácie.

Meno, *priezvisko* a *telefón* je kontakt na majiteľa inzerátu, teda pravdepodobne prenajímateľa nehnuteľnosti alebo realitnú kanceláriu.

Entita *Používateľ* má jediný atribút, ktorým je jednoznačná identifikácia používateľa v platforme Google. Autentifikácia používateľa sa nachádza v klientskej časti aplikácie (pozri sekciu 7.3). Pri prihlasovaní sa do aplikácie pomocou Google účtu musí používateľ súhlasiť s poskytnutím základných údajov, akými sú meno, priezvisko, e-mailová adresa, či fotka. Všetky tieto informácie si aplikácia vyžiada až v prezentačnej vrstve. Do databázy pre účely tejto aplikácie stačí ukladať Google ID používateľa.

Ako je možné vidieť na obrázku 6.1 zo vzťahu medzi entitami *Používateľ* a *Notifikácia*, používateľ môže mať vytvorenú v rovnakom čase maximálne jednu notifikáciu. Tá obsahuje informácie o filtroch, ktorými sa jednoznačne určí užší výber inzerátov spomedzi všetkých novo vložených, a tie sa uložia do zoznamu nových pre daného používateľa. Používateľ má zároveň prístup k zoznamu inzerátov, ktoré si pridal medzi svoje obľúbené.

6.2 Funkcie aplikácie

Pred návrhom architektúry aplikácie je potrebné určiť, čo bude aplikácia vykonávať a akými funkciami bude aplikácia disponovať. Na obrázku 6.2 je možné vidieť diagram prípadov použitia.

Používateľ sa môže ocitnúť v dvoch rolách. Ak príde na stránku po prvýkrát, je neprihlásený. Okrem prihlásenia môže inzeráty vyhľadávať a prehľadávať. Inzeráty je možné filtrovať na základe určitých parametrov a zoradiť podľa niekoľkých kritérií. Taktiež je možné vybrať niekoľko inzerátov na porovnanie, a potom prejsť na stránku samotného porovnania. Po kliknutí na konkrétny inzerát sa zobrazí jeho detail. Po zobrazení detailu inzerátu, systém automaticky označí tento inzerát ako videný pre daného používateľa. Používateľ tak môže učiniť aj sám, a to aj v prípade, ak detail inzerátu nezobrazí.

Po prihlásení používateľ dostáva možnosť pridávať a odoberať jednotlivé inzeráty do zoznamu obľúbených. Následne má možnosť zobraziť spomedzi všetkých inzerátov len jeho obľúbené. Prihlásený používateľ môže pridávať osobné poznámky a subjektívne názory týkajúce sa inzerátu, ktoré nebudú verejné.

Ďalšou dôležitou funkciou je nastavenie notifikácií pre nové inzeráty. Používateľ musí ako prvé nastaviť parametre filtrovania inzerátov. Potom pomocou týchto filtrov môže zapnúť notifikácie. Navrhovaná aplikácia bude vedieť používateľa upozorniť na novú ponuku, ktorá bude vyhovovať jeho požiadavkám. Používateľ bude mať možnosť, podobne ako pri obľúbených inzerátoch, zobraziť len tie nové, na ktoré ho upozorní notifikácia.

Obr. 6.2: Diagram prípadov použitia

6.3 Architektúra

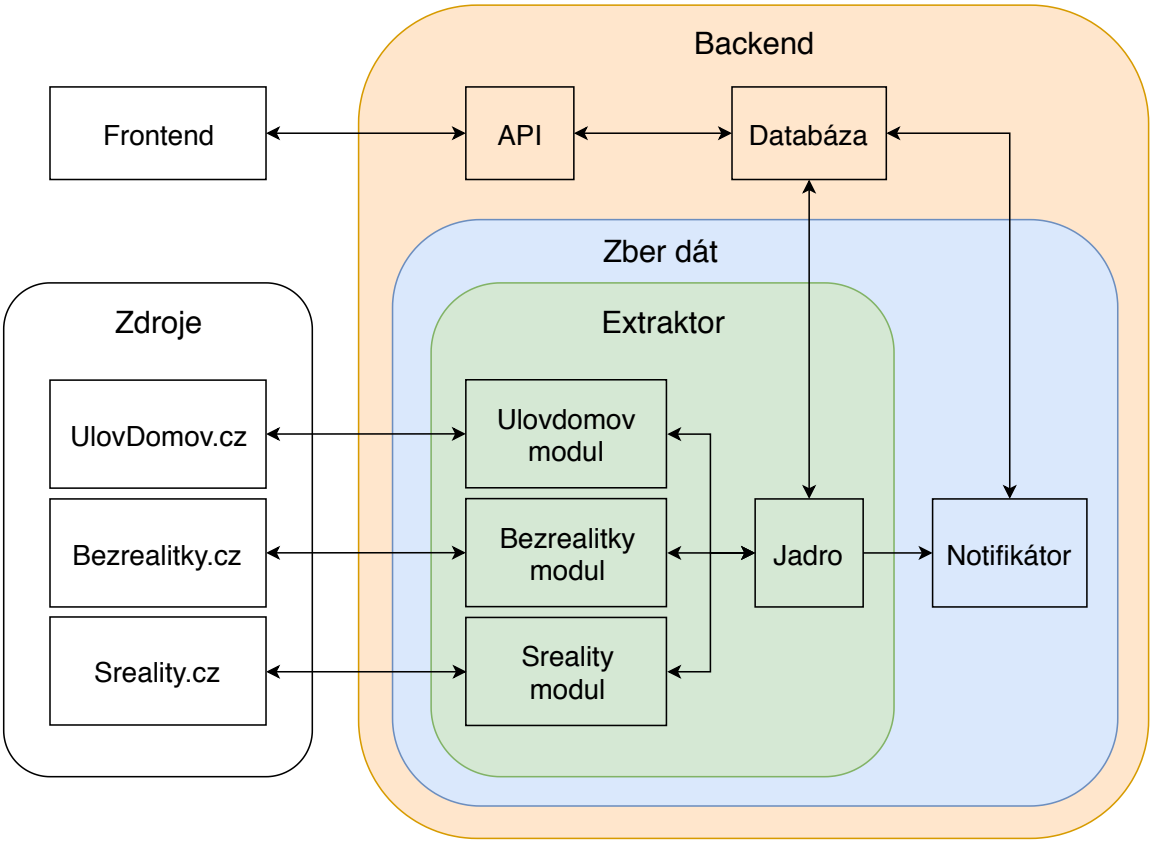
Dôležitou otázkou, ktorú je potrebné riešiť pri návrhu architektúry, je spôsob zbierania dát. Sú dve možnosti ako a kedy zbierať dáta. Prvou možnosťou je získavať dáta až za behu podľa požiadaviek používateľa. Až po tom, čo používateľ odošle parametre vyhľadávania, pošle sa požiadavka na serverovú časť aplikácie (backend). Tam začne získavanie dát zo zdrojových portálov a po chvíli sa výsledok pošle klientskej časti aplikácie (frontend). Tento spôsob sa nazýva synchrónny a kvôli blokovanému volaniu je pomalý. Používateľ musí čakať niekoľko sekúnd, možno až minút na odpoveď. Výhodou sú vopred známe parametre vyhľadávania, čím je možné presne špecifikovať požiadavku na zdrojový portál. Tento postup sa dá zrýchliť zavedením paralelizmu a *cachovania*. Paralelizmus je dnes možné jednoducho riešiť pomocou *serverless* technológie[1]. Ďalšou výhodou je, že takýmto spôsobom sa do databázy aplikácie uložia len tie inzeráty, ktoré sú reálne vyhľadávané.

Alternatívou k tomuto prístupu je zbieranie dát pravidelne. V databáze sú uložené úplne všetky inzeráty, aj také, ktoré by žiaden používateľ nevyhľadal. Tento spôsob je vhodnejší pre implementáciu navrhnutej aplikácie. Výhodou je, že frontend dostáva dáta priamo z databázy aplikácie, a nečaká na dokončenie extrakcie dát zo všetkých webových zdrojov. Je to rýchlejší prístup, používateľ nečaká zbytočne dlho za výsledkom vyhľadávania.

Pri pravidelnom ukladaní všetkých inzerátov je potrebné rozlišovať dva stavy inzerátu. Prvý stav predstavuje inzerát len so základnými informáciami. Pomocou jednej požiadavky na jeden webový zdroj je možné získať len základné informácie o všetkých inzerátoch z daného portálu. Pre získanie detailov je potrebné poslať ďalšiu požiadavku, a to pre každý inzerát zvlášť. Ak sú načítané všetky dostupné informácie o inzeráte, jeho stav sa zmení. Je tak možné rozlíšiť inzeráty, pre ktoré je potrebné načítať detaily, a ktoré už tieto detaily majú uložené v databáze aplikácie.

Celú aplikáciu je možné rozdeliť na niekoľko komponentov. Tieto komponenty spolu s ich vzťahmi je možné vidieť na obrázku 6.3.

[1] https://techcrunch.com/2015/11/24/aws-lamda-makes-serverless-applications-a-reality/



Obr. 6.3: Architektúra aplikácie - komponenty a ich vzťahy

Frontend predstavuje webovú stránku, cez ktorú používateľ komunikuje so zvyškom aplikácie. Využíva rozhranie API formou požiadavka-odpoveď. API spracováva požiadavky používateľa, potrebné dáta získava z databázy a taktiež do databázy zapisuje.

Dôležitou súčasťou backendu je zber dát (angl. *data acquisition*). Zahŕňa nástroje *Extraktor* a *Notifikátor*.

Extraktor je zodpovedný za získavanie dát z webových zdrojov. Pozostáva z *Jadra* a modulov. Každý modul zaisťuje získavanie dát z konkrétneho zdroja dát, tzn. pre každý zdroj dát existuje jeden modul. *Jadro* predstavuje spoločnú časť kódu, ktorá sa má vykonať pre ľubovoľný modul. Dáta získané modulmi ukladá do databázy. Návrh počíta s možnosťou pridania nového modulu pre nový zdroj dát. Moduly môžu získavať dáta buď z formátu JSON priamo, alebo z dokumentu HTML pomocou wrappera. Všetky uložené dáta sú štruktúrované a majú spoločnú podobu.

Extraktor po dokončení práce predáva zoznam nových inzerátov *Notifikátoru*. Ten zistí, ktoré z týchto inzerátov vyhovujú používateľom, ktorí majú zapnuté notifikácie. Svoje zistenia následne premietne do databázy.

Kapitola 7

Implementácia

Implementácia navrhnutej aplikácie prebiehala v iteráciách. Cieľom bolo v čo najkratšom čase vytvoriť produkt, ktorý je funkčný, hoci by bola funkcionalita z veľkej časti obmedzená. V ďalších fázach vývoja sa postupne aplikácia zlepšovala, pridávali sa nové funkcie, vlastnosti, aplikácia sa stávala robustnejšou. Takýto princíp si často osvojujú začínajúce firmy (angl. *startup*). Pre nich je nevyhnutné zistiť skutočnú hodnotu vyvíjaného produktu a potenciálny rast na trhu čo najskôr. Robia tak vytvorením minimálneho produktu (angl. *Minimum Viable Product*, MVP), ktorý dokáže poslúžiť ako demonštrácia finálnej verzie [30].

V nasledujúcich troch sekciách je postupne popísaný spôsob implementácie jednotlivých komponentov z obrázku 6.3. V ďalšej sekcii sa potom nachádza návod na použitie serverovej časti aplikácie.

7.1 Zber dát

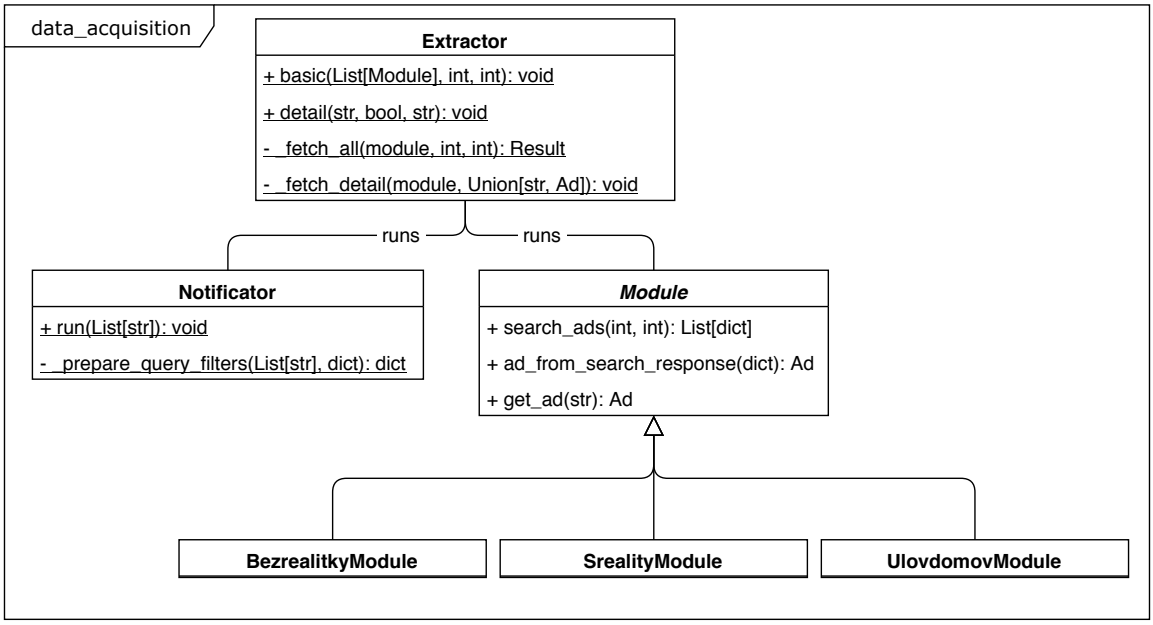
Zber dát je kľúčovou súčasťou aplikácie. Bez dát by implementovaná aplikácia nemala zmysel. Dáta sa extrahujú pomocou skriptov napísaných v jazyku **Python**. Na obrázku 7.1 je diagram tried napísaný v jazyku UML, ktorý popisuje, aké triedy sa na zbere dát podieľajú.

Trieda `Extractor` predstavuje jadro zberu dát. Prostredníctvom svojich metód vykonáva úlohy spoločné pre všetky moduly. Moduly sú implementované každý zvlášť v samostatnom súbore, resp. Python module. Každý modul obsahuje triedu zdedenú z abstraktnej triedy `Module`. Tá definuje spoločné rozhranie pre všetky moduly, pričom triedy zdedené z tejto rodičovskej triedy musia implementovať tri metódy (pozri obr. 7.1). Výstupom týchto metód sú objekty triedy `Ad`, ktorá predstavuje model inzerátu. `Extractor` pomocou štandardného balíčka `importlib` importuje konkrétny modul do svojho skriptu a priamo z kódu volá jednotlivé metódy daného modulu. Výsledkom týchto volaní je inzerát alebo skupina inzerátov, ktoré sa následne vložia do databázy.

Jednotlivé moduly implementujú abstraktné metódy vlastným spôsobom. Avšak prakticky sa vždy vyvolá API požiadavka a následne prebehne transformácia odpovede do spoločnej podoby. Odpoveď býva vo formáte JSON alebo HTML. Spoločnú podobu reprezentuje už spomínaná modelová trieda `Ad`.

Pre ukladanie inzerátov bola zvolená dokumentová databáza **MongoDB**[1], aj kvôli dôvodom spomenutým v sekcii 4.1. Dokumentová databáza je vhodná pre tento typ aplikácie, pretože inzeráty sa štrukturalizujú do dokumentov. MongoDB poskytuje flexibilitu dátového modelu. Čiastočné modifikácie existujúcich dokumentov sú časovo nenáročné.

[1] https://www.mongodb.com/



Obr. 7.1: Diagram tried systému zberu dát

Pre prácu s MongoDB databázou v jazyku Python bol zvolený balíček **MongoEngine**[2], ktorý slúži na objektovo dokumentové mapovanie (ODM). Aplikuje podobný princíp ako objektovo relačné mapovanie (ORM) pri relačných databázach. Pomocou deklaratívneho API je možné skriptu v jazyku Python napovedať, aké dokumenty databáza obsahuje, v akých kolekciách sa nachádzajú, z čoho sú zložené a podobne. Výsledkom sú triedy, ktoré je možné inštanciovať. Inštancie predstavujú dokumenty, s ktorými je možné pracovať pomocou metód, čítať ich atribúty a podobne.

Model inzerátu je navrhnutý tak, aby spoločné atribúty dostupné na každom portáli mali jasne definovaný názov a dátový typ. Atribúty, ktoré sa líšia pre každý portál, sú ukladané do generického dátového typu, akým je napríklad objekt. Príklad takého atribútu je vybavenie nehnuteľnosti, resp. bytu (pozri sekciu 6.1).

Adresa inzerátu je v každom webovom zdroji uložená inak. Aby implementovaná aplikácia dokázala s adresou efektívne narábať, bolo nutné ju previesť do jednotného tvaru. Využíva sa pre to projekt **OpenStreetMap**[3]. Poslaním tohto projektu je tvorba, skladovanie a udržiavanie množstva geografických dát, ktoré sú voľne dostupné. Voľne dostupný je aj zobrazovací nástroj **Nominatim**[4] v podobe webovej stránky, ktorý poskytuje aj webové API. Práve pomocou tohto nástroja jednotlivé moduly transformujú dostupné informácie na pevne definovanú štruktúru. Nielenže je výsledkom jednotný tvar adresy, zároveň je možné získať doplňujúce informácie, napríklad o poštovom smerovacom čísle, či kraji.

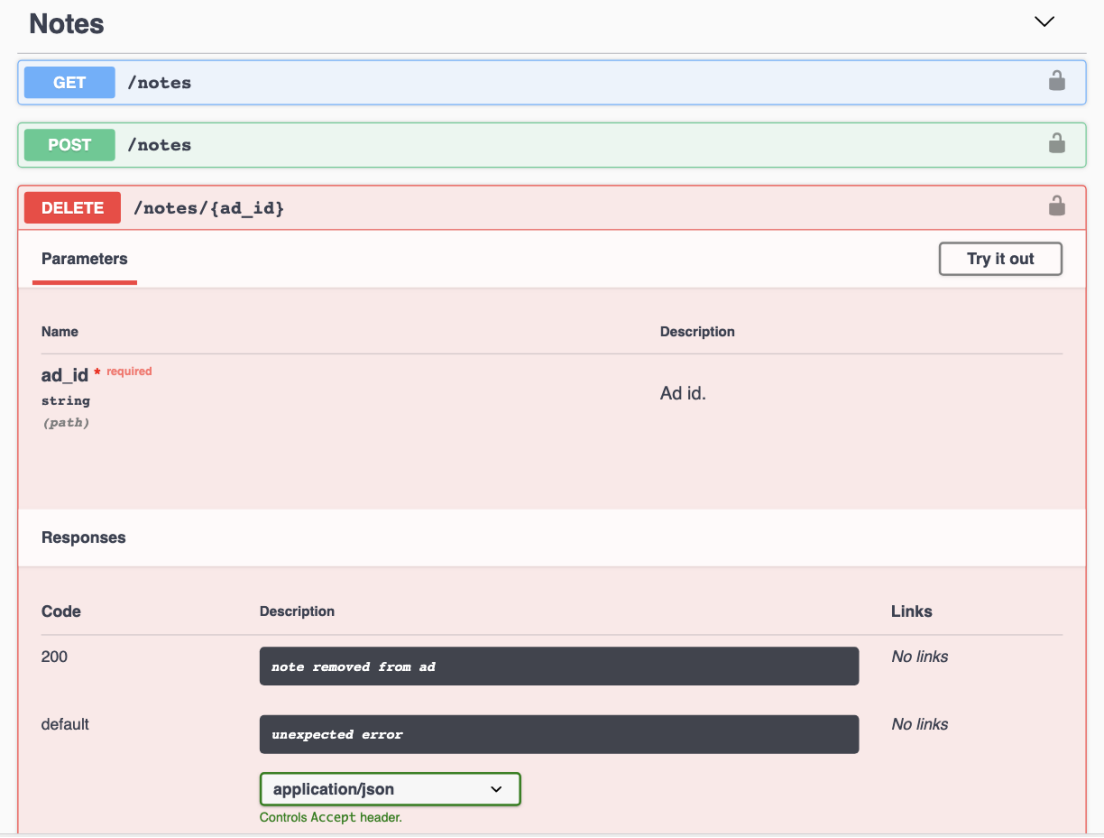
Na API nástroja Nominatim sa odošle požiadavka, ktorej parametre sú názov ulice a názov mesta. Tieto informácie je možné väčšinou získať z každého inzerátu, či už priamo alebo pomocou regulárneho výrazu. V prípade, že sa daná lokalita nájde, v odpovedi sú vrátené všetky dostupné informácie o nej.

[2] http://mongoengine.org/
[3] https://www.openstreetmap.org/about
[4] https://nominatim.openstreetmap.org/

7.2 Aplikačné rozhranie

Aplikačné rozhranie serverovej časti (API) slúži pre komunikáciu klientskej časti aplikácie so serverovou časťou. Rozhranie využíva architektúru REST. Celé API je definované deklaratívnym popisom vo formáte YAML podľa špecifikácie **OpenAPI**[5]. V jednom súbore sa nachádzajú informácie o všetkých dostupných koncových bodoch, operáciách, vstupných parametroch, či o formáte odpovedí. Súčasťou je aj popis autentifikácie a dokumentácia všetkých prvkov. Najprv sa teda API navrhne a až potom z tohto návrhu vznikne kód. Opačný prístup by bol najprv napísať kód a potom dodatočne tento kód zdokumentovať. Oba prístupy majú svoje výhody aj nevýhody.

Ak sa API najprv zdokumentuje a popíše napríklad podľa špecifikácie OpenAPI, je následne možné použiť nástroje pre automatické generovanie jadra servera alebo generovanie klientov, resp. knižníc klientov, a to v desiatkach rôznych programovacích jazykov. Projekt **Swagger**[6] dokonca umožňuje automatické generovanie interaktívnej dokumentácie API, pomocou ktorej si používateľ API môže vyskúšať vyvolanie jednotlivých operácií priamo vo svojom webovom prehliadači. Náhľad takejto webovej stránky je zobrazený na obrázku 7.2.



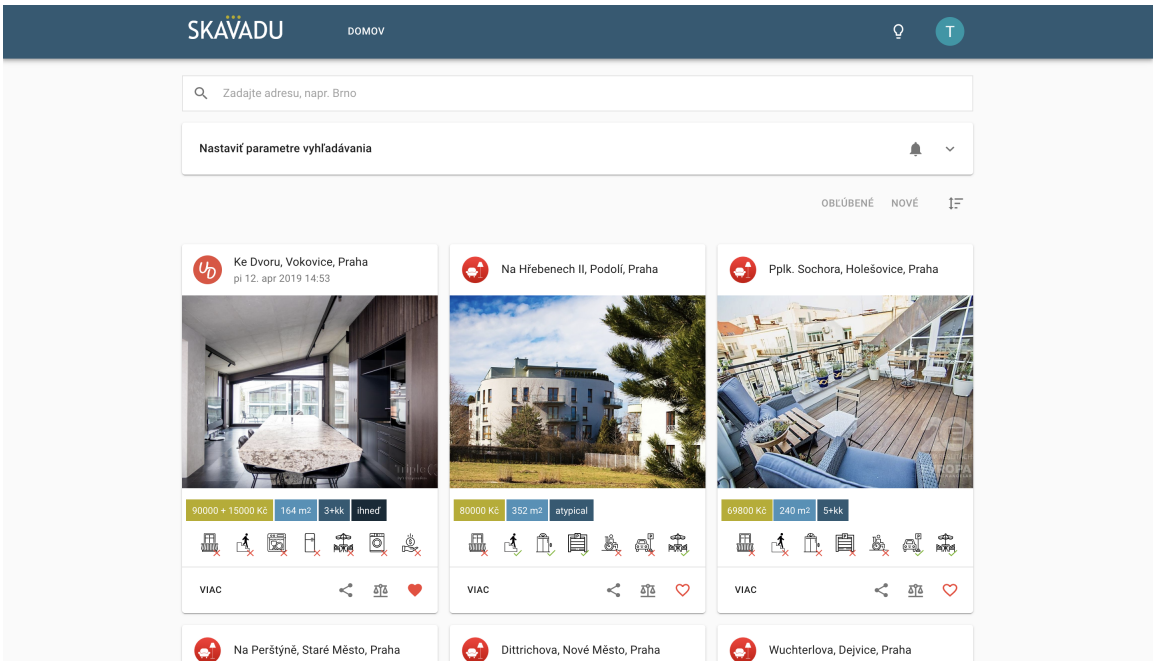
Obr. 7.2: Náhľad vizualizácie interaktívneho API **Swagger UI**[7]

[5] https://swagger.io/docs/specification/about/
[6] https://swagger.io/
[7] https://swagger.io/tools/swagger-ui/

V práci je použitý Python framework **Connexion**[8] v spolupráci s frameworkom **Flask**[9], ktoré automaticky spracovávajú prichádzajúce požiadavky podľa špecifikácie OpenAPI.

7.3 Klientská časť

Klientská časť aplikácie (frontend) je implementovaná pomocou knižnice **React**[10] napísanej v jazyku **JavaScript**. React je orientovaný komponentovo. Komponenty dokáže vykresľovať asynchrónne, čím je možné jednoducho vytvárať dynamické webové stránky. Pre prácu so stavom prezentačnej vrstvy aplikácie bola zvolená technológia **Redux**[11]. Vzhľad komponentov je založený na princípoch vizuálneho jazyka **Material Design**[12] od firmy Google. Ukážku vytvorenej webovej aplikácie je možné vidieť na obrázku 7.3. Ďalšie snímky je možné nájsť v prílohe tejto práce.



Obr. 7.3: Ukážka vytvorenej webovej stránky

Prihlasovanie používateľa je výhradne na strane frontendu. Používateľ sa môže prihlásiť pomocou svojho Google účtu. Pri prihlasovaní prebieha komunikácia medzi webovou stránkou a serverom Google. Ak je prihlásenie úspešné, webová stránka si do úložiska prehliadača uloží základne informácie o prihlásenom používateľovi. Súčasťou týchto informácií sú aj dva tokeny JWT[13] (*JSON Web Token*). *Access* token slúži na overenie prihláseného používateľa. Ten sa posiela v hlavičke `Authorization` s každou požiadavkou klienta na server. Na strane servera prebieha overenie platnosti tokenu. Ak je token neplatný alebo

[8] https://github.com/zalando/connexion
[9] http://flask.pocoo.org/
[10] https://reactjs.org/
[11] https://redux.js.org/
[12] https://material.io/design/
[13] https://jwt.io/

exspiroval, server požiadavku odmietne s kódom `401`. Ak klient zachytí takúto odpoveď, na token zabudne a používateľa prinúti sa znovu prihlásiť. Token exspiruje po hodine. Lepším riešením je použiť druhý token, tzv. *refresh* token, k získaniu nového access tokenu. Refresh token má taktiež svoju expiračnú dobu, tá je obvykle dlhšia, a tak môže byť používateľ prihlásený viac ako hodinu.

Používateľ môže vidieť v jednom čase na obrazovke len malú skupinu inzerátov. Je zbytočné načítať všetky inzeráty zo servera naraz. Inzeráty sa načítajú postupne po menších skupinách. Pri posielaní požiadaviek na REST API sú využité parametre pre stránkovanie. Keď sa používateľ pohne na obrazovke smerom dole, načítajú sa ďalšie inzeráty. Tým sa zrýchľujú reakcie aplikácie na požiadavky používateľa, keďže nedochádza k prenosu veľkého objemu dát.

