UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

Janoš Žibrat

SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH

Diplomsko delo

Maribor, december 2017

SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH

Diplomsko delo

Študent:	Janoš Žibrat
Študijski program:	Univerzitetni študijski program Računalništvo in informacijske tehnologije
Mentor:	doc. dr. Tomaž Kosar
Lektor:	Urška Škvorc, prof. slov.

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija

FERI

Številka: E1031029
Datum in kraj: 25. 09. 2017, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Statut UM – UPB 12, Ur. l. RS, št. 29/2017) izdajam:

SKLEP O ZAKLJUČNEM DELU

1. **Janošu Žibratu,** študentu študijskega programa prve stopnje UN RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE, se dovoljuje izdelati zaključno delo.

2. Tema zaključnega dela je pretežno s področja Inštituta za računalništvo.

3. **MENTOR:** **doc. dr. Tomaž Kosar**

4. **Naslov zaključnega dela:**
SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH

5. **Naslov zaključnega dela v angleškem jeziku:**
WEB SERVICE FOR WEBSITE DATA PARSING AND PRESENTATION ON MOBILE DEVICE

6. Zaključno delo je potrebno izdelati skladno z "Navodili za izdelavo zaključnega dela". Skladno s 7. členom *Pravilnika o postopku priprave in zagovora zaključnega dela na študijskih programih prve in druge stopnje Univerze v Mariboru,* je bilo odobreno podaljšanje roka za oddajo zaključnega dela do 31. 12. 2017. Zaključno delo študent odda v treh izvodih (dva trdo vezana izvoda in en v spiralo vezan izvod) ter en izvod elektronske verzije v referatu za študentske zadeve. Hkrati se odda tudi izjava mentor-ja/-ice (in morebitnega somentor-ja/-ice) o ustreznosti zaključnega dela.

Pravni pouk: Zoper ta sklep je možna pritožba na Senat članice v roku 10 delovnih dni od dneva prejema sklepa.

UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE
6

Dekan:
red. prof. dr. Borut Žalik

Obvestiti:
- kandidata,
- mentor-ja/-ico,
- odložiti v arhiv.

ZAHVALA

Najprej bi se rad zahvalil svojemu mentorju docentu dr. Tomažu Kosarju za vso strokovno pomoč, podporo ter potrpljenje pri nastajanju diplomskega dela.

Zahvaljujem se tudi vsem profesorjem in asistentom, ki so pripomogli k mojemu napredku v računalništvu.

Posebno se zahvaljujem tudi svojim staršem in bratu Timu za neizmerno podporo, potrpljenje, ljubezen, vero ter veselje, ki so mi ga nudili skozi vsa leta tako pri šolanju kot športu.

Na koncu bi se rad zahvalil vsem svojim prijateljem, predvsem Maticu Naraločniku, Sebastijanu Jurendiću, Piji Vrezner, Ireni Ciglič ter Aljažu Kovačiču, ki so mi vsa leta tesno stali ob strani ter vame verjeli.

Spletna storitev za razčlenjanje spletne strani in prikaz na mobilnih napravah

Ključne besede: razpoznavanje podatkov, mobilna aplikacija, android

UDK: 621.397.7-026-26:004.777(043.2)

Povzetek

Interakcija s spletnimi aplikacijami na mobilnih napravah je dandanes zelo pogosta, saj se njihova uporaba povečuje, kljub temu pa veliko spletnih aplikacij še vedno ni prilagojenih za mobilne naprave. Implementirana je bila rešitev, ki predstavlja način, kako s pomočjo razpoznavanja podatkov rešimo težave spletnih aplikacij na mobilnih napravah, ne da bi spreminjali njihovo trenutno implementacijo. Rešitev s pomočjo spletne storitve razpozna podatke iz izbrane spletne aplikacije ter jih nato prikaže v mobilni aplikaciji. S tem zmanjšamo prenos podatkov in čas uporabe ter izboljšamo mobilno izkušnjo spletne aplikacije.

Web service for website data parsing and presentation on mobile devices

Key words: data parsing, mobile application, android

UDK: 621.397.7-026-26:004.777(043.2)

Abstract

The interaction with web applications on mobile devices is very common nowadays, but even as the use of the latter increases, many web applications are still not mobile friendly. A solution, which represents a way to solve the problems that web applications have on mobile devices with the help of data parsing, was utilized without changing their current implementation. The solution, using a web service, parses data from the chosen web application, and then displays them in the mobile application. As a consequence, data transfer and usage times are reduced, as well as improves the mobile experience of the web application.

KAZALO VSEBINE

KAZALO PROGRAMSKE KODE

KAZALO TABEL

KAZALO SLIK

SEZNAM UPORABLJENIH KRATIC

URL - Uniform Resource Locator

API - Application Programming Interface

HTML - Hypertext Markup Language

JSON - JavaScript Object Notation

DOM - Document Object Model

SQL - Structured Query Language

HTTP - Hypertext Transfer Protocol

LTE - Long-Term Evolution

PHP - PHP: Hypertext Preprocessor

1. UVOD

V današnjem času se povečuje uporaba mobilnih naprav, kar prinaša težave pri prikazu spletnih aplikacij na le-teh. Ogromno spletnih aplikacij (predvsem starejših) ni prilagojenih za mobilne naprave, saj:

1. imajo prikaz, prilagojen samo za večje zaslone namiznih računalnikov, medtem ko imajo mobilne naprave manjše zaslone, kar omejuje prikaz informacij;
2. količina podatkov (npr. slike, stili, skripte itd.), ki se prenaša, je po navadi prevelika, saj mobilne naprave uporabljajo mobilna omrežja, ki so plačljiva in imajo omejeno količino prenosa podatkov (npr. 3G, LTE (*Long-Term Evolution*)).

Obstajajo različni načini za rešitev naštetih težav. Ti so:

1. razviti mobilno verzijo spletne aplikacije, ki je dostopna na drugem naslovu URL (*Uniform Resource Locator*) in ima prilagojen videz za mobilno napravo;
2. razviti spletno aplikacijo s tako imenovanim prilagodljivim videzom (angl. *Responsive Design*), ki s pomočjo določenih pravil prilagodi prikaz velikosti zaslona;
3. razviti mobilno aplikacijo ter vmesnik API (*Application Programming Interface*), po katerem mobilna aplikacija pridobiva podatke. Vmesnik neposredno dostopa do podatkov iz spletne aplikacije (npr. podatkovne baze, datoteke itd.).

Vendar včasih teh metod ni enostavno implementirati v že obstoječo aplikacijo, ker ta tega na nek način ne dopušča. Obstaja možnost prenove celotne spletne aplikacije, vendar tudi ta rešitev ni tako enostavna, saj zahteva ogromno časa ter stroškov.

Naslednja težava spletnih aplikacij so spletne forme (angl. *Web Forms*), ki se pogosto uporabljajo ter nam omogočajo iskanje oz. filtriranje vsebine, izdelkov, oglasov itd.

Pojavi se težava, da bomo vsakič, ko obiščemo spletno stran s to formo, vnašali enake podatke, kar je zelo moteče in zamudno. Naslednja težava pri izpolnjevanju forme je ta, če v brskalniku pritisnemo gumb nazaj, da bi popravili ali odstranili kakšen parameter, ugotovimo, da moramo na novo izpolniti celotno formo. Na mobilni napravi se lahko spletne forme rešijo v obliki filtrov, ki vsebujejo parametre iz spletne forme. Te izpolnimo ter shranimo v podatkovno bazo. S tem prihranimo čas izpolnjevanja, kot tudi spreminjanje kakšnega parametra, saj se dajo urejati.

Namen diplomske naloge je, da se na izbrani spletni aplikaciji, ki nima videza in uporabe, prilagojene za mobilne naprave, prikaže dodaten način, kako lahko z metodo razpoznavanja podatkov (angl. *Data Parsing*) iz dokumentov HTML (*Hypertext Markup Language*), izluščimo poljubne podatke ter jih prikažemo v mobilni aplikaciji. Način prikaže tudi enostavno rešitev težave spletnih form v obliki filtrov. Način je zelo uporaben, kajti obstoječa implementacija spletne aplikacije ostane nedotaknjena.

