Univerzita Hradec Králové
Fakulta informatiky a managementu
Název katedry

Dynamický web pro podporu výuky hry na kytaru

Bakalářská práce

Autor: Lukáš, Šára
Studijní obor: Informační management (3)

Vedoucí práce: doc. Mgr. Tomáš, Kozel, Ph.D.

Hradec Králové duben 2014

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 24.4.2015 Lukáš Šára

Poděkování:

Děkuji vedoucímu bakalářské práce panu doc. Mgr. Tomáši Kozlovi, Ph.D. za metodické vedení práce, praktické rady a zkušenosti.

Anotace

Účelem této práce bylo vytvořit a popsat proces vývoje dynamického webu pro podporu výuky hry na kytaru, který by sloužil jako pomocník pro kytarové hráče specializující se na akordovou hru. Webová aplikace je psaná na platformě JAVA, běží ve webovém kontejneru Apache Tomcat a je postavena na architektonickém vzoru MVC. Práce nejprve popisuje stručný úvod do hudební teorie, který je nutné znát pro efektivní využívání aplikace. Dále jsou specifikovány požadavky na dynamický web a následuje kapitola analyzující partie výuky hry na kytaru vhodné pro webové zpracování. Kapitola o návrhu softwarového řešení seznamuje čtenáře se základními kameny, na kterých aplikace stojí. Dále jsou uvedeny a stručně rozebrány technologie použité v aplikaci. V kapitole popisující proces implementace jsou tyto technologie detailněji rozebrány, často pomocí názorných příkladů převzatých z dynamického webu. Použité technologie byly Spring rámec, Hibernate JPA, Twitter Bootstrap, HTML5 canvas, Apache Maven, AJAX, JavaScript a JQuery.

Annotation

Title: Dynamic web for guitar play learning

The purpose of this bachelor's thesis is to create and describe the process of development of dynamic web for guitar play learning, which is supposed to help especially guitarists playing chords. Web application is written to run on JAVA platform, uses web container Apache Tomcat and is built-up by using MVC architecture. Thesis firstly describes outline of basic music theory to take advantage of the practical using of this application. Next, the requirements for the application are specified. The following chapter analyses suitable methods to improve guitar playing. These methods are developed into design model describing implementation's details. Thesis also contains a chapter about used technologies during implementation of the dynamic web, often with an example helping to understand the way the technology works. Technologies used are Spring framework, Hibernate JPA, Twitter, Bootstrap, HTML5, canvas, Apache Maven, AJAX, JavaScript and JQuery.

Obsah

Seznam obrázků

1 Úvod

Smyslem práce je popsat proces vývoje dynamického webu pro podporu výuky hry na kytaru. Dynamickým webem je myšlen praktický projekt, který je součástí této práce. V průběhu práce je na tento projekt často odkazováno.

Práce nejprve seznamuje čtenáře se základy hudební teorie, které jsou nezbytné pro pochopení fungování celého projektu. Následně jsou specifikovány požadavky na projekt a analyzovány výukové nástroje hry na kytaru vhodné pro webové zpracování. Čtenář se zde dozví, které funkce bude webový projekt podporovat.

Kapitola o návrhu softwarového řešení popisuje základní kameny, na kterých je projekt vystavěn. Dále jsou stručně představeny technologie, které byly v projektu použity. Kapitola popisující proces implementace detailněji rozebírá jednotlivé technologie. Je zde uvedeno množství názorných příkladů převzatých z praktického projektu, které pomohou čtenáři rychle pochopit, jak daná technologie pracuje. Poslední částí této práce jsou ukázky výstupu vybraných částí aplikace.

2 Cíl práce

Cílem práce je analyzovat možnosti webové podpory výuky hry na kytaru, navrhnout vhodné výukové nástroje a implementovat je jako dynamickou webovou aplikaci.

Práce si navíc dává za úkol pomoci čtenáři pochopit, jak technologie použité v praktickém projektu fungují.

3 Stručný úvod do hudební teorie

Kapitola stručně popíše základy hudební teorie a vysvětlí, jak tyto poznatky uplatnit na kytarovém pražci. V jejím průběhu budou také vysvětleny pojmy, které jsou použity v praktickém projektu.

3.1 Základ z hudební teorie

Existuje celkem dvanáct tónů [1]:

C, C#, D, D#, E, F, F#, G, G#, A, A#, H.

Každé dva sousední tóny jsou od sebe vzdáleny o půltón. Tóny s křížky se dají zapsat i jinak: stačí vzít tón o půltón výš a přidat k němu znak b [1].

Tón F# se například rovná tónu Gb.

Když se zvýší poslední dvanáctý tón opět o půltón, vznikne oktáva [2], která se značí stejným symbolem jako tón první. Oktáva je tón, který je o poznání vyšší než tón základní, ale zní skoro stejně.

Akord je obecně souzvuk minimálně třech tónů [3]. Pokud tedy znějí v jednom okamžiku alespoň tři tóny, jedná se o akord. Nejjednodušším typem akordů jsou takzvané triády (kvintakordy). Jedná se o souzvuk právě třech tónů [3]. V praktickém projektu byly použity triády typu:

dur, dur +5, moll, zmenšený akord, sus2 a sus4.

Všechny ostatní typy akordů mají více než tři tóny.

První tón akordu se označuje jako základní tón. Akord se tedy skládá ze základního tónu a typu akordu.

Například typ akordu maj7 a základní tón D vytvoří akord D maj7.

Akordy se liší jeden od druhého nejen počtem tónů, ale také charakteristickými intervaly. Pojmem charakteristické intervaly je myšlena vzdálenost ostatních tónů akordu od základního tónu [4]. Tento pojem vyžaduje hlubší pochopení problematiky harmonie a tvorby akordů, což není možné v této práci popsat. Zájemci o tuto

problematiku mohou využít dobře zpracovaný a poměrně jednoduše napsaný Kurz harmonie bez not [3].

3.2 Kytarový pražec

Kytarový pražec tvoří šest strun postupně od struny nejhlubší [5]:

E, A, D, G, H, E

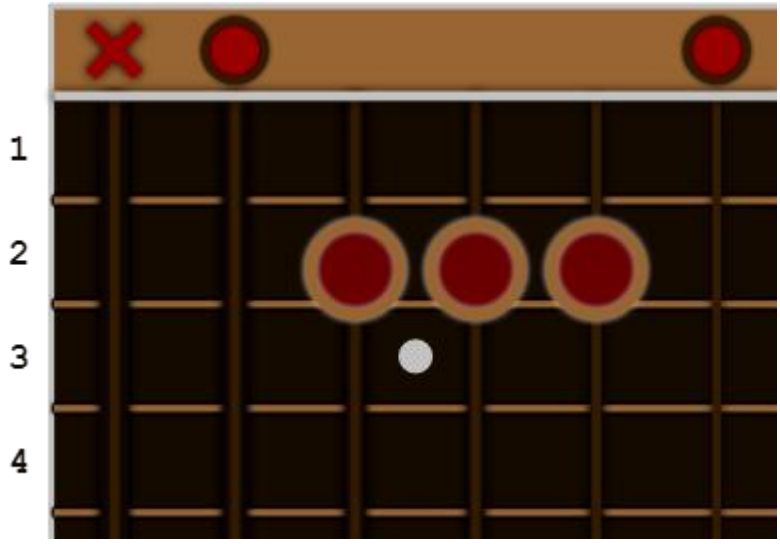
Název struny odpovídá tónu, který struna vydává bez přimáčknutí struny k tělu pražce. Kytarový krk je dále tvořen nejméně patnácti pražci. Na každém pražci zní daná struna o půltón výš.

Například přimáčknutím struny A na třetím pražci vznikne tón C.

Protože je dvanáct základních tónů a patnáct pražců na kytaře, obsahuje kytarová struna minimálně jednou každý tón. Akord na kytaře vznikne rozechvěním alespoň tří strun současně. Struny, na které se nehraje, se označují symbolicky křížkem.
Kytarový hmat je uchopení jednotlivých strun na konkrétních pražcích tak, aby vznikl akord. Například hmat akordu A dur se může zapsat jako množina čísel pražců na jednotlivých strunách od nejhlubší struny E:

x, 0, 2, 2, 2, 0 – na strunu E se nehraje, tóny akordu: A, E, A, C#, E.

Každý hmat je možné zobrazit také graficky. Na obrázku číslo 1 je zobrazen právě předchozí hmat akordu A dur.

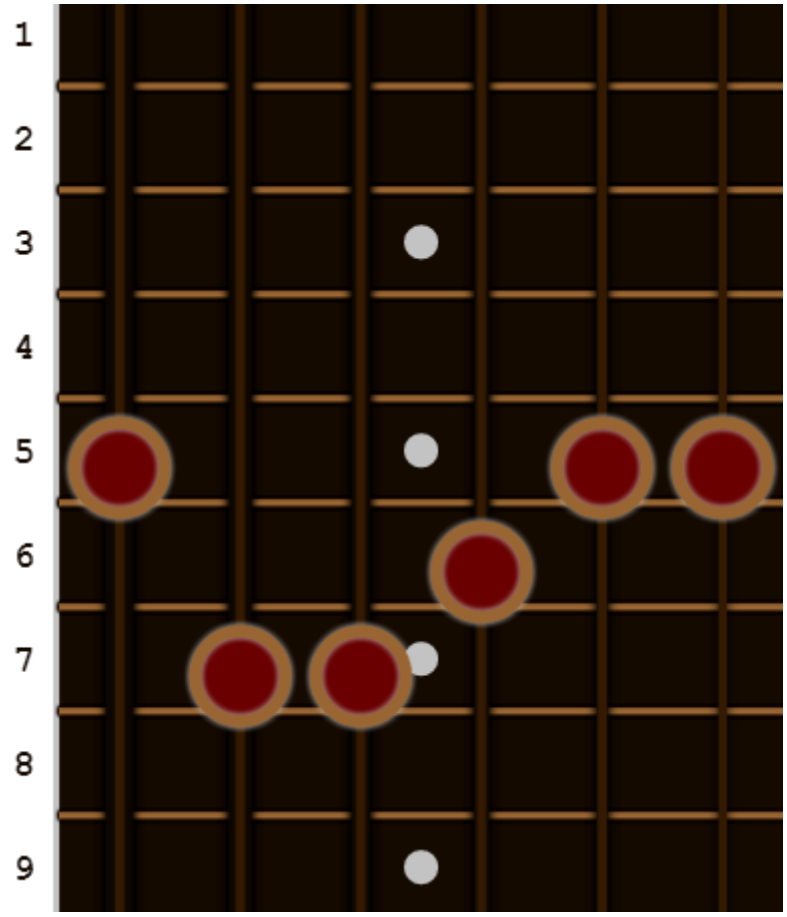


Obr. 1 Akord A dur první poloha
Zdroj: autor, vlastní zpracování

V praktickém projektu se používá pojem poloha akordu. Poloha akordu je synonymem pro hmat. Kytara umožňuje zahrát jeden akord více hmaty [6]. Existuje tedy více poloh akordu. Například předchozí akord A dur se může zahrát i jiným hmatem:

5, 7, 7, 6, 5, 5 – tóny akordu: A, E, A, C#, E, A

Tato poloha akordu je znázorněna na obrázku číslo 2.



Obr. 2 Akord A dur druhá poloha
Zdroj: autor, vlastní zpracování

4 Specifikace požadavků

Existuje mnoho způsobů, jak se učit hrát na kytaru. Ne všechny jsou však vhodné pro webové zpracování. Kapitola představí partie výuky hry na kytaru, které budou použity v praktickém projektu. Tyto výukové partie by měly sloužit jako rychlý pomocník především kytaristům, kteří se zaměřují na akordovou hru.

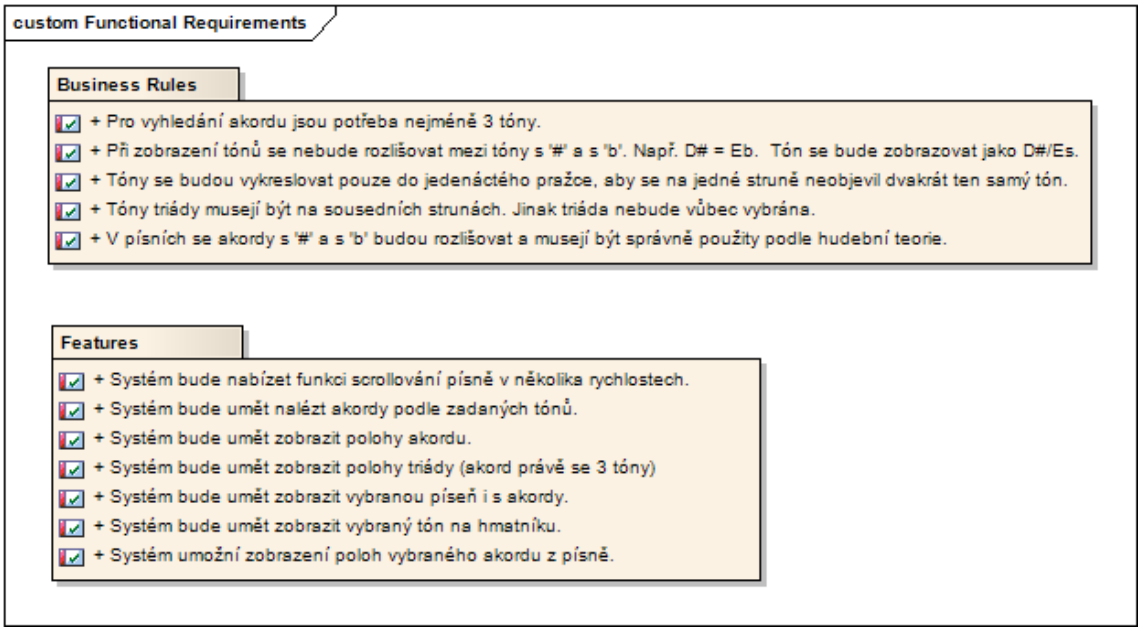
Pomocí specifikace UML (The Unified Modeling Language), která pomáhá modelovat a lépe navrhovat budoucí software [7], budou nejprve představeny funkční a non funkční požadavky na praktický projekt. Dále budou blíže popsány funkční požadavky a na závěr kapitola nabídne diagram případů užití, který specifikuje požadavky na systém z hlediska budoucího uživatele [8].

4.1 Funkční a non funkční požadavky

Funkční požadavky by měly popsat budoucí zamýšlené chování systému. Popisují, co by systém měl umět [9]. Non funkční požadavky shrnují nároky na kvalitu systému a vycházejí z firemního prostředí. Typickými non funkčními požadavky mohou být požadavky na spolehlivost, dostupnost, výkon, bezpečnost nebo persistenci dat [10].

4.1.1 Funkční požadavky

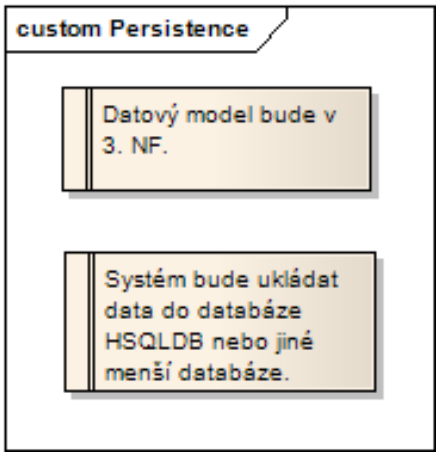
Funkční požadavky jsou znázorněny na obrázku číslo 3 v záložce „Features". Záložka „Business Rules" obsahuje pravidla, která úzce souvisejí s funkčními požadavky. Jedná se o specifické požadavky na systém či různá omezení, která musejí být v budoucím systému dodržena [11].



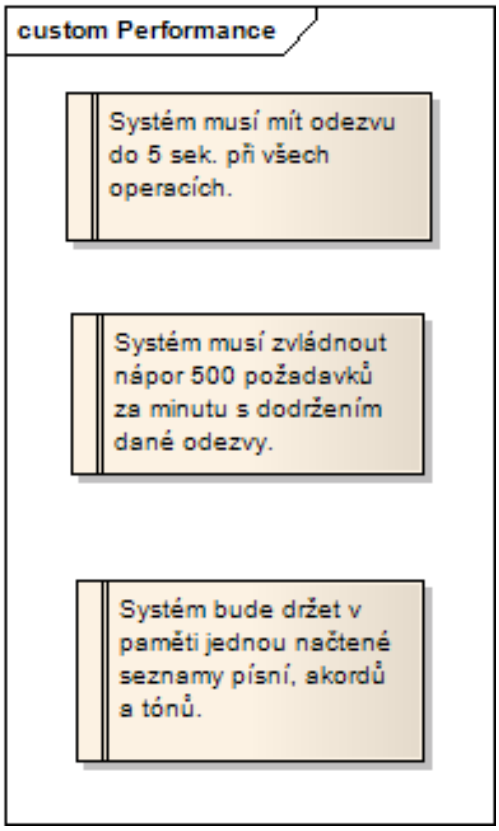
Obr. 3 Funkční požadavky
Zdroj: autor, vlastní zpracování

4.1.2 Non funkční požadavky

Budoucí systém by měl dodržet non funkční požadavky z hlediska persistence dat (obrázek číslo 4) a z hlediska výkonu (obrázek číslo 5). Na další kategorie nebyly kladeny speciální požadavky.



Obr. 4 Non funkční požadavky na persistenci dat
Zdroj: autor, vlastní zpracování



Obr. 5 Non funkční požadavky na výkon
Zdroj: autor, vlastní zpracování

4.2 Popis funkčních požadavků na systém

Funkční požadavky, které byly znázorněny v předchozí kapitole, zde budou detailněji vysvětleny a popsány.

4.2.1 Zobrazení akordu a jeho poloh

První a základní partií, která bude implementována v praktickém projektu, je zobrazení akordu na hmatníku. Uživatel má na výběr všechny možné tóny a typy akordů, ze kterých si zvolí akord. Tento akord se následně zobrazí na kytarovém pražci, přičemž jsou k dispozici všechny polohy akordu. Součástí zobrazení akordu je také název akordu a všechny tóny, které do akordu patří.

Uživatel například zvolí typ akordu: dur od tónu: C. Uživateli se zobrazí základní poloha akordu C dur, bude však možné zobrazit i jinou polohu akordu C dur.

4.2.2 Vyhledání akordu podle zadaných tónů

Další partií výuky hry na kytaru je vyhledání a zobrazení akordu podle zadaných tónů. Jde tedy o opačný postup než v předchozí kapitole. Uživatel si vybere tóny, které mají do akordu patřit. V okamžiku vybrání tří tónů jsou nalezeny všechny akordy, které dané tóny obsahují. Uživatel může výběr akordů dále specifikovat výběrem dalších tónů. Pokud uživatel klikne na konkrétní akord, je zobrazen hmat tohoto akordu s dalšími polohami a informacemi o akordu (viz předchozí kapitola).

Uživatel například zvolí tóny: C, E, G. Budou zobrazeny všechny akordy s těmito tóny: C dur, Ami7, G13 atd.

4.2.3 Zobrazení triády

Uživatel si tentokrát vybírá pouze z typů akordů, které obsahují právě tři tóny. V okamžiku, kdy si uživatel zvolí konkrétní triádu, zobrazí se triáda se všemi polohami a informacemi na kytarovém pražci. Pravidla pro zobrazování triád jsou jiná než u akordů. Každý tón triády se zobrazí pouze jednou a zobrazené tóny musejí být na sousedních strunách.

4.2.4 Zobrazení jednoho tónu

V praktickém projektu bude dále implementována partie týkající se zobrazení jednoho tónu na hmatníku. Uživatel si vybere jeden tón, který se vykreslí právě jednou na každé

struně. Tóny se vykreslují pouze do jedenáctého pražce, aby se na jedné struně neobjevil dvakrát ten samý tón.

4.2.5 Písně

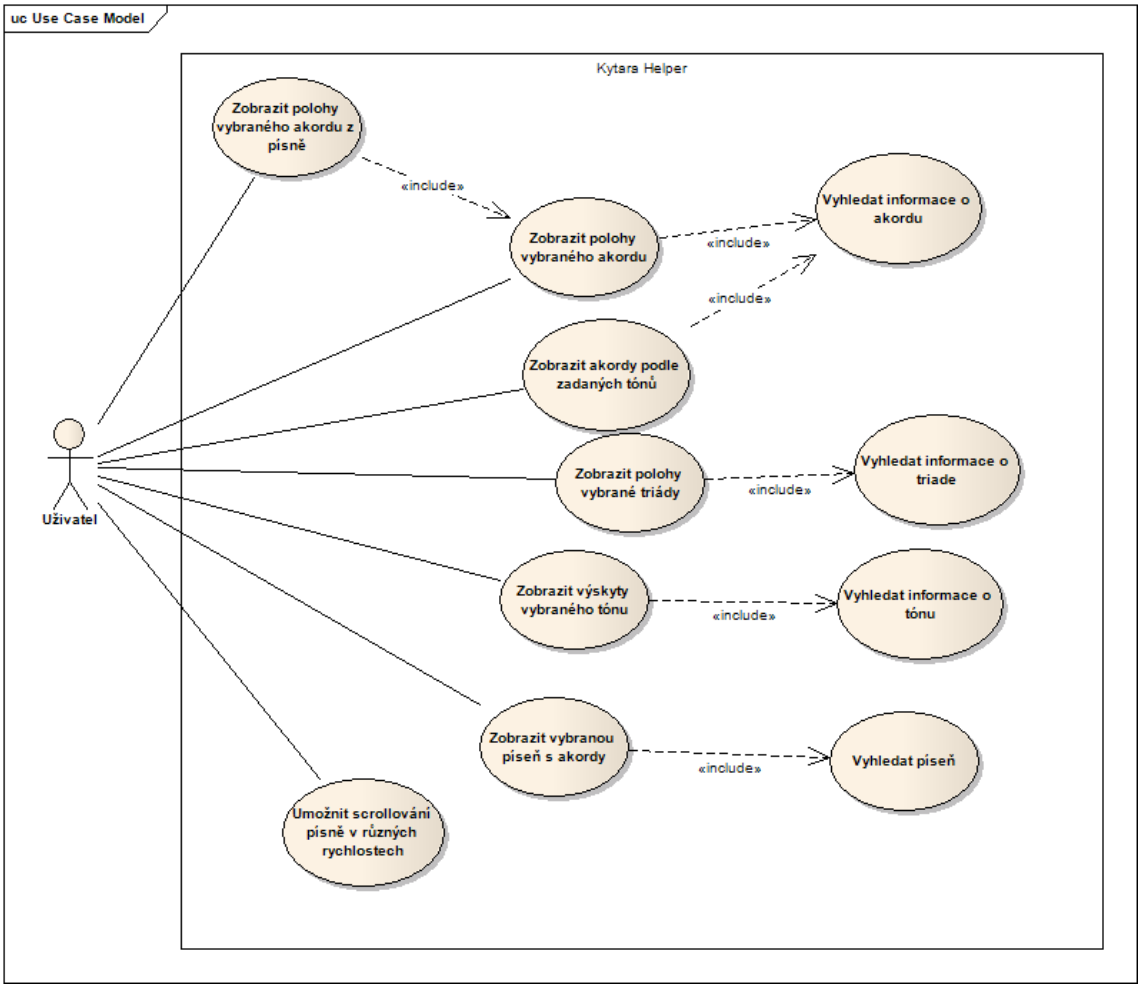
Poslední partií v praktickém projektu budou texty vybraných písní. Jde o seznam názvů písní. Uživatel si vybere konkrétní píseň, jejíž text a název se zobrazí. V textu budou umístěny také akordy, podle kterých si uživatel danou píseň bude moci zahrát. Když uživatel klikne na akord, zobrazí se hmat akordu s jednotlivými polohami a informacemi o akordu. Jako součást zobrazení písně bude systém nabízet funkcionalitu rolování textu od shora dolů v několika rychlostech.

4.3 Diagram případů užití

Diagram případů užití zachycuje přehledně funkcionalitu mezi systémem a uživatelem [12]. Diagram je vhodné použít jako prezentace pro zákazníka, neboť odpovídá na otázku, co vlastně zákazník od systému očekává. Diagram je tvořen třemi základními elementy [12]:

aktér, typová úloha a asociace.

Aktérem je nečastěji osoba, ale může jím být také externí systém, organizace nebo čas. Aktér stojí mimo systém, ale komunikuje s ním. Typovou úlohou se rozumí akce v systému, která přinese aktérovi užitek. Asociace jsou vztahy mezi aktérem a typovou úlohou. Tyto tři základní elementy dále doplňuje hraniční box systému, což je ohraničení vymezující systém od jeho okolí [12]. Obrázek číslo 6 znázorňuje diagram případů užití budoucího praktického projektu.



Obr. 6 Diagram případů užití
Zdroj: autor, vlastní zpracování

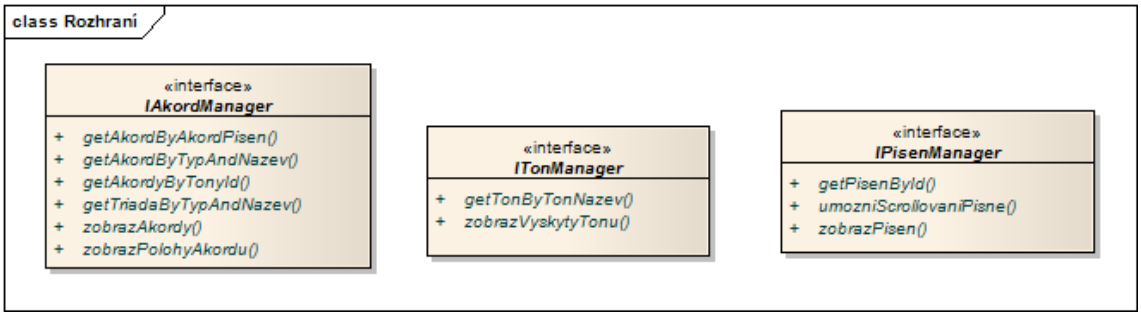
Z obrázku je patrné, že systém bude mít jednoho aktéra, který je napojen na sedm typových úloh přinášejících aktérovi užitek. Vazba „include“ se používá pouze mezi typovými úlohami a znamená, že se vložená typová úloha vloží do základní typové úlohy [13]. Například pro zobrazení akordu je nutné vyhledat informace o tomto akordu.

5 Analýza partií výuky vhodných pro webové zpracování

Již jsou vytvořeny požadavky na budoucí software. Nyní je třeba analyzovat, jak by software mohl vypadat. Kapitola zobrazí a popíše analytický model vytvořený pomocí jazyka UML a způsob, jak tento model vznikl.

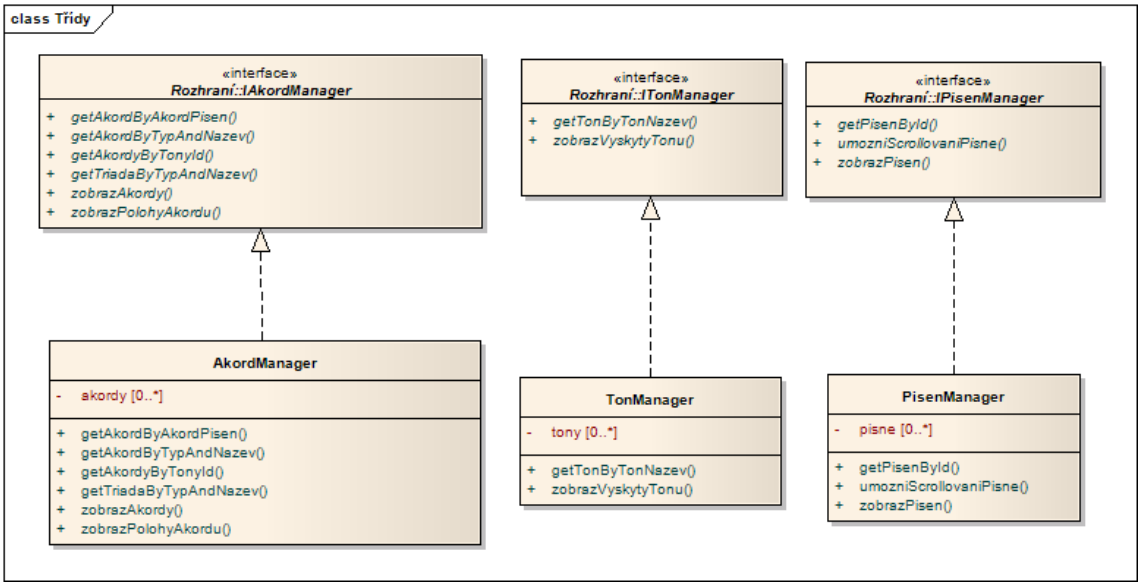
5.1 Rozhraní a třídy

Aby mohl vzniknout analytický model, je třeba nejprve definovat rozhraní a třídy, které budou součástí tohoto modelu. Rozhraní by měla vzniknout metodou realizace případů užití, která popisuje interakci mezi objekty pro dosažení požadovaného výstupu jedné typové úlohy [14]. Je důležité vytvořit rozhraní, která jsou vnitřně soudržná a mají jasně definovanou množinu odpovědností neboli operací [14]. Obrázek číslo 7 znázorňuje rozhraní pro budoucí praktický projekt.



Obr. 7 Analytická rozhraní
Zdroj: autor, vlastní zpracování

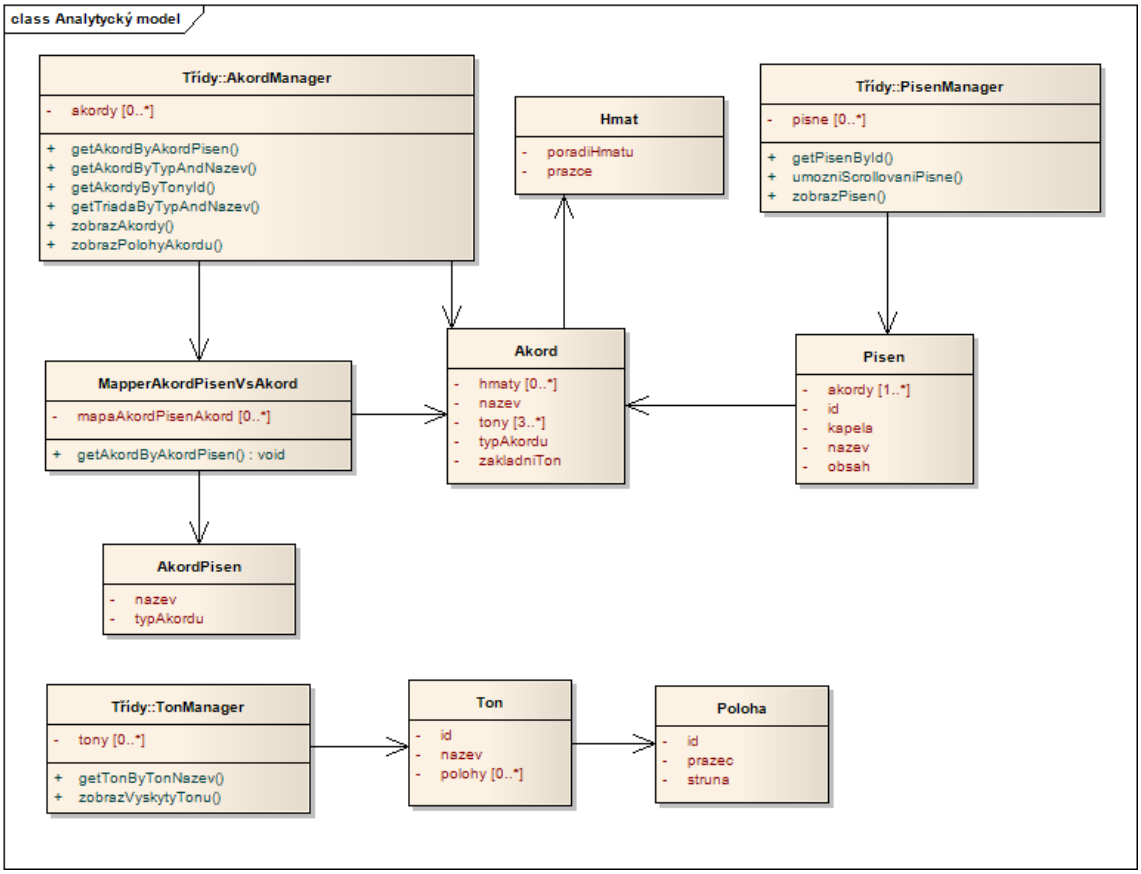
Jakmile existují rozhraní, vytvoří se třídy, které tato rozhraní budou implementovat. Analytické třídy jsou znázorněny na obrázku číslo 8.



Obr. 8 Analytické třídy
Zdroj: autor, vlastní zpracování

5.2 Analytický model

Nyní je čas na tvorbu analytického modelu. Analytický neboli doménový model by měl znázorňovat business logiku dané domény a udržovat důležité informace o této doméně [15]. Model by měl být snadný pro pochopení klíčové problematiky a neměl by tedy zatěžovat implementačními detaily – implementační nezávislost [16]. Analytický model je složen ze tříd a z relací mezi nimi. U tříd se obvykle modelují klíčové atributy a operace, které je však možné vynechat, je-li model zaměřen pouze na znázornění relací [17]. Analytický model pro budoucí praktický projekt znázorňuje obrázek číslo 9.



Obr. 9 Analytický model
Zdroj: autor, vlastní zpracování

Operace na obrázku splňují všechny požadavky, které byly popsány v předešlé kapitole a které byly znázorněny v diagramu případů užití. Za zmínku stojí fakt, že aplikace si bude uchovávat mapu mezi akordy v písni a akordy v databázi. Důvodem je odlišné pojmenování obou skupin akordů, které plyne z funkčních požadavků na systém.

6 Návrh softwarového řešení

Návrh softwarového řešení se již bude zabývat implementačními detaily. Kapitola bude rozdělena do dvou částí. Nejprve bude stručně charakterizována vybraná platforma společně s prostředím, ve kterém se praktický projekt bude vyvíjet, a základním architektonickým vzorem, nad kterým bude projekt postaven. Druhá část se bude věnovat návrhovému modelu, který by měl zpřesnit analytický model o implementační detaily.

6.1 Návrh platformy a prostředí

Praktický projekt bude napsán nad platformou JAVA. Projekt se bude vyvíjet v IDE Eclipse a speciálně view vrstva pomocí IDE Sublime_text. Celý projekt poběží ve webovém kontejneru Apache Tomcat verze 7.0.

Projekt bude dále postaven na architektonickém vzoru MVC (Model-View-Controller). Hlavní myšlenkou MVC je oddělení logiky od view vrstvy, ke které má přístup uživatel. Aplikace postavená na MVC zaručuje větší bezpečnost a má přehlednější a znovupoužitelný kód. Každá komponenta v MVC architektuře je něčím charakteristická [18]:

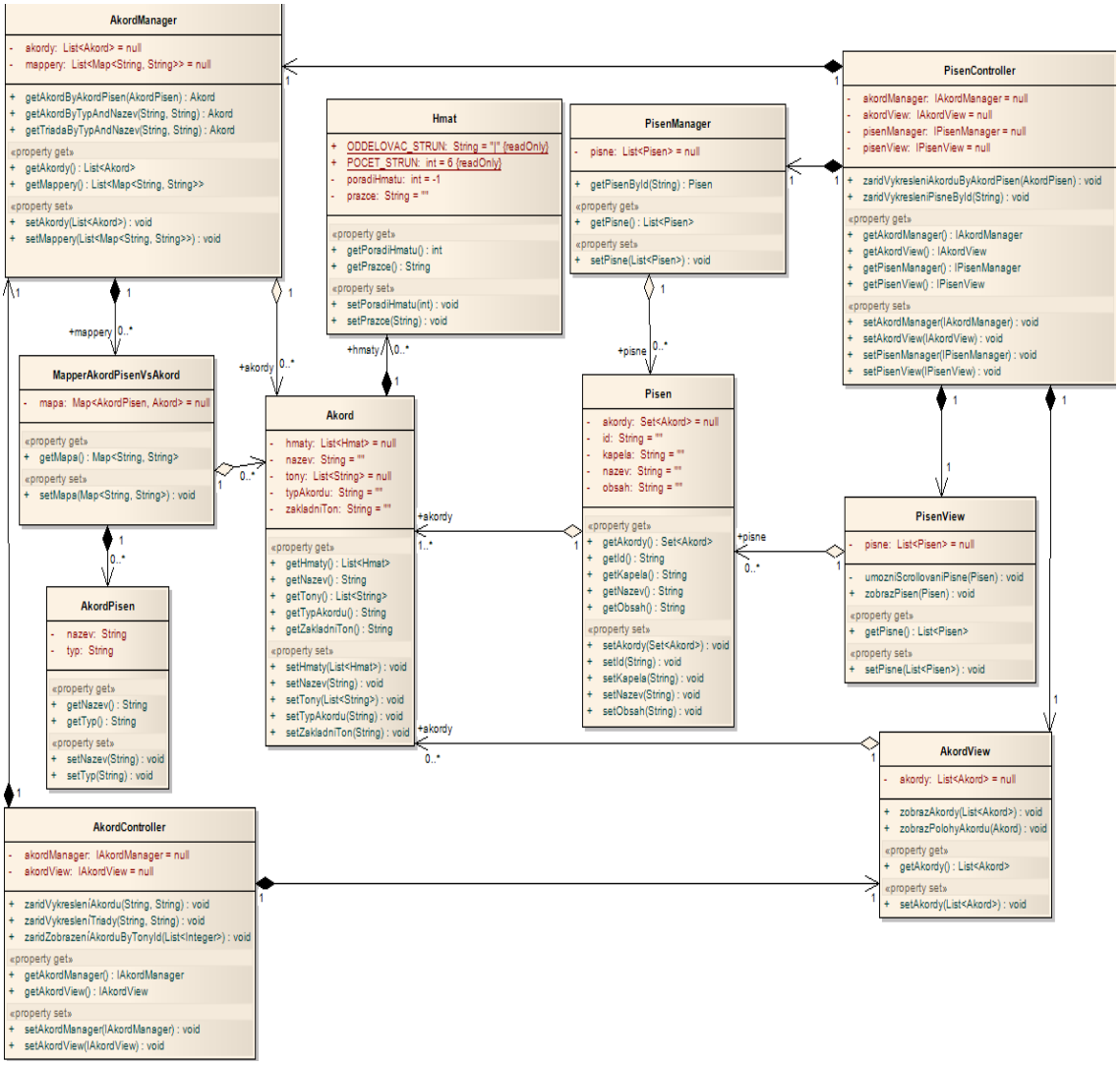
Model – obsahuje veškerou logiku, přistupuje k databázi, nemá ponětí o dalších vrstvách.
View – stará se o zobrazení dat uživateli, nemá ponětí o dalších vrstvách, nejčastěji používá JSP šablonu, ze které se generuje formát HTML.
Controller – komunikuje s ostatními komponentami, drží celý systém pohromadě.

MVC funguje následovně [18]: Uživatel pošle Http požadavek s parametry, který kontrolér odchytí a přes model vrstvu získá příslušná data. Dále kontrolér zavolá view vrstvu, která data vloží do připravené šablony. View sestaví stránku a zobrazí ji uživateli. Data, která uživatel může měnit, jsou trvale uložena v databázi. Při každé akci vyvolané uživatelem jsou z databáze vytažena příslušná data, která jsou následně uživateli zobrazena. Bude se tedy jednat o dynamický web [19], který reaguje na uživatelovy vstupy.

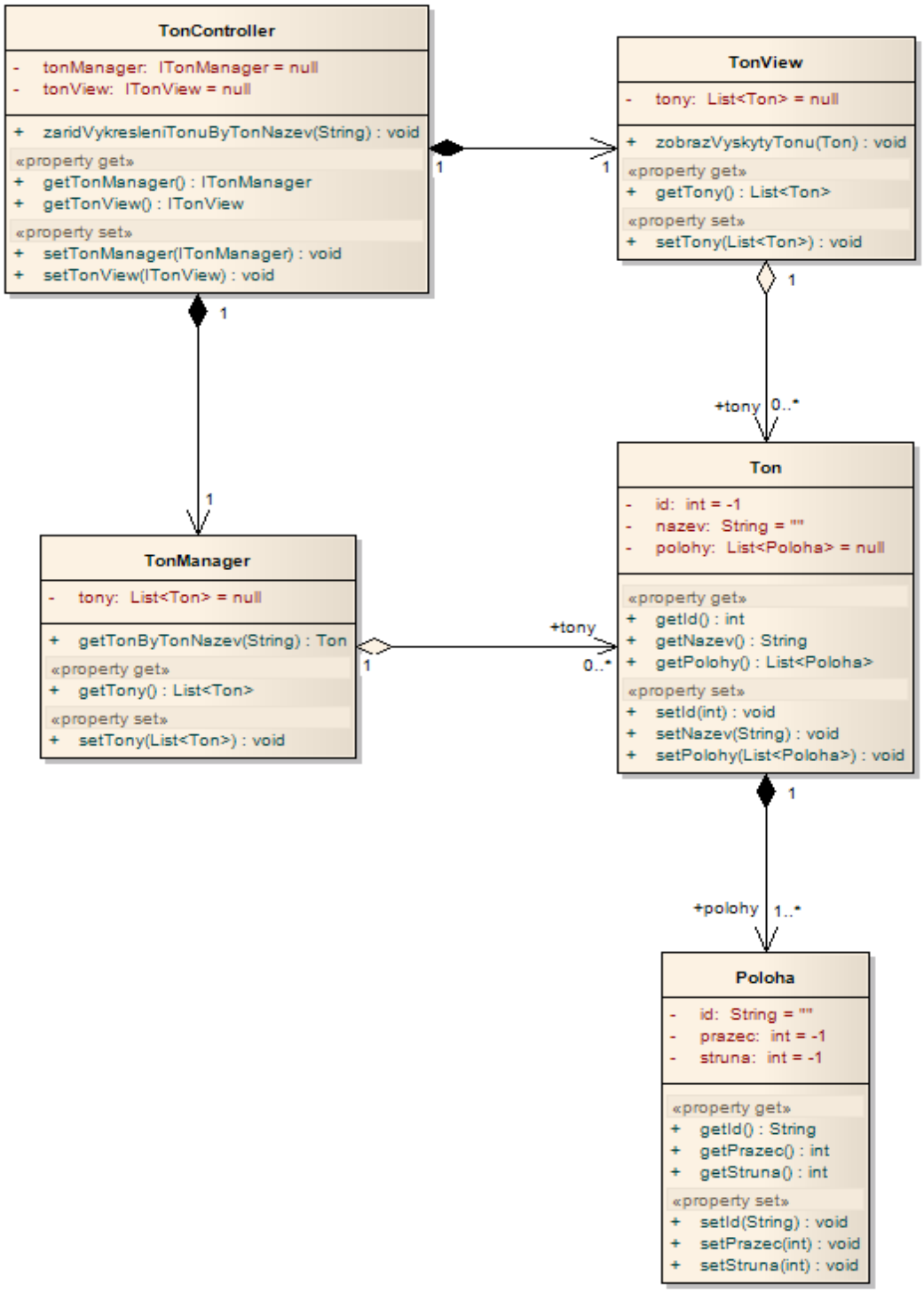
6.2 Návrhový model

Návrhový model vzniká postupným zpřesňováním a rozšiřováním analytického modelu [14]. Návrhová třída již obsahuje viditelnost u atributů a metod, typy atributů

a parametrů v metodách, návratové typy a další detaily závislé na dané implementaci [14]. Dále dochází ke zpřesňování analytických relací. Obecně se obyčejná asociace nahradí agregací nebo kompozicí a doplní se multiplicita. Tyto speciální relace značí vazbu celku a části, kde celek využívá služeb části [14]. Návrhový model je oproti analytickému rozšířen o třídy zajišťující uživatelské rozhraní a třídy obsluhující systémové události [17], už se tedy nejedná pouze o doménovou část aplikace. Návrhový model praktického projektu je rozdělen do dvou částí. Je znázorněn na obrázku 10 a 11.



Obr. 10 Návrhový model část A
Zdroj: autor, vlastní zpracování



Obr. 11 Návrhový model část B
Zdroj: autor, vlastní zpracování

Na modelu je patrná MVC architektura webové aplikace, kde kontrolér má na starost přerozdělování úkolů. Nejprve volá manažera, který vrátí požadovaný objekt. Tento objekt předá třídě z view vrstvy s požadavkem o vykreslení.

7 Výběr vhodných technologií pro implementaci

Kapitola stručně popíše a zdůvodní výběr technologií, které byly použity pro implementaci praktického projektu. V následující kapitole budou jednotlivé technologie detailněji rozebrány.

7.1 Apache Maven

Pro centrální správu závislostí a sestavení projektu bude použit software Apache Maven. Maven umí zjistit nejnovější verzi potřebné knihovny a následně ji také stáhnout. Knihovny se nemusí ručně kopírovat do cesty projektu (classpath), což vede zvláště u velkých projektů k větší přehlednosti a zjednodušení vývoje [20]. Maven se automaticky stará o všechny další závislosti, které je třeba nainstalovat. Tato vlastnost se označuje jako Transitive Dependencies [21]. Alternativní technologií pro Maven je software Apache Ant. Technologie Apache Maven oproti Apache Ant používá princip „konvence nad konfigurací“ („Convention over configuration“), což znamená využívání stále stejné struktury projektu [22]. Tento fakt byl hlavním důvodem pro volbu technologie Apache Maven. Další informace o softwaru Apache Maven jsou k dispozici na oficiálních stránkách [23].

7.2 Spring rámec

Jak bylo již řečeno, praktický projekt je založený na MVC architektuře. Ústředním rámcem, který pokrývá všechny vrstvy MVC je Spring. Spring byl uveřejněn v roce 2003 Rodem Johnsonem [24]. Výhoda Springu spočívá v jeho komplexnosti, která ovšem nepodmiňuje jeho použití v celé aplikaci. Spring se tak může kombinovat i s dalšími technologiemi, jako je například EJB (Enterprise Java Beans), Stripes či Struts [25]. Spring řeší v praktickém projektu kromě jiného i připojení k databázi. K samotnému připojení je použita technologie Hibernate JPA. Spring rámec však zajišťuje podporu této technologie pomocí vlastního adaptéru [26].

Ve velké části MVC architektury může Springu konkurovat technologie Stripes. Stripes je open source prezentační rámec, jehož hlavním cílem je zjednodušení vývoje webových aplikací [27]. Stripes vylepšuje dříve velmi oblíbený rámec Apache Struts tím, že využívá anotace. Nemusí se tedy udržovat žádné konfigurační soubory. Další výhodou Stripes je, že uchovává v jedné třídě více akcí, které zpracovávají data od prohlížeče [28]. Zatím poslední verze Stripes 1.5.8 byla vydána 7. 7. 2014 [27].

Technologie Stripes je zaměřena výhradně na MVC architekturu, ve které může plně konkurovat Springu. Pro práci s MVC se tak Stripes jeví jako jednodušší a elegantnější technologie [29]. Díky zájmu autora blíže poznat Spring však bude v praktickém projektu použita právě Spring technologie. Další informace o technologii Stripes může čtenář nalézt například v tutoriálu: Stripes: a lean, mean Java web framework [30].

7.3 HSQLDB

Další použitou technologií bude HSQLDB (HyperSQL DataBase), což je malá relační databáze kompletně napsaná v jazyce JAVA [31]. Databázi je možné spustit ve dvou základních módech: server mód a samostatný (standalone) mód [32]. Pro účely praktického projektu budou data spravována pomocí HSQLDB manažera, který běží v server módu. HSQLDB je vhodná databáze na menší projekty, což je hlavní důvod jejího použití v praktickém projektu.

7.4 HTML5 canvas

Pro vykreslení kytarového pražce bude použita technologie HTML5 canvas, která umožňuje vykreslování grafiky pomocí JavaScriptu [33]. Jako alternativa ke canvasu existuje SVG (Scalable Vector Graphics), což je vektorový grafický formát [34]. Obvykle se pro uživatelská rozhraní používá právě SVG, zejména pro její nezávislost na rozlišení obrazovky a formát založený na xml [33]. V praktickém projektu bude však použita technologie HTML5 canvas. Budou vykreslovány jen části stránek, navíc budou tyto části s využitím technologie Twitter Bootstrap nezávislé na rozlišení.

7.5 Twitter Bootstrap

Technologie Twitter Bootstrap bude použita pro zrychlení vývoje JSP stránek. Jedná se o moderní front-end rámec, který kromě jiného řeší responzivní design na různě velkých výstupních zařízeních [35]. Alternativou k technologii Twitter Bootstrap je Foundation. Obě jsou velmi populární, Bootstrap má větší komunitu a je pravděpodobné, že se v budoucnu bude rychleji rozvíjet [36].

7.6 JQuery, AJAX

V JSP stránkách bude použita knihovna JQuery zefektivňující práci s JavaScriptem [37]. JQuery podporuje technologii AJAX (Asynchronous JavaScript and XML), která umožňuje asynchronní zpracování webových stránek a tím změnu pouze části těchto stránek bez

nutnosti obnovení celé stránky [38]. Technologie AJAX bude v praktickém projektu využita pro rychlé vyhledání a zobrazení množiny akordů podle zadaných tónů.

8 Popis procesu implementace

Technologie představené v minulé kapitole budou nyní důkladně rozebrány. Nepůjde o praktický projekt jako takový, ale spíše o pochopení, jak daná technologie pracuje. Ke každé technologii je uveden dostatečný počet příkladů (často převzatých z praktického projektu), které by měly pomoci zorientovat se a pochopit základní souvislosti.

8.1 Apache Maven

Aby mohl vývojář používat Maven, stačí v prostředí Eclipse vytvořit nový Maven projekt [39]. Automaticky se ve struktuře projektu vytvoří soubor pom.xml, což je hlavní konfigurační soubor pro Maven [40]. Pro přidání závislosti stačí v tomto souboru kliknout na pole přidat závislost a napsat, jakou závislost je třeba přidat. Maven by měl dohledat závislost a nabídnout aktuální verzi. Dále Maven dohledá a stáhne všechny ostatní závislosti, které jsou pro build projektu potřebné.
Po přidání závislosti vytvoří Maven záznam o této závislosti do souboru pom.xml:

```
<dependency>
  <groupId>org.hsqldb</groupId>
  <artifactId>hsqldb</artifactId>
  <version>2.3.2</version>
</dependency>
```

Maven kromě jiného umožňuje vyčlenění verze knihovny do atributu v pom.xml, což je výhodné pro udržování verze projektu na jednom místě [40]. Vhodným příkladem by mohl být Spring rámec, který je rozdělen na více samostatných závislostí se stejnou verzí:

```
<properties>
  <spring.version>4.1.2.RELEASE</spring.version>
</properties>
<!-- Spring rámec -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${spring.version}</version>
</dependency>
```

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${spring.version}</version>
</dependency>
```

8.2 Spring

8.2.1 Úvod do Springu

Nejvíce využívaným rámcem v této práci je Spring. Jedná se o modulární aplikační rámec, který umožňuje kompletní vývoj jakékoliv aplikace – ať už webové či desktopové. Spring mimo jiné umožňuje aspektově orientované programování, deklarativní management transakcí či různé možnosti ukládání dat [41]. Aktuální verze rámce Spring je 4.1.4 [42].

Rámec Spring podporuje všechny vrstvy aplikace, od prezentační po business logiku [43]. K tomu Spring integruje další nástroje – např. Hibernate. Takové nástroje lze kdykoliv využít. Důležité ale je, že si Spring nevynucuje využívání celého rámce, ale pouze jeho libovolné části.

8.2.2 Konfigurace Springu

Pro využívání rámce je nutné nejprve springový rámec nakonfigurovat. Samotný Spring nabízí dva rozdílné pohledy na konfiguraci [44]. Starší, ale více využívanou konfiguraci pomocí xml souborů, nebo novější konfiguraci pomocí java kódu. V práci bude představena pouze první možnost pomocí xml souborů, která byla také využita v praktickém projektu.

8.2.2.1 Načtení kontextu

Základním konfiguračním souborem, který Spring očekává v adresáři Nazev_projektu\src\main\webapp\WEB-INF je web.xml [45]. Tento soubor neslouží pouze pro Spring, ale je hlavním konfiguračním souborem obecně pro webové aplikace. Spring od něj očekává načtení rodičovského (parent) a dětského kontextu. Rodičovský kontext se vytvoří přidáním ContextLoaderListener do web.xml [46]:

```
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

Rodičovský kontext se mimo jiné stará o start či ukončení aplikačního kontextu Springu, čtení bean a anotací či aspektově-orientované programování.
Načtení dětského kontextu probíhá pomocí DispatcherServlet ve web.xml [47]:

```
<servlet>
  <servlet-name>kytara</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <init-param>
    <param-name>namespace</param-name>
    <param-value>kytara-servlet</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Dětský kontext platí pouze v rámci daného sevletu. Konfigurace k tomuto servletu a tedy i k celému dětskému kontextu je uvedena v souboru kytara-servlet.xml ve stejném adresáři jako web.xml. Servlet je dále nutné namapovat pomocí:

```
<servlet-mapping>
  <servlet-name>kytara</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

Dětský kontext je namapován na všechny http požadavky.

8.2.2.2 Konfigurace rodičovského kontextu

Kontexty jsou již načteny. Nyní je potřeba vytvořit konfiguraci pro oba kontexty. Nejprve je tedy nutné vytvořit konfiguraci rodičovského kontextu. V této konfiguraci by měla být obecná nastavení platící v celém projektu. Ve složce WEB-INF/conf vznikne soubor applicationContext.xml s následujícím obsahem:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mybatis="http://mybatis.org/schema/mybatis-spring"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd
    http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-
spring.xsd
    http://www.springframework.org/schema/util
  http://www.springframework.org/schema/util/spring-util-3.2.xsd">
  <context:annotation-config />
  <context:component-scan base-package="cz.cr.sara.kytara">
  </context:component-scan>
  <bean id="propertyPlaceholderConfigurer"
  class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
```

```
      <list>
        <value>classpath:/properties/default.properties</value>
        <value>classpath:/properties/db/hsqldb.properties</value>
      </list>
    </property>
    <property name="fileEncoding" value="utf-8" />
    <property name="ignoreUnresolvablePlaceholders" value="true" />
    <property name="ignoreResourceNotFound" value="true" />
    <property name="nullValue" value="" />
  </bean>
</beans>
```

Annotation-config umožňuje automatickou injektáž (autowiring) bean, které jsou obsaženy v xml souboru [48]. Spring dokáže injektovat beanu z xml souboru do objektu v javě, který je označen anotací @Autowired například podle typu rozhraní. Ještě silnější vlastnost je component-scan, při níž Spring prohledává libovolný balíček projektu a všechny jeho podbalíčky se stejným cílem jako u annotation-config [48]. Výhodou u tohoto principu je fakt, že se beany nemusejí vkládat do xml souboru, ale Spring je nalezne přímo v projektu.

Posledním nastavením pro rodičovský kontext, které bylo použito v praktickém projektu, je propertyPlaceholderConfigurer. Pokud Spring najde v projektu speciální regulární výrazy, bude se je snažit nahradit hodnotou ze souboru properties podle daného klíče [49]. Vše slouží k oddělení nastavení či prostého textu od výkonného kódu. V případě například změny databáze stačí pouze upravit soubor properties a není třeba upravovat konfigurační soubory.

Aby Spring věděl o konfiguraci rodičovského kontextu, je třeba přidat následující tag do web.xml:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/conf/applicationContext.xml</param-value>
</context-param>
```

8.2.2.3 Konfigurace dětského kontextu

Jak bylo popsáno v kapitole o web.xml, Spring hledá dětský kontext v souboru kytara-servlet.xml a tento soubor platí pro všechny http požadavky. Obecné schéma souboru vypadá následovně:

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc
  http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context2.5.xsd">
</beans>
```

Pro nastavení lokalizace je nutné do této šablony vložit:

```
<bean id="messageSource"
  class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basename" value="properties/locale/messages" />
</bean>
```

Pro českou lokalizaci je třeba v daném adresáři vytvořit soubor s názvem messages_cs_CZ.properties. Vždy je nutné vytvořit také defaultní soubor messages.properties.

Další nezbytnou konfigurací je InternalResourceViewResolver, který hledá jsp stránky pouze v uvedené složce. Navíc umožňuje kódování a možnost odkazovat se na jsp pouze jejich jménem a ne příponou [50]. Stačí tedy použít pouze název a spring sám vrátí název.jsp. InternalResourceViewResolver vypadá následovně:

```
<bean id="viewResolver"
  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="viewClass"
  value="org.springframework.web.servlet.view.JstlView"></property>
  <property name="prefix" value="/WEB-INF/jsp/"></property>
```

```
  <property name="suffix" value=".jsp"></property>
  <property name="contentType" value="text/html;charset=UTF-8"></property>
</bean>
```

Pro rozpoznání anotací v celé MVC architektuře je třeba přidat:

```
<mvc:annotation-driven />
```

Spring má obvykle problémy s načítáním statických souborů jako jsou JavaScript, css či obrázky. Řešením je tag resources [51], který explicitně namapuje složku s těmito soubory:

```
<mvc:resources mapping="/resources/**" location="/resources/" />
```

Pro připojení statického souboru poté stačí v jsp stránce zadat: resources/nazev.css. Samotné mapování http požadavků a vracení příslušných JSP stránek probíhá v kontroléru. Pokud ale není nutné provádět žádnou logiku či vracet data z databáze před voláním jsp stránky, stačí přidat tento tag [52]:

```
<mvc:view-controller path="/" view-name="kytara" />
```

Požadavek s url adresou kytara Spring přesměruje na kytara.jsp. Tento princip se dá také použít pro přesměrování na defaultní jsp stránku, pokud uživatel zadá neplatnou url adresu. Stačí do souboru web.xml přidat:

```
<error-page>
  <error-code>404</error-code>
  <location>/</location>
</error-page>
```

Každé neexistující url bude nahrazeno základním url /, které chytí dětský kontext a podle výše uvedeného tagu view-controller je vrácena stránka kytara.jsp.

8.2.3 Práce s databází

Spring je komplexní rámec nabízející mnoho různých variant řešení v rámci architektury MVC. Pro práci s databází Spring například podporuje technologii JDBC přes JdbcTemplate (knihovna org.springframework.jdbc.core.JdbcTemplate) nebo Hibernate

JPA (knihovna javax.persistence) [53]. Kapitola popíše a vysvětlí druhý přístup přes technologii Hibernate JPA. Tento přístup byl implementován v praktickém projektu.

8.2.3.1 Hibernate JPA přístup

Tento příklad by měl demonstrovat práci s databází pomocí Hibernate JPA. Nejprve je třeba vytvořit konfigurační soubor xml pro databázové spojení. Součástí této konfigurace bude nastavení dataSource [47]:

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
  <property name="driverClass">
    <value>${jdbc.driver.className}</value>
  </property>
  <property name="jdbcUrl">
    <value>${jdbc.url}</value>
  </property>
  <property name="user">
    <value>${jdbc.username}</value>
  </property>
  <property name="password">
    <value>${jdbc.password}</value>
  </property>
</bean>
```

Hodnoty pro jednotlivé atributy nalezne Spring v souboru properties, který byl definován v rodičovském kontextu. Dále je třeba vytvořit beanu entityManagerFactory, která do sebe zapouzdřuje již vytvořený dataSource [47]:

```
<bean id="jpaEntityFactory"
  class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="packagesToScan" value="cz.cr.sara.kytara.domain" />
  <property name="jpaVendorAdapter">
    <bean class="org.springframework.orm.jpa.vendor.Hibernate
    JpaVendorAdapter" />
  </property>
  <property name="jpaProperties">
```

```
    <props>
      <prop key="hibernate.hbm2ddl.auto">update</prop>
      <prop key="hibernate.dialect">${jdbc.hibernate.dialect}</prop>
      <prop key="hibernate.show_sql">false</prop>
      <prop key="hibernate.format_sql">true</prop>
      <prop key="hibernate.archive.autodetecion">class, hbm</prop>
      <prop key="connection.provider_class">org.hibernate.connection.C3P0
      ConnectionProvider
      </prop>
      <prop key="hibernate.c3p0.acquire_increment">1</prop>
      <prop key="hibernate.c3p0.min_size">5</prop>
      <prop key="hibernate.c3p0.max_size">20</prop>
      <prop key="hibernate.c3p0.timeout">300</prop>
      <prop key="hibernate.c3p0.max_statements">50</prop>
      <prop key="hibernate.c3p0.idle_test_period">3000</prop>
      <prop key="hibernate.c3p0.acquireRetryAttempts">1</prop>
      <prop key="hibernate.c3p0.acquireRetryDelay">250</prop>
    </props>
  </property>
</bean>
```

Beana definuje balíček, ve kterém budou repository (třídy pracující s databází). Dále je zde definován jpaVendorAdapter, což je adaptér propojující technologie Spring a Hibernate JPA [54].

Na takto vytvořený konfigurační soubor je nutné přidat referenci v souboru web.xml podobně jako reference na rodičovský kontext Springu:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
  /WEB-INF/conf/applicationContext.xml
  /WEB-INF/conf/db_config.xml
  </param-value>
</context-param>
```

Nyní, když je vytvořena entityManagerFactory, bude ukázán příklad, jak s touto entitou pracovat. Cílem bude zavolání metody getAllTony(), která zajistí vybrání všech objektů Tón z databáze. Nejprve musí vzniknout entita Ton:

```
@Entity
@Table(name = "ton")
public class Ton implements Serializable {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "id", nullable = false)
  private int id; // ID tonu

  @Column(name = "nazev", nullable = false)
  private String nazev; // Nazev tonu

  @OneToMany(mappedBy = "ton", cascade = CascadeType.ALL,
  fetch = FetchType.EAGER)
  @OrderBy(clause = ("struna ASC"))
  private List<Poloha> polohy; // Seznam poloh
--------getry, setry, konstruktor-------
}
```

Hibernate JPA pracuje pouze s objektem. Ten se musí pomocí anotací namapovat na konkrétní tabulku v databázi. Atributy se dále namapují na sloupce této tabulky. Samotný mechanismus převodu na objekty potom zajišťuje Hibernate JPA [47]. Nevytváří se tedy žádný RowMapper, jako v předchozím příkladu.
Třída, která přistupuje do databáze, bude mít atribut s entityManagerFactory:

```
@PersistenceUnit(unitName = "jpaEntityFactory")
private EntityManagerFactory emf;
```

Do atributu emf bude injektována beana z konfiguračního souboru podle id [47].
Pokud se přistupuje k databázi z více míst (více metod pracující s databází), musí se na každém takovém místě zajistit otevření a zavření spojení, transakční zpracování požadavku, commit či rollback. Dochází zde k opakování kódu a navíc může lehce nastat chyba. Stačí, když programátor zapomene uzavřít spojení do databáze a aplikace po

chvíli selže z důvodu vyčerpání celkového počtu současně otevřených spojení do databáze. Z tohoto důvodu vznikla abstraktní třída TransactionCallback, která tyto záležitosti řeší na jednom místě:

```
public abstract class TransactionCallback {
 protected EntityManager em;
 /**
  * Spusteni definovaneho zpracovani v DBS v transakci.
  * @param emf - tovarna entit pro praci s DBS.
  * @return Vrati vyhledana nebo informacni data.
  * @throws RuntimeException
  */
 public Object run(EntityManagerFactory emf) throws RuntimeException {
  if (emf == null) {
   throw new NullPointerException("Chybne vstupni parametry");
  }
  em = emf.createEntityManager();
  EntityTransaction tx = null;
  try {
   tx = em.getTransaction();
   tx.begin();
   Object o = runInTransaction();
   tx.commit();
   return o;
  }
  catch (Exception ex) {
   if (tx != null && tx.isActive()) {
    tx.rollback();
   }
   throw new RuntimeException(ex);
  }
  finally {
   em.close();
  }
 }
 /**
```

```
 * Definovane zpracovani v DBS.
 * @return Vrati vyhledana nebo informacni data.
 */
 public abstract Object runInTransaction();
}
```

Třída navíc vytváří objekt entityManager, přes který se už přímo přistupuje do databáze. Nyní zbývá jen implementovat metodu getAllTony(), která bude přes TransactionCallback přistupovat do databáze:

```
public List<Ton> getAllTony() {
 Object data = new TransactionCallback() {
  @Override
  public Object runInTransaction() {
   List<Ton> tony = em.createNamedQuery("Ton.all",
     Ton.class).getResultList();
   return tony;
  }
 }.run(emf);

 List<Ton> tony = (data != null)
   ? (List<Ton>) data
   : null;
 if (tony == null) {
  throw new EntityNotFoundException("Nebyl vyhledan zadny ton");
 }
 return tony;
}
```

Opět je třeba zdůraznit, že se pracuje s objektem Ton, nikoli s tabulkou v databázi ton. Výsledkem dotazu je list objektů Ton.

Pro úplnost zbývá objasnit, co to jsou NamedQueries a jak se používají. Jedná se o SQL dotazy, ve kterých se můžou objevit parametry [55]. Běžně se definují pomocí anotací v entitě, se kterou souvisejí. Například v entitě ***Ton*** jsou následující NamedQueries:

```
@NamedQueries({ @NamedQuery(name = "Ton.tonById", query = "FROM Ton t WHERE t.id = :id"),
```

```
  @NamedQuery(name = "Ton.idTonByNazev", query = "SELECT t.id FROM Ton t WHERE t.nazev = :nazev"),
  @NamedQuery(name = "Ton.allNazev", query = "SELECT t.nazev FROM Ton t ORDER BY t.id"),
  @NamedQuery(name = "Ton.all", query = "FROM Ton t ORDER BY t.id") })
```

Ve třídě repository se stačí podle jména odkázat na NamedQuery a případně doplnit parametry.

8.2.4 Typy anotací komponent

Spring umožňuje skenování bean přímo v Java kódu, jak bylo popsáno v konfiguraci souboru web.xml. Spring poté injektuje vybranou beanu do atributu označeného anotací @Autowired. Spring skenuje pouze takové beany (třídy), nad kterými je jedna z anotací z balíčku [56]:

```
org.springframework.stereotype
```

Tato kapitola se zaměřuje na typy jednotlivých anotací. Jaké jsou mezi nimi rozdíly a kdy je vhodné použít kterou anotaci.

8.2.4.1 Představení anotací

Pro automatické skenování bean přímo v JAVA kódu používá Spring následující anotace [56]:

```
@Component, @Repository, @Service, @Controller
```

@Component je základním typem anotace. Další anotace jsou speciálním typem anotace @Component a měly by se používat na specifickém místě v aplikaci:

```
@Repository – pro třídy přistupující do databáze
@Service – pro třídy v roli managera
@Controller – pro controllery
```

V rozsáhlých aplikacích vznikají jednotlivé, na sobě nezávislé, vrstvy – datová, servisní, prezentační. Pokud má být daná třída použita pro automatické skenování, měla by být použita právě ta anotace, která do dané vrstvy patří. Obecná anotace @Component se

používá u tříd, které plní jinou roli než ostatní tři typy tříd. Například třída řešící logické výpočty.

Mezi jednotlivými anotacemi není v podstatě žádný rozdíl. Spring je všechny bere jako anotaci @Component. Nicméně každá anotace by se měla používat na správném místě i z důvodu, že Spring v budoucí verzi může přidat další funkcionalitu k jednotlivým typům [57].

8.2.4.2 Injektování podle jména

Při injektování hledá Spring beanu defaultně podle jména beany:

```
@Service
public class AkordManagerImpl implements AkordManager {
  // ...
}
```

Pro injektování této třídy stačí v jiné třídě napsat:

```
@Autowired
private AkordManager akordManagerImpl;
```

Spring ale dovoluje explicitně nastavit jméno beany přímo v anotaci:

```
@Service("akordManager")
public class AkordManagerImpl implements AkordManager {
  // ...
}
```

Pro automatickou injektáž je pak nutné napsat:

```
@Autowired
private AkordManager akordManager;
```

Stejná pravidla platí pro všechny typy anotací bean.

8.2.5 Formuláře ve view

Jak už bylo výše zmíněno, Spring je součástí celé MVC architektury. Co se týče view vrstvy, Spring umožňuje její nezávislost na zbytku MVC architektury[58]. Spring využívá šablonu (template) JSP pro dynamické HTML stránky. Dynamická data získaná pomocí

Springu jsou podle JSP šablony na serveru převedena do formátu HTML a poslána na klienta, který se stará pouze o jejich zobrazení.

Jedna z možností, kterou Spring rámec ve view vrstvě nabízí, jsou formuláře v JSP stránce. Aby mohl Spring tyto formuláře v JSP šabloně používat, musí do každé JSP přidat odkaz na svou knihovnu [59]:

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
```

Pod daným prefixem odkazujícím na knihovnu potom Spring používá vlastní formulářové tagy:

```
<form:input type="hidden" path="ton" id="ton" value="${akord.ton}" />
```

8.2.6 Kontrolér a Spring

Aby byla třída Spring kontrolérem, musí mít anotaci @Controller z balíčku org.springframework.stereotype.Controller [60]:

```
@Controller
public class AkordFormController {
  //...
}
```

Vhodný způsob, jak namapovat kontrolér na URL adresu, je použít následující anotaci před názvem třídy:

```
@RequestMapping("/akordy")
```

Třída AkordFormController bude zavolána, pokud v URL za názvem projektu bude hodnota: /akordy [60]. Třída kontroléru se sice zavolá, ale není v něm ještě žádná metoda pro zpracování HTTP požadavku. Tato metoda může vypadat například takto:

```
protected ModelAndView akordyFind(@ModelAttribute("akordInfo") AkordInfo akordInfo){
  //logika
  return new ModelAndView("/akordy",
    "akord",
    akordView);
}
```

Protože obvykle taková metoda přesměruje volání na JSP stránku, která vrací nějaká data, používá se jako návratový typ ModelAndView z balíčku [61]:

```
org.springframework.web.servlet
```

První parametr konstruktoru ModelAndView znamená, kterou JSP stránku metoda vrací, druhý parametr jméno objektu, na který se odkazuje JSP stránka a třetí parametr samotný objekt.

Dalším důležitým rysem metody je anotace @ModelAttribute z balíčku [62]:

```
org.springframework.web.bind.annotation.ModelAttribute
```

Jedná se o objekt, který má být naplněn v okamžiku zavolání metody, a se kterým metoda dále pracuje. Toto bude rozebráno v následující kapitole Interakce mezi kontrolérem a JSP.

Předchozí metoda neobsahuje žádné mapování. Pokud metoda v kontroléru není namapována, chová se jako defaultní metoda. Defaultní metoda je zavolána pokaždé, když není nalezena v daném kontroléru metoda jiná. Taková defaultní metoda bude zavolána na následující URL adresy:

```
název_projektu/akordy/
název_projektu/akordy/necoCoNeniNamapovano
```

Namapování metody na dané URL se provede (podobně jako u kontroléru) následující anotací nad názvem metody:

```
@RequestMapping(value = { "/akordy" }, method = { RequestMethod.POST,
  RequestMethod.GET })
```

Metoda bude zavolána, pokud adresa URL bude mít hodnotu:

```
název_projektu/akordy/akordy
```

Dále anotace říká, že metoda bude reagovat pouze na HTTP požadavky typu POST a GET.

8.2.7 Interakce mezi kontrolérem a JSP

V této kapitole bude vysvětleno, jak fungují Spring formuláře v JSP stránce a jak tyto formuláře spolupracují s kontroléry. Vše začíná tím, že je zavolána JSP stránka s formulářem pomocí metody v kontroléru:

```
protected ModelAndView akordy () {
 // Získání objektu AkordView
 return new ModelAndView("/akordy/akordy",
  "akord",
  akordView);
}
```

Do JSP stránky je poslán objekt AkordView, který má následující strukturu:

```
public class AkordView {
 private final String nazev; // Nazev akordu
 private final String ton; // Ton
 private final String typAkordu; // Typ akordu

 private final List<String> tony; // Seznam tonu
 private final List<AkordView.Hmat> hmaty; // Seznam hmatu
 private final VyberAkordu vyberAkordu; // Vyber akordu
//...
}
```

Stránka akordy.jsp obsahuje následující Spring formulář:

```
<form:form method="post" commandName="akordInfo"
  action="/kytara/akordy/akordy" class="form form-horizontal"
  role="form">
<form:input type="hidden" path="ton" id="ton" value="${akord.ton}" />
</form:form>
```

Po odeslání formuláře bude zpracována URL adresa s hodnotou:

```
/kytara/akordy/akordy
```

Bude tedy volán kontrolér namapovaný na hodnotu: akordy. V něm se bude volat následující metoda:

```
@RequestMapping(value = { "/akordy" }, method = { RequestMethod.POST })
protected ModelAndView akordyFind(@ModelAttribute("akordInfo") AkordInfo
akordInfo) {
  //...
```

```
}
```

CommandName se odkazuje na anotaci @ModelAttribute v metodě kontroléru a říká, že po potvrzení formuláře se atribut označený touto anotací automaticky naplní [63]. Přesněji se vezme hodnota path v každém tagu input, která naplní stejnojmenný atribut objektu. Například pokud má path hodnotu ton, zavolá se metoda setTon(Ton ton).
V celé stránce je možné získávat data z objektu, který byl do JSP stránky předán v kontroléru. Tento princip je vidět u atributu value [63]:

```
value="${akord.ton}"
```

Volá se objekt AkordView podle jména akord. Z tohoto objektu je následně zavolána metoda getTon().

8.3 Twitter Bootstrap

8.3.1 Úvod do Twitter Bootstrap

Twitter Bootstrap je moderní front-end rámec pro rychlejší vývoj webových aplikací [35]. Vyžaduje alespoň základní znalost CSS a HTML. Podporuje také JavaScript a JQuery.
V moderních webových aplikacích se stále dokola používají základní frontendové komponenty (tabulky, formuláře, tlačítka). Poměrně pracně se upravuje vzhled a umístění těchto komponent pomocí CSS. Bootstrap přichází s myšlenkou jednoduchého generování těchto komponent. Stačí pouze přidat odkaz na CSS soubor vytvořený Bootstrapem. K tomu přidává rámec i další komponenty jako například: Navigation, Pagination, Carousel nebo Headers [35]. Bootstrap tak umožňuje zrychlení vývoje webových aplikací pomocí předem připravených souborů.
Obrovskou předností Twitter Bootstrap je možnost responzivního designu. Bootstrap používá předdefinované šablony, které umožní snadnou tvorbu layoutů, založených na 12 sloupcích. Rámec pozná podle typu výstupního zařízení, jak velkou vykreslovací plochu může použít a na různě velké plochy aplikuje rozdílné CSS třídy. Uživatel tak může rozlišit vzhled na extra malých, malých, středních a velkých zařízeních [64].
Používání předpřipravených šablon způsobuje, že mnohé projekty používající Twitter Bootstrap vypadají podobně. Tato vlastnost bývá rámci často vytýkána. Na druhou stranu vývojářům nic nebrání v tom předefinovat určité CSS třídy podle vlastního uvážení a tím docílit odlišného vzhledu svých komponent.

8.3.2 Stažení Twitter Bootstrap

Nejběžnější formou, jak získat Bootstrap, je stáhnout zkompilovanou a zmenšenou verzi Bootstrapu. Po rozbalení obsahuje tato varianta tři složky [65]:

css – soubory s kaskádními styly,
js – soubory s JavaScriptem,
fonts – obsahuje glyphicony (obrázky pro použití v html stránce).

Běžnou praxí je vytvořit si například složku resources a do ní vložit verzi Bootstrapu tak, jak byla stažena. Dále stačí vzít soubory mající v názvu slovo min a importovat je do každé JSP stránky:

```
<link rel="stylesheet" type="text/css"
  href="${pageContext.request.contextPath}/resources/css/bootstrap.min.css" />
<link rel="stylesheet" type="text/css"
  href="${pageContext.request.contextPath}/resources/css/bootstrap-theme.min.css" />
<script
  src="${pageContext.request.contextPath}/resources/js/bootstrap/
  bootstrap.min.js">
</script>
```

Pokud je dodržena základní struktura, Bootstrap sám nalezne cestu ke svým fontům. Nyní je vše připraveno, aby se Bootstrap mohl použít.

8.3.3 Použití Twitter Bootstrap

Bootstrap se používá velmi snadno. K HTML prvku stačí přidat atribut class, který bude mít hodnotu z knihovny Bootstrap:

```
<button type="button" class="btn btn-default">
```

Takto se HTML prvek upraví pomocí daného kaskádního stylu. Na oficiálních stránkách Bootstrapu [66] je k nalezení velké množství příkladů a návodů pracujících nejen s kaskádními styly, ale také s JavaScriptem. Pro základní představu zde bude ukázáno několik příkladů, jak naformátovat vybrané HTML elementy pomocí kaskádních stylů.
Prvním příkladem Bootstrapu je založení tabulky přes celou stránku (bude zabírat všech 12 sloupců obrazovky) se speciálním stylem:

```
<div class="table-responsive col-sm-12">
  <table class="table table-striped">
```

Jak je vidět, odstavec s tabulkou je definovaný pro malá zařízení (col-sm). Bootstrap funguje tak, že všechna větší zařízení ponechá stejná (tedy na 12 sloupců). Zařízení, která jsou menší, upraví tak, že co se na obrazovku nevejde, přidá na další řádek. Dále automaticky vytvoří rolovací lištu. Z hlediska programátora stačí definovat jeden typ zařízení, pro který má být aplikace primárně určena. Pokud je cílem vytvořit dokonalé vykreslení pro všechna zařízení, je možné vzájemně kombinovat více CSS atributů u elementu.

Stejně jednoduše je možné pomocí Bootstrapu přidat do tabulky sloupeček se zabarvením do zelena pomocí třídy success:

```
<tr class="success">
  <th>Polohy akordu</th>
</tr>
```

Odstavec s výstražným textem se vytvoří pomocí třídy alert alert-warning:

```
<div class="alert alert-warning" id="polohy">
```

Do JSP stránky se dají načíst také glyphicony. Opět velmi snadno:

```
<span class="glyphicon glyphicon-forward"></span>
```

Posledním příkladem bude ukázka navigačního menu, které Bootstrap opět dokáže vytvářet v závislosti na typu zařízení. Vytvoří se odstavec s vnořeným seznamem s odkazy na další JSP stránky:

```
<div class="collapse navbar-collapse">
  <ul class="nav navbar-nav">
    <li><a href="akordy/akordy"
      title="Kytarové akordy">Akordy</a></li>
    <li><a href="triady/triady"
      title="Výběr triád">Triády</a></li>
    <li><a href="tony/tony"
      title="Výběr tónů">Tóny</a></li>
    <li><a href="findAkord/tony"
```

```
      title="Vyhledání akordu">Výběr akordu</a></li>
  </ul>
</div>
```

Pokud se zmenší šířka zobrazovací plochy natolik, že se odkazy nevejdou vedle sebe, vytvoří Bootstrap rozbalovací seznam se všemi odkazy. Pokud jsou potřeba rolovací lišty, přidá je do seznamu.

Jak je vidět z příkladů, uživatel rámce Bootstrap pouze využívá předpřipravené styly. Není však problém některé styly přepsat vlastním CSS souborem.

8.4 HTML5 canvas

8.4.1 Úvod do HTML5 canvas

Specifikace pro HTML5 přináší celou řadu nových vlastností. Snad nejradikálnější z nich je element canvas, který umožňuje vykreslování grafiky pomocí JavaScriptu do HTML stránky [33]. Pro vykreslování je nutné pro každý canvas element inicializovat kontext. Kontext může být buď 2D (podporují běžné prohlížeče) nebo 3D (vzniká pouze experiment v Opeře) [33].

8.4.2 Základní použití

Pro práci s HTML5 canvasem je nutná alespoň základní znalost JavaScriptu. Nejprve je třeba založit samotný element canvas s určitou šířkou a výškou:

```
<canvas id="myCanvas" width="1000" height="726">
    Your browser does not support the HTML5 canvas tag.
</canvas>
```

Uvnitř je text, který se zobrazí v případě, že prohlížeč nebude canvas podporovat. Ke kreslení do elementu bude použit JavaScript. Nejprve je třeba získat referenci na daný canvas pomocí getElementById. Dále se inicializuje context na tomto elementu [33]. Po vytvoření kontextu je vše připraveno k vykreslování grafiky.

Následující příklad vykreslí červený obdélník uvnitř elementu canvas:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle ="red";
```

```
ctx.fillRect(20, 20, 100, 200);
```

8.4.3 Vybrané funkce

V této kapitole budou ukázány vybrané možnosti práce s HTML5 canvasem, které byly využity v praktickém projektu.

8.4.3.1 Kreslení obrazců

Canvas disponuje funkcemi na vykreslování obrazců. Již bylo ukázáno, jak nakreslit obdélník pomocí funkce fillRect(x, y, sirka, vyska). Podobně se vykresluje také kružnice:

```
ctx.arc (x, y, poloměr, začátek vykreslování v radiánech,
  konec vykreslování v radiánech);
```

Obrazcům se dají nastavit styly. Existují dva druhy stylů:

```
strokeStyle – definuje styl obrysu obrazce
fillStyle – definuje styl vyplněné oblasti
```

Následující příklad demonstruje nakreslení kruhu pomocí obou těchto stylů:

```
ctx.arc(xStredPrazdnehoTonu,
  yStredPrazdnehoTonu, polomerTonuSPrazdnouStrunou, 0, 2 * Math.PI);
ctx.fillStyle = barvaPrazdnaStruna;
ctx.fill();
ctx.lineWidth = prazdnyTonSirkaOhraniceni;
ctx.strokeStyle = barvaPrazdnaStrunaOhraniceni;
ctx.stroke();
```

Nejprve se vykreslí kružnice, která se následně vyplní určenou barvou. Dále se definuje šířka čáry a barva ohraničení. Nakonec se vykreslí ohraničení.

8.4.3.2 Kreslení vlastních tvarů

Pomocí Canvasu je možné nakreslit jakýkoliv tvar. Tvary se kreslí pomocí jednotlivých čar, které se skládají za sebe. Následující kód vysvětluje, jak se čáry kreslí:

```
ctx.beginPath();
ctx.moveTo(x, y); //čára od bodu A
ctx.lineTo(x1, y1); //čára do bodu B
```

```
ctx.lineTo(x2, y2); //další čára od B do C
ctx.closePath();
```

Kreslení čar se může kombinovat s jednotlivými styly popsanými v předešlé kapitole. Tento postup demonstruje následující funkce, která má za úkol nakreslit křížek s určitými styly:

```
function vykresliKrizek(poradi) {
  ctx.beginPath();
  ctx.shadowBlur = stinyTloustka;
  ctx.shadowColor = stinyBarvaKrizek;
  ctx.lineWidth = 7;
  ctx.strokeStyle = barvaKrizek;
  ctx.moveTo(stredBunky - rozpetiKrize + poradi * sirkaBunky + xHmatniku,
  yKrizekOd);
  ctx.lineTo(stredBunky + rozpetiKrize + poradi * sirkaBunky + xHmatniku,
  yKrizekDo);

  ctx.moveTo(stredBunky + rozpetiKrize + poradi * sirkaBunky + xHmatniku,
  yKrizekOd);
  ctx.lineTo(stredBunky - rozpetiKrize + poradi * sirkaBunky + xHmatniku, yKrizekDo);
  ctx.stroke();
  ctx.closePath();
}
```

Nejprve se nakreslí dvě čáry, z nichž vznikne tvar kříže. Následně dojde k vykreslení obrysu s nastavenými styly. Jedním z těchto stylů je také stínování, u něhož se nastavuje šířka a barva stínu.

8.4.3.3 Obnovení kontextu

V předešlých kapitolách bylo ukázáno, jak lze kontextu nastavit určité vlastnosti. Často nastává situace, kdy v různých funkcích jsou potřeba stejné vlastnosti. Aby vývojář nemusel psát stále ty samé vlastnosti na různých místech, byla vymyšlena funkce obnovení kontextu. Princip spočívá v uložení daného kontextu kdekoli v kódu a jeho následného načtení na jiném místě:

```
ctx.save(); //uložení kontextu
```

```
ctx.restore(); //načtení uloženého kontextu
```

8.5 JQuery

8.5.1 Úvod do JQuery

JQuery je knihovna JavaScriptu, která ulehčuje práci s JavaScriptem tím, že se volají již předpřipravené funkce z API JQuery. S JQuery se tedy dá napsat to samé s minimálním množstvím kódu. Další projekty založené na JQuery jsou JQuery User Interface, QUnit unit testing a JQuery mobile [37].

8.5.2 JQuery použití

Pro používání JQuery v JSP stránce stačí z oficiálních stránek [37] stáhnout nejnovější knihovnu JQuery a následně ji přidat do JSP stránky:

```
<script
  src="${pageContext.request.contextPath}/resources/js/jquery/jquery2.1.1.js">
</script>
```

Nyní již může být knihovna volána v JSP stránce. Jednou z mnoha funkcí, které JQuery nabízí, je funkce each, která projde všechny prvky v kolekci a s každým prvkem pracuje. Nativní funkce implementovaná přímo v JavaScriptu je více než 10x rychlejší [67], na druhé straně je funkce each úspornější a lépe čitelná:

```
function enableTon() {
  $("#tony .buttons").each(function() {
    $(this).attr("class", "buttons buttons_upgrade");
  });
}
```

JQuery nabízí celou řadu funkcí nahrazujících klasické funkce v JavaScriptu. Mimoto podporuje problematiku zvanou AJAX, která bude rozebrána v následující kapitole.

8.6 AJAX

8.6.1 Úvod do AJAXU

AJAX (asynchronous JavaScript and XML) je technologie umožňující výměnu dat se serverem a aktualizaci části webové stránky bez nutnosti obnovení celé stránky [68].

JSON je formát pro výměnu dat, který umí přenášet objekty nebo kolekce objektů. Používá přitom univerzální datové struktury. Na nich je založen na jazyce nezávislý výměnný formát [69].

Princip technologie AJAX by se dal popsat následovně [70]:

Prohlížeč vytvoří a pošle Http požadavek.
Server zpracuje požadavek, vytvoří odpověď a pošle data zpět do prohlížeče.
Prohlížeč pomocí JavaScriptu zpracuje data a pozmění obsah stránky.

8.6.2 Použití AJAXU

V této kapitole bude ukázán příklad použití technologie AJAX v rámci Spring rámce. Protože se budou přenášet datové objekty, je třeba nejprve přidat do pom.xml některou knihovnu JSON:

```
<!-- Google GSON -->
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.3</version>
</dependency>
```

8.6.2.1 Klientská část

Pokud nastane určitá událost, bude zavolána metoda s funkcí AJAX. Tato funkce je podporována knihovnou JQuery a vypadá následovně:

```
$.ajax({
  url : 'findAkordy',
  type : 'POST',
  data : {
    tony : tony,    //parametr typu String
  },
  dataType : 'text',
  success : function(data) {  //při úspěšném vrácení dat
  //vlastní zpracování dat ze serveru
  },
  error : function(e) { //při chybě
    alert('CHYBA: ' + e.status);
```

```
  }
});
```

Atribut url specifikuje, která metoda v kontroléru se zavolá. Jelikož se bude posílat parametr typu String, nastaví se datový typ na hodnotu text.

8.6.2.2 Server

Jakmile se odešle tento požadavek, zavolá se následující metoda v kontroléru:

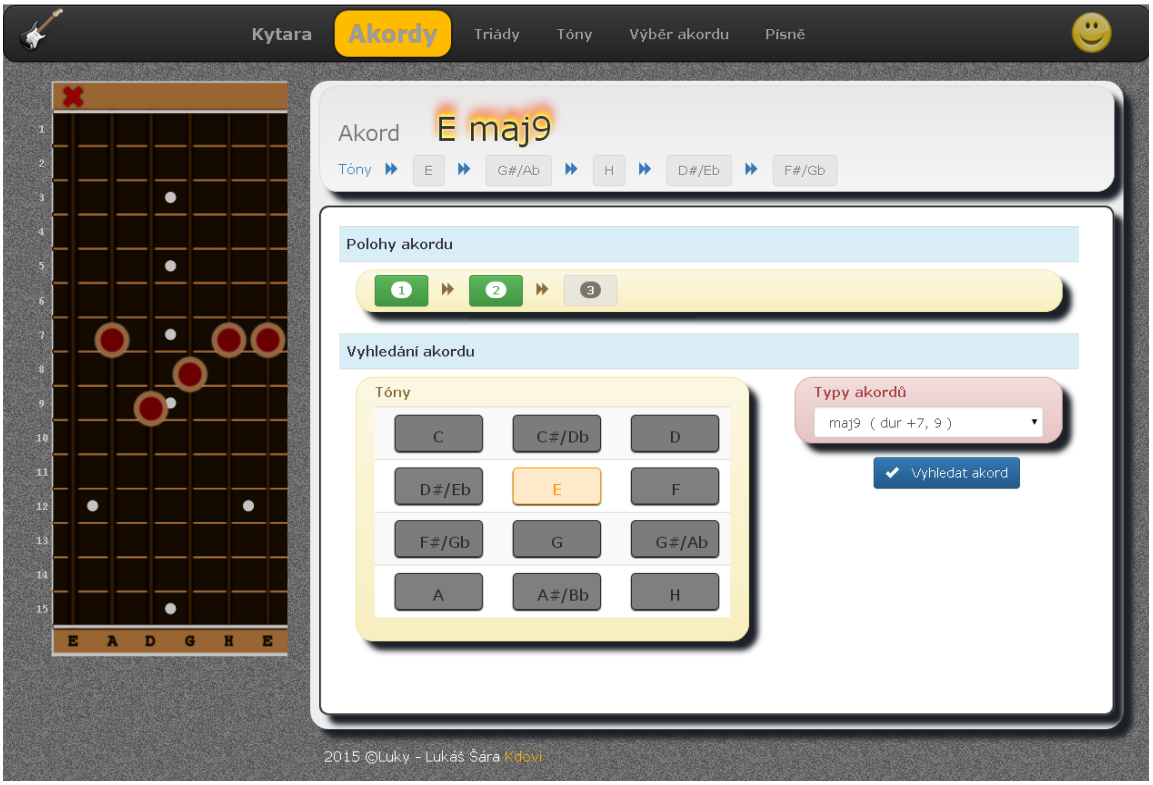
```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@RequestMapping(value = { "/findAkordy" }, method = RequestMethod.POST)
 protected @ResponseBody
   List<AkordInfo> akordFind(@RequestBody String tony) {
    List<AkordInfo> akordy = //naplnění kolekce objektů podle parametru tony
    return akordy;
}
```

Do parametru označeného anotací @RequestBody se vloží hodnota posílaného parametru. Podle tohoto parametru dojde k naplnění kolekce objektů AkordInfo. Aby Spring vrátil kolekci jako objekt, je nutné k návratovému typu metody přidat anotaci @ResponseBody. Spring si sám zavolá potřebné metody a objekt pošle zpět. V případě úspěšného poslání dat zpět na klientskou část dojde ke zpracování těchto dat a aktualizaci JSP stránky.

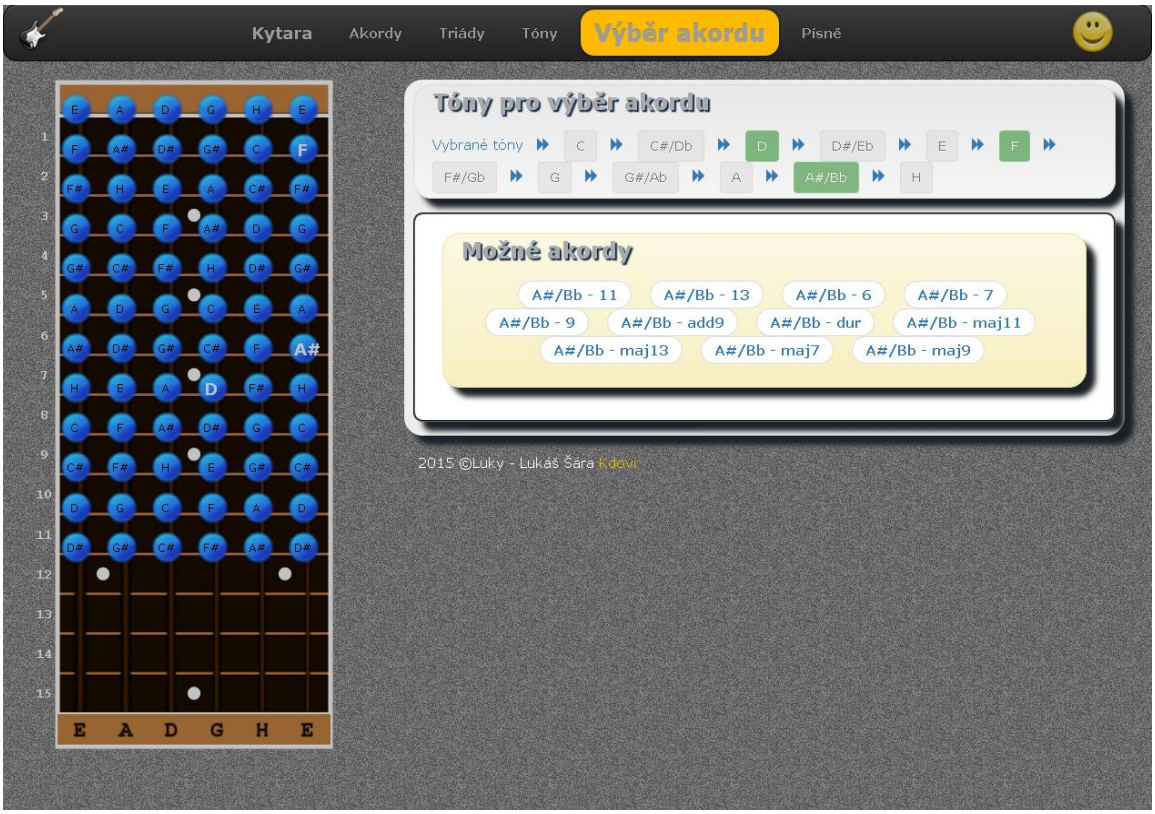
9 Ukázky výstupu

Praktický projekt byl úspěšně implementován. Veškeré ikony použité v projektu byly vytvořeny autorem nebo převzaty jako volně dostupné ze zdroje [71]. Kapitola znázorní ukázky výstupu pro vybrané partie aplikace. Konkrétně se bude jednat o tři ukázky. První z nich se týká zobrazení akordu podle tónu a typu akordu. Tato ukázka je znázorněna na obrázku číslo 12.



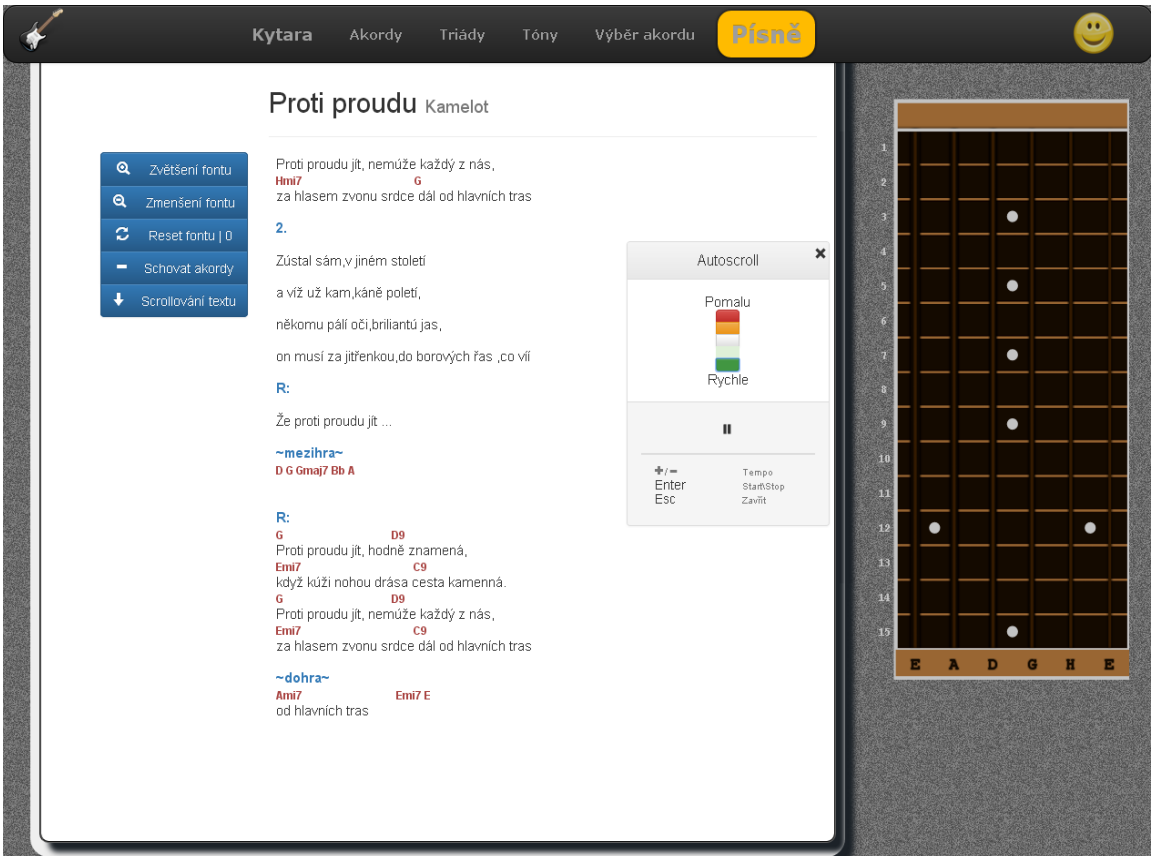
Obr. 12 Ukázka výstupu: zobrazení akordu
Zdroj: autor, vlastní zpracování

Obrázek číslo 13 znázorňuje druhou ukázku, která se týká výběru všech možných akordů podle zadaných tónů.



Obr. 13 Ukázka výstupu: výběr akordů podle tónů
Zdroj: autor, vlastní zpracování

Poslední ukázka je znázorněna na obrázku číslo 14. Jedná se o ukázku písně při automatickém rolování stránky. Rolovací menu se nachází v pravé polovině stránky.



Obr. 14 Ukázka výstupu: píseň
Zdroj: autor, vlastní zpracování

10 Shrnutí výsledků

Podařilo se implementovat dynamický web pro podporu hry na kytaru pomocí popsaných technologií. Aplikace slouží jako pomocník pro kytaristy specializující se na akordovou hru. Aplikace nabízí vyhledání a zobrazení akordových hmatů podle uživatelova výběru. Uživatel si navíc u každého akordu může projíždět jeho jednotlivé polohy.

Podobně aplikace nabízí zobrazování triád (akordů obsahujících tři tóny) a jednotlivých tónů na pražci. Uživatel si dále může zadat konkrétní tóny a aplikace mu zobrazí akordy, které tyto tóny obsahují.

Poslední funkcí webové aplikace jsou písně. Aplikace obsahuje vybrané písně i s akordy, které si uživatel může zobrazit a zahrát. Pokud klikne na akord, zobrazí se mu hmat tohoto akordu. Aplikace umožňuje také automatické rolování písní podle zvolené rychlosti.

Dále se podařilo popsat technologie použité v praktickém projektu a přiblížit je uživateli. Technologie obsahují navíc příklady převzaté z praktického projektu.

10.1 Budoucí uplatnění

Pro praktický projekt popisovaný v této práci by bylo jistě vhodné, kdyby nesloužil pouze k bakalářské práci, ale žil si vlastním životem – uplatnil se v praxi. Kapitola se pokusí stručně nastínit možné budoucí využití projektu v praxi.

Dynamický web poslouží spíše začínajícím kytaristům, kteří budou potřebovat zobrazit hmat libovolného akordu či tónu. Aplikace by proto mohla sloužit jako doplňující modul internetových kytarových prodejen (například kytary.cz [72], audiotek.cz [73] či music-city.cz [74]). Zisk z prodeje aplikace internetovým prodejcům by zajistil příležitost pro další rozvoj tohoto portálu. Z hlediska spotřebitelského chování by měl prodejce oproti ostatním výhodu ve fázi „hledání informací“ [75], neboť ho kytaristé budou lépe znát a neopomenou se při výběru kytary podívat na jeho internetový obchod. V důsledku toho by prodejce mohl očekávat pozitivní poptávkový šok, který by vedl k expanzi poptávky po kvalitnějších kytarách.

Další možností budoucího uplatnění by mohlo být provozovat aplikaci na vlastní náklady a prodávat internetovým prodejcům reklamu. Zisk z prodeje reklamy by měl opět zajistit finanční prostředky na další rozvoj portálu. Prodejce by díky reklamě mohl očekávat větší poptávku zejména po kvalitnějších kytarách.

11 Závěry a doporučení

Práce popisuje vývoj dynamického webu pro podporu výuky hry na kytaru. Čtenáři dostatečně přibližuje proces vývoje od požadavků na software přes analýzu a návrh partií výuky až po použité technologie.

Práce popisuje jednotlivé technologie. Jsou to technologie Apache Maven, Spring rámec, Twitter Bootstrap, HTML5 canvas, JQuery, AJAX a Stripes. Současně s pomocí názorných příkladů podrobně vysvětluje možnosti použití těchto technologií. Po důkladném nastudování by měl být čtenář schopen umět tyto technologie použít.

Velmi přínosnou částí této práce je bezesporu samotná aplikace nabízející funkce, které začínající kytarista specializující se na akordovou hru jistě využije.

Dále je velice zajímavý popis technologie AJAX v prostředí Spring rámce. Je zde podrobně nastíněn princip, na kterém AJAX funguje i s názorným příkladem využívajícím anotace @ResponseBody ve Spring kontroléru.

Bohužel v práci již nezbylo místo na důkladnější vysvětlení procesu tvorby akordů, které by napomohlo lepšímu pochopení odlišností jednotlivých akordů uvedených v aplikaci. Čtenář byl alespoň seznámen se základní teorií a nasměrován k dalším zdrojům.

Práce může pomoci středně pokročilým i začínajícím vývojářům nad platformou JAVA, jelikož jednotlivé postupy jsou podrobně vysvětleny. Těm zkušenějším přináší zajímavou možnost srovnání.

První přílohou práce je přiložený kompaktní disk (CD) se zdrojovými kódy a informacemi ohledně spuštění aplikace. Disk je nedílnou součástí práce. Druhou přílohu tvoří oskenované zadání bakalářské práce.

12 Seznam použité literatury

[1] Karel Čaloud. Chromatická stupnice: Kurz harmonie bez not [online]. [cit. 2015-01-30]. Dostupné z: http://www.webhouse.cz/kurz-harmonie/chromaticka_stupnice.htm

[2] http://fyzika.fce.vutbr.cz/file/martinek/frekvanalzvuku.pdf [online]. [cit. 2015-02-09]. Dostupné z: http://fyzika.fce.vutbr.cz/file/martinek/frekvanalzvuku.pdf

[3] Karel Čaloud. Akordy: Kurz harmonie bez not [online]. [cit. 2015-01-15]. Dostupné z: http://webhouse.cz/kurz-harmonie/akordy.htm

[4] INTERVALY – PRAKTICKÝ PŘEHLED | ZUŠ V.Petrželky [online]. [cit. 2015-01-30]. Dostupné z: http://www.zusvpetrzelky.cz/materialy-ke-stazeni?id=con504

[5] Dušan Janovský. Ladění kytary [online]. [cit. 2015-01-30]. Dostupné z: http://dusan.pc-slany.cz/hudba/kytara_ladeni.htm

[6] Petr Krumphanzl. Kytarová škola – díl 3 [online]. [cit. 2015-01-30]. Dostupné z: http://www.folktime.cz/serialy/kytarova-skola-dil-3.html

[7] OMG UML [online]. [cit. 2015-02-05]. Dostupné z: http://www.omg.org/gettingstarted/what_is_uml.htm

[8] UML - Use Case Diagrams [online]. [cit. 2015-02-11]. Dostupné z: http://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

[9] Bredemeyer Consulting. http://www.bredemeyer.com/pdf_files/functreq.pdf [online]. [cit. 2015-02-05]. Dostupné z: http://www.bredemeyer.com/pdf_files/functreq.pdf

[10] TechMasala - Technology Spice Rack >> Non Functional Requirements (NFRs) [online]. [cit. 2015-02-05]. Dostupné z: http://www.techmasala.com/2006/05/23/non-functional-requirements-nfrs/

[11] Scott W. Ambler. Business Rules: An Agile Introduction [online]. [cit. 2015-02-05]. Dostupné z: http://agilemodeling.com/artifacts/businessRule.htm

[12] Scott W. Ambler. UML 2 Use Case Diagrams: An Agile Introduction [online]. [cit. 2015-02-05]. Dostupné z: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

[13] UML use case include relationship shows that behavior of the included use case is inserted into the behavior of the including use case. [online]. [cit. 2015-02-05]. Dostupné z: http://www.uml-diagrams.org/use-case-include.html

[14] Gábina Janků. Arlow Jim, Neustadt Ila: UML 2 a unifikovaný proces vývoje aplikací | PŘEČTENO. COM [online]. [cit. 2015-02-09]. Dostupné z: http://www.precteno.com/arlow-jim-neustadt-ila-uml-2-a-unifikovany-proces-vyvoje-aplikaci-1577

[15] [Modeling your domain models in UML | Capgemini Worldwide [online]. [cit. 2015-02-09]. Dostupné z: http://www.capgemini.com/blog/capping-it-off/2011/05/modeling-your-domain-models-in-uml

[16] Class charakteristic [online]. [cit. 2015-02-09]. Dostupné z: http://skat.ihmc.us/rid=1226963868250_1894953268_69941/Class%20charakteristic.cmap

[17] Petra Rejnková. Příklady použití diagramů UML 2.0 [online]. [cit. 2015-02-09]. Dostupné z: http://uml.czweb.org/diagram_trid.htm

[18] Čápka, David. MVC archi}[tektura [online]. [cit. 2015-01-02]. Dostupné z: http://www.itnetwork.cz/mvc-architektura-navrhovy-vzor

[19] TechTerms.com. Dynamic Website Definition [online]. [cit. 2015-02-09]. Dostupné z: http://techterms.com/definition/dynamicwebsite

[20] Vojtěch Hordějčuk. ing. Vojtěch Hordějčuk – Maven [online]. [cit. 2015-01-15]. Dostupné z: http://voho.cz/wiki/maven/

[21] The Apache Software Foundation. Maven – Introduction to the Dependency Mechanism [online]. [cit. 2015-01-02]. Dostupné z: http://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html

[22] Adam Bien. Maven vs. Ant : Adam Bien's Weblog [online]. [cit. 2015-01-23]. Dostupné z: http://www.adam-bien.com/roller/abien/entry/maven_vs_ant

[23] The Apache Software Foundation. Maven – Welcome to Apache Maven [online]. [cit. 2015-01-15]. Dostupné z: http://maven.apache.org/

[24] Skills Matter Ltd. Skills Matter [online]. [cit. 2015-01-02]. Dostupné z: https://skillsmatter.com/legacy_profile/rod-johnson#overview

[25] Roman Pichlík. Spring Framework – představení J2EE lightweight kontejneru [online]. [cit. 2015-01-23]. Dostupné z: http://interval.cz/clanky/spring-framework-predstaveni-j2ee-lightweight-kontejneru/

[26] Eugen Paraschiv. Spring 4 and JPA with Hibernate | Baeldung [online]. [cit. 2015-01-23]. Dostupné z: http://www.baeldung.com/2011/12/13/the-persistence-layer-with-spring-3-1-and-jpa/

[27] Rick Grashel. Home – Stripes Wiki - Stripes Framework Wiki [online]. [cit. 2015-01-04]. Dostupné z: https://stripesframework.atlassian.net/wiki/display/STRIPES/Home

[28] Stripes – FI WIKI [online]. [cit. 2015-01-04]. Dostupné z: http://kore.fi.muni.cz/wiki/index.php/Stripes

[29] Christian Nelson. Stripes: A Successful First Project [online]. [cit. 2015-01-23]. Dostupné z: http://blog.carbonfive.com/2009/02/26/stripes-a-succesful-first-project/

[30] Daoud Fred. Tutorial - Stripes: a lean, mean Java web framework – JAXenter [online]. [cit. 2015-01-04]. Dostupné z: http://jaxenter.com/tutorial-stripes-a-lean-mean-java-web-framework-42991.html

[31] The HSQLDB Development Group. HSQLDB [online]. [cit. 2015-01-02]. Dostupné z: http://www.hsqldb.org/

[32] Fred Toussi. Chapter 1. Running and Using Hsqldb [online]. [cit. 2015-01-28]. Dostupné z: http://hsqldb.org/doc/guide/ch01.html

[33] Opera Software ASA. Dev. Opera — HTML5 Canvas — the Basics [online]. [cit. 2015-01-09]. Dostupné z: https://dev.opera.com/articles/html5-canvas-basics/

[34] Martin Malý. SVG, nebo Canvas? Vyberte si – Zdroják [online]. [cit. 2015-01-28]. Dostupné z: http://www.zdrojak.cz/clanky/svg-nebo-canvas-vyberte-si/

[35] Twitter Bootstrap Tutorial for beginners | w3resources [online]. [cit. 2015-01-02]. Dostupné z: http://www.w3resource.com/twitter-bootstrap/tutorial.php

[36] Marc Schenker. Bootstrap vs. Foundation: Which framework is Better? [online]. [cit. 2015-01-28]. Dostupné z: http://bootstrapbay.com/blog/bootstrap-vs-foundation/

[37] The jQuery Foundation. jQuery [online]. [cit. 2015-01-09]. Dostupné z: http://jquery.com/

[38] jQuery AJAX Introduction [online]. [cit. 2015-01-28]. Dostupné z: http://www.w3schools.com/jquery/jquery_ajax_intro.asp

[39] Alex Bahdanovich. Create a New Maven Project in Eclipse | a Tech-Recipes Tutorial [online]. [cit. 2015-02-11]. Dostupné z: http://www.tech-recipes.com/rx/39279/create-a-new-maven-project-in-eclipse/

[40] The Apache Software Foundation. Maven – Introduction to the POM [online]. [cit. 2015-02-11]. Dostupné z: http://maven.apache.org/guides/introduction/introduction-to-the-pom.html

[41] Spring Framework Reference Documentation [online]. [cit. 2014-12-27]. Dostupné z: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/spring-introduction.html

[42] Spring Framework [online]. [cit. 2014-12-27]. Dostupné z: http://projects.spring.io/spring-framework/

[43] Tutorialspoint. Spring Quick Guide [online]. [cit. 2015-02-11]. Dostupné z: http://www.tutorialspoint.com/spring/spring_quick_guide.htm

[44] Configuration (Spring Framework 4.1.4.RELEASE API) [online]. [cit. 2015-02-11]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/annotation/Configuration.html

[45] Jonathan Hui. Spring MVC - Application Setup | Jonathan Hui [online]. [cit. 2015-02-14]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-http://jonathanhui.com/spring-mvc-application-setup

[46] Helicaltech.com. Helical IT Solutions Pvt. Ltd. [online]. [cit. 2015-02-14]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-http://helicaltech.com/about-multiple-contexts-of-the-spring-mvc-framework/

[47] WALLS, Craig. Spring in action. 3rd ed. Shelter Island: Manning, c2011, 400 p. ISBN 1935182358.

[48] Lokesh Gupta. Spring MVC: Difference between <context:annotation-config> vs <context:component-scan> - How To Do In Java [online]. [cit. 2015-02-14]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-http://howtodoinjava.com/2014/07/19/spring-mvc-difference-between-contextannotation-config-vs-contextcomponent-scan/

[49] PropertyPlaceholderConfigurer (Spring Framework API 2.5) [online]. [cit. 2015-02-14]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-http://docs.spring.io/spring-framework/docs/2.5.x/api/org/springframework/beans/factory/config/PropertyPlaceholderConfigurer.html

[50] InternalResourceViewResolver (Spring Framework API 2.5) [online]. [cit. 2015-02-17]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-http://docs.spring.io/spring-framework/docs/2.5.6/api/org/springframework/web/servlet/view/InternalResourceViewResolver.html

[51] Tutorialspoint. Spring Static Pages Example [online]. [cit. 2015-02-17]. Dostupné z: http://www.tutorialspoint.com/spring/spring_static_pages_example.htm

[52] Keith Donald. MVC Simplifications in Spring 3.0 [online]. [cit. 2015-02-17]. Dostupné z: http://spring.io/blog/2009/12/21/mvc-simplifications-in-spring-3-0/

[53] Pivotal Software. Spring Data [online]. [cit. 2015-02-17]. Dostupné z: http://projects.spring.io/spring-data/

[54] JpaVendorAdapter (Spring Framework 4.1.4.RELEASE API) [online]. [cit. 2015-02-17]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/orm/jpa/JpaVendorAdapter.html

[55] ObjectDB Software. JPA Named Queries (@NamedQuery, @NamedQueries annotations) [online]. [cit. 2015-02-17]. Dostupné z: http://www.objectdb.com/java/jpa/query/named

[56] org.springframework.stereotype (Spring Framework 4.1.4.RELEASE API) [online]. [cit. 2015-02-17]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/package-summary.html

[57] Joseph Kulandai. Spring @Component, @Service, @Repository, @Controller Difference [online]. [cit. 2014-12-27]. Dostupné z: http://javapapers.com/spring/spring-component-service-repository-controller-difference/

[58] 18. View technologies [online]. [cit. 2014-12-27]. Dostupné z: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/view.html

[59] David Winterfeldt. 2. JSP Example [online]. [cit. 2015-02-17]. Dostupné z: http://www.springbyexample.org/examples/spring-web-flow-subflow-webapp-jsp-example.html

[60] Tutorialspoint. Spring MVC Tutorial [online]. [cit. 2015-02-18]. Dostupné z: http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[61] ModelAndView (Spring Framework 4.1.4.RELEASE API) [online]. [cit. 2015-02-18]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/ModelAndView.html

[62] ModelAttribute (Spring Framework 4.1.4.RELEASE API) [online]. [cit. 2015-02-18]. Dostupné z: http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/ModelAttribute.html

[63] Alexey Zvolinskiy. Spring MVC: Form handling with @ModelAttribute annotation [online]. [cit. 2015-02-18]. Dostupné z: http://fruzenshtein.com/spring-mvc-form-handling/

[64] CSS · Bootstrap [online]. [cit. 2015-01-02]. Dostupné z: http://getbootstrap.com/css/

[65] Getting started · Bootstrap [online]. [cit. 2015-01-09]. Dostupné z: http://getbootstrap.com/getting-started/

[66] Bootstrap · The world's most popular mobile-first and responsive front-end framework. [online]. [cit. 2015-01-30]. Dostupné z: http://getbootstrap.com/

[67] Giulio Bai. 10 Ways to Instantly Increase Your jQuery Performance - Tuts+ Code Tutorial [online]. [cit. 2015-02-18]. Dostupné z: http://code.tutsplus.com/tutorials/10-ways-to-instantly-increase-your-jquery-performance--net-5551

[68] AJAX Tutorial [online]. [cit. 2015-01-09]. Dostupné z: http://www.w3schools.com/ajax/default.asp

[69] JSON [online]. [cit. 2015-01-12]. Dostupné z: http://www.json.org/json-cz.html

[70] AJAX Introduction [online]. [cit. 2015-01-12]. Dostupné z: http://www.w3schools.com/ajax/ajax_intro.asp

[71] Systém vyhledávání ikon - Stáhnout 475 450 Ikony zdarma, PNG ikony, Webové ikony [online]. [cit. 2015-01-19]. Dostupné z: http://findicons.com/

[72] *Hudební nástroje [online]. [cit. 2015-01-19]. Dostupné z: http://www.kytary.cz*

[73] *Kytary a hudební nástroje | Elektrické kytary | AUDIOTEK MEGASTORE | AUDIOTEK MEGASTORE [online]. [cit. 2015-01-19]. Dostupné z: http://www.audiotek.cz/*

[74] *MUSIC CITY...pro muzikanty | Hudební nástroje, kytary, bicí a audio [online]. [cit. 2015-02-07]. Dostupné z: http://www.music-city.cz/*

[75] [halek.info] Prezentace ke cvičení z předmětu MARKETING [online]. [cit. 2015-02-07]. Dostupné z: http://halek.info/www/prezentace/marketing-cviceni4/mcvp4-print.php?projection&l=05

13 Přílohy

1) Kompaktní disk (CD)

Obsah CD:

- **/bakalarska_prace**
- **/dbScripty**
- **dbSpusteni**

2) Oskenované zadání práce

FIM UHK
UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Lukáš Šára**

Obor studia: Informační management (3)
Jméno a příjmení vedoucího práce: **Tomáš Kozel**

Název práce:
Dynamický web pro podporu výuky hry na kytaru

Název práce v AJ:
Dynamic web for guitar play learning

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Analyzovat možnosti webové podpory výuky hry na kytaru, navrhnout vhodné výukové nástroje a implementovat je jako dynamickou webovou aplikaci.

Osnova práce:
1. Úvod
2. Stručný úvod do hudební teorie
3. Analýza partií výuky vhodných pro webové zpracování
4. Návrh softwarového řešení
5. Výběr vhodných technologií pro implementaci
6. Popis procesu implementace
7. Výsledky a závěr

Projednáno dne: 14.10.2014

Podpis studenta L. Šára

Podpis vedoucího práce