Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

# Grafické uživatelské rozhraní a front-end webové aplikace pro práci s notovými zápisy

***Vojtěch Hejda***

Vedoucí práce: Mgr. Petr Matyáš

11. května 2015

# Poděkování

Panu Petru Matyášovi, vedoucímu této práce, za podporu, nadšení pro věc a za to, že dal dohromady skupinu skvělých lidí, kteří se na tomto projektu podíleli. Mohitu Muthannovi Cheppudirovi za to, že stvořil VexFlow, a umožnil tak tisícům lidí po celém světě, včetně mě, pracovat s úžasně vypadající hudební notací.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 11. května 2015 .....

### Odkaz na tuto práci

Hejda, Vojtěch. *Grafické uživatelské rozhraní a front-end webové aplikace pro práci s notovými zápisy.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Tato práce se zabývá návrhem a implementací front-endu webové aplikace pro práci s hudební notací. Hlavní důraz je kladen na tvorbu interaktivního editoru notových zápisů pomocí moderních webových technologií.

Výstupem této práce je webová aplikace pro tvorbu a správu notových zápisů, obsahující grafický editor, který ověřuje použitelnost vybraných technologií v této oblasti softwarového inženýrství.

**Klíčová slova** grafické uživatelské rozhraní, editor, hudební notace, webová aplikace

# Abstract

This bachelor thesis focuses on designing and implementing music notation web based application‘s front-end. Main concern is in creating an interactive music sheet editor using the lastest web technologies.

Output of this paper is a web application for formation and management of music sheets, containing a graphical-interface editor. This editor attests usability of chosen technologies in this area of software engineering.

**Keywords** graphical user interface, editor, music notation, web application

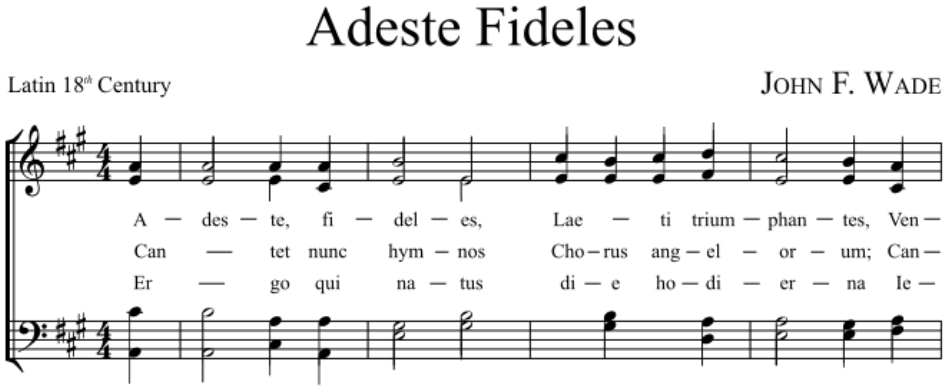
# Obsah

# Seznam obrázků

# Úvod

V dnešním světě je hudba minimálně stejně důležitá, jako kdykoliv jindy v historii lidstva. Díky moderním technologiím ji navíc můžeme poslouchat doma, v autě nebo třeba i při běhání v parku z nějakého minimalistického hudebního přehrávače.

Hudební notace se, na rozdíl od trendů v moderní hudbě, za posledních několik stovek let prakticky nezměnila. Je natolik známá, že se symbol noty (či jejich seskupení) z klasické hudební notace stal nejčastějším grafickým zobrazením pojmu „hudba“ a čímkoliv s ním souvisejícím. Tomuto symbolu totiž rozumí i lidé, kteří se o hudbu aktivně vůbec nezajímají; stal se jejím synonymem.




Obrázek 0.1: Příklad klasické hudební notace. [1]

Kromě hudby samotné se tento motiv přenesl i do obecné roviny zvuku jako takového, jakkoliv nemusí s hudbou přímo souviset. Když má dnes člověk, zodpovědný za návrh uživatelského prostředí, navrhnout vhodný piktogram k ovládacím prvkům, jakými je např. ztlumení či nastavení zvuku, bez váhání

volí nějakou kombinaci obsahující symbol noty.

Počet lidí zabývajících se tvorbou hudby (ať už profesionálně či amatérsky) je nyní obrovský. S tím souvisí i rostoucí poptávka po co nejjednodušších způsobech kreativního vyžití v této oblasti. Lidé si v dnešní hektické době již nechtějí kupovat notový papír a zaznamenávat svá díla tužkou. Raději vytáhnou notebook či tablet, a pomocí chytrých aplikací a internetu mohou svou tvorbu během pár minut nejen zaznamenat, ale i rozšířit do celého světa. A tak se i hudební notace posunula do nové éry; éry digitální.

## Motivace

Tato sekce je rozdělena do dvou částí. Neformální úvod popisuje mou osobní motivaci pro práci na tomto tématu, kdežto druhá část, požadavky na aplikaci, již formálně popisuje motivaci pro vznik této aplikace jako celku.

### Neformální úvod – osobní motivace

Tímto tématem by se nezabýval nikdo, kdo sám nemá kladný vztah k hudbě. Již od malička jsem hudbu miloval a byl jsem vychováván jako aktivní muzikant. Hrál jsem na různé hudební nástroje (od flétny až po piano) a také jsem zpíval v několika dětských a školních sborech. S hudební notací jsem přicházel do styku prakticky celý svůj život a noty jsem se naučil zapisovat již v předškolním věku.

S notovými zápisy se setkávám denně, například při hře na klavír. Už si ani napamatuji, kdy jsem naposledy zapisoval noty ručně. I já jsem podlehl duchu moderní doby, a při potřebě tvorby či úpravy notových záznamů sahám po různých počítačových nástrojích.

Již několik let se také amatérsky zabývám počítačovou grafikou a grafickým designem. Můj pohled na svět je tímto poznamenán, takže když používám všemožné aplikace pro práci s hudební notací, automaticky přemýšlím, jak tyto programy vykreslují notové záznamy. Používají vektorovou grafiku, či bitmapy? Skládají noty ze statických obrázků, nebo je dynamicky vykreslují? A umožňuje tato technologie interaktivní práci s těmito záznamy v reálném čase?

Když jsem na webu závěrečných prací FIT ČVUT zahlédl rámcové téma zabývající se prací s hudební notací v počítačovém světě, věděl jsem, že je to něco pro mě. Ukázalo se, že nejsem sám, na informační schůzku se zadavatelem, Mgr. Petrem Matyášem, nás dorazilo hned několik, motivovaných naším společným zájmem o hudbu ve spojitosti s informačními technologiemi.

Takto vznikl tým 7 studentů ČVUT. Za cíl máme vytvoření webové aplikace pracující s notovými zápisy (ve formě šesti bakalářských a jedné diplomové práce).

### Požadavky na aplikaci

Primární požadavky jsou podřízeny formě a také účelu aplikace. Výsledná aplikace bude tvořena moduly (jednotlivé závěrečné práce), čímž bude zajištěna částečná samostatná funkčnost těchto prací (a do určitého bodu také nezávislý vývoj), a zároveň i možná budoucí rozšiřitelnost o další rozličné moduly (např. případné budoucí závěrečné práce). Modulární architektura aplikace je zároveň výhodná vzhledem k samotnému vývoji aplikace – snadnější testování, nasazování, separace odpovědností za jednotlivé funkce.

### Účel aplikace

Účel tvorby této aplikace by měl být především vzdělávací, a to jak pro její uživatele, tak pro samotné tvůrce. Aplikace si neklade za cíl masové rozšíření mezi veřejnost a nasazení výsledné aplikace na stálý server není jejím primárním cílem.

U většiny modulů jde hlavně o teoretické ozkoušení technologií, postupů či algoritmů, které se při tvorbě programů či aplikací zabývajících se problematikou hudební notace používají.

Jednotlivé moduly reflektují požadavky zadavatele. Aplikace by měla umět pracovat s různými formáty souborů ukládajících hudební notaci (např. *MusicXML*, *MIDI* či *GuitarPro*), vzájemně porovnávat skladby na podobnost melodií a přehrávat zvuk k jednotlivým notovým záznamům. Noty budou tvořeny, vykreslovány a upravovány přímo ve webovém prohlížeči, což je jedním z hlavních předmětů této konkrétní závěrečné práce.

Kapitola 1

# Cíl práce

Cílem této bakalářské práce je návrh, implementace a otestování webového front-endu a grafického uživatelského rozhraní (zkráceně GUI) výše popsané aplikace. Ta by měla být přenositelná a spustitelná na většině zařízení s webovým prohlížečem. Tomu bude podřízen návrh front-endu a jeho implementace, i použité technologie.

Standardní hudební notace obsahuje nepřeberné množství prvků a způsobů zápisu, které není možné zpracovat kompletně v rámci jedné závěrečné práce jedním člověkem. Tato práce si proto klade za cíl použití pouze vybrané podmnožiny prvků z této notace, a to především pro účely ozkoušení vhodnosti použitých technologií pro toto odvětví softwarového průmyslu.

Aplikace by zároveň měla poskytovat i nějaké praktické užití, jakkoliv je její primární účel akademický. Neměla by proto uživatele nijak omezovat v jejím používání, např. nějakými prázdnými ovládacími prvky či nedokončenou funkcionalitou. Hlavní bod této práce, editor hudební notace, bude uživateli umožňovat vytvoření, uložení a případně i sdílení notového zápisu a jeho metadat s dalšími uživateli aplikace.

Kapitola 2

# Současný stav řešení problematiky

Aplikací zpracovávajících a zobrazujících notové zápisy existuje spousta, ať už placených, či zdarma k použití. Většinou se však jedná o poměrně robustní, desktopové aplikace, jejichž složité uživatelské rozhraní může spoustu příležitostných uživatelů odradit od jejich používání.

Existují i jednoduché aplikace, webové i jiné (např. pro chytré telefony), ale ty jsou zpravidla vyvíjené jednotlivci a jejich funkční možnosti jsou obvykle velice omezené nebo jednoúčelové, nehledě na malou portabilitu. To se týká především aplikací zaměřených na „smartphony“ či tablety, které jsou většinou nepřenosné na jiné operační systémy a jejichž cílovou uživatelskou skupinou není profesionální muzikant.

Tato kapitola je strukturována od obecného ke konkrétnějšímu a je rozdělena na tři sekce: *Desktopové aplikace*, *Webové aplikace* a *Aplikace používající podobné technologie*. V poslední jmenované jsou představena řešení, která, stejně jako tato aplikace, používají k vykreslování hudební notace renderovací API VexFlow. Samotné API a další užité technologie podrobněji popisuje kapitola *Zvolené technologie*.

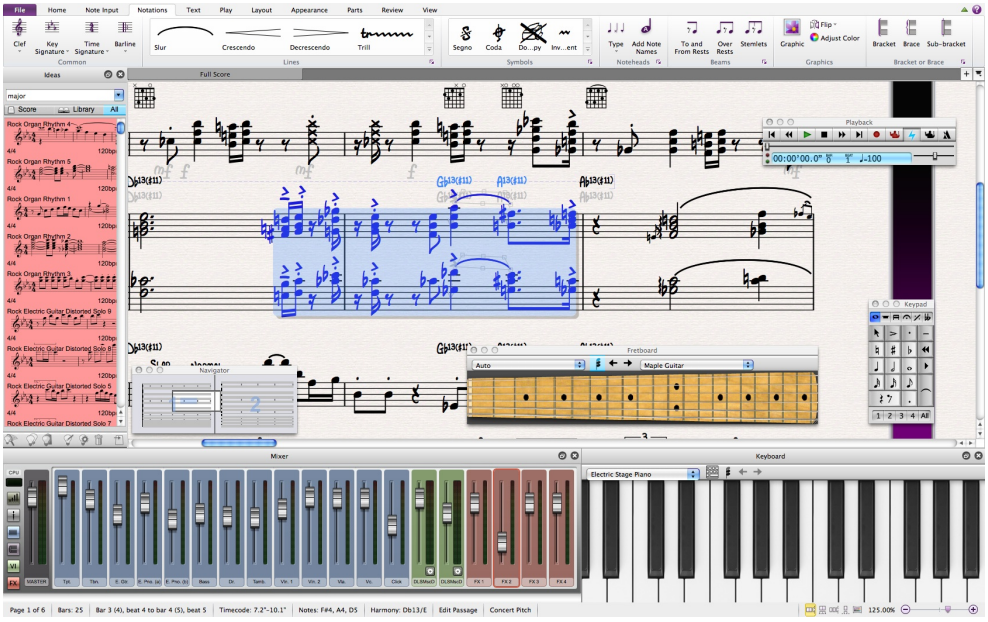
## 2.1 Desktopové aplikace

V oblasti desktopových aplikací existují desítky alternativ, od malých open-sourcových projektů, jejichž instalace nezabírají více než pár MB, po robustní placený software zahrnující kompletní zvukové knihovny a WYSIWYG editory. Použitými vykreslovacími technologiemi těchto programů se zde zabývat nebudu, jelikož předmětem této práce je webová aplikace, které v této oblasti používají odlišné technologie. Uvedu pouze pár majoritních programů, k dokreslení kontextu oboru elektronické hudební notace.

### 2.1.1 Placený software

V té druhé kategorii mají dlouhodobě pravděpodobně největší podíl na trhu profesionální programy **Finale** [3] a **Sibelius** [4]. Přesná čísla prodejů nejsou k dispozici, ale naznačuje to počet webových stránek a článků, které se jimi zabývají, a jemuž se neblíží žádný další placený software. Jejich vzájemnou rivalitu není těžké dohledat. Na webových stránkách obou aplikací je vám přímo vysvětlováno, proč je tato aplikace lepší než ta druhá, konkurenční.

Většina profesionálních aplikací, jak již označení *profesionální* vypovídá, je určena především hudebníkům-profesionálům, kteří je používají na denní bázi pro tvorbu rozsáhlých a komplexních děl. Jejich uživatelská rozhraní jsou komplikovaná, což je při jejich obsáhlé funkcionalitě pochopitelné. I tato rozhraní však procházejí v nejnovějších verzích většími i menšími změnami, především za účelem přehlednějšího a intuitivnějšího ovládání. Uživatelské rozhraní programu Sibelius ukazuje *Obrázek 2.1*.



Obrázek 2.1: Uživatelské rozhraní programu **Sibelius v7.5**

### 2.1.2 Volně dostupný software

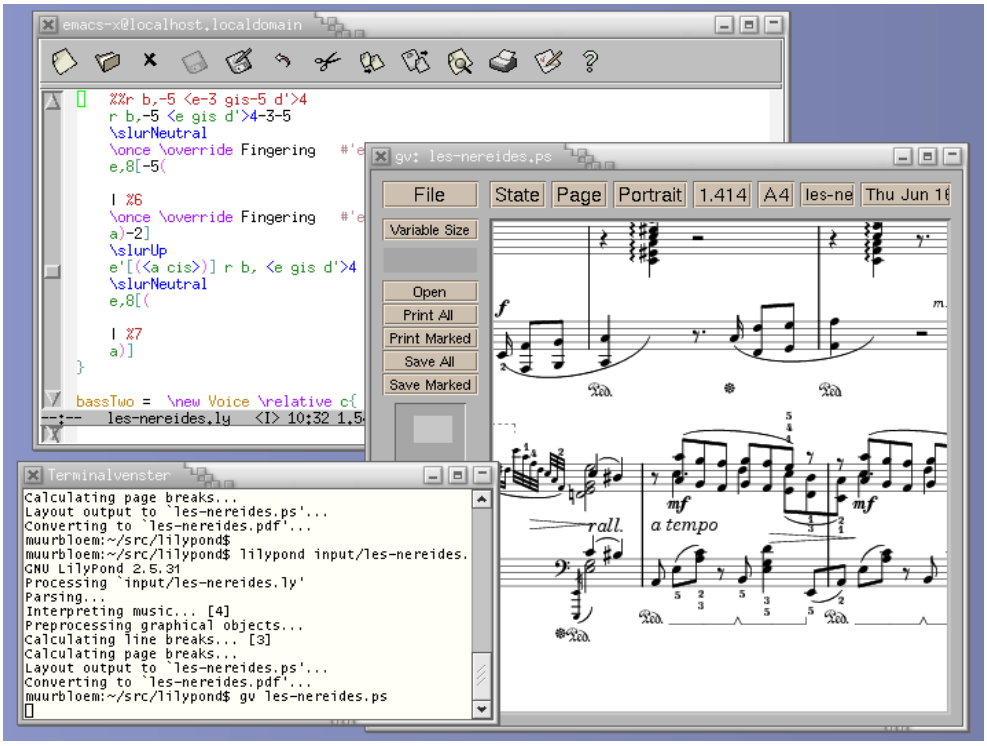
Je potřeba zmínit i některé zajímavé projekty, které nejsou placené. Velice populárním pro komunitu amatérských hudebníků se stalo **MuseScore** [5].

Jedná se o open source software na vytváření a tisk hudebních záznamů, který je ale zároveň mnohem více. Nabízí mimo jiné možnost sdílení těchto záznamů mezi uživateli a obecně rozsáhlou online komunitu, která za tímto projektem stojí. Jsou k dispozici také aplikace pro *Android*, *iPad* a *iPhone*.

Pozitivní je zde garance existence neplaceného účtu i do budoucnosti, pro zájemce ale existuje i placený účet s určitými výhodami, jako je neomezený online prostor či absence reklamy. Vše je dostupné ze stránky *musescore.org*.

**LilyPond** je dalším open-sourcovým projektem, zabývajícím se hudební notací. Je založen na textové editaci notových záznamů pomocí důmyslného značkovacího jazyka ve stylu jazyka LaTeX(viz *Obrázek 2.2*).

Nabízí plně profesionální notový výstup, nicméně absence grafického uživatelského rozhraní může být pro někoho, kdo nemá zkušenosti s koncepcí značkovacích jazyků, překážkou. Za dobu své existence se LilyPond stal v hudebních kruzích již téměř kultovní záležitostí [6].




Obrázek 2.2: Značkovací jazyk, terminál a zobrazený výstup programu **LilyPond**

## 2.2 Webové aplikace

Modernější přístup představují webové aplikace. Není potřeba instalace žádné desktopové aplikace a ke svým notám se dostanete prakticky odkudkoli. S přesunem do webových prohlížečů ale vyvstávají nové problémy, které je potřeba řešit.

Kompatibilita různých prohlížečů a jejich verzí a omezenější možnosti použitelných technologií jsou jen některé z věcí, které je potřeba vzít do úvahy. Dalšími mohou být například nižší rychlost některých technologií oproti desktopovým alternativám, nebo nutnost instalace dalších programů či plug-inů, jako je např. **Adobe Flash** nebo **Microsoft Silverlight**.

**Noteflight** [7] je webovým editorem notových záznamů, který pro zobrazování svého obsahu využívá v současné době právě softwarovou platformu Adobe Flash. To mu umožňuje interaktivní práci s notami pomocí myši, přehrávání zvuku v prohlížeči či zajímavé efekty a animace. Nevýhodou tohoto řešení je nutnost instalace platformy Flash jako plug-inu, který ale nepodporují všechny prohlížeče, hlavně na mobilních platformách.

Dalším příkladem je online notový editor **Scorio** [8], který využívá poměrně originální přístup: Jednotlivé noty a další prvky notace jsou skládány ze statických obrázků přímo do HTML struktury stránky. JavaScript se stará o uživatelský vstup a dynamicky mění DOM strukturu stránky. Pozitivní je zde přenositelnost editoru, který nepotřebuje žádné externí plug-iny a měl by fungovat v každém konvenčním webovém prohlížeči. Častá manipulace s DOM má však za následek nižší rychlost editoru, která je znát už při základních operacích typu přidání či přesun noty.

## 2.3 Aplikace používající podobné technologie

Kromě výše představených řešení je zde pro úplnost potřeba uvést ještě řešení, která používají k renderování notace VexFlow API, které je použito i v této práci. Spíše, než o aplikace, se jedná o technologická dema, naznačující možný potenciál tohoto API.

### 2.3.1 HTML5 Cloud Composer

V srpnu 2011 v rámci *Google HTML5 Hackathonu* vzniknul tento jednoduchý program, umožňující uživateli přidávat na osnovu různě dlouhé a vysoké noty a následně je označit myší a smazat. Program neměl žádné komerční aspirace, spíše chtěl ukázat možnost využití HTML5 canvasu k renderování a editování hudební notace. Hudba by podle autora měla jít následně v některých prohlížečích i přehrát pomocí web audio api [9].

### 2.3.2 „An in-browser sheet music and tablature editor“

O něco ambiciózněji vypadá editor od Cyrila Silvermana, který je sám jedním z přispěvatelů do kódu VexFlow API. Jeho čistě JavaScriptová aplikace umožňuje vytvořit a editovat většinu notových prvků na dvou fixních osnovách pomocí myši a grafického rozhraní. Program však neumožňuje jejich uložení, a po obnovení stránky vám veškerá vztvořená notace zmizí [10].

Kapitola 3

# Možnosti řešení

## 3.1 Aplikace jako celek

Vzhledem k potřebě přenositelnosti celé aplikace byla už od začátku plánování jasná a daná volba webové aplikace, která bude spouštěna ve webovém prohlížeči uživatele. Architektur takovýchto aplikací existují spousty. Uvedu zde uvažované možnosti:

1. **Klasická architektura webové aplikace:**

   MVC (model-view-controller), kde vzdálený server obsahuje databázi uložených souborů notových zápisů. Server zároveň obsluhuje požadavky na operace s těmito soubory, provádí veškerou business logiku a klientovi posílá pouze výslednou prezentační stránku. Každý uživatelský vstup se rovná novému požadavku na server.

2. **„Tlustý“, nebo také „Silný“ klient:**

   Většina výpočtů a operací probíhá přímo pomocí skriptů na přístroji klienta. Tímto se omezí objem komunikace mezi serverem a klientem pravděpodobně pouze na ukládání a stahování notových souborů či načítání jednotlivých stránek. Editace not či tvorba zvuku se vypočítávají na stroji uživatele. Je nutné dbát na vhodnou implementaci a použité technologie, aby nebyl klient těmito operacemi přetížen.

## 3.2 Front-end aplikace

Každá webová aplikace musí nějak dynamicky generovat výslednou webovou stránku, na základě uživatelského vstupu a také dat uložených v databázi. Jednou z možností je napsat celou aplikaci od nuly, například pomocí jazyka PHP, včetně základních operací s databází a jejím návrhem.

Další metodou je použití nějakého webového frameworku, což je v dnešní době často používaná možnost. Frameworků existuje celá řada, od nízkoúrovňových projektů až po velice mocné nástroje, v kterých se dá základní kostra aplikace zhotovit doslova za pár minut. Jazyka PHP využívají frameworky jako **Symfony** či **Nette** a mnoho dalších. Další frameworky používají populární skriptovací jazyk **Python**.

Většina robustnějších frameworků používá pro zápis stránek systému **šablon** (angl. templates). Tyto šablony obvykle umožňují skládání stránek z jednotlivých bloků a proměnných, dodaných *kontrolerem* této stránky (kontroler je část programu či funkce, zodpovědná za obsluhu dané stránky). Zároveň je v nich možné používat základních programovacích konstrukcí jako jsou podmínky a cykly. Hlavním cílem těchto šablon je zeštíhlení kódu a co největší jednoduchost zápisu stránek, tudíž i ušetření času pro vývojáře a kodéry.

## 3.3 Zobrazení a editace hudební notace

Co se této části týče, opět existuje celá řada možností. V této sekci budou popsány ty, mezi kterými bylo vybíráno.

### 3.3.1 Technologie

1. **HTML statické obrázky + JavaScript**

   Tuto možnost zmiňuji spíše na okraj a pro zajímavost, jelikož jsem se s ní setkal při rešerši existujících řešení. V tomto řešení (Scorio [8]) jsou noty vykreslovány pomocí jednotlivých obrázků v HTML elementu *img*.

   Pro některé objekty, jako jsou např. notové linky a nožičky not, toto řešení využívá možnosti škálování obrázků pomocí HTML a CSS. Program načte pouze jedno-pixelový černý obrázek a ten potom roztáhne do potřebné šířky či výšky.

   Pro statické vykreslování not by to byla validní, i když poměrně zastaralá možnost. Pro použití k interaktivní editaci pokládám toto řešení za ne příliš vhodné. Neustálá manipulace s DOM stránky pomocí JavaScriptu celý editor výrazně zpomalí a navíc toto řešení pouze složitě emuluje podobnou funkcionalitu, kterou nabízejí jiná řešení nativně.

2. **Adobe Flash**

   Spoustu užitečné funkcionality nabízí Adobe Flash. Toto řešení umožňuje tvorbu kompletního interaktivního prostředí v rámci internetových stránek, tedy i zobrazování not a jejich editaci či přehrávání zvuku. K jeho fungování je potřebná instalace plug-inu do webového prohlížeče.

   Tento plug-in není k dispozici na všechny platformy a prohlížeče. Má problémy především na mobilních zařízeních. Soukromá společnost *Adobe*,

která Flash vyvíjí jako uzavřený standard, navíc do budoucnosti nepočítá s další podporou jeho vývoje na mobilních platformách. Chce se věnovat modernějším HTML5 alternativám [11].

3. **SVG + JavaScript**

   Scalable Vector Graphics (zkráceně SVG) je formát vektorového souboru na bázi XML. Navíc podporuje animace a umožňuje interakci s vykreslenými objekty. Jelikož se jedná o grafiku definovanou vektory, může se tento obsah libovolně škálovat beze ztráty kvality.

   SVG může být samostatným souborem, nebo se může do HTML stránky zapsat v rámci elementu object. V HTML5 je dokonce možné přímé vložení *svg* elementu, reprezentujícího plátno SVG objektů, do HTML dokumentu. Podpora SVG v prohlížečích se rok od roku lepší, nicméně v některých starších prohlížečích, či na mobilních zařízeních, není dostupná celá funkcionalita tohoto rozsáhlého standardu.

   Jednotlivé prvky SVG grafiky jsou zároveň DOM elementy. Výhodou je možnost editace těchto prvků či přímé navěšení listenerů uživatelského vstupu. Při editaci, především při větším rozsahu notace, se DOM opakovaně upravuje a mění se i náročnost výpočtu výsledného obrazu. To může vést ke zpomalení, obdobně jako v případě *č. 1.*

   Další nevýhoda spočívá v tom, že SVG standard neumožňuje přímo, tedy bez dalších knihoven, exportovat výsledný obraz do obrázkového souboru. To je vlastnost, která by neměla v implementaci notového editoru chybět.

4. **HTML5 canvas + JavaScript**

   Tato možnost je nejmladší ze všech uvedených. HTML5 standard se sice stále ještě formuje, ale je masivně podporován ve všech konvenčních webových prohlížečích, včetně mobilních platforem. Na jeho vývoji se podílejí společnosti jako **Google**, **Microsoft** či **Apple** [12], takže se dá předpokládat velká podpora i v budoucnosti.

   Konkrétně se, pro potřeby této aplikace, jedná o využití HTML5 elementu *canvas*[1] a jeho JavaScriptového API (označováno jako **Canvas API**), které umožňuje vykreslování bitmapové grafiky do pomyslného „plátna“ v HTML stránce.

   Na rozdíl od SVG je vše v rámci jednoho HTML elementu, jehož výpočetní náročnost se mění pouze v případě změny jeho velikosti. S HTML se prakticky nemanipuluje (vyjma právě změny velikosti elementu *canvas*), samotné plátno nemá žádný přehled o grafu scény. Ten si musí program vytvořit a udržovat sám.

[1] Pozn.: **Canvas** = plátno (angl.)

Grafika je pouze rastrová a má statické rozměry v pixelech, což může být překážka responsivnosti na různých zobrazovacích zařízeních. Velikost plátna se však dá měnit, jako jakýkoliv jiný HTML element, pomocí *JavaScriptu*. Po změně je pouze potřeba z udržovaného modelu obsah plátna vykreslit znovu.

Canvas API nenabízí nativní podporu pro animace, ani pro uživatelskou interakci s objekty. Na druhou stranu *canvas* umožňuje přímo exportovat plátno jako bitmapový obrázek ve formě Base64 řetězce.

### 3.3.2 Knihovny

1. **Alpha Tab** [13]

   Alpha Tab je open-sourcová renderovací knihovna. Existuje ve dvou verzích, *JavaScript* a *.NET*, a umožňuje renderovat hudební notaci přímo v prohlížeči. Je poměrně pestrá, co se výběru vykreslovacích pláten týče. Na výběr mají vývojáři hned ze čtyř možností:

   - **HTML5 canvas (pouze JS verze)** - renderuje rastrově
   - **SVG Canvas (obě verze)** - renderuje vektorově
   - **GDI Canvas (pouze .net verze)** - renderuje rastrově
   - **WPF Canvas (pouze .net verze)** - renderuje vektorově

   Problémem této knihovny je poměrně malá komunita uživatelů a přispěvatelů (prakticky jediný autor), což se odráží i na nepříliš rozsáhlé dokumentaci a pomalém vývoji a odstraňování chyb. Tato knihovna neobsahuje žádné API přímo určené k interaktivní editaci hudební notace, spíše se zaměřuje na co nejkompletnější vykreslování a implementaci zvukových funkcí.

2. **HTML5 MusicXML Viewer** [14]

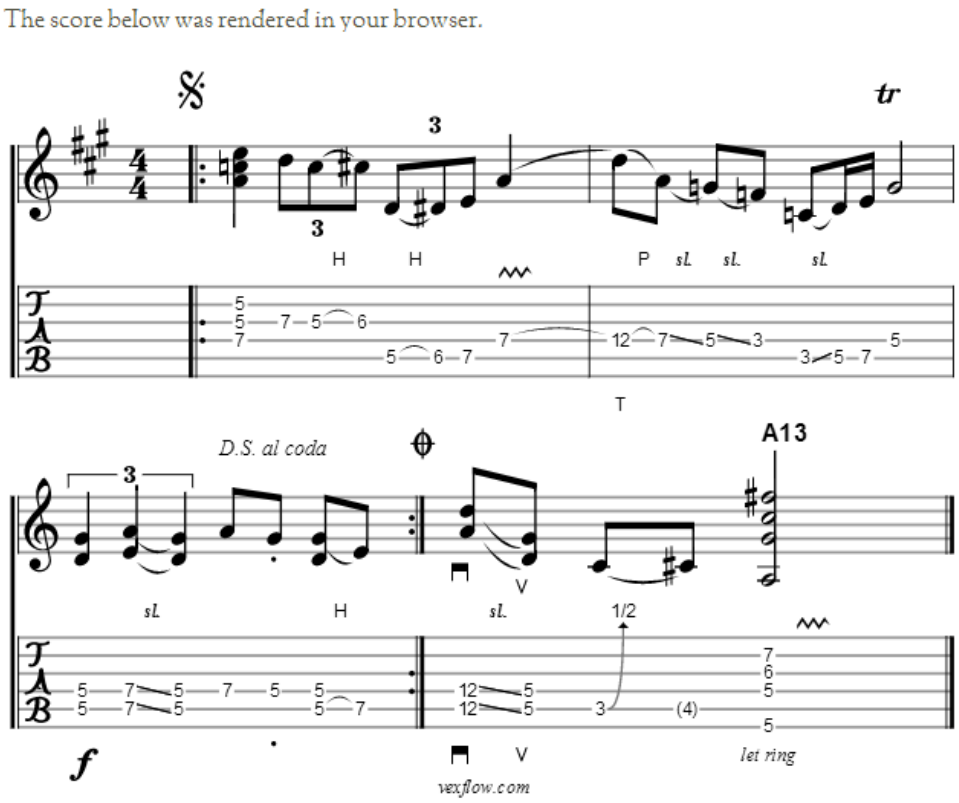
   Za tímto názvem se ukrývají dvě JavaScriptové knihovny: **score-library** a **score-div**. První jmenovaná je zodpovědná za načtení *MusicXML*[2] souboru a ta druhá za vykreslení načteného modelu do HTML5 plátna.

   Program je sice oficiálně distribuován pod *GNU GPL* licencí, nicméně se mi nepodařilo dohledat jinou, než minifikovanou verzi zdrojového kódu. I stručná dokumentace, která je k dispozici na stránkách projektu, se věnuje pouze použití knihovny pro koncového uživatele k vykreslení MusicXML souboru na webových stránkách. Knihovna proto není příliš vhodná k použití pro vývoj notačního editoru.

[2] Pozn.: **MusicXML** je nejrozšířenější typ souboru hudební notace

3. **VexFlow** [2]

   Vexflow je nízkoúrovňové API, jehož implementace je open-source a je pro nekomerční využití volně k dispozici pod *MIT* licencí [15]. Je napsané kompletně v *JavaScriptu* a notaci vykresluje přímo v prohlížeči do HTML5 canvasu. Ve spojení s další JS knihovnou – **Raphaël** (*raphaeljs.com*) – dokáže renderovat i vektorově do SVG plátna.




Obrázek 3.1: Ukázka notace, vykreslené pomocí VexFlow [2]

Kapitola 4

# Zvolené technologie

Tato část práce představuje technologie, které jsou použity při tvorbě jádra aplikace a také modulu, jehož se tato práce týká. Kromě stručného popisu je zde také vysvětleno, z jakých důvodů byly zvoleny.

## 4.1 Jádro aplikace

Stručný popis zvolených technologií back-endu aplikace. Ačkoliv se tato práce nezabývá primárně návrhem ani tvorbou jádra, byl jsem součástí rozhodovacího procesu, který tyto technologie vybíral a v rámci rozdělení práce jsem do jádra také přispíval.

1. **Programovací jazyk**

   Jako hlavní programovací jazyk jádra aplikace byl vybrán **Python** (konkrétně verze *3.4*). Jedná se o moderní, vysoko-úrovňový programovací jazyk, jehož hlavní filosofií je čitelnost kódu. Umožňuje rychlou tvorbu aplikací, v kratším čase a s menším objemem kódu, než většina programovacích jazyků [16].

   Python je k dispozici pro většinu systémů (u většiny Linuxových distribucí je rovnou součástí instalace) a existuje pro něj nespočet rozličných standardních i komunitních knihoven. Je používán ve významných webových projektech, které používají miliony lidí každý den:

   **YouTube:** *„Python je pro naše stránky dostatečně rychlý a umožňuje nám produkovat udržovatelné funkce v rekordním čase, s minimem vývojářů.“* [3]

[3] **Cuong Do**, Software Architect, *YouTube.com* (přeloženo z angličtiny, citováno z [16])

> **Google:** *„Python byl důležitou součástí Googlu již od jeho počátku, a nadále jí zůstává, jak systém roste a vyvíjí se. Dnes používají Python tucty Google inženýrů a hledáme další, kteří tento jazyk ovládají.“* [4]

Pro všechny výše zmíněné výhody byl jazyk Python vybrán k použití v tomto projektu. Mimo jiné například i pro svou snadnou čitelnost, která je při projektu, na kterém pracuje více lidí, výhodou.

2. **Framework**

   Při tvorbě webových aplikací v Pythonu je zapotřebí nějakého frameworku, který obslouží HTTP požadavky a poskytne rozumný způsob generování HTML stránek. Psát webovou aplikaci v Pythonu bez frameworku se v podstatě rovná napsání svého vlastního frameworku, což není účelem tohoto projektu.

   K tomuto účelu byl vybrán framework **Django** (verze *1.8*). Jedná se o velice robustní a komplexní vysoko-úrovňový nástroj, určený především pro rychlý vývoj aplikací. Django splňuje všechny požadavky na moderní webový framework, jako je použití „pěkných URL“, vestavěné administrační rozhraní nebo podpora více-jazyčných aplikací [17].

   Velkou výhodou Djanga je také rozsáhlá dokumentace a celá škála oficiálních i neoficiálních tutoriálů, které činí práci s ním velice pohodlnou. To byl také jeden z důvodů volby tohoto frameworku pro vývoj této aplikace.

   Jedna instalace Djanga (nazývaná „projekt“) může obsahovat více částí či modulů, označovaných v Django terminologii jako „aplikace“. Toho zde bylo využito pro různé části aplikace, které tak mohou být přidávány a odebírány z projektu, aniž by se narušovala funkčnost a celistvost jádra.

## 4.2 Front-end aplikace

„Templatovací“ (nebo také „šablonovací“[5]) systém Django frameworku je založen na svém vlastním značkovacím jazyce, nikoliv na Python syntaxi, jak by někoho mohlo napadnout. Jeho hlavním účelem je zjednodušení tvorby souborů textových formátů, jako např. (X)HTML či XML. Povoluje některé základní programovací konstrukce, jako jsou podmínky či cykly, které ale slouží pouze k prezentační, nikoliv programové logice. Ta musí být oddělená v samostatných kontrolerech (v Djangu zvané *Views*).

Jednotlivým kontrolerům je přiřazen URL klíč, po jehož požadavku klientem na server se daný kontroler spustí a případně i zavolá potřebná šablona.

[4] **Peter Norvig**, Director of search quality, *Google, Inc.* (přeloženo z angličtiny, citováno z [16])

[5] *template* = *šablona* angl.

V té se použijí proměnné dodané kontrolerem (včetně pythonovských slovníků a dalších datových typů). Kontroler zároveň naslouchá uživatelskému vstupu (např. HTML formuláře) a obsluhuje jeho obsah (včetně kontroly, zda-li se nejedná např. o *cross-site request forgery*).

Vzhledem k vybraným technologiím vykreslování notových záznamů (více v dalších částech této práce) je šablonovací systém použit k tvorbě stránek ve standardu HTML5. HTML5 je zatím posledním standardem HTML, a byl navrhnut pro tvorbu bohatého obsahu webových stránek bez nutnosti přídavných knihoven či plug-inů [18].

HTML5 si dokáže poradit se zvukem, videem nebo složitější grafikou, a navíc přináší nové HTML elementy, které daleko více reflektují skutečnou strukturu dnešních stránek (elementy jako *header*, *footer*, *article* nebo *nav* a mnoho dalších). Staré prvky, které se v praxi nepoužívají, naopak ze standardu vyřazuje, čímž ho zpřehledňuje [19].

HTML5 samo o sobě samozřejmě všechny požadované věci neobslouží. K manipulaci s DOM stránek a k obsluze dalších prvků (např. vykreslování) bude použito skriptovacího jazyka **JavaScript**.

JavaScript obstarává spouštění skriptů na stroji klienta, čímž umožňuje dynamickou změnu stránek bez nutnosti dalších požadavků na server. To je velice výhodné při tvorbě uživatelského rozhraní notačního editoru. Ve spojení s HTML ovládacími prvky, jako jsou tlačítka či rozbalovací seznamy, je JavaScript schopen zpracovávat uživatelské vstupy.

Editor tedy bude fungovat na klientské straně a serveru bude využito pouze k načítání a ukládání vytvořeného notového zápisu. Tyto požadavky může JavaScript na server posílat a zpracovávat asynchronně, bez nutnosti opětovného načítání celé stránky.

Užívání JavaScriptu, jazyka, který byl navrhnut v roce 1995 během 10 dní [20], se za poslední roky hodně posunulo. Právě díky technologiím jako HTML5 ho již dnes stále více programátorů používá pro tvorbu komplexních, objektově orientovaných programů. Některá objektová funkcionalita však byla do JavaScriptu dodána postupně, a má značná omezení.

JavaScript nemá třídy, ale pouze objekty a prototypy, které dokáží většinu vlastností tříd z jiných jazyků nahradit, ale pro programátora to není příliš intuitivní ani pohodlné. Kopírování objektů navíc probíhá pouze referencí, takže pro klonování objektů, které má mít možnost uživatel poté separátně upravovat, je potřeba napsat vlastní kopírovací funkce, které ručně přiřadí veškerý obsah objektu.

Pro interakci čistě JavaScriptového editoru s Django aplikací, například pro ukládání vytvořené notace do databáze, je také potřeba nějaký univerzální formát, který bude posílán mezi serverem a klientem. V úvahu připadalo buďto **XML**, nebo **JSON**. Vzhledem k faktu, že jak Python, tak JavaScript umožňují manipulaci s formátem JSON (převedení vlastních objektů do JSON řetězce a naopak) přímo bez dalších knihoven, byl zvolen tento formát.

K transportu těchto dat bude využito asynchronního volání serveru (jak pro načtení, tak pro uložení notace) pomocí **AJAX**[6] technologií. K tomuto účelu bude využito knihovny **JQuery**, která tento proces podstatně usnadňuje. Knihovna se zároveň využije i v rámci dalších front-end skriptů, například pro správu vyskakovacích dialogových boxů aplikace.

## 4.3 Vykreslování hudební notace

### 4.3.1 Vykreslovací technologie

Hlavním předmětem této práce je vykreslování standardní (také někdy označované jako západní) hudební notace ve webovém prohlížeči. Ze všech výše popsaných uvažovaných metod jsem zvolil rastrové vykreslování pomocí **Canvas API**.

Tato možnost má dle mého osobního názoru vhodnou kombinaci přenositelnosti a výkonu. Navíc je pro mě výzvou vyzkoušet tuto stále poměrně novou technologii v oblasti, kde ještě není tolik prozkoumána. Většina současných řešení stále využívá starší technologie, jakými je třeba *Adobe Flash*.

Canvas API je velice vhodným nástrojem pro jednorázovou tvorbu 2D bitmapového obrazu (umožňuje i 3D kontext, ale to se netýká této práce), takže je velice vhodná k jednorázovému vykreslení statického notového záznamu. Z prvků hudební notace dokáže velice rychle sestavit výsledný obraz do plátna, reprezentovaného elementem *canvas*).

Pro použití při tvorbě interaktivního editoru, kde se hudební notace dynamicky mění, je zapotřebí vytvoření vlastních metod pro opětovné překreslování obsahu plátna a uživatelskou interakci s jednotlivými prvky. Rychlost zde bude patrně klesat kvůli opakovanému překreslování *canvasu*, ale měla by být stále rozumná, jelikož se nemění DOM stránky, pouze obsah plátna.

### 4.3.2 Použité API

Po zvážení množství různých prvků, které standardní hudební notace obsahuje, a s přihlédnutím k faktu, že obsahem této závěrečné práce není pouze statické vykreslování notových zápisů, rozhodl jsem se použít externí renderovací API **VexFlow**.

VexFlow dokáže vykreslit prakticky cokoliv z hudební notace (další funkce neustále přibývají), ale je potřeba mu to explicitně přikázat. Mezi největší úkoly implementace tak patří korektní logika hudební notace a další formátovací aspekty notových záznamů, které vedou k dobré interpretaci zápisu do obrazové formy.

Nejnáročnější úlohou pak je vytvoření pro uživatele jednoduchého a intuitivního editoru, který umožní tvorbu a editaci hudební notace za použití

[6] *Asynchronous JavaScript and XML*. Při použití JSONu namísto XML občas označováno také jako *AJAJ*, tento termín se však nikdy masově neuchytil

GUI a uživatelského vstupu myší. Zde se bude notace také vykreslovat pomocí VexFlow, ale bude zapotřebí udržovat editovatelný objektový model vytvářené notace, který bude udržovat přehled o upravovaných částech záznamu.

Kapitola 5

# Návrh a realizace

## 5.1 Aplikace a pracovní postup

Do vytvořeného Django projektu potřebovalo mít najednou přístup více lidí. Byl tedy vytvořen repozitář verzovacího nástroje **Subversion (SVN)**, pomocí kterého se dají zaznamenávat a verzovat lokální změny v projektu.

Mezitím byla vytvořena základní struktura aplikace. Jádro systému sestává z databázového modelu, obsahujícího informace o uživatelích a uživatelských skupinách a také jejich uložených albech a skladbách. Součástí každé skladby, kromě jejích metadat (tedy názvu, popisu, a dalších údajích) je i editovatelný notový záznam. Editace probíhá pomocí editoru, popsaného v dalších sekcích této kapitoly.

Toto jádro tvoří základní část projektu (tedy „aplikaci“ v Django terminologii), nazvanou pro její nejvýraznější součást *Editor*. Samostatné moduly, jako například OMR či algoritmy zabývající se porovnáváním skladeb, tvoří vlastní aplikace v tomto projektu. Naopak, práce zabývající se hudebními souborovými formáty či syntézou zvuku by měly být součástí jádra.

Ačkoliv to nebylo přímou součástí zadání této práce, podílel jsem se i na tvorbě jádra aplikace, převážně na URL dispečeru a kontrolerech pro některé akce, které přímo souvisely s touto prací. Bylo tak rozhodnuto proto, že tvorba jednotlivých šablon v Djangu je provázána s tvorbou kontrolerů a jejich návrh je podřízen jejich obsahu. Naopak, ze stejného důvodu byly některé šablony napsány mými kolegy. Toto mělo vést k větší efektivitě práce a časové úspoře.

## 5.2 Front-end aplikace

Po dohodě s ostatními členy vývojového týmu bylo rozhodnuto, že navrhnu základní šablonu aplikace a její HTML strukturu. Tato základní, „base“, šablona bude rozšiřována obsahem jednotlivých šablon. V tomto kroku byla navržena také grafická podoba aplikace a příslušné *CSS* styly a vytvořeno logo celého

projektu, který dostal název **FIT Music**. Také bylo zapotřebí napsat některé JavaScriptové skripty.

### 5.2.1 Vykreslování hudební notace - *notation.js*

Jak je již zmíněno výše, VexFlow je velice povedené, ale poměrně hodně nízkoúrovňové API. Zabývá se pouze vykreslováním hudebních elementů. Korektnost a logiku notace si musí ohlídat sama aplikace. Toto je „by design“ (zamýšlené), v rámci principu oddělení business a prezentační logiky. Výsledný render působí čistě a velice profesionálně (viz *obrázek 3.1*), dle mého názoru možná i lépe, než některé placené programy.

Je tedy zapotřebí sestavit objektový model notace a notační logiku, která zaručí správné vykreslování notace. Je proto vytvořeno jakési primitivní, objektově orientované, API (*notation.js*), jež je určeno k převedení objektového modelu editoru (*editor.js*) do VexFlow objektů. Může být však využito např. i ke statickému vykreslení již uloženého zápisu.

K tvorbě obou programů bude použito čistého jazyka JavaScript, bez jakýchkoliv externích knihoven. V JavaScriptu neexistují třídy, proto k definici objektového modelu využiji zapouzdřených funkcí („konstruktorů“), obsahujících JavaScriptové prototypy. Tímto způsobem je napsané i samotné VexFlow.

VexFlow používá anglické názvosloví hudební notace, takže se jí v rámci jisté koherence kódu budu držet. Slovník použitých pojmů najdete přiložen.

Základním prvkem VexFlow je *Stave* (notová osnova). Do té se dají kreslit noty, pomlky, klíče, předznamenání a spousta dalších prvků notace. VexFlow je vykreslí tak, jak mu zadáte, i když by to bylo v rozporu s pravidly hudební notace (např. směr notových nožek, rozložení trámců atp.). Pokud už VexFlow něco samo hlídá (např. maximální délka taktu), řeší to pouze vyhozením výjimky, takže editor stejně potřebuje ošetřovat takové věci sám.

Notové záznamy jsou tvořeny **takty**. Ve VexFlow nic jako takt neexistuje, jelikož to už patří do notační logiky. Existují dvě možnosti tvorby taktů za použití VexFlow:

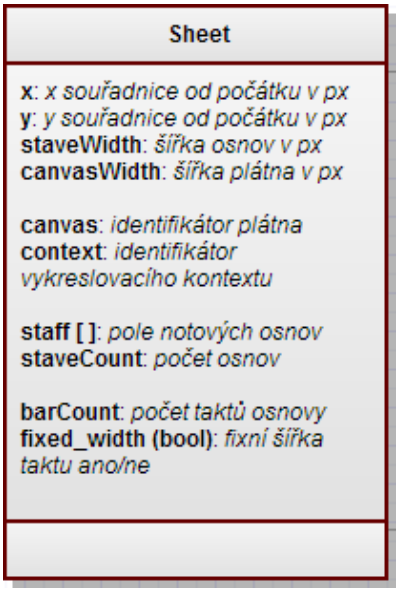
1. Do jedné instance *Stave* se vykreslí instance *Barnote* („taktové noty“), což jsou vlastně plovoucí taktové čáry, které je možné umístit mezi noty. Toto řešení je vhodné pouze pro krátké, jedno-osnovové záznamy, jelikož *Barnotes* je obtížné přesně pozicovat pod sebe.

2. Každá *Stave* reprezentuje jeden samostatný takt. Nastavuje se jí typ počáteční a koncové taktové čáry (*Barline*). Je potřeba takty za sebe korektně navázat (ukládat si jejich pozici a šířku). Výhodou je přesná kontrola šířky taktu a možnost samostatného překreslování jednotlivých taktů.

Z informací uvedených výše jasně plyne nutnost využití druhé možnosti, kvůli potřebě tvorby editoru. Budou tedy vytvořeny následující entity:

- **Sheet**

  Tato entita (viz *obrázek 5.1*) reprezentuje **celý notový záznam**. Udržuje v sobě všechny notové osnovy (jednotlivé nástroje či ruce) a celkovou šířku záznamu. Je odpovědná za správné navazování nově přidávaných taktů.




Obrázek 5.1: Entita *Sheet*

- **Stave**

  Tato entita (viz *obrázek 5.2*) reprezentuje jednu **notovou osnovu** (noty pro jeden nástroj či jednu ruku (u piana)). Obsahuje všechny takty v dané konkrétní osnově a metody pro operace s nimi.

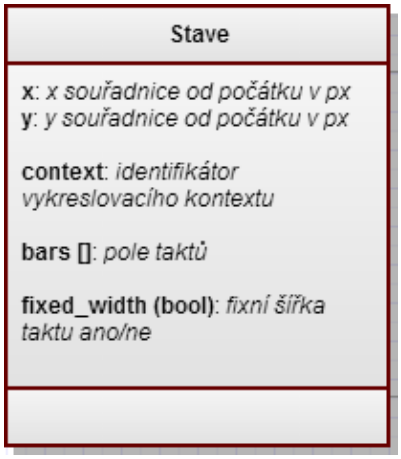
- **Bar**[7]

  Tato entita (viz *obrázek 5.3*) představuje jeden **takt** (myšleno pouze na jedné notové osnově). Obsahuje všechny hlasy na této osnově v tomto taktu.

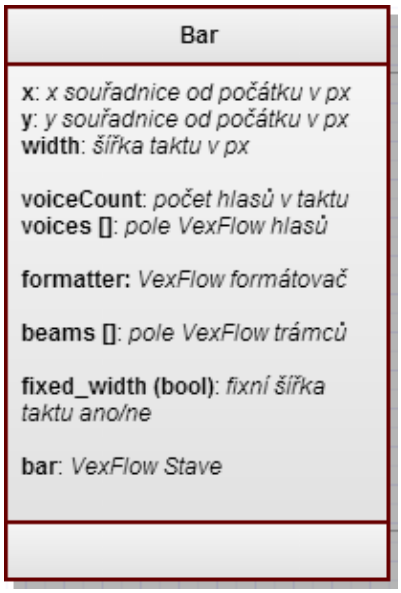
Pro všechny podřízené prvky (noty, klíče, atd.) má již VexFlow své vlastní třídy. Jejich správné použití a vykreslení kontrolují výše uvedené entity.

Celý návrh je podřízen faktu, že s použitím tohoto „API“ bude realizován notový editor. Kromě metod, obstarávajících statické, jednorázové vykreslování, proto obsahuje i metody pro přepisování jednotlivých entit, jako jsou například takty či noty.

[7]Takt se v anglické terminologii nazývá *bar*, nebo také *measure*.

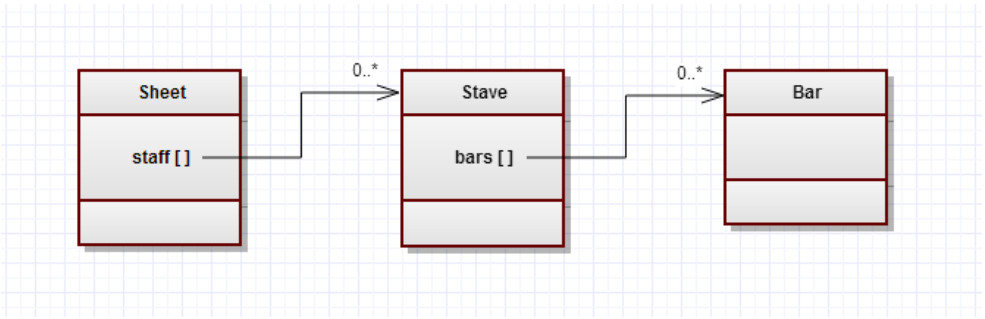



Obrázek 5.2: Entita *Stave*




Obrázek 5.3: Entita *Bar*

Při implementaci notového editoru se objevily nedostatky návrhu a implementace tohoto API, které vedly k jeho téměř kompletnímu přepsání do současné podoby. Stále se však jedná jen o prototyp, počet podporovaných prvků je zatím omezen jen na ty, potřebné ke tvorbě editoru. Mezi výjimky patří například připravená podpora pro více notových osnov než jen dvě, a pro více hlasů v rámci jedné osnovy (editor používá jen jeden).




Obrázek 5.4: Vztah mezi notačními entitami

## 5.3 Editor hudební notace - *editor.js*

### 5.3.1 Účel editoru

Jedná se o malý interaktivní editor, určený primárně příležitostným uživatelům pro tvorbu základních notových záznamů. Editor by měl mít jednoduché, intuitivní ovládání s minimalistickým uživatelským prostředím. Dané ovládání si musí být schopen osvojit do několika minut, buďto intuitivně, nebo s krátkou nápovědou.

K jeho ovládání by měla postačit počítačová myš[8]. Základní operace, jako přidávání a odebírání not, lze obstarat přímo klikáním na notovou osnovu. Pro složitější úpravy slouží ovládací prvky grafického uživatelského panelu.

Editor bude představovat klavírní notový zápis. Fixně v něm budou dvě notové osnovy, které představují noty pro levou a pravou ruku.

### 5.3.2 Požadované funkce

Seznam minimálních funkčních požadavků notového editoru:

- přidání/odebrání taktu z notové osnovy.
- výběr/označení taktu k editaci pomocí myši
- změna klíče, stupnice(předznamenání) a rytmu(metrum) označeného taktu
- vyznačení začátku či konce repetice v označeném taktu
- volba délky přidávané noty/pomlky
- přidání/odebrání noty/pomlky z notové osnovy pomocí myši
- označení noty/pomlky pomocí myši

[8] *myší* je zde (i dále v této sekci) myšlena počítačová myš, touchpad či jiné zařízení podobného účelu, včetně dotekových displejů

- přidání či odebrání posuvky (křížky a béčka) označené notě
- kopírování/vymazání celého obsahu taktu
- možnost exportovat vytvořený záznam jako obrázek k uložení

### 5.3.3 Návrh

Hlavní myšlenkou tohoto návrhu je práce ve dvou objektových „vrstvách“. VexFlow objekty (a objekty notačního API nad ním) se vytvářejí až při samotném vykreslování. Je tak potřeba si vytvořit paralelní objekty, které budou udržovat graf scény, přímo v editoru. Tyto objekty budou editovatelné, a při jejich změně se vytvoří nové VexFlow objekty, které překreslí ty původní.

Budou proto vytvořeny následující entity:

- **Editor**

  V notovém editoru vznikne pouze jedna instance třídy *Editor*. Ta inicializuje editor a nastaví potřebné **event-listenery** pro kontrolu uživatelského vstupu. Zároveň obsahuje stavové atributy (např. který takt či nota jsou označeny) a **event-handlery**(metody spouštěné event-listenery) pro obsluhu vstupu uživatele.

- **Editor.Bar**

  Tato entita představuje jeden takt editoru pro obě notové osnovy (nad sebou). Obsahuje informace společné pro oba takty (rytmus, předznamenání, začátek a ukončení taktu).

- **Editor.BarStave**

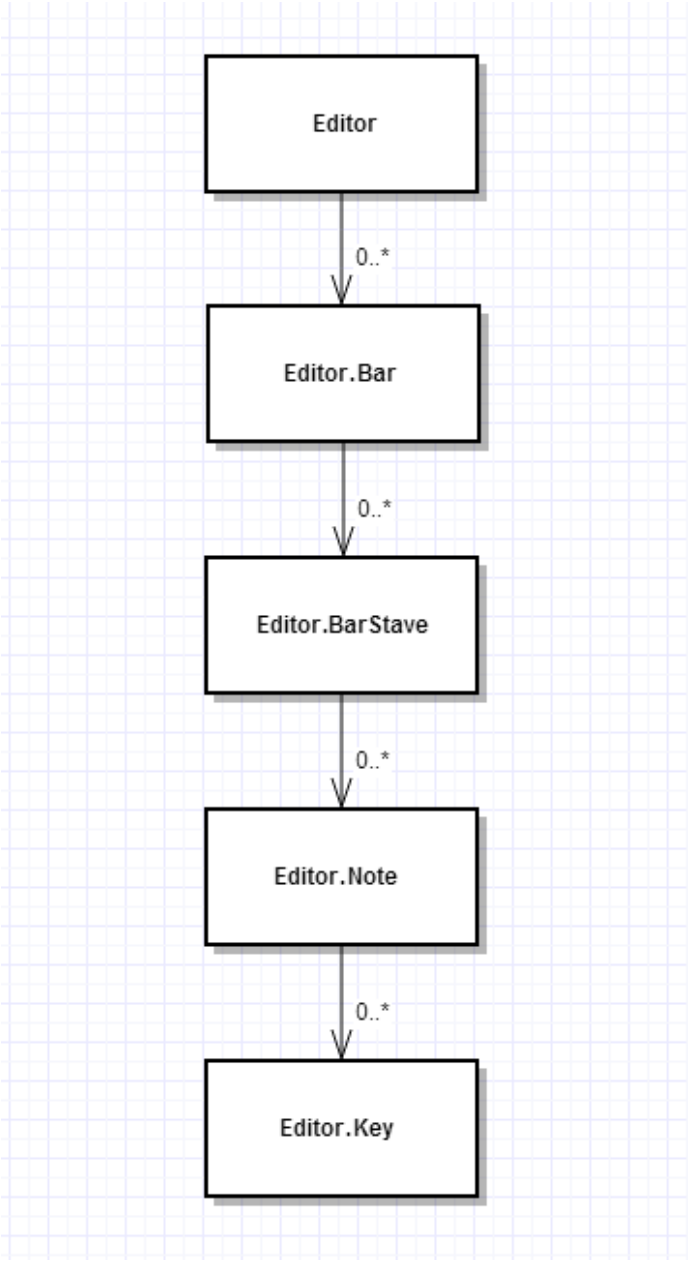
  *Editor.BarStave* představuje jeden konkrétní takt na jedné notové osnově. Obsahuje sety not a informace specifické jen pro něj (např. klíč).

- **Editor.Note** *Editor.Note* představuje celý souzvuk not (akord). Obsahuje informace o své délce a také jednotlivé noty.

- **Editor.Key** Jedna „nota“ v daném souzvuku. Obsahuje informace o výšce a oktávě noty. Název „key“ je zde použit po vzoru VexFlow a má naznačovat jednu zmáčknutou klávesu (angl. *key*) při hře na klavír.

Tento návrh je potřeba trochu vysvětlit, jelikož může být matoucí. Jak ukazuje *obrázek 5.5*, základem editoru je instance třídy *Editor*. Jelikož se délka skladby mění v obou osnovách stejně (pro levou i pravou ruku), i v tomto editoru se přidávají a odebírají oba takty jako celek. Tento celek představuje právě entita *Editor.Bar*.

*Editor.BarStave* je jeden konkrétní takt (na jedné osnově), a je paralelou pro entitu *Bar* z notačního API. Vztah je tady tedy opačný, *Editor.Bar* obsahuje vícero *Editor.BarStave*, kdežto v *notation.js* je vztah *Stave -> Bar*.




Obrázek 5.5: Vztah mezi entitami editoru

### 5.3.4 Omezení návrhu

Tento návrh obsahuje některá omezení, která jsou dána jak zvolenými technologiemi, tak i samotným zaměřením této aplikace:

- Fyzická délka taktu je v editoru nastavena fixně. Toto omezení je zde

záměrně, aby bylo možné vždy editovat a překreslovat jen jeden takt, což vede ke zrychlení aplikace.

- Počet hlasů je omezen na jeden. Vzhledem k vybranému ovládání pomocí myši a požadované jednoduchosti editoru by bylo obtížné rozlišit, která nota patří do kterého hlasu. Navíc se nepředpokládá, že by někdo využil tento nástroj k tvorbě složitých, vícehlasých partitur.

- Počet osnov je omezen na dvě. To je nejčastější forma notového záznamu. Tvorba složitějších partitur způsobem použitým v tomto editoru je tímto omezením vyloučena.

### 5.3.5 Realizace

Prototyp notového editoru je implementován za použití výše popsaného notačního API. Hlavním principem, na kterém tato implementace stojí, je překreslování vždy jen té části záznamu, ve které se něco změní. Tou jednotkou je konkrétně jeden takt. Toto řešení by mělo zaručit konstantní náročnost editoru i při delších skladbách, jelikož se znovu renderuje jen jeden takt, nehledě na délku skladby.

Aktuální podoba skladby se udržuje v objektovém modelu editoru. Uživatel myší zvolí jeden konkrétní takt, který je poté k dispozici pro editaci. Při změně v označeném taktu se aktuálně vykreslený takt „vymaže“ pomocí Canvas API metody **clearRect**, sestaví se nová logika taktu a ten se celý znovu kompletně překreslí.

Jak je již uvedeno dříve, Canvas API nenabízí nativní metody pro označování/výběr vykreslených prvků, jelikož si neudržuje žádný graf scény. Nemá žádný záznam o tom, co do něj bylo v minulosti vykresleno, pouze bitový přehled o barvách jednotlivých pixelů. Je proto potřeba si v objektovém modelu udržovat souřadnice jednotlivých elementů na plátně. VexFlow objekty jsou k tomuto přizpůsobené. Tyto souřadnice jsou ale relativní k počátku[9] elementu *canvas*.

Pro udržování souzvuků not v rámci jednoho taktu, stejně jako jednotlivých not v rámci souzvuku, byl vybrán spojový seznam, jako vhodná datová struktura pro vymazávání a přidávání seřazených prvků. Aktuální spojový seznam not souzvuku se před vykreslením celý projde a sestaví se z něj notový řetězec pro VexFlow notaci.

Noty jsou do spojového seznamu vkládány neseřazené, nehledě na svou výšku, pouze podle pořadí jejich vytvoření. Proto byla vytvořena metoda, která navrátí notu (entita key) podle její výšky v souzvuku, ať už se nachází ve spojovém seznamu kdekoliv. Metoda k tomu používá tento optimalizovaný algoritmus:

[9] dle specifikace levý horní roh

```
getKeyOnPosition: function (k) {
    var p1, p2;
    var i=0;
    p1 = p2 = this.firstKey;
    while (p1!=null) {
        while (p2!=null) {
            if ((p1.octave > p2.octave) || (
                (p1.octave == p2.octave) &&
                (p1.keyValue > p2.keyValue)
                )) {
                i++;
            }
            p2 = p2.nextKey;
        }
        if (i==k) {
            return p1;
        }
        i=0;
        p1 = p1.nextKey;
        p2 = this.firstKey;
    }
    return null;
}
```

Uživatelský panel je tvořen HTML ovládacími prvky jako jsou elementy *button*, *select* nebo *input*. Jejich změny „poslouchají“ editorové event-listenery, které potom mění objektový model na základě uživatelského vstupu.

Po každé změně, včetně pohybu myši po označeném taktu, se celý takt znovu překresluje. Základní kreslící metodou je proto **drawBar()**, která má na starosti korektní sestavení VexFlow prvků, jejich zformátování a nakonec i vykreslení.

Při rozšiřování notového záznamu přidáváním nových taktů je potřeba nějak vyřešit scrollování plátnem. Toho je v tomto prototypu dosaženo zvětšováním či zmenšováním samotného elementu *canvas* pomocí změny jeho atributu **width** v DOM. Blokový element, který plátno obaluje, má nastavenu CSS vlastnost *overflow* na hodnotu „auto“ a má fixní šířku. Tím je zajištěno scrollování při zvětšení plátna.

Při změně šířky *canvasu* dojde k vymazání celého obsahu plátna a je potřeba vykreslit celý obsah znovu. Je to ale stále nejvhodnější varianta, protože Canvas API žádné vnitřní metody pro scrollování nemá a jejich implementace by znamenala nutnost překreslování obsahu ne pouze při zvětšení, ale při každém posunu.

Kapitola 6

# Testování

Výstupem operací, prováděných uživatelem v editoru notových zápisů pomocí grafického uživatelského rozhraní a myši, je objektový model reprezentující jednotlivé prvky hudební notace pro danou skladbu. Tento model je možné ukládat a načítat z databáze ve formě JSON řetězce.

Korektnost převedení daného modelu na standardní hudební notaci však nezávisí pouze na jeho obsahu. Spoustu prvků notace ovlivňují také okolní objekty, ať už to jsou další noty, vedlejší takty, či různá znaménka v taktu a předtaktí. O malou část této logiky se stará samotné VexFlow API, většinu však musí vyřešit samotný editor.

Správnost aktuálně vykreslené notace se proto nedá ověřit pouhým zaznamenáním a ověřením aktuálního stavu objektového modelu skladby. Pro potřeby testování je tudíž nutné hodnotit až výslednou vykreslenou hudební notaci. Její korektnost musí určit člověk, který se vyzná v hudební teorii. Pro potřeby tohoto testování postačí znalosti autora této práce, jelikož se práce zabývá primárně programováním, nikoliv hudební teorií. Není proto zapotřebí rigorózní hudební terminologii ani znalosti.

Některé prvky ani objektivně hodnotit nelze, jelikož je zvyklosti hudební notace přesně nespecifikují, či pro ně existuje vícero interpretací[10]. Takové prvky jsou vždy na uvážení autora notace či notačního programu, nicméně by měly být konzistentní v celém záznamu (potažmo ve všech záznamech programu).

Jako vhodná metoda testování byla zvolena metoda simulace scénářů užití. Scénáře byly voleny tak, aby pokryly veškeré možné uživatelské operace v editoru, a co největší počet jejich kombinací a možných mezních situací. Testování tímto způsobem probíhalo po každé iteraci vývoje, vždy s přidáním další funkce byla otestována funkčnost celého editoru. V důsledku tohoto testování byla implementace několikrát upravena a znovu otestována.

[10] pozn.: Hudební notace nemá žádné mezinárodně ani národně platné standardy, které by bylo možné v takovýchto případech citovat.

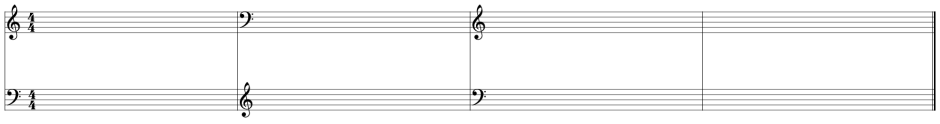
V následující sekci práce jsou uvedeny jednotlivé testované případy užití, rozdělené do podsekcí podle funkčních skupin. Editor umožňuje exportovat vytvořenou notaci do obrazového souboru ve formátu *PNG*. Takto vygenerovaný obrázek je přiložen ke všem scénářům, aby dokládal výsledek testování. Předchází jej stručný popis uživatelem provedené akce, a po obrázku následuje krátký komentář výsledku.

## 6.1 Vlastnosti taktu

### 6.1.1 Notové klíče

1. **Provedená akce:** Taktu č. 2 změněn původní klíč (pro obě osnovy).

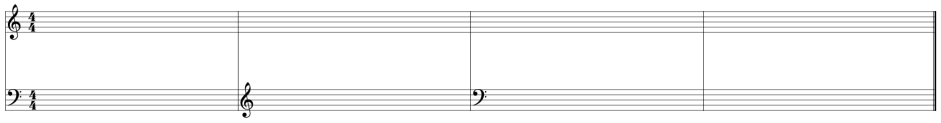
   **Výsledek:**



   **Komentář:** Editor korektně vykreslil klíče v druhém taktu a také v taktu následujícím, pro který je stále nastaven původní klíč, který je nyní potřeba opět vyznačit.

2. **Provedená akce:** Pokračování předchozího testu. Taktu č. 2 změněn klíč zpět na původní (tentokrát pouze pro vrchní osnovu).

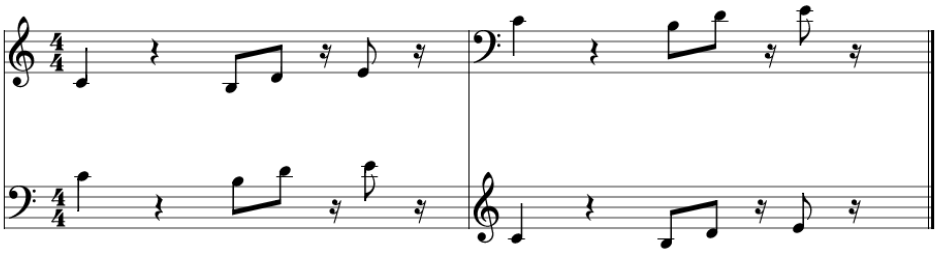
   **Výsledek:**



   **Komentář:** Na horní osnově editor korektně změnil klíč 2. taktu zpět na houslový a nevykreslil ho, jelikož je již vyznačen v taktu předchozím. Klíč je skryt také ve 3. taktu, jelikož je tento takt nastaven jako houslový. Editor vyznačuje klíče nezávisle pro obě osy.

3. **Provedená akce:** Vytvořen takt s notami a pomlkami v 1. taktu horní osnovy. Takt zkopírován na dolní osnovu s odlišným klíčem a také do 2. taktu na obě osnovy. Ve druhém taktu byly posléze změněny klíče na obou osnovách.

**Výsledek:**

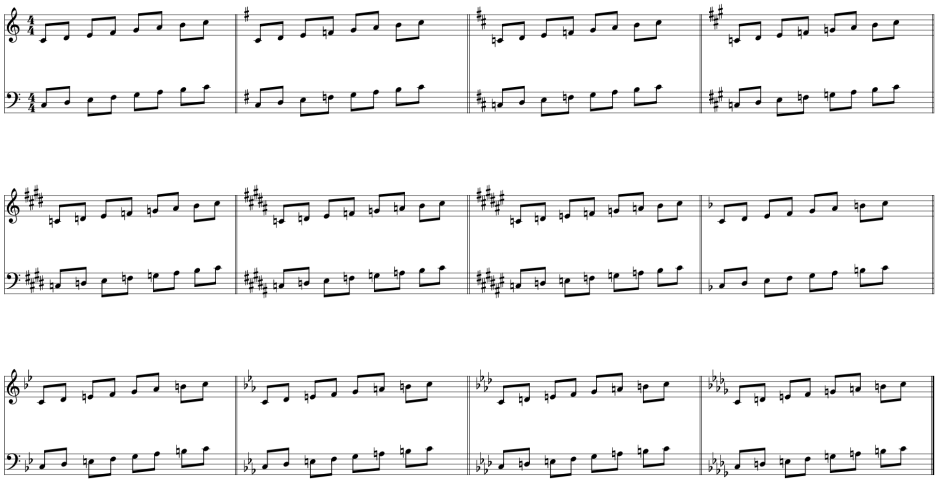


**Komentář:** Editor korektně vykreslil všechny 4 takty. Při změně klíče bylo také potřeba znovu pozicovat pomlky, jelikož VexFlow API umožňuje jejich vykreslení na jakékoliv lince a korektnost samo nehlídá.

### 6.1.2 Stupnice a předznamenání

1. **Provedená akce:** Stupnice *C-dur* zapsána pomocí osminových not do jednoho taktu a zkopírována do všech vytvořených taktů. Každému taktu posléze nastaveno jiné předznamenání.

**Výsledek:**



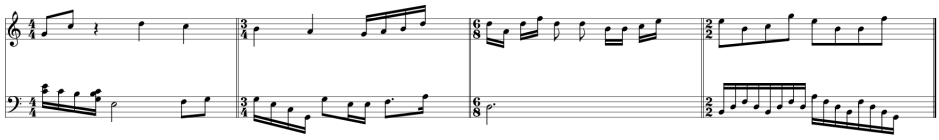
**Komentář:** Editor doplní znaménka odrážek před noty tak, aby stupnice *C-dur* zůstala stupnicí *C-dur* ve všech taktech. Jakákoliv standardní stupnice nebude obsahovat žádné odrážky pouze v tom

taktu, ve kterém je pro danou stupnici její vlastní předznamenání (zde to platí pro 1. takt).

### 6.1.3 Doby a metrum

1. **Provedená akce:** Nastavení některých z obvyklejších taktů: *4/4, 3/4, 6/8, 2/2.*

   **Výsledek:**

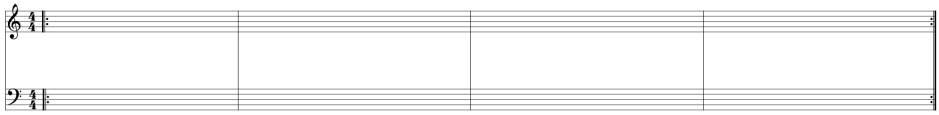


   **Komentář:** Editor korektně vykresluje *metrum* v předtaktí všech taktů. Zároveň hlídá, aby délka taktu nepřekročila omezení dané touto dobou. Noty jsou svázané tak, aby reflektovali zvolené *metrum* (např. šestnáctinové noty jsou ve 2., 3. a 4. taktu pokaždé svázány po jiných počtech, protože jedna doba v každém z těchto taktů reprezentuje jinou délku - *1/4*, *1/8* a *1/2*).

### 6.1.4 Repetice

1. **Provedená akce:** Prvnímu taktu nastaven začátek repetice.

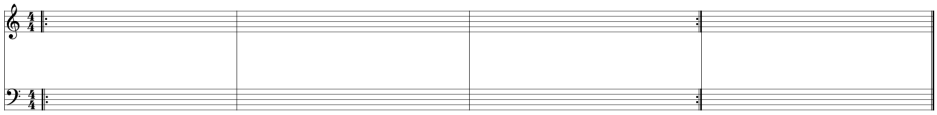
   **Výsledek:**



   **Komentář:** Editor automaticky nastavil konec repetice na konec skladby, protože nenašel žádný bližší konec repetice, ani začátek další repetice.

2. **Provedená akce:** Pokračování předchozího testu. 3. taktu nastaven konec repetice.

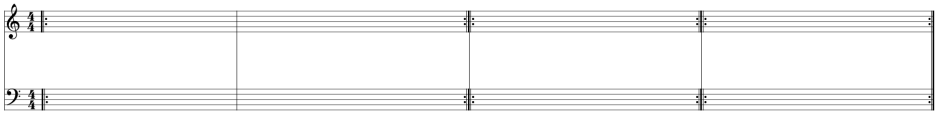
   **Výsledek:**



   **Komentář:** Editor ukončil repetici 3. taktem a ve 4. taktu zrušil původní konec repetice.

3. **Provedená akce:** Pokračování předchozích testů. 4. taktu nastaven začátek repetice, poté 2. taktu nastaven konec repetice a nakonec 3. taktu nastaven začátek repetice.

   **Výsledek:**



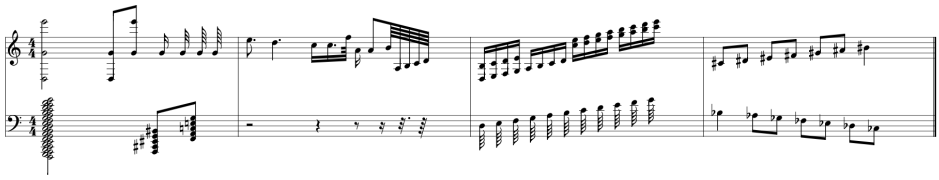
   **Komentář:** Editor nastavil konec repetice započaté ve 4. taktu na konec toho samého taktu, jelikož se zde nachází i konec skladby. Ve 3. taktu editor nastavil také konec repetice, protože ve 4. taktu narazil na začátek další repetice.

## 6.2 Přidávání/odebírání not

### 6.2.1 Přehled

1. **Provedená akce:** Přehled vykreslení not a pomlk všech editorem přípustných délek a výšek, včetně extrémních situací.

   **Výsledek:**

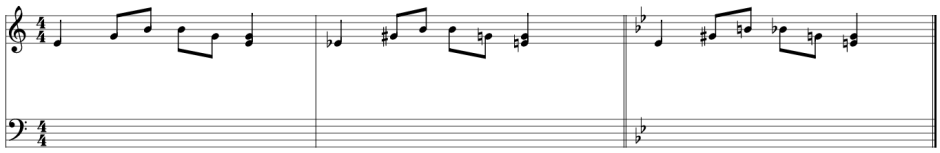


   **Komentář:** Editor zobrazí většinu extrémních situací korektně. Vykreslení některých extrémů však zvyk v hudební notaci jednoznačně neurčuje. Ty pak mohou vypadat po vykreslení zvláštně. V běžných případech se s nimi ale prakticky nelze setkat.

### 6.2.2 Předznamenání, posuvky a odrážky

1. **Provedená akce:** Vytvořen 1. takt (viz výsledek). Takt zkopírován do 2. taktu, poté ve 2. taktu nastaveno béčko první notě a křížek druhé notě. 2. takt je následně zkopírován do 3. taktu, kterému je změněno předznamenání na 2 béčka a u 4. noty nastaveno béčko.

   **Výsledek:**



   **Komentář:** Editor korektně vykreslí posuvky notám, kterým byla nastavena, a odrážku stejným notám v témže taktu, pokud následují po zvýšené/snížené notě. Pokud je navíc nastaveno nějaké předznamenání, editor nevykresluje předznamenané posuvky, dokud nenarazí na odrážku téže výšky. Příkladem je 4. nota 3. taktu, které je nastaveno béčko i přesto, že je v předznamenání již vyznačeno, nicméně se u předchozí noty stejné výšky nachází odrážka, která

ruší platnost předznamenání nejen pro tuto notu, ale i pro noty následující po ní v tomtéž taktu.

# Závěr

Tato práce si vzala za cíl implementaci front-endu aplikace pro práci s notovými zápisy za pomoci moderních webových technologií. Tento cíl se podařil splnit a použité technologie si vedly nad očekávání dobře.

*Python* je jazyk s obrovským potenciálem, nejen co se webu týče. Práce s ním je až neuvěřitelně rychlá, téměř jako jeho rychlost při samotném běhu programu v něm napsaném. Také často kritizovaný *JavaScript* se za svých 20 let existence dostal do podoby, kdy je práce s ním, hlavně zásluhou existujících knihoven a vývojových nástrojů, velice efektivní.

Výsledek této práce také dokládá, jak mocným standardem je HTML5, ačkoliv ještě stále není uzavřený. A že budoucnost *world wide webu* je v jeho rukou. To ostatně ukazuje i podpora, kterou mu věnují velké společnosti, které určují směr (nejen) webu v posledních letech či desetiletích. Doufám, že se v následujících letech dočkáme dalších zajímavých rozšíření HTML5 standardu a samotného elementu *canvas*.

I v oblasti notačních editorů a obecně interaktivního webu má HTML5 podle mého názoru co nabídnout. Tento jednoduchý editor to dokládá, jelikož si v některých ohledech vede nad moje očekávání. Například rychlost vykreslování a celková plynulost jsou velice pozitivní.

Otázkou však zůstává, jestli by vhodnější technologií nakonec nebylo vektorové *SVG*. Rastrovou formu vykreslení nepokládám v tomto případě za velké omezení, potřebu zvětšovat tento editor bude mít asi jen málokdo a i při zvětšení jsou noty čitelné. Avšak *SVG* si udržuje graf scény, pomocí kterého je možné již jednou vykreslené prvky notace přesouvat či zakrývat/odkrývat nebo mazat.

Teoreticky je *SVG* sice pomalejší než HTML5 canvas, ale bylo by potřeba praktického srovnání, aby se ukázalo, zda-li je toto zdržení natolik znatelné, aby ho uživatel vůbec dokázal zaznamenat. Nejnovější verze moderních prohlížečů již *SVG* optimalizovaly natolik, že se jeho rychlost prakticky vyrovná

HTML5 *canvasu*[11].

Django Framework mohu doporučit komukoliv, vývoj v něm je velice intuitivní a rychlý. Kdo má zkušenosti např. s nějakým *PHP MVC* frameworkem, dokáže Django začít používat během několika desítek minut. Jeho dokumentace je velice důkladná a podle mého názoru bude v budoucnu stale více vývojářů přecházet z PHP právě na tento i jiné Python frameworky.

[11]Pozn.: Srovnání k dispozici např. zde: http://bl.ocks.org/stepheneb/1296930

# Literatura


[1] *Adeste Fideles sheet music sample* [online]. Květen 2008. Dostupné z: http://commons.wikimedia.org/wiki/File:Adeste_Fideles_sheet_music_sample.svg

[2] *Finale - The World Standard in Music Notation Software* [online]. [cit. 2014-05-10]. Dostupné z: http://www.finalemusic.com/

[3] *Sibelius - the leading music composition and notation software* [online]. [cit. 2014-05-10]. Dostupné z: http://www.sibelius.com/home/index_flash.html

[4] *MuseScore - Tvořte, hrajte a tiskněte krásné noty* [online]. [cit. 2014-05-10]. Dostupné z: http://musescore.org/cs

[5] Kirn, P.: *Lilypond: Free, Beautiful Music Notation Engraving for Anyone* [online]. Květen 2010, [cit. 2014-05-10]. Dostupné z: http://createdigitalmusic.com/2010/05/lilypond-free-beautiful-music-notation-engraving-for-anyone/

[6] *Noteflight - Your music, everywhere.* [online]. [cit. 2014-05-10]. Dostupné z: http://www.noteflight.com/

[7] *Scorio - Music Notation for Everyone - simple, online, anywhere!* [online]. [cit. 2014-05-10]. Dostupné z: http://www.scorio.com/web/scorio/

[8] Jopa, G.: *HTML5 Cloud Composer App* [online]. [cit. 2015-05-8]. Dostupné z: http://www.gregjopa.com/2011/08/html5-cloud-composer-app/

[9] Silverman, C.: *An in-browser sheet music and tablature editor with JavaScript – Version 0.01 – Early Prototype* [online]. [cit. 2015-05-8]. Dostupné z: http://www.cyrilsilverman.com/tag/vexflow/

[10] Winokur, D.: *Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5* [online]. Listopad 2011, [cit. 2014-05-10]. Dostupné z: `http://blogs.adobe.com/conversations/2011/11/flash-focus.html`

[11] *HTML5 - A vocabulary and associated APIs for HTML and XHTML* [online]. [cit. 2014-05-10]. Dostupné z: `http://www.w3.org/TR/html5/`

[12] *alphaTab | music notation for everyone* [online]. [cit. 2015-05-8]. Dostupné z: `http://www.alphatab.net/`

[13] OpenSource: *HTML5 MusicXML Viewer* [online]. [cit. 2015-05-8]. Dostupné z: `http://www.musicxml-viewer.com/`

[14] Cheppudira, M. M.: *VexFlow - HTML5 Music Engraving* [online]. [cit. 2014-05-10]. Dostupné z: `http://vexflow.com/`

[15] *GitHub - VexFlow* [online]. [cit. 2014-05-10]. Dostupné z: `https://github.com/0xfe/vexflow`

[16] *Quotes about Python* [online]. [cit. 2014-05-10]. Dostupné z: `https://www.python.org/about/quotes/`

[17] *Meet Django*. [cit. 2014-05-10]. Dostupné z: `https://www.djangoproject.com/`

[18] *HTML5 Introduction* [online]. [cit. 2014-05-10]. Dostupné z: `http://www.w3schools.com/html/html5_intro.asp`

[19] *HTML5 New Elements* [online]. [cit. 2014-05-10]. Dostupné z: `http://www.w3schools.com/html/html5_new_elements.asp`

[20] Severance, C.: *JavaScript: Designing a Language in 10 Days. Computer*, ročník 45, č. 2, Únor 2012: s. 7–8.

Příloha **A**

# Seznam použitých zkratek

**GUI** Graphical user interface - *Grafické uživatelské rozhraní*

**WYSIWYG** What you see is what you get - *Výsledný dokument je při editaci vidět na obrazovce*

**HTML** HyperText Markup Language - *HyperTextový značkovací jazyk*

**XML** Extensible Markup Language - *univerzální značkovací jazyk*

**DOM** Document Object Model - *Objektová reprezentace HTML nebo XML dokumentu*

**CSS** Cascading Style Sheets - *Jazyk pro zápis vzhledu a formátování webového dokumentu*

**MVC** Model-View-Controller - *Druh softwarové architektury*

**PHP** Hypertext Preprocessor - *Skriptovací programovací jazyk*

**SVG** Scalable Vector Graphics - *Značkovací jazyk a formát souboru pro vektorovou grafiku*

**URL** Uniform Resource Locator - *Řetězec s umístěním a informacemi o zdroji na internetu*

**API** Application Programming Interface - *Specifikuje rozhraní mezi softwarovými komponenty*

**AJAX** Asynchronous JavaScript and XML - *Webová technologie pro změnu obsahu webových stránek bez jejich kompletního znovunačítání*

Příloha **B**

# Slovník použitých pojmů z hudební notace

*Slovník použitých pojmů z hudební notace s krátkým popisem a anglickým ekvivalentem:*

**notová osnova** *(stave, mn.č. staff)* - systém 5 vodorovných čar určený k notovému zápisu

**takt** *(bar, measure)* - úsek hudební skladby, na notové osnově ohraničen taktovými čarami

**taktová čára** *(barline)* - svislá čára, oddělující takty na notové osnově

**doba** *(beat)* - časový interval, základní jednotka rytmu

**nota** *(note)* - grafická značka tónu. Skládá se z **hlavičky** *(head)*, případně také **nožičky** *(stem)* a **praporce** *(flag)*

**pomlka** *(rest)* - rytmický interval, který znamená ticho

**trámec** *(beam)* - graficky spojuje osminové a kratší noty podle rytmu a také ukazuje jejich délku

**tečka** *(dot)* - symbol tečky za notou či pomlkou jí prodlužuje o polovinu její doby

**posuvka** *(accidental)* - „posunuje“ tón noty nahoru(křížek) či dolů(béčko) o půl noty

**notový klíč** *(clef)* - symbol pro identifikaci klíče k dekódování noty na tón

**předznamenání** *(key)* - symbol určující tóninu skladby

**metrum** *(time signature)* - symbol určující počet dob a délku základní doby jednoho taktu

**repetice** *(repetition, repeat barline)* - dvě svislé čáry s dvojtečkou, označující začátek a konec opakující se část skladby (dvojtečka směruje do opakované části)

Příloha C

# Obsah přiloženého CD

- `ctimne.txt` ..... stručný popis obsahu CD
- `impl`
  - `muzikologie` ..... adresář s Django projektem
  - `sandbox` ... adresář se spustitelným prototypem a zdrojovými soubory
- `latex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP Hejda Vojtech 2015.pdf` ..... text práce ve formátu PDF