Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Omerzel

Mobilna interaktivna predstavitvena aplikacija

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Robert Rozman

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Večpredstavnost igra v našem vsakdanjem življenju vse pomembnejšo vlogo, še posebej na področju izobraževanja na daljavo s pomočjo interaktivne predstavitve izobraževalnih vsebin. Te uporabniku lahko predstavijo izbrano področje na strukturiran, interaktiven in posledično bolj zanimiv način. V okviru diplomske naloge razvijte mobilno aplikacijo, ki bo omogočala predstavitev tovrstnih vsebin uporabniku. Pri tem posebno pozornost posvetite možnosti, da se prikazana vsebina določi brez posebnega programerskega znanja in s tem omogoči tudi programersko manj veščim uporabnikom ustvarjalno oblikovanje predstavitvenih vsebin. Najprej analizirajte obstoječe rešitve in potem zasnujte ter implementirajte sistem v obliki mobilne interaktivne mobilne aplikacije.

Zahvaljujem se mentorju za pomoč pri izdelavi diplomske naloge in družini za vso podporo za časa študija. Posebna zahvala gre partnerki Evi, brez katere te diplomske naloge ne bi dokončal.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AoT	Ahead-of-Time	vnaprej
API	Application Programming Interface	aplikacijski programski vmesnik
ART	Android Runtime	okolje izvajanja za Android
CSS	Cascading Style Sheets	kaskadne stilske podloge
DRM	Digital Rights Management	upravljanje digitalnih pravic
EPUB	Electronic Publication	elektronska publikacija
FTP	File Transfer Protocol	protokol za prenos datotek
HAL	Hardware Abstraction Layer	aparaturna abstraktna plast
HTML	Hypertext Markup Language	jezik za označevanje nadbesedila
IDE	Integrated Development Environment	integrirano razvojno okolje
IDPF	International Digital Publishing Forum	Mednarodno združenje za standarde v digitalni založniški industriji
JIT	Just-in-Time	ravno pravi čas
MathML	Mathematical Markup Language	matematični označevalni jezik
OEBPS	Open eBook Publication Structure	publikacijska struktura odprte e-knjige
PDA	Personal Digital Assistant	osebni digitalni asistent ali dlančnik
PDF	Portable Document Format	prenosni format dokumentov

RSS	Rich Site Summary	zgoščeni povzetek strani
SDK	Software Development Kit	orodja za razvoj programske opreme
SFTP	Secure File Transfer Protocol	varni protokol za prenos datotek
SVN	Subversion	sistem za upravljanje z izvorno kodo
URI	Uniform Resource Identifier	enotni identifikator virov
WebDAV	Web Distributed Authoring and Versioning	protokol za spletno porazdeljeno avtorstvo in različice
XML	Extensible Markup Language	razširljiv označevalni jezik
XSD	XML Schema Definition	shematska definicija XML

Povzetek

Naslov: Mobilna interaktivna predstavitvena aplikacija

Avtor: Jernej Omerzel

V diplomskem delu je predstavljena aplikacija, ki se lahko uporablja na mobilnih napravah z operacijskim sistemom Android in je namenjena predvsem za uporabo v izobraževalnem procesu. Aplikacija namreč poleg običajnih omogoča tudi posamezne funkcije, ki jih pri obstoječih formatih e-knjig, katerim je sicer podobna, ne najdemo (npr. interaktivnost in možnost prikaza oddaljenih vsebin).

V diplomskem delu so najprej predstavljeni različni formati e-knjig in primerjava aplikacije s formatom EPUB. V nadaljevanju je opisan operacijski sistem Android in zgradba ter lastnosti aplikacij Android. Opisana so tudi razvojna orodja, s pomočjo katerih smo programirali aplikacijo.

Drugi del diplomskega dela je namenjen predstavitvi razvoja aplikacije. Opisane so zahteve, ki smo jih pred programiranjem aplikacije postavili, načrt zgradbe aplikacije in posamezne faze razvoja oziroma t.i. razredi, iz katerih je zgrajena aplikacija. Prav tako so predstavljeni primeri uporabe funkcij aplikacije, tj. v kakšnih oblikah lahko v aplikaciji prikažemo učno vsebino. Na koncu je prikazano še, kako poteka ustvarjanje konkretne vsebine aplikacije. V sklepu so nakazane še možne nadgradnje aplikacije.

Ključne besede: mobilna aplikacija, interaktivna aplikacija, Android, aplikacija Android.

Abstract

Title: Mobile Interactive Presentation Application

Author: Jernej Omerzel

The subject of diploma thesis is an Android application, which is mostly used on mobile phones and is intended to be used in the educational process. The application functions are similar to those of e-books, with the addition of interactivity and ability to display remote content.

In the diploma thesis, we first present different e-book formats and the functional comparison of the developed application with the well-known EPUB format. In addition, we describe the Android operating system and the structure of Android applications. We also describe tools, which we used for the application development.

In the second part of diploma thesis, development of the application is presented. We describe specifications, which we set before programming the application, the development plan and selected steps in the development process, i.e. classes that compose the application. Furthermore, we present how certain functions of the application can be used, i.e. in which form the content can be presented in the application. In the end, we describe the process of creating the content of the application. In the final chapter, we elaborate on possible upgrades of the application.

Keywords: mobile application, interactive application, Android, Android application .

Poglavje 1

Uvod

Dandanes ima skoraj vsakdo v svojem žepu pametni mobilni telefon. Le-ta med drugim omogoča prebiranje različnih vsebin, kot so na primer spletne strani, *zgoščeni povzetki strani (ang. Rich Site Summary, RSS)* in e-knjige. Vendar pa na mobilnih telefonih ni zaslediti aplikacije, ki bi bralcu predstavila tako vsebino za branje kot tudi interaktivno vsebino. Ker menimo, da bi navedena možnost lahko koristila izobraževalnemu procesu, smo se odločili, da takšno aplikacijo (v nadaljevanju: mobilna interaktivna predstavitvena aplikacija ali krajše aplikacija) razvijemo v diplomskem delu.

Aplikacija je po svoji zasnovi podobna e-knjigam, ki so zapisane v različnih formatih, npr. v *elektronski publikaciji (ang. Electronic Publication, EPUB)*, Kindlu, LITu, *prenosnemu formatu dokumentov (ang. Portable Document Format, PDF)* in Mobipocketu. Tudi ti formati e-knjig namreč omogočajo prikaz učne vsebine na mobilnih napravah, vendar ne podpirajo posameznih funkcij, s katerimi bi bilo po našem mnenju mogoče izobraževalni proces optimizirati in ga bolj približati uporabniku. Nobeden izmed opisanih formatov e-knjig tako npr. ne omogoča prikaza oddaljenih vsebin, s čimer bi lahko omogočili hitro spremembo učne vsebine na mobilnem telefonu uporabnika, ne da bi uporabnik sam za to moral kar koli storiti. Zato smo pri razvoju aplikacije posebno pozornost namenili funkcijam, ki jih obstoječi formati e-knjig ne podpirajo, bi pa pripomogle k bolj dinamičnem in uporab-

niku prijaznem izobraževalnem procesu. Poleg tega smo tudi ostale lastnosti aplikacije zastavili na način, da bi le-ta omogočala kar najboljši prikaz učne vsebine na mobilnih napravah. Med drugim smo pri načrtovanju aplikacije postavili cilj, da naj le-ta omogoča več oblik podajanja izobraževalnih vsebin (tekstovne, zvočne in vizualne) ter podpira interaktivne teste znanja. Rezultat praktičnega dela diplomskega dela je tako aplikacija, ki po svoji naravi oziroma zgradbi spada med e-knjige, vendar je dodatno obogatena z zvočno in vizualno vsebino ter poleg tega omogoča še prikaz vsebine JavaScript in prikaz oddaljenih vsebin.

Opisni (teoretični) del diplomskega dela je sestavljen iz več poglavij z naslednjo vsebino. V podpoglavju 1.1 najprej podajamo definicijo e-knjige in delitev e-knjig glede na različne dejavnike, v podpoglavju 1.2 pa smo pojasnili, katere funkcije pogrešamo pri obstoječih formatih e-knjig. V istem poglavju smo opisali še bistvene lastnosti aplikacije in nakazali, katere funkcije, ki jih pri obstoječih formatih e-knjig nismo zasledili, omogoča aplikacija. V drugem poglavju so nato podrobneje predstavljene lastnosti, prednosti in pomanjkljivosti obstoječih formatov e-knjig ter primerjava naše aplikacije z najbolj razširjenim formatom e-knjig, tj. EPUB. V tretjem poglavju je opisan operacijski sistem Android, osnovna zgradba tovrstnih aplikacij in razvojna orodja, s pomočjo katerih smo programirali aplikacijo. Poudarek je na razvojnih orodjih Android Studio in Android SDK, saj smo ti orodji pri programiranju aplikacije največ uporabljali. Razvoj in zgradba aplikacije sta nadalje predstavljena v četrtem poglavju: pri programiranju aplikacije smo najprej specificirali zahteve, ki naj bi jih aplikacija izpolnjevala (podpoglavje 4.1), nato smo naredili načrt zgradbe aplikacije (podpoglavje 4.2) ter se na koncu lotili samega programiranja aplikacije. Zgradba aplikacije je po posameznih sklopih podrobno opisana v podpoglavju 4.3. Aplikacija, kot pojasnjeno zgoraj, omogoča predstavitev vsebine v različnih oblikah, in sicer kot besedilo, zvok, slike, vizualne (video) vsebine in vsebine JavaScript. Posamezne oblike prikaza vsebin oziroma t.i. prezentacijskih elementov so predstavljene v petem poglavju. V šestem poglavju smo poleg tega opisali

še, kako poteka praktični postopek tvorbe prezentacijske vsebine, ki bo prikazana z aplikacijo. V sklepu smo na kratko analizirali, v kolikšni meri smo izpolnili zastavljene cilje, tj. v kolikšni meri aplikacija izpolnjuje zahteve, ki smo jih postavili v načrtu, ter nakazali možnost izboljšav oziroma nadgradenj aplikacije.

1.1 Definicija e-knjige

Obstoječa literatura podaja več definicij termina e-knjiga. V Mednarodni federaciji knjižničnih združenj in institucij pojasnjujejo, da je e-knjiga digitalna verzija tekstovnega dela, ki je javno dostopna, kot ločeno delo [1]. M. Fidler in T. Fidler v strokovnem članku podajata naslednjo definicijo: "Elektronska knjiga (e-knjiga, e-book) je v elektronski obliki shranjena knjiga, lahko tudi kateri koli drug dokument"[2]. S. K. Zadravec, T. Buzina in D. Seiter-Šverko so v svojem delu opozorile na različne definicije e-knjige in s tem na delitev glede na različne dejavnike, in sicer delitev glede na [3]:

- **Način nastanka**
  - digitalizirana e-knjiga (e-knjiga, pretvorjena iz fizične oblike)
  - pristna e-knjiga (že v svoji izvorni obliki zapisana kot e-knjiga)
  - e-knjiga v aplikacijah
- **Medij izdaje**
  - e-knjige dostopne na optičnih nosilcih
  - e-knjige prosto dostopne na internetu
  - e-knjige dostopne na internetu proti plačilu
- **Vsebino**
  - zvočne knjige
  - e-knjige, obogatene z zvočno ali vizualno vsebino

  - tekstovne e-knjige

- **Bralno napravo**
  - prirejene za branje na osebnem računalniku
  - prirejene za branje na prenosnih bralnih napravah

- **Upravljanje z digitalnimi pravicami**
  - prosto dostopne, brez *upravljanja digitalnih pravic (ang. Digital Rights Management, DRM)*
  - komercialno dostopne e-knjige, podvržene upravljanju digitalnih pravic

1.2 Obogatitev koncepta e-knjige

Cilj diplomskega dela je programirati aplikacijo, ki bi bralcu predstavila tako vsebino za branje kot tudi interaktivno vsebino. Takšna aplikacija bi bila namreč lahko, kot je bilo to pojasnjeno v uvodu, uporabna predvsem v izobraževalnem procesu, saj bi z njeno pomočjo učitelji in profesorji učencem ter študentom učno vsebino lahko prikazali v njim prijaznem formatu - aplikaciji za mobilne naprave, običajno pametne telefone.

Takšen prikaz učne vsebine sicer deloma že omogočajo e-knjige, vendar slednje ne podpirajo posameznih funkcij, s katerimi bi lahko po našem mnenju izobraževalni proces izboljšali. To bi bilo med drugim mogoče storiti tako, da je v aplikaciji mogoče hitro spremeniti oziroma dopolniti učno vsebino in da ima urednik možnost preveriti, ali so uporabniki zahtevano vsebino razumeli. Slednjega namreč e-knjige ne omogočajo, prav tako tudi ne funkcije prikaza oddaljenih vsebin. Zato smo so odločili za razvoj aplikacije, s katero bi obstoječe formate e-knjig nadgradili.

Posamezne lastnosti, prednosti in pomanjkljivosti različnih formatov e-knjig bodo podrobneje predstavljene v naslednjem poglavju diplomskega

dela, opis osnovnih značilnosti aplikacije, ki smo jo programirali, pa zaradi boljšega razumevanja delovanja aplikacije podajamo v naslednjem odstavku.

Aplikacija po svoji zasnovi spada med e-knjige. Njene načrtovane bistvene značilnosti so naslednje:

- aplikacija omogoča več oblik podajanja izobraževalnih vsebin: vsebina je lahko tekstovna, zvočna ali vizualna (video) vsebina,
- aplikacija podpira interaktivne teste znanja, narejene v programskem jeziku JavaScript,
- aplikacija omogoča prikaz oddaljenih vsebin, ki omogočajo uredniku prikaz novih vsebin na napravi bralca, ne da bi le-ta moral kaj storiti. Tako lahko npr. slednji na mobilno napravo bralca - učenca postopoma dodaja izobraževalne vsebine, za katere želi, da se le-ta z njimi seznanja postopoma,
- uporabniški vmesnik menijev se lahko enostavno spreminja glede na želje urednika aplikacije.

