

Aplikacija za pomoć učenju sviranja klavira

Bagić, Ana

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:168:388506>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-21**



Repository / Repozitorij:

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 675

APLIKACIJA ZA POMOĆ UČENJU SVIRANJA KLAVIRA

Ana Bagić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 675

APLIKACIJA ZA POMOĆ UČENJU SVIRANJA KLAVIRA

Ana Bagić

Zagreb, lipanj 2024.

DIPLOMSKI ZADATAK br. 675

Pristupnica: **Ana Bagić (0036515780)**
Studij: Računarstvo
Profil: Računarska znanost
Mentor: izv. prof. dr. sc. Marko Čupić

Zadatak: **Aplikacija za pomoć učenju sviranja klavira**

Opis zadatka:

Učenje sviranja klavira zahtjevan je zadatak jer istovremeno podrazumijeva koordinaciju niza aktivnosti kod učenika, koje uključuju koordinaciju obje ruke, njihovu vremensku usklađenost, istovremeno čitanje i razumijevanje notnog zapisa te niz drugih. Računalne aplikacije učeniku mogu pomoći u savladavanju ovih aktivnosti, omogućavajući praćenje pritisaka pojedinih tipki, analizirajući njihov korektan odnos i vremensku dinamiku, prikazivanjem alternativnih načina notnog zapisa, praćenjem uspješnosti sviranja neke dionice i nuđenjem povratne informacije o uspješnosti iskazanoj kroz različite mjere. U okviru ovog diplomskog rada potrebno je odabrati neki od navedenih ili srodnih aspekata te razviti računalni program koji će učeniku olakšati postupak učenja. U okviru rada potrebno je napraviti predloženo programsko ostvarenje te ga opisati. Radu je potrebno priložiti izvorni i izvršni kod razvijenog programa, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

1. Uvod	3
2. Opis problema	4
2.1. Postojeće aplikacije	4
2.1.1. Synthesia	4
2.1.2. Simply Piano	5
2.1.3. Flowkey	6
3. Korištene tehnologije	8
3.1. Apache Maven	8
3.2. JavaFX	8
3.3. MusicXML i ProxyMusic	9
3.4. MIDI protokol	11
3.5. SMuFL specifikacija i Bravura font	12
4. Implementacija programa	15
4.1. Početni zaslon	15
4.1.1. Navigacija aplikacijom	16
4.2. Zaslon za odabir pjesme s računala	17
4.2.1. Podržani i nepodržani formati	18
4.3. Zaslon za odabir pjesme iz knjižnice	20
4.4. Zaslon za postavke klavijature	21
4.4.1. Upravljanje MIDI uređajima	23
4.4.2. Obrada MIDI poruka	25
4.5. Glavni zaslon	26
4.5.1. Alatna traka	26

4.5.2. Notni zapis	29
4.5.3. Klavijatura	36
4.5.4. Povratna informacija o uspjehu sviranja	38
5. Analiza rezultata	39
5.1. Prikaz rezultata	39
5.2. Mogućnost nadogradnje	41
6. Zaključak	44
Literatura	45
Sažetak	47
Abstract	48

1. Uvod

Učenje sviranja klavira složen je proces koji od učenika zahtijeva koordinaciju niza kognitivnih i motoričkih vještina, uključujući sinkronizaciju pokreta obje ruke, precizno usklađivanje ritma, kontrolu dinamike, te istovremeno čitanje i razumijevanje notnog zapisa. Ovi zadaci mogu biti prilično zahtjevni, posebno za početnike, koji često moraju savladati više aspekata u isto vrijeme.

U današnjem digitalnom dobu, računalne aplikacije nude značajnu pomoć u procesu učenja sviranja raznih instrumenata, a zbog njegove popularnosti, najčešće klavira. One pružaju vizualne i zvučne povratne informacije koje pomažu učenicima u prepoznavanju i ispravljanju grešaka, praćenju napretka i usavršavanju tehnike sviranja. One također omogućuju interaktivni i personalizirani pristup učenju i mogu značajno unaprijediti učinkovitost, zadovoljstvo i zabavu u procesu učenja. Iako ove aplikacije ne mogu u potpunosti zamijenitiiskusne učitelje i tradicionalne škole, one su pristupačnije široj publici i omogućuju neformalno i fleksibilno učenje. Neke od aplikacija usmjerene su na poučavanje teorije glazbe, neke na praktične elemente, ali većina njih na neki način kombinira oba pristupa.

U ovom diplomskom radu istražila sam kako se računalne tehnologije mogu koristiti za nadopunu tradicionalnih metoda, s ciljem olakšavanja procesa učenja sviranja klavira. Razvila sam aplikaciju koja koristi praktičan pristup, specifično uči korisnika svirati odabranu skladbu. Ona na dva načina prikazuje što treba odsvirati: pomičnim notnim zapisom i bojom istaknutim tipkama klavira, te analizom odsviranog prati uspješnost korisnika. U radu sam predstavila svoje programsko rješenje, prikazala postignute rezultate, analizirala njegove prednosti i nedostatke, te navela ideje za poboljšanje i nadogradnju. Uz rad priložen je izvorni i izvršni kod razvijenog programa.

2. Opis problema

Cilj ovog diplomskog rada je razviti aplikaciju koja korisnicima omogućuje učinkovito učenje sviranja odabranih skladbi kombinirajući tradicionalne metode s modernim tehnologijama. Korisnik u aplikaciji odabire skladbu u MusicXML formatu (detaljno objašnjen u odjeljku 3.3.) koju želi savladati, dok mu aplikacija istovremeno nudi dvije različite vizualne reprezentacije onoga što treba odsvirati: pomični notni zapis i vizualno istaknute tipke na klaviru. Aplikacija kontinuirano prati korisnikovu izvedbu putem klaviature povezane s računalom MIDI protokolom (detaljno objašnjeno u odjeljku 3.4.), analizira njegov napredak i daje povratne informacije o točnosti sviranja odabrane skladbe. Ovaj rad fokusira se na razvoj aplikacije koja pojednostavljuje proces učenja sviranja, čineći ga interaktivnim i pristupačnim.

U nastavku ovog poglavlja bit će predstavljene postojeće slične aplikacije, zajedno s njihovim značajkama i mogućnostima, te usporedba s programskim rješenjem razvijenim u ovom radu, kako bi se definirale ključne razlike između njih.

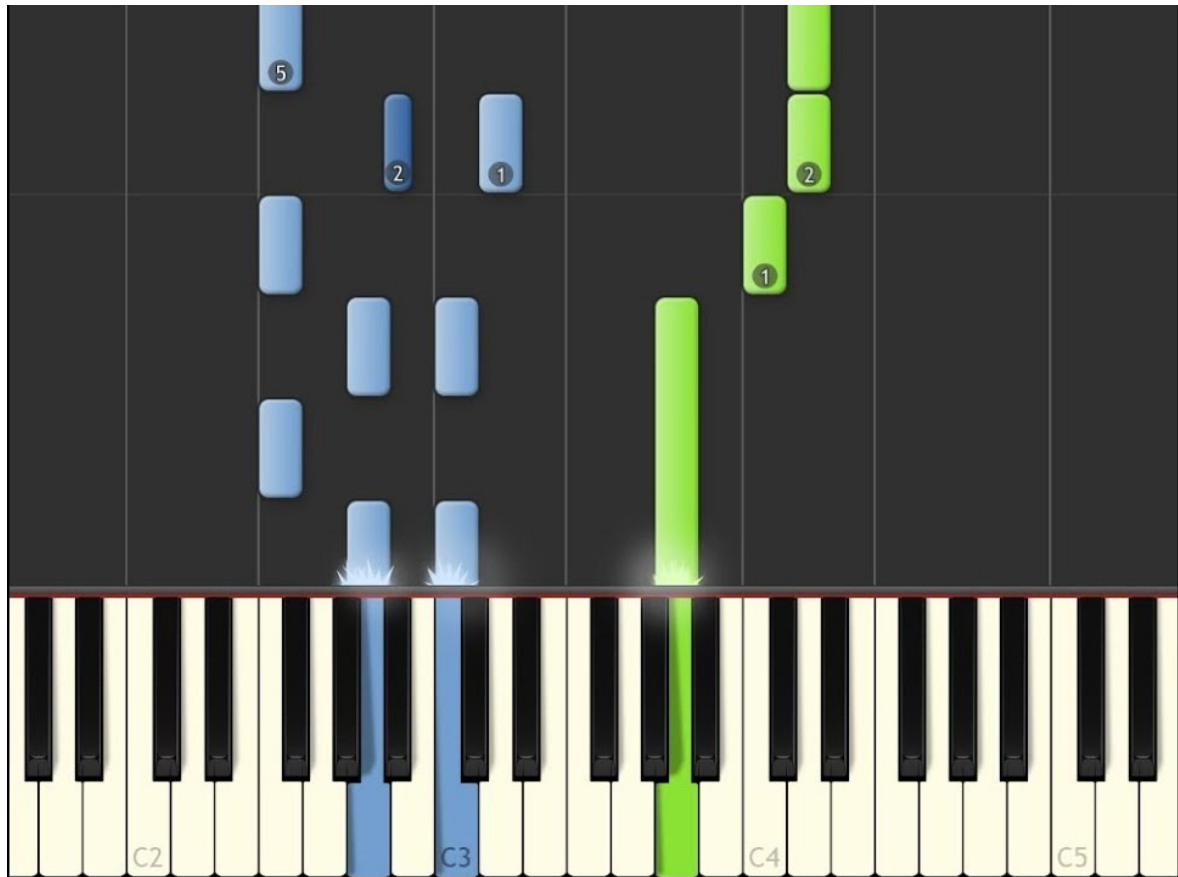
2.1. Postojeće aplikacije

Postoji veliki broj aplikacija koje se bave sličnom tematikom, međutim u nastavku navesti ću neke od najpoznatijih i najkorištenijih.

2.1.1. Synthesia

Synthesia [1] je aplikacija koja korisnicima omogućuje sviranje skladbi uz pomoć vizualnih uputa na ekranu. Osim za učenje, mnogi glazbenici koriste ju za digitalizaciju i prikaz svojih skladbi ili aranžmana tuđih, a na Youtube-u se često mogu vidjeti videozapisi izrađeni pomoću te aplikacije. Note se prikazuju u obliku padajućih blokova iznad

klavijature, pomažući korisnicima da prate koje tipke treba pritisnuti, što je prikazano na slici 2.1. Osim bogate biblioteke pjesama koju sadrži, Synthesia podržava i učitavanje MIDI datoteka, omogućujući korisnicima sviranje raznih skladbi. Aplikacija također prati korisnikov napredak, omogućuje prilagodbu tempa i vježbanje specifičnih dijelova skladbe, što je korisno za početnike i samouke glazbenike.



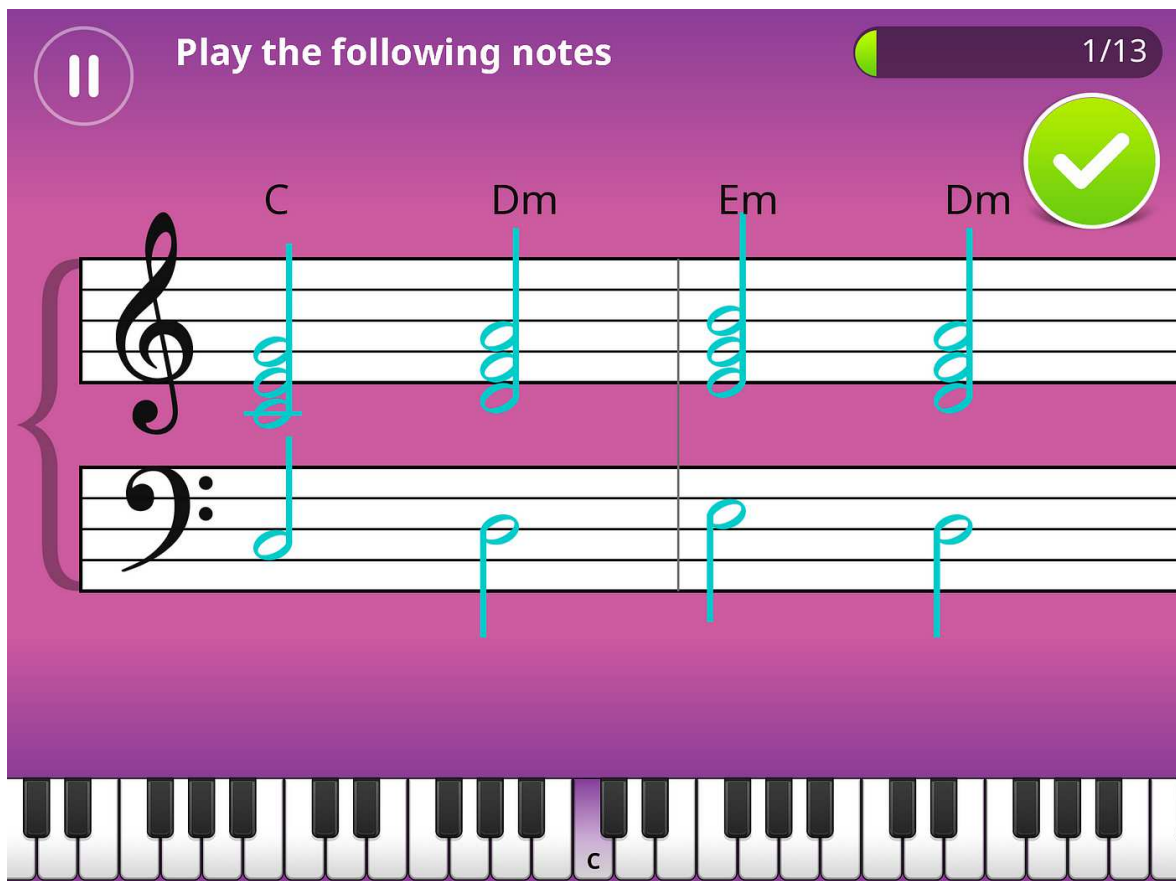
Slika 2.1. Synthesia [1]

Slično mojoj aplikaciji, Synthesia koristi bojom istaknute tipke na klaviru, ali se više oslanja na prikaz padajućih nota umjesto na tradicionalni notni zapis. Obje aplikacije nude praćenje korisnikove izvedbe na povezanoj klavijaturi i pružaju povratne informacije. Međutim, dok Synthesia nudi učitavanje isključivo MIDI datoteka, moja aplikacija koristi MusicXML format koji omogućuje precizniji prikaz notnog zapisa.

2.1.2. Simply Piano

Simply Piano [2] je popularna mobilna aplikacija za učenje sviranja klavira koja nudi interaktivne lekcije i vježbe prilagođene različitim razinama vještine, a primjer korištenja može se vidjeti na slici 2.2. Korisnici mogu učiti skladbe iz opsežnog kataloga pje-

sama, a aplikacija koristi tehnologiju prepoznavanja zvuka za analizu izvedbe i pružanje povratne informacije u stvarnom vremenu. S obzirom da ta metoda nije skroz ispravna kod određivanja odsviranih nota, od nedavno Simply Piano podržava i praćenje korisnika putem MIDI protokola kao i moja aplikacija, međutim pošto je namijenjena isključivo mobilnim uređajima, spajanje klavijature na uređaj može biti složenije. Moja aplikacija također omogućuje učitavanje skladbi u MusicXML formatu, dok Simply Piano podržava samo pjesme iz svog kataloga.

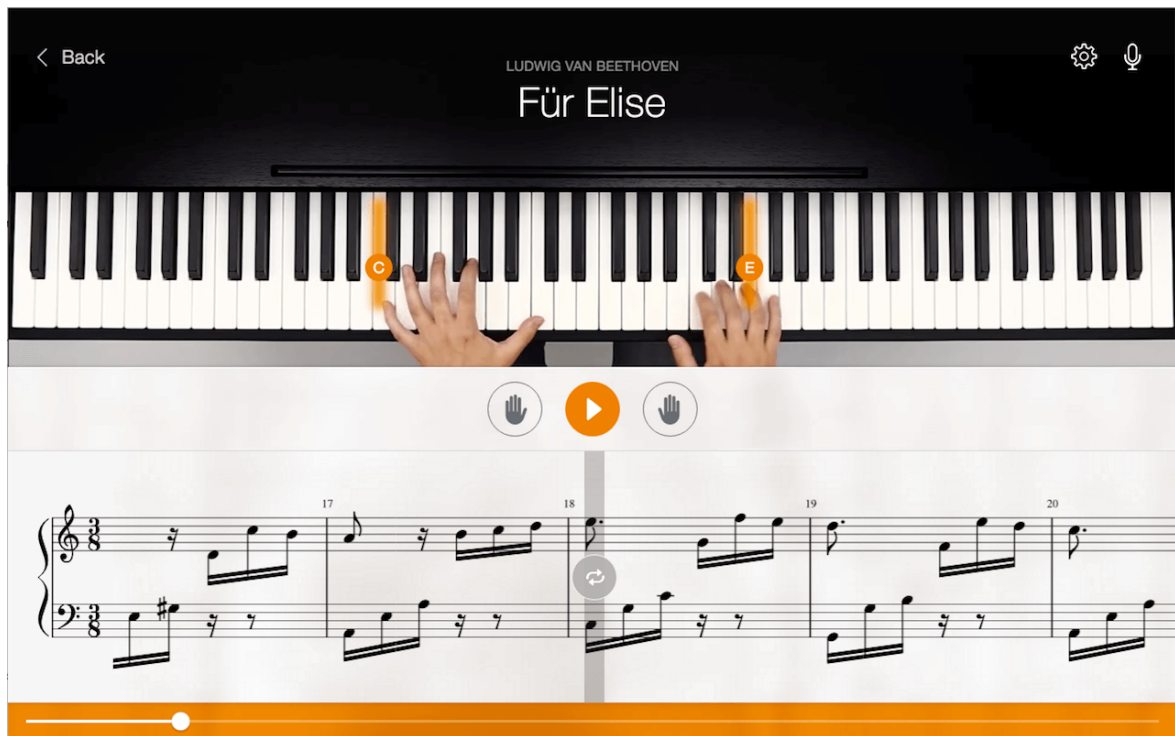


Slika 2.2. Simply Piano [3]

2.1.3. Flowkey

Flowkey [4] je aplikacija za učenje sviranja klavira koja kombinira interaktivne lekcije, i širok katalog skladbi. Primjer korištenja prikazan je na slici 2.3. Aplikacija omogućuje korisnicima učenje sviranja pjesama uz pomoć video vodiča (engl. *tutorial*) i dinamičkih prikaza notnih zapisa. Koristi tehnologiju prepoznavanja zvuka, kao i MIDI vezu s računalom kako bi analizirala korisnikovu izvedbu u stvarnom vremenu i pružala povratne informacije.

Flowkey nudi različite razine težine, od početnika do naprednih korisnika, te omogućuje odabir specifičnih dijelova skladbe za vježbanje, što moja aplikacija trenutno ne podržava. Aplikacija je dostupna na više platformi, uključujući mobilne uređaje i računala, što olakšava pristup različitim korisnicima. Za razliku od moje aplikacije, Flowkey ne podržava učitavanje vlastitih skladbi, što može biti ograničenje za korisnike koji žele raditi s notnim zapisima izvan Flowkey biblioteke.



Slika 2.3. Flowkey [5]

3. Korištene tehnologije

Aplikaciju za učenje sviranja klavira razvila sam koristeći Javu 21, trenutno (kolovoz 2024.) najnoviju verziju programskog jezika s dugoročnom podrškom (engl. *LTS*). Upravljanje projektom ostvarila sam putem Apache Maven alata, dok sam za izradu korisničkog sučelja koristila JavaFX biblioteku. Kao ulazni format datoteka upotrijebila sam MusicXML, koji sam deserijalizirala pomoću ProxyMusic biblioteke. Za prepoznavanje unosa s klavijature i reprodukciju glazbe koristila sam MIDI protokol, a prikaz notnog zapisa olakšala sam si korištenjem Bravura fonta, koji slijedi SMuFL specifikaciju.

U nastavku ovog poglavlja navesti ću osnovne informacije o navedenim tehnologijama i objasniti njihovu primjenu u ovom projektu.

3.1. Apache Maven

Maven [6] je alat za upravljanje programskim projektima i automatizaciju izgradnje. Razvijen je od strane Apache Software Foundationa, a prvi put je pušten u rad u srpnju 2004. godine. Prvenstveno je dizajniran za Java projekte, ali može se upotrijebiti i za rad u drugim programskim jezicima kao što su C#, Scala, Ruby i drugi. Jedna od ključnih značajki Mavena je njegova sposobnost za automatsko upravljanje procesom izgradnje, što uključuje prevođenje koda, izvođenje testova i pakiranje aplikacija. Međutim, možda najvažnija prednost Mavena je njegova funkcionalnost upravljanja ovisnostima (engl. *dependency*) prema vanjskim bibliotekama.

3.2. JavaFX

JavaFX [7] je biblioteka razvijena za programski jezik Java, namijenjena izgradnji korisničkih sučelja aplikacija. Razvila ju je kompanija Oracle, a prva verzija objavljena je

u prosincu 2008. godine. JavaFX se može koristiti za razvoj desktop, web, mobilnih i ugrađenih (engl. *embedded*) aplikacija.

Jedna od ključnih prednosti JavaFX-a je mogućnost odvajanja logike aplikacije od vizualnog prikaza. Vizualni elementi često se definiraju u FXML datotekama, koje koriste poseban XML format, te u kombinaciji s tim i SceneBuilder – alat koji koristi povuci-i-ispusti (engl. *drag-and-drop*) tehniku za jednostavno kreiranje i uređivanje izgleda aplikacije. Međutim, u svom radu odlučila sam komponente izravno napisati u kodu, bez korištenja FXML-a i SceneBuilder-a. Ovaj pristup mi je omogućio precizniju kontrolu nad izgledom i ponašanjem komponenata, te jednostavniju nadogradnju i jasniji pregled svega što je definirano u kodu kako bi komponenta izgledala na željeni način. JavaFX podržava stiliziranje komponenata putem CSS-a, no tu sam mogućnost koristila jedino kada neki dizajn nije bilo moguće napraviti putem koda.

3.3. MusicXML i ProxyMusic

MusicXML (Music Extensible Markup Language) [8] je format datoteke temeljen na XML specifikaciji, dizajniran za reprezentaciju zapadnjačkog notnog zapisa na način koji je razumljiv i ljudima i računalima. Ovaj format je osmislio Michael Good, a razvila ga je tvrtka Recordare LLC. Prva verzija MusicXML-a objavljena je u siječnju 2004. godine.

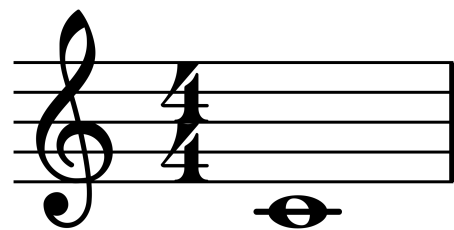
MusicXML je danas postao standardni format za digitalni notni zapis, koji koriste brojne aplikacije za kreiranje, reprodukciju i dijeljenje notnog zapisa. Mnoge od tih aplikacija omogućuju učitavanje datoteka u MusicXML formatu, uređivanje notnog zapisa, reprodukciju te izvoz u različite formate poput PDF-a, XML-a, MIDI-ja, mp3-a i drugih. S vremenom su se mogućnosti i značajke MusicXML-a značajno proširile, tako da sada podržava većinu oznaka i simbola korištenih u klasičnim notnim zapisima.

Struktura MusicXML formata slična je drugim XML formatima. Svaki element u MusicXML-u sastoji se od oznake, opcionalnih atributa u obliku `naziv="vrijednost"`, te opcionalnog sadržaja unutar elementa. Na slici 3.1. prikazan je primjer MusicXML datoteke koja definira jedan takt u četveročetvrtinskoj mjeri i C-dur ljestvici, unutar kojeg se nalazi jedna cijela nota. Zbog sažetijeg prikaza iz datoteke sam izbacila sve oznake sa metapodacima. Notni zapis te datoteke može se vidjeti na slici 3.2.

S obzirom da tekstualni prikaz MusicXML-a može biti vrlo dugačak, MusicXML v2.0 uvodi komprimirani format sa sufiksom .mxl, koji može smanjiti veličinu datoteka na otprilike jednu dvadesetinu veličine nekomprimirane verzije. Međutim, zbog pojednostavljenja implementacije moja aplikacije ne podržava taj format.

```
<score-partwise version="4.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
          <mode>major</mode>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>6</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Slika 3.1. MusicXML datoteka



Slika 3.2. Notni zapis [8]

Slika 3.3. MusicXML primjer

MusicXML format sadrži velik broj oznaka i kompleksnih struktura, što ručnu deserializaciju čini izuzetno zahtjevnom. Kako bih si to olakšala, koristila sam biblioteku koja

automatski obavlja te zadatke. ProxyMusic je biblioteka koja omogućava jednostavnu serijalizaciju, deserijalizaciju i uređivanje MusicXML datoteka, međutim ne nudi nikakve naprednije mogućnosti obrade i analize skladbe. ProxyMusic je osim za Javu, dostupna i za Python, pružajući programerima u oba jezika moćan alat za rad s MusicXML formatom. Upute za preuzimanje biblioteke, kao i dokumentacija i primjeri upotrebe dostupni su na GitHub profilu korisnika *Audiveris* na sljedećoj poveznici [9]. Biblioteka je također dostupna na Maven repozitoriju i može se pronaći na ovoj poveznici [10].

3.4. MIDI protokol

MIDI (Musical Instrument Digital Interface) [11] je standardni protokol za povezivanje i komunikaciju između elektroničkih glazbenih instrumenata, računala i drugih uređaja koji se koriste za stvaranje, reprodukciju i obradu glazbe. Razvijen je 1980-ih i prenosi digitalne informacije, poput tona, dinamike i kontrolnih uputa, a ne sam zvuk, što čini MIDI datoteke i poruke malim i učinkovitim. MIDI kabel može prenositi do 16 kanala, a svaki može kontrolirati različite instrumente, što omogućava sinkronizaciju više uređaja u glazbenim sustavima. Njegova standardiziranost osigurava kompatibilnost među različitim uređajima i aplikacijama, čineći ga neophodnim alatom u modernoj glazbenoj industriji. MIDI omogućava preciznu kontrolu i fleksibilnost u stvaranju i reproduciranju glazbe.

Java nativno podržava MIDI putem Java Sound programskog sučelja (engl. *API*), konkretno kroz *javax.sound.midi* paket, koji je jednostavan za korištenje. U svojoj aplikaciji koristila sam MIDI protokol na dva načina: za primanje podataka sa klavijature o tome što korisnik svira i za implementaciju reprodukcije glazbe. Detaljna programska implementacija ovih funkcionalnosti prikazana je u pododjeljcima 4.4.1., 4.4.2. i 4.5.3.

Klavijature se mogu spojiti s računalom na dva načina. Većina novijih klavijatura opremljena je USB tip B priključkom (engl. *port*) na stražnjoj strani, a kabel s USB-B priključkom na jednoj strani i USB-A na drugoj lako je dostupan. Međutim, stariji modeli klavijatura, uključujući onu s kojom sam radila, nemaju USB priključak, nego samo MIDI priključke. Nažalost, kabel s MIDI priključcima, puno je teže pronaći i većina trgovina s glazbenom opremom ga ne nudi. Stoga sam bila prisiljena naručiti ga iz Slovenije.

Na slici 3.4. prikazani su spomenuti priključci. S lijeve strane nalaze se MIDI priključci, dok je s desne strane USB-B priključak. Noviji modeli klavijatura sadrže oba priključka, omogućujući korisnicima da odaberu najprikladniji način spajanja. Ako se koriste MIDI priključci, važno je zapamtiti da **MIDI OUT** priključak na kabelu treba biti povezan s **MIDI IN** priključkom na klavijaturi, dok **MIDI IN** na kabelu treba biti povezan s **MIDI OUT** na klavijaturi. U suprotnom, informacije neće biti prenesene.



Slika 3.4. Priključci na klavijaturi [12]

3.5. SMuFL specifikacija i Bravura font

SMuFL (Standard Music Font Layout)[13] je specifikacija koja standardizira mapiranje tisuća glazbenih simbola korištenih u konvencionalnoj notaciji unutar Unicode standarda, točnije unutar privatnog područja (engl. *PUA - Private Use Area*). PUA je dio Unicode standarda rezerviran za prilagođene znakove i simbole koje nisu obuhvaćeni standardnim Unicode rasponom. Ovaj prostor omogućava korisnicima i aplikacijama da definiraju vlastite simbole bez univerzalnog značenja, kao što su specijalni glazbeni znakovi ili logotipi. PUA je podijeljena na tri glavna područja, međutim SMuFL koristi samo jedno od njih poznato kao *Basic Multilingual Plane*, s rasponom vrijednosti od U+E000 do U+F8FF.

Glavna svrha ove specifikacije je osigurati dosljednost i kompatibilnost između različitih fontova i aplikacija koje koriste glazbene simbole, kao što su notacijski programi i aplikacije za obradu glazbe. SMuFL definira točne pozicije simbola unutar fonta, omogućujući lako prebacivanje između fontova bez gubitka kvalitete ili ispravnosti simbola i time značajno olakšava rad svim korisnicima. Primjenjuje se u modernim notacijskim

aplikacijama kao što su Dorico¹, Finale², i MuseScore³, osiguravajući da glazbeni zapisi budu prikazani točno i konzistentno, bez obzira na korišteni font.

Specifikacija obuhvaća širok raspon simbola, uključujući note, pauze, ključeve, oznake za tempo i dinamiku, specifične oznake za pojedine instrumente, kao i specijalizirane simbole za suvremenu i povijesnu glazbu, pokrivajući praktički sve što bi ikada moglo biti potrebno u glazbenoj notaciji. Puna specifikacija, na koju sam se referirala prilikom razvoja svog programa, dostupna je na sljedećoj poveznici [14]. Simboli su organizirani po kategorijama, a uz svaki simbol priložena je slika, objašnjenje i kodna točka.

Bravura [15] je prva fontna obitelj u potpunosti usklađena sa SMuFL specifikacijom, koju je dizajnirao Daniel Spreadbury iz Steinberga za njihovu aplikaciju za glazbenu notaciju Dorico. Inspirirana je europskom glazbenom gravurom iz 19. i ranog 20. stoljeća, a ističe se debljim i snažnijim izgledom u usporedbi s ostalim glazbenim fontovima. Zbog tog dizajna Bravura nudi poboljšanu čitljivost, pogotovo kada se koristi na većim udaljenostima.

Trenutna verzija 1.392 implementira sve glazbene simbole iz SMuFL verzije 1.4, kao i one iz Unicode raspona od U+1D100 do U+1D1DD. Fontna obitelj Bravura sastoji se od dva osnovna fonta: Bravura, namijenjena za notacijske aplikacije, i Bravura Text, prilagođena za uporabu u tekstualnim aplikacijama. Ovi fontovi dostupni su u OpenType (OTF), SVG, te u Web Open Font (WOFF) formatu. U svojoj aplikaciji koristila sam Bravura Text font u OTF formatu.

Bravura fontna obitelj dostupna je pod SIL Open Font licencom, što omogućuje slobodno preuzimanje, korištenje, ugrađivanje, redistribuciju s drugim softverom te stvaranje izvedenih verzija uz određena ograničenja. Font u svakom od prethodno navedenih formata i upute za instalaciju mogu se pronaći na GitHub profilu korisnika *steinbergmedia* na poveznici [16].

¹<https://www.steinberg.net/dorico/>

²<https://www.finalemusic.com/>

³<https://musescore.org/hr>

U nastavku je prikazan primjer korištenja Bravura fonta. Ispod se nalazi Java string čiji se prikaz na ekranu može vidjeti na slici 3.5. Prikazano je redom: crtovlje, violinski tj. G ključ, oznaka koja se koristi za prikaz mjere, povisilica, dvije četvrtinke sa drškom (engl. *stem*) prema gore i dolje i četvrtinska pauza.

```
Text text = new Text("\uE014 \uE050 \uE084 \uE262 \uE1D5 \uE1D6 \uE4E5");
```



Slika 3.5. Primjer korištenja Bravura fonta

U ovom slučaju upotrijebila sam prvu kodnu točku za prikaz notnog crtovlja, kako bih jasnije prikazala pozicioniranje ostalih elemenata. Međutim, u implementaciji aplikacije crtovlje sam ručno "nacrtala", jer generiranje crtovlja korištenjem Bravura fonta nije praktično i ne preporuča se u praksi, s obzirom na to da je širine samo jednog elementa, kao što se vidi na prethodnoj slici.

Simboli su vertikalno pozicionirani tako da se svi centriraju oko treće linije crtovlja. Međutim, note imaju različite tonove i time se trebaju moći vertikalno pomicati po crtovlju. Zbog toga Bravura font nudi naprednu podršku za OpenType ligature koje omogućuju vertikalno pozicioniranje simbola. Ligature su vrsta zamjene znaka u fontu, gdje se dvije ili više kodnih točaka zamjenjuju jednim simbolom. U Bravura fontu se koriste tako što se prvo unese kodna točka koja određuje željenu promjenu vertikalnog položaja, a zatim kodnu točku za sam simbol koji se želi pozicionirati.

Nažalost, nakon puno istraživanja, zbog ograničenja JavaFX biblioteke, nisam uspjela omogućiti korištenje ligatura u svojoj aplikaciji, pa sam sva vertikalna pozicioniranja radila ručno.

4. Implementacija programa

Za razvoj ove aplikacije koristila sam arhitektonski obrazac (engl. *architectural pattern*) MVC (Model-View-Controller). On razdvaja aplikaciju na tri komponente: model, koji definira i upravlja podacima, pogled (engl. *view*), koji prikazuje podatke korisniku i kontroler (engl. *controller*), koji posreduje između modela i pogleda, definira logiku, te omogućuje interakciju korisnika s aplikacijom. Ovaj obrazac poboljšava organizaciju koda te omogućuje lakše održavanje i testiranje aplikacija.

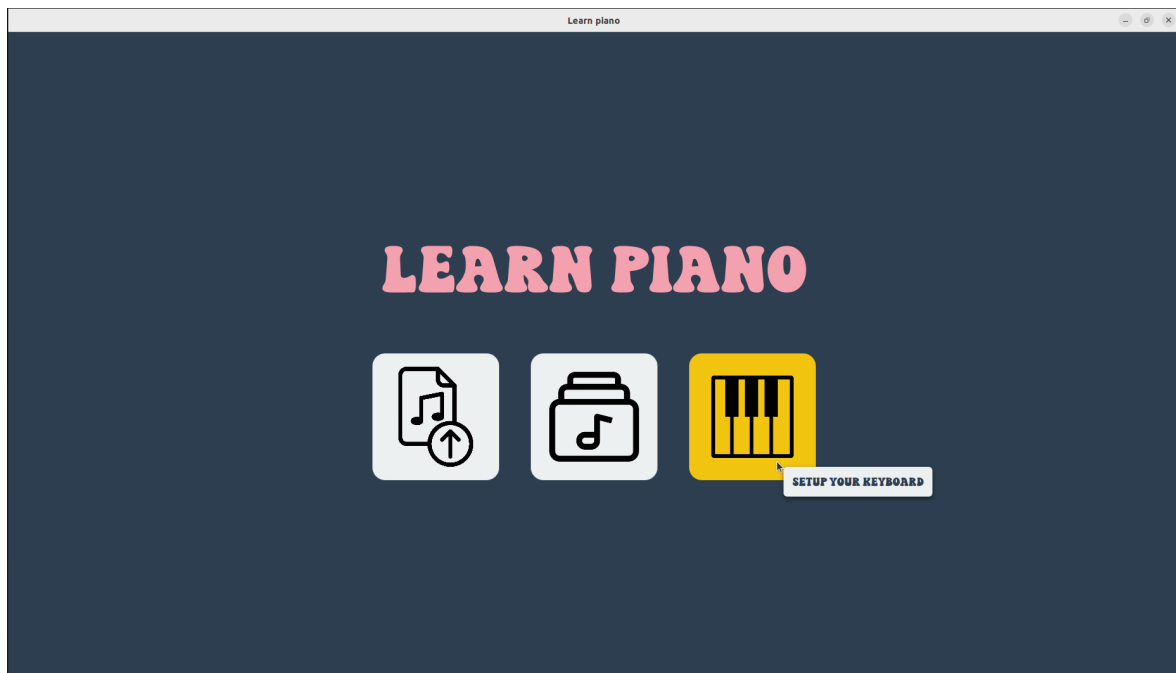
Korisničko sučelje dizajnirala sam na engleskom jeziku kako bih aplikaciju učinila pristupačnijom široj publici. Za pomoć pri odabiru palete boja koristila sam web alat *Coolors*¹. Sve ikone korištene u dizajnu aplikacije slobodne su za upotrebu u privatne i komercijalne svrhe.

U ovom poglavlju detaljno ću objasniti funkcioniranje svakog dijela aplikacije i prikazati sve zaslone. Poglavlje sam strukturirala prema zaslonima aplikacije, kako bih detaljnije objasnila sve komponente i njihove uloge. Zaslone s glavnom logikom aplikacije, objašnjen u odjeljku 4.5., razradit ću kroz pododjeljke radi bolje jasnoće.

4.1. Početni zaslon

Kada se aplikacija pokrene, prikaže se početni zaslon, koji se može vidjeti na slici 4.1. Ovaj zaslon je jednostavan i sastoji se od naslova, te tri gumba. Pritiskom na prvi gumb otvara se zaslon za odabir datoteke s pjesmom s računala, što je detaljnije objašnjeno u odjeljku 4.2. Drugi gumb otvara zaslon s izborom pjesama iz male knjižnice koju sam pripremila, a koji je detaljno objašnjen u odjeljku 4.3. Treći gumb vodi na zaslon s postavkama klavijature, o kojem je više riječi u odjeljku 4.4.

¹<https://coolors.co/>



Slika 4.1. Početni zaslon aplikacije

Prelaskom kursora preko gumba, pozadina se mijenja u žutu kako bi se jasno istaknuo mogući odabir. Umjesto teksta, odlučila sam koristiti ikone za označavanje gumba, što poboljšava vizualni dizajn, ali može smanjiti jasnoću. Zbog toga sam dodala opisne oznake (engl. *tooltip*) koje se prikazuju kada se kursor nalazi iznad gumba. Mijenjanje pozadine gumba u žutu boju i dodavanje opisnih oznaka na gume s ikonama konzistentno je primijenjeno u cijeloj aplikaciji.

4.1.1. Navigacija aplikacijom

Kako bih si olakšala navigaciju među različitim zaslonima, implementirala sam *NavigationController* koji preuzima svu logiku navigacije. Ovaj kontroler je implementiran koristeći oblikovni obrazac (engl. *design pattern*) *singleton*. Za njega i sve ostale *singleton* objekte u aplikaciji koristila sam enumeraciju (engl. *enum*) s jednom vrijednošću (engl. *case*) pod nazivom *INSTANCE*. Time sam osigurala jednostavan pristup kontroleru iz bilo kojeg dijela aplikacije gdje je potreban.

NavigationController interno pohranjuje stog (engl. *stack*) drugih kontrolera odgovornih za upravljanje zaslonima. Svi oni implementiraju sučelje *BaseViewController*, koje zahtijeva implementaciju metode *Pane getView()*, koja vraća pogled kojim taj kontroler upravlja. *NavigationController* nudi tri jednostavne metode za upravljanje navigaci-

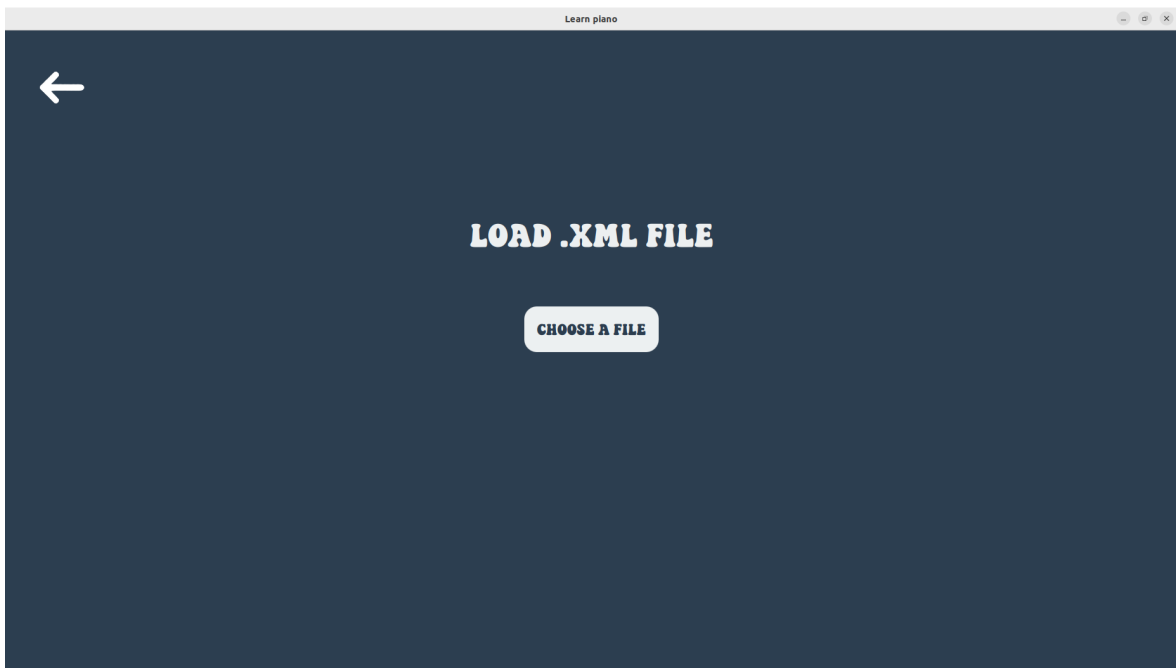
jom, navedene u nastavku, kao i metodu *Stage* `getStage()` koja omogućuje dohvat objekta *Stage* iz bilo kojeg dijela aplikacije.

```
void init(Stage stage, BaseViewController viewController);  
void push(BaseViewController viewController);  
void pop();
```

Prva metoda koristi se za inicijalizaciju navigacije početnim *stage-om* i kontrolerom, u mojem slučaju *TitleViewController-om* koji upravlja početnim zaslonom. Metoda *push* dodaje predani kontroler na interni stog i prikazuje njegov pogled. Slično tome, metoda *pop* uklanja posljednji kontroler sa stoga i prikazuje pogled kontrolera koji se nalazi neposredno ispod njega, vraćajući aplikaciju na prethodni zaslon.

4.2. Zaslon za odabir pjesme s računala

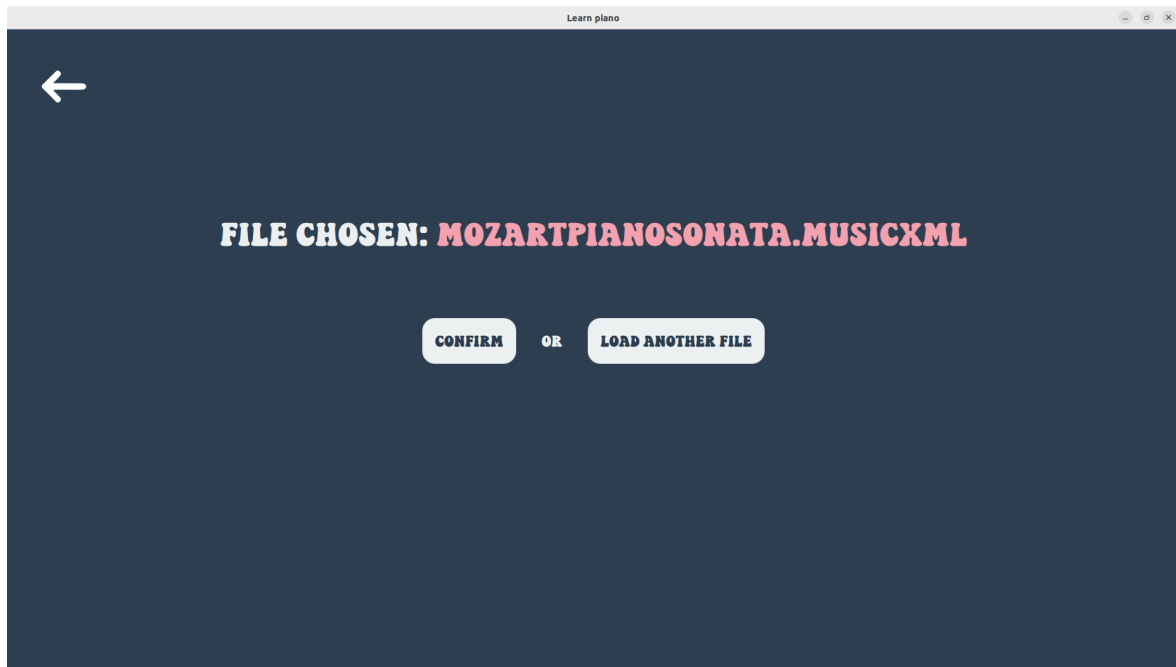
Odabirom lijevog gumba na početnom zaslonu otvara se zaslon za odabir pjesme s računala, prikazan na slici 4.2. Klikom na gumb *Choose a file* otvara se datotečni izbornik u kojemu se mogu odabrati datoteke s nastavcima *.xml* i *.musicxml*. U gornjem lijevom kutu zaslona smješten je gumb sa ikonom strelice, koji omogućava povratak na prethodni zaslon. Ovaj gumb prisutan je na svakom zaslonu osim na početnom.



Slika 4.2. Zaslon za odabir pjesme s računala

U slučaju da odabrana datoteka odgovara formatu koji moja aplikacija podržava, pri-

kazuje se zaslon kao na slici 4.3. Na zaslonu se prikazuje naziv odabrane datoteke te su dostupna dva gumba. Klikom na gumb *Confirm*, potvrđuje se odabir datoteke i otvara se glavni zaslon aplikacije. Gumb *Load another file* omogućuje ponovno otvaranje datotečnog izbornika za odabir druge datoteke.

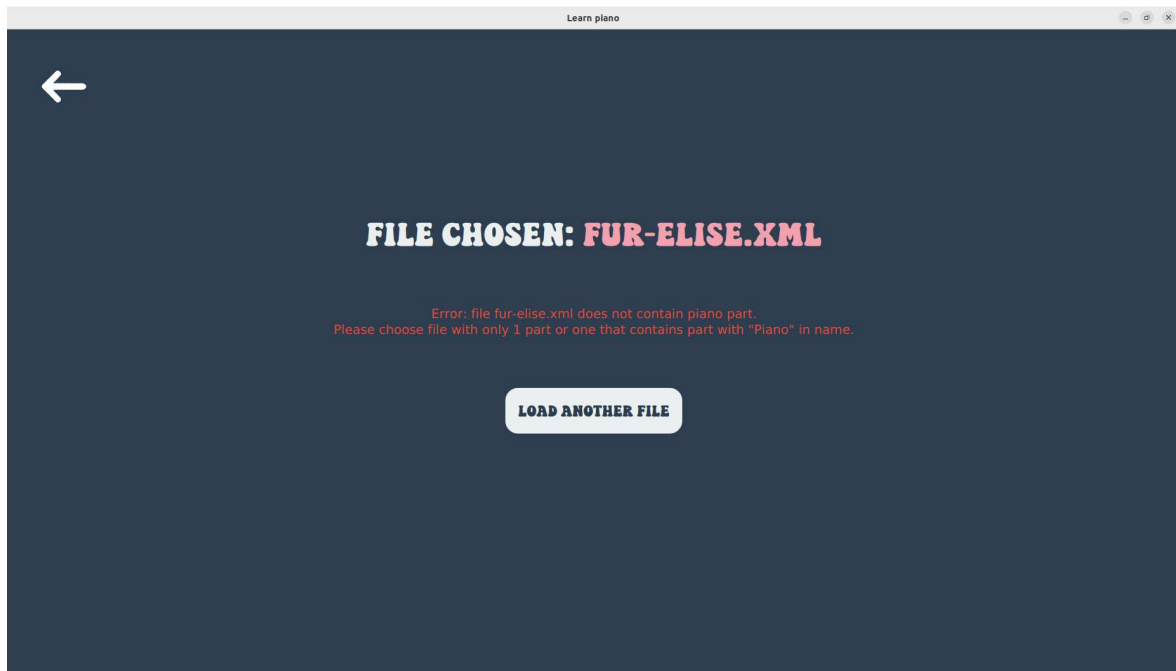


Slika 4.3. Zaslon za potvrdu pjesme

4.2.1. Podržani i nepodržani formati

Ako format odabrane datoteke nije podržan u aplikaciji, na zaslonu se prikazuje pogreška, što je vidljivo na slici 4.4. Mogu se pojaviti tri različite vrste pogrešaka. Prva se javlja ako datoteka nije uopće u MusicXML formatu ili ako njezin korijenski element nije `<score-partwise>`. Iako datotečni sustav filtrira datoteke u formatima `.musicxml` i `.xml`, format `.xml` može sadržavati razne vrste datoteka, pa postoji mogućnost da korisnik odabere datoteku koja nije u MusicXML formatu.

MusicXML datoteke mogu biti organizirane na dva različita načina, ovisno o korijenskom elementu. Element `<score-partwise>` strukturira notni zapis prema dijelovima (engl. *part*), poput različitih instrumenata ili vokalnih linija. Unutar svakog dijela, zapisi su organizirani po taktovima. Ovaj format je najčešći i najintuitivniji za većinu korisnika, jer slijedi tradicionalnu organizaciju partitura. S druge strane, `<score-timewise>` strukturira notni zapis prema taktovima. Svaki takt sadrži informaciju za sve dijelove u tom



Slika 4.4. Zaslona za odabir pjesme sa pogreškom

taktu. Ovaj format je koristan za obradu glazbenih podataka gdje je važno imati pristup informacijama prema vremenskoj liniji.

U nastavku je prikazana metoda koja deserijalizira odabranu datoteku u *ScorePartwise* objekt koristeći biblioteku *ProxyMusic*. Ona se nalazi u razredu *XMLConverter* koji je implementiran kao *singleton* objekt kroz enumeraciju, kako bi deserijalizirani *ScorePartwise* objekt mogao biti dostupan u bilo kojem dijelu aplikacije. Ako dođe do iznimke (npr. datoteka nije u ispravnom formatu, deserijalizacija ne uspije ili dođe do problema s ulazno-izlaznim operacijama), metoda vraća *false* i aplikacija obavještava korisnika o pogrešci.

```
public boolean unmarshall(File file) {
    try (InputStream inputStream = new FileInputStream(file)) {
        Object unmarshalledObject = Objects.requireNonNull(Marshalling.
            unmarshal(inputStream));
        score = (ScorePartwise) unmarshalledObject;
        return true;
    } catch (Marshalling.UnmarshallingException | ClassCastException |
        NullPointerException | IOException e) {
        return false;
    }
}
```


Druga kontrola koju aplikacija provodi odnosi se na prisutnost klavirskog dijela u datoteci. Budući da su datoteke formata *<score-partwise>* organizirane po dijelovima, na početku se nalazi obavezan element *<part-list>* koji sadrži popis svih dijelova skladbe.

Ako datoteka sadrži samo jedan dio, aplikacija automatski pretpostavlja da je taj dio namijenjen klaviru, budući da nema drugih kriterija za identifikaciju. Međutim, ako datoteka sadrži više dijelova (npr. dodatni instrument ili vokalnu liniju), aplikacija pretražuje imena dijelova kako bi provjerila sadrži li neki od njih riječ "Piano" te ako ga pronađe, taj dio smatra klavirskim. U slučaju da aplikacija ne uspije pronaći dio za klavir, korisniku će prikazati poruku o pogrešci, kao što je prikazano na slici 4.4.

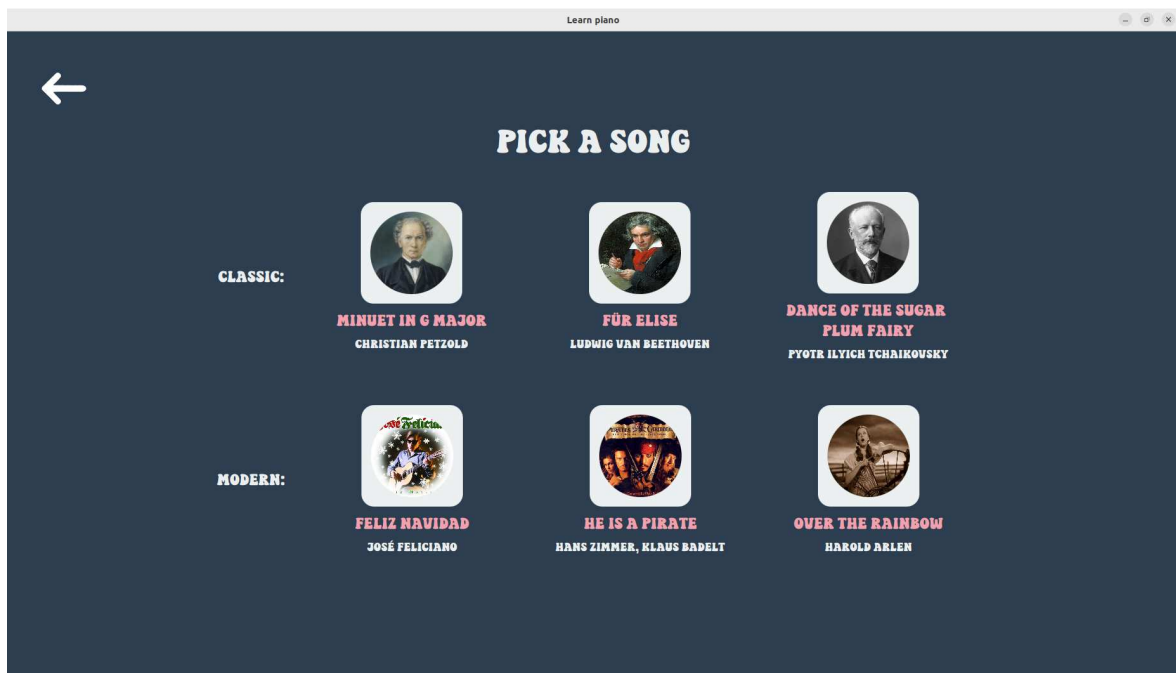
Treća i posljednja provjera odnosi se na broj crtovlja (engl. *staff*) u dijelu. Prvi takt svakog dijela sadrži element *<attributes>*, koji pruža ključne informacije o pjesmi, poput ljestvice, vremenske oznake, ključeva, te elementa *<staves>*. Ovaj element sadrži broj koji označava koliko crtovlja pjesma ima. U slučaju da taj element nije prisutan, pretpostavlja se da pjesma ima jedno crtovlje.

Aplikacija podržava prikaz jednog ili dva crtovlja. Ako pjesma ima jedno crtovlje, ono je namijenjeno desnoj ruci. Ako ima dva crtovlja, prvo je za desnu ruku, a drugo za lijevu. U slučaju da *<attributes>* element nije pronađen ili broj crtovlja prelazi dva, aplikacija prikazuje odgovarajuću poruku o pogrešci na ekranu.

Uz pogrešku, na ekranu je također prikazan naziv odabrane datoteke te gumb *Load another file* kojim se putem datotečnog izbornika može odabrati druga datoteka.

4.3. Zaslون za odabir pjesme iz knjižnice

Odabirom srednjeg gumba na početnom zaslonu otvara se zaslon za odabir pjesme iz knjižnice, koji je prikazan na slici 4.5. Iako je prvotna ideja aplikacije bila da korisnik učita datoteku sa željenom pjesmom, odlučila sam implementirati i malu knjižnicu s nekoliko unaprijed ponuđenih pjesama. Idealno ova knjižnica sadržavala bi puno veći broj pjesama, ali to bi zahtijevalo značajan trud u pronalaženju i pripremi velikog broja datoteka, što nadilazi opseg ovog rada. Ipak, knjižnica služi kao dokaz koncepta i olakšava testiranje funkcionalnosti aplikacije.



Slika 4.5. Zaslona za odabir pjesme iz knjižnice

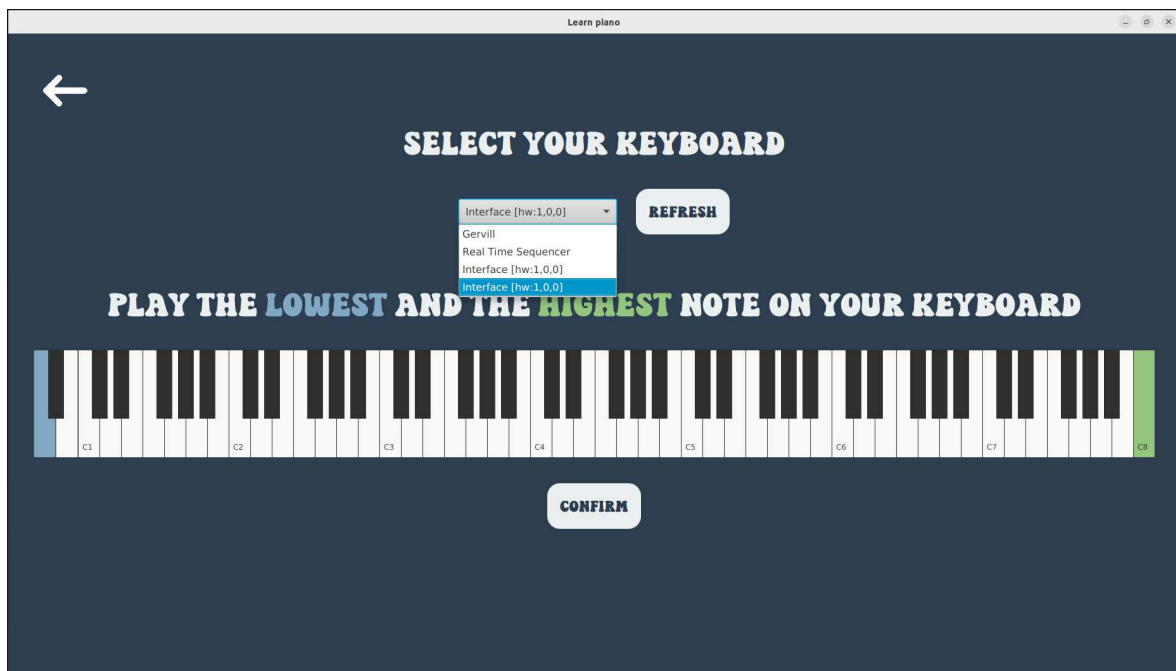
Odlučila sam u knjižnicu dodati tri klasične skladbe i tri moderne pjesme. Pjesme su različitih težina gdje je prva u oba žanra najjednostavnija za sviranje, a zadnja najteža. Za svaku pjesmu navedeni su naziv, ime umjetnika (skladatelja/izvođača/pjevača), te fotografija umjetnika, koja služi kao gumb za odabir te pjesme. Odabirom pjesme klikom na fotografiju, odgovarajuća datoteka se učitava iz resursa aplikacije i deserijalizira na način opisan u odjeljku 4.2.1., te se otvara glavni zaslon aplikacije. Za ove datoteke nije potrebno provoditi dodatne provjere, budući da su već unaprijed provjerene i sigurno ispravne.

Pjesme odnosno skladbe koje sam odabrala su redom: "Menuet u G duru" skladatelja Christiana Petzolda, "Za Elizu" skladatelja Ludwiga van Beethovena, "Ples šećerne vile" skladatelja Petra Iljiča Čajkovskog, "Feliz Navidad" pjevača i tekstopisca Joséa Feliciano, "He is a pirate" skladatelja Klause Badelta i Hansa Zimmera za film "Pirati s Kariba: Prokletstvo Crnog bisera" i zadnja "Over the rainbow" skladatelja Harolda Arena za film "Čarobnjak iz Oza". Sve skladbe preuzete su sa stranice MuseScore[17].

4.4. Zaslona za postavke klavijature

Odabirom desnog gumba na početnom zaslonu otvara se zaslon za odabir klavijature i postavljanje njenog raspona, koji je prikazan na slici 4.6. Na ovome zaslonu korisnik

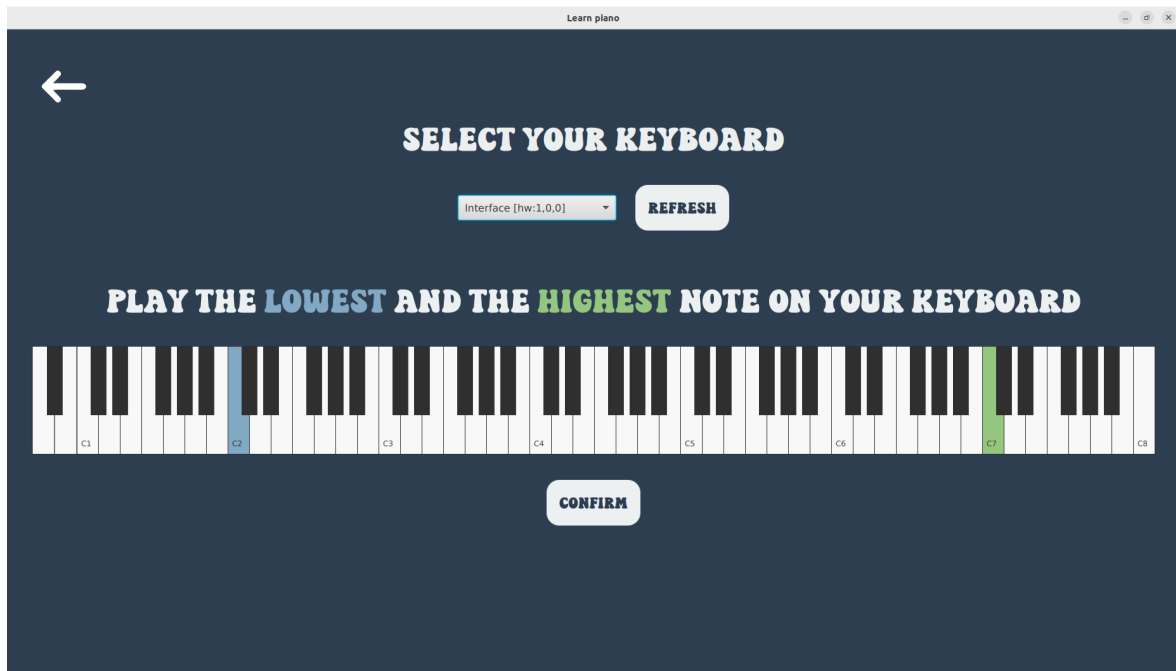
može odabrati spojenu klavijaturu iz padajućeg izbornika koji prikazuje sve dostupne MIDI uređaje. Java nativno podržava rad s MIDI uređajima i protokolom putem paketa *javax.sound.midi*. Desno od padajućeg izbornika nalazi se gumb za osvježavanje liste dostupnih uređaja. Zadani odabir je posljednji uređaj na listi, budući da sam na temelju testiranja ustanovila da je to obično uređaj koji je fizički spojen na računalo. Prije njega nalaze se MIDI uređaji koji su aplikacijski ili virtualni, odnosno već ugrađeni u operacijski sustav. Promjenom odabira uređaja u padajućem izborniku, mijenja se uređaj u aplikaciji i taj proces je detaljnije objašnjen u pododjeljku 4.4.1.



Slika 4.6. Zaslona za postavke klavijature

Ispod izbornika za odabir MIDI uređaja nalazi se uputa za korisnika da odsvira najnižu i najvišu notu na svojoj klavijaturi. To je važno jer nisu sve klavijature punog opsega koncertnog klavira (A0 - C8), što znači da neke vrlo niske ili visoke tonove na njima nije moguće odsvirati. Ovo može predstavljati problem kod načina sviranja koji čeka da korisnik odsvira točnu notu, jer ju on možda neće moći odsvirati.

Na slici 4.7. prikazan je rezultat odabira raspona, pri čemu je plavom bojom označena najniža nota, a zelenom najviša. Kada korisnik pritisne tipku na klavijaturi i ako je ona različita od prethodno pritisnute, ažurira se raspon nota tako da nova i prethodna tipka definiraju najnižu i najvišu notu. Zaslona se zatim osvježava kako bi prikazao te promjene, dinamički prilagođavajući plavu i zelenu oznaku prema korisnikovom odabiru.



Slika 4.7. Zaslom za postavke klavijature s odabranim rasponom

Klikom na gumb *Confirm* sprema se korisnikov odabir raspona klavijature, nakon čega se aplikacija vraća na prethodni, odnosno početni zaslon. Također, korisnik se može vratiti na prethodni zaslon pritiskom na strelicu za povratak u gornjem lijevom kutu, ali u tom slučaju korisnikov odabir raspona klavijature neće biti spremljen. Postavljanje ovih postavki nije obavezno, jer će aplikacija u većini slučajeva ispravno funkcionirati i sa zadanim postavkama: posljednji uređaj na listi biti će automatski odabran, a raspon klavijature postavljen na standardni koncertni klavir.

4.4.1. Upravljanje MIDI uređajima

Za upravljanje MIDI uređajima unutar aplikacije napisala sam razred *MidiDeviceManager*, koji je također implementiran kao *singleton* objekt, osiguravajući da postoji samo jedna instanca razreda tijekom rada aplikacije. Ovaj razred omogućuje centralizirano upravljanje MIDI uređajima, uključujući povezivanje, postavljanje i zatvaranje uređaja kroz sljedeće metode:

```
List<MidiDevice.Info> getDeviceInfos();
MidiDevice.Info getChosenDeviceInfo();
void setDevice(MidiDevice.Info deviceInfo);
MidiInputReceiver getReceiver();
void close();
```

Metoda *getDeviceInfos* vraća listu informacija svih dostupnih MIDI uređaja koje sustav prepozna, a to se može napraviti ovako: `MidiSystem.getMidiDeviceInfo()`. Metoda *getChosenDeviceInfo* vraća informacije o trenutno odabranom uređaju. Ako nijedan uređaj nije odabran, ili se odabrani uređaj više ne nalazi u listi dostupnih uređaja, automatski se odabire posljednji uređaj s liste dostupnih, što odgovara zadanom ponašanju prethodno opisanim u tekstu.

Postavljanje korisnikovog odabira postiže se koristeći metodu *setDevice*, koja postavlja i otvara odabrani MIDI uređaj na temelju predanih informacija. U slučaju da već postoji otvoreni uređaj, on se zatvara prije otvaranja novog. Otvaranje uređaja provodi se na način naveden u nastavku. Nakon što se dohvati i otvori željeni uređaj, postavlja se prijemnik (engl. *receiver*), za predajnik (engl. *transmitter*) uređaja.

```
try {
    assert chosenDeviceInfo != null;
    currentDevice = MidiSystem.getMidiDevice(chosenDeviceInfo);
    currentDevice.open();

    Transmitter transmitter = currentDevice.getTransmitter();
    transmitter.setReceiver(receiver);
} catch (MidiUnavailableException | NullPointerException ignored) {
}
```

Metoda *getReceiver* dohvaća prijemnik, a ako on nije već postavljen, inicijalizira MIDI uređaj na zadane postavke. Prijemnik se implementira kroz sučelje *Receiver*, što sam to napravila u razredu *MidiInputReceiver*. Detaljnija implementacija ovog sučelja može se pronaći u pododjeljku 4.4.2.

Na kraju, metoda *close* omogućuje zatvaranje trenutno otvorenog MIDI uređaja, ako je otvoren, čime se osigurava pravilno upravljanje resursima kod gašenja aplikacije.

Osim metoda za rad s MIDI uređajima, ovaj razred pruža i pristup modelu klavijature putem metode *getKeyboardModel*. Ako model klavijature još nije definiran, automatski se kreira novi model s rasponom koncertnog klavira. Metoda *setKeyboardModel* omogućuje postavljanje novog modela klavijature.

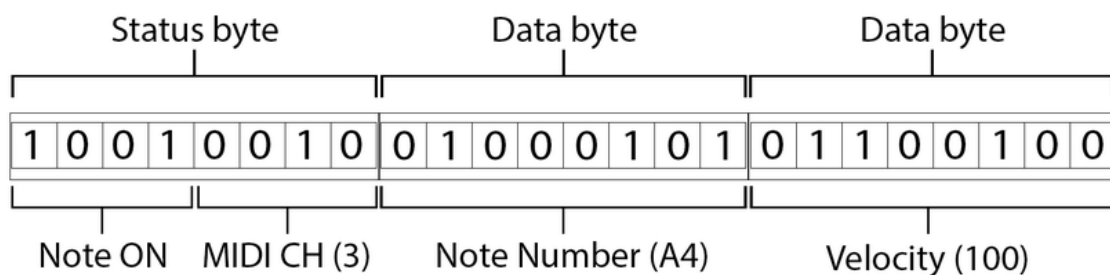
4.4.2. Obrada MIDI poruka

Razred *MidiInputReceiver* predstavlja implementaciju sučelja *Receiver*, zaduženog za primanje i obradu MIDI poruka, specifično onih koje se odnose na pritisak i otpuštanje klavirskih tipki. Ključna metoda koju je potrebno implementirati u ovom razredu je *send*. Ona prima i obrađuje primljene MIDI poruke, a njena implementacija prikazana je u nastavku.

```
@Override
public void send(MidiMessage message, long timeStamp) {
    int status = message.getStatus();
    if (status == 128 || status == 144) {
        byte[] data = message.getMessage();
        int key = data[1] & 0xFF;
        int speed = data[2] & 0xFF;

        if (status == 144 && speed != 0 && keyPressedListener != null) {
            keyPressedListener.onAction(key);
        } else if (keyReleasedListener != null) {
            keyReleasedListener.onAction(key);
        }
    }
}
```

Primjer sadržaja MIDI poruke prikazan je na slici 4.8. Poruka se obično sastoji od tri bajta: prvi je statusni bajt, koji određuje tip poruke i kanal, dok druga dva bajta sadrže parametre, kao što su vrijednost pritisnute tipke i brzina pritiska. Statusni bajt uvijek započinje s bitom 1, dok podatkovni bajtovi s bitom 0.



Slika 4.8. Primjer MIDI poruke [18]

Metoda *send* prvo provjerava status poruke. Iako postoji više mogućih statusa, za moj

rad su ključni status 128, koji označava otpuštanje tipke, i status 144 koji označava pritisak tipke. Zbog pojednostavljenja implementacije, neki modeli klavijatura (uključujući onaj koji koristim) ne koriste status 128 za označavanje otpuštanja tipke. Umjesto toga, koriste kombinaciju statusa 144 s brzinom pritiska postavljenom na 0. Stoga, u mojoj aplikaciji oba ova pristupa prepoznajem kao otpuštanje tipke.

Unutar metode, poruka se raščlanjuje pomoću metode *getMessage*, koja vraća byte polje podataka. Prvi element u tom polju sadrži prethodno dobiven status, dok drugi element *data[1]* sadrži vrijednost pritisnute tipke. Ta vrijednost je u rasponu od 0 do 127, pri čemu klavijature obično koriste interval od 21 do 108: vrijednost 21 označava tipku A0, a vrijednost 108 tipku C8. Srednja tipka klavijature C4 ima MIDI vrijednost 60. Treći element polja *data[2]* sadrži informaciju o brzini pritiska tipke, od 0 do 100, no u ovom je slučaju ta informacija relevantna samo za detekciju otpuštanja tipke.

Na temelju tih podataka, metoda pokreće odgovarajuće akcije, ovisno o tome je li tipka pritisnuta ili otpuštena. Ovo omogućuje da se reakcije na pritisak ili otpuštanje tipki prenesu na ostatak sustava koristeći slušače (engl. *listener*) koje vanjske komponente mogu postaviti.

4.5. Glavni zaslون

Odabirom odgovarajuće datoteke s računala ili iz knjižnice aplikacije otvara se glavni zaslon aplikacije prikazan na slici 4.9. Za potrebe prikaza funkcionalnosti koristila sam početak skladbe "An Chloe" skladatelja Wolfganga Amadeusa Mozarta.

Zaslon je podijeljen u tri glavna dijela: u gornjem dijelu nalazi se alatna traka za upravljanje reprodukcijom pjesme, u srednjem dijelu nalazi se naziv pjesme i notni zapis, dok je u donjem dijelu prikaz klavijature. Svaki od tih dijelova bit će detaljnije objašnjen u sljedećim pododjeljcima.

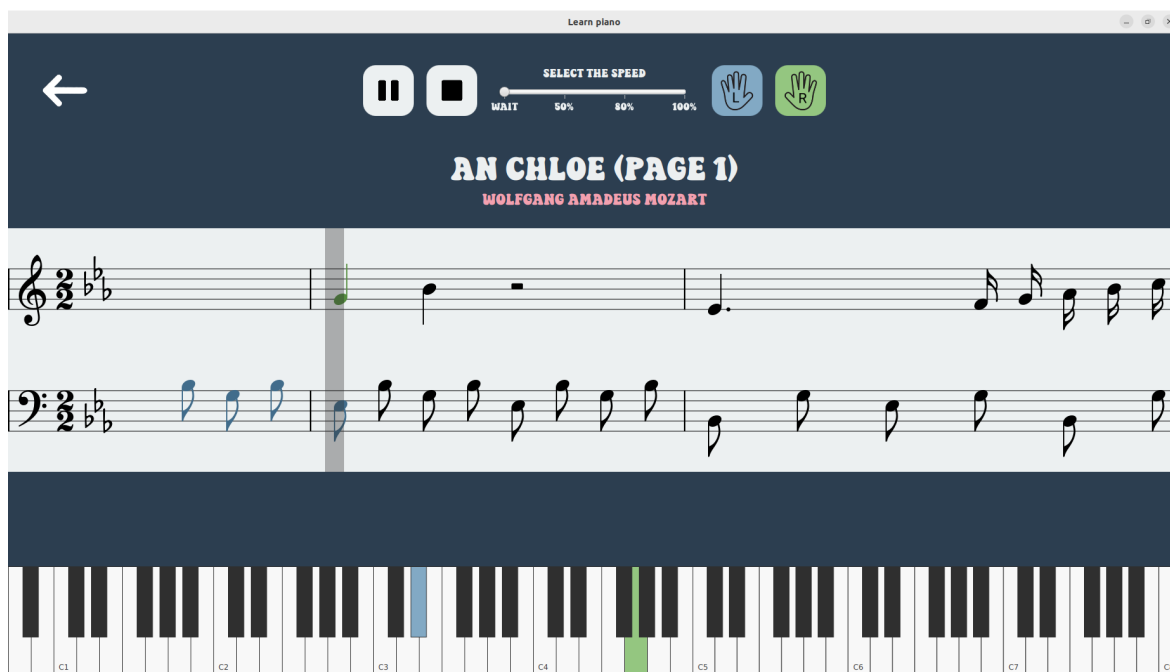
4.5.1. Alatna traka

Na vrhu zaslona nalazi se alatna traka koja upravlja reprodukcijom pjesme. Kao i na ostalim zaslonima, osim početnog, u gornjem lijevom kutu nalazi se gumb za povratak na prethodnu stranicu.



Slika 4.9. Glavni zaslon aplikacije

Prvi gumb u alatnoj traci služi za pokretanje i pauziranje reprodukcije. Kada pjesma nije u reprodukciji, gumb prikazuje ikonu za početak reprodukcije, što je vidljivo na slici 4.9. Tijekom reprodukcije, gumb prikazuje ikonu za pauziranje, vidljivu na slici 4.10.



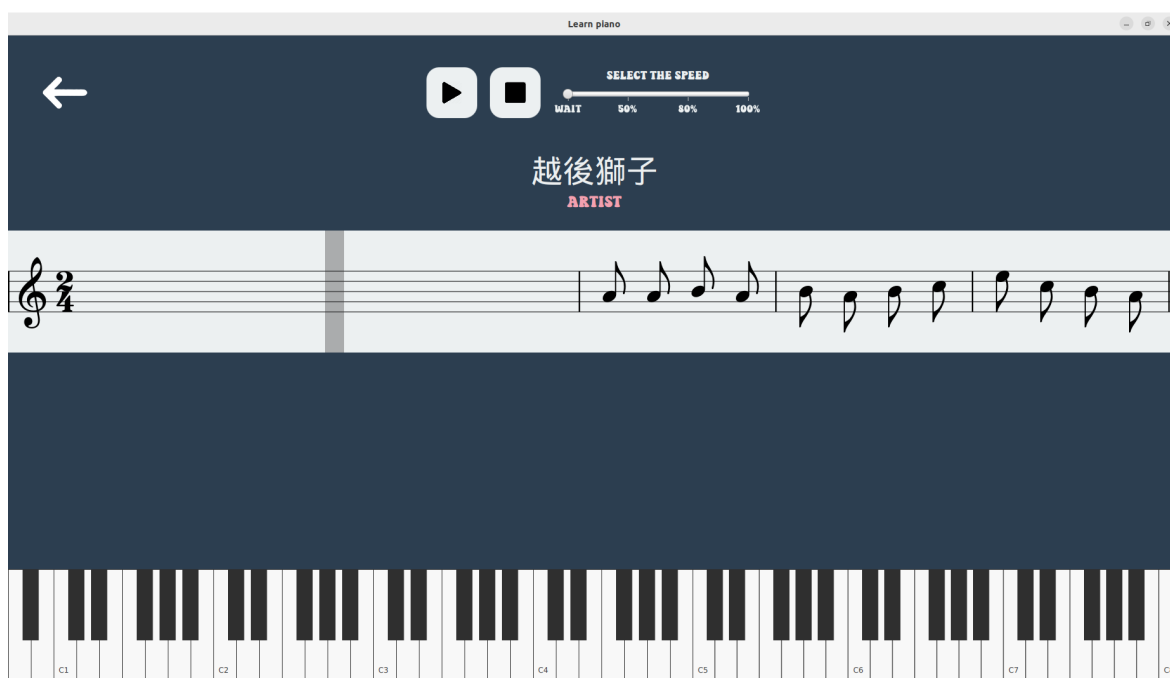
Slika 4.10. Glavni zaslon aplikacije za vrijeme reprodukcije

Sljedeći gumb služi za zaustavljanje reprodukcije, što uključuje pauziranje i vraćanje

notnog zapisa na početak, odnosno resetiranje reprodukcije.

U sredini alatne trake nalazi se klizač (engl. *slider*) za odabir brzine reprodukcije. Prva opcija, a ujedno i zadana, označena je oznakom *Wait* i omogućuje reprodukciju gdje aplikacija čeka da korisnik odsvira točne tipke, dok se prijelazi između odsviranih nota odvijaju normalnom brzinom. Ostale tri opcije klizača su *50%*, *80%* i *100%*, što označava različite brzine reprodukcije. Kod početka učenja pjesme, sporija reprodukcija može olakšati praćenje, dok će napredniji korisnici moći svirati punom brzinom.

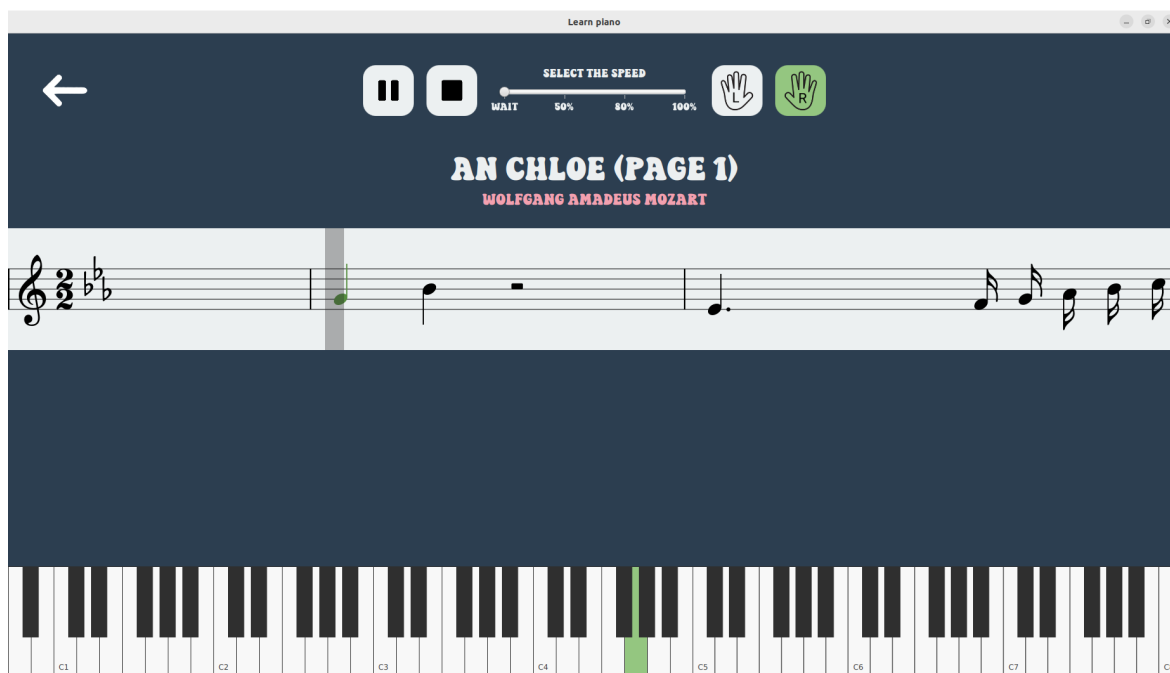
Zadnja dva gumba alatne trake služe za odabir ruke koju se želi vježbati. Sviranje svake ruke zasebno olakšava proces učenja, jer omogućuje fokusiranje na tehniku i točnost svake ruke posebno prije nego što se pređe na zajedničko sviranje. U slučaju da se pjesma svira samo jednom rukom, ti gumbi se neće prikazati, kao što je prikazano na slici 4.11., i tada se drugo crtovlje također ne prikazuje.



Slika 4.11. Glavni zaslon kad pjesma sadrži note za samo jednu ruku

Kada je pjesma koja sadrži note za obje ruke prikazana, oba gumba su aktivna, što se može vidjeti na slici 4.9., gdje je gumb za lijevu ruku obojen plavo, a gumb za desnu ruku zeleno. Isključivanjem jedne od ruku, gumb te ruke postaje bijeli, a crtovlje za tu ruku više se ne prikazuje. Na slici 4.12. prikazano je kako izgleda kad je lijeva ruka isključena, pa se prikazuje samo crtovlje za desnu ruku. Ako bi desna ruka bila isključena, prikazalo bi se samo crtovlje za lijevu ruku, koje bi se automatski pomaknulo na mjesto crtovlja

desne ruke, odnosno pozicioniralo više.



Slika 4.12. Glavni zaslon kad pjesma sadrži note za samo jednu ruku

Alatnom trakom upravlja kontroler *PlayViewController*, koji također nadzire cijeli zaslon. Ovaj kontroler prima korisničke akcije iz alatne trake i prosljeđuje odgovarajuće upute dalje. Uz njega, na upravljanju glavnim zaslonom sudjeluju još dva kontrolera: *SheetMusicController*, koji upravlja notnim zapisom, i *PianoKeyboardController*, koji kontrolira prikaz klavijature.

4.5.2. Notni zapis

U sredini zaslona nalazi se notni zapis. Ispod notnog zapisa, a iznad alatne trake, prikazani su naziv pjesme i ime autora. Ovi podaci u MusicXML datoteci nalaze se u zaglavlju koje sadrži različite metapodatke. Naziv pjesme može se pronaći na dva mjesta: unutar oznake `<movement-title>` ili u oznaci `<work-title>`, koja je smještena unutar oznake `<work>`, a aplikacija ih oba uzima u obzir.

Oznaka `<creator>` u MusicXML dokumentu, smještena unutar oznake `<identification>`, koristi se za označavanje osoba koje su doprinijele stvaranju skladbe. Može se koristiti za označavanje različitih uloga, poput kompozitora, tekstopisca ili aranžera, putem atributa *type* (npr. *composer*, *lyricist*, *arranger*). Unutar oznake `<creator>` navodi se ime osobe koja je ispunila tu ulogu. Budući da se uloga *composer* pojavljuje u gotovo

svim MusicXML dokumentima, bilo da se radi o klasičnim skladbama ili modernim pjesmama, ime navedeno u ovoj oznaci prikazuje se kao autor pjesme na zaslonu.

Ispod naslova pjesme nalazi se notni zapis, koji može sadržavati jedan ili dva dijela (crtovlja), ovisno o tome izvodi li se skladba s jednom ili dvije ruke. Svaki dio notnog zapisa sastoji se od četiri elementa. Prvi element je notno crtovlje, koje se sastoji od pet horizontalnih linija na kojima se prikazuju note i druge glazbene oznake.

Određivanje atributa

Drugi element odnosi se na stalne attribute pjesme koji se nalaze na lijevom kraju crtovlja, čija se pozicija i vrijednosti ne mijenjaju tijekom izvođenja. Ovi atributi uključuju osnovne informacije o pjesmi, kao što su ključ, mjera i tonalitet (prikazan kroz povisilice i snizilice), a nalaze se na početku notnog zapisa.

Svi ovi atributi mogu se pronaći unutar oznake `<attributes>` u MusicXML datoteci. Ova oznaka smještena je unutar prvog takta klavirskog dijela skladbe. Primjer strukture oznake `<attributes>` prikazan je na slici 4.13., dok je njen odgovarajući notni zapis generiran u aplikaciji prikazan na slici 4.14.

Prvi element `<divisions>`, s vrijednošću 24, definira relativno trajanje četvrtinke. Iako ova oznaka nije izravno vezana za attribute na početku notnog zapisa, od ključne je važnosti za kasniji izračun tempa. Slijedi oznaka `<key>`, koja određuje tonalitet skladbe i sadrži dva podelementa. Prvi `<fifths>`, označava broj povisilica (ako je vrijednost pozitivna) ili snizilica (ako je vrijednost negativna) u tonalitetu. Drugi element `<mode>`, određuje je li tonalitet durski (engl. *major*) ili molski (engl. *minor*). U ovom primjeru, vrijednost *fifths* iznosi -3, što znači da tonalitet sadrži tri snizilice. U kombinaciji s vrijednošću *major*, to označava tonalitet Eb-dur.

Tonalitet je sustav organizacije tonova u skladbi, gdje se svi tonovi usmjeravaju prema tonici, glavnom tonu. Ovisno o ljestvici, tonalitet može biti durski (vedar) ili molski (tamniji). Prepoznaje se prema predznacima (povisilicama \sharp i snizilicama b) iza ključa u notnom zapisu, koji pokazuju povišene ili snižene tonove. Na primjer, C-dur i njegov paralelni a-mol nemaju predznake, dok G-dur ima jednu povisilicu (F \sharp). Postoji 30 različitih tonaliteta, ali zbog enharmonije neki zvuče isto, iako imaju različite nazive (npr.

```

<attributes>
  <divisions>24</divisions>
  <key>
    <fifths>-3</fifths>
    <mode>major</mode>
  </key>
  <time symbol="cut">
    <beats>2</beats>
    <beat-type>2</beat-type>
  </time>
  <staves>2</staves>
  <clef number="1">
    <sign>G</sign>
    <line>2</line>
  </clef>
  <clef number="2">
    <sign>F</sign>
    <line>4</line>
  </clef>
</attributes>

```

Slika 4.13. Atributi u MusicXML datoteci



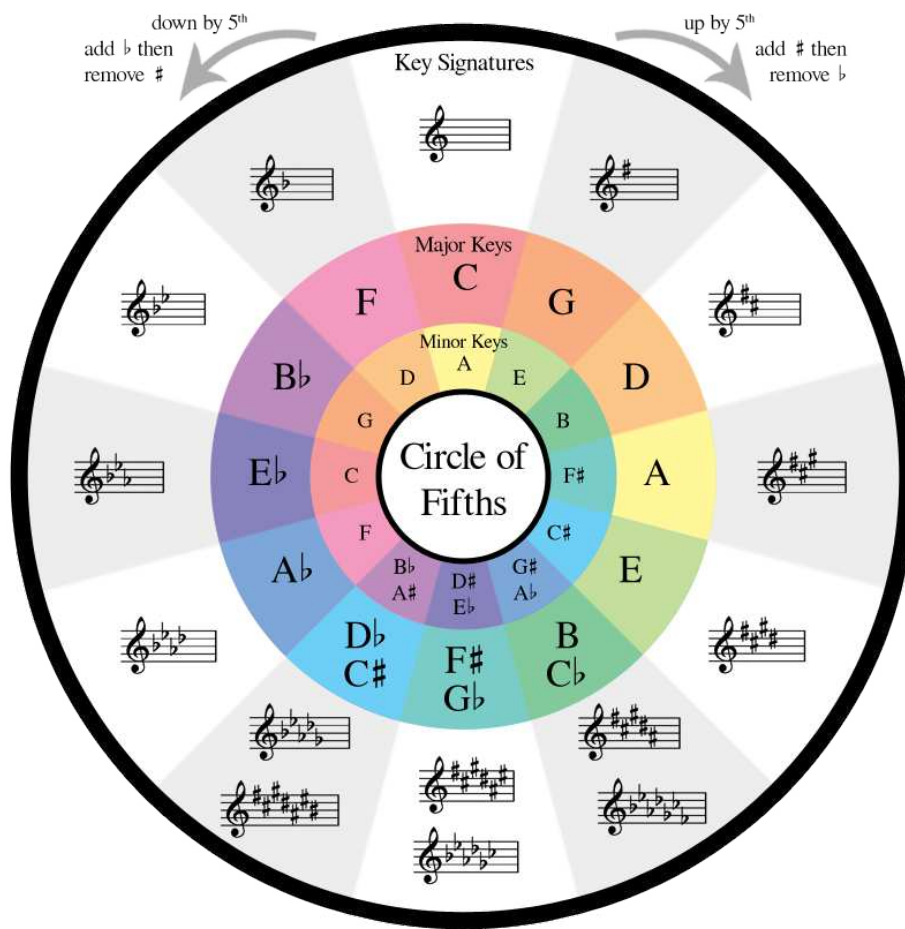
Slika 4.14. Notni zapis atributa u aplikaciji

Slika 4.15. Primjer atributa skladbe

C \sharp -dur i D \flat -dur). Zbog toga se zapravo koristi 24 različita tonaliteta, 12 durskih i 12 molskih, prikazanih na slici 4.16.

Tonaliteti se organiziraju u kvintni (pomak za kvintu dodaje povisilice) i kvartni krug (pomak za kvartu dodaje snizilice). Ova dva kruga su blisko povezana, jer se do istog tonaliteta može doći pomakom kvinte unatrag ili kvarte unaprijed. Na primjer, C-dur prelazi u F-dur pomakom kvarte unaprijed ili kvinte unatrag. Zbog toga se često spominje samo kvintni krug, gdje se pomaci u oba smjera prikazuju jednom vrijednošću s pozitivnim ili negativnim predznakom, kao što je to slučaj u MusicXML formatu.

Oznaka *<time>* definira mjeru pomoću dva elementa. Element *<beats>* određuje broj udaraca (engl. *beat*) i odgovara brojnika mjere, dok element *<beat-type>* određuje vrstu udarca (npr. 2 za polovinku, 4 za četvrtinku) i predstavlja nazivnik mjere. U ovom



Slika 4.16. Prikaz kvintnog i kvartnog kruga [19]

primjeru riječ je o dvopolovinskoj mjeri (2/2).

Oznaka `<staves>` s vrijednošću 2 označava da skladba koristi dva notna crtovlja, a detaljnije je objašnjena u pododjeljku 4.2.1.

Oznaka `<clef>` koristi se za određivanje ključa u notnom zapisu, pri čemu je podelement `<sign>` ključan za identifikaciju. Vrijednost ovog elementa određuje tip ključa pri čemu se najčešće koriste: *G* za violinski ključ i *F* za bas ključ, oba koja su korištena u ovom radu. Podelement `<line>` određuje na kojoj liniji notnog crtovlja se ključ postavlja. Violinski ključ (*G*) se standardno nalazi na drugoj liniji od dna, dok je bas ključ (*F*) smješten na četvrtoj liniji, kao što je i u primjeru.

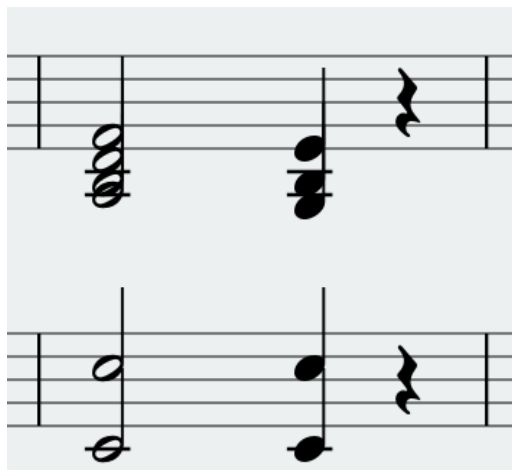
Ako neki od ovih atributa nije naveden postavlja se na zadane postavke: *divisions* na 8, *fifths* na 0, *beats* na 4, *beat-unit* na 4, *staves* na 1 i *sign* na *G*.

Određivanje i pozicioniranje notnog zapisa

Desno od atributa na notnom crtovlju nalazi se ostatak notnog zapisa koji sadrži same note. Korijenjski element MusicXML datoteke, `<score-partwise>`, koji koristi moja aplikacija, organizira sadržaj u popis dijelova (`<part>`), od kojih jedan mora biti namijenjen klaviru. Element `<part>` sastoji se od niza elemenata `<measure>`, koji su vremenski porredani. Svaki `<measure>` element predstavlja jedan takt, što je osnovna ritmička jedinica u glazbi. Moja aplikacija na sličan način organizira note po taktovima i prikazuje ih na ekranu. Primjer strukture jednog takta u MusicXML formatu prikazan je na slici 4.17., dok se odgovarajući notni zapis može vidjeti na slici 4.18.

```
3 <score-partwise version="4.0" > 3 <score-partwise version="4.0" > 3 <score-partwise version="4.0" > 3 <score-partwise version="4.0" >
1019 <part id="P2"> 1019 <part id="P2"> 1019 <part id="P2"> 1019 <part id="P2">
2121 <measure number="6" width="194"> 2121 <measure number="6" width="194"> 2121 <measure number="6" width="194"> 2121 <measure number="6" width="194">
2122 <note default-x="13"> 2122 <note default-x="80"> 2122 <note default-x="13"> 2122 <note default-x="13">
2123 <pitch> 2123 <type>quarter</type> 2123 </note> 2123 <pitch>
2124 <step>4</step> 2124 <stem>up</stem> 2124 <note default-x="13"> 2124 <octave>3</octave>
2125 <alter>-1</alter> 2125 <staff>1</staff> 2125 </chord/> 2125 </pitch>
2126 <octave>3</octave> 2126 </note> 2126 <step>E</step> 2126 <stem>up</stem>
2127 </pitch> 2127 <note default-x="80"> 2127 <duration>48</duration> 2127 <type>half</type>
2128 <duration>48</duration> 2128 <chord/> 2128 <alter>-1</alter> 2128 <octave>3</octave>
2129 <type>half</type> 2129 <pitch> 2129 <step>E</step> 2129 <stem>up</stem>
2130 <stem default-y="0">up</stem> 2130 <alter>-1</alter> 2130 <duration>48</duration>
2131 <staff>1</staff> 2131 <octave>4</octave> 2131 <type>half</type> 2131 <stem>up</stem>
2132 </note> 2132 </pitch> 2132 <staff>2</staff> 2132 <duration>24</duration>
2133 <note default-x="25"> 2133 </note> 2133 <pitch> 2133 <step>E</step>
2134 <chord/> 2134 <duration>24</duration> 2134 <type>quarter</type> 2134 <stem>up</stem>
2135 <pitch> 2135 <step>B</step> 2135 <staff>1</staff> 2135 <octave>3</octave>
2136 <step>B</step> 2136 <stem>up</stem> 2136 </note> 2136 </pitch>
2137 <alter>-1</alter> 2137 <pitch> 2137 <duration>24</duration>
2138 <octave>3</octave> 2138 <stem>up</stem> 2138 <type>quarter</type>
2139 </pitch> 2139 <note default-x="123"> 2139 <staff>1</staff> 2139 <step>E</step>
2140 <duration>48</duration> 2140 <rest/> 2140 <alter>-1</alter>
2141 <type>half</type> 2141 <duration>24</duration> 2141 <type>quarter</type>
2142 <stem>up</stem> 2142 <staff>1</staff> 2142 <pitch> 2142 <step>E</step>
2143 <staff>1</staff> 2143 <octave>2</octave> 2143 <stem default-y="18">up</stem>
2144 </note> 2144 </note> 2144 <duration>24</duration> 2144 <type>quarter</type>
2145 <note default-x="13"> 2145 <backup> 2145 <pitch> 2145 <step>E</step>
2146 <chord/> 2146 <duration>96</duration> 2146 </note> 2146 <alter>-1</alter>
2147 <step>B</step> 2147 </pitch> 2147 <note default-x="80"> 2147 <octave>3</octave>
2148 <octave>4</octave> 2148 <chord/> 2148 <pitch> 2148 <duration>24</duration>
2149 </pitch> 2149 <pitch> 2149 <step>E</step>
2150 </pitch> 2150 <duration>48</duration> 2150 <alter>-1</alter>
2151 <duration>48</duration> 2151 <type>quarter</type> 2151 <stem>up</stem>
2152 <type>half</type> 2152 <octave>2</octave> 2152 <pitch> 2152 <step>E</step>
2153 <stem>up</stem> 2153 </pitch> 2153 <duration>48</duration>
2154 <staff>2</staff> 2154 <duration>24</duration> 2154 <type>quarter</type>
2155 </note> 2155 <note default-x="123"> 2155 <rest/>
2156 </pitch> 2156 <duration>24</duration> 2156 <type>quarter</type>
2157 <chord/> 2157 <stem>up</stem> 2157 <duration>24</duration>
2158 <pitch> 2158 <staff>2</staff> 2158 <type>quarter</type>
2159 <step>F</step> 2159 </note> 2159 <note default-x="123"> 2159 <pitch>
2160 <octave>4</octave> 2160 <rest/> 2160 <pitch> 2160 <step>E</step>
2161 </pitch> 2161 <pitch> 2161 <duration>24</duration> 2161 </measure>
```

Slika 4.17. Prikaz takta u MusicXML datoteci



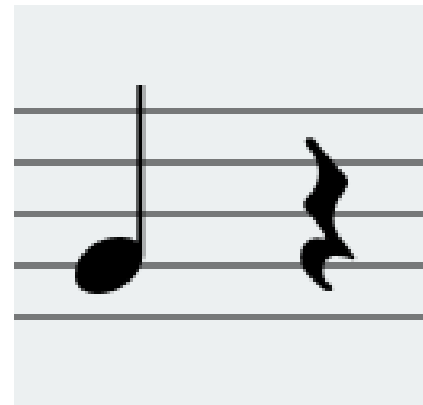
Slika 4.18. Prikaz takta u aplikaciji

Element `<measure>` može sadržavati razne podelemente, poput `<attributes>`, ali dva ključna za prezentaciju takta u mom slučaju su `<note>` i `<backup>`. Kako bih pojednostavila prikaz i praćenje reprodukcije taktova, kreirala sam razred `MeasurePairModel`. Ovaj model uključuje listu nota za desnu i, ukoliko se koristi, listu za lijevu ruku. Budući da su taktovi u obje ruke jednake širine i trajanja, taj model također sadrži informacije o tim parametrima.

`MeasurePairModel` ne pohranjuje izravno listu nota, već listu objekata tipa `MusicNodeModel`. Ovaj objekt obuhvaća sve note, kao i dodatne elemente, poput pomoćnih crta, koje se sviraju istovremeno. Na primjeru prikazanom na slici 4.18., u gornjem crtovlju prikazan je prvi akord. Taj akord je predstavljen jednim objektom `MusicNodeModel`, koji sadrži četiri note i dvije pomoćne crte, što ukupno čini šest objekata tipa `NoteModel`.

```
<note default-x="133">
  <pitch>
    <step>B</step>
    <alter>-1</alter>
    <octave>2</octave>
  </pitch>
  <duration>24</duration>
  <type>quarter</type>
  <stem default-y="5">up</stem>
  <staff>2</staff>
</note>
<note default-x="189">
  <rest/>
  <duration>24</duration>
  <type>quarter</type>
  <staff>2</staff>
</note>
```

Slika 4.19. Primjer note i pauze u MusicXML datoteci



Slika 4.20. Notni zapis note i pauze

Slika 4.21. Primjer note i pauze

`NoteModel` predstavlja bilo koji element notnog zapisa, bilo da se radi o notama, pa-

uzama ili drugim pomoćnim elementima. Na slici 4.19. prikazan je MusicXML zapis jedne note i jedne pauze, dok je na slici 4.20. prikazana ista nota i pauza u notnom zapisu unutar aplikacije.

Note uključuju element `<pitch>`, koji određuje visinu tona koristeći tri svojstva: ton unutar oktave (`<step>`), broj oktave (`<octave>`) te opcionalno predznak (`<alter>`), gdje je -1 predznak za sniženi ton, a 1 za povišeni. U prethodnom primjeru radi se o tonu H_b2 (engl. B_b2). Budući da je pjesma napisana u tonalitetu koji već uključuje sniženi ton H, na notnom crtovlju nije dodatno označeno sniženje.

Note mogu opcionalno imati oznaku `<stem>` s vrijednostima *up* ili *down*, koja određuje smjer drške note – prema gore ili dolje. Pauze se označavaju korištenjem elementa `<rest/>`.

Sljedeća tri elementa zajednička su za note i za pauze. Element `<duration>` označava relativno trajanje note ili pauze, `<type>` definira tip note (npr. *quarter* za četvrtinku ili četvrtinsku pauzu), a `<staff>` označava crtovlje na kojem se nota ili pauza nalazi. U prikazanom primjeru, broj 2 označava drugo crtovlje, odnosno ono za lijevu ruku.

Note unutar takta pozicioniraju se uzimajući u obzir najkraću notu u taktu. Note te duljine smještaju se na jediničnu udaljenost jedna od druge, dok su ostale note raspoređene proporcionalno njihovom trajanju. Ovaj pristup osigurava ravnomjeren vizualni ritam notnog zapisa, omogućujući da se zapis pomiče ujednačenom brzinom tijekom reprodukcije takta. Reprodukcijska pjesma, odnosno pomicanje notnog zapisa ulijevo, odvija se brzinom koja ovisi o tempu skladbe i širini takta. Budući da postoji dodatna udaljenost između početka takta i prve note, prijelaz između zadnje note prethodnog takta i prve note sljedećeg takta vizualno se čini bržim.

Kontrolna linija

Četvrti element notnog crtovlja je kontrolna linija, prozirno sive boje, koja se proteže preko oba crtovlja i služi kao orijentir za sviranje nota. Svaki takt sadrži slušač (engl. *listener*), čija je uloga obavješćavanje *PianoKeyboardControllera* o tome kada treba odsvirati ili prekinuti sviranje određenih nota.

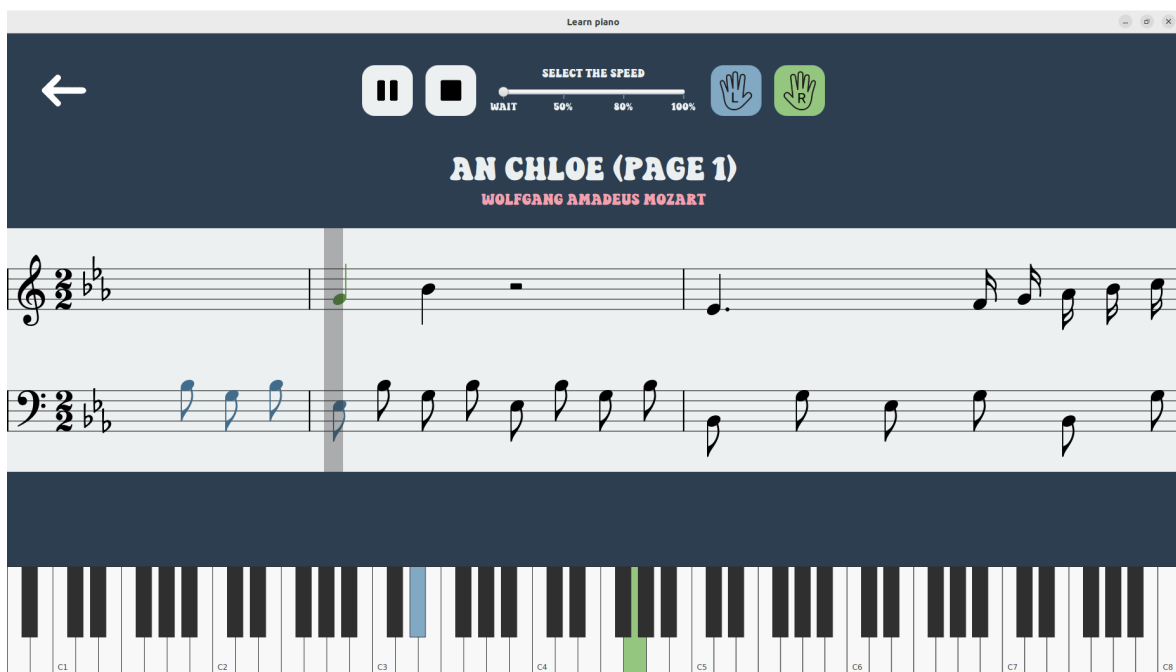
Kada objekt *MusicNodeModel* uđe u područje kontrolne linije, a note još nisu odsvi-

rane, slušač tog takta detektira ovaj događaj i obavještava sustav o notama koje treba odsvirati. U tom trenutku te se note označuju kao odsvirane, kako bi se izbjeglo njihovo ponovno aktiviranje. Neposredno prije nego što *MusicNodeModel* uđe u kontrolnu liniju, aktivira se drugi slušač, zadužen za obavještavanje o prekidu sviranja prethodnih nota, kako bi se sustav mogao pripremiti za nadolazeće note.

4.5.3. Klavijatura

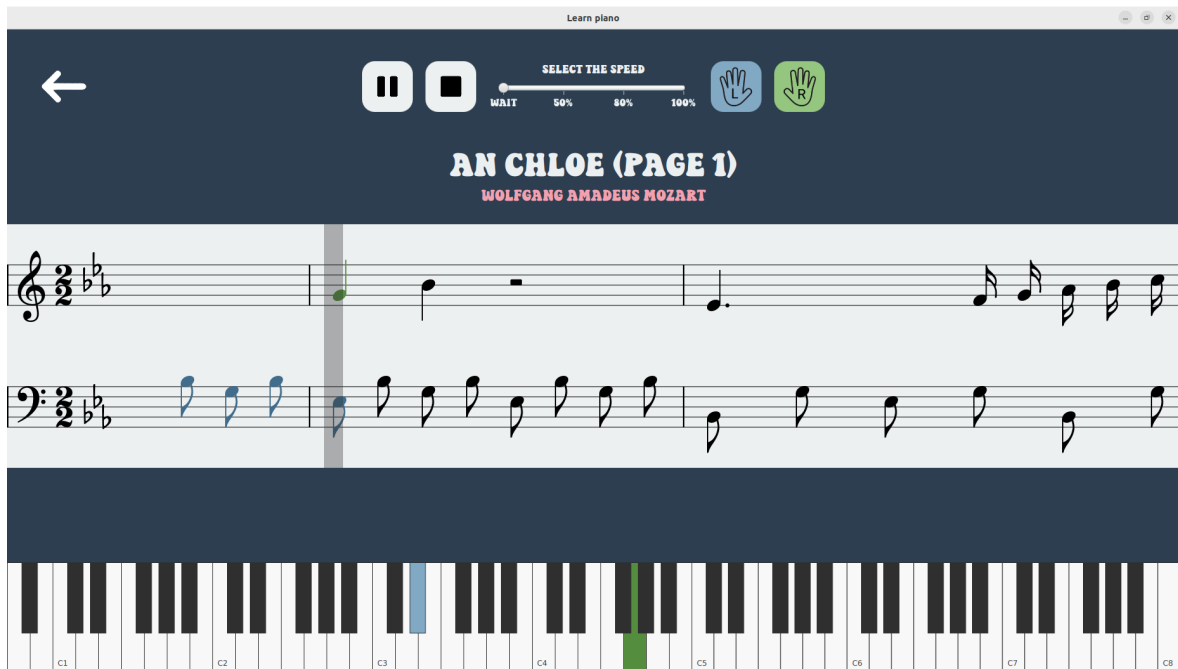
Na dnu zaslona nalazi se prikaz klavijature, koja obuhvaća cijeli raspon koncertnog klavira, od A0 do C8. Veličina klavijature dinamički se prilagođava širini prozora, osiguravajući optimalan prikaz bez obzira na veličinu zaslona. Klavirske tipke implementirane su kao instance razreda *Rectangle* iz JavaFX biblioteke. Radi lakšeg snalaženja, tipke s tonom C sadrže oznaku na dnu, a uz to je na njima ispisan i broj oktave.

Tijekom reprodukcije skladbe, tipke koje treba odsvirati ističu se drugačijom bojom. Na slici 4.22. prikazan je izgled istaknutih tipki prije nego što ih korisnik odsvira. Tipke koje se sviraju lijevom rukom označene su plavom bojom, dok su tipke za desnu ruku označene zelenom. Kako bih osigurala lakše praćenje, te su boje dosljedno korištene u cijeloj aplikaciji, uključujući gumbe za odabir ruke i prikaz odsviranih nota na notnom zapisu.



Slika 4.22. Prikaz istaknutih tipki na klavijaturi dok se ne sviraju

Kada korisnik pritisne ispravne tipke, one promijene boju na tamniju, što se može vidjeti na slici 4.23. Pritisnuta je tipka G4, te je promijenila boju u tamno zelenu. Ako korisnik otpusti ispravnu tipku, a nota još nije završila sa sviranjem, boja tipke vraća se na svjetliju varijantu.



Slika 4.23. Prikaz istaknutih tipki na klavijaturi za vrijeme sviranja

Pritisak tipki koje se ne sviraju u tom trenutku ne prikazuje se na klavijaturi. Kada nota završi, boja pripadajuće tipke odmah se vraća u normalno stanje. Upravljanje bojama tipki obavlja *PianoKeyboardController*, koji također upravlja zvučnom reprodukcijom skladbe putem singleton razreda *MidiPlayback*.

Razred *MidiPlayback* odgovoran je za otvaranje MIDI kanala i slanje MIDI poruka za reprodukciju. Proces otvaranja kanala uključuje sljedeće korake: prvo se inicijalizira *Synthesizer*, zatim se učitava instrument 0 (klavir), i na kraju se dohvaća prvi kanal. Kod za taj proces prikazan je u nastavku.

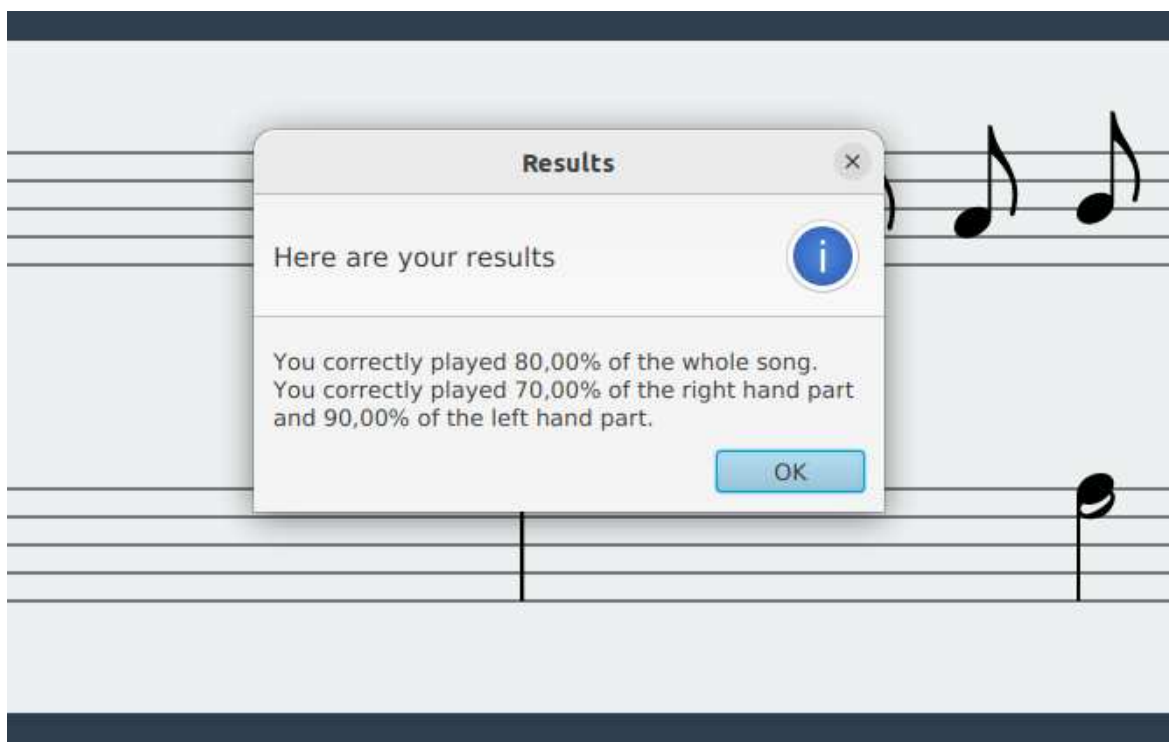
```
try {  
    synthesizer = MidiSystem.getSynthesizer();  
    synthesizer.open();  
    Instrument[] instruments = synthesizer.getDefaultSoundbank().  
        getInstruments();  
    synthesizer.loadInstrument(instruments[0]);  
    channel = synthesizer.getChannels()[0];  
}
```

```
channel.programChange(0);  
} catch (MidiUnavailableException ignored) {}
```

Kada je kanal otvoren, reprodukcija note ostvaruje se pomoću metode `channel.noteOn(midiKey, volume)`, gdje je `midiKey` numerička vrijednost tipke u rasponu od 0 do 127, a `volume` predstavlja glasnoću izvedbe u rasponu od 0 do 100. Za zaustavljanje reprodukcije note koristi se metoda `channel.noteOff(midiKey)`.

4.5.4. Povratna informacija o uspjehu sviranja

Nakon što korisnik završi sviranje skladbe, otvara se prozor s povratnom informacijom o kvaliteti izvedbe. Prikaz tog prozora može se vidjeti na slici 4.24., a radi bolje preglednosti, slika je uvećana.



Slika 4.24. Prikaz prozora s povratnom informacijom

Trenutno moja aplikacija prikazuje samo postotak ispravno odsviranih nota. Ako skladba sadrži note za jednu ruku, prikazuje se samo poruka o ukupnoj uspješnosti izvedbe. Međutim, ako skladba uključuje note za obje ruke, prozor prikazuje i detaljne informacije o uspješnosti svake ruke zasebno, uz ukupan rezultat. Tijekom sviranja skladbe, aplikacija prati koje je note korisnik ispravno odsvirao i na temelju toga izračunava postotak uspješnosti.

5. Analiza rezultata

Pokretanjem aplikacije koristeći različite datoteke, uočila sam da notni zapis, a time i prikaz na klavijaturi, nije uvijek jednak očekivanom. Međutim to nije toliki problem, jer da bih pokrila sve mogućnosti notnog zapisa u svojoj aplikaciji, bilo bi potrebno puno više vremena, poznavanja teorije glazbe i iskustva u programiranju.

U odjeljku 5.1. navesti ću rezultate pokretanja aplikacije, te dobre i loše strane implementacije programa. U odjeljku 5.2. iznijeti ću neke prijedloge za unaprijeđenje i nadogradnju aplikacije u budućnosti. Za prikaz rezultata koristiti ću početak skladbe "An Chloe" skladatelja Wolfganga Amadeusa Mozarta.

5.1. Prikaz rezultata

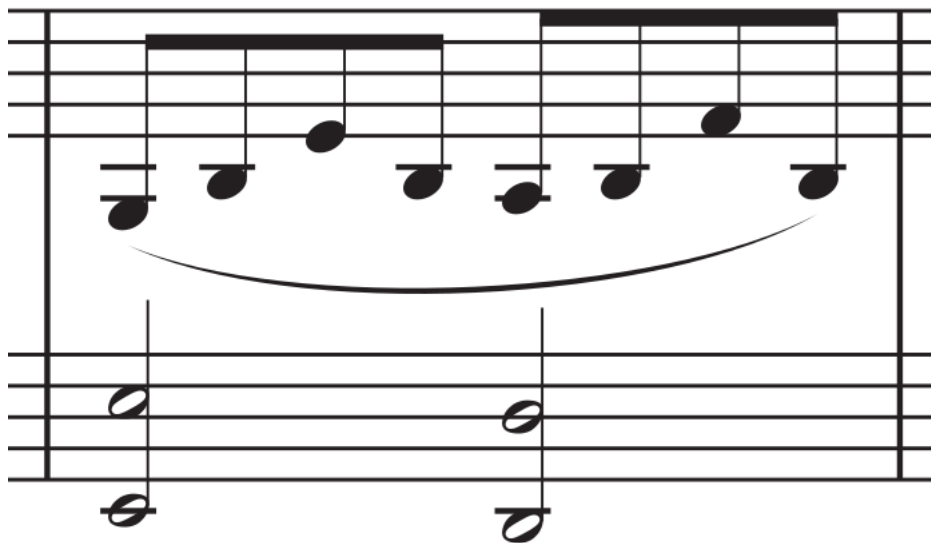
Ako korisnik učita datoteku koja je u ispravnom formatu, opisanom u pododjeljku 4.2.1., aplikacija će ispravno funkcionirati. Prema standardu, notni zapis za različite ruke ili crtovlja namijenjen istom instrumentu treba biti unutar istog dijela, dok se razlika između crtovlja označava oznakom *<staff>* unutar svake note. Međutim, mogu se pronaći datoteke koje ne prate ovaj standard, gdje su note za različita crtovlja smještene u različite dijelove. To predstavlja izazov prilikom deserijalizacije dokumenta, a trenutna verzija moje aplikacije ne podržava takav format. Ako korisnik učita takvu datoteku, ona se ili neće učitati – ako nijedan od dijelova ne sadrži naziv "Piano" – ili će prikazivati samo jedno crtovlje, odnosno jednu ruku.

Notni zapis se u većini slučajeva prikazuje ispravno, što je prikazano na slici 5.1. Na toj slici vidi se prikaz jednog takta unutar moje aplikacije. Za usporedbu, na slici 5.2. prikazan je isti taj takt u tradicionalnom notnom zapisu. Može se primijetiti da su u aplikaciji sve note ispravno pozicionirane i odgovarajućeg trajanja. Također, pomoćne

crte, koje se koriste za prikaz nota izvan standardnog raspona notnog crtovlja, ispravno su prikazane.



Slika 5.1. Prikaz takta u aplikaciji

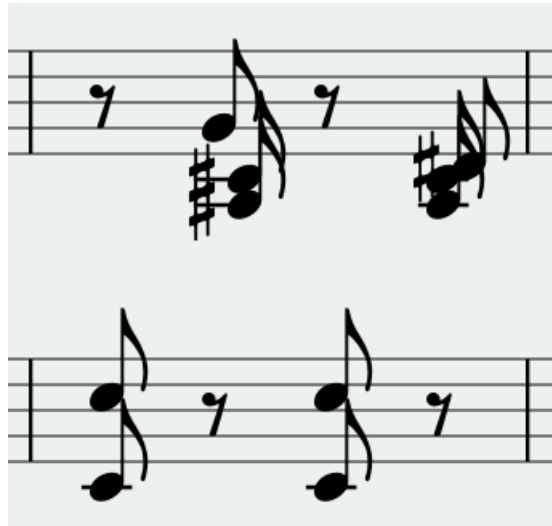


Slika 5.2. Prikaz takta u postojećem notnom zapisu

Kada se više osminki ili kraćih nota pojavljuje uzastopno, one se obično povezuju linijama, kao što je prikazano na slici 5.2. U mojoj aplikaciji, međutim, to nisam uspjela implementirati, pa se svaka nota prikazuje odvojeno. Zbog tehničkih ograničenja, nisam mogla koristiti font Bravura za ovu funkcionalnost, pa bi povezivanje nota trebalo ručno nacrtati. U budućnosti smatram da bi bilo korisno implementirati ovu značajku kako bi prikaz bio jasniji korisnicima.

Još jedan problem koji bih željela ispraviti je prikaz osminki i kraćih nota kada su dio

akorda. Na slici 5.3. prikazana su četiri akorda s osminkama, no svaka nota ima svoju vlastitu dršku i zastavicu, što otežava čitljivost zapisa. Umjesto toga, sve note unutar jednog takta trebale bi dijeliti istu dršku s jednom zajedničkom zastavicom, kako bi notni zapis bio jasniji i pregledniji.



Slika 5.3. Prikaz neispravno prikazanih osminki

Također, predznaci ispred nota (npr. povisilice na slici 5.3.) ne bi smjeli utjecati na položaj same note. Nota bi trebala zadržati svoju uobičajenu poziciju, dok bi se predznak trebao pravilno uskladiti s njom, bez promjene razmaka.

Unatoč ovim problemima, notni zapis se u većini slučajeva ispravno prikazuje, a zvučna reprodukcija i označene tipke na klaviru većinom funkcioniraju kako treba. S obzirom na to da glazba obuhvaća veliki broj različitih elemenata, teško je uvijek uzeti u obzir sve aspekte i osigurati njihovu ispravnu izvedbu.

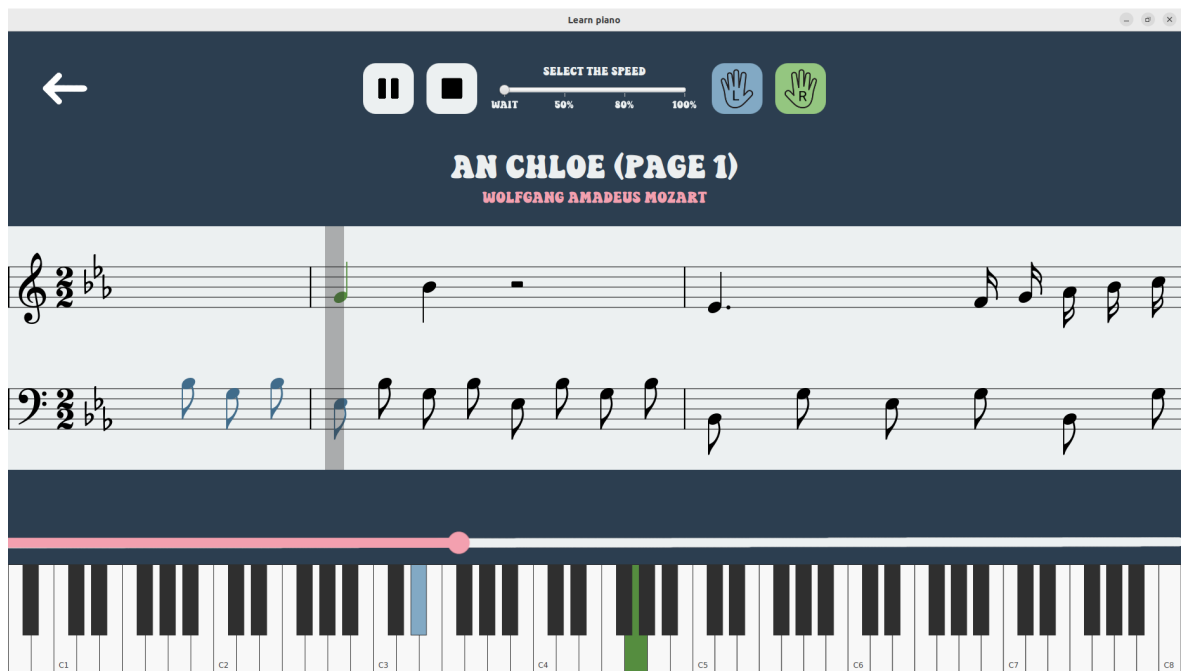
5.2. Mogućnost nadogradnje

S obzirom na ograničeno vrijeme za izradu ovog diplomskog rada, nisam bila u mogućnosti implementirati sve funkcionalnosti koje bih htjela. Iako sam zadovoljna trenutnim stanjem aplikacije i vjerujem da sam uspjela razviti solidan program, svjesna sam da postoji prostor za daljnje nadogradnje. U nastavku ću navesti neke od glavnih funkcionalnosti koje bih voljela uključiti u budućim verzijama aplikacije.

Prva stvar koju bih htjela unaprijediti u aplikaciji je proširenje biblioteke pjesama. Većina sličnih aplikacija nudi velik izbor pjesama različitih razina zahtjevnosti, koje ko-

risnici mogu svirati. Iako bih htjela povećati broj dostupnih pjesama, želim zadržati primarni fokus aplikacije na mogućnosti učitavanja vlastitih datoteka, jer mislim da je to velika prednost ove aplikacije nad mnogima koje već postoje. Međutim, zbog kompleksnosti implementacije te funkcionalnosti, proširenje biblioteke mi nije prioritet.

Prilikom učenja nove skladbe, glazbenici često ne vježbaju od samog početka, već se fokusiraju na određene dijelove kompozicije. Trenutno, moja aplikacija omogućava reprodukciju isključivo od početka skladbe. Zbog toga bih u budućim verzijama željela implementirati mogućnost "navigacije" kroz skladbu, koja bi omogućila reprodukciju od odabranog dijela. Koncept ove funkcionalnosti prikazan je na slici 5.4., gdje bi korisnici povlačenjem pokazivača mogli odrediti točku od koje žele nastaviti reprodukciju.



Slika 5.4. Ideja prikaza pokazivača reprodukcije

Iduća funkcionalnost koju bih željela implementirati je metronom. Metronom pomaže glazbenicima održavati ritam proizvodnjom ritmičkih zvučnih signala. Tijekom pokušaja implementacije, naišla sam na izazov jer nisam uspjela osigurati da metronom radi ispravno. Zbog potrebe za višedretvenošću (engl. *multithreading*), otkucaji metronoma nisu bili sinkronizirani s reprodukcijom i prikazom glazbe. Shvatila sam da, kako bi otkucaji metronoma bili potpuno usklađeni s ostatkom reprodukcije, metronom bi trebao raditi unutar iste dretve i u zajedničkoj petlji s ostatkom koda. Međutim, zbog trenutne strukture aplikacije, ova implementacija nije bila moguća.

S obzirom na to da je u aplikaciji već implementirana zvučna reprodukcija glazbe, dodavanje metronoma moglo bi stvoriti poteškoće glazbenicima pri koncentraciji zbog istovremenog slušanja oba izvora zvuka. Zbog toga mislim da bi optimalno rješenje bilo uvesti dva gumba: jedan za isključivanje metronoma i drugi za isključivanje reprodukcije glazbe. Na taj način, glazbenici bi imali fleksibilnost da sami biraju žele li slušati samo metronom, samo glazbenu reprodukciju, oboje ili nijedno.

Posljednja funkcionalnost koju bih htjela implementirati odnosi se na poboljšanje povratnih informacija o kvaliteti sviranja. Trenutno aplikacija prikazuje samo postotak ispravno odsviranih nota na kraju skladbe, no smatram da bi korisnicima bilo korisno pratiti svoj napredak tijekom vremena. Kako bi se to omogućilo, aplikacija bi trebala pohranjivati podatke o izvedbama, što bi zahtijevalo implementaciju trajne memorije, poput lokalne baze podataka ili integracije s oblakom. Mislim da bi ova funkcionalnost motivirala korisnike na daljnje vježbanje.

6. Zaključak

Izrada ove aplikacije omogućila mi je razumijevanje izazova povezanih s primjenom računalnih tehnologija u kreativnim i subjektivnim područjima poput glazbe. Dok računala zahtijevaju preciznost i determinizam, glazba često odstupa od tih pravila, što čini razvoj ovakvih aplikacija relativno zahtjevnim.

Unatoč tim izazovima, uspjela sam razviti aplikaciju koja, iako ne može zamijeniti tradicionalne metode učenja i čitanje pravog notnog zapisa, predstavlja korisno sredstvo za brže i lakše savladavanje novih skladbi. Aplikacija omogućuje korisnicima učenje sviranja klavira na interaktivan i zabavan način, pružajući im mogućnost praćenja napretka i ispravljanja pogrešaka. Iako aplikacija ne može uvijek generirati 100% točne upute za sviranje, smatram da je izuzetno korisna za one koji žele brzo naučiti željene pjesme, štedeći im vrijeme i trud koji bi inače uložili u učenje čitanja notnog zapisa.

Na kraju, ponosna sam na postignute rezultate i smatram da je ovaj rad značajan doprinos u primjeni računalne tehnologije u glazbenom obrazovanju. Budući da uživam u sviranju klavira i učenju novih pjesama, planiram nastaviti koristiti ovu aplikaciju, što će me potaknuti da ju i dalje unapređujem. Postoje brojne mogućnosti za poboljšanje i nadogradnju koje bih voljela implementirati u budućnosti. Vjerujem da bi ove nadogradnje dodatno povećale funkcionalnost, korisnost i zadovoljstvo korištenja aplikacije. Iako uvijek postoji prostor za napredak, aplikacija koju sam napravila predstavlja dobar temelj za njen daljnji razvoj.

Izvorni kod aplikacije implementirane u sklopu ovoga rada, kao i buduće verzije mogu se pronaći na poveznici [20].

Literatura

- [1] Synthesia, <https://synthesiagame.com/>, [pristupljeno 26. kolovoza 2024.].
- [2] Simply Piano, <https://www.hellosimply.com/>, [pristupljeno 26. kolovoza 2024.].
- [3] Yoni Tsafir, “A look under-the-hood of Simply Piano (Part 1)”, <https://medium.com/hellosimply/a-look-under-the-hood-of-simply-piano-part-1-7050d297f611>, svibanj 2018., [pristupljeno 26. kolovoza 2024.].
- [4] Flowkey, <https://www.flowkey.com/en>, [pristupljeno 26. kolovoza 2024.].
- [5] “flowkey - Learn piano”, <https://chromewebstore.google.com/detail/jlmilmpimaepjkihnncinbaooibgelhm>, studeni 2016., [pristupljeno 26. kolovoza 2024.].
- [6] Wikipedia, “Apache Maven”, https://en.wikipedia.org/wiki/Apache_Maven, srpanj 2024., [pristupljeno 27. kolovoza 2024.].
- [7] —, “JavaFX”, <https://en.wikipedia.org/wiki/JavaFX>, kolovoz 2024., [pristupljeno 28. kolovoza 2024.].
- [8] —, “MusicXML”, <https://en.wikipedia.org/wiki/MusicXML>, srpanj 2024., [pristupljeno 28. kolovoza 2024.].
- [9] Audiveris, “ProxyMusic - GitHub”, <https://github.com/Audiveris/proxymusic>, ožujak 2022., [pristupljeno 28. kolovoza 2024.].
- [10] —, “ProxyMusic - Maven Repository”, <https://mvnrepository.com/artifact/org.audiveris/proxymusic>, veljača 2022., [pristupljeno 29. kolovoza 2024.].

- [11] Wikipedia, “MIDI”, <https://en.wikipedia.org/wiki/MIDI>, kolovoz 2024., [pristupljeno 28. kolovoza 2024.].
- [12] Simply Piano, “MIDI\USB Connection Support”, <https://welcome.hellosimply.com/midi-recognition/>, [pristupljeno 29. kolovoza 2024.].
- [13] “SMuFL”, <https://www.smufl.org/>, [pristupljeno 29. kolovoza 2024.].
- [14] Daniel Spreadbury, “Standard Music Font Layout (SMuFL)”, <https://w3c.github.io/smufl/latest/index.html>, [pristupljeno 30. kolovoza 2024.].
- [15] “SMuFL- Fonts”, <https://www.smufl.org/fonts/>, [pristupljeno 30. kolovoza 2024.].
- [16] steinbergmedia, “bravura”, <https://github.com/steinbergmedia/bravura>, svibanj 2022., [pristupljeno 30. kolovoza 2024.].
- [17] “MuseScore”, <https://musescore.org/hr>, pristupljeno 10. rujna 2024.].
- [18] Bernardo Breve, Stefano Cirillo, Mariano Cuofano, “Perceiving space through sound: mapping human movements into MIDI”, https://www.researchgate.net/publication/343709022_Perceiving_space_through_sound_mapping_human_movements_into_MIDI, srpanj 2020., pristupljeno 4. rujna 2024.].
- [19] J. Walker, “Composing music - circle of fifths”, <https://www.derbyguitartutor.co.uk/post/2018/07/30/how-to-use-the-circle-of-fifths>, srpanj 2018., pristupljeno 6. rujna 2024.
- [20] Ana Bagić, “GitHub - Diplomski rad”, <https://github.com/ana-bagic/diplomski-rad>, rujan 2024., [GitHub repozitorij sa izvornim kodom aplikacije].

Sažetak

Aplikacija za pomoć učenju sviranja klavira

Ana Bagić

U ovom radu istraživala sam problem razvoja aplikacije za učenje sviranja klavira, specifično usmjerene na vježbanje izvođenja odabrane skladbe. Razvila sam aplikaciju koja omogućuje korisniku učitavanje željene skladbe u MusicXML formatu, nakon čega aplikacija pomoću pomičnog notnog zapisa i vizualno istaknutih tipki na klaviru prikazuje koje note treba odsvirati. Analizom izvedbe u stvarnom vremenu, aplikacija prati izvedbu korisnika koristeći klavijaturu povezanu s računalom putem MIDI protokola i nudi povratne informacije o točnosti sviranja. Ovaj rad detaljno opisuje razvijenu aplikaciju, uključujući odabrane tehnologije i metode implementacije, te pruža analizu dobivenih rezultata. U radu su također razmotrene mogućnosti za daljnje poboljšanje i nadogradnju aplikacije, s ciljem povećanja njezine funkcionalnosti i prilagodljivosti različitim razinama vještine korisnika.

Ključne riječi: glazba; klavir; midi; musicxml; javafx; desktop aplikacija

Abstract

Application to assist with learning to play the piano

Ana Bagić

In this thesis, I explored the development of an application designed for learning piano playing, specifically aimed at practicing the performance of selected pieces. I developed an application that allows users to load their chosen piece in MusicXML format, after which the application displays the notes to be played using a scrolling sheet music view and visually highlighted piano keys. By analyzing the performance in real-time, the application evaluates the user's playing using a keyboard connected to the computer via MIDI protocol and provides feedback on the accuracy of the performance. This thesis provides a detailed description of the developed application, including the chosen technologies and implementation methods, and offers an analysis of the obtained results. The thesis also considers possibilities for further improvement and enhancement of the application, with the goal of increasing its functionality and adaptability to different skill levels of users.

Keywords: music; piano; midi; musicxml; javafx; desktop application