Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Výuková webová aplikace pro hudebníky

Bc. Matouš Kyncl

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Matouš Kyncl**
Osobní číslo:	**I17208**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Výuková webová aplikace pro hudebníky**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem práce je vytvořit webovou výukovou aplikaci zaměřenou na oblast hudby. Aplikace by měla disponovat možností výuky studenta v oblastech intonace, rytmu, hudební teorie a dalších. V teoretické části práce bude provedena rešerše obdobných výukových aplikací zaměřených na oblast hudby. V rámci rešerše je možné zahrnout i desktopové nebo mobilní aplikace. Dále v teoretické části bude popis vybraných technologií pro realizaci webové aplikace a bude proveden její návrh. V teoretické části bude dále popsána problematika přípravy nebo generování hudebních cvičení. V praktické části bude implementována webová aplikace umožňující výuku v oblastech intonace, rytmu, hudební nauky, aj. Webová aplikace bude umožňovat základní sledování schopností studenta a postupu v jeho procvičování. Pro realizaci webové aplikace budou využity vhodné technologie a databáze pro perzistenci dat.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50-60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

ZAKAS, Nicholas C, Understanding ECMAScript 6: the definitive guide for JavaScript . San Francisco: No Starch Press, 2016. ISBN 9781593277987. CROCKFORD, Douglas, JavaScript the Good Parts. Sebastopol: O'Reilly Media, 2008. ISBN 9780596554873. COOK, Fabian Node.js Essentials. Birmingham, UK: Packt Publishing, 2015. ISBN 9781785285943. AMUTHAN, G. ,Spring MVC Beginner's Guide Packt Publishing, 2014. ISBN 1-78328-488-9.

Vedoucí diplomové práce: **Ing. Roman Diviš**
Katedra softwarových technologií

Datum zadání diplomové práce: **22. října 2018**

Termín odevzdání diplomové práce: **18. května 2019**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 17. listopadu 2018

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 15. 05. 2019

Matouš Kyncl

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat panu inženýrovi Romanu Divišovi za cenné rady při zpracování diplomové práce, které mi poskytoval velice ochotně a rychle. Také bych chtěl poděkovat celé své rodině za pomoc s korekturou diplomové práce.

ANOTACE

Tato práce se zabývá problematikou výuky hudební teorie a hudební nauky. Obsahem této práce je tvorba webové aplikace, jež umožňuje výuku v oblastech hudební nauky (například: stupnic, intervalů, akordů), nebo v oblastech hudební teorie (spoje akordů a jejich funkce). Aplikace není zcela dokončena, ale již v této fázi může posloužit jak studentům hudební teorie a nauky, tak také hudebním nadšencům.

KLÍČOVÁ SLOVA

Hudební teorie, hudební nauka, výuka, webová aplikace.

TITLE

Educational web application for musicians.

ANNOTATION

This work is focused on teaching music theory and music science. The content of this work is to create a web application that allow learning in music science (for example: scales, intervals and chords), or in music theory (chords connection and chords function). Application is not finished yet, but already this phase could serve students, or music enthusiasts.

KEYWORDS

Music theory, music science, teaching, web application

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
BSON	Binary JSON
CSS	Cascading Style Sheets
EA	Enterprise Architect
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JWT	JSON Web Token
NPM	Node Package Manager
SPA	Single Page Application
SVG	Scalable Vector Graphics
ŠVP	Školní vzdělávací program
URL	Uniform Resource Locator
ZUŠ	Základní umělecká škola

ÚVOD

Tato práce se zabývá problematikou výuky hudební teorie s využitím informačních technologií, konkrétně webové aplikace. Cílem této práce je vytvoření webové aplikace, která bude sloužit k načerpání nových znalostí nebo k ověření správnosti znalostí stávajících. Aplikace by do budoucna mohla být přístupná nejen studentům, ale i nadšencům, kteří se zajímají o hudební teorii.

Důvodem k vytvoření této práce byla absence podobné aplikace v oblasti výuky, procvičování, intonace, rytmu a obecně hudební teorie určené konzervatořím. Především neexistuje komplexní aplikace v české lokalizaci. Již existující aplikace, které budou v práci uvedeny, se zaměřují spíše na hudební teorii základních uměleckých škol a problematika „pokročilejší" hudební teorie, spadající do ŠVP konzervatoří, se v nich nevyskytuje. Proto je cílem v této práci (aplikaci), klást největší důraz právě na hudební teorii spadající do ŠVP konzervatoří a pokusit se tuto problematiku zapracovat do aplikace.

V práci budou nejprve představeny podobné projekty zabývající se touto problematikou, jak bylo uvedeno výše, dále jednotlivé technologie, na kterých bude aplikace vystavěna. Jedná se konkrétně o *HTML 5, CSS* – framework *Bootstrap*, programovací jazyk *JavaScript* na straně klienta a na straně serveru interpret jazyka *JavaScript – Node.js*. Na straně klienta bude využita JavaScriptová opensource knihovna *VexFlow*, pro vykreslování a práci s notami a notovou osnovou. Na straně serveru bude poté využita opensource knihovna *Zazate.js*, pro zpracování zadaných not, která je podpůrnou knihovnou pro hudební teorii.

Aplikace bude fungovat dle následujícího scénáře. Uživatel vstoupí do aplikace a bude mít možnost pracovat s aplikací ve dvou režimech. Může se zaregistrovat, nebo pokračovat jako neregistrovaný uživatel (host). V režimu neregistrovaného uživatele bude mít takovýto uživatel možnosti procvičování hudební teorie, částečný přístup do výukové části, ale nebude mít přístup k výukovým kurzům. V režimu registrovaného uživatele bude aplikace nabízet mnohem více možností než jen strohé procvičování. Pro takového uživatele bude plně dostupný modul výuky a také možnost absolvovat jakýkoli výukový kurz.

1 VÝUKOVÉ APLIKACE ZAMĚŘENÉ NA HUDEBNÍ TEORII

V této kapitole budou představeny vybrané aplikace, které jsou nějakým způsobem zaměřené na výuku a procvičování hudební teorie.

1.1 Auralia a Musition

Tyto dva programy tvoří společně ucelený systém k výuce, testování a učení se hudební teorie. Jedná se o webové aplikace provozovaných v cloudu. Tyto programy jsou vyvíjeny společností Rising Software a patří mezi komerční projekty. Celé rozhraní aplikace je lokalizováno v anglickém jazyce a licence obou dvou programů (sady) vyjde cca na 4400 Kč (ke dni 16. 3. 2019) [1].

1.1.1 Auralia

Prvním z dvojice programů je Auralia. Jedná se o program určený k trénování poslechu se 43 tématy, která pokrývají oblasti hudební teorie, jako jsou: akordy, stupnice, ladění, rytmus (různá rytmická cvičení), průběh kadence, harmonie, jazzové postupy, nebo přepis melodie. Jednotlivá cvičení jsou odstupňována dle náročnosti, tím je program vhodný jak pro začátečníky, tak pro pokročilé studenty. Pro učitele a lektory je dostupné navolit si třídu studentů, sledovat jejich pokroky a také je hodnotit. Zároveň je možné, aby si učitel navolil [1], nebo vytvořil své vlastní úkoly a usnadnil si tak práci při přípravě na vyučovací hodiny.

1.1.2 Musition

Musition je program určený k výuce hudební teorie a rytmu. Zahrnuje oblasti jako je například: čtení partitury, rozpoznávání časových značek (doba trvání not a pomlk), rozpoznávání intervalů, orientace v notové osnově s různými notačními klíči a mnoho dalších. Stejně jako Auralia je Musition vhodný [3] jak pro začátečníky, tak pro pokročilé.

1.2 Musictheory.net

Webová aplikace musictheory.net je zdarma dostupná webová aplikace pro kohokoliv s lokalizací v anglickém jazyce. Nabízí 2 moduly – výukový a procvičovací. Výukový modul obsahuje základy hudební nauky (notová osnova, klíče, délka not, tečky, postupy a posuvky atd.), rytmus a metrum, stupnice a předznamenání stupnic, intervaly, akordy plus jejich funkce, postupy a spoje. Procvičovací modul obsahuje okruhy pro procvičování not, zápisu stupnic, akordů a předznamenání do notových osnov, orientaci na klaviatuře, orientaci na kytarovém

hmatníku a poté poslech stupnic, akordů, intervalů s následným zápisem do notové osnovy, nebo určením správné klávesy na klaviatuře.

Pro modul výuky je v nabídce offline aplikace Theory Lessons pro operační systém iOS a pro modul procvičování je k dispozici aplikace Tenuto, taktéž pro iOS. Obě aplikace jsou zpoplatněny [4] a stejně jako celá webová aplikace jsou v anglickém jazyce. Webová aplikace i obě mobilní aplikace jsou stále v provozu a poslední aktualizace proběhla v lednu roku 2019 (k datu 16. 3. 2019).

1.3 uTheory

Webová aplikace určená pro vzdělávání v oblasti hudební teorie. Pro výuku hudební teorie jsou zde k dispozici tutoriály obsahující text a krátká videa, která mají za cíl nastínit a vysvětlit studentovi danou problematiku. Pro procvičení nabytých zkušeností jsou zde k dispozici příklady k jednotlivým tématům ve 3 základních kategoriích: rytmus – rozeznávání rytmu v různých metrech; harmonie a vzdálenosti – rozlišování intervalů; stupnic, akordů atd.; poslech – poslech a rozlišování stupnic, intervalů, akordů a dalších. V obou částech aplikace – jak v části procvičovací, tak v části vyučovací, je sledován pokrok studenta – např. kolik tematických okruhů již navštívil, jak byl úspěšný při procvičování a tyto údaje jsou zobrazeny přehledně v grafech na dashboard stránce.

Pro vyzkoušení s omezenými možnostmi je tato aplikace zdarma (obsahuje pouze 10 výukových videí a nejsou dostupná všechna praktická cvičení). Pro plnou funkcionalitu aplikace je potřeba být registrován [5] a mít zaplacený poplatek. Měsíční předplatné stojí cca 140 Kč, roční pak okolo 910 Kč. Tato aplikace je v anglické lokalizaci.

1.4 Teoria.com

Jedna z dalších webových aplikací, která je zaměřena na hudební teorii, je zdarma v anglické a španělské lokalizaci. Nabízí výuku ve formě tutoriálů a procvičování ve formě cvičných příkladů. Je možné se také registrovat/přihlásit a získat tak možnost uložit si informace o tom, jak úspěšný byl uživatel v dané problematice. Od ostatních výše popsaných aplikací je tato specifická také tím, že obsahuje část, kde se nacházejí články, které nějakým způsobem souvisejí obecně s hudbou – je možno se zde setkat s články na téma muzikoterapie nebo články zaměřené hudebně-filozoficky a velká část příspěvků se týká analýzy děl světových autorů – buď jejich částí, nebo celých děl. Teoria.com také obsahuje komponentu [6], ve které jsou zobrazeny hudební, historické milníky vztahující se k aktuálnímu datu. Webová aplikace je

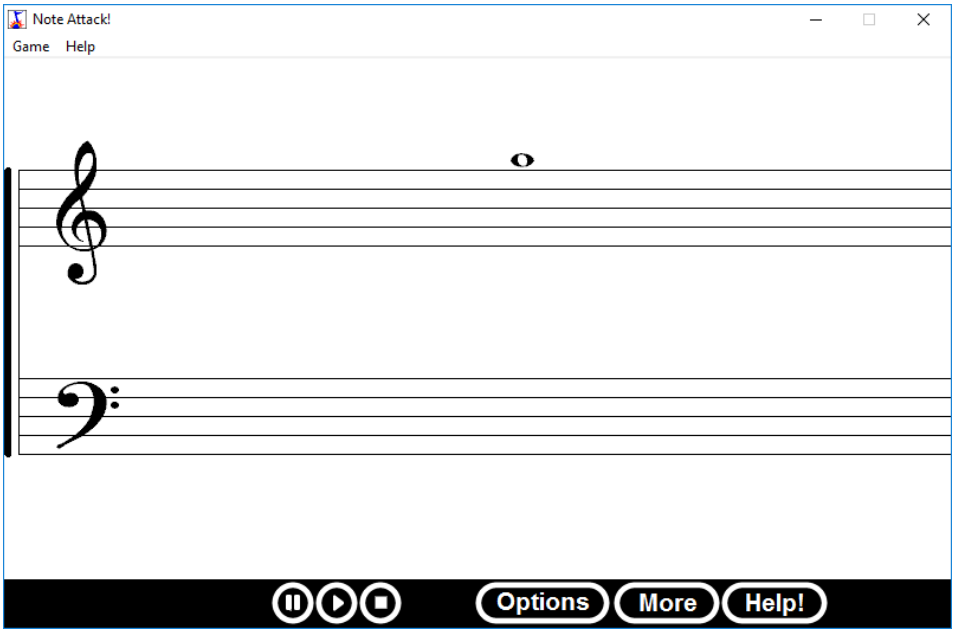
stále aktivní a průběžně aktualizována – poslední revize stránek proběhla 15. 3. 2019 (ke dni 16. 3. 2019).

1.5 Dave Conservatoire

Jedná se o webovou aplikaci, která je zcela zdarma. Vytvořena byla ve Velké Británii a je celá v anglické lokalizaci. Tato aplikace byla dle autora inspirována vzdělávací aplikací *Khan Academy* (aplikace *Khan Academy* sdružuje několik oborů přes přírodní vědy, matematiku a obsahuje také lekce z oblasti hudby). Dostupná je i v české lokalizaci (viz https://cs.khanacademy.org/). Uživatelům je k dispozici přes 300 video tutoriálů, které vysvětlují základní hudebně-teoretické problémy a testy pro ověření znalostí. Pokud chce student znát svůj pokrok ve studiu, tak se může bezplatně zaregistrovat [7] a to např. přes existující účet na Gmailu nebo Facebooku.

1.6 Note Attack

Note Attack je desktopová aplikace, která je zaměřena pouze na rozeznávání not v basovém a houslovém klíči. Jedná se o aplikaci, vysvětlující tuto problematiku pomocí hry. V pravé části okna jsou generovány náhodně noty (včetně zvukové ukázky) a ty se posouvají postupně směrem ke klíčům (doleva). Úkolem uživatele je správně identifikovat notu a zadat její název do té doby než „narazí“ do klíče. Zadávat noty se může pomocí klávesnice PC, nebo pomocí připojené MIDI klaviatury. Aplikace je celá v anglické lokalizaci.



Obrázek 1 - Okno aplikace Note Attack! (zdroj vlastní)

1.7 Aplikace s českou lokalizací

Aplikací s českou lokalizací zabývajících se hudební teorií není mnoho. Většina z těchto aplikací je zaměřena buď na jeden konkrétní problém, anebo je koncipována jako sada testových otázek. Mezi aplikace [8], které se zabývají tématem procvičování not, patří webové aplikace: *Procvičování not* ([9]), *Pugův notační trénink* ([10]) anebo *Online učitel k procvičování not* ([11]). Formou testů je vystavěna aplikace *Testi.cz*, která zahrnuje kromě jiného i testy z oblasti hudby – obsahuje testy týkající se hudebních dějin, stupnic, intervalů atd.

Společnost *euroDIDACT s.r.o.* vyvíjí výukové programy pro mnoho oborů. Jedním z nich je i hudba. Aplikace Hudební nauka umožňuje interaktivní procvičování základů hudební nauky, základů hry na různé nástroje a další oblasti. Aplikace je vydávána pod třemi licencemi a zpoplatněna. Při zaplacení multilicence pro školy, je možnost využít aplikaci v offline režimu [12].

1.8 Mobilní aplikace

V oblasti mobilních aplikací lze říci, že žádná z aktuálně dostupných aplikací není komplexní, protože je většinou zaměřena na jeden specifický problém. Některými z těchto aplikací mohou být: *Musical dictation lite* – zaměřená na hudební diktáty, *Circle of 5ths* – zaměřená na kvartkvintový kruh (vztahy mezi stupnicemi) nebo *Walkband* – zaměřená na teorii hudebních nástrojů (jak daný nástroj zní, jak se např. na kytaře zahraje určitý akord a podobně). Výjimku tvoří aplikace *Perfect Ear*.

1.8.1 Perfect Ear

Původní záměr této aplikace bylo zdokonalit hudební sluch pomocí cvičení (identifikace podle sluchu, zazpívání dle zadání a další). Postupně se ale aplikace začala rozšiřovat a stala se z ní poměrně komplexní aplikace zahrnující trénink intervalů, stupnic, akordů, orientace na kytarovém hmatníku, také hudební diktáty a další. V neposlední řadě disponuje rozsáhlou hudební teorií [13].

Nevýhodou této aplikace se může jevit pouze anglická lokalizace a zpoplatnění některých funkcionalit.

1.8.2 Mobilní aplikace s českou lokalizací

Mobilních aplikací s českou lokalizací se v dnešní době vyskytuje velice málo. Aplikace, které by obsahovaly nějaké prvky interaktivity a umožňovaly uživateli procvičení v různých oblastech, téměř nenajdeme. Spíše jsou to opět aplikace zaměřené na jeden problém, a i tak je

jich velice málo. Jedním z příkladů je aplikace – *Učit se číst hudbu* (https://play.google.com/store/apps/details?id=org.anddev.android.solfa) pro platformu *Android, Windows Phone* a *iOS*. Jedná se o aplikaci určenou pro procvičování not a notového zápisu. Aplikace ovšem není pro platformy iOS a Windows Phone zdarma a její použití je zpoplatněno. Aplikací, která oblast učení a rozpoznávání not rozšiřuje o možnost rozpoznávání not dle poslechu, je aplikace – *Noutee* (https://play.google.com/store/apps/details?id=com.noutee.noutee) pro platformy *Android*, *iOS*, ale i *Windows* a *MacOS* a zároveň funguje i jako webová aplikace. Kromě práce s notami nabízí ještě procvičování v oblasti rytmu nebo v oblasti stupnic.

1.9 Porovnání aplikací

V následující tabulce (Tabulka 1) jsou porovnány aplikace, které byly popsány výše. Jsou porovnávány dle těchto kritérií: *podporované platformy* – zda jsou určeny pro mobilní, či desktopová zařízení, nebo se jedná o webové aplikace; *licence* – zda jsou nějakým způsobem zpoplatněny, nebo zcela zdarma; *hudební nauka* – zda obsahují alespoň jednu oblast, spadající pod hudební nauku a zároveň je možno si tuto oblast v rámci aplikace procvičit; *pokročilejší hudební teorie* – jestli je možnost si procvičit alespoň jednu oblast řadící se do pokročilejší hudební teorie, resp. hudební teorie pro vyšší vzdělání; *lokalizace* – v jakých jazykových mutacích jsou aplikace dostupné.

Tabulka 1 - Porovnání jednotlivých aplikací (zdroj - vlastní)

	Podporované platformy	Licence	Hudební nauka	Pokročilejší hudební teorie	Lokalizace
Auralia , Musition	webová aplikace	zpoplatněno	ano	ano	angličtina, španělština
Musictheory.net	webová aplikace	zdarma	ano	ano	angličtina
uTheory	webová aplikace	zdarma, pro plné využití aplikace zpoplatněno	ano	ano	angličtina
Teoria.com	webová aplikace	zdarma	ano	ano	angličtina, španělština
Dave Conservatoire	webová aplikace	zdarma	ano	ne	angličtina, čeština
Note Attack	Windows	zdarma	ano	ne	angličtina
Procvičování not	webová aplikace	zdarma	ano	ne	čeština
Pugův notační trénink	webová aplikace	zdarma	ano	ne	čeština
Online učitel k procvičování not	webová aplikace	zdarma	ano	ne	čeština
Hudební nauka - euroDIDACT	webová aplikace	zpoplatněno	ano	ne	čeština
Perfect Ear	iOS, Android	částečně zpoplatněno	ano	ne	angličtina
Učit se číst hudbu	Android, iOS, Windows Phone	Android – zdarma, pro Windows Phone a iOS zpoplatněno	ano	ne	čeština
Noutee	Android, iOS, Windows, MacOS a zároveň i jako webová aplikace	zdarma	ano	ne	čeština

Z tabulky lze odvodit, že jedním z nejlepších řešení pro výuku a procvičování v oblasti hudební nauky a teorie se jeví dvě aplikace, které tvoří ucelený systém, Auralia a Musition. Díky širokému spektru oblastí může uživatel získat spoustu znalostí z hudební teorie. Největší nevýhodou se ovšem jeví zpoplatnění celé aplikace (zadarmo je uživateli pouze omezená zkušební verze). Další nevýhodou může být, pro anglicky nemluvícího uživatele, anglická mutace. Druhou aplikací, která v porovnání s ostatními nabízí nejvíce možností především v pokročilé hudební teorii, je aplikace Teoria.com. Její nevýhodou může opět být pouze cizojazyčná mutace, avšak kromě angličtiny je v nabídce také španělština. O pomyslnou třetí pozici by se společně mohly podělit aplikace musictheory.net a uTheory, nabízející oproti předešlým méně kategorií z oblasti pokročilé hudební teorie. U obou těchto aplikací, pak může být opět problémem cizojazyčná mutace.

Aplikace, která je vytvářena v rámci této práce, by měla poskytnout především českou lokalizaci a to zcela zdarma. Cílem není tuto aplikaci úplně dokončit, protože například modul Výuka je předmětem jiné práce. Hlavním záměrem je implementace kategorií, které se řadí především do pokročilé hudební teorie a nevyskytují se v již existujících aplikacích, zvláště pak s českou lokalizací.

2 POUŽITÉ TECHNOLOGIE

2.1 HTML

HTML je standardizovaný značkovací jazyk pro vytváření webových stránek. Pomocí značek popisuje strukturu webových stránek. *HTML* elementy jsou reprezentovány tagy. Pomocí nich lze webové stránky stavět do bloků, jako jsou – hlavička, odstavec, tabulka, zápatí atd. [14]. Aktuální verzí tohoto jazyka je verze *HTML 5*, která je využita i v této práci.

2.2 CSS

CSS je jazyk, popisující styl *HTML* dokumentu. Popisuje, jak by měly být jednotlivé elementy zobrazeny na obrazovce, papíře, nebo dalších médiích. Zjednodušuje práci, protože může najednou kontrolovat velké množství stránek [15]. Pro ještě větší zjednodušení práce existuje v dnešní době velká spousta frameworků, které obsahují velké množství předdefinovaných stylů pro úpravu designu webových stránek. Zároveň jsou součástí těchto frameworků pluginy pro UI, realizované např. pomocí javascriptové knihovny *jQuery*. Mezi nejpoužívanější *CSS* frameworky patří: *Bootstrap, Semantic-UI, Foundation, Materialize, Pure* a další [16].

2.2.1 Bootstrap

Bootstrap je sada nástrojů sloužící k vývoji aplikací pracující s *HTML, CSS* a *JavaScriptem*. Umožňuje rychle vytvořit a navrhnout design celé aplikace pomocí *Sass* proměnných a přísad, responsivního mřížkového systému, rozsáhlých předem sestavených komponent a výkonných pluginů postavených na *jQuery* [17].

2.3 JavaScript

JavaScript je programovací jazyk definovaný ve standardu ECMA-32, nesoucí název *ECMAScript*. Aktuální verze *ECMAScriptu* je verze *ES2015* (uvádí se také *ES6*). Jedná se o skriptovací jazyk, který je určen [18] pro vývoj interaktivních, webových aplikací a v dnešní době je jejich nedílnou součástí.

2.3.1 TypeScript

TypeScript je nadstavbou *JavaScriptu* a vychází ze samotného *JavaScriptu*. *TypeScript* je kompilován jako čistý *JavaScript*. Vývojářům usnadňuje práci díky statické typové kontrole. Typová kontrola není povinná, ale umožňuje například odlišit běžné *JavaScriptové* knihovny od *TypeScriptových* komponent. Podporuje všechny [19] nejnovější funkce, nejnovější verze *ECMAScriptu,* jako jsou asynchronní funkce, dekorátory atd. *TypeScript* využívá framework *Angular*, který byl použit ve vytvořené aplikaci na straně klienta.

2.4 MEAN

MEAN je zkratka pro balíček, sadu nástrojů, zahrnující několik technologií, konkrétně *MongoDB, Express, Angular* a *Node.js*. Může posloužit jako jednoduchý startovní bod k vytvoření plnohodnotné javascriptové aplikace určené pro cloud [20].

2.4.1 MongoDB

MongoDB je škálovatelná a flexibilní open source databáze. Jedná se o *NoSQL* databázi, čili jednotlivá data nejsou uložena v relačních tabulkách, ale v dokumentech. Dokumenty jsou ukládány v datovém formátu *JSON* (ve formátu: klíč-hodnota), resp. *BSON* (v binární podobě), což znamená, že pro každý dokument může mít odlišnou strukturu dat. Model dokumentu je podobný objektu v programovacím jazyce, a proto zjednodušuje práci s daty. *MongoDB* je distribuovaná databáze s vysokou mírou dostupnosti a díky tomu může existovat v clusteru [21].

2.4.2 Express

Express je *JavaScriptový*, minimalistický, webový framework pro *Node.js*, který poskytuje robustní sadu nástrojů pro webové a mobilní aplikace. *Express* poskytuje thin layer („tenkou vrstvu“) pro základní funkcionality webových aplikací, aniž by překrýval funkcionality samotného *Node.js*. Díky velkému množství nástrojů http a middleware [22] je snadné pomocí *Express* rychle vytvořit komplexní API.

2.4.3 Angular

Angular je frontendový framework pro tvorbu webových aplikací. Architektura tohoto frameworku vychází z konceptu komponent, služeb a framework využívá jazyk *TypeScript*. Předchůdcem *Angularu* byl *Angular.js*, který využíval čistý *JavaScript* a jeho architektura je zcela odlišná od stávajícího *Angularu* [23]. Z této informace vyplývá, že *Angular* a *Angular.js* jsou navzájem nekompatibilní.

Architektura je vystavěna na komponentách (component), jejichž základem je programová třída (class) a staví tak na základech objektově orientovaného programování. K této třídě může být poté navázána vlastní *HTML* šablona a *CSS* stylopis pomocí direktiv. Tím lze webovou aplikaci rozdělit na samostatné celky. Pro logickou část aplikace slouží služby (services), a to např. pro komunikaci mezi klientskou aplikací v *Angularu* se serverovým API. Nakonec lze služby [23] a komponenty sloučit do jednotlivých modulů (module).

Pro správu a jednodušší práci na aplikaci v *Angularu* slouží rozhraní pro příkazovou řádku *Angular CLI* [23].

2.4.4 Node.js

Node.js je prostředí umožňující spouštět *JavaScript* mimo webový prohlížeč (jedná se tedy o interpret jazyka *JavaScript*). Je postaven na *Chrome V8 JavaScript enginu*, což je engine, využívající webový prohlížeč *Google Chrome*. Primární použití tohoto prostředí je tvorba serverové části webových aplikací. Na rozdíl od jazyka *PHP*, který je také určen pro tvorbu serverové části, je *Node.js* vysoce škálovatelný a schopný obsloužit množství klientů. Pro svoji výkonnost a uvedené vlastnosti je velmi oblíbený pro tvorbu tzv. API serverů [24] pro klientské single page aplikace v *JavaScriptu*. *Node.js* využívá *NPM* (správce JavaScriptových balíčků) a v následujících podkapitolách budou stručně popsány některé z využitých balíčků pro tuto aplikaci.

2.4.4.1 Mongoose

Tento balíček slouží k modelování *MongoDB* objektů pro *Node.js*., resp. slouží ke komunikaci s *MongoDB*. Umožňuje modelování dat z aplikace postavené na schématech. Zároveň usnadňuje práci při sestavování dotazů do databáze [25], přetypovávání a další.

2.4.4.2 Bcrypt

Jedná se o balíček, jež pomáhá při hašování hesel. Samotný název vychází z hašovací funkce *Bcrypt*. Tato funkce je postavena na hašovacím algoritmu *Blowfish*. Do procesu hašování vstupuje samotné heslo, hodnota salt (tzv. kryptografická sůl = několik náhodných bitů, sloužících jako vstup do jednosměrné funkce) a počet iterací – tato hodnota udává počet iterací, kolikrát bude heslo hašováno, čímž zvyšuje bezpečnost, protože zvyšuje nároky na hardware k prolomení hesla (počet iterací ovlivňuje dobu hašování, resp. prolomení hesla). V současnosti se jedná o jeden z nejbezpečnějších způsobů hašování hesel [26].

2.4.4.3 Passport

Balíček *Passport* je autentizační middleware pro *Node.js*. Usnadňuje práci při autentizaci uživatele na základě jeho hesla a uživatelského jména. Kromě jiného podporuje také strategie pro autentizaci účtů ze sociálních sítí *Facebook, Twitter* nebo webu *Google* [27].

2.4.4.4 Jsonwebtoken

Jedná se o balíček, který umožňuje práci s *JWT*. Poskytuje funkce ke generování, dekódování a ověřování *JWT*. *JWT* je otevřený standard, sloužící k bezpečné komunikaci mezi dvěma

stranami. Balíček *Jsonwebtoken* ve spojení s *Passport* balíčkem jsou poté hlavními autentizačními nástroji, při přístupu do aplikace [28].

2.5 Angular Material

Angular Material je framework, využívající *Material Design* (vizualizační jazyk) pro komponenty *Angularu*, pro který je optimalizován. Funguje napříč mnoha platformami, je rychlý a plně kompatibilní s moderními prohlížeči. Zároveň umožňuje jednotlivé komponenty nastylovat dle vlastního uvážení a potřeb [29].

2.6 VexFlow

VexFlow je open-source, online, hudební, notační vykreslovací API. Je kompletně napsáno v *JavaScriptu* a běží na straně klienta (v prohlížeči). *VexFlow* podporuje *HTML5* a jeho elementy canvas a SVG [30]. Mezi základní a hlavní funkce tohoto API patří vykreslování notových osnov, not, notačních značek, posuvek, notových klíčů a jejich stylování.

2.7 Zazate.js

Zazate.js je hudebně-teoretický a notační balíček, nebo modul pro *JavaScript* sloužící programátorům, muzikantům nebo i vědcům ke zkoumání hudby. Nabízí velké množství funkcí, souvisejících s určováním intervalů, akordů, not nebo také stupnic. Například lze vygenerovat intervaly, akordy (kvintakordy, septakordy) různých druhů. Rozpoznat zadané intervaly, akordy nebo i stupnice [31] a také rozpoznat harmonické funkce jednotlivých akordů.

3 HUDEBNÍ TEORIE A JEDNOTLIVÉ OBLASTI ZAMĚŘENÍ TÉTO PRÁCE

Dle definic odborníků je hudební teorie soubor teoretického vědění o hudbě. Dělí se na několik složek, například: nauka o harmonii, nauka o kontrapunktu a nauka o hudebních formách. Dále sem může spadat nauka o rytmu, nauka o melodii, nauka o zvuku. Definic existuje nespočet, a proto tento pojem může mít spoustu významů. Často dochází k záměně pojmů hudební nauka a hudební teorie [32]. Pro účel této práce bude hudební nauka součástí hudební teorie a jednotlivé oblasti zaměření budou částečně vysvětleny, nastíněny v následující kapitole.

3.1 Hudební nauka

V této práci je hudební nauka chápána jako základní poznání nebo vědění z hudební teorie [32]. Patří sem především oblasti spadající pod kategorii ZUŠ. Pojem hudební nauka může mít různé definice. Zároveň je hudební nauka také vyučovacím předmětem na ZUŠ (v názvech se může lišit) a znalosti načerpané v tomto předmětu jsou prostředkem k pochopení pokročilejších problémů, které spadají především do oblasti harmonie.

3.1.1 Noty

Jednotlivé tóny (= zvuky) zapisujeme notami do notové osnovy. Notová osnova je tvořena pěti linkami, čtyřmi mezerami a pomocnými linkami nad a pod notovou osnovou. Umístění noty do notové osnovy udává její jméno. Zároveň je na začátku každé notové osnovy klíč, udávající, jak je daná nota vysoko [33]. Výšku noty můžou změnit posuvky, kterých rozlišujeme pět („𝄫“ - *dvojité bé* snižuje notu o dva půltóny, „♭“ - *bé* – snižuje notu o jeden půltón, „♮“ – *odrážka* ruší předchozí předznamenání, nebo posuvku, „♯“ - *křížek* zvyšuje notu o jeden půltón, „𝄪“ - *dvojkřížek* zvyšuje notu o 2 půltóny). Tvar not udává délku (= trvání) jednotlivých not.

V této práci je jedním z výukových modulů také modul Procvičování not. Tento modul slouží k trénování pojmenovávání a zápisu not. Jakým způsobem vše funguje, bude vysvětleno v kapitole 4.

3.1.2 Intervaly

Interval v hudbě znamená vzdálenost mezi dvěma tóny. Například vzdálenost mezi tóny *c-g* se nazývá čistá kvinta. Pojmenování intervalu se sestává vždy ze dvou slov. Jedno slovo vyjadřuje číslovku, vyjadřující relativní vzdálenost mezi dvěma notami a druhé slovo je přídavné jméno, které tuto vzdálenost upřesňuje. Pro označení intervalů se používají italské číslovky (pro intervaly v jedné oktávě): *prima (1), sekunda (2), tercie (3), kvarta (4), kvinta (5), sexta (6),*

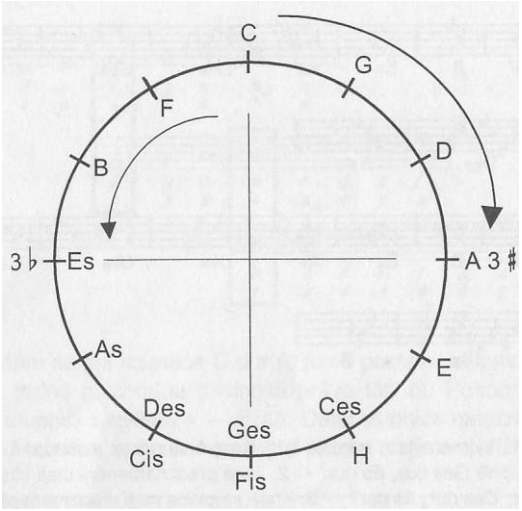
septima (7), oktáva (8). Dále pak jsou pro upřesnění rozlišovány intervaly základní a ze základních pak odvozené. Základní intervaly mohou být *čisté* a *velké*. Čisté intervaly mohou být zvětšené, zmenšené, dvojzvětšené nebo dvojzmenšené a velké mohou být ještě navíc malé (jinak stejné jako u čistých intervalů). Dle směru („od shora dolů“, „od spodu nahoru“) pojmenování se rozlišují intervaly vzestupné a sestupné. Pokud interval patří do dané stupnice (o stupnicích v následující podkapitole), pak je nazýván jako *doškálný* (=„patří do škály“), v opačném případě *nedoškálný*.

V této práci je oblast problematiky intervalů sdružena do modulu Intervalů.

3.1.3 Stupnice

Stupnice je řada stoupajících, nebo klesajících tónů jdoucích za sebou. Existuje celá řada stupnic, mohou být pětistupňové, šestistupňové, nebo sedmistupňové. Rozlišují se stupnice diatonické (obsahují půltóny, ale i celé tóny), nebo celotónové (obsahují pouze celé tóny). Zároveň je možnost dělit stupnice na durové, mollové, cikánské, staré (nebo také církevní) a další. Stupnici je možné vytvořit od kteréhokoli tónu, užívají se však stupnice s tóny nejvýše dvakrát sníženými, nebo dvakrát zvýšenými. V takovém případě je nutné k jednotlivým notám doplnit posuvky, aby daná stupnice odpovídala pravidlům, resp. vzdálenostem mezi jednotlivými tóny např. v durových, nebo mollových tóninách. Mluví se o tzv. předznamenání, které nám určuje specifickou tóninu. K určení správného předznamenání k dané stupnici se používá kvart-kvintový kruh, kdy stupnice s křížky postupují od základního tónu po kvintách směrem nahoru a stupnice s béčky po kvartách směrem dolů. Pro durové stupnice je základním tónem tón C a pro mollové stupnice je základním tónem tón A. Úzce souvisejícím termínem se stupnicí je tónina [35]. Tónina je vlastně neuspořádaná stupnice, čili jednotlivé tóny nejsou seřazeny podle výšek za sebou, ale objevují se ve volném pořadí.

V této práci bude důraz zaměřen na stupnice durové, mollové a jejich příbuzné stupnice, jimiž jsou stupnice mollová harmonická a mollová melodická. Tyto dvě stupnice se vyznačují oproti mollové pravidly takovými, že mollová harmonická stupnice má zvýšený sedmý stupeň a melodická mollová stupnice má zvýšený šestý a sedmý stupeň.



Obrázek 2 – Kvart-kvintový kruh (zdroj [35])

Pro účely této práce je pozornost zaměřena především na stupnice, resp. tóniny durové a mollové.

3.1.4 Akordy

Akord je dle definic souzvuk nejméně tří tónů. Podle počtu tónů lze akordy dělit na: trojzvuky, čtyřzvuky, pětizvuky a vícezvuky. Nejběžnější akordy jsou sestaveny z tercií a jejich název vychází ze vzdálenosti nejnižšího a nejvyššího tónu. Dělí se na: kvintakordy, septakordy, nebo také nónové akordy a další [36]. Dále lze obecně akordy dělit dle vzdálenosti (intervalu) krajních tónů na zmenšené, zvětšené, durové, mollové atd. a také je možné přeuspořádat jednotlivé tóny akordu a vznikne tak obrat akordu.

V této práci se bude klást důraz především na trojzvuky a čtyřzvuky, a to na kvintakordy a septakordy. Na jejich správný notový zápis a opačně, na rozpoznávání a správné pojmenování dle notového zápisu.

3.1.5 Rytmus

Slovo rytmus se v hudbě i mimo hudbu užívá v různých významech. Je základní složkou hudby. Hudební skladby jsou členěny na úseky, zvané takty. Takty jsou dále členěny na doby. V hudební nauce je rytmus popsán jako střídání dob různých délek. Takt se označuje zlomkem, v němž je v čitateli uveden počet dob a ve jmenovateli hodnota jednotlivých dob v notách. Např.: *2/4, 3/4, apod.* To znamená, že *3/4* je označení pro „tříčtvrťový“ takt, který obsahuje 3 doby čtvrťových not. S pojmem rytmus souvisí také pojem *metrum* = střídání dob přízvučných a nepřízvučných [37].

V této práci bude zaměřena pozornost především na správné určení taktového označení na základě vygenerovaného rytmického motivu.

3.2 Harmonie

„Harmonie je složka hudby, která se projevuje souzněním tónů (= akordy, souzvuky) a časovým sledem souzvuků (= harmonická věta)“. Lze říci, že harmonie využívá základní znalosti z hudební nauky k popsání složitějších jevů a pravidel harmonie. Popisuje například: vztahy mezi jednotlivými tóny, akordy, intervaly a jejich funkce v rámci nějakého celku. Z historického hlediska se může harmonie dělit na klasickou (období baroka, klasicismu) a moderní (20. století) [38]. V této práci je důraz kladen především na harmonii klasickou, protože z ní čerpá i harmonie moderní.

3.2.1 Funkce v harmonické větě

Jeden akord může být v rámci jedné tóniny jednou funkcí a v rámci druhé funkcí jinou. *„Ucelený a plynulý sled akordů uspořádaný tak, aby dával smysl jako uzavřený hudební celek, nazývejme harmonickou větou“.* Zpravidla harmonická věta začíná a končí tónikou, což je funkce prvního stupně v tónině. Nejzákladnějšími funkcemi v rámci harmonické věty jsou: výše zmíněná tónika, subdominanta = čtvrtý stupeň a dominanta = pátý stupeň. *Kadence* je nejzákladnějším možným průběhem harmonické věty v pořadí – tónika, subdominanta, dominanta, tónika. Všechny ostatní funkce, které se v rámci tóniny objevují – druhý stupeň, třetí stupeň, šestý stupeň a sedmý stupeň, jsou zástupci hlavních funkcí.

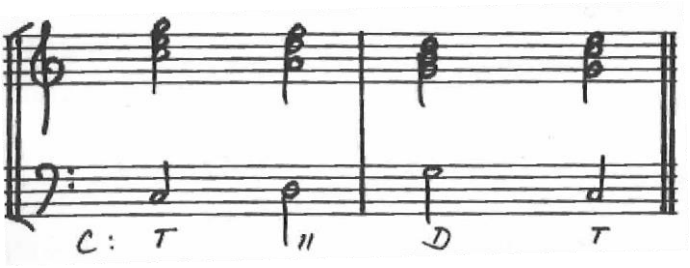
V této práci bude kladen důraz na správné rozeznání funkce v rámci tóniny a také na správný zápis zadané funkce do notové osnovy.

3.2.2 Spoje akordů

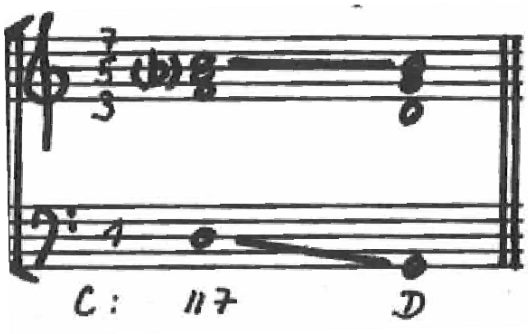
Spojování jednotlivých akordů, ať už např. kvintakordů nebo septakordů vychází z několika pravidel klasické harmonie. Ve čtyřhlasé sazbě, na kterou se zaměřuje i tato práce platí následující pravidla, mezi něž můžou patřit: zakázaný rovnoběžný pohyb v intervalech kvinty a oktávy, resp. primy mezi jednotlivými hlasy = *paralelní kvinty a oktávy, postup v paralelních sekundách a septimách, postup v antiparalelních kvintách a oktávách* nebo postup v tzv. *skrytých oktávách,* dále pak zákaz zdvojování citlivého tónu, zákaz postupu v jednom hlasu o zvětšenou sekundu. Každý z postupů má kromě obecně zákazových pravidel také svá specifická pravidla (některá popsána dále v této kapitole).

Pro přísný spoj, který se používá mezi příbuznými akordy = akordy, jež mají společný tón, platí, že společné tóny se zadrží, bas je veden po základních tónech akordů a zbývající hlasy jsou vedeny nejkratším směrem do cílového akordu. Pro nepříbuzné akordy = akordy, které nemají jediný společný tón, se používá tzv. „pravidlo protipohybu“, kdy je bas veden v základních tónech obou akordů [39], a vrchní hlasy jsou vedeny protipohybem k basu. Specifickými vybranými spoji mezi akordy jsou spoj druhého stupně a dominanty v mollové, popř. harmonické tónině, spoj dominanty a šestého stupně = „klamný spoj“, spoj sedmého stupně a prvního stupně (dále již tóniky). Mezi další vybrané spoje byly zařazeny spoj tóniky a subdominanty a spoj tóniky a dominanty.

Při spoji druhého stupně a dominanty v mollové tónině dochází k problému postupu v některém z hlasů o zvětšenou sekundu neboli „hiatus“. Tomuto problému se musí předejít tak [39], že všechny hlasy jsou vedeny směrem dolů. Jedná se tedy o tzv. volný spoj.

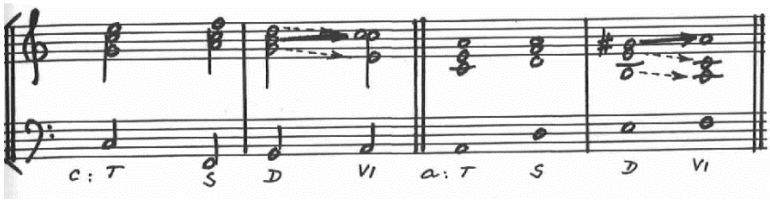


Obrázek 3 - Spoj druhého stupně s dominantou (zdroj [39])

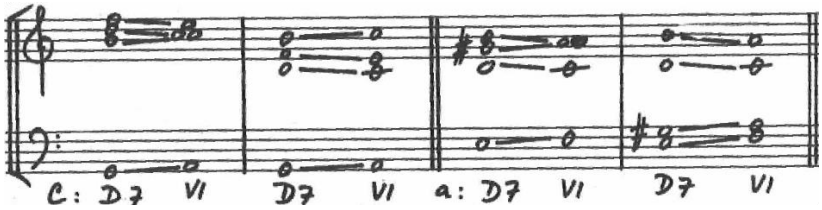


Obrázek 4 - Spoj septakordu druhého stupně s dominantou (zdroj [40])

Pro „klamný spoj“ platí, že citlivý tón dominanty musí být rozveden směrem nahoru, do následujícího akordu a zbylé hlasy musí postupovat protipohybem = přísné vedení hlasů. V tomto případě dojde ke zdvojení tercie šestého stupně. Pokud se citlivý tón nenachází v sopránu (nejvyšší hlas), nebo v basu (nejspodnější hlas), lze rozvod citlivého hlasu zastoupit jiným hlasem a ostatní hlasy mohou být vedeny volně = volné vedení hlasů [41].

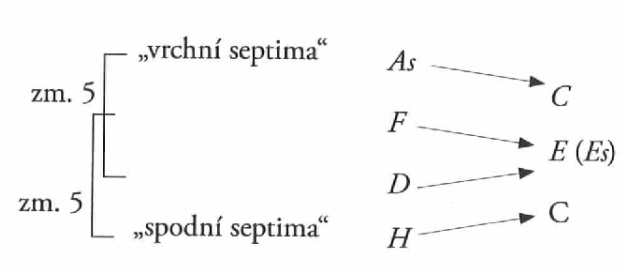


Obrázek 5 - Spoj dominanty s šestým stupněm (zdroj [41])



Obrázek 6 - Spoj dominantního septakordu s šestým stupněm (zdroj [41])

Pro spoj sedmého stupně a tóniky ať v mollové nebo durové tónině je užíváno tzv. kombinovaného rozvodu. Jedná se o přísný rozvod septakordu. To znamená, že prima a tercie septakordu sedmého stupně se rozvádějí směrem nahoru do tóniky (neboli do primy a tercie tóniky) a kvinta a septima se rozvádějí směrem dolů (čili do tercie a kvinty tóniky). Z toho vyplývá, že při tomto spoji dochází ke zdvojení tercie tóniky [42].



Obrázek 7 - Kombinovaný rozvod 7. stupně s tónikou (zdroj [42])

Pro spoje kvintakordů tóniky se subdominantou a tóniky s dominantou platí pravidla pro rozvod příbuzných akordů vysvětlená výše. Pokud se jedná o septakordy, tak poté pro spoj dominanty s tónikou platí, že vrchní septima je rozvedena směrem dolů, kvinta je volně vedena směrem dolů do tóniky (může být i nahoru, ale dochází ke zdvojení tercie), tercie (neboli citlivý tón v tónině) je vedena směrem nahoru do primy, resp. oktávy a základní tón septakordu, pokud je v basu, postupuje skokem na základní tón tóniky, nebo ho lze zadržet [43]. V případě spoje tónického septakordu se subdominantou platí opět, že vrchní septima je vedena směrem dolů, tercie s kvintou jsou rozvedeny směrem dolů do tónů subdominanty a základní tón postupuje skokově do základního tónu subdominanty v případě, že se nachází v base, nebo je zadržen, protože se jedná o společný tón tóniky a subdominanty.

V této práci bude pozornost soustředěna právě na výše uvedené spoje. Jak na jejich rozeznávání, tak na správný zápis vybraného spoje a jeho správný rozvod.

4 APLIKACE

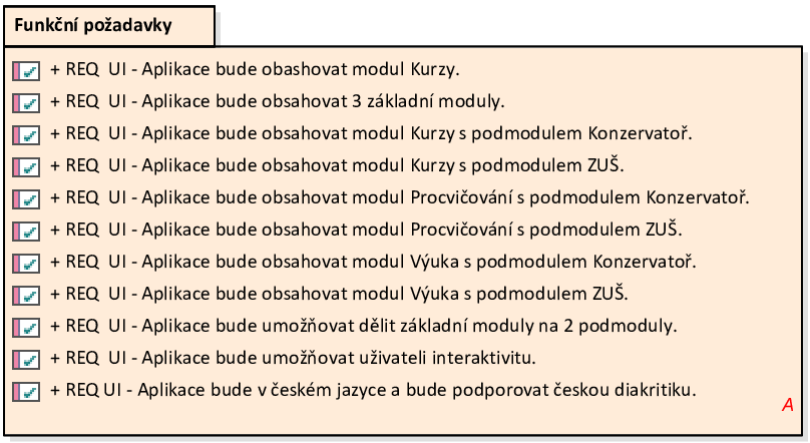
4.1 Návrh a model aplikace

Celá aplikace byla navržena pomocí programu *Enterprise Architect (EA)* - nástroj pro analýzu a návrh systému, reflektující jednotlivé etapy vývoje.

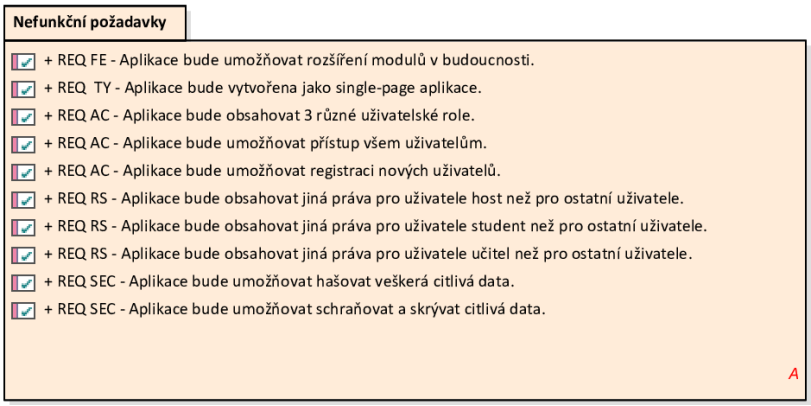
Celý návrh v EA odpovídá jednotlivým etapám vývoje. Nejprve byly sepsány požadavky, co by měla aplikace umožňovat danému uživateli s konkrétní rolí. Poté byly k těmto požadavkům vytvořeny případy užití a nakonec model analytických tříd.

4.1.1 Požadavky

Požadavky byly seskupeny do dvou předepsaných kategorií, které se běžně používají při vývoji – funkční a nefunkční. Funkční byly dále rozděleny na požadavky spadající do kategorie Rozhraní. Nefunkční pak na kategorie: Přístup (AC), Bezpečnost (SEC), Práva (RS), Rozšiřitelnost (FE) a Technologie (TY).



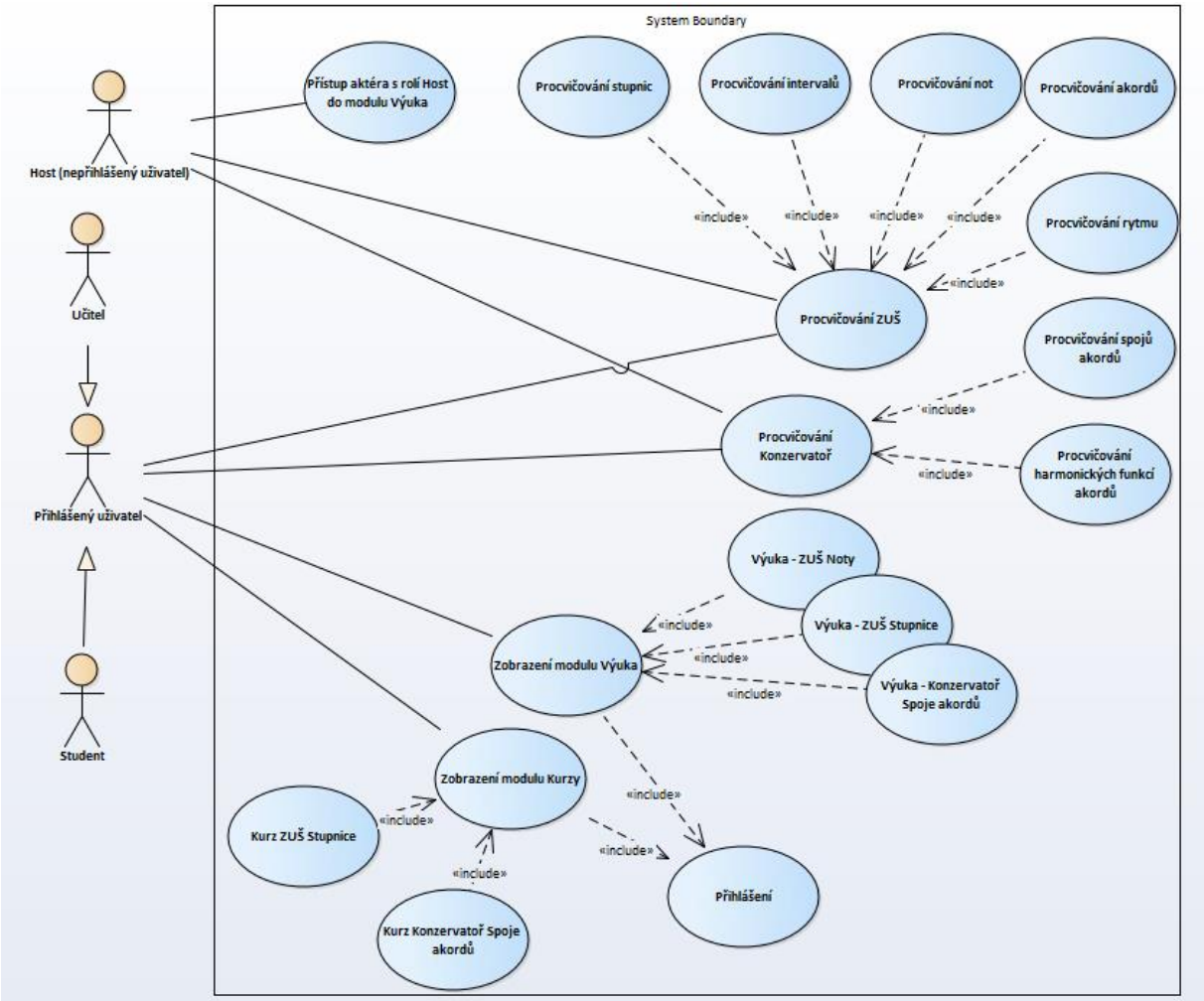
Obrázek 8 - Funkční požadavky (zdroj – vlastní)



Obrázek 9 - Nefunkční požadavky (zdroj – vlastní)

4.1.2 Případy užití

Na základě stanovených požadavků byly vytvořeny případy užití a také aktéři. Neboli uživatelé, kteří budou přistupovat do aplikace, resp. jejich role. Jednotlivé případy užití mají scénáře, podle nichž by měla aplikace fungovat. Některé případy užití (z důvodu vysokého počtu, ne všechny) byly zobrazeny na souhrnném diagramu, kde lze rozeznat, na co má daný uživatel právo.



Obrázek 10 - Případy užití (zdroj - vlastní)

4.1.3 Model analytických tříd

Návrh tříd byl přizpůsoben k možnostem jazyka *TypeScript* a frameworku *Angular*. Hlavním modulem (v rámci *Angularu*) je modul AppModule. Je kořenovým modulem celé aplikace. Sdružuje nezbytné komponenty, služby a ostatní moduly. V tomto modulu se nachází komponenta (= dekorátor, který třídě přiřazuje html a css šablonu) AppComponent, což je komponenta, která se objeví při spuštění aplikace (používá ji výchozí stránka index.html). Proto

je v návrhu tato komponenta (dále již třída) na vrcholu pomyslného stromu tříd (pojmenovaná jako App bez označení Component). Tato třída zároveň zahrnuje ještě Home třídu, jež je třídou pro zobrazení úvodní stránky. Taktéž třída Navigation je součástí třídy App a zobrazuje základní navigační menu, přístup k přihlašovací a registrační třídě (třídy Login a Registration). Třída App poté předává veškeré řízení třídám View, Learn a Courses a tyto třídy rozhodují o tom, jaké ostatní komponenty, služby, a jak se budou na základě preferencí uživatele zobrazovat.

4.1.3.1 View

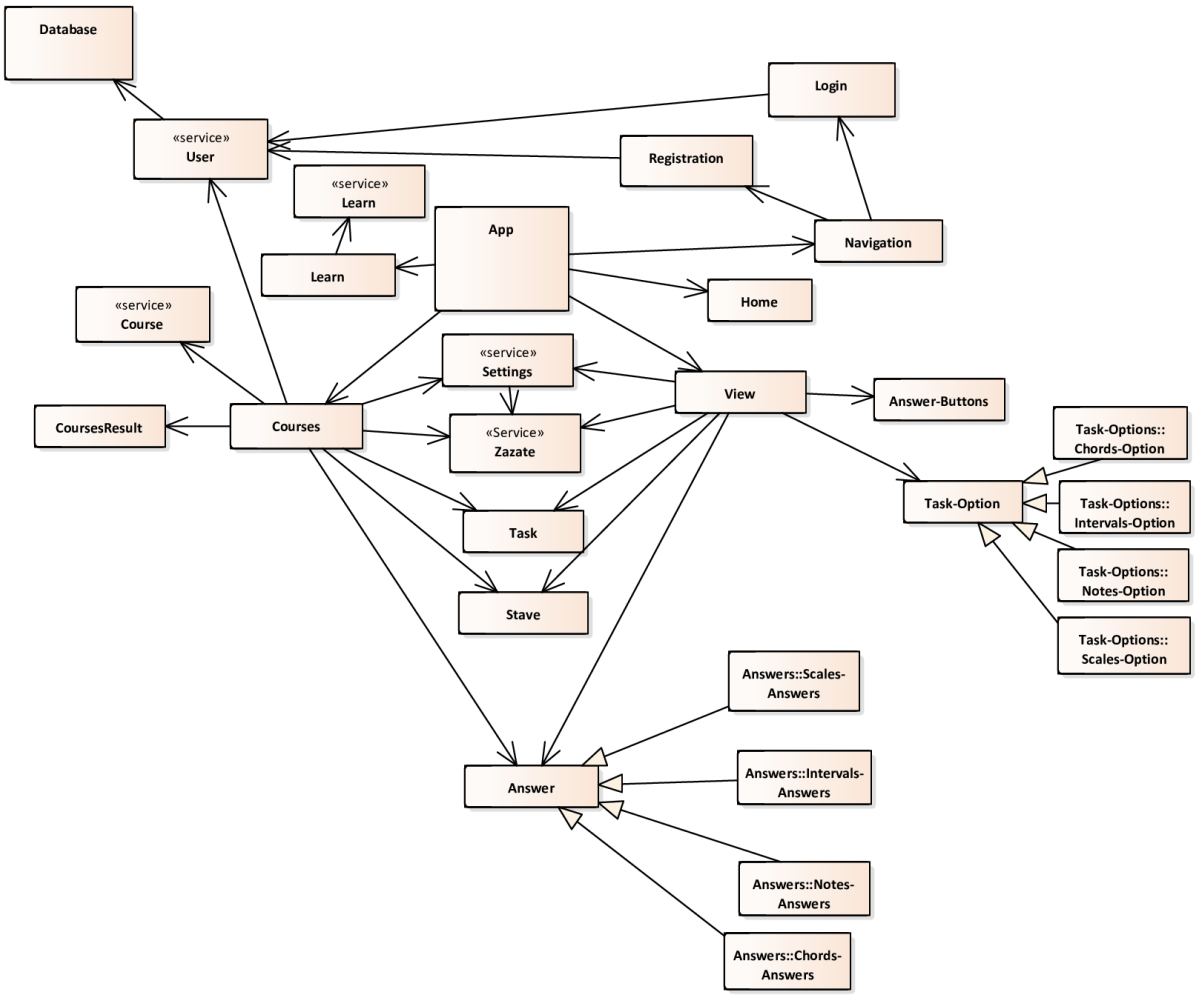
Třída *View* slouží primárně pro zobrazování komponent v rámci modulu Procvičování. Mezi komponenty, se kterými pracuje třída *View*, patří např.: *Task* (generování otázek příkladů), *Answer* (komponenta pro generování dalších komponent pro zodpovězení otázky), *Answer-Buttons* (komponenta obsluhy příkladů – odstartování procvičování, kontrola aktuálního příkladu a generování dalšího příkladu), *Stave* (komponenta pro vykreslování notových osnov a interakci s nimi), *Task-Option* (komponenta pro přednastavení příkladu). A pro některé z těchto komponent jsou poté definovány specifičtější potomci, kteří zahrnují požadavky dané kategorie (opět z důvodu velkého počtu tříd jsou zobrazeny jen některé). Důležitými asociacemi třídy *View* jsou asociace se třídou *ZazateService* a *SettingsService*. Jedná se o služby, které jsou logickou částí aplikace v *Angularu* a v této aplikaci slouží především k udržování nastavení daných příkladů a jejich generování (bude popsáno v následujících kapitolách).

4.1.3.2 Courses

Třída *Courses* obsluhuje modul Kurzy. Využívá již existujících komponent, stejných jako třída *View* a díky těmto komponentám je poté vygenerován kurz. Zároveň využívá službu *CourseService*, která uchovává jednotlivé odpovědi v rámci právě probíhajícího kurzu a na konci kurzu se tyto výsledky zobrazí uživateli díky komponentě *CourseResult*. Třída *Courses* má také asociaci na službu *UserService* a tato služba slouží k připojení do databáze, resp. k přihlášení a registraci uživatele.

4.1.3.3 Learn

Třída *Learn* je třída sloužící k vykreslení obsahu pro modul Výuka. Tato třída je pouze návrhem pro další zpracování v budoucnosti.



Obrázek 11 - Model analytických tříd (zdroj - vlastní)

4.2 Důležité komponenty a služby

V této kapitole budou vybrány některé stěžejní komponenty a služby, obsahující hlavní logiku aplikace, důležité části a ty, s kterými se velmi často pracuje.

4.2.1 Komponenta Stave

Komponenta Stave je jednou z hlavních komponent, která se objevuje v rámci celé aplikace. Každé cvičení, jež je v rámci této aplikace k dispozici, pracuje s komponentou Stave. Tato komponenta se stará o vykreslení notové osnovy. Zároveň naslouchá událostem, jako jsou: pohyb myši nebo stisknutí tlačítka myši. Tím uživateli umožňuje vykreslovat noty po stisknutí a zobrazovat noty při pohybu myši dle dané pozice. V případě neinteraktivních úkolů dokáže tyto události odfiltrovat a pouze vykreslit zadaný příklad. Také slouží k vykreslení pomocných

tlačítek, které slouží k přidání posuvek k notám. Základní vykreslení notové osnovy viz zdrojový kód metody *paintStave()*.

```
paintStave() {
  // Create an SVG renderer and attach it to the DIV element named
     'divStave'.

  const div = this.divStave.nativeElement;
  const renderer = new VF.Flow.Renderer(div,
    VF.Flow.Renderer.Backends.SVG);

  // Prepare width stave by type of task
  let width = 200;

  // for category scales
  if (this.category === CATEGORIES[5].id) {
    width = 500;
  } else if (this.category === CATEGORIES[6].id) {
    // for category rhythm
    width = 640;
  }
  div.style.width = width;

  // Size our svg:
  renderer.resize(width + 21, HEIGHT);

  // And get a drawing context:
  this.context = renderer.getContext();
  this.context.svg.style.pointerEvents = 'bounding-box';

  // Create a stave at position X, Y of width width on the canvas.
  this.staves[0] = new VF.Flow.Stave(X, Y, width);

  // Add a clef.
  let clefType = CLEFS[0].clef;
  if (this.clef !== UNDEFINED && this.clef) {
    clefType = this.clef;
  }
  this.staves[0].addClef(clefType);

  // Connect it to the rendering context and draw!
  this.staves[0].setContext(this.context).draw();
}
```

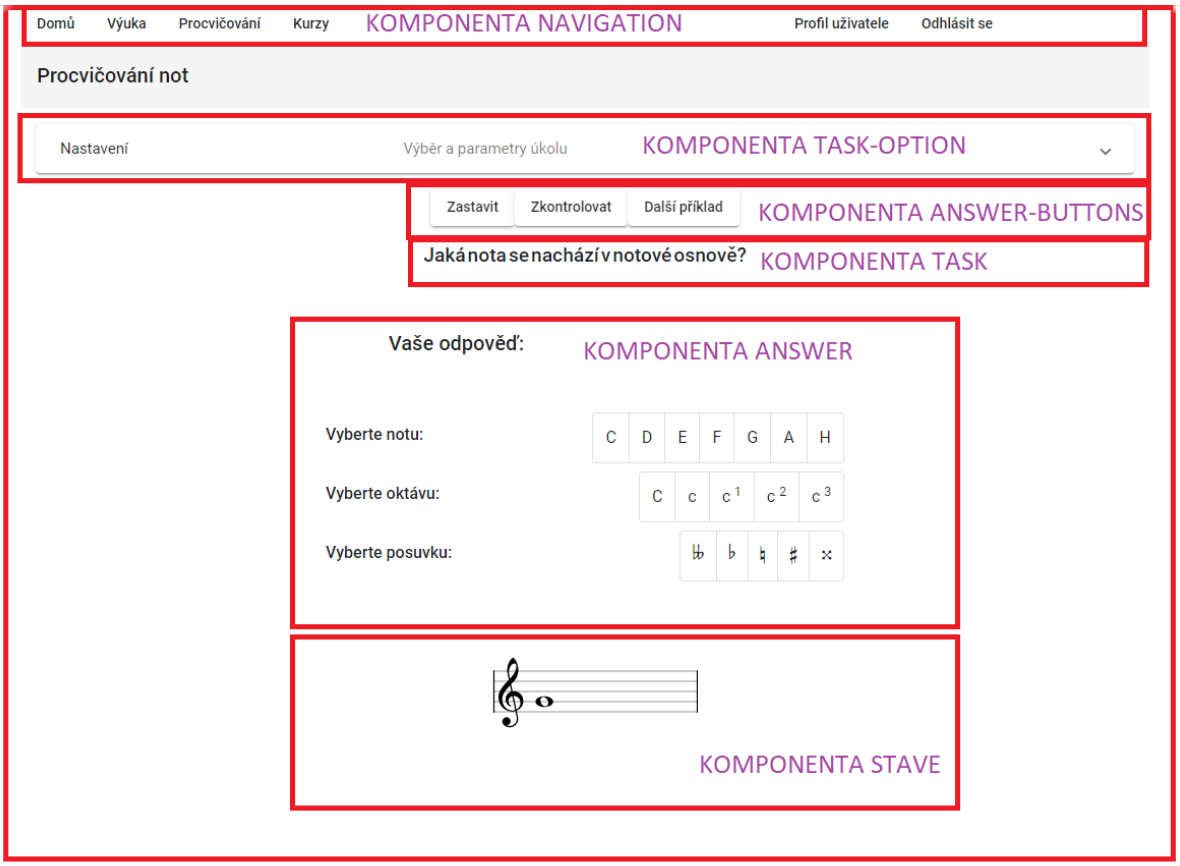
4.2.2 Komponenta View

Komponenta View je základní třídou pro modul Procvičování. Využívá několik dalších komponent pro vykreslení výsledného obsahu. Po výběru uživatele kterékoli kategorie z procvičovacího modulu, je řízení předáno této komponentě. Zde je na základě adresy URL zjištěno, jakou kategorii uživatel zvolil. K dané kategorii je poté přidán příslušný titulek stránky a k vykreslení obsahu jsou volány komponenty *Info, Task-Option, Task, Answer-Buttons, Answer a Stave*. Tyto komponenty (kromě komponent *Stave, Answer-Buttons* a *Info*) se poté rozhodnou, jaký konkrétní potomek se bude starat o vykreslení výsledného obsahu. Rozhoduje

se dle vstupního parametru, kterým je URL. Poskládáním všech těchto komponent dohromady, poté vznikne samotný úkol pro danou kategorii spadající pod modul Procvičování.

```
<mat-toolbar>
  {{title}}
</mat-toolbar>
<div class="text-center col-sm my-2">
  <div class="text-right">
    <app-info [category]="url"></app-info>
  </div>
  <app-task-option
      *ngIf="showTaskOption()"
      [parent]="url"
      [clef]="settingsService.getClef()"
      (taskTypeChange)="settingsService.setTaskType($event)"
      (typeChordChange)="settingsService.setTypeChord($event)"
      (keyTypeChange)="settingsService.setKeyType($event)"
      [isStarted]="settingsService.getIsStarted()"
      (clefChange)="settingsService.setClef($event)"
      [isPossibleMoreStaves]=
            "settingsService.getIsPossibleMoreStaves()">
  </app-task-option>
  <app-answer-buttons
      *ngIf="settingsService.getShowAnswerButtons()"
      (startHandler)="settingsService.setIsStarted($event)"
      (checkHandler)="this.zazateService.checkAnswer()"
      (nextHandler)="this.nextHandler()">
  </app-answer-buttons>
  <app-task *ngIf="settingsService.getIsStarted()"
            [parent]="url"
            [generatedTask]="zazateService.getGeneratedTask()"
            [isInteractive]="settingsService.getIsInteractive()"
            [category]="category"
            (taskChangeEvent)="zazateService.setTaskText($event)">
  </app-task>
  <app-answer [parent]="url"
              [isInteractive]="settingsService.getIsInteractive()"
              [isRightAnswer]="zazateService.getIsRightAnswer()"
              [isStarted]="settingsService.getIsStarted()"
              [typeChord]="settingsService.getTypeChord()"
              (answerHandler)="this.zazateService.setAnswer($event)"
              [isNext]="isNext">
  </app-answer>
  <app-stave
      *ngIf="settingsService.getTaskType()"
      [taskType]="settingsService.getTaskType()"
      [clef]="settingsService.getClef()"
      [isStarted]="settingsService.getIsStarted()"
      [category]="category"
      [generatedTask]="zazateService.getGeneratedTask()"
      [isInteractive]="settingsService.getIsInteractive()"
      (answerHandler)="zazateService.setAnswer($event)"
      [isPossibleMoreStaves]=
            "settingsService.getIsPossibleMoreStaves()">
  </app-stave>
</div>
```

(Na podobném principu jako komponenta *View* funguje i hlavní komponenta modulu Kurzů a to komponenta *Course*.)

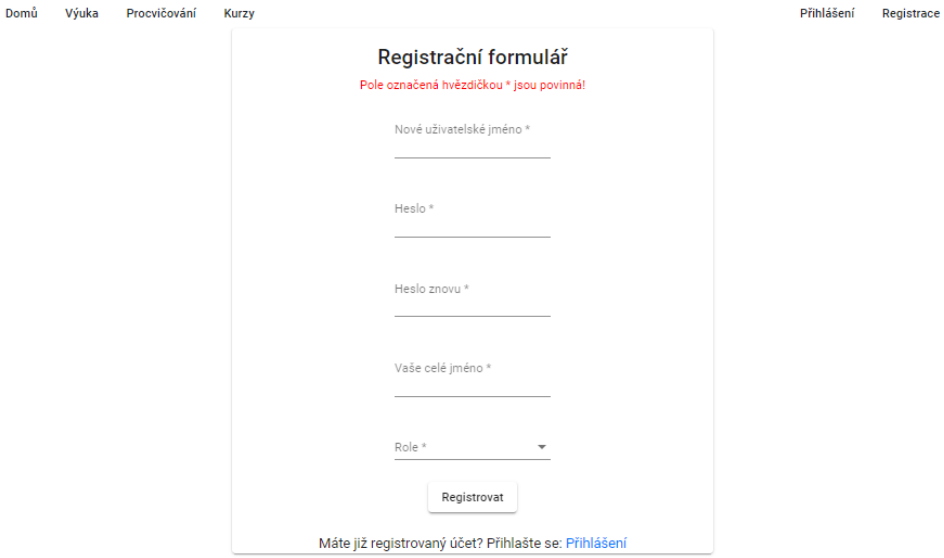


Obrázek 12 - komponenta view (zdroj vlastní)

4.2.3 Komponenty Registration a Login

O přihlášení a registraci do aplikace se starají komponenty *Registration* a *Login*.

Komponenta *Registration* především vykresluje registrační formulář a kontroluje, zda jsou zadaná data validní. Pakliže jsou, tak jsou tato data zaslána na server, kde jsou zpracována a uložena do databáze.



Obrázek 13 - registrační formulář (zdroj - vlastní)

Komponenta *Login* očekává od uživatele dva vstupy, kde jedním z nich je uživatelské jméno a druhým heslo. Po stisknutí tlačítka Přihlásit se, jsou tato data zaslána na server, kde jsou ověřena a pokud jsou správně zadána, je uživatel přihlášen do aplikace.

4.2.4 Komponenta Profile

Tato komponenta je určena pro zobrazení uživatelského profilu. Po úspěšné autentizaci, je uživatel přesměrován na stránku s vlastním profilem. Zde se nachází základní informace o uživateli, tedy: uživatelské jméno, celé jméno uživatele, role uživatele a v neposlední řadě přehled výsledků kurzů, které uživatel absolvoval. K zobrazení těchto výsledků jsou využity komponenty frameworku *Angular Material*, a to především komponenty modulu *MatExpansionModule*. V tomto modulu se nachází komponenta *MatExpansionPanel*. Jedná se o panel, který lze rozvinout (viz následující obrázek).

Obrázek 14 - Ukázka rozvinovacího panelu (zdroj - vlastní)

Na obrázku je vidět, že nejprve je v prvním panelu zobrazen název panelu a počet pokusů, kolikrát ho již uživatel absolvoval. A dále pak jsou jednotlivé pokusy zabaleny do dalších

panelů, po jejichž rozvinutí může uživatel vidět jednotlivé otázky a identifikátory správnosti odpovědí z daného kurzu.

4.2.5 Služba SettingsService

Třída *SettingsService* slouží k uložení nastavení příkladů. Uživatel si může na začátku procvičování ve stejnojmenném modulu nastavit např.: jaký chce používat notační klíč v notové osnově, zda chce procvičovat pouze durové, resp. mollové stupnice v kategorii stupnic, nebo jestli chce procvičovat pouze kvintakordy, resp. septakordy v rámci kategorie spojů akordů. Také informaci o typu příkladu – zda se jedná o interaktivní verzi, nebo neinteraktivní, udržuje tato třída. Kromě těchto nastavení je v této třídě také uložen stav, který nese informaci o tom, zda uživatel spustil procvičování, nebo vybraný kurz.

4.2.6 Služba ZazateService

Tato třída (service) je určena pro generování příkladů. Zde jsou uchovávány vygenerované hodnoty, na základě kterých je poté vytvořen příklad. Zadaná kritéria od uživatele pak tvoří omezující podmínky pro generování hodnot. Např. dur/moll stupnice, nebo kvintakordy/septakordy jsou právě ony zmiňované omezující podmínky. Důležitou metodou této třídy je metoda *generateRandom(min, max, excludedNumbers)*, sloužící pro generování pseudonáhodných číselných hodnot, jež jsou poté následně v jednotlivých metodách převedeny na hodnoty potřebné pro daný příklad. Syntaxe jednotlivých metod pro generování příkladů vychází z předpisu *generateX*, kde *X* může být např.: stupnice, interval, nota atd. (v anglickém jazyce). Zároveň se tato třída stará o veškeré vyhodnocování úkolů a to se děje v metodě *checkAnswer()*, která vrací true/false hodnotu, dle toho jestli byl příklad zadán správně, nebo špatně.

4.2.7 Služba UserService

Tato služba, resp. třída umožňuje komunikaci se serverovou částí aplikace. Slouží především k práci s uživatelskými účty. Pomocí metod této třídy jsou volány *HTTP* požadavky na server. Mezi těmito požadavky jsou požadavky na přihlášení, registraci, nebo profil uživatele. Zároveň se pomocí této třídy přistupuje k uživatelskému tokenu, jež slouží pro autentizaci a autorizaci uživatele.

4.3 Popis funkcionalit aplikace

Aplikace nabízí funkcionality ve třech modulech: Výuka, Procvičování, Kurzy. Každý modul je ještě rozdělen na dva podmoduly, a to podmoduly ZUŠ a Konzervatoř. Podmodul ZUŠ bude vždy zahrnovat nižší úroveň vzdělávání, a naopak podmodul Konzervatoř vyšší úroveň.

4.3.1 Registrace a přihlášení

Jak bylo popsáno v kapitole výše, o registraci a přihlášení uživatele se starají komponenty *Login* a *Registration*. V obou případech tyto komponenty komunikují se službou *UserService*, která volá samotné požadavky na server, resp. do databáze.

Při registraci je volán *HTTP* (může být i *HTTPS*) požadavek typu POST, obsahující objekt sestavený na základě vstupních polí z registračního formuláře. Server tento požadavek přijme a sestaví si z přijatých dat objekt uživatele. V případě, že uživatel se stejným přihlašovacím jménem není registrován, je nový uživatel úspěšně uložen do databáze uživatelů, a je registrován. Při implementaci registračního modulu bylo počítáno s nasazením do reálného provozu, a proto je heslo ukládáno v hašované podobě pomocí hašovacího algoritmu *bcrypt* (částečně popsán v kapitole o použitých balíčcích v *Node.js*).

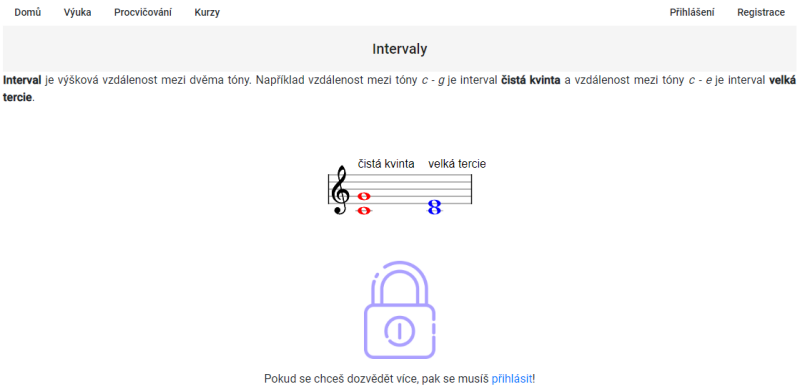
Při přihlášení je volán *HTTP* (může být i *HTTPS*) požadavek typu POST, který obsahuje přihlašovací údaje – tedy uživatelské jméno a heslo uživatele. Přijatá data jsou nejprve ověřena, zda se nachází v databázi. Jestliže ne, tak je uživatel vyzván k opravě přihlašovacích údajů, nebo k registraci nového uživatelského účtu. Pokud uživatel v databázi existuje, tak je na základě id uživatele + času expirace, vygenerován *JWT* token, čímž je uživatel autentifikován a přesměrován na stránku s uživatelským profilem (obsluhuje komponenta *Profile*). Vygenerovaný token je uložen do cookie (= informace uložená webovou stránkou do PC) a na základě tohoto *JWT* tokenu je uživatel autorizován, např. k přístupu do modulu Kurzů. V případě odhlášení uživatele, nebo vypršení expirace, dochází ke smazání *JWT* tokenu z cookies uživatele, který již není autorizován k přístupu do modulu Kurzů.

4.3.2 Modul Výuka

Modul Výuka by měl sloužit k nabytí teoretických znalostí z oblasti hudební teorie, nauky atd. Uživatel bude mít k dispozici materiál, který může posloužit jako učebnice hudební teorie. Co nejlépe srozumitelnou formou bude vysvětlena daná problematika i s ukázkovými příklady, vztahující se k dané problematice.

Konkrétní příklad je ukázán na problematice intervalů. O zobrazení se stará komponenta *Learn*. Tato komponenta nejprve dle vybrané URL nastaví kategorii, týkající se dané problematiky. Kategorii získá z konstanty *LEARN*, kde jsou mapovány jednotlivé hodnoty pro jednotlivé kategorie. Jsou zde tyto parametry: id kategorie - *id*, její název (tak jak se vyskytuje v URL) - *nameCategory*, titulek - *title*, dále viditelný obsah - *visibleContent* a neviditelný obsah - *invisibleContent*. Podle URL je tedy nastaveno id kategorie. Dle této kategorie jsou získány

jednotlivé hodnoty a na základě nich je poté utvořen obsah stránky. Pro registrované uživatele je přístupný vždy celý obsah vybrané kategorie a pro neregistrované, nepřihlášené uživatele pouze obsah vyskytující se v parametru viditelného obsahu (*visibleContent*). Pro zobrazení specifického obsahu (notové osnovy apod.) se používá třída, služba *LearnService*.



Obrázek 15 - uzamčený obsah pro nepřihlášeného uživatele (zdroj - vlastní)

V rámci této práce, resp. aplikace bude pouze nastíněna podoba tohoto modulu, ale její zpracování bude náplní jiné závěrečné práce.

4.3.3 Modul Procvičování

Modul Procvičování bude sloužit k procvičení získaných vědomostí a znalostí. Jak bylo uvedeno výše, je rozdělen do dvou podmodulů a zahrnuje několik kategorií, resp. oblastí.

4.3.3.1 Noty

Kategorie Noty je určena pro nižší úroveň vzdělání ZUŠ. Slouží pro identifikaci not ze zápisu v notové osnově, ale také pro zápis not do notové osnovy dle zadání.

Pokud uživatel vybere, že chce zapsat notu do notové osnovy dle zadání, tak scénář může vypadat takto. Nejprve si vybere, s jakým klíčem v notové osnově chce pracovat. Poté co spustí příklad, aplikace vygeneruje pseudonáhodné číslo od nuly do šesti. Takovéto číslo značí id mapovaných základních not v konstantě *BASICKEYS*. K notám bez posuvky je pak následným dalším generováním určeno, jestli bude nota obsahovat posuvku a popřípadě jakou. Důležitou součástí, která musí být k samotné notě a posuvce doplněna, je oktáva. Protože framework *Vexflow* potřebuje pro vykreslení noty zároveň i její oktávu, určující, kde přesně v notové osnově bude nota umístěna – na jaké lince, resp. v jaké mezeře. Rozhodnutí o tom, jaká oktáva bude notě přiřazena je obsahem metody *generateOctave(clef)*, kde jsou určeny oktávy, dle

vybraného klíče. O celé vygenerování noty s posuvkou a oktávou se stará funkce *generateFullNoteWithOctave()* ze třídy (služby) *ZazateService*.

```
generateFullNoteWithOctave() {
    const noteNumber = this.generateRandom(0, BASICKEYS.length - 1, []);
    let octave = this.generateOctave(this.clef);
    let noteName = Helper.noteHtoB(BASICKEYS[noteNumber].name);
    octave = this.editOctaveByClef(this.clef, octave, noteNumber);

    // set accidental
    noteName += this.generateAccidental();
    return noteName + SLASH + octave;
    }
```

Služba ZazateService si tuto vygenerovanou notu uloží do proměnné a čeká na interakci uživatele, jež spočívá v zadání noty do notové osnovy. Pokud dojde ke shodě, je uživateli oznámeno, že odpověděl správně, v opačném případě, že odpověděl špatně (použita značka křížku a fajfky).

Pokud si uživatel vybere, že chce rozpoznávat (identifikovat) noty dle zápisu v notové osnově, pak proběhne obdobné vygenerování jako v případě zápisu not, s tím rozdílem, že vygenerovaná nota je zapsána do notové osnovy a uživatel má možnost pomocí tlačítek, rozevíracích seznamů a dalších, které obsluhuje komponenta *Answer*, odpovědět a zadat správný název noty vykreslené v notové osnově. Stejně jako při zápisu je uživateli oznámeno, jestli odpověděl správně, nebo špatně.

4.3.3.2 Intervaly

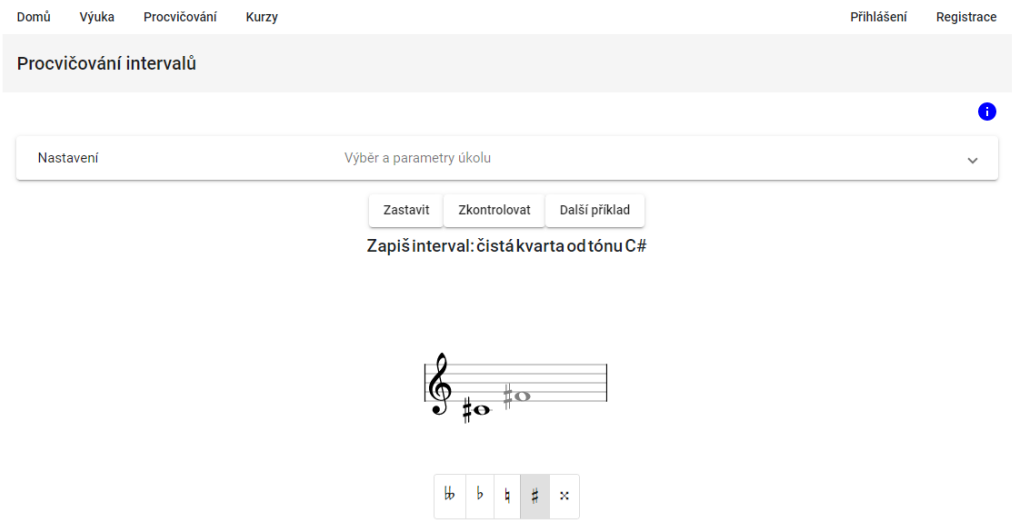
Kategorie Intervaly je určena pro nižší úroveň vzdělávání ZUŠ. Opět se nabízí interaktivní forma, ve které uživatel zapíše interval dle zadané předlohy do notové osnovy, nebo neinteraktivní forma, kde je úkolem uživatele správně pojmenovat zapsaný interval (vzdálenost mezi dvěma notami).

V přednastavení procvičování intervalů si může uživatel vybrat, v jakém notovém klíči chce pracovat a zda chce interval zapisovat, nebo rozeznávat.

V případě zápisu zadaného intervalu do notové osnovy je nejprve vygenerována nota s posuvkou (popsáno v předchozí kapitole), od které se bude vytvářet interval. Poté je vygenerováno pseudonáhodné číslo od 0 do 7, odpovídající intervalu, jež má být zapsán (prima, sekunda, tercie, …). Třetím generovaným číslem je číslo od 0 do 6, které určuje, jestli se bude jednat o základní interval, nebo o odvozený (zvětšená tercie, malá sexta, … = přívlastek). O tento celý proces se stará metoda *generateIntervalForPainting()* ze třídy ZazateService.

```
generateIntervalForPainting() {
  const note = this.generateNoteOnlyWithAccidental();
  const idIntervalName =
              this.generateRandom(0, INTERVALS.length - 1, []);
  let idTypeOfInterval;
  // for perfect intervals - 1,4,5,8 (intervals)
  if (idIntervalName === INTERVALS[0].id) {
    idTypeOfInterval = this.generateRandom(0, INTERVALS.length – 2,
                       [INTERVALS[1].id, INTERVALS[2].id,
                       INTERVALS[3].id, INTERVALS[5].id]);
  } else if (idIntervalName === INTERVALS[7].id) {
    idTypeOfInterval = this.generateRandom(0,INTERVALS.length - 2,
                       [INTERVALS[1].id, INTERVALS[2].id,
                       INTERVALS[4].id, INTERVALS[6].id]);
  } else if (idIntervalName === INTERVALS[3].id
             || idIntervalName === INTERVALS[4].id) {
    idTypeOfInterval = this.generateRandom(0, INTERVALS.length - 2,
                       [INTERVALS[1].id, INTERVALS[2].id]);
  } else {
    // for non-perfect intervals - 2, 3, 6, 7 (intervals)
    if (idIntervalName === 1) {
      idTypeOfInterval = this.generateRandom(1, INTERVALS.length - 2,
                         [INTERVALS[3].id, INTERVALS[5].id]);
    } else if (idIntervalName === INTERVALS[6].id) {
      idTypeOfInterval =
            this.generateRandom(1, INTERVALS.length - 3, []);
    } else {
      idTypeOfInterval =
            this.generateRandom(1, INTERVALS.length - 2, []);
    }
  }
  return [note, idIntervalName, idTypeOfInterval];
}
```

Určení správné odpovědi probíhá v metodě *checkAnswer()* třídy ZazateService, kde jsou zadané noty převedeny na čísla. Nejprve je ze zadání příkladu získána vzdálenost na základě vygenerovaného intervalu (mapováno v konstantách *INTERVALS* a *TYPE_OF_INTERVALS* dle názvu intervalu a jeho přívlastku) a poté za pomocí funkce *measure(note1, note2)* frameworku *Zazate* je spočítána vzdálenost mezi dvěma zadanými notami. Pokud se obě vypočítané vzdálenosti neliší, je uživateli oznámena správná odpověď, v opačném případě špatná.



Obrázek 16 - Ukázka z aplikace – zadávání intervalu (zdroj – vlastní)

Pokud si uživatel vybere, že chce interval identifikovat dle zápisu v notové osnově, tak je nejprve vygenerována nota, od které bude určován zadaný interval a poté je vygenerován samotný interval – jeho název (mapován v konstantách *INTERVALS*) a přívlastek (mapován v konstantách *TYPE_OF_INTERVALS*).

```
export const INTERVALS = [{id: 0, number: 0, name: 'prima'},
  {id: 1, number: 2, name: 'sekunda'},
  {id: 2, number: 4, name: 'tercie'},
  {id: 3, number: 5, name: 'kvarta'},
  {id: 4, number: 7, name: 'kvinta'},
  {id: 5, number: 9, name: 'sexta'},
  {id: 6, number: 11, name: 'septima'},
  {id: 7, number: 12, name: 'oktáva'}];

export const TYPE_OF_INTERVALS = [{id: 0, number: 0, name: 'čistá'},
  {id: 1, number: 0, name: 'velká'},
  {id: 2, number: -1, name: 'malá'},
  {id: 3, number: -2, name: 'zmenšená'},
  {id: 4, number: 1, name: 'zvětšená'},
  {id: 5, number: -3, name: 'dvojzmenšená'},
  {id: 6, number: 2, name: 'dvojzvětšená'}];
```

Na základě těchto vygenerovaných hodnot je vytvořen příklad, dle kterého má uživatel zapsat noty do notové osnovy. Jakmile jsou noty zadány, uživatel má možnost si zkontrolovat odpověď. O kontrolu odpovědi se stará metoda *checkAnswer()*, kde dojde k porovnání vzdáleností mezi dvěma napsanými notami (určena jako absolutní rozdíl mezi výškami těchto not) a vzdáleností, která je získána z konstant s názvy a přívlastky intervalu.

4.3.3.3 Stupnice

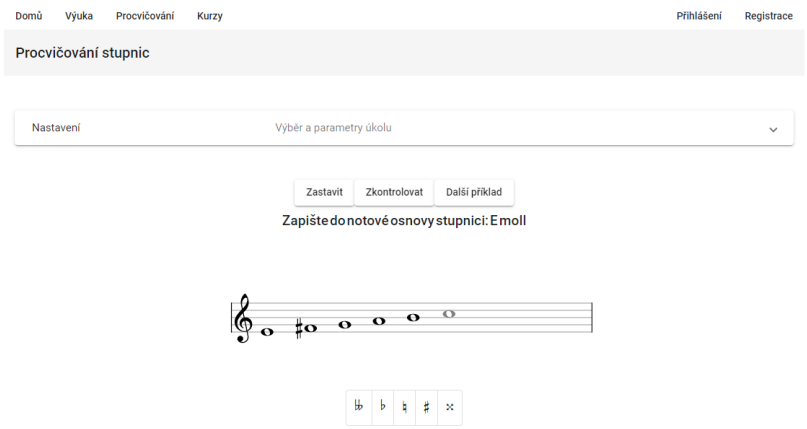
Kategorie Stupnice je určena pro nižší úroveň vzdělávání ZUŠ. Uživatel má na výběr procvičování stupnic mollových a durových. Mezi mollovými stupnicemi jsou zařazeny ještě

mollová harmonická a mollová melodická – vysvětleno v kapitole 3.1.3. Na výběr jsou opět dva typy procvičování – interaktivní (zápis stupnic do notové osnovy) a neinteraktivní (rozeznání stupnice ze zápisu).

V případě interaktivní volby je na základě předvolby uživatele, tedy jestli chce procvičovat durové, nebo mollové stupnice, vygenerována daná stupnice. Pokud vybere oba druhy stupnic, pak musí být nejprve vygenerováno číslo od nuly do jedné (proměnná *majorMinor*), udávající, jaká stupnice bude ve skutečnosti generována. Pokud je generovaná stupnice mollová, musí se určit, o jakou mollovou stupnici se jedná. V nabídce jsou: harmonická moll, melodická moll, nebo přirozená moll (= aiolská) – určeno pomocí vygenerovaného čísla od nuly do dvou (proměnná *isHarmonicMelodic*). Ve chvíli, kdy jsou známy tyto parametry, je dle nich vygenerován úkol pro uživatele. Celý proces generování stupnice probíhá v metodě *generateScale()* třídy *ZazateService*. Jakmile uživatel zapíše stupnici, má možnost zkontrolovat, zda zapsal stupnici správně dle zadání. Kontrola probíhá takto. Zapsaná stupnice je vstupním parametrem funkce *determine(scale)* frameworku *Zazate* a tato funkce vrací pojmenování stupnice ve tvaru – *„TÓNINA POJMENOVÁNÍ_STUPNICE“* (textový řetězec). Nejprve je určena shoda počátečního tónu stupnice s vygenerovaným příkladem, resp. tónina. Poté je získaný textový řetězec z funkce *determine(scale)* rozdělen podle mezer a porovnán s textovým řetězcem (taktéž rozdělen dle mezer), který vychází z vygenerovaných hodnot. Mapování hodnot, resp. názvů stupnic zajišťuje konstanta *MAJORMINOR*, zahrnující hodnoty (id) stupnic, jejich český název a zároveň název odpovídající návratovým hodnotám funkce *determine(scale)* frameworku *Zazate* pro durové a mollové stupnice.

```
export const MAJORMINOR = [
  {id: 0, name: 'moll', zazateName: 'aeolian natural'},
  {id: 1, name: 'dur', zazateName: 'ionian'},
  {id: 2, name: 'moll harmonická', zazateName: 'harmonic minor'},
  {id: 3, name: 'moll melodická', zazateName: 'melodic minor'}
];
```

Pokud se první tón stupnice shoduje s prvním tónem vygenerovaného příkladu a pokud se shodují textové řetězce, tak podobně jako u všech ostatních příkladů je sděleno uživateli, zda se jedná o správnou odpověď, nebo o špatnou.



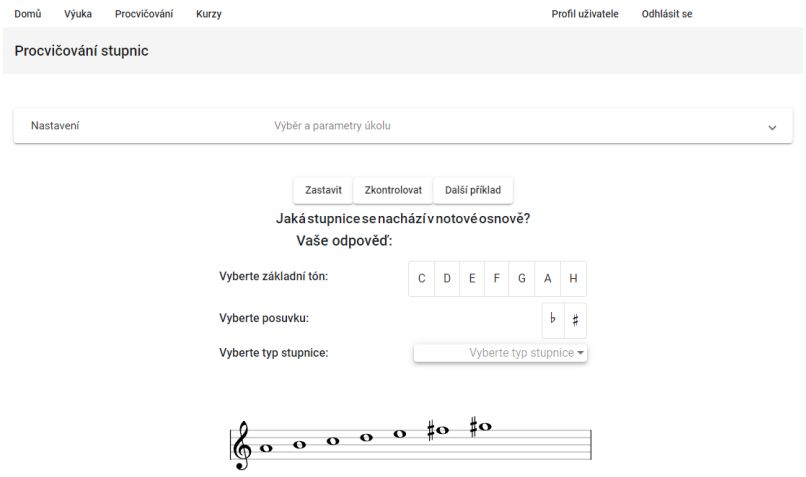
Obrázek 17 - zápis stupnice (zdroj - vlastní)

V případě, že chce uživatel stupnici pouze identifikovat, generování stupnice probíhá následujícím způsobem. Dle přednastavených hodnot – zda chce procvičovat pouze stupnice mollové, nebo durové, anebo oba druhy zároveň, jsou vygenerovány hodnoty obdobným způsobem, jako v případě zápisu stupnic popsaného výše (v metodě *generateScale()*). Na základě těchto hodnot jsou vytvořeny samotné stupnice pomocí funkcí frameworku *Zazate*. Konkrétně funkce *inonian(key)* – pro vytvoření durové stupnice, *natural_minor(key)* – pro vytvoření mollové (aiolské) stupnice, *harmonic_minor(key)* – pro vytvoření mollové harmonické stupnice a *melodic_minor(key)* – pro vytvoření mollové melodické stupnice. Tato

stupnice je uložena do proměnné *generatedScale()* a pro správné vykreslení do notové osnovy musí být každé notě ze stupnice přidána správná oktáva.

```
// ČÁST METODY generateScale()
  // for non-interactive form
   let generatedScale = [];
  // for interactive form
  const generatedScaleKeyName = [];
  let majorMinor;
  if (this.getKeyType() === MAIN_KEY_TYPES[0].id) {
    majorMinor = 1;
  } else if (this.getKeyType() === MAIN_KEY_TYPES[1].id) {
    majorMinor = 0;
  } else {
    majorMinor = this.generateRandom(0, 1, []);
  }
  const numberKey = this.generateRandom(0, 14, []);
  if (!majorMinor) {
    const harmonicMelodic = this.generateRandom(0, 2, []);
    generatedScaleKeyName[0] = MINORKEYS[numberKey].name;
    if (harmonicMelodic === 0) {
      generatedScale =
        zazate.scales.natural_minor(MINORKEYS[numberKey].name);
      generatedScaleKeyName[1] = MAJORMINOR[0].id;
    } else if (harmonicMelodic === 1) {
      generatedScale =
        zazate.scales.harmonic_minor(MINORKEYS[numberKey].name);
      generatedScaleKeyName[1] = MAJORMINOR[2].id;
    } else {
      generatedScale =
        zazate.scales.melodic_minor(MINORKEYS[numberKey].name);
      generatedScaleKeyName[1] = MAJORMINOR[3].id;
    }
  } else {
    generatedScale = zazate.scales.ionian(MAJORKEYS[numberKey].name);
    generatedScaleKeyName[0] = MAJORKEYS[numberKey].name;
    generatedScaleKeyName[1] = MAJORMINOR[1].id;
  }
```

Pak je tato stupnice vykreslena do notové osnovy a uživatel může pomocí tlačítek s jednotlivými tóny a posuvkami a pomocí rozbalovacího menu s názvy stupnic odpovědět (obstarává komponenta *Answer*, resp. její potomek), jaká stupnice je vykreslena v notové osnově. Stejně jako všechny kontroly probíhá kontrola ověření správné odpovědi v metodě *checkAnswer()* třídy *ZazateService*. Nejprve je určeno, jaká stupnice byla vygenerována – pomocí fuknce *determine(scale)* frameworku *Zazate*. Je získán textový řetězec, který je porovnán s příchozím řetězcem. Oba jsou opět rozděleny dle mezer a porovnány. Řetězec získaný v odpovědi vychází z namapovaných hodnot v konstantách *BASICKEYS* (obsahuje všech sedm základních tónů: *C, D, E, F, G, A, H*), posuvky a *MAJORMINOR* konstanty (vysvětleno výše). Pokud jsou řetězce shodné, je uživateli oznámena správná odpověď, v opačném případě špatná.



Obrázek 18 - identifikace stupnic (zdroj - vlastní)

4.3.3.4 Akordy

Kategorie Akordy spadá pod nižší úroveň vzdělávání, tedy ZUŠ. Uživatel má možnost si procvičit základní druhy akordů, resp. septakordů a jejich obratů. V nabídce jsou mollové, durové, zvětšené a zmenšené kvintakordy a tvrdě velké, tvrdě malé, měkce malé, měkce velké a zmenšené (= zmenšeně zmenšené) septakordy. Uživatel si může v přednastavení vybrat, zda chce procvičovat kvintakordy, nebo septakordy zvlášť, popřípadě obojí dohromady. A zároveň jako u většiny typů úkolů, zda chce akordy zapisovat do notové osnovy, nebo je ze zápisu not v notové osnově identifikovat. Oba dva způsoby procvičování jsou aplikovány pouze na jedné notové osnově pro snazší pochopení dané problematiky.

V případě, že chce uživatel zapisovat akordy do notové osnovy, je mu vygenerován dle předvoleb daný typ akordu. Pokud vybere oba typy akordů, je nutné vybrat, zda bude generován kvintakord, nebo septakord. K tomuto účelu slouží pseudonáhodné číslo od nuly (kvintakord) do jedné (septakord) uložené do proměnné *typeChord*. Dalším generovaným číslem je pseudonáhodné číslo od nuly do dvou pro kvintakordy a od nuly do třech pro septakordy. Toto číslo uložené v proměnné *inversion*, značí obrat (např.: sextakord, kvintsextakord apod.), nebo základní tvar daného akordu. Dle typu akordu je generováno číslo pro konkrétní charakteristiku kvintakordu, resp. septakordu. Pro kvintakordy je generováno číslo od nuly do tří a pro septakordy od nuly do čtyř (uloženo v proměnné *characteristicChord*). Tyto hodnoty vycházejí z mapování typů kvintakordů a septakordů v konstantě *TRIAD_SEPTA_TYPES*. Z vytvořeného akordu je dle proměnné *inversion* utvořen obrat pomocí funkcí frameworku *Zazate*: *first_inversion(chord), second_inversion(chord), third_inversion(chord)*, nebo zanechán původní základní tvar akordu. Vygenerovaná čísla opět odpovídají namapovaným hodnotám v konstantě *CHORDS_INVERSION*. Generování akordu obstarává metoda *generateChord()*

třídy *ZazateService*. Na základě hodnoty o typu akordu (kvintakord/septakord), hodnoty o charakteristice akordu (durový kvintakord/měkce-malý septakord), hodnoty obratu akordu a noty, od které má být daný akord vytvořen, je v komponentě *Task* (jejím potomkovi *ChordTask*) vygenerován příklad, který má uživatel zadat (splnit). Pokud uživatel napíše zadaný akord a nechá si ho zkontrolovat, tak je pouze v metodě *checkAnswer()* ve třídě *ZazateService* zjištěno, zda se zadaný akord shoduje s vygenerovaným akordem, neboli jestli obsahuje stejné noty ve stejném pořadí a má stejnou délku - tato kontrola probíhá v metodě *isFirstChordSameLikeSecond(chord1, chord2)*.

V případě, že chce uživatel identifikovat akord zapsaný v notové osnově, je takový akord vygenerován obdobným způsobem, jako při zápisu akordu. S tím rozdílem, že k vygenerovanému akordu, resp. každé jeho notě, je přidána správná oktáva pro zápis do notové osnovy – zajišťuje metoda *addNoteToChord(chord, clef)*. Uživateli je v komponentě *ChordsAnswer* (potomek komponenty *Answer*) umožněno vybrat o jaký typ akordu se jedná (zvětšený, zmenšený, tvrdě velký, tvrdě malý atd.) a o jaký obrat (sextakord, terckvartakord apod.). Obě tyto hodnoty jsou mapovány v konstantách *CHORDS_INVERSION* a *TRIAD_SEPTA_TYPES* (viz výše). Kontrola potom probíhá stejně jako v jiných případech v metodě *checkAnswer()*, kde se porovnává vygenerovaný obrat se zadaným obratem a vygenerovaný typ kvintakordu, resp. septakordu se zadaným typem. V případě shody těchto hodnot je uživateli oznámeno, že se jedná o správnou odpověď, v opačném případě, že se jedná o špatnou odpověď.

4.3.3.5 Rytmus

Kategorie Rytmus se řadí do nižší úrovně vzdělávání. Je zaměřena na rozlišení, určení správného taktu dle rytmického motivu zapsaného v notové osnově. V aplikaci je k dispozici právě rozlišení a určení správného taktu podle motivu bez jakýchkoli dalších nastavení.

Po spuštění procvičování je uživateli vygenerován krátký rytmický úsek (rytmický motiv). A to tak, že v metodě *generateRhythm()* třídy *ZazateService* je nejprve vygenerován druh taktu

(např. 3/4 - „tříčtvrťový“, 2/4 – „dvoučtvrťový“). Na základě tohoto taktu jsou poté vygenerovány hodnoty not, které se shodují s dobou trvání taktu.

```
generateRhythm() {
  const bar = this.generateRandom(0, BASIC_METRICS.length - 1, []);
  const totalResult = BASIC_METRICS[bar].countBeats /
                      BASIC_METRICS[bar].metric;
  const generatedNotes = [];
  let generatedResult = 0;
  while (generatedResult < totalResult) {
    const oneNote = this.generateRandom(0,
                    BASIC_NOTE_DURATION.length - 1, []);
    const valueOneNote = 1 / BASIC_NOTE_DURATION[oneNote].value;
    const tempResult = (valueOneNote) + generatedResult;
    if (tempResult <= totalResult) {
      generatedNotes.push(BASIC_NOTE_DURATION[oneNote].value);
      generatedResult += (valueOneNote);
    }
  }
  return [generatedNotes, bar];
}
```

Tyto hodnoty jsou uloženy do pole a na základě tohoto pole jsou zakresleny do notové osnovy noty daných hodnot (výška noty je staticky nastavena na notu G). Toto vykreslení zajišťuje komponenta *Stave*. Uživatel má pak možnost z rozbalovacího menu vybrat o jaký typ taktu se dle zapsaného motivu jedná (obstarává potomek komponenty *Answer*, konkrétně *RhythmAnswer*). Pokud se uživatel rozhodne pro kontrolu příkladu, tak je porovnáno, zda vybral stejný typ taktu, jaký byl vygenerován.

4.3.3.6 Spoje akordů

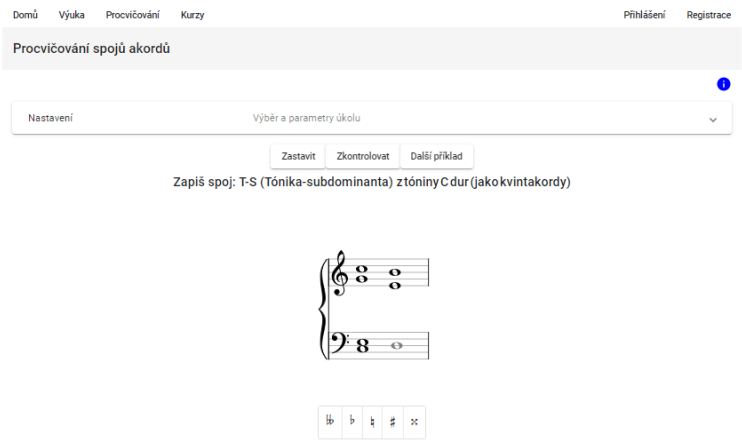
Kategorie Spoje akordů je určena pro vyšší úroveň vzdělávání, tedy pro konzervatoře. Od ostatních kategorií se liší tím, že si uživatel může vybrat, zda chce spoje procvičovat pro kvintakordy, pro septakordy, nebo pro obě dvě možnosti zároveň. Tuto volbu si lze navolit spolu s výběrem typu úkolu (zápis spojů, identifikace spojů) v nastavení, o které se stará komponenta *TaskOption*, resp. její potomek *ChordsConnectionTaskOption*.

Pro oba typy procvičování probíhá generování příkladu velice podobně. Jako první je vygenerováno pseudonáhodné číslo od nuly do jedné, vyjadřující z jaké tóniny budou dané akordy (z mollové, nebo durové). Na základě tohoto čísla je poté vygenerováno další pseudonáhodné číslo, určující základní tón tóniny. Jednotlivé tóniny jsou mapovány v konstantách zvlášť pro mollové (konstanta *MINORKEYS*) a zvlášť pro durové tóniny (konstanta *MAJORKEYS*). Následuje vygenerování pseudonáhodného čísla určujícího obrat kvintakordu, resp. septakordu, dále pak generování pseudonáhodného čísla od nuly do jedné, značící v případě mollové tóniny, zda se bude jednat o klasickou mollovou tóninu, nebo o

harmonickou mollovou tóninu. Na závěr je vygenerováno číslo, které definuje konkrétní spoj, např.: D-T (dominanta s tónikou), D-VI (dominanta s šestým stupněm) apod. (mapováno v konstantě *TYPE_OF_CONNECTIONS*). Na základě těchto vygenerovaných hodnot se vytvoří dva akordy, které splňují pravidla klasické harmonie pro vygenerovaný spoj. Veškeré generování probíhá v metodě *generateChordConnection()* třídy *ZazateService*.

Pokud uživatel vybere, že chce spoj identifikovat, jsou mu v notové osnově vykresleny dva vygenerované akordy a na uživateli je určit, jaký spoj představují v zadané tónině. Zadaný spoj je převeden na číslo pomocí konstanty *TYPE_OF_CONNECTIONS* a porovnán s vygenerovaným spojem.

Pokud uživatel vybere zápis spoje, je spoj vygenerován opět v metodě *generateChordConnection()* a uživateli je v komponentě *Task*, resp. *ChordsConnectionTask* zobrazeno, jaký spoj a v jaké tónině má zapsat. Poté co si nechá svoje řešení zkontrolovat, tak kontrola proběhne následujícím způsobem. Nejprve je první zapsaný akord porovnán s prvním vygenerovaným akordem, resp. jestli zapsaný akord obsahuje správné noty a poté je určeno dle pravidel klasické harmonie, jestli jsou tyto noty rozvedeny správně do druhého akordu. Veškerá logika ověření správnosti probíhá v metodě *checkAnswer()* třídy ZazateService, resp. tato metoda volá *checkChordsConnection()*, kde je kontrolován správný rozvod (spoj) akordů dle zadaných kritérií.



Obrázek 19 - zápis spoje akordů (zdroj - vlastní)

4.3.3.7 Funkce akordů v tónině

Kategorie Funkce akordů v tónině spadá do vyšší úrovně vzdělání, tedy pro konzervatoře. Jedná se o kategorii, kde si uživatel může procvičit a správně identifikovat jednotlivé funkce v tónině. Buď zápisem dané konkrétní funkce, nebo rozpoznáním o jakou funkci se jedná. Procvičování

probíhá pro jednodušší pochopení pouze v jedné notové osnově, podobně jako v kategorii Akordy. Uživateli se v přednastavení nabízí možnost kromě interaktivního a neinteraktivního procvičování vybrat typ funkčních akordů ke cvičení (kvintakordy, nebo septakordy) a k tomu navíc z jaké tóniny (dur nebo moll). Typ akordů a také typ tóniny může být vybrán pro všechny případy (typ akordů pro kvintakordy i septakordy zároveň a typ tóniny durové i mollové zároveň).

Generování příkladů pro tuto problematiku probíhá podobně jako generování akordů. Ovšem zde se ještě navíc přihlíží k charakteru tóniny. Pokud uživatel vybere, že chce funkce z obou typů tónin (durové i mollové), musí být vygenerováno pseudonáhodné číslo od nuly do jedné, které určí, o jakou tóninu se bude jednat. Dle charakteru tóniny jsou poté generovány akordy, stojící na jednotlivých stupních dané tóniny. Pokud je typem akordu kvintakord, jsou generovány funkce kvintakordů stojících na daných stupních tóniny. Naopak pokud je typem akordu septakord, jsou generovány funkce septakordů stojících na daných stupních tóniny. Pro určení funkce (stupně) v dané tónině, je generováno číslo od nuly do šesti udávající jednotlivé stupně tóniny, resp. stupnice. Obdobně jako při generování akordů je ještě vygenerováno pseudonáhodné číslo, jenž udává o jaký obrat daného akordu, na daném stupni se jedná. Proces generování funkce v tónině probíhá v metodě *generateChordFunction()*.

V případě, že chce uživatel zapsat danou funkci, je mu vytvořen, z vygenerovaných hodnot, příklad zobrazený v komponentě *ChordsFunctionTask* (potomek komponenty *Task*), nesoucí informaci o tónině a funkci, která má být vytvořena. Uživatel zapíše akord, který je zadanou funkcí v dané tónině a může si nechat tuto funkci zkontrolovat. Kontrola v metodě *checkAnswer()* probíhá pouze porovnáním vygenerovaného akordu se zapsaným akordem a je určena shoda jednotlivých not a délky obou akordů (opět užita metoda podobně jako v kategorii Akordy - *isFirstChordSameLikeSecond(chord1, chord2)*).

V případě, že chce uživatel rozpoznat danou funkci, je vygenerován akord dle postupu popsaného výše a ke každé notě tohoto akordu se ještě přidává oktáva pro správné vykreslení. Uživatel má možnost odpovědět, o jakou se jedná funkci, díky rozbalovacímu menu v komponentě *ChordsFunctionAnswer* (potomek komponenty *Answer*). Rozbalovací menu je naplněno hodnotami z konstanty *LEVELS*. V metodě *checkAnswer()* dochází k porovnání vygenerované funkce (byla generována taktéž ve shodě s konstantou *LEVELS*) se zadanou funkcí. Správná, nebo špatná odpověď je oznámena uživateli.

4.3.4 Modul Kurzy

Modul Kurzy slouží k souhrnnému procvičování nabytých zkušenosti a znalostí studenta. Ve skutečnosti modul Kurzů funguje velice podobně jako modul Procvičování. S tím rozdílem, že student nemá možnost si navolit, zda chce v dané kategorii zapisovat do notové osnovy, nebo rozpoznávat dle zápisu v notové osnově. Ale může si vybrat právě jednu kategorii, ve níž je mu vygenerován kurz s několika příklady, ke kterým je ještě vygenerováno, jestli se bude jednat o zapisovací příklad, resp. interaktivní, nebo identifikační příklad. K sestavení obsahu jednotlivých kategorií modulu Kurzy je využito stejných komponent jako v modulu procvičovacím, konkrétně jsou využity komponenty: *Task, Answer* a *Stave*. Tyto komponenty sdružuje a jejich výběr řídí jedna hlavní komponenta (podobně jako v modulu Procvičování) a to komponenta *Courses*. Dále je tento modul obohacen o tlačítka k obsluze kurzu a také možností, resp. tlačítkem k uložení výsledků do databáze. Pro demonstraci byly vybrány dva kurzy, popisující princip fungování.

4.3.4.1 Kurz stupnice

Tento kurz je postaven na stejných komponentách jako kategorie stupnic z procvičovacího modulu. Poté co si uživatel vybere tento kurz, zobrazí se mu stránka s krátkým popisem kurzu a tlačítkem pro zahájení kurzu. Po stisknutí tohoto tlačítka se vygenerují potřebné hodnoty k vytvoření příkladu. Nejprve se generuje pseudonáhodná hodnota od nuly do jedné, udávající, o jaký typ příkladu se bude jednat, resp. jestli bude interaktivní, nebo identifikační. To vše obstarává metoda *generateValues()* ze třídy, komponenty *Courses*.

```
generateValues() {
  if (this.isMultiDisciplinary) {
    this.generateCategory();
  }
  let isInteractive = this.zazateService.generateRandom(0, 1, []);
  if (this.category === CATEGORIES[6].id) {
    isInteractive = 0;
  }
  (isInteractive) ? this.settingsService.setTaskType('writing')
    : this.settingsService.setTaskType('indetifying');
  this.settingsService.setIsInteractive(isInteractive);

  this.zazateService.generateTask(
    this.settingsService.getIsInteractive());
}
```

Ve chvíli, kdy je hodnota interaktivity vygenerována, spustí se kurz a zobrazí se první příklad spolu s tlačítkem přesunu na další příklad. (Generování samotného příkladu již poté probíhá stejně jako v případě generování příkladů v procvičovacím modulu.) Uživatel zadá, zodpoví danou otázku a tlačítkem se posouvá dál. Dojde k uložení odpovědi aktuálního příkladu – tato

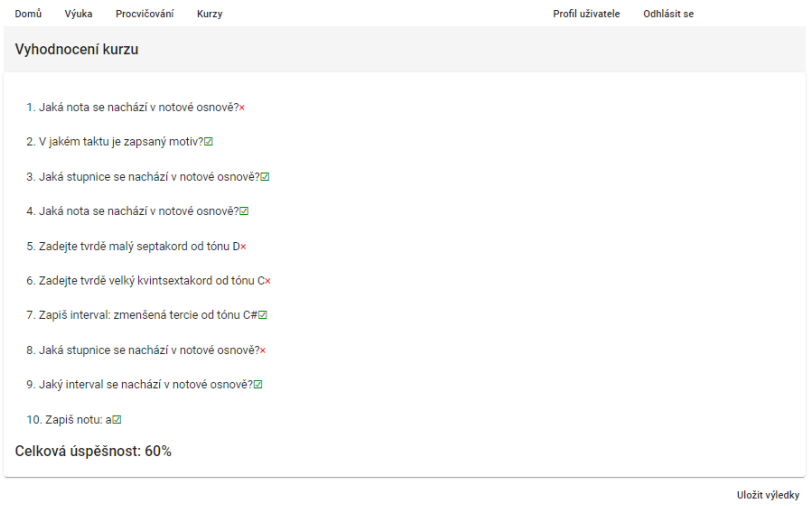
odpověď je uložena ve třídě, službě, se kterou hlavní komponenta spolupracuje a je jí třída *CourseService*. Po uložení se volá opět metoda *generateValues()* a pro uživatele je vytvořen další příklad. Ve chvíli, kdy dorazí na konec kurzu, zobrazí se tlačítko Vyhodnotit a tímto tlačítkem se přesune uživatel do komponenty *CourseResult,* sloužící k vykreslení výsledků kurzu s jednotlivými otázkami ve formě tabulky. U těchto výsledků je zároveň zobrazeno tlačítko pro možnost uložení výsledků do databáze (popsáno v kapitole Uložení výsledků).

4.3.4.2 Kurz hudební nauky

Tento kurz využívá opět stejné komponenty jako ostatní kategorie spadající pod modul procvičování, ale rozdíl oproti všem ostatním kurzům je ten, že je složen z více kategorií. Sdružuje následující kategorie: stupnice, intervaly, noty, akordy a rytmus. Uživateli je tedy nakombinován multidisciplinární kurz. Po zahájení kurzu je nejprve vygenerováno, z jaké kategorie bude daný příklad – generuje se číslo od nuly do pěti (mimo čísla tři a čtyři, jež jsou kategoriemi, nespadající do hudební nauky), které představuje jednotlivé kategorie dostupné v rámci aplikace. Tyto kategorie spolu s jejich číselnými identifikátory jsou mapovány v konstantě *CATEGORIES*. Poté probíhá obdobné generování příkladů a zakončení kurzu, jako bylo popsáno v kurzu stupnic.

4.3.4.3 Uložení výsledků

Uživatel se může rozhodnout, zda chce uložit výsledky právě dokončeného kurzu, nebo chce tyto výsledky ztratit. Samotný proces probíhá pomocí *HTTP* požadavku POST, kdy je do databáze zasílán objekt *Result*, s atributy: *datum a čas dokončení kurzu; položené otázky v kurzu a indikátor* (zda byly odpovězeny správně)*; úspěšnost kurzu* (vyjádřena procentuálním zastoupením správných odpovědí) a *identifikátor uživatele*, který plnil tento kurz.



Obrázek 20 - vyhodnocení kurzu (zdroj - vlastní)

Poté co uživatel zvolí, že chce uložit výsledky, je přesměrován na stránku s vlastním profilem, kde může nalézt všechny dosud absolvované kurzy, včetně data absolvování kurzu a jeho úspěšnosti.

5 ZÁVĚR

Tato práce si kladla za cíl vytvořit webovou aplikaci, která bude umožňovat výuku v oblasti hudební nauky a teorie. Zároveň by měla zahrnovat některé pokročilejší oblasti, týkající se například problematiky harmonie.

Vytvořená aplikace využívá moderních technologií, které jsou v dnešní době trendem při vývoji webových aplikací (*Node.js, Angular, MongoDB* a další). Nabízí několik kategorií, ve kterých se může uživatel vzdělávat. Jak kategorie, které se řadí spíše k základnímu vzdělání v oblasti hudby, tak ale i pokročilejší, které jsou součástí hlubšího vědění při bádání a chápání hudby. Zvláště pak kategorie Spoje akordů a Funkce akordů jsou jedním ze stavebních kamenů při tvorbě hudby nové, nebo i analýze a pochopení hudby starých mistrů jakými byli Johann Sebastian Bach, Wolfgang Amadeus Mozart, Joseph Haydn a další. Aplikace se zaměřuje na základní a nejrozšířenější oblast, která je součástí školních, vyučovacích, hudebních programů po celém světě, tedy klasickou harmonii.

Uživatel si může nejen procvičit všechny znalosti, ale také získat povědomí o pokroku ve studiu. Má tak přehled o tom, zda je připraven dobře analyzovat složitější hudební útvary a pochopit vztahy jednotlivých not v rámci melodie, resp. hudební myšlenku.

Aplikace je vytvořena jako prototyp určený k neustálému rozšiřování a doplňování nových funkcionalit. I již implementované funkcionality mohou být rozšířeny. Jedním z takových příkladů může být kategorie Procvičování spojů akordů, kde je zohledněna pouze jedna oblast v rámci této problematiky. Tato kategorie se zaměřuje především na přísné spoje, ovšem v běžné hudební praxi se objevují častěji spoje volné, ale i spoje netypické, které mají svůj význam v kontextu daného hudebního úryvku. Je tedy zřejmé, že implementace všech plnohodnotných kategorií a oblastí hudby, vyžaduje veliké úsilí.

Hlavním cílem této práce bylo započít dílo, resp. aplikaci, která přesahuje rozsah této práce a s největší pravděpodobností i možnosti jednoho člověka. Aplikaci, která by mohla být pomocníkem studentů, ale také učitelů hudební teorie (z čímž je v budoucnu v aplikaci také počítáno). Zároveň je tato práce (aplikace) připravena jako podklad pro jinou závěrečnou práci zaměřující se na konkrétní specifickou oblast, zpracování teoretických informací z oblasti hudby, a její implementaci do webové aplikace. Z důvodu kontinuálního rozšiřování aplikace nebyl kladen důraz na samotný design aplikace. Spíše než design byla zohledněna celková funkcionalita aplikace s plánem doplnění designu v budoucnosti, při nasazení do reálného provozu.

LITERATURA

[1] *RisingSoftware* [online]. Melbourne, Australia: Rising Software, ©2019 [cit. 2019-03-25]. Dostupné z: https://www.risingsoftware.com/.

[2] Auralia5: Ear training with real music. *RisingSoftware* [online]. Melbourne, Australia: Rising Software, ©2019 [cit. 2019-03-16]. Dostupné z: https://www.risingsoftware.com/auralia/.

[3] Musition 5: Fun ways to learn music theory & musicianship. *RisingSoftware* [online]. Melbourne, Australia: Rising Software, ©2019 [cit. 2019-03-16]. Dostupné z: https://www.risingsoftware.com/musition/.

[4] *Musictheory* [online]. Musictheory.net, ©2000-2018 [cit. 2019-03-16]. Dostupné z: https://www.musictheory.net/.

[5] *UTheory* [online]. uTheory.com, LLP, ©2015-2019 [cit. 2019-03-16]. Dostupné z: https://utheory.com/.

[6] *Teoría: Music Theory Web* [online]. Caribbean island of Puerto Rico: MERLOT, ©1997-2019 [cit. 2019-03-16]. Dostupné z: http://teoria.com/.

[7] *Dave Conservatoire* [online]. London: Dave Conservatoire, ©2019 [cit. 2019-03-18]. Dostupné z: http://www.daveconservatoire.org/.

[8] ŠIMEČEK, Mgr. Karel. Aplikace k procvičení a testování hudební teorie a praxe. *Metodický portál: Články*[online]. 2019 [cit. 2019-03-18]. ISSN 1802-4785. Dostupné z: https://clanky.rvp.cz/clanek/c/U/21866/aplikace-k-procviceni-a-testovani-hudebni-teorie-a-praxe.html/.

[9] *Cvičebna* [online]. Buštěhrad: ZUŠ Buštěhrad, ©2019 [cit. 2019-03-25]. Dostupné z: http://cvicebna.zusbustehrad.cz/.

[10] *Pug's notation training* [online]. Ivan Kuckir, ©2012 [cit. 2019-03-25]. Dostupné z: http://notes.musicpug.com/.

[11] Online učitel k procvičování not. *Jankoweb* [online]. JK, ©2008-2019 [cit. 2019-03-25]. Dostupné z: http://jankoweb.wz.cz/blog/skripty/online-ucitel-k-procvicovani-not/.

[12] *EuroDIDACT* [online]. Olomouc: euroDIDACT, ©2019 [cit. 2019-03-18]. Dostupné z: https://www.eurodidact.cz.

[13] ŠIMEČEK, Mgr. Karel. Mobilní aplikace k procvičení a testování hudební teorie a praxe. *Metodický portál: Články* [online]. 2019 [cit. 2019-03-18]. ISSN 1802-4785. Dostupné z: https://clanky.rvp.cz/clanek/c/U/21869/MOBILNI-APLIKACE-K-PROCVICENI-A-TESTOVANI-HUDEBNI-TEORIE-A-PRAXE.html/.

[14] HTML Introduction. *W3Schools* [online]. W3schools, ©1999-2019 [cit. 2019-02-21]. Dostupné z: https://www.w3schools.com/html/html_intro.asp.

[15] CSS Introduction. *W3Schools* [online]. W3schools, ©1999-2019 [cit. 2019-02-21]. Dostupné z: https://www.w3schools.com/css/css_intro.asp.

[16] ARSENAULT, Cody. Top 10 Front-End Frameworks of 2018. *KeyCDN* [online]. Proinity, ©2019, 4. 9. 2018 [cit. 2019-02-21]. Dostupné z: https://www.keycdn.com/blog/front-end-frameworks.

[17] Bootstrap. *Bootstrap* [online]. Bootstrap team, ©2019 [cit. 2019-02-26]. Dostupné z: https://getbootstrap.com/.

[18] ZAKAS C., Nicholas. *Understanding ECMAScript 6: The definitive guide for JavaScript developers*. San Francisco: No Starch Press, [2016], XXI. ISBN 1-59327-757-1.

[19] TypeScript. *TypeScript* [online]. Microsoft, ©2012-2019 [cit. 2019-02-26]. Dostupné z: http://www.typescriptlang.org/.

[20] Welcome to the mean stack. *MEAN* [online]. linnovate, ©2019 [cit. 2019-03-04]. Dostupné z: http://mean.io/.

[21] What is MongoDB?. *MongoDB* [online]. MongoDB, ©2019 [cit. 2019-03-12]. Dostupné z: https://www.mongodb.com/what-is-mongodb

[22] Express: Fast, unopinionated, minimalist web framework for Node.js. *Express* [online]. StrongLoop, IBM, and other expressjs.com contributors, ©2017 [cit. 2019-03-11]. Dostupné z: https://expressjs.com/.

[23] MÁCA, Jindřich. Lekce 1 - Úvod do Angular frameworku. *ITnetwork.cz* [online]. Praha: David Čápka, ©2019 [cit. 2019-03-11]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku.

[24] MÁCA, Jindřich. Lekce 1 - Úvod do Node.js. *ITnetwork.cz* [online]. Praha: David Čápka, ©2019 [cit. 2019-03-11]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.

[25] *Mongoose* [online]. LearnBoost, ©2011 [cit. 2019-04-29]. Dostupné z: https://mongoosejs.com/.

[26] Passlib.hash.bcrypt - BCrypt. *PassLib* [online]. Assurance Technologies, ©2008-2017 [cit. 2019-04-29]. Dostupné z: https://passlib.readthedocs.io/en/stable/lib/passlib.hash.bcrypt.html.

[27] HANSON, Jared. Passport. *GitHub* [online]. San Francisco: GitHub, ©2019 [cit. 2019-04-29]. Dostupné z: https://github.com/jaredhanson/passport.

[28] Jsonwebtoken. *GitHub* [online]. San Francisco: GitHub, ©2019 [cit. 2019-04-29]. Dostupné z: https://github.com/auth0/node-jsonwebtoken#readme.

[29] Angular Material. *Angular Material* [online]. Google, ©2010-2019 [cit. 2019-03-12]. Dostupné z: https://material.angular.io/.

[30] What is VexFlow?. *VexFlow* [online]. oxfe, ©2019 [cit. 2019-03-12]. Dostupné z: http://www.vexflow.com/.

[31] BOUTGLAY, Wael. Zazate.js. *GitHub* [online]. San Francisco: GitHub, ©2019 [cit. 2019-03-12]. Dostupné z: https://github.com/btwael/zazate.js.

[32] *Hudební teorie jako součást hudební nauky na ZUŠ* [online]. Olomouc, 2014 [cit. 2019-03-31]. Dostupné z: https://theses.cz/id/rofz18/Hudebn_teorie_jako_soust_hudebn_nauky_na_ZU_-_Kristna_Ode.pdf. Bakalářská práce. Univerzita Palackého v Olomouci, Fakulta pedagogická. Vedoucí práce PaedDr. Lena Pulchertová, Ph.D.

[33] ZENKL, Luděk. *ABC hudební nauky*. 7. revidované a doplněné vydání. Praha: Editio Bärenreiter, 2000, s. 9-24. ISBN 80-86385-01-9.

[34] ZENKL, Luděk. *ABC hudební nauky*. 7. revidované a doplněné vydání. Praha: Editio Bärenreiter, 2000, s. 78-80. ISBN 80-86385-01-9.

[35] ZENKL, Luděk. *ABC hudební nauky*. 7. revidované a doplněné vydání. Praha: Editio Bärenreiter, 2000, s. 55-62. ISBN 80-86385-01-9.

[36] ZENKL, Luděk. *ABC hudební nauky*. 7. revidované a doplněné vydání. Praha: Editio Bärenreiter, 2000, s. 96. ISBN 80-86385-01-9.

[37] ZENKL, Luděk. *ABC hudební nauky*. 7. revidované a doplněné vydání. Praha: Editio Bärenreiter, 2000, s. 28-36. ISBN 80-86385-01-9.

[38] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 21-22. Hudební pedagogika. ISBN 978-80-7331-199-5.

[39] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 80-81. Hudební pedagogika. ISBN 978-80-7331-199-5.

[40] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 126. Hudební pedagogika. ISBN 978-80-7331-199-5.

[41] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 80-81. Hudební pedagogika. ISBN 978-80-7331-199-5.

[42] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 130. Hudební pedagogika. ISBN 978-80-7331-199-5.

[43] Harmonicky myslet a slyšet. TICHÝ, Vladimír. *Harmonicky myslet a slyšet*. 2., opr. a rozš. vyd. Praha: Nakladatelství Akademie múzických umění, 2011, s. 110-136. Hudební pedagogika. ISBN 978-80-7331-199-5.