Razvoj algoritma za generiranje online obrasca na temelju XML predloška

Nagy, Daniel

SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET

DANIEL NAGY

RAZVOJ ALGORITMA ZA GENERIRANJE ONLINE OBRASCA NA TEMELJU XML PREDLOŠKA

DIPLOMSKI RAD

Zagreb, 2016.

Sveučilište u Zagrebu
Grafički fakultet

DANIEL NAGY

RAZVOJ ALGORITMA ZA GENERIRANJE ONLINE OBRASCA NA TEMELJU XML PREDLOŠKA

DIPLOMSKI RAD

Mentor:

doc. dr. sc. Tibor Skala

Student:

Daniel Nagy

Zagreb, 2016.

ZAHVALA

Veliku zahvalnost dugujem svom mentoru doc. dr. sc. Tiboru Skali i asistentu Vladimiru Cviljušcu, mag. ing., koji su svojim stručnim savjetima, strpljenjem i susretljivošću oblikovali ideju i pomogli u izradi ovog diplomskog rada.

Posebnu zahvalnost iskazujem svojoj obitelji koja mi je bila najveća potpora tijekom čitavog mog školovanja i bez čije podrške sve ovo što sam postigao ne bi bilo moguće.

Također, želim se zahvaliti i svim kolegama, prijateljima i cimerima na nesebičnosti i prijateljstvu tijekom studiranja.

SAŽETAK

Polaganje državne mature za srednjoškolce zahtijeva puno rada i učenja, a u ovom diplomskom radu razvijena je web aplikacija koja će kroz interaktivan i zabavan način pomoći učenicima da se što bolje pripreme za ispite mature. Glavno svojstvo aplikacije je ažuriranje baze pitanja i generiranje novih obrazaca od strane administratora bez potrebe za promjenom programskog koda. Pri izradi korištene su najnovije tehnologije koje su potrebne za izradu jedne ovakve web aplikacije. U teorijskom dijelu rada opisane su sve te tehnologije s naglaskom na programski jezik ActionScript 3.0 i XML preko kojih će se dohvaćati i generirati *online* obrasci odnosno pitanja u kvizu. Također, navedeno je i razvojno okruženje potrebno za izradu jedne ovakve aplikacije. U eksperimentalnom dijelu rada objašnjen je cijeli postupak izrade aplikacije, od ideje pa sve do same distribucije. Prije izrade same web aplikacije obavljeno je ispitivanje tržišta odnosno istraženo je postoje li već slične web ili mobilne aplikacije za pomoć maturantima pri učenju za ispite državne mature. Nad pronađenim aplikacijama izvršena je analiza i međusobna usporedba te su opisane njihove prednosti i nedostatci, a rezultati istraživanja su upotrijebljeni pri izradi korisničkog sučelja i funkcionalnosti aplikacije.

Ključne riječi:

- Državna matura
- Interaktivno učenje
- XML obrasci
- Actionscript 3

ABSTRACT:

For high school students, passing the state matura exam requires a lot of hard work and studying. In this thesis it has been developed a web application which will through an interactive and interesting way help students to prepare themselves as best as possible for their matura state exam. Main feature of this application is updating the question database and generating new patterns by the administrator without needing to change the application code. Latest technologies, required for making this type of application, were used during this application's development. All of those technologies, with an emphasis on Action Script 3.0 programming language, are described in theoretical part of the thesis. From this programming language online forms and question will be retrieved and generated in the quiz. Furthermore, development enviroment required for making this type of application, is specified. The whole procedure of application's development, from idea to distribution, is explained in the experimental part of thesis. However, before creating web application itself, application market had been explored in search for similar web or mobile applications that could also help students with preparing and learning for the state matura exams. Found applications were analyzed and mutually compared, their pros and cons were described, and results of research have been used in the process of designing user interface and other funcionalities of the application.

Keywords:

- State Matura
- Interactive learning
- XML forms
- ActionScript 3

SADRŽAJ

1. UVOD

Državna matura je skup ispita iz određenih nastavnih predmeta koje je učenik učio tijekom svoga najmanje četverogodišnjega srednjoškolskog obrazovanja i ona označava završetak srednjoškolskog obrazovanja u gimnazijskim programima. Učenici četverogodišnjih strukovnih i umjetničkih srednjih škola također mogu polagati ispite državne mature. Državna matura provodi se polaganjem ispita obveznog i izbornog dijela na jednoj od dviju razina i to na višoj (A) i osnovnoj (B) razini. Sva visoka učilišta u Republici Hrvatskoj prihvatila su rezultate ispita državne mature kao jedan od uvjeta za rangiranje kandidata za upis na studijske programe. [1]

U današnje vrijeme, na tržištu rada se sve češće kao osnovni uvjet za zapošljavanje traži visokoškolsko obrazovanje što svake godine rezultira sve većom zainteresiranosti za polaganjem državne mature. Polaganje ispita državne mature važan je korak u životu srednjoškolaca i zahtjeva puno rada i učenja. Svaki učenik se na svoj način priprema za te ispite, a uglavnom je to čitanje raznih knjiga, skripti, natuknica i slično. Ipak, u današnjem modernom, digitalnom svijetu, gdje brzina dolaska do informacije ovisi jedino o brzini internetske veze, postoje i drugačiji načini učenja. Tradicionalno učenje iz knjiga sve više zamjenjuje interaktivno učenje putem mobilnih i web aplikacija. Napredak tehnologije ponudio je učenicima interaktivniji odnosno kvalitetniji način učenja od standardnog čitanja knjiga, a cilj ovog diplomskog rada bio je razviti i opisati postupak izrade web aplikacije koja na interaktivniji i zabavniji način olakšava učenje za ispite državne mature. Web aplikacija je izrađena u obliku kviza nalik na popularne televizijske kvizove i kvizove dostupne na internetu. Nakon odgovorenog posljednjeg pitanja, korisnik kao rezultat dobiva omjer točnih i netočnih odgovora, postotak riješenosti i vrijeme potrebno za rješavanje ispita. Ona korisnicima omogućava brzo međudjelovanje tako da se pomoću XML obrasca omogućava stvaranje novih obrazaca od strane administratora bez potrebe za promjenom programskog koda. U radu je opisano razvojno okruženje potrebno za izradu jedne takve aplikacije.

Aplikacija je dostupna na https://goo.gl/3qFIoL.

2. RAZVOJNO OKRUŽENJE

2.1. Internet

Iako je za većinu pojam Interneta zbunjujući, objašnjenje je u suštini vrlo jednostavno. On je svjetski zbornik računalnih mreža tzv. *mreža svih mreža* gdje se digitalne informacije dijele putem mrežnih i softverskih protokola. Ono što je jedinstveno za Internet je koji god računalni sustav koristili pri spajanju na Internet, računala komuniciraju putem identičnih internetskih protokola što nam omogućuje razmjenu informacija i datoteka s računalima diljem planete Zemlje. [2]

Internet je nastao krajem 60-ih prošlog stoljeća u Sjedinjenima Američkim Državama kao eksperiment izgradnje snažne mreže između računala koja bi izdržala oštećenje mreže s mogućnošću nastavka komunikacije između računala koja su ostala u funkciji. Taj eksperiment je doživio ogroman uspjeh, ali glavni nedostatak te snažne mreže je njezina ograničena veličina odnosno djelokrug. Njome su se u početku mogle koristiti jedino akademske institucije i ministarstvo obrane, a tada je zvana ARPAnet (*Advanced Research Projects Agency Network of the Department of Defense*). [2]

Rastom ARPAnet-a 70-ih godina raste i broj povezanih računala, što je rezultiralo veliki rast i broja mreža koje se povezuju u jednu zajedničku mrežu te sukladno tome dolazi do promjene naziva u *Internet*. Naziv Internet dolazi od riječi Interconnected što u prijevodu znači međusobno povezani i riječi *Networks* što znači mreže, mrežni sustav. Nagli razvoj Interneta počinje početkom 90-ih godina prošlog stoljeća nakon što je britanski fizičar Tim Berners - Lee predložio ideju internet servisa koja se bazira na poveznicama. Na tom servisu su smještene sve internetske stranice, a ono što je potaknulo nagli razvoj Interneta je mogućnost pristupa servisu za sve ljude. U današnje vrijeme mnogi poistovjećuju naziv Internet i World Wide Web što nije točno. WWW je samo jedna od usluga koje se nalaze na Internetu. Među ostalim najpoznatijim usluga su elektronička pošta, čavrljanje, prijenos podataka itd.

2.2. HTML jezik

Za stvaranje dokumenta odnosno internetskih stranica na World Wide Webu koristi se prezentacijski jezik HTML, a počeo se koristiti od samih početaka WWW-a. Kratica HTML dolazi od engleskog izraza *HyperText Markup Language*. HTML jezikom određuje se struktura, oblikuje se sadržaj i stvaraju se hiperveze dokumenata.

Hiperveze dokumenata su tekstualne strukture koje se sastoje od međusobno povezanih jedinica informacije prikazana na nekom elektroničkom uređaju. Za razliku od jednostavnog tradicionalnog teksta, hipertekst nema jedinstven redoslijed čitanja, nego ga čitatelj dinamički određuje, tj. određuje ga tijekom čitanja. Zato kažemo da je tradicionalni tekst sekvencijalan, a hipertekst nesekvencijalan. [4] HTML dokument je moguće stvoriti najobičnijim uređivačem teksta kao što je Notepad, a dokument mora sadržavati ekstenziju .html ili .htm. Niz HTML dokumenata predstavlja web sjedište kojem se može pristupiti preko web preglednika (eng. *browser*). Osnovna zadaća HTML-a je uputiti preglednik kako prikazati dokument, a pri tome se nastoji da taj dokument izgleda jednako u svim preglednicima.

Prva verzija HTML-a objavljena je 1993. godine. Trenutna verzija HTML-a je HTML 5, koja se razvija od 2008. godine, a postala je konačnom preporukom World Wide Web Consortiuma (W3C) u 2014. godine. Podržavaju ga svi moderni preglednici. HTML 5 dodatno naglašava semantičku stranu HTML-a pa su dodani novi elementi poput *header*, *footer, article* itd. [4]

Većina ljudi HTML zove programskim jezikom, a shodno tome ljude koji ga koriste zovu programerima što je pogrešno iz razloga što ne sadrži ni najjednostavnije funkcije poput osnovnih matematičkih operacija. Kao što je već spomenuto, HTML je opisni jezik. HTML je jednostavan za upotrebu i lako se uči iz čega proizlazi njegova popularnost.

2.2.1. HTML sintaksa

Sav sadržaj unutar HTML dokumenta, kojeg želimo prikazati na pregledniku, okružuju oznake (eng. *tags*). HTML oznake se nalaze unutar šiljatih zagrada i najčešće dolaze u paru. Prva oznaka u paru je početna oznaka, a sljedeća je završna i ona mora sadržavati znak / nakon prve šiljate zagrade. Početna i završna oznaka zajedno tvore HTML element. Također ih zovemo oznake otvaranja i zatvaranja po uzoru na englesko nazivlje (*opening* i *closing tags*). Između tih oznaka *web developeri* mogu dodavati tekst, komentare i drugi sadržaj na temelju teksta: <oznaka>sadržaj</oznaka>. Primjer jednostavnog HTML elementa možemo vidjeti u ispisu broj 1. Također, postoje i pojedinačne oznake koji se sastoje samo od početne oznake. Najjednostavniji primjer takve oznake je
 koja predstavlja prelazak u novi red. [5]

```
<h1> Ovo je naslov </h1>
```

Ispis 1 – Isječak koda: HTML element

HTML oznake i atributi mogu se pisati malim i velikim slovima, međutim preporuka World Wide Web Consortiuma (poznatija kao W3C), organizacije koja se bavi standardizacijom tehnologija korištenih na webu, je da se elementi pišu mali slovima. [6]

2.2.2. Struktura HTML dokumenata

HTML dokument svojim izgledom podsjeća na obiteljsko stablo. Svaki element može sadržavati druge elemente i to ga čini *parent* elementom, dok je element unutar njega *child* element. Dakako, grananje može ići još dalje pa tako *child* element može biti *parent* element nekom drugom *child* elementu. Prije svega potrebno je napisati *doctype* deklaraciju koja omogućava pregledniku da prepozna tip i inačicu dokumenta koji prikazuje. Svaki HTML dokument se sastoji od 2 dijela: zaglavlja (eng. *head*) i tijela (eng. *body*), a potrebno ih je smjestiti unutar <html> elementa. Sve ono što napišemo u zaglavlju dokumenta, unutar <head></head> oznaka, neće se prikazati u pregledniku već služi samo da pruži informacije o stranici npr. naslov stranice. Valja napomenuti da to nije naslov koji će se prikazati kao sadržaj web stranice već kao tekst u naslovnoj liniji preglednika. S druge strane, sve što napišemo unutar tijela tj. unutar <body></body> elementa predstavlja tijelo dokumenta i pojavit će se kao sadržaj web stranice u prozoru preglednika. [5]

```
<!DOCTYPE html>
<html>
    <head>
        <title>Naslov</title>
    </head>
    <body>
        <h1>Ovo je naslov</h1>
        <p>Ovo je paragraf teksta.</p>
    </body>
</html>
```

Ispis 2 – Isječak koda: Jednostavan HTML dokument

Valjanost HTML dokumenta provjeravamo na https://validator.w3.org/.

2.2.3. Upotreba HTML jezika pri izradi aplikacije

HTML jezik se nije direktno koristio pri izradi aplikacije. Nakon izvoza (eng. *export*) aplikacije dobivamo nekoliko izlaznih datoteka među kojima se nalazi i HTML dokument. Taj HTML dokument je zadužen za učitavanje *flash* aplikacije odnosno koji će učitati *flash* (.swf) dokument.

2.3. CSS tehnologija

HTML određuje strukturu dok je CSS (*Cascading Style Sheets*) zadužen za prezentacijski aspekt stranice – opisuje kako će se pojedini HTML elementi prikazati (eng. *render*) na zaslonu, papiru ili u drugim medijima. [4] Dakle, HTML jezikom oblikuje tj. opisuje se sadržaj i stvaraju se hiperveze dokumenta, a izgled se definira putem CSS stilova. Sintaksa pisanja CSS-a je vrlo jednostavna i sastoji se od selektora koji je najčešće element, ID ili klasa elementa i deklaracija koja se definira u obliku { svojstvo:vrijednost; }. Kao što se može primijetiti sintaksa CSS-a je vrlo jednostavna, potrebno je samo zapamtiti i poznavati CSS svojstva. Vrlo bitno je ne zaboraviti znak točka-zarez koji označava završetak naredbe. U sljedećem ispisu možemo vidjeti kako smo paragraf elementu definirali veličinu fonta od 16 piksela.

```
p { font-size:16px; }
```

Ispis 3 – Isječak koda: Sintaksa definiranje stila u CSS tehnologiji

Vrlo važni CSS selektori su ID i klasa. Selektor ID koristimo za definiranje svojstva samo jednog elementa, dok se klase koriste za definiranje svojstva više elemenata unutar istog dokumenta.

```
<!DOCTYPE html>
<html>
        <head>
                <style type="text/css">
                        #plavo {color: blue;}
                        .zeleno {color: green;}
                </style>
        </head>
        <body>
                <h1 id=“plavo“> Ovaj naslov je plave boje </h1>
                <h1 class=“zeleno“> Ovaj naslov je zelene boje </h1>
                <h1 class=“zeleno“> Ovaj naslov je također zelene boje </h1>
        </body>
</html>
```

Ispis 4 – Isječak koda: CSS selektori - ID i klasa

CSS je moguće implementirati na tri načina pa tako može biti pisan unutar atributa *style* pojedinog elementa (tzv. *inline* CSS), unutar oznaka <style></style> u zaglavlju HTML dokumenta te u dodatnoj datoteci koja sadrži .css ekstenziju. Implementacija vanjske CSS datoteke je vrlo jednostavna. U zaglavlje HTML dokumenta, unutar *head* elementa potrebno je dodati liniju koda prikazanu u ispisu 6. Vrijednost *href* atributa je adresa gdje se nalazi vanjska CSS datoteka. Zbog preglednosti i odvajanja stila od sadržaja, najčešće se koriste dodatne, vanjske datoteke. Ipak, *inline* i CSS unutar HTML-a imaju veći prioritet od vanjskog.

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

Ispis 5 – Isječak koda: Linija koda za implementaciju vanjske CSS datoteke

2.4. XML jezik

XML je kratica od *Extensible Markup Language* što bi u prijevodu značilo proširivi jezik za označavanje. XML-om zapisujemo dokumente i podatke u tekstualnom formatu čitljiv za uređivače teksta kao što su npr. MS Word, Notepad i slični. XML dokument se najčešće sprema kao tekstualna datoteka koja se sastoji od sadržaja i oznaka. Iako je prvenstveno namijenjen za programsku obradu, XML format je čitljiv i ljudima. [7]

Kao i HTML, XML pripada obitelji jezika za označavanje i za njihovu se standardizaciju brine W3C. Oba jezika opisuju sadržaj dokumenta tj. datoteke pomoću oznaka. XML dijeli neke zajedničke osobine sa HTML-om, ali je dizajniran za drugačije potrebe i nije izravna zamjena za HTML. Razlika između ta dva jezika je u tome što se XML orijentira na to što je podatak dok se HTML orijentira na to kako podatak izgleda. Osnovna ideja koja stoji iza XML-a je bila stvoriti jezik za pohranu i razmjenu gotovo svih vrsta podataka koji će i ljudi i računalni programi jednostavno čitati, a uz to je i potpuno neovisan o računalnoj platformi na kojoj se nalazi te o operacijskom sustavu kojeg koristimo. XML je otvoreni standard čija je specifikacija javna i dostupna svima.

2.4.1. Povijest XML jezika

Šezdesetih godina prošlog stoljeća IBM je imao velik problem s ogromnom količinom različite tehničke dokumentacije koja se za svaku posebnu namjenu morala prepisivati i nanovo uređivati za što su trošili ogromnu količinu ljudskih resursa. Ideja do koje su došli bila je prvi šire korišteni jezik za označavanje podataka. Korisni sadržaj uokvirio bi se određenim oznakama koje će ga opisivati. Nakon što se to učini, za određene namjene jednostavno će se povlačiti sadržaji određenog tipa. GML (*Generalized Markup Language*) oznake opisivale su određene dijelove dokumenta kao npr. poglavlja, liste, tablice itd. Korištenjem GML-a iz istog sadržaja mogla se dobiti ispisana različita dokumentacija, tehnička, ali i korisnička. [8]

GML se pokazao kao uspješan proizvod tako da je nastavljen razvoj u tom smjeru. 80-ih godina 20. stoljeća American National Standards Institute (ANSI) radio je na razvoju

standarda jezika za označavanje podataka. Nastali jezik nazvan je Standard Generalized Markup Language (SGML) i 1986. objavljen je kao međunarodna norma ISO 8879. Problem SGML-a, kao i mnogih sličnih proizvoda koji su razvijani na takav način od strane organizacija za standardizaciju, je što je bio golem, opširan i složen za korištenje. Zbog tih osobina SGML nije bio jako raširen u praksi i korisnici SGML-a bile su uglavnom velike kompanije, državne službe i znanstvene institucije. [8]

S paralelnim razvojem HTML-a, SGML je i dalje bio bolji izbor za opisivanje sadržaja, ali problem je bio taj što je bio preopširan za korištenje i izvršavanje u web pregledniku. Javila se potreba za kreiranjem jezika koji će biti dovoljno malen za izvršavanje u pregledniku, a s druge strane dovoljno prilagodljiv da se može proširivati korisničkim oznakama. XML je zadržao značajku SGML-a da se njime mogu opisivati novi jezici, ali je izostavljeno sve ono što je komplicirano za razumjeti i teško za implementirati. Time bi se objedinila jednostavnost HTML-a i snaga SGML-a. Pri razvoju XML-a određeno je deset ciljeva:

1. XML mora biti izravno primjenjiv preko interneta.
2. XML mora podržavati širok spektar primjena.
3. XML mora biti kompatibilan sa SGML-om.
4. Postupak pisanja programa za procesiranje XML dokumenata mora biti jednostavan.
5. Broj dodatnih svojstava u XML-u mora biti minimalan, u idealnom slučaju jednak nuli.
6. XML dokumenti bi trebali biti čitljivi i jasni ljudima.
7. Izrada XML-a bi trebala biti brza.
8. Izrada XML-a mora biti formalna i jezgrovita.
9. Izrada XML dokumenata mora biti jednostavna.
10. Sažetost u strukturi XML-a je od minimalne važnosti.

1998. godine objavljena je prva verzija XML-a od strane W3C-a, a trenutni standardi su peto izdanje 1.0 inačice i drugo izdanje 1.1 inačice.

2.4.2. Sintaksa i struktura XML dokumenta

XML dokument se sastoji od jednog ili više elemenata. Elementi koji nisu prazni označuju se na već poznati način, sadržaj elementa je okružen s dvije oznake,

početne i završne oznake. Naziv elementa se nalazi unutar znakova manje od (<) i više od (>). Završna oznaka se razliku od početne po tome što sadrži znak / nakon znaka manje od i prije imena elementa. Kao i kod HTML-a, sadržaj između oznaka se smatra dijelom elementa te se obrađuje sukladno s pravilima tog elementa. Za razliku od HTML-a, XML oznake nisu unaprijed definirane. Ipak, postoji nekoliko pravila koja se odnose na imenovanje XML elemenata:

- Imena mogu sadržavati slova, brojeve i druge znakove.
- Imena ne smiju počinjati brojem ili interpunkcijskim znakom.
- Imena ne smiju počinjati slovima „xml“.
- Imena ne smiju imati razmak u sebi.

Također, važno je napomenuti da su u XML-u oznake osjetljive na velika i mala slova (eng. *case-sensitive*) što znači da <mojaoznaka> nije isto što i <MojaOznaka>. Pravilnu sintaksu XML elementa možemo vidjeti u sljedećem primjeru.

```
<oznaka> Sadržaj elementa </oznaka>
```

Ispis 6 – Isječak koda: Sintaksa XML elementa

XML dokument se sastoji od dva dijela, zaglavlja i sadržaja dokumenta. U zaglavlju XML dokumenta se navodi XML deklaracija, tj. podaci koji opisuju dokument, poput inačice koja se koristi, a poželjno je navesti tip znakova koje koristimo u zapisu dokumenta, pogotovo zato što naš jezik ima puno dijakritika koji bi se bez te deklaracije krivo prikazivali. Zaglavlje XML dokumenta je opcionalno, ali ako postoji mora se nalaziti na samom početku dokumenta.

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
```

Ispis 7 – Isječak koda: Zaglavlje XML dokumenta

Ono što svaki XML mora imati je jedan korijenski (eng. *root*) element, element unutar kojeg su svi ostali elementi. Prilikom izrade dokumenta važno je da su svi elementi pravilno ugniježđeni. Da bi neki dokument bio XML dokument moraju biti zadovoljeni minimalni uvjeti koji se skraćeno nazivaju "dobro formiran" (eng. *well formed*).

Valjanost XML dokumenta provjeravamo na službenoj stranici W3C-a - https://validator.w3.org/.

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<imenik>
  <kontakt vrsta=“posao“>
    <ime>Ivan</ime>
    <prezime>Radoš</prezime>
    <brojMobitela>099123456</brojMobitela>
  </kontakt>
  <kontakt vrsta=“fakultet“>
    <ime>Luka</ime>
    <prezime>Seidl</prezime>
    <brojMobitela>098654321</brojMobitela>
  </kontakt>
</imenik>
```

Ispis 8 – Isječak koda: Jednostavan XML dokument

U gore navedenom primjeru možemo vidjeti jedan jednostavan XML dokument koji predstavlja adresar u kojeg spremamo kontakte. Prva linija koda predstavlja zaglavlje odnosno deklaraciju, a element <imenik> je korijenski element ovog XML dokumenta. Osim elemenata i sadržaja, u ovom dokumentu možemo primijetiti i atribute. Atributi se pridružuju elementima kako bi pružili dodatne informacije o svojstvima elemenata. Oni imaju svoj naziv i vrijednost koja se upisuje pod navodnicima. U ispisu broj 8 možemo primijetiti atribute imena *vrsta* koji nam opisuju o kakvoj vrsti kontakta se radi u dokumentu.

2.4.3. XML imenski prostori

Iz razloga što XML jezik dozvoljava autorima da stvaraju vlastite elemente, često dolazi do sukoba u imenovanju između dva elementa istog imena. Na primjer, jedna osoba može stvoriti element naziva *predmet* i kao njegov sadržaj unijeti naziv školskog predmeta npr. matematika. Druga osoba također stvori element istog naziva, ali kao sadržaj unese kuhinjski predmet npr. nož. Ako se oba elementa nađu u istom dokumentu doći će do sukoba imenovanja odnosno neće se moći odrediti koju vrstu

podataka element zahtijeva. XML imenski prostori (eng. *namespace*) su stvoreni kako bi osigurali jedinstvenost između XML elemenata te time sprječavaju koliziju imenovanja. [7]

Imenski prostori nisu obavezni, ali ih je pametno koristiti. Oni nam omogućavaju da spajamo dijelove jednog XML dokumenta s dijelovima drugog XML dokumenta. Imenski prostori se međusobno razlikuju jedinstvenim identifikatorom, a to je jednostavnije rečeno njihovo ime. U primjeru iz prošlog paragrafa bismo tako mogli stvoriti dva imenska prostora, jedan za školske predmete i jedan za kuhinjske predmete.

```
<škola:predmet> Matematika </škola:predmet>
<kuhinja:predmet> Nož </kuhinja:predmet>
```

Ispis 9 – Isječak koda: Korištenje imenskih prostora u XML jeziku

U ispisu 9 možemo vidjeti na koji način se koriste imenski prostori unutar XML dokumenta. Za oba elementa smo dodali prefiks elementu *predmet* s imenom imenskog prostora kojeg želimo koristiti. Imena imenskog prostora se formiraju prema konvenciji o web adresama. Ta imena ne moraju pokazivati na neku stvarnu web stranicu.

```
<predmeti xmlns=“www.mojastranica.hr/skola“
xmlns:kuhinja="www.mojastranica.hr/kuhinja">
  <predmet>Matematika<predmet>
  <predmet>Geografija<predmet>
  <kuhinja:predmet>Nož</kuhinja:predmet>
  <kuhinja:predmet>Vilica</kuhinja:predmet>
</predmeti>
```

Ispis 10 – Isječak koda: Pozivanje i korištenje imenskog prostora

U ispisu 10 vidimo kako smo pozvali dva imenska prostora: *www.mojastranica.hr/skola* i *www.mojastranica.hr/kuhinja*. Ispred prvog je namjerno izostavljen prefiks *škola* pa se on smatra početnim (eng. *default*) imenskim prostorom.

Dakle, za svaki element kojemu je izostavljen prefiks pripada imenskom prostoru *škola*, dok element s prefiksom *kuhinja*, pripada imenskom prostoru *kuhinja*.

2.4.4. Provjera ispravnosti XML dokumenta

Osim formalne ispravnosti, ispravnost XML dokument se može kontrolirati u odnosu na propisanu shemu. Shema određuje koje elemente smije sadržavati XML dokument te redoslijed i broj tih elemenata. Ispravnost se provjerava u odnosu na određenu DTD ili XML shemu. DTD je kratica od *Document Type Description*, a odnosi se na dio XML dokumenta koji detaljnije opisuje i ograničava dozvoljeni sadržaj XML dokumenta [8]

```
<!DOCTYPE imenik [
 <!ELEMENT imenik (kontakt)* >
 <!ELEMENT kontakt (ime, prezime, brojMobitela)>
 <!ELEMENT ime (#PCDATA) >
 <!ELEMENT prezime (#PCDATA) >
 <!ELEMENT brojMobitela (#PCDATA) >
 <!ATTLIST kontakt vrsta CDATA #REQUIRED >
]>
```

Ispis 11 – Isječak koda: DTD Schema XML dokumenta

Gore navedeni ispis opisuje dozvoljeni sadržaj XML dokumenta. Shema je rađena po XML dokumentu u ispisu broj 8. U ovoj shemi se određuje da XML dokument mora započeti s korijenskim elementom naziva *imenik* i on može sadržavati nula ili više elemenata naziva *kontakt*. Nadalje, element *kontakt* mora sadržavati tri elementa: *ime*, *prezime* i *brojMobitela.* Sadržaj tih elemenata može biti bilo kakav tekst, a to smo definirali ključnom riječi #PCDATA. U zadnjoj liniji koda smo definirali atribut naziva *vrsta* za element imena *kontakt*, ključnom riječi #REQUIRED smo zadali da je on neophodan tom elementu. DTD shema se ne mora nalaziti u istoj datoteci s XML dokumentom već se može definirati u zasebnoj datoteci.

Novija tehnika za opis dozvoljenog formata XML dokumenta je XML Schema. Za razliku od DTD-a, XML Schema je pisana kao dobro formiran XML dokument. XML Schema je tekstualni opis koji definira dozvoljenu strukturu XML dokumenata i ima istu funkciju kao i DTD, ali je novija i opsežnija od DTD-a. [8] XML Schema pruža više

mogućnosti od DTD-a, a jedna od njih određivanje tipa sadržaja koji se može unijeti u pojedini element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="imenik">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="kontakt"  maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
          <xs:element name="ime" type="xs:string"/>
              <xs:element name="prezime" type="xs:string"/>
       <xs:element name="brojMobitela" type="xs:positiveInteger"/>
         </xs:sequence>
      <xs:attribute name="vrsta" type="xs:string" use="required" />
        </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Ispis 12 – Isječak koda: XML shema

Kao što možemo primijetiti u gornjem ispisu, XML Schema ima XML deklaraciju, jedan korijenski element <xs:schema> i definira svoj imenski prostor s prefiksom *xs*. Shema u ovom ispisu određuje da dokument mora započeti s elementom *imenik* koji može sadržavati beskonačan broj elemenata *kontakt*. Svaki element *kontakt* mora sadržavati tri elemenata: *ime* i *prezime* kojima je zadano da su tipa *string* (niz znakova) te element *brojMobitela* kojemu je zadano da je tipa *positiveInteger* što znači da se kao sadržaj tog elementa može upisati samo pozitivan broj i ništa drugo.

Upravo XML Schema i DTD su mehanizmi pomoću kojih korisnici mogu striktno odrediti vlastite formate zapisa podataka i dokumenata. Ovi striktni opisi dozvoljenih formata su ključni za strojnu obradu razmijenjenih podataka. Programi mogu provjeriti

da li je primljena informacija u određenom formatu i tako izbjeći potencijalne greške. [8]

2.4.5. Budućnost XML jezika

Budućnost XML jezika je i dalje nejasna zbog konfliktnih mišljenja između korisnika XML-a. U zadnjih pet godina dogodila su se značajna postignuća. XML je omogućio upravljanje velikom količinom informacija koja nije kompaktna s bazama podataka. Zbog silnih promjena XML nije više tako jednostavan, a potrebno je i puno više vremena za izradu XML alata. Prednost XML-a nad ostalim tehnologijama je i dalje njegova neovisnost, otvorenost i prenosivost. Jedan dio korisnika zagovara prestanak nadogradnje i pojednostavljivanje, dok se drugi korisnika zalaže za daljnji napredak te čak traže novu verziju, XML 2.0. Budućnost XML-a leži u web izdavaštvu. Danas su u preglednicima integrirane videoigre, alati za obradu teksta itd. Paralelno s razvojem weba razvijat će se i XML. [10]

2.5. Actionscript 3.0 i objektno orijentirano programiranje

Programiranje je fazni postupak stvaranja računalnog programa odnosno softvera. Faze programiranja su:

1. Definiranje problema koji se želi riješiti
2. Stvaranje algoritma
3. Pisanje programa (kodiranje)
4. Provjera ispravnosti (testiranje) programa
5. Dokumentiranje programa
6. Implementiranje, odnosno puštanje programa u rad
7. Korištenje i održavanje programa

Objektno orijentirano programiranje (OOP) je jedan od osnovnih programerskih koncepata koji svaki programer mora savladati kako bi bio konkurentan na tržištu rada. Pojedini programski jezici se razlikuju po sintaksi, međutim koncept objektno orijentiranog programiranja primjenjiv je na skoro svaki programski jezik. OOP je metoda programiranja kojoj je temeljni princip da se klasa definira kao samostalna programska cjelina odnosno postupak izrade programa upotrebom skupa objekata koji međusobno razmjenjuju poruke. Osnovna svojstva OOP-a su klase, objekti, nasljeđivanje, inkapsulacija i polimorfizam.

ActionScript 3.0 (AS3) je objektno orijentiran programski jezik koji je neovisan o hardveru i softveru. Prevedeni (eng. *compiled*) Actionscript kod se može izvršiti na svakoj platformi (hardver + operacijski sustav) na kojoj je instaliran Adobe AIR *runtime*. Razlika između aplikacija za određenu platformu je u samom načinu na koji se izvršava aplikacija. Desktop aplikacija je program koji se izvršava na računalu, mobilne aplikacije se izvršavaju na mobilnom uređaju dok se web aplikacije izvršavaju u web pregledniku.

ActionScript programski jezik nam služi za stvaranje svih vrsta interaktivnih aplikacija smještene na webu. Ovo su samo neke od mogućnosti: MP3 svirač, aplikacije za crtanje, 3D šetnja kućom, online trgovina, ploča s porukama, HTML uređivač, Pac-Man igra itd. Pomoću AS3 jezika možemo stvarati interaktivne elemente kao što su gumbi, padajući izbornik, razne liste, *checkboxeve*, sadržaj koji se animira u skladu s kretnjama miša, obrasce i mnoge druge. [11]

2.5.1. Objekti i klase

Objektno orijentirano programiranje je način programiranja koji se fokusira na korištenju objekata za dizajn i izradu aplikacija. Zamislimo objekt kao primjerak ideje, zbivanja ili stvari iz stvarnog života koji su značajni za vašu aplikaciju. Svaki od tih objekata ima dvije karakteristike: ponašanje (funkcije) i stanje (varijable). Oni su temeljni gradivni blokovi aplikacija. [12]

Klase su predlošci ili set instrukcija koji nam služe za stvaranje objekata iste vrste. Svaki objekt nastaje iz klase. Svi objekti imaju isti niz funkcija i stanja naslijeđenih od klase. Svaka klasa bi trebala biti dizajnirana i programirana kako bi postigla samo jednu stvar. Budući da su klase programirane za samo jednu dužnost potrebno je mnogo klasa kako bi se izradila cijela aplikacija. Instance su specifični objekti nastali iz specifične klase. Njoj je dodijeljena referentna varijabla koja se koristi kako bismo imali pristup svim svojstvima i metodama instance. U AS3 programskom jeziku instance se stvaraju pomoću ključne riječi *new*. [12]

```
public class MojaPrvaKlasa {
        public var mojaPrvaVarijabla:int;
        public function mojaPrvaFunkcija():void {};
}
public var mojaPrvaInstanca: MojaPrvaKlasa = new MojaPrvaKlasa ();
mojaPrvaInstanca.mojaPrvaVarijabla = 3;
mojaPrvaInstanca.mojaPrvaFunkcija();
```

Ispis 13 – Isječak koda: Stvaranje instance u AS3

U ispisu 13 smo definirali klasu naziva *MojaPrvaKlasa.* Klasa se stvara tako da prvo napišemo ključnu riječ koja predstavlja doseg (eng. *scope*) klase. Nakon toga slijedi ključna riječ *class* i nakon toga ime same klase. Bitno je napomenuti da se imena klasa pišu veliki početnim slovom tzv. *CamelCase*. Unutar vitičastih zagrada definiramo varijable i funkcije naše klase. U ovom primjeru smo deklarirali varijablu imena *mojaPrvaVarijabla* koja je tipa *integer* i funkciju naziva *mojaPrvaFunkcija* koja u ovom slučaju ne radi ništa. Nakon zatvorene vitičaste zagrade tj. kraja naše klase, stvaramo instancu klase *MojaPrvaKlasa* stvarajući referentnu varijablu koju smo nazvali *mojaPrvaInstanca.* Kao i u većini OOP jezika, za pristup svojstvima novonastale instance *mojaPrvaInstanca* stvorene od klase *MojaPrvaKlasa* koristimo tzv*. dot notation* tj. pristupamo na način da nakon imena instance napišemo točku.

2.5.2. Inkapsulacija, nasljeđivanje i polimorfizam

Integracija funkcija i stanja u objekt zove se inkapsulacija (eng. *encapsulation*). Inkapsulacija omogućava skrivanje stanja i funkcija koji ne trebaju biti vidljivi vanjskim procesima i jedan je od tri glavna (uz nasljeđivanje i polimorfizam) svojstva OO programiranja.

Različiti objekti često imaju određenu količinu sličnosti s drugim objektima. Na primjer, različiti poligoni (trokut, kvadrat, pentagon) dijele iste karakteristika poligona npr. broj stranica. Ipak, svaki od tih poligona se po nekim karakteristikama razlikuje od drugih. Mehanizam pomoću kojega je na temelju postojećih klasa moguće definirati nove i proširene klase. Klasa koja nasljeđuje svojstva zove se podklasa (eng. *child class*), a klasa čija su svojstva naslijeđena se naziva superklasa (eng. *parent class*). Za

nasljeđivanje u AS3 jeziku, a i većini OOP jezika, koristi se ključna riječ *extends*. Korištenjem svojstva nasljeđivanja kod je bolje strukturiran i lakše čitljiv.

```
public class Zivotinje {
        private var brojNogu:int;
}
public class Pas extends Zivotinje {
        private function lajanje();
}
```

Ispis 14 – Isječak koda: Nasljeđivanje klasa u AS3

U ispisu 14 možemo vidjeti jednostavan primjer nasljeđivanja. Klasa *Pas* nasljeđuje klasu *Zivotinje*. U ovom slučaju klasa *Zivotinje* je super klasa, a klasa *Pas* je podklasa. Klasa *Pas* nasljeđuje svojstvo *brojNogu* jer je to svojstveno za sve životinje.

Moć poprimanja više oblika naziva se polimorfizam. Polimorfizam se koristi kada želimo koristiti različite objekte s istim nazivom funkcije, ali različitom funkcionalnošću. Primjerice, sve se životinje kreću, ali ne kreću se na isti način npr. neke trče, neke plivaju, neke skaču. Kako ne bismo različito imenovali slične funkcionalnosti, koji u osnovi imaju isti rezultat, koristimo polimorfizam.

2.5.3. Komentari

Komentari služe programerima i osobama koji će kasnije održavati program da se lakše snalaze među linijama programskog koda. U praksi je poželjno ostavljati komentar nakon svakog zahtjevnijeg bloka koda. U AS3 i većini OOP jezika postoje 3 vrste komentara.

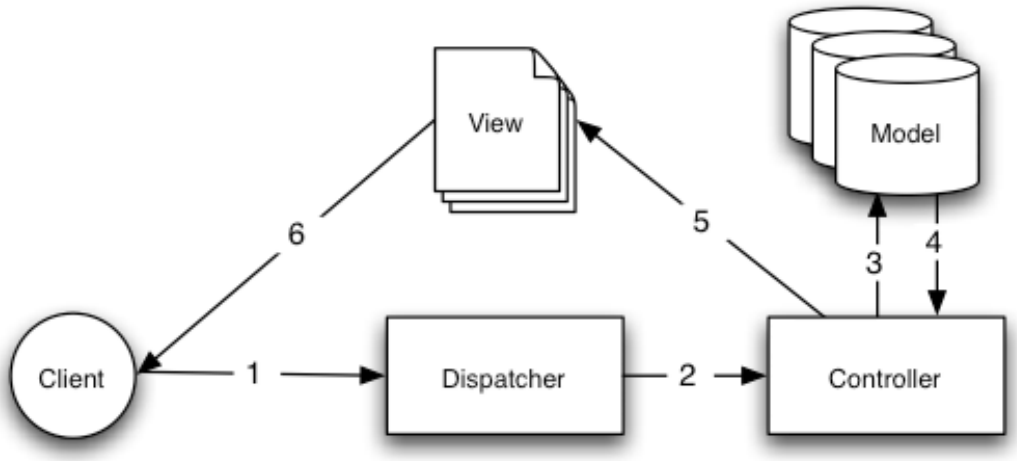
```
/** Komentar za dokumentaciju. */
// Jednolinijski komentar
/* Višelinijski
   komentar */
```

Ispis 15 – Isječak koda: Komentiranje koda u AS3

Prva vrsta komentara su komentari za dokumentaciju. Prevoditelj ignorira sve što se nađe između znakova /** i */ odnosno tu vrstu komentara. Jednolinijske komentare prevoditelj ignorira sve od znaka // do kraja tekuće linije. Kod višelinijskih komentara prevoditelj ignorira ignorira sve između znakova /* i */ kao i kod komentara za dokumentaciju.

2.5.4. MVC obrazac

MVC (*Model-View-Controller*) arhitektonski obrazac koji se koristi u programskoj potpori i najčešći je obrazac u svijetu programiranja web aplikacija. Svrha tehnologije je razdvajanje korisničkog sučelja od poslovne logike čime je u velikoj mjeri poboljšana preglednost samoga kôda, a što zapravo rezultira olakšanim održavanjem kao i nadogradnjom aplikacije. MVC struktura razdvaja aplikaciju na tri osnovna dijela – *Model, View i Controller*. Svaki dio ima svoja svojstva i zadaću koju mora izvršiti. Na kraju cijela struktura predstavlja jedan izlaz (eng. *output*) u odnosu na zahtjeve krajnjeg korisnika. [18] Grafički prikaz MVC obrasca možemo vidjeti na slici broj 1.



Slika 1: MVC arhitektura

(izvor: http://book.cakephp.org/2.0/en/cakephp-overview/understanding-model-view-controller.html)

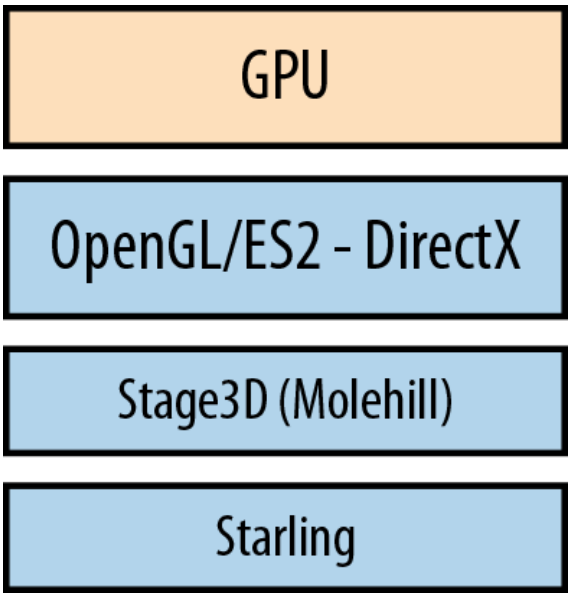
Controller (upravitelj) je zadužen za upravljanje akcijama unutar programa i odgovoran je za njegov tok. U web aplikacijama je prvi sloj koji se poziva kada preglednik pozove URL. Također, *controller* upravlja korisničkim zahtjevima (HTTP GET ili POST). Upravitelj osluškuje zahtjeve korisnika i šalje naloge *modelu* i ukoliko *model* promijeni stanje o tome obavještava *view*.

Model je odgovoran za rad s podacima poput povezivanja na bazu podataka, izvršavanje upita itd., ali se on ne može direktno pozvati. *Controller* je taj koji od *modela* zahtjeva određene podatke, *model* obrađuje zahtjeve i vraća podatke *controlleru*.

View (pogled) je posljednji sloj MVC arhitekture koji sadrži korisničko sučelje aplikacije, odnosno osigurava različite načine za prikaz podataka koje dobije od *modela*. U web aplikacijama *view* sadrži HTML, CSS, JavaScript, XML ili JSON, itd. *View* je vidljiv od strane korisnika, dok su *model* i *controller* skriveni.

2.6. Starling framework, Feathers UI library i GPU ubrzanje

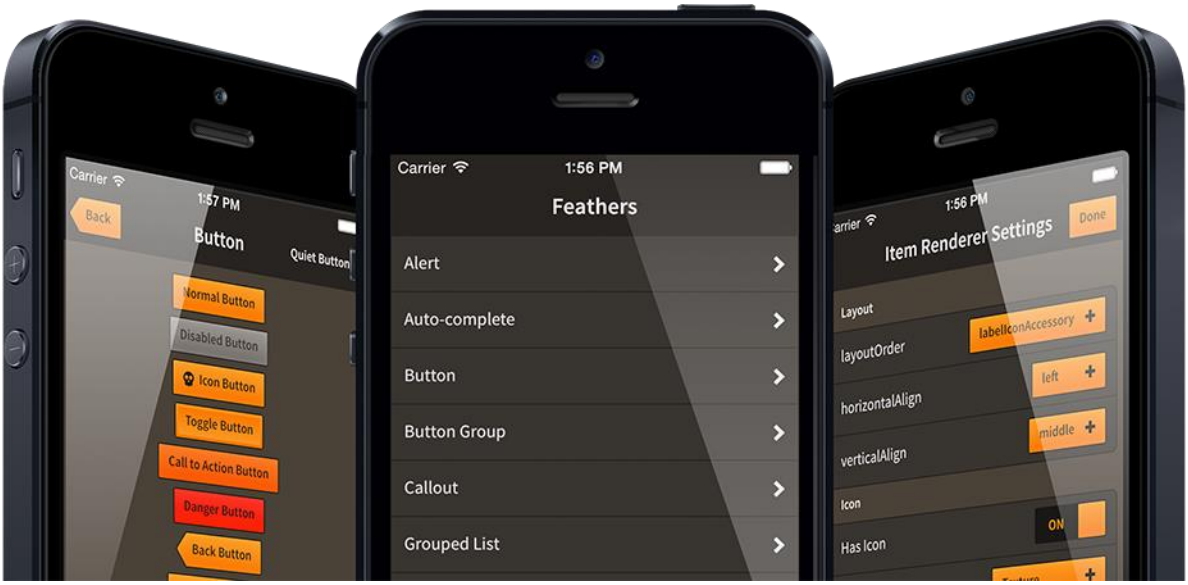
Starling je ActionScript 3 *open-source framework*, okvir otvorenog koda, razvijen 2011. godine i koristi se za izradu 2D igara koje se mogu pokrenuti na različitim web preglednicima i mobilnim platformama. Osim za igrice, Starling se koristi i za izradu ostalih vrsta aplikacija. Starling se u potpunosti bazira na AS3, a razvijen je na Stage3D (Molehill) koji je nižerazredni GPU API (*Graphics Processing Unit Application Programming Interface*) pokrenut povrh OpenGL i DirectX namijenjenih za računala i OpenGL ES2 namijenjenog za mobilne uređaje. Grafički prikaz hijerarhije vidljiv je na slici 2. Kao rezultat dobivamo da se cijeli sadržaj renderira kroz grafičku procesnu jedinicu (GPU) što poboljšava brzinu i performanse renderiranja. [17]



Slika 2: Hijerarhija Starling razvojne cjeline u odnosu na Stage3D, OpenGL i GPU (Izvor: https://www.safaribooksonline.com/library/view/introducing-starling/9781449320904/ch01s04.html)

Osim izvršavanja kroz GPU, Starling API nam nudi još niz drugih mogućnosti kao što su *multitouching*, kreiranje animacija, specijalnih efekata, tekstura, 3D efekata, filtera, atlasa tekstura (kombinacija malih tekstura - *spritesheet*) itd. Starling je besplatan, intuitivan i jednostavan, a za njegovo korištenje nije potrebno ništa osim Flash Player 11 ili AIR 3 (3.2 verzija za mobilne aplikacije). Ipak potrebno je i osnovno znanje AS3 programskog jezika. Starling je idealan za prikaz aplikacija na različitim rezolucijama jer odabire optimalan set tekstura u odnosu na rezoluciju. Najpoznatija video igra izrađena pomoću Starling okvira je *Angry Birds*.

Postoji nekoliko biblioteka koji proširuju funkcionalnost Starling okvira, a jedan od njih je i Feathers. Feathers je, kao i Starling, otvorenog koda, a često se koristi u kombinaciji sa samim Starlingom. U svojoj osnovi je veoma sličan Starlingu, a omogućava nam korištenje velikog broja unaprijed definiranih UI (eng. *User Interface*) komponenti odnosno omogućava nam izradu korisničkog sučelja za mobilne, desktop, web video igre i aplikacije bez obzira na veličinu ekrana. [18] Neke od mnogobrojnih komponenti možemo vidjeti na slici broj 3. Također, Feathers podržava interakcije mišem i dodirom.



Slika 3: Feathers UI komponente
(Izvor: http://feathersui.com/ui-components/)

Sa Starling API i Feathers UI u kombinaciji moguće je izraditi potpuno samostalne nativne aplikacije za sve operacijske sustave i aplikacije koje se pokreću preko web preglednika s instaliranim Adobe Flash Playerom. Većina *flash* programera želi biti u

mogućnosti koristiti snagu GPU ubrzanja (kroz Stage3D), bez potrebe za pisanjem višerazrednih *frameworkovima* i zadiranja u nižerazredni Stage3D API. ActionScript3, Starling i Feathers tvore snažnu kombinaciju koja sažima složenost Stage3D i omogućava jednostavno i intuitivno programiranje.

Budući da su obje otvorenog koda tj. besplatne, preuzimanje datoteka je dostupno na njihovim službenim web stranicama.

2.7. PHP skriptni jezik

PHP je *server-side* skriptni programski jezik otvorenog koda namijenjen za dinamičko generiranje HTML koda. To znači da je pomoću PHP-a možete kreirati HTML stranicu na serveru, popunjenu dinamičkim sadržajem, prije nego što je ona poslana klijentu. Ovim načinom generiranja sadržaja klijent ne može vidjeti kod (skriptu) koji je generirao sadržaj koji gleda već ima pristup čistom HTML kodu. Svojom sintaksom je sličan mnogim drugim programskim jezicima, čak i ima istoznačne (iste po sintaksi i funkcionalnosti) funkcije kao i neki drugi jezici kao što su C ili Perl. PHP je bogat funkcijama za manipuliranje mnogo različitih tipova sadržaja. Od rukovanja grafikom (png, jpg, flash…) do učitavanja .NET modula i rada sa XML-om. Ono što PHP stavlja još više ispred ostalih web skriptnih tehnologija je njegova podrška za baratanje širokom paletom baza podataka. Podržava sve popularnije baze podatak kao MySQL, PostgreSQL, dBase, Oracle, ODBC… Isto tako njegova neovisnost o operacijskom sustavu i pristupačna cijena (besplatan je) ga čini među prvim izborom velikih i malih kompanija za izradu vlastitih mrežnih sustava.

2.7.1. PHP kao *server-side* skripta

PHP je jedna od najnaprednijih i najkorištenijih *server-side* skriptnih tehnologija danas u upotrebi. Kao što možemo vidjeti na slici broj 4, 2015. godine je više od 75% popularnih web sjedišta koristilo PHP kao *server-side* programski jezik. *Server-side* skripte se izvršavaju na serveru (poslužitelju) kada poslužitelj primi zahtjev za PHP dokumentom. Nakon primitka zahtjeva za PHP dokumentom poslužitelj izvršava PHP kod i na osnovu njega generira HTML kod i šalje ga klijentu. To znači da stranica koja se prikazuje u pretraživaču klijenta ne postoji u tom obliku nigdje na serveru odakle ju je klijent primio. [5]

Slika 4: Popularnost PHP jezika u odnosu na ostale server-side programske jezike (izvor: http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/)

2.7.2. GET i POST metode prosljeđivanja podataka

Svaka HTML forma stvara se oznakom <form> koja obavezno sadrži atribut *name* koji definira naziv forme te atribute *method* i *action* koji definiraju obradu forme na serverskoj strani. *Action* atributom specificiramo kamo će biti proslijeđeni podaci iz forme, a metoda kojom se podaci šalju iz forme prema serveru je definirana atributom *method*. Dvije osnovne metode za prosljeđivanje podataka forme nekom drugom dokumentu (skripti) su POST i GET metode. Odabirom metode utječemo na koji način će se podaci forme poslati, neovisno o jeziku u kojem će se ti podaci prihvatiti i obraditi. GET metodom podaci iz obrasca se šalju kroz komandnu liniju (eng. *query string*) tj. iza znaka upitnik u adresnoj traci (eng. *adress bar*) web preglednika. Odabirom POST metode podaci nisu vidljivi u adresnoj traci već se šalju transparentno kroz zaglavlje HTTP zahtjeva (eng. *HTTP header request*) i samim time se na njih ne može utjecati izmjenom podataka u adresnoj traci preglednika. Odabir metode ovisi o zahtjevima pošiljatelja. GET metoda je korisna za pretraživanje web sjedišta i u situacijama kada se želi omogućiti posjetiteljima spremanje stranica u biblioteke zajedno s podacima iz HTML obrasca. Ta mogućnost ju čini vrlo nesigurnom metodom jer posjetitelj lako može izmijeniti podatke unutar adresne trake stoga ju nije preporučljivo koristiti pri prosljeđivanju korisničkog imena i lozinke u obrascima za prijavu i registraciju. Sigurnost podataka slanjem POST metodom ovisi o sigurnosti HTTP-a stoga se ona koristi kod svih obrazaca za slanje osjetljivih podataka. Ukoliko je potrebno proslijediti veću količinu podataka nije preporučljivo koristiti GET metodu zbog ograničenja na

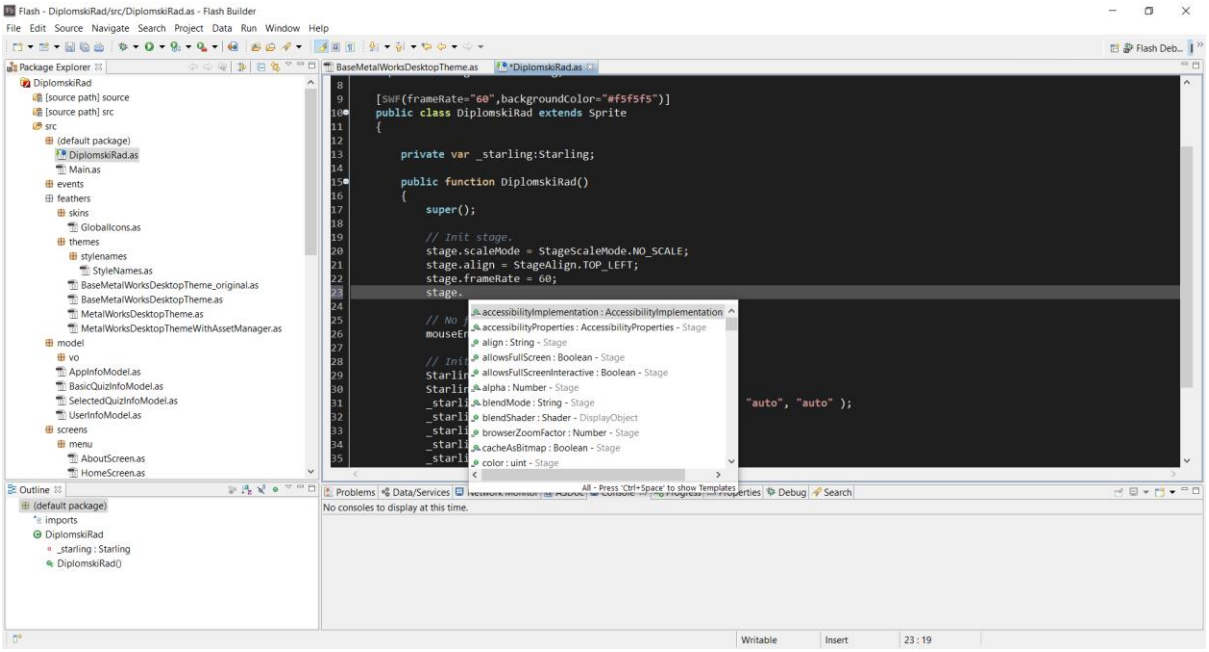
količinu podataka koja se može poslati tom metodom dok POST metoda nema takvo ograničenje.

Za izradu ove web aplikacije, PHP programski jezik je korišten kao skripta na serveru za primanje podataka o rezultatu riješenog ispita i adresi elektroničke pošte na koju će se poslati ti rezultati. Dio te PHP skripte možemo vidjeti u ispisu broj 28.

2.8. Korišteni programi

Flash Builder 4.7 je popularni IDE (*Integrated Development Environment*) programski alat, koji služi za brzu izradu opsežnih web aplikacija i nativnih računalnih i mobilnih aplikacija. Razvijen je na Eclipse platformi i koristeći se ActionScript programskim jezikom i Flex *frameworkom* omogućava nam razvoj visoko kvalitetnih softverskih rješenja neovisnih o platformi. Bržu i efikasniju izradu omogućava nam niz alata kao što su predlaganje i završavanje naredbi (eng. *code hinting*), automatsko javljanje pogrešaka, obojana sintaksa za lakše snalaženje u kodu, više od 100 gotovih predložaka i isječaka za kodiranje itd. Flash Builder također sadrži interaktivni razotkrivač i otklanjač problema (eng. *debbuger*) koji omogućava programerima nadzor koda tijekom izvršavanja naredbi te izbornike za kontrolu učinkovitosti i potrošnju memorije.

Flash Builder je dostupan u dvije inačice: Standard i Premium. Razlikuju se u nekoliko stvari, ali ponajviše u tome što Standard inačica nema potpuno automatizirano razvojno okruženje za testiranje memorije, performansi i otkrivanja pogrešaka. Za ovaj projekt korištena je Premium inačica koja je dostupna na službenim stranica tvrtke Adobe. Korisničko sučelje programa je vidljivo na slici broj 5. Preuzimanje je besplatno, ali uz probni rok u trajanju od 60 dana, a nakon toga je potrebno nadoplatiti kako bi se i dalje koristio na računalu.



Slika 5. Korisničko sučelje Flash Builder 4.7 Premium programa i code hinting

2.9. Adobe AIR SDK i Flash player plugin

Adobe AIR (*Adobe Integrated Runtime*) predstavlja osnovni podskup Adobe platforme koji korisnicima nudi mogućnost pokretanja prevedenih programa (eng. *runtime*).

Adobe AIR SDK (*Software Development Kit*) predstavlja nadskup opisane platforme. SDK sadrži sve što je potrebno kako bi programer mogao prevoditi izvorni kod programa te kako bi mogao izvoditi programe. To znači da SDK u sebi uključuje implementaciju prevodioca i drugih pomoćnih alata. On također služi programerima za pakiranje istog koda u nativne aplikacije i video igre za računala i mobilne uređaje neovisne o operacijskom sustavu.

Flash Player Plugin je dodatak web preglednicima koji im dopušta prikazivanje Flash medijskih sadržaja na web stranicama, a najčešće se koristi za prikaz animacija, video uradaka i video igara. *Debug* inačica dodatka donosi nam niz svojstava pogodnih za razvoj aplikacija kao što je skočni prozor s obavijestima o pogreškama. Također, pregršt ActionScript 3 naredbi će raditi jedino uz instaliranu *debug* inačicu. Svaki web preglednik zahtjeva svoju inačicu Flash debugger dodatka pa tako za Google Chrome je potrebno instalirati PPAPI, a za Mozilla Firefox NPAPI inačicu. Navedeni dodaci i alati dostupni su svima na službenim stranicama tvrtke Adobe Systems, a posebno

važno je imati najnoviju inačicu dodataka koji se redovito nadograđuju ponajviše iz sigurnosnih razloga.

3. RAZVOJ APLIKACIJE

3.1. Ideja aplikacije

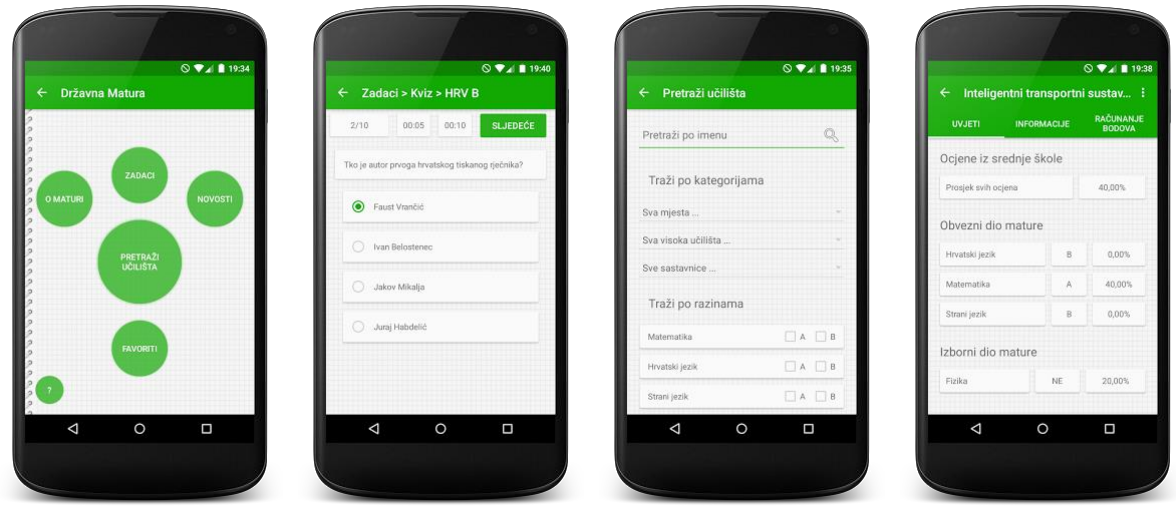
Web aplikaciju čija je izrada opisana u radu zamišljena je kao pomagalo učenicima u pripremi za ispite državne mature. Popis literature potrebnu za učenje učenici dobivaju unaprijed, a dostupni su im i svi ispiti koji su se pisali prijašnjih godina. Na osnovu tih prijašnjih ispita, početna ideja je bila napraviti bazu pitanja iz koje će se generirati ispiti iz pojedinog predmeta. Ispit će se sastojati od desetak pitanja s četiri ponuđena odgovora od kojih je samo jedan točan. Ono što je specifično za aplikaciju je generator nasumičnih pitanja i odgovora (eng. *random generator*) i jednostavna nadogradnja baze pitanja bez potrebe za mijenjanjem programskog koda. Pomoću *random generatora* svaki novi ispit je različit od prošlog, a i ponuđeni odgovori unutar pitanja svaki puta imaju novi raspored. Nakon završenog ispita korisnik dobije rezultat svojeg ispita u obliku omjera točnih odgovora i ukupnog broja pitanja te postotka točnosti. Osim toga, korisniku se daje mogućnost slanja elektroničke pošte s rezultatima ispita čime izaziva druge korisnike da se okušaju u rješavanju ispita. Time se stvara dodatna kompetitivnost, a samim time i želja za vježbanjem za ispite mature putem ove aplikacije. Dakle, ciljana skupina su učenici završnih razreda srednje škole koji namjeravaju polagati državnu maturu. Naziv aplikacije je riječ *Maturana* koja je nastala spajanjem riječi *matura* i *teretana* čime se opisuje i sama svrha ove aplikacije (teretana tj. vježbalište za maturu).

3.2. Istraživanje tržišta

Trenutno (15. kolovoza 2016.) se na tržištu nalazi nekoliko (manje od pet) aplikacija namijenjenih maturantima u svrhu lakše, bolje i efikasnije pripreme za ispite državne mature. Ispitano je tržište među aplikacijama namijenjenim Android mobilnom operacijskom sustavu preko Google Play trgovine. Ono što se može uočiti kod svih aplikacija je nedostatak kvalitete u izradi korisničkog sučelja aplikacije i nedostatak maštovitosti pri imenovanju. Kod većine testiranih aplikacija prevladava minimalistički staromodan dizajn tamnoplave, sive i crne boje. Također, nisu pridržavana osnovna načela izrade korisničkog sučelja za mobilne aplikacije kao što su veličina gumbova ili razmak između njih što otežava korisnicima korištenje istih. Nadalje, većina njih ne posjeduje nikakve dodatne mogućnosti osim generiranja ispita. Ipak, valja izdvojiti

dvije aplikacije među testiranima. Kao što je spomenuto, zbog nedostatka maštovitosti pri imenovanju te dvije aplikacije se nalaze pod istim nazivom *Drzavna Matura*.

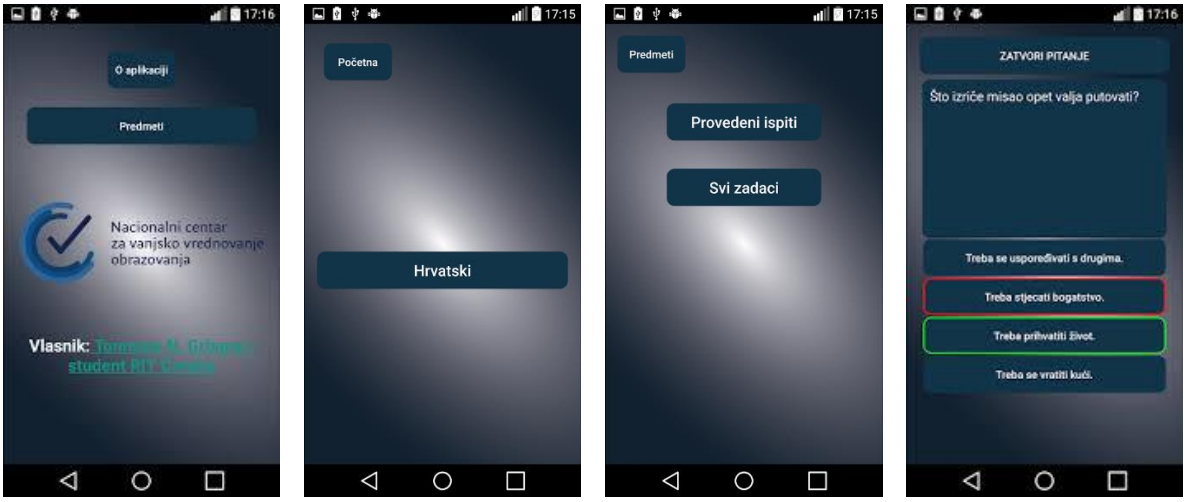
Prva izdvojena aplikacija je po broju mogućnosti i kvalitete izrade korisničkog sučelja daleko ispred svih ostalih testiranih aplikacija. Već i vizualno se aplikacija izdvaja od ostatka konkurencije. To je jedina aplikacija u kojoj ne prevladavaju tamna paleta boja. Kao što je vidljivo na slici broj 6, aplikacija obiluje raznim mogućnostima kao što su pregled svih učilišta, pretraživanje ispita državne mature, rješavanje kviza uz objavu i prikaz rezultata, popis bitnih datuma za pojedini ispitni rok, računanje bodova za pojedino učilište, novosti sa službenih stranica NCVVO (*Nacionalni centar za vanjsko vrednovanje obrazovanja*) i mnogo drugih korisnih stvari.



Slika 6: Istraživanje tržišta: 1. primjer

(Izvor: https://play.google.com/store/apps/details?id=com.vedranvidakovic.drzavnamatura)

Druga aplikacija koju ćemo izdvojiti je ogledni primjerak ostatka aplikacija ove tematike, a ono po čemu je jedinstvena je to što je izrađena u suradnji s NCVVO-om iako kvalitetom izrade uopće ne upućuje na to već izgleda kao samo jedna u nizu drugih nekvalitetnih aplikacija. Ako izuzmemo aplikaciju iz prvog primjera, sve ostale aplikacije su manje-više jednake po broju mogućnosti i izgledom korisničkog sučelja. Kao što je spomenuto, u dizajnu prevladava minimalizam s tamnim nijansama boja. Zbog manjka opcija korisničko sučelje izgleda siromašno i nedovršeno što se može vidjeti na slici broj 10.



Slika 7: Istraživanje tržišta: 2. primjer

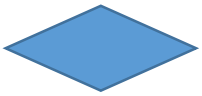
(Izvor: https://play.google.com/store/apps/details?id=matura.dm)

Navedene aplikacije se mogu besplatno preuzeti preko trgovine za Android mobilne uređaje, a ostale testirane aplikacije su također *Drzavna matura* i *DrzavnApp* koja dolazi u 2 inačice (besplatna s probnim rokom od 15 dana i inačica s potrebom plaćanja).

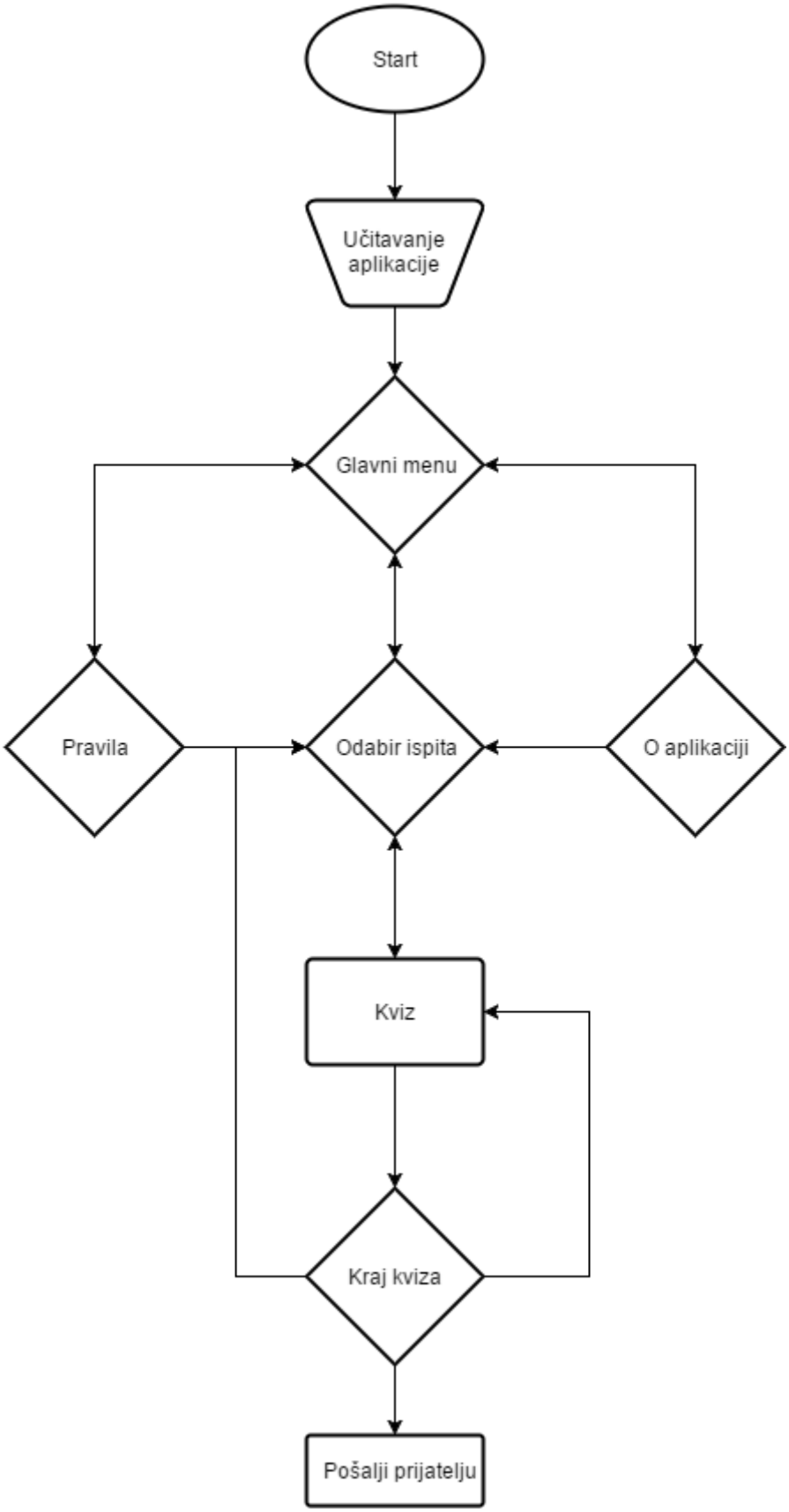
3.3. Izrada dijagrama toka aplikacije

Kako bi si olakšali stvaranje koncepta programa za rješenje zadanog problema koristimo se izradom dijagrama toka aplikacije. Dijagram toka je grafički prikaz zadanog algoritma i sastoji je od niza međunarodno dogovorenih simbola i strelica. Ti simboli su jednostavni geometrijski likovi, a povezani su crtama koje prikazuju tok rješavanja zadatka. Takav prikaz je jednostavno za analizirati, pronaći pogreške i modificirati stoga je izrada dijagrama toka aplikacije prvi korak pri realizaciji jednog ovakvog projekta čime se smanjuje mogućnost pogreške u daljnjem razvoju aplikacije.

Tablica 1: Osnovni simboli za izradu dijagrama toka

	Ovalni oblik označava početak ili kraj programa
	Pravokutnik označava radnju koje je određena programom
	Romb označava odluku ili grananje programa
	Trapez s duljom gornjom osnovicom označava ulaznu radnju
	Trapez s duljom donjom osnovicom označava izlaznu radnju

Tijekom razrade ideje aplikacije, nastojalo se što više usredotočiti na njenu glavnu funkcionalnost što je rezultiralo jednostavnijom izradom dijagrama toka aplikacije. Aplikacija sastoji od malog broja zaslona što znači i mali broj grananja unutar aplikacije odnosno ograničenog kretanja unutar aplikacije. Usprkos tome, korisniku su na svakom zaslonu, osim unutar kviza, dane dvije opcije koje mu daju mogućnost jednostavnijeg daljnjeg kretanja unutar aplikacije. Na slici 8 prikazan je dijagram toka razvijene aplikacije.

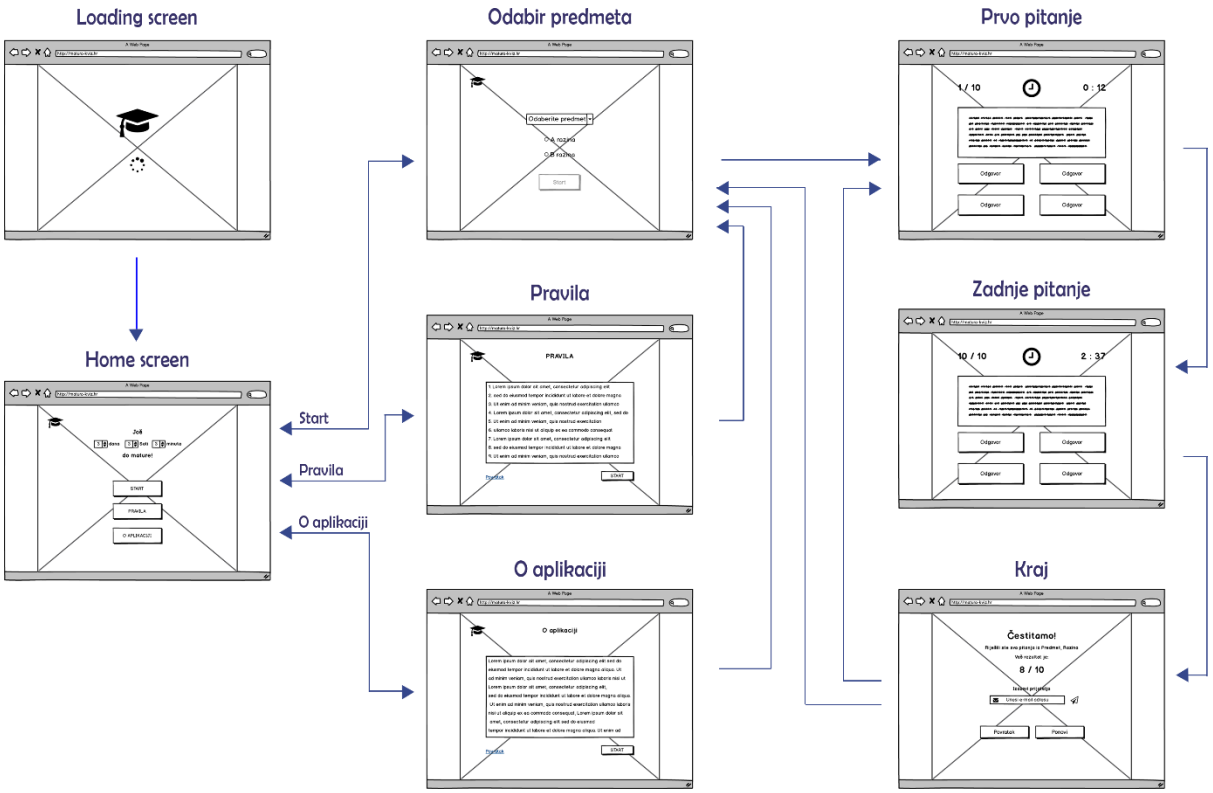


Slika 8: Dijagram toka aplikacije

3.4. Izrada skica i shema kretanja unutar aplikacije

Nakon izrade dijagrama toka aplikacije uslijedilo je skiciranje zaslona aplikacije. Skica (eng. *wireframe*) je pojednostavljeni prikaz dizajna proizvoda koji treba sadržavati jasnu prezentaciju i može poslužiti kao dio procesa izrade proizvoda. Njen cilj je predstaviti glavne grupe informacija, izgled tj. strukturu informacija i osnovnu vizualizaciju i opis interakcija korisničkog sučelja. [19] Skice se obično sastoje od nekoliko blokova sive palete boja koji na prvi pogled izgledaju besmisleno. Sadržaj se uglavnom ne unosi već se ostavlja za kasnije faze izrade. Budući da se mogu lako i brzo stvarati i uređivati potrebno ih je koristiti na samom početku izrade proizvoda, a odlični su za dobivanje prvih povratnih informacija od strane klijenta. Danas postoji nekoliko specijaliziranih alata koji nude stvaranje interaktivnih skica. Takve skice mogu poslužiti kao testni materijal za proučavanje korisničkog iskustva prilikom korištenja aplikacije. Među najpoznatijim alatima za izradu interaktivnih *wireframeova* su UXPin, InVision, Proto.io. Za izradu statičnih skica aplikacije korištena je web aplikacija Balsamiq, a izgled skica i kretanje među zaslonima je prikazan na slici 9.

Kako bi se što tečnije i brže kretalo unutar aplikacije poželjno je što više podataka učitati pri pokretanju aplikacije prije samog ulaska u aplikaciju. Za prikaz stanja trenutačne faze učitavanja podataka korišten je zaslon učitavanja (eng. *loading screen*). Na tom zaslonu korisniku je onemogućena ikakva interakcija. Nakon završenog učitavanja aplikacije korisnik dolazi na početni zaslon aplikacije (eng. *home screen*). Dolazak na zaslon učitavanja više nije moguć osim prilikom ponovnog pokretanja aplikacije. Na početnom zaslonu korisniku su dane tri opcije unutar glavnog izbornika. Dvije od tih opcija korisnika vode na zaslone na kojima korisnik može pročitati sadržaj koji opisuje pravila kviza te informacije o samoj aplikaciji i njenom autoru. Preko oba zaslona moguće je izravno doći na treću opcija unutar glavnog izbornika, a to je odabir ispita i početak kviza. Na tom zaslonu korisnik odabire ispit koji želi riješiti i pokreće kviz. Unutar kviza odgovori na pitanja predstavljaju poveznice na sljedeće pitanje i nemoguće je vratiti se na prošlo pitanje. Nakon odgovorenog zadnjeg pitanja korisniku se prikazuje zaslon s rezultatom ispita i opcijama za povratak na odabir novog ispita ili ponavljanje riješenog ispita te slanje elektroničke pošte.



Slika 9: Kretanje unutar aplikacije

3.5. Preuzimanje biblioteka i razvojnih cjelina

Prije pisanja prve linije koda nužno je pripremiti sve potrebne materijale koji će se koristiti pri razvoju aplikacije. Pod te materijale ubrajamo sve tehnologije, biblioteke, razvojne cjeline potrebne za razvoj aplikacije. Osim toga, potrebno je osmisliti sadržaj i izraditi grafiku koji će se prikazivati na zaslonima aplikacije. Naravno, sve to bi bilo beskorisno bez preuzetog i instaliranog programskog alata u kojem će se izraditi aplikacija. Redoslijed izvedbe nije striktno određen već je samo važno da se sve izvrši prije pisanja prve linije koda.

Prvi korak u izradi aplikacije je stvaranje mape proizvoljnog imena na računalu, unutar koje će se pohraniti kasnije stvoreni ActionScript projekt i njemu pripadajuće datoteke. Nadalje, unutar te mape stvaramo novu mapu imena *libs* što je skraćenica od engleske riječi *libraries* koja u prijevodu znači biblioteke. Preuzete komponente spomenute u poglavlju 2.6, razvojna cjelina Starling i biblioteka komponenti korisničkog sučelja Feathers, spremamo unutar mape *libs*. Inačice datoteka koju su preuzete su Starling 1.7 i Feathers 2.3.

Idući korak je preuzimanje Adobe AIR SDK razvojne cjeline opisane u poglavlju 2.9. Bez nje je nemoguće razvijati i izvoditi *flash* aplikacije. Nakon preuzimanja datoteka potrebno ih je smjestiti, time i zamijeniti postojeću inačicu, u određenoj mapi unutar instaliranog programskog alata Flash Builder. Putanju spomenute mape možemo vidjeti na slici broj 10.

Slika 10. Putanja spremanja Adobe AIR SDK razvojne cjeline

Posljednja datoteka koju je nužno preuzeti je Flash Player Debugger dodatak koji je također opisan u poglavlju 2.9. Inačica dodatka se može odabrati prema omiljenom web pregledniku odnosno pregledniku pomoću kojeg će se izvoditi i testirati aplikacija u razvoju. Preuzetu datoteku dovoljno je spremiti bilo gdje na računalu i pokrenuti čime se započinje kratkotrajna instalacija dodatka.

3.6. Dizajn aplikacije

3.6.1. Odabir palete boja

Boja oživljava dizajn. Ona može privući pozornost, odrediti raspoloženje te čak utjecati na naše emocije i opažanje. Vrlo važno i teško je odabrati paletu boja prikladnu za izradu projekta jer je potrebno udovoljiti nekoliko zahtjeva. Jedan od tih zahtjeva je odabir palete prikladan ciljanoj skupini korisnika. Ciljanu skupinu korisnika ove aplikacije čine mlade osobe, većinom učenici završnih razreda srednjih škola stoga je kao inspiracija za paletu boja odabrana fotografija jarkih nijansi s ciljem da dizajnu daju mladenačku energiju. Ipak, potrebno je pažljivo dozirati boje unutar aplikacije kako ukupni učinak ne bi previše naglašavao tu energiju. Kao pozadinska boja dugmadi odabrana je boja plave kupine (heksadekadska vrijednost 008DCB) i ona se proteže kroz cijelu aplikacije. Boja cvijeta biljke Fuksije korištena je kao pozadinska boja tanke linija koja služi za odvajanje naslova i ostatka sadržaja, a pojavljuje se na većini zaslona aplikacije. Njena heksadekadska vrijednost je E1315B. Nadalje, narančasta boja heksadekadse vrijednosti F47D4A upotrijebljena je kao pozadinska boja odbrojavača na početnom zaslonu te zaglavlja unutar kviza. Slika koja je poslužila kao inspiracija i paletu boja možemo vidjeti na slici 11. Uz navedene, korištene su i

bijela, siva i crna boja. Bijelom bojom je obojana pozadina prozorčića na svakom zaslonu dok je sva tipografija crne boje. Za neaktivno dugme korištena je siva boja.

Slika 11: Paleta boja

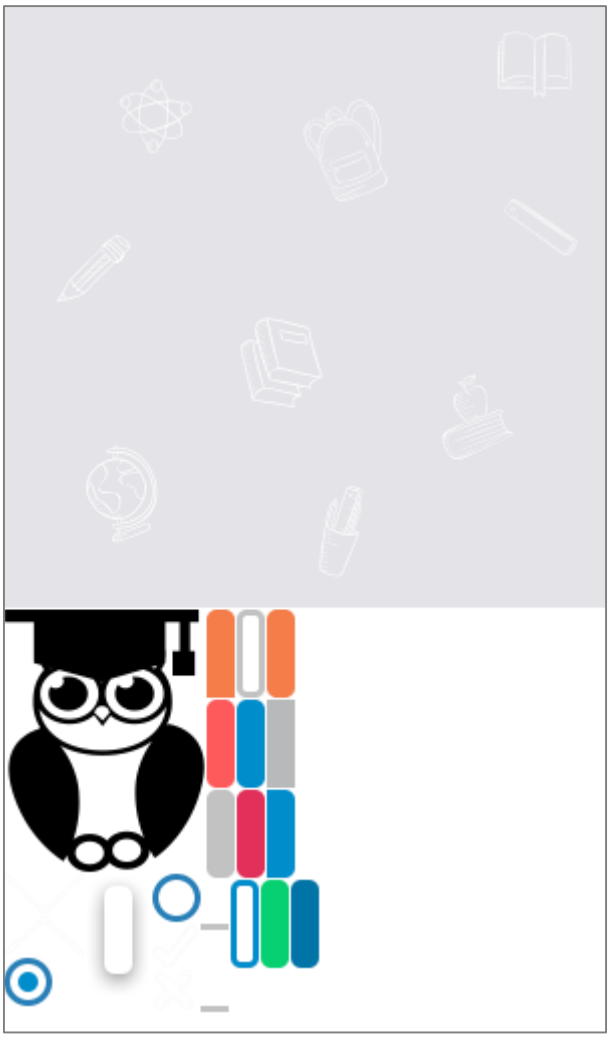
(izvor: https://designschool.canva.com/blog/100-color-combinations/)

3.6.2. Izrada i pohrana atlasa tekstura

Za izradu i obradu grafičkih elemenata korišten je Adobe Photoshop CS6 u kombinaciji sa ShoeBox programskim alatom. Adobe Photoshop je rasterski grafički alat i softver koji se koristi u svijetu digitalnog crtanja i slikanja te obrade istih. Uzevši u obzir snažne alate i jednostavno grafičko korisničko sučelje koje sadrži, ovaj program je već desetljećima u samom vrhu grafičke industrije te ga nije potrebno detaljnije predstavljati. ShoeBox je besplatan programski alat namijenjen izradi atlasa tekstura za izradu elemenata korisničkog sučelja i video igara. Ovaj snažan, a malen programski alat sadrži mnoštvo funkcija kao što je *ekstraktiranje* isječaka tekstura iz slika, razdvajanje PSD datoteka, izradu animacija itd.

Postupak izrade atlasa tekstura je sljedeći. Pomoću Adobe Photoshop programa stvoreno je nekoliko sličica u PNG formatu koje predstavljaju teksture određenih elemenata grafike na zaslonima aplikacije. Razlog iz kojeg su spremljene baš u taj

format je kako bi se sačuvala prozirnost sličica čije teksture ne ispunjavaju sličicu cijelom svojom površinom. Nakon što smo izvezli sve željene teksture sljedeći korak je povući sve teksture i ispustiti ih iznad opcije *Sprite Sheet* unutar ShoeBox programa. Potvrdnim odabirom program spaja sličice u jedan veliki atlas i nakon kratkog vremena kao produkt dobivamo dvije datoteke. Jedna od njih je atlas tekstura koji možemo vidjeti na slici 12, a druga je XML datoteka koja sadrži vrijednosti pozicija i veličina tekstura i pomoću nje dohvaćamo željene teksture iz atlasa.



Slika 12. Izgled atlasa tekstura

3.6.3. Tipografija

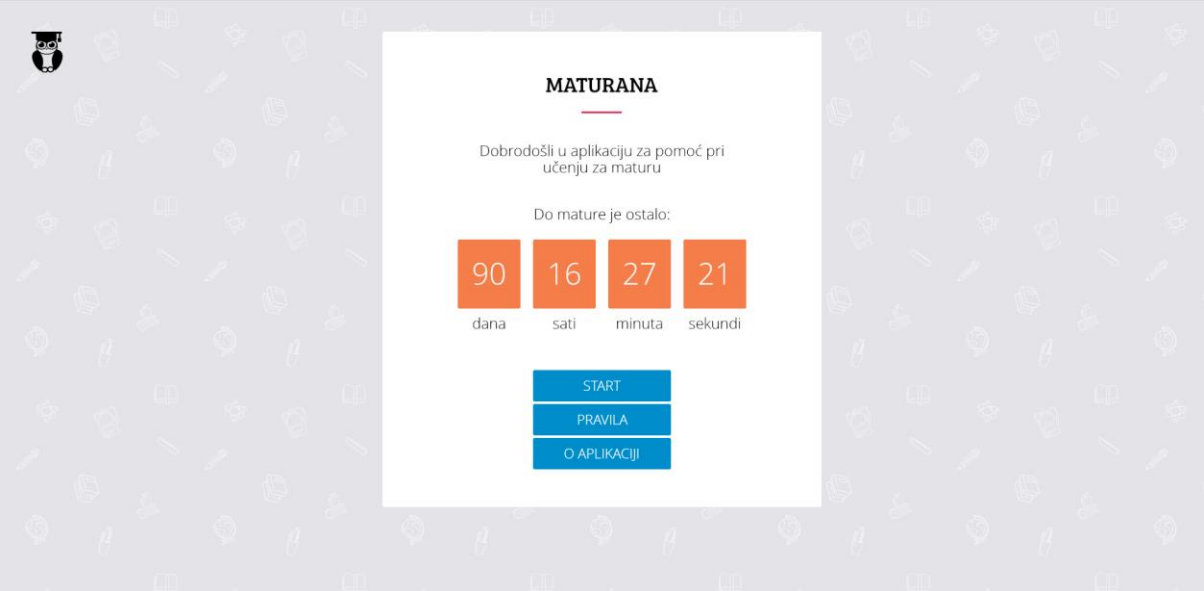
Tipografija je osnovni dio web dizajna, no kao i paletu boja izazov je odabrati pravu za vaš dizajn. Zbog svoje svestranosti, Open Sans je sveprisutan font na webu čime je olakšan odabir prvog fonta, a kako bismo ga uparili s prikladnim fontom istražena su

razna web sjedišta kao što je www.fontpair.co. Odlučeno je da će tipografski par za ovu web aplikaciju činiti Bree Serif i Open Sans. Bree Serif je serifni font i korišten je samo za naslove zaslona dok je Open Sans font bez serifa i korišten je za sve ostale tekstualne sadržaje. Oba fonta su besplatna i dostupna za preuzimanje preko web alata Google Fonts.

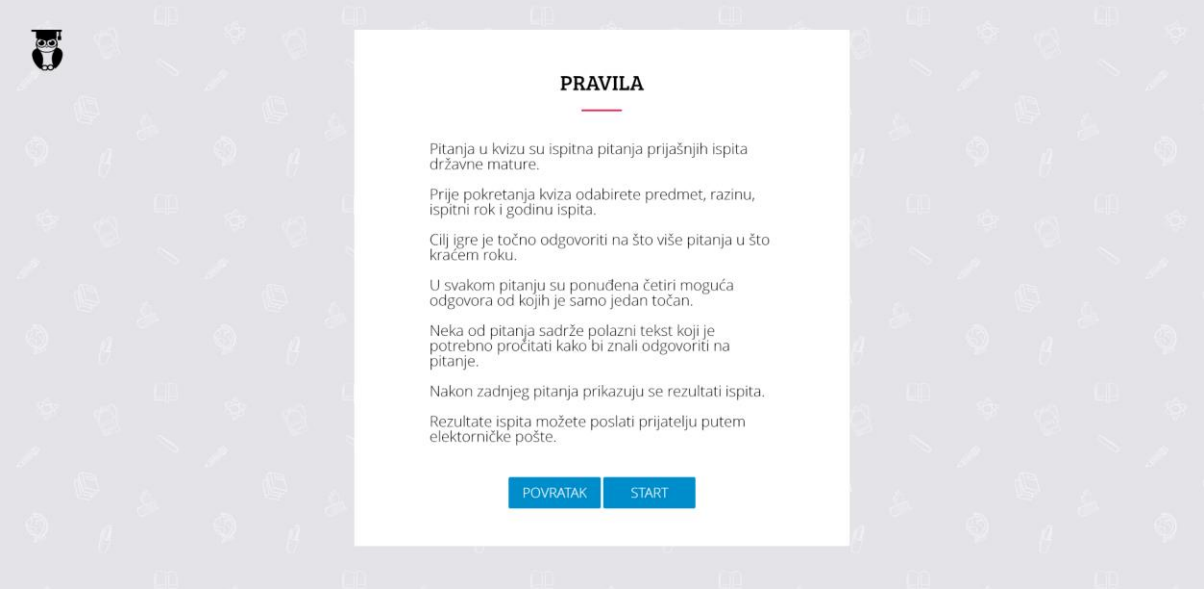
3.6.3. Izgled zaslona aplikacije

Na slikama od broja 13 do broja 26 prikazan je konačni izgled zaslona aplikacije.

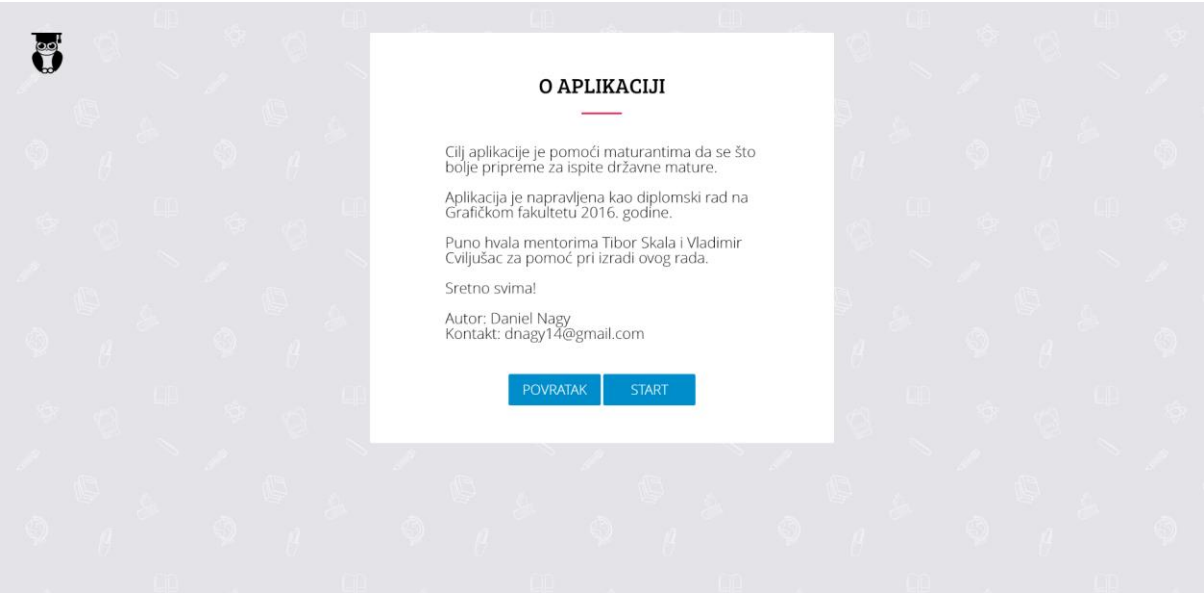
Slika 13: Izgled zaslona učitavanja



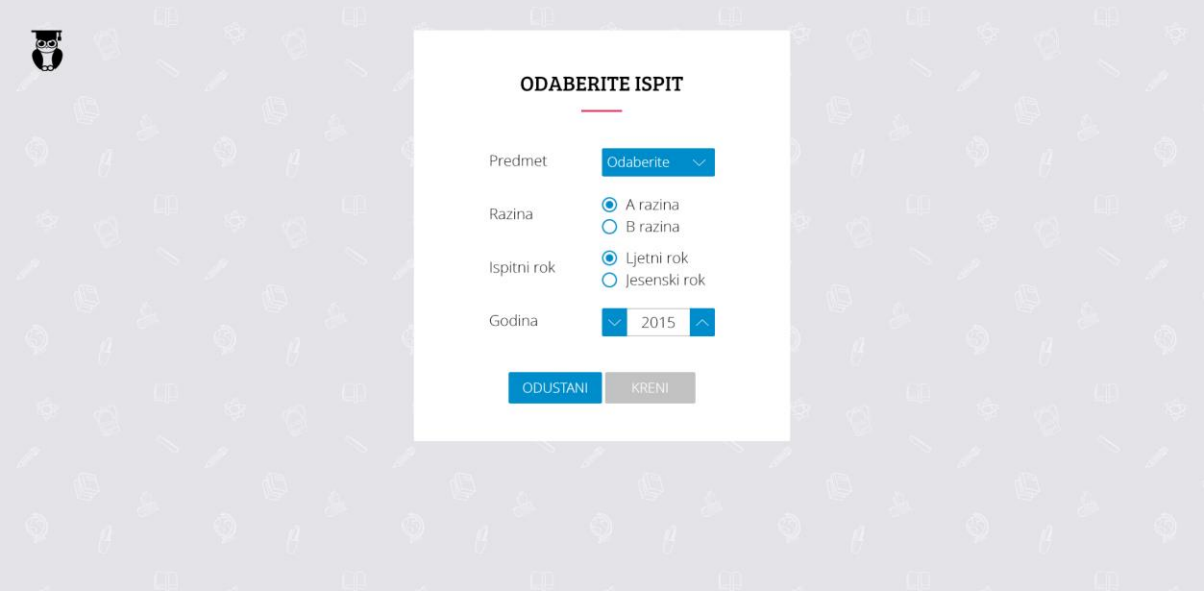
Slika 14: Izgled početnog zaslona



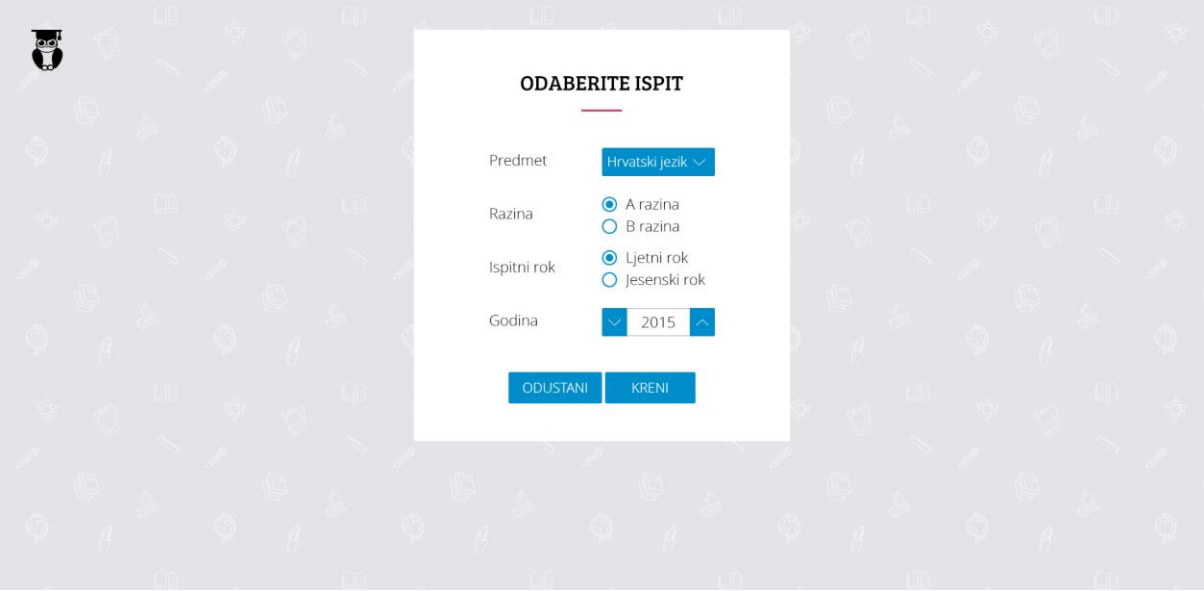
Slika 15: Izgled zaslona s pravilima kviza



Slika 16: Izgled zaslona s informacijama o aplikaciji i autoru

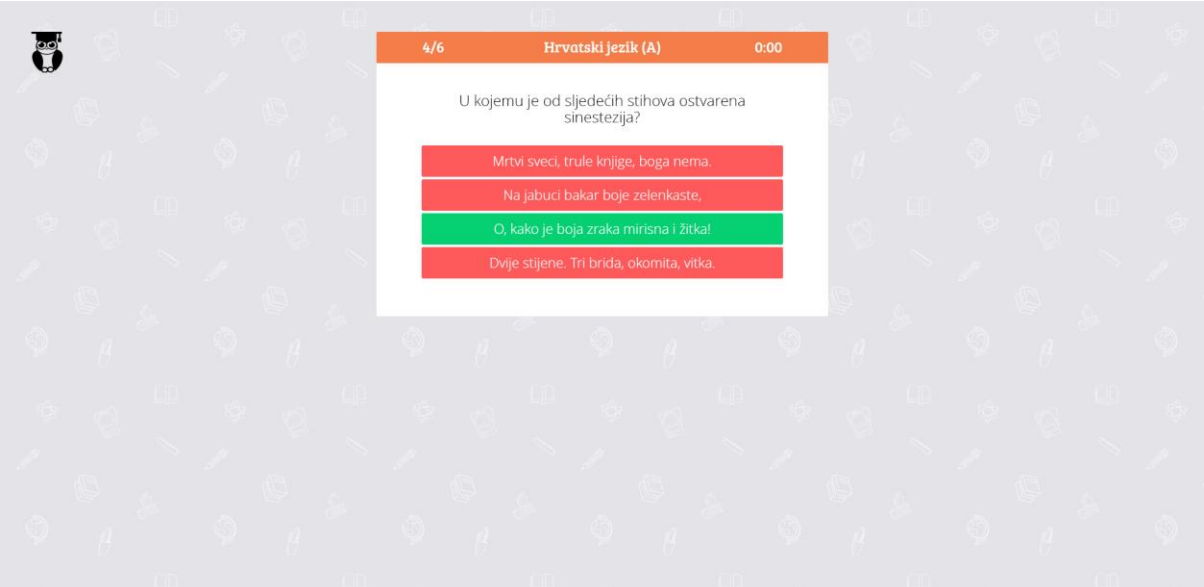


Slika 17: Izgled zaslona za odabir ispita

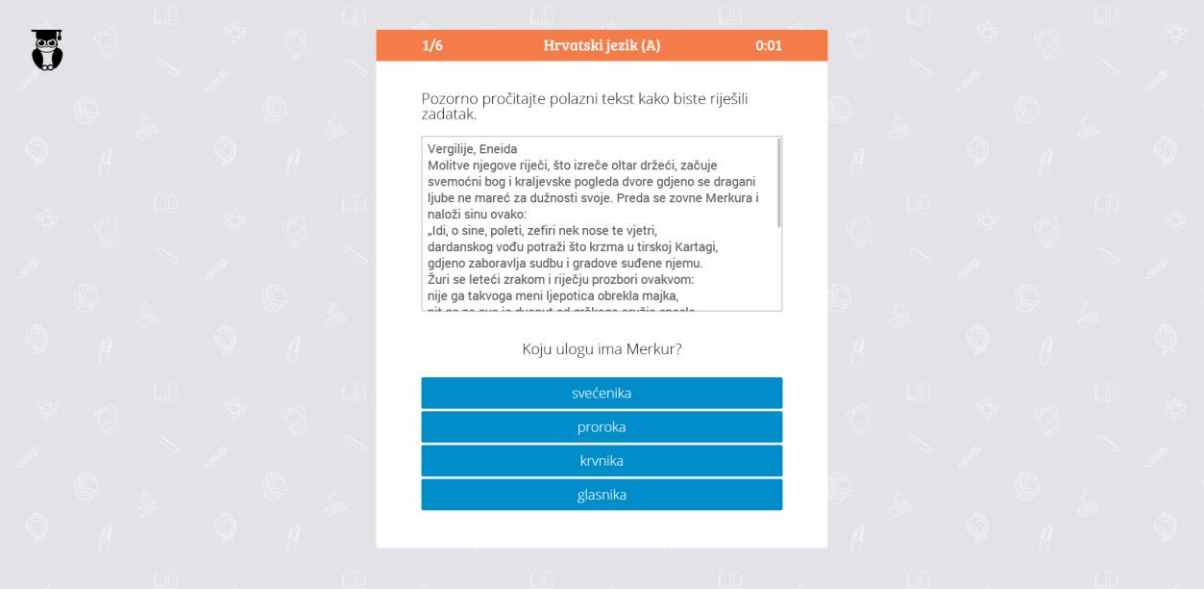


Slika 18: izgled zaslona za odabir ispita s odabranim ispitom

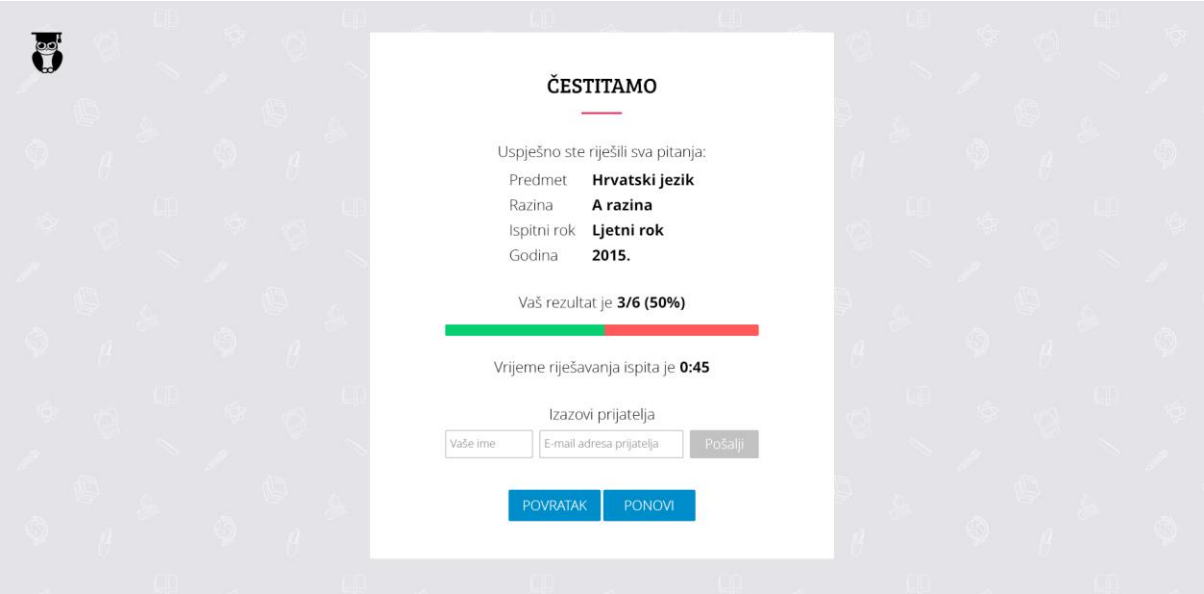
Slika 19: Izgled zaslona u kvizu



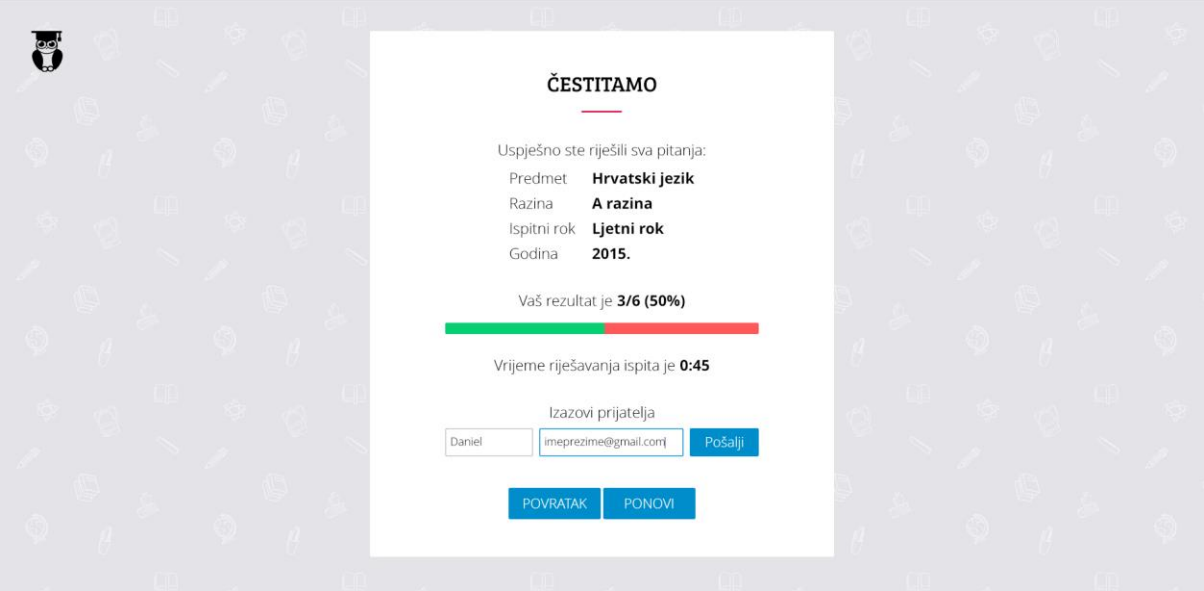
Slika 20: Izgled zaslona nakon odgovora na pitanje ili isteka vremena



Slika 21: Izgled zaslona u kvizu s pitanjem koje sadrži polazni tekst



Slika 22: Izgled zaslona s rezultatima kviza



Slika 23: Izgled zaslona s popunjenim obrascem za slanjem elektroničke pošte

Slika 24: Izgled zaslona nakon poslane elektroničke pošte

3.7. Izrada XML dokumenata

3.7.1. Datoteka s tekstualnim sadržajem aplikacije

Sav tekstualni sadržaj koji možemo vidjeti na zaslonima aplikacije spremljen je u jednoj XML datoteci. Time nam je olakšano rukovanje sadržajem u slučaju potrebe za izmjenom postojećeg sadržaja. Budući da se svi podaci nalaze na jednom mjestu eliminirali smo potrebu za pronalaskom i izmjenom sadržaja unutar samog koda. S obzirom da datoteka sadrži podatke aplikacije simbolično je nazvana AppData.xml, a dio njenog sadržaj možemo vidjeti u ispisu 16. Naziv oznaka je proizvoljan, a poželjno ih je nazvati prema podacima koje sadrže kako bi si olakšali kasniji dohvat podataka. Prva linija koda je komentar koja nam daje do znanja da se sljedeći dio dokumenta odnosi na podatke korištene na početnom zaslonu aplikacije. Ovdje su definirani naslov zaslona, kratak opis aplikacije i tekst dugmadi, a isti postupak je ponovljen za sve ostale zaslone.

```
<!-- Home Screen data -->
<homeScreenTitle>MATURANA</homeScreenTitle>
<homeScreenDescription>Dobrodošli u aplikaciju za pomoć pri učenju za maturu
</homeScreenDescription>
<homeScreenTimerDays>dana</homeScreenTimerDays>
<homeScreenTimerHours>sati</homeScreenTimerHours>
<homeScreenTimerMinutes>minuta</homeScreenTimerMinutes>
<homeScreenTimerSeconds>sekundi</homeScreenTimerSeconds>
<homeScreenTimerTextUntil>Do mature je ostalo:</homeScreenTimerTextUntil>
<homeScreenStartButtonLabel>START</homeScreenStartButtonLabel>
<homeScreenRulesButtonLabel>PRAVILA</homeScreenRulesButtonLabel>
<homeScreenAboutButtonLabel>O APLIKACIJI</homeScreenAboutButtonLabel>
```

Ispis 16 – Isječak koda: Dio sadržaja AppData.xml dokumenta

3.7.2. Datoteka s parametrima kviza

Svi parametri ispita su smješteni unutar istog XML dokumenta. S obzirom da je aplikacija zasnovana na ispitima državne mature bilo je potrebno definirati nekoliko parametara, a to su popis školskih predmeta, razina, ispitnih rokova i godina izrađenih

ispita. Uz njih je još definirano vrijeme početka održavanja predstojećih ispita državne mature koje se koristi za odbrojavanje vremena na početnom zaslonu. Elementima je unutar početne oznake dan atribut imena *source* preko kojeg se unutar programskog koda manipulira podacima iz XML dokumenta. Na primjer, *source* atribut elementa koji označava školski predmet sadrži vrijednost koja označava putanju do XML datoteke koja sadrži ispite iz tog školskog predmeta. Datoteka je također nazvana prema sadržaju unutar nje (*QuizData.xml*).

```
<dateOfExam>06.06.2017</dateOfExam>
<timeOfExam>09:00:00</timeOfExam>
<nextQuestionDelay>1</nextQuestionDelay>
<subjectsList>
   <value source="subjectsXML/HrvatskiJezik.xml" maxNumeOfQuestions="">Hrvatski
jezik</value>
</subjectsList>
<levelsList>
   <value source="A">A razina</value>
   <value source="B">B razina</value>
</levelsList>
<examSeason>
  <value source="1">Ljetni rok</value>
  <value source="2">Jesenski rok</value>
  </examSeason>
<examYears>
  <value source="2014">2014.</value>
  <value source="2015">2015.</value>
</examYears>
```

Ispis 17 – Isječak koda: QuizData XML dokument

3.7.3. Datoteka s ispitnim pitanjima

Posljednji XML dokument predstavlja ispite iz jednog školskog predmeta s pitanjima i odgovorima koja će se nalaziti u kvizu i podatak o vremenu unutar kojeg je dozvoljeno odgovoriti na pitanje. Dakle, broj ovakvih datoteka nije ograničen već ovisi o tome iz koliko različitih školskih predmeta želimo izraditi pitanja u kvizu. U ispisu 18 prikazan

je uzorak pomoću kojeg se unose podaci koji će predstavljati pitanja, odgovore i vremensko ograničenje odgovora na pitanje. Element naziva *questions* predstavlja sami ispit i sadrži tri atributa *level*, *examSeasonId* i *examYear* koji redom označavaju razinu ispita, ispitni rok i godinu ispita. Unutar njega je ugniježđen element *question* unutar kojeg se nalaze elementi s podacima o vremenu i tekstu pitanja te mogućim odgovorima na to pitanje. Element *questionText* je neobvezan jer će on sadržavati polazni tekst za pitanja koja se odnose na taj tekst. Ukoliko postoji polazni tekst potrebno je dati i vrijednost atributu *author* koji označava naslov i autora tog polaznog teksta. Unutar elementa *questionAnswers* se nalaze mogući odgovori na pitanje i pri tome je važno da element kojem atribut *type* ima vrijednost *true* označava točan odgovor na to pitanje. Poredak pitanja i odgovora unutar XML dokumenta nije presudan s obzirom da se korištenjem algoritma za nasumičan raspored, za svaki novi ispit, generira novi poredak pitanja i odgovora u ispitu.

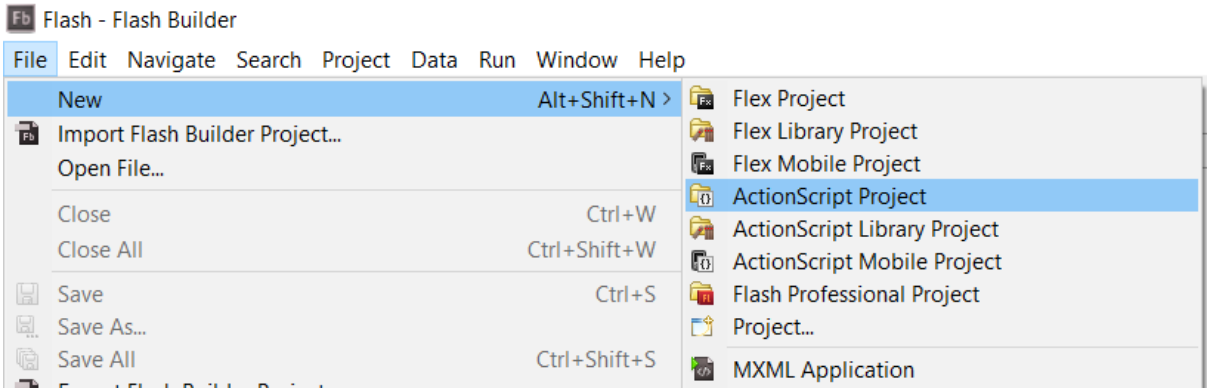
```
<questions level="" examSeasonId="" examYear="">
        <question>
                <questionTime>
                        <questionMins></questionMins>
                        <questionSecs></questionSecs>
                </questionTime>
                <questionText author=""></questionText>
                <questionTitle></questionTitle>
                <questionAnswers>
                        <value type="false"></value>
                        <value type="false"></value>
                        <value type="true"></value>
                        <value type="false"></value>
                </questionAnswers>
        </question>
</questions>
```

Ispis 18 – Isječak koda: Uzorak za unos ispitnih pitanja i odgovora

3.8. Stvaranje projekta u programu Flash Builder

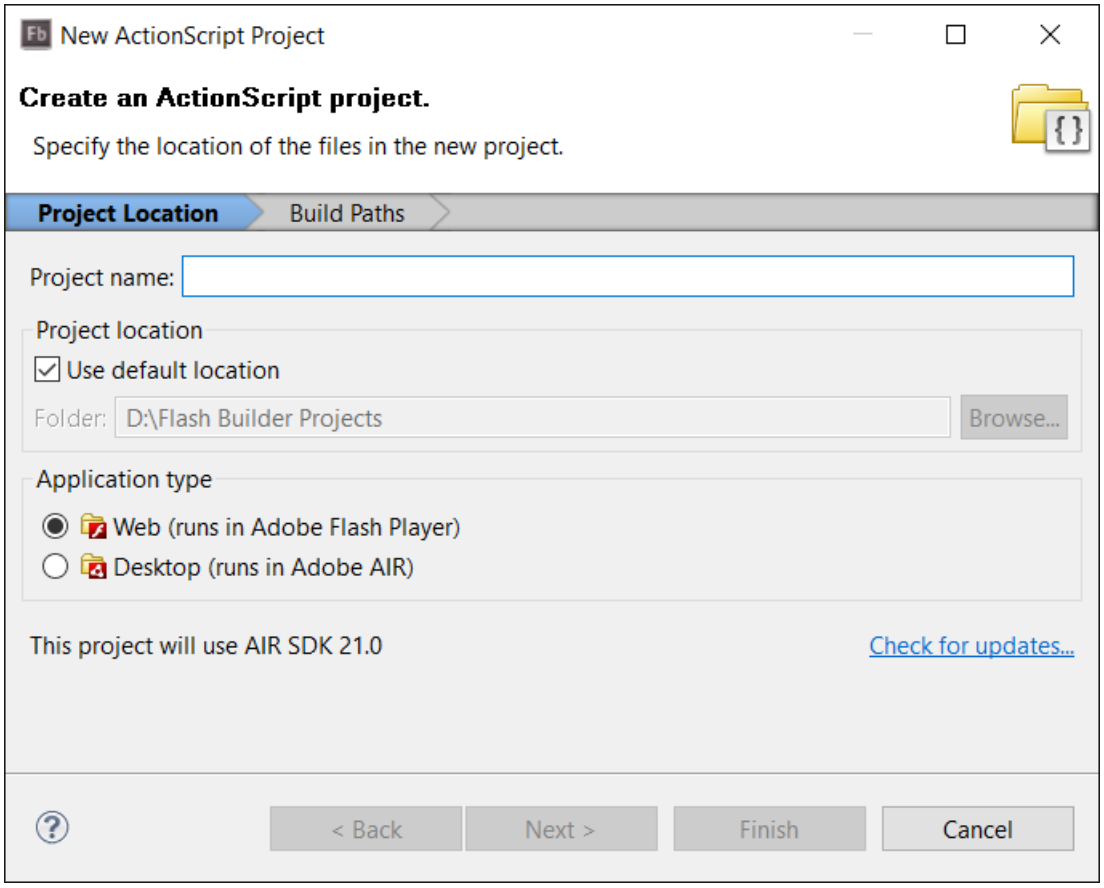
3.8.1. Stvaranje novog ActionScript projekta

Nakon pripreme i izrade svih potrebnih datoteka spremni smo na stvaranje AS3 projekta odnosno naše aplikacije. U Flash Builder 4.7 programu klikom na opciju *File* u alatnoj traci na vrhu programa pokrećemo padajući izbornik u kojem klikom na opciju *New* otvaramo novi izbornik gdje odabiremo opciju *ActionScript Project* među popisom svih mogućih vrsta projekata koje je moguće izraditi pomoću Flash Builder programa. Navedeni postupak možemo vidjeti na slici broj 25.



Slika 25: Stvaranje novog AS3 projekta

Odabirom na tu opciju prikaže nam se skočni prozor u kojem prvo imenujemo projekt te odabiremo lokaciju projekta i vrstu aplikacije koju ćemo izraditi. Za lokaciju pohrane projekta odabiremo mapu koju smo stvorili u poglavlju 3.5. Budući da će se ova aplikacija pokretati unutar web preglednika odabiremo opciju *Web (runs in Adobe Flash Player)*. Ukoliko smo uspješno uvezli preuzetu Adobe AIR SDK razvojnu cjelinu čiji je postupak opisan u poglavlju 3.5, na dnu prozora bismo trebali vidjeti informaciju o inačici koju će ovaj projekt koristiti. Kao što možemo vidjeti na slici 26, razvojna cjelina je uspješno uvezena.



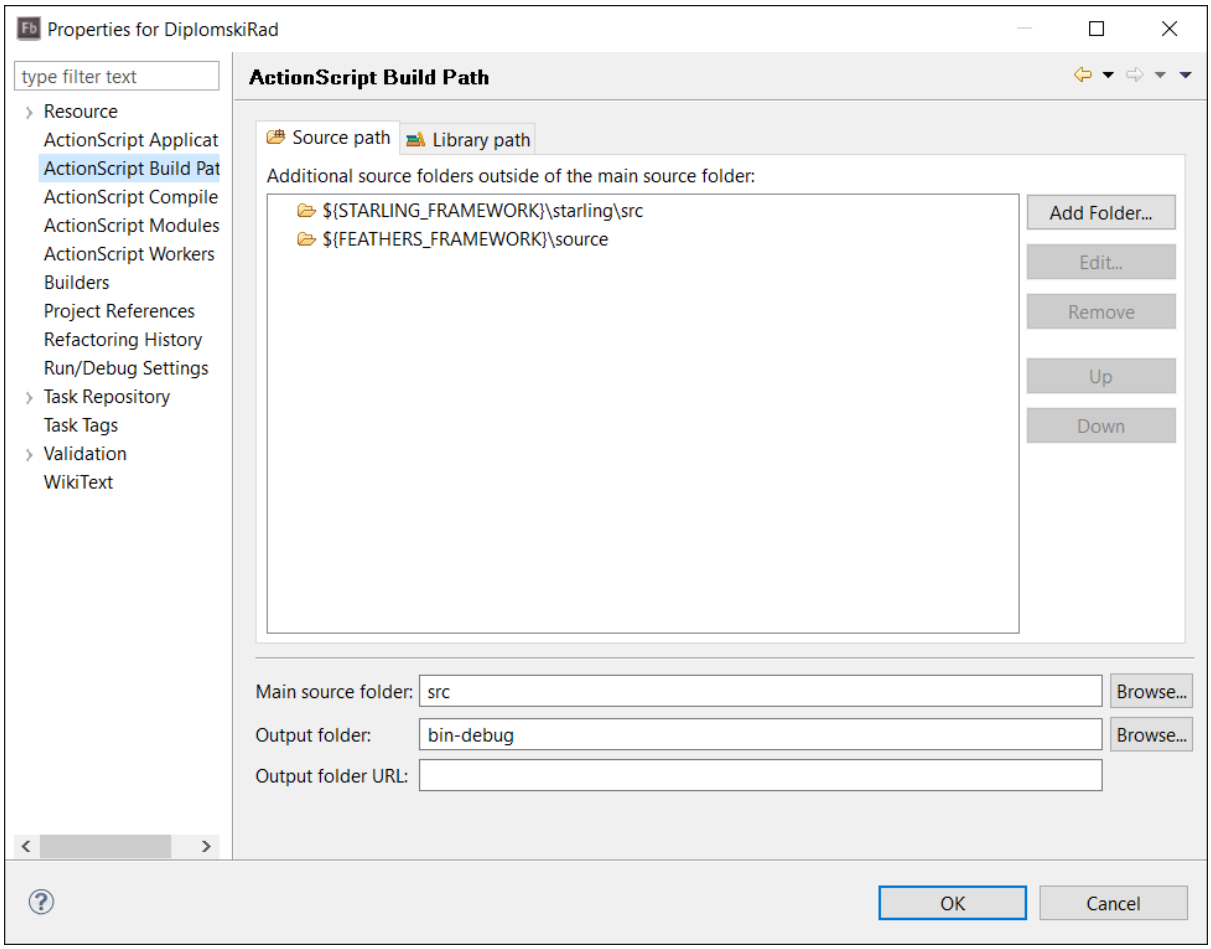
Slika 26: Imenovanje AS3 projekta i odabir vrste aplikacije

Nakon imenovanja projekta i odabira lokacije pohrane projekta i vrste aplikacije pritišćemo na opciju *Finish* u izborniku na dnu prozora i time smo uspješno stvorili AS3 projekt.

3.8.2. Uvoz okvira, biblioteke i XML dokumenata

Sljedeći korak je uvoz Starling 1.7 okvira i Feathers 2.3 biblioteke. U lijevom dijelu prozora, pritiskom na desnu tipku miša iznad mape novostvorenog projekta pojavljuje se padajući izborniku u kojem odabiremo opciju *Properties*. U izborniku s lijeve strane skočnog prozora odabiremo *ActionScript Build Path* opciju. Poslije toga odabiremo na karticu *Source Path* i pritišćemo na gumb *Add Folder* u izborniku s desne strane. Nakon pronalaska mape *libs* u kojoj su spremljeni Starling i Feathers, prvo odabiremo *src* mapu koja se nalazi u Starling mapi te potvrdimo klikom na gumb *OK*. Isti postupak vrijedi i za Feathers, a naziv mape koju ovaj put označujemo je *source*. Na slici 27 možemo vidjeti uspješno uvezene komponente. Klikom na gumb *OK* potvrđujemo promjene i zatvaramo skočni prozor. U međuvremenu su izašle nove inačice obje

komponente, no zbog kompleksnosti migracije koda iz jedne inačice u drugu ostali smo pri preuzetim inačicama.



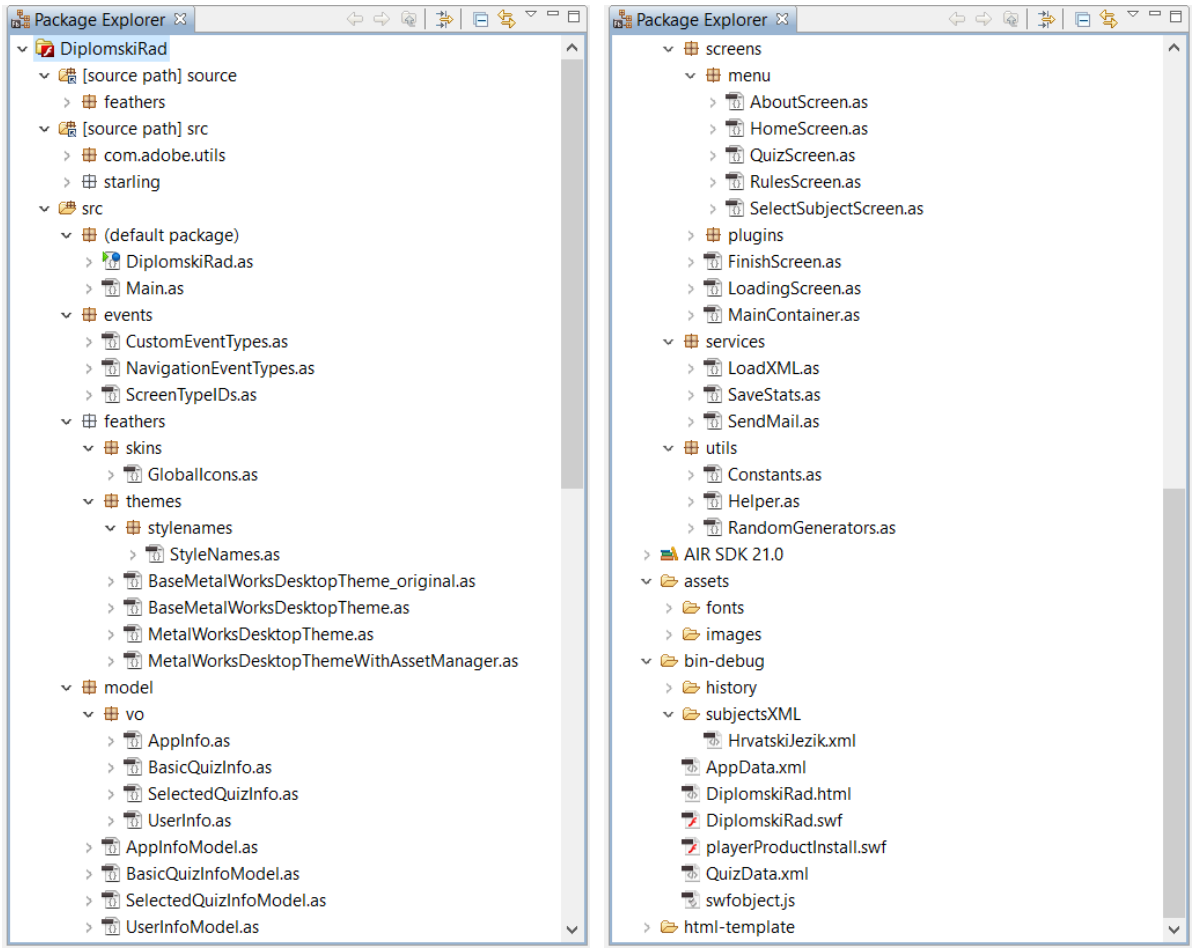
Slika 27: Uvoženje Starling okvira i Feathers biblioteke

Nakon uvezenog Starling okvira i Feathers biblioteke potrebno je smjestiti preuzete fontove i stvoren atlas tekstura u *assets* mapu stvorenog projekta. Za lakše snalaženje među datotekama, unutar mape *assets* napravljena je nova mapa *fonts* u kojoj su smješteni svi preuzeti fontovi i mapa *images* u kojoj je smješten atlas tekstura i pripadajući XML dokument.

Nadalje, XML dokumente *AppData* i *QuizData* opisane u poglavlju broj 3.7 smještamo u mapu *bin-debug* koja se također nalazi u mapi stvorenog projekta. Za pohranu XML dokumenata koji sadrže ispitna pitanja i odgovore, unutar mape *bin-debug* izrađena je nova mapa imena *subjectsXML*.

3.8.3. Stvaranje paketa i klasa

Sljedeći korak je stvaranje paketa (eng. *package*) odnosno mapa u koje ćemo spremati sve klase. Prvi korak je isti kao i kod stvaranja projekta samo što ovaj put odabiremo opciju *Package* umjesto *ActionScript Project*. Uz zadani paket i Feathers paket stvoreno je još pet novi paketa naziva *events*, *model*, *screens*, *services* i *utils*. Unutar paketa *model* ugniježđen je novi paket *vo* (kratica od *value object*), a unutar *screens* paketa novi paket naziva *menu*. Postupak izrade klasa je sličan kao i kod stvaranja paketa. Pritiskom na desnu tipku miša iznad paketa u kojem želimo stvoriti klasu otvara se izbornik u kojem odabiremo opciju *New* i nakon toga *ActionScript Class*. Hijerarhiju datoteka projekta možemo vidjeti na slici broj 28.



Slika 28: Hijerarhija datoteka aplikacije u programu Flash Builder

Zadani package sadrži dvije klase koje se brinu za inicijalizaciju Starling okvira, klase za navigaciju između zaslona i teme korisničkog sučelja. Unutar *events* paketa nalaze se klase koje sadrže varijable vezane uz događaje koji se mogu pojavljivati kroz cijelu

aplikaciju. Pomoću *model* paketa dohvaćamo podatke iz XML dokumenata koje prosljeđujemo u varijable koje se nalaze u klasama unutar *vo* paketa. *Screens* paket je zadužen da drži na okupu klase koje predstavljaju zaslone aplikacije. Servisi kao što su učitavanje XML dokumenata i slanje elektroničke pošte nalaze se u *services* paketu, a varijable s konstantnim vrijednostima i algoritam nasumičnosti su smješteni u *utils* paketu.

3.9. Najvažniji isječci programskog koda

3.9.1. Pokretanje Starling okvira

Prve linije programskog koda napisane u Flash Builder programu odnose se na inicijalizaciju Starling okvira i napisane su u zadanoj klasi koja je istog imena kao naziv stvorenog AS3 projekta i ona se prva pokreće pri pokretanju aplikacije. Razlog inicijalizacije Starlinga je što on ne koristi *Flash DisplayObject*. Starling koristi Stage3D te iz toga razloga je potrebna početna inicijalizacija gdje se napušta standardni *Flash DisplayObject* čime se dobiva GPU ubrzanje. Za početak je potrebno uvesti nekoliko klasa, odrediti brzinu izvršavanja programskog koda i boju pozadine. Unutar klase je prvo deklarirana varijabla *_starling* koja je instanca klase *Starling.* Scena je pozicionirana i počinje u gornjem lijevom kutu web pregledniku. Kako bismo u svakom trenutku imali uvid u stanje potrošnje memorije i brzinu izvršavanja koda pokrenuta je metoda *showStats*. Konstruktor Starling klase zahtjeva nekoliko parametara, a prvi od njih je ime klase koja predstavlja portal u Starlingov svijet GPU ubrzanja. Konačno, metodom *start* pokrenut je Starling okvir. Starling je potrebno samo jednom inicijalizirati na početku te mu kasnije možemo pristupati kroz cijeli kod aplikacije.

```
package
{
	import flash.display.
	import flash.display.StageAlign;
	import flash.display.StageScaleMode;
	import starling.core.Starling;

	[SWF(frameRate="60", backgroundColor="#e4e3e8")]
	public class DiplomskiRad extends Sprite
	{
		private var _starling:Starling;

		public function DiplomskiRad()
		{
			super();
			stage.scaleMode = StageScaleMode.NO_SCALE;
			stage.align = StageAlign.TOP_LEFT;
			stage.frameRate = 60;
			mouseEnabled = mouseChildren = false;
			Starling.handleLostContext = false;
			Starling.multitouchEnabled = false;
			_starling = new Starling( Main, stage, null, null, "auto", "auto" );
			_starling.enableErrorChecking = false;
			_starling.showStats = true;
			_starling.simulateMultitouch = false;
			_starling.start();
		}
	}
}
```

Ispis 19 – Isječak koda: Pokretanje Starling okvira

3.9.2. Postavljanje uzorka pozadine i navigatora između zaslona

Za dizajn pozadine aplikacije izrađen je jednostavan uzorak koji se kao instanca *TiledImage* klase ponavlja duž cijelog vidnog polja web preglednika. Osim toga

stvoren je navigator koji je zaslužan za kretanje tranzicije između zaslona i oboje su postavljeni na scenu.

```
var background:TiledImage = new TiledImage (GlobalIcons.bgPattern);
background.height = stage.stageHeight;
background.width = stage.stageWidth;
this.addChild(background);

this.navigator = new StackScreenNavigator();
this.addChild(this.navigator);
```

Ispis 20 – Isječak koda: Postavljanje pozadine i navigatora

Budući da ne želimo mogućnost vraćanja na zaslon učitavanja, glavnom navigatoru su dodana dva zaslona. Jedan koji predstavlja zaslon učitavanja i jedan koji predstavlja kontejner za ostale zaslone.

```
var loadingScreen:StackScreenNavigatorItem = new
StackScreenNavigatorItem(LoadingScreen);
loadingScreen.addPopEvent(Event.COMPLETE);
this.navigator.addScreen(ScreenTypeIDs.LOADING_SCREEN, loadingScreen);

var mainContainer:StackScreenNavigatorItem = new
StackScreenNavigatorItem(MainContainer);
mainContainer.addPopEvent(Event.COMPLETE);
this.navigator.addScreen(ScreenTypeIDs.MAIN_CONTAINER, mainContainer);
```

Ispis 21 – Isječak koda: Dodavanje zaslona navigatoru

Metodom *setScreenIDForPushEvent* navigatoru je rečeno da osluškuje događaje koji su zapravo prikazivanje željenog i uklanjanje aktivnog zaslona.

```
loadingScreen.setScreenIDForPushEvent(NavigationEventTypes.SHOW_MAIN_CON
TAINER, ScreenTypeIDs.MAIN_CONTAINER);

mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_HOME_SC
REEN, ScreenTypeIDs.HOME_SCREEN);

mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_SELECT_S
UBJECT_SCREEN, ScreenTypeIDs.SELECT_SUBJECT_SCREEN);

mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_RULES_SC
REEN, ScreenTypeIDs.RULES_SCREEN);

mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_QUIZ_SCR
EEN, ScreenTypeIDs.QUIZ_SCREEN);

mainContainer.setScreenIDForPushEvent(NavigationEventTypes.SHOW_FINISH_SC
REEN, ScreenTypeIDs.FINISH_SCREEN);
```

Ispis 22 – Isječak koda: Događaji u navigatoru koji se osluškuju

Sav navedeni kod se nalazi u *Main* klasi, a postavljanje zaslona u *mainContainer* navigatoru se odvija u istoimenoj klasi.

3.9.3. Petlja za prikazivanje željenog zaslona

Metoda kojom se koristimo kako bi prikazali željeni zaslon je *switch* petlja. Kao ulazni parametar joj se prosljeđuje objekt koji sadrži naziv zaslona kojeg želimo prikazati i njegovu vrijednost provjerava u svojem tijelu. Ako se podudaraju ulazni parametar i parametar unutar tijela petlje pokreće se metoda kojom se prikazuje zaslon čiji je naziv jednak vrijednosti ulaznog parametra. Ukoliko se ulazni parametar ne slaže s nijednom vrijednosti u tijelu petlje, petlja javlja pogrešku. Dakle, ova petlja zna koji zaslon je trenutno aktivan i koji zaslon je potrebno prikazati.

```
switch(responseObj.screenToShowScreenTypeID){
case ScreenTypeIDs.HOME_SCREEN:
this.navigator.showScreen(ScreenTypeIDs.HOME_SCREEN);
                        break;
        case ScreenTypeIDs.SELECT_SUBJECT_SCREEN:
this.navigator.showScreen(ScreenTypeIDs.SELECT_SUBJECT_SCREEN);
                break;
        case ScreenTypeIDs.RULES_SCREEN:
        this.navigator.showScreen(ScreenTypeIDs.RULES_SCREEN);
                break;
        case ScreenTypeIDs.ABOUT_SCREEN:
        this.navigator.showScreen(ScreenTypeIDs.ABOUT_SCREEN);
                break;
        case ScreenTypeIDs.QUIZ_SCREEN:
        this.navigator.showScreen(ScreenTypeIDs.QUIZ_SCREEN);
                break;
        case ScreenTypeIDs.FINISH_SCREEN:
        this.navigator.showScreen(ScreenTypeIDs.FINISH_SCREEN);
                break;
        default:
        throw new Error("Pogreška! Pokušavaš pozvati zaslon koji ne postoji!");
                break;
}
```

Ispis 23 – Isječak koda: Switch petlja za prikazivanje zaslona

3.9.4. Učitavanje XML dokumenata

Klasa *LoadXML* je servis koji u sebi ima metode za dohvat podataka s određene lokacije koja može biti disk računala, server, USB memorija i slično. Uz dohvat podataka klasa je zadužena da prati je li preuzimanje uspješno izvršeno i ukoliko dođe do pogreške istu javlja i ispisuje. Po završetku preuzimanja podataka servis *LoadXML* prosljeđuje podatke u metodu iz klase *AppInfoModel* za spremanje podataka u *AppInfo*.

```
public function load(url:String, _modelToLoad:Object):void{
        modelToload = _modelToLoad;
        xmlLoader = new URLLoader();
        xmlLoader.load(new URLRequest(url));
        xmlLoader.addEventListener(Event.COMPLETE, processXML);
        xmlLoader.addEventListener(HTTPStatusEvent.HTTP_STATUS,
httpStatus_Handler);
        xmlLoader.addEventListener(IOErrorEvent.IO_ERROR, ioError_Handler );
        xmlLoader.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
securityError_Handler );
}
```

Ispis 24 – Isječak koda: Metoda za učitavanje XML dokumenata

```
private function processXML(e:Event):void   {
        loadedXML = new XML(e.target.data);
        modelToload.getInstance().loadData(loadedXML);
        xmlLoader.removeEventListener(Event.COMPLETE, processXML);
        xmlLoader.removeEventListener(HTTPStatusEvent.HTTP_STATUS,
httpStatus_Handler);
        xmlLoader.removeEventListener(IOErrorEvent.IO_ERROR, ioError_Handler );
        xmlLoader.removeEventListener(SecurityErrorEvent.SECURITY_ERROR,
securityError_Handler );
        this.dispatchEventWith(CustomEventTypes.DOWNLOAD_FINISHED,true);
}
```

Ispis 25 – Isječak koda: Prosljeđivanje podataka iz XML dokumenta

3.9.5. MVC arhitektura u aplikaciji

View odnosno grafičko sučelje aplikacije su sve klase koje se nalaze u paketu *screens* i one sadrže grafičke elemente koje dodajemo. U paketu *model* nalaze se klase koje predstavljaju modele koji sadrže funkcije ili metode. Objekti koji sadrže podatke nalaze se u paketu *vo.* Na primjer, *AppInfoModel* klasa je zadužena za „komunikaciju“ između *viewa* i objekta gdje se spremaju vrijednosti ili stanja. *Model* se sastoji od funkcija koje

izvršavaju neku radnju nad podacima koji se nalaze u objektu *AppInfo* koje se pozivaju iz recimo *HomeScreen* klase. *AppInfoModel* prilikom pozivanja stvara instancu klase u slučaju da se poziva prvi puta, a kasnije provjerava da li postoji u memoriji te ukoliko postoji vraća postojeću klasu. Pohranjeni podaci u vrijednosni objekt u *AppInfo* klasi dostupni su kroz cijeli kod aplikacije.

```
public static function getInstance():AppInfoModel    {
        if(!_instance){
                _instance = new AppInfoModel();
                }
        else{
                // ne čini ništa
        }
        return _instance;
}
```

Ispis 26 – Isječak koda: Provjera postojanja instance klase AppInfoModel

3.9.6. Metoda za slanje elektroničke pošte

Kako bismo uspješno poslali elektroničku poštu preko sučelja aplikacije potrebne su nam dvije datoteke, a u ovom slučaju to su *SendMail* klasa i PHP skripta koja prima podatke na serveru. Unutar *SendMail* klase nalazi se funkcija koja sadrži *URLLoader* koji šalje podatke iz objekta *URVariables* na zadanu *URLRequest* adresu PHP skripte na serveru. Osim toga potrebno je definirati metodu slanja podataka i događaje koji se osluškuju pri pozivu metode.

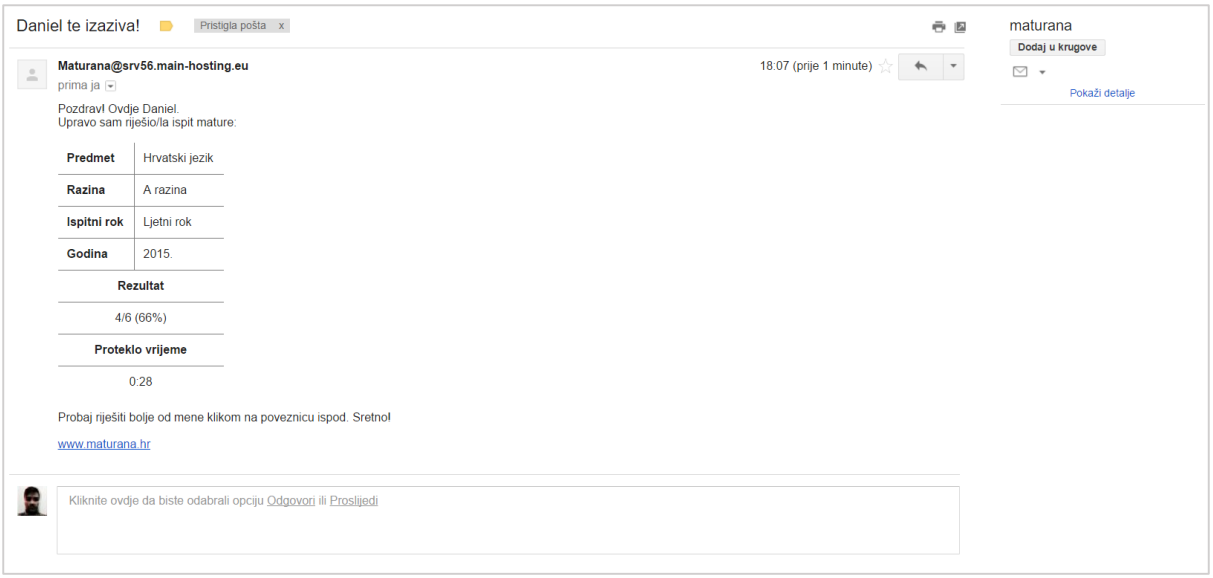
```
public function send():void {
        sendDataLoader = new URLLoader();
        sendDataLoader.dataFormat = URLLoaderDataFormat.TEXT;
        urlRequest = new URLRequest("http://danielnagy.xyz/skripta.php");
        urlRequest.method = URLRequestMethod.POST;
        urlRequest.data = urlVariables;
        sendDataLoader.addEventListener(Event.COMPLETE, sendDataCompleted);
        sendDataLoader.addEventListener(HTTPStatusEvent.HTTP_STATUS,
httpStatus_Handler);
        sendDataLoader.addEventListener(IOErrorEvent.IO_ERROR,
ioError_Handler);
        sendDataLoader.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
securityError_Handler);
        sendDataLoader.load(urlRequest);
}
```

Ispis 27 – Isječak koda: Send metoda za slanje podataka skripti na serveru

Ukoliko je metoda uspješno izvršena, podaci su proslijeđeni PHP skripti na serveru i spremljeni u varijable nakon dodatne provjere. Te varijable prosljeđujemo u ugrađenu *mail* metodu kao ulazne parametre i ako nema pogreške u kodu elektronička pošta je uspješno poslana, a njen izgled možemo vidjeti na slici 29.

```
$mailTo = isset($_POST['mailTo']) ? $_POST['mailTo'] : '';
$examResult = isset($_POST['examResult']) ? $_POST['examResult'] : '';
$fromName = isset($_POST['name']) ? ucFirst($_POST['name']) : '';
$mailSubject = $fromName . " te izaziva!";
$message = $examResult;
$headers = "From: Maturana\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/html; charset=UTF-8\r\n";
mail($mailTo, $mailSubject, $message, $headers);
```

Ispis 28 – Isječak koda: Dio PHP skripte koja prima podatke i šalje e-poštu



Slika 29: Izgled primljene elektroničke pošte

3.9.7. Algoritam za generiranje ispita

Jedna od najvažnijih funkcionalnosti aplikacije je generiranje ispita koji je rasporedom pitanja i odgovora različit od prošlog ispita. Time smo spriječili ponavljanje ispita i dodali dozu neizvjesnosti pri rješavanju ispita. Za tu funkcionalnost stvorena je klasa naziva *RandomGenerators.* Ta klasa sadrži metodu *shuffleArray* koja kao ulazni parametar prihvaća niz ispunjen podacima, a vraća isti niz, ali s novim poretkom podataka. Unutar metode su inicijalizirane dvije nove varijable. Prva varijabla je novi prazan niz kojem je određena dužina u skladu s dužinom niza kojeg smo proslijedili metodi kao ulazni parametar. Druga varijabla je brojčanog tipa i ona nam služi kao pokazivač. Koristeći *for* petlju dohvaćamo podatke unutar niza i izvršavanje petlje se ponavlja onoliko puta koliko je dugačak novi prazan niz. Svakim pokretanjem petlje varijabli *randomPos* pridodan je novi broj čija je vrijednost cijeli broj u rasponu od nule do broja koji označava dužinu ulaznog niza. Razlog iz kojeg raspon počinje od broja nula je što se indeksiranje elemenata u nizu započinje baš od nule. Ovime smo odabrali element niza čija je vrijednost indeksa jednaka trenutnoj vrijednosti varijable *randomPos*. Ovisno o rednom broju ponavlja petlje novom nizu je pridodan novi nasumični element iz ulaznog niza, a pozicija u novom polju je jednaka rednom trenutnom broju ponavljanja izvršavanja petlje. Ugrađenom metodom *splice* izabrani element brišemo iz ulaznog polja. Uzevši u obzir da *splice* metoda kao rezultat vraća niz potrebno joj je definirati dvije vrijednosti. Prva je pozicija elementa u nizu kojeg

želimo obrisati, a druga je broj elemenata koje želimo obrisati. U ovom slučaju vraća samo jedan član zato što smo joj definirali da obriše jedan element iz ulaznog polja kojemu je vrijednost indeksa jednaka vrijednosti varijable *randomPos*. Nakon što se isprazni ulazni niz, petlja se prestaje izvršavati i novi niz napunjen podacima metoda vraća kao rezultat izvršavanja funkcije.

```
public class RandomGenerators {
        public static function shuffleArray (inputArray:Array):Array {
        var shuffledArray:Array = new Array(inputArray.length);
                var randomPos:Number = 0;
                for (var i:int = 0; i < shuffledArray.length; i++) {
                        randomPos = int(Math.random() * inputArray.length);
                        shuffledArray[i] = inputArray.splice(randomPos, 1)[0];
                        }
                        return shuffledArray;
}
```

Ispis 29 – Isječak koda: Algoritam nasumičnosti

3.10. Izvoz aplikacije

Nakon što je aplikacija u potpunosti isprogramirana i uspješno istestirana spremna je na izvoz. Izvoz prevedene verzije aplikacije je jedan jednostavan postupak koji zahtjeva samo tri pritiska lijeve tipke miša. Odabirom na opciju *Project* u alatnoj traci na vrhu Flash Builder programa otvara nam se padajući izbornik u kojem odabiremo opciju *Export Release Build.* Pritom se pojavljuje skočni prozor u kojem označujemo projekt kojeg želimo izvesti i mapu u koju želimo pohraniti izlazne datoteke. Nakon kratkotrajnog procesa, u željenoj mapi smo dobili nekoliko datoteka koje je potrebno smjestiti na *hosting* račun kako bi aplikacija bila dostupna svima putem web preglednika i internetske veze. Osim navedenih datoteka, vrlo bitno je smjestiti i izrađene XML dokumente i PHP skriptu inače aplikacija neće funkcionirati.

4. ZAKLJUČAK

Uzevši u obzir da je čovjek u stanju zapamtiti oko 75% informacija ako ih je vidio, čuo i aktivno koristio, interaktivno učenje putem mobilnih, desktop i web aplikacija se pokazalo kao efikasniji način učenja od standardnog učenja iz knjiga i skripti. Cilj ovog diplomskog rada bio je izraditi web aplikaciju koja će maturantima olakšati učenje te ih na interaktivniji način pripremiti za ispite državne mature. Koristeći se ActionScript 3 programskim jezikom te implementacijom razvojnih cjelina, alata, XML dokumenata i ostalog sadržaja uspješno je razvijena jedna takva aplikacija. Najvažnija njena funkcionalnost je da korisnicima nudi brzu interakciju na način da se pomoću XML obrasca omogućuje generiranje novih obrazaca od strane administratora bez potrebe za izmjenom programskog koda. Uz to, njena prednost je i modularnost koja nam omogućava izmjene postojećeg i dodavanja novog sadržaja aplikacije na vrlo jednostavan način putem XML dokumenta. Time aplikacija nije ograničena samo na izradu replika ispita državne mature već je mogu koristi različite tvrtke za testiranje svojeg osoblja i slično. Kroz diplomski rad objašnjeni su svi koraci nužni za izradu ovakve aplikacije. Unatoč svemu, aplikacija nije razvijena na najbolji mogući način. Prostora na napredak možemo primijetiti u tzv. responzivnom dizajnu odnosno prilagodljivosti sadržaja zaslona aplikacije na veličinu preglednika u kojem je pokrenuta. Zasloni su inicijalno centrirani u svakom pregledniku, ali pri smanjivanju veličine preglednika dio sadržaja nestaje iz vidnog polja i potrebno je pomicati vidno polje kako bi se prikazao sav sadržaj što je loše korisničko iskustvo pri korištenju manjih uređaja kao što su mobiteli ili tableti. Također, pri unosu novog sadržaja u XML dokumente važno je pridržavati se zadanog kostura kako ne bi došlo do pogreške u unošenju podataka čime nastaju problemi u prikazivanju sadržaja na zaslonima. Za napredak tog dijela aplikacije potrebno je izgraditi sustav za upravljanje sadržajem tzv. CMS (*Content Managment System*) odnosno izraditi sučelje putem kojeg bi administrator upravljao podacima u bazama i time bi se otklonila sva mogućnost pogreške.

Aplikacija je dostupna na https://goo.gl/3qFIoL.

5. LITERATURA

[1] *** http://public.mzos.hr/Default.aspx?sec=2246 – MZOS - Državna matura, 2. kolovoza 2016.

[2] Kennedy, B. Musciano C. (2002): HTML & XHTML: The Definitve Guide, 5th Edition

[3] Nielsen, J. (2005): Multimedia and Hypertext: The Internet and Beyond.

[4] Duckett, J. (2011): HTML and CSS: Design and Build Websites

[5] Nagy, D. (2012): Dinamičko generiranje izvješća u CSS tehnologiji

[6] *** https://www.w3.org/Consortium/ - W3C, 2. kolovoza 2016.

[7] Dietel, Deitel, Nieto, Lin, Sadhu (2001): XML How to program

[8] *** http://www.fer.unizg.hr/_download/repository/mipro_xml_tekst.pdf - XML tehnologija i primjena u sustavima procesne informatike, 2. kolovoza 2016

[9] *** https://hr.wikipedia.org/wiki/XML - Povijest XML-a, 3. kolovoza 2016.

[10] *** http://www.totalxml.net/future-of-xml.php - The future of XML, 3. kolovoza 2016.

[11] Moock, C. (2002): ActionScript for Flash MX: The Definitive Guide, 2nd Edition

[12] *** http://www.adobe.com/devnet/actionscript/learning/oop-concepts/objects-and-classes.html – AS3 – Objects and classes, 3. kolovoza 2016.

[13] *** http://www.adobe.com/devnet/actionscript/learning/as3-fundamentals/functions.html - AS3 – Functions 4. kolovoza 2016.

[14] *** https://sites.google.com/site/sandasutalo/osnove-programiranja/grananje/jednostruko-uvjetno-grananje - Osnove programiranja, jednostruko grananje, 4. kolovoza 2016.

[15] *** https://sites.google.com/site/sandasutalo/osnove-programiranja/grananje/visestruko-uvjetno-grananje - Osnove programiranja, višestruko grananje, 4. kolovoza 2016.

[16] *** https://sites.google.com/site/sandasutalo/osnove-programiranja/grananje/naredba-switch-case - Osnove programiranja, Switch-case, 4. kolovoza 2016.

[17] Imbert, T. (2012): Introducing Starling, Building GPU Accelerated Applications

[18] Nagy, D. (2016): Seminarski rad: Starling API i Feathers UI

[19] *** http://www.popwebdesign.net/popart_blog/2014/06/wireframe-vs-prototip-u-cemu-je-razlika/ - Wireframe vs prototip – u čemu je razlika, 8. kolovoza 2016.

6. POPIS SLIKA

7. POPIS ISJEČAKA KODA

9. POPIS TABLICA