Snimanje i opis procesa snimanja glazbenog albuma putem modernih web tehnologija

Svjetličić, Domagoj

VISOKO UČILIŠTE ALGEBRA

VISOKA ŠKOLA ZA PRIMIJENJENO RAČUNARSTVO

ZAVRŠNI RAD

Snimanje i opis procesa snimanja glazbenog albuma putem modernih web tehnologija

Domagoj Svjetličić

Zagreb, listopad 2017.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 25.01.2018.

Predgovor

Zahvaljujem se svim profesorima Visokog učilišta Algebra, osobito profesorima i asistentima koji su predavali na multimedijskom računarstvu, jer su tijekom mog studija posvetili velik dio svog vremena kako bih dobio potrebno znanje koje danas svakodnevno koristim, kako na poslu, tako i za realizaciju završnog rada. Posebno bih zahvalio profesorima Bojanu Fulanoviću, Ivanu Božajiću, Vedranu Dakiću i Damiru Begoviću jer su mi znanja njihovih kolegija bila najpotrebnija za uspješnu izradu ovog rada. Također bih htio zahvaliti i kolegama iz Web Development odjela firme Degordian, koji su mi na dnevnoj bazi doprinosili nova znanja i vještine koja su uvelike doprinijela lakšoj i kvalitetnijoj izradi završnog rada. Na kraju bih htio istaknuti i zahvale svojim roditeljima Mirjani i Davoru Svjetličiću, koji su mi omogućili i priuštili studij koji mi je donio iznimno veliku količinu znanja koje je bilo potrebno da bih mogao raditi posao kojeg volim i bez kojeg realizacija ovog završnog rada ne bi bila moguća.

Sažetak

Ideja i cilj ovog rada je izrada glazbenog albuma, te web aplikacije koja će na interaktivan način prikazivati snimljene pjesme sa albuma. Samim time korisnici aplikacije će dobiti realni i praktični uvid kako započeti snimati vlastite pjesme i razviti web aplikaciju koristeći moderne i najpopularnije web development tehnologije kao što su PHP, MySql, JavaScript, Vue.js, HTML i CSS (SCSS).

Ključne riječi: glazba, glazbeni album, web development, pjesme, web aplikacija, programiranje.

Summary

The idea and main goal of this paper is producing and recording music album and web application that shows recorded songs in an interactive way for users. Application users will get real and practic insight of how to get started with recording their own songs and develop a web application using modern and popular web development technologies such as PHP, mySql, JavaScript, Vue.js, HTML and CSS (SCSS).

Key Words: music, musical album, web development, songs, web application, programming.

Sadržaj

1. Uvod

Tema rada je izrada glazbenog albuma i razvoj web aplikacije koja će na interaktivan način korisnicima prikazivati sam sadržaj albuma, te ih educirati sa ciljem da dobiju znanje kako bi i sami stekli vještine i znanja za samostalnu izradu projekta ovakvoga tipa. Primarni cilj je kroz ovaj rad prikazati proces razvoja projekta koji uključuje izradu glazbenog albuma i razvoj web aplikacije koja na interaktivan način prezentira finalni produkt glazbenog albuma. Isto tako veliki je fokus stavljen na edukaciju o tehnologijama i alatima koji su korišteni u ovome radu, te prikazati najbolje načine korištenja relevantnih tehnologija i alata. Korisnici gotove aplikacije i čitaoci ovog rada će biti upućeni i educirani o svim navedenim tehnologijama, alatima i instrumentima koji su bili potrebni za izradu ovog rada, te ih pripremiti i potaknuti na samostalan razvoj projekata sličnog ili istog tipa.

Temeljni razlog odabira navedene teme ovog završnog rada je dugogodišnje iskustvo u skladanju, sviranju i snimanju glazbenih kompozicija, te naročito sviranju gitare. Također velika motivacija za realizaciju ovakvog tipa projekta je velika osobna zainteresiranost prema tehnologiji i glazbi, zbog čega mi je ovaj rad idealna kombinacija različitih znanja i vještina kojima se dobiva rad kreativnog i tehnološkog temelja. Izradom rada se nastojalo spojiti dvije meni najzanimljivije grane multimedije kako bi se u konačnici rezultiralo smislenim, korisnim i kreativnim finalnim produktom. Također jedan od predefiniranih ciljeva je stvoriti programski sustav koji će biti korišten za višekratnu uporabu, odnosno da ukoliko dođe do širenja glazbenog albuma i programskog sustava, to bude realizirano na što jednostavniji i brži način.

2. Izrada glazbenog albuma

Glazbeni album (engl. *Album*) je skup povezanih glazbenih kompozicija u jednu raščlanjenu cjelinu. Proces samog snimanja glazbenog albuma varira od umjetnika do umjetnika, te nema strogo definirana pravila izrade istog. Nije rijetkost da umjetnici koriste svoju umjetničku slobodu i kreativnost pri snimanju albuma što često rezultira iznimno velikoj raznolikosti žanrova i stilova kompozicija unutar istog glazbenog albuma. U mom slučaju, album će se sastojati isključivo od instrumentalnih kompozicija. Sami redoslijed procesa snimanja biti će ideja, skladanje, izrada kostura kompozicije, odabir pjesama i ambijenta, snimanje audio materijala, te naposljetku i post – produkcija (engl. *Post – production*) audio materijala.

2.1. Ideja

Sama ideja kreiranja ovog glazbenog albuma nije samo doći do finalnog produkta, odnosno gotovih pjesama, nego i usput pokazati korake i zahtjeve koji su potrebni kako bi se do tog cilja došlo na što kvalitetniji mogući način. Iako postoji mnogo načina i rasporeda procesa razvijanja glazbenog albuma kojim se služe razni amaterski i profesionalni glazbenici, ja ću prikazati sistem realizacije glazbenog albuma koji meni osobno najviše odgovara, te ga podijeliti putem ovog rada. Od jednake, ako ne i veće važnosti, je prikazati način i proces same izrade nego finalni produkt.

Prilikom samog razvijanja ideje glazbenog albuma, definirao sam odredbe da će pjesme biti isključivo instrumentalnog tipa, da će se u velikoj većini koristiti samo gitara i bas gitara od glazbenih instrumenata, te da će bubnjevi biti pisani virtualno unutar Apple Logic Pro X[1] aplikacije. Također sam definirao da će svaka pjesma biti zasebnog žanra, ali će također svakoj pjesmi glavni i lako uočljiv temeljni utjecajni žanr biti Rock glazba.

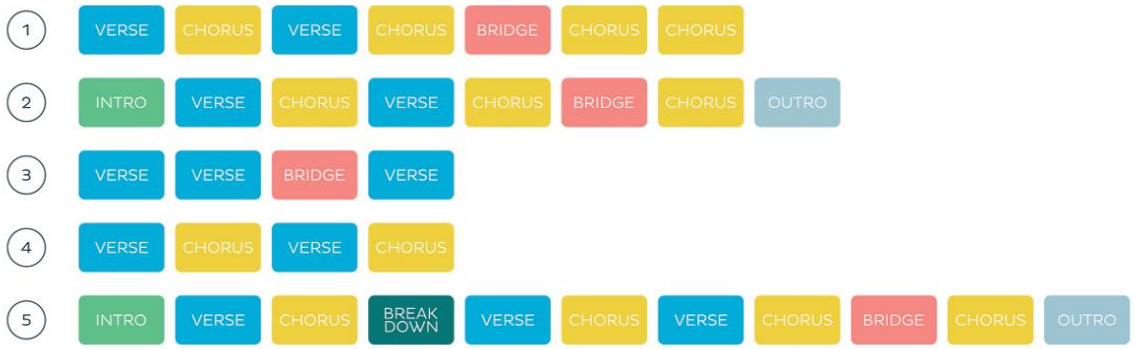
2.2. Skladanje

[1] Programsko sučelje za studijsko snimanje audio signala (engl. *Recording Studio Workstation*)

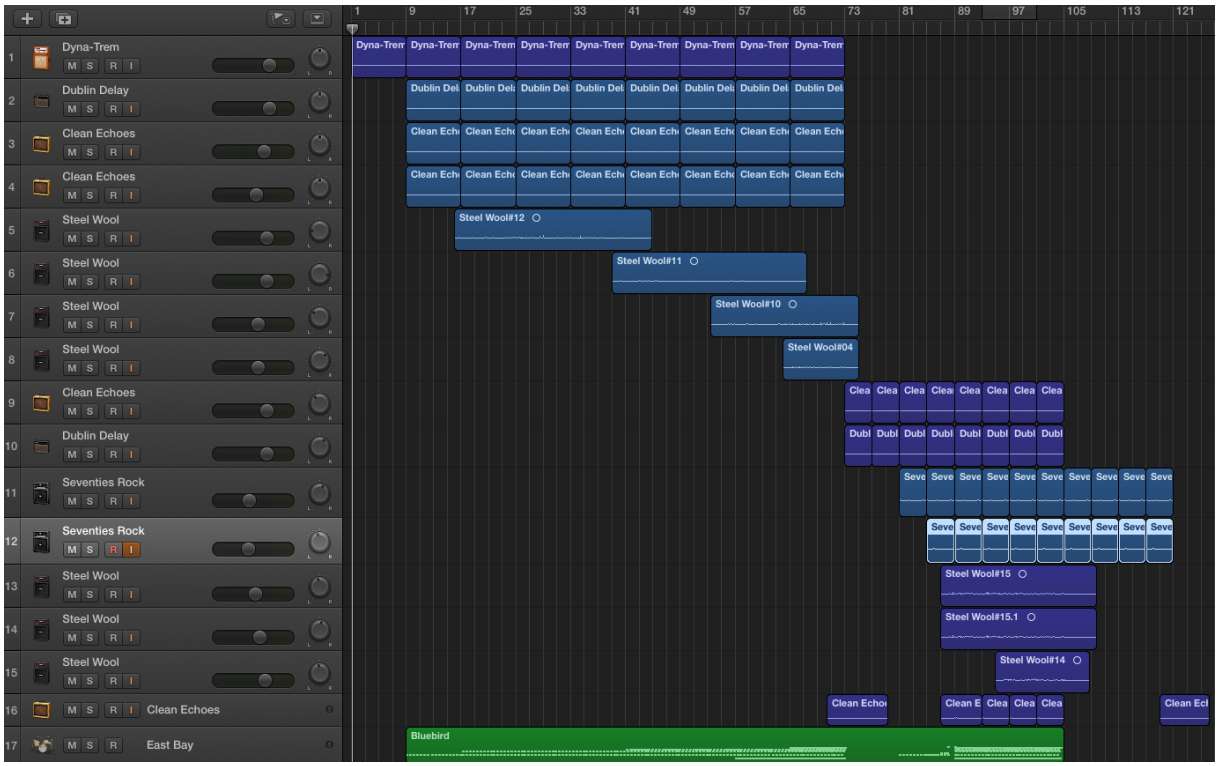
Skladanje (engl. *Composing*) je proces smišljanja i razvijanja nove glazbene kompozicije. Nakon jasno definirane ideje, počinje faza skladanja. Temeljem definiranih odredba tokom faze ideje, svaka od pjesama započinje sa skladanjem na način da se improvizacijskim sviranjem gitare traži temeljna melodija prema kojoj će se graditi ukupna glazbena kompozicija. Nakon uspješnog pronalaska temeljne melodije i dionice kompozicije, započinje se sa pronalaskom najidealnije glazbene pratnje koja uključuje akorde, glazbene prijelaze (engl. *Music gradation*), bas dionice, ritam, dinamiku i bubanj dionice. Samim time se prelazi u novu fazu koja se zove izrada kostura kompozicije.

2.3. Izrada kostura kompozicije

Kostur kompozicije (engl. *Song skeleton*) je proces grananja dijelova glazbene kompozicije u logične sljedove i prikaz istoga. Tokom ove faze definiramo mnoga načela i pravila same kompozicije, kao što su uvod, refren, završetak, dinamika, dinamičke i melodijske promjene, glazbena ljestvica, glazbena pratnja, akordi, bas dionice, ritam, bubanj dionice, glazbeni redoslijed kompozicije i još mnogo relativnih pojmova.



Slika 2.1 Najkorištenije strukture pjesama



Slika 2.2 Struktura pjesme u Apple Logic Pro X alatu

Iz iznad navedenih slika, osobito iz "Slika 2.2 struktura pjesme u Apple Logic Pro X alatu" lako možemo iščitati strukturu pjesme, te samim time dobiti okvirne informacije o dinamičkim promjenama, ritmu i glazbenom redoslijedu kompozicije, iako u tome trenu ne znamo o kojim točno glazbenim instrumentima (engl. *Musical instrument*) i žanru se radi. Na slici vidimo da je od 1. do 9. polja aktivna samo jedna dionica, čime odmah možemo zaključiti da se radi o uvodu. Nakon toga vidimo da se broj dionica postepeno povećava i izmjenjuje od polja 9 do polja 73, te taj dio možemo shvatiti kao glavni dio kompozicije prije prije refrena[2]. Na 73. polju vidimo nagli pad broja dionica čime zaključujemo da je došlo do naglog prijelaza, odnosno pred - refrena[3]. Zatim vidimo veliki porast broja dionica na polju 89 što nam govori da smo došli do glavnoga djela kompozicije, odnosno refrena. Na posljetku imamo i zaključak koji je vidljiv na polju 105 i koji je lako uočljiv jer imamo nagli pad broja dionica, te je i to zadnji strukturalni dio kompozicije.

[2] Refren (engl. *Chorus*), glavni i najisticaniji dio glazbene kompozicije
[3] Pred – refren (engl. *Pre – chorus*), uvodni dio refrena

Struktura prikazana na slici “Slika 2.2 struktura pjesme u Apple Logic Pro X“ alatu je zapravo temeljna struktura svih pjesama albuma, sa manjim odskocima.

Temeljna struktura se okvirno zadaje tokom faze skladanja, te prilikom faze razvijanja kostura kompozicije i faze samog snimanja, ona prima puni i gotovi oblik prema ranije definiranim pravilima. Nekada sami kostur, odnosno struktura kompozicije može imati manje odskoke i nepoklapanja sa ranije definiranim pravilima, što zapravo ne utječe negativno na sami proces i kvalitetu izrade kompozicija jer je u nekim slučajevima potrebno biti u samoj fazi snimanja kako bi se odredila najpogodnija struktura same pjesme.

2.4. Odabir instrumenata i ambijenta kompozicije

Na odabir instrumenata i ambijenta kompozicije utječu faktori kao što su ritam, dinamika, žanr i glazbena ljestvica (engl. *Music scale*). Odabirom različitih instrumenata i ambijenta, kompozicija može dobiti različite poruke i osjećaje koje želimo prenijeti slušaocu.

Odabirom instrumenata možemo utjecati na sami žanr, ambijent i dojam glazbene kompozicije. Na primjer, ako imamo melodiju koju želimo istaknuti kao snažnu vodeću Rock dionicu, logično je da ćemo za takav tip glazbene dionice koristiti gitaru sa distorzijom[4]. Ukoliko za tu dionicu koristimo neki drugi instrument poput flaute (engl. *Flute*), izgubit ćemo željeni dojam snažne vodeće Rock glazbene dionice. Kako odabir instrumenata utječe na žanr, ambijent i dojam glazbene kompozicije, tako i navedeni faktori koji trebaju biti unaprijed definirani, utječu na izbor instrumenata kompozicije. Iz toga zaključujemo da je ključno definirati stil, žanr, ambijent i dojam kompozicije prije samog snimanja kako bi kvalitetno mogli odabrati relevantne glazbene instrumente.

Ambijentom kompozicije također možemo utjecati na žanr i dojam glazbene kompozicije. Definiranje ambijenta zadaje se za vrijeme i nakon odabira instrumenata za određenu

[4] Distorzija (engl. *Distortion*), glazbeni efekt koji se najčešće aplicira na električnu gitaru kako bi stvorio žestoki, hrapavi Rock zvuk.

kompoziciju. Ambijent se najviše realizira glazbenom ljestvicom, ritmom, dinamikom, i odabirom instrumenata i efekata koji utječu na izvorni zvuk glazbenih instrumenata.

Kompozicija daje potpuno različiti dojam ako je temeljna glazbena ljestvica u molu[5] naspram dura[6], i obrnuto. Isto tako brži ritam i veća dinamika utječu na ambijent kompozicije na način da ju doživljavamo kao življu i veseliju što je ritam brži i dinamika veća. Na ambijent također u velikoj mjeri utječu i glazbeni efekti koji se prilikom same izvedbe, snimanja ili za vrijeme post – produkcije dodaju na izvorni zvuk određenih instrumenata. Ti efekti su najčešće *reverb*[7]*, delay*[8]*, tremolo*[9] i *distortion.* Glazbenim efektima se može utjecati i na sami žanr kompozicije, što možemo zaključiti činjenicom da je najveća posljedica utjecaja Rock žanra na neku kompoziciju upravo *distortion* efekt na električnoj gitari.

2.5. Snimanje audio materijala

Nakon što smo definirali i realizirali ideju, skladanje, kostur kompozicije, odabir instrumenata i ambijenta kompozicija, slijedi faza snimanja audio materijala.

Za snimanje audio materijala potrebni su nam razni instrumenti i alati kako bi ovu fazu mogli uspješno i kvalitetno realizirati. U mom procesu snimanja audio materijala koristio sam slijedeće alate i instrumente:

1. Električna gitara (engl. *Electric guitar*) Squier by Fender Bullet Strat
2. Električna bas gitara (engl. *Electric bas guitar*) Ibanez GSR180
3. Akustična gitara Ibanez (engl. *Acoustic guitar*) V70-NT
4. USB[10] zvučna kartica (engl. *USB sound card*) Focusrite Scarlett Solo 2nd generation
5. Mikrofon (engl. *Microphone*) Focusrite Scarlett Solo 2nd generation
6. Slušalice (engl. *Headset*) Focusrite Scarlett Solo 2nd generation
7. Laptop MacBook Pro Early 2015 macOS Sierra

[5] Mol (engl. *Minor*), dijatonska glazbena ljestvica sa početnom notom šeste note relativne dur glazbene ljestvice
[6] Dur (engl. *Major*), najkorištenija dijatonička glazbena ljestvica koja se sastoji od sedam nota
[7] Glazbeni efekt koji audio signalu daje dojam prostora
[8] Glazbeni efekt koji audio signalu dodaje zakašnjele, ponavljajuće i identične audio signale
[9] Glazbeni efekt koji audio signalu daje dojam vibracije
[10] USB (engl. *Universal Serial Bus,* skraćeno USB), univerzalni model konektora u računarstvu

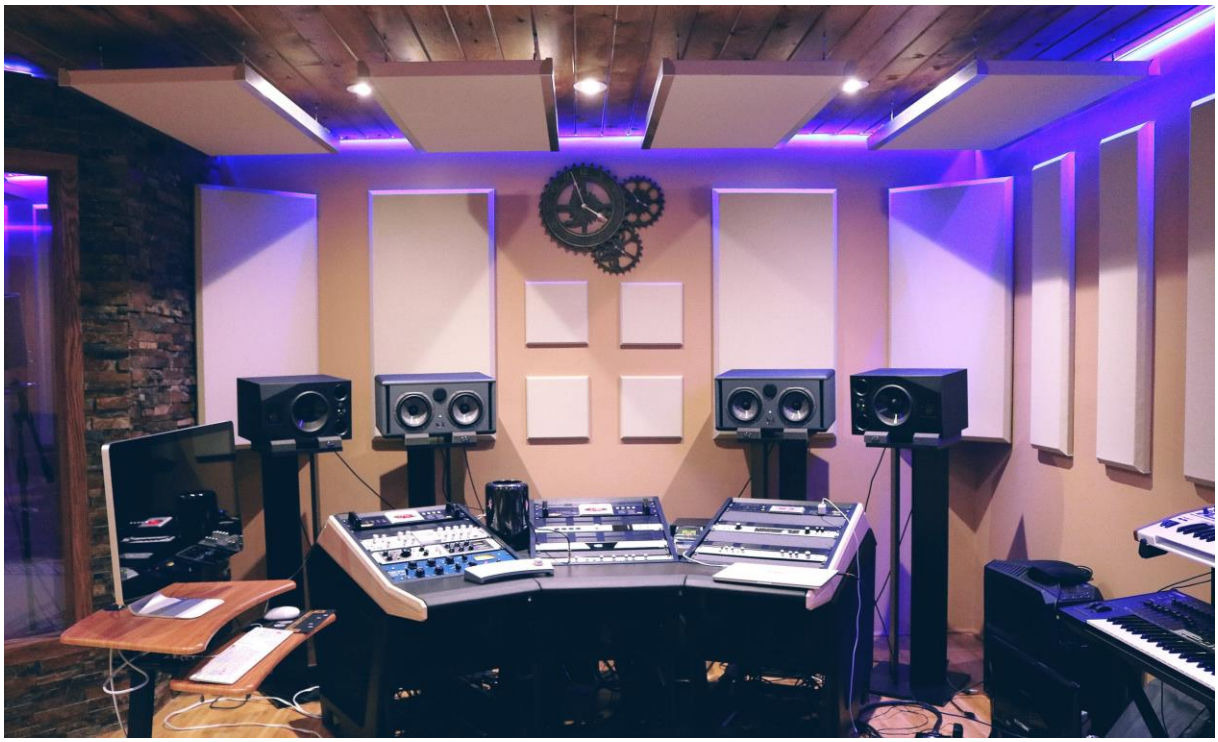
8. Programska radna stanica za snimanje zvuka Apple Logic Pro X

2.5.1 Načini snimanja audio materijala

Postoje mnogi amaterski i profesionalni načini snimanja audio materijala kao što su snimanje u profesionalnom glazbenom studiju, snimanje direktno putem eksterne zvučne kartice i snimanje izlaznog audio signala pojačala (engl. *Amplifier,* skraćeno *Amp*) putem zvučne kartice ili izravnim dinamičkim mikrofonom. Svaki način snimanja audio materijala ima svoje prednosti i mane, dok je snimanje u profesionalnom studio neosporivo najkvalitetniji način snimanja, ali zato i financijski najskuplji.

Iako sam podvornik snimanja izlaznog audio signala samog pojačala putem eksterne zvučne kartice sa zasebnim fantomskim napajanjem i kondenzatorskim mikrofonom, jer na taj način dobijemo najsličniji zvuk i ambijent kojeg bi stvorili i svirajući uživo, za ovaj rad sam se odlučio za snimanje direktno putem zvučne kartice. Razlog tome je što nam alat Logic Pro X daje iznimno veliku količinu virtualnih pojačala, instrumenata i efekata kojim možemo utjecati na ulazni audio signal što nam daje veliku slobodu pri manipulaciji audio signala.

Kao što je već ranije navedeno, različiti procesi i načini snimanja imaju zasebne prednosti i mane. Odabir načina snimanja ovisi o cilju koji se želi postići sa finalnim produktom.
Ukoliko je taj cilj profesionalno bavljenje glazbom i ciljana skupina slušaoca je šira javnost, najidealnije, čak i obvezno, je snimanje audio materijala u profesionalnom glazbenom studiju gdje je na raspolaganju dostupan i profesionalni producent. Ovakav način će vam pružiti najkvalitetniji proces snimanja albuma i željeni finalni produkt, ali zato će financijski biti najskuplji. Svaki studio ima svoje zasebne instrumente, računala, i tehnologije s kojima vrši snimanje i post – produkciju audio materijala.



Slika 2.3 Primjer profesionalnog glazbenog studija

U slučaju da se ne bavite ili nemate namjeru baviti se glazbom profesionalno, nego samostalno i amaterski snimati audio materijale, postoji više načina za realizaciju istoga.

Jedan od načina je direktno koristiti dinamički mikrofon koji snima audio izlaz pojačala ili izvorni zvuk instrumenta. Prednosti ovog načina snimanja su jednostavnost i jako mali financijski ulog, dok je mana prilično loša kvaliteta zvuka.

Drugi način je koristiti eksternu zvučnu karticu sa zasebnim fantomskim napajanjem, te kvalitetni kondenzatorski mikrofon za snimanje izvornog zvuka instrumenata. Prednosti ovog načina snimanja je puno bolja kvaliteta zvuka i manipulacija audio signala naspram direktnog dinamičkog mikrofona, dok je jedina mana veći financijski ulog.

Koji god način snimanja koristili, uvijek je potrebno imati programski alat za snimanje zvuka, a najpoznatiji su Audacity, Apple Logic Pro X, Steinberg Cubase i Avid Pro Tools.

2.5.2 Oprema za snimanje

Opremom za snimanje (engl. *Sound recording equipment*) definiramo sve alate i instrumente koji su neophodni u procesu snimanja audio signala.

Oprema za snimanje audio signala varira svojom kvalitetom i načinom snimanja. Od raznih alata i instrumenata koji se mogu koristiti za proces snimanja zvuka, uvijek je od opreme potrebno minimalno imati računalo, programski alat za snimanje zvuka i mikrofon.

Prema ranije zadanim definicijama, snimanje vlastitog glazbenog albuma je realizirano putem direktnog signala električnih instrumenata do eksterne zvučne kartice, te za snimanje akustičnih instrumenata putem kondenzatorskog mikrofona i eksterne zvučne kartice.

Za takav proces snimanja koristi se programski alat Apple Logic Pro X iz razloga što ima iznimno širok spektar virtualnih zvučnih pojačala, instrumenata i efekata, te je vrlo zahvalan i kvalitetan pri manipulaciji audio signala. Također ima korisničko sučelje i opcije koje idealno odgovaraju meni zadanom finalnom produktu i načinu snimanja.

Eksterna zvučna kartica koju sam koristio je Focusrite Scarlett Solo 2nd generation jer se sa njom dobije najveća moguća kvaliteta audio signala s obzirom na financijski ulog za istu. Kupovinom navedene zvučne kartice dobiju se slušalice i kondenzatorski mikrofon koji svojom kvalitetom idealno odgovaraju mom procesu snimanja.

Slika 2.4 Focusrite Scarlett Solo 2nd generation Studio

Najkorišteniji glazbeni instrument prilikom snimanja glazbenog albuma bila je električna gitara Squier by Fender Bullet Strat. Navedenu gitaru sam koristio iz razloga što odgovara mom stilu sviranja i žanru pjesama koje se nalaze u završnom albumu.

Slika 2.5 Squier by Fender električna gitara

U pojedinim pjesmama za dodatne prateće dionice korištena je akustična gitara Ibanez V70-NT.

Slika 2.6 Ibanez V70-NT akustična gitara

Za glavne prateće dionice i niskofrekventne audio signale u svakoj pjesmi korištena je električna bas gitara Ibanez GSR180.

Slika 2.7 Ibanez GSR180 električna bas gitara

Za računalo na kojem se nalazi programski alat za snimanje audio signala korišten je laptop MacBook Pro Early 2015 macOS Sierra koji se pokazao vrlo dobrim i pouzdanim za ovakav tip projekta.

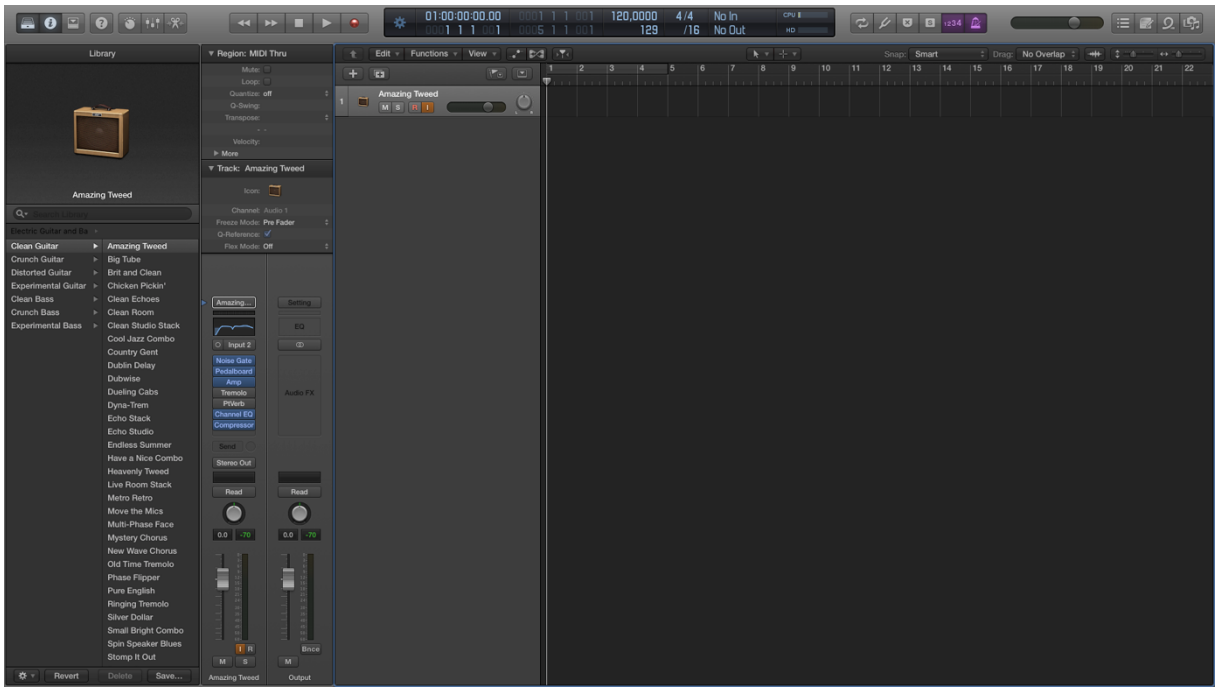
Slika 2.8 MacBook Pro Early 2015 macOS Sierra

2.5.3 Snimanje demoa

Nakon što su uspješno definirane i realizirane sve prethodne faze, dolazi dio procesa izrade glazbenog albuma gdje se izvršava snimanje audio signala.

Za uspješnu realizaciju snimanja zvuka potrebno je na računalu pokrenuti aplikacijski program za snimanje zvuka, otvoriti novi projekt i aktivirati prvu input dionicu audio ulaznog signala. Eksterna zvučna kartica mora biti spojena sa računalom za input i output audio signale jer istu koristimo za snimanje, ali i za slušanje ulaznih audio signala. Nakon uspješnog povezivanja zvučne kartice sa programom za snimanje zvuka, potrebno je spojiti instrument sa zvučnom karticom. Ukoliko se spaja električni instrument, potrebno ga je samo povezati 6.3mm[11] TRS[12] priključkom, jer će se Logic Pro X svojim virtualnim pojačalima i efektima pobrinuti za daljnju manipulaciju audio signalom na nama željeni način. Ukoliko snimamo akustični instrument, potrebno je spojiti mikrofon sa zvučnom karticom i aktivirati fantomsko napajanje[13] na zvučnoj kartici kako bi kondenzatorski mikrofon mogao dobiti napajanje potrebno za uspješan rad. Također je potrebno spojiti slušalice na zvučnu karticu, kako bi bilo moguće dobiti audio signale za vrijeme snimanja i post – produkcije. Alternativa slušalicama su zvučnici, koje ne bih preporučio za sami proces snimanja i post – produkcije iz razloga što će glazbeni instrumenti, osobito akustični, zvučati drugačije za vrijeme snimanja u odnosu na gotove snimljene audio materijale, te se time može dobiti neželjeni produkt. Apple Logic Pro X ima mogućnost slušanja obrađenog audio signala u stvarnom vremenu tokom snimanja audio signala, što rezultira identičnom zvuku tokom i nakon snimanja željene glazbene dionice. Kvalitetne slušalice će izolirati vanjske zvukove kako bi se moglo fokusirati samo na audio signal za kojeg je sigurno da će identično zvučati tokom i nakon snimanja audio signala.

[11] Milimetar (engl. *Milimeter*), mjerna jedinica za duljinu
[12] TRS (engl. *Tip, ring and sleeve,* skraćeno TRS) model priključka kabla
[13] Fantomsko napajanje (engl. *Phantom Power*), interno napajanje zvučne kartice za kondenzatorske mikrofone



Slika 2.9 Početno stanje novokreiranog projekta u programu Logic Pro X

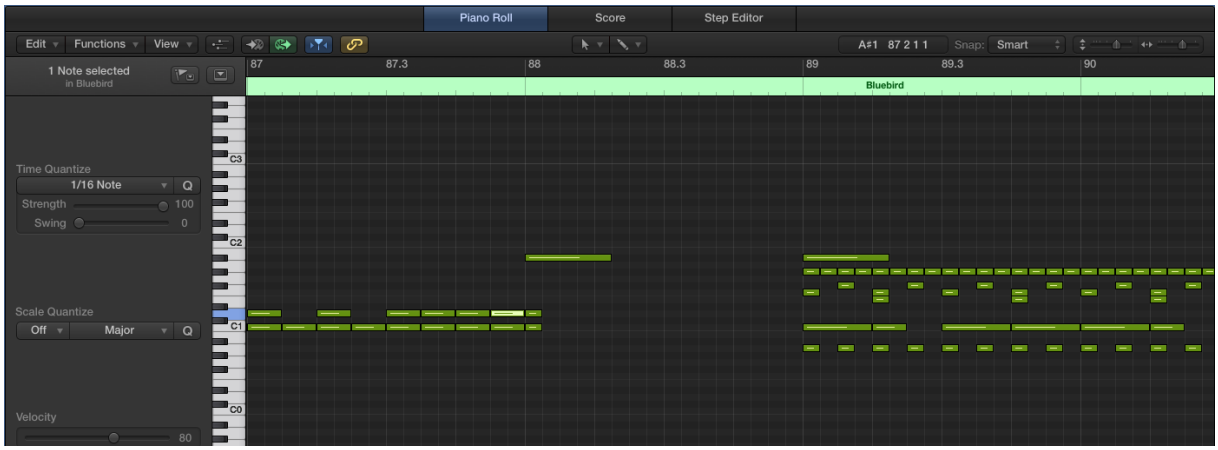
Kako su skladanje, struktura i ambijent pjesama ranije definirani, temeljem tih definicija se odabire brzina i takt[14] pjesme unutar novokreiranog projekta. Brzinu i takt pjesama potrebno je odabrati iz razloga što Logic Pro X može raditi na temelju taktova pjesama što omogućuje laku manipulaciju glazbenim dionicama unutar projekta, te osobito izradu ponavljajućih petlja glazbenih dionica. Takva opcija rezultira velikom uštedom vremena jer dionice koje se ponavljaju unutar pjesme nije potrebno svirati kroz cijelu kompoziciju, nego ih se na vrlo lak i pouzdan način postavi na željene ponavljajuće taktove.

Prije početka snimanja glazbene dionice potrebno je odabrati odgovarajući audio input eksterne zvučne kartice koji će zaprimati željene audio signale. Nakon što smo podesili potrebne zahtjeve, potrebno je samo kliknuti gumb za snimanje unutar programskog sučelja ili pritisnuti tipku “R“ na tipkovnici za početak snimanja aktivne glazbene dionice projekta.

[14] Takt (engl. *Tact*), ponavljajući period vremena kompozicije na kojem se temelji vrijeme i ritam

Za dodavanje nove glazbene dionice potrebno je kliknuti gumb namijenjen toj funkciji unutar programskog sučelja, te odabrati željeni oblik i stil dionice. Odabir virtualnih pojačala i efekata vrši se na lijevoj strani programskog sučelja gdje imamo mnogobrojan izbor raznih rješenja kao što su razni predefinirani zvukovi električne i akustične gitare i pripadajućih pojačala i efekata.

Za odabir virtualnih instrumenata potrebno je tokom kreiranja nove dionice odabrati *"Software Instrument"* opciju, te nakon kreiranja dionice odabrati jedan od virtualnih instrumenata kao što su klavijature, električna gitara, violončelo, viola, bubnjevi, itd. Kako virtualne instrumente ne možemo snimati istim načinom kao realne glazbene instrumente, njihove dionice se pišu unutar samog programskog sučelja programa za snimanje zvuka. To je izvedivo na način da uključimo opciju *"Piano Roll"* koja nam otvara novi aplikacijski prozor unutar kojeg je moguće pisati note, odnosno cjelokupne glazbene dionice za određeni virtualni instrument.



Slika 2.10 Primjer *"Piano Roll"* prozora unutar Logic Pro X projekta

2.6. Post produkcija audio materijala

Post – produkcija audio materijala je finalna faza procesa snimanja zvučnog signala. Za vrijeme ove faze se manipulira snimljenim audio signalima tako da se isti uređuju na način koji nije moguć tokom samog procesa snimanja. Unutar post – produkcije spadaju pod – faze dosnimavanje, montaža, miksing i mastering.

2.6.1 Dosnimavanje

Dosnimavanje je pod-faza post – produkcije gdje se vrši snimanje dodatnih glazbenih dionica nad temeljnom snimljenom glazbenom matricom. Najčešći način i proces snimanja u profesionalnom glazbenom studiju je da se istovremeno snimaju svi članovi nekog glazbenog sastava kako bi se inicijalnom snimkom dobila temeljna glazbena matrica više različitih glazbenih instrumenata kao što su na primjer gitara, bas gitara, klavijature, bubnjevi i ljudski vokal. Nakon inicijalne glazbene matrice vrši se dosnimavanje kako bi se dodale dodatne glazbene dionice poput više gitarskih dionica istovremeno.

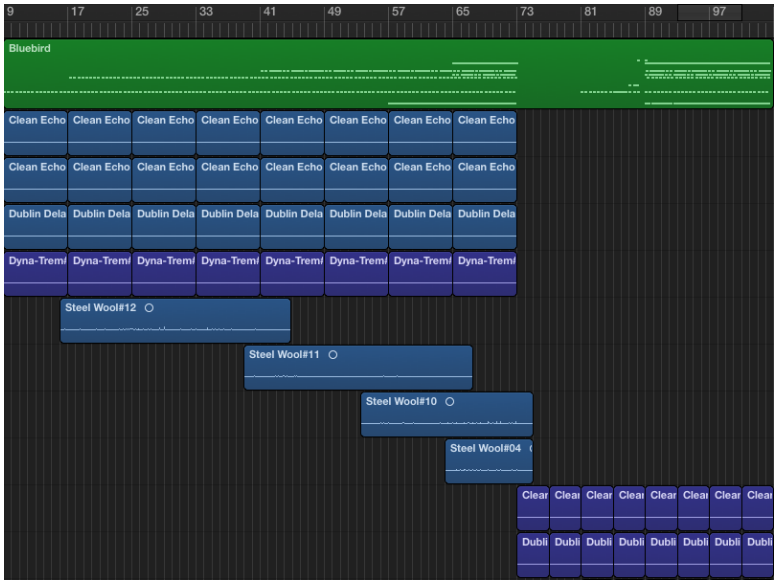
Na vlastitom glazbenom albumu proces dosnimavanja odrađuje se na poprilično drugačiji način jer je samo jedna osoba uključena u proces snimanja glazbenog albuma, tako da nije moguće kao rezultat inicijalne glazbene matrice dobiti više glazbenih dionica, nego samo jednu. Prema tome se može smatrati da u slučaju kada jedna osoba snima glazbeni album, svako snimanje dionice može se smatrati dosnimavanjem.

Kako imamo samo jednu osobu aktivnu u procesu snimanja glazbenog albuma i možemo snimati maksimalno jednu glazbenu dionicu istovremeno, od velike je važnosti znati koji instrument ćemo koristiti kao prvu glazbenu dionicu. Gledajući strukturu većine pjesama, lako je zaključiti da je najlogičnije kao prvu dionicu snimiti bubnjeve, a zatim neki od glavnih pratećih instrumenata kao što je u mom slučaju bas gitara. Razlog tome je što bubnjevima dobivamo ritam i dinamiku bez koje ne možemo znati brzinu i takt pjesme, dok nam bas gitara daje temeljnu pratnju i ljestvicu kako bi se lakše i kvalitetnije moglo odraditi snimanje vodećih glazbenih dionica kao što je vodeća gitara. U slučaju da imamo pjesmu u kojoj se nalaze dijelovi bez bubanj dionica, ili pjesma uopće nema bubanj kao jedan od glazbenih instrumenata, obvezno je uključiti metronom[15] unutar Logic Pro X aplikacije kako bi se dobio osjećaj za brzinu i ritam pjesme.

[15] Metronom (engl. *Metronome*), sprava za mjerenje brzine ritma glazbene kompozicije

2.6.2 Montaža

Montaža (engl. *Montage*) u smislu procesa snimanja zvuka je povezivanje, raspoređivanje i ponavljanje više glazbenih dionica unutar jedne cjelokupne glazbene kompozicije. Radeći sa alatom Logic Pro X montaža igra veliku ulogu iz razloga što spomenuti programski alat jako dobro radi sa dionicama na temelju vremena i taktova, te zato što imamo veliku mjeru korištenja procesa montaže jer se većina snimanja odvija na način dosnimavanja u određenim taktovima.



Slika 2.11 Primjer montaže i ponavljajućih petlji dionica u alatu Logic Pro X

Iz iznad prikazane slike uočljiv je redoslijed snimanja glazbenih dionica. Prva dionica je bubanj, zatim slijede dvije dionice bas gitare, te dvije prateće dionice električne gitare. Ovdje je vidljivo pravilo koje smo ranije definirali da kao prvu dionicu snimamo bubanj, te temeljnu prateću dionicu bas gitare, zatim slijedi sporedna pratnja električne gitare, i kao najviši nivo imamo vodeće dionice električne gitare. Također je uočljivo ponavljanje identičnih dionica kroz taktove u pratećim dionicama prikazane kompozicije.

2.6.3 Miksing

Miksing (engl. *Miksing*) je pod – proces post – produkcije audio materijala gdje se više glazbenih dionica kombinira unutar jedne ili više dionica, odnosno kanala. To nam omogućuje kvalitetniju obradu cijele pjesme kao jedne cjelokupne cjeline. U ovoj fazi se često audio signali obrađuju u području ujednačenosti, niveliranja i kompresije audio signala. Navedene obrade mogu u velikoj mjeri utjecati na finalnu kvalitetu kompozicije.

Slika 2.12 Primjer jednostavne miksing konzole

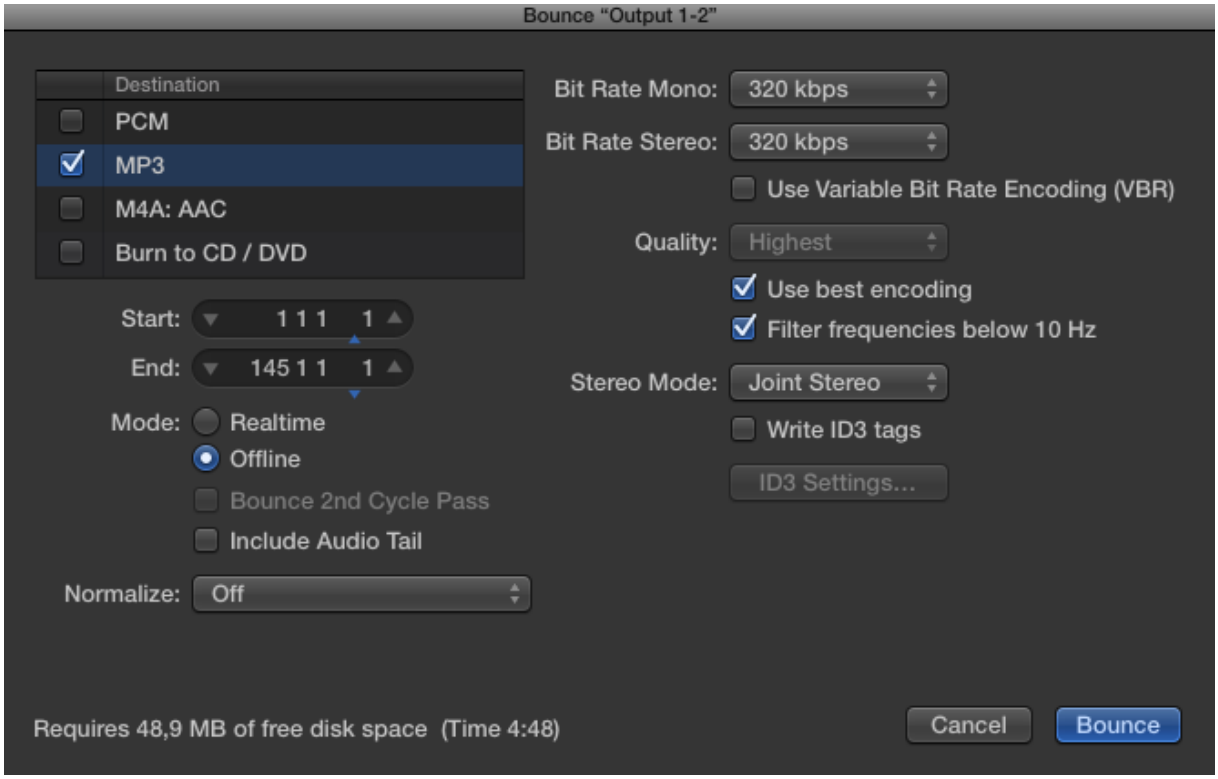
U svijetu audio miksinga, veliku pozitivnu revoluciju doveo je novitet višekanalno snimanje. Do tada su se cjelokupne kompozicije snimale unutar jednog kanala, odnosno kao jedna zasebna dionica. Sa takvim načinom snimanja dolazi do velikih poteškoća jer nije bilo moguće ili je bilo iznimno zahtjevno izvršiti dosnimavanje, dionice različitih glazbenih instrumenata se nisu mogli izolirati za kvalitetniju i specifičniju obradu audio signala, te nije bilo moguće snimiti album sa samo jednom osobom uključenom u proces kao što sam ja snimao, jer se dionice dodaju kao zasebni audio signali i kanali.

Radeći sa programskim alatom za snimanje zvuka kao što je Logix Pro X, relativno je lako odraditi fazu miksinga. U takvim radnim stanicama razni efekti i spajanje kanala može se raditi na nivou koji sami lako zadate. Ukoliko je to zasebni kanal, više kanala ili cjelokupna kompozicija nad kojom želite vršiti proces miksinga, potrebno je samo odabrati željene kanale i dionice na koje će utjecati proces miksinga.

2.6.4 Mastering

Audio mastering je finalni pod – proces post – produkcije gdje se vrši pripremanje snimljenog audio materijala za izvoz, te sami izvoz audio materijala u željene formate i medije sa definiranim svojstvima finalnog audio produkta kojeg nazivamo *master*.

Unutar programskog alata Logic Pro X mastering vršimo na način da nakon što smo završili fazu miksinga, uključili željene efekte na određene kanale i dionice, te ispravili sve nepravilnosti audio signala ukoliko ih je bilo, radimo izvoz cjelokupne glazbene kompozicije navigacijom kroz programsko sučelje do gumba *"Bounce"* ili istovremenim pritiskom tipki "CMD" i "B". Tom radnjom otvara nam se novi programski prozor gdje biramo razne formate, kvalitete i svojstva glazbene kompozicije za izvoz finalnog produkta.



Slika 2.13 *"Bounce"* prozor za izvoz finalnog audio produkta

Na iznad prikazanoj slici vidimo svojstva i kvalitetu audio formata kojeg sam koristio za izvoz svih svojih pjesama uključenih u navedeni glazbeni album. Format audio datoteke je

MP3[16]. Kvaliteta je maksimalna koju nam Logic Pro X pruža za ovaj format, a to je 320 kbps[17], te ima stereo svojstvo kako bi se u finalnom produktu primili efekti niveliranja stereo vrijednosti.

Razlog zašto sam se odlučio za ovakav format finalnog audio izvoza je zato što je sa tim formatom moguće dobiti željenu kvalitetu sa manjom veličinom datoteke u odnosu na ostale audio formate. Isto tako ciljani web[18] preglednici (engl. *Web Browser*) za web aplikaciju podržavaju ovaj audio format.

[16] Digitalni audio format
[17] kbps (engl. *Kilobit per second,* skraćeno kbps), jedinica kvalitete i brzine audio transfera
[18] web (engl. *Web*), kratica za *World Wide Web* poput www

3. Izrada web aplikacije

Za prikaz gotovih snimljenih pjesama, odnosno gotovog završnog glazbenog albuma i procesa stvaranja istog, izrađujemo web aplikaciju prema ciljanoj publici i predodređenim načinom i pravilima, koristeći današnje moderne i najzastupljenije web development tehnologije.

3.1. Ideja

Temeljna ideja i cilj web aplikacije je prikazati glazbeni album korisnicima na interaktivan način, gdje će svaku pjesmu glazbenog albuma moći zasebno komentirati, pustiti, preuzeti i ocijeniti, te će moći naći informacije koje su potrebne kako bi i sami korisnici dobili uvodno i okvirno znanje za uspješno samostalno kreiranje vlastitog glazbenog albuma.

Pored funkcionalnosti koje će web aplikacija pružati korisniku, u cilju nam je također i razviti sustav na način da samom administratoru aplikacije bude što brži i jednostavniji uvoz snimljenih pjesama u sustav web aplikacije.

3.2. Proces kreiranja web aplikacije

Kako bi što kvalitetnije i u što kraćem vremenskom periodu proveli razvoj i produkciju web aplikacije, iznimno je važno prije samog razvoja definirati određena pravila po kojima ćemo razvijati web aplikaciju. Ta pravila se nazivaju planovi i utječu na razne aspekte web rješenja.

3.2.1 Plan strategije

Planom strategije (engl. *Strategy plan*) uključujemo istraživanje koje rezultira informacijama što korisnici žele vidjeti i pronaći na našem web rješenju, te samim time i što mi kao kreatori želimo pružiti korisnicima putem našeg web rješenja.

Cilj navedenog web rješenja je prikazati korisnicima gotovi glazbeni album na interaktivan način, te ih time potaknuti na dulji boravak i što veću interakciju na aplikaciji kako bi što veći postotak korisnika napravio radnje poput komentiranja, ocjenjivanja, preuzimanja, te informiranja o izradi i razvoju glazbenog albuma.

Identitetom branda, odnosno glazbenog albuma u ovom slučaju, želimo korisnicima kreirati emotivne reakcije. Većina vizualnih elemenata mora korisniku davati pozitivni dojam da se radi o aplikaciji koja na korisnike ne gleda kao izvor profita, nego da je najveća uloga potaknuti korisnika na jednu od interakcija vezanih za cilj aplikacije kao što su slušanje, komentiranje, ocjenjivanje i preuzimanje pjesama. Korisnika želimo zainteresirati za ono što mu nudimo preko web rješenja, bez da pomisli da sama aplikacija gleda na njega kao jednu od vrsta zarade. Cilj UX[19]-a i UI[20]-a je vizualno držati se teme glazbenog albuma.

3.2.2 Plan opsega

Planom opsega (engl. *Scope plan*) definiramo od kojih će se sve elemenata i funkcionalnosti web rješenje sastojati. S obzirom na ideju i cilj web rješenja predviđamo potrebne funkcionalnosti istoga. Važno je znati da je plan opsega određen planom strategije.

Ranije utvrđenim temeljima plana strategije definiramo sljedeće funkcionalnosti web rješenja.

Kako su jedne od temeljnih funkcionalnosti web aplikacije ocjenjivanje i komentiranje pjesama glazbenog albuma, neizbježno je razviti internu društvenu mrežu kako bi razlikovali korisnike i njihove interakcije poput komentiranja i ocjenjivanja zasebnih pjesama unutar web rješenja. Kako bi to bilo ostvarivo, potrebna je implementacija registracije i identifikacije korisnika, za što je odlučeno da će se odraditi preko Google Sign-In[21]

[19] UX (engl. *User Experience*, skraćeno UX), definira razinu korisnikova zadovoljstva grafičkim sučeljem
[20] UI (engl. *User Interface*, skraćeno UI), kratica za korisničko sučelje
[21] Metoda registracije korisnika preko Google sustava

integriranog sustava, odnosno JavaScript open-source[22] paketa vue-google-auth iz razloga što će klijentska strana aplikacije biti pisana u JavaScript okružju Vue.js. Nama najbitnije informacije i funkcionalnosti koju nam nudi vue-google-auth[23] su idToken[24] korisnika, odnosno unikatni skup karaktera za svakog od korisnika koji nam omogućuje identifikaciju kroz back – end[25] MySql[26] bazu podataka, te automatiziranost registracijskog sustava korisnika koju umjesto aplikacije odrađuje Google API[27] REST[28]. Kako bi riješili problem da korisnik ne mora svakim dolaskom raditi registraciju, određene vrijednosti će se spremati u lokalno skladište (engl. *Local storage*) web preglednika.

```
updateCurrentUser(context, payload) {

    const googleUser = payload.googleUser;

    localStorage.zavrsniRadUser = JSON.stringify(googleUser);

    const request = {
        firstName: googleUser.firstName,
        lastName: googleUser.lastName,
        googleId: googleUser.googleId
    };

    Vue.http.post(context.rootGetters.login, request)
        .then(response => response.json())
        .then(response => {

            console.log(response);

            context.commit('updateCurrentUser', {
                currentUser: response
```

[22] Programsko rješenje kojem je javno dostupan izvorni aplikacijski kod
[23] Vue.js eksterni instalacijski paket za provedbu Google Sign-In sustava
[24] Identifikacijska unikatna vrijednost korisnika unutar Google Sign-In sustava
[25] Serverska strana programskih rješenja
[26] Programski jezik za menadžment vrijednostima baze podataka
[27] API (engl. *Application Programming Interface,* skraćeno API), programsko sučelje aplikacije
[28] REST (engl. *Representational state transfer*, skraćeno REST), web servis koji omogućuje komunikaciju više aplikacija

```
            });

            EventBus.$emit('userUpdated');

        }, error => {
            console.error(error);
        });

}
```

Kôd 3.1 Registracijski sustav sa vue-google-auth i vue.js okružjem

Vjerojatno najvažnija i najistaknutija funkcionalnost web rješenja bi zasigurno bila mogućnost sviranja pjesama albuma, jer se sve ostale funkcionalnosti temelje upravo na tome. Za implementaciju navedene funkcionalnosti potrebno je razviti audio player[29] koji će biti implementiran tehnologijama JavaScript (ES6), Vue.js, CSS (SCSS), i HTML5.

```
initSong() {

    this.song = new Audio(this.songObject.url);
    this.song.load();

    this.song.addEventListener('loadedmetadata', () => {
        this.songDuration = this.song.duration;
    });

}
```

Kôd 3.2 Inicijalizacija pjesmama na klijentskoj strani aplikacije

Ocjenjivanje korisnika će biti implementirano na način da će svaki od korisnika biti u mogućnosti ocijeniti pjesmu od 1 – 10. Nakon ocjenjivanja, HTTP[30] zahtjevom će se sa klijentske strane slati JSON[31] objekt koji sadrži ocjenu, identifikacijsku informaciju korisnika i pjesme na eksternu PHP skriptu koja će spremati informacije u back-end MySql bazu podataka. Svaki korisnik je u mogućnosti zasebnu pjesmu ocijeniti maksimalno

[29] Programska logika koja omogućuje sviranje i upravljanje audio datotekama kroz programsko sučelje
[30] HTTP (engl. *Hypertext Transfer Protocol,* skraćeno HTTP), aplikacijski protokol za komunikaciju putem *World Wide Web-a*
[31] JSON (engl. *JavaScript Object Notation,* skraćeno JSON), format vrijednosti za transport objekata

jednom, iz razloga što je zaključeno da bi mogućnost neprestanog ocjenjivanja jednog korisnika iste pjesme bila loša funkcionalnost. Svaki od korisnika je u mogućnosti vidjeti ukupnu prosječnu ocjenu svih pjesama.

```
gradeSong(context, payload) {

    const grade = payload.count;
    const songId = payload.songId;
    const currentUser = context.rootGetters.currentUser;

    const gradeSongRequest = {
    userId: currentUser.id,
        songId,
        grade
    };

    Vue.http.post(context.rootGetters.gradeSong,
gradeSongRequest)
        .then(response => response.json())
        .then(response => {
            console.log(response);
            context.dispatch('getSongGrades', {
                songId
            });
        }, error => {
            console.error(error);
        });

}
```

Kôd 3.3 Ocjenjivanje pjesama na klijentskoj strani

Komentiranje pjesama će također biti implementirano kao i ocjenjivanje, na način da će se HTTP zahtjevom sa klijentske strane aplikacije slati JSON objekt koji sadrži tekst komentara, identifikacijsku vrijednost korisnika i pjesme na PHP skriptu koja će primljene vrijednosti spremati u MySql bazu podataka. Svaki od korisnika je u mogućnosti neograničeno komentirati, te će korisnicima biti vidljivi svi komentari i imena korisnika koji su komentirali određene pjesme.

```
commentSong(context, payload) {

    const songId = payload.songId;
    const comment = payload.comment;
    const userId = context.rootGetters.currentUser.id;

    const request = {
        songId,
        comment,
        userId
    };

    Vue.http.post(context.rootGetters.commentSong, request)
        .then(response => response.json())
        .then(response => {

            context.dispatch('updateSongComments', {
                songId
            });

            EventBus.$emit('newCommentAdded');

        }, error => {
            console.error(error);
        });

}
```

Kôd 3.4 Zahtjev novog komentara na klijentskoj strani

Kako bi sve prethodne funkcionalnosti vezane za pjesme albuma bilo moguće realizirati, potrebno je razviti sustav za uvoz pjesama u web rješenje. Za tu funkcionalnost biti će kreirana posebna PHP skripta kojoj omogućen pristup ima samo administrator web aplikacije. PHP skripta će kroz HTML5 formu primati MP3 datoteku i tekst koji sadrži informacije o žanru pjesme koju uvozimo u aplikaciju. Navedena skripta će iz primljenih vrijednosti dobiti podatke o veličini, ukupnom trajanju, ime, putanju, te žanr uvezene datoteke. Skripta će navedene podatke spremati u bazu podataka, te će također spremati

cjelokupnu audio datoteku u zasebni folder kako bi istu mogli dohvatiti za klijentske strane aplikacije.

```
if (isset($_POST['submit'])) {

    $files = getMultiple_FILES($_FILES['images'])['images'];
    $genre = $_POST['genre'];

    pre($files);

    foreach ($files as $file) {

        $mp3file = new MP3File($file['tmp_name']);
        $duration2 = $mp3file->getDuration();
        $songDuration = substr(MP3File::formatTime($duration2),
3);

        move_uploaded_file($file['tmp_name'], "../songs/" .
$file['name']);
        saveSongToDatabase($file['name'], $dbPath .
$file['name'], $file['size'], $genre, $songDuration);

    }

}
```

Kôd 3.5 Glavna logika PHP skripte za uvoz nove pjesme

Zadnja primarna funkcionalnost web rješenja je preuzimanje audio datoteke na lokalni uređaj, što će biti implementirano na način da se na HTML5 link element kreiran JavaScript sintaksom zakače atributi *"download"*, te *"href"* kojem je vrijednost apsolutna putanja do određene audio datoteke.

```
downloadSong() {

    let aEl = document.createElement('a');
    aEl.setAttribute('href', this.songObject.url);
    aEl.setAttribute('download', this.songObject.name);
```

```
        aEl.click();

    }
```

Kôd 3.6 Primjer koda koji omogućuje preuzimanje pjesme

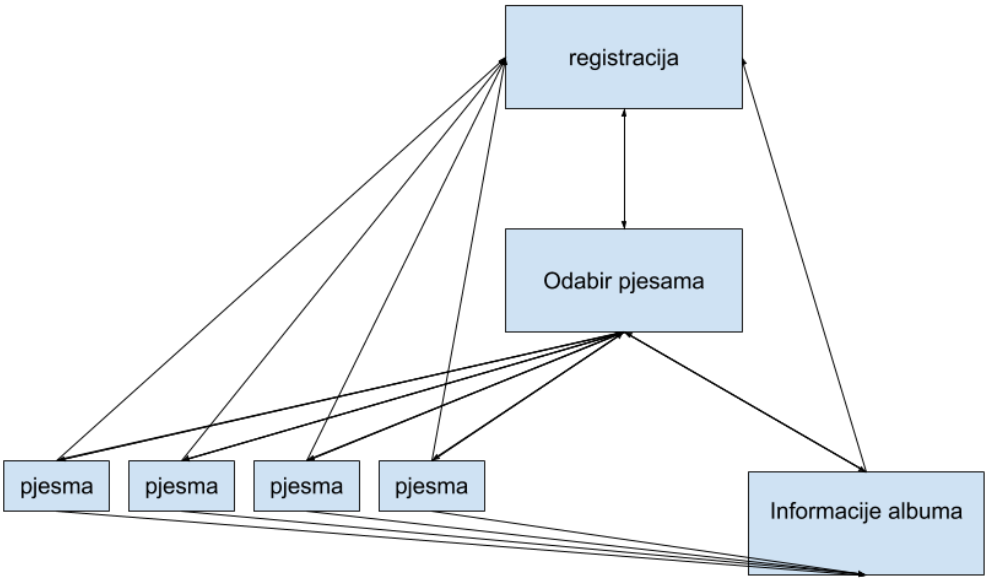
Kako je web rješenje većinski interaktivnog tipa, nije stvorena potreba za velikom količinom sadržaja na našoj aplikaciji. Sadržaj koji će se nalaziti unutar aplikacije biti će informativnog tipa prema korisniku da mu okvirno prikaže proces snimanja albuma kako bi korisnici mogli sami započeti sa razvojem sličnog tipa projekta. Od ostalog primarnog sadržaja aplikacija će sadržavati audio datoteke sa prethodno snimljenog glazbenog albuma i razne vrijednosti vezane za isto.

3.2.3 Struktura stranice

Planom strukture (engl. *Structure Plan*) definiramo razvoj konceptualne strukture web rješenja, korisnikova očekivanja o informacijama koje mogu pronaći na web rješenju, moguća ponašanja korisnika u sustavu web rješenja, te odgovor sustava na korisnikove očekivane ili neočekivane interakcije unutar web rješenja.

Informacijskom strukturom definiramo što korisnici očekuju od informacija, gdje se nalaze, kako doći do njih unutar naše strukture web rješenja, te kreiranje organizacijskih i navigacijskih ruta i shema koje vode do određenog sadržaja.

Tokom ranijih planova je već utvrđeno da će se aplikacija razvijati u smjeru kako bi korisnici od sadržaja mogli očekivati edukativne informacije o procesu izrade glazbenog albuma, te snimljene pjesme glazbenog albuma nad kojima će moći vršiti razne interakcije.



Slika 3.1 Navigacijska shema web rješenja

Kako je prikazano na slici iznad, zaključivo je da će se pri ulasku u aplikaciju kao prva ruta korisniku pružiti registracijska stranica, te nakon uspješne registracije korisnik se automatski prebacuje na rutu za odabir pjesama, gdje može birati zasebnu pjesmu kao novu rutu, ili rutu za informacije o albumu. Od velike je važnosti da korisnik bude u mogućnosti odjave na svakoj ruti aplikacije, te da se nakon uspješne odjave korisnik automatski prebaci na početnu rutu za registraciju.

Interakcijskim dizajnom opisujemo moguća ponašanja korisnika i odgovor sustava web rješenja na korisnikova ponašanja, odnosno interakcije. Interakcijski dizajn se dijeli na:

1. Konceptualne modele, odnosno očekivane korisničke interakcije
2. Upravljanje greškama, odnosno neočekivane korisničke interakcije

Od konceptualnih modela očekujemo uspješnu registraciju i odjavu korisnika, uspješnu komunikaciju klijentske i serverske strane aplikacije pod što spadaju sve CRUD[32] operacije, navigaciju kroz aplikaciju putem glavnog menija, odabir određenih pjesama i sve vezane

[32] CRUD (engl. *Create Read Update Delete,* skraćeno CRUD), definira skup programskih operacija

interakcije kao što su sviranje, ocjenjivanje, komentiranje i preuzimanje pjesama. Također sa administratorske strane očekujemo uvoz novih datoteka i vrijednosti nad kojim se temelji čitava aplikacija.

Od upravljanja greškama, očekujemo nevažeću registraciju od strane korisnika, nepovezanost ili propalu komunikaciju između serverske i klijentske strane aplikacije, pokušaj korisnika da dođe do nedozvoljenih ruta unutar aplikacije i neželjene interakcije kao što su ocjenjivanje iste pjesme više puta i dodavanje praznih komentara.

U slučaju nevažeće registracije ili prijave korisnika u aplikaciju, većinu upravljanja greškama izvodi Google API REST, preko kojega implementiramo prvi sloj registracije. U slučaju da dođe do greške na razini same aplikacije i njenih serverskih skripta, kao odaziv prema klijentskoj strani šaljemo negativni status i okvirni opis greške do koje je došlo, te klijentska strana aplikacije reagira na te odazive na određene načine kao što su nedozvoljavanje korisniku da se locira u sljedeću rutu, ili forsiranje korisnikove lokacije nazad na registracijsku rutu.

```
function insertNewUser($firstName, $lastName, $googleId)
{

    global $conn;

    $sql = "INSERT INTO korisnici (firstName, lastName,
googleId)
            VALUES ('$firstName', '$lastName', '$googleId')";

    $result = $conn->query($sql);

    if ($result) {

        getUserIdAndRespond($googleId);

    } else {
```

```
            ej([
                'desc' => 'USER_NOT_INSERTED',
                'status' => false,
            ]);

        }

    }
```

Kôd 3.7 Primjer vraćanja odaziva u slučaju greške na PHP skripti

U slučaju da klijentska strana aplikacije dobije objekt kojem je statusna vrijednost negativna, klijentska skripta reagira na to raznim metodama, ograničenjima i limitiranjem prema trenutnom korisniku.

```
EventBus.$on('updateStarsOnStart', grade => {
    if (grade === false || grade === 0) {
        return;
    }
    else {
        this.updateStars(grade);
    }
});
```

Kôd 3.8 Primjer reakcije klijentske strane u slučaju negativnog odgovora PHP skripte

U iznad ispisanom kodu je vidljivo da ukoliko je dohvaćena negativna vrijednost ocjene, željena metoda se neće izvršiti. Također se pri samom dohvaćanju odgovora sa servera ispisuje greška u konzolu, ukoliko je do greške došlo.

```
getAllSongs(context) {

    Vue.http.get(context.rootGetters.getAllSongs)
        .then(response => response.json())
        .then(response => {

            context.commit('getAllSongs', {
                songs: response.songs
            });
```

```
        }, error => {
            console.error(error);
        });

    }
```

Kôd 3.9 Direktni ispis serverske greške u konzolu pri dohvatu odgovora

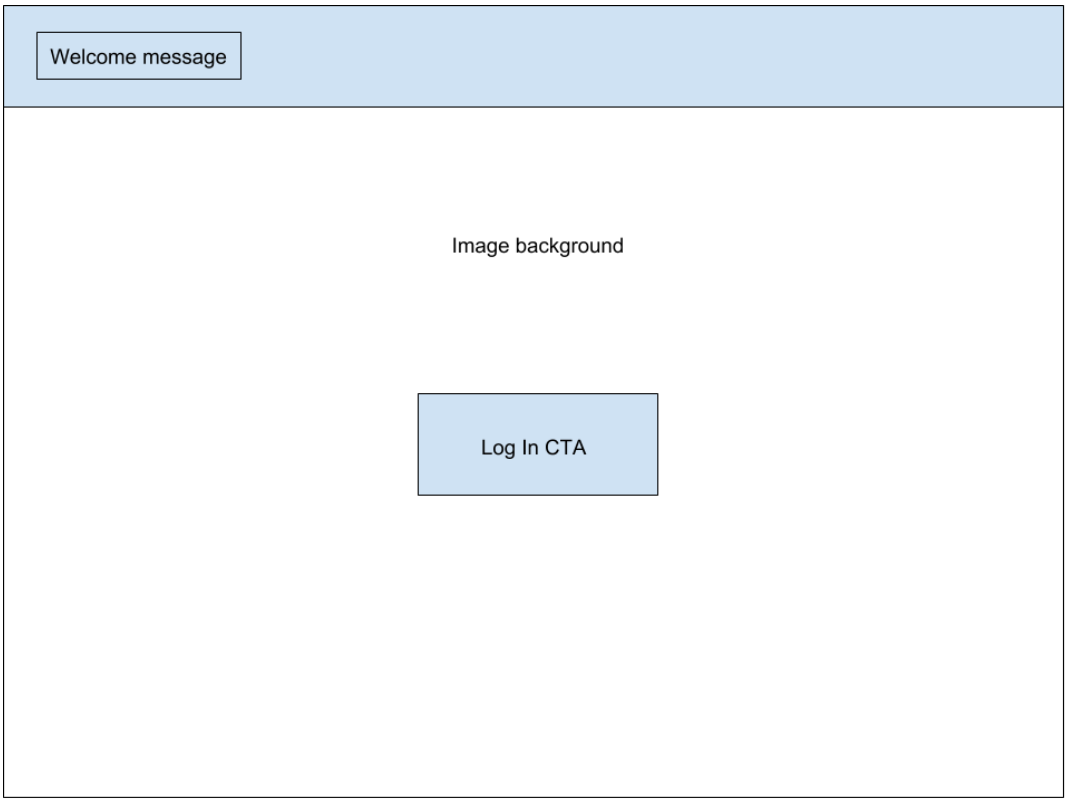
Kako treba biti oprezan i izvršiti validaciju podataka u smjeru serverske strane prema klijentskoj, isto treba napraviti i u slučaju obrnutog smjera kako bi osigurali da serverske skripte ne prime neželjene vrijednosti. Iako se validacija radi i na serverskom nivou, uvijek je poželjno napraviti validaciju i na klijentskoj strani prije samog HTTP AJAX[33] zahtjeva prema serverskim skriptama.

3.2.4 Kostur stranice

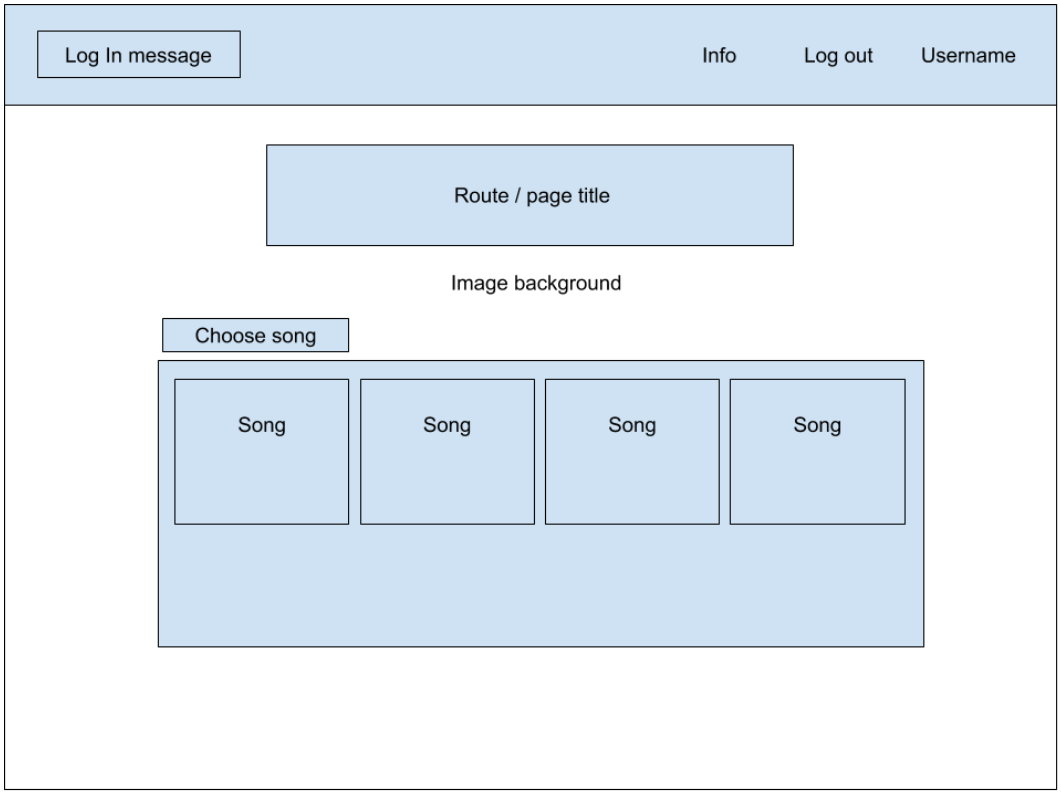
Planom kostura (engl. *Skeleton plan*) stranice detaljnije definiramo i razvijamo strukturu, dizajnerske zahtjeve, navigaciju, i prezentiranje informacija prema korisnicima od strane web rješenja. Svaka zasebna definicija kostura stranice se prikazuje kao vizualni element naziva *wireframe*. Kostur stranice se dijeli na tri djela:

1. Korisničko sučelje koje definira razne komponente sučelja, odnosno vizualni okvir
2. Dizajn navigacije koji definira putanje korisnika kroz web rješenje
3. Dizajn informacija koji definira predstavljanje i grupiranje informacija za efektniju svrhu komunikacije web rješenja sa korisnikom

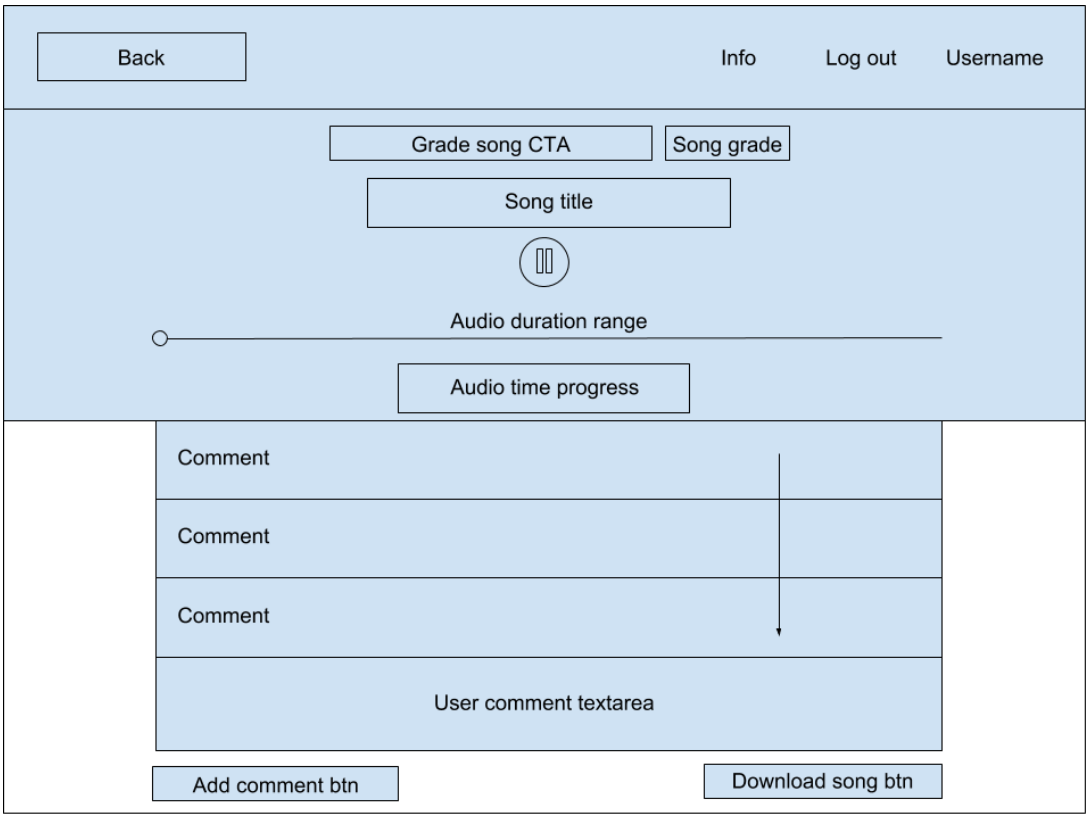
[33] AJAX (engl. *Asynchronous JavaScript And XMLHttpRequest,* skraćeno AJAX), metoda komunikacije klijentske i serverske strane web programskih rješenja



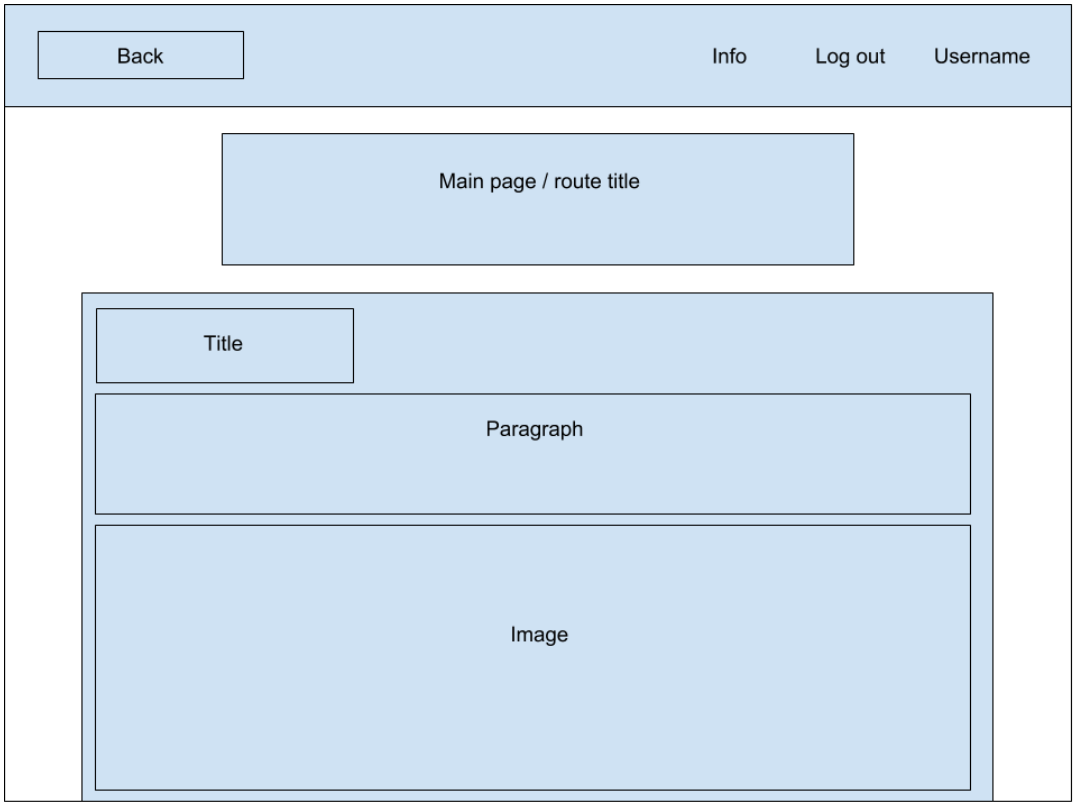
Slika 3.2 *Wireframe* početne stranice za registraciju i prijavu



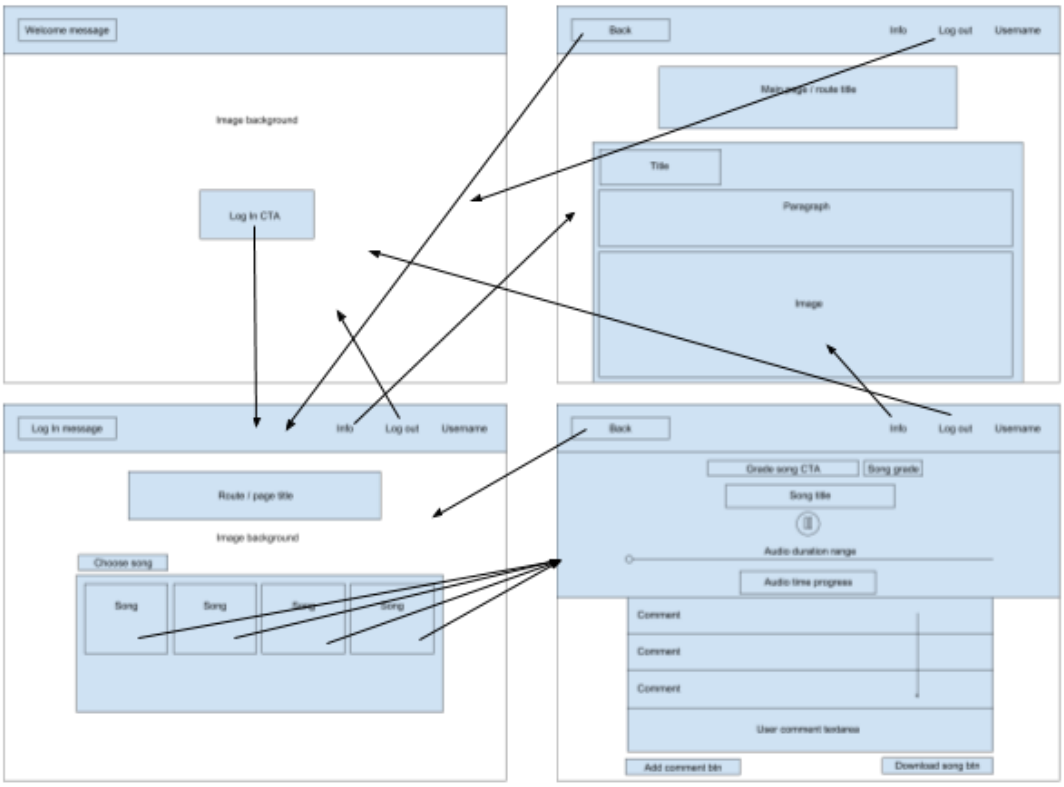
Slika 3.3 *Wireframe* stranice za biranje pjesama



Slika 3.4 *Wireframe* stranice za zasebne pjesme



Slika 3.5 *Wireframe* info stranice



Slika 3.6 Detaljna shema navigacije kroz aplikaciju

Za kvalitetniji i privlačniji UX, koriste se tranzicije i animacije raznih elemenata korisničkog sučelja koje su implementirane pomoću tehnologija GSAP[34], Vue.js tranzicija te CSS tranzicija. Animacije i tranzicije se aktiviraju korisničkom interakcijom kao što su klik ili micanje kursorom, te dolaskom na određene rute aplikacije.

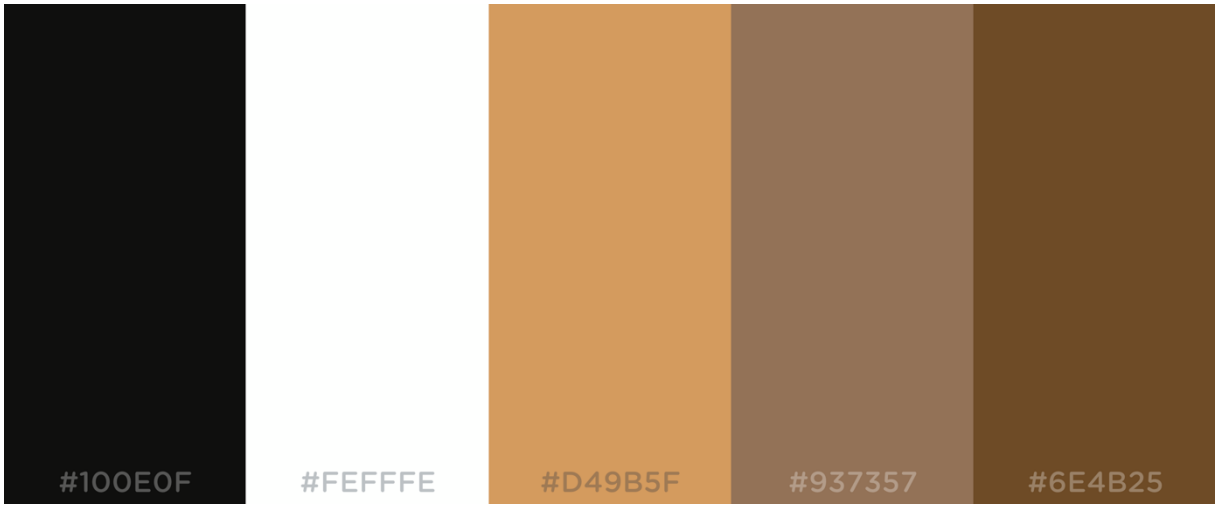
3.2.5 Površinski plan

Površinski plan (engl. *Surface plan*) definira dizajn stranice, te dizajn statičnog sadržaja unutar web rješenja.

Dizajnom stranice se korisniku pokušava ostaviti što veći dojam da se radi o glazbenom sadržaju, te interakciji sa istim. Bojama korištenim u web rješenju se nastoji dati dojam

[34] JavaScript eksterni instalacijski paket za razvoj animacija i tranzicija

nekog glazbenog instrumenta. Kako je najkorišteniji glazbeni instrument tokom snimanja glazbenog albuma električna gitara *sunburst*[35] boje, te je navedena boja također najkorištenija i najpopularnija među bojama električnih gitara, nastoji se napraviti paleta boja koja odgovara toj kombinaciji.



Slika 3.7 Paleta boja korištena za web rješenje

Pod *sunburst* paletu boja spada i crvena boja, ali kako na većini ruta imamo sliku kao pozadinu koja sadrži poprilično velik nivo žutih i crvenih nijansi, odlučeno je da se na same elemente neće dodavati crvena boja jer stvara asocijaciju na greške unutar web rješenja.

Za bržu, lakšu i kvalitetniju implementaciju dizajniranja elemenata programskog sučelja, korišteno je eksterno CSS okružje Materialize.css koje ima već složena CSS i JavaScript svojstva. Navedeno okružje se nije implementiralo na sve elemente programskog sučelja, nego najviše na one koji zahtijevaju veću količinu CSS koda za dizajniranje. To su elementi poput interakcijskih polja kao što je polje za unos teksta. Materialize.css okružje ima implementirani grid[36] sustav koji uvelike olakšava responzivnost[37] web rješenja i postavljanje različitih elemenata programskog sučelja na željena mjesta. Materialize.css

[35] Najpopularnija paleta boja za električne gitare koja se sastoji od crvenih, smeđih, crnih i bijelih nijansi
[36] Sustav raspoređivanja HTML elemenata unutar web preglednika
[37] Sposobnost prilagodbe web rješenja za prikaz elemenata unutar različitih dimenzija ekrana, uređaja i prozora web preglednika

također ima implementiranu skupinu mnogobrojnih ikona koje se koriste u web sučelju, što uvelike podiže kvalitetu UI-a i UX-a.

Dizajnom se nastoji izbjeći *scroll*[38] korisnika kako bi se što više istaknuo dojam da se radi o aplikaciji, a ne prezentacijskom web rješenju. Na svim rutama je uspješno postignut takav dizajn, osim na info ruti gdje se nalazi prevelika količina tekstualnog sadržaja da bi se izbjegao *scroll* korisnika.

3.3. Web razvoj

Web razvoj (engl. *Web development*) je proces razvijanja web rješenja koristeći web development tehnologije. Web razvoj se može vršiti raznim današnjim web development tehnologijama, te je upravo zato odlučeno korištenje najpopularnijih i najzastupljenijih web development tehnologija za izradu ovog web rješenja.

3.3.1 Početna struktura

Početna struktura definira inicijalno radno okružje u kojem se započinje raditi development web rješenja.

Najvažniji alat pri izradi svih programskih rješenja je tekstualni editor[39] ili IDE[40]. IDE koji se koristi za izradu ovog web rješenja je PHP Storm 2017.3 koji je jedan od najkvalitetnijih IDE-a za razvoj web rješenja. Navedeni IDE ima iznimno kvalitetnu podršku za klijentsku stranu programiranja, za back-end jezik PHP, te za manipulaciju bazama podataka sa SQL sintaksom.

[38] Interakcija korisnika unutar programskog sučelja na način da se prikazuje sadržaj koji nije vidljiv inicijalnim dimenzijama sučelja
[39] Programski alat za pisanje teksta, u ovom kontekstu pisanje koda
[40] IDE (engl. *Integrated Development Environment,* skraćeno IDE), radno okruženje za pisanje programskog koda sa visokim nivoom tehnološke i programske podrške

Kako imamo dvije strane aplikacije, klijentsku i serversku, potrebno je stvoriti kvalitetno radno okružje za obje strane.

Klijentska strana aplikacije je pisana u JavaScript okruženju Vue.js, koje ima vlastiti CLI[41] alat. Preduvjet uspješne instalacije Vue.js CLI-a na lokalnu mašinu je instaliran Node.js i NPM[42] na lokalnoj mašini. Sa instaliranim Vue.js CLI-om imamo mogućnost kreiranja novog projekta koji sadrži većinu potrebnih tehnologija za uspješnu realizaciju web aplikacije. Sve dodatne programske pakete i tehnologije koje su nam potrebne za uspješnu realizaciju web rješenja je moguće instalirati putem NPM-a.

```
npm i –save vue-google-auth
```

Kôd 3.10 Primjer instalacije programskog paketa putem NPM-a

Vue.js CLI ima vlastite NPM skripte za pokretanje aplikacije u razvojnom okruženju, te za stvaranje aplikacije za produkciju.

```
npm start
npm run build
```

Kôd 3.11 Primjer komanda za pokretanje Vue.js NPM skripti

Kada smo stvorili potrebno radno okruženje za rad na klijentskoj strani aplikacije, potrebno je stvoriti radno okruženje za serversku stranu. Prvi uvjet stvaranja serverskog programskog okruženja na lokalnoj mašini je Apache server, kako bi bilo moguće pokretati PHP skripte i stvoriti komunikaciju sa bazom podataka. Vrlo popularni alati za takvo okruženje su XAMPP[43], MAMP[44] i WAMP[45]. Navedeni programski alati omogućuju pokretanje radnog okruženja servera putem korisničkog sučelja ili terminala. Serversko radno okruženje koje se koristilo prilikom izrade ovog završnog rada je direktna instalacija Apache servera i PHP-a na lokalnu mašinu, iz razloga što takvo radno okruženje omogućava više mogućnosti

[41] CLI (engl. *Command – line interface,* skraćeno CLI), komandno programsko sučelje
[42] NPM (engl. *Node Package Manager,* skraćeno CLI), menadžer za instalacijske pakete na bazi Node.js-a
[43] XAMPP (engl. *Cross-Platform Apache MariaDB PHP Perl,* skraćeno XAMPP), razvojni server za PHP i Perl
[44] MAMP (engl. *Mac Apache MariaDB PHP,* skraćeno MAMP), razvojni server za PHP na Mac OS
[45] WAMP (engl. *Windows Apache MariaDB PHP,* skraćeno WAMP), razvojni server za PHP na Windows OS

tokom rada kao što je direktno manipuliranje bazama podataka i PHP skriptama putem terminala. Pri postavljanju Apache servera, ključno je definirati glavni direktorij koji će servirati PHP skripte i bazu podataka. Računalo za razvoj web rješenja je MacBook Pro Early 2015 macOS Sierra[46], koji u svom sustavu već ima instaliran Apache server, te ga je za uspješan rad potrebno samo postaviti i pokrenuti.

Korišteni alat za održavanje baza podataka ovog završnog rada je PhpMyAdmin[47], koji je najkorišteniji alat za tu namjenu u ovakvom programskom radnom okruženju.

3.3.2 Korištene razvojne web tehnologije

Za web programski projekt ovakvog tipa potrebno je mnogo instalacijskih paketa i tehnologija za kvalitetan i uspješan rad finalne web aplikacije. Svaki aspekt i dio aplikacije zahtjeva vlastite alate i tehnologije.

3.3.2.1 HTML5

HTML5 je markup[48] jezik koji služi isključivo za prezentaciju i strukturiranje elemenata na web pregledniku. Danas je aktivna peta verzija spomenutog jezika, te se ona smatra standardom među svim web preglednicima. Izdan je u listopadu 2014. godine, te je svojim dolaskom doveo mnoge novitete u web preglednike i razvoj web rješenja, osobito vezano za multimedijalne sadržaje i manipulaciju DOM[49] elemenata.

HTML5 je korišten za prezentaciju vizualnih elemenata sučelja web rješenja prema web pregledniku u kombinaciji sa Vue.js razvojnim okruženjem. Vue.js razvojno okruženje

[46] Primarni operacijski sustav Apple Mac računala
[47] Primarno grafičko sučelje baza podataka u razvoju sa PHP i SQL tehnologijama
[48] Vrsta programskog jezika za prezentaciju HTML elemenata u web preglednik
[49] DOM (engl. *Document Object Model,* skraćeno DOM), sustav elemenata namijenjenih prezentaciji kroz web preglednik

omogućava veću manipulaciju HTML elemenata i mogućnost pisanja programske sintakse zajedno sa HTML5 sintaksom. Kombinacijom tih dviju tehnologija stvaramo programsko okruženje koje donosi kvalitetniju, bržu i pregledniju manipulaciju elementima programskog sučelja unutar web preglednika.

```
<h3
        class="col s12 song-player__song-name gsap-title"
        v-if="songObject"
        ref="songTitle"
>{{ songObject.name | songName}}</h3>

<i
        class="material-icons col s2 offset-s5 song-
player__play-pause-icon"
        @click="playPauseSong"
>
    {{ songIsPlaying ? playPauseStrings.pause :
playPauseStrings.play }}
</i>
```

Kôd 3.12 Primjer kombinacija Vue.js i HTML5 tehnologija

3.3.2.2 CSS3 (SCSS)

CSS je style sheet jezik koji se koristi za stiliziranje i opisivanje prezentacije HTML5 elemenata u web pregledniku. Trenutna standardna verzija je CSS3 koja se na dnevnoj bazi unaprjeđuje i dobiva razne novitete. Sa navedenim jezikom je uz stiliziranje, moguće raditi i animacije i tranzicije elemenata programskog sučelja. CSS jezik ima i mogućnost reagiranja na korisničke interakcije unutar web preglednika.

SCSS je pred-procesorski style sheet jezik dizajniran od strane Hampton Catlina i razvijen od strane Natalie Weizenbaum. SCSS jezik zahtjeva kompajler kako bi kao finalni format dobio CSS sintaksu, koja je jedina čitljiva web pregledniku. Velike prednosti i novitete koje SCSS sadrži za razliku od klasičnog CSS-a su mogućnost pisanja varijabli, mixina, funkcija

i mnogih matematičkih opracija unutar sintakse. SCSS doslovno omogućuje programiranje vizualnih elemenata i njihovih stilova i dizajna na matematičkoj bazi. Upravo zato je danas najkorišteniji style sheet jezik u web developmentu.

Pisanje SCSS sintakse unutar web rješenja završnog rada je vršeno BEM[50] metodom. BEM metoda je metoda nazivanja CSS klasa na jedinstven način koji omogućuje veću preglednost i kvalitetniju sintaksu CSS koda. Odvija se na način da se elementi kojima se zadaju CSS klase dodaju dvije donje crte ako je element dijete blok elementa, te dvije povlake kako bi se zakačio modifikator za stiliziranje.

```
&__play-pause-icon {

    text-align: center;
    font-size: 40px;
    transition: all .3s ease-in-out;
    color: $black;

  &:hover {
        transform: scale(1.2);
        cursor: pointer;
        color: $white-almost;
        transform: rotate(360deg);
    }
}
```

Kôd 3.13 Primjer pisanja SCSS sintakse sa BEM metodom

3.3.2.3 JavaScript (ES6)

JavaScript je programski jezik namijenjen web preglednicima. Ovim programskim jezikom moguće je manipulirati DOM elementima i određenim sustavima web preglednika. Iznimno je dinamičan i ključan za razvoj interaktivnosti web rješenja sa korisnicima. Na dnevnoj bazi

[50] BEM (engl. *Block Element Modifier,* skraćeno BEM), sustav imenovanja CSS klasa i konstruiranja CSS koda

se sve više koristi u raznim programskim okruženjima koji nisu samo web development, nego i game development i back-end serverska okruženja. JavaScript programskom jeziku razne programske ekstenzije poput Node.js-a omogućuju razvoj izvan klijentske strane.

Iako je danas još uvijek standardna verzija ES5, iz dana u dan sve više web programera koristi verziju ES6 koja omogućuje bolju i kvalitetniju sintaksu JavaScripta. Noviteti poput programskih klasa, *Arrow* funkcija, *const* i *let*[51] varijabli čine JavaScript ES6 sintaksu daleko naprednijom od prethodne verzije. Razlog zašto je ES5 verzija danas i dalje standard je zato što se nisu svi web preglednici prilagodili svim novitetima unutar sintakse koje ES6 donosi. Način na koji se danas ES6 koristi je uz kompajler koji pretvara ES6 sintaksu na prethodnu verziju kako bi ista bila pregledna što širem spektru web preglednika.

JavaScript se za web rješenje završnog rada pisao u ES6 verziji, u kombinaciji za JavaScript ekstenzijom Vue.js. JavaScript ima mogućnost komuniciranja sa serverskim skriptama putem HTTP AJAX zahtjeva, što je u korišteno u mnogobrojnim situacijama u web rješenju završnog rada s obzirom na tip aplikacije. Vue.js radno okružje ima implementiran ES6 kompajler kako bi se pokrio što veći broj web preglednika koji podržavaju korištene tehnologije.

```
created() {

    this.$store.dispatch('getSongGrades', {
        songId: this.songId
    });

    this.$store.dispatch('updateSongComments', {
        songId: this.songId
    });

    EventBus.$on('updateStarsOnStart', grade => {
        if (grade === false || grade === 0) {
            return;
```

[51] JavaScript inicijalizacija varijabli ES6 sintaksom

```
            }
            else {
                this.updateStars(grade);
            }
        });

    }
```

Kôd 3.14 Primjer ES6 sintakse u kombinaciji sa Vue.js okruženjem

3.3.2.4 Vue.js

Vue.js je open-source progresivno JavaScript okruženje za razvoj korisničkih sučelja. Najviše se koristi za razvoj SPA[52] web rješenja. Dizajniran je i razvijen da bude iznimno brz i prihvatljiv unutar raznih drugih razvojnih okruženja kao što je PHP Laravel. Zadnjih godinu i pol je najviše dobio na popularnosti pored svih ostalih eksternih JavaScript programskih instalacijskih paketa kao što su React i Angular.js. Trenutno je najbrže razvojno programsko okruženje za web korisnička sučelja. Iznimno pojednostavljuje i umanjuje broj programskih linija. Najviše se fokusira na najviši sloj web aplikacija koji služi za prikaz i manipulaciju elementima korisničkog sučelja. Radi na temelju komponenti koje su u mogućnosti međusobnog komuniciranje i interakcije u opsegu cijele aplikacije.

Neke od najboljih funkcionalnosti i mogućnosti koje ovo okruženje donosi su:

1. Dvostrano vezivanje vrijednosti
2. Menadžment ruta aplikacije
3. Menadžment vrijednosti unutar opsega cijele aplikacije
4. Podjela aplikacije na međusobno povezane komponente
5. Kvalitetno i pouzdano komuniciranje sa serverskom stranom aplikacije
6. Kombiniranje programske sintakse sa HTML sintaksom

[52] SPA (engl. *Single Page Application*, skraćeno SPA), Aplikacija koja se prezentira kroz jednu ishodišnu datoteku

Vue.js okruženje je korišteno za web rješenje ovog završnog rada iz razloga što je idealan za projekt ovakvog tipa. Web aplikacija ovog tipa zahtjeva visoki nivo interakcije s korisnikom, veliku brzinu navigacije i kretanja kroz aplikaciju, te stabilnu i pouzdanu komunikaciju sa serverskom stranom aplikacije, a ovo programsko okruženje nam sve to pruža.

Kao gotovu aplikaciju spremnu za produkciju Vue.js razvija jednu HTML datoteku, jednu CSS datoteku i 3 JavaScript datoteke. Produkcijska verzija aplikacije donosi minimalnu količinu programskog koda u minimiziranom formatu što uvelike utječe na konačnu brzinu aplikacije.

3.3.2.5 PHP

PHP je serverski programski jezik kreiran od strane Rasmusa Lerdorfa 1994. godine. PHP sintaksu je moguće pisati u kombinaciji sa HTML sintaksom, ili odvojeno kao API REST okruženje. Danas je najkorišteniji serverski programski jezik u web developmentu koji uzima više od 75% web projekata na internetu. Jedan od najvećih faktora njegovoj popularnosti je što su najpoznatiji CMS[53] sustavi poput WordPressa pisani u PHP-u. PHP ima mogućnost komuniciranja sa klijentskom stranom i raznim bazama podataka putem programskih jezika kao što je SQL.

Potreba za PHP-om kod razvoja web rješenja ovog završnog rada je aplicirana iz razloga što web aplikacija ima funkcionalnosti i mogućnosti spremanja trajnih podataka kao što su audio datoteke, korisnici aplikacije, ocjene pjesama i komentari. Na klijentskoj strani nije moguće trajno spremanje takvih vrsta podataka.

Način na koji se PHP implementirao unutar aplikacije završnog rada je API REST, odnosno potpuno odvojeno od klijentske strane aplikacije, iako ga je moguće kombinirati sa

[53] CMS (engl. *Content Management System,* skraćeno CMS), programska aplikacija koja dopušta korisniku modificiranje njenog sadržaja

klijentskom stranom. Razlog takvog načina implementacije PHP-a je ta što je na klijentskoj strani korišteno Vue.js okruženje koje može razviti prilično interaktivniju i bržu web aplikaciju za razliku od PHP-a. Sve vrijednosti sa servera se šalju na klijentsku stranu u određenim trenutcima, te Vue.js radi manipulaciju tim vrijednostima i na brz i interaktivan način ih prikazuje korisnicima putem web preglednika.

```
function saveSongToDatabase($name, $songUrl, $size, $genre,
$songDuration)
{

    global $conn;

    $sql = "INSERT INTO songs (name, songUrl, songSize, genre,
duration)
            VALUES ('$name', '$songUrl', '$size', '$genre',
'$songDuration')";

    $result = $conn->query($sql);

    if ($result) {

        echo '<h5>Song <b>' . $name . '</b> has been
successfully inserted into database!<br>';

    } else {

        echo '<b>ERROR</b> - song <b>' . $name . '</b> isn\'t
inserted to database!<br>';

    }

}
```

Kôd 3.15 Primjer PHP sintakse za spremanje pjesme u bazu podataka

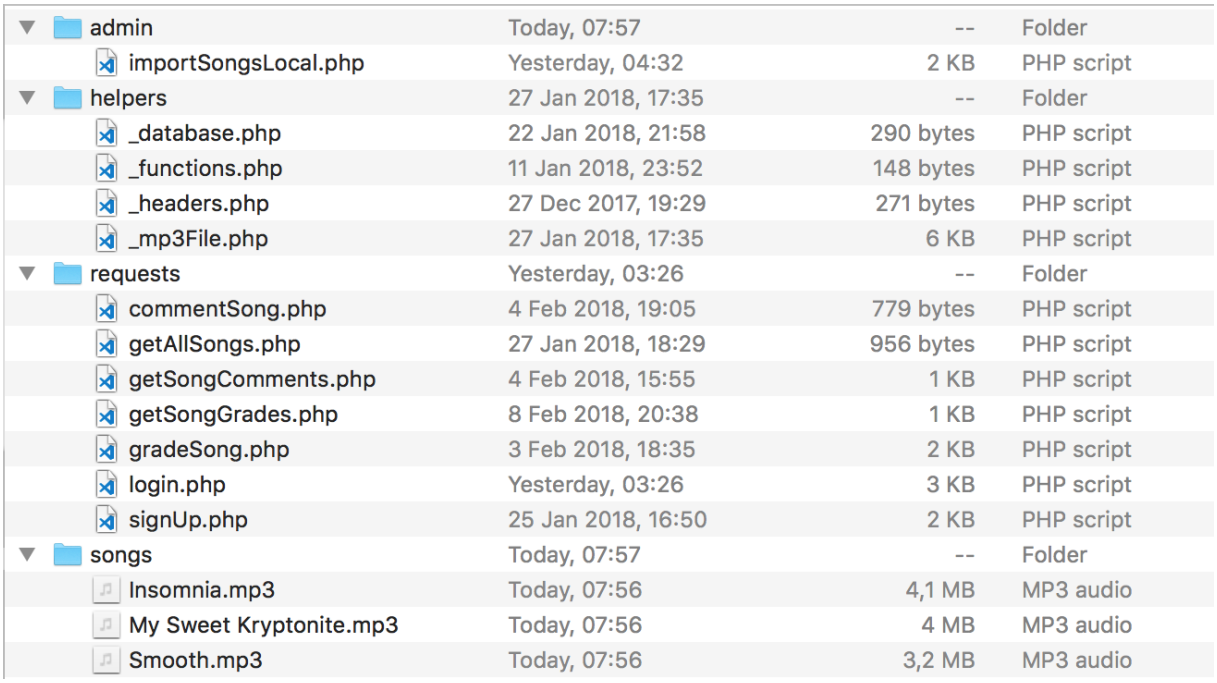
3.3.3 Logika programskog koda

Razvoj web aplikacije radi se u dva odvojena okruženja, serverska strana i klijentska strana. Najveći razlog tome su tehnologije koje se koriste na obje strane, i koje donose kvalitetnijem i bržem radu web aplikacije.

Klijentska strana aplikacije služi za prikaz informacija, sadržaja i elemenata prema korisniku putem web preglednika na interaktivan način. Vodi korisnika kroz rute aplikacije, i prati njegove reakcije unutar aplikacije. Dohvaća podatke sa serverske strane aplikacije, te ih također i šalje. Pisana je u klijentskim web development tehnologijama Vue.js, JavaScript (ES6), HTML5 i CSS3 (SCSS). Koristi razne JavaScript i CSS instalacijske pakete poput GSAP i Materilaize.css za kvalitetniji, brži i privlačniji rad aplikacije.

Serverska strana aplikacije je razvijana potpuno odvojeno od klijentske strane. Pisana je na način da se u zasebnim PHP skriptama nalazi programska logika koja sluša i prima zahtjeve sa klijentske strane, te reagira na njih na određeni način. Apache aktivni direktorij nalazi se na "127.0.0.1" putanji, odnosno na localhostu[54]. Direktorij koji je namijenjen aplikaciji završnog rada je na putanji "localhost/zavrsni/" te se kod komunikacije klijentske strane sa serverskom koristi fiksna putanja "http://localhost/zavrsni/requests" gdje se se nalaze sve PHP skripte koje slušaju zahtjeve klijentske strane aplikacije i reagiraju na njih.

[54] Tekstualno ime za lokalnu putanju računala 127.0.0.1



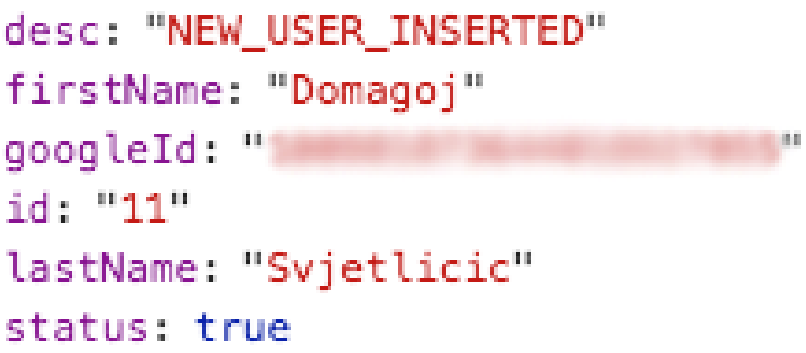
Slika 3.8 Struktura direktorija serverske strane aplikacije

Kako je vidljivo na iznad prikazanoj slici, struktura direktorija je strukturirana po namjeni.

Direktorij “admin“ sadrži samo jednu PHP skriptu koja služi za uvoz novih pjesama u sustav web aplikacije i bazu podataka, te same datoteke u određeni direktorij. Ovaj direktorij je obavezno zaštititi identifikacijom jednom kada se aplikacija stavi na aktivni povezani hosting račun, najviše iz sigurnosnih razloga kako bi samo kreator aplikacije imao dozvolu unosa novih pjesama.

Direktorij “helpers“ sadržava razne metode, klase, logiku i funkcije koje su potrebne PHP skriptama koje slušaju klijentsku stranu aplikacije. Razlog odvajanja ovih skripti u zaseban direktorij je kako bi se izbjeglo pisanje istog programskog koda na više mjesta, te veći broj programskih linija unutar jedne PHP skripte. Na primjer skripta “_database.php“ sadrži logiku i vrijednosti za spajanje na željenu bazu koja je potrebna u svakoj PHP skripti koja sluša klijentsku stranu, te bi loš primjer pisanja programskog koda bio kada bi taj identičan kod bio raspisan u svakoj od skripta koje se nalaze u direktoriju “requests“. U takvom slučaju kada bi morali promijeniti vrijednosti i logiku za spajanje na bazu bi mijenjali kod unutar više datoteka umjesto u samo jednoj.

Direktorij “requests“ sadrži sve skripte koje su povezane sa klijentskom stranom i komuniciraju sa istom. Na primjer kada korisnik napravi interakciju na klijentskoj strani koja zahtjeva unos komentara, klijentska strana će slati potrebne podatke na “http://localhost/zavrsni/requests/commentSong.php“. U trenu kada navedena PHP skripta dobije takav zahtjev klijentske strane, ona prima poslane vrijednosti, te ih određenom logikom sprema u bazu podataka. I u slučaju pozitivnog i negativnog ishoda spremanja informacija u bazu podataka, PHP skripta je obavezana odgovoriti klijentskoj strani kako bi i druga strana bila informirana o pozitivnom ili negativnom ishodu, te na određeni način reagirati na dobiveni odgovor.



Slika 3.9 Odgovor serverske strane klijentskoj u slučaju uspješne registracije korisnika

Direktorij “songs“ je javno dostupan direktorij gdje se spremaju MP3 datoteke nakon uvoza pjesama kroz sustav aplikacije. Klijentska strana pristupa ovom direktoriju u navratima kada se odabere pjesma na ruti za odabir pjesama kako bi ista mogla biti odsvirana interakcijom korisnika u programskom sučelju, te kada korisnik interakcijom pošalje zahtjev za preuzimanje određene MP3 datoteke na lokalno računalo.

Za komunikaciju PHP skripti sa bazom podataka se koristi MySql sintaksa. To je programski jezik namijenjen dizajniranju i menadžmentu informacija koje su spremljene u formi baza podataka. Za svaki dohvat, uvoz ili provjeru vrijednosti baze podataka se koristi navedeni jezik.

```
INSERT INTO songs (name, songUrl, songSize, genre, duration)
        VALUES ('$name', '$songUrl', '$size', '$genre',
'$songDuration')
```

Kôd 3.16 Primjer SQL sintakse koja sprema informacije uvezene pjesme u bazu podataka

Klijentska strana pomoću JavaScript programskog koda stvara razne HTTP AJAX zahtjeve prema serverskoj strani aplikacije na što obje strane reagiraju na određene načine. Određenom interakcijom korisnika ili njegovom trenutnom lokacijom unutar aplikacije šalju se zahtjevi prema serverskoj strani aplikacije. Na primjer kada korisnik odabere određenu pjesmu koju želi preslušati, klijentska strana aplikacije šalje serverskoj strani informacije da je korisnik zatražio zahtjev o informacijama za određenu pjesmu, te mu serverska strana odgovara sa informacijama koje su potrebne klijentskoj strani za daljnji uspješan rad aplikacije.

4. Zaključak

Današnjim tehnologijama i programskim alatima je moguće razviti gotovo svakakve tehnološke inovacije, sustave, programska rješenja i kreativne finalne produkte poput rezultata ovog završnog rada. Ovim radom je prikazan proces stvaranja glazbenog albuma i razvoj web aplikacije za prikaz, promociju, edukaciju i interakciju kroz web aplikaciju finalnog produkta glazbenog albuma.

Postoje mnogi načini i procesi stvaranja projekata tipa kao što je ovaj završni rad, te je cilj bio prikazati određene načine realizacije ovakvog tipa rada za koje se smatra da su kvalitetniji i uspješniji od ostalih načina i procesa razvoja. I kada bi se ovakav projekt krenulo raditi sa drugim tehnologijama i alatima, bez obzira na korištenje drugačijih metoda i načina, ovaj rad pruža edukaciju i znanje koje se može iskoristiti u velikim mjerama u različitim tehnologijama.

Ovim radom se prikazuje činjenica da ovakav projekt zahtjeva veliku količinu stečenog znanja i informiranosti u području glazbe, snimanja zvuka i osobito web developmenta. Dok je nešto jednostavnije, brže i lakše steći znanje za izradu glazbenog albuma, stjecanje znanja u web developmentu koje je potrebno za izradu ovakvog tipa projekta vrši se mjesecima, čak i godinama. Najviše iz razloga što web development sadrži dosta opširan opseg zahtijevanog znanja koje se dijeli u mnogo područja web developmenta. Ovaj rad zahtjeva kvalitetno znanje u području serverske i klijentske strane web developmenta, što dodatno otežava realizaciju finalnog produkta rada.

Tablica 4.1. Tehnologije i vještine relevantne za realizaciju rada

Izrada glazbenog albuma	**Izrada web aplikacije**
Poznavanje glazbe u aspektima skladanja i arhitekture glazbenih ljestvica	Razvojno okruženje za pisanje programskog koda
Programski alat za snimanje audio signala	Znanje u tehnologiji HTML5
Poznavanje cjelokupnog procesa snimanja glazbenog albuma	Znanje u tehnologiji CSS (SCSS)
Sposobnost sviranja jednog ili više glazbenih instrumenata	Znanje u tehnologiji JavaScript (ES6)
Oprema za snimanje i reprodukciju audio signala	Znanje u tehnologiji PHP
	Znanje u tehnologiji SQL
	Znanje u tehnologiji Vue.js

Postoje mnoga slična rješenja ovome radu u aspektu načina izrade glazbenog albuma, te načina izrade web aplikacije i korištenja vezanih tehnologija. Programskih sučelja za studijsko snimanje audio signala postoji mnogo na tržištu, i širok spektar njih je popularan. To su alati poput Pro Tools, Ableton Live, Bitwig Studio, Cubase, FL Studio, Reaper i mnogi drugi. Ključan razlog osobnog odabira Apple Logic Pro X-a kao programskog sučelja za studijsko snimanje audio signala ovog rada je iznimno velik odabir i manipulacija ulaznog

audio signala električnih instrumenata priključenih direktno na eksternu zvučnu karticu. Postoje i ostali programski alati koji imaju takvu mogućnost, ali Logic Pro X se pokazao najkvalitetnijim alatom za takvu uporabu, osobito zbog utjecaja navedenih efekata na ulazni audio signal u stvarnom vremenu sa minimalnom latencijom. Kada bi ideja i cilj glazbenog albuma bile različite od navedene, velika je mogućnost da bi se koristio neki drugi programski alat za snimanje audio signala. Za ovakav projekt Logic Pro X se pokazao idealnim s obzirom na svoje funkcionalnosti, način rada, brzinu snimanja audio signala, lakoću post – produkcije i jednostavnost različite manipulacije i montaže audio kanala.

Za izradu web aplikacije korištene su primarne tehnologije HTML5, CSS (SCSS), JavaScript, Vue.js, PHP i SQL. Razlog odabira navedenih tehnologija je taj što su se navedene pokazale kao najpopularnije, najkorištenije, te neke od tehnologija i jedine moguće za izradu ovog tipa projekta. Na primjer tehnologije HTML, CSS i JavaScript su neizbježne u korištenju da bi se uspješno mogla realizirati klijentska strana web aplikacije, dok su tehnologije serverske strane aplikacije zamjenjive drugim tehnologijama kao što su Java, Go, Python i mnogi drugi programski jezici. Osobni odabir PHP programskog jezika je donesen iz razloga što je PHP najkorišteniji i najpopularniji serverski programski jezik za razvoj web rješenja, te moja količina znanja u pisanju i funkcioniranju PHP sintakse naspram ostalih serverskih programskih jezika.

Navedenim načinom razvoja web aplikacije nastojalo se razviti sustav koji će se na lak i brz način moći proširiti i ažurirati prema određenim potrebama. Web rješenje nije razvijeno na način da je limitirano samo jednim glazbenim albumom i određenom količinom pjesama, nego je logikom pisanja programskog koda omogućen daljnji unos novih pjesama, glazbenih albuma i dodatnih ruta u sustav web aplikacije. Takvo web rješenje je dugoročno rješenje za promociju i prezentaciju vlastitih snimljenih pjesama i educiranje u povezanim tehnologijama.

Kako se tehnologija unaprjeđuje na dnevnoj bazi, samo je pitanje vremena kada će se tehnologije korištene u ovom radu smatrati zastarjelima. Upravo je zato ovaj rad rađen putem

modernih, zastupljenih i popularnih tehnologija i alata, kako bi ovaj rad što dulji vremenski period imao pozitivni edukativni utjecaj kojeg je poželjno slijediti.

Popis kratica

PHP	*Hypertext Preprocessor*	
HTML	*Hypertext Markup Language*	
CSS	*Cascading Style Sheets*	
SCSS	*Syntactically Awesome Style Sheets*	
USB	*Universal Serial Bus*	
TRS	*Tip, ring and sleeve*	
CMD	*Command*	Komanda
MP3	*Moving Picture Experts Group Layer-3 Audio*	
UX	*User Experience*	Korisničko iskustvo
UI	*User Interface*	Korisničko sučelje
API	*Application Programming Interface*	Programsko sučelje aplikacije
REST	*Representational state transfer*	Predstavnički prijenosni transfer
ES	*EcmaScript*	
HTTP	*Hypertext Transfer Protocol*	
JSON	*JavaScript Object Notation*	
CRUD	*Create, Read, Update, Delete*	Stvori, Čitaj, Ažuriraj, Izbriši
AJAX	*Asynchronous JavaScript And XMLHttpRequest*	
IDE	*Integrated Development Environment*	Integrirano razvojno okruženje
SQL	*Structured Query Language*	Strukturirani jezik upita
CLI	*Command – line interface*	Sučelje naredbene linije
NPM	*Node Package Manager*	
XAMPP	*Cross-Platform Apache MariaDB PHP Perl*	
MAMP	*Mac Apache MariaDB PHP*	
WAMP	*Windows Apache MariaDB PHP*	

DOM	*Document Object Model*	Model dokumenta objekta
BEM	*Block Element Modifier*	Modifikator bloknog elementa
SPA	*Single Page Application*	Jednostranična aplikacija
CMS	*Content Management System*	Sustav za upravljanje sadržajem

Popis slika

Popis tablica

Popis kodova

Literatura

[1] Vue.js Core Team, Vue.js 2 Guide: https://vuejs.org/v2/guide, rujan 2016.

[2] GreenSock Team, TweenMax documentation: https://greensock.com/docs/TweenMax, rujan 2015.

[3] Peter Cowburn, Mehdi Achour, Friedhelm Betz, PHP documentation: http://php.net/manual/en, srpanj 2004.

[4] MDN Durable Team, JavaScript Guide: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide, 2015.