7.4 Použitie serverovej časti

Skripty pre zber dát je možné spúšťať centrálne pomocou jedného skriptu `manager.py`. Je to ďalší skript napísaný v jazyku Python. Automatické generovanie príkazov pre používateľské rozhranie v príkazovom riadku (angl. *Command-line interface*, CLI) zabezpečuje knižnica **Python Fire**[14] od firmy Google. Táto knižnica umožňuje jednoduché vytváranie CLI z ľubovoľného objektu. V tabuľke 7.1 je zoznam príkazov, pomocou ktorých je možné manipulovať s modulmi.

Príkaz	**Poznámka**
`python manager.py modules new --name NAME [--enabled ENABLED]`	pridanie nového modulu
`python manager.py modules list`	zobrazenie všetkých modulov
`python manager.py modules get --name NAME`	získanie informácií o danom module
`python manager.py modules delete --name NAME`	zmazanie modulu
`python manager.py modules enable --name NAME`	zapnutie modulu
`python manager.py modules disable --name NAME`	vypnutie modulu

Tabuľka 7.1: Príkazy pre manipuláciu s modulmi

Po vytvorení nového modulu sa automaticky vygeneruje Python modul s predpripravenými metódami, ktorým je potrebné napísať vlastnú implementáciu. Zároveň sa pridá záznam o module do databázy. Modul je možné ľubovoľne vypínať a zapínať, a to napríklad pre potreby nutnej aktualizácie v prípade nájdených chýb.

Pomocou manažéra je možné spúšťať aj samotnú extrakciu dát prostredníctvom modulov. Prehľad dostupných príkazov je v tabuľke 7.2.

V prípade základných informácií je možné určiť zoznam modulov, prostredníctvom ktorých má prebehnúť samotná extrakcia. Ak sa tento parameter nezadá, extrakcia prebieha zo všetkých webových zdrojov, ktorých moduly sú v databáze povolené.

Detailné informácie je možné získať pre všetky inzeráty, ktoré sú uložené v databáze, a ktorých detail ešte nebol načítaný. To sa zistí pomocou príznaku *detail načítaný*. Toto nastavenie je možné prepísať pomocou parametra `force` pri spúšťaní extrakcie detailných informácií.

Detail je možné získať aj pre jediný inzerát. Slúži na to parameter `id`. Ak je jeho hodnota zložená z dvoch častí oddelených dvojbodkou, extrakcia prebieha bez ohľadu na

[14] https://github.com/google/python-fire

to, či sa daný inzerát v databáze vyskytuje alebo nie. V takom prípade je prvá časť pred dvojbodkou názov poskytovateľa, resp. názov modulu a druhá časť predstavuje unikátny interný identifikátor inzerátu generovaný poskytovateľom.

Príkaz	**Poznámka**
`python manager.py modules extract basic`	získanie základných informácií zo všetkých modulov
`python manager.py modules extract basic *MODULES`	získanie základných informácií len z určitých modulov
`python manager.py modules extract basic --page PAGE --limit LIMIT`	stránkovanie
`python manager.py modules extract detail`	získanie detailných informácií o všetkých uložených inzerátoch zo všetkých modulov
`python manager.py modules extract detail --id AD_ID`	získanie detailných informácií o jednom inzeráte uloženom v databáze
`python manager.py modules extract detail --id PROVIDER:ID`	získanie detailných informácií o jednom inzeráte ešte neuloženom v databáze
`python manager.py modules extract detail --force True`	získanie detailných informácií o všetkých uložených inzerátoch bez ohľadu na príznak *detail načítaný*

Tabuľka 7.2: Príkazy pre spúšťanie extrakcie dát

Pre spustenie API a databázy sa odporúča mať nainštalovanú platformu **Docker**[15], ktorá stavia na izolácií aplikácií do tzv. *kontajnerov*. V súbore `docker-compose.yml` sú definované služby pre REST API a MongoDB databázu. Po spustení kontajnerov pomocou príkazu `docker-compose up --build` je API dostupné na adrese `0.0.0.0:8080` a databáza na adrese `localhost:27017`. Frontend je možné spustiť pomocou príkazu `npm start`.

[15] https://www.docker.com/

Kapitola 8

Testovanie

Táto kapitola sa zaoberá testovaním aplikácie, pričom kladie dôraz na extrakciu dát z vybraných realitných portálov. Popisuje aj praktické problémy, ktoré sa počas testovania vyskytli.

8.1 Priebeh testovania

Všetky úrovne implementovanej aplikácie boli testované postupne. Implementácia prebiehala v niekoľkých iteráciách s cieľom vytvoriť za čo najkratší čas minimálne funkčnú aplikáciu a postupne pridávať funkcionalitu (pozri kapitolu 7).

Po pridaní nových atribútov inzerátov sa celá aplikácia otestovala, od získania údajov z reálnych inzerátov, po uloženie do databázy, a následné zobrazenie v klientskej časti aplikácie. Toto testovanie prebiehalo vždy na vzorke 100 inzerátov z daného portálu.

Inzeráty vo fáze testovania boli získavané z troch najznámejších realitných portálov: *Ulovdomov.cz*, *Bezrealitky.cz* a *Sreality.cz*. Po podrobnej analýze všetkých troch portálov bola určená najvhodnejšia metóda extrakcie dát (pozri sekciu 5.2).

Tak ako už bolo spomenuté, aplikácia je zameraná na inzeráty s prenájmami bytov. Ide teda o podmnožinu všetkých inzerátov dostupných na webových stránkach vyššie spomenutých portálov. Aplikácia teda nepočíta s inzerátmi, ktoré ponúkajú predaj bytov či spolubývanie. Počty dostupných inzerátov z jednotlivých portálov sú prehľadne znázornené v tabuľke 8.1.

Názov portálu	**Všetky inzeráty**	**Inzeráty s prenájmom bytu**
`Ulovdomov.cz`	8 290	7 822
`Bezrealitky.cz`	7 049	2 257
`Sreality.cz`	86 291	9 379

Tabuľka 8.1: Počet dostupných inzerátov na testovaných portáloch z dňa 16.5.2019

Extrakcia dát prebiehala v dvoch fázach (pozri sekciu 6.3). Najprv boli získané základné informácie o inzerátoch. Pre každý zdroj inzerátov sa vytvorila jedna požiadavka na API alebo webový server daného portálu. Po získaní zoznamu inzerátov spolu s ich základným informáciami muselo dôjsť k transformácií adresy na jednotný tvar (pozri sekciu 7.1). To si vyžaduje vytvorenie ďalšej požiadavky pre každý inzerát zvlášť. Pre extrakciu základných informácií na vzorke 50 inzerátov je teda potrebné dokopy vytvoriť 51 HTTP požiada-

viek. Počet takto získaných inzerátov je možné regulovať pomocou atribútov slúžiacich na stránkovanie výsledkov.

V druhej fáze sa testovalo získavanie detailov, ktoré prebiehalo podobne ako získavanie základných informácií. Tento zber dát sa aplikoval na všetky inzeráty uložené v databáze.

Ak pri extrakcii dát došlo k chybe, skript preskočil problematický inzerát (pozri sekciu 8.2.4). Dôvodom mohlo byť napríklad nenájdenie povinného atribútu alebo nečakané hodnoty. Chyba sa zapísala do žurnálu. Po skončení extrakcie sa v rámci testovania tento žurnál preštudoval, zistilo sa kde dochádza k chybám a následne boli tieto problémy odstránené.

Pri opravovaní chýb sa využili aj ladiace výpisy (angl. *debug logs*) do štandardného výstupu systému. Tie je možné zapnúť nastavením premennej prostredia (angl. *environment variable*) s názvom `DEBUG` na hodnotu `1`. Súčasťou ladiacich výpisov sú všetky potrebné informácie, ktoré pomáhajú pri spätnej analýze vzniknutej chyby. Patria sem argumenty posielaných požiadaviek, odpovede zo serverov, medzikroky rôznych výpočtov a funkcií.

Súčasťou testovania neboli žiadne automatizované testy. Validácia prebiehala ručne porovnávaním získaných dát s informáciami, ktoré boli uvedené na originálnych stránkach inzerátu. Pre tieto dôvody sa testovalo len na desiatkach inzerátov z každého portálu.

Testovanie prebiehalo počas niekoľkých dní, a tak bolo možné otestovať aj správnosť neaktuálnych inzerátov. Po niekoľkých dňoch môže dôjsť k stiahnutiu inzerátu z portálu. Dôvodov môže byť niekoľko – najčastejšie sa nájde záujemca o danú ponuku. V tom prípade portál prestane poskytovať údaje o danom inzeráte. Informácie nie je možné získať ani z ich rozhrania serverovej časti. Na webových stránkach portálov je tento inzerát zobrazený ako neaktuálny. Implementovaná aplikácia je schopná sa s touto situáciou vysporiadať a v tomto prípade označí v databáze daný inzerát ako neaktuálny. Všetky doposiaľ získané informácie o danom inzeráte sú ale zachované.

8.2 Praktické problémy

Počas testovania aplikácie sa vyskytlo niekoľko otázok a problémov, ktoré bolo potrebné riešiť. V nasledujúcich sekciách sú spomenuté tie najdôležitejšie z nich.

8.2.1 Duplicity

Aplikácia agreguje inzeráty z viacerých informačných webových zdrojov na jednom mieste. Nie je vylúčené, že jeden konkrétny inzerát s jednou konkrétnou ponukou môže byť uverejnený na viacerých portáloch zároveň. Aplikácia by ideálne nemala zobrazovať duplicitné inzeráty. Je potrebné určiť unikátny identifikátor inzerátu, ktorý by bol platný naprieč všetkými zdrojmi.

Na konštrukciu identifikátoru je vhodné použiť len údaje, ktoré sú spoločné pre všetky zdroje, a teda všetky zdroje musia tieto údaje poskytovať. Z tabuľky 6.1 vyplýva, že je možné použiť kombináciu niektorých základných údajov. Dispozícia, výmera, podlažie a kontakt by mali byť uvedené vo všetkých zdrojoch rovnako. Toto pozorovanie je možné využiť na konštrukciu unikátneho identifikátoru.

Je pravdepodobné, že bude dochádzať k situáciám, kedy majiteľ alebo realitný maklér bude v ten istý čas ponúkať na prenájom niekoľko bytov, ktoré sa môžu zhodovať vo všetkých spomenutých parametroch. Ak by identifikátor zahŕňal aj adresu nehnuteľnosti, stále nie je možné tento problém spoľahlivo vyriešiť. Teoreticky môže nastať situácia, kedy dva byty s rovnakými parametrami sídlia na zhodnej adrese.

Pri testovaní dokonca nastala situácia, kedy z jedného webového zdroja bolo stiahnutých niekoľko nápadne podobných inzerátov. Inzerované byty sa pravdepodobne nachádzali v rovnakom bytovom dome. Ich fotky, popis, adresa a dispozícia boli takmer identické.

V rámci jedného portálu je možné duplicity detekovať pomocou identifikátoru alebo absolútnej adresy inzerátu. Tieto atribúty sú vždy dostupné. Naprieč viacerými portálmi sa ale duplicity vylúčiť úplne nedajú.

Jediným spoľahlivým riešením je potom spolupráca s používateľmi aplikácie. Ak by používateľ natrafil na inzerát, ktorý sa v ponuke opakuje, mohol by mať možnosť nahlásiť tento inzerát. Následne by došlo k posúdeniu, či sa naozaj jedná o duplicitný inzerát a ponuka by sa stiahla.

8.2.2 Adresa

Všetky informácie o adrese sa získavajú pomocou API tretej strany (pozri sekciu 7.1), s čím súvisia dva problémy.

Po prvé, API môže byť nedostupné. Server nemusí vždy zvládať spracovávať všetky požiadavky, ktoré dostane, resp. ktoré naň boli odoslané.

Druhým problémom je, že nie vždy sa podarí adresu preložiť, resp. nájsť lokalitu odpovedajúcu danej adrese.

Dobrým príkladom je ulica Doktorky Milady Horákové v obci Poděbrady. V takomto formáte Nominatim nie je schopný lokalizovať túto ulicu. Ako názov ulice je potrebné zadať „Dr. Horákové Poděbrady“. Extraktor tak zlyhá pri získaní lokality a keďže adresa je povinným parametrom inzerátu, tento inzerát preskočí. Aplikácia ho neuvažuje, nie je teda zobrazený ani používateľom.

8.2.3 Paralelizmus

Ďalším problémom je rýchlosť získavania dát. Keďže implementovaná platforma extrakcie dát využíva webové API tretích strán, napríklad pre získavanie jednotnej formy adresy, rýchlosť vykonávania skriptov úzko súvisí s rýchlosťou pripojenia k internetu.

Celý proces spracovania inzerátov spomaľuje aj transformácia adresy na jednotný tvar. Využíva sa na to nástroj Nominatim (pozri sekciu 7.1). Podmienky používania tohto nástroja zakazujú hromadné posielanie požiadaviek [1]. Je možné poslať maximálne jednu požiadavku za sekundu. Ak to používateľ poruší, má na nejaký čas zablokovaný prístup k tomuto nástroju.

Pre tento dôvod nie je možné použiť žiadnu techniku paralelného spracovania, ktorá by to urýchlila. Riešením je použitie iného nástroja bez takýchto obmedzení. Nevýhodou je, že takáto služba býva spoplatnená.

Ak by sa tento problém vyriešil, bolo by možné implementovať spôsob paralelného spracovania. Jednotlivé inzeráty nie sú na sebe závisle a tak je možné získavať informácie súbežne.

Modernou technológiou v tomto smere je dnes *serverless* výpočtová technika. Nasadením aplikácie takouto formou odpadáva povinnosť vývojára zaoberať sa dostupnými zdrojmi servera akými sú veľkosť úložiska dát, pamäte, výkon databáze, počet a výkon procesorov a podobne. Vývojár zároveň platí len za čas, resp. zdroje, ktoré program reálne využíva.

Firma Amazon prostredníctvom svojho projektu **AWS**[1] prišla s riešením s názvom **Lambda**[2], kedy je možné púšťať súbežne množstvo menších operácií, resp. funkcií na základe rôznych spúšťacích udalostí. Medzi tieto udalosti patrí napríklad HTTP požiadavka, čas, pridanie novej položky do fronty a podobne. Vývojár nahrá kód na server, a ten automaticky a spoľahlivo spracuje všetky požiadavky na vykonanie daného kódu. Kód, ktorý beží v takomto prostredí sa nazýva *lambda funkcia*. Táto funkcia sa môže vykonávať niekoľkokrát súbežne. Záleží teda od množstva požiadaviek, ktorými sa funkcia spúšťa. Ak je tých požiadaviek málo, využíva sa menej výpočtovej sily. Ak ich je viac, automaticky sa do výpočtu zapoja ďalšie servery. Takémuto prístupu sa hovorí *autoscaling*. [28]

8.2.4 Spoľahlivá extrakcia

Pri extrakcií dát dochádza k sťahovaniu dokumentov z webových zdrojov. Skripty, ktoré dáta zbierajú, sa spoliehajú na nemennú štruktúru týchto dokumentov. Takto vie byť extrakcia dát presná a spoľahlivá.

Štruktúra sa ale môže časom zmeniť. Môže dôjsť k zmene atribútov alebo hodnôt. V tom prípade môže extraktor prestať spoľahlivo fungovať, čo predstavuje problém. Túto situáciu je potrebné vyriešiť s čo najmenšími následkami.

Extrakcia dát nemusí zlyhávať ako celok. Skripty môžu prestať fungovať len pre konkrétny inzerát. Niektoré dokumenty, s ktorými extraktor pracuje, obsahujú informácie o niekoľkých inzerátoch naraz. Vstupom je teda zoznam inzerátov v nejakom formáte získaný z určitého webového zdroja. Zoznam sa nespracováva ako celok, naopak, extrakcia prebieha vždy na úrovni jedného inzerátu. Preto má abstraktná trieda `Module` dve metódy venujúce sa získaniu základných informácií o všetkých inzerátoch, pozri 7.1. Jednou dochádza k získaniu zoznamu nespracovaných objektov a druhá už spracováva tieto objekty samostatne. Všetko to riadi jadro extraktora. V prípade, že dôjde k chybe, skript neskončí, ale pokračuje ďalším inzerátom. Extraktor týmto spôsobom vynechá inzeráty, ktoré nie je schopný spracovať, a tie ostatné spracuje riadne. Informácie o zlyhaniach sa ukladajú do žurnálu (angl. *log*). Neskôr je možné tento žurnál spätne analyzovať, zistiť, kde nastala chyba a túto chybu eventuálne opraviť.

8.2.5 Klientská časť a používateľská skúsenosť

Aj pri implementácii prezentačnej vrstvy aplikácie dochádzalo ku komplikáciám. Knižnica React je založená na komponentoch, pozri 7.3. Existuje niekoľko návrhových vzorov, podľa ktorých je možné komponenty implementovať. Jedným z nich je rozdelenie komponentov na *kontajnery* a *prezentačné komponenty* [50].

Kontajner predstavuje dátovú a logickú vrstvu. Má stav a často získava dáta HTTP požiadavkami. Jeho výstupom sú prezentačné komponenty. Tie nemajú stav a pri vykresľovaní používajú argumenty, ktoré im predáva rodičovský kontajner. Zároveň definujú štýly, ktoré formujú vzhľad výsledného komponentu.

React prekreslí komponent v prípade, že sa zmenia jeho atribúty alebo stav [3]. Preto je potrebné celú aplikáciu vhodne rozdeliť do menších komponentov tak, aby sa prekresľovali len tie neaktualizované.

Prvotná verzia implementovanej aplikácie obsahovala jeden veľký kontajner s množstvom parametrov, ktoré sa postupne predávali do prezentačných komponentov. Problém

[1] https://aws.amazon.com/
[2] https://aws.amazon.com/lambda/

bol, že ak sa zmenil čo i len jeden z týchto atribútov, prekreslila sa prakticky celá stránka. Ako aplikácia rástla, zvýšovala sa záťaž pre prehliadač. Stránka sa nenačítala plynule, akákoľvek zmena viedla k spomaľovaniu animácií a celková používateľská skúsenosť, resp. spokojnosť (angl. *user experience*) bola zlá. Tento problém sa vyriešil vhodným rozdelením veľkého kontajnera na menšie časti. Každá časť má menšiu množinu vstupných parametrov. K prekresľovaniu nedochádza tak často a prekresľujú sa naozaj len tie časti webovej stránky, ktoré sa zmenili.

Kapitola 9

Záver

V rámci tejto práce som naštudoval možnosti extrakcie dát z webových zdrojov. Urobil som komplexnú analýzu portálov poskytujúcich inzeráty s prenájmami bytov. Bolo zistené, akým spôsobom si v rámci týchto služieb posielajú dáta klient so serverom. Podrobne som zdokumentoval parametre, ktoré sa pri tomto prenose využívajú. Na základe týchto zistení a na základe dostupných možností bol vytvorený návrh aplikácie, ktorá agreguje inzeráty s prenájmami bytov z niekoľkých webových zdrojov.

Stanovil som ciele, vlastnosti, resp. funkcie, ktorými má aplikácia disponovať. Ako úložisko získaných inzerátov bola zvolená dokumentová NoSQL databáza. Navrhol som architektúru aplikácie a popísal jednotlivé komponenty celej aplikácie. Zistilo sa, že bude lepšie, ak sa databáza inzerátov bude pravidelne automaticky aktualizovať bez toho, aby prišla požiadavka z klientskej časti aplikácie (frontend) od používateľa.

Navrhnutá aplikácia bola po zvolení vhodných technológií implementovaná. Výstupom sú skripty pre získavanie dát o inzerátoch z troch veľkých realitných portálov: *UlovDomov.cz*, *Bezrealitky.cz* a *Sreality.cz*. Získané dáta sú jednotnou formou uložené v *MongoDB* databáze. Pomocou REST API je možné k týmto údajov pristupovať. Taktiež sa podarilo vytvoriť plne funkčnú webovú stránku napísanú pomocou technológie *React*. Tá vhodným spôsobom zobrazuje agregované údaje o inzerátoch s prenájmami bytov z viacerých zdrojových portálov. Používateľ môže prostredníctvom webovej stránky pridávať inzeráty do zoznamu obľúbených, pridávať poznámky k inzerátom, alebo filtrovať a triediť inzeráty pomocou niekoľkých parametrov. Ak má používateľ záujem o upozornenia na nové inzeráty, môže si nastaviť notifikácie na základe zvolených filtrov.

V práci sa venujem aj praktickým problémom, ktoré nastali počas testovania aplikácie, a popisujem spôsob, akým som tieto problémy vyriešil. Zároveň sú spomenuté ďalši situácie, ktoré môžu v budúcnosti nastať a ako sa s nimi vysporiadať.

Pri finalizácii tejto práce som rozmýšľal nad možnými rozšíreniami vzniknutej aplikácie. Ak by došlo k zmene inzerátu, napríklad znížením ceny alebo upravením popisu, bolo by vhodné s touto skutočnosťou oboznámiť používateľa. Riešiť sa to dá zakódovaním objektu inzerátu určitou hašovacou funkciou a potom už len porovnávať výstupy tejto funkcie.

Pridanou hodnotou tejto aplikácie by mohli byť aj informácie o inzerátoch, resp. bytoch, získaných od používateľov. Samotní používatelia aplikácie by mohli byť sekundárnym zdrojom informácií, ktoré by skripty nedokázali automaticky získať. Získavanie by mohlo prebiehať formou otázok kladených v momente kedy používateľ potvrdil vykonanú prehliadku bytu.

Vylepšením aplikácie by mohlo byť aj použitie lepšieho API na získavanie presných informácií o vyhľadanej lokalite. Použitý nástroj *Nominatim* má svoje nedostatky, ktoré

boli v tejto práci spomenuté. Jednoduchým testovaním bolo zistené, že lepšie výsledky má vyhľadávač od spoločnosti Google. Za ich služby je ale potrebné zaplatiť.

Komunikácia medzi frontendom a serverovou časťou aplikácie by mohla byť efektívnejšia nahradením *REST API* za *GraphQL*. Prenášalo by sa menšie množstvo dát a nemuseli by sa implementovať nové koncové body na strane API v prípade pridávania novej funkcionality na strane frontendu.

V rámci tejto práce som si vyskúšal pracovať s modernými technológiami pri tvorbe stredne veľkej aplikácie. Prešiel som si celým vývojom, od získavania informácií, návrhu, až po implementáciu. Naučil som sa pracovať systematicky a pochopil som problematiku extrakcie dát z webových zdrojov.

Literatúra

[1] Allan, A.; Poole, S.; Hoffmann, S.: Nominatim Usage Policy. Október 2018, [Online; navštívené 17.5.2019].
URL https://operations.osmfoundation.org/policies/nominatim/

[2] Arpaci-Dusseau, R. H.: *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, August 2012, ISBN 9781105979125, 686 s.

[3] Bain, L.: How does React decide to re-render a component? Február 2017, [Online; navštívené 12.5.2019].
URL https://lucybain.com/blog/2017/react-js-when-to-rerender/

[4] Berners-Lee, T.; Hendler, J.; Lassila, O.: The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. *ScientificAmerican.com*, Máj 2001: str. 3.

[5] Bezrealitky.cz: O realitním serveru bezrealitky.cz. 2019, [Online; navštívené 16.1.2019].
URL https://www.bezrealitky.cz/informace/o-nas

[6] Birrell, A.; Nelson, B.: Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, ročník 2, č. 1, 1984: s. 39–59, ISSN 1557-7333.

[7] Brown, M. S.: Get The Basics On NoSQL Databases: Wide Column Store Databases. https://www.forbes.com/sites/metabrown/2018/03/31/get-the-basics-on-nosql-databases-wide-column-store-databases/, Marec 2018, [Online; navštívené 6.1.2019].

[8] Buckler, C.: SQL vs NoSQL: The Differences. September 2015, [Online; navštívené 6.1.2019].
URL https://www.sitepoint.com/sql-vs-nosql-differences/

[9] Buna, S.: *Learning GraphQL and relay*. Packt Publishing Ltd, 2016.

[10] Burget, R.: Information extraction from HTML documents based on logical document structure. 2004.

[11] Doyle, B.; Lopes, C.: Survey of Technologies for Web Application Development. Marec 2017.
URL http://www.escholarship.org/uc/item/8mh6n58m

[12] Fette, I.; Melnikov, A.: The WebSocket Protocol. RFC 6455, RFC Editor, December 2011.
URL https://www.rfc-editor.org/info/rfc6455

[13] Fielding, R.; Gettys, J.; Mogul, J.; aj.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, Jún 1999.
URL https://www.rfc-editor.org/info/rfc2616

[14] Garrett, J. J.: Ajax: A New Approach to Web Applications. Február 2005: str. 5.

[15] Google: Secure your site with HTTPS: Protect your site and your users. 2018, [Online; navštívené 22.12.2018].
URL https://support.google.com/webmasters/answer/6073543?hl=en

[16] GraphQL.org: Explore GraphQL: The API for modern apps. [Online; navštívené 24.12.2018].
URL https://www.graphql.com/

[17] Hickson, I.; Berjon, R.; Faulkner, S.; aj.: HTML5. Technická správa, W3C, Marec 2018, [Online; navštívené 27.12.2018].
URL https://www.w3.org/TR/2018/SPSD-html5-20180327/

[18] Hock-Chuan, C.: A Quick-Start Tutorial on Relational Database Design. September 2010, [Online; navštívené 6.1.2019].
URL https://www.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html

[19] Ian: What is a Column Store Database? Jún 2016, [Online; navštívené 6.1.2019].
URL https://database.guide/what-is-a-column-store-database/

[20] Kennedy, J.; Satran, M.: What is a Transaction? Máj 2018, [Online; navštívené 6.1.2019].
URL https://docs.microsoft.com/en-gb/windows/desktop/Ktm/what-is-a-transaction

[21] Kushmerick, N.: *Wrapper Induction for Information Extraction*. Dizertačná práca, Seattle, WA, USA, 1997.

[22] Lassila, O.; Swick, R. R.: Resource Description Framework(RDF) Model and Syntax Specification. Technická správa, W3C, Január 1999, [Online; navštívené 27.12.2018].
URL https://www.w3.org/TR/1999/PR-rdf-syntax-19990105/

[23] Lie, H. W.; Bos, B.: Cascading Style Sheets, level 1. Technická správa, W3C, September 2018, [Online; navštívené 27.12.2018].
URL https://www.w3.org/TR/REC-CSS1/

[24] Loshin, P.: Simple object access protocol. *Computerworld*, ročník 34, č. 36, 2000, ISSN 00104841.
URL
http://web.a.ebscohost.com.ezproxy.lib.vutbr.cz/ehost/detail/detail?vid=0&sid=93585a54-e9b1-4509-9641-8fc14cd62505%40sdc-v-sessmgr05&bdata=Jmxhbmc9Y3Mmc2l0ZT1laG9zdC1saXZl#AN=3565507&db=a9h

[25] Masse, M.: *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. Ö'Reilly Media, Inc.", 2011.

[26] Melendez, S.: The Difference Between Dynamic & Static Web Pages. August 2018, [Online; navštívené 22.12.2018].
URL https://smallbusiness.chron.com/difference-between-dynamic-static-pages-69951.html

[27] Mesbah, A.; van Deursen, A.: An Architectural Style for Ajax. IEEE, 2007, ISBN 0769527442, str. 9.

[28] Miller, R.: AWS Lambda Makes Serverless Applications A Reality. 2016, [Online; navštívené 11.5.2019].
URL https://techcrunch.com/2015/11/24/aws-lamda-makes-serverless-applications-a-reality/

[29] MongoDB, Inc.: NoSQL Databases Explained. 2018, [Online; navštívené 6.1.2019].
URL https://www.mongodb.com/nosql-explained

[30] Moogk, D. R.: Minimum viable product and the importance of experimentation in technology startups. *Technology Innovation Management Review*, ročník 2, č. 3, 2012.

[31] Neo4j, Inc.: Graph Database Use Cases and Solutions. 2019, [Online; navštívené 6.1.2019].
URL https://neo4j.com/use-cases/

[32] Pritchett, D.: BASE: An Acid Alternative. *Queue*, ročník 6, č. 3, Máj 2008: s. 48–55, ISSN 1542-7730, doi:10.1145/1394127.1394128.
URL http://doi.acm.org/10.1145/1394127.1394128

[33] Q-Success: Usage Statistics of CSS for Websites, December 2018. December 2018, [Online; navštívené 27.12.2018].
URL https://w3techs.com/technologies/details/ce-css/all/all

[34] Reese, G.: *Database Programming with JDBC & Java*. O'Reilly Media, Inc., 2000, ISBN 1565926161.

[35] Rescorla, E.: HTTP Over TLS. RFC 2818, RFC Editor, Máj 2000.
URL https://www.rfc-editor.org/info/rfc2818

[36] Rouse, M.: What is graph database? Jún 2016, [Online; navštívené 6.1.2019].
URL https://whatis.techtarget.com/definition/graph-database

[37] Rychlý, M.; Kolář, D.: NoSQL databáze. Přednáška pro PDB, Október 2013, [Online; navštívené 7.1.2019].
URL http://www.fit.vutbr.cz/~rychly/public/docs/slides-nosql-databases/slides-nosql-databases.print.pdf

[38] Sasaki, B. M.: Graph Databases for Beginners: Why Graph Technology Is the Future. Júl 2018, [Online; navštívené 6.1.2019].
URL https://neo4j.com/blog/why-graph-databases-are-the-future/

[39] Schulz, J.: Cassandra Use Cases: When To Use and When Not To Use Cassandra. Marec 2018, [Online; navštívené 6.1.2019].
URL https://blog.pythian.com/cassandra-use-cases/

[40] Seznam.cz, a.s.: O službě Sreality.cz. 2019, [Online; navštívené 16.1.2019].
URL https://napoveda.seznam.cz/cz/sreality/o-sluzbe-sreality.cz

[41] SimilarWeb LTD: Heureka.cz Analytics - Market Share Stats & Traffic Ranking. 2019, [Online; navštívené 9.1.2019].
URL https://www.similarweb.com/website/heureka.cz

[42] Strauch, C.: NoSQL Databases. 2011, [Online; navštívené 7.1.2019].
URL http://www.christof-strauch.de/nosqldbs.pdf

[43] Sturgeon, P.: Understanding RPC Vs REST For HTTP APIs. September 2016, [Online; navštívené 24.12.2018].
URL https://www.smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/

[44] Sullivan, D.; Sullivan, J.: NoSQL Key-Value Database Simplicity vs. Document Database Flexibility. September 2015, [Online; navštívené 6.1.2019].
URL http://www.informit.com/articles/article.aspx?p=2429466

[45] Techopedia Inc.: Static Web Page. 2018, [Online; navštívené 22.12.2018].
URL https://www.techopedia.com/definition/5399/static-web-page

[46] The Open Group: *Soa Source Book*. Van Haren Publishing, 2009, ISBN 9087535031.

[47] Violino, B.: How to choose the right NoSQL database. Marec 2018, [Online; navštívené 6.1.2019].
URL https://www.infoworld.com/article/3260184/nosql/how-to-choose-the-right-nosql-database.html

[48] W3C OWL Working Group: OWL 2 Web Ontology Language Document Overview (Second Edition). Technická správa, W3C OWL Working Group, December 2012, [Online; navštívené 27.12.2018].
URL https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[49] W3Techs: Usage Statistics of HTTP/2 for Websites. December 2018, [Online; navštívené 22.12.2018].
URL https://w3techs.com/technologies/details/ce-http2/all/all

[50] Whatley, W.: React component patterns. Jún 2018, [Online; navštívené 12.5.2019].
URL
https://medium.com/teamsubchannel/react-component-patterns-e7fb75be7bb0

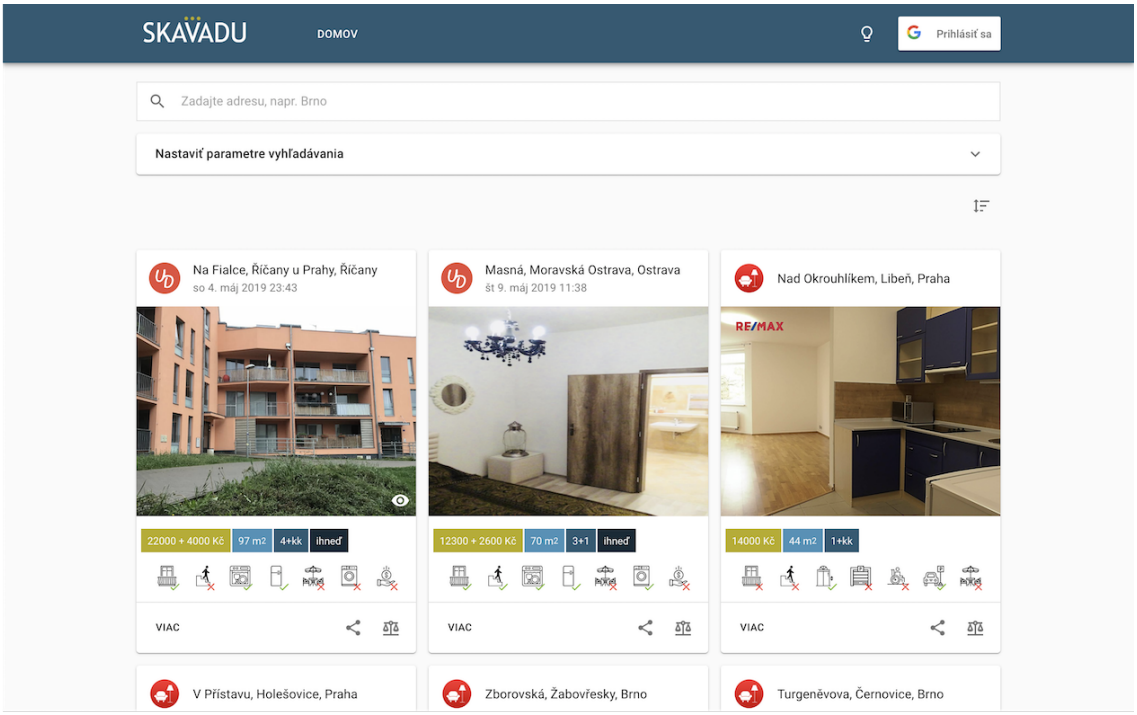
Príloha A

Obsah priloženého pamäťového média

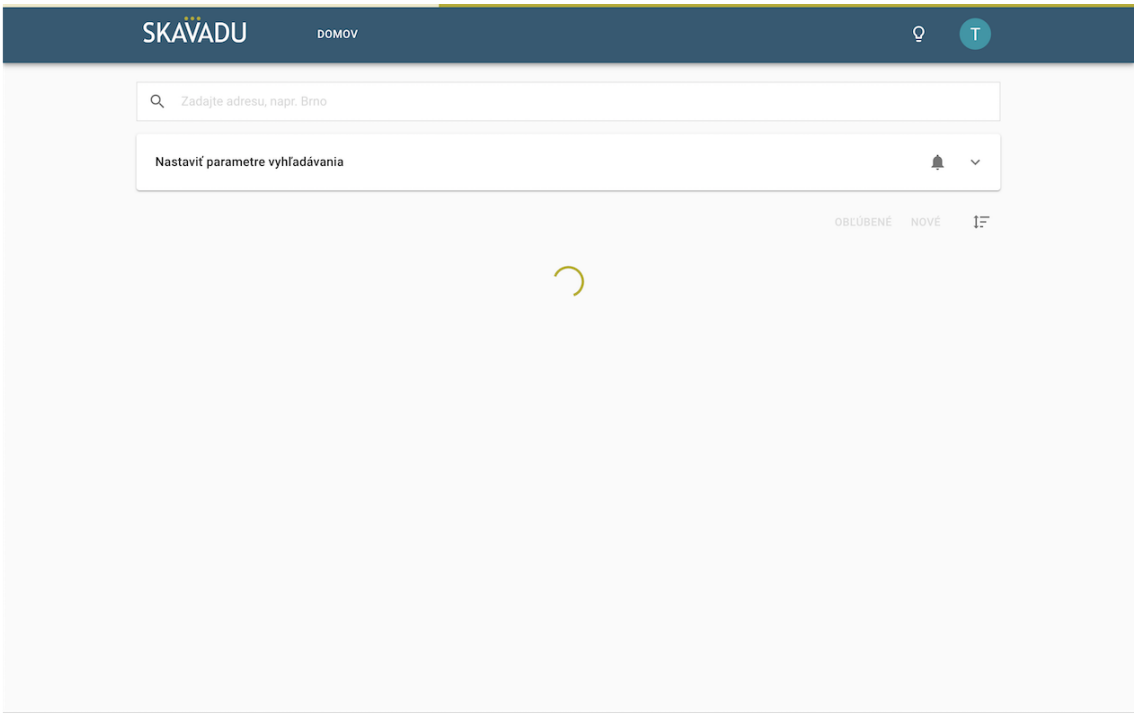
- `src/` – zdrojové súbory implementovaného systému
- `doc/` – zdrojové súbory technickej správy
- `LICENSE` – BSD licencia
- `README.md` – manuál k implementovanému systému
- `xmikit01-Portal-pro-agregaci-dat-z-webovych-zdroju.pdf` – technická správa
- `xmikit01-Portal-pro-agregaci-dat-z-webovych-zdroju-PRINT.pdf` – technická správa bez farebných odkazov

Príloha B

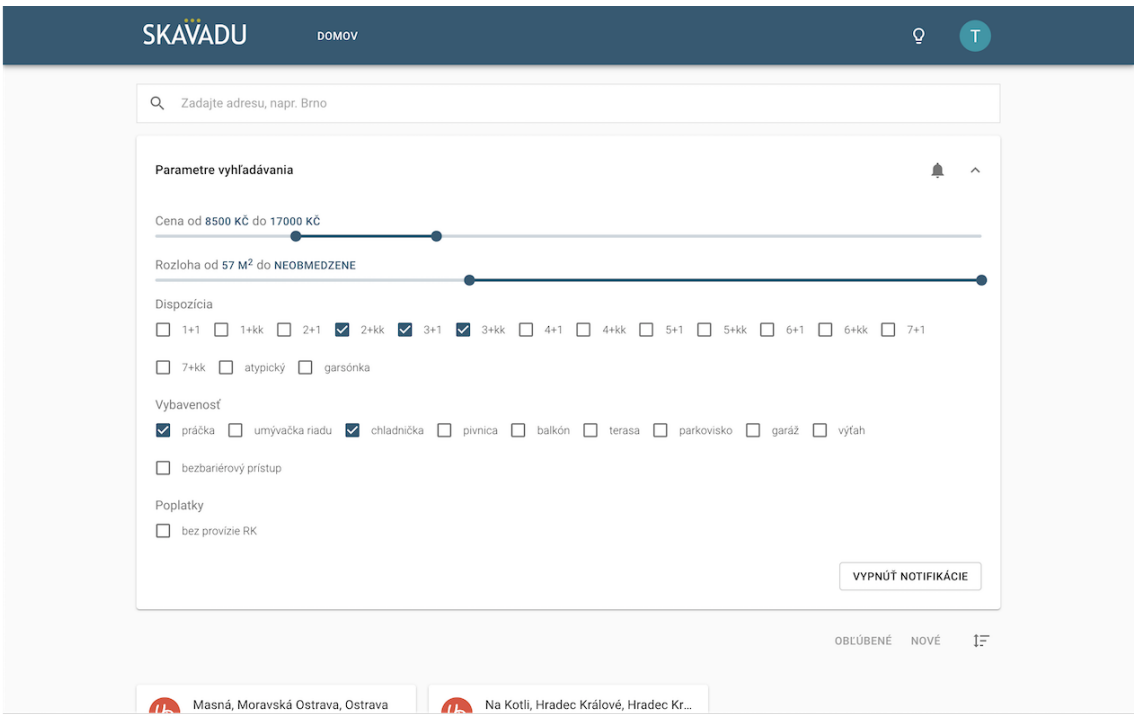
Snímky webovej aplikácie



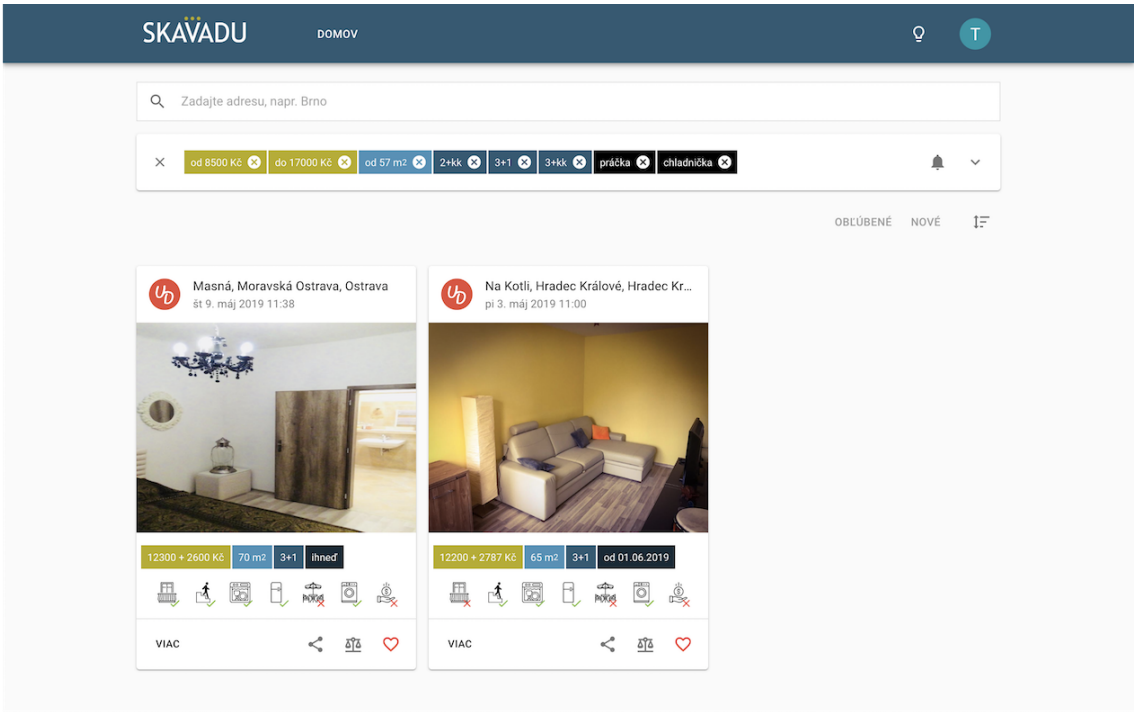
Obr. B.1: Úvodná obrazovka pre neprihláseného používateľa



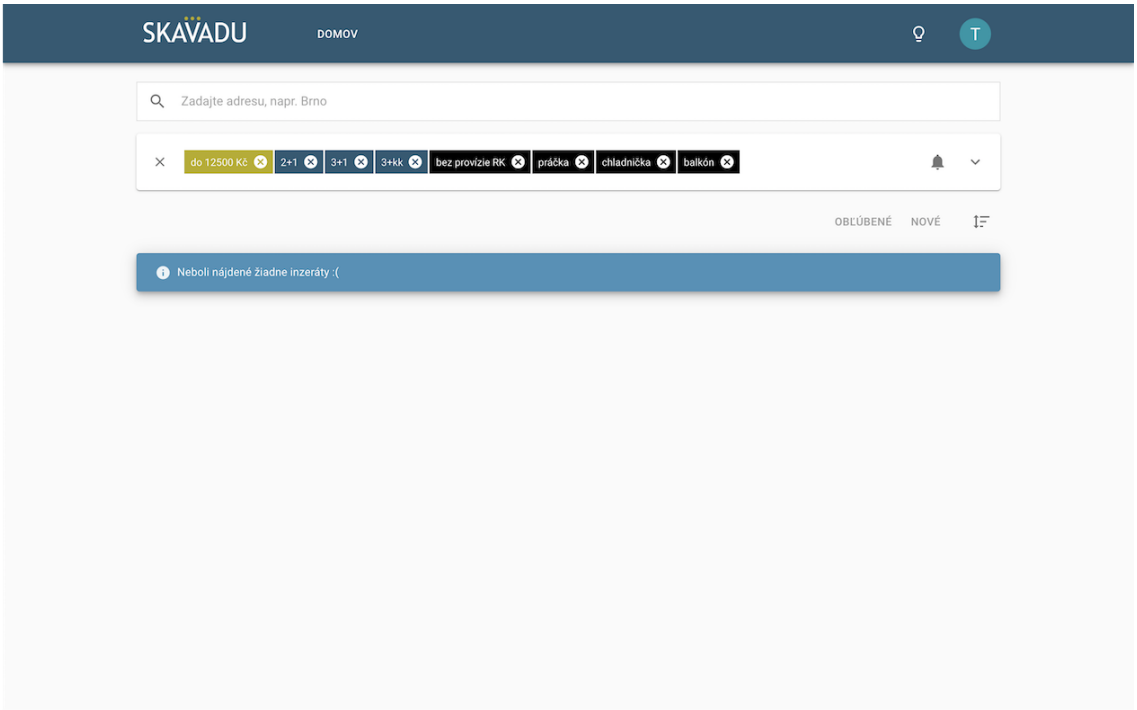
Obr. B.2: Načítanie ponuky inzerátov počas obnovenia stránky po prihlásení používateľa



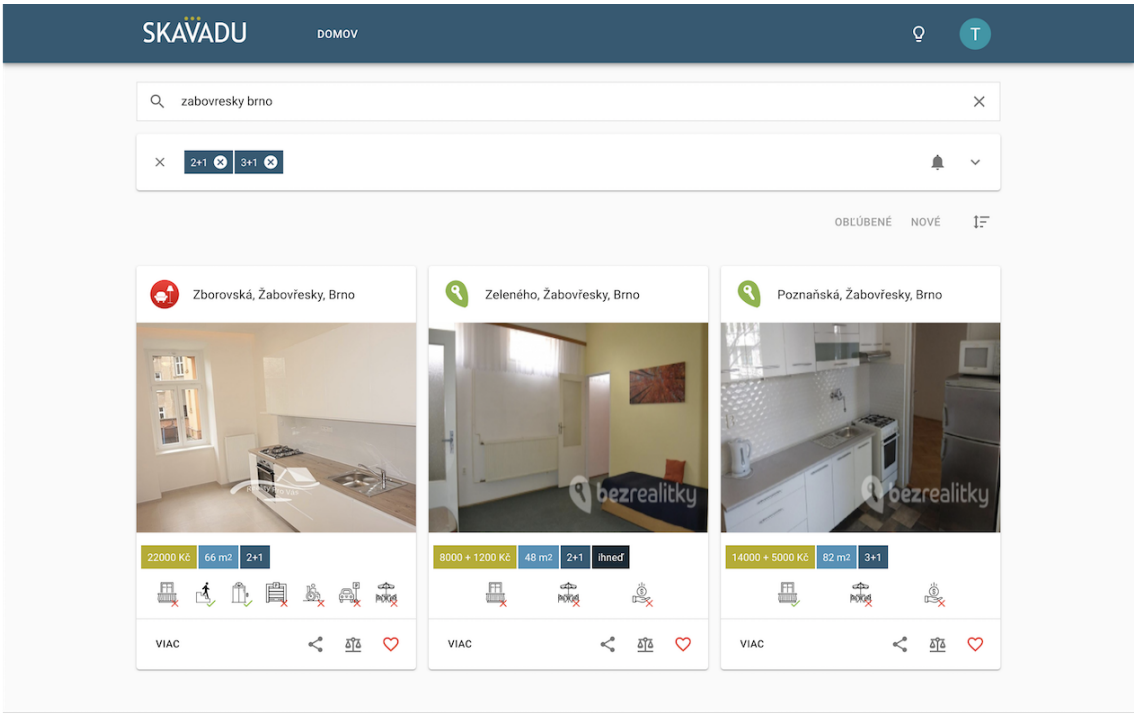
Obr. B.3: Nastavenie parametrov vyhľadávania



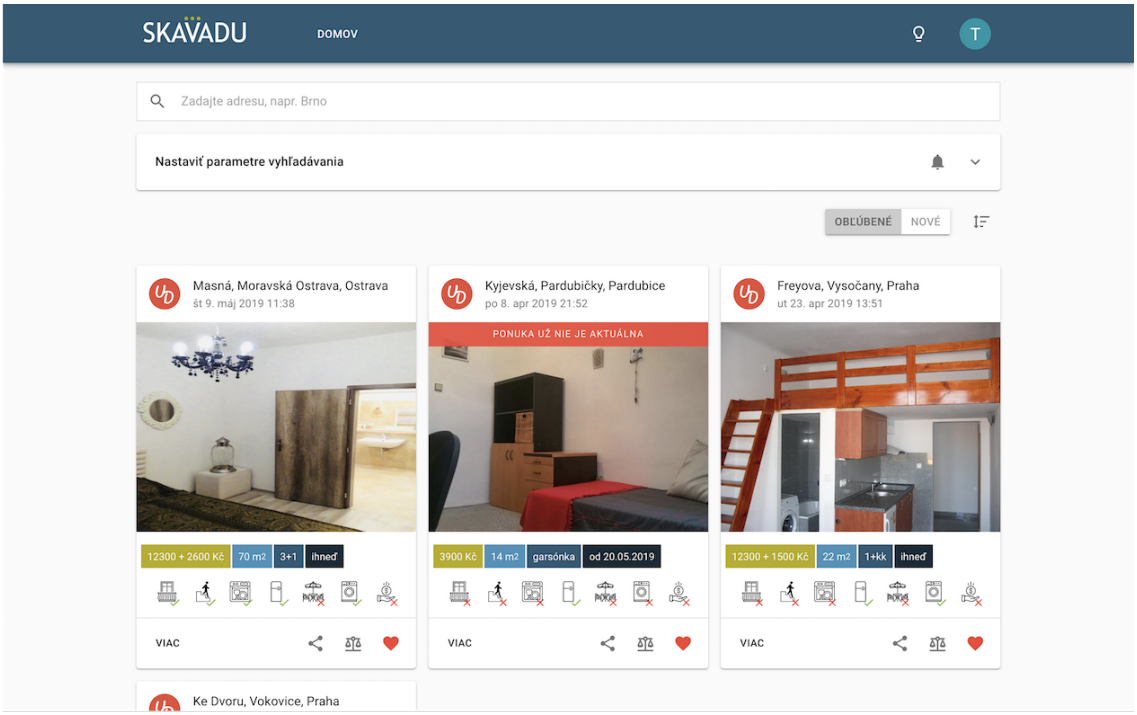
Obr. B.4: Nájdené inzeráty podľa zadaných parametrov



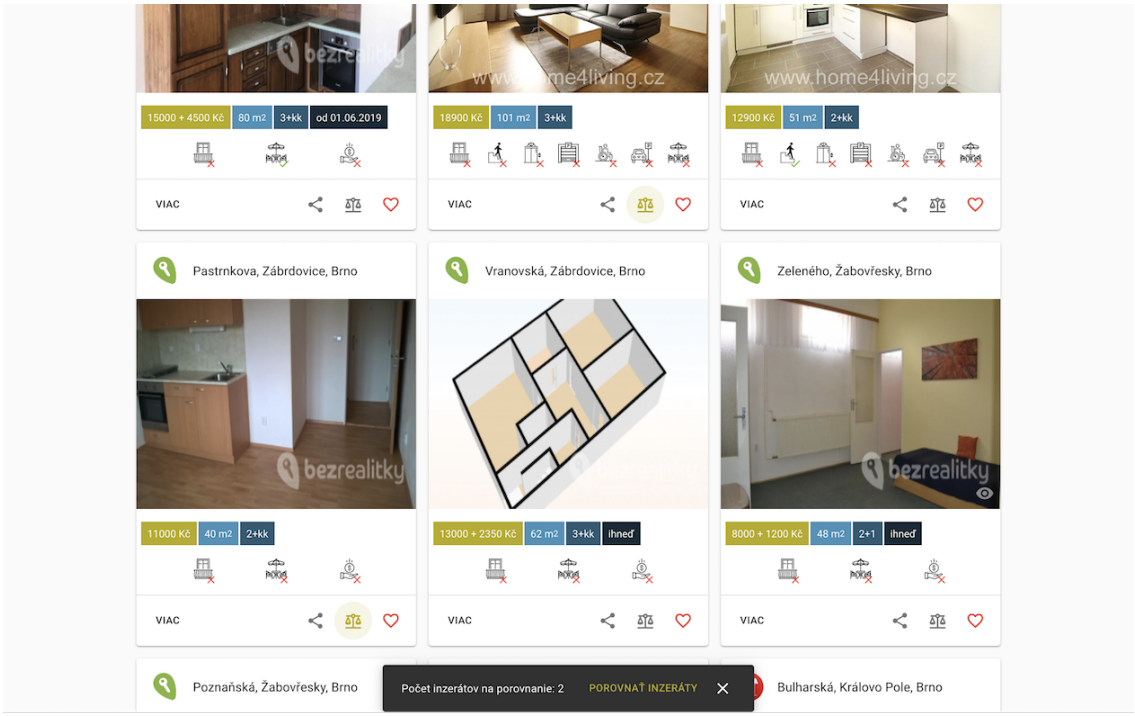
Obr. B.5: Žiadne inzeráty nevyhovujú zadaným parametrom



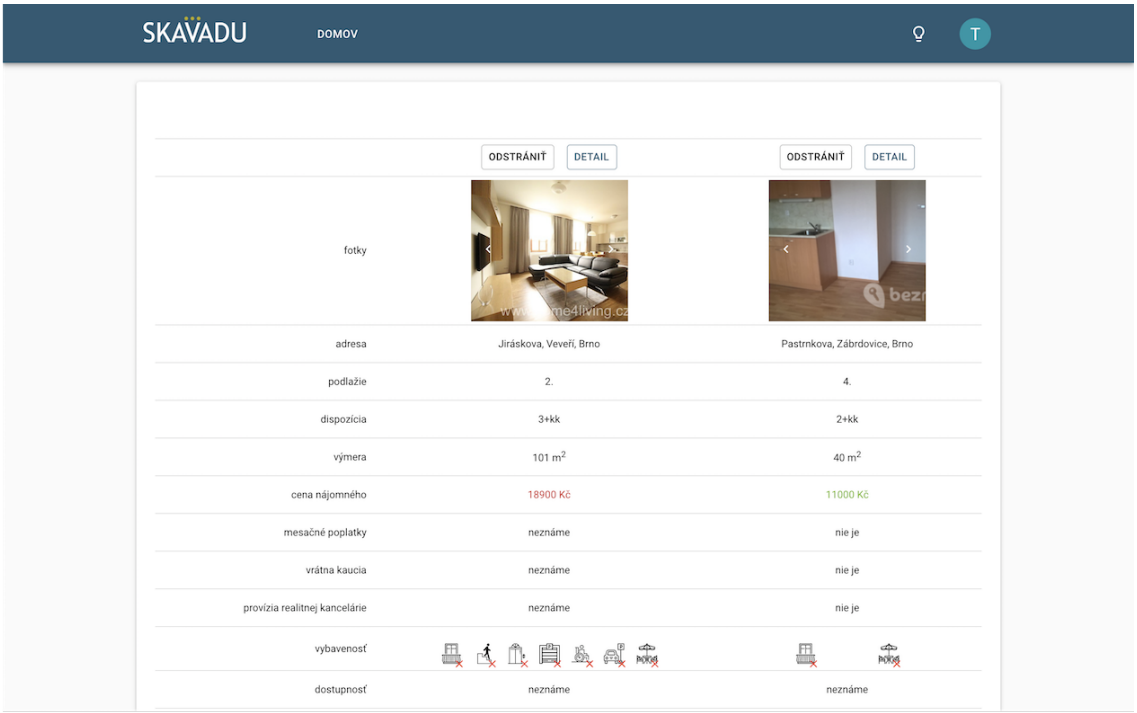
Obr. B.6: Vyhľadávanie inzerátov podľa adresy



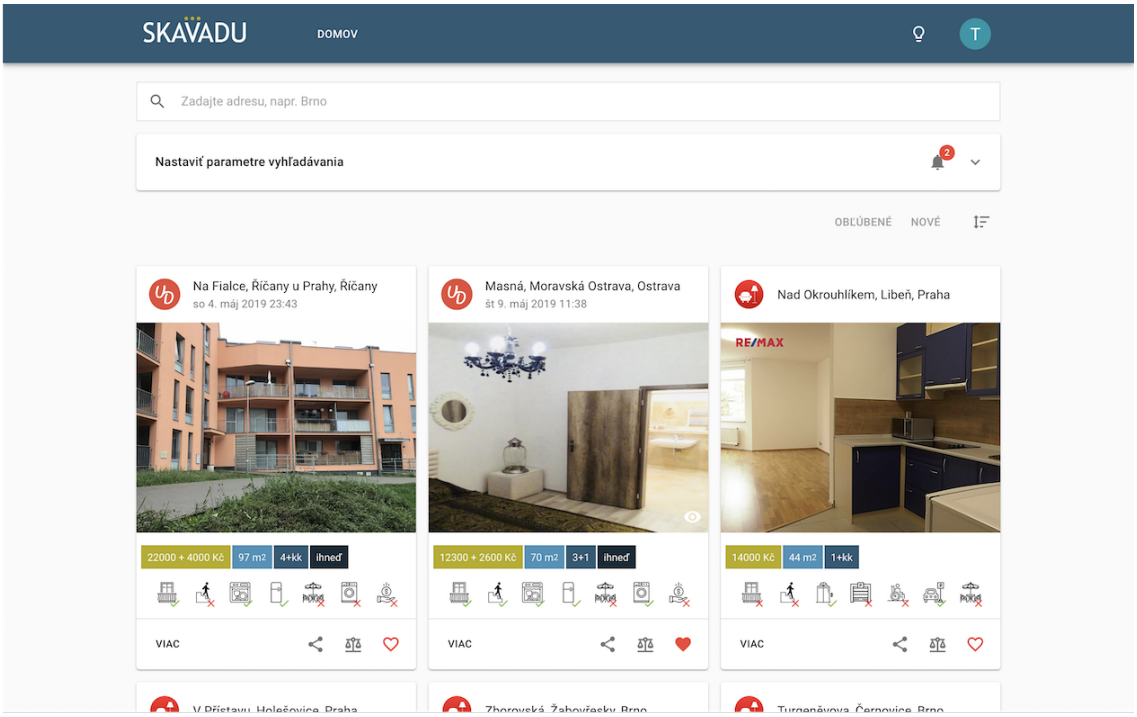
Obr. B.7: Zobrazenie obľúbených inzerátov a ukážka neaktuálnej ponuky



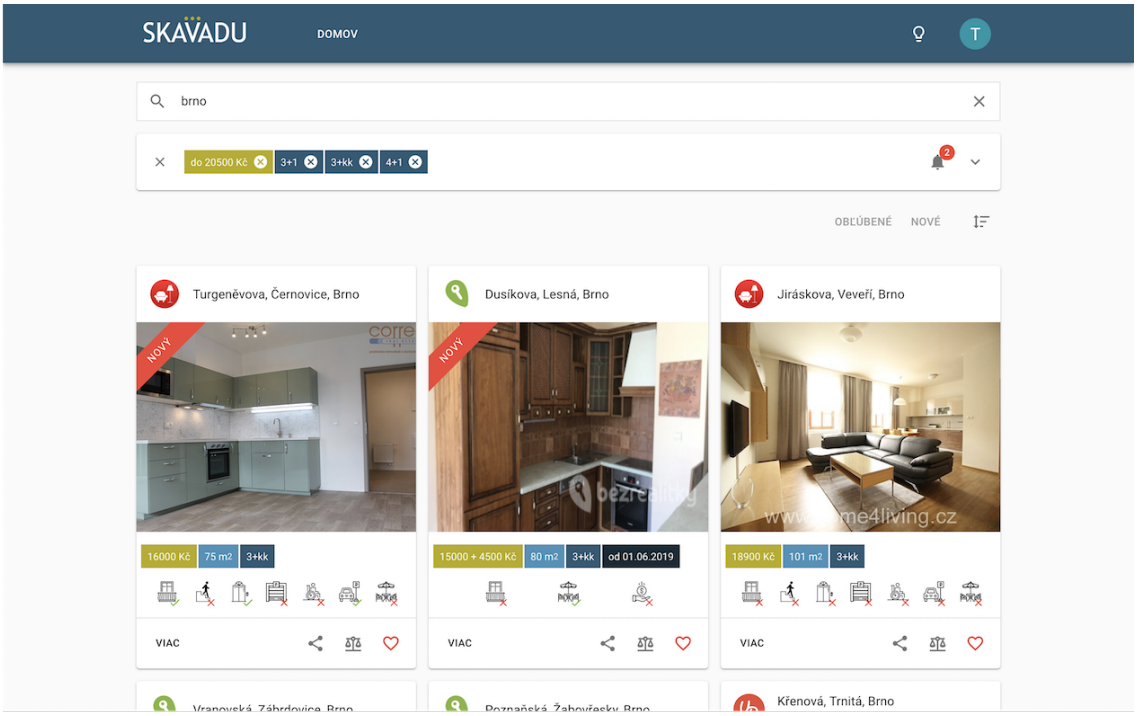
Obr. B.8: Výber inzerátov na porovnanie



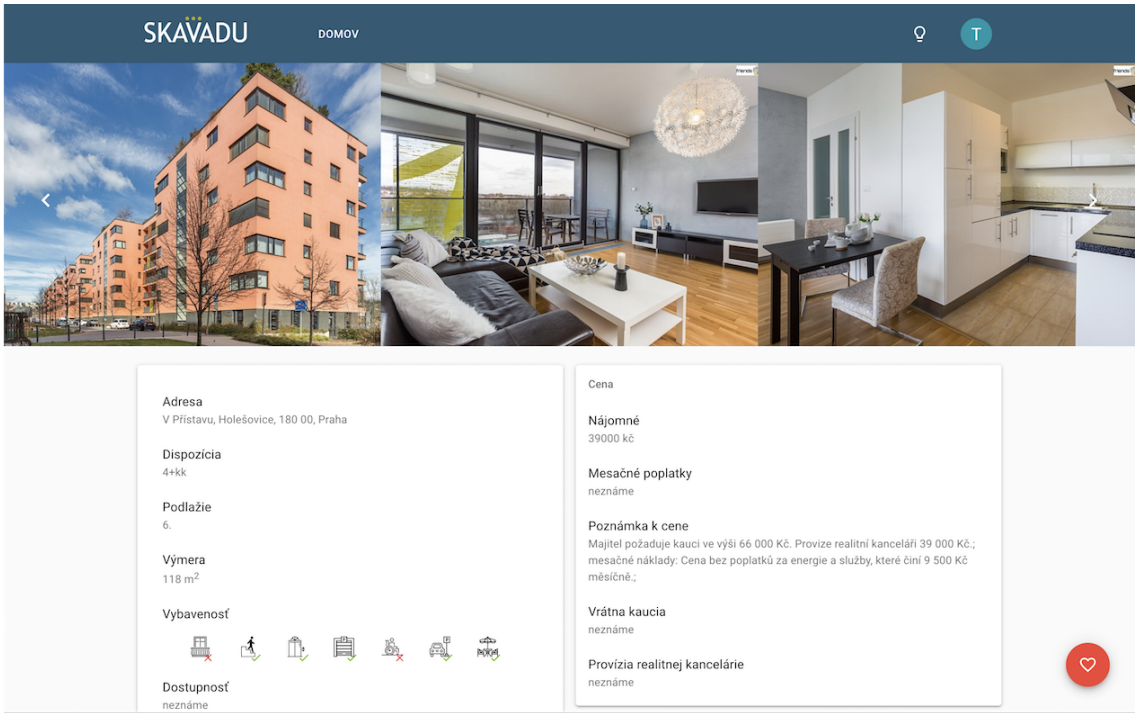
Obr. B.9: Porovnávanie vybraných inzerátov



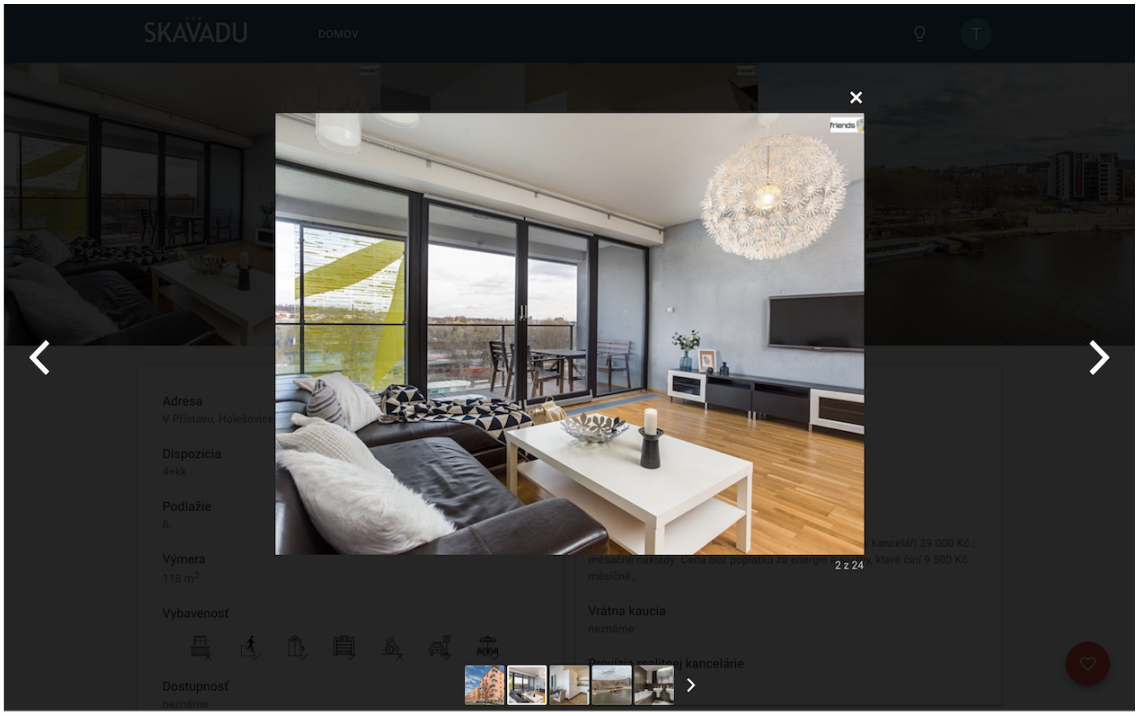
Obr. B.10: Upozornenie na nové inzeráty



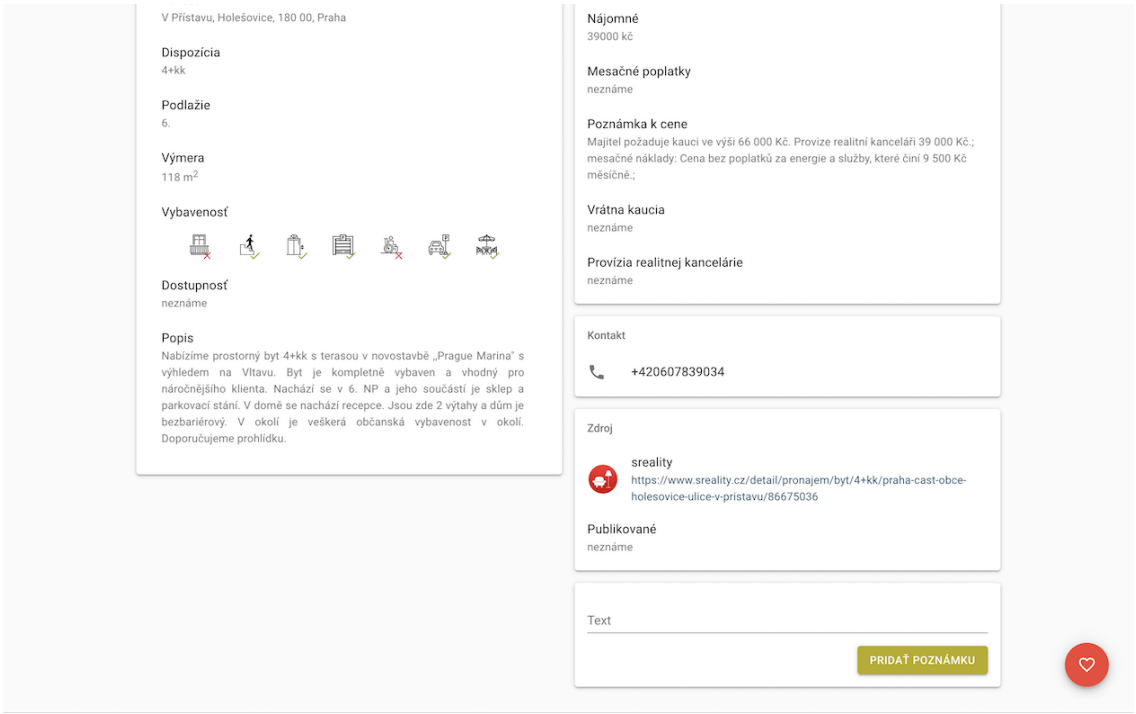
Obr. B.11: Zobrazenie nových inzerátov



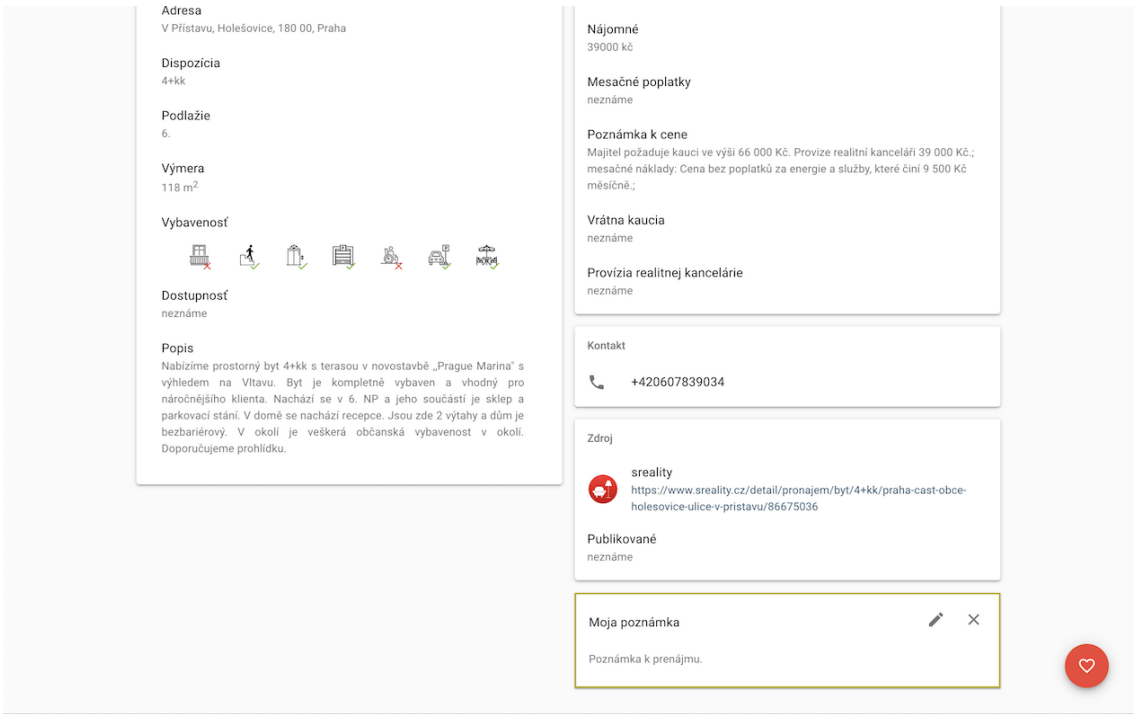
Obr. B.12: Detail inzerátu



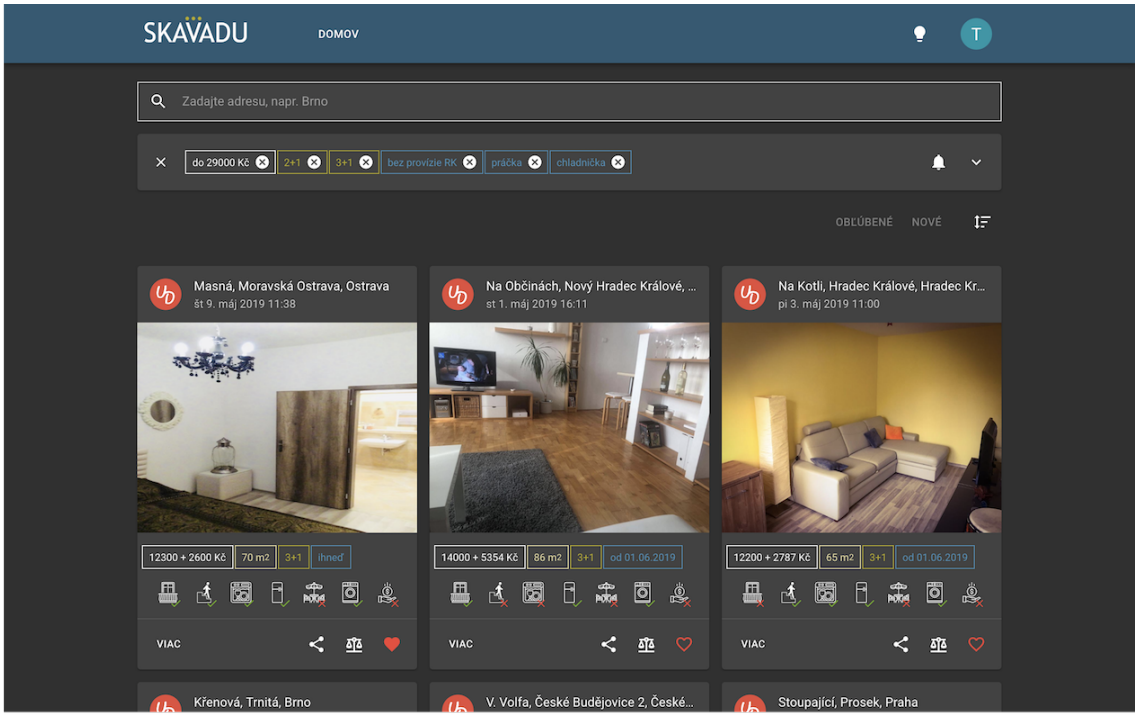
Obr. B.13: Galéria v detaile inzerátu



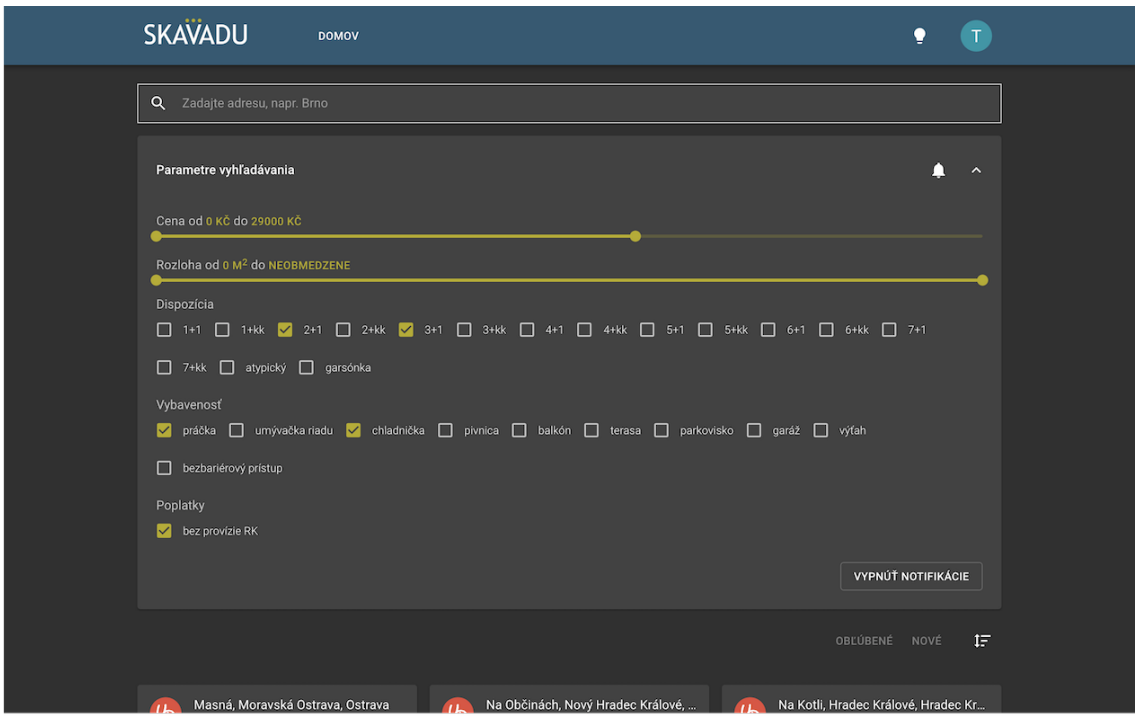
Obr. B.14: Detail inzerátu s možnosťou pridania poznámky



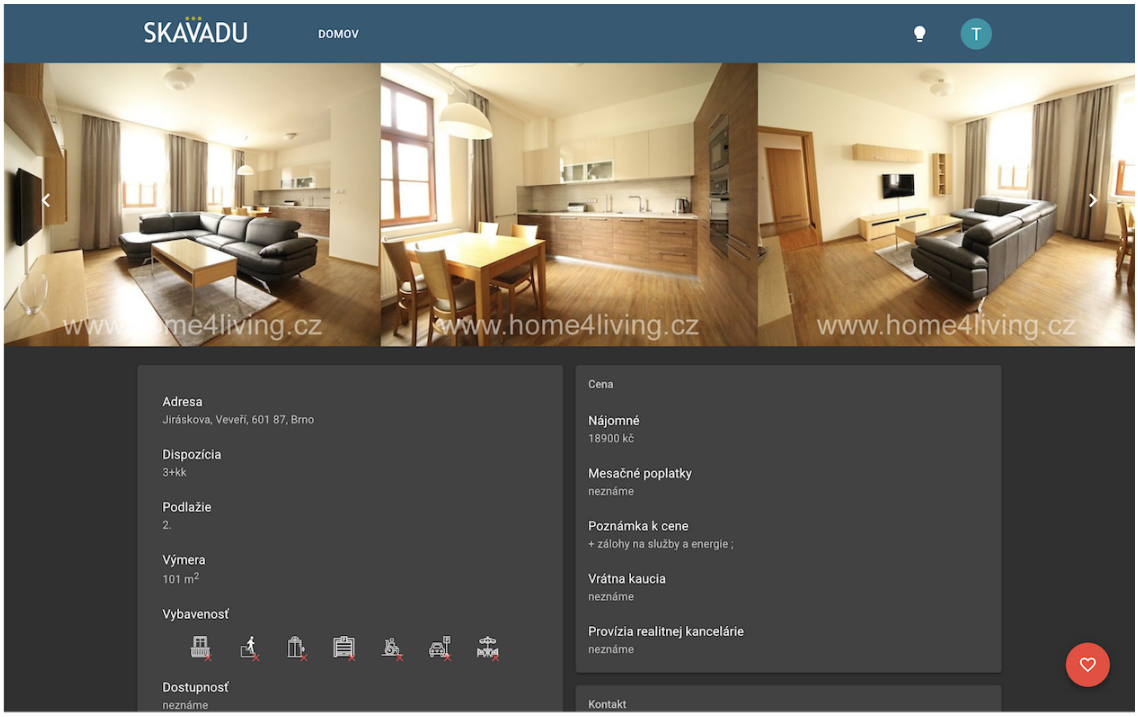
Obr. B.15: Pridaná poznámka k inzerátu



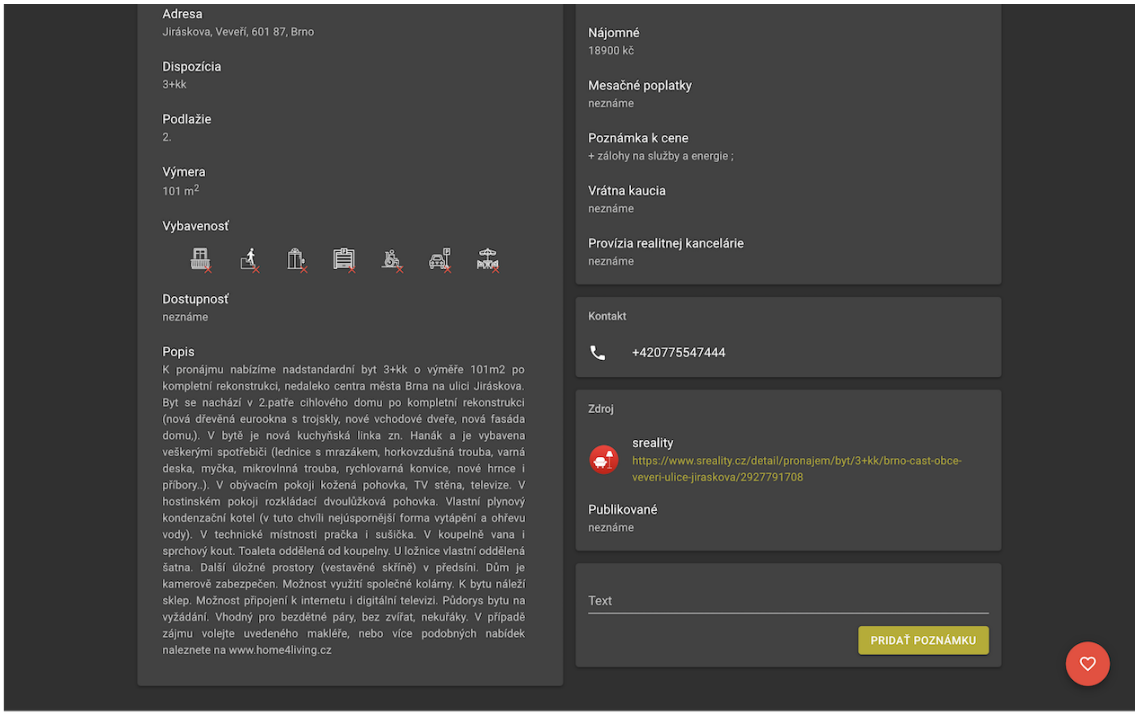
Obr. B.16: Nastavené parametre vyhľadávania v tmavom režime



Obr. B.17: Nastavenie parametrov vyhľadávania v tmavom režime



Obr. B.18: Detail inzerátu v tmavom režime



Obr. B.19: Pokračovanie detailu inzerátu v tmavom režime