Upoštevaje zgoraj opisane značilnosti lahko tako zasnovano aplikacijo označimo kot e-knjigo, ki je obogatena z zvočno in vizualno vsebino in poleg tega omogoča še vsebine JavaScript in pa predvsem funkcijo prikaza oddaljenih vsebin. Prav po slednji se namreč aplikacija razlikuje od obstoječih formatov e-knjig.

Poglavje 2

Zasnova aplikacije

V tem poglavju podajamo pregled nekaj najbolj znanih obstoječih formatov e-knjig, skupaj z njihovimi bistvenimi značilnostmi, v drugem delu pa primerjavo aplikacije z enim izmed bolj znanih formatov e-knjige, tj. EPUB.

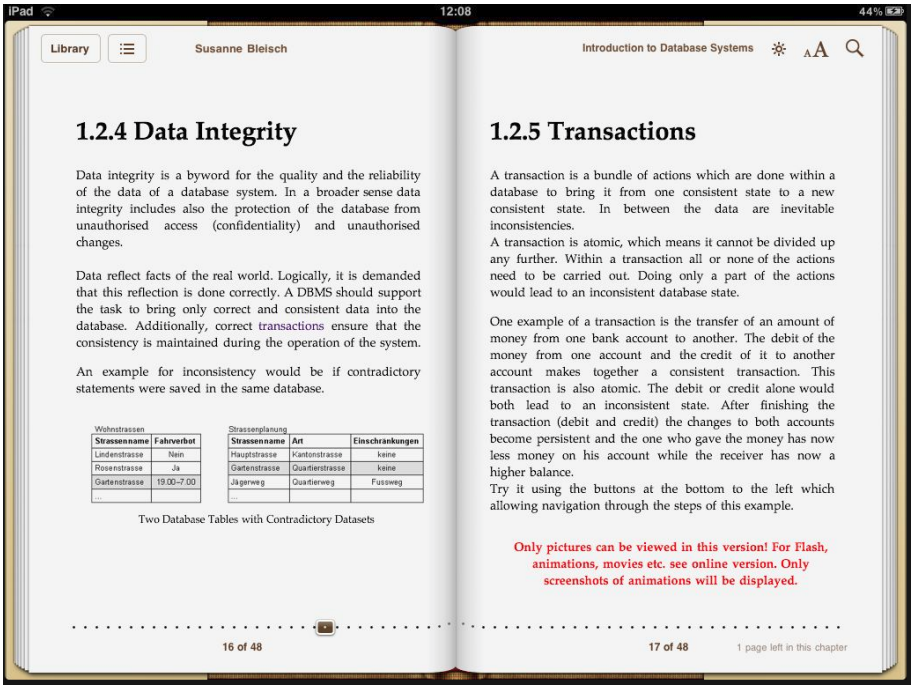
2.1 Pregled obstoječih formatov e-knjig

Dandanes na trgu obstoji veliko različnih formatov e-knjig. Njihova skupna značilnost je, da omogočajo predstavitev besedila in slikovnih vsebin. Nekateri izmed njih omogočajo tudi dodatne funkcije ali pa so narejeni posebej za določene naprave. V nadaljevanju podrobneje predstavljamo tiste formate e-knjig, ki so ali so bili v preteklosti tržno najbolj zanimivi in razširjeni. Poleg tega bomo te formate e-knjig med seboj primerjali po najpomembnejših lastnostih v obliki tabele.

Najbolj zanimivi in razširjeni formati e-knjig so po naši presoji naslednji:

- **EPUB** - je odprti format, ki ga je razvila in ga vzdržuje skupina "International Digital Publishing Forum". Osnovan je na *razširljivem označevalnem jeziku (ang. Extensible Markup Language, XML)* in *jeziku za označevanje nadbesedila verzije 5 (ang. Hypertext Markup Language, HTML5).* Značilnost tega formata je v tem, da celotno e-knjigo

EPUB predstavlja ena sama datoteka: v tej datoteki so vsebovani vsi potrebni sestavni elementi e-knjige [4].



Slika 2.1: Primer prikaza e-knjige v formatu EPUB

- **Kindle** - Amazonov format Kindle [5] je bil narejen ekskluzivno za Amazon Kindle bralno napravo. Gre za format, ki temelji na Mobipocket formatu (opisan spodaj) in uporablja visoko kompresijo. Aplikacije Kindle so bile narejene tudi za druge operacijske sisteme. Zato lahko datoteke s tem formatom beremo tudi na mobilnih telefonih (Android in iOS), osebnih računalnikih, tablicah in televizijskih predvajalnikih z operacijskim sistemom Android.

- **LIT** - je format e-knjig, ki je bil razvit posebej za aplikacijo Microsoft Reader. To je aplikacija za e-knjige, ki je bila prvič izdana leta 2000. Microsoft Reader uporablja patentirano tehnologijo ClearType, ki izboljša kakovost prikaza e-knjige. Le-to formatu ni pomagalo pri

popularnosti, zato je Microsoft leta 2011 prekinil razvoj in uporabo tega formata [6].

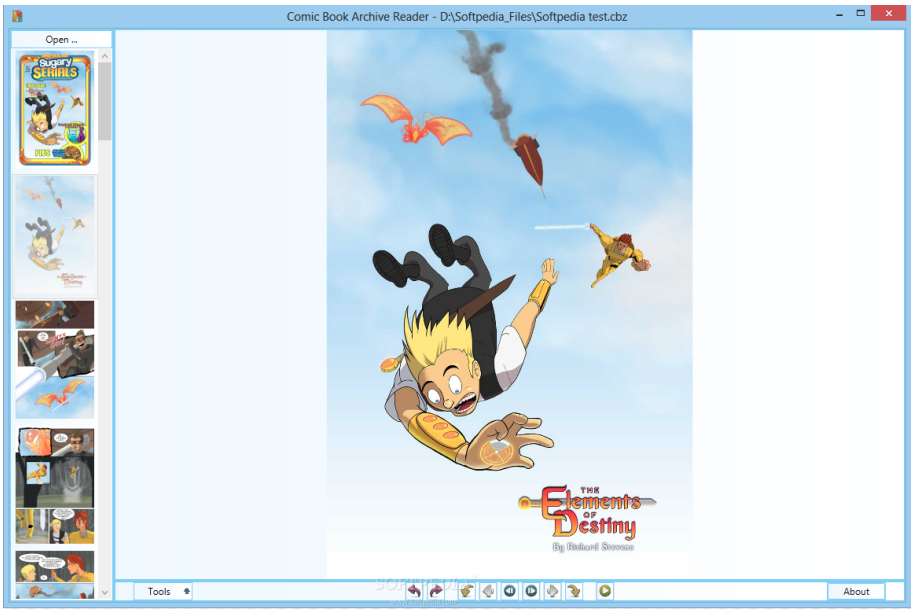
Slika 2.2: Primer LIT e-knjige

- **PDF** - je široko popularen format e-knjige, saj je standardni bralni dokument na osebnih računalnikih. Ustrezna aplikacija za branje obstaja na praktično vseh platformah. Slabost PDF formata je v tem, da njegova vsebina temelji na A4 velikosti, zaradi česar je težko berljiva na malih ekranih mobilnih telefonov in manjših tablicah.

- **Mobipocket** je format e-knjige, ki se uporablja na *dlančnikih (ang. Personal Digital Assistant, PDA)* in drugih pametnih napravah (npr. BlackBerry, iOS). Originalna programska oprema za branje tega formata je Mobipocket Reader, sicer pa so možne tudi Stanza, FBReader, Kindle za osebni računalnik in STDU viewer. Slabost formata je neenotno prikazovanje na različnih bralnikih. Mobipocket format podpira upravljanje digitalnih pravic - DRM [6].

- **Comic Book Archive file** je arhivski tip formata, ustvarjen z namenom sekvenčnega branja stripov, shranjenih v slikah. Sestoji iz slik,

tipično formata PNG ali JPEG [7]. Primer prikaza v tem formatu je na sliki 2.3.



Slika 2.3: Primer Comic Book Archive e-knjige

V tabeli 2.1 primerjamo večino obstoječih formatov e-knjig glede na naslednje lastnosti: možnost prikaza slik, možnost prikaza tabel, podpora zvočnim vsebinam, interaktivnost, podpora Word Wrapu in vizualna podpora. Primerjava pokaže, da so med najbolj dovršenimi formati Kindle, EPUB in iBooks, saj vsebujejo vse prej omenjene lastnosti oziroma funkcije. Sledi format PDF, ki podpira vse funkcije razen Word Wrapa. Tudi nekateri drugi formati imajo večino teh lastnosti (npr. DOCX in HTML), vendar pa se primarno ne uporabljajo kot e-knjige, zato jim v diplomskem delu nismo posvetili posebne pozornosti.

Format	Končnica datoteke	Slike	Tabele	Zvok	Interaktivnost	Podpora Word Wrap	Vizualna podpora
Comic Book Archive	.cbr, .cbz, .cb7	Da	Ne	Ne	Ne	Ne	Ne
DjVu	.djvu	Da	Da	Ne	Ne	Ne	?
DOC	.doc	Da	Da	?	?	Da	?
DOCX	.docx	Da	Da	Da	?	Da	Da
EPUB	.epub	Da	Da	Da	Da	Da	Da
Fiction Book	.fb2	Da	Da	Ne	Ne	Da	?
HTML	.html	Da	Da	Da	Ne	Da	Da
iBooks	.ibook	Da	Da	Da	Da	Da	Da
INF	.inf	Da	Da	Ne	?	Da	Ne
Kindle	.azw	Da	Da	Da	Da	Da	Da
Microsoft Reader	.lit	Da	?	Ne	Ne	Da	?
Mobi pocket	.prc, .mobi	Da	Da	Ne	Da	Da	?
eReader	.pdb	Da	?	Ne	Ne	Da	?
Plain text	.txt	Ne	Ne	Ne	Ne	Da	Ne
Plucker	.azw	Da	Da	Ne	Da	Da	?
PDF	.pdf	Da	Da	Da	Da	Ne	Da
PostScript	.ps	Da	?	Ne	Ne	Ne	?
Tome Raider	.tr2, .tr3	Da	?	Ne	Ne	Da	?
Open XPS	.oxps, .xps	Da	Da	?	Ne	Ne	?

Tabela 2.1: Primerjava lastnosti e-knjig [6]

2.2 Primerjava s formatom EPUB

EPUB je format e-knjige, zapisan v datoteki z običajno končnico *.epub*. Primeren je za branje na večini naprav, kot so osebni računalniki, pametni mobilni telefoni, tablice in e-bralniki. Postal je uradni standard *Mednarodnega združenja za standarde v digitalni založniški industriji (ang. International Digital Publishing Forum, IDPF)*. Leta 2007 je nasledil format z imenom *"publikacijska struktura odprte e-knjige"(ang. Open eBook Publication Structure, OEPBS)* [4]. Funkcije so mu dodajali postopno. Trenutno je v verziji 3.1 (Jan 2017). Funkcijsko je zelo podoben naši mobilni interaktivni predstavitveni aplikaciji. Trenutno EPUB podpira naslednje funkcije:

- **prikaz HTML strani:** rasterske in vektorske slike, zvok, vizualna vsebina, interaktivnost, *kaskadne stilske podloge (ang. Cascading Style Sheets, CSS)* in meta podatki,
- **zaznamki strani**,
- **opombe**,
- **knjižnica** (hrani knjige in omogoča iskanje),
- ***matematični označevalni jezik (ang. Mathematical Markup Language, MathML)***,
- ***upravljanje digitalnih pravic (ang. Digital Rights Management, DRM)***.

Podobnosti in razlike med obema formatoma so prikazane v tabeli 2.2. Aplikacija in EPUB imata po eni strani veliko skupnih točk. Bistvene funkcije so namreč pri obeh formatih zelo podobne: HTML5 (zvočna, slikovna in vizualna vsebina), interaktivnost ter *Word Wrap*[1]. Aplikacija, tako kot

[1]Word wrap ali tako imenovano lomljenje vrstice je postopek preoblikovanja dela besedila v vrsticah, tako da se bo prilegalo na razpoložljivo širino strani, okna ali drugega prikazovalnega področja.

	Mobilna interaktivna predstavitvena aplikacija	EPUB
HTML5	Da	Da
Word Wrap	Da	Da
Upravljanje digitalnih pravic	Ne	Da
Zaznamki	Ne	Da
Knjižnica	Da	Da
Oddaljena vsebina	Da	Ne

Tabela 2.2: Primerjava mobilne interaktivne predstavitvene aplikacije s formatom EPUB

EPUB, nadalje omogoča več oblik podajanja izobraževalnih vsebin: vsebina je lahko tekstovna, zvočna ali vizualna (video) vsebina.

Formata pa se po drugi strani razlikujeta pri upravljanju digitalnih pravic, zaznamkih, knjižnici in oddaljeni vsebini. Lahko bi sicer rekli, da ima tudi aplikacija knjižnico, saj lahko v sistem menijev in strani spravimo zelo veliko "knjig"oziroma strani, ki niso omejene z velikostjo vsebine.

Funkcije formata EPUB nadalje poleg zgoraj navedenega obsegajo še optimiziranje teksta za določen ekran, zaznamke in izpis jedra. Vsega tega naša aplikacija nima, saj je bil razvojni čas omejen. Temeljne prednosti aplikacije v primerjavi z EPUB pa so naslednje:

- možnost poljubne določitve uporabniškega vmesnika menijev,
- možnost lokalnega ali spletnega shranjevanja strani oziroma vsebine,
- podpora interaktivnim testom znanja, ki so narejeni v programskem jeziku JavaScript,
- možnost prikaza oddaljenih vsebin.