Poglavje, ki sledi, opiše uporabljene tehnologije, ki smo jih potrebovali za uspešno implementacijo rešitve. Tretje poglavje vsebuje predstavitev izbrane spletne aplikacije. Navedeni so kriteriji, po katerih smo izbirali aplikacijo, ter razlogi za našo odločitev. Četrto poglavje vsebuje analizo implementacije spletne storitve, ki podatke iz izbrane spletne aplikacije razpozna, ter mobilne aplikacije, ki razpoznane podatke prikazuje. V zadnjem poglavju so opisani rezultati dela, analiza uspešnosti ter uporabnost, ki smo jih pridobili z implementacijo rešitve.

2. UPORABLJENE TEHNOLOGIJE

Pred začetkom implementacije smo morali določiti in pripraviti delovno okolje. Rešitev je možno implementirati v več programskih jezikih in okoljih. Odločili smo se, da spletno storitev implementiramo v programskem jeziku JavaScript. Izvajala se bo na spletnem strežniku v okolju Node.js.[1]

Programski jezik JavaScript smo izbrali predvsem zaradi:

- hitrosti izvajanja,
- privzeto deluje z objekti JSON (*JavaScript Object Notation*),
- ima močno podporo za regularne izraze.

Mobilna aplikacija je implementirana za platformo Android, katere minimalna verzija, potrebna za izvajanje aplikacije, znaša 5.0 (Lollipop). Mobilna aplikacija ni in ne bo objavljena v android trgovini (služi samo v raziskovalne namene).

2.1 Strežnik Node.js in uporabljene knjižnice

Node.js je odprtokodno okolje, namenjeno za izvajanje kode JavaScript na strežniku. Njegovi začetki segajo v leto 2009, kjer ga je prvič predstavil njegov avtor Ryan Dahl. Avtor je kritiziral najbolj popularni spletni strežnik *»Apache HTTP Server«* glede omejitve sočasnih povezav (do 10.000 in več) ter zaporedno izvajanja kode, ki blokira glavni proces [1].

[1] Verzija 8.9.3 LTS. https://nodejs.org/en/

Okolje deluje na principu dogodkov, ki so glavni del pri izvajanju kode JavaScript. Dogodki se izvajajo paralelno, kar omogoča več kot milijon sočasnih povezav. Takšno izvajanje kode ne blokira glavnega procesa, kar je glavna razlika med npr. popularnim skriptnim jezikom PHP (*PHP: Hypertext Preprocessor*) [1].[2] Node.js se primarno uporablja za implementacijo spletnih programov (npr. spletnih strežnikov).

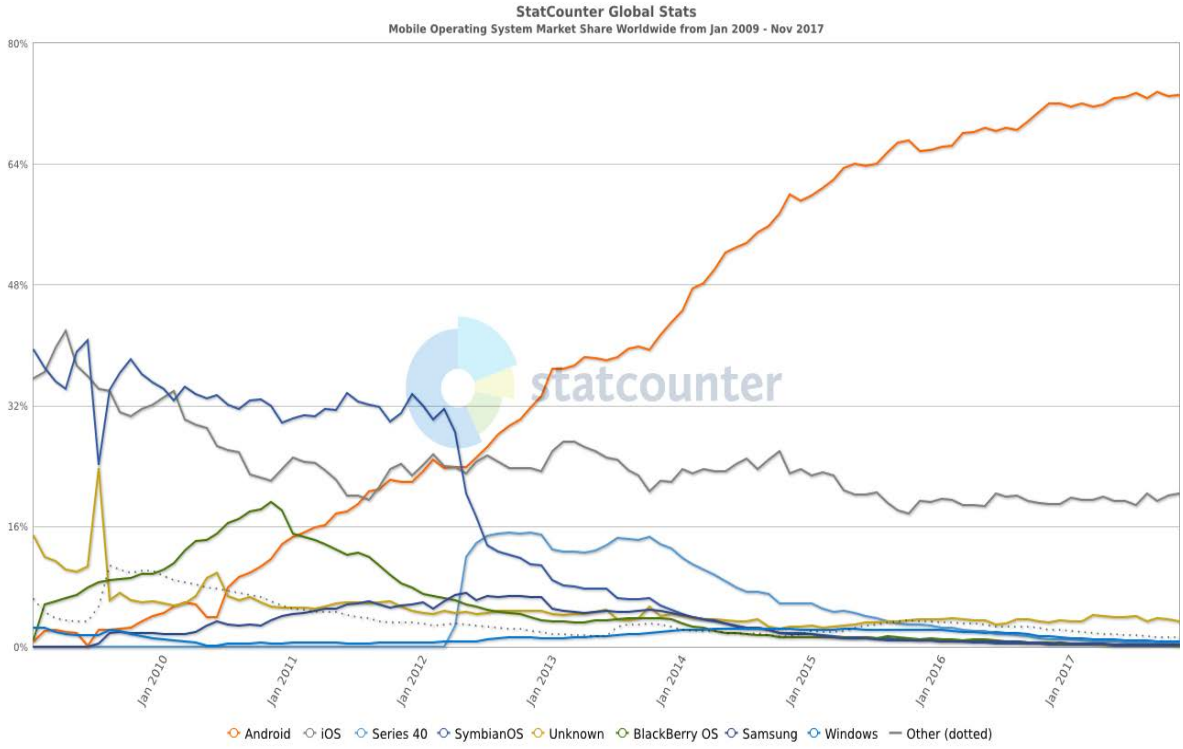
Knjižnice Node.js, ki so nam pomagale pri implementaciji spletne storitve, so:

1. Hapi (https://hapijs.com/), ogrodje za grajenje aplikacij ter storitev. Zadolženo je za inicializacijo spletne storitve ter sprejemanje in procesiranje klicev HTTP;
2. Cheerio (https://cheerio.js.org/), implementacija jedra knjižnice jQuery, namenjena za strežnik. Izvajala bo iskanje značk (angl. *Tags*) po drevesu DOM (*Document Object Model*);
3. Underscore (http://underscorejs.org/), zbirka uporabnih funkcij za delo z objekti ter zbirkami JavaScript;
4. Iconv-lite (https://github.com/ashtuchkin/iconv-lite), zbirka funkcij za kodiranje znakov.

2.2 Mobilna platforma Android

Platforma Android je operacijski sistem, ki ga razvija podjetje Google LLC [2]. Namenjen je predvsem za naprave na dotik, kot so pametni telefoni ter tablice. Prva uradna verzija je izšla leta 2008. Od leta 2012 naprej je najbolj popularna platforma za pametne telefone, kar prikazuje graf na sliki 2.1 [3].

[2] Večina funkcij PHP blokira glavni proces.



Slika 2.1: Tržni delež mobilnih operacijskih sistemov

3. IZBRANA SPLETNA APLIKACIJA

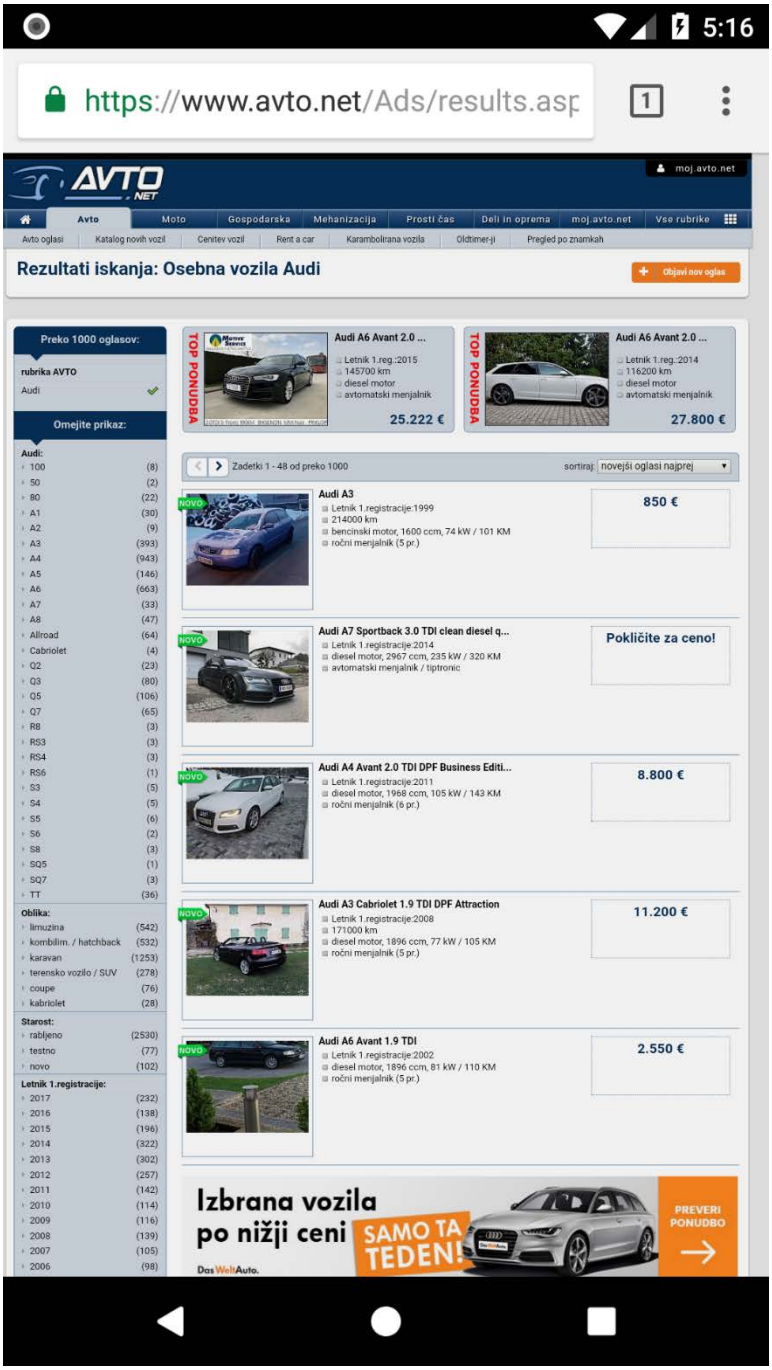
Za uspešno implementacijo rešitve smo potrebovali spletno aplikacijo, ki zadošča naslednjim kriterijem:

- v fazi izbiranja ne sme imeti implementirane mobilne izkušnje, kar pomeni, da ima prikaz vsebine prilagojen samo večjim zaslonom;
- med svojo uporabo mora prenašati veliko količino podatkov;
- ponujati mora zanimivo vsebino, po možnosti izdelke ali oglase, ki se posodabljajo dnevno;
- ponujati mora obširen iskalnik, s katerim lahko filtriramo njeno vsebino;
- v Sloveniji mora biti vsaj malo komercialno poznana.[3]

Po temeljitem premisleku ter analizi trga smo izbrali spletno aplikacijo Avto.net (https://www.avto.net/), ki izpolnjuje vse naštete kriterije:

- nima implementirane primerne mobilne izkušnje, saj je prikaz (slika 3.1) neprimeren za male zaslone, kar otežuje prebiranje informacij ter interakcijo;
- prenaša veliko količino podatkov (~ 2 MB na naloženo stran);
- ponuja veliko število oglasov, ki se objavljajo redno (celo večkrat na minuto) [4];
- ponuja zelo obširen iskalnik, ki vsebuje veliko (65) različnih parametrov, s katerimi lahko natančno filtriramo oglase;
- je zelo komercialno poznana, saj je ena najbolj obiskanih slovenskih spletnih aplikacij;[4]

[3] Zaradi večje uporabniške atraktivnosti.
[4] Več kot 100.000 različnih obiskovalcev dnevno [4].



Slika 3.1: Prikaz aplikacije Avto.net na mobilni napravi

4. IMPLEMENTACIJA

4.1 Spletna storitev

Spletna storitev je jedro celotnega projekta. Zadolžena je za razpoznavo podatkov iz izbrane spletne aplikacije in pošiljanje v mobilno aplikacijo.

Prva stopnja je bila analiza spletne aplikacije Avto.net. Ugotoviti smo morali, kako pridobiva in prikazuje podatke posameznih oglasov.

Ugotovili smo, da:

- vsako povpraševanje s pomočjo iskalnika izvede povpraševanje na skripto *results.asp*. Ta nam vrne dokument HTML z želenimi oglasi. Naslov URL (odrezek kode 4.1) vsakega povpraševanja ima vedno 65 parametrov, s katerimi lahko natančno določimo, kakšne oglase želimo imeti prikazane ter kako naj bodo ti v dokumentu HTML urejeni (npr. po datumu objave, proizvajalcu, ceni, letniku itd.);
- vsako povpraševanje vrne maksimalno 48 oglasov (na posamično stran);
- ob ustvarjanju naslova URL po iskalniku je možno izbrati proizvajalce ter modele avtomobilov, ki imajo poseben ključ, s katerim se identificirajo.

```
https://www.avto.net/Ads/results.asp?znamka=&model=&modelID=&tip=katerikoli%20tip&znamka2=&model2=&tip2=katerikoli%20tip&znamka3=&model3=&tip3=katerikoli%20tip&cenaMin=0&cenaMax=999999&letnikMin=0&letnikMax=2090&bencin=0&starost2=999&oblika=0&ccmMin=0&ccmMax=99999&mocMin=&mocMax=&kmMin=0&kmMax=9999999&kwMin=0&kwMax=999&motortakt=&motorvalji=&lokacija=0&sirina=&dolzina=&dolzinaMIN=&dolzinaMAX=&nosilnostMIN=&nosilnostMAX=&lezisc=&presek=&premer=&col=&vijakov=&vozilo=&airbag=&barva=&barvaint=&EQ1=1000000000&EQ2=1000000000&EQ3=1000000000&EQ4=1000000000&EQ5=1000000000&EQ6=1000000000&EQ7=1110100120&EQ8=1010000001&EQ9=1000000000&KAT=1010000000&PIA=&PSLO=&akcija=&paketgarancije=&broker=&prikazkategorije=&kategorija=&zaloga=&arhiv=&presort=3&tipsort=DESC&stran
```

Odrezek kode 4.1: Primer naslova URL aplikacije Avto.net z vsemi parametri

4.1.1 Proizvajalci

Pomembno vlogo pri spletni storitvi ima ustvarjanje seznama proizvajalcev ter njihovih modelov. Seznam uporablja mobilna aplikacija pri ustvarjanju filtrov, kot tudi samo razpoznavanje oglasov. Seznam spletna storitev hrani v začasnem spominu (angl. *Cache*) (odrezek kode 4.2) [5]. Posodobi se na intervalu treh dni[5]. S tem smo pospešili odzivni čas vračanja seznama iz ~ 1300 ms na ~ 70 ms.

```
HapiServer.method('getManufacturers', Manufacturers.getManufacturers, {
  cache: {
    expiresIn: 1000 * 60 * 60 * 24 * 3, // 3 dni
    generateTimeout: 1000 * 60
  }
});
```

Odrezek kode 4.2: Inicializacija začasnega spomina za seznam proizvajalcev

Pri razpoznavanju proizvajalcev smo morali ugotoviti, kako jih med seboj ločiti v dokumentu HTML. Ugotovili smo, da se seznam vseh proizvajalcev ter njihovih modelov nahaja na vstopni strani spletne aplikacije Avto.net. Obkrožen je z značko `<select>`, ki ima identifikator *»make«* (odrezek kode 4.3). Vsak proizvajalec je obdan z značko `<option>`, ki vsebuje identifikator pod atributom *»value«*.

```
<select id="make" name="znamka">
    <option value="Audi">Audi</option>
    <option value="BMW">BMW</option>
    <option value="Citroen">Citroen</option>
    <option value="Ford">Ford</option>
    <option value="Mercedes-Benz">Mercedes-Benz</option>
    <!--// ... -->
</select>
```

Odrezek kode 4.3: Značka `<select>`, ki zajema vse proizvajalce

[5] V primeru, da se doda nov proizvajalec ali model.

Modeli so tako kot proizvajalci zajeti v znački *<select>* (odrezek kode 4.4), ki ima identifikator *»model«*. Vsak model je obkrožen z značko *<option>*, ki v atributu *»value«* vsebuje identifikator ter v atributu *»class«* identifikator proizvajalca.

```
<select id="model" name="model">
    <!--// ... -->
    <option value="145" class="Alfa Romeo">145</option>
    <option value="146" class="Alfa Romeo">146</option>
    <option value="147" class="Alfa Romeo">147</option>
    <!--// ... -->
    <option value="A4" class="Audi">A4</option>
    <option value="A5" class="Audi">A5</option>
    <option value="A6" class="Audi">A6</option>
    <option value="A7" class="Audi">A7</option>
    <option value="A8" class="Audi">A8</option>
    <!--// ... -->
</select>
```

Odrezek kode 4.4: Značka `<select>`*, ki zajema vse modele*

Z zbranimi informacijami smo nato implementirali funkcijo (odrezek kode 4.6), ki razpozna vse proizvajalce ter modele. Po končani operaciji seznam vsebuje vse proizvajalce. Vsak proizvajalec vsebuje svoje modele v podseznamu *»models«*. Seznam se nato kot objekt JSON shrani v začasni spomin. Dostop do seznama je možen z metodo HTTP (Hypertext Transfer Protocol) GET na dostopno točko *»/manufacturers/get«* (odrezek kode 4.5) [6].

```
HapiServer.route({
  method: 'GET',
  path: '/manufacturers/get',
  handler: function (request, reply) {
    HapiServer.methods.getManufacturers(function (error, manufacturers) {
      if (error) throw error;
      reply(manufacturers);
    })
  }
});
```

Odrezek kode 4.5: Dostopna točka do proizvajalcev

```
function parseManufacturersAndModels() {
  let key, model, manufacturer,
    manufacturers = [];

  $("select#make option").each(function () {
    manufacturer = {};
    manufacturer.key = $(this).attr("value");
    manufacturer.name = $(this).text().trim();
    manufacturer.models = [];

    if (manufacturer.key) {
      key = manufacturer.key.replace(" ",".");

      $(`select#model option.${key}`).each(function () {
        model = {};
        model.key = $(this).attr("value");
        model.name = $(this).text().trim();

        if (!/seznamu/.test(model.key)) {
          manufacturer.models.push(model);
        }
      });

      manufacturers.push(manufacturer);
    }
  });

  manufacturers = _.uniq(manufacturers, false, function (manufacturer) {
    return manufacturer.name;
  });

  return manufacturers;
}
```

Odrezek kode 4.6: Razpoznavanje proizvajalcev ter modelov

4.1.2 Oglasi

Glavno vlogo pri spletni storitvi ima razpoznavanje oglasov. Razpoznane oglase bo nato prikazovala mobilna aplikacija. Razpoznavanje se ob uporabi spletne storitve izvede ob vsakem povpraševanju po oglasih. Izvesti se mora hitro ter natančno.

Prva stopnja implementacije je generiranje naslova URL. Ta nam bo ob klicu vrnil dokument HTML z želenimi oglasi. Za generiranje naslova URL je potrebno vključiti vseh 65 parametrov. Najprej smo ustvarili seznam vseh parametrov (odrezek kode 4.7), ki so nastavljeni na privzete vrednosti.

```
const defaultParameters = {
  znamka: "",
  model: "",
  modelID: "",
  tip: "katerikoli%20tip",
  cenaMin: 0,
  cenaMax: 999999,
  letnikMin: 0,
  letnikMax: 2090,
  bencin: 0,
  ccmMin: 0,
  ccmMax: 99999,
  mocMin: "",
  mocMax: "",
  kmMin: 0,
  kmMax: 9999999,
  stran: 1,
  // ...
};
```

Odrezek kode 4.7: Seznam parametrov s privzetimi vrednostmi

Ob vsakem povpraševanju nam nato funkcija (odrezek kode 4.8) generira naslov URL. Generiranje upošteva vrednosti želenih parametrov. Prejetim parametrom nato doda ostale privzete parametre ter jih spoji v pravilno obliko [7].

```
const ADS_URL = "https://www.avto.net/Ads/results.asp?";
// ...
function generateUrl(parameters) {
  let combinedParameters = [];
  parameters = parameters || {};

  _.defaults(parameters, defaultParameters);
  _.each(parameters, function (value, key) {
    combinedParameters.push(key + "=" + value);
  });
  return ADS_URL + combinedParameters.join("&");
}
```

Odrezek kode 4.8: Generiranje naslova URL

Z generiranim naslovom URL se nato opravi povpraševanje na spletno aplikacijo Avto.net. Ta nam vrne dokument HTML, ki vsebuje oglase za podan filter. Naloga je bila, da v dokumentu HTML najdemo vzorec, kako najlažje ločiti oglase od ostalih informacij. Po temeljitem pregledu značk v dokumentu HTML smo ugotovili, da je vsak oglas obdan z značko * *. Značko je možno identificirati s ključno besedo *»ResultsAd«*, ki je shranjena v atributu *»class«* (odrezek kode 4.9).

```
<div class="ResultsAd">
  <div class="ResultsAdPhotoContainer">
    <div class="ResultsAdPhotoTop">
      <a href="../Ads/details.asp?id=12695428">
        <img src="https://images.avto.net/12695428/1_160.jpg">
      </a>
    </div>
  </div>
  <div class="ResultsAdData">
    <a class="Adlink" href="../Ads/details.asp?id=12695428">
      <span>Audi A1 ...</span>
    </a>
    <ul>
      <li>Letnik 1.registracije:2015</li>
      <li>20000 km</li>
      <li>bencinski motor, 999 ccm, 70 kW  / 95 KM</li>
      <li>ročni menjalnik (5 pr.)</li>
    </ul>
    <!--//... -->
  </div>
  <div class="ResultsAdPriceLogo">
    <div class="ResultsAdPrice">15.990 €</div>
    <!--//... -->
  </div>
</div>
```

Odrezek kode 4.9: Značke posameznega oglasa

Po ugotovitvah smo implementirali funkcijo (odrezek kode 4.10), ki pregleda vse značke * *, ki v atributu *»class«* vsebujejo ključno besedo, ter za vsak posamezni oglas izvede razpoznavo podatkov [8]. Preostali del dokumenta HTML nas ne zanima.

```
function parseAds(manufacturers) {
  let ads = [];

  $("div.ResultsAd").each(function () {
    let ad = {
      id : parseId($("a.Adlink", this).attr("href")),
      price : parsePrice($("div.ResultsAdPrice", this).text())
    };
        ad.thumb = parseThumb($("div.ResultsAdPhotoContainer img",
this).attr("src"), ad.id);
        ad.url = getUrl(ad.id);
    _.extend(ad, parseManufacturerAndModel($("a.Adlink > span",
this).text(), manufacturers));
    _.extend(ad, parseAdInfo($("div.ResultsAdData > ul > li", this)));
    ads.push(ad);
  });

  return ads;
}
```

Odrezek kode 4.10: Razpoznavanje oglasov

Za vsak oglas posebej smo želeli pridobiti čim več uporabnih podatkov. Po pregledu oglasov na spletni aplikaciji smo se odločili, da za vsak oglas pridobimo naslednje podatke:

- identifikator oglasa,
- proizvajalca ter model,
- ceno,
- letnik,
- prevoženi kilometri,
- menjalnik,
- gorivo,
- prostornino ter moč motorja,
- naslov URL, ki vodi do fotografije.

Za vsak podatek smo potrebovali natančen regularni izraz, ki nam bo pridobil podatek iz vsebine ali atributa značk [9].

4.1.2.1 Identifikator

Vsak oglas vsebuje identifikator. Ta se nahaja na več mestih v oglasu. Izbrali smo značko *<a>* (odrezek kode 4.9). Značko identificiramo po atributu *»class«* s ključno besedo *»Adlink«*. Vsebina atributa *»href«* se pošlje funkciji (odrezek kode 4.11), ki z regularnim izrazom vrne identifikator.

```
// haystack = $("a.Adlink", this).attr("href")
function parseId(haystack) {
  return Number(/id=(\d+)/.exec(haystack)[1]);
}
```

Odrezek kode 4.11: Pridobivanje identifikatorja

4.1.2.2 Cena

Vsak oglas vsebuje ceno, včasih pa ta ni podana v numerični obliki (npr. 10.000 €), temveč kot besedna zveza »Pokliči za ceno!« Cena je podana v znački * * (odrezek kode 4.9). Značko identificiramo po atributu *»class«* s ključno besedo *»ResultsAdPrice«*. Vsebina značke se pošlje funkciji (odrezek kode 4.12), ki z regularnim izrazom preveri, ali gre za numerično valuto,[6] ter vrne pravilno vrednost.

```
// haystack = $("div.ResultsAdPrice", this).text()
function parsePrice(haystack) {
  return (/€/.test(haystack)) ?
    /\d+\.?\d+ €/.exec(haystack)[0] : "";
}
```

Odrezek kode 4.12: Pridobivanje cene

[6] Vedno vsebuje znak *»€«*.

4.1.2.3 Fotografija

Vsak oglas vsebuje fotografije, vendar če uporabnik ni dodal fotografije, vsebuje privzeto. Naslov URL, ki vodi do fotografije, je podan v znački <*img*> (odrezek kode 4.9), znotraj značke <*div*>, ki v atributu *»class«* vsebuje ključno besedo *»ResultsAdPhotoContainer«*. Vsebina značke se pošlje funkciji (odrezek kode 4.13), ki z regularnim izrazom preveri, ali gre za privzeto fotografijo, in nato generira pravilen naslov URL.

```
// haystack = $("div.ResultsAdPhotoContainer img", this).attr("src")
function parseThumb(haystack, id) {
  return (!/_graphics/.test(haystack)) ?
    `http://images.avto.net/${id}/1.jpg` : "";
}
```

Odrezek kode 4.13: Pridobivanje naslova URL, ki vodi do fotografije

4.1.2.4 Proizvajalec ter model

Vsak oglas vsebuje proizvajalca ter model. Podana sta v vsebini značke <*span*> (odrezek kode 4.9), znotraj značke <*a*>, ki v atributu *»class«* vsebuje ključno besedo *»Adlink«*. Nato nam funkciji (odrezek kode 4.14) iz seznama proizvajalcev z regularnimi izrazi pridobita pravilnega proizvajalca ter model.

```
// haystack = $("a.Adlink > span", this).text()
function parseManufacturer(haystack, manufacturers) {
  for (let i = 0; i < manufacturers.length; i++) {
    if (new RegExp("\\b" + manufacturers[i].name + "\\b").test(haystack))
{
      return manufacturers[i];
    }
  }
}

function parseModel(haystack, models) {
  for (let i = 0; i < models.length; i++) {
    if (new RegExp("\\b" + models[i].name + "\\b").test(haystack)) {
      return models[i];
    }
  }
}
```

Odrezek kode 4.14: Pridobivanje proizvajalcev ter modelov

4.1.2.5 Ostali podatki

Vsi ostali podatki so podani v seznamu z značkami <*li*> (odrezek kode 4.9), ki nimajo atributov in so brez indikatorjev, kaj vsebujejo. Poiskali smo unikatne ključne besede, ki smo jih uporabili v regularnih izrazih (tabela 4.1) za sklepanje ter razpoznavo podatkov.

Tabela 4.1: Regularni izrazi za razpoznavo preostalih podatkov

Podatek	Regularni izraz
Letnik	`/registracije:(\d+)/`
Prevoženi kilometri	`/(\d+) km/`
Menjalnik	`/avtomatski/`
Gorivo	`[/plin/, /bencinski/, /diesel/, /hibridni/, /elektro/]`
Prostornina + moč	`/(\d+) ccm/, /(\d+) kW/, /(\d+) KM/`

4.1.2.6 Uporaba storitve

Povpraševanje se izvede na dostopno točko *»/ads/get«* (odrezek kode 4.15) z metodo HTTP POST ter filtrom JSON (odrezek kode 4.16). Spletna storitev po razpoznavanju vrne objekt JSON (odrezek kode 4.17). Ta vsebuje seznam oglasov za želen filter.

```
HapiServer.route({
  method: 'POST',
  path: '/ads/get',
  handler: function (request, reply) {
    HapiServer.methods.getManufacturers(function (error, manufacturers) {
      if (error) throw error;
      Ads.get(request.payload, manufacturers, function (error, ads) {
        if (error) throw error;
        reply(ads);
      })
    })
  }
});
```

Odrezek kode 4.15: Dostopna točka do oglasov

```
{
  "znamka":"Audi",
  "model":"A6",
  "cenaMin":"1500",
  // ...
}
```

Odrezek kode 4.16: Primer filtra v formatu JSON

```
[
  {
    "id": 12692415,
    "manufacturer": "Audi",
    "model": "A4",
    "price": "2.890 €",
    "thumb": "http://images.avto.net/12692415/1.jpg",
    "year": 2001,
    "transmission": 0,
    "fuel": 1,
    "distance": 299000,
    "engine": 1896,
    "power": 96,
    "horsepower": 131,
    "url": https://www.avto.net/Ads/details.asp?id=12692415
  },
  // ...
]
```

Odrezek kode 4.17: Končni objekt JSON, ki vsebuje seznam oglasov

4.2 Mobilna aplikacija

Za mobilno aplikacijo smo se odločili zaradi naslednjih razlogov:

1. spletna aplikacija v mobilnem spletnem brskalniku ne more izkoristiti androidovih funkcij, kot so različni senzorji, podatkovne baze, notifikacije itd., medtem ko jih mobilna aplikacija lahko;
2. spletna aplikacija ob vsaki interakciji prenese celotni dokument HTML, stile, fotografije itd. Mobilna aplikacija na drugi strani prenese samo razpoznane podatke o oglasih, kar se pozna pri preneseni količini podatkov s spleta;
3. mobilna aplikacija omogoča veliko boljšo interakcijo ter uporabniško izkušnjo.

Z mobilno aplikacijo smo želeli izboljšati uporabniško izkušnjo, ki jo imamo ob uporabi spletne aplikacije Avto.net v mobilnem brskalniku.

Zastavili smo si, da mora mobilna aplikacija izpolnjevati naslednje pogoje:

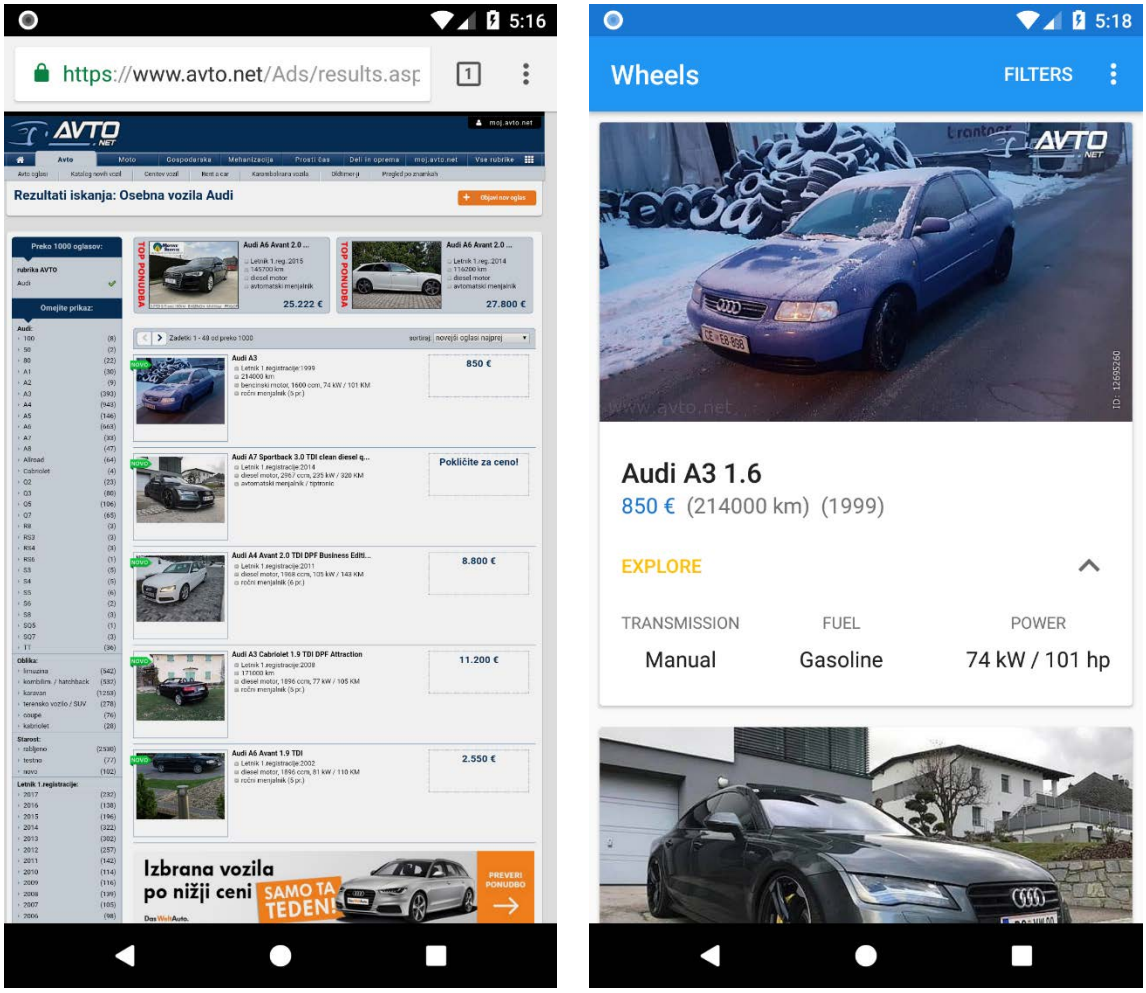
1. zaradi mobilnih omrežij mora prenašati minimalno količino podatkov;
2. posamezen oglas se mora prikazati prilagojen manjšemu zaslonu in pomembnosti informacij;
3. omogočiti mora sprotno nalaganje oglasov za pomikanje po njih;
4. omogočiti mora dodajanje filtrov, ki se lahko poljubno aktivirajo;
5. obveščati nas mora o novih oglasih za trenutno aktiven filter;
6. posamezen zapis mora vsebovati neposredno povezavo na spletno aplikacijo Avto.net, kjer si uporabnik lahko podrobno ogleda celoten oglas.

4.2.1 Prikaz oglasov

Zasloni mobilnih naprav so majhni in omejeni, kar pomeni, da je potrebno spletne aplikacije prilagoditi, če želimo ohraniti prijetno uporabniško izkušnjo. Spletna aplikacija Avto.net nima prilagojenega prikaza za mobilne naprave (slika 3.1). Zelo težko je razbrati posamezne podatke o oglasih, tudi sama interakcija z aplikacijo je otežena in neprijetna.
Implementirali smo prilagojen prikaz oglasa, ki prikaže minimalen nabor informacij o oglasu na uporabniku prijazen način. Vsak oglas vsebuje tudi gumb *»Explore«*, ki nam ob interakciji zažene mobilni brskalnik ter prikaže več podrobnosti o izbranem oglasu v spletni aplikaciji Avto.net.
Po temeljitem razmisleku ter posvetovanju smo se odločili:

1. najpomembnejši podatek pri oglasu je fotografija. Ta v naši aplikaciji zaseda več kot 50 % zapisa;
2. sledi proizvajalec ter model, ki sta poudarjena z večjo in odebeljeno pisavo;
3. sledi cena, poudarjena z drugo barvo;
4. ob ceni sta pomembna tudi podatka o prevoženih kilometrih ter letnik prve registracije in sta zapisana v oklepajih;
5. podatki o menjalniku, gorivu ter moči motorja so privzeto skriti. Prikažemo jih lahko s pritiskom na puščico.

Primerjava med prikazom oglasov v mobilnem brskalniku ter implementirani aplikaciji je na sliki 4.1.



Slika 4.1: Primerjava prikaza oglasov

4.2.2 Nalaganje naslednjih oglasov

Večina spletnih aplikacij, ki prikazujejo nek seznam (npr. izdelkov, oglasov itd.), prikazuje omejeno število elementov na posamično stran (angl. *Page*), zato imajo ponavadi pod ter/ali nad oglasi prikazan seznam strani (angl. *Paging*), do katerih lahko dostopamo. Takšen primer imamo tudi pri prikazu oglasov na spletni aplikaciji Avto.net. Težava se pojavi, kadar uporabljamo spletno aplikacijo v mobilnem brskalniku, saj so številke ostalih strani tako majhne, da lahko pomotoma pritisnemo dve hkrati ali napačno. Na splošno je uporaba takšnih spletnih aplikacij neprijetna, kajti vsakič, ko želimo prikazati naslednjih nekaj elementov, moramo počakati, da se celotna stran na novo prenese in naloži.

To težavo smo v mobilni aplikaciji rešili tako, da smo implementirali nalaganje naslednjih oglasov (angl. *Endless Scrolling*). Ob premikanju po seznamu navzdol nam aplikacija preverja, koliko elementov je še do konca seznama, ki še niso prikazani. Če prekoračimo omejitev petih oglasov, naložimo dodatne. Implementacija preverjanja ter nalaganja naslednjih oglasov je prikazana na odrezku kode 4.18, kjer vsebuje spremenljivka *»visibleThreshold«* omejitev prikazanih oglasov.

```
// ...
private int visibleThreshold = 5;
// ...
if (!loading
    && (lastVisibleItemPosition + visibleThreshold) > totalItemCount)
{
  currentPage++;
  onLoadMore(currentPage, totalItemCount, view);
  loading = true;
}
// ...
```

Odrezek kode 4.18: Nalaganje naslednjih oglasov

4.2.3 Filtri

Spletna aplikacija Avto.net ima zelo obsežno spletno formo, ki vsebuje veliko (65) parametrov, s katerimi lahko natančno določimo prikaz oglasov. Težava nastane pri vsakodnevni uporabi spletne aplikacije, saj moramo vedno znova izpolniti formo.
Težavo smo rešili z implementacijo filtrov, ki jih hranimo v podatkovni bazi (angl. *Database*). So zelo pomemben del aplikacije, kajti hranijo parametre (ki jih poljubno nastavimo), s katerimi se izvaja povpraševanje na spletno storitev (odrezek kode 4.19).

```
filterDao.getActive()
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribeWith(new DisposableSingleObserver<Filter>() {
      @Override
      public void onSuccess(Filter filter) {
        filter.setPage(nextPage);
        // ...
      }
      // ...
    });
```

Odrezek kode 4.19: Pridobivanje aktivnega filtra

Pri uporabi androidove podatkovne baze nam je pomagala Googlova knjižnica *»Room«* [10]. Na podlagi opomb (angl. *Anotations*) v kodi nam knjižnica generira razrede, s katerimi dostopamo do baze. Opombe za povpraševanja SQL (*Structured Query Language*), ki so bila potrebna za pravilno delovanje filtrov, so prikazana na odrezku kode 4.20.

```
// filterDao
@Query("SELECT * FROM filter")
Single<List<Filter>> getAll();

@Query("SELECT * FROM filter WHERE active = 1")
Single<Filter> getActive();

@Insert(onConflict = OnConflictStrategy.REPLACE)
void insert(Filter filter);

@Insert(onConflict = OnConflictStrategy.REPLACE)
void insertAll(List<Filter> filters);

@Delete
void delete(Filter filter);
// ...
```

Odrezek kode 4.20: Povpraševanja SQL za upravljanje s filtri

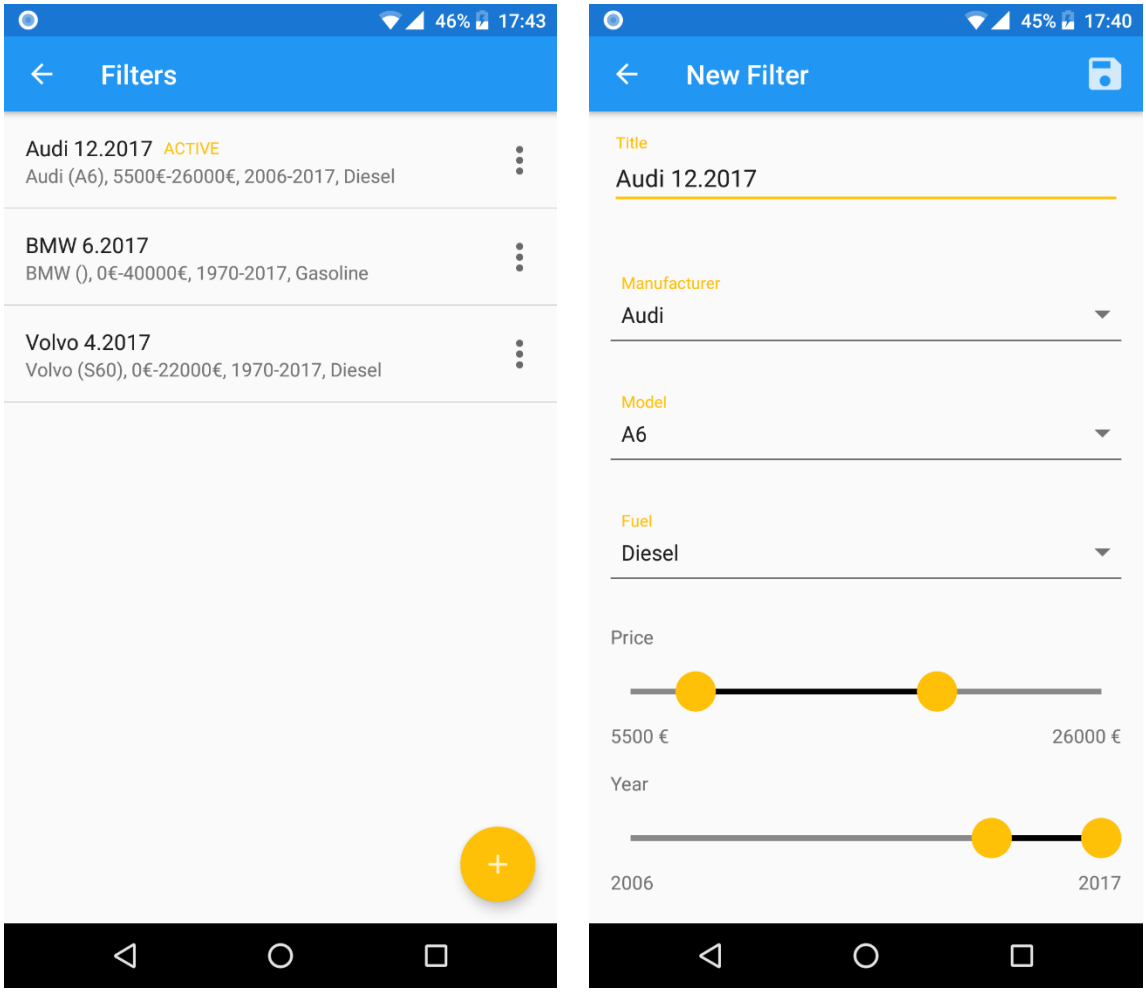
Za pravilno delovanje filtrov potrebujemo tudi seznam vseh proizvajalcev ter njihovih modelov, ki jih pridobimo s spletno storitvijo. Seznam se prenese ob ustvarjanju prvega filtra, nakar se hrani v podatkovni bazi. Opombe za povpraševanja SQL, so prikazana na odrezku kode 4.21.

```
// manufacturerDao
@Query("SELECT * FROM manufacturer ORDER BY name ASC")
Single<List<Manufacturer>> getAll();
// ...

// modelDao
@Query("SELECT * FROM model WHERE manufacturerId = :manufacturerId")
Single<List<Model>> getAll(Long manufacturerId);
// ...
```

Odrezek kode 4.21: Povpraševanja SQL, potrebna za pridobivanje proizvajalcev

Prikaz dodanih filtrov ter ustvarjanje novega filtra lahko vidimo na sliki 4.2.



Slika 4.2: Seznam dodanih filtrov ter dodajanje novega filtra

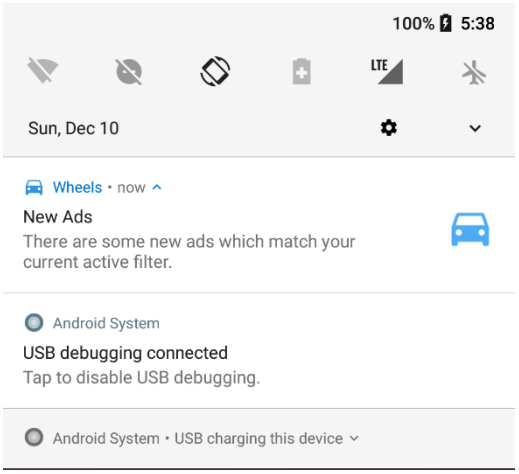
4.2.4 Obveščanje o novih oglasih

Spletna aplikacija Avto.net je namenjena iskanju želenega avtomobila, kar pomeni, da bomo aplikacijo uporabljali večkrat dnevno[7], dokler avtomobila ne najdemo oz. kupimo. Težavo smo rešili z implementacijo preverjanja o novih oglasih in tako prihranili čas, ki ga porabimo za njihovo preverjanje na spletni aplikaciji. Mobilna aplikacija nam v obliki notifikacij (slika 4.3) sporoči, ko se pojavi nov oglas.

Ob vsakem nalaganju oglasov, bodisi aktivnem ali pasivnem (v ozadju), se nam shrani zadnji prenesen identifikator oglasa. Vsak nov oglas ima vedno višji identifikator kot zadnji preneseni, kar preverimo (odrezek kode 4.22) ter obvestimo uporabnika.

```
public static boolean isGreaterThenLastId(Context context, int newId) {
  SharedPreferences prefs = PreferenceManager.getDefaultSharedPrefer-
ences(context);
  int lastId = prefs.getInt(KEY_LAST_ID, 0);
  return (newId > lastId);
}
```

Odrezek kode 4.22: Preverjanje o novih oglasih



Slika 4.3: Notifikacija o novih oglasih

[7] Zaradi dodajanja novih oglasov.

5. SKLEP

Implementacijo rešitve, ki je predstavljena v diplomski nalogi, smo se lotili z namenom, da prikažemo način, kako s pomočjo razpoznavanja podatkov preprosto in efektivno prikažemo ter optimiziramo spletno aplikacijo na mobilni napravi.

Razlogi, zakaj podatke razpoznava spletna storitev in ne mobilna aplikacija, so:

- Mobilni aplikaciji ni potrebno prenašati celotnega dokumenta HTML, temveč samo izluščene oglase. S tem prihranimo na prenosu podatkov.
- Spletna storitev ima večjo operacijsko moč kot mobilna naprava. S tem prihranimo na času razpoznavanja ter pri porabi baterije.

Spletna storitev samo razpozna podatke in jih pošlje mobilni aplikaciji. Optimalna rešitev bi bila, da bi spletna storitev razpoznane podatke hranila v lokalni podatkovni bazi. To ni možno zaradi pravnih pravil spletne aplikacije, kar bi v našem primeru pomenilo krajo podatkov ter podvajanje.

S samim razpoznavanjem podatkov smo ugotovili, da se za eno povpraševanje, kjer se v spletni aplikaciji poprečno prenese 2 MB, prenese samo 13 KB podatkov (kar je za 1987 KB oz. 154-kratno zmanjšanje prenosa). Seveda v naš prenos nismo upoštevali fotografij, ki jih prenaša mobilna aplikacija, vendar moramo omeniti, da spletna aplikacija Avto.net vedno prenese približno 2 MB podatkov, ker prenese vse fotografije oglasov. Naša aplikacija prenese samo tiste fotografije, katerih oglase trenutno aktivno prikazujemo na zaslonu aplikacije (kar je lahko nekaj fotografij). Pri preverjanju novih oglasov naša aplikacija preverja v ozadju in ne prenaša nobenih fotografij, saj to ni potrebno.

Z implementacijo filtrov v mobilni aplikaciji smo eliminirali čas, ki ga potrebujemo za dnevno izpolnjevanje obširne spletne forme. S samodejnim preverjanjem in obveščanjem o novih oglasih, smo eliminirali še preostali čas, ki smo ga potrebovali za dnevno uporabo spletne aplikacije.

Implementacija rešitve nam prihrani tako na prenosu podatkov kot času, ki je potreben za uporabo spletne aplikacije. Nudi tudi prijetno uporabniško izkušnjo, saj sta v mobilni aplikaciji prikaz ter interakcija dosti bolj prilagojena za mobilni zaslon, kot sta pri spletni aplikaciji Avto.net.

VIRI

1. **Wikipedia.** Node.js. *Wikipedia.* [Elektronski] [Navedeno: 17. december 2017.] https://en.wikipedia.org/wiki/Node.js.

2. —. Android (operating system). *Wikipedia.* [Elektronski] [Navedeno: 17. december 2019.] https://en.wikipedia.org/wiki/Android_(operating_system).

3. **StatCounter.** Mobile Operating System Market Share Worldwide. *StatCounter.* [Elektronski] [Navedeno: 17. december 2017.] http://gs.statcounter.com/os-market-share/mobile/worldwide.

4. **Avtonet d.o.o.** Cenik - mali oglasi. *Avto.net.* [Elektronski] 1. Januar 2017. [Navedeno: 13. december 2017.] https://www.avto.net/cenik/cenikOglasi2017.pdf.

5. **Hapi.js.** Server Methods. *Hapi Tutorials.* [Elektronski] [Navedeno: 17. december 2017.] https://hapijs.com/tutorials/server-methods?lang=en_US.

6. —. Routing. *Hapi Tutorials.* [Elektronski] [Navedeno: 17. december 2017.] https://hapijs.com/tutorials/routing?lang=en_US.

7. **Underscore.js.** Underscore.js. *Underscore.js.* [Elektronski] [Navedeno: 17. december 2017.] http://underscorejs.org/#defaults.

8. **Cheerio.js.** Cheerio. *Cheerio.* [Elektronski] [Navedeno: 17. december 2017.] https://cheerio.js.org/.

9. **Mozilla Corporation.** Regular Expressions - JavaScript. *MDN web docs.* [Elektronski] [Navedeno: 17. december 2017.] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions.

10. **Google LLC.** Room Persistence Library. *Android Developers.* [Elektronski] [Navedeno: 17. december 2017.] https://developer.android.com/topic/libraries/architecture/room.html.

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

FERI

IZJAVA O AVTORSTVU IN ISTOVETNOSTI TISKANE IN ELEKTRONSKE OBLIKE ZAKLJUČNEGA DELA

Ime in priimek študent-a/-ke: JANOŠ ŽIBRAT
Študijski program: RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE
Naslov zaključnega dela: SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH
Mentor: DOC. DR. TOMAŽ KOSAR
Somentor: /

Podpisan-i/-a študent/-ka JANOŠ ŽIBRAT

- izjavljam, da je zaključno delo rezultat mojega samostojnega dela, ki sem ga izdelal/-a ob pomoči mentor-ja/-ice oz. somentor-ja/-ice;
- izjavljam, da sem pridobil/-a vsa potrebna soglasja za uporabo podatkov in avtorskih del v zaključnem delu in jih v zaključnem delu jasno in ustrezno označil/-a;
- na Univerzo v Mariboru neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico ponuditi zaključno delo javnosti na svetovnem spletu preko DKUM; sem seznanjen/-a, da bodo dela deponirana/objavljena v DKUM dostopna široki javnosti pod pogoji licence Creative Commons BY-NC-ND, kar vključuje tudi avtomatizirano indeksiranje preko spleta in obdelavo besedil za potrebe tekstovnega in podatkovnega rudarjenja in ekstrakcije znanja iz vsebin; uporabnikom se dovoli reproduciranje brez predelave avtorskega dela, distribuiranje, dajanje v najem in priobčitev javnosti samega izvirnega avtorskega dela, in sicer pod pogojem, da navedejo avtorja in da ne gre za komercialno uporabo;
- dovoljujem objavo svojih osebnih podatkov, ki so navedeni v zaključnem delu in tej izjavi, skupaj z objavo zaključnega dela;
- izjavljam, da je tiskana oblika zaključnega dela istovetna elektronski obliki zaključnega dela, ki sem jo oddal/-a za objavo v DKUM.

Uveljavljam permisivnejšo obliko licence Creative Commons:_____(navedite obliko)

Začasna nedostopnost:

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, zaščite poslovnih skrivnosti, varnosti ljudi in narave, varstva industrijske lastnine ali tajnosti podatkov naročnika:

_____(naziv in naslov naročnika/institucije) ne sme biti javno dostopno do _____(datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela). To se nanaša na tiskano in elektronsko obliko zaključnega dela.

Temporary unavailability:

To ensure competition priority, protection of trade secrets, safety of people and nature, protection of industrial property or secrecy of customer's information, the thesis _____(institution/company name and address) must not be accessible to the public till_____(delay date of thesis availability to the public must not exceed the period of 3 years after thesis defense). This applies to printed and electronic thesis forms.

Datum in kraj: MARIBOR, 20.12.2017

Podpis študent-a/-ke:

Podpis mentor-ja/-ice: _____
(samo v primeru, če delo ne sme biti javno dostopno)

Ime in priimek ter podpis odgovorne osebe naročnika in žig:

_____
(samo v primeru, če delo ne sme biti javno dostopno)

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija

FERI

IZJAVA O USTREZNOSTI ZAKLJUČNEGA DELA

Podpisani mentor/-ica : DOC. DR. TOMAŽ KOSAR
(ime in priimek mentor-ja/-ice)

in somentor/-ica (eden ali več, če obstajajo): /
(ime in priimek somentor-ja/-ice)

Izjavlja-m/-va/-mo, da je študent/-ka

Ime in priimek: JANOŠ ŽIBRAT, *ID številka:* E1031029,

vpisna številka: E1031029, *na študijskem programu:*

RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE

izdelal/-a zaključno delo z naslovom:

SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH

(naslov zaključnega dela v slovenskem jeziku)

v skladu z odobreno temo zaključnega dela, navodili o pripravi zaključnih del in mojimi (najinimi/našimi) navodili.

Preveril/-a/-i in pregledal/-a/-i sem/sva/smo poročilo o preverjanju podobnosti vsebin z drugimi deli (priloga) in potrjujem/potrjujeva/potrjujemo, da je zaključno delo ustrezno.

Datum in kraj:
MARIBOR, 20.12.2017

Podpis mentor-ja/-ice:

Datum in kraj:

Podpis somentor-ja/-ice (če obstaja):

Priloga:
- Poročilo o preverjanju podobnosti vsebin z drugimi deli.

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija

FERI

IZJAVA O OBJAVI OSEBNIH PODATKOV

Ime in priimek diplomant-a/ magistrant-/-ke: JANOŠ ŽIBRAT

ID številka: E1031029

Študijski program: RAČUNALNIŠTVO IN INFORMACIJSKE TEHNOLOGIJE

Naslov zaključnega dela: SPLETNA STORITEV ZA RAZČLENJANJE SPLETNE STRANI IN PRIKAZ NA MOBILNIH NAPRAVAH

Mentor/-ica: DOC. DR. TOMAŽ KOSAR

Somentor/-ica: /

Podpisan-i/-a izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto zaključka študija, naslov zaključnega dela) na spletnih straneh Univerze v Mariboru in v publikacijah Univerze v Mariboru.

Datum in kraj: MARIBOR, 20.12.2017

Podpis diploman-ta/magistran-ta/-ke: