Mendelova univerzita v Brně
Provozně ekonomická fakulta

Interaktivní e-learningová aplikace pro podporu hudební nauky na ZŠ

Diplomová práce

Vedoucí práce:
Ing. Ondřej Popelka, Ph.D.

Bc. Michal Fíbek

Brno 2015

Děkuji především svému vedoucímu, Ing. Ondřeji Popelkovi, Ph.D., za trpělivost, odborné rady a konstruktivní připomínky při vypracování této diplomové práce. Dále bych chtěl poděkovat Mgr. Martinu Grobárovi za prvotní námět, didaktický pohled při návrhu her a praktické otestování s jeho žáky ve výuce. Děkuji také Ing. Martinu Ševčíkovi za konstruktivní diskuze o dílčích postupech a algoritmech a celé mé rodině a přítelkyni za obrovskou podporu a trpělivost. V konečné řadě bych rád poděkoval všem žákům, kteří se zapojili do praktického testování aplikace a jejichž zpětná vazba mě velmi motivovala v průběhu dokončování práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Interaktivní e-learningová aplikace pro podporu hudební nauky na ZŠ**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 19. května 2015 .....

Abstract

Fíbek, M. Interactive e-learning application to support music education for elementary schools. Brno, 2015

This thesis deals with the design and implementation of multi-platform web application based on HTML5, HTML5 Audio and Web Audio API, designed for elementary school students to develop their musical skills, especially musical hearing. The first part compares current educational programs designed to practise musical hearing and to educate musical notation. The following section develops design of music games and application interface, which is based on requirements, followed by analysis of web technologies for sound playback. Last part describes implementation of application in PHP (Nette Framework) with MySQL database and user testing.

Keywords

Web application, music education, responsive design, Web Audio API, HTML5 Audio, Nette Framework.

Abstrakt

Fíbek, M. Interaktivní e-learningová aplikace pro podporu hudební nauky na ZŠ. Brno, 2015

Tato práce obsahuje návrh a implementaci multiplatformní webové aplikace využívající HTML5, HTML5 Audio a Web Audio API, určené pro rozvoj hudebních dovedností žáků na základní škole s důrazem na hudební sluch. V první části jsou porovnány současné výukové programy, určené k procvičování hudebního sluchu a notového zápisu. Na základě požadavků je proveden návrh hudebních her, prostředí aplikace a rozbor technologií přehrávání zvuku na webu. V konečné části je aplikace implementována pomocí PHP (Nette Framework) a MySQL a je provedeno uživatelské testování.

Klíčová slova

Webová aplikace, hudební výchova, responzivní design, Web Audio API, HTML5 Audio, Nette Framework.

Obsah

1 Úvod

Vyučovací hodiny na základních školách jdou technologicky dopředu. Učitelé stále častěji využívají interaktivní tabule, tablety a další technické pomůcky, které zvyšují názornost a atraktivitu informací, které předávají dětem.

V hudební nauce však stále není využito maximum potenciálu, které tyto nástroje nabízí. Vyučující se může obrátit na komplexní výukové balíky, které jsou z drtivé většiny anglicky a omezené pouze na jednu platformu, nebo na jednoduché hudební hry, které nejsou plně přizpůsobeny školnímu prostředí a jejím požadavkům a jsou spíše určeny k pobavení.

U výuky hudby je velkou přidanou hodnotou, pokud aplikace dokáže žáka opravdu zaujmout a vzbudí v něm zájem s ní pracovat také mimo školní prostředí. Toto je problém, který je částečně řešitelný důrazem na vysokou uživatelskou přívětivost a návrhem aplikace, která dokáže vhodně zacílit na motivace žáka. Neméně důležitá je také podpora platforem, které má žák doma k dispozici.

Toto téma je pro mne zajímavé z toho důvodu, že díky mému zájmu o audiovizuální tvořivost, tvorbu webových stránek a obor uživatelské přívětivosti je pro mě výzvou navrhnout aplikaci, kterou budou žáci rádi používat a která je skutečně něco v tomto oboru naučí.

1.1 Cíl práce

Cílem práce je implementovat aplikaci, která bude formou několika hudebních her rozvíjet hudební sluch žáků a napomáhat při výuce hudební nauky na základních školách. Aplikace by měla být uživatelsky přívětivá a pro žáky přirozeně motivující k používání a tréninku jejich schopností.

Řešení musí být pohodlně použitelné i pro učitele, aby jim na základě aktivity žáků umožnilo průběžné vyhodnocování dosažených výsledků a pomohlo jim také jako jeden z faktorů při hodnocení žáka ve škole.

Důležitá je kompatibilita a multiplatformost aplikace – kromě klasických počítačů by měla být spustitelná na většině mobilních zařízení s moderními internetovými prohlížeči, s podporou dotykového i nedotykového ovládání.

Nedílnou součástí práce bude také otestování získaných poznatků v praxi, tedy v reálné výuce samotnými žáky a učitelem. Z tohoto praktického experimentu získám výstupy, které vyhodnotím a rozvedu do dalších možností budoucího rozvoje.

2 Porovnání hudebních výukových aplikací

V současné době existuje větší množství aplikací, které pracují s výukou hudební nauky. Mé srovnání by mělo přinést objektivní pohled na celou problematiku z hlediska klíčových faktorů a jejich vah.

Do srovnání zařadím různorodé aplikace, které jsou alespoň částečně přizpůsobeny pro školní prostředí. Cíl, kterého by měly všechny dosáhnout, je rozvoj hudebního sluchu u žáků, přičemž konkrétní postup a metoda se může velmi lišit.

Pojmem *aplikace* budu v této práci označovat obecně všechny programy nezávisle na platformě a způsobu použití (dotyková/nedotyková).

Pro označení konkrétní výukové funkce dané aplikace, tedy nejčastěji jedné z voleb v hlavním menu, budu pracovat obecným s výrazem *výuková hra*, příp. *výukový test*. Výukovou hrou nebo testem může být např. poznávání not, vytleskávání rytmu, skládání hudební melodie, test na hudební teorii apod.

Všechny aplikace určené pro osobní počítače testuji v systému Microsoft Windows 8.1, v případě webových aplikací v prohlížeči Google Chrome 42.

2.1 Metodika hodnocení

Při porovnávání aplikací vycházím z klasické rozhodovací analýzy s využitím metody známkování. Většina faktorů je kvalitativních, pouze cena se dá řadit do kvantitativní kategorie.

Následují kritéria jsou označena váhami v intervalu 1–5, kde 5 je nejdůležitější a 1 nejméně důležité kritérium. Hodnocení provádím analogicky – známka 0 je naprosto nevyhovující, naopak 5 nejlepší výsledek.

2.1.1 Platforma

Jedním z charakteristických faktorů při prvním výběru aplikace je šíře podpory zařízení a systémů. Pokud by byla aplikace např. ryze desktopová pro systém Mac OS X, nelze očekávat, že jej bude moci provozovat většina uživatelů.[1]

Naopak aplikace určené pro provoz v internetovém prohlížeči mají výhodu vysoké šíře podporovaných platforem, protože nejsou vázány na konkrétní API[2] operačního systému.

Váha: 2/5

2.1.2 Výukové možnosti

Klíčovým parametrem je rozsah výukových testů nebo her, které daná aplikace zahrnuje. Nižší počet značí vyšší specializaci aplikace, což nemusí být vždy špatně – při splnění kritéria příznivé cenové politiky může úspěšně fungovat jako doplněk ke

[1] Vycházím z globálních statistik podílu operačních systémů od ledna 2014 do ledna 2015 [1]

[2] Application Programming Interface

stávající výuce. Pro mé srovnání je ale žádoucí mít ideálně jednu aplikaci, která zahrnuje více oblastí hudební teorie do nabídky obsažených výukových her.

Váha: 4/5

2.1.3 Uživatelská přívětivost

Důraz na vysokou uživatelskou přívětivost, kterou se zabývá oblast *UX (User Experience)*[3], musí být nedílnou součástí návrhu každé aplikace. Při návrhu e-learningových aplikací je tento důraz ještě vyšší, protože aplikace se správným designem dokáže učení zpříjemnit a zjednodušit do té míry, že se tato činnost nestává frustrující, nudnou či obtěžující, ale naopak skutečně učí hrou [3, s. 3–4].

U hodnocení uživatelské přívětivosti je obvykle těžké vyvarovat se subjektivizaci – z tohoto důvodu budu využívat metodiky evaluace dosažení cílů [2, s. 116–119]. Jako cíle si stanovím první spuštění aplikace, pochopení a splnění úkolu v první zvolené výukové hře a zobrazení osobního hodnocení.

Váha: 3/5

2.1.4 Rozhraní pro učitele

Pro použití ve školním prostředí je velmi důležité, aby byla aplikace dobře přístupná pro učitele, který by mohl sám přizpůsobovat její obtížnost, nastavovat obsah výukových her, spravovat přístupy žáků a sledovat jejich aktivitu a rozvoj jejich schopností.

Váha: 3/5

2.1.5 Česká lokalizace

Protože srovnávám aplikace pro nasazení v české základní škole, je velmi žádoucí, aby produkt obsahoval podporu českého jazyka. Důležité je, že jazyková bariéra by neměla být nejen na straně samotného uživatelského rozhraní, ale ani v notovém zápisu. Rozdíl je například ve značení noty, která je v České republice známá pod označením *h* – v USA, Spojeném království a dalších anglosaských zemích je značena písmenem *b* [4]. Formát, používaný u nás, bývá někdy nazýván jako *německá notace*, v textu na něj dále odkazuji jako na *českou notaci*.

Váha: 5/5

2.1.6 Cena

Jedním z nejdůležitějších kritérií je cenová dostupnost – je zřejmé, že v ideálním případě by aplikace měla být zcela zdarma, případně by její licenční politika mohla

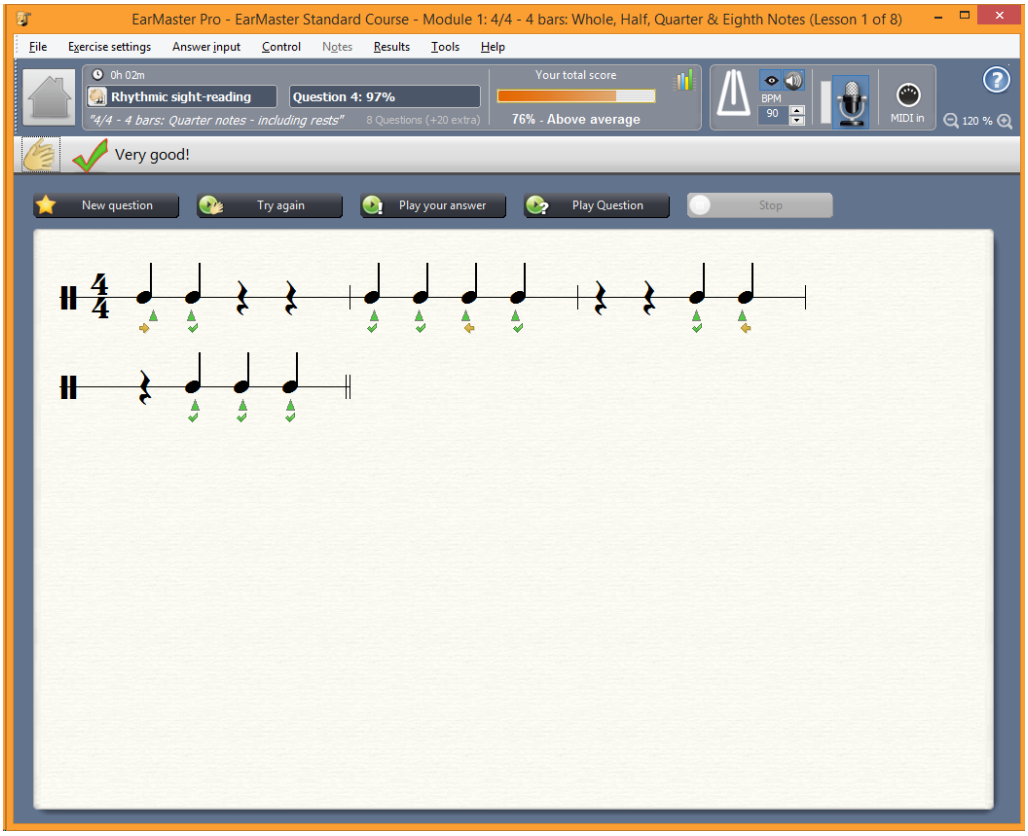
[3] UX je součástí velkého vědního oboru HCI (Human-Computer interaction), jenž zahrnuje obecně veškerou interakcí uživatele s počítačem [2, s. ix].

umožňovat zakoupení hromadných multilicencí pro všechny žáky a učitele dané školy. Tímto způsobem se zbavím důležité demotivační překážky pro samostatný trénink žáků doma.

Při srovnávání uvádím z důvodu objektivity ceny uvedené na oficiálních webových stránkách, přestože mají některé z aplikací také českého distributora.

Váha: 3/5

2.2 EarMaster Pro 6



Obrázek 1: EarMaster Pro 6 – hra rytmického vytleskávání

Komerční aplikace EarMaster [5] je nabízena ke stažení v několika různých edicích s více cenovými politikami. Následující hodnocení provádím na základě zkušební verze pokročilejší verze EarMaster Pro 6 se zmínkou základní varianty EarMaster Essential 5.

2.2.1 Platforma

Všechny aplikace EarMaster jsou určeny pro operační systémy Windows a Mac OS X. Při použití tabletu se systémem Windows lze předpokládat i funkčnost

dotykového ovládání, aplikace ale není svým designem k tomuto způsobu práce přizpůsobena.

Hodnocení: 2/5

2.2.2 Výukové možnosti

EarMaster Essential nabízí sadu čtyř základních testů – rozpoznávání intervalů, identifikace intervalů, rozpoznávání akordů a čtení rytmu. EarMaster Pro se navíc dělí na oblasti *intervaly*, *akordy a stupnice*, *rytmus* a *melodie*, přičemž v každé z nich jsou 3–4 výukové hry.

Nespornou výhodou této aplikace je aktivnější zapojení uživatele, protože využívá vstupu z mikrofonu u her se zpěvem tónů a vytleskávání rytmů. Tleskání funguje velmi spolehlivě, u zpěvu je však aplikace citlivější na kvalitu mikrofonu a hlas uživatele.

Samotné hry se blíží svým pojetím běžné výuce, využívají klasickou notovou osnovu, takže mohou být pro žáka na první pohled méně atraktivní. Základní motivační systém je obsažen ve formě zobrazení průměrného procentního hodnocení úspěšnosti u aktuální spuštěné hry a možnosti otevřít statistiky všech her z historie, jedná se tedy opět spíše o tradičnější přístup k hodnocení.

Hodnocení: 3/5

2.2.3 Uživatelská přívětivost

Pro stažení libovolné zkušební verze aplikace EarMaster je nutné zadat vlastní e-mail, na který je poslán odkaz ke stažení. Z pohledu prvního spuštění aplikace tak jde o komplikaci, která brání okamžitému použití. Samotná instalace již probíhá běžným způsobem.

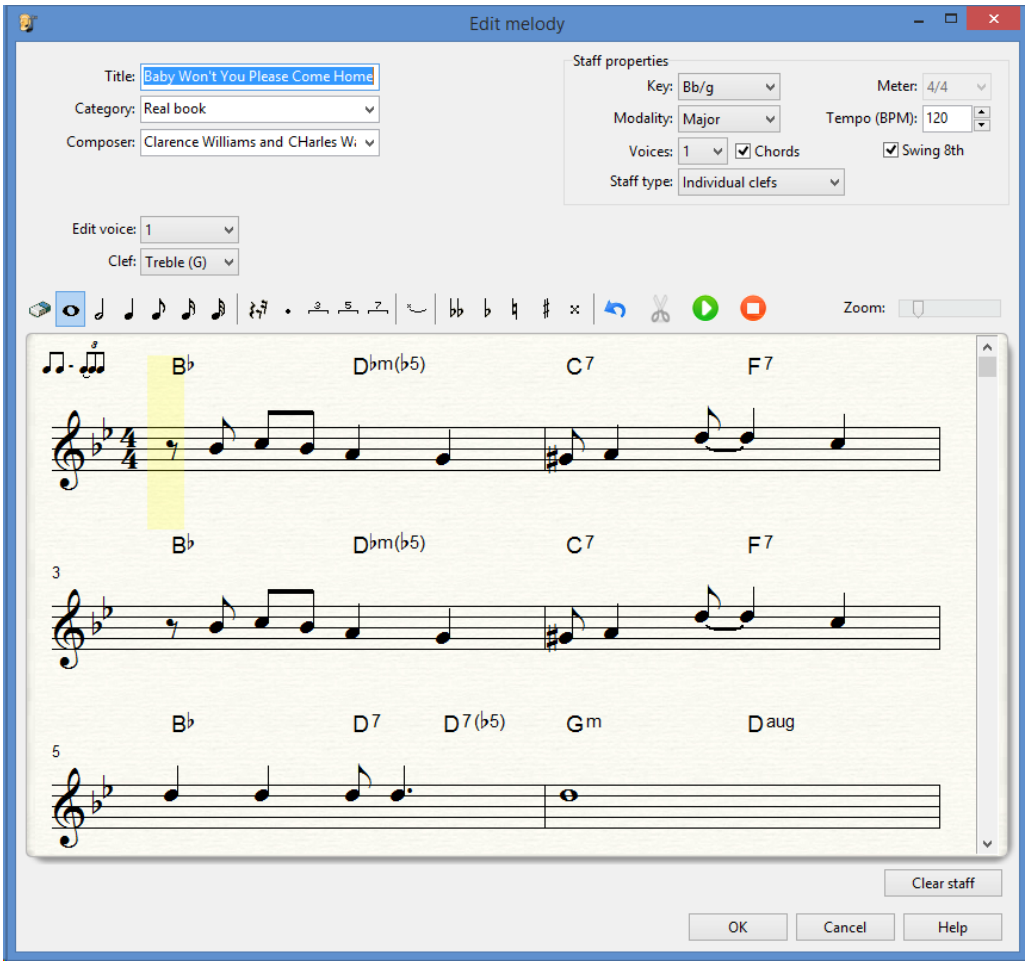
Při prvním spuštění EarMaster Pro program vyzve k potvrzení zkušebního režimu, poté průvodce postupně nabídne konfiguraci zvukového výstupu, výběr cvičeného nástroje a nastavení hlasového rozsahu. Konfigurace je pro laika relativně jednoduchá, vše je podrobně popsáno. Při opakovaných spuštěních aplikace nás již průvodce neobtěžuje a dostáváme se přímo do hlavního menu.

Navigace v hlavním menu je jednoduchá, dominují jí jednotlivé aktivity, tedy výukové hry. Při prvním použití her s mikrofonem však narážím na problém, že mikrofon není okamžitě rozpoznán, je nutné jej tedy zvlášť kalibrovat v jeho samostatném průvodci, což mi aplikace nejprve nedává najevo.

Spuštění libovolné hry je přímočaré, v hlavním menu zvolím výukovou hru a poté dostávám v modálním okně nabídku s výběrem lekce, včetně příkladu obsahu. Pochopení principu her není vždy snadné, některé mají složitější a nepříliš zřejmé ovládání. Hodnocení je zobrazeno v průběhu hry, historii všech her se zobrazením průměrného výsledku mohu zobrazit v menu *Results – Statistics*.

Hodnocení: 3/5

2.2.4 Rozhraní pro učitele



Obrázek 2: EarMaster Pro – editor melodií

V Earmaster Essential chybí jakákoliv možnost ovlivňovat výuku učitelem – aplikace je je připravena spíše k individuální výuce. Earmaster Pro obsahuje rozlišení uživatelských účtů na žáka a učitele. Rozhraní učitele obsahuje:

- editor kurzů (lekcí),
- databázi melodií,
- přehled výsledků studentů,
- správu uživatelů,
- správu tříd.

V editoru kurzů má učitel možnost vytváře vlastní lekce, ve kterých nastavuje typ aktivity (výukové hry), počet otázek a také obtížnost pomocí procenta správně zodpovězených otázek žákem, které jsou nutné ke schválení výsledku.

Databáze melodií obsahuje možnost vytvářet a upravovat použité melodie pomocí grafického notového zápisu (obr. 2) a importovat nové melodie ve formátu MusicXML [6].

Velkou výhodou je synchronizace výsledků a kurzů přes internet u verze EarMaster Cloud. Díky tomu může mít učitel aktuální přehled o činnosti žáků i v případě, kdy budou aplikaci používat v rámci cloudu doma.

Hodnocení: 5/5

2.2.5 Česká lokalizace

Aplikace nabízí velkou škálu podporovaných jazyků, je však zvláštní, že zatímco verze Essential 5 mezi nimi zahrnuje také češtinu, ve verzi Pro 6 požadovaný jazyk chybí. Právě tohle je velká slabina použitelnosti programu pro základní školy, protože jedině verze Pro je součástí multi a cloudové licence.

Hra neobsahuje nastavení zobrazení noty *h* podle české notace – toto nastavení je ovlivněno podle zvoleného jazyku. V často kladených otázkách na oficiálních webových stránkách je zmíněna možnost měnit kompletní terminologii a ostatní texty programu ruční úpravou textového souboru `English.lan`, kde se nachází také značení not. U popisu produktu na webu je uveden seznam podporovaných jazyků, s upozorněním, že další budou přidány – lze tedy předpokládat, že česká lokalizace (společně s českou notací) bude doplněna i pro EarMaster Pro 6.

Hodnocení: 2/5

2.2.6 Cena

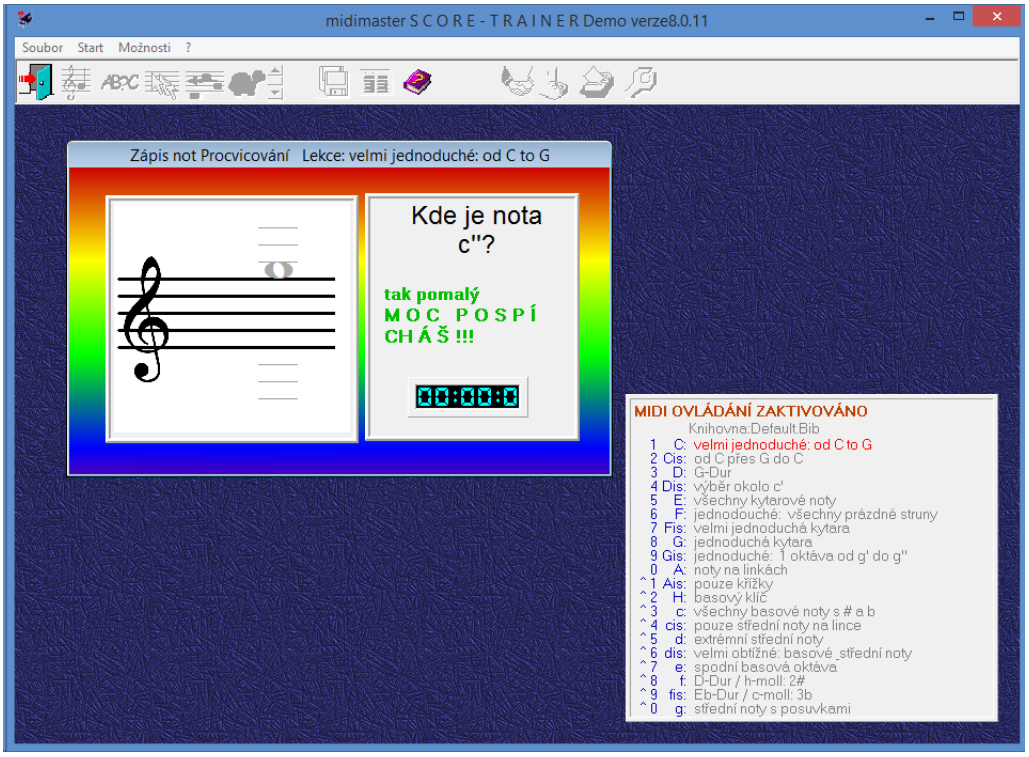
Základní časově neomezené „krabicové“ verze jsou dostupné ve variantách:

- EarMaster Pro 6 – 59,95 €
- EarMaster 6 Teacher – 69,95 €
- EarMaster Essential 5 – 29,95 €

Verze Teacher a Pro je možné zakoupit v multilicenci vázané na vybraný počet počítačů s časově neomezenou platností, která je označena jako *Lab Pack*.

V nabídce je také cloudová licence, která např. ve variantě 100 licencí vychází cenově 5 €/osoba/rok (nezávisle na tom, zda jde o studenta nebo učitele); program povoluje současnou instalaci doma i ve výuce.

Hodnocení: 4/5



Obrázek 3: Score Trainer – hra Zápis not

2.3 Score Trainer

Aplikace Score Trainer od společnosti Midimaster [7] je nabízena ve dvou edicích – Standard a Professional. Zde hodnotím demoverzi ScoreTrainer 8.0.11 (z 15. 3. 2013), která je funkčně identická s Professional edicí.[4]

2.3.1 Platforma

Aplikace je dle oficiálních webových stránek [7] určena pro systému Windows 98/NT/ME/2000/XP – je tedy možné, že nestabilita a některé popisované chyby nejsou tak markantní ve starších verzích Microsoft Windows. Je překvapivé, že přestože poslední verze vyšla v roce 2013 a aplikace je stále komerčně nabízena, není uvedena podpora novějších verzí systému.

Hodnocení: 1/5

2.3.2 Výukové možnosti

Score Trainer Professional obsahuje čtyři základní hry:

- hádání not,

[4]Demoverze je po dobu 30 dní a pouze volby tisk, ukládání a načítání nejsou přístupné.

- hádání slov zapsaných v notách,
- zápis not (obr. 3),
- poznávání řady not[5].

Každou z těchto her lze spouštět v režimu *Procvičování*, *Kvíz* nebo *Učit se*. Hry nejsou příliš zábavné a poutavé, pracují jen s notami a jejich poznáváním, ale pro jejich naučení mohou být přínosné.

Hodnocení: 1/5

2.3.3 Uživatelská přívětivost

Stažení zkušební verze z domovské stránky probíhá standardním způsobem, není nutné se registrovat. Web je uspořádán poměrně nepřehledně a po prvním načtení se zobrazí v německém jazyce; je však možné jej přepnout do angličtiny. Instalace probíhá běžným způsobem pomocí průvodce a již na začátku nabídne uživateli českou verzi.

Při spuštění se aplikace dotáže, zda mám licenční číslo na plnou verzi. Po zamítnutí se spustí průvodce, který slouží k nastavení programu. Je nutné zvolit MIDI zařízení, způsob značení not, používaný nástroj a další. Také je zobrazeno okno pro zadání jmen až pěti učitelů, přičemž není zcela zřejmé, z jakého důvodu a zda je toto nastavení povinné. Po potvrzení se dostávám do hlavní nabídky.

V hlavní nabídce není na první pohled vůbec zřejmé, jakou volbou se spouští výukové hry. V samotném středu okna obsahuje informaci o MIDI ovládání a seznam lekcí z pohledu obtížnosti a použitých not, což není na první pohled zcela zřejmé. Při najetí kurzoru na horní lištu ikon se zobrazují nápovědy v podobě bubliny. Ikony jsou vizuálně málo charakterizující, ale po chvíli práce s aplikací je možné se jejich význam naučit.

Po výběru každé výukové hry musím ve svém případě pokaždé znovu označovat, že chci používat Demo verzi. Při spuštění hry jsem nucen zvětšit okno aplikace, ta totiž obsahuje další vložená okna, která se svojí velikostí nevejdou na základní obrazovku.

Samotné hry jsou velmi neintuitivní a není na první pohled zřejmé, jak s nimi pracovat. Hodnocení je zobrazeno v průběhu každé hry a také v okně *Zobrazení nejvyššího score*.

Aplikace je vzhledově velmi zastaralá a neatraktivní, svým designem je podobná zpracování aplikací v Microsoft Windows 95.[6] Je také špatně odladěná – při pouhé změně velikosti okna nebo potvrzení dialogu někdy dochází k jejímu pádu, samotné hry se chovají nepředvídatelně.

Hodnocení: 1/5

[5] Ve hře označeno jako *Změna na posouvající se noty*.

[6] Nemohl jsem zjistit, od jakého roku byla vyvíjena, ale vzhled silně evokuje původ právě ve Windows 95, nebo dokonce Windows 3.1.

2.3.4 Rozhraní pro učitele

Aplikace neobsahuje uživatelské účty, ale je možné ji modifikovat v rámci nastavení lekcí – učitel může aplikaci nastavit na určitý počet správných not nutných ke splnění, zvolit časový limit, předznamenání, tóninu, klíč a také omezovat použité noty na vlastní výběr.

Hodnocení může zobrazovat pouze v podobě žebříčku nejlepších výsledků pro každou z lekcí.

Hodnocení: 2/5

2.3.5 Česká lokalizace

Webové stránky aplikace jsou v němčině a angličtině, samotná aplikace je již od instalace plně lokalizována do češtiny. Používá ale poměrně nestandardní názvy, např. *Realizuj* pro potvrzení operace, a komunikuje velmi zmateným způsobem, např. výraz *tak pomalý – moc pospícháš* v některé z her. Aplikace v průvodci při spuštění nabízí zobrazení noty *h* podle české notace.

Hodnocení: 3/5

2.3.6 Cena

- Standard – 19,90 €
- Professional – 74,90 €.

Hodnocení: 3/5

2.4 Musition 4 a Auralia 4

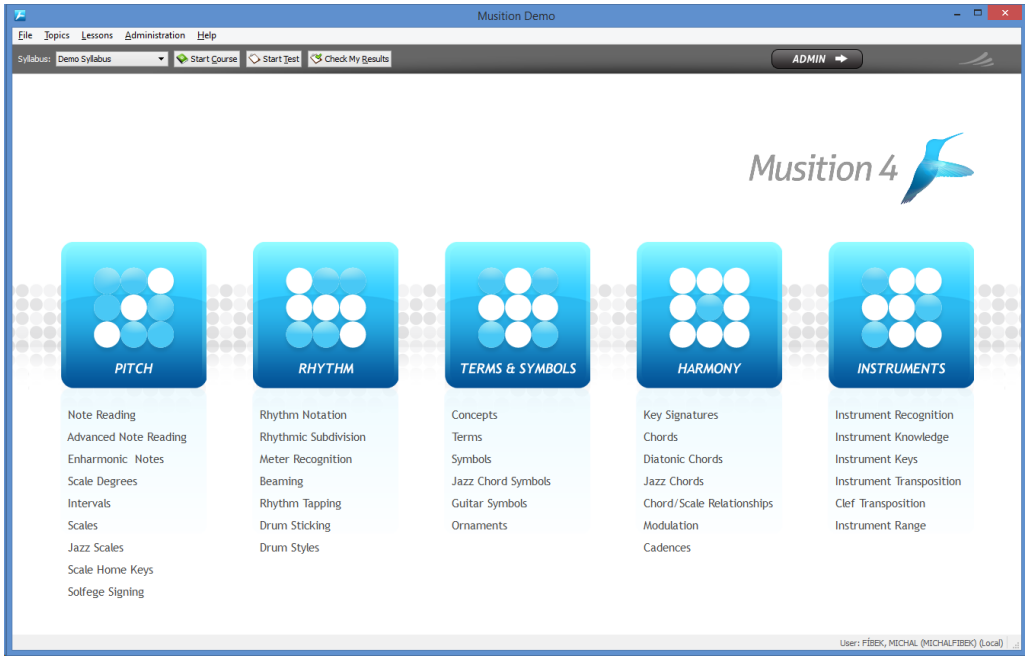
Protože jsou aplikace Musition 4 [8] a Auralia 4 [9] prodávány u společnosti Sibelius[7] společně v jednom balení, mají stejné ovládání, grafické rozhraní, pojetí lekcí i administraci a hlavní rozdíl je tak v samotném obsahu výukových her, zahrnuji je do srovnání jako jeden produkt.

Při průzkumu jsem zjistil, že jsou nabízeny také u Rising Software [11], kde je možné je zakoupit navíc samostatně nebo v cloudové licenci. Z oficiálních internetových stránek není zřejmé, která ze společností je oficiálním autorem aplikací.

2.4.1 Platforma

Musition i Auralia jsou k dispozici pro platformy Windows a Mac OS X. Programy nejsou přizpůsobeny pro dotykové ovládání, na tabletu se systémem Windows by bylo možné je s nižším komfortem používat.

[7] Společnost Sibelius patří od roku 2006 pod Avid. [10]



Obrázek 4: Musition 4 – výběr her

Společnost Rising Software nabízí také varianty pro iPhone a iPad, kde každá aplikace zastupuje jednu výukovou lekci a obsahově jsou velmi podobné variantám pro osobní počítače.

Hodnocení: 3/5

2.4.2 Výukové možnosti

Musition se zaměřuje především na hudební teorii a znalost not, Auralia pracuje s praktickým rozvojem hudebního sluchu a zpěvu.

Oba programy nabízí 5 základních skupiny výuky:

- výška,
- rytmus,
- termíny a symboly,
- harmonie,
- nástroje.

Každá z nich obsahuje v průměru 6–8 výukových her pro obě aplikace – Musition 34 her, Auralia 41 her. Tyto hry lze spustit také v podobě testů (skupina několika her za sebou).

Dále jsou k dispozici teoretické lekce hudební nauky, v názorné textové podobě s obrázky a ukázkami melodií (např. popis hudební stupnice u hry se stupnicemi). Lekce spolu s testy tvoří kompletní kurzy, které kombinují výuku s testování znalostí.

Hodnocení: 5/5

2.4.3 Uživatelská přívětivost

Obrázek 5: Auralia 4 – hra poznávání intervalů

Pro získání obou aplikací ve zkušební verzi ze stránek společnosti Sibelius je nutné zadat základní osobní údaje a e-mail; poté je nabídnuta ke stažení verze pro Windows a Mac. Instalace je pouze v anglickém jazyce, ale probíhá pomocí průvodce běžným způsobem.

Po prvním spuštění je zobrazen průvodce, který nabízí volbu jazyka (pouze anglický a německý), terminologii v pojmenování not (US a UK způsob) a přepnutí noty *h* podle české notace. Dále provede nastavením MIDI výstupu a zvuků a založení uživatelského účtu, kde je nutné zadat přihlašovací údaje včetně hesla a také výšku vlastního hlasu.

Po ukončení průvodce jsem vyzván k prvnímu přihlášení a aplikace zobrazí podrobné informace o obsahu demo verze. Poté se již zobrazí úvodní obrazovka s výběrem her (obr. 4). Po výběru každé z nich aplikace zobrazí dialog s výběrem obtížnosti, případně je možné zobrazit teoretické lekce z hudební oblasti, která se hrou souvisí.

Samotné hry probíhají v modálních oknech se strohou, jednoduchou grafikou. Princip většiny her je intuitivně pochopitelný a je zde vizuálně výrazně oddělena otázka od možných odpovědí. Úvodní obrazovka, která je na pozadí, má zašedlý text, ale obrázky jsou stále výrazné; to může působit mírně rušivě na koncentraci žáka.

Po každém jednotlivém úkolu ve výukové hře je zobrazeno jeho hodnocení. Zde má uživatel možnost pokračovat na další úkol, nebo hru ukončit. Po ukončení se zobrazí celkové hodnocení ke všem proběhlým úkolům s průměrnou úspěšností v procentech, nebo pouze počtem správně odpovězených otázek (v závislosti na typu hry).

Z úvodní obrazovky je možné spustit také test ve vybrané oblasti. V testech je žákovi předloženo několik různých her za sebou, každá po určitém počtu úkolů, bez okamžité odezvy o správnosti; výsledné hodnocení se žák dozví až na úplném konci testu v podobě procentní úspěšnosti.

Podobným způsobem pracují kurzy – ty jsou kombinací teoretických lekcí, které žáka uvádí do určité oblasti, a samotných testů, které okamžitě ověřují získané znalosti. V kurzech je v levém panelu zřetelně označený aktuální postup žáka (kolik položek kurzu splnil a kolik mu ještě zbývá).

Osobní hodnocení je možné zobrazit také z úvodní obrazovky – to je zobrazeno v podrobné tabulce se seznamem témat (výukových her) a hodnocením jejich úspěšnosti. Hodnocení lze exportovat do formátu PDF nebo vytisknout.

Hodnocení: 3/5

2.4.4 Rozhraní pro učitele

Obě aplikace nabízí komplexní uživatelskou administraci, kde se uživatelé dělí do skupin Žák, Učitel, Administrátor a Systémový administrátor. Dále je možné dělit žáky podle školních tříd.

Učitelé mají přístup k editaci sylabu a kompletního obsahu jednotlivých lekcí, mohou upravovat teoretické testy, včetně přidělování časů, kdy jsou testy otevřeny.

Aplikace generuje podrobná hodnocení z libovolné oblasti, a to na úrovni žáků i celých tříd.

Veškeré nastavení s uživateli je možné ukládat v rámci lokálního počítače, do vlastní databáze na lokální (školní) síti, využívající databázový server Firebird, nebo do hostované cloudové služby v případě zakoupení cloudové edice[8] programu. Databáze je vždy společná pro Musition i Auralia.

Hodnocení: 5/5

2.4.5 Česká lokalizace

Aplikace je pouze v anglickém a německém jazyce, včetně veškerých popisů a lekcí. Je však možné při prvním spuštění zvolit formát not podle české notace.

Hodnocení: 2/5

[8] Cloudová edice je dostupná pouze od společnosti Rising Software. [11]

2.4.6 Cena

U společnosti Sibelius jsou aplikace Auralia 4 a Musition 4 nabízeny společně:

- oba programy standardní edice – $249
- oba programy pro studenty – $159

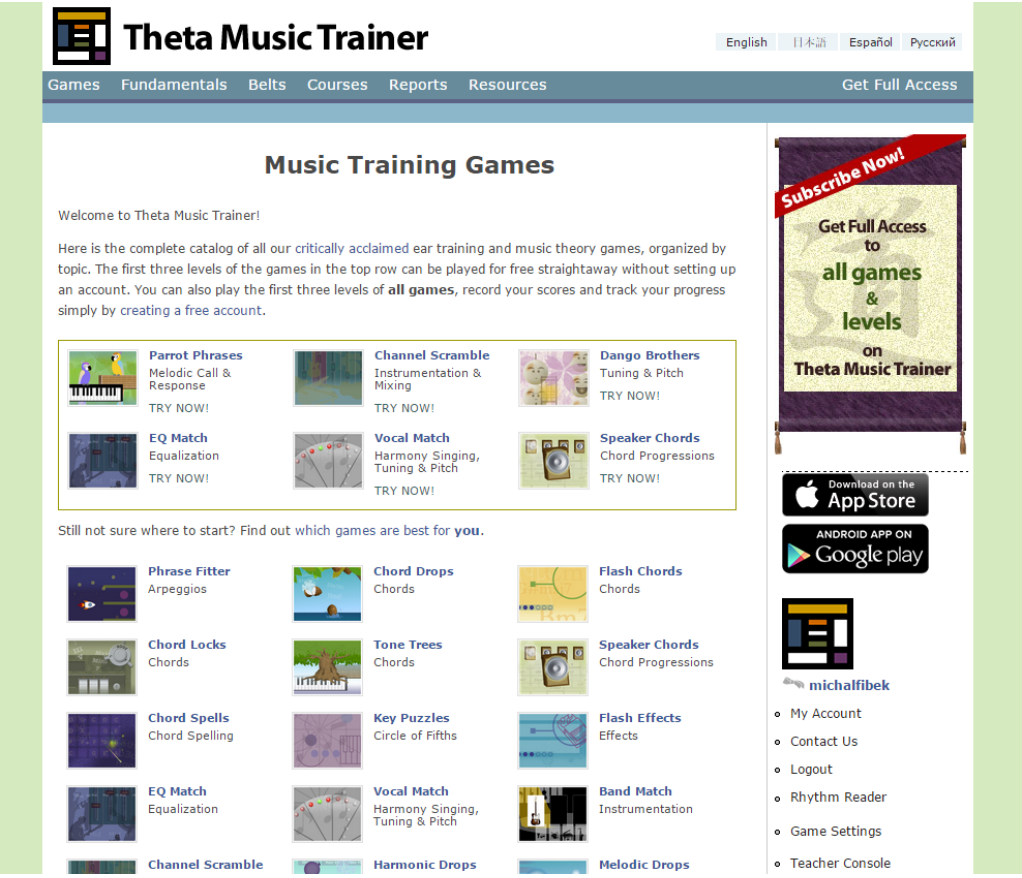
U Rising Software jsou ceny společného balení totožné, další ceny jsou:

- Auralia nebo Musition standardní edice – $149
- Auralia nebo Musition pro studenty – $99
- oba programy v cloudové edici na 1 rok (1 licence) – $49

Mobilní aplikace pro iPhone a iPad jsou nabízeny většinou v ceně $1,99 nebo $0,99 za samostatnou lekci.

Hodnocení: 2/5

2.5 Theta Music Trainer



Obrázek 6: Theta Music Trainer – výběr her

2.5.1 Platforma

Aplikace Theta Music Trainer [12] je dostupná online jako webová stránka a také jako nativní aplikace pro mobilní systémy iOS a Android. Ve své webové verzi, na kterou se v tomto srovnání zaměřím nejvíce, využívá technologii Flash.

Flash je využíván v hudebních ukázkách, hrách i podrobném hodnocení. Z tohoto důvodu je aplikace funkční také ve starších prohlížečích, ale nekompatibilní s většinou mobilních zařízení.

Mobilní aplikace obsahuje všechny hry, které jsou ve webové verzi. Uživatel má po spuštění možnost přihlásit se do stejného uživatelského účtu, jako ve webové verzi, nebo může aplikaci používat bez přihlášení.

Hodnocení: 4/5

2.5.2 Výukové možnosti

Celá aplikace je navržena stylem školy hrou a zaměřuje se především na typ her, které pracují s poslechem. Je zde 43 výukových her, které se dělí do 11 skupin podle tématu zaměření. Každá skupina pak obsahuje ještě detailnější rozdělení do podskupin. Mezi výukové hry patří například:

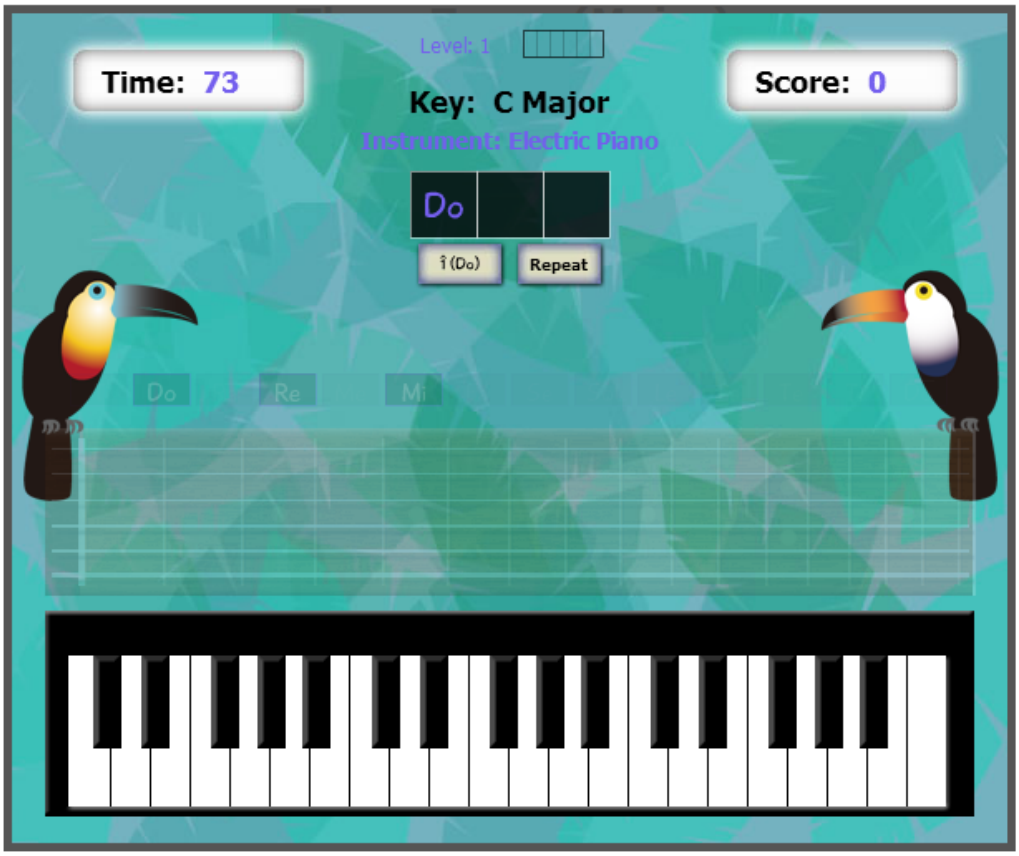
- poznávání tónů a not,
- poznávání harmonických a melodických intervalů,
- poznávání akordů,
- opakování melodie na základě poslechu,
- poznávání rytmu,
- hlasová cvičení - tóny,
- poznávání hudebních efektů,
- poznávání hudebních nástrojů,
- a další.

Každá z her pracuje s těmito oblastmi odlišným způsobem – některé vypadají téměř jako běžné počítačové hry, většina ale funguje vyváženějším stylem, zaměřeným také na výuku. U her, které jsou silně inspirované počítačovými hrami, je jejich vzdělávací účinnost velmi sporná.[9]

Také je možné spustit celý kurz, který se skládá z řady her na určité téma – jsou zde kurzy *Basic Ear Training*, *Music Theory* a *Sound & Audio*. Dále je zde zahrnuto několik kapitol hudební teorie i s ukázkami melodií, přičemž ke každé z nich je zobrazena skupina her, které slouží k tréninku.

Hodnocení: 5/5

[9]Například hra *Paddle Pitch*, která je inspirována starou počítačovou hrou *Breakout*, kde má hráč za úkol pomocí pohyblivé plošinky a kuličky, která se o ni odráží, postupně zničit všechny kostky.



Obrázek 7: Theta Music Trainer – hra poznávání posloupností

2.5.3 Uživatelská přívětivost

Pro spuštění libovolné hry není nutné stažení ani registrace. Vytvoření uživatelského účtu je ale výhodné, protože přináší možnost ukládání výsledků žáka a sledování postupu ve hře. Registrace proběhne okamžitě po zadání uživatelských údajů, není nutné potvrzení e-mailem a žák je hned přihlášen.

Každá hra se při prvním spuštění nejprve chvíli načítá, poté je zobrazeno herní menu. Zde je možné zvolit obtížnost (na škále 1–20), zobrazit nápovědu k dané hře a spustit hru v režimu cvičení nebo bodovaně. Po zvolení režimu je zahájena samotná hra. Hry jsou ztvárněny barevnou, kreslenou grafikou, vyznačují se jednoduchostí, názorností a použití je většinou intuitivní. U hlasových her, využívajících mikrofon, není potřeba žádná kalibrace – i méně kvalitní mikrofon funguje okamžitě a bez problémů.

Bodovaný režim je subjektivně náročný a bývá vymezen časovým limitem nebo maximálním počtem pokusů pro danou hru. Hodnocení je zobrazeno v průběhu samotné hry v podobě bodového hodnocení. Body jsou zobrazeny v případě úspěšného splnění také po ukončení hry, včetně 5 nejlepších výsledků hráče a jeho postup k dalšímu pásku. Pokud si hráč nevedl v určité oblasti příliš dobře, některé z her

mu zobrazí také konkrétní doporučení ke zlepšení.[10]

Hodnocení je celkově děleno na body a celkový počet úrovní, které žák splnil. Na základě počtu splněných úrovní je každému uživateli přiřazen pásek – od bílého po černý, celkem 10 barev. V aplikaci je možné otevřít podrobný žebříček, který zobrazuje všechny uživatele a jejich bodové hodnocení. Také si uživatel může zobrazit osobní žebříčky pro každou z trénovaných her, nebo vlastní postup, kde mu aplikace vykreslí jeho poslední výsledky ve vybrané oblasti během časového období.

Hodnocení: 4/5

2.5.4 Rozhraní pro učitele

Aplikace obsahuje podporu uživatelských účtů pro učitele. Při registraci učitel zvolí školu, pod kterou patří on i jeho žáci. Poté má možnost žákům vytvářet uživatelské účty. To lze zadáním jejich e-mailových adres, na které jim přijde pozvánka, nebo zadáním seznamu názvů uživatelských účtů, které budou vytvořeny.

Učitel má možnost upravit žákům nastavení, včetně změny jména a hesla. Žáky je možné přiřazovat do tříd, které učitel v aplikaci vytvoří. Každému uživateli lze přidělovat hry, které dostane za úkol na konkrétní datum – k úkolu může uživatel psát instrukce a zvolit upozornění žáka na e-mail. Podobně je možné přiřazovat žákům vybrané kurzy, což jsou skupiny výukových her.

Učitel má možnost prohlížet hodnocení na základě uživatelských účtů žáků – zobrazen je počet žákem hraných her, celkový počet úrovní, které splnil a celkový počet získaných bodů. Také může zobrazit podrobný seznam záznamů, jaké hry hrál žák naposledy a jakého výsledku v nich dosáhl.

Učitel také může prohlížet žebříček nejlepších žáků pro danou školu, opět s celkovým počtem bodů a počtem splněných úrovní.

V možnostech nastavení v rámci celé školy je změna značení hudební stupnice, změna značení římských čísel, vypnutí zvuků pro správnou nebo špatnou odpověď a vypnutí herních tipů.

Hodnocení: 4/5

2.5.5 Česká lokalizace

Aplikace nepodporuje český jazyk a není zde ani možnost nastavit zobrazení noty *h* podle české notace.

Hodnocení: 0/5

[10] Například u intervalů dostane hráč seznam problematických intervalů a může okamžitě spustit novou hru, která se na ně zaměřuje.

2.5.6 Cena

Základní verze aplikace s přístupem do prvních tří úrovní obtížnosti u každé hry je zdarma. Pro přístup do všech 20 úrovní je nutné zakoupení některé z licencí, které jsou pouze na bázi předplatného.

Jednouživatelské licence:

- měsíční předplatné – $7,95/měsíc
- roční předplatné – $4,08/měsíc

Školní licence:

- Theta Studio (50 účtů) – $99/rok
- Theta School (200 účtů) – $199/rok

Hodnocení: 5/5

2.6 Závěrečné porovnání

Při hledání výukových aplikací, které by byly opravdu použitelné pro české prostředí (především podporou českého formátu not), jsem nedokázal najít další zástupce vhodné ke srovnání. Trh je zaměřený především na ostatní státy, protože značení hudební abecedy, používané v České republice a okolních státech, je silně minoritní. Nalezl jsem mimo jiné soubor multiplatformních hudebních aplikací *Foriero* [13] od českého autora, ale i přesto, že webové stránky jsou v českém jazyce, aplikace nabízí výuku pouze v angličtině s anglickou notací.

2.7 Shrnutí vlastností

I přes relativně malý počet her je **EarMaster** uživatelsky dobře použitelným programem, který nabízí výuku spíše tradičnější formou. Všechny hry pracují s notami, což může být nevýhodou, pokud hledáme zábavnější přístup. Grafické rozhraní je strohé a málo „hravé“, aplikace je tak vhodná spíše pro starší ročníky. Česká lokalizace zde chybí, dá se však očekávat, že bude doplněna. Pořizovací náklady jsou poměrně nízké, především při zakoupení cloudové licence ve větším počtu.

Vzhledem ke své ceně je aplikace **Score Trainer** velmi špatnou volbou, přestože je stále nabízena českým distributorem. Od doby svého vzniku se změnila nejspíše jen nepatrně, má velmi zastaralé uživatelské rozhraní, málo her a s absencí uživatelských účtů není vůbec vhodná pro použití ve škole. Jedinou výhodou je česká lokalizace, ta však není ideální a celkovému dojmu příliš nepomáhá.

Musition a **Auralia** nabízí vysokou přidanou hodnotu v teoretických popisech jednotlivých hudebních oblastí, které nabízí samostatně, ale také v rámci kurzů. Celkově jsou kurzy velmi dobře koncipované a mohou být zajímavou přidanou hodnotou pro domácí výuku. Aplikacím bych vytknul strohé uživatelské rozhraní a příliš

tradiční přístup k výuce, který žákům nenabízí více kreativity a hravosti v rámci výukových her – jsou tak vhodné spíše pro vyšší ročníky základních škol a střední školy. Vyšší pořizovací cena i v případě cloudové licence je zde poměrně adekvátní, vzhledem k hodnotě, kterou tato dvojice aplikací nabízí; větší překážkou tak může být především rozhraní v anglickém jazyce.

Theta Music Trainer je aplikace navržená pro velmi hravý a zábavný styl výuky. Přirozeně motivuje žáka ke zlepšování pomocí systému pásků, které žák získává za každou splněnou úroveň hry. Výhodou je dostupnost odkudkoliv díky platformě webové aplikace, mezi klady patří také mobilní aplikace, které obsahují stejnou nabídku her a společný účet. Největší nevýhodou je chybějící podpora českého stylu značení not, takže výukové hry, které pracují přímo s notami, nejsou příliš použitelné pro českou základní školu. Dalším problémem je použitá technologie Flash, která omezuje použití webové aplikace pouze na podporované prohlížeče s tímto doplňkem. Cenově tato aplikace vychází při zakoupení hromadné licence pro školu relativně výhodně.

2.7.1 Porovnání na základě metriky

Ve srovnání všech aplikací používám jako hlavní měřítko ceny cloudovou verzi, pokud ji aplikace nabízí – předpokládám, že je pro většinu škol nejvýhodnější využívat program v hromadných licencích. Všechny ceny jsem normalizoval přepočtem na jednoho uživatele; u většiny multilicenčních programů je nutné pořídit licenci pro nejméně 100 uživatelů, než bude této ceny dosaženo.

Tabulka 1: Porovnání základních parametrů aplikací

Název programu	**Platforma**	**Výukové možnosti**	**Česká lokalizace**	**Cena**
Earmaster Pro 6	Windows, Mac	14 her	ne	4,99 €/rok 1 licence
ScoreTrainer Pro 8	Windows	4 hry	ano	74,90 € 1 licence
Musition 4 a Auralia 4	Windows, Mac, iOS	34 + 41 her	ne	$49/rok 1 licence
Theta Music Trainer	Web (Flash), Android, iOS	43 her	ne	$0,995/rok 1 licence

2.7.2 Hodnocení

Při srovnání aplikací na základě udělených bodů a stanovené váhy jednotlivých kritérií vychází následující pořadí:

1. Theta Music Trainer – 67 bodů
2. Musition 4 a Auralia 4 – 66 bodů
3. Earmaster Pro 6 – 62 bodů
4. ScoreTrainer Pro 8 – 39 bodů

Toto srovnání ukazuje vysokou bodovou vyrovnanost prvních dvou aplikací. Je však nutné brát na zřetel, že požadavek na českou lokalizaci, který jsem nastavil jako nejdůležitější kritérium, nebyl u aplikace *Theta Music Trainer* vůbec splněn. Proto je nutné zvážit ze strany učitele veškerá rizika, které to může ve výuce obnášet a na základě nich se individuálně rozhodnout, jaké aplikaci dá přednost. Podrobné srovnání aplikací je uvedeno v příloze A.

3 Analýza požadavků

Pro správný návrh aplikace nejprve analyzuji typického uživatele a zjistím dílčí činnosti, které se k jeho používání aplikace budou vázat. Podle nich rozeberu funkční i nefunkční požadavky. Při určování požadavků budu spolupracovat s učitelem hudební výchovy Mgr. Martinem Grobárem, který vyučuje na Základní škole Tomáše Garrigua Masaryka Blansko.

3.1 Analýza uživatele

Typický uživatel se bude dělit do dvou primárních skupin:

- Žák navštěvující 3.–8. třídu ZŠ, kterého nebaví učení běžnou cestou a hledá nějakou zábavnou alternativu.
- Učitel ve věku od 20 let, který hledá jednoduchý způsob, jak vzdělávat své žáky v hudební nauce.

3.1.1 Účel aplikace

Rozvinout u žáka hudební sluch nenásilnou a jednoduchou formou. Vyhodnocovat dosažené úspěchy objektivní metrikou.

3.1.2 Dílčí činnosti

- **Žák** – výběr hry, hraní hry, prohlížení osobního hodnocení, volba obtížnosti, přizpůsobení profilu.
- **Učitel** – prohlížení hodnocení žáků, přidávání a odebírání žáků do aplikace, správa podkladů použitých k výuce.

3.2 Funkční požadavky

Požadavky obecně definují chování, které bude aplikace vykazovat pod určitými podmínkami. *Požadavek je specifikace toho, co by mělo být implementováno. Popisuje chování systému nebo jeho vlastnost. Požadavky mohou určovat omezení vývojového procesu systému* (volný překlad) [14, s. 6]. Funkční požadavky jsou obecně chápány jako skupina požadavků, které jsou vnímány z pohledu uživatele.

3.2.1 Uživatelské účty

Žák i učitel se budou moci přihlásit do aplikace pod svým uživatelským účtem. Každý účet bude evidovat základní informace o uživateli. Žákovské účty budou ukládat informace o školní třídě, kterou navštěvuje, aby bylo možné žáka přesně

identifikovat. Každý uživatel bude mít možnost změnit své heslo a zvolit osobního *avatara*[11].

3.2.2 Používání her

Uživatelé budou mít přístup k jednotlivým výukovým hrám. Každá hra bude obsahovat stručné vysvětlení jejího použití a po jejím splnění zobrazí uživateli hodnocení. Každý uživatel bude mít možnost zjistit zpětně své celkové hodnocení.

3.2.3 Učitel – Prohlížení hodnocení žáků

Učitel bude mít možnost prohlížet osobní hodnocení žáka na základě jeho používání výukových her v aplikaci. Bude moci porovnávat jednotlivé žáky mezi sebou a zjišťovat úspěchy žáků podle tříd.

3.2.4 Učitel – Přidávání a odebírání žáků

Učitel bude mít možnost vytvářet nové uživatelské účty žákům. Bude moci v aplikaci vytvářet nové třídy a jednotlivým žákům nastavovat, do které třídy patří.

3.2.5 Učitel – Správa herních podkladů

Učitel bude mít možnost ovlivňovat obsah her jejich nastavováním nebo nahráváním příslušných podkladů. Podklady bude moci ovlivňovat i zpětně po jejich uložení.

3.3 Nefunkční požadavky

Nefunkční požadavky zahrnují další vlastnosti aplikace, které se netýkají přímo funkcí z pohledu uživatele. Nefunkční požadavky mohou zahrnovat chování systému v určitém prostředí, jako například platformu, přenositelnost, kompatibilitu, výkon a další [14, s. 10].

Aplikace by měla mít jednoduchý a moderní design, který by měl být zajímavý i z pohledu žáka. Písmo by mělo být dostatečně velké a čitelné.

Měla by být naprogramována co nejvíce platformě nezávisle, a to z pohledu zařízení (mobilní telefon, tablet, osobní počítač), prohlížeče (fungování v nejpoužívanějších prohlížečích) i ovládání (podpora klávesnice a myši i dotykového ovládání). Odezva aplikace by měla být dostatečně rychlá.

Aplikace by měla být naprogramována bezpečně, být odolná vůči útokům a chránit uživatele před zneužitím jejich účtů. Měla by mít čitelný, vhodně okomentovaný zdrojový kód. Kód aplikace by měl používat vhodné návrhové vzory, aplikace by měla být modulární pro možnosti dalšího rozšiřování. Pro možnosti vyhodnocování testování a ladění by měla zaznamenávat dostatečně podrobné informace o uživatelích.

[11] Avatar je postava, která reprezentuje virtuální podobu uživatele

4 Návrh řešení

Na základě analýzy požadavků v kapitole 3 navrhnu základní strukturu aplikace, v níž se bude uživatel pohybovat. Poté rozeberu konkrétní hry, které budou v aplikaci obsaženy a na základě nich rozeberu jednotlivé složky aplikace, jako je modul pro hodnocení a administrace.

4.1 Uživatelské rozhraní

Jak je patrné z analýzy uživatelů, navrhovaná aplikace musí klást maximální důraz na jednoduchost, aby se s jejím prostředím bez problému naučily pracovat i malé děti navštěvující první stupeň základní školy. Proto je nezbytné navrhnout maximálně přívětivé a intuitivní uživatelské rozhraní.

4.1.1 Logická struktura aplikace

Jak vyplývá ze statistik, v dnešní době jsou žáci již od útlého věku zvyklí používat různá mobilní zařízení.[12] Právě mobilní operační systémy používají velmi často navigaci postavenou na vzoru *Hub and spoke*.

Hub and spoke vypisuje všechny hlavní části stránky nebo aplikace na domovské stránce, tzv. *hubu*. Uživatel klikne nebo prstem označí danou volbu, provede požadovaný úkon a vrátí se zpět do hubu pro pokračování jinou možností. Obrazovky *spoke* jsou naopak zaměřené konkrétně na vykonání určitého úkonu a velmi opatrně zachází s prostorem.[13]

Vzhledem k tomu, že je mým cílem navrhnout aplikaci multiplatformně a tak, aby plně podporovala přístroje s dotykovým rozhraním a připomínala svojí jednoduchostí jejich ovládání, volím právě vzor Hub and spoke. Jako hub bude sloužit hlavní obrazovka, ve které se uživatel dostane jedním gestem na obrazovku s vybranou hrou nebo do stránky s žebříčkem jejího hodnocení.

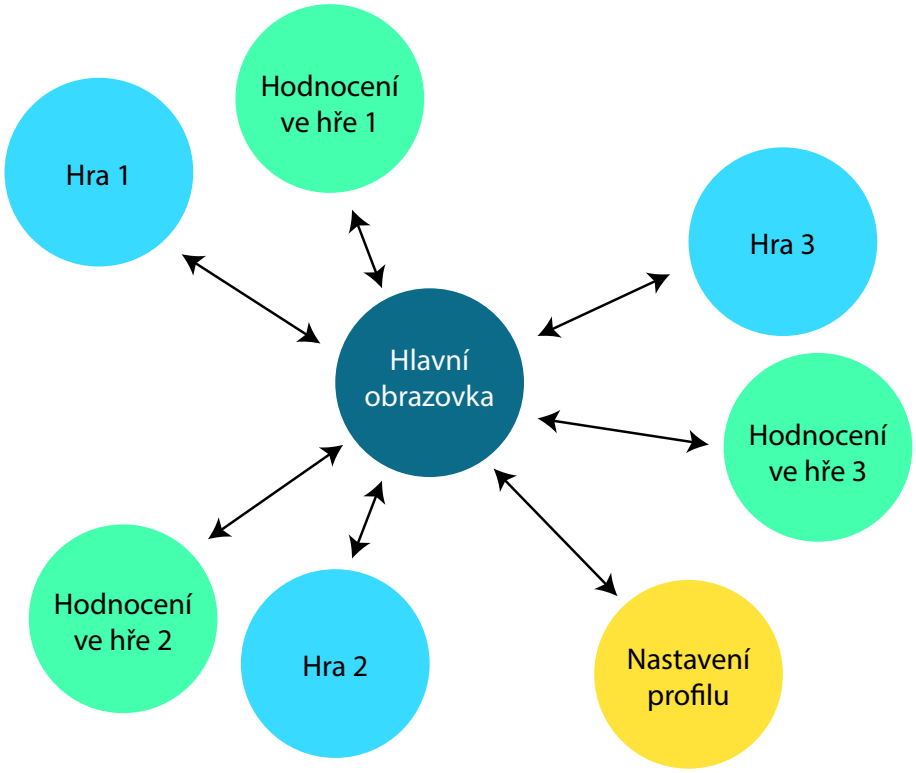
4.1.2 Hlavní obrazovka

Při návrhu hrubé podoby aplikace pracuji s principem *mobile first*. Mobile first [17] je založen na vývoji ve stylu tzv. *progressive enhancement* [18] – zaměřuji se primárně na obsah, kostru navrhnu tak, aby byla po implementaci okamžitě použitelná na mobilních zařízeních, teprve poté ji responzivně přizpůsobím větším úhlopříčkám.

Když vycházím z potřeb uživatele, v centru jeho pozornosti jsou samotné hry, které bude chtít po otevření aplikace spouštět. Tyto tedy umístím ve výrazně viditelných blocích (odlišených od ostatních elementů jak velikostí, tak barvou) do

[12] Očekává se, že v roce 2020 bude více než 90 % dětí starších 6 let používat mobilní telefon [15].

[13] *[...] this architecture lists all the major parts of the site or app on the home screen, or "hub." The user clicks or taps through to them, does what she needs to do, and comes back to the hub to go somewhere else. The "spoke" screens focus tightly on their jobs, making careful use of space – they may not have room to list all the other major screens.* [16, s. 80]



Obrázek 8: Diagram Hub and spoke

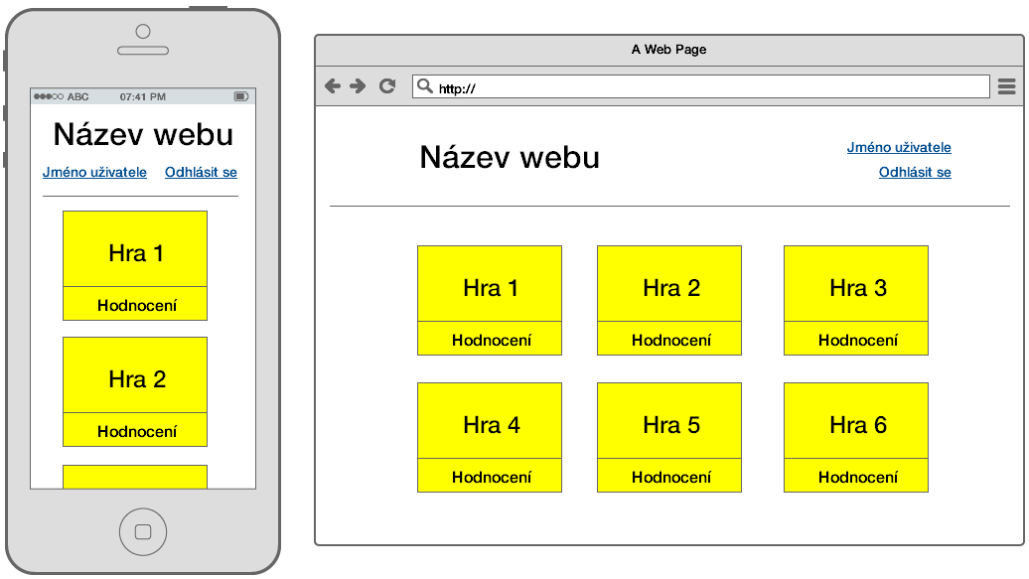
největšího prostoru *viewportu* – k nim vizuálně svážu pole odkazující na hodnocení v dané hře.

Protože bude z pohledu učitele žádoucí mít o každém z uživatelů unikátní přehled, musím zavést uživatelské účty, a tedy i přihlašování a odhlašování. Dále mu chci předat informaci o tom, že je přihlášeným právě on zobrazením jeho jména, a samotné jméno bude také sloužit jako odkaz do jeho uživatelského profilu, což by mělo být vnímáno implicitně.

Pro návrat z jiné obrazovky zpět na hlavní bude sloužit nadpis, který využívá vzoru *Escape hatch*.[14] Právě na mobilních zařízeních, kde je málo prostoru pro velké množství ovládacích prvků, je tento vzor obzvláště důležitý a ukazuje uživateli známý vizuální prvek, který jej vždy přivede zpátky na začátek, ať už je kdekoliv.

Všechny tyto prvky zpracuji do vizuální podoby pro mobilní obrazovku, kterou následně přizpůsobím také pro běžnou obrazovku webového prohlížeče na počítači (návrh na obr. 9).

[14] *On each screen that has limited navigation options, place a button or link that clearly gets the user out of that screen and back to a known place. [...] Websites often use clickable site logos as home-page links, usually in the upper left of a page. These provide an Escape Hatch in a familiar place, while helping with branding.* [16, s. 104]



Obrázek 9: Mockup mobilní a desktopové podoby aplikace

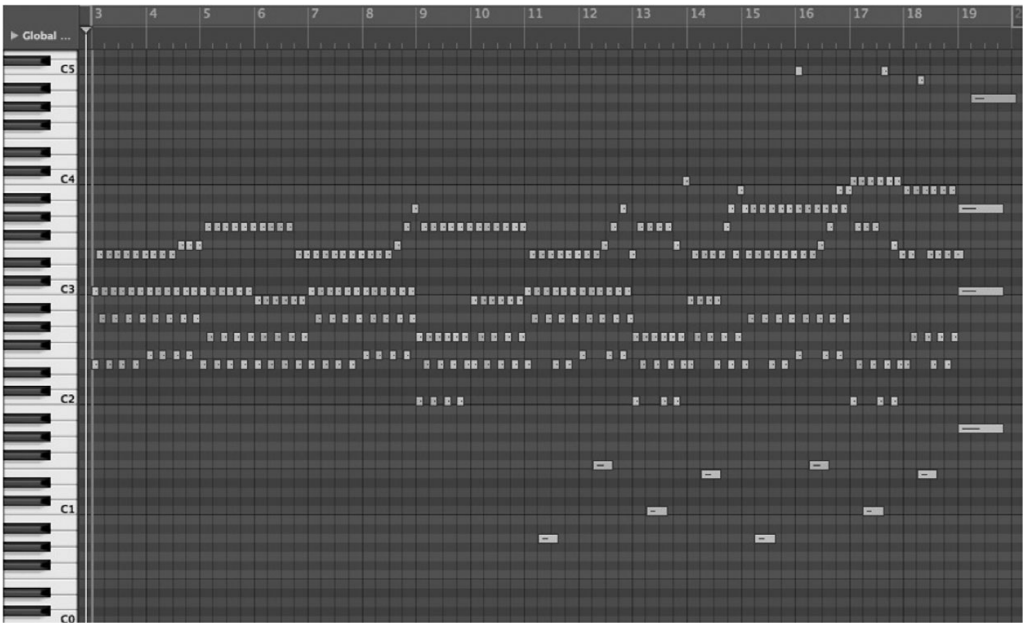
4.2 Výukové hry

Než postoupím dále v návrhu samotné aplikace, musím mít jasnou představu o výukových hrách, které budu implementovat. Z aktuálních poznatků jednoznačně vyplývá, že pro rozvoj hudebního sluchu, kreativity a obecně vyjadřovacích schopností v oblasti hudby hraje velký význam interaktivita a hravost [19, 20, 21]. Interaktivní výukové hry mohou prokázaně pomáhat rozvíjet hudební schopnosti, znalosti a jsou pro žáky z hlediska stylu výuky mnohem zajímavější.[15]

Notový zápis sice není zcela překonaný, ale v rámci hudební nauky může pro spoustu žáků představovat těžce překonatelnou překážku, protože je na první pohled velmi neintuitivní a náročný na naučení. Z tohoto důvodu se čím dál častěji preferuje ve výuce zápis melodie v názornější podobě, např. ve formě *piano roll* (obr. 10), který skládá jednotlivé tóny do mřížky, kde horizontální délka udává délku tónu, vertikální pozice jeho výšku.

Celkově tedy budu svoji aplikaci uvažovat v rovině školy hrou, tedy nabízet výuku primárně intuitivním a herním způsobem, který je pro žáka více poutavý, ale přitom jej podvědomě vzdělává a posouvá ve vnímání hudby – tedy například v rozlišování tónů, jejich délek, výšek, tvorby melodií, poznávání vzorů apod.

[15] *Results indicate that the games may help to develop some music skills and knowledge and that the games are of high interest and importance to students. The music teachers acknowledged the potential of interactive music video games to be incorporated into traditional music curriculum but they each expressed a belief that ideally the technology needs further development, including a greater capacity to compose and create using the programs. These findings suggest that, based on constructivist learning theories, there may be a place for these games in music education.* [20]



Obrázek 10: Zobrazení Piano roll [21]

4.2.1 Rámcový vzdělávací program

Pro použití aplikace na ZŠ a jako přímou podporu hudební výuky je nutné pracovat s požadavky, které bývají kladeny pro tento stupeň vzdělání. V Rámcovém vzdělávacím programu pro základní vzdělávání [22, s. 64–65], který předepisuje obecné požadavky, které by měl žák splňovat, jsou uvedeny mimo jiné následující body (výběr z textu pro 1. i 2. stupeň):

- *rozlišuje jednotlivé kvality tónů, rozpozná výrazné tempové a dynamické změny v proudu znějící hudby,*
- *rozpozná v proudu znějící hudby některé hudební nástroje, odliší hudbu vokální, instrumentální a vokálně instrumentální,*
- *rozpozná hudební formu jednoduché písně či skladby,*
- *rozpozná v proudu znějící hudby některé z užitých hudebních výrazových prostředků, upozorní na metrorytmické, tempové, dynamické i zřetelné harmonické změny,*
- *reprodukuje na základě svých individuálních hudebních schopností a dovedností různé motivy, témata i části skladeb, vytváří a volí jednoduché doprovody, provádí jednoduché hudební improvizace,*
- *realizuje podle svých individuálních schopností a dovedností písně a skladby různých stylů a žánrů,*

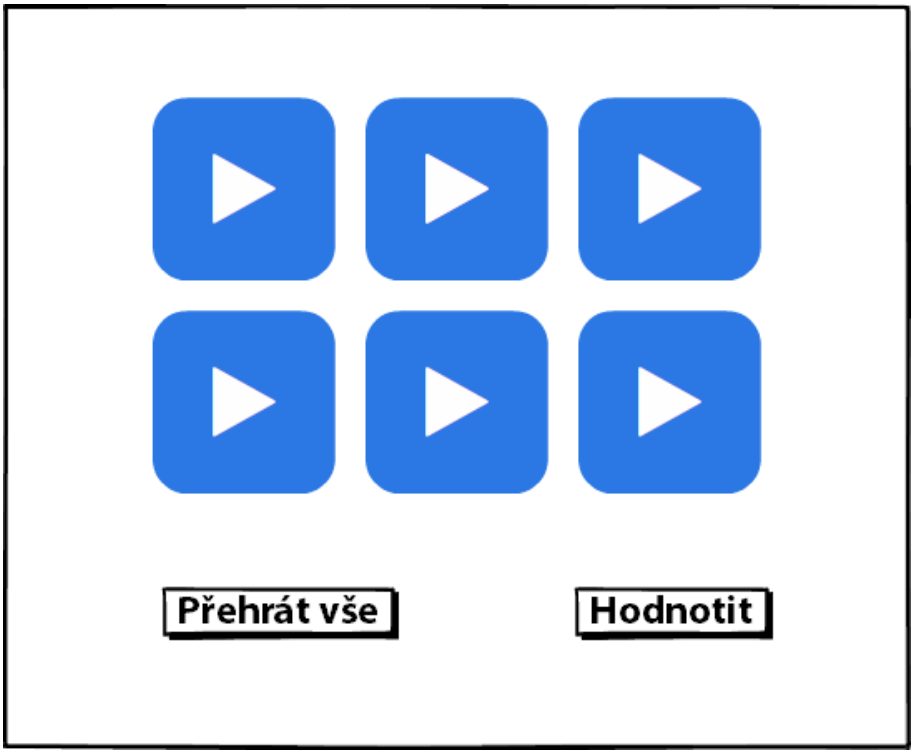
- *orientuje se v proudu znějící hudby, vnímá užité hudebně výrazové prostředky a charakteristické sémantické prvky, chápe jejich význam v hudbě a na základě toho přistupuje k hudebnímu dílu jako k logicky utvářenému celku,*
- *zařadí na základě individuálních schopností a získaných vědomostí slyšenou hudbu do stylového období a porovnává ji z hlediska její slohové a stylové příslušnosti s dalšími skladbami.*

Z toho vyplývá, že je žádoucí implementovat hry, které nějakým způsobem přispívají k rozvoji vnímání tempa a rytmu znějící hudby, rozlišování hudební formy, výrazových prostředků, orientaci v proudu znějící hudby a jiné.

Všechny tyto požadavky mají společný znak – primárně pracují s rozvojem vnímání hudby z hlediska poslechové stránky. Právě na tuto stránku se tedy od této chvíle primárně zaměřím ve své práci.

Při návrhu všech následujících her z hlediska herního designu i didaktického hlediska budu spolupracovat s Mgr. Martinem Grobárem, jehož náměty a postřehy z pedagogické praxe zanesu do konečné podoby her.

4.3 Hra Melodické kostky



Obrázek 11: Melodické kostky – návrh hry

První hrou, která pracuje s rozvojem hudebního sluchu, nazvu *Melodické kostky*. Bude sloužit k rozvíjení schopnosti žáka vnímat melodii a zlepšování jeho hudební paměti.

V každém herním kole bude jedna skladba, která bude rozstřižena na určitý počet částí. Každá tato část bude reprezentována jednou kostkou, přičemž všechny tyto kostky budou vzájemně promíchány. Úkolem žáka bude kostky poskládat opět do správného pořadí, aby vytvořil původní posloupnost melodie a splnil tak zadané kolo. Žák bude mít možnost přehrávat jak jednotlivé kostky, tak celou sestavenou melodii.

4.3.1 Variace obtížnosti

V závislosti na zvolené obtížnosti bude skladba rozstřižena na určitý počet částí – např. 3 části pro nízkou obtížnost, 6 částí pro vysokou obtížnost.

4.3.2 Hodnocení

Hodnocení bude probíhat na základě celkového času pro vyřešení úlohy a počtu přesunutí kostek, které žák pro vyřešení udělal. K výpočtu celkového počtu bodů bude sloužit následující vzorec:

$$\text{body} = \text{maxbodů} - \text{časová penalizace} - (\text{počet kroků} \cdot 30) \tag{1}$$

Limitující čas pro počítání penalizace bude pro tuto hru 5 minut, konkrétní vzorec výpočtu časové penalizace je uveden v kapitole 4.7.3 (vzorec 9). Maximální počet bodů bude stanoven na hodnotu 1000 bodů, od které budou penalizace odečítat další body.

4.3.3 Vizuální podoba

Vizuální stránka hry je nastíněna na obr. 11 – herní pole s kostkami by mělo představovat dominantní prvek celé hry, přičemž bude nutné nějakým způsobem odlišit právě hranou kostku. To bude možné například pulzováním, jejím zvětšením nebo například změnou barvy – důležité je, aby žák dostal tuto informaci, na základě čehož se může lépe rozhodovat v samotné hře při skládání melodie.

Hodnocení spustí žák sám kliknutím na tlačítko *Hodnotit* – toto je nutné pro alespoň částečnou eliminaci nahodilého promíchávání kostek; zároveň se tímto způsobem může sledovat hypoteticky rostoucí hudební sebejistota žáka (při vyšší jistotě správnosti svého řešení by měl stále méně častěji slepě klikat na tlačítko pro hodnocení). Dále zde bude tlačítko pro přehrání celé melodie v aktuálně sestaveném pořadí.

4.3.4 Administrace

Administrace učitele pro tuto hru bude obsahovat formulář pro nahrávání a ukládání souborů s hudebními podklady. Jak jsem zjistil při konzultaci s Mgr. Martinem Grobárem, aby hra fungovala hudebně správně, bude nutné skladbu dělit podle melodie – nemůže dojít k rozdělení v polovině tónu. Toto je problém, který lze vyřešit dvěma způsoby – algoritmem pro spektrální analýzu zvukové nahrávky a následným automatickým dělením skladby, nebo systémem pro vytváření uživatelských značek pro daná místa předělů. Z důvodu větší kontroly nad obsahem zvolím právě ruční vytváření značek učitelem, který dále rozeberu v kapitole 4.8.1.

4.4 Hra Pexeso

Obrázek 12: Pexeso – návrh hry

Další hra, která bude sloužit pro trénování hudební paměti a vnímání melodie, bude založena na principu klasické hry pexeso. Žák bude mít za úkol „otáčením“ jednotlivých herních polí zjišťovat, která dvě pole k sobě hudebně patří – půjde

o jednu skladbu, rozdělenou na dvě části. Jeho úkolem tedy bude označit za sebou obě pole z páru, navíc v hudebně správném pořadí.

4.4.1 Variace obtížnosti

Volba obtížnosti bude pracovat na základě celkového počtu herních polí k vyřešení, které žákovi aplikace předloží – s vyšší obtížností bude jejich počet růst, čímž se bude zvyšovat i kognitivní zátěž, která bude na žáka kladena.[16]

4.4.2 Hodnocení

Hodnocení bude probíhat na základě celkového času pro vyřešení úlohy a celkového počtu otočení polí žákem; ve výpočtu je zahrnut také počet párů herních polí. K výpočtu celkového počtu bodů bude sloužit následující vzorec:

$$\text{body} = \text{maxbodů} - \text{časová penalizace} - ((\text{počet kroků} - \text{počet párů}) \cdot 12) \quad (2)$$

Limitující čas pro počítání penalizace bude pro tuto hru proměnlivý v závislosti na celkovém počtu párů:

$$\text{limitující čas} = (\text{počet párů} \cdot 60 \cdot 1000) \cdot 0{,}8 \quad (3)$$

Konkrétní vzorec výpočtu časové penalizace je uveden v kapitole 4.7.3 (vzorec 9). Maximální počet bodů bude stanoven na hodnotu 1000 bodů, od které budou penalizace odečítat další body.

4.4.3 Vizuální podoba

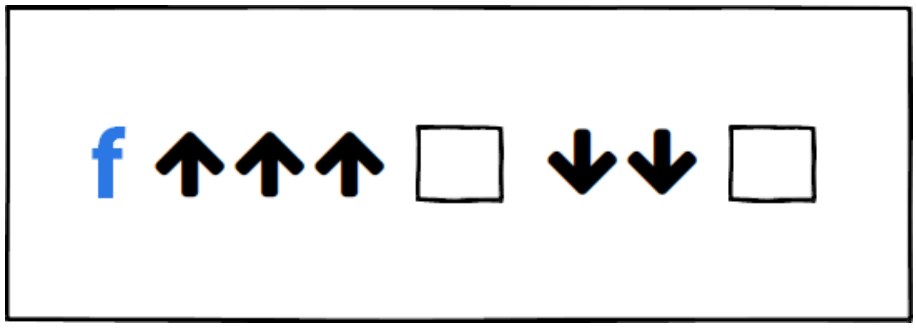
U této hry nebude třeba doplňovat rozhraní dalšími ovládacími prvky, kromě samotných herních polí pexesa. Ta budou po označení přehrávat konkrétní část skladby, kterou pole reprezentuje. Opět bude nutné pro vyšší názornost zvýraznit aktivní pole, jehož melodie se v danou chvíli přehrává. Úspěšné vyřešení pozná hra bez potvrzování uživatelem sama a zobrazí potvrzovací dialog. Vizuální návrh je znázorněn na obr. 12.

4.4.4 Administrace

Nahrávání hudebních souborů bude probíhat učitelem pomocí formuláře. Dospěli jsme k rozhodnutí, že zde, na rozdíl od *Melodických kostek*, není nutné rozdělovat jednotlivé hudební podklady melodicky, ale postačí prosté dělení na dva časově stejně dlouhé úseky. V této hře totiž nebude klíčová přesná návaznost jednotlivých částí melodie, ale žák bude nucen vnímat melodii jako celek.

[16] Vyšší kognitivní zátěž se projeví ve vyšším počtu asociací melodií k polím, které si žák bude muset v daný moment pamatovat.

4.5 Hra Krokování not



Obrázek 13: Krokování not – návrh hry

Oproti ostatním hrám bude *Krokování not* sloužit k jednoduchému učení posloupnosti hudební abecedy.[17]

Žákovi bude zobrazena výchozí nota, následovaná množinou šipek nahoru nebo dolů, jejichž směr a počet udává směr a velikost posunu v hudební abecedě (např. $f \uparrow \uparrow \uparrow h$). Za každou skupinou šipek bude následovat pole, do kterého žák vyplní písmeno hudební abecedy po určeném posunu. Jeho úkolem bude vyplnit správně všechna tato pole.

4.5.1 Variace obtížnosti

Variace obtížnosti bude na základě počet šipek nahoru nebo dolů, které budou vygenerovány pro jeden posun.

4.5.2 Hodnocení

Hodnocení bude probíhat na základě celkového času pro vyřešení úlohy a počtu špatně zadaných písmen. K výpočtu celkového počtu bodů bude sloužit následující vzorec:

$$\text{body} = \text{maxbodů} - \text{časová penalizace} - (\text{počet špatných zadání} \cdot 30) \qquad (4)$$

Limitující čas pro počítání penalizace bude pro tuto hru 5 minut, konkrétní vzorec výpočtu časové penalizace je uveden v kapitole 4.7.3 (vzorec 9). Maximální počet bodů bude stanoven na hodnotu 1000 bodů, od které budou penalizace odečítat další body.

[17] Ve formátu hudební abecedy, který se používá mimo jiné v České republice, jsou noty zapsány v pořadí *ahcdefg*

4.5.3 Vizuální podoba

Grafická podoba je nastíněna na obr. 13 – hra nebude obsahovat žádné ovládací prvky, kromě názvu první noty, šipek a pole pro zadávání odkrokovaných not uživatelem. Každé z těchto polí bude vizuálně napovídat uživateli, zda je zadaná nota správná – např. zvýrazněním rámečku pole nebo barevným pozadím.

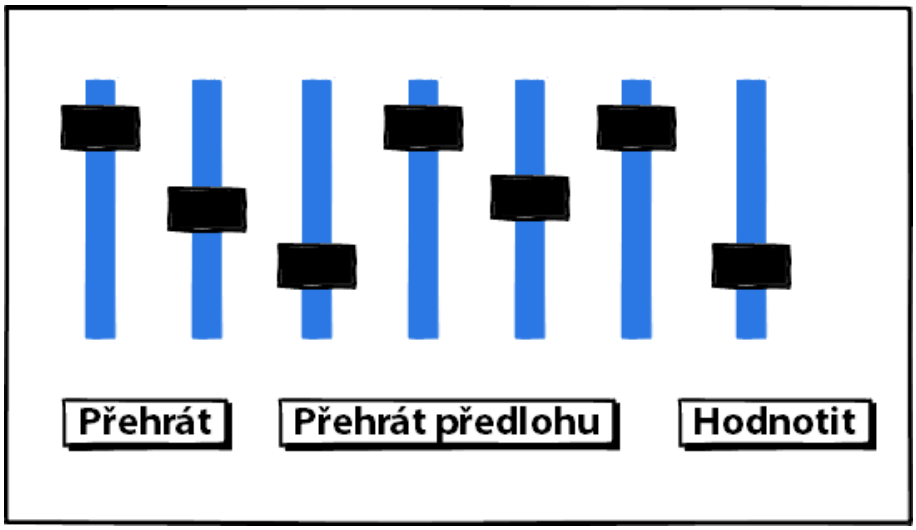
Vyřešení hry nebude nutné nijak potvrzovat, hra samotná by měla poznat správnost veškerých not a dle toho zobrazit uživateli potvrzení o vyřešeném herním kole.

4.5.4 Administrace

K této hře není nutné implementovat žádné administrační rozhraní pro přidávání podkladů, je však nutné naprogramovat správné a logické generování posloupností. Po konzultaci s Mgr. Martinem Grobárem jsme se rozhodli, že při nejnižší obtížnosti bude možné, aby docházelo ke generování návratových šipek, tedy např. posloupností „třikrát nahoru, třikrát dolů“ ($f \uparrow \uparrow \uparrow h \downarrow \downarrow \downarrow f$); při vyšší obtížnosti k této posloupnosti nebude docházet.

Grobár vysvětlil, že hru na podobném principu již zkoušel se žáky v podobě ručně psaných značek a i když nám může tato zpětná posloupnost připadat na první pohled zcela zřejmá, většině žáků trvalo nějakou dobu, než ji zpracovali správně.

4.6 Hra Posuvníky



Obrázek 14: Posuvníky – návrh hry

Další hrou budou posuvníky. Ty budou mít za cíl využívat podvědomě znalostí z hudební teorie (posloupnost not), ale především rozvíjet hudební sluch z hlediska přesného rozeznávání jednotlivých tónů, jejich délek a skládání celkové melodie.

Žákovi bude předvedena hudební melodie, která pro něj poslouží jako předloha. Melodie se bude skládat z mnoha tónů o různých délkách a výškách (podobně jako notový zápis, jen velmi zjednodušeně). Ta bude „vynulována“ do soustavy posuvníků, které zachovají jen délku tónu, výška však bude pro všechny nově vytvořené posuvníky stejná (např. tón, který je průměrem mezi nejnižším a nejvyšším tónem dané předlohy). Úkolem žáka bude nastavit každý z posuvníků tak, aby se svou výškou shodoval s tónem v předloze a ve výsledku se mu podařilo sestavit celou melodii do správné podoby.

4.6.1 Variace obtížnosti

Obtížnost bude definována počtem posuvníků, tedy počtem not (délkou melodie). Při nižší obtížnosti bude aplikace žákům předkládat jednodušší melodie, které nemusí mít vzor v nějaké existující skladbě, s rostoucí obtížností naopak poroste kromě délky také „reálnost“ melodií.

4.6.2 Hodnocení

Hodnocení bude probíhat na základě celkového času pro vyřešení úlohy a počtu změn pozice každého posuvníku žákem; ve výpočtu je zahrnut také celkový počet posuvníků. K výpočtu celkového počtu bodů bude sloužit následující skupina vzorců:

$$\mathrm{P} = 15 - \text{počet posuvníků} \tag{5}$$

$$\text{body} = \text{maxbodů} - \text{časová penalizace} - (\mathrm{P} \cdot \text{počet posuvů}) \tag{6}$$

Protože se u této hry počítá s velkým počtem kroků, tedy změn pozice některého z posuvníků, zohledňuje vzorec č. 5 celkový počet posuvníků. Jako hranice pro výpočet je zde určen počet 15. V případě, že bude celkový počet posuvníků roven nebo vyšší 15, bude proměnná `P` dosažena automaticky číslem 1 (nepředpokládá se, že by hra obsahovala větší počet posuvníků, než 25).

Limitující čas pro počítání penalizace bude pro tuto hru proměnlivý v závislosti na celkovém počtu posuvníků:

$$\text{limitující čas} = (\text{počet posuvníků} \cdot 60 \cdot 1000) \cdot 0{,}8 \tag{7}$$

Konkrétní vzorec výpočtu časové penalizace je uveden v kapitole 4.7.3 (vzorec 9). Maximální počet bodů bude stanoven na hodnotu 1000 bodů, od které budou penalizace odečítat další body.

4.6.3 Vizuální podoba

Hra sestává ze soustavy posuvníků, kde každý značí jeden tón. Posuvníky budou postaveny horizontálně vedle sebe a každý z nich bude mít drážku ve vertikální rovině, aby více připomínaly notovou osnovu, která bývá také čtena zleva doprava

(obr. 14). Z důvodu názornosti bude vhodné, aby byl každý posuvník jinak široký v závislosti na délce reprezentovaného tónu.

Pro přehlednější práci s melodií bude posuvník, který reprezentuje právě hraný tón, zvýrazněn například barvou. Každý posuvník také při změně stavu zobrazí, jaká nota je k němu přiřazena – z hlediska herní logiky půjde o méně klíčovou funkci, která však bude mít didaktický rozměr, tedy podvědomé přiřazování žákem slyšených tónů ke konkrétní notové reprezentaci.

Hra bude obsahovat tlačítko pro přehrání aktuální sestavy posuvníků, přehrání předlohy a vyhodnocení žákem. Sebehodnocení, podobně jako u *Melodických kostek*, by mělo sloužit k eliminaci náhodného zkoušení posuvů a ke zvyšování hudebního sebevědomí žáka.

4.6.4 Administrace

Tato hra bude z hlediska technické implementačně odlišná od *Melodických kostek* i *Pexesa*, protože bude nutné vyřešit přehrávání všech základních tónů a jejich půltónových posuvů[18], více oktáv a také více základních délek [19].

Při návrhu s Mgr. Martin Grobárem navíc docházíme k závěru, že bude třeba pro melodie ovlivňovat délky not tzv. tečkami[20] a také používat noty jako trioly[21].

Pokud budeme uvažovat o použití jednotlivých zvukových souborů, dostaneme se z hlediska počtu kombinací na velmi vysoký, implementačně obtížně zvládnutelný počet – výpočet ilustruje následující rovnice:

$$\text{Kombinace tónů} = \text{tón} \cdot \text{oktáva} \cdot \text{délka} \cdot \text{změna délky} = 12 \cdot 11 \cdot 5 \cdot 3 = 1980 \quad (8)$$

Takové množství kombinací může být náročné nejen z pohledu tvorby zvukových podkladů, ale také datově náročné z hlediska stahování podkladů v prohlížeči žáka.[22]

Jednou možností, jak tento problém řešit, je ukládání tónů pouze o jedné délce, která bude rovna nejdelšímu přípustnému tónu (tedy např. celému tónu s tečkou). Při umělém zkrácení to bohužel přinese daň v podobě neexistence plynulého poklesu tónu, nazývaného také *decay*[23], s celkovým počtem kombinací se však dostaneme na

[18] 7 tónů *cdefgah* a posuny pomocí ♯ a ♭ tvoří celkem 12 různě znějících tónů pro jednu oktávu

[19] celkem 5 základních délek – celá, půlová, čtvrťová, osminová a šestnáctinová

[20] Tečka za notou ji prodlužuje o její polovinu

[21] *Skupina tří stejně dlouhých not hraných v době jinak určené pro dvě noty, např. tři osminové noty hrané celkově na jednu dobu* [23].

[22] Pokud uvážíme, že jeden tón bude mít při zvukové kvalitě 128 kbps v průměru 40 kB a v rámci hry definujeme rozsah 2 oktávy, které budou používány, stále se jedná o celkových 14 MB, které by musel prohlížeč uživatele stáhnout.

[23] Nota se dělí na attack, timbre a decay – zjednodušeně řečeno *nájezd*, *tělo* a *zakončení*. *Decay je závěrečná část noty hrané na daný hudební nástroj.* [24, s. 205]

počet 132. To je již téměř totožné se standardním rozhraním MIDI, které obsahuje celkových 128 not (označených 0–127)[24] pro každý ze 128 programů[25].

Z praktických důvodů tedy bude vhodné implementovat přehrávání s využitím rozhraní MIDI, které je univerzálně uznávané a přímo definuje jednotlivé noty na základě číselného vyjádření.

V kapitole 4.8.2 navrhnu konkrétní systém vkládání hotových melodií učitelem pro použití ve hře. Mezi možné varianty patří nahrávání samotných souborů *MIDI* nebo přímé vkládání hudební notace. Z hlediska modifikovatelnosti a větší přímočarosti bude vhodnější využít přímého vkládání notového zápisu, který nespoléhá na zpracování binárního souboru, ale bude možné při tvorbě skladbu okamžitě reprodukovat, kontrolovat správnost zadání a provádět opravy.

4.7 Motivace uživatele

Při návrhu aplikace, kterou budou používat děti, je motivační složka velmi důležitá. Do celkové motivace v tomto případě zařazuji jak motivaci hrát jednotlivé hry, tak také pozitivní přístup k používání aplikace jako celku. Rozeberu zde jednotlivé složky, které by měly společně vést k tomu, aby žáci aplikaci používali zcela přirozeně a automaticky, aniž by na ně musel být veden vnější tlak ze strany rodiče či učitele.

4.7.1 Interakce

Protože chci, aby byla aplikace pro žáky skutečně přátelská a osobní, zvolím pro ni postavu člověka nebo zvířete, který ji bude reprezentovat, otiskne do ní svoji osobnost a bude sloužit k interakci mezi žákem a aplikací jako takový „učitel“.[26] Takový maskot by měl sám o sobě charakterizovat práci s hudbou nebo zvukem – tedy například zvíře, které má velmi citlivé sluchové ústrojí.

Jako hlavního maskota tedy nakonec volím sovu. Patří mezi živočichy s nejlépe rozvinutým sluchem a také bývá často reprezentována jako symbol moudrosti a vědomostí. Její vzhled však musím navrhnout v komiksovém duchu, aby na žáky nepůsobila přísně, ale spíše přátelsky a hravě.

Veškerá její komunikace se žáky bude zásadně osobní (tykáním) a bude se odehrávat na více místech v celé aplikaci – sova bude žáky motivovat ke hraní her, oznamovat jim jejich výsledky a říkat různé další „hlášky“ určené pouze pro pobavení. Důležité je, aby si žáci sovu dostatečně zapamatovali a měli ji s aplikací asociovanou.

[24] Zbývající čtyři noty jsou vynechány z konce stupnice, kde jsou vzhledem ke své výšce velmi zřídka používané.

[25] *Program je určité nastavení syntetizéru, které produkuje konkrétní hudební zvuk* [25, s. 184]. Programem zde může být např. klavír, basa, elektrická kytara, ale také zvuk helikoptéry, výstřelu z pistole nebo zvonícího telefonu.

[26] V obecnější rovině zmiňuje důležitost *osobnosti* uživatelského rozhraní [16, s. 6] – *Before you start the design process, consider your overall approach. Think about how you might design the interface's overall interaction style – its personality, if you will.*

4.7.2 Personalizace

Protože je dobré žákovi poskytnout jistou volnost při práci s aplikací a zároveň mu ji ještě více přiblížit, poskytnu mu možnost vybírat si sovu hned z několika vizuálních variací – např. různé „převleky“ nebo módní doplňky. Tím docílím, že si každý žák aplikaci představí právě pod jednou konkrétní sovou, kterou si sám zvolil a bude vědět, že když se do aplikace vrátí, přivítá jej právě ona. Sovy budu dále nazývat slovem *avatar*, které bývá obecně používáno k označení postavy, jenž reprezentuje virtuální podobu uživatele.

4.7.3 Hodnocení

Nejdůležitějším prvkem motivace žáka bude hodnocení, které získá okamžitě po dohrání každé hry. Hodnocení je nutné zobrazit čitelnou a jednoduchou formou, která žákovi okamžitě zodpoví, jak dobře si vedl. [3, s. 263]

Jak jsem rozebral v analýze konkrétních her, společnou metrikou hodnocení každé z nich bude čas, za který žák úlohu splnil, sekundární metrika se však u každé z her mírně liší – obvykle ji značí počet kroků, které musel žák pro splnění udělat, z principu her zde však bude použit pro každý případ jiný výpočet. Např. u *Krokování not* se počítá počet špatných kroků (předpokládá se, že žák bude schopen tuto hru zahrát i naprosto bez chyby), v *Posuvnících* se naopak s jistým experimentováním, a tedy i vyšším počtem kroků, počítá přímo v návrhu.

Obě tyto metriky budou důležité pro hodnocení, pro žáka by však byly příliš „neurčité“; pokud by navíc došlo v budoucnu k rozšíření o další hry, a tedy i potenciální doplnění dalších metrik v rámci jedné hry, problém srozumitelnosti by lineárně rostl. Z těchto důvodů hodnocení shrnu výpočtovým vzorcem do počtu bodů, tedy jednoho čísla, které bude jednoduše čitelné. Konkrétní vzorce jsem uvedl v návrhu samotných her (od kap. 4.3), společný vzorec pro výpočet časové penalizace je následující:

$$\text{časová penalizace} = \text{dosažený čas} \cdot \frac{\text{limitující čas}}{500} \tag{9}$$

Proměnná *limitující čas* se pro každou hru liší a je vždy uvedena v návrhu dané hry – jedná se o čas, po jehož překročení dostane žák plnou časovou penalizaci o hodnotě 500 bodů. U časových hodnot budu vždy používat hodnotu v milisekundách, takže např. 5 minut bude vyjádřeno jako $5 \cdot 60 \cdot 1000 = 300000$ ms.

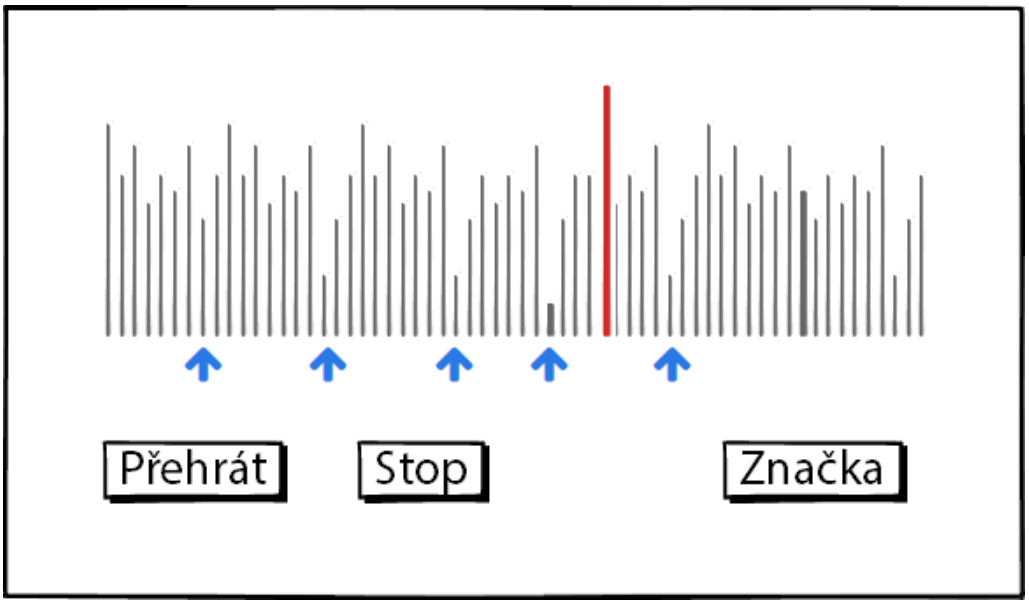
Abych povzbudil soutěživost mezi žáky a zároveň jim ukázal jednoduchý přehled těch nejúspěšnějších z nich, bude se body pro každého z nich zapisovat na jejich uživatelský účet rekord v dané hře a nastavené obtížnosti, a žebříček těch nejlepších bude vypsán na speciální stránce.

Tento koncept se nazývá *hall of fame* a jedná se o jeden ze základních způsobů motivace hráče.[27] Osobní bodové hodnocení pro uživatele navíc zobrazím i na úvodní obrazovce v přehledné formě.

4.8 Administrační rozhraní

Nezbytnou součástí aplikace bude jednoduše použitelné administrační rozhraní, ve kterém bude učitel pracovat s uživatelskými účty žáků, přidávat podklady pro jednotlivé hry a sledovat hodnocení a statistiky.

4.8.1 Tvorba předělových značek



Obrázek 15: Návrh rozhraní pro vytváření značek

Protože pracují hry *Melodické kostky* i *Pexeso* s podklady ve formě hudebních souborů (např. mp3), bude probíhat jejich nahrání přes klasický formulář. Pro *Melodické kostky* je však nutné vyřešit specifikum v podobě tvorby značek určujících předěly ve skladbě.

Vytváření značek bude probíhat během přehrávání hudebního podkladu. Aplikace spektrálně analyzuje hudební průběh skladby, z čehož vytvoří spektrogram, jenž bude sloužit učiteli jako pomůcka pro vizuální kontrolu umístění značek. Po spuštění přehrávání se bude po spektrogramu pohybovat časový kurzor, kliknutím

[27] V knize *Teaching Music with Technology* [26, s. 129] je popsáno použití stejného konceptu při výuce hudební výchovy ve školní třídě – učitel vybral hudební hru, která se vztahuje k jeho osnovám, a dal ji k dispozici žákům před a po výuce. Na základě výsledků v této hře měl na zdi v učebně žebříček nejlepších tříd. Tímto způsobem posílil soutěživost celých tříd – udělal z nich vzájemně soupeřící *týmy*.

na tlačítko *Značka* vytvoří uživatel značku na aktuálním místě kurzoru. Navržené rozhraní je zobrazeno na obr. 15.

Bude zde nutné zahrnout také mazání značek pro případné opravy – to implementuji pomocí seznamu všech značek přiřazených ke skladbě a jejích časů, kde dostane učitel možnost každou z nich jednoduše smazat.

4.8.2 Vkládání not

Pro vkládání not, které budou přehrávány rozhraním MIDI, se inspiruji u ostatních programů, které pracují non-WYSIWYG s notovým zápisem.[28] Open source hudebně notační program Lilypond [27] používá následující syntaxi:

```
\relative c" { % relativní oktáva, od které se dále odvozují noty
  d4 b8 c'4 a8 | d,4 b g4. % notový zápis
}
```

V tomto zápisu je označením `c'4` definována nota *c* s posunem o jednu oktávu nahoru o délce *1/4* taktu. Na prvním místě je tedy vždy název noty, následovaný nepovinnými znaky `,` nebo `'`, které slouží pro posun o oktávu nahoru nebo dolů. Dále následuje číslo označující podíl v délce noty (např. `1` pro notu celou nebo `16` pro notu šestnáctinovou), který nemusí být povinný – délka se v případě vynechání určí dle poslední noty s definovanou délkou. Zápis noty končí nepovinným modifikátorem délky, kde tečka značí tečkovanou notu (nota s délkou o polovinu větší). A konečně, znak `|` v tomto příkladu slouží k oddělení taktu. Protože je tento zápis velmi názorný a v praxi často používaný, využiji jeho základ k návrhu formátu notace do mé aplikace.

V navrhované aplikaci bude sloužit k definici relativní oktávy pro celou skladbu jednoduché roletové menu, které nahradí složitý textový zápis. Oddělovač taktu zde potřeba nebude, protože se nejedná o zápis pro vykreslení notace, ale pro přímé přehrávání; zbývá tedy pouze zápis samotných not. Ten bude prakticky totožný jako u Lilypondu, s tím rozdílem, že kromě tečky přidám i modifikátor `t` pro zápis trioly.[29]

Kromě toho doplním k notám znak, který bude definovat pomlku. V programu Lilypond je mezera označena písmenem `r` místo názvu noty, já však pro vyšší názornost použiji podtržítko. Zápis délky bude probíhat totožně jako u not, tedy např. `_16`.

Výsledný formát hudební melodie pro moji aplikaci bude tedy:

```
g4 g g eb8. b16 g4 eb8. b16 g4
```

[28] Vykreslení not z textové deklarace do grafické reprezentace.

[29] Vzhledem k požadavku na přesné grafické vykreslení jsou trioly v programu Lilypond definovány syntaxí `\tuplet 3/2 { b4 b b }`, ta je však pro můj účel příliš složitá.

4.9 Výběr platformy

Na základě výše uvedeného návrhu je možné využít hned několika základních platforem pro implementaci:

1. Běžná mobilní aplikace pro iOS, Android, Windows Phone.[30]
2. Běžná desktopová aplikace pro Windows nebo Mac OS.
3. Webová aplikace.
4. Kombinace více přístupů – nejčastěji webová aplikace pro desktopová zařízení a běžná mobilní aplikace pro použití na mobilních zařízeních.

Protože by vývoj mobilní aplikace pro každý z mobilních systémů byl velmi časově náročný, neefektivní a náchylný k roztříštěnosti, použiji implementaci založené na platformě webové aplikace. Ty lze dále fragmentovat podle použité technologie – spousta stávajících výukových a herních aplikací je založena na platformě Flash, ta je však stále více na ústupu, obsahuje spoustu bezpečnostních chyb a není podporována na většině mobilních zařízení [28].

Kvůli požadavkům na funkčnost v mobilních zařízeních pro práci s multimediálním obsahem použiji HTML5, které je podporováno ve všech mobilních operačních systémech [29]. Tento značkovací jazyk doplním při tvorbě multimediálního obsahu skripty v jazyce JavaScript.

[30]Běžnou mobilní aplikací zde myslím aplikaci, která využívá aplikační rozhraní dané mobilní platformy a je distribuována přes platformě specifický online obchod s aplikacemi – App Store, Google Play nebo Windows Phone Store

5 Možnosti přehrávání zvuku na webu

Než přistoupím k implementaci, je důležité prozkoumat současné technologie pro práci se zvukem v prostředí webu. Dále zde rozebírám a porovnávám knihovny, které nad těmito těmito technologiemi vytváří další vrstvu a zjednodušují tak jejich použití.

5.1 Průzkum webových technologií

Technologie, které dále popisuji, využívají zvukové soubory v mnoha různých formátech. Tyto formáty jsou definovány standardem MIME (Multipurpose Internet Mail Extensions), který původně sloužil k definici rozšíření pro elektronickou poštu. Mezi nejpoužívanější formáty patří [30]:

- `audio/webm` – Kontejner WebM, který obsahuje zvukový soubor kódovaný kodekem Vorbis nebo Opus; slouží také pro přenos videa. Není podporován v prohlížečích Safari a Internet Explorer.[31]
- `audio/ogg (audio/vorbis)` – Kontejner Ogg, v němž je zvuk kódovaný kodekem Vorbis. Není podporován v prohlížečích Safari a Internet Explorer.
- `audio/mpeg` – Soubor v uzavřeném formátu MP3, jehož použití je v některých zemích vázáno licenčními poplatky.[32] Je podporován většinou komerčních prohlížečů, z licenčních důvodů je jeho podpora v prohlížeči Firefox závislá na podpoře formátu ve vrstvě operačního systému.
- `audio/wave` – Kontejner WAVE, který obsahuje nekomprimovaný zvuk kódovaný kodekem PCM. Není podporován v prohlížeči Internet Explorer.[33]

5.1.1 HTML5 Audio

Obrázek 16: Zobrazení `<audio>` přehrávače v prohlížeči Google Chrome

Základní možností přehrávání hudby na stránkách je využití značky `<audio>`, která využívá rozhraní `HTMLAudioElement` a jejíž podporu přinesl standard HTML5.

[31] V Internet Exploreru verze 9 a vyšší je možné docílit podpory pomocí doplňku od společnosti Google – `https://tools.google.com/dlpage/webmmf/`

[32] Společnost Technicolor si stále nárokuje licenční poplatky [31], přestože v některých zemích platnost patentů již vypršela; poslední patenty ve Spojených státech, které se vážou k tomuto formátu, vyprší v roce 2017 [32].

[33] Podpora bude zahrnuta v následující verzi prohlížeče Internet Explorer, která bude obsahovat nové vykreslovací jádro Edge [33].

Tento prvek slouží k vložení zvukového obsahu do dokumentu; je zde možné specifikovat více zvukových zdrojů (formátů) pomocí prvku `<source>`, přičemž prohlížeč sám zvolí ten, který je pro něj nejvhodnější z hlediska podpory [34].

Ukázka použití (výsledek na obr. 16):

```
<audio id="prehravac" controls autoplay>
   <source src="skladba.mp3" type="audio/mpeg">
   Váš prohlížeč nepodporuje značku audio.
</audio>
```

Tento způsob je určen především pro vkládání jednoduchého přehrávače zvukového souboru do webové stránky. Také je možný objektový přístup k deklarovanému prvku pomocí DOM (Document Object Model) [35, s. 96]:

```
prehravac = document.getElementById('prehravac');
prehravac.pause();
```

Přehrávání je možné vytvářet a řídit jen s pomocí JavaScriptu, což je vhodné například v případě, kdy není žádoucí vytvářet element v rámci kontextu stránky:

```
prehravac = new Audio();
if (prehravac.canPlayType('audio/mpeg') != }) {
   prehravac.src = 'skladba.mp3';
   prehravac.play();
} else {
   alert('Váš prohlížeč nepodporuje formář MP3.')
}
```

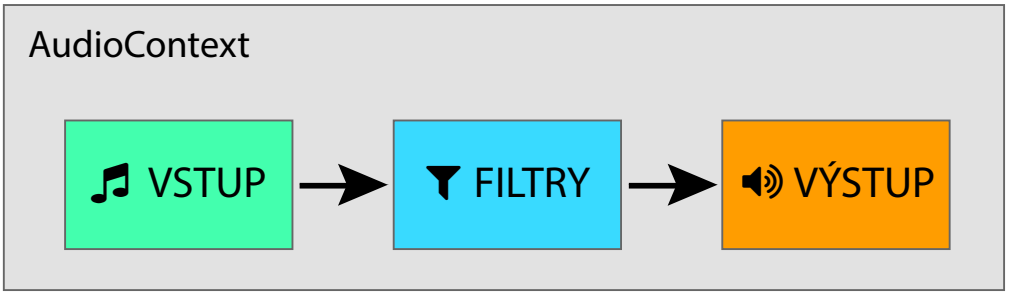
Standard HTML5 Audio zajišťuje u zvukových souborů např. zobrazení aktuálního času skladby, přehrávání, pozastavování nebo změnu hlasitosti – nedokáže však manipulovat se zvukem na pokročilejší úrovni, např. zvuk generovat, modifikovat, analyzovat nebo přesněji kontrolovat časování zvuků [34].

Podpora HTML5 Audio v prohlížečích je velmi vysoká, ke dni 7. 5. 2015 dosahuje v České republice 95,66 % (viz příloha B).

5.1.2 Web Audio API

Standard Web Audio API přistupuje k práci se zvukem na nižší úrovni abstrakce, než HTML5 Audio. Jeho činnost je založena na *audio kontextu*, který vytváří graf složený z řetězce uzlů (obr. 17), jenž vede od zdroje zvuku, přes filtry nebo analyzátory, až po výstup[34] [36].

[34] Živá ukázka tohoto principu je demonstrována v aplikaci Web Audio Playground `http://webaudioplayground.appspot.com/`



Obrázek 17: Schéma řetězce Web Audio API kontextu

Samotné zpracování zvuku probíhá v kódu prohlížeče, který je napsán v jazyku nižší úrovně, než je JavaScript (typicky v C, C++ nebo jazyku symbolických adres – často označován jako Assembler). Zdrojem zvuku může být [37]:

- oscilátor
- živý vstup (např. z mikrofonu)
- HTML5 Audio (prvek `<audio>`)
- přímé generování JavaScript kódem

Pokud chce programátor přehrávat pouze zvukový soubor, ale nechce použít jako zdroj zvuku značku `<audio>`, použije např. následující implementaci:

```
var source = 'http://server.cz/skladba.mp3';
var context = new AudioContext();

var soundFile = new XMLHttpRequest();
soundFile.open('GET', source, true);
soundFile.responseType = 'arraybuffer';

soundFile.onload = function() {
    context.decodeAudioData(soundFile.response, function(buffer){
        var soundBuffer = context.createBufferSource();
        soundBuffer.buffer = buffer;
        soundBuffer.connect(context.destination);
        soundBuffer.start(0);

    });
}

soundFile.send();
```

Zvukový zdroj není načítán přímo, ale skrze flexibilnější rozhraní `XMLHttpRequest` – to umožňuje například využití asynchronních požadavků nebo dynamické stahování zvukového obsahu v závislosti na průběhu skriptu.

Připojení jednotlivých filtrů probíhá velmi jednoduše – místo připojení zvukového souboru přímo na výstup je vytvořen další uzel, přes který je zvuk zpracován:

```
// [...] načtení bufferu
var soundBuffer = context.createBufferSource();
soundBuffer.buffer = buffer;
var filter = context.createBiquadFilter();
filter.type = filter.LOWPASS;
filter.frequency.value = 1000;
filter.Q.value = 30 * 0.6;
soundBuffer.connect(filter);
filter.connect(context.destination);
soundBuffer.start(0);
// [...]
```

Stejným způsobem se pracuje i s dalšími zdroji zvuku, jako je oscilátor. Pro generování posloupnosti tónů sinusového průběhu, kde první začíná na frekvenci 440 Hz a další jsou vždy o 30 Hz vyšší[35], lze postupovat následovně:

```
var context = new AudioContext();
var oscillators = [];

for (var i = 1; i < 5; i++) {
    var time = context.currentTime + i * 0.6;
    oscillators[i] = context.createOscillator();
    oscillators[i].type = "sine";
    oscillators[i].frequency.value = 440 + i*30;
    oscillators[i].connect(context.destination);
    oscillators[i].start(time);
    oscillators[i].stop(time + 0.2);
}
```

V tomto případě je použito interní časování API, které funguje v porovnání s běžnou funkcí JavaScriptu `setTimeout` velmi přesně.[36] Web Audio API je díky své rychlosti vhodné také pro použití v počítačových hrách a dalších aplikacích, které pracují se zvukem v reálném čase.

Protože byl standard ještě nedávno ve fázi vývoje a některé prohlížeče jej implementovaly v nehotové podobě, používaly pro implementaci rozpracovaného stan-

[35] 440 Hz odpovídá hudebnímu tónu *a'*, ostatní tóny již nejsou svojí frekvencí *hudebně správně*, protože hudební stupnice je uspořádána logaritmicky.

[36] U funkce `setTimeout` se špatné časování projevuje nejvíce ve chvíli, kdy se v prohlížeči uživatel přepne do jiné záložky – z důvodu úspory zdrojů prohlížeč zpomaluje interní časovač. Jinou možností přesného časování je technologie *HTML5 Web Workers*. [38]

dardu prefixovaný název třídy, tedy například `webkitAudioContext` pro prohlížeče založené na jádru Webkit (typicky Google Chrome a Safari) [39].

Během ustálení standardu došlo také ke změnám některých metod, které byly během vývoje upraveny – například dříve platné metody pro přehrávání zvuku `noteOn()` (příp. `noteGrainOn()`) a zastavení zvuku `noteOff()` byly přejmenovány na společné `start()` a `stop()`.[37]

Jak bylo otestováno v Google Chrome 42, v současné době zde funguje prefixovaný název, ale také standardní `AudioContext`; zastaralé metody jsou již nefunkční. V konzoli *Developer Tools* je zobrazeno varování pro vývojáře, upozorňující na zastaralost prefixu.[38]

Podpora v prohlížečích je v době psaní této práce stále podstatně nižší, než podpora HTML5 Audio – ke dni 7. 5. 2015 dosahuje v České republice 74,97 % (viz příloha C). Do prohlížeče Internet Explorer bude zahrnuta v následující verzi, která bude obsahovat nové vykreslovací jádro Edge [40].

5.2 Průzkum knihoven pro přehrávání zvuku

Zvukové knihovny slouží k abstrakci přístupu ke konkrétní technologii přehrávání zvuku. Jsou určeny pro přímočarou implementaci – programátor se nemusí zabývat podrobnými technickými detaily technologie, ale využije hotové funkce knihovny připravené pro nejčastější případy použití.

5.2.1 SoundManager 2

Knihovna SoundManager 2 [41] slouží k přehrávání zvukových souborů v prohlížeči pomocí HTML5 Audio a technologie Flash. Zvukům je možné nastavovat identifikátor a volat je pomocí parametru metody objektu `soundManager`, nebo je přiřazovat do vlastní proměnné:

```
soundManager.setup({
    url: '/swf-soubory/',
    onready: function() {
        var skladba = soundManager.createSound({url: 'skladba.mp3'});
        skladba.play();
    }
});
```

Knihovna obsahuje velmi podrobný systém událostí – lze je volat po načtení souboru, spuštění nebo dokončení přehrávání a další. Také je možné vytvářet *liste-*

[37] Na stránce `https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Porting_webkitAudioContext_code_to_standards_based_AudioContext` je přehled změn v kontextu Webkitu.

[38] `'webkitAudioContext' is deprecated. Please use 'AudioContext' instead.`

nery pro časové pozice ve skladbách a spouštět tak vlastní callback funkci například ve 2. sekundě skladby:

```
soundManager.onPosition('skladba', 2000, function(eventPosition){
   console.log('Casova znacka ' + eventPosition + ' milisekund.');
});
```

Knihovna neimplementuje přímočaré použití hotových zvukových spritů[39], ale při zavolání metody pro přehrávání lze definovat čas začátku a konce stopy v milisekundách:

```
skladba.play({from: 1200, to: 4000});
```

V dokumentaci je uvedeno varování, že konečný čas může být prodloužen až o 200 ms a více, v závislosti na rychlosti běhu skriptu. Autor popisuje možnost zpřesnění času nastavením `useHighPerformance`, které dává vyšší prioritu Flash přehrávači.

Dokumentace je velmi podrobná, včetně popisu API a mnoha praktických ukázek použití. Autor demonstruje také vizuálně zajímavé aplikace, kterých lze s využitím knihovny docílit.

Protože knihovna využívá proprietární doplněk Flash, je možné ji používat i ve starších prohlížečích. V mobilních zařízeních je nutné spolehnout se pouze na HTML5 Audio, které má omezené možnosti časování.

5.2.2 Wad

Wad [42] je knihovna, která zajišťuje abstrakci nad Web Audio API – místo sítě uzlů, které Web Audio využívá, pracuje pomocí instancí objektu *wad*, které reprezentují konkrétní zdroje zvuku. Spuštění přehrávání zvukového souboru pracuje následovně:

```
var skladba = new Wad( {source : 'http://server.cz/skladba.mp3'} );
skladba.play();
```

Podobně je možné vytvářet syntetizované zvuky – Wad poskytuje přístup k notám podle jejich názvů, zároveň umožňuje nastavení *attack*, *decay* i dalších hudebních vlastností [43]:

```
var bass = new Wad({
   source: 'sine',
   env: {
      attack: .02,
      decay: .1,
```

[39] Viz vysvětlení v popisu knihovny Howler.

```
        sustain: .9,
        hold: .4,
        release: .1
    }
})
bass.play({ pitch : 'C2' })
```

Tato knihovna neimplementuje podporu událostí a callbacků, není tedy možné volat vlastní funkci například po skončení přehrávání skladby. Neexistuje zde způsob zjištění aktuální časové pozice v přehrávané skladbě.

Dokumentace knihovny je velmi podrobná, na jedné stránce jsou rozebrány všechny metody včetně příkladů použití. Protože Wad pracuje pouze s Web Audio API, má stejnou podporu v prohlížečích, jako tato technologie – žádný další doplněk není vyžadován.

5.2.3 Howler.js

Howler.js [44] je knihovna, která pracuje se zvukem pomocí Web Audio API s *fallbackem* [40] na HTML5 Audio. Podobně jako Wad vytváří pro každý zdroj zvuku další instanci objektu, s výjimkou spritů. Následující ukázky kódu jsou demonstrovány na verzi 2.0.

```
var skladba = new Howl({
  src: ['http://server.cz/skladba.mp3']
});
skladba.play();
```

Howler.js podporuje pouze přehrávání zvukových souborů. Soubory je možné deklarovat ve více formátech, jenž jsou použity podle podpory v prohlížeči.

Velkou předností této knihovny je přímočaré použití zvukových spritů – do jednoho zvukového souboru je možné umístit za sebou více nezávislých zvuků a jejich časy v milisekundách přiřadit pod vlastní názvy, prohlížeč tak pro stažení vykoná pouze jeden HTTP požadavek, ale přehrává zvlášť jednotlivé zvuky:

```
zvuky = new Howl({
    src: ['http://server.cz/zvuky.wav'],
    sprite: {
        sirena: [0, 3509],
        motorka: [3509, 6499],
        auto: [3509, 6499],
    }
```

[40] Fallback ve volné interpretaci znamená, že v případě neexistence podpory jedné technologie, je využita technologie *náhradní*; frázi *to fall back on* lze přeložit jako *uchýlit se k nějakému řešení, spolehnout se na něco.*

```
    onload: function() {
        zvuky.play('sirena');
    }
});
```

Knihovna implementuje systém událostí – je možné volat vlastní funkce pomocí callbacků, například po načtení skladby, zastavení přehrávání, spuštění přehrávání a další. Pro zjištění nebo nastavení aktuální časové pozice ve skladbě lze volat metodu `seek([seek], [id])`.

Dokumentace knihovny je velmi podrobná, na jedné stránce jsou rozebrány všechny metody včetně příkladů použití. Díky použití Web Audio API i HTML5 Audio je knihovna kompatibilní s drtivou většinou moderních prohlížečů, nefunguje však ve starších verzích, které neimplementují ani jednu z těchto technologií.

5.2.4 MIDI.js

V porovnání s ostatními zástupci je knihovna MIDI.js [45] zaměřena pouze na zvuk přehrávaný na bázi rozhraní MIDI a využívá Web MIDI API, Web Audio API a HTML5 Audio. Slouží ke skriptovanému přehrávání notového zápisu i přímému přehrávání souborů *MIDI*.

Ukázka kódu pro přehrání tónu *c'* o délce 3/4 sekundy:

```
MIDI.loadPlugin({
    soundfontUrl: 'midi-soundfonts/FluidR3_GM/',
    instrument: 'acoustic_grand_piano',
    onsuccess: function() {
        var note = MIDI.noteToKey['C4'];
        var velocity = 127;
        MIDI.setVolume(0, 127);
        MIDI.noteOn(0, note, velocity, 0);
        MIDI.noteOff(0, note, 0.75);
    }
```

Podklady pro jednotlivé tóny získává MIDI.js ze zvukových fontů ve speciálním formátu. Jeden hudební nástroj (v terminologii MIDI *program*) obsahuje v průměru 89 zvukových souborů, kde každý představuje jeden tón. Všechny tyto tóny jsou pomocným skriptem zabaleny do jediného JS souboru, ve kterém jsou zakódovány pomocí *base64*. Každý zvukový font obsahuje reprezentaci tónů přístupnou podle čísla MIDI (0—127) nebo názvu noty (A0—Gb7).

Dva ukázkové zvukové fonty jsou součástí knihovny, další je nutné stáhnout z externích zdrojů – v dokumentaci je uveden odkaz na kolekci *MIDI.js Soundfonts*[41], která obsahuje všech 128 programů podle standardu General MIDI. Datový tok

[41] Dostupné z URL `https://github.com/gleitz/midi-js-soundfonts`.

jednotlivých tónů je 33 kbps, což zaručuje nižší datovou velikost každého z nich, ale také velmi nízkou kvalitu.

Knihovna umožňuje přehrávání celých akordů – k tomu slouží metoda `MIDI.chordOn()`. Knihovna obsahuje implementaci událostí pro přehrávač MIDI souborů, kdy je možné sledovat jednotlivé noty, jejich přehrávání, zastavení, zastavení skladby apod.Dokumentace je velmi špatná, popis je zmatečný a neúplný a je nutné podrobněji studovat zdrojové kódy pro zjištění způsobů použití u některých z metod.

Díky široké podpoře standardů je knihovna kompatibilní s drtivou většinou moderních prohlížečů. V dokumentaci je zmíněna spolupráce s knihovnou Sound Manager 2, což by mělo umožňovat alternativní použití Flashe pro starší prohlížeče, v kódu knihovny však tato funkčnost není nikde zahrnuta.

5.2.5 Výběr knihovny

Na základě výše uvedeného průzkumu jsem se rozhodl použít knihovnu *Howler.js* (kap. 5.2.3) pro hry využívající přehrávání hotových skladeb. Výhodou je využití Web Audio API s alternativním použitím HTML5 Audio, takže má vysokou podporu i na mobilních zařízeních a Internet Exploreru od verze 9.

U hry *Posuvníky*, která je navržena k práci s hudbou založenou na MIDI rozhraní, jsem využil knihovnu *MIDI.js* (kap. 5.2.4). Díky použití Web Audio API, HTML5 Audio a Web Midi API je zaručena vysoká kompatibilita a přehrávání funguje relativně přesně i na mobilních zařízeních.

6 Implementace aplikace

Aplikaci jsem implementoval v *PHP 5.4* a *Nette Frameworku 2.3* [46] s využitím databázového serveru *MySQL 5.5*, mimo jiné z důvodu zajištění vysoké kompatibility do budoucna, pro případy nasazování aplikace na většině běžných webových hostingů.

Vývoj probíhal lokálně ve virtuálním serveru, který běžel na *Ubuntu 14.04 LTS „Trusty Tahr“*. Pro účely testování ve výuce jsem aplikaci nasadil na virtuální privátní server od společnosti DigitalOcean[42] s totožnou konfigurací Linuxové distribuce. Pro jednoduché nasazování aplikace a rozdělení produkční a vývojové větve kódu jsem využil služby *GitHub*, která slouží k hostování repozitáře pro vlastní kód, založený na distribuovaném verzovacím systému *Git*.

Protože jsem nebyl, na rozdíl od volby konvenčního webového hostingu, vázán k použití webového HTTP serveru *Apache*, který je nasazen na drtivé většině z nich, využil jsem server *Nginx 1.9.0*. Ten vykazuje v mnoha případech nižší paměťové nároky, vyšší výkon a zvládne tak v rámci stejného časového úseku obsloužit větší počet klientů [48].

Veškeré knihovny třetích stran, stejně jako celý Nette Framework (dále také zkráceně jako *Nette*), jsem stahoval přes nástroj *Composer* [49], který slouží ke správě závislostí v PHP. Pro správu Front-end balíčků (využívajících JavaScriptu nebo CSS) jsem použil aplikaci *Bower* [50].

Při tvorbě jsem použil CSS framework *Bootstrap* [51], který usnadňuje tvorbu responzivního designu. Samotné kaskádové styly jsem psal pro *CSS* preprocesor *LESS*, automatickou kompilaci LESS souborů do CSS zajišťoval program *Gulp*. Grafické symboly, které aplikace obsahuje, jsou součástí CSS knihovny *Font Awesome* [52].

Z pohledu skriptování prohlížeče pracuji s JavaScriptem a knihovnou *jQuery 2.1.3*. V některých skriptech, především pro herní logiku, dále využívám obecnou deklaraci třídy Daniela Steigerwalda [53] funkcí `$class`. Ostatní knihovny, které zde nezmiňuji, uvádím přímo u konkrétních případů použití.

6.1 Diagram entit a vztahů

Na základě požadavků a provedeného návrhu jsem vytvořil diagram entit a vztahů pro strukturu databáze. Grafickou podobu diagramu jsem vytvářel v programu *MySQL Workbench 6.2* [54], pomocí nějž jsem také exportoval hotovou databázovou strukturu do skriptu pro MySQL.

Protože jsem zvolil k implementaci Nette Framework, rozhodl jsem se pro pojmenování tabulek na základě konvence, která se u tohoto frameworku běžně používá [55]. Veškeré tabulky jsem označil anglickým názvem v jednotném čísle (singuláru),

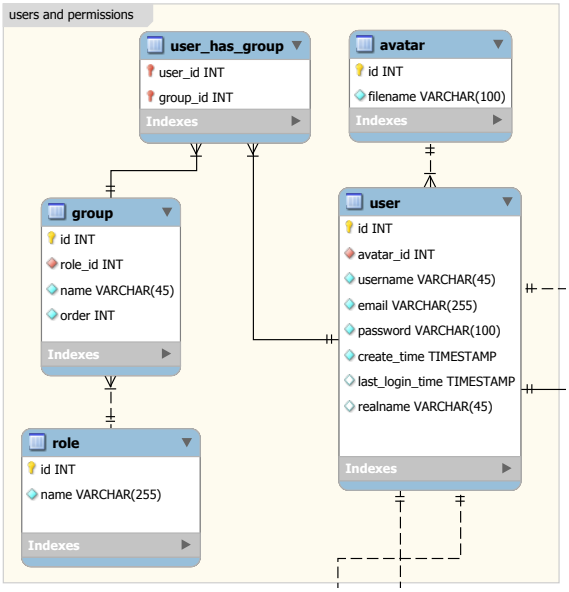
[42] Předplatné VPS jsem získal pro vzdělávací účely v rámci *Student Developer Pack* od společnosti GitHub na několik měsíců zdarma [47]

podobně, jako bývají obvykle značeny třídy objektů v kódu programu. Primární klíč je u všech tabulek nazván `id`.

Pro spojování tabulek jsem využil formátu `tabulka2_id` pro sloupec cizího klíče, spojovací tabulku jsem vždy označil názvem `tabulka1_has_tabulka2`. Toto je výhodné především při využití třídy Nette Database, která na základě konvencí v `Nette\Database\Conventions\DiscoveredConventions` dokáže automaticky nalézat tabulky, které jsou spolu v relaci. [56, 55]

V této části popíšu jednotlivé sekce diagramu, jejich použití a význam v aplikaci. Kompletní ERD je v příloze D.

6.1.1 Uživatelé a oprávnění



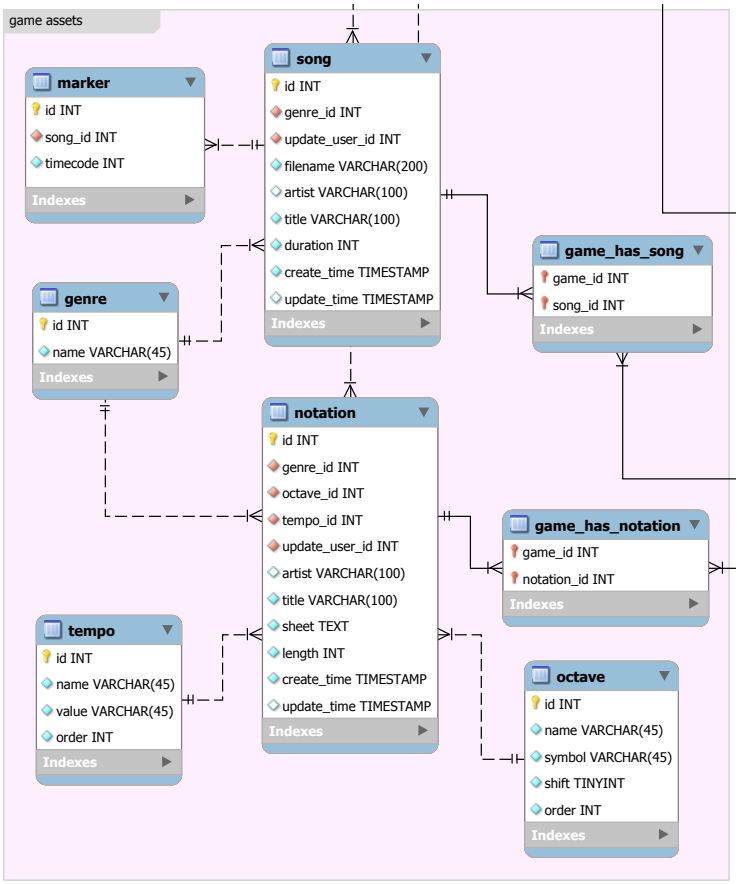
Obrázek 18: ERD – sekce *uživatelé a oprávnění*

Sekce *users and permissions* definuje způsob uložení jednotlivých uživatelských účtů a jejich vztahů v rámci rolí a skupin (obr. 18). Každý uživatel je veden tabulce `user`, která ukládá jeho základní údaje, včetně přihlašovacího jména a hesla, a také eviduje jeho poslední čas přihlášení.

Sloupec `avatar_id` odkazuje na *avatara*, kterého si uživatel může zvolit (koncept avatarů popisuji v kapitole 4.7.2). Tabulka avatarů odkazuje pouze na název souboru, pod kterým je obrázek uložen.

Každý uživatel může být součástí několika skupin (vazba M:N). To je výhodné např. z důvodu vazby učitele na třídu – učitel je součástí skupiny *učitelé* a zároveň ve „své“ třídě. Každá skupina má definovanou určitou roli (role dále rozebírám v kapitole 6.3.2).

6.1.2 Zdroje her



Obrázek 19: ERD – sekce *zdroje her*

V sekci *game assets* jsou uloženy veškeré zdroje, které jsou využívány jednotlivými hrami při jejich činnosti (obr. 19). Tabulka `song` slouží k ukládání hudby, která je využívána v rámci her *Melodické kostky* (kap. 4.3) a *Pexeso* (kap. 4.4). Tabulka eviduje název fyzického souboru s hudbou, který je uložen na server, a základní údaje o skladbě, včetně jejího času.

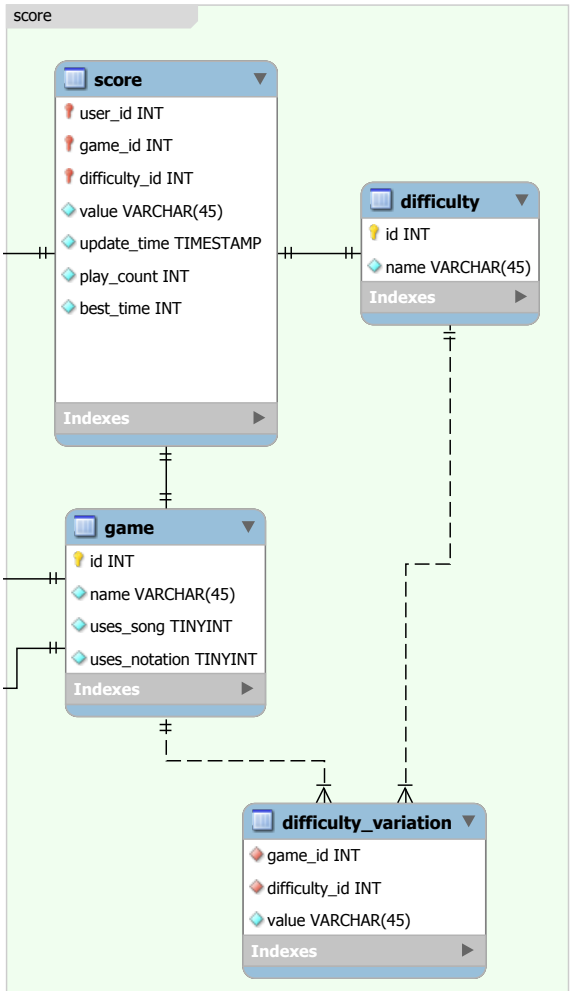
Žánr skladby je uložen v samostatná číselníkové tabulce `genre`. Pro *Melodické kostky* je zde navíc tabulka `marker`, která slouží k ukládání časových kódů předělových značek.

V tabulce `notation` jsou uloženy notové zápisy jednotlivých skladeb, které využívá pouze hra *Posuvníky* (kap. 4.6). Nejdůležitější je sloupec `sheet`, který ukládá samotný zápis; na základě něj je uložena také hodnota délky skladby ve sloupci `length`. Jak bylo u této hry navrženo, délka skladby slouží k rozlišování obtížnosti. Dále je pro každou skladbu definované tempo a základní oktáva, které jsou uloženy v číselníkových tabulkách.

Pro každou notaci i skladbu lze nastavit jednu nebo více her, ve které může být využívána – to je důležité například při ukládání časově různě dlouhých skladeb, kde

může mít vyučující jiné požadavky pro *Pexeso* a jiné pro *Melodické kostky*. Zároveň se jedná o prvek, který do budoucna výrazně zjednoduší potenciální implementaci dalších her.

6.1.3 Hodnocení



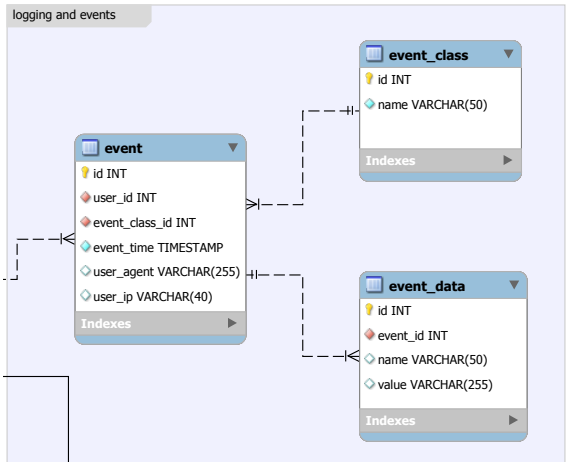
Obrázek 20: ERD – sekce *Hodnocení*

Hlavním účelem tabulky `score` je ukládání jednoduše čitelných rekordů, které hráči získají v konkrétních hrách. Ty se skládají z označení konkrétní hry (tabulka `game`), obtížnosti, v jaké je rekord veden (tabulka `difficulty`) a reference na samotného uživatele podle jeho primárního klíče. Pod touto kombinací je nejvyšší počet bodů, který uživatel získal, jeho nejlepší herní čas, čas poslední aktualizace a celkový počet dohraných her.

Tabulka `difficulty_variation` udržuje ke každé hře informaci, jak se má za dané obtížnosti zobrazovat – např. pro hru *Melodické kostky* definuje počty předělů,

pro *Posuvníky* délky skladeb podle počtu not. Další informace o principu této tabulky popisuji u společného rozhraní her (kap. 6.5) a u implementace jednotlivých her.

6.1.4 Záznamy a události



Obrázek 21: ERD – sekce *Záznamy a události*

Základní tabulka `event` slouží k ukládání událostí, které daný uživatel vyvolá. Události jsou děleny podle tříd, které udržuje číselníková tabulka `event_class` – např. třídy *autentizace*, *splnění hry* a další. Každá událost ukládá čas, kdy byla vyvolána, informace o prohlížeči uživatele a jeho IP adresu – toto jsou důležité poznatky pro testování a ladění aplikace.

Každá třída události obsahuje určitý soubor dat, které se k ní vážou – v případě spuštění hry to je například název dané hry, počáteční nastavení, obtížnost a další. Tímto způsobem je možné velmi detailně vyhodnocovat, jakým způsobem uživatelé s aplikací pracují. Další informace k implementaci událostí jsou uvedeny v kap. 6.4.

6.2 Struktura aplikace

Kód mé aplikace využívá architekturu Model-View-Controller (MVC), která je přímo podporována v Nette Frameworku [57]. MVC[43] zajišťuje oddělení vrstvy zobrazující data, obsluhy a aplikační logiky. Základní bloky Nette Frameworku, které jsou analogické k MVC architektuře, se dělí na *modely*, *presentery* a *šablony*.

V mé aplikaci jsou všechny aplikační třídy součástí jmenného prostoru `\App`. Tímto způsobem je vytvořena hierarchická struktura tříd a nedochází ke kolizím ve jménech.

[43] V rámci terminologie Nette někdy označováno jako MVP – *Model-View-Presenter*.

6.2.1 Modely

Modely v Nette Frameworku obsahují základní aplikační logiku – starají se o komunikaci s databází, ověřování uživatele, ukládání nových souborů a další. Protože se při práci s databází často opakují stejné úkony, vytvořil jsem abstraktní modelovou třídu nazvanou `Base`. Ta obsahuje metody pro získávání dat, aktualizaci záznamů, vkládání záznamů a další.

Od této třídy dále dědí třídy další, které jsou označeny stejně, jako tabulky v databázi (kap. 6.1). Některé z nich implementují nové metody, které jsou pro ně specifické – například ukládání souborů na disk u třídy *Song*. Tyto modelové třídy jsou součástí jmenného prostoru `\App\Model`.

6.2.2 Presentery

V Nette Frameworku jsou prvkem, který zajišťuje komunikaci mezi uživatelem a aplikační logikou, tzv. *presentery*. Presenter obsahuje metody, které zajišťují zpracování požadavků od uživatele, na základě nichž spouští odpovídající akce a jejich výstup zobrazuje pomocí *šablon*.

V mé aplikaci jsem se rozhodl presentery rozdělit do samostatných modulů, které rozlišují jejich použití. Na nejvyšší úrovni jsou moduly *Front* a *Admin* a pomocný modul *Base*, který obsahuje abstraktní třídy `BasePresenter` a `BaseGamePresenter`. `BasePresenter` dědí od třídy `Nette\Application\UI\Presenter`, jenž obsahuje základní implementaci presenterů v Nette.

Moduly jsou rozděleny jmennými prostory tříd a také na úrovni adresářů. V rámci adresářové struktury obsahuje každý z modulů složku *templates*, ve které jsou uloženy šablony patřící k presenterům. *Front* obsahuje prezentační logiku, starající se o aplikaci především z pohledu běžného uživatele, tedy žáka. Součástí *Front* je podmodul *Game*, který se skládá z presenterů jednotlivých her.

Modul *Admin* je určen pro práci učitele a dalších uživatelů s privilegovaným oprávněním – obsahuje presentery, které pracují s uživatelskými účty, skupinami, slouží k nahrávání hudebních podkladů, prohlížení statistik a událostí a další.

6.2.3 Dependency Injection

V případě, že jedna třída vyžaduje použití jiné třídy, vyžádá si předání závislostí pomocí *Dependency Injection* (DI). Tento návrhový zdroj zaručuje přehledný přístup ke službám objektů, jenž minimalizuje skryté závislosti. Nette Framework využívá Systémový kontejner, což je statický DI kontejner, *ve kterém se nacházejí všechny služby a parametry potřebné pro běh aplikace* [58].

Tento kontejner je generován pomocí třídy `Nette\Configurator`. Nastavení Systémového kontejneru je definováno v konfiguračních souborech `config.neon` (hlavní konfigurace aplikace) a `config.local.neon` (konfigurace specifická pro lokální prostředí – např. přístupové údaje k databázi).

V Nette Frameworku existuje několik způsobů, jak používat Dependency Injection při předávání služby [59]:

- konstruktorem,
- setterem či nastavením proměnné,
- metodou `inject*`,
- anotací `@inject`.

Každý způsob je vhodný pro jinou komponentu v Nette. V mé aplikaci na základě doporučení v dokumentaci [59] předávám závislosti u presenterů pomocí anotace `@inject`:

```
class EventsPresenter extends BasePresenter {
  /** @inject @var Model\Event */
  public $eventModel;
}
```

Tento způsob je nejkratší a dobře čitelný, dochází však k porušení principu zapouzdření – u takto předávané služby je nutné nastavit veřejnou viditelnost proměnné. Hlavním důvodem, proč se v Nette u presenterů nepředávají závislosti konstruktorem, je problematika dědičnosti. Pokud rodičovský presenter získává závislosti konstruktorem, je nutné je znovu vyjmenovat v konstruktoru potomka a komplikuje se tak signatura konstruktoru [59].[44]

V ostatních třídách, především těch modelových, využívám doporučeného předávání konstruktorem:

```
abstract class Base extends Nette\Object {
  protected $db;
  public function __construct(Nette\Database\Context $connection) {
    $this->db = $connection;
  }
}
```

6.3 Uživatelské účty

Při práci s uživateli využívám základních bezpečnostních postupů. *Autentizace* zajišťuje ověřování uživatele pomocí jeho uživatelského jména a hesla; ověřuje, zda se jedná o existujícího uživatele. *Autorizace* zajišťuje schvalování přístupu k jednotlivým zdrojům aplikace, tedy např. rozlišení, zda může uživatel přistupovat ke správě aplikace, či nikoliv.

[44] Tento problém nastává v případech, kde DI využívá rodič i potomek – pokud potomek nepotřebuje získávat další službu, konstruktor se vynechá a je použit ten, který je definován v rodičovské třídě.

6.3.1 Autentizace

Autentizaci zajišťuje třída `Authenticator`, která implementuje rozhraní `IAuthenticator`. Rozhraní definuje povinnou metodu `authenticate`, která je při ověřování zavolána – po úspěšném ověření vrací identitu uživatele.

Autentifikační třída přistupuje k databázi uživatelů pomocí třídy `Model\User`, jenž je předána v konstruktoru (kap. 6.2.3). Pro vytvoření otisku (*hashe*) hesla a jeho ověření využívá pomocné třídy `Nette\Security\Passwords` – otisk je vytvořen pomocí algoritmu `bcrypt`. Ten je v současné době vnímán za jeden z nejbezpečnějších algoritmů pro vytváření otisků hesel. [60]

Objekt `Nette\Security\Identity`, kterou autentizační metoda vrací, obsahuje ID uživatele, seznam jeho rolí a další základní údaje (kromě hesla), které jsou uloženy v tabulce `user` (kap. 6.1.1).

6.3.2 Autorizace

Autorizační třída `Acl`, kterou používám ve své aplikaci, dědí od třídy `Permission`, která implementuje v Nette implementuje rozhraní `IAuthorizator`. Tato třída vytváří statický seznam rolí, které jsou přidělovány jednotlivým uživatelům podle jejich členství ve skupinách (kap. 6.1.1). Role jsou vytvořeny hierarchicky, kde vyšší role dědí od té nižší (má stejná privilegia):

```
$this->addRole('guest');
$this->addRole('student', 'guest');
$this->addRole('teacher', 'student');
$this->addRole('editor', 'teacher');
$this->addRole('admin', 'editor');
```

Dále jsou zde definovány zdroje, které označuji identicky, jako cesty k jednotlivým presenterům[45]. Součástí každého zdroje je operace, což jsou v mém případě konkrétní akce, které presentery vykonávají. [61]

Každá role má přidělené určité zdroje a operace, ke kterým má přístup. Příklad definice přístupu pro hosty:

```
$this->allow('guest', array(
    'Front:Default', 'Front:Profile', 'Front:Game:MelodicCubes',
    ↪ 'Front:Game:Pexeso', 'Front:Game:NoteSteps',
    ↪ 'Front:Game:Faders', 'Front:Rating'));
```

Administrátor má přístup ke všem rolím pomocí následující definice:

```
$this->allow('admin', Permission::ALL, Permission::ALL);
```

[45] Každá cesta je uvedena ve formátu `Front:Game:MelodicCubes` – tedy *modul*, příp.*podmodul* a *presenter*.

6.3.3 Přihlašování

Z pohledu uživatele probíhá přihlašování pomocí přihlašovací obrazovky, která je zobrazena při prvním spuštění aplikace. O zpracování přihlašovací obrazovky se stará presenter `DefaultPresenter`, který je součástí modulu *Front*. Na přihlašovací obrazovce se nachází klasický formulář, který je v presenteru vytvořen jako komponenta pomocí Nette Forms. Po úspěšném vyplnění formuláře je zavolána metoda `processLoginForm`, která posílá zadané údaje na autentizaci. Tu nevolá ručně – o proces přihlášení se postará metoda `login`, která je součástí třídy `Nette\Security\User`.

Pokud dojde k úspěšnému přihlášení, Nette přidělí uživateli *identitu* a uživatel je přesměrován na domovskou stránku.[46] Pokud není přihlášen, například kvůli špatně zadanému heslu, aplikace mu zobrazí konkrétní chybovou zprávu.

Formulář obsahuje zaškrtávací políčko s popisem *Uložit heslo* (vstupní prvek typu `checkbox`), který zde funguje jako přepínač doby expirace přihlášení (*session*). Ta je nastavena metodou `$user->setExpiration()` – v případě neoznačení pole je uživatel automaticky odhlášen po 90 minutách nečinnosti nebo zavření okna prohlížeče, pokud pole označí, automatické odhlášení proběhne až po dvou týdnech.

Zde jsem narazil na problém, že přihlášení nezávisle na nastavené délce expirace vypršelo již po 24 minutách. To je způsobeno implicitním nastavením serveru – Ubuntu a jiné distribuce založené na systému Debian spouští každých 25 minut *cron* úlohu v souboru `/etc/cron.d/php5`. Ta zajišťuje smazání cache v závislosti na proměnné PHP `maxlifetime`, která je implicitně nastavena na hodnotu 1440 sekund [62]. Problém jsem vyřešil nastavením session mimo implicitní adresář v konfiguračním souboru `config.neon`:

```
session:
  expiration: 14 days
  savePath: %appDir%/../sessions
```

6.4 Události

Pro práci s událostmi využívám knihovnu `Kdyby\Events` od Filipa Procházky [63]. Ta převádí všechny události, napsané formátem pro Nette `onNazevUdalosti`, na události globální. Na vyvolání těchto událostí čekají *listenery*, které jsou v případě `Kdyby\Events` definovány v samostatných třídách.

Třída `UserListener`, implementující `Kdyby\Events\Subscriber`, poslouchá jednotlivé uživatelské události, jako je přihlášení a odhlášení uživatele, úpravy profilu a spuštění nebo ukončení hry. Tyto události dále předává modelu `Event`, která na jejich základě zapisuje záznamy do tabulky událostí 6.1.4, z níž jsou získávány výstupy pro další statistiky.

[46] Pro domovskou stránku používám stejný presenter, jako pro přihlašovací formulář – rozlišování probíhá především na úrovni šablony.

Jednotlivé události, které listener poslouchá, jsou definovány v metodě `getSubscribedEvents()`. Ukázka principu:

```
public function getSubscribedEvents() {
  return array(
    'App\Module\Base\Presenters\BaseGamePresenter::onGameStart'
    // další listenery
  );
}
public function onGameStart($result) {
  $this->eventModel->saveGameStart($this->user, $result);
}
```

6.5 Společné rozhraní her

Všechny herní presentery dědí od abstraktní třídy `BaseGamePresenter`. Ta implementuje metody, které se starají o společnou komunikaci her, práci s jejich historií[47], variace obtížnosti a další.

6.5.1 Variace obtížnosti

Obtížnost je u každé hry nastavena URL parametrem `difficulty`, který je předán implicitní akci presenteru. Difficulty může být číslo v rozsahu 1–3, na jehož základě je z databáze pomocí třídy `Score` získána konkrétní variace, která se pro danou obtížnost a hru váže:

```
public function getDifficultyVariation($gameId, $difficultyId) {
  return $this->db->table('difficulty_variation')
    ->where('game_id', $gameId)
    ->where('difficulty_id', $difficultyId)
    ->fetch()->value;
}
```

Získaná hodnota je poté použita specificky podle presenteru dané hry – např. hra Melodické kostky tuto variaci používá pro získání určitého počtu předělových značek pro danou skladbu.

6.5.2 Signály presenteru

Po zahájení každé hry je herním skriptem odeslán HTTP požadavek, který oznamuje, že hra byla úspěšně načtena a spuštěna (kap. 6.5.3). Tento požadavek zpracovává signál `handleGameStart` vyvolávající událost, kterou přijímá `UserListener` (kap. 6.4).

[47] Historie je implementována pomocí *sessions* a je vysvětlena především u Melodických kostek v kap. 6.6.1.

Signál explicitně ukončuje činnost presenteru metodou `terminate()`, protože by jinak v rámci jeho životního cyklu [57] došlo k dalšímu zpracování až po vykreslení šablonou, což je pro tento případ nežádoucí. Od signálu `handleGameEnd` naopak klient očekává odpověď s výpočtem bodů a případnou informací, jestli uživatel dosáhl nového rekordu. Tu aplikace posílá ve formátu JSON:

```
public function handleGameEnd(array $result) {
  $gameId = $this->game->getByName($result['gameName'])->id;
  // výpočet bodů a zjištění rekordu
  $score = $this->score->processGameEndResult($this->user->getId(),
   ↪  $gameId, $result);
  // přidání počtu bodů pro záznam události
  $result['score'] = $score['score'];
  // vyvolání události
  $this->onGameEnd($result);
  // odeslání odpovědi prohlížeči a ukončení činnosti presenteru
  $this->sendResponse(new JsonResponse($score));
}
```

Po nějaké době jsem v rámci práce na aplikaci došel k poznatku, že se aplikace chovala nepředvídatelně a po načtení her zpracovávala dvakrát některé části herních presenterů. To bylo způsobeno tím, že se před volanými signály prováděla implicitní akce `actionDefault`. Toto chování je nutné obejít následujícím způsobem:

```
public function startup() {
  parent::startup();
  if ($this->isSignalReceiver($this, 'gameStart') ||
   ↪  $this->isSignalReceiver($this, 'gameEnd') ||
   ↪  $this->isSignalReceiver($this, 'gameForceEnd')) {
    $this->processSignal();
  }
}
```

6.5.3 Odesílání herních událostí

Posílání HTTP požadavků z herních skriptů jsem implementoval pomocí společné třídy `AjaxLogger`, která využívá skript `nette.ajax.js` [64], postavený na jQuery. Třída obsahuje metodu `sendResult` a jejím parametrem je adresa, na kterou má být požadavek zaslán, obsah odeslané informace a nepovinný callback funkce, které je při příjmu odpovědi předán výsledek požadavku ve formátu JSON. Příjemcem těchto požadavků jsou signály deklarované ve třídě `BaseGamePresenter` (kap. 6.5.2).

Tento callback je využíván při zobrazování výsledného počtu bodů a informace, zda uživatel dosáhl nového rekordu v dialogovém okně (kap. 6.5.6). Při výpisu výsledného počtu bodů využívám knihovnu `animateNumber` [65], která se stará o ani-

maci růstu čísla od nuly po získaný počet bodů; parametr 800 definuje délku animace v milisekundách a třetím parametrem je callback funkce, která se po dokončení animace stará o zobrazení informace o osobním nebo celkovém rekordu, v případě, že jej uživatel dosáhl.

Dále využívám vlastnost prohlížečů, která umožňuje zavolat libovolnou funkci při události zavření okna nebo přechodu na jinou stránku. Tímto způsobem ukládám informaci o opuštění hry uživatelem:

```
$(window).on('beforeunload', function() {
  if (!scope.gameSolved)
    logger.sendResult(gameForceEndHandler, getResult());
});
```

6.5.4 Výpočet času

Další společnou funkcí, kterou využívají všechny hry, je počítadlo času, deklarované třídou `GameTimer`, která obsahuje metody pro spuštění a zastavení stopek a metodou pro získání výsledného času. Ten lze získat ve formátu sekund nebo milisekund, v závislosti na parametru. Sekundové vyjádření času dostává žák na konci splněné hry, naopak milisekundy se ukládají do databáze a jsou využívány k přesnému výpočtu bodů.

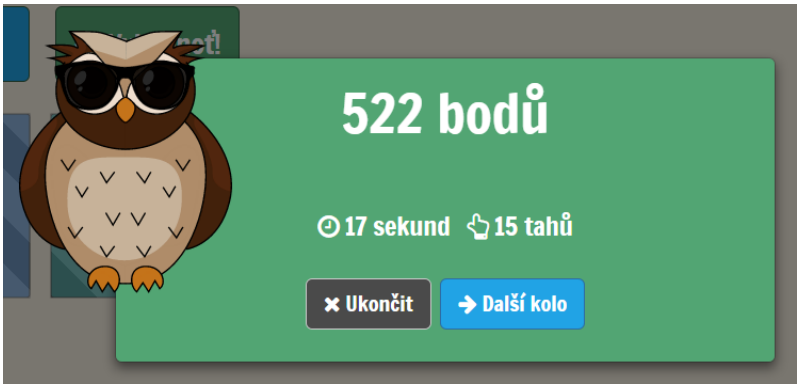
6.5.5 Výpočet bodů

Výpočet bodového ohodnocení (návrh v kap. 4.7.3) probíhá v modelové třídě `Score.php`. Po úspěšném splnění hry je z příslušného signálu (kap. 6.5.2) zavolána metoda `processGameEndResult`, která na základě přijatého záznamu vybere metodu, která se váže k dané hře a v argumentech jí předá proměnné pro výpočet pomocí konkrétního vzorce.

Po získání hotového počtu bodů metoda zjistí, zda je současná hodnota vyšší, než rekord uživatele a v případě kladného výsledku rekord v databázi aktualizuje. Na základě počtu bodů také zjistí, zda došlo k vytvoření nového rekordu v rámci celé hry a všechny tyto výsledky vrací zpět signálu, který se stará o jejich předání do záznamu událostí a do hry pro dialog s výsledkem.

6.5.6 Dialog s výsledkem

Všechny hry využívají společný dialog pro zobrazení výsledného počtu bodů a dalších ukazatelů úspěšnosti ve hře (obr. 22). Ten je uložen jako Latte šablona v souboru `solved.latte`. O jeho naplnění výsledky se stará skript, zavolaný po úspěšném vyřešení hry (kap. 6.5.3). Součástí dialogu je avatar se sovou, kterou si uživatel zvolil. Na základě počtu vyřešených kol za jednu hru se tomto dialogu zobrazuje výzva ke zvýšení obtížností – tato výzva se opakuje každých 7 kol.



Obrázek 22: Dialog s výsledkem hry

Melodické kostky a Posuvníky navíc využívají dialog upozorňující na špatnou odpověď `wrong.latte`, protože v těchto hrách uživatel ověřuje správnost svého řešení ručně. Tento dialog je vykreslen s výrazně červeným pozadím, které okamžitě evokuje špatnou odpověď.

6.6 Hra Melodické kostky

Melodické kostky (návrh v kap. 4.3) využívají hotových hudebních souborů, které jsou přehrávány pomocí knihovny *Howler.js* (kap. 5.2.3).

6.6.1 Presenter a model

Hudební podklady jsou získávány buď náhodně, nebo na základě vybrané skladby[48]. Náhodný výběr probíhá pomocí metody `getRandom` v modelové třídě `Song`.

V presenteru je dále zajištěno, že v dalších hrách nedochází k opakování již vyřešených skladeb. Při spuštění první a každé následující hry je do aktuálního *session* uloženo identifikační číslo skladby. Tato čísla jsou při spuštění každé nové hry předány modelu, který je při náhodném výběru vynechává. Pokud je spuštěna zcela nová hra z hlavní obrazovky, obsah session je vymazán a aplikace začne ukládat historii znovu. Jestliže uživatel dohraje všechny skladby, které jsou pro tuto hru připraveny, aplikace mu nabídne zopakování všech her.

6.6.2 Herní logika

Každá jednotlivá část skladby, která je reprezentována kostkou, je přehrávána knihovnou *Howler.js* jako sprite. Pole s definicí spritů je vytvořeno pomocí makra šablonovacího systému *Latte*, který Nette Framework využívá. Toto pole je spolu s URL hudebního souboru předáno objektu, v němž je přehrávač implementován.

[48] K výběru má přístup pouze administrátor, který tak může ověřovat práci s konkrétní skladbou.

Jakmile je celá skladba pro přehrávač načtena, je zavolána metoda `showGame`, která skryje indikátor načítání a dojde ke spuštění stopek.

V rámci metody `initButtons`, která inicializuje funkčnost jednotlivých kostek, také inicializuji přesouvání kostek mezi sebou (řazení). V původní implementaci jsem pracoval s knihovnou Sortable [66], která pro přesouvání využívá nativní *HTML5 drag and drop API*. Při testování aplikace na více zařízeních však knihovna vykazovala nižší výkon a v dotykové verzi Internet Exploreru 12 nefungovala vůbec. Z těchto důvodů jsem nakonec přistoupil k použití knihovny jQuery UI Sortable [67].

Při zahájení přesunu kostky dojde k zastavení přehrávání, pokud uživatel změní pozici kostky, počítadlo přesunů je inkrementováno. Kontrola správnosti hry je spuštěna ve chvíli, kdy uživatel klikne na tlačítko *Vyhodnotit*. Při testování aplikace jsem narazil na problém s nefunkčním přesouvání kostek v dotykové verzi prohlížeče Internet Explorer 10 a vyšší. Řešení spočívalo v přidání CSS parametru pro třídu kostky:

```
.single-cube {
  -ms-touch-action: none;
}
```

Tento parametr potlačuje běžné chování prohlížeče, které vykazuje při tažení prstem po obrazovce [68] – na místo posunu celé stránky je gesto správně použito pro vybrání kostky a zahájení jejího přesunu.

6.6.3 Grafické rozhraní

Každé pole je vytvořeno z blokového prvku `<div>`, který obsahuje tlačítko reprezentované elementem `<a>`. Uvnitř tlačítka je také prvek tzv. *spinneru*, který pomocí CSS3 animace ztvárňuje otáčející se šipky během přehrávání skladby (ikona šipek je využita z knihovny *Font Awesome*). Dále jsou zde dvě ovládací tlačítka s popisky, *Hrej* a *Vyhodnoť*.

Responzivní zobrazení je zde implementováno tak, že na menších obrazovkách se herní pole zmenšují a zmizí popisky u ovládacích tlačítek, která jsou také zmenšena. V příloze E je hotová grafická podoba této hry.

6.7 Hra Pexeso

Pexeso (návrh v kap. 4.4) využívá hotových hudebních souborů, které jsou přehrávány pomocí knihovny *Howler.js* (kap. 5.2.3).

6.7.1 Presenter a model

Hudební podklady jsou vybírány náhodně – presenter volá metodu `getRandom`, které v parametru předává požadovaný počet párů. Ten je určen na základě va-

riace obtížnosti (kap. 6.5.1). Pro každou ze skladeb jsou získány základní informace – název umělce a skladby, cesta k jejímu souboru a její délka. Právě délka je dále využívána v herní logice pro vytváření přesných předělů.

6.7.2 Herní logika

Každá skladba je definována jako prvek pole, který je předán knihovně *Howler.js*. V poli je předáno URL zvukového souboru a seznam spritů. Ty podle délky definují přehrávání celé skladby a přehrávání první a druhé poloviny.

```
{foreach $songs as $s}
songTitles[{$iterator->counter-1}] = {$s->artist} + ' - ' +
 ↪  {$s->title};
songPairs[{$iterator->counter-1}] = {
  src: ['/assets/sounds/songs/{$s->filename|noescape}.mp3'],
  sprite: {
    complete: [0, {$s->duration}],
    partA: [0, {$s->duration/2}],
    partB: [{$s->duration/2}, {$s->duration}]{sep},{/sep}
} }
{/foreach}
```

Objekt s herní logikou při svém vytvoření volá metody pro inicializaci jednotlivých tlačítek a promíchání herních polí. Promíchání pracuje pomocí jQuery doplňku `shuffle` [69]. Pomocí selektoru jsou vybrána všechna pole a funkce se postará o jejich náhodné pořadí.

Protože jsem v návrhu hry uvedl, že je požadováno označovat obě pole páru ve správném pořadí (kap. 4.4), kontroluji tuto správnost metodou `isPair`. Té jsou jsou v parametrech předány vlastnosti `id` obou polí, např. `song3A`. Tato `id` jsou porovnána podle jejich textové hodnoty bez posledního znaku a v případě, že se shodují, a poslední znaky jsou u nich v pořadí A–B, je správnost potvrzena.

6.7.3 Grafické rozhraní

Základ grafického rozhraní je totožný jako u hry Melodické kostky – herní pole jsou vykreslena stejným způsobem (kap. 6.6.3). Pexeso indikuje správně vybraný pár silným zvýrazněním polí. Při správném výběru také zobrazí autora a název vybrané skladby. V příloze F je hotová grafická podoba hry.

6.8 Hra Krokování not

Krokování not (návrh v kap. 4.5) nepracuje se zdroji, získávanými z databáze. Na straně serveru je vygenerována výchozí nota a kombinace posloupnosti kladných a záporných čísel – série posunů. Na základě nich jsou pak ve skriptu zjištěny správné noty.

6.8.1 Presenter a model

Metoda `getAssetsRandom`, která slouží ke generování série posunů, nejprve náhodně vybere výchozí notu jako číslo v rozsahu 97–103, což jsou ordinální hodnoty písmen *a*–*g*. Poté ve smyčce generuje sérii posunů pomocí o velikosti, kterou určuje číslo násobené proměnnou `$stepRatio` získanou z variace zvolené obtížnosti. Vygenerované posuny jsou funkcí vráceny v tomto formátu:

```
array('firstLetter' => "d",
     'shiftSigns'  => array("+4","+3","-2","-4"))
```

Třída herního presenteru obsahuje vlastnost `$noteBtoH`. V případě, že je rovna `true`, je nota *b* vždy nahrazena notou *h*. Hodnota vlastnosti `$noteBtoH` je dále posílána také hernímu skriptu pomocí proměnné předávané šabloně. Tímto způsobem jsem zajistil „zadní vrátka“ pro případnou budoucí lokalizaci do jiného jazyka – bude možné doplnit do nastavení volbu, která toto chování přepíná v závislosti na jazyku prostředí.

6.8.2 Herní logika

Výpočet správných not, které mají být žákem zadány po jednotlivých posunech, probíhá v rámci metody `initCorrectNotes`. S notou *h* se zde interně pracuje opět jako s notou *b*, protože je stupňování prováděno pomocí ordinálních hodnot malých písmen (stejně jako v případě generování základní noty na straně serveru):

```
for(var i = 0; i < noteCount; i++) {
  // základ posunu -> výchozí nota
  if (i == 0) var baseNote = firstLetter.toLowerCase();
  // základ posunu -> předchozí nota
  else var baseNote = correctNotes[i-1].toLowerCase();
  if (baseNote == 'h') baseNote = 'b';
  // výpočet posunu podle značky směru a čísla
  if (shiftSigns[i][0] == '+')
    var shiftedNote = baseNote.charCodeAt(0) +
     ↪  parseInt(shiftSigns[i].slice(1));
  if (shiftSigns[i][0] == '-')
    var shiftedNote = baseNote.charCodeAt(0) -
     ↪  parseInt(shiftSigns[i].slice(1));
  // ošetření přetečení přes nejvyšší nebo nejnižší notu
  while (shiftedNote < 97)
    shiftedNote = 103 - (97 - shiftedNote) + 1;
  while (shiftedNote > 103)
    shiftedNote = 97 + (shiftedNote - 103) - 1;
```

```
  // převod noty z ord. do znakové reprezentace
  correctNotes[i] = String.fromCharCode(shiftedNote);
}
```

Správnost zadaných not je ověřována při každé změně obsahu vstupních polí volanou funkcí. Tato funkce zajišťuje v případě zadání dalšího znaku automatické přepsání pole (uživatel nemusí poslední zadané písmeno mazat) a zvýrazňuje pole v závislosti na správnosti vstupu. Pokud uživatel zadá správnou notu, je označeno pro zadávání následující pole (označení je známo pod anglickým výrazem *focus*). Implementoval jsem zde také normalizaci vstupu – pokud žák zadá velké písmeno, je převedeno na malé. Pole neakceptuje, a tedy ani nezobrazí vstupy písmen a ostatních znaků, které nepatří do hudební abecedy.

6.8.3 Grafické rozhraní

Jednotlivá pole pro zadávání not jsou vytvořeny klasickými prvky `<input>` pro vstup textu, generovanými spolu s šipkami pomocí Latte makra procházení pole:

```
{foreach $shiftSigns as $sign}
  {$sign|shiftArrows|noescape}
  <input id="step-{$iterator->counter}" type="text" class="step-input"
  ↪  placeholder="?" data-pos="{$iterator->counter}">
{/foreach}
```

Filtr `shiftArrows`, který je zde použit, jsem implementoval v presenteru hry pro vykreslení určitého počtu symbolů šipek na základě aktuální hodnoty v poli. Samotné šipky jsou graficky vytvořeny pomocí symbolů z knihovny *Font Awesome*.

Pole jsou upravena kaskádovými styly – není zde zobrazen kurzor (aktivní pole je zvýrazněno barevným pozadím) a pole je výrazně větší, než je jeho základní velikost. Při zmenšení obrazovky prohlížeče se responzivně zmenší všechny vizuální prvky. Hra je tak pohodlně použitelná i na mobilních telefonech, aniž by uživatel musel během hraní posunovat viditelnou obrazovku (*viewport*). V příloze G je hotová grafická podoba hry.

6.9 Hra Posuvníky

Posuvníky (návrh v kap. 4.6) získávají jako zdroj notové zápisy, uložené v databázové tabulce `notation`; přehrávání je zpracováno knihovnou *MIDI.js* (kap. 5.2.4).

6.9.1 Presenter a model

Posuvníky využívají implementace historie na stejném principu, jako Melodické kostky (kap.6.6.1). Tím je vyloučeno, aby uživatel dostal stejnou notaci v dalším kole hry. Notový zápis je získáván náhodně z tabulky `notation`. Výběr probíhá

s podmínkou na délku notace, která je určena variací aktuální obtížnosti jako číselný rozsah:

```
$this->getAll()->where('length >= ? AND length <= ?', $min, $max);
```

6.9.2 Herní logika

Pro přehrávání notového zápisu jsem vytvořil třídu `NotationPlayer`. Ta je pro tuto hru použita ve dvou instancích – přehrávač originální skladby a přehrávač skladby, kterou vytváří uživatel. Objektu je při zahájení hry předán notový zápis v původní podobě, jak jsem navrhnul v kap. 4.8.2. Ten je zpracován ve smyčce pomocí regulárního výrazu, který z něj extrahuje parametry každé noty:

```
^([cdefgah]is|[cdfg]es|es|as|[cdefgah]#|[cdefga]b|[bcdefgah]|_)
([,]{0,7}|[\']{0,7})?(16|8|4|2|1)?([.t])?$
```

Vizualizace tohoto výrazu pomocí nástroje *Debuggex* [70] je v příloze H. Při zpracování notového zápisu jsou vytvořeny interní reprezentace pro každý tón v podobě pole, které obsahuje proměnné `key`, `length` a `multiplier`. `Key` obsahuje konkrétní MIDI číslo, kterým je nota reprezentována – to je zjištěno na základě převodní tabulky, která je součástí skriptu. Do `length` je uložena délka noty jako jmenovatel v podílu délky celého taktu (např. `4` pro $\frac{1}{4}$ taktu), která je dále ovlivňována násobkem `multiplier`, který bývá implicitně `1` – u trioly je roven `1/3`, u noty s tečkou `1,5`.

Při samotném přehrávání je již přímo využívána knihovna *MIDI.js* a její metody `noteOn` a `noteOff`. Přehrávání probíhá ve smyčce, kde je procházeno výše zmíněné pole. Časování délky jsem realizoval pomocí běžné funkce `setTimeout`.

Ve třídě jsem implementoval systém událostí – při každém přehrávání noty, zastavení noty, zastavení skladby a dalších událostech je možné zavolat vlastní callback funkci. Toho dále využívá herní skript – na základě událostí volá funkce, které se starají o přepínání stavu tlačítka *Přehrát/Stop*, zvýrazňování aktivního posuvníku podle přehrávané noty a další.

Herní skript pracuje s oběma objekty přehrávače not. Toho je využito především při vyhodnocování hráčem vytvořené kombinace posuvníků – aktuální pozice posuvníků jsou porovnány s původními tóny, které jsou uloženy v objektu notového přehrávače.

6.9.3 Grafické rozhraní

Posuvníky jsou vytvořeny pomocí knihovny *noUiSlider* [71]. Ta zajišťuje jejich použitelnost i na mobilních zařízeních a velmi dobře přizpůsobitelné grafické rozhraní. Pomocí systému událostí této knihovny zajišťuji změny not ve hře.

Pro větší názornost jsem potřeboval realizovat každý posuvník v jiné šířce, v závislosti na délce noty. Zde jsem narazil problém názornosti – noty jsou v délce

od celé po šestnáctinovou a samotné posuvníky samozřejmě nemohu zobrazovat v tomto lineárním rozsahu, protože by velikostní rozdíly mezi nimi byly příliš velké. Využil jsem možností CSS preprocesoru *LESS* a každou šířku posuvníku počítám dynamicky pomocí jednoduchého matematického výrazu:

$$\text{šířka} = \frac{140\text{px}}{\text{délka noty}^{0,5}} \tag{10}$$

Tento výraz je obsažen v LESS *mixinu*, což je funkce, která expanduje definovaný soubor pravidel s využitím zadaného parametru. Každou šířku zde odvozuji od základní hodnoty 140px. Zjednodušeně je výpočtový mixin implementován tímto způsobem:

```
.note-slider-mixin(@divided-by) {
  width: 140px / ( pow(@divided-by, 0.5));
}
.noUi-handle {
  &.length-16 { .note-slider-mixin(16); }
  &.length-8 { .note-slider-mixin(8); }
  &.length-4 { .note-slider-mixin(4); }
  // další noty
}
```

Podobným způsobem dále ošetřuji šířku posuvníku při použití tečkované noty. V příloze I je hotová grafická podoba hry.

6.10 Administrace

Administrační rozhraní aplikace jsem implementoval v presenterech, které jsou součástí modulu `Admin`. Pro všechny sekce, které vypisují seznam zdrojů k úpravě (uživatelské účty, skupiny, nahrané skladby aj.), jsem použil knihovnu Grido pro Nette Framework [72], která zajišťuje přímočarou implementaci tabulkového výpisu (datové mřížky) z databáze. Hlavní obrazovka administrace je zpracována stylem *dashboard* – je zde zobrazen přehled nejdůležitějších událostí s odkazy do samotných sekcí administrace. V příloze J je grafická podoba této obrazovky.

6.10.1 Statistiky

Všechny grafy, které používám ve statistických přehledech, jsou vykresleny do prvku `<canvas>` pomocí knihovny *Chart.js* [73]. V přehledu na úvodní obrazovce administrace je stručný seznam posledních her, které žáci vyřešili, s jejich bodovým ohodnocením. Dále je zde týdenní přehled ve formě sloupcového grafu, který zobrazuje, kolik her za tento nebo minulý týden žáci hráli (a nebo pouze zahájili a nedohráli). Posledním přehledovým prvkem je výsečový graf, který zobrazuje srovnání obliby jednotlivých her.

Data pro týdenní přehled jsou získána v metodě `getEventClassByWeek` třídy `Event`. Výběr událostí pro aktuální týden probíhá pomocí následujícího SQL požadavku:

```
$evtCount = $this->db->table('event')
  ->where('event_class_id', $eventId);
  ->where('event_time BETWEEN DATE_ADD(CURDATE(), INTERVAL
   ↪  2-DAYOFWEEK(CURDATE()) DAY) AND DATE_ADD(CURDATE(), INTERVAL
   ↪  7-DAYOFWEEK(CURDATE()) DAY)');
  ->select('COUNT(id) AS event_count, DAYOFWEEK(event_time) AS
   ↪  event_day')
  ->group('event_day')
  ->fetchPairs('event_day', 'event_count');
```

V proměnné `$eventId` je uloženo číslo typu události (v tomto případě události vyřešení hry). Požadavek ošetřuje základní stav, kdy je jako první den v týdnu definována podle amerického formátu neděle.

Učitel má v administraci dále přístup k záznamům posledních aktivit, kde vidí události přihlášení a odhlášení žáků a poslední splněné a nesplněné hry. Také má přístup do tabulky bodových výsledků žáků, které může filtrovat na základě tříd, a pro každého žáka vidí počty konkrétních vyřešených her. Při výběru konkrétního žáka se dostane do jeho herního profilu, kde jsou opět ve formě grafů zobrazeny: celkový počet bodů, statistika jeho nejoblíbenějších her a poměr splněných ku nesplněným hrám.

6.10.2 Nahrávání hudby

Presenter `SongsPresenter` zajišťuje obsluhu rozhraní pro nahrávání hudby a tvorbu značek. Pro nahrávání souborů používám knihovnu *FineUploader* [74]. Při vkládání nového hudebního souboru je zobrazeno pouze pole pro jeho výběr, po vybrání je soubor okamžitě nahrán a zpracování je předáno jako HTTP požadavek na signál `handleUploadFile`. Ten využívá třídy *UploadHandler*, která je součástí knihovny FineUploader, a soubor uloží do dočasného umístění.

Popis umístění souboru je odeslán metodě `save`, která se postará o přesunutí souboru do adresáře určeného pro hudební zdroje. Metoda soubor dále předá třídě `SongTagHandler`, která z něj pomocí knihovny `getID3` [75] extrahuje informace o jménu umělce, názvu skladby a délce v milisekundách. Výsledek je uložen do databáze a zároveň odeslán jako součást odpovědi signálu na původní HTTP požadavek.

Pole pro výběr souboru je skryto a zobrazí se vyplněný formulář s údaji jména umělce a názvu skladby (v případě, že byly u souboru zadány). Zároveň je pomocí funkce `XMLHttpRequest` stažen zpracovaný soubor, který uživatel nahrál; soubor ukládám do binárního tvaru (*blob*). Obsah tohoto souboru je předán knihovně

`Howler.js` pro jeho okamžité přehrávání [49] a také metodě `generateWaveform`, která s využitím knihovny *SoundCloud-Waveform-Generator* [76] vygeneruje do prvku `<canvas>` spektrogram pro danou skladbu.

Při zahájení přehrávání je ve spektrogramu zobrazen kurzor, který se pohybuje podle času skladby. Uživatel v momentu přehrávání vytváří kliknutím na tlačítko *Marker* nebo klávesou *M* značku pro aktuální pozici kurzoru. Opakovaným klikáním tedy docílí vytvoření značek pro všechna požadovaná umístění ve skladbě. Značky jsou stejně jako samotný spektrogram vykresleny na *canvasu* a jejich časy jsou vypsány v samostatném seznamu, kde je možné je mazat.

6.10.3 Tvorba notových zápisů

Notový zápis je vytvářen v podobném formuláři, jako u hudebních podkladů. Zápis uživatel zadává do textového pole (značka `<textarea>`). Zde je pomocí jQuery doplňku *highlightTextarea* [77] docíleno podbarvování zadaného zápisu na základě správnosti syntaxe, kterou doplněk ověřuje regulárním výrazem. V případě, že je některá z not špatně zadaná, je červeně zvýrazněna v samostatném poli.

Rozhraní využívá stejnou třídu pro přehrávání not, jako hra *Posuvníky* (kap. 6.9.2). Při každé změně notového zápisu je změna poslána objektu přehrávače not, který přepočítá interní reprezentaci tónů. Také je zde možné přímo modifikovat tempo a základní oktávu jejich výběrem ze seznamu.

6.11 Uživatelské rozhraní

V uživatelském rozhraní využívám v největší míře základní skupiny barev, které jsou definovány jako LESS proměnné v souboru `variables.less`. Barvy jsou výrazně kontrastní, časté použití hnědo-oranžové barvy má žákům evokovat les (pojítko s avatarem sovy). Obtížnosti jsou v této hře značeny na všech místech konstantě stejnými barvami – zelená, oranžová a červená. Tyto barvy jsou také definovány jako proměnné, odkud je dále expanduji do deklarace barev konkrétních prvků.

Uživatel má ve svém profilu na výběr ze šesti základních avatarů. Každý avatar je ztvárněn sovou s jinou grafickou variací. Základní sovu jsem vytvořil v programu Adobe Illustrator na základě postupu na internetové stránce *wikiHow* [78]. Pro další variace jsem použil volně šiřitelné vektorové prvky, které jsem přidal k základní sově.

Všechny obrázky sov jsou v aplikaci použity ve vektorovém formátu *SVG*, který má velmi dobrou podporu v prohlížečích [79]. Díky tomu je mohu libovolně zvětšovat bez ztráty kvality a soubor s grafikou je velmi malý.

Při komunikaci se žáky jsem využil písmo *Francois One* z Google Fonts [80], které má odlehčený a hravý charakter, přesto je velmi dobře čitelné. Pro aplikaci jsem vytvořil ikony ve formátu *favicon*, *Apple touch icon* a *mstile*, na kterých je v různých

[49] Howler.js implicitně nepodporuje přímé přehrávání binárních data. Problém jsem ošetřil použitím funkce prohlížeče `FileReader`, která pomocí metody `readAsDataURL` zpracuje obsah souboru jako datové URL (ve formátu base64), které jsem již předal knihovně ke zpracování.

velikostech zobrazena část sovy. Tyto ikony jsou použity např. při připnutí webové aplikace jako dlaždice v systému Windows 8.1.

6.12 Žebříčky hodnocení

Souhrn bodů pro všechny hry je zobrazen v rohu úvodní obrazovky, kde je umístěn avatar hráče. Uživatelské rekordy pro každou hru jsou vizuálně připojeny k tlačítku dané hry a barva pozadí u počtu bodů odkazuje na barvu, kterou je reprezentována daná obtížnost. Při ukázání kurzorem myši na tuto oblast se animací vysune odkaz *Tvůj rekord*, přes něj uživatel přejde na žebříček nejlepších hráčů pro danou hru.[50]

Žebříčky s bodovým výpisem nejlepších hráčů jsou zpracovány v presenteru `RatingPresenter`. Ten vytváří tabulky žebříčků (rozdělené podle obtížnosti) pomocí vícenásobných instancí komponenty `RatingChart`. Komponenta používá k vykreslení tabulky knihovnu *Grido* [72], kterou využívám také v administraci. Seznam nejlepších hráčů získává modelová třída `Score` ze stejnojmenné databázové tabulky pomocí jednoduchého seřazení na základě rekordů za danou obtížnost.

Na stránce s žebříčky jsou také zobrazeny stupně vítězů se třemi nejlepšími hráči – stupně vítězů jsou jednoduchý vektorový obrázek ve formátu *SVG*, který je vložen přímo do kódu. Každý hráč je zde reprezentován svým jménem, počtem bodů a vlastním avatarem.

6.13 Nasazení aplikace

Aplikaci jsem průběžně testoval v lokálním virtuálním serveru, který jsem spouštěl pomocí hypervizoru *VirtualBox* od společnosti Oracle. Ve fázích dokončování aplikace, kdy byla základní funkčnost již hotová, jsem zakoupil doménu *sluchohry.cz*, kterou jsem nasměroval na virtuální privátní server (VPS) připravený pro testovací provoz s žáky.

6.13.1 Konfigurace serveru

VPS používá systém Ubuntu 14.04. Tato verze systému využívá stabilní a méně aktuální programové balíčky, protože jsem se však rozhodl využít HTTP server *Nginx* v nejnovější verzi, musel jsem provést kompilaci zdrojového kódu. Nginx jsem zkompiloval spolu s modulem *PageSpeed* od společnosti Google [81], který zajišťuje několika způsoby optimalizaci rychlosti načítání stránek. Mezi optimalizace patří automatické spojení a komprese (*minifikace*) veškerých CSS a JavaScriptových souborů, rozšířená vyrovnávací paměť, optimalizace obrázků a další.

Velkou výhodou modulu PageSpeed, kterou jsem využil při testování s žáky, bylo automatické zneplatnění neaktuálních souborů s kaskádovými styly nebo skripty, které jsem za provozu potřeboval měnit (z důvodu přidávání funkcí nebo

[50] U dotykových zařízení je tento odkaz ve výchozím stavu již zobrazen – detekci provádím v souboru obecných skriptů aplikace `main.js`.

opravy chyb). Za běžné situace by bylo nutné vynutit u prohlížeče klienta stažení nové verze souboru např. doplněním URL parametru. Toto zajišťoval PageSpeed automaticky pomocí pravidelné kontroly změn v souborech.

Protože aplikace operuje s osobními údaji (uživatelská jména a hesla), nastavil jsem na serveru použití bezpečného protokolu HTTPS, na který jsem přesměrovával ze základního HTTP. Vycházel jsem z návodu Filipa Procházky [82] a získal jsem SSL certifikát od společnosti Simplia zdarma na 90 dní. Na základě návodu jsem v serveru Nginx nastavil použití protokolu SPDY, který umožňuje např. simultánní požadavky a kompresi hlaviček požadavků, takže dochází k dalšímu zrychlení odezvy stránek [83].

6.13.2 Dostupnost aplikace

Kompletní zdrojové kódy aplikace jsou k dispozici na přiloženém CD (příloha L). Na adrese `https://sluchohry.cz/` je v provozu produkční verze aplikace, která byla testována žáky. Pro účely výzkumu poznatků této diplomové práce je aplikace dostupná také na adrese `http://dp.fibek.cz/`[51] s přístupem pomocí následujících přihlašovacích údajů:

Uživatel administrátor:

- Přihlašovací jméno: dpadmin
- Heslo: KCVGAwTDYyuuve

Uživatel žák:

- Přihlašovací jméno: dpzak
- Heslo: DdlNUvXLKlWQCg

Demonstrované hudební zdroje v aplikaci mohou být chráněným obsahem třetích stran. Jména a IP adresy, které jsou v testovací verzi aplikace uloženy, jsou z důvodu ochrany osobních údajů anonymizovány. Záznamy o práci s aplikací, bodové výsledky a další data jsou reálná, která byla získána v průběhu používání aplikace žáky.

[51] Tato verze nezahrnuje podporu bezpečného protokolu HTTPS a souvisejících optimalizací.

7 Výsledky

V hotové aplikaci jsem vygeneroval sadu přístupových údajů pro vybrané žáky. Výsledky, které aplikace shromáždila během mého testování, jsem vymazal a nechal tak záznamy v aplikaci „čisté“. Veškeré hudební zdroje pro testování připravil Mgr. Martin Grobár, který aplikaci naplnil 63 melodiemi pro hry Pexeso a Melodické kostky a 28 notovými zápisy pro hru Posuvníky.

7.1 Uživatelské testování

Uživatelské testování proběhlo na Základní škole Tomáše Garrigua Masaryka Blansko, kde ji se žáky otestoval Mgr. Martin Grobár. Pro testování byly zvoleny ročníky 6. C a 6. B, jejichž žáci věkově patří do mnou provedené analýzy uživatele v kapitole 3.1.

7.1.1 Představení aplikace

Aplikace byla 27. 4. 2015 představena v hodině Hudební výchovy třídě 6. C, 30. 4. 2015 třídě 6. B. Žáci byli obeznámeni učitelem se základy jejího použití, byly jim demonstrovány jednotlivé hudební hry a vysvětlen jejich princip. Vyučující rozdal vygenerovaná přihlašovací jména pro všechny žáky a vyzval je k otestování přímo ve výuce, pokud disponují vlastním mobilním zařízením (škola poskytuje žákům volné wifi připojení).

Při testování na těchto zařízeních bylo hned prakticky zjištěno, že se objevují problémy s přehráváním hudebních podkladů na mobilním zařízení *Aligator S4700* (operační systém Android 4.2.2), kde zvuk vůbec nefungoval. To může být způsobeno nižší kompatibilitou zvukových API ve hrách, které jsou i přes vysokou podporu mnoha standardů stále do jisté míry omezeny a navíc mají vyšší požadavky na výkon zařízení.

Na konci hodiny byl žákům předložen dotazník (příloha K), který na místě vyplnili a odevzdali vyučujícímu. Žáci byli také informováni, že se bodování získané v této aplikaci může projevit do jejich hodnocení, kdy nejúspěšnější z nich dostanou diplom.

Z výsledků (součástí přílohy L) vyplývá spousta zajímavých zjištění. Žáky nejvíce zaujala hra *Melodické kostky*, naopak nejméně *Posuvníky*. Navzdory původním předpokladům byl první dojem z čistě teoretické hry *Krokování not* poměrně pozitivní.

Drtivá většina žáků je z principu výuky hrou nadšena – průměrná známka byla 1,3. Při hodnocení her dostáváme průměr 1,6; výkon aplikace 1,3 a grafika 1,4. Nedílnou součástí evaluace byla také personifikační složka aplikace, tedy uživatelský avatar – odlehčené *hlášky* sovy byly hodnoceny známkou 1,3, při subjektivním popisu vyučujícího bylo také zdůrazněno nadšení žáků z možností volit si převleky pro svého avatara.

Z pohledu platforem plánuje 10 žáků aplikaci otestovat na tabletu, 8 na mobilním telefonu a 8 na počítači. Toto je důležité vodítko, které koreluje s rozhodnutím pracovat s responzivním designem a snažit se o podporu co největší šíře zařízení.

V individuálním poli, které se žáků táže, co v aplikaci postrádají, bylo velmi často zmíněno chybějící přehrávání melodie v *Pexesu* po označení správné dvojice. Někteří žáci by také uvítali více her nebo dokonce týmové hry.

7.1.2 Reakce po představení

Již odpoledne po ukončení vyučování bylo v administraci aplikace zjištěno, že všichni oslovení žáci 6. C, kteří odpověděli kladně na otázku zájmu o domácí testování aplikace, se začali ještě ten den postupně přihlašovat a zkoušet jednotlivé hry.

Navzdory dotazníkovému šetření, kdy byl poměrně velký zájem o hru *Posuvníky*, bylo zjištěno, že jej málokterý žák vydrží hrát opravdu dlouho a někteří se jí vyhnuli úplně. Vyučující uvedl, že se této hře věnoval v hodině jako poslední a nezbylo na ni tolik prostoru, jako na ostatní. Tento nižší zájem je tedy možné přisuzovat nižší znalosti hry, ale také vyšší časovou náročností jednoho hraného úseku (kola) nebo dokonce vyšší náročnosti na hudební schopnosti žáka, kdy je nutné do hloubky přemýšlet o poslouchaných předlohách a dlouho zkoušet různé varianty tónů.

Naopak *Krokování not* vzbudilo překvapivě vysoký zájem. Je to možná proto, že hru žáci znalé hudební abecedy vnímají jako poměrně jednoduchý způsob, jak si nahrát vysoký počet bodů.

Je zajímavostí, že jedna žákyně vydržela aplikaci používat celkem hodinu a půl, kdy hrála v součtu všech her celých 96 kol.

7.1.3 Hodnocení po týdnu používání

Po jednom týdnu od představení byli žáci dotázání na shrnující poznatky z používání aplikace. Nejvíce žáků uvedlo, že hrou průměrně strávili 10–30 minut denně. Ze všech her se jim nejvíce líbilo pexeso, jako důvod uvedli například největší počet písní, které poznávali. Nejméně oblíbené byly Posuvníky, které hodnotilo záporně 10 žáků – mezi důvody uvedli, že hra byla nudná, nezáživná a obtížná. Jedna žákyně konkrétně uvedla zklamání, že měla napoprvé pouze 232 bodů.

V preferenci úrovní obtížnosti pro jednotlivé hry nejčastěji zvolili střední pro Melodické kostky, vysokou pro Pexeso, vysokou pro Krokování not a nízkou pro Posuvníky. Při dotazu na technické problémy byla jedna zmínka o problému funkčnosti Pexesa, jednomu žákovi nefungoval zvuk v mobilním telefonu u hry Posuvníky a jedna žákyně informovala, že si nastavila Pexeso na vysokou obtížnost, ale hra se spustila v nízké obtížnosti.

Při pohledu na platformy 9 žáků uvedlo, že nejčastěji aplikaci používali na osobním počítači nebo notebooku, 7 žáků na tabletu a 1 na mobilním telefonu. V dotazu, zda si žák myslí, že se v některé hře zlepšil, uvedlo 12 žáků kladnou odpověď, 5 zápornou; nejčastěji uvedli, že se zlepšili v Pexesu, ale výsledky byly velmi vyrovnané.

V otevřené otázce, co žáci doporučují pro zlepšení aplikace, nejčastěji uvedli více her a také více podob sovy na výběr. Podrobné výsledky dotazníkového šetření jsou součástí přílohy L.

7.2 Hodnocení výsledků

Při interpretaci výsledků jsem narazil také na zajímavý problém – v dotazníku někteří žáci hodnotili aplikace, přestože pro jejich uživatelský účet nebylo zaznamenáno jediné přihlášení. Je možné, že si žáci navzájem půjčovali svá zařízení a hráli pod společným uživatelským účtem. Toto je těžko řešitelný problém; jedním z možných způsobů řešení by mohl být pokročilejší motivační systém, který by více vybízel k soutěživosti mezi samotnými žáky. Pokud však žáci v dotazníku neuváděli pravdivou odpověď, jedná se o zajímavý výstup pro vyučujícího.

Ve výsledcích a statistice aplikace je zřejmé, že se hra Posuvníky nesetkala kvůli své obtížnosti s příliš kladným přijetím. Možné zjednodušení by šlo realizovat tak, že by posuvníky nabízely např. méně možností výběru špatných not a eliminovala tak nutnost vysoké přesnosti odhadu.

Žáci se nejvíce poptávali po doplnění dalších her, což by je jistě motivovalo k ještě aktivnějšímu používání aplikace. Velmi je zaujaly kreslené avatary sov, které si s oblibou vybírali, nelíbil se jim ale malý jejich výběr.

Z testování je zřejmé, že aplikace zacílila na oblasti, které žákům v hudební nauce chybí – jednoduché učení vnímání hudby hravou formou. Důležité také je, že ocenili příjemné a hravé uživatelské rozhraní, které responzivně funguje i na jejich vlastních mobilních zařízeních, a bodovou motivaci, kterou vnímali jako velmi důležitou.

7.3 Budoucí rozvoj

Pro použitelnost do reálné výuky by mohla aplikace více pracovat s rozvojem znalostí hudební teorie. Je možné implementovat testovou hru, která by na základě databáze otázek a možných odpovědí, zadaných učitelem, testovala teoretické znalosti žáka.

Aby byla aplikace atraktivnější pro žáky, je vhodné přidat další hry, které by je zaujaly. Žáky nejvíce bavilo Pexeso, pracující s více skladbami, které žáci poznávali – mohly by je tedy zaujmout další hry na podobném principu, např. poznávání hudebních žánrů podle skladeb.

Dalším prvkem, který by mohl žáky aktivně zapojit do hudební kreativity, by mohla být hra sloužící k vlastní tvorbě hotové hudby. Zde bych mohl pro implementaci využít hotovu třídu zpracovávající zápis not, a přidat k ní jednoduché uživatelské rozhraní, kde by žáci přesouváním symbolů not na notovou osnovu nebo polí na *piano roll* (kap. 4.2) vytvářeli vlastní posloupnost melodií, které by si mohli v rámci aplikace ukládat a např. porovnávat hotové melodie mezi sebou. Pro žáky, používající dotyková zařízení, by mohlo být zajímavé virtuální piano, které by mohlo být s tvorbou melodií provázáno.

V rámci motivace žáků lze aplikaci vylepšit *gamifikačním* systémem, kde by žáci dostávali *trofeje* např. na základě počtu vyřešených her, celkového počtu bodů, dosažení rekordu a dalších měřítek, které by byly sledovány a vyhodnocovány v rámci systému událostí. Tímto způsobem by bylo možné vylepšit jejich osobní profil, který by navštěvovali pro prohlížení svých ocenění a výsledků.

Z pohledu administrace učitele lze aplikaci vylepšit pokročilejším způsobem vyhodnocování výsledků, který by se přibližoval principům *Business intelligence* – např. zjišťování nejoblíbenějších hudebních podkladů a jejich žánrů, obliba her na základě věku a hudebních zkušeností žáka, vliv dosaženého počtu bodů na další motivaci žáka a další. Také by bylo možné implementovat pokročilejší systém událostí, kde by byly zaznamenány přesné kroky, které žáci vykonají v rámci používání her; záznamy by bylo možné zpětně přehrávat a analyzovat ze strany vyučujícího.

8 Závěr

V první části práce (kapitola 2) jsem analyzoval současný stav e-learningových aplikací pro podporu hudební výuky základních škol. Zjistil jsem, že při požadavku na české prostředí a podporu českého způsobu zápisu not je současná nabídka aplikací velmi omezená.

V kapitole 3 jsem provedl analýzu uživatelů a jejich funkčních a nefunkčních požadavků a na základě nich jsem v kap. 4 navrhl podobu aplikace. V návrhu jsem kladl důraz na jednoduchou navigační strukturu a uživatelské rozhraní použitelné i na mobilních zařízeních. Navrhl jsem jednotlivé hry a způsob vyhodnocování a motivace žáků, který pracuje s bodovým hodnocením na základě výpočtových vzorců.

V kapitole 5 jsem rozebral technologie a knihovny pro přehrávání zvuku na webových stránkách a na základě účelu použití pro jednotlivé hry jsem zvolil dvě knihovny, které jsem dále použil při implementaci. Aplikaci jsem implementoval (kap. 6) pomocí programovacího jazyka PHP a knihovny Nette Framework a hotovou implementaci jsem ve spolupráci s učitelem hudební výchovy otestoval se žáky 6. tříd základní školy. Aplikace je dostupná na adrese `https://sluchohry.cz/`.

Na základě hodnocení aplikace žáky vyplynulo, že největší zájem je o větší počet výukových her. Aplikace je v tomto směru připravena především na rozvoj o hry založené na hudebních notách nebo zvukových souborech. Jak ukázaly výsledky, pro základní motivaci žáků je bodové hodnocení dostatečné, ke zvýšení atraktivity aplikace a ještě vyšší motivaci by však bylo vhodné implementovat pokročilejší systém hodnocení, například získáváním virtuálních odměn podle dosažených výsledků.

Aplikace obsahuje základní systém pro vyhodnocování aktivit a dosažených výsledků, který učiteli zobrazuje dosažený počet bodů pro každého žáka a jeho herní aktivitu. Na základě těchto údajů se může učitel rozhodovat o dalším hodnocení žáka. Pro budoucí rozšíření by bylo vhodné zpracovat hodnocení komplexnějším způsobem, nabídnout například porovnání v rámci celých tříd a zobrazovat vývoj výsledků žáků vzhledem k času strávenému používáním výukových her.

Testování ukázalo, že nejméně žáky motivovala hra, která pro ně byla nejobtížnější a výsledku zde bylo dosaženo v nejdelším čase. Pro praktické použití je tedy nutné aplikaci rozvíjet výukovým směrem, ale klást také důraz na hratelnost, která dokáže přirozeně udržet pozornost žáků.

9 Literatura

1 NET APPLICATIONS.COM. *Desktop Operating system market share – January, 2014 to January, 2015* [online]. 2015 [cit. 2015-02-07]. Dostupný z WWW: ⟨http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpsp=2014&qpnp=2&qptimeframe=Y&qpcustomd=0⟩.

2 JOHNSON, Jeff. *Designing with the Mind in Mind*. 2nd. Amsterdam: Elsevier Inc., 2014. ISBN 9780124079144.

3 HORTON, William. *E-Learning by Design*. 2nd. San Francisco, CA: Pfeiffer, 2012. ISBN 9780470900024.

4 SENGPIEL, Eberhard. *Note names of musical notes keyboard piano frequencies* [online]. 2014 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.sengpielaudio.com/calculator-notenames.htm⟩.

5 EARMASTER APS. *EarMaster 6 - Ear Training and Sight-Singing Software* [online]. 1996–2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.earmaster.com/⟩.

6 MAKEMUSIC, INC. *MusicXML for Exchanging Digital Sheet Music* [online]. 2015 [cit. 2015-05-09]. Dostupný z WWW: ⟨http://www.musicxml.com/⟩.

7 MIDIMASTER-SOFTWARE. *Midimaster - music education software* [online]. 2015 [cit. 2015-05-09]. Dostupný z WWW: ⟨http://www.midimaster.de/indexe.htm⟩.

8 AVID TECHNOLOGY, INC. *Musition* [online]. 2015 [cit. 2015-05-10]. Dostupný z WWW: ⟨http://www.sibelius.com/products/musition/index.html⟩.

9 AVID TECHNOLOGY, INC. *Auralia* [online]. 2015 [cit. 2015-05-10]. Dostupný z WWW: ⟨http://www.sibelius.com/products/auralia/index.html⟩.

10 AVID TECHNOLOGY, INC. *Avid Acquires Sibelius, Expands Reach into Global Education Market* [online]. 2006 [cit. 2015-05-10]. Dostupný z WWW: ⟨http://www.sibelius.com/news/press100.html⟩.

11 RISING SOFTWARE AUSTRALIA PTY LTD. *Ear Training Software and Music Theory Software* [online]. 1996–2015 [cit. 2015-05-10]. Dostupný z WWW: ⟨http://www.risingsoftware.com/⟩.

12 THETA MUSIC TECHNOLOGIES, INC. *Theta Music Trainer — Online Ear Training Games* [online]. 2015 [cit. 2015-05-11]. Dostupný z WWW: ⟨http://trainer.thetamusic.com/en⟩.

13 LEDVINA, Marek. *Foriero Hudební Hry* [online]. 2010-2012 [cit. 2015-05-14]. Dostupný z WWW: ⟨http://www.foriero.com/cs/⟩.

14 WIEGERS, Karl; BEATTY, Joy. *Software Requirements*. 3. vyd. Redmond, Washington: Microsoft Press, 2013. ISBN 978-0-7356-7966-5.

15 WOODS, Ben. *By 2020, 90% of world's population aged over 6 will have a mobile phone: Report* [online]. 2014 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://thenextweb.com/insider/2014/11/18/2020-90-worlds-population-aged-6-will-mobile-phone-report/⟩.

16 TIDWELL, Jenifer. *Designing interfaces*. 2nd. Sebastopol, CA: O'Reilly, 2011. ISBN 9781449379704.

17 FROST, Brad. *Mobile-First Responsive Web Design* [online]. 2011 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://bradfrost.com/blog/web/mobile-first-responsive-web-design/⟩.

18 GUSTAFSON, Aaron. *Understanding Progressive Enhancement* [online]. 2008 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://alistapart.com/article/understandingprogressiveenhancement⟩.

19 HEWITT, Allan. Some features of children's composing in a computer-based environment: the influence of age, task familiarity and formal instrumental music tuition. *Journal of Music, Technology and Education* [online]. 2009, [cit. 2015-05-07], s. 5–24. Dostupný z WWW: ⟨http://strathprints.strath.ac.uk/id/eprint/8147⟩. ISSN 17527066.

20 GOWER, Lily; MCDOWALL, Janet. Interactive music video games and children's musical development. *British Journal of Music Education* [online]. 2012, roč. 29, [cit. 2015-05-07], s. 91–105. Dostupný z WWW: ⟨http://journals.cambridge.org/article_S0265051711000398⟩. ISSN 1469-2104.

21 KARDOS, Leah. How music technology can make sound and music worlds accessible to student composers in Further Education colleges. *British Journal of Music Education* [online]. 2012, roč. 29, [cit. 2015-05-07], s. 143–151. Dostupný z WWW: ⟨http://journals.cambridge.org/article_S0265051712000186⟩. ISSN 1469-2104.

22 VÝZKUMNÝ ÚSTAV PEDAGOGICKÝ V PRAZE. *Rámcový vzdělávací program pro základní vzdělávání* [online]. 2007 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPZV_2007-07.pdf⟩.

23 SCHONBRUN, Marc. *The Everything Music Theory Book: A Complete Guide to Taking Your Understanding of Music to the Next Level* [online]. 2. vyd. 2006 [cit. 2015-05-07]. ISBN 9781593376529.

24 PILHOFER, Michael; DAY, Holly. *Music Theory For Dummies*. 2015. ISBN 978-1-118-99113-8.

25 ROTHSTEIN, Joseph. *MIDI: A Comprehensive Introduction*. 1995. Computer Music and Digital Audio Series. ISBN 9780895793096.

26 RUDOLPH, T.E. *Teaching Music with Technology*. 2004. ISBN 9781579993139.

27 LILYPOND. *LilyPond – Music notation for everyone* [online]. 2003-2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.lilypond.org/⟩.

28 BOŘÁNEK, Roman. *Flash na ústupu: Mozilla nabídne implementaci v HTML5* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.root.cz/clanky/flash-na-ustupu-mozilla-nabidne-implementaci-v-html5/⟩.

29 FIRTMAN, Maximiliano. *Mobile HTML5 compatibility on iPhone, Android, Windows Phone, BlackBerry, Firefox OS and other mobile devices* [online]. 2014 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://mobilehtml5.org/⟩.

30 MOZILLA DEVELOPER NETWORK AND INDIVIDUAL CONTRIBUTORS. Media formats supported by the HTML audio and video elements. *Web developer guide – Mozilla Developer Network* [online]. 2005–2015, [cit. 2015-05-07]. Dostupný z WWW: ⟨https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats⟩.

31 TECHNICOLOR. *mp3licensing.com - FAQ* [online]. 2002–2009 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://www.mp3licensing.com/help/developers.html⟩.

32 COGLIATI, Josh. *US Patent Expiration for MP3, MPEG-2, H.264* [online]. 2011 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://www.osnews.com/story/24954/US_Patent_Expiration_for_MP3_MPEG-2_H_264⟩.

33 MICROSOFT. *WAV Audio Support – Internet Explorer Web Platform Status and Roadmap - status.modern.IE* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://status.modern.ie/wavaudiosupport⟩.

34 MOZILLA DEVELOPER NETWORK AND INDIVIDUAL CONTRIBUTORS. <audio> - HTML (HyperText Markup Language). *Web developer guide – Mozilla Developer Network* [online]. 2005–2015, [cit. 2015-05-07]. Dostupný z WWW: ⟨https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio⟩.

35 LUBBERS, Peter. *HTML5 : programujeme moderní webové aplikace*. Brno: Computer Press, 2011. ISBN 9788025135396.

36 MOZILLA DEVELOPER NETWORK AND INDIVIDUAL CONTRIBUTORS. Web Audio API. *Web developer guide – Mozilla Developer Network* [online]. 2005–2015, [cit. 2015-05-07]. Dostupný z WWW: ⟨https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API⟩.

37 SMUS, Boris. *Web audio API*. Sebastopol, CA: O'Reilly Media, 2013. Dostupný také z WWW: ⟨http://chimera.labs.oreilly.com/books/1234000001552/index.html⟩. ISBN 9781449332686.

38 NYMAN, Robert. *Using HTML5 Web Workers To Have Background Computational Power* [online]. 2010 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://robertnyman.com/2010/03/25/using-html5-web-workers-to-have-background-computational-power/⟩.

39 CHINNATHAMBI, Kirupa. *Vendor Prefixes and JavaScript — kirupa.com* [online]. 2012 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.kirupa.com/html5/vendor_prefixes_javascript.htm⟩.

40 MICROSOFT. *Web Audio API – Internet Explorer Web Platform Status and Roadmap - status.modern.IE* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://status.modern.ie/webaudioapi⟩.

41 SCHILLER, Scott. *SoundManager 2: JavaScript Sound For The Web* [online]. 2014 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.schillmania.com/projects/soundmanager2/⟩.

42 SEROTA, Raphael. *Wad.js Demo Song* [online]. 2014 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.codecur.io/us/songdemo⟩.

43 SEROTA, Raphael. *Wad* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://github.com/rserota/wad⟩.

44 SIMPSON, James. *howler.js at 2.0* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://github.com/goldfire/howler.js/tree/2.0⟩.

45 DEAL, Michael. *MIDI.js* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://github.com/mudcube/MIDI.js⟩.

46 NETTE FOUNDATION. *Nette Framework* [online]. 2008, 2015 [cit. 2015-05-09]. Dostupný z WWW: ⟨http://nette.org/⟩.

47 GITHUB, INC. *GitHub Student Developer Pack - GitHub Education* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨https://education.github.com/pack⟩.

48 THE ORGANIC AGENCY. *Apache vs Nginx – Performance Comparison* [online]. 2013 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://www.theorganicagency.com/apache-vs-nginx-performance-comparison/⟩.

49 ADERMANN, Nils; BOGGIANO, Jordi. *Composer* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://getcomposer.org/⟩.

50 BOWER. *Bower – A package manager for the web* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://bower.io/⟩.

51 OTTO, Mark; THORNTON, Jacob; BOOTSTRAP CONTRIBUTORS. *Bootstrap – The world's most popular mobile-first and responsive front-end framework.* [online]. 2015 [cit. 2015-05-09]. Dostupný z WWW: ⟨http://getbootstrap.com/⟩.

52 GANDY, Dave. *Font Awesome, the iconic font and CSS toolkit* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨http://fortawesome.github.io/Font-Awesome/⟩.

53 STEIGERWALD, Daniel. *Třídy, dědičnost a OOP v Javascriptu - III - Zdroják* [online]. 2010 [cit. 2015-05-14]. Dostupný z WWW: ⟨http://www.zdrojak.cz/clanky/oop-v-javascriptu-iii/⟩.

54 ORACLE CORPORATION AND/OR ITS AFFILIATES. *MySQL :: MySQL Workbench* [online]. 2015 [cit. 2015-02-05]. Dostupný z WWW: ⟨https://www.mysql.com/products/workbench/⟩.

55 NETTE FOUNDATION. *Database–Table — Nette Framework* [online]. 2008, 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://doc.nette.org/cs/2.3/database-table⟩.

56 SKRASEK, Jan. *Class Nette–Database–Conventions–DiscoveredConventions — Nette 2.3.1 API* [online]. 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://api.nette.org/2.3.1/Nette.Database.Conventions.DiscoveredConventions.html⟩.

57 NETTE FOUNDATION. *MVC aplikace & presentery — Nette Framework* [online]. 2008, 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://doc.nette.org/cs/2.3/presenters⟩.

58 NETTE FOUNDATION. *Konfigurace — Nette Framework* [online]. 2008, 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://doc.nette.org/cs/2.3/configuring⟩.

59 NETTE FOUNDATION. *Získávání závislostí — Nette Framework* [online]. 2008, 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://doc.nette.org/cs/2.3/di-usage⟩.

60 HALE, Coda. *How To Safely Store A Password* [online]. 2010 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://codahale.com/how-to-safely-store-a-password/⟩.

61 NETTE FOUNDATION. *Přihlašování & oprávnění uživatelů — Nette Framework* [online]. 2008, 2015 [cit. 2015-05-12]. Dostupný z WWW: ⟨http://doc.nette.org/cs/2.3/access-control⟩.

62 THE PHP GROUP. *PHP: Runtime Configuration - Manual* [online]. 2001-2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://php.net/manual/en/session.configuration.php`⟩.

63 PROCHÁZKA, Filip. *Eventy a Nette Framework - Filip Procházka* [online]. 2013 [cit. 2015-05-12]. Dostupný z WWW: ⟨`https://filip-prochazka.com/blog/eventy-a-nette-framework`⟩.

64 DOBEŠ, Vojtěch. *vojtech-dobes/nette.ajax.js* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`https://github.com/vojtech-dobes/nette.ajax.js`⟩.

65 BORISOV, Alexandr. *aishek/jquery-animateNumber* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`https://github.com/aishek/jquery-animateNumber`⟩.

66 KONSTANTIN, Lebedev. *RubaXa/Sortable* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`https://github.com/RubaXa/Sortable`⟩.

67 THE JQUERY FOUNDATION. *Sortable — jQuery UI* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`https://jqueryui.com/sortable/`⟩.

68 MORRIS, Charles. *Adapting your WebKit-optimized site for Internet Explorer 10* [online]. 2012 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://blogs.windows.com/buildingapps/2012/11/15/adapting-your-webkit-optimized-site-for-internet-explorer-10/`⟩.

69 PADOLSEY, James. *Shuffle DOM Elements — CSS-Tricks* [online]. 2010 [cit. 2015-05-14]. Dostupný z WWW: ⟨`https://css-tricks.com/snippets/jquery/shuffle-dom-elements/`⟩.

70 TOARCA, Sergiu. *Debuggex: Online visual regex tester. JavaScript, Python, and PCRE.* [online]. 2013 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://www.wikihow.com/Draw-an-Owl`⟩.

71 GERSEN, Léon. *noUiSlider - Getting Started — Refreshless.com* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://refreshless.com/nouislider/`⟩.

72 BUGYÍK, Petr. *Grido* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://o5.github.io/grido-examples/`⟩.

73 DOWNIE, Nick. *Chart.js — Open source HTML5 Charts for your website* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://www.chartjs.org/`⟩.

74 WIDEN. *Fine Uploader* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://fineuploader.com/`⟩.

75 HEINRICH, James. *getID3() - The PHP media file parser* [online]. 2014 [cit. 2015-05-14]. Dostupný z WWW: ⟨`http://www.getid3.org/`⟩.

76 AHMED, Adnan. *SoundCloud-Waveform-Generator* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://github.com/Idnan/SoundCloud-Waveform-Generator⟩.

77 SOREL, Damien. *jQuery highlightTextarea* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨http://mistic100.github.io/jquery-highlighttextarea/⟩.

78 WIKIHOW. *How to Draw an Owl (with Pictures) - wikiHow* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨http://www.wikihow.com/Draw-an-Owl⟩.

79 DEVERIA, Alexis. *Can I use... Support tables for HTML5, CSS3, etc* [online]. 2015 [cit. 2015-05-07]. Dostupný z WWW: ⟨http://caniuse.com/⟩.

80 ADAMS, Vernon. *Google Fonts Francois One* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://www.google.com/fonts/specimen/Francois+One⟩.

81 GOOGLE. *PageSpeed Module - PageSpeed Module – Google Developers* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://developers.google.com/speed/pagespeed/module⟩.

82 PROCHÁZKA, Filip. *Nginx + HTTPS + SPDY + HSTS = Security++* [online]. 2014 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://filip-prochazka.com/blog/nginx-https-spdy-hsts-security⟩.

83 GOOGLE. *SPDY - Protocols – Google Developers* [online]. 2015 [cit. 2015-05-14]. Dostupný z WWW: ⟨https://developers.google.com/speed/spdy/⟩.

PŘÍLOHY

A Podrobné bodové srovnání aplikací

Název programu	**Platforma**	**Výukové možnosti**	**Uživatel. přívětivost**	**Rozhraní pro učitele**	**Česká lokalizace**	**Cena**	
Earmaster Pro 6	2	3	3	5	2	4	
ScoreTrainer Pro 8	1	1	1	2	3	3	
Musition 4 a Auralia 4	3	5	3	5	2	2	
Theta Music Trainer	4	5	4	4	0	5	
Váha kritéria	**2**	**4**	**3**	**3**	**5**	**3**	**Výsledné hodnocení**
Earmaster Pro 6	4	12	9	15	10	12	**62**
ScoreTrainer Pro 8	2	4	3	6	15	9	**39**
Musition 4 a Auralia 4	6	20	9	15	10	6	**66**
Theta Music Trainer	8	20	12	12	0	15	**67**

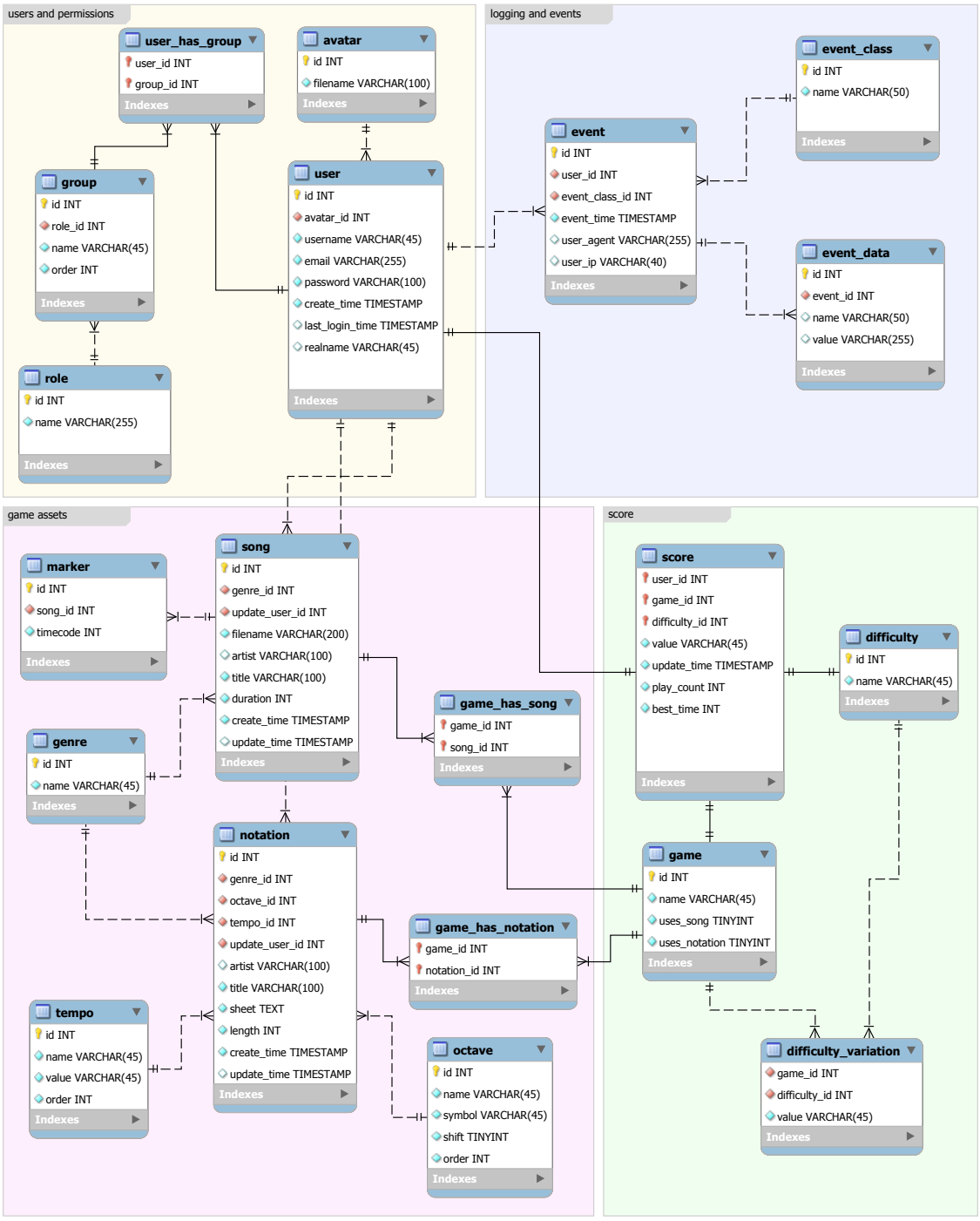
B Podpora HTML5 Audio v prohlížečích

Podpora HTML5 Audio v prohlížečích [79] (7. 5. 2015)

C Podpora Web Audio API v prohlížečích

Podpora Web Audio API v prohlížečích [79] (7. 5. 2015)

D Diagram entit a vztahů aplikace

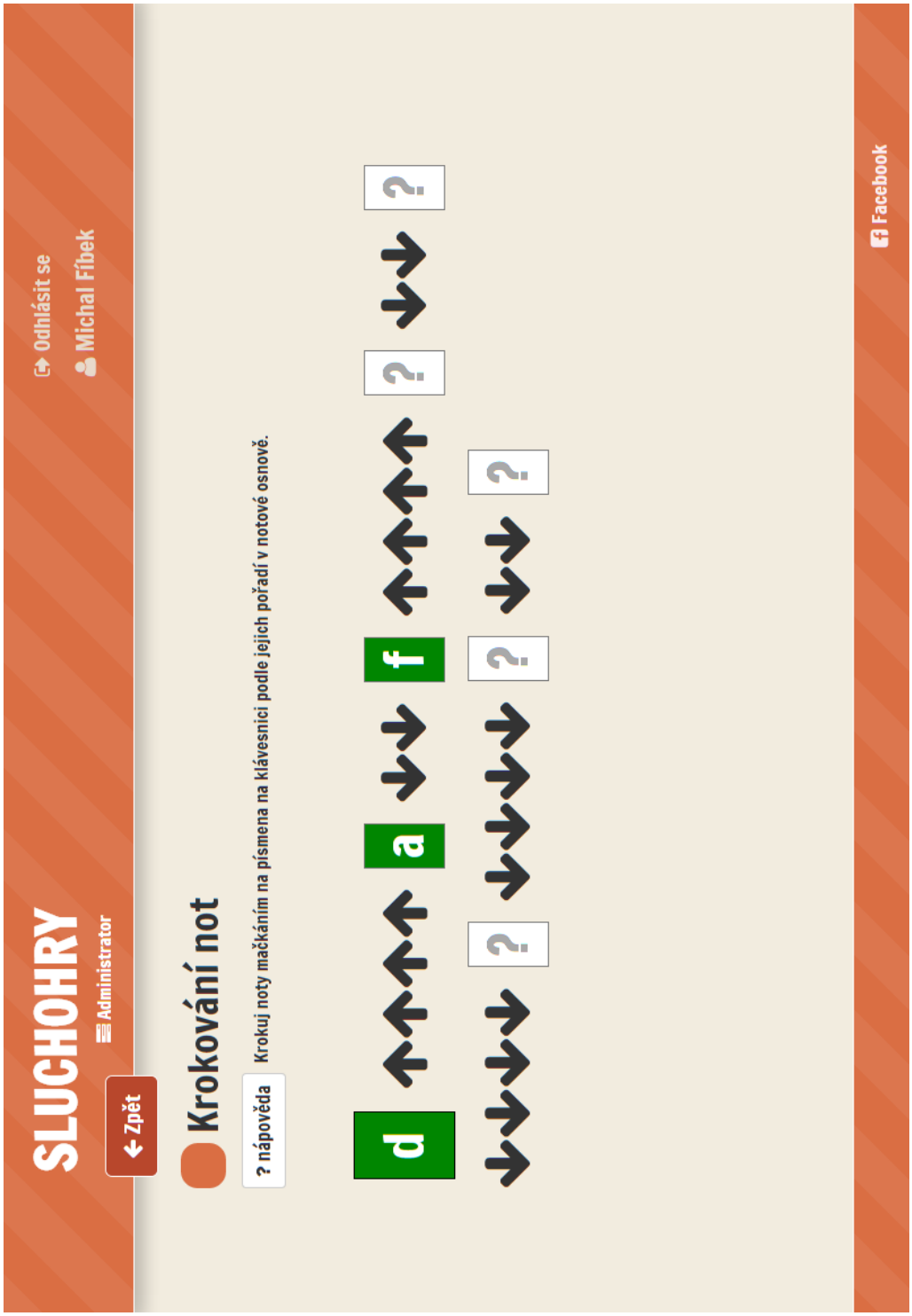


E Hra Melodické kostky

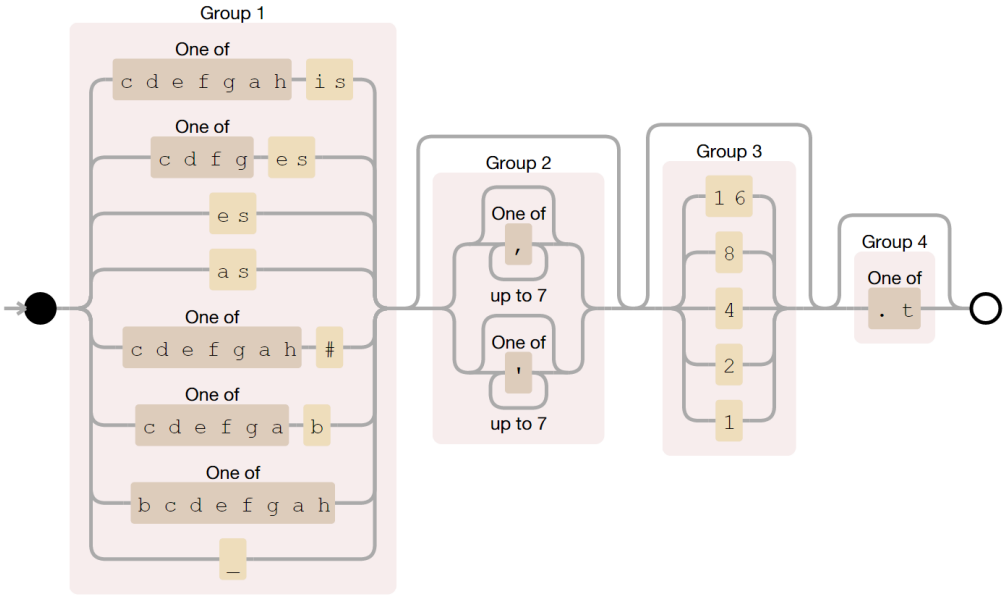
F Hra Pexeso

SLUCHOHRY
Administrator
Odhlásit se
Michal Fibek
← Zpět
Hudební pexeso
? nápověda
Najdi kompletní písničku schovanou pod dvěma kostkami a označ je ve správném pořadí.
Black Violin - A-Flat

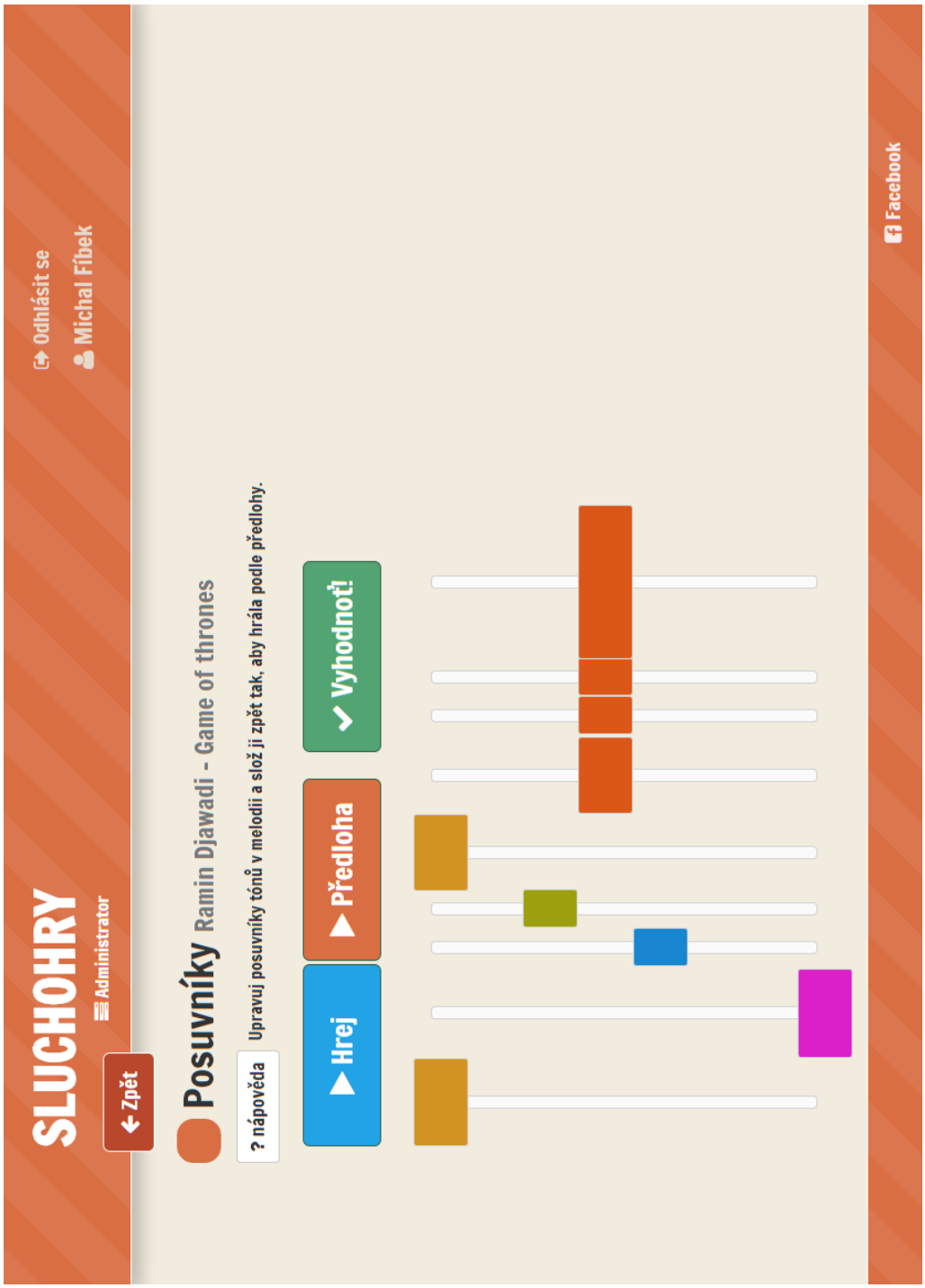
G Hra Krokování not



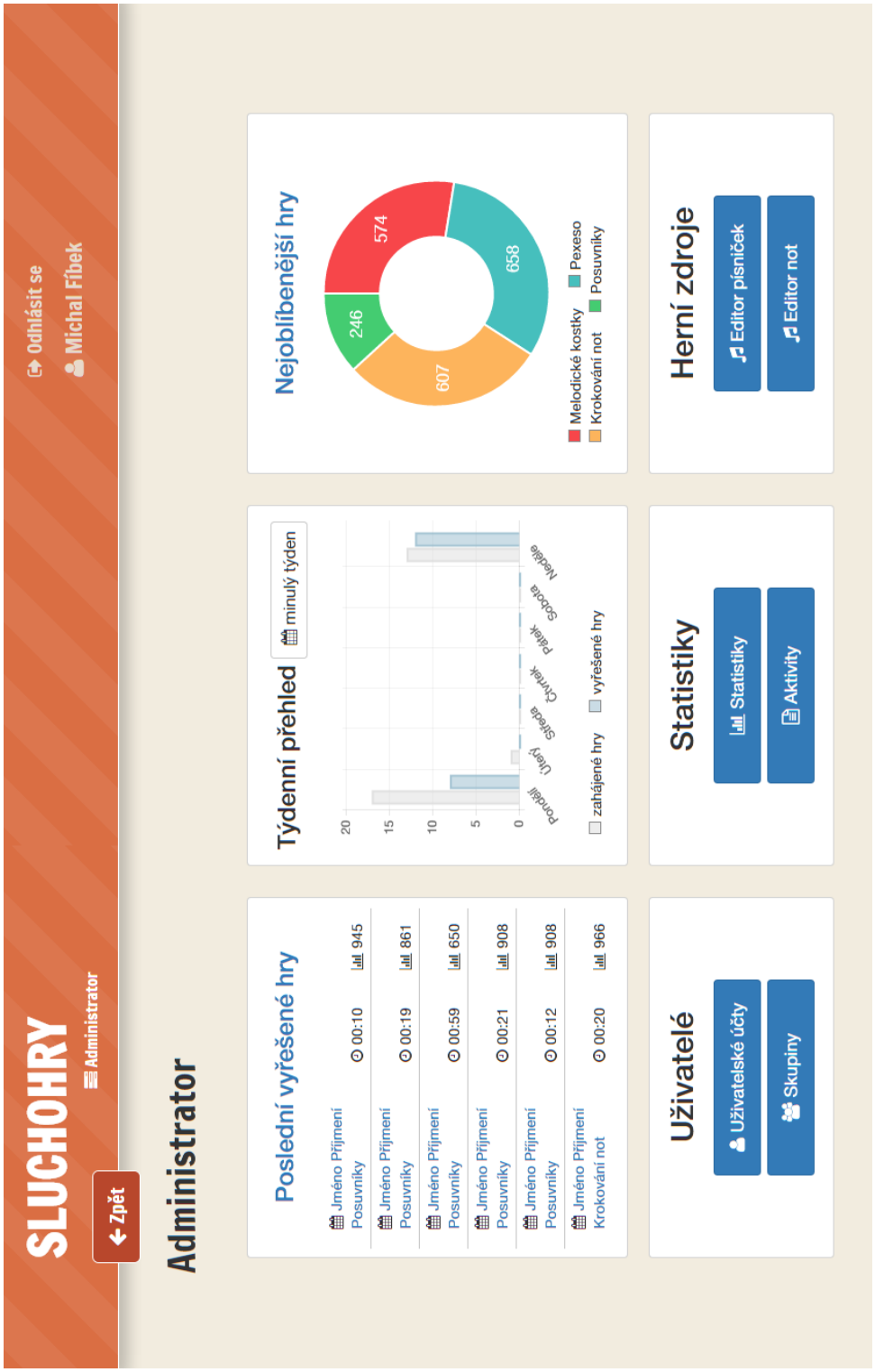
H Vizualizace regulárního výrazu pro zpracování zápisu not



I Hra Posuvníky



J Administrace – úvodní obrazovka



K Dotazník – hodnocení výuky žákem

Hodnocení výuky žákem

Jsi vybrán jako první hodnotitel hudební pomůcky sluchohry.cz. Prosím tě, aby ses zamyslel nad otázkami, které ti předkládám a svými odpověďmi ji pomohl zlepšit. Čeká tě jen šest otázek a vyplnění tak zabere jen pár minut.

Jméno a příjmení: _____

Třída: _____ Věk: _____ Nástroj: _____

1. Jaký máš celkový dojem z webové aplikace sluchohry.cz?

☐ může něco naučit a do HV patří.

☐ je to hloupost, která v HV jen otravuje. Proč? _____

2. Jsi rozhodnut vyzkoušet své schopnosti pomocí sluchoher také doma?

☐ ANO ☐ NE

3. Na jakém zařízení si hru doma vyzkoušíš? (více odpovědí)

☐ Mobil

☐ Tablet

☐ Počítač

4. Která hra tě zaujala nejvíce?

☐ Melodické kostky

☐ Pexeso

☐ Krokování

☐ Posuvníky

5. Hodnoť jednotlivé prvky aplikace sluchohry.cz?

	😀	🙂	😐	😕	😠
myšlenka výuky hrou	☐	☐	☐	☐	☐
hra melodické kostky	☐	☐	☐	☐	☐
hra pexeso	☐	☐	☐	☐	☐
hra krokování	☐	☐	☐	☐	☐
hra posuvníky	☐	☐	☐	☐	☐
bodování výkonu	☐	☐	☐	☐	☐
grafika	☐	☐	☐	☐	☐
hlášky sovy	☐	☐	☐	☐	☐
adaptabilní design[1)]	☐	☐	☐	☐	☐
výběr hudebních ukázek	☐	☐	☐	☐	☐

6. Co ve hře postrádáš?

_____

_____

_____

_____

1) Hra funguje na všech digitálních zařízeních - na mobilech, tabletech, notebookách a osobních počítačích.

L CD se zdrojovým kódem aplikace

Disk, který je přiložen k této práci, obsahuje následující adresáře:

- `aplikace` - kód aplikace
- `databaze` - struktura a testovací data aplikace
- `prace` - text diplomové práce ve formátu PDF
- `dotazniky` - dotazník s finálními výstupy testování
- `screenshoty` - snímky vybraných obrazovek v aplikaci