Takšna primerjava značilnosti EPUB in aplikacije pokaže, da je EPUB funkcijsko veliko bolj dovršen, saj navsezadnje aplikacija uvaja čisto nov format in njegova nadaljnja dodelava zahteva svoj čas. Vendar ima tudi aplikacija določene prednosti, saj omogoča nekatere funkcije, ki jih EPUB ne podpira. Zato lahko aplikacija s pomočjo funkcije oddaljenih vsebin dopolnjuje EPUB format pri podajanju hitro spreminjajočega izobraževalnega gradiva in v primerih, ko uporabnik na strani potrebuje ali želi tudi podporo programskemu jeziku JavaScript.

Poglavje 3

Uporabljena orodja in tehnologije

Mobilno interaktivno predstavitveno aplikacijo smo programirali v *integriranem razvojnem okolju (ang. Integrated Development Environment, IDE)* Android Studio in z uporabo Android *orodja za razvoj programske opreme (ang. Software Development Kit, SDK)*. Vsebino aplikacije smo naredili v Oxygen XML Editorju (menu.xml in pripadajoča .xsd shema) in v Adobe Dreamweaverju (strani s prezentacijsko vsebino).

V naslednjih podpoglavjih bomo predstavili operacijski sistem Android, za katerega smo naredili aplikacijo, splošno zgradbo tovrstnih aplikacij in ustrezna razvojna orodja.

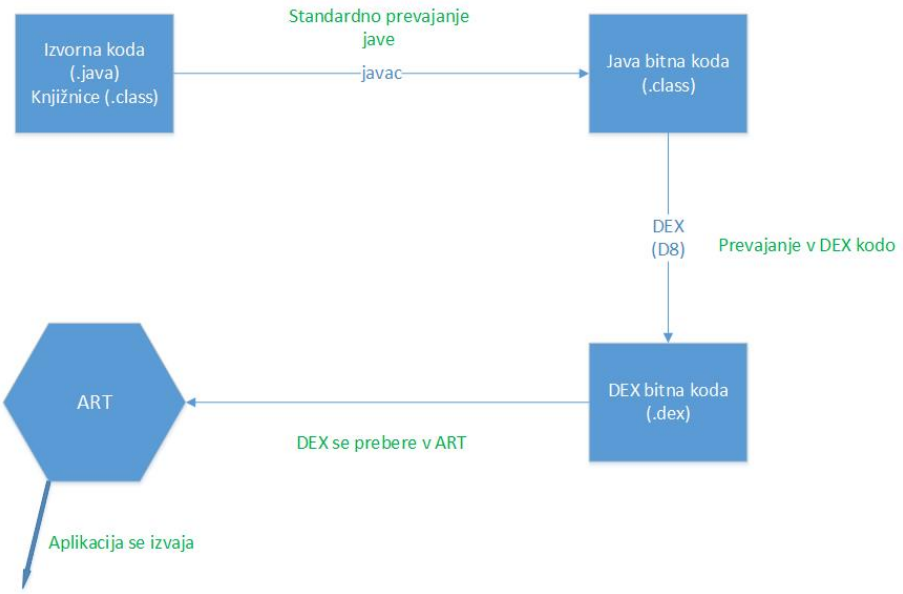
3.1 Operacijski sistem Android

Android je zelo priljubljen mobilni operacijski sistem, ki temelji na linuxovem jedru. Nameščen je na napravah številnih proizvajalcev, kot so npr. Samsung, HTC, LG, Sony Ericsson, Huawei in drugi. Verzija linuxovega jedra se razlikuje od naprave do naprave, najbolj pogosto uporabljeni verziji pa sta 3.18 in 4.4 [8].

Operacijski sistem Android ima številne prednosti. Največja med njimi

je ta, da je odprtokoden. S tem je močno olajšano razvijanje aplikacij. Navedena lastnost operacijskega sistema nam je bila v pomoč tudi pri razvoju naše aplikacije.

Prevajanje aplikacije za Android se izvede v več korakih. Prva faza je klasično prevajanje javanske kode s prevajalnikom "javac". V tem koraku izvorne datoteke združimo z javanskimi knjižnicami tako, da dobimo Java bitno kodo. Naslednji korak je prevajanje Java bitne kode s prevajalnikom "D8", tj. najnovejšo verzijo prevajalnikov tipa DEX, v DEX bitno kodo. Leta se nato prebere na navideznem stroju ART in prevede v strojno kodo tega navideznega stroja, nato pa se na njem izvrši. Koraki prevajanja aplikacije za Android so prikazani na sliki 3.1.



Slika 3.1: Koraki prevajanja aplikacije

Operacijski sistem Android je sestavljen iz več delov, ki so eden na drugega naloženi kot torta ali sklad (ang. stack). Ti deli so naslednji [9]:

- **Linux jedro** - skrbi za upravljanje vse strojne opreme mobilne naprave. Sem sodi procesor, pomnilnik, upravljanje z energijo, zvok, slika,

povezljivost (WiFi, Bluetooth, mobilna omrežja, USB), kamera in tipkovnica.

- ***aparaturna abstraktna plast (ang. Hardware Abstraction Layer, HAL)*** - je povezovalni element med strojno in programsko opremo. S pomočjo standardnih vmesnikov ali knjižnic omogoča komunikacijo med aplikacijo in strojno opremo na naslednji način: aplikacija uporabi višjenivojski Java *aplikacijski programski vmesnik (ang. Application Programming Interface, API)*, ta uporabi standardni vmesnik v HALu, ki nato komunicira s strojno opremo mobilne naprave.

- **Knjižnice C/C++** - posamezni deli operacijskega sistema Android so napisani v programskem jeziku C/C++, zato za svoje delovanje potrebujejo temu programskemu jeziku lastne knjižnice. V te knjižnice je mogoče dostopati preko Java API okvirja (ang. Java API Framework).

- ***okolje izvajanja za Android (ang. Android Runtime, ART)*** - je izvajalno okolje delovanja, ki ga uporablja Android. V izvajanje dobi .dex datoteke, ki so iz javanske bitne kode prevedene s prevajalnikom DEX. V verziji Android 5.0 Lollipop je ART nadomestil izvajalno okolje Dalvik, ki se ne razvija več. ART in Dalvik sta kompatibilni izvajalni okolji, kar pomeni, da lahko poganjata iste aplikacije. Bistvena razlika med njima je v načinu prevajanja programa. ART uporablja izvajanje *vnaprej (ang. Ahead-of-Time, AoT)*, Dalvik pa prevajanje *ravno pravi čas (ang. Just-in-Time, JIT)* [10], [11], [12].

- **Java API okvir** - je celota razredov in vmesnikov, ki jih potrebujemo za razvijanje aplikacij v programskem jeziku Java za Android. Vmesniki tega okvirja omogočajo, kot pojasnjeno zgoraj, dostop do strojne opreme naprave, razredi tega okvirja pa so osnovni gradniki aplikacije (npr. razreda Activity, Service in drugi). Zgradba aplikacij Android bo podrobneje predstavljena v podpoglavju 3.2.

- **sistemske aplikacije** - v sklopu operacijskega sistema Android nava-

dno najdemo še sistemske aplikacije, kot so npr. aplikacija za klicanje, pošiljanje SMS sporočil, koledar, brskanje po internetu in nastavitve - skupaj imenovane *Googlove sistemske aplikacije (ang. Google Applications, GAPPS)*. Sistemske aplikacije ne služijo zgolj uporabniku, temveč tudi razvijalcu: s pomočjo vmesnikov teh aplikacij lahko namreč razvijalec uporabi funkcije, ki jih posamezne sistemske aplikacije imajo, in mu jih tako ni potrebno na novo razviti.

3.2 Zgradba in lastnosti aplikacij Android

Večina aplikacij Android je napisanih v programskem jeziku Java. Poleg tega vsako aplikacijo Android sestavljajo tudi druge datoteke, kot so npr. XML datoteke uporabniškega vmesnika in slikovne ter druge podatkovne datoteke. Kodo, ki je napisana v programskem jeziku Java, se tako dopolni s prej omenjenimi datotekami, vse navedene elemente pa se s pomočjo razvojnih orodij združi v enotno datoteko. Ta datoteka ima končnico .apk in omogoča namestitev aplikacije na napravi z operacijskim sistemom Android.

Da bi izboljšali varnost in zaščitili napravo ter podatke, ki se na njej nahajajo, pred zlonamernimi aplikacijami (ang. malware) in vdori, je vsaka aplikacija izolirana od drugih aplikacij na napravi in lahko dostopa le do operacijskega sistema. Pravimo, da vsaka aplikacija živi v svojem t.i. varnostnem peskovniku [13].

Z namenom zagotavljanja čim večje varnosti lahko poleg tega aplikacije Android dostopajo samo do tistih delov operacijskega sistema, ki jih nujno potrebujejo za svoje delovanje. Gre za t.i. pravilo najmanjšega privilegija (ang. principle of least privilege), ki se izvaja preko dovoljenj za dostop: aplikacija za vsak dostop do delov operacijskega sistema potrebuje uporabnikovo dovoljenje. Le-to smo za dostop do zunanje shrambe in interneta uporabili tudi pri razvoju naše aplikacije.

Navedeno pravilo o omejevanju dostopa ni absolutno. Za nemoteno delovanje aplikacij je namreč večkrat potrebna delitev datotek med aplikacijami.

To se lahko omogoči s tem, da si dve ali več aplikacij deli uporabniško ime. S tem pridobijo dostop do medsebojnih datotek.

3.2.1 Komponente aplikacij

Vsaka aplikacija je sestavljena iz različnih komponent. Le-te se delijo na štiri različne tipe [13]:

- aktivnosti (ang. activities),
- storitve (ang. services),
- sprejemniki dogodkov (ang. broadcast receivers) in
- ponudniki vsebin (ang. content providers).

Vsak tip komponent ima svoj točno določen namen in svoj specifičen življenjski cikel. V spodnjih odstavkih jih bomo podrobneje predstavili.

Aktivnosti Aktivnost je prva komponenta, s katero se uporabnik sreča pri vstopu v aplikacijo. Ko se uporabnik premika po aplikaciji, se na zaslonu mobilne naprave izriše različna vsebina. Posamezni vsebini oziroma zaslonu odgovarja ena aktivnost, ko pa se vsebina ali zaslon spremeni, se zamenja tudi aktivnost. Zato pravimo, da vsaka aktivnost predstavlja en zaslon uporabniškega vmesnika [13]. To je mogoče ponazoriti z naslednjim primerom: aplikacija kontakti ima eno aktivnost za pregled vseh kontaktov, drugo za urejanje podatkov o uporabnikih in tretjo za dodajanje uporabnikov. Vse aktivnosti skupaj predstavljajo enotno aplikacijo kontakti. Druge aplikacije lahko, če to aplikacija kontakti dovoli, dostopajo do posamezne aktivnosti. Aplikacija telefon (ang. phone) npr. dostopa neposredno do seznama kontaktov in s tega seznama prebere podatke o imenu, priimku in telefonski številki kontakta. Uporabi jih tako, da telefonsko številko pokliče, ime in priimek pa izpiše na zaslonu. Vsaka aktivnost je podrazred razreda Activity.

Pri razvoju naše aplikacije smo od vseh komponent aplikacij največ uporabljali prav aktivnost. Tako npr. eno aktivnost naše aplikacije predstavlja prikazovalnik vsebine, drugo prikazovalnik menijev in tretjo nastavitve aplikacije. Aktivnosti se v naši aplikaciji v veliki meri prilagajajo vhodnim podatkom, ki jih pridobijo z namenom (ang. intent)[1]. S tem smo uresničili enega izmed ciljev, ki smo si jih zadali pri načrtovanju aplikacije, in sicer da naj aplikacija omogoča hitro in enostavno spreminjanje vsebine prezentacijskih elementov.

Storitve Storitev je komponenta oziroma del aplikacije, ki nima uporabniškega vmesnika in skrbi za izvajanje tistih operacij aplikacije, ki so uporabniku skrite. Primeri so transakcija podatkovne baze, prenosi datotek, prikaz ure in predvajanje glasbe, medtem ko uporabnik v drugi aplikaciji brska po internetu. Na storitev, ki teče v ozadju (ang. background), uporabnik navadno ni opozorjen. Operacijski sistem lahko storitev zaradi pomanjkanja pomnilnika uniči. Storitve so podrazred razreda Service.

Sprejemniki dogodkov Sprejemnik dogodkov je komponenta, ki nima uporabniškega vmesnika in omogoča aplikaciji, da se odzove na sistemsko obvestilo. Z vidika operacijskega sistema je tudi sprejemnik dogodkov vstopna točka v aplikacijo. Ta komponenta omogoča vstop tudi v aplikacije, ki se trenutno ne izvajajo, kot npr. v aplikacijo Whatsapp. Ta pri svoji namestitvi zahteva sms sporočilo od sistemskega strežnika aplikacije, da potrdi avtentičnost uporabnika. Aplikacija se v operacijskem sistemu naroči na obvestilo o sms sporočilu, ki je poslano s številke sistemskega strežnika. Ko je sms sporočilo dostavljeno, sprejemnik dogodkov to sporoči aplikaciji, ki nato iz sms sporočila prebere validacijsko kodo. Uporabniku tako ni potrebno posebej vnesti oziroma prepisati validacijske kode, temveč to namesto njega naredi aplikacija sama. Namesto uporabniškega vmesnika lahko sprejemniki dogodkov ustvarijo obvestilo v vrstici stanja (ang. status bar), ki je name-

[1] Več o namenu v podpoglavju 3.2.2.

njeno seznanitvi uporabnika s predmetnim dogodkom. Sprejemniki dogodkov so podrazred razreda BroadcastReceiver. Vsako obvestilo je dostavljeno kot namen (ang. intent).

Ponudniki vsebin Večina aplikacij za svoje delovanje uporablja tudi vsebino, ki ni shranjena v aplikaciji sami. Pri tem gre največkrat za datoteke v datotečnem sistemu naprave, lahko pa tudi za podatke z interneta ali podatke, ki so shranjeni v podatkovnih bazah. Povezovalni element med aplikacijo in podatki, ki niso shranjeni v sami aplikaciji, so ponudniki vsebin. S pomočjo te komponente lahko aplikacije iščejo, spreminjajo ali dodajajo vsebino, ki jo posamezni ponudnik vsebin omogoča. Primer je ponudnik vsebin Contextimpl, ki smo ga uporabili tudi v naši aplikaciji. Omogoča nam, da iz *enotnega identifikatorja virov (ang. Uniform Resource Identifier, URI)* pridobimo naslov do datoteke z vsebino. Ponudniki vsebin so podrazred razreda ContentProvider.

Pri razvoju operacijskega sistema Android se že od njegovega nastanka poudarja modularnost in t.i. recikliranje funkcij aplikacij. Če ima neka aplikacija funkcijo, ki jo potrebuje tudi druga aplikacija, lahko le-ta do te funkcije dostopa preko operacijskega sistema in jo uporabi. Preko APIja aplikacije zažene le komponento, ki jo potrebuje. Komponenta nato opravi delo za aplikacijo, ki jo je zagnala. S tem je olajšano razvijanje novih aplikacij, saj ni potrebno razviti tistih funkcij, ki jih že omogočajo druge aplikacije. Takšnega deljenja komponent aplikacij ne pozna noben drug operacijski sistem.

3.2.2 Aktiviranje komponent

Komponente aplikacij se aktivirajo na različne načine:

- aktivnosti, storitve in sprejemniki dogodkov se aktivirajo s **sporočilom namen (ang. Intent)**,
- ponudniki vsebin se aktivirajo z **razrešiteljem vsebin (ang. Content Resolver)**.

Sporočilo namen se ustvari z razredom Intent. Namen ponavadi vsebuje podatke, ki jih aktivirana komponenta potrebuje, ali pa podatke o tem, kaj je potrebno storiti. Takšno navodilo oziroma sporočilo v namenu je lahko namenjeno bodisi posamezni komponenti bodisi določenemu tipu komponent. Namen za posamezno komponento se imenuje eksplicitni namen, namen za določen tip komponent pa implicitni namen. Namen, ki je namenjen aktivnostim in storitvam, se od namena, namenjenega sprejemnikom dogodkov, loči po vsebini podatkov: prvi aktivnostim in storitvam posreduje posamezno navodilo in podatke, drugi pa zgolj obvestilo o dogodku.

Ponudniki vsebin so, kot pojasnjeno zgoraj, aktivirani z razrešiteljem vsebine. Tako mora aplikacija, ki želi določeno vsebino, le-to pridobiti z metodami razreda ContentResolver.

Pri razvoju naše aplikacije smo velikokrat uporabili namen. Z njegovo pomočjo smo prenašali podatke med aktivnostmi. Prav tako smo za aktiviranje komponente ponudnik vsebin uporabili razrešitelja vsebin tipa VariableResolver.

3.2.3 Datoteka manifest

Vsaka aplikacija operacijskega sistema Android ima datoteko manifest.xml. V njej so definirane komponente aplikacije, njena varnostna dovoljenja, najnižja možna raven API stopnje, ki jo aplikacija zahteva za svoje delovanje, podatki o strojni in programski opremi, ki jo bo uporabljala aplikacija, ter podatki o zunanjih knjižnicah, ki so uporabljene v aplikaciji [13]. Datoteka manifest naredi komponente vidne operacijskemu sistemu. Nahajati se mora v korenski mapi aplikacije.

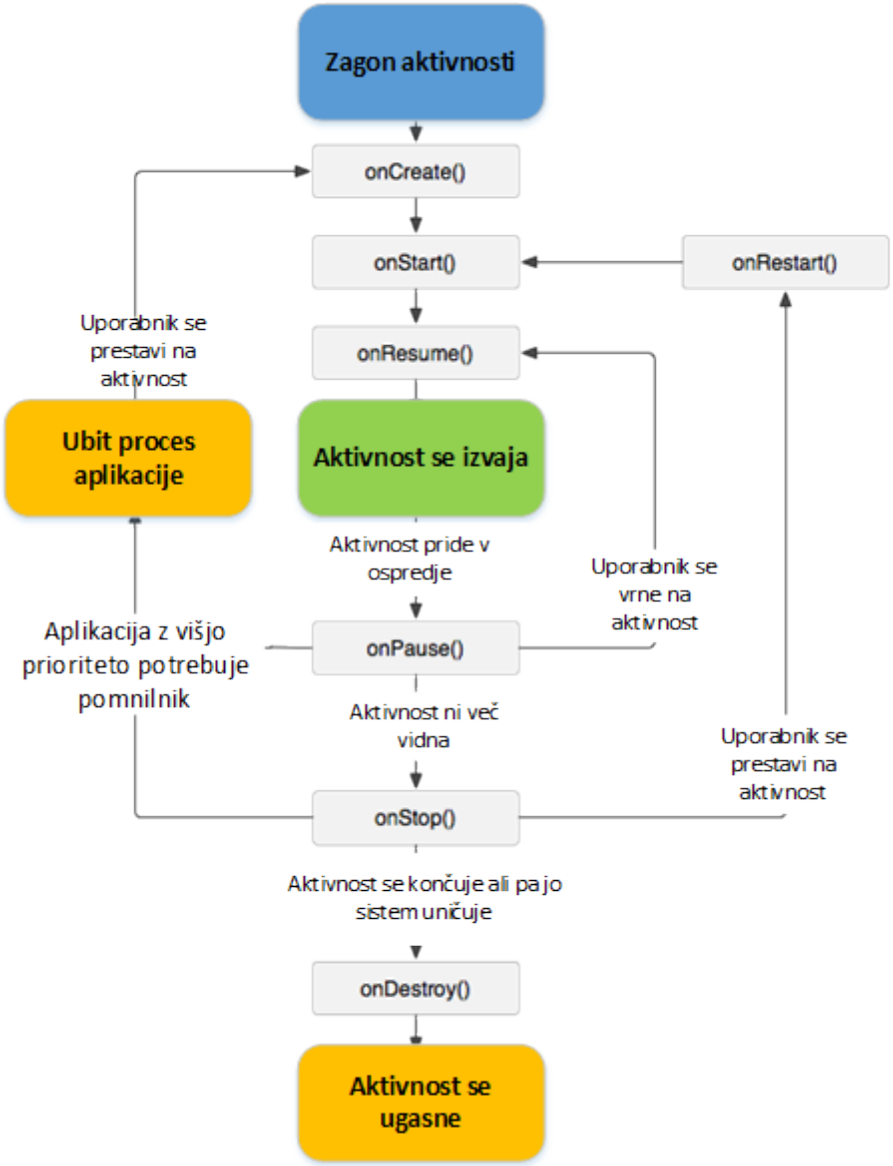
Aktivnosti, storitve in ponudniki vsebin morajo biti nujno definirani v datoteki manifest. V nasprotnem primeru jih namreč operacijski sistem ne zazna in ne morejo delovati. Za razliko od njih lahko sprejemniki dogodkov delujejo tudi, če niso definirani v datoteki manifest, če so ustvarjeni programsko kot objekti tipa BroadcastReceiver.

3.2.4 Življenjski cikel aktivnosti

Z življenjskim ciklom aktivnosti označujemo različna stanja aktivnosti. Ta stanja se spreminjajo glede na to, kaj se z aplikacijo v posameznem trenutku dogaja. Tako je npr. aktivnost pri zagonu aplikacije v drugem stanju kot pri sami uporabi aplikacije. Zato se v literaturi poudarja, da med uporabnikovim premikanjem po aplikaciji aktivnosti aplikacije prehajajo med različnimi stanji v svojem življenjskem ciklu [14]. Kot je bilo to opisano že v prejšnjih podpoglavjih, se vsaka aktivnost ustvari kot podrazred razreda Activity. Leta zagotavlja več metod (klicev), s pomočjo katerih se aktivnost seznani z njenim dogajanjem oziroma s procesom, ki jo gosti. Možna stanja aktivnosti so [14], [15], [16]:

- stanje kreiranja (ang. created) - je prvo stanje po zagonu aplikacije,
- začeto stanje (ang. started) - sledi stanju kreiranja,
- aktivno stanje (ang. resumed) - samo ena aktivnost je lahko aktivna na enkrat. To je takrat, ko se aktivnost pojavi na zaslonu mobilne naprave in je pripravljena na uporabo. Pravimo, da je v tem stanju aktivnost v ospredju (ang. foreground),
- stanje mirovanja (ang. paused) - ko uporabnik določene aktivnost ne uporablja več, se ta zaustavi in preide v stanje mirovanja,
- stanje ustavitve (ang. stopped) - če je aktivnost popolnoma prekrita z drugo aktivnostjo, se aktivnost ustavi [16]. Operacijski sistem ohrani vrednosti vseh elementov aktivnosti, tako da ob ponovnem zagonu aktivnosti nismo izgubili podatkov,
- stanje uničenja (ang. destroyed) - če se aktivnost sama konča ali uporabnik pritisne gumb nazaj (ang. back), se aktivnost uniči. Do uničenja aktivnosti lahko poleg tega pride tudi zaradi pomanjkanja sistemskih virov, kot je npr. pomnilnik. V takšnem primeru si operacijski sistem zapomni, da je aktivnost obstajala, zato lahko ponovno ustvari enako

aktivnost z enakimi podatki. Zanimivo pri tem je, da se uničenje in ponovno ustvarjanje aktivnosti zgodi vsakič, ko uporabnik obrne zaslon za 90 stopinj (in povzroči drugačno postavitev vsebine), saj mora operacijski sistem v takšnem primeru na novo naložiti sistemske vire.



Slika 3.2: Prehodi med stanji aktivnosti [13]

Prehode med stanji aktivnosti spremljajo metode razreda Activity. Slika 3.2 ponazarja prehode med zgoraj opisanimi stanji aktivnosti in posamezne metode.

3.2.5 Metode življenjskega cikla aktivnosti

Aktivnosti spreminjajo svoja stanja iz več razlogov. Najbolj pogost razlog je ravnanje uporabnika, ki s svojo uporabo aplikacije povzroči, da posamezna aktivnost spremeni stanje (npr. iz aktivnega v stanje mirovanja). Poleg navedenega so lahko razlogi za spremembo stanja aktivnosti tudi sledeči: izklop zaslona zaradi neuporabe naprave, izklop naprave zaradi pomanjkanja energije, varčevalec baterije in drugi. Ko aktivnost spremeni stanje, se sprožijo metode, ki so definirane v razredu Activity in po potrebi še enkrat definirane oziroma implementirane (ang. Override) v vsaki aktivnosti. Možne metode spremembe stanja aktivnosti so naslednje [17]:

- **onCreate()** - je prva metoda, ki se sproži po zagonu aktivnosti. V tej metodi se opravi osnovna začetna inicializacija spremenljivk in ostalih potrebnih elementov aktivnosti. Bistvo te metode je v tem, da pripravi vse, kar je potrebno, za osnovni zagon aktivnosti. Za delovanje metode je pomemben parameter savedInstanceState, ki vsebuje podatke o prejšnjem stanju aktivnosti. S tem je omogočeno, da se pri zagonu aktivnosti izhaja iz stanja, ki je obstajalo v prejšnjem življenjskem ciklu aktivnosti. Če aktivnost prejšnjega življenjskega cikla ni imela, operacijski sistem kljub temu pošlje parameter savedInstanceState, ki pa je v tem primeru prazen. Po izvedeni metodi onCreate() se takoj izvede metoda onStart(), nato pa še metoda onResume().

- **onStart()** - metoda se izvede vsakič po metodi onCreate() in onRestart(), ko aktivnost preide v začeto stanje (ang. started). Temeljna funkcija metode onStart() je v tem, da aktivnost prikaže na zaslonu mobilne naprave. To stori s tem, da inicializira uporabniški vmesnik. Metoda se konča zelo hitro, saj aplikacija nikoli ne ostane v začetem

stanju, temveč takoj za tem izvede naslednjo metodo - onResume().

- **onResume()** - to metodo operacijski sistem izvede takrat, ko aktivnost preide v aktivno stanje (ang. resumed) in se pojavi na zaslonu mobilne naprave. Aktivnost je s tem pripravljena na uporabo in je na voljo uporabniku vse dokler se ne zgodi t.i. prekinitveni dogodek. Primeri prekinitvenih dogodkov so sledeči: uporabnik preide na novo aktivnost, zaslon naprave se ugasne ali uporabnik prejme telefonski klic.

- **onPause()** - ta metoda se izvede takrat, ko se zgodi eden izmed prekinitvenih dogodkov. Njena osnovna funkcija je sproščanje sistemskih virov. Pomembno je namreč, da se določene operacije, ki porabijo veliko sistemskih virov, ne izvajajo, če uporabnik ne uporablja aktivnosti. Kot primer takšnih operacij se v literaturi omenjajo navigacija GPS, animacije v aplikaciji in kamera [17]. Izvajanje te metode je časovno zelo omejeno, zato v njej ni smiselno izvajati časovno zahtevnih operacij, kot so bazne transakcije in shranjevanje podatkov. To se opravi v metodi onStop(). Aktivnost ostane v stanju mirovanja tudi po končani metodi onPause(). Iz stanja mirovanja lahko aktivnost preide nazaj v aktivno stanje. To se zgodi takrat, ko uporabnik ponovno začne uporabljati aktivnost. Druga možnost je, da aktivnost iz stanja mirovanja preide v stanje ustavitve (kadar aktivnost popolnoma nadomesti oziroma prekrije druga aktivnost) ali v stanje uničenja (kadar operacijski sistem potrebuje sistemske vire).

