VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

TVORBA TISKOVÝCH SESTAV NA PLATFORMĚ JAVA

PRINT REPORT CREATION ON THE JAVA PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAKUB MAJZLÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Majzlík Jakub**
Program: Informační technologie
Název: **Tvorba tiskových sestav na platformě Java**
Print Report Creation on the Java Platform
Kategorie: Informační systémy

Zadání:

1. Seznamte se s platformou Java EE pro tvorbu informačních systémů. Zaměřte se na nástroje a knihovny pro vytváření tiskových sestav ve formátu PDF.
2. Prostudujte existující nástroje pro zpracování HTML a CSS předloh na platformě Java.
3. Navrhněte architekturu řešení pro vytváření tiskových PDF sestav na základě šablony ve formátu HTML a CSS s možností snadné integrace do informačních systémů na platformě Java EE.
4. Implementujte navržené řešení a jednoduchou demonstrační aplikaci umožňující vytvoření tiskové sestavy z existující databáze.
5. Proveďte testování vytvořeného řešení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Červinka, Z.: Generování PDF dokumentů z webových stránek, bakalářská práce, Brno, FIT VUT v Brně, 2015
- Juneau, J.: Java EE 7 Recipes, Apress, 2013
- Bien, A.: Real World Java EE Patterns: Rethinking Best Practices, lulu.com, 2009

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 16. října 2019

Abstrakt

Cieľom tejto bakalárskej práce je navrhnúť a implementovať riešenie tvorby tlačových zostáv vo formáte PDF na platforme Java s využitím šablóny v HTML a CSS. V práci sa nachádza opis návrhu, implementácie navrhnutého riešenia tvorby PDF dokumentov z HTML šablóny. Výstupom tejto práce je knižnica a webová aplikácia, ktorá demonštruje funkčnosť navrhnutého riešenia.

Abstract

The goal of this Bachelor's thesis is design and implementation of solution of creating print reports on Java EE platform with use of template written in HTML and CSS. There is a description of the design and implementation of the proposed solution. The output of this thesis is Java library and web application, where functionality of the proposed solution will be demonstrated.

Kľúčové slová

Java EE, HTML, CSS, Tlačové zostavy, Spring Boot, CSSBoxPDF, OpenHTMLtoPDF

Keywords

Java EE, HTML, CSS, Print reports, Spring Boot, CSSBoxPDF, OpenHTMLtoPDF

Citácia

MAJZLÍK, Jakub. *Tvorba tiskových sestav na platformě Java.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Tvorba tiskových sestav na platformě Java

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje z ktorých som čerpal.

.....
Jakub Majzlík
26. mája 2020

Poďakovanie

Rád by som poďakoval vedúcemu práce Ing. Radekovi Burgetovi, Ph.D. za odborné vedenie pri tvorbe tejto bakalárskej práce a za konzultácie.

Obsah

Kapitola 1

Úvod

Vo svete informačných technológií sa nachádza veľa dát. Firmy disponujú gigabajtami ak nie aj terabajtami dát. Napríklad môžu to byť dáta o objednávkach, ktoré treba vytlačiť a poslať zákazníkovi aj s objednaným tovarom. Samotné dáta na papieri môžu byť neprehľadné a nemusia samé o sebe dávať význam. K tomu aby sme zlepšili čitateľnosť týchto dát nám pomôže tlačová zostava. Tlačovou zostavou môžeme chápať šablónu dokumentu s určitým formátom.

Cieľom tejto práce je navrhnúť a implementovať riešenie tvorby tlačových zostáv vo formáte PDF na platforme Java. V súčasnej dobe existuje veľa programov a knižníc určených na tvorbu PDF, no nie všetky sú ideálnym riešením. Ak programátor chce vytvoriť PDF dokument za pomoci knižnice na to určenej, často si musí preštudovať dokumentáciu a samotná tvorba zložitejších dokumentov môže byť zdĺhavá. Hlavným motívom tejto práce je zabezpečiť jednoduchšiu tvorbu tlačových zostáv a jednoduchšiu integráciu do existujúceho informačného systému.

Jednoduchým riešením je generovanie PDF zostáv s použitím HTML šablón. Výhodou generovania PDF zostáv z HTML šablóny je, že samotná tvorba šablóny nie je zložitá. Programátor, ktorý vytvára webové aplikácie sa s jazykom HTML už určite stretol a ak nie tak naučiť sa tento jazyk nie je zložité. Nevýhodou použitia HTML šablón pre generovanie PDF zostáv môže byť samotné spracovanie jazyka knižnicou. Výsledný dokument nemusí stopercentne odpovedať šablóne a pri použití viacerých knižníc na generovanie PDF dokumentov nemusí byť výsledok rovnaký.

Samotný preklad bude prebiehať vo filtri. Tento filter si programátor zaregistruje v kóde webovej aplikácie na adresu, ktorá bude vracať HTML kód vyplnenej šablóny. Náročnosť tvorby tlačovej zostavy nie je vysoká, lebo samotná aplikácia sa musí zaoberať len doplnením dát do šablóny.

Kapitola 2 sa bude zaoberať existujúcimi riešeniami tvorby tlačových zostáv vo formáte PDF. V kapitole 3 sa bude nachádzať popis použitých programovacích, či značkovacích jazykov, vrátane použitých technológií a knižníc. Návrhom riešenia a demonštračnej aplikácie sa bude zaoberať kapitola 4. V kapitole 5 bude popísaná implementácia jednotlivých filtrov, demonštračnej aplikácie a použitých štruktúr. Nakoniec v kapitole 6 otestujeme demonštračnú aplikáciu a porovnáme výsledky jednotlivých filtrov.

Výstupom tejto práce je knižnica, ktorá obsahuje dve implementácie filtrov, ktoré generujú PDF dokument z HTML kódu a webová aplikácia, v ktorej si budeme demonštrovať funkčnosť vytvoreného riešenia.

Kapitola 2

Existujúce riešenia

V tejto kapitole budú stručne popísané niektoré existujúce riešenia na tvorbu tlačových zostáv na platforme Java.

2.1 Knižnice na manipuláciu s PDF dokumentami

Možným riešením tvorby PDF dokumentov je aj využitie knižníc na to určených. S pomocou týchto knižníc je možné vytvoriť PDF dokument od základov. Tvorba PDF dokumentov týmto spôsobom je pracná a časovo náročnejšia, ako použitie nástrojov s užívateľským rozhraním, či generovaním na základe XML, alebo HTML šablóny. Je možné aj dosadenie dát na určité súradnice v už existujúcom PDF dokumente, avšak nájdenie tej správnej pozície zaberie čas.

Príkladom knižníc určených na tvorbu a manipuláciu s PDF dokumentami môže byť knižnica `iText`, alebo `PDFBox`.

2.2 Big Faceless Report Generator

Big Faceless Report Generátor je Java nástroj na generovanie PDF reportov z jazyka XML. Vyžíva technológiu JSP a ASP, pomocou ktorej je možné vytvárať dynamické PDF reporty[1]. Navrhnutú šablónu si vo webovej aplikácií načítame, pripojíme k nej zdroj dát a následne môžeme vyexportovať do formátu PDF.

2.3 JasperReports

JasperReports je open source nástroj určený na tvorbu „pixel-perfect“ PDF reportov. Umožňuje exportovanie reportov nielen do PDF, ale aj dokument formátu ako napríklad HTML, Excel, Word[5]. Šablóny pre tento nástroj sa ukladajú do súborov s príponou `.jrxml`. Túto šablónu si následne môžeme načítať za pomoci JasperReports Library do Java projektu, pripojiť k nej zdroj dát a vyexportovať, ako PDF dokument. Súčasťou projektu je JasperReport Designer, ktorý umožňuje tvorbu šablóny v grafickom editore.

Kapitola 3

Použité technológie

Táto kapitola sa bude zaoberať použitými technológiami v tejto práci. Nachádza sa tu stručný opis použitých programovacích a značkovacích jazykov, technológií, frameworkov a knižníc. Dôraz sa bude brať na platformu Java EE a niektoré technológie súvisiace s touto platformou (Java Server Pages a Servlets).

3.1 Java

Programovací jazyk Java vyvinula spoločnosť Sun Microsystems a momentálne Javu spravuje firma Oracle. Jedná sa o jeden z najpoužívanejších a najobľúbenejších jazykov na svete [10].

Java nie je len programovací jazyk, ale aj platforma, ktorá poskytuje veľa knižníc, ktoré obsahujú množstvo znovu použiteľných kódov a takzvaný „Execution Enviroment“, ktorý zabezpečuje bezpečnosť, portabilitu a automatický garbage collector[4].

Java nám ponúka široké spektrum platforiem, kde môžeme vyvíjať aplikácie, a to platforma Java ME, čo je platforma určená na vývoj aplikácií s obmedzenou pamäťou a výkonom (mikrokontroléry, senzory, tlačiarne, a tak ďalej.)[7], JavaCard, ktorá sa využíva na vývoj a bezpečný beh aplikácií pre čipové karty. Ďalšou platformou je Java SE, ktorá je vhodná na vývoj desktopových a serverových aplikáciií a Java EE, ktorá je určená na tvorbu podnikových aplikácií.

Program napísaný v Jave je preložený do tzv. „bytecodu“, ktorý je následne interpretovaný a beží na virtuálnom stroji Javy (Java Virtual Machine). Táto vlastnosť nám zabezpečuje to, že program je multiplatformový. Nachádza sa tu takzvaný „garbage collector“, ktorý sa stará o automatickú správu pamäti. Keď zanikne posledná referencia na objekt, „garbage collector“ sa postará o uvoľnenie objektu z pamäte.

Jazyk Java je silno typový programovací jazyk, čo znamená, že program striktne vyžaduje vopred vymedzené dátové typy premenných, parametrov, a tak ďalej. Typová kontrola je statická, ktorá sa vykonáva počas kompilácie. Tento typ kontroly nám umožňuje zachytiť typové chyby už počas kompilácie. Java obsahuje osem primitívnych dátových typov (boolean, byte, char, short, int, long, float a double), všetky ostatné dátové typy sú objektové.

Pre spustenie programu napísaného v Jave, je potrebné aby program obsahoval triedu s metódou `main`. Bez tejto metódy nie je možné spustiť program.

Príklad jednoduchej triedy:

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3        System.out.println("Hello world!")
4      }
5  }
```

Program napísaný v Jave je zapísaný v súbore s príponou .java, bytecode sa nachádza v súbore s príponou .class a spustiteľný súbor má príponou .jar, alebo .war.

3.1.1 Platforma Java EE

Primárna platforma tejto práce je Java EE, tak si v tejto podkapitole zhrnieme niektoré technológie tejto platformy. O ostatných technológiách môžeme nájsť viac informácií v špecifikáciách, ktoré sa nachádzajú na stránkach Java Community Proccess[1], alebo v tutoriáli zo stránky Oraclu[2].

Java EE je súbor špecifikácií, ktoré rozširujú štandardnú edíciu Javy (Java SE) o technológie a nástroje vhodné na vývoj informačných systémov, serverových a podnikových aplikácií. Java EE aplikácie sú založené na modeli klient-server.

Java EE technológie sú podľa oficiálnej stránky Oracle rozdelené do piatich skupín[8]:

(a) Java EE platforma – Java Enterprise Edition Platform

(b) Technológie webových aplikácií – Java Servlet, Java Server Faces, Java Server Pages, a ďalšie

(c) Technológie podnikových aplikácií – Dependency Injection, Bean Validation, Java Persistence, a ďalšie

(d) Technológie webových služieb – Java API for RESTful Web Services(JAX-RS), Web Services Metadata for the Java Platform, Java API for XML Registries, a ďalšie

(e) Technológie správy a bezpečnosti – Java EE security API, Java Authetication Service Provider Interface for Containers, Java EE Aplication Deploytment, a ďalšie

(f) Špecifikácie spojené s Javou EE z Javy SE – Java Management Extension (JMX), Java API for XML Processing (JAXP), Java Architecture for XML Binding (JAXB), a ďalšie

Benefity aplikácií založených na platforme Java EE:

- Takéto aplikácie sú platformovo nezávislé. Aplikácia pobeží na všetkých systémoch, pre ktoré sú k dispozícii aplikačné servery.
- Aplikácia môže bežať na rôznych aplikačných serveroch, ktoré implementujú Java EE štandard.
- Java EE špecifikácia nám poskytuje veľké množstvo rozhraní vhodných na tvorbu, podnikových aplikácií (webové služby, práca s databázou, bezpečnosť).
- Existuje veľké množstvo frameworkov, nástrojov a ready-to-use aplikácií, ktoré uľahčia vývoj.

[1] https://www.jcp.org/en/home/index
[2] https://docs.oracle.com/javaee/7/JEETT.pdf

3.1.2 Aplikačný server

Keďže Jave EE je len špecifikácia, potrebujeme aj jej implementáciu. Táto implementácia sa nachádza na takzvaných aplikačných serveroch. Tieto aplikačné servery implementujú celú špecifikáciu Javy EE, alebo len jej časť.

Medzi známe aplikačné servery patrí:

- Apache Tomcat
- GlassFish (referenčná implementácia Javy EE)
- JBoss Application Sever
- IBM Application Server
- Oracle WebLogic

3.1.3 Servlet

Java Servlet je jedna z technológií webových aplikácií Javy EE. Servlet je trieda, ktorá rozširuje možnosti webového serveru hosťujúceho webovú aplikáciu. Táto trieda spracováva HTTP požiadavky a generuje HTML kód odpovede.

Na tvorbu vlastného servletu nám poslúžia balíčky `javax.servlet` a `javax.servlet.http`[6]. Implementácia servletu môže vyzerať nasledovne:

```
1  public class MyServlet extends HttpServlet {
2      @Override
3      public void doGet(HttpServletRequest request, HttpServletResponse response)
           throws IOException {
4          response.getWriter().println("Hello World");
5      }
6  }
```

3.1.4 Filter

Filter je súčasťou Java Servlet technológií. Filtrom sa rozumie objekt, ktorý sa používa na transformáciu hlavičky a obsahu požiadavku, alebo odpovede. Tento objekt sa prichytí k ľubovoľnému webovému zdroju, ktorému poskytne svoju funkčnosť. Webový zdroj môže využívať viacero filtrov, ktoré sa následne budú reťaziť.

Možné využitie filtrov je napríklad logovanie, autentifikácia, kompresia dát, šifrovanie, a tak ďalej[6].

Pri použití rozhrania `Filter`, je potrebné implementovať nasledujúce metódy:

- **public void init(FilterConfig filterConfig)** – Metóda, ktorá sa zavolá hneď po tom, ako sa inicializuje filter. Môžu sa tu nachádzať inicializácie objektov, alebo napríklad záznam o inicializácií filtra.
- **public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)** – Metóda sa zavolá vždy, keď filter zachytí požiadavku od klienta, alebo odpoveď od servera.
- **public void destroy()** – Metóda sa zavolá pred tým, ako je filter zničený. V metóde sa môže nachádzať kód na uvoľnenie zdrojov, alebo napríklad záznam o zničení filtra.

Filter si môžeme zaregistrovať v súbore `web.xml`. Tento súbor popisuje, na ktorých adresách je potrebné autorizovať používateľov, registruje filtre, mapovanie servletov a iné informácie.

Filter v súbore `web.xml` môže byť zapísaný nasledovne:

```
<filter>
  <filter-name>MyFilter</filter-name>
  <filter-class>sk.jakubmajzlik.filter.MyFilterImplementation</filter-class>
</filter>

<filter-mapping>
  <filter-name>MyFilter</filter-name>
  <url-pattern>/path/to/resource</url-pattern>
</filter-mapping>
```

Ďalším možným spôsobom ako zaregistrovať filter je použitie anotácie `@WebFilter` na triede, ktorá reprezentuje daný filter.

Ukážková trieda môže vyzerať nasledovne:

```
  @WebFilter(urlPatterns = {"/path/to/resource"})
  public class MyFilterImplementation implements Filter {
    public void init(FilterConfig filterConfig) throws ServletException { }

    public void destroy() { }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
      // Spracovanie poziadavku

      chain.doFilter();

      // Spracovanie odpovede
    }
  }
```

3.1.5 Java Server Pages

Java Server Pages (ďalej už len JSP) je ďalšia z technológií webových aplikácií Javy EE. JSP sa využíva na generovanie dynamického webového obsahu[3]. Pre účely doplňovania dát do šablón budeme v demonštračnej aplikácii využívať túto technológiu. Alternatívou, k tejto technológii môže byť Java šablónovací engine Thymeleafe.

3.2 HTML

Pre návrh šablón tlačových zostáv sa bude využívať značkovací jazyk HTML.

Hypertext Markup Language (ďalej už len HTML) je značkovací jazyk, ktorý popisuje štruktúru dokumentu, ktorý sa má zobraziť vo webovom prehliadači. Celosvetovo je využívaný na tvorbu webových stránok. HTML súbor má príponu `.htm`, alebo `.html`. Tento dokument sa skladá zo značiek (tagov), ktoré sa skladajú zo znakov „`<`“ a „`>`“, medzi ktorými sa nachádza názov tagu. Tagy môžu byť párové (napr. `<h1></h1>`), alebo nepárové

(napr. `<img>`). Každý tag môže mať svoje vlastnosti (atribúty), napríklad id (identifikátor tagu), class (trieda do ktorého patrí tag), style (definuje štýl tagu) a tak ďalej. Znakové entity sa zapisujú ako „`&`“, nasleduje krátky zápis danej entity a končí znakom „`;`“ (napr. „©“ sa zapisuje, ako `&copy;`).

HTML dokument sa skladá z niekoľkých častí. Na začiatku HTML dokumentu sa nachádza informácia pre prehliadač v tvare `<!DOCTYPE html>`, ktorá špecifikuje typ súboru, aby prehliadač vedel, ako má spracovať daný súbor. Nasleduje párový tag `<html>`, v ktorom sa nachádza párový tag `<head>` a `<body>`. V tele tagu `<head>` sa nachádzajú odkazy na súbory, nastavenie titulky stránky, definície štýlov v CSS a iné metadáta. V tele tagu `<body>` sa nachádza obsah samotnej stránky.

HTML dokument často dopĺňajú súbory so štýlmi, napísanými v jazyku CSS a súbory obsahujúce skripty, napísané v jazyku JavaScript, ktoré sa používajú na tvorbu animácií, efektov, a tak ďalej.

Príklad jednoduchej webovej stránky:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Website title</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

3.3 CSS

Cascading Style Sheets (ďalej už len CSS) je jazyk, ktorý sa používa pre špecifikáciu vzhľadu prvkov jazyka HTML, XHTML, alebo XML. Tento jazyk umožňuje zápis štýlov pre konkrétny tag, pre tagy, ktoré sa nachádzajú v určitej triede, alebo pre tag s určitým identifikátorom. Tieto štýly sa môžu zapisovať do súboru s koncovkou .css, ktorý si s pomocou tagu `<link>` môžeme vložiť do HTML súboru, alebo priamo v HTML súbore v tele párového tagu `<style>`, ktorý sa nachádza v tele párového tagu `<head>`.

Štýly v CSS sa zapisujú, ako postupnosť pravidiel. Každé pravidlo sa skladá zo selektoru a bloku, kde sa deklarujú vlastnosti selektora. Každá vlasnosť sa skladá z mena a z oboru hodnôt, ktoré môže nadobúdať. Názov od hodnoty sa oddeľuje znakom „`:`“. Na stránke W3C sa môžeme dočítať, že existuje niekoľko typov selektorov, ako napríklad[11]:

- typový selektor – pravidlo sa aplikuje na prvok určitého typu (selektorom je názov tagu), napríklad nastavenie farby písma pre nadpis prvej úrovne na modrú sa použije pravidlo: `h1 { color: blue; }`
- triedny selektor – pravidlo sa aplikuje na všetky prvky, ktoré sa nachádzajú v určitej triede, čiže na prvky, ktoré majú nastavený atribút class na názov selektora. Selektor sa skladá z „`.`“ a názvu. Napríklad pravidlo, ktoré nastaví farbu písma pre všetky prvky, ktoré patria do triedy „`red`“, bude vyzerať nasledovne: `.red { color: red;}`
- ID selektor – pravidlo sa aplikuje na prvok, ktorý obsahuje atribút id, ktorého hodnota sa rovná názvu selektora. Selektora sa skladá z „`#`“ a názvu. Napríklad pravidlo, ktoré

nastaví farbu písma na zelenú pre prvok, ktorého identifikátor sa rovná „`green`“, bude vyzerať nasledovne: `#green { color: green; }`

- univerzálny selektor – pravidlo sa aplikuje na všetky prvky. Selektor sa skladá zo znaku „`*`“. Napríklad pravidlo, ktoré zmení font písma na „`Courier New`“ , pre všetky prvky v dokumente bude vyzerať nasledovne: `* { font-family: Courier New; }`
- a iné

3.4 Spring Boot

Pre uľahčenie vývoja demonštračnej aplikácie som sa rozhodol použiť framework Spring Boot. Tento framework je postavený na frameworku Spring Framework, ktorý slúži na uľahčenie vývoja moderných Java aplikácií. Spring Boot obsahuje celú radu prednastavených konfiguračných súborov, ktoré by sme pre spustenie webovej aplikácie vo Spring Frameworku museli písať ručne.

Pre vytvorenie jednoduchej webovej aplikácie potrebujeme len zahrnúť v Maven projekte závislosť `spring-boot-starter-web`. Na spustenie Spring Boot webovej aplikácie nepotrebujeme mať nainštalovaný webový server v počítači. Spring Boot pri použití závislosti `spring-boot-starter-web` automaticky obsahuje vstavaný Tomcat Server. Tomcat je predvoleným web serverom pre Spring Boot, no je ho možné nahradiť webovými servermi Jetty (`spring-boot-starter-jetty`), alebo Undertow (`spring-boot-starter-undertow`)[9].

3.5 Knižnice na generovanie PDF z HTML

Pri nasadení budeme mať možnosť výberu z viacerých implementácií filtrov. V tejto práci budú naimplementované dva filtre, kde každý bude využívať inú knižnicu na generovanie PDF z HTML.

V následujúcej časti si stručne popíšeme knižnice OpenHTMLtoPDF a CSSBoxPDF, s pomocou ktorých budú naimplemntované generovanie PDF vo filtroch.

3.6 Knižnica OpenHTMLtoPDF

Open HTML to PDF je open source Java knižnica, ktorá umožňuje generovanie PDF súbory z XML/XHTML, HTML a CSS súborov. Knižnica je postavená na knižniciach Flying Soucer a Apache PDF-Box.

Open HTML to PDF pracuje s w3c DOM Document objektom. Samotná knižnica nevie pracovať s HTML5, ale s použitím knižnice `JSoup` si vieme HTML5 rozpársovať na w3c DOM Document.

Existujú aj rôzne rozšírenia pre túto knižnicu, ako napríklad JFreeChart, ktoré slúži na vkladanie koláčových, alebo stĺpcových diagramov. Ďalším rozšírením je Latex, ktoré prevádza LaTeX na XHTML, MathML, CSS a rozšírenie SVG Images, ktoré zaisťuje podporu SVG formátu v HTML[12].

Knižnica je dostupná pod LGPL licenciou.

3.7 Knižnica CSSBoxPDF

CSSBoxPDF je rozšírenie knižnice CSSBox o možnosť generovať výstup vo formáte PDF. Samotná knižnica CSSBox je XHTML/HTML/CSS renderovací engine napísaný v Jave. Engine spracováva DOM model a súbory s CSS vlastnosťami na objektovo-orientovaný model webovej stránky[2].

Knižnica je dostupná pod GNU Lesser General Public License verzia 3 licenciou.

Kapitola 4

Návrh

Nasledujúca kapitola sa zaoberá návrhom. Bude rozdelená na dve časti, architektúra riešenia tvorby tlačových zostáv a architektúra demonštračnej aplikácie.

V architektúre riešenia sa môžme dočítať, ako prebieha požiadavka na získanie vyplnenej PDF šablóny.

Architektúra demonštračnej aplikácie sa bude zaoberať návrhom databázy pre dáta určené na vyplnenie šablón, akým spôsobom je možné vytvoriť šablónu a návrhom samotnej aplikácie.

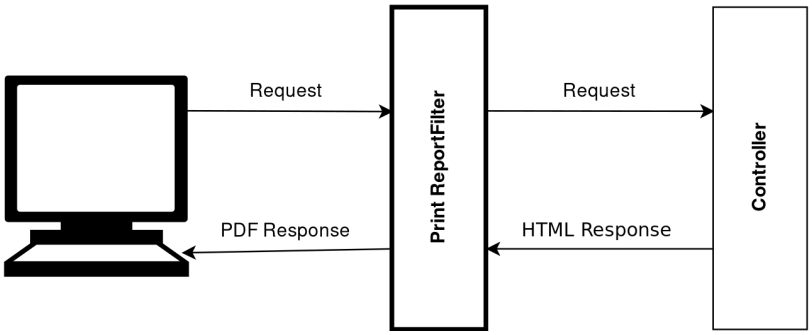
4.1 Architektúra riešenia

Základom riešenia je filter, v ktorom prebieha preklad HTML a CSS kódu na PDF za pomoci knižnice OpenHTMLtoPDF, alebo knižnice CSSBoxPDF. Výber knižnice bude závisieť od vybranej implementácie filtra.

Ak bude chcieť programátor použiť inú knižnicu na generovanie PDF dokumentov z HTML, nebude musieť implementovať celý filter. Implementácia filtra a proces získania kódu vyplnenej šablóny z odpovede servera sa totiž bude nachádzať v abstraktnej triede `PrintReportFilter`. Programátor si len vytvorí novú triedu, ktorá bude dediť zo spomínanej triedy a bude musieť naimplementovať abstraktnú metódu `getPDF`. V tejto triede sa inicializujú potrebné objekty, vytvorí sa DOM strom z HTML kódu, prebehne postprocessing, ak bude potrebné, vygeneruje sa PDF dokument a vráti sa ako `ByteArrayOutputStream`.

Pri použití filtra je splnená požiadavka zo zadania, že riešenie má byť ľahko integrovateľné do informačného systému na platforme Java. Pre integráciu do informačného systému je potrebné len zaregistrovať filter a je treba zabezpečiť aby adresa, na ktorej je filter zaregistrovaný, vracala HTML kód vyplnenej šablóny.

Ako môžeme vidieť na obrázku 4.1, keď zadáme požiadavku na získanie PDF dokumentu, tak požiadavka prejde bez zmeny cez filter do kontroléru, kde sa vyplní šablóna s požadovanými dátami z databázy a vygeneruje sa HTML kód. Následne je odpoveď s vyplnenou šablónou vo formáte HTML odoslaná klientovi. Túto odpoveď zachytí filter, ktorý vezme HTML, s pomocou knižnice OpenHTMLtoPDF, alebo CSSBoxPDF vytvorí PDF dokument, upraví sa hlavička odpovede na „`application/pdf`“ a odošle vytvorený PDF dokument klientovi.



Obr. 4.1: Diagram návrhu

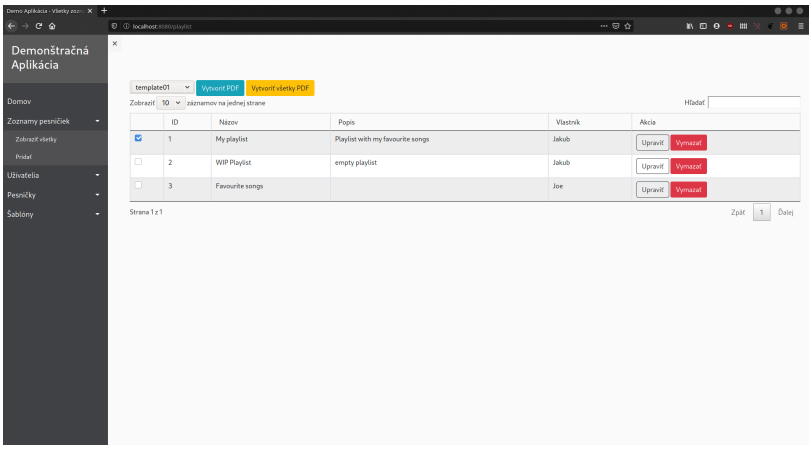
4.2 Demonštračná aplikácia

Súčasťou práce je aj aplikácia, ktorá má demonštrovať funkčnosť navrhnutého riešenia. Jej hlavnou úlohou je tvorba tlačových zostáv z existujúcej databázy. Keďže sa jedná len o demonštračnú aplikáciu, nie sú v nej implementované časti, ako napríklad prihlásenie, alebo kontrola rolí, ktoré by sa za normálnych okolností nachádzali v informačnom systéme.

Demonštračná aplikácia bude predstavovať časť informačného systému hudobnej streamovacej služby. Medzi hlavné funkcie, ktoré bude aplikácia vykonávať, patrí správa používateľov streamovacej služby, správa pesničiek a zoznamov hudby. Medzi ďalšie funkcie patrí tvorba zostáv pre jeden záznam, alebo viac záznamov z tabuľky a správa šablón. Aplikácia bude postavená na návrhovom vzore MVC (Model-View-Controller).

Pri vývoji demonštračnej aplikácie bude využitý framework Spring Boot. Použitie tohto frameworku nám značne uľahčí prácu s databázov a so spracovaním jednotlivých požiadavok.

O pohľady rovnako ako o šablóny sa bude starať technológia JSP. JSP nám umožní si vytvoriť vlastný tag (`page.tag`[1]), ktorý nám zadefinuje predvolené rozloženie a vzhľad stránky. Na obrázku 4.2 môžeme vidieť rozloženie stránky. Bude pozostávať z dvoch častí, a to navigačné menu, ktoré sa bude nachádzať na ľavej strane každej stránky a obsahu samotnej stránky.



Obr. 4.2: Rozloženie stránky

[1] Adresa súboru je na adrese: „src/main/webapp/WEB-INF/tags/page.tag“

4.2.1 Tvorba zostavy

V aplikácii bude možné vytvárať tlačové zostavy pre všetky záznamy, alebo len pre vybrané. Zakliknutím checkboxov záznamov v aplikácii, vyberieme tie záznamy, z ktorých sa majú vytvoriť tlačové zostavy. Po výbere šablóny a kliknutí na tlačítko `Vytvoriť PDF`, sa vyplní model s informáciami o vybraných záznamoch, názvu šablóny a odošle sa na validáciu[2] a následne na generovanie HTML kódu[3]. Pre vytvorenie zostáv pre všetky záznamy stačí si len vybrať šablónu a kliknúť na tlačítko `Vytvoriť všetky PDF`. Po kliknutí na tlačítko JavaScript zaklikne za nás všetky checkboxy a odošle vyplnený model na validáciu a následne na generovanie HTML kódu.

Pri validácii sa bude kontrolovať, či bol vybraný názov šablóny a či model obsahuje aspoň jedno id záznamu, z ktorého sa má vytvoriť tlačová zostava. Ak nebude splnená niektorá z týchto podmienok, zobrazí sa nám opätovne stránka s chybovou hláškou, kde doplníme chýbajúce údaje.

4.2.2 Stránky

Demonštračná aplikácia bude obsahovať nasledujúce stránky:

- **Úvodná stránka** – Úvodná stránka s informáciami o aplikácií. Nachádzať sa bude na adrese „`/`“.
- **Správa šablón** – Možnosť náhľadu, pridania a vymazania šablóny. Nachádzať sa bude na adrese „`/template`“.
- **Správa užívateľov** – Možnosť pridania, úpravy, vymazania užívateľa a možnosť tvorby tlačovej zostavy. Nachádzať sa bude na adrese „`/user`“.
- **Správa pesničiek** – Možnosť pridania, úpravy, vymazania pesničky a možnosť tvorby tlačovej zostavy. Nachádzať sa bude na adrese „`/song`“.
- **Správa zoznamov s hudbou** – Možnosť pridania, úpravy, vymazania zoznamu s hudbou a možnosť tvorby tlačovej zostavy. Nachádzať sa bude na adrese „`/playlist`“.

4.2.3 Návrh databázy

Vďaka Spring Bootu potrebujeme minimum SQL kódu. Jedinú vec o kturú sa programátor musí postarať je vytvorenie databázy a používateľa[4]. O vytvorenie tabuliek sa postará Spring Boot pri prvom spustení demonštračnej aplikácie. Tabuľky budú vytvorené na základe atribútov entitných tried.

Databáza bude obsahovať nasledujúce tabuľky, ktoré budú popísané nižšie:

- playlists
- songs
- songs_in_playlist
- users
- users_user_playlists

[2] Adresa validácie: „/[kategória]/validate“
[3] Adresa generovania šablón: „/[kategória]/make“
[4] Script pre vytvorenie databázy a užívateľa sa nachádza na adrese „src/main/sql/db_init.sql“

Tabuľka playlists

Tabuľka `playlists` bude obsahovať informácie o zoznamoch s hudbou. Vygenerovaná bude na základe entitnej triedy `PlaylistEntity.java`[5]. Bude obsahovať nasledujúce stĺpce:

- **id** – Primárny kľúč zoznamu s hudbou.
- **description** – Popis zoznamu s hudbou.
- **last_updated** – Dátum a čas kedy bol zoznam s hudbou upravený.
- **name** – Názov zoznamu s hudbou.
- **owner_id** – Cudzí kľúč vlastníka zoznamu s hudbou. Tento kľúč sa odkazuje na tabuľku `users`.

Tabuľka songs

Tabuľka `songs` bude obsahovať informácie o pesničkách. Vygenerovaná bude na základe entitnej triedy `SongEntity.java`[6]. Bude obsahovať nasledujúce stĺpce:

- **id** – Primárny kľúč pesničky.
- **album_name** – Meno albumu, do ktorého pesnička patrí.
- **artist_name** – Názov autora, alebo autorov pesničky.
- **duration** – Dĺžka pesničky.
- **song_name** – Názov pesničky.

Tabuľka songs_in_playlist

Tabuľka `songs_in_playlist` bude obsahovať informácie o tom, ktorá pesnička sa nachádza v akom zozname s hudbou. Vygenerovaná bude na základe atribútu `listOfSongs` entitnej triedy `PlaylistEntity.java`[7]. Bude obsahovať nasledujúce stĺpce:

- **playlist_id** – Cudzí kľúč zoznamu s hudbou, do ktorej má pesnička patriť. Tento kľúč sa odkazuje na tabuľku `playlists`.
- **song_id** – Cudzí kľúč pesničky, ktorá má patriť do zoznamu s hudbou. Tento kľúč sa odkazuje na tabuľku `songs`.

[5] Adresa entitnej triedy: „src/main/java/sk/jakubmajzlik/printreport/entity/PlaylistEntity.java“
[6] Adresa entitnej triedy: „src/main/java/sk/jakubmajzlik/printreport/entity/SongEntity.java“
[7] Adresa entitnej triedy: „src/main/java/sk/jakubmajzlik/printreport/entity/PlaylistEntity.java“

Tabuľka users

Tabuľka `users` bude obsahovať informácie o užívateľovi. Vygenerovaná bude na základe entitnej triedy `UserEntity.java`[8]. Bude obsahovať nasledujúce stĺpce:

- **id** – Primárny kľúč užívateľa.
- **email** – Email užívateľa.
- **first_name** – Krstné meno užívateľa.
- **last_name** – Priezvisko užívateľa.

Tabuľka users_user_playlists

Tabuľka `users_user_playlists` bude obsahovať informácie o tom, ktoré zoznamy s hudbou vlastní užívateľ. Vygenerovaná bude na základe atribútu `userPlaylists` entitnej triedy `UserEntity.java`[9]. Bude obsahovať nasledujúce stĺpce:

- **user_id** – Cudzí kľúč užívateľa, ktorý vlastní zoznam s hudbou. Tento kľúč sa odkazuje na tabuľku `users`.
- **playlist_id** – Cudzí kľúč zoznamu s hudbou, ktorý vlastní užívateľ. Tento kľúč sa odkazuje na tabuľku `playlists`.

4.2.4 Šablóny

Šablóny je možné pridať do systému dvoma spôsobmi, a to buď skopírovaním súboru do zložky so šablónami, alebo skopírovaním kódu a nahranie cez webovú aplikáciu[10].

V demonštračnej aplikácii budú šablóny rozdelené do troch kategórií, a to kategória pre zoznamy s hudbou, kategória pre pesničky a kategória pre užívateľov. To, do ktorej kategórie bude šablóna patriť, závisí od toho v akom podpriečinku sa bude šablóna nachádzať. Pri pridávaní šablóny cez webové rozhranie, stačí len vybrať kategóriu z drop-down listu. Na základe kategórie, do ktorej šablóna patrí, bude vytvorený model s dátami a odoslaný na doplnenie do šablóny.

Štýly v šablónach je možné zapisovať do hlavičky medzi tag `<style>`, alebo aj do externých súborov a mali by sa nachádzať pri šablóne. Ak pridáme štýl cez webové rozhranie, aplikácia vytvorí súbor, ktorý bude mať rovnaké meno ako šablóna a bude umiestnený na rovnakom mieste. Pridanie štýlu do šablóny bude vyzerať nasledovne:

```
1  ...
2  <link rel="stylesheet" href="<c:url value='/[kategoria]/[nazov sablony].css'/>">
3  ...
```

[8] Adresa entitnej triedy: „src/main/java/sk/jakubmajzlik/printreport/entity/UserEntity.java“
[9] Adresa entitnej triedy: „src/main/java/sk/jakubmajzlik/printreport/entity/UserEntity.java“
[10] Adresa stránky na pridanie šablóny: „/template/add“

Kapitola 5

Implementácia

5.1 Filtre

Knižnica bude obsahovať dve implementácie filtrov, a to CSSBoxPDF implementáciu a OpenHTMLtoPDF implementáciu. Tieto triedy, ktoré budú prekladať HTML na PDF dokument. Samotné triedy nebudú implementovať rozhranie `Filter`, ale budú dediť z triedy `PrintReportFilter`[1] , ktorá obsahuje implementáciu tohto rozhrania.

V nasledujúcich podkapitolách budú popísané implementácie jednotlivých filtrov, abstraktnej triedy `PrintReportFilter` a jednej pomocnej triedy z projektu `printreportlib`.

5.1.1 PrintReportFilter

Abstraktná trieda `PrintreportFilter` bude implementovať rozhranie `Filter`. V triede sa bude nachádzať okrem metód `init`, `destroy` a `doFilter` z rozhrania `Filter` aj abstraktná metóda `ByteArrayOutputStream getPDF(String, String)` a metóda `void setPDFResponse(HttpServletResponse, ByteArrayOutputStream)`.

Abstraktnú metódu `getPDF` bude potrebné naimplementovať v triedach, ktoré budú reprezentovať jednotlivé implementácie filtrov. V tejto metóde by sa mal nachádzať kód, ktorý vygeneruje PDF dokument z HTML kódu a vráti ho ako `ByteArrayOutputStream`.

Metóda `setPDFResponse` nastaví hlavičku odpovede. Upravia sa atribúty hlavičky odpovede `Content-Disposition`, `Content-Length` a `Content-Type`.

Metódy `init` a `destroy` nie je potrebné implementovať a preto ostanú prázdne. V metóde `doFilter` bude prebiehať spracovanie odpovede servera. Po tom, ako filter zachytí odpoveď, získame si HTML kód vyplnenej šablóny, ktorý použijeme ako parameter metódy `getPDF` na získanie PDF dokumentu. Následne upravíme hlavičku odpovede s pomocou metódy `setPDFResponse` a odošleme vygenerovaný dokument klientovi.

5.1.2 CSSBoxPDF implementácia

Implementáciu filtra s využitím knižnice CSSBoxPDF bude reprezentovať trieda `PrintReportCSSBoxPDFFilter`[2] z projektu `printreportlib`.

Trieda bude dediť z abstraktnej triedy `PrintReportFilter` a implementuje abstraktnú metódu `getPDF`.

[1] Adresa súboru: „src/main/java/sk/jakubmajzlik/printreportlib/PrintReportFilter.java“

[2] Adresa súboru: „src/main/java/sk/jakubmajzlik/printreportlib/PrintReportCSSBoxPDFFilter.java“

V nasledujúcom zozname budú uvedené všetky inicializované objekty potrebné na vygenerovanie PDF dokumentu:

- **inputStream** – Vstupný tok dát s HTML kódom pre objekt `documentSource` typu `ByteArrayInputStream`.
- **outputStream** – Výstupný tok dát, ktorý bude obsahovať pdf dokument vyplnenej šablóny.
- **media** – Špecifikácia mena výstupného zariadenia.
- **documentSource** – Objekt reprezentujúci dokument typu `StreamDocumentSource`.
- **parser** – Parser, pomocou ktorého získame DOM z objektu `documentSource`, typu `DefaultDOMSource`.
- **domAnalyzer** – Objekt, ktorý bude analyzovať DOM, typu `DOMAnalyzer`.
- **pdfEngine** – Objekt, ktorý sa postará o vytvorenie PDF dokumentu, typu `PDFEngine`.

V triede `getPDF` prebehne inicializácia objektov spomínaných vyššie, s pomocou objektu `pdfEngine` sa vygeneruje PDF dokument a uloží sa do `outputStream`. Vygenerovaný dokument následne metóda vráti.

Okrem naimplementovanej metódy `getPDF` z abstraktnej triedy `PrintReportFilter`, sa v triede nachádzajú aj nasledujúce dve pomocné metódy:

- **void prepareDOMAnalyzer(DOMAnalyzer, MediaSpec)** – Metóda, ktorá nakonfiguruje DOM Analýzer.
- **void preparePDFEngine(PDFEngine)** – Metóda, ktorá nakonfiguruje PDF Engine

5.1.3 OpenHTMLtoPDF implementácia

Implementáciu filtra s využitím knižnice OpenHTMLtoPDF bude reprezentovať trieda `PrintReportOpenHTMLtoPDFFilter`[3] projektu `printreportlib`.

Trieda bude dediť z abstraktnej triedy `PrintReportFilter` a implementuje abstraktnú metódu `getPDF`.

V nasledujúcom zozname budú uvedené všetky inicializované objekty potrebné na vygenerovanie PDF dokumentu:

- **outputStream** – Výstupný tok dát, ktorý bude obsahovať pdf dokument vyplnenej šablóny.
- **builder** – Objekt typu PdfRendererBuilder, s pomocou ktorého bude prebiehať generovanie PDF dokumentu z HTML kódu. Nastavíme tu použitie rýchleho módu, vstupné dáta a výstup, kde sa má vygenerovaný PDF dokument uložiť.

Spomenuté inicializované objekty zo zoznamu sa budú nachádzať v triede `getPDF`. V tejto triede prebehne aj generovanie PDF dokumentu s pomocou objektu `builder` a uloženie vygenerovaného dokumentu do `outputStream`, ktorý metóda následne vráti.

[3] Adresa súboru: „src/main/java/sk/jakubmajzlik/printreportlib/PrintReportOpenHTMLtoPDFFilter.java“

Ako bolo už spomenuté v kapitole 3.6, na to aby sme mohli pracovať aj s HTML5, potrebujeme získať w3c DOM Documet za pomoci knižnice Jsoup a ten použiť na generovanie PDF dokumentu. Získanie tohto objektu a nastavenie builderu bude vyzerať nasledovne:

```
1  builder.withW3cDocument(new W3CDom().fromJsoup(Jsoup.parse(servletResponse)),
       url);
```

5.1.4 CharResponseWrapper

Keď server odpovie klientovi, potrebujeme z odpovede dostať HTML kód, avšak keď použijeme bežnú implementáciu `ServletResponse`, nevieme si tento kód získať. Dôvodom je, že metóda `toString()` nám len vráti hash kód objektu. Z tohoto dôvodu si musíme vytvoriť vlastnú implementáciu `ServletResponse`.

Naša vlastná implementácia sa bude nachádzať v triede `CharResponseWrapper`[4], ktorá bude dediť z triedy `HttpServletResponseWrapper`. Táto zdenená trieda nám poskytne implementáciu `ServletResponse`.

Vytvoríme si dva atribúty:

- **PrintWirter printWriter** – objekt, ktorý nám bude zapisovať do bufferu `output`
- **CharArrayWriter output** – buffer, ktorý bude obsahovať odpoveď zo servera

Bude potrebné prepísať implementáciu metódy `toString()`, aby nám vracala obsah bufferu `output` a taktiež metódu `getWriter()` zdedenej triedy, aby vracala nami vytvorený objekt `printWritter`.

5.2 Demonštračná aplikácia

Samostatným projektom bude demonštračná aplikácia. V následujúcich podkapitolách budú popísané naimplementované časti, konfigurácia a spustenie demonštračnej aplikácie.

5.2.1 Kontroléry

Kontroléry budú predstavovať triedy z balíčka `sk.jakubmajzlik.printreport.controller`. Tieto triedy sa budú starať o prichádzajúce požiadavky a budú označené anotáciou `@Controller` a `@RequestMapping("[adresa]")`. Keď použijeme anotáciu `@RequestMapping` na úrovni triedy, tak pri metódach už nemusíme udávať celú adresu požiadavku, ale len adresu konkrétnej úlohy, ktorú majú spracovávať. Pri adrese „/user/edit/5“ bude parameter anotácie na úrovni triedy nastavený na „/user“ a na metóde, ktorá má spracovávať úpravu používateľa, bude nastavený na „/edit/{id}“.

TemplateController

Kontrolér `TemplateController` sa bude starať o požiadavky určené na prácu so šablónami. Parameter anotácie `@RequestMapping` na úrovni triedy bude nastavený na hodnotu „/template“.

[4] Adresa triedy: „src/main/java/sk/jakubmajzlik/printreportlib/CharResponseWrapper.java“

Trieda bude spracovávať nasledujúce adresy požiadaviek:

- **/template/** – Stránka s tabuľkou, ktorá obsahuje všetky dostupné šablóny. O zobrazenie sa postará metóda `showIndex`.
- **/template/add** – Formulár na pridanie novej šablóny. O zobrazenie sa postará metóda `showAddTemplateForm`.
- **/template/add/save** – Validácia a uloženie novej šablóny, o čo sa postará metóda `addTemplate`.
- **/template/view/{category}/{templateName}** – Náhľad šablóny v prehliadači (nie PDF dokumentu), o čo sa postará metóda `showTemplate`.
- **/template/edit/{category}/{templateName}** – Formulár na úpravu šablóny. O zobrazenie a vyplnenie formulára údajmi sa postará metóda `showEditTemplateForm`.
- **/template/edit/save** – Validácia a uloženie zmien úpravy šablóny, o čo sa postará metóda `editTemplate`.
- **/template/delete/{category}/{templateName}** – Vymazanie šablóny, o čo sa postará metóda `deleteTemplate`.

PlaylistController

Kontrolér `PlaylistController` sa bude starať o požiadavky určené na prácu so zoznammi s hudbou. Parameter anotácie `@RequestMapping` na úrovni triedy bude nastavený na hodnotu „/playlist“.

Trieda bude spracovávať nasledujúce adresy požiadaviek:

- **/playlist/** – Úvodná stránka s tabuľkou, ktorá obsahuje všetky zoznamy s hudbou. O zobrazenie sa postará metóda `showIndex`.
- **/playlist/add** – Formulár na pridanie nového zoznamu s hudbou do databázy. O zobrazenie a o vyplnenie šablóny formulára (zoznam všetkých pesničiek, zoznam užívateľov, model) sa postará metóda `showAddPlaylistForm`.
- **/playlist/add/save** – Validácia a uloženie nového zoznamu do databázy, o čo sa postará metóda `addPlaylist`.
- **/playlist/edit/{id}** – Formulár na úpravu zoznamu s hudbou. O zobrazenie a vyplnenie šablóny a formulára údajmi sa postará metóda `showEditPlaylistForm`.
- **/playlist/edit/save** – Validácia a uloženie zmien úpravy zoznamu s hudbou, o čo sa postará metóda `editPlaylist`.
- **/playlist/delete/{id}** – Vymazanie zoznamu s hudbou, o čo sa postará metóda `deletePlaylist`.
- **/playlist/validate** – Zvaliduje sa model s informáciami o tlačovej zostave pre zoznam s hudbou a presmeruje požiadavok na „/make“.
- **/playlist/make** – Vyplnenie šablón údajmi o zoznamoch s hudbou a odoslanie HTML kódu filtra, o čo sa postará metóda `fillTemplatesAndSendToFilter`.

SongController

Kontrolér `SongController` sa bude starať o požiadavky určené na prácu s hudbou. Parameter anotácie `@RequestMapping` na úrovni triedy bude nastavený na hodnotu „/song“.

Trieda bude spracovávať nasledujúce adresy požiadaviek:

- **/song/** – Úvodná stránka s tabuľkou, ktorá obsahuje všetky pesničky. O zobrazenie sa postará metóda `showIndex`.
- **/song/add** – Formulár na pridanie novej pesničky do systému. O zobrazenie a pridanie modelu do šablóny sa postará metóda `showAddSongForm`.
- **/song/add/save** – Validácia a uloženie novej pesničky do databázy, o čo sa postará metóda `addSong`.
- **/song/edit/{id}** – Formulár na úravu pesničky. O zobrazenie a vyplnenie formulára údajmi sa postará metóda `showEditSongForm`.
- **/song/edit/save** – Validácia a uloženie zmien úpravy pesničky, o čo sa postará metóda `editSong`.
- **/song/delete/{id}** – Vymazanie pesničky, o čo sa postará metóda `deleteSong`.
- **/song/validate** – Validácia modelu s informáciami o tlačovej zostave pre pesničky a presmerovanie na „/make“. O toto sa postará metóda `validate`.
- **/song/make** – Vyplnenie šablón údajmi o pesničkách a odoslanie HTML kódu filtra, o čo sa postará metóda `fillTemplatesAndSendToFilter`.

UserController

Kontrolér `UserController` sa bude starať o požiadavky určené na prácu s užívateľmi. Parameter anotácie `@RequestMapping` na úrovni triedy bude nastavený na hodnotu „/user“.

Trieda bude spracovávať nasledujúce adresy požiadaviek:

- **/user/** – Stránka s tabuľkou, ktorá obsahuje všetkých užívateľov. O zobrazenie sa postará metóda `showIndex`.
- **/user/add** – Formulár na pridanie nového užívateľa. O zobrazenie sa postará metóda `showAddUserForm`.
- **/user/add/save** – Validácia a uloženie nového užívateľa, o čo sa postará metóda `addUser`.
- **/user/edit/{id}** – Formulár na úpravu existujúceho užívateľa. O zobrazenie a vyplnenie formulára údajmi sa postará metóda `showEditUserForm`.
- **/user/edit/save** – Validácia a uloženie zmien úpravy užívateľa, o čo sa postará metóda `editUser`.
- **/user/delete/{id}** – Vymazanie užívateľa, o čo sa postará metóda `deleteUser`.
- **/user/validate** – Validácia modelu s informáciami o tlačovej zostave pre užívateľa a presmerovanie na „/make“. O toto sa postará metóda `validate`.
- **/user/make** – Vyplnenie šablón údajmi o užívateľoch a odoslanie HTML kódu filtra, o čo sa postará metóda `fillTemplatesAndSendToFilter`.

5.2.2 Služby

Služby predstavujú triedy, ktoré separujú biznis logiku aplikácie od kontrolérov. Tieto triedy sa nachádzajú v balíčku `sk.jakubmajzlik.printreport.service` a budú označené anotáciou `@Service`.

PlaylistService

Služba `PlaylistService` bude vykonávať operácie spojené so zoznamami s pesničkami.
Metódy, ktoré obsahuje táto trieda:

- **PlaylistEntity getPlaylistByID(int)** – Vráti entitnú triedu zoznamu s hudbou na základe ID.
- **PlaylistAddEditModel getPlaylistByIDAsModel(int)** – Vráti model zoznamu s hudbou na základe ID.
- **void savePlaylist(PlaylistAddEditModel)** – Uloží zoznam pesničiek do databázy. Ak tento zoznam v databáze už existuje, tak sa len upraví.
- **List<PlaylistEntity> getllPlaylists()** – Vráti všetky zoznamy s hudbou, ktoré sú v databáze.
- **void deletePlaylist(PlaylistEntity)** – Vymaže daný zoznam s hudbou z databázy.

SongService

Služba `SongService` bude vykonávať operácie spojené s pesničkami.
Metódy, ktoré obsahuje táto trieda:

- **SongEntity getSongByID(int)** – Vráti entitnú triedu pesničky na základe ID.
- **SongModel getSongByIDAsModel(int)** – Vráti model pesničky na základe ID.
- **void saveSong(SongEntity)** – Uloží pesničku do databázy. Ak táto pesnička v databáze už existuje, tak sa len upraví.
- **List<SongEntity> getAllSongs()** – Vráti všetky pesničky, ktoré sú v databáze.
- **void deleteSong(SongEntity)** – Vymaže pesničku z databázy.

TemplateService

Služba `PlaylistService` bude vykonávať operácie spojené so šablónami.
Metódy, ktoré obsahuje táto trieda:

- **void saveTemplate(TemplateModel)** – Uloží HTML a CSS kód šablóny na určité miesto[5].
- **TemplateModel getTemplateModel(String, String)** – Vráti model šablóny na základe kategórie a názvu.
- **List<TemplateModel> getAllTemplates()** – Vráti modely všetkých šablón.

[5]Miesto, kde sa sablóny budú ukladať, určuje konštanta PATH z triedy TemplatePath.

- **List<String> getAllTemplateNamesOfCategory(String)** – Vráti zoznam mien všetkých šablón z určitej kategórie.
- **void deleteTemplate(TemplateModel)** – Vymaže danú šablónu.

UserSerice

Služba `PlaylistService` bude vykonávať operácie spojené s užívateľmi.

Metódy, ktoré obsahuje táto trieda:

- **UserEntity getUserByID(int)** – Vráti entitnú triedu užívateľa na základe ID.
- **UserAddEditModel getUserByIDAsModel(int)** – Vráti model užívateľa na základe ID.
- **void saveUser(UserEntity)** – Uloží užívateľa do databázy. Ak tento užívateľ v databáze už existuje, tak sa len upