

**SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STUČNE STUDIJE
ODSJEK ZA INFORMACIJSKU TEHNOLOGIJU**

DUJE MILATIĆ

**ZAVRŠNI RAD
IZRADA ANDROID APLIKACIJE**

Split, rujan 2015.

**SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STUČNE STUDIJE
ODSJEK ZA INFORMACIJSKU TEHNOLOGIJU**

PREDMET: Oblikovanje web stranica

ZAVRŠNI RAD

KANDIDAT: Duje Milatić

TEMA ZAVRŠNOG RADA: Izrada Android aplikacije

MENTOR: Ljiljana Despalatović, predavač

Split, rujan 2015.

Sadržaj

Sažetak	4
Summary	4
1. Uvod	5
2. Korištene tehnologije	6
2.1. HTML	6
2.2. CSS	7
2.3. JavaScript	9
2.4. SQLite	10
2.5. Phonegap	10
3. Opis praktičnog rada	12
3.1 Planiranje aplikacije	12
3.2. Opis izrade aplikacije	14
3.2.1 Izrada HTML dokumenta	14
3.2.2 Izrada i rad sa bazom podataka	16
3.2.3 Dodavanje, brisanje i pretraživanje zabilješki	17
3.2.4 Dodavanje i pretraživanje kategorija	19
3.2.5 Dodavanje obavijesti zabilješke	22
3.2.6 Dizajniranje aplikacije	23
3.3 Prikaz aplikacije	28
4. Zaključak	33
5. Literatura	34

Sažetak

Naziv završnog rada je "Izrada Android aplikacije". Tema Android aplikacije je aplikacija za vođenje bilježaka u koju se zapisuju zabilješke te se iste zabilješke pohranjuju u SQLite bazu pomoću JavaScripta. Zabilješke se pohranjuju u SQLite bazu kako bi ostali spremljeni te kako se ne bi izgubili kod zatvaranja, odnosno ponovnog otvaranja same aplikacije. Aplikacija omogućava pisanje novih zabilješki te čitanje, uređivanje i brisanje starih zabilješki.

Aplikacija je napravljena pomoću HTML5 i JavaScripta. Za pisanje samih HTML i JavaScript datoteka je korišten program Notepad++. Da bi se HTML datoteka na Android uređaju mogla pregledavati kao aplikacija potrebno je koristiti okruženje koje to omogućava. U ovom slučaju je korišteno okruženje Phonegap. Aplikacija se kompajlirala kompajlerom Phonegap iz komandne linije (eng. *Command prompt*).

Summary

The title of the thesis is Developing Android application. Developed Android application presents a note application where user can add, edit and delete new notes. All notes are stored in the SQLite database using JavaScript. Notes are stored in a SQLite database, so they would not get lost when closing or reopening the application. The application allows you to write new notes and read, edit and delete old records.

The application was made by using HTML5 and JavaScript. The application is written using Notepad++ software. To view HTML files on an Android smartphone as an application, it is necessary to use a framework that makes it possible. Phonegap framework is used in this case. The application is compiled using Command prompt (cmd).

1. Uvod

Tema završnog rada je oblikovati Android aplikaciju, u ovom slučaju Android aplikaciju za upravljanjem zabilješkama. Aplikacija omogućava dodavanje novih zabilješki, te brisanje, čitanje i uređivanje ranije dodanih zabilješki. Uz to, aplikacija omogućava i mijenjanje izgleda samih zabilješki te i određivanja u koju kategoriju spadaju zabilješke, kako bi ih bilo lakše pretraživati. Također, unutar zabilješke se može odabrati datum i vrijeme te se korisniku javi obavijest na uređaju u odabranom vremenu. Sama aplikacija je namijenjena da se koristi od strane vlasnika uređaja te nije moguće da više korisnika sprema svoje zabilješke na jednom uređaju. Aplikacija je napravljena pomoću HTML-a i JavaScripta te oblikovana pomoću CSS-a.

Pri dodavanju zabilješke, zabilješka se sprema u SQLite bazu pomoću JavaScripta. Baza je kreirana unutar JavaScripta te se sprema na mobilnom uređaju, što ju čini lokalnom aplikacijom. Sama baza podataka će biti kasnije detaljno opisano u trećem poglavlju poglavlju.

Sama aplikacija je pisana u Notepad++ aplikaciji, koja omogućava lakše pisanje HTML i JavaScript datoteka. Nakon toga se aplikacija kompajlira pomoću komandne linije. Kako bi to bilo moguće bilo je potrebno instalirati Java Development Kit (JDK) i Phonegap.

U drugom poglavlju su opisane korištene tehnologije, a u trećem i četvrtom prikazuje sami praktični rad, odnosno pripadna izrada aplikacije te dizajniranje i izgled aplikacije.

2. Korištene tehnologije

2.1. HTML

HTML (*Hyper Text Markup Language*) je prezentacijski jezik koji služi za izradu web stranica. HTML nije programski jezik, već njime opisujemo strukturu i izgled web dokumenta. HTML daje upute web pregledniku na koji način treba prikazivati web stranicu. Osnovni elementi svakog HTML dokumenta su oznake (eng. *tags*) koje označavaju i opisuju kako će se nešto prikazati u web pregledniku.

Oznake su prvi put spomenute 1991., a prva verzija HTML-a je objavljena 1993. godine. HTML je postao standard 2000. godine, nakon što je predstavljen HTML4.01. Najnovija verzija HTML-a je HTML5 koji je predstavljen krajem 2008. godine, a krajem 2014. godine se predstavlja kao stabilna preporuka. Posljednja verzija HTML-a daje mnoštvo novih mogućnosti, odnosno oznaka kao što su `<video>`, `<audio>` i `<canvas>` koje daju mogućnost dodavanja multimedije unutar HTML-a bez korištenja različitih dodataka. Također su dodane oznake koje omogućuju bolju semantiku, kao što su `<header>`, `<nav>`, `<section>`, `<footer>`, `<article>`. Neke od najosnovnijih oznaka koje se koriste gotovo u svakom HTML dokumentu su: `<h1>` do `<h6>` oznake koji se koriste za oblikovanje naslova, `<p>` koji služi za definiranje odlomaka, `` služi za oblikovanje dijelova teksta unutar teksta, `<a>` je HTML oznaka za link, `` služi za umetanje slike unutar HTML dokumenta, `<div>` oznaka služi kao spremnik koji sadržava tekst, slike i slično.

Sve HTML oznake moraju biti napisane unutar znakova `< i >`, a definiraju se kao početne i završne oznake. Primjer početne oznake je `<html>`, a završne `</html>`. Jedina razlika između početne i završne oznake je ta što završna oznaka ima znak `/` ispred imena.

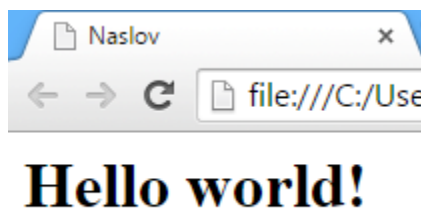
Osnovna struktura HTML-a se sastoji od *Head* i *Body* dijela. Primjer osnovnog HTML dokumenta:

```
<html>
  <head>
    <title> Naslov </title>
  </head>

  <body>
    <h1>Hello world!</h1>
  </body>
</html>
```

Ispis kôda 1. Osnovna HTML stranica

Na slici 1. vidljivo je kako izgleda Ispis kôda 1 u web pregledniku.



Slika 1. Prikaz osnovne HTML stranice

2.2. CSS

CSS (*Cascading Style Sheets*) je stilski jezik koji služi za opis prezentacije dokumenta napisanog pomoću HTML jezika. Pomoću CSS-a definira se način prikazivanja HTML elemenata u web pregledniku. CSS je napravljen od strane W3C (*World Wide Web Consortium*) institucije kako bi HTML dokument bio pregledniji, te kako bi se izbjeglo korištenje HTML oznaka za uređivanje unutar dokumenta, te kako bi se u konačnici odvojilo oblikovanje od sadržaja.

CSS je prvi put spomenut 1994., a CSS1 je predstavljen 1996. godine. CSS2 i njegova nadogradnja CSS2.1 su prezentirani 1998. godine. Najnovija verzija je CSS3 koja je predstavljena 1999. godine, no bitnije nadogradnje su krenule tek 2011. godine, te se od tada redovno nadograđuje sa novim *modulima*. Neke od novosti koje su dodane u CSS3 su: dodavanje sjene tekstu koje se ostvaruje pomoću svojstva *text-shadow*, korištenje više pozadinskih slika za isti element, dodavanje pozadinske slike s nijansama (eng. *linear gradient*), oblikovanje prozirnosti elementa pomoću svojstva *opacity*, dodavanje zaobljenih rubova pomoću *border-radius*. Jedno od bitnijih novih mogućnosti CSS3-a su CSS transformacije koje pomoću svojstva *transform* omogućavaju rotiranje slika na stranici, mijenjanje veličine elementa, pomicanje elementa horizontalno ili vertikalno te ukošavanje elementa. Također velika novost je svojstvo *animation* koje donosi velike promjene budući da više nije potrebno koristiti Flash ili JavaScript kako bi se napravile animacije. Također, tu je i novo svojstvo *transition* koje omogućava mijenjanje već postavljenih atributa kroz CSS. Mana navedenih svojstava je kompatibilnost te je potrebno navoditi različite prefikse za različite preglednike kako bi se svojstva prikazala kako treba. Prefiksi su: *-ms-* za Internet Explorer,

-moz- za Mozillu Firefox, *-webkit-* za Google Chrome, Safari i Operu.

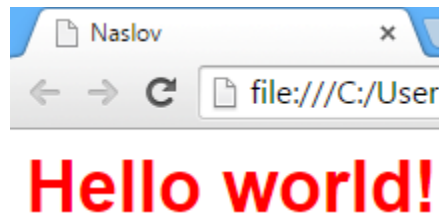
CSS sintaksa se sastoji od selektora (eng. *selector*) i svojstva (eng. *property*). Selektor se odnosi na HTML oznaku koju se želi definirati, primjerice *body*, *p*, *h1*, *img*. Svojstvo je atribut koji se želi promijeniti. Primjer osnovnog CSS dokumenta gdje je *h1* selektor, odnosno oznaka koju se želi definirati, a *color*, *font-family* i *font-size* su svojstva, je prikazano sa ispisom kôda 2.

```
h1{
color: red;
font-family: Arial;
font-size:36px;
}
```

Ispis kôda 2. Osnovni CSS dokument

CSS kôd je povezan sa HTML dokumentom iz prethodnog poglavlja, te stranica u web pregledniku izgleda kao što je prikazano na slici 2. HTML dokument je povezan sa CSS dokumentom pomoću sljedeće linije kôda:

```
<link rel="stylesheet" type="text/css" href="main.css">
```

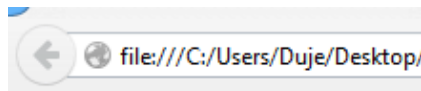


Slika 2 Primjer osnovnog CSS oblikovanja

CSS, također, omogućava korištenje identifikatora i klase. Identifikatori se koriste kada je potrebno definirati svojstva samo jednom elementu, a klase kada je potrebno definirati svojstva više elemenata unutar HTML dokumenta.

```
<html>
<head>
<style>
#naslov{
color: red;
font-family: Arial;
font-size:36px;
}
</style>
</head>
<body>
<h1 id="naslov"> Hello world!</h1>
<h1> Hello world!</h1>
</body>
</html>
```

Ispis kôda 3. Primjer korištenja identifikatora



Hello world!

Hello world!

Slika 3 Primjer korištenja identifikatora

2.3. JavaScript

JavaScript je skriptni programski jezik koji se izvršava u web pregledniku kako bi učinio web stranicu što dinamičnijom. JavaScript se najčešće upotrebljava za: različite efekte sa slikama, otvaranje novih pop-up prozora, rad sa obrascima (npr. provjera ispravnosti upisanog sadržaja u polje), promjena svojstava pojedinih oznaka. Varijable u JavaScriptu se deklariraju sa ključnom riječju `var` (*var znak*) gdje znak označava naziv same varijable. Posebnost JavaScripta je ta što se u tu varijablu mogu spremati različite vrste varijabli, tj. u istu varijablu se može spremiti i nekakav broj i tekst i znak. JavaScript je osjetljiv na mala i velika slova (eng. *case-sensitive*) što znači da varijabla znak nije ista kao i varijabla Znak. Da bi se spriječilo da se cijeli JavaScript kôd izvršava prilikom učitavanja web stranice koriste se funkcije. Funkcije su dijelovi kôda koji se izvršavaju samo kada su pozvani, primjerice prilikom pritiska na gumb ili upisivanja teksta. Funkcije se definiraju sa oznakom *function* i sastoje se od imena funkcije, prosljeđenih parametara koji se nalaze u zagradama te izrada tijela funkcije koji se nalazi unutar vitičastih zagrada.

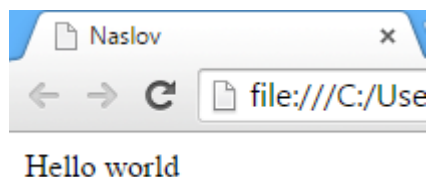
JavaScript je prvi put 1996. godine prezentirao Brendan Eich iz Netscape-a koji ga je izmislio za tada najkorišteniji web preglednik Netscape Navigator 2.0. JavaScript se može pisati unutar HTML dokumenta, a može i u zasebnom dokumentu. Ukoliko se piše unutar HTML-a tada mora započeti sa oznakom `<script>`, odnosno završiti sa oznakom `</script>`

```
<html>
  <head>
    <title> Naslov </title>
  </head>

  <body>
    <script type="text/JavaScript">
      document.write("Hello world");
    </script>
  </body>
</html>
```

Ispis kôda 4. Primjer Javascript ispisa „Hello world“

Primjer JavaScript kôda koji ispisuje „Hello world“ u web pregledniku je prikazan na slici 4.



Slika 4 Primjer JavaScript ispisa „Hello world“

2.4. SQLite

SQLite je relacijska baza podataka temeljena na C programskoj biblioteci. Prednosti SQLite baze podataka su to što su vrlo jednostavne za korištenje, a istovremeno puno brže izvode operacije od ostalih popularnih baza podataka. Također, prednost je što je cijela baza spremljena u samo jednoj datoteci na uređaju te zahtijeva malu memoriju u toku izvršavanja operacija, što ju čini popularnim izborom za baze podataka za uređaje poput mobitela ili MP3 playera.

Ono što čini SQLite bazu bržom od ostalih baza podataka je to što nema klijent/server arhitekturu. Većina ostalih baza podataka ima veliki serverski paket koji je glavni pokretač baze podataka dok SQLite nema odvojen server nego sama aplikacija, kojoj je potrebna pristup bazi, pokreće samu bazu.

2.5. Phonegap

Phonegap je okruženje (eng. *framework*) koje služi za oblikovanje aplikacija za različite platforme (Android, iOS, Windows Phone) pomoću HTML5 i JavaScripta. Kada se aplikacija kreira, ona postaje hibrid Android izvorne aplikacije i web stranice na mobitelu. Programska podrška (eng. *software*) na kojem se Phonegap temelji, je program otvorenog kôda (eng. *open source*) Apache Cordova.

Phonegap je 2009. godine na Web 2.0 konferenciji predstavio Nitobi Software-a, a 2011. godine ih otkupljuje Adobe.

Jedna od velikih prednosti Phonegap-a je što unatoč tome što HTML i JavaScript ne mogu pristupati različitim funkcijama uređaja, npr. kameri, mikrofONU, Phonegap to omogućava kroz mnogobrojne dodatke (eng. *plugin*) koji se mogu dodavati. Prednost je također što se sa istim HTML i JavaScript dokumentima u istom trenutku mogu kreirati aplikacije za više platformi.

Kako bi se aplikacija mogla kompajlirati potrebno je preko komandne linije aplikacije doći do mape gdje su spremljene datoteke aplikacije, odnosno HTML, CSS i JavaScript datoteke. Nakon toga pozivom funkcije *phonegap build*, Phonegap započinje kompajliranje aplikacije.

```
C:\Users\Duje\Documents\Notes>phonegap build android
```

Slika 5 Pozivanje funkcije za kompajliranje

```
C:\Users\Duje\Documents\Notes>phonegap build android
[phonegap] detecting Android SDK environment...
[phonegap] using the local environment
[phonegap] compiling Android...
Buildfile: C:\Users\Duje\Documents\Notes\platforms\android\build.xml

-set-mode-check:

-set-debug-files:

-check-env:
 [checkenv] Android SDK Tools Revision 22.3.0
 [checkenv] Installed at F:\backup\Desktop\Zavrsni\android\sdk

-setup:
 [echo] Project Name: CordovaApp
 [gettype] Project Type: Application

-set-debug-mode:

-debug-obfuscation-check:

-pre-build:

-build-setup:
[getbuildtools] Using latest Build Tools: 19.0.0
 [echo] Resolving Build Target for CordovaApp...
[gettarget] Project Target:   Android 4.4
[gettarget] API level:       19
 [echo] -----
 [echo] Creating output directories if needed...
 [echo] -----
 [echo] Resolving Dependencies for CordovaApp...
[dependency] Library dependencies:
[dependency] -----
[dependency] Ordered libraries:
[dependency] -----
 [echo] -----
 [echo] Building Libraries with 'debug'...

nodeps:

-set-mode-check:

-set-debug-files:

-check-env:
 [checkenv] Android SDK Tools Revision 22.3.0
 [checkenv] Installed at F:\backup\Desktop\Zavrsni\android\sdk
```

Slika 6 Kompajliranje aplikacije

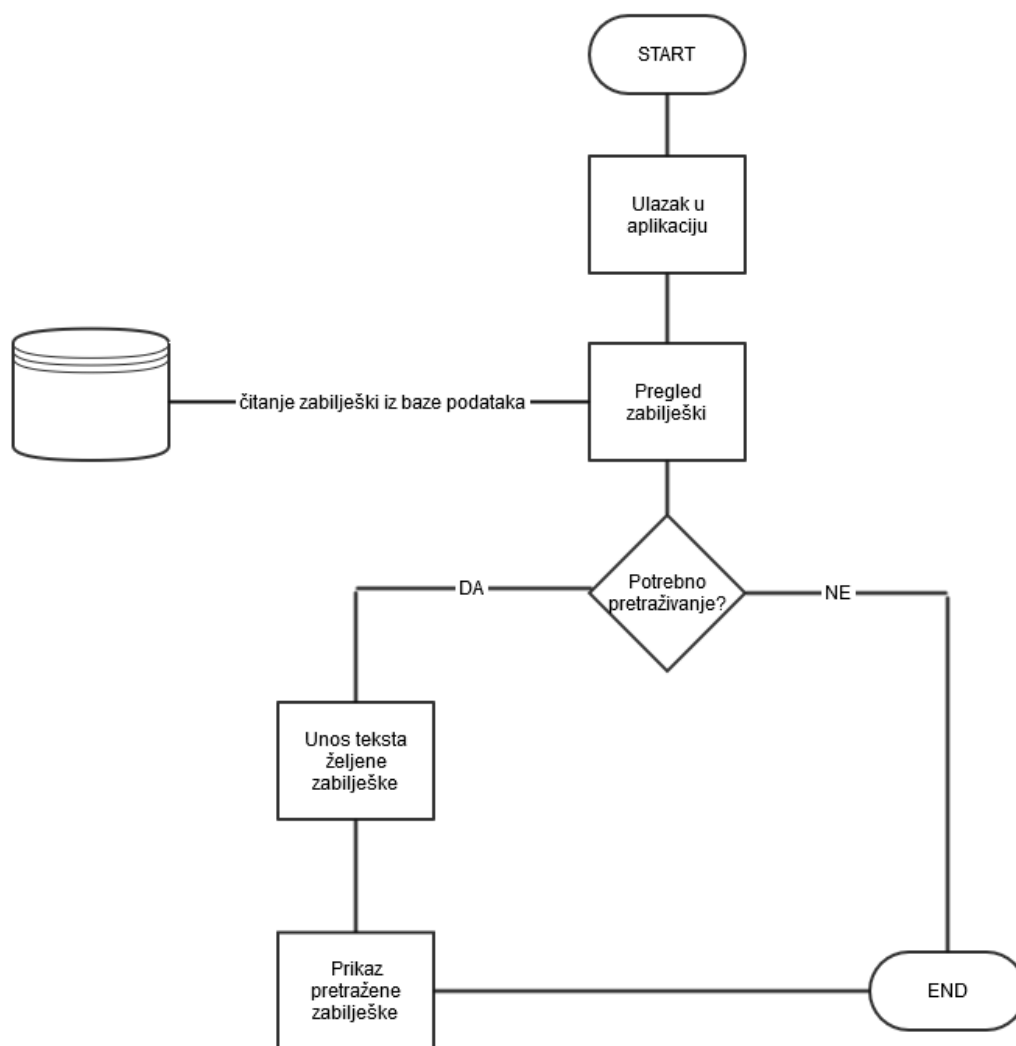
3. Opis praktičnog rada

3.1 Planiranje aplikacije

Osnovni zadatak je bio izrada aplikacije za zapisivanje zabilježaka koja bi pomogla u svakodnevnom radu te omogućila zapisivanje zabilježki bilo gdje i bilo kada. Aplikacija omogućuje spremanje zabilježki, pretraživanje te brisanje samih zabilježki.

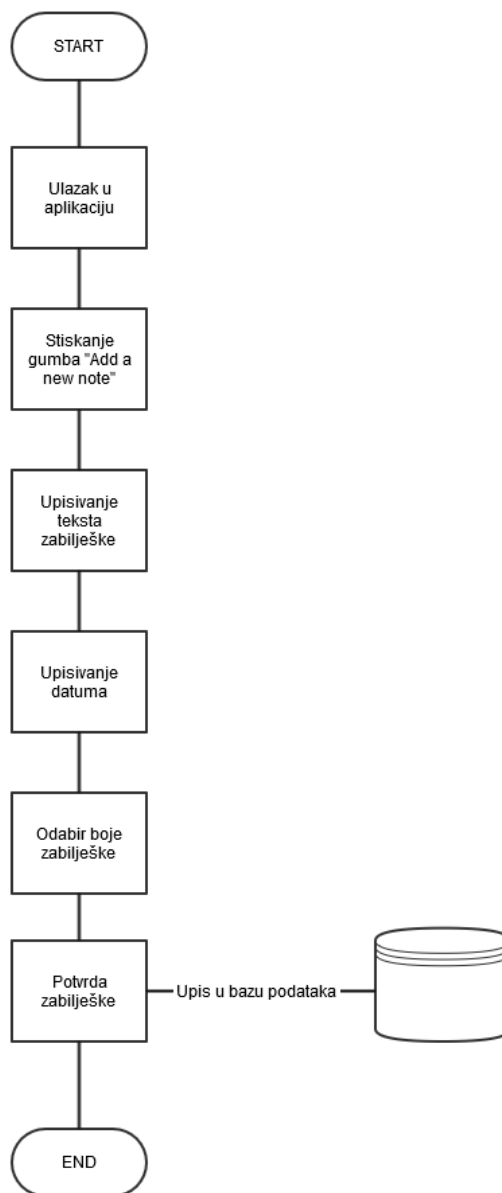
U nastavku su dijagrami toga koji prikazuju čitanje zabilježki, dodavanje nove zabilješke, te brisanje.

Dijagram toka na slici 7. prikazuje čitanje zabilježki. Proces započinje ulaskom u aplikaciju gdje se čitaju zabilješke iz baze podataka te se prikazuju na zaslonu, ukoliko je potrebno, korisnik može pretraživati zabilješke.



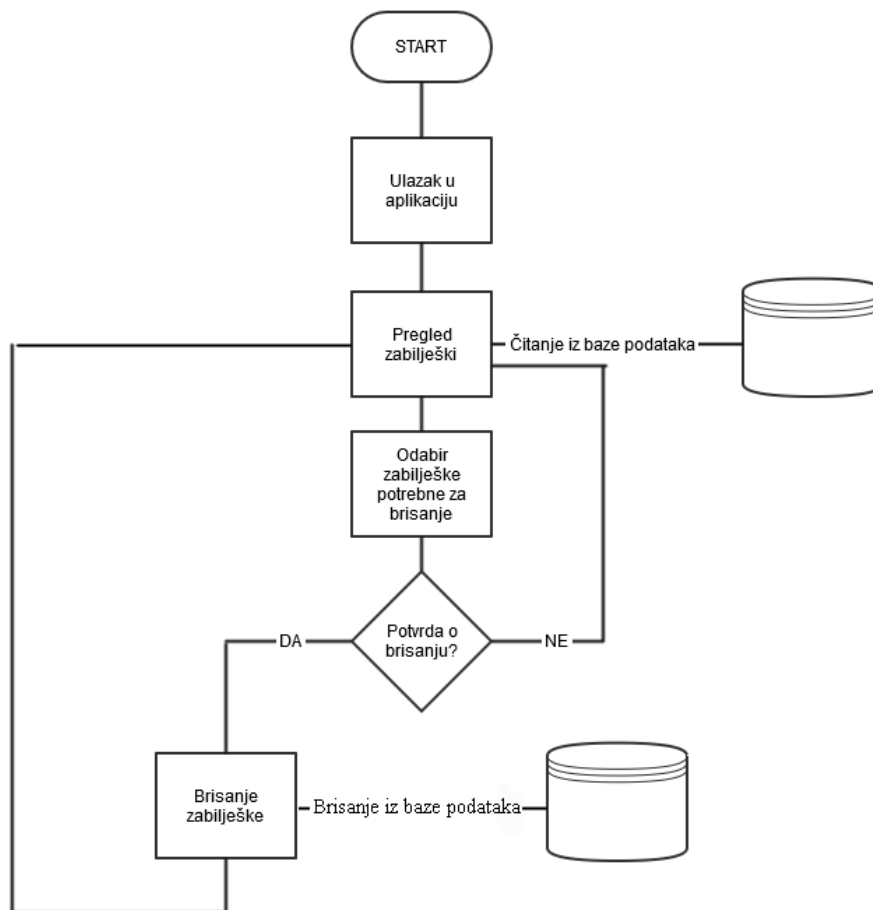
Slika 7 Dijagram toka čitanja zabilježki

Dijagram toka za upisivanje novih zabilješki je vidljiv na slici 8. gdje korisnik ulazi u aplikaciju te mu se pritiskom na gumb „Add a new note“ otvara forma za dodavanje nove zabilješke, nakon što korisnik unese tekst zabilješke i datum te odabere boju, potvrdom se zabilješka upisuje u bazu podataka.



Slika 8 Dijagram toka upisivanja novih zabilješki

Dijagram toka za brisanje zabilješki je prikazan na slici 9. Kada korisnik uđe u aplikaciju, prikažu mu se zabilješke. Prilikom klika na zabilješku, nudi mu se mogućnost brisanja zabilješke te ukoliko korisnik potvrdi da želi brisati zabilješku, zabilješka se briše iz baze podataka.



Slika 9 Dijagram toka brisanja zabilješki

3.2. Opis izrade aplikacije

3.2.1 Izrada HTML dokumenta

Kao što je rečeno u prethodnim poglavljima, sama aplikacija je napravljena uz pomoć HTML-a, JavaScripta te SQLite-a pomoću kojeg je kreirana baza podataka u koju se spremaju zabilješke.

HTML dokument je vrlo jednostavan, te se sastoji od četiri *div* spremnika. U prvom *div-u* se nalaze gumbi koji su vidljivi pri samom ulasku u aplikaciju.

```

<div id="main">
  <button id="button_note" onclick="show_note()">Notes </button></a>
  <button id="button_new" onclick="new_note()">Add a new note
</button></a>
</div>

```

Ispis kôda 5. Prvi div spremnik

Između prve i druge oznake *div* se nalazi polje koje služi za pretraživanje zabilješki. Prilikom unosa bilo kojeg znaka u polje, poziva se funkcija koja pretražuje zabilješke. Također, pored polja za pretraživanje se nalazi gumb za prikaz svih kategorija. Prilikom pritiska na gumb otvara se *overlay* gdje se ispisuju sve kategorije

```

<input type="text" id="search" placeholder="Search.."
oninput="search_note()" /><br/>
<a href="#overlay" id="open-overlay">    <input type="button" value="List
all categories" id="catlist" onclick="showcatlist()" ></input></a>

<div id="overlay">
  <a href="#" class="close">&times;</a>
  <div style="height:20%"></div>
  <ul id="container"> </ul>
</div>

```

Ispis kôda 6. Polje za pretraživanje zabilješki i prikaz kategorija

U drugom *div-u* se nalazi forma za dodavanje novih zabilješki, tj. za dodavanje teksta zabilješke i datuma zabilješke. Osim toga, može se i odabrati boja zabilješke te i kategorije u koju tu zabilješka spada. U ovom *div-u* se, također, nalazi gumb i forma za dodavanje nove kategorije te gumb koji potvrđuje dodavanje nove zabilješke, te se lista zabilješki automatski i ažurira.

```

<div id="notes_new">
  <form class="notes">
    Note: <input type="text" class="txtinput" id="txtinput" />    <input
type="checkbox" id="todo" name="vehicle" value="todo"> To - do list<br>
<br>
    Date: <input id="dateinput" class="dateinput" type="datetime-local"
><br>

    Note Color: <input type="color" class="colorinput" id="colorinput"
value="#F9EFAF" />
    Select Category:  <select id="select">    </select>
    <input type="button" value="Add new category"
onclick="showcat()" ></input>
  </form>
  <form id="new_cat" onblur="addcategory()" style="visibility:hidden;">

    <p id="cattekst" style="visibility:hidden;">New Category name:</p>
    <input type="text" id="catname" style="visibility:hidden;">  <br>

    <input type="button" id="catsubmit" style="visibility:hidden;"
class="buttonClass" value="submit" onclick="catend()" ></form>

    <input type="button" class="buttonClass" value="submit"
onclick="opendbs()" >
</div>

```

Ispis kôda 7. Drugi div spremnik

Posljednja dva *div-a* su ugniježdene, tj. jedan *div* se nalazi unutar drugog. Unutarnji *div* je prazan u HTML dokumentu jer se pomoću JavaScripte unutar njega ispisuju zabilješke.

```
<div class="segment" id="segment_note">
  <div id="output"> </div>
</div>
```

Ispis kôda 8. Spremnik za ispis zabilješki

3.2.2 Izrada i rad sa bazom podataka

Kako bi JavaScript na Androidu mogla pristupiti SQLite bazi potrebno je aplikaciji dodati dodatak koji to omogućava. U ovom slučaju je korišten BrodySoft SQLite Plugin. Dodatak omogućava da se unutar JavaScripte koriste već kreirane funkcije za rad sa SQLite bazama.

Unutar JavaScripta baza je kreirana pomoću funkcije koja je prikazana na ispisu kôda 9.

```
function.opendbs()
{
  var db = window.sqlitePlugin.openDatabase("dbnotes", "1.0", "Noter", -1);
  db.transaction(populateDB, errorCallback, successCB);
}
```

Ispis kôda 9. JavaScript funkcija za izradu baze podataka

Glavna funkcija unutar *JavaScripta* je *populateDB* koja služi kako bi se kreirale tablice unutar baze podataka te ista ta tablica napunila podacima. Funkcija *executeSql* kreira 3 nove tablice. Prva tablica je *Notes* koja se sastoji od 5 atributa, a to su id (integer), Note (text), Date (integer), Colour (text) i Checked (text). Druga je *Category*, koja se sastoji od minimalno tri kategorije, koje su već postavljene unutar aplikacije, a to su Private, Home i Work te od dodatnih kateogrija ukoliko ih korisnik sam doda. Tablica *Category* se sastoji od id (integer) i Category (text). Treća tablica je tablica *Notes_Category*, gdje su povezane tablice *Category* i *Notes*, odnosno primarni ključevi iz tih tablica.

```
function populateDB(tx) {
  tx.executeSql('CREATE TABLE IF NOT EXISTS Notes (id integer primary key,
  Note text, Date integer, Colour text, Checked text)');
  tx.executeSql('CREATE TABLE IF NOT EXISTS Category (id integer primary key,
  Category text)');
  tx.executeSql('CREATE TABLE IF NOT EXISTS Notes_Category (id integer
  primary key, id id_note, id id_category)');
  tx.executeSql('INSERT IGNORE INTO Category(Category) VALUES (?, ?, ?)',
  ['private', 'home', 'work']);
  var checkbox = document.getElementById("todo").checked;
  var note=document.getElementById('txtinput').value;
  var date=document.getElementById('dateinput').value;
  var colors=document.getElementById('colorinput').value;
  var select=document.getElementById('select').value;
  if(note=='')
  {
    queryDB(tx);
  }
  else{
    var category= tx.executeSql('SELECT Category from Category WHERE id
  like ?', [select]);
    var category_id= tx.executeSql('SELECT id from Category WHERE
  Category like ?', [category]);
```



```

        var note_id= tx.executeSql('SELECT id from Note WHERE Note like ?',
[note]);

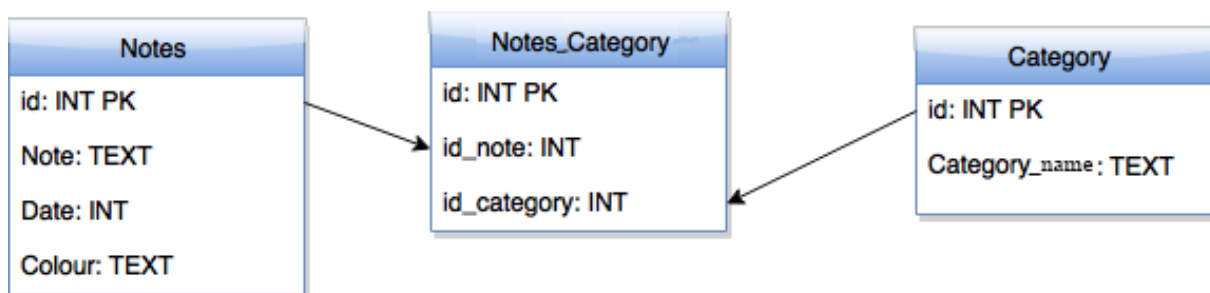
        tx.executeSql('INSERT INTO Notes (Note, Date, Colour, Checked) VALUES
(?,?,?,?)', [note, date, colors, checkbox]);
        tx.executeSql('INSERT INTO Notes_Category (id_note, id_category)
VALUES (?,?)', [note_id,category_id]);
        queryDB(tx);
    }

    document.getElementById('txtinput').value = "";
    document.getElementById('dateinput').value ="";
    document.getElementById('colorinput').value ="";
    document.getElementById("notes_new").style.visibility="hidden";
    document.getElementById("segment_note").style.visibility="visible";
}

```

Ispis kôda 10. JavaScript funkcija za kreiranje tablica

Sama baza podataka se sastoji od tri tablice. Tablice *Notes* u koju se upisuju zabilješke, a tablica se sastoji od *id*-a koji označava redni broj zabilješke, *Note*-a u koji se sprema zabilješka, *Date*-a u koji se sprema datum upisivanja zabilješke u bazu te *Colour*-a u koji se sprema odabrana boja zabilješke. Sastoji se i od tablice *Category* koja se sastoji od *id*-a kategorije te samog naziva kategorije, *Category_name*. Treća tablica je tablica *Notes_Category* u koju se sprema *id* zabilješke te *id* kategorije.



Slika 10 Izgled baze podataka

Nakon dohvaćanja podataka unesenih u formu te ukoliko je polje gdje se zabilješka unosi prazno, znači da korisnik nije unio nikakvu zabilješku te se poziva funkcija *queryDB* koja dohvaća sve zabilješke iz baze podataka. Ukoliko je korisnik aplikacije unio nekakvu zabilješku, tada se prvo ta zabilješka pomoću funkcije *executeSql* upisuje u ranije kreiranu tablicu. Nakon spremanja zabilješke u tablicu, zatvara se forma za upis nove zabilješke te se prikazuje lista svih zabilješki.

```

function queryDB(tx) {
    tx.executeSql("SELECT id, Note, Date, Colour, Category, Checked from
Notes;", [], querySuccess, errorCallback);
}

```

Ispis kôda 11. JavaScript funkcija za dohvaćanje zabilješki

3.2.3 Dodavanje, brisanje i pretraživanje zabilješki

Funkcija *queryDB* dohvaća sve zabilješke iz baze podataka, odnosno tablice Notes te poziva funkciju *querySuccess* koja te zabilješke ispisuje na ekran. Nakon ispisivanja na ekran, funkcija poziva funkciju *scheduledelayed* koji sprema te i aktivira datum i vrijeme prikazivanja obavijesti zabilješke što će biti detaljnije objašnjeno u poglavlju 3.2.5.

```
function querySuccess(tx, results) {

var slika = "<img src=close.png>";
var len = results.rows.length;
document.getElementById("output").innerHTML="";
for (var i = 0; i < len; i++)
{
var x = results.rows.item(i).id;
if(results.rows.item(i).Checked === "true")
{
document.getElementById("output").innerHTML += "<table id='newTasks'
class='newTasks' onmousedown='deleterow(this)'; ><tr
id='editable'><td><li>" +results.rows.item(i).Note + "</td><td></td><td>"
+ results.rows.item(i).Date + "</li></td><td id='newTasks' >" + slika +
"</td></tr></table>";
}
else{
document.getElementById("output").innerHTML += "<table id='newTasks'
class='newTasks' onmousedown='deleterow(this)'; ><tr id='editable' ><td>"
+results.rows.item(i).Note + "</td><td></td><td>" +
results.rows.item(i).Date + "</td><td id='newTasks' >" + slika +
"</td></tr></table>";
}

document.getElementById('newTasks').setAttribute('id',x);
document.getElementById(x).style.background = results.rows.item(i).Colour;
var time = results.rows.item(i).Date;
var category = tx.executeSql("SELECT id_note, id_category from
Notes_Category WHERE id_category like?;"[results.rows.item(i).Category;],
querySuccess, errorCallback)var notetxt = results.rows.item(i).Note;
var d = new Date();
var n = d.getTime();

if (time == null || time == undefined || time == "")
{
continue;
}
else{
var da = new Date(time);
var na = da.valueOf();
if (n < na)
{
scheduleDelayed(x,time,category,notetxt);
}
}
}
}
```

} Ispis kôda 12. JavaScript funkcija za ispis zabilješki

Funkcija *querySuccess* ispisuje sve zabilješke dohvaćene iz baze podataka na ekran tj. od dohvaćenih zabilješki kreira tablicu unutar HTML-a. To je učinjeno pomoću for petlje koja za svaku zabilješku kreira novi redak te ih tako ispisuje. Također je unutar dodavanja novog reda omogućeno da se klikne na jedan redak, odnosno zabilješku, te se poziva funkcija *deleterow* koja poziva funkciju *deleterow*. To je učinjeno tako što je za *id* atribut elementa retka postavljen *id* iz baze podataka, dakle *id* retka neke zabilješke je isti kao i *id* te zabilješke unutar baze podataka.

```
function deleterow(x)
{
    var r = confirm("Are you sure you want to delete this note?");
    if (r == true) {
        var db = window.sqlitePlugin.openDatabase("dbnotes", "1.0", "Notes", -1);
        id1 = x.id;
        db.transaction(deletenote,errorCB, successCB);
    }
}
```

Ispis kôda 13. JavaScript funkcija za brisanje zabilješki

Korisnicima je, također, omogućeno dodavanje nove kategorije zabilješki, prilikom klika na gumb unutar forme za dodavanje nove zabilješke, korisniku se otvara forma za dodavanje nove kategorije.

3.2.4 Dodavanje i pretraživanje kategorija

```
function showcat()
{
    document.getElementById("notes_new").style.visibility="hidden";
    document.getElementById('catname').style.visibility="visible"
    document.getElementById('cattekst').style.visibility="visible"
    document.getElementById('catsubmit').style.visibility="visible"
    document.getElementById("new_cat").style.visibility="visible";
    addcategory();
}
```

Ispis kôda 14. Prikaz forme za dodavanje nove kategorije

Prilikom klika na gumb za potvrdu dodavanja nove kategorije, poziva se funkcija *addcategory*, koja nakon što otvori bazu podataka za pisanje, poziva funkciju *categoryDB* koja upisuje novu kategoriju u tablicu *Category*.

```
function addcategory()
{
    var db = window.sqlitePlugin.openDatabase("dbnotes", "1.0", "Notes", -1);
    db.transaction(categoryDB, errorCallback, successCB);
}

function categoryDB(tx)
{
    var selectcat=document.getElementById('catname').value;
    if(selectcat == "")
    {
        queryCat(tx);
    }
}
```

```

        else{
            tx.executeSql('INSERT INTO Category (category_name) VALUES (?)',
[selectcat]);
            document.getElementById('catname').value = "";

            queryCat(tx);
        }
    }
}

```

Ispis kôda 15. JavaScript funkcija za dodavanje nove kategorije

Funkcija *categoryDB* poziva funkciju *queryCat* koja izvršava upit (eng. *query*) za ispis svih kategorija te ukoliko je izvršavanje upita uspješno, poziva se funkcija *catSuccess*. Unutar *catSuccess* funkcije se pomoću *for* petlje učitavaju sve kategorije te se one dodaju u izbornik za odabir kategorija te u listu kategorija na početnom zaslonu. Također, nakon dodavanja se provjerava jesu li sve kategorije jedinstvene, tj. ukoliko ima kategorija sa istim nazivom, duple se odbacuju.

```

function queryCat(tx) {
    if(cnt ===true)
    {
        tx.executeSql("SELECT id, category_name from Category;", [], catSuccess,
errorCB);}

        else{

            tx.executeSql('INSERT INTO Category(category_name) VALUES (?)',
["Private"]);

            tx.executeSql('INSERT INTO Category(category_name) VALUES (?)',
["Home"]);

            tx.executeSql('INSERT INTO Category(category_name) VALUES (?)',
["Work"]);

            cnt = true;

            tx.executeSql("SELECT id, category_name from Category;", [],
catSuccess, errorCB);
        }
    }
}

```

Ispis kôda 16. JavaScript funkcija za ispis svih kategorija

```

function catSuccess(tx, results) {

    var len = results.rows.length;

    selectrows= document.getElementById("select");
    selectrows1= document.getElementById("selects");

    for (var i = 0; i < len; i++)

        {
            var y= document.getElementById("select");
            var z= document.getElementById("selects");

```

```

        var option = document.createElement("option");
        option.text = results.rows.item(i).category_name;
        y.add(option);
        z.add(option);
    }
    for (var j = 0; j < len; j++)
    {
        var jid = results.rows.item(j).id;

        document.getElementById("container").innerHTML += "<li id='liid'
onclick='deletecat(this);'>" + results.rows.item(j).category_name + "</li>"

        document.getElementById('liid').setAttribute('id',jid);
    }
    var usedNames = {};
    $("select > option").each(function () {
        if(usedNames[this.text]) {
            $(this).remove();
        }
        else {
            usedNames[this.text] = this.value;
        }
    });
}

```

Ispis kôda 17. JavaScript funkcija za dodavanje nove kategorije

Funkcija *deleterow* otvara prozor u kojem korisnik potvrđuje da želi ukloniti zabilješku, ukoliko korisnik potvrdi da želi poziva se funkcija *deletenote*. Također, dohvaća se *id* od retka koji je potrebno izbrisati.

```

function deletenote(tx)
{
    tx.executeSql('DELETE FROM Notes WHERE id = ?', [id1]);
    window.location.reload(true);
    queryDB(tx);
}

```

Ispis kôda 18. JavaScript funkcija za brisanje zabilješki

Funkcija *deletenote* briše zabilješku iz baze podataka te se nakon toga osvježava aplikacija kako bi se uklonio i redak iz tablice.

Funkcija koja se poziva prilikom upisivanja teksta u polje za pretraživanje je *search_note*. Ta funkcija poziva funkciju *search* koja po unesenom tekstu pretražuje bazu podataka te ispisuje samo one zabilješke u kojima se nalazi uneseni tekst.

```

function search_note()

```

```

{
    var db = window.sqlitePlugin.openDatabase("dbnotes", "1.0", "Notes", -1);
    db.transaction(search,errorCB, successCB);
}

```

Ispis kôda 19 JavaScript funkcija za pretraživanje zabilješki

```

function search(tx)
{
    search_text = document.getElementById('search').value;
    tx.executeSql("SELECT id, Note, Date, Colour from Notes WHERE Note like
?;", [search_text], querySuccess, errorCB);
    if(!search_text){
        tx.executeSql("SELECT id, Note, Date, Colour from Notes;", [],
querySuccess, errorCB);
    }}

```

Ispis kôda 20. JavaScript funkcija za pretraživanje zabilješki

Također je moguće pretraživati po kategorijama. Na početnom zaslonu je moguće odabrati kategorije te će se prikazati sve zabilješke u odabranoj kategoriji. Funkcija prvo dohvaća *id* od kategorije koja je odabrana te se nakon toga kroz tablicu *Notes_Category* dohvaća *id* zabilješke koji je vezan uz tu kategoriju.

```

function select_note()
{
    var db = window.sqlitePlugin.openDatabase("dbnotes", "1.0", "Notes",
1);
    db.transaction(select,errorCB, successCB);
}

```

Ispis kôda 21. JavaScript funkcija za pretraživanje zabilješki po kategoriji

```

function select(tx)
{
    selections = document.getElementById('selects').value;
    if(selections=='All')
    {
        queryDB(tx);
    }
    else{
        var select_id = tx.executeSql("SELECT id, category_name from Category
WHERE category_name like ?;", [selections], querySuccess, errorCB);
        var category_id = tx.executeSql("SELECT id_note, id_category from
Notes_Category WHERE id_category like?;"[select_id], querySuccess, errorCB)

        tx.executeSql("SELECT id, Note, Date, Colour from Notes WHERE id like ?;",
[category_id], querySuccess, errorCB);    }
        tx = null;
    }
}

```

Ispis kôda 22. JavaScript funkcija za pretraživanje zabilješki po kategoriji

3.2.5 Dodavanje obavijesti zabilješke

Dodavanje obavijesti zabilješke omogućava korisniku da odabere datum i vrijeme kako bi mu se prikazala zabilješka u to odabrano vrijeme.

Obavijest prvo sprema kategoriju, tekst zabilješke i vrijeme prikazivanja obavijesti te kategoriju prikazuje u naslovu obavijesti, a tekst zabilješke se prikazuje ispod kategorije.

```

scheduleDelayed = function (x,time,category,notetxt) {
var d = new Date(time);
var n = d.valueOf();

document.getElementById("sched").innerHTML = n;

var sound = device.platform == 'Android' ?
'file://sound.mp3' : 'file://beep.caf';

cordova.plugins.notification.local.schedule({
    id: x,
    title: category,
    text: notetxt,
    at: n,
    sound: sound,
});
};

update = function () {
cordova.plugins.notification.local.update({
id: 1,
text: 'Updated Message 1',
json: { updated: true }
});
};

updateInterval = function () {
cordova.plugins.notification.local.update({
    id: 1,
    text: 'Updated Message 1',
    every: 'minute'
});
};

setDefaultTitle = function () {
cordova.plugins.notification.local.setDefaults({
    title: 'New Default Title'
});
};

document.addEventListener('deviceready', function () {

    cordova.plugins.notification.local.on('schedule', function
(notification) {

    });

}, false); app.initialize();

```

Ispis kôda 23. JavaScript funkcija za dodavanje obavijesti zabilješki

3.2.6 Dizajniranje aplikacije

Dizajn aplikacije obuhvaća oblikovanje izgleda za gumbove, izgleda formi za dodavanje novih zabilješka te izgleda samih zabilješki.

Izgled gumbova, vidljivih pri samom ulasku u aplikaciju, a koji služe za prikaz i dodavanje ovih zabilješki, određen je sa kôdom 24.

```
#button_note, #button_new{
```

```

width:10em;
height: 2em;
color: #000000;
opacity: 0.5;
font:3vw 'Gloria Hallelujah', cursive;
box-shadow: 10px 10px 5px #888888;
margin: 2px;
border-radius:10px;
background-color:#fff;
border:#878787 solid 1px;
}

```

Ispis kôda 24. Oblikovanje izgleda gumbova



Slika 11 Izgled gumbova

Izgled zabilješki je oblikovan sa sljedećim kôdom, gdje se definira veličina, širina zabilješki, kao i početna boja zabilješke te sami razmak između svake zabilješke.

```

.newTasks{
  display: inline-block;
  padding:30px 30px 40px;
  margin: 20px;
  width:10em;
  height:10em;
  font:20px 'Gloria Hallelujah', cursive;
  background: #F9EFAF;
  box-shadow: 10px 10px 15px #888888;
  transition:box-shadow 0.5s ease;
  -webkit-border-radius: 50px;
  -moz-border-radius: 50px;border-radius: 50px;
  max-width:520px;
  max-height:250px;
  color: #000000;
}

```

Ispis kôda 25. Oblikovanje izgleda zabilješki

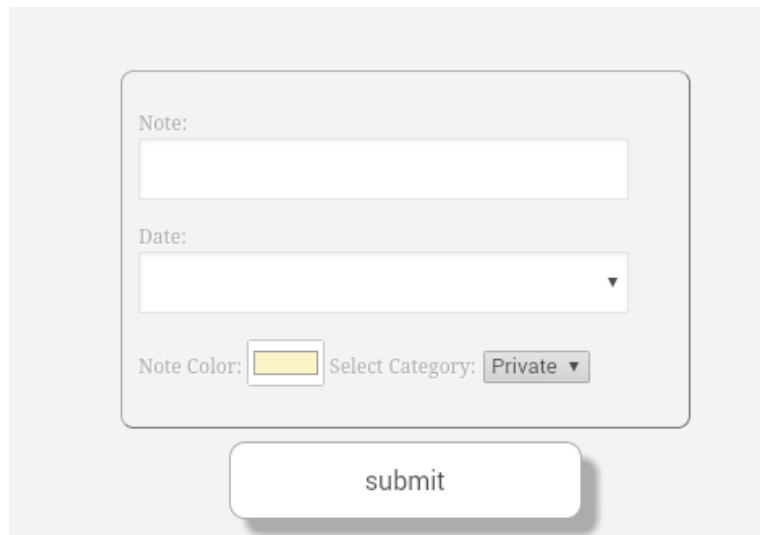


Slika 12 Izgled zabilješki

Izgled forme za dodavanje novih zabilješki je definiran pomoću ispisa kôda 26. te je sam izgled vidljiv na slici 13. U ispisu kôda 26. je definiran obrub forme, boja teksta te margine i udaljenost od ostalih elemenata.

```
.notes input[type="text"], .notes input[type="date"]{
  border: 1px solid #DADADA;
  color: #888;
  height: 30px;
  margin-bottom: 16px;
  margin-right: 6px;
  margin-top: 2px;
  outline: 0 none;
  padding: 3px 3px 3px 5px;
  width: 90%;
  font-size: 12px;
  line-height: 25px;
  box-shadow: inset 0px 1px 4px #ECECEC;
  -moz-box-shadow: inset 0px 1px 4px #ECECEC;
  -webkit-box-shadow: inset 0px 1px 4px #ECECEC;}
```

Ispis kôda 26. Oblikovanje izgleda forme

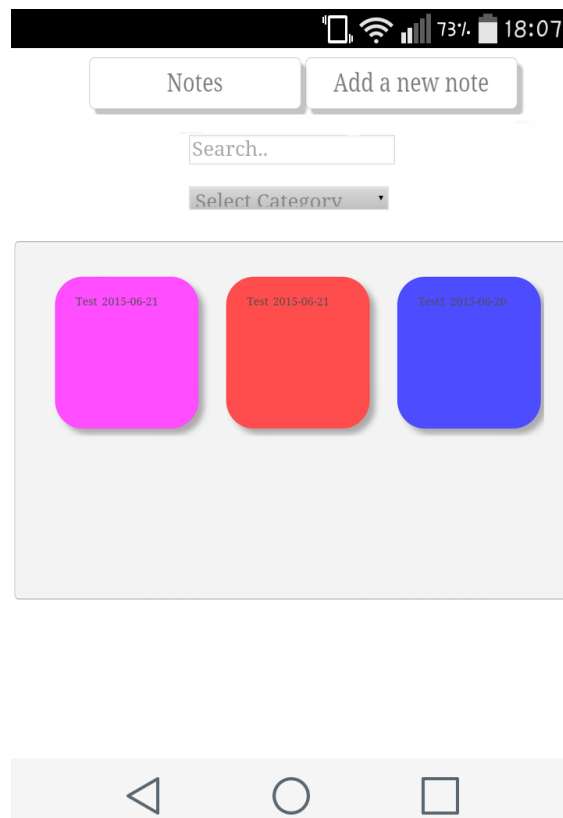


Slika 13 Izgled forme za dodavanje zabilješki

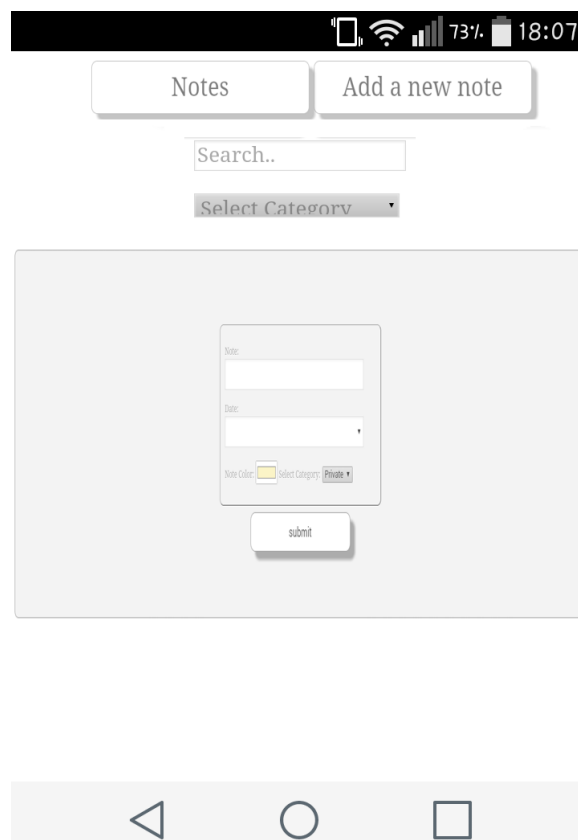
Kako bi se omogućilo korištenje aplikacije na uređajima različitih veličina ekrana potrebno je u HTML dokument dodati ispis kôda 27. Tim se kôdom definira *viewport* te se aplikacija, odnosno HTML dokument automatski oblikuje da bude podjednake veličine na svim ekranima. U ovom slučaju je postavljeno da aplikacija nije skalabilna, što znači da korisnik ne može približavati (eng. *zoom in*) odnosno udaljavati (eng. *zoom out*) sadržaj stranice, kao što je moguće na web stranicama preko web preglednika. Također je postavljeno da je visina, odnosno širina aplikacije jednaka veličini i širini ekrana uređaja. Nakon toga je potrebno u CSS definirati medija upite (eng. *media queries*), pomoću kojih se detaljno oblikuje HTML dokument tako da bude podjednak na svim veličinama zaslona. Medija upiti su CSS 3 modul koji omogućava renderiranje sadržaja kako bi se prilagodio različitim stanjima kao što je različita rezolucija ekrana ili broj bitova po boji koje može prikazati uređaj. Medija upiti se najčešće koriste kako bi web stranica bila postavljena tako da jednako izgleda na svim veličinama zaslona.

```
<meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width, height=device-height, target-densitydpi=device-dpi"/>
```

Ispis kôda 27. Oblikovanje izgleda forme



Slika 14 Prikaz zabilješki na zaslonu veličine 720x1280 px



Slika 15 Prikaz dodavanja novih zabilješki na zaslonu veličine 720x1280 px

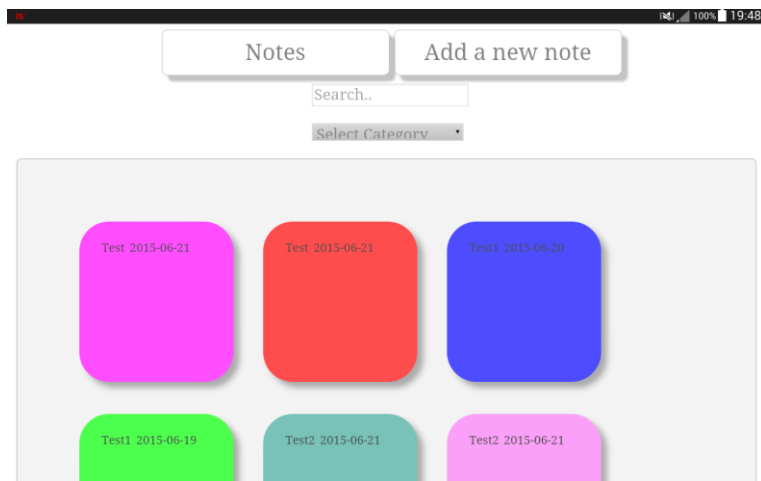
Kada je zaslon veličine 720x1280 px, kao u ovom slučaju, tada se aplikacija oblikuje po CSS-u iz ispisa kôda 28.

```
@media screen and (min-width:720px){  
  
    .notes{  
        margin-left:auto;  
        margin-right:auto;  
        max-width: 30%;  
        border:#878787 outset 1px;  
        border-radius:6px;  
        padding: 20px 10px 20px 5px;  
        font: 12px Georgia, "Times New Roman", Times, serif;  
        color: #888;  
        text-shadow: 1px 1px 1px #FFF;  
        position:relative;  
        margin-top:5px;  
  
        .newTasks{  
            display: inline-block;  
            padding:15px 15px 20px;  
            margin: 12px;  
            width:5em;  
            height:5em;  
            font:15px 'Gloria Hallelujah', cursive;  
            background: #F9EFAF;  
            box-shadow: 7px 7px 10px #888888;  
            transition:box-shadow 0.5s ease;  
            -webkit-border-radius: 50px;  
            -moz-border-radius: 50px;  
            border-radius: 50px;  
            max-width:300px;  
            max-height:150px;  
            color: #000000;  
            text-align:center;  
        }  
    }  
}
```

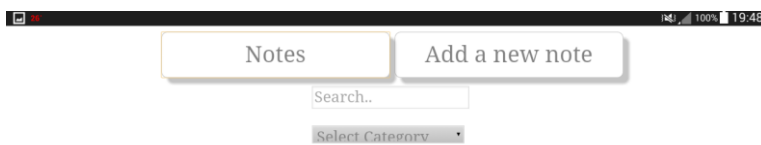
Ispis kôda 28. Primjer medija upita

3.3 Prikaz aplikacije

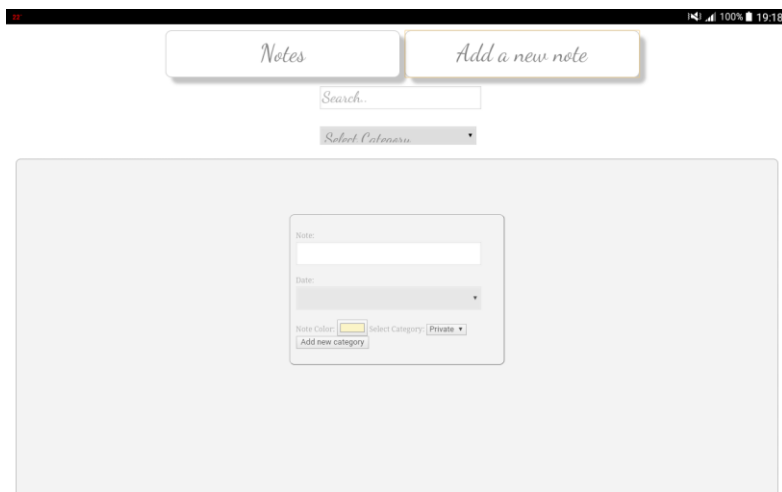
Pri samom ulasku u aplikaciju su vidljiva dva gumba, i to jedan za prikazivanje, odnosno „sakrivanje“ liste zabilješki, koje se automatski prikazuju pri pokretanju aplikacije te drugi za prikazivanje i „skrivanje“ forme za dodavanje nove zabilješke, te je to vidljivo na slikama 16, 17 i 18.



Slika 16 Ulazak u aplikaciju

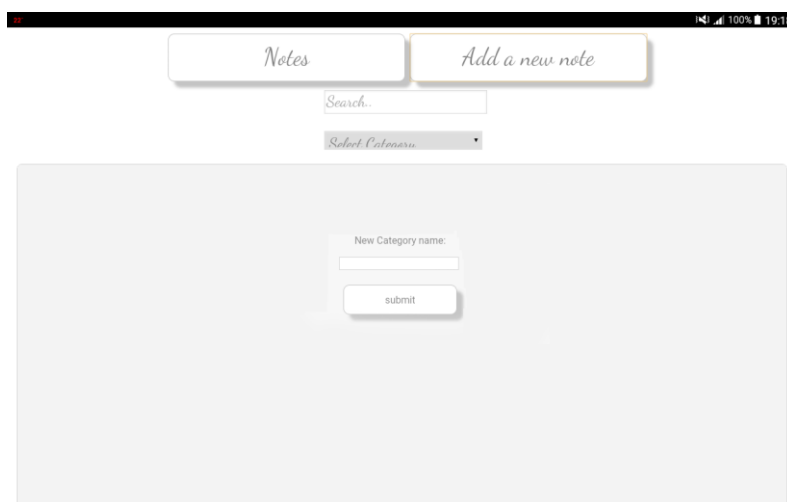


Slika 17 "Sakrivanje" zabilješki



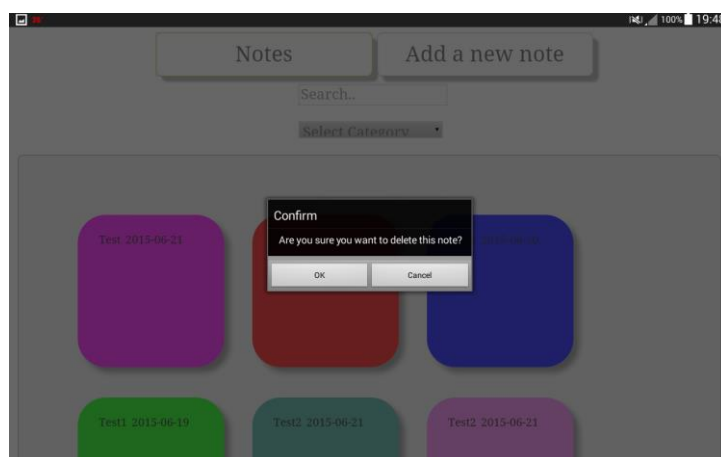
Slika 18 Dodavanje nove zabilješke

Prilikom dodavanja je moguće dodati i novu kategoriju, prilikom klika na gumb *Add new category*, prikazuje se forma za dodavanje kategorije.



Slika 19 Dodavanje nove kategorije

Nakon dodavanja nove zabilješke vidljiva je lista svih zabilješki. Prilikom pritiska na zabilješku, moguće je samu zabilješku i izbrisati, što je vidljivo na slici 20 gdje se javlja poruka koju korisnik mora potvrditi ukoliko želi izbrisati zabilješku.

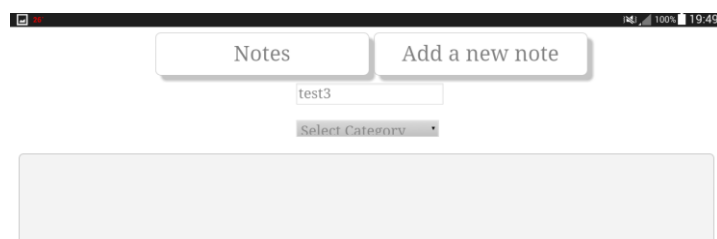


Slika 20 Brisanje zabilješke

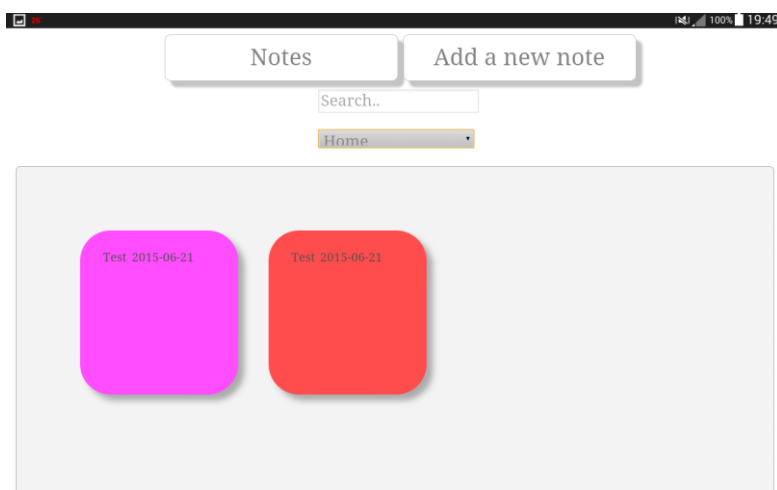
Aplikacija također omogućuje pretraživanje zabilješki po nazivu što je vidljivo na slikama 21 i 22 te pretraživanje po kategorijama što je vidljivo na slikama 23 i 24.



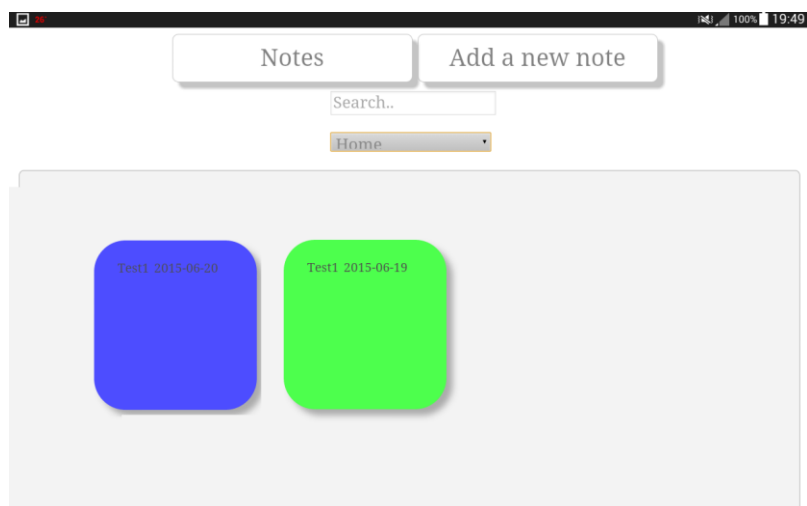
Slika 21 Pretraživanje zabilješki po nazivu



Slika 22 Pretraživanje zabilješki po nazivu



Slika 23 Pretraživanje zabilješki po kategoriji



Slika 24 Pretraživanje zabilješki po kategoriji

4. Zaključak

Osvrćući se na cjelokupni rad tj. aplikaciju koja se temelji na zapisu u notes, Android aplikacije uvelike olakšavaju svakodnevni život, posebice ako je riječ o aplikaciji čija funkcionalnost i namjena je ekvivalenta standardnim svakodnevnim aktivnostima, čime je korištenjem pripadne aplikacije ista aktivnost ubrzana i olakšana. Aplikacija omogućuje jednostavan zapis zabilježki, koje se spremaju u bazu podataka, te brisanje istih, na način da korisnik aplikacije mora pamtiti što manju količinu informacija te da se informacije više ne zapisuju na papiriće kao podsjetnici koje je vrlo lako izgubiti.

Glavna prednost aplikacije oblikovane za Android u HTML5 i JavaScriptu u odnosu na Android aplikaciju je ta što HTML aplikacija zauzima puno manje prostora, brže se pokreće, te isto tako je i sami rad u aplikaciji dosta brži. Kao što je i rečeno, jedina mana ovakve aplikacije je to što se ne može pristupati svim mogućnostima Android uređaja te je tako kompliciranije napraviti aplikaciju. Zbog toga se kod programiranja Android aplikacija u HTML-u koriste različita okruženja, u ovom slučaju Phonegap.

Android aplikacije su aplikacije drugačijeg značaja i funkcionalnosti nego aplikacije za računala. Mobilni uređaji rade na baterije i imaju slabije procesore, ali imaju jako velike mogućnosti i funkcionalnosti, te samim time aplikacije rađene za mobilne uređaje su aplikacije sa mnogo većim mogućnostima (aplikacije povezane s GPS-om, aplikacije vezane za kameru, aplikacije koje omogućuju direktno povezivanje na društvene mreže i još pregršt sličnih). Izrada mobilnih aplikacija, u ovom slučaju za Android operacijske sustave, nije jednostavan posao programskim stručnjacima jer se u obzir treba uzeti različita veličina zaslona mobilnih uređaja, drugačijih hardverskih specifikacija i mobilne programske podrške te promjena unutar platformi samih mobilnih uređaja.

5. Literatura

1. Tittel E., Noble J., *HTML, XHTML & CSS for Dummies*, Wiley, 2010.
2. West M., *HTML5 Foundations*, Wiley, 2012.
3. Harris A., *JavaScript and AJAX for Dummies*, Wiley, 2010.
4. Srinivas Sriparasa S., *JavaScript and JSON Essentials*, Packt Publishing, 2013.
5. Lunny A., *PhoneGap Beginner's Guid*, Packt Publishing, 2011.
6. Williams G., *Learn HTML5 and JavaScript for Android*, Apress, 2012.
7. Android platform guide, 15. travnja 2015.,
http://docs.phonegap.com/en/edge/guide_platforms_index.md.html