- **onStop()** - ko aktivnost preide v stanje ustavitve, ni več vidna na zaslonu mobilne naprave. To se zgodi tedaj, ko je aktivnost opravila svoje delo in se zaradi tega konča, ali pa kadar uporabnik ustvari novo aktivnost. Takrat operacijski sistem izvede metoda onStop(). Če je aktivnost v stanju ustavitve, jo lahko operacijski sistem uniči, da sprosti potreben delovni pomnilnik. Pri tem je pomembno, da se tudi v tem primeru podatki o izvajani aktivnosti ne izgubijo, saj jih operacijski sitem shrani v parametru savedInstanceState. Zaradi tega se lahko

aktivnost ponovno zažene (z metodo onRestart()) ali pa se uniči.

- **onDestroy()** - je zadnja metoda, ki se izvede preden pride do uničenja aktivnosti. Je obenem tudi zadnja metoda, ki se izvede v življenjskem ciklu aktivnosti. Funkcija te metode je podobna funkciji metod onPause() in onStop(), tj. varčevanje s sistemskimi viri. Metoda onDestroy() namreč dokončna nalogo, ki sta jo deloma opravili že prej omenjeni metodi onPause() in onStop(), saj sprosti vse tiste sistemske vire, ki jih posamezna aktivnost še uporablja. Metoda onDestroy() je nepovratna: ko aktivnost enkrat preide v stanje uničenja, se ugasne in jo mora uporabnik, če jo ponovno potrebuje, ustvariti na novo.

3.3 Razvojna orodja

Za razvoj aplikacije smo uporabili različna orodja. Programerski del razvoja smo naredili v Android Studiu, skupaj z Android SDKjem. Pripravo kontrolne menijske datoteke smo naredili z Oxygen XML Editorjem, samo vsebino aplikacije pa z Adobe Dreamweaverjem.

3.3.1 Android Studio

Android Studio je uradni IDE za operacijski sistem Android [18]. Zgrajen je na IntelliJ IDEA razvojnem okolju, ki je profesionalno razvojno okolje za Javo, proizvajalca JetBrains. Nadomestil je razvojno okolje Eclipse in vtičnik ADT (Eclipse Android Development Tools) kot primarno razvojno okolje za Android [19]. V literaturi se poudarja, da ima trenutna verzija naslednje funkcije [18],[19], [20]:

- skriptni sistem izgradnje aplikacije imenovan Gradle,
- emulator, ki omogoča preizkus aplikacije na vseh vrstah navideznih naprav Android,

- funkcija hitri zagon (ang. Instant Run), ki omogoča spremembe v aplikaciji, ne da bi jo bilo potrebno ponovno prevesti,
- integracija z GitHub razvojno platformo,
- orodja Lint za iskanje težav s hitrostjo ali siceršnjim delovanjem aplikacije ali težav s kompatibilnostjo verzij,
- urejevalnik XML datotek uporabniškega vmesnika, ki omogoča funkciji povleci-in-spusti (ang. drag-and-drop) in predogled postavitve,
- analiza izjem (ang. exception),
- samodejno zaključevanje kode,
- analizator apk datotek za enostavni pregled vsebine apk-ja, ki ugotovi velikost vsake komponente, omogoča pregled DEX datotek in primerjavo dveh apk-jev.

Pri razvoju naše aplikacije smo uporabljali razvojno orodje Android Studio, konkretno naslednje funkcije tega orodja: Gradle, emulator, urejevalnik XML datotek uporabniškega vmesnika, analizo izjem in samodejno zaključevanje kode.

3.3.2 Android SDK

Android SDK je niz orodij za razvijanje aplikacij, ki delujejo v operacijskem sistemu Android. Vsebuje naslednje elemente [21], [22]:

- potrebne knjižnice,
- razhroščevalnik (ang. debugger),
- emulator navideznega stroja ART,
- aplikacijski programski vmesnik API,
- vzorce izvorne kode.

Verzije SDKja sledijo verzijam APIjev. Ko izide nova verzija operacijskega sistema Android, istočasno izideta tudi novi verziji SDKja in APIja.

3.3.3 Oxygen XML Editor

Oxygen XML Editor je urejevalnik vseh tipov XML datotek in shematskih kontrolnih datotek tipa *shematska definicija XML (ang. XML Schema Definition, XSD)*. S tem razvojnim orodjem je mogoče preveriti pravilnost zgradbe XML dokumenta in njegovo skladnost s XSD shemo. Možni načini pregledovanja in urejanja datotek so naslednji [23]:

- **tekstovni pogled** (ang. Text View) - je osnovni pogled urejanja dokumentov. Poleg prikaza dokumenta v tekstovni obliki omogoča tudi zaključevanje oznak (ang. tag),
- **mrežni pogled** (ang. Grid View) - prikaže dokument v obliki mreže. V stolpcih imamo gnezdene oznake (ang. tag), v vrsticah pa vsebino, ki spada v to oznako,
- **avtorski pogled** (ang. Author View) - za ta pogled datoteka potrebuje CSS datoteko, v kateri se določi vrsto podatkov za vsako oznako v XML datoteki. Pri našem razvoju aplikacije tega načina pregledovanja in urejanja datotek nismo uporabljali,
- **oblikovalski pogled** (ang. Design View) - je na voljo samo za shematske datoteke XSD. Ta način prikaže shemo datoteke in s tem omogoči lažje razumevanje, kako poteka preverjanje skladnosti XML datoteke s shemo.

3.3.4 Adobe Dreamweaver

Adobe Dreamweaver je spletno razvojno orodje, ki ga je leta 1997 prvič izdalo podjetje Macromedia. Omogoča razvoj najbolj enostavnih spletnih strani HTML, prav tako spletnih strani, obogatenih s CSS slogi in vsebinami

JavaScript. Poleg tega to razvojno orodje podpira različne strežniške skriptne jezike, kot so npr. ASP JavaScript, ASP VBScript, ASP.NET C#, ASP.NET VB, Coldfusion, Scriptlet in PHP [24].

Zasnova tega orodja je takšna, da omogoča izdelavo spletnih strani brez poznavanja programskih jezikov (npr. HTML). Tako spletno stran zgradimo iz gradnikov, ki jih ponuja to orodje. Gradnike vnašamo na spletno stran z miško po principu povleci in spusti (ang. drag-and-drop). Na podlagi izbranih gradnikov nato Dreamweaver avtomatsko zgradi kodo spletne strani. V Dreamweaverju lahko uporabnik na enem mestu oblikuje, kodira in upravlja s spletnimi stranmi. Vsebino datotek spreminja lokalno in jih nato naloži na spletni strežnik z uporabo *protokola za prenos datotek (ang. File Transfer Protocol, FTP)*, *varnega protokola za prenos datotek (ang. Secure File Transfer Protocol, SFTP)* ali *protokola za spletno porazdeljeno avtorstvo in različice (ang. Web Distributed Authoring and Versioning, WebDAV)*. Omogoča timski razvoj spletnih strani, saj ima tudi *sistem za upravljanje z izvorno kodo (ang. Subversion, SVN)*.

Med pomembnejšimi značilnostmi tega orodja je funkcija, ki se imenuje "živi pogled"(ang. Live View). Ta funkcija uporabniku omogoča, da že med samo izdelavo spletne strani vidi spremembe, ki jih je napravil, ter tako najlažje odpravi napake v spletnih straneh.

Poglavje 4

Razvoj in zgradba aplikacije

Kot je bilo to pojasnjeno že v uvodu, je bil cilj diplomskega dela razviti aplikacijo, ki bi bralcu predstavila tako vsebino za branje kot tudi interaktivno vsebino. Menili smo namreč, da bi lahko učitelji in profesorji s pomočjo takšne aplikacije učencem ter študentom učno vsebino prikazali v njim prijaznem formatu - aplikaciji za mobilne naprave. Da bi navedeni cilj izpolnili, smo predvideli tudi druge specifične funkcije aplikacije, ki jih obstoječi formati e-knjig ne omogočajo. Gre predvsem za t.i. funkcijo oddaljenih vsebin. Ta funkcija omogoča uredniku prikaz novih vsebin na napravi bralca, ne da bi le-ta moral kaj storiti. Tako lahko na primer slednji na mobilno napravo bralca - učenca postopoma dodaja izobraževalne vsebine, za katere želi, da se z njimi seznani postopoma. Na ta način je mogoče v aplikaciji hitro spremeniti oziroma dopolniti učno vsebino. Poleg navedenega smo želeli še, da aplikacija podpira interaktivne teste znanja, saj lahko na ta način urednik preveri, ali so uporabniki izobraževalno vsebino ustrezno prebrali in razumeli.

4.1 Specifikacija zahtev aplikacije

Glede na zgoraj opisane cilje smo pred programiranjem aplikacije postavili naslednje zahteve, ki naj jih naša aplikacija izpolnjuje:

- aplikacija bo delovala na mobilnih napravah z operacijskim sistemom

Android, saj je le-ta najbolj uporabljen operacijski sistem mobilnih telefonov. Zahtevana verzija operacijskega sistema naj bo 5.0 Lollipop - API nivo 21 (82% vseh aktivnih Android operacijskih sistemov je na tej verziji ali višji). Ciljna skupina naprav bodo mobilni telefoni,

- aplikacija bo omogočala prikaz vsebine, ki je shranjena lokalno na napravi ali na oddaljenem strežniku,
- aplikacija bo zahtevala dostop do interneta in bralni ter pisalni dostop do zunanje shrambe,
- uporabniški vmesnik se zgradi iz vhodne XML datoteke, ki omogoča grupiranje in gnezdenje vsebine,
- uporabniški vmesnik bo prilagodljiv, tako da lahko urednik ureja celotni uporabniški vmesnik, razen aplikacijskih gumbov nazaj in nastavitve,
- aplikacija naj omogoča branje in razčlenitev vhodne datoteke,
- sistem menijev se zgradi na podlagi vhodne datoteke,
- aplikacija naj podpira naslednje vrste vsebin: tekstovno, zvočno, slikovno, vizualno (video) in interaktivno vsebino,
- aplikacija naj omogoča shranjevanje več različnih vsebin hkrati, npr. po predmetih učnega ali študijskega leta,
- strani z vsebino naj bodo shranjene v formatu HTML,
- tipka nastavitve naj omogoča prenos nove vsebine v aplikacijo, tako da je mogoče v aplikacijo prenesti novo menijsko datoteko XML, ki je shranjena na napravi (t.i. lokalna vsebina) ali na strežniku (t.i. oddaljena vsebina),
- aplikacija naj uredniku omogoča enostaven vnos novih vsebin,

- kontrola pravilnosti vhodne XML datoteke naj se izvaja z XSD shemo,
- aplikacija naj bo pregledna in uporabna.

4.2 Sestavni deli aplikacije

Ob upoštevanju zahtev, ki smo jih opisali v podpoglavju 4.1, smo naredili načrt zgradbe aplikacije. Pri načrtovanju smo najprej upoštevali, da mora biti aplikacija sestavljena iz več delov. Ti deli so sledeči:

- prikazovanje vsebine,
- prikazovanje spremenljivega uporabniškega vmesnika in
- podpora prikazovanju vsebine ter uporabniškega vmesnika.

Nadalje smo določili temeljno zasnovo aplikacije. Predvideli smo, da bi aplikacija delovala na naslednji način:

- pri zagonu naj aplikacija iz trenutno aktivne datoteke prebere menijsko zgradbo,
- na podlagi prebrane menijske zgradbe aplikacija zgradi sistem menijev, podmenijev in povezav do vsebine,
- vsebina se nahaja v ločenih datotekah, ki se preberejo in prikažejo samo po potrebi oziroma uporabnikovi želji,
- uporabnik z gumbom nazaj in gumbi v meniju brska po vsebini.

Načrtovanje in programiranje aplikacije je potekalo po principu "od zgoraj navzdol"(ang. Top-Down), kar pomeni, da smo najprej določili, kakšen bo končni izgled in uporaba aplikacije, kasneje pa smo po teh zahtevah naredili podporno logiko.

Pri načrtovanju prikazovanja vsebine smo najprej želeli upoštevati zahtevi, da naj aplikacija omogoča prikaz veliko različnih oblik vsebine in da

naj bo kreiranje vsebine enostavno za urednika. Sprva smo razvoj aplikacije načrtovali tako, da bi vsaka posamezna stran podpirala zgolj določen tip vsebine (npr. tekstovno, zvočno ali vizualno (video) vsebino). Vendar pa smo nato takšen načrt spremenili in se odločili, da naj aplikacija vsebino prikazuje kot HTML spletne strani. S tem je namreč omogočeno prikazovanje več različnih tipov vsebin hkrati, aplikacija pa je bolj pregledna in uporabna, s čimer v večji meri sledimo zahtevam, ki smo jih postavili pred programiranjem aplikacije.

V okviru načrtovanja prikazovanja vsebine smo nadalje upoštevali, da prikaz HTML strani v Androidu omogoča več različnih razredov. Izbrali smo najbolj naprednega, tj. razred Webview.

Pri načrtovanju spremenljivega uporabniškega vmesnika smo se najprej soočili s težavo, kako bi uredniku omogočiti enostavno spreminjanje oziroma dodajanje novih vsebin in povezav v aplikacijo. Pri tem smo morali določiti, kakšna bi bila vhodna datoteka za izgradnjo uporabniškega vmesnika ter na kakšen način bi iz enostavne vhodne tekstovne datoteke zgradili menije in povezave do vsebinskih delov aplikacije. Vhodno datoteko smo najprej nameravali zastaviti tako, da bi uporabniški vmesnik zapisali v navadni tekstovni datoteki tipa .txt. Po tehtnem premisleku pa smo se odločili za datoteko tipa .xml, saj za njegovo urejanje obstaja veliko urejevalnikov, ki omogočajo kontrolo zgradbe XML datoteke s shematskimi datotekami tipa XSD.

4.3 Zgradba aplikacije

V tem poglavju bomo opisali delovanje aplikacije po posameznih funkcijskih sklopih:

- zagon aplikacije,
- branje in razčlenitev XML datoteke,
- izgradnja menijev,
- premikanje po menijih,

- prikaz vsebine in
- nastavitve in dostop do oddaljene vsebine.

Na sliki 4.1 je prikazana zgradba moje aplikacije po posameznih razredih. Razredi aplikacije so med seboj povezani, kar bo pojasnjeno v naslednjih podpoglavjih in je prav tako razvidno iz slike 4.1.

4.3.1 Zagon aplikacije

Vstopna točka aplikacije je aktivnost **FirstActivity.java**. Ta aktivnost najprej preveri dovoljenje aplikacije za dostop do zunanje shrambe (ang. External Storage) in dovoljenje za dostop do interneta. Nato aktivnost preveri, ali je aplikacija v namenu prejela vhodne podatke *enotnega identifikatorja virov (ang. Uniform Resource Identifier, URI)*. Te podatke je lahko aplikacija pridobila samo, v kolikor se je vrnila na začetno točko zaradi uporabnikove želje po spremembi vsebine. Vhodni podatki tipa URI izvirajo iz aktivnosti **SettingsCustom.java** in vsebujejo lokacijo nove vhodne datoteke. Aktivnost FirstActivity.java nato prebere in razčleni vhodno datoteko, ki je bila vanjo poslana preko namena, ali pa jo, če se aktivnost zažene prvič, pridobi v korenskem imeniku aplikacije.

4.3.2 Branje in razčlenitev XML datoteke

Branje datoteke je narejeno z razredom InputStream. Razčlenitev XML datoteke je kompleksna operacija zaradi lastnosti te datoteke. XML datoteka ima namreč gnezdene oznake, brez omejitve globine nivojev gnezdenja, zaradi česar je potrebna rekurzija. Oznake XML datoteke se v pomnilniku hrani v razredu **Item.java**. Ta razred zaradi zahteve o možnosti prenašanja podatkov med aktivnostmi preko namena implementira vmesnik Parcelable. Vmesnik Parcelable zahteva implementacijo novega konstruktorja, metode writeToParcel() in metode describeContents().

Razčlenitev XML datoteke se opravi v razredu **NodeParser.java**. Metoda parse(InputStream in) prejme podatke v obliki InputStream-a. Nato

prebere vhodne podatke in poišče oznako "catalog", ki je korenska oznaka naše XML datoteke. Z uporabo razreda DocumentBuilder se datoteka prebere v objekt tipa NodeList, iz katerega je nato potrebno podatke prebrati in zapisati v objekt tipa Item. Potek branja je naslednji:

1. ustvarimo seznam tipa ArrayList<Item> z imenom "entries", kamor bomo shranjevali objekte tipa Item,

2. nad objektom tipa NodeList pokličemo metodo readCatalog,

3. readCatalog pregleda vse naslednike v korenskem objektu in se glede na vrsto naslednika odloči, kaj je potrebno storiti: če je naslednik tipa meni (oziroma oznaka v XML datoteki "item") izvede klic entries.add(readItem()), če pa je objekt tipa vsebina (oziroma oznaka v XML datoteki "page"), izvede klic entries.add(readPage()),

4. metoda readItem najprej prebere vse atribute korenske oznake, nato pa naredi rekurzivni klic metode readCatalog. Objekte, ki jih prejme od metode readCatalog, shrani v objekt tipa Item, ki ga skupaj z atributi pošlje v vrnitvenem objektu,

5. metoda readPage prebere vse atribute korenske oznake. Naslednikov te oznake ni, zato metoda vrne objekt tipa Item s prej prebranimi atributi,

6. readCatalog vrne seznam tipa ArrayList<Item>, kamor smo shranili vse podatke iz datoteke XML.

4.3.3 Izgradnja menijev

Za izgradnjo menija je najprej potrebno določiti, kako se bo posamezna menijska vrstica imenovala in kam bo vodila (v podmenije ali na stran z vsebino). Ta dva podatka pridobimo iz pravkar prebranega objekta entries. V aplikaciji se posamezna menijska vrstica izriše kot gumb, na katerem se izpiše prej prebrani atribut "text" XML datoteke. Ta gumb vodi v želeni podmeni

ali na stran z vsebino. Nato ustvarimo vse gumbe za trenutni meni, jim dodamo prožilnike na klik (ang. onClickListener) in jih vnesemo v objekt tipa ListView, ki se izriše na ekranu. Prožilnik ob pritisku na gumb izvede klic metode modulActivityCreate in ji v argument zapiše, kateri gumb je bil pritisnjen. Ta metoda ustvari objekt namen, mu doda vse naslednike izbranega menija in zažene novo aktivnost: če je bila izbrana stran z vsebino, ustvari aktivnost Page2, sicer pa aktivnost Menus. Opisana metoda novo ustvarjeni aktivnosti pošlje predhodno ustvarjen namen.

4.3.4 Premikanje po menijih

Ob uporabnikovi izbiri podmenija se ustvari aktivnost Menus. Le-ta prebere podatke, ki jih je dobila z namenom. Iz namena pridobi objekt Item, iz njega pa podatke o vseh naslednikih. S temi podatki enako kot že aktivnost FirstActivity zgradi menije in nanje veže prožilnik. Ob sproženem prožilniku aktivnost Menus naredi enako kot aktivnost FirstActivity: v namen zapakira podatke, ki jih potrebuje naslednja aktivnost, in se glede na uporabnikovo izbiro podmenija ali vsebinske strani odloči, komu jih pošlje: novi aktivnosti Menus ali Page2.

4.3.5 Prikaz vsebine

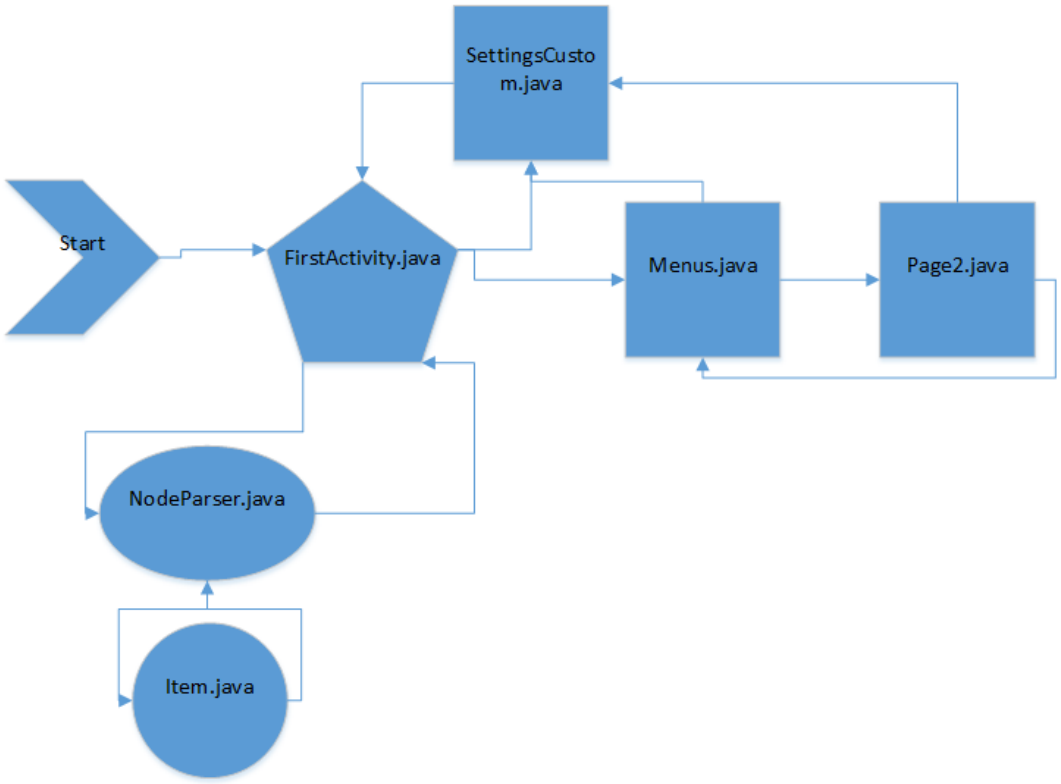
Ko aplikacija pride do aktivnosti Page2, smo na enem izmed zunanjih vozlišč datoteke XML. Preberejo se podatki iz namena, ki vsebujejo URI naslov do vsebine, ki naj bi se prikazala. Ta naslov je lahko lokalni ali spletni. Aktivnost za dostop do spletnih naslovov potrebuje dostop do interneta. Prikaz vseh vrst vsebine se naredi z razredom WebView. Slednjemu moramo še z argumenti omogočiti JavaScript in prikazovanje vizualnih (video) vsebin.

4.3.6 Nastavitve in dostop do oddaljene vsebine

Uporabnik lahko v aplikaciji z gumbom, ki se nahaja v zgornjem desnem kotu, vedno dostopa do aktivnosti SettingsCustom. V tej aktivnosti lahko

uporabnik novo datoteko XML prenese s spleta. To stori bodisi aktivnost SettingsCustom, ki vsebuje prostor za vpis spletnega naslova nove XML datoteke in njen prenos, ali pa uporabnik prenese XML datoteko na druge načine, kot je npr. s pomočjo spletnega brskalnika ali elektronske pošte. Nato uporabnik datoteko XML poišče na zunanji (lokalni) shrambi in potrdi vnos nove XML datoteke v aplikacijo. S tem z novo vhodno datoteko vrne aplikacijo v začetno stanje, tj. v aktivnost FirstActivity. S tem je omogočen prikaz novih vsebin, saj lahko urednik na določenem naslovu oddaljene vsebine, ki ga vsebuje aplikacija v meniju, po želji spreminja vsebino. Vsebina se spremeni, ko se uporabnik po ponovnem zagonu aplikacije vrne v isto točko ali meni.

Z gumbom "nazaj", ki se nahaja v zgornjem levem kotu, lahko uporabnik preide na prejšnjo aktivnost oziroma na prejšnji menijski nivo.



Slika 4.1: Prikaz razredov aplikacije

Poglavje 5

Prezentacijski elementi aplikacije

Učno vsebino lahko v aplikaciji oblikujemo, preoblikujemo in prikažemo v različnih oblikah, in sicer kot besedilo, zvok, slike, vizualne vsebine ali vsebine, oblikovane v programskem jeziku JavaScript. Posamezne oblike prikaza vsebin bodo predstavljene v nadaljevanju tega poglavja.

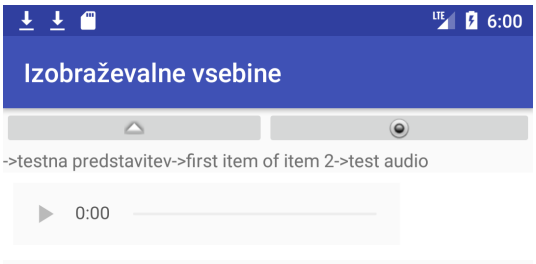
5.1 Besedilo

Besedilo lahko oblikujemo na vse načine, ki jih omogočata standarda HTML in HTML5. Pri HTML lahko uporabimo naslednje elemente: "color", "face" in "size". HTML5 omogoča uporabo analognih elementov, poleg njih pa lahko nastavimo še mnogo drugih, kot na primer "opacity", "outline-style", "position" ali "visibility".

Slika 5.1: Primer prikaza tekstovnih oblik

5.2 Zvok

Aplikacija preko formata HTML5 omogoča enostavno predvajanje zvočnih datotek. Element "audio"omogoča nastavitev začetka in konca predvajanja, nastavitev glasnosti in poljubno spreminjanje pozicije zvočne datoteke. Primer je prikazan na sliki 5.2.



Slika 5.2: Primer prikaza zvoka

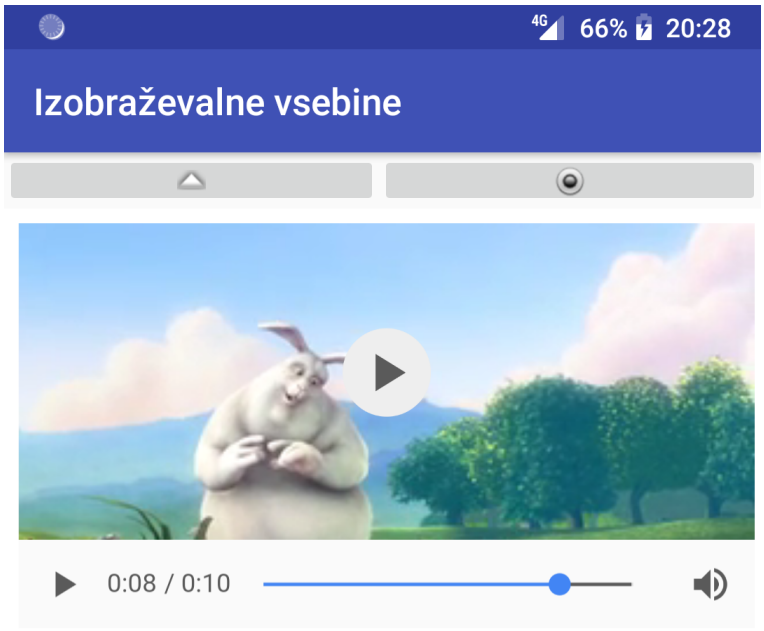
5.3 Slike

Slike se v aplikaciji prikažejo preko HTML in HTML5 formata, z uporabo elementov "img" ali "picture". "Picture" je novejša oznaka standarda HTML5, ki omogoča prilagajanje slike različnim dimenzijam ekrana. Primer uporabe je prikazan na sliki 5.3.

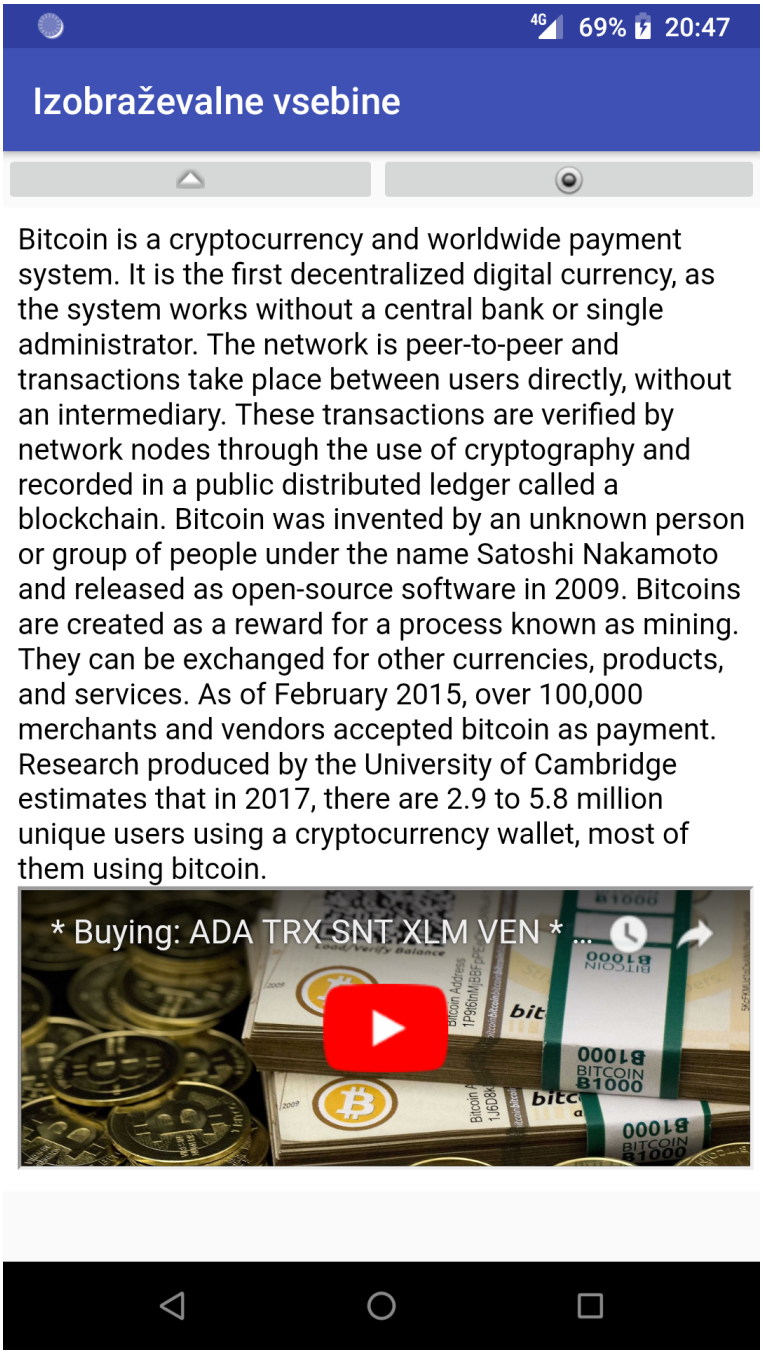
Slika 5.3: Primer prikaza slike s spremljajočim tekstom

5.4 Vizualne (video) vsebine

Aplikacija omogoča tudi predavanje vizualnih oziroma t.i. video vsebin. Za predvajanje uporablja predvajalnik HTML5. Predvaja lahko vse tri formate, ki so podprti v tem standardu, in sicer MP4, WebM in Ogg. Slika 5.4 prikazuje predvajanje videa tipa MP4. Poleg tega aplikacija omogoča tudi predvajanje vgrajenih, a oddaljenih vizualnih (video) vsebin, npr. s portala *YouTube.com*. Primer takega prikaza skupaj s tekstom se nahaja na sliki 5.5.



Slika 5.4: Primer prikaza vizualne vsebine MP4



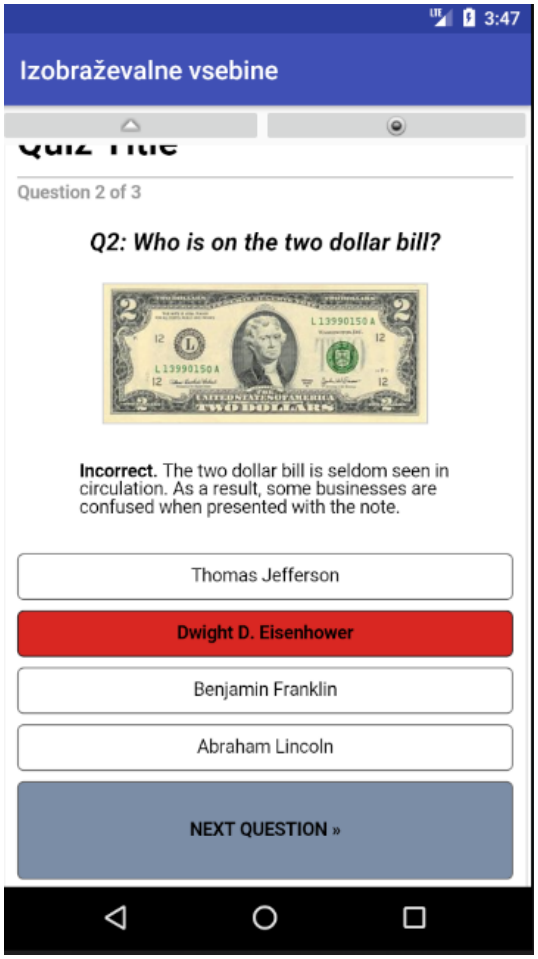
Slika 5.5: Primer prikaza vgrajene vizualne (video) vsebine s portala *YouTube.com*

5.5 Vsebine JavaScript

Aplikacija omogoča tudi prikaz vsebin, ki so določene s kodo v programskem jeziku JavaScript. Primer uporabe je prikazan na enostavnem primeru na sliki 5.6, ki s pritiskom na gumb spremeni vsebino prikazanega besedila na strani. Primer kode na spodnjem delu slike 5.6 nam prikaže stran, na kateri lahko s klikom na gumb s kodo JavaScript spremenimo vsebino teksta na strani. Prav tako je na isti sliki zgoraj prikazano delovanje te kode v aplikaciji.

Slika 5.6: Primer prikaza vsebine, oblikovane v programskem jeziku JavaScript

Slika 5.7 prikazuje primer kviza, ki se naredi s pomočjo programskega jezika JavaScript. Kviz poda vprašanje in štiri možne odgovore, po izbiri odgovora preveri njegovo pravilnost in nato na koncu vseh vprašanj naredi še analizo uspešnosti kviza. Z omenjenim programskim jezikom bi lahko



Slika 5.7: Primer prikaza kviza, narejenega v programskem jeziku JavaScript

stran priredili tudi tako, da bi po prebrani vsebini naredili kratek test in tako preverili bralčevo znanje vsebine, ki si jo je pravkar ogledal. Rezultat kviza bi lahko prenesli na strežnik, kjer bi urednik dobil rezultate vseh bralcev (lahko anonimno, lahko pa tudi poimensko). S temi rezultati bi lahko analiziral prikazano v smislu odziva uporabnikov na dane učne vsebine ali ocenil njihovo delo. To funkcijo bi bilo potrebno narediti v sami vsebini oziroma na strani s kvizom, tj. z ustrezno kodo v jeziku JavaScript.

Poglavje 6

Postopek tvorbe prezentacijske vsebine

Vsebina aplikacije nastaja v dveh delih. Urednik vsebine aplikacije mora poskrbeti za kreiranje glavne datoteke, tj. datoteke "menu.xml". Le-ta vsebuje zgradbo (nivoje) menijev in strani. V to glavno datoteko urednik poveže strani z vsebino. V nadaljevanju je prikazan praktičen primer tvorbe prezentacijske vsebine v celoti. Najprej je prikazan primer vsebine glavne konfiguracijske datoteke "menu.xml" in zatem še podrobnosti nastanka vsebine aplikacije oziroma t.i. prezentacijskih elementov.

6.1 Nastanek menijev

Menije aplikacija ustvari iz datoteke tipa XML. Oznake, ki se uporabljajo v datoteki, so sledeče:

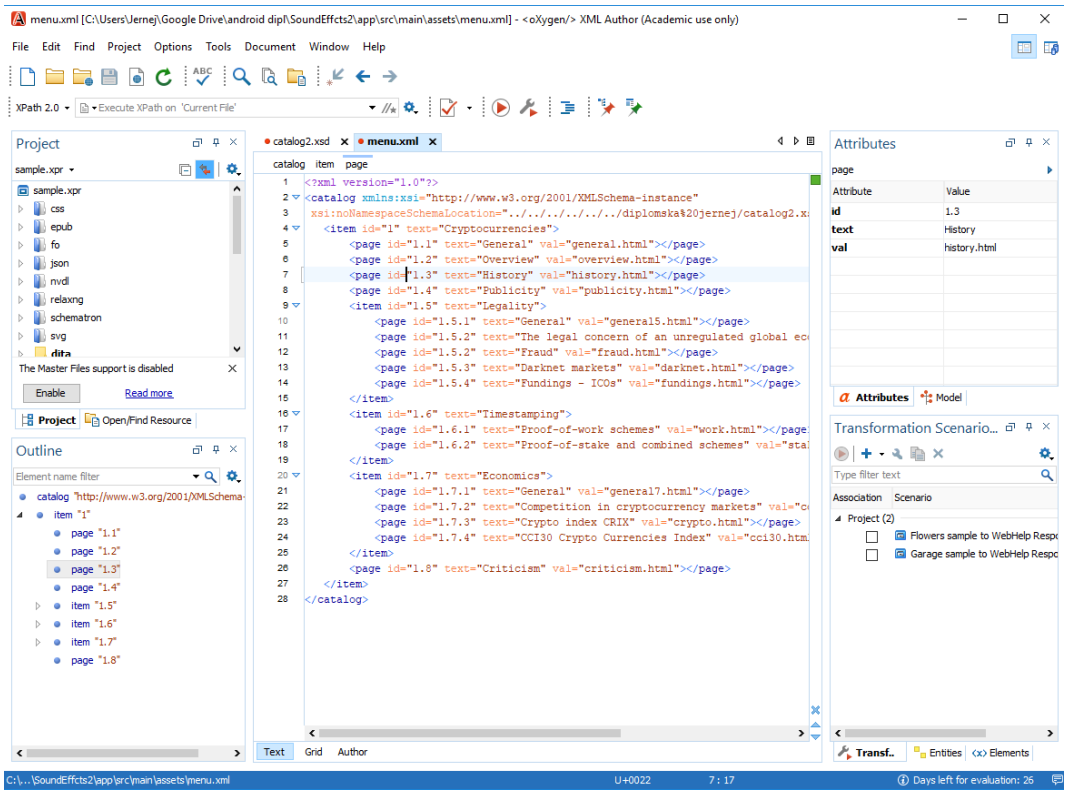
- **catalog**: oznaka za začetek in konec datoteke meni.xml. Ne vsebuje nobenih atributov,
- **item**: oznaka za vrstico v meniju, preko katere se dostopa v podmenije in strani. Njena atributa sta "id", ki je številčna oznaka vrstice, in "text", v katerega zapišemo besedilo te vrstice menija,

- **page**: oznaka za vrstico menija, ki vodi na stran z vsebino. Ima tri atribute: "id" in "text" sta identična kot pri oznaki "item", v atribut "val" pa shranimo vrednost, ki nas vodi do vsebine (lokalna ali oddaljena spletna stran).

Glavni urednik lahko ustvari novo datoteko "menu.xml" s pomočjo naslednjih orodij in predlog:

- XML urejevalnika "oxygen XML Author", ki je prikazan na sliki 6.1,
- XSD preverjevalne sheme na sliki 6.3,
- že narejenega konkretnega primera datoteke "menu.xml"na sliki 6.2.

V datoteki "menu.xml" lahko urednik uporabi oddaljene ali lokalne strani. Če uporabi oddaljene strani, bralcu pošlje le datoteko "menu.xml", vsebina pa se na bralčevo napravo prenese preko interneta. Če se urednik odloči uporabiti lokalno vsebino, mora poleg glavne datoteke bralcu poslati še vse spremljajoče datoteke z vsebino. Slika 6.2 prikazuje primer datoteke, iz katere nastanejo meniji, slika 6.3 pa prikazuje kontrolno shemo za izgradnjo glavne datoteke.



Slika 6.1: Primer uporabe Oxygen XML Author programa

```
<?xml version="1.0"?>
<catalog>
<item id="1" text="Cryptocurrencies">
<page id="1.1" text="General" val="general.html"></page>
<page id="1.2" text="Overview" val="overview.html"></page>
<page id="1.3" text="History" val="history.html"></page>
<page id="1.4" text="Publicity" val="publicity.html"></page>
<item id="1.5" text="Legality" val="">
<page id="1.5.1" text="General" val="general5.html"></page>
<page id="1.5.2" text="The legal concern of an unregulated
global economy" val="legal.html"></page>
<page id="1.5.2" text="Fraud" val="fraud.html"></page>
<page id="1.5.3" text="Darknet markets" val="darknet.html"></page>
<page id="1.5.4" text="Fundings - ICOs" val="fundings.html"></page>
</item>
<item id="1.6" text="Timestamping">
<page id="1.6.1" text="Proof-of-work schemes" val="work.html"></page>
<page id="1.6.2" text="Proof-of-stake and combined schemes"
val="stake.html"></page>
</item>
<item id="1.7" text="Economics">
<page id="1.7.1" text="General" val="general7.html"></page>
<page id="1.7.2" text="Competition in cryptocurrency markets"
val="competition.html"></page>
<page id="1.7.3" text="Crypto index CRIX" val="crypto.html"></page>
<page id="1.7.4" text="CCI30 Crypto Currencies Index" val="cci30.html"></page>
</item>
<page id="1.8" text="Criticism" val="criticism.html"></page>
</item>
</catalog>
```

Slika 6.2: Primer vsebine datoteke "menu.xml"

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<xs:element name="catalog">
<xs:complexType>
<xs:sequence>
<xs:element ref="item"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="item">
<xs:complexType>
<xs:choice maxOccurs="unbounded">
<xs:element ref="item"/>
<xs:element ref="page"/>
</xs:choice>
<xs:attribute name="id" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="text" use="required" type="xs:NCName"/>
</xs:complexType>
</xs:element>
<xs:element name="page">
<xs:complexType>
<xs:attribute name="id" use="required" type="xs:NMTOKEN"/>
<xs:attribute name="text" use="required"/>
<xs:attribute name="val" use="required" type="xs:NCName"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Slika 6.3: Vsebina kontrolne datoteke XSD

6.2 Tvorba vsebine s prezentacijskimi elementi

Aplikacija lahko kot vsebino prikaže skoraj vsako spletno stran. Omejitev se pojavi pri kompleksnih JavaScript straneh (interakcija z uporabnikom), pri JavaScript straneh, ki niso pripravljene za mobilne naprave, in pri tistih straneh, ki so namenjene za visoke ločljivosti.

Urednik aplikacije mora poskrbeti, da se vsebine na strani lahko dobro prikažejo. To mora narediti za vsako vrsto vsebine posebej. Pri tekstovnih vsebinah to stori tako, da vrednosti atributa "font-size" doda pripono vw (primer - font-size: 10vw). Vrednost 10vw predstavlja velikost pisave, ki ustreza desetim odstotkom širine zaslona naprave. Pri slikovnih vsebinah pa se v atributu "width"določi odstotek površine zaslona, ki naj ga pokriva slika. Na podoben način je mogoče prilagajati tudi ostale vrste vsebin.

Vsebino za bralca urednik vnese na HTML stran. Le-to lahko ustvari na mnogo načinov. Najenostavnejši je z namenskimi programi, kot je na primer Adobe Dreamweaver ali spletnimi stranmi, specializiranimi za ustvarjanje novih spletnih strani. Urednik, ki je vešč programiranja, pa lahko spletne strani ustvari tudi v urejevalnikih besedila, kot je Notepad++.

6.2.1 Lokalna vsebina

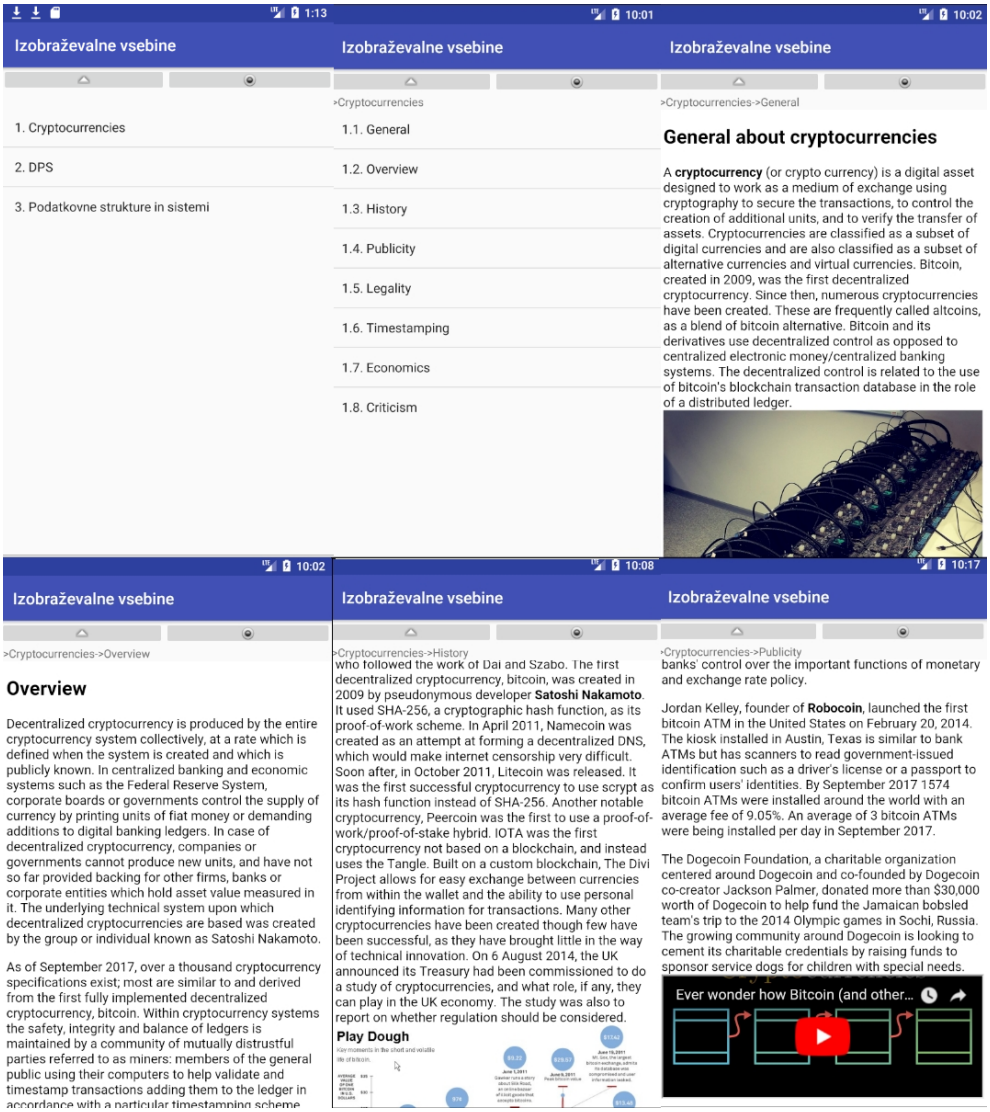
Pod lokalno vsebino štejemo vse strani, ki so shranjene na napravi sami in so glede na vsebino ter stil prilagojene prikazu v aplikaciji. Prikaz strani je specifičen, saj je aplikacija napisana za mobilne naprave Android, ki imajo zelo različne velikosti zaslonov.

6.2.2 Oddaljena vsebina

Oddaljena vsebina se od lokalne razlikuje po lokaciji hrambe podatkov. Podatki oddaljene vsebine so shranjeni na internetu. Ustvari se tako, da se v datoteki menu.xml pod element "val"vpiše celoten naslov spletne strani (npr. https://www.ibm.com/design/). Oddaljena vsebina uredniku aplikacije omogoča spreminjanje vsebine, pri čemer uporabniku obenem ni po-

trebno nadgraditi ali spremeniti same aplikacije. Urednik tako sam na oddaljeni lokaciji spremeni vsebino, ki se potem spremeni tudi pri uporabniku. Navedena funkcija aplikacije poleg tega omogoča tudi, da poleg urednika, ki skrbi za celotno aplikacijo, del vsebine ureja še nekdo drug.

Slika 6.4 prikazuje aplikacijo, ki ima naloženih več tem, vsa podpoglavja pa se nanašajo na kriptovalute. V zgornji polovici slike si zasloni iz leve proti desni sledijo na naslednji način: prvi prikazuje nabor različnih tem v aplikaciji, drugi podpoglavja omenjene teme o kriptovalutah, tretji pa uvodni tekst tega poglavja skupaj s sliko. V spodnji polovici slike zasloni od leve proti desni prikazujejo: prvi tekst podpoglavja pregled (na sliki: Overview), drugi zgodovino kriptovalut in tretji tako lokalno kot oddaljeno vsebino o publiciteti kriptovalut. Lokalna vsebina je besedilo, oddaljena pa vgrajen video iz portala *YouTube.com*.



Slika 6.4: Primer vsebine

Poglavje 7

Sklep

V diplomskem delu smo razvili aplikacijo, ki se lahko uporablja na mobilnih napravah z Android operacijskim sistemom in je zaradi svojih značilnosti namenjena za uporabo v izobraževalnem oziroma predstavitvenem procesu. Postopek programiranja je potekal v več fazah: najprej smo postavili specifikacijo zahtev, ki naj bi jih aplikacija izpolnjevala, nato smo naredili načrt zgradbe aplikacije in se na koncu lotili samega programiranja. Najzahtevnejši del sta predstavljala omogočanje vsebin JavaScript in prikaz vizualnih (video) vsebin. Pri programiranju aplikacije in posameznih problemih, ki se v zvezi s tem pojavijo, so v pomoč različne spletne vsebine. Med najkoristnejšimi je zagotovo https://developer.android.com [25], saj so na navedeni spletni strani podrobno, jasno in razumljivo opisani postopki in metode programiranja aplikacij Android.

Glede zahtev, ki smo jih postavili pred programiranjem aplikacije, lahko ugotovimo, da so bile implementirane vse želene lastnosti aplikacije. S tem v zvezi velja izpostaviti zlasti funkcijo prikaza oddaljenih vsebin, ki omogoča hitro spremembo učne vsebine na mobilnem telefonu uporabnika, ne da bi uporabnik sam za to moral kar koli storiti. Poleg tega aplikacija, kot je bilo to prav tako načrtovano, omogoča več oblik podajanja izobraževalnih vsebin (tekstovne, zvočne in vizualne) ter podpira interaktivne teste znanja. Vse to so funkcije aplikacije, s katerimi sledimo zastavljenim ciljem, tj. približati

izobraževalni proces uporabnikom in za potrebe boljšega izobraževalnega procesa izkoristiti hiter razvoj tehnologije ter široko uporabo pametnih mobilnih telefonov.

Aplikacija sicer omogoča tudi določene nadgradnje, ki so predvsem v predlogah kvizov v programskem jeziku JavaScript. Za urednika bi bilo smotrno pripraviti predloge kvizov, saj je programiranje le-teh v programskem jeziku JavaScript kompleksno in zato ni primerno za urednika brez znanja tega programskega jezika.

Upoštevaje navedeno je mogoče zaključiti, da smo uspeli uresničiti cilj diplomskega dela, saj aplikacija izpolnjuje vse zahteve in lastnosti, ki smo jih postavili. Tovrstno programiranje poleg tega pomeni nadgradnjo osvojenih metod programiranja na fakulteti, saj se s programiranjem Android naprav tekom študija nismo podrobneje seznanili.

Literatura

[1] (2017) Ifla elending background paper. Dostopno na: https://www.ifla.org/files/assets/hq/topics/e-lending/documents/ifla-elending-background-paper-aug-2014-rev.pdf. Dostopano 5.10.2017.

[2] M. Ločniškar-Fidler and T. Fidler, *ELEKTRONSKA KNJIGA – KNJIGA BREZ PAPIRJA*. Zveza bibliotekarskih društev Slovenije, 2003.

[3] S. Zadravec, T. Buzina, and D. Seiter-Šverko. (2014) E-book ingest module at the national and university library in zagreb. Dostopno na: http://library.ifla.org/860/1/087-klarin-en.pdf. Dostopano 28.12.2017.

[4] (2017) Epub. Dostopno na: https://en.wikipedia.org/wiki/EPUB". Dostopano 22.11.2017.

[5] (2012) Kindle. Dostopno na: https://en.wikipedia.org/wiki/Kindle_File_Format. Dostopano 6.5.2017.

[6] (2017) Comparison of e-book formats. Dostopno na: https://en.wikipedia.org/wiki/Comparison_of_e-book_formats. Dostopano 5. 6. 2017.

[7] (2017) Comic book archive. Dostopno na: https://en.wikipedia.org/wiki/Comic_book_archive. Dostopano 6.7.2017.

[8] M. Gaventa, *Learning Android*. O'Reily publishing, 2011.

[9] (2018) Platform architecture. Dostopno na: https://developer.android.com/guide/platform/index.html. Dostopano 1.3.2018.

[10] (2018) Android runtime. Dostopno na: https://en.wikipedia.org/wiki/Android_Runtime. Dostopano 28.2.2018.

[11] (2018) Art and dalvik. Dostopno na: https://source.android.com/devices/tech/dalvik/. Dostopano 28.2.2018.

[12] (2018) Closer look at android runtime: Dvm vs art. Dostopno na: https://android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924. Dostopano 28.2.2018.

[13] (2018) Application fundamentals. Dostopno na: https://developer.android.com/guide/components/fundamentals.html. Dostopano 28.2.2018.

[14] (2018) The activity lifecycle. Dostopno na: https://developer.android.com/guide/components/activities/activity-lifecycle.html. Dostopano 2.3.2018.

[15] (2018) Understanding the activity lifecycle. Dostopno na: http://www.informit.com/articles/article.aspx?p=2168079&seqNum=4". Dostopano 2.3.2018.

[16] J. Talbot and J. Mclean, *Learning Android Application Programming*. Addison-Wesley Publishing, 2013.

[17] (2018) The activity lifecycle. Dostopno na: https://developer.android.com/guide/components/activities/activity-lifecycle.html. Dostopano 2.3.2018.

[18] (2018) Meet android studio. Dostopno na: https://developer.android.com/studio/intro/index.html. Dostopano 3.3.2018.

[19] (2018) Android studio. Dostopno na: https://en.wikipedia.org/wiki/Android_Studio. Dostopano 3.3.2018.

[20] (2018) Everything you need to build on android. Dostopno na: https://developer.android.com/studio/features.html. Dostopano 3.3.2018.

[21] (2018) Android sdk tutorial for beginners. Dostopno na: https://www.androidauthority.com/android-sdk-tutorial-beginners-634376/. Dostopano 4.3.2018.

[22] (2018) Android software development. Dostopno na: https://en.wikipedia.org/wiki/Android_software_development. Dostopano 4.3.2018.

[23] (2018) Oxygen xml editor. Dostopno na: https://en.wikipedia.org/wiki/Oxygen_XML_Editor. Dostopano 4.3.2018.

[24] (2018) Adobe dreamweaver. Dostopno na: https://en.wikipedia.org/wiki/Adobe_Dreamweaver. Dostopano 4.3.2018.

[25] (2018) Developers. Dostopno na: https://developer.android.com/index.html. Dostopano 1.3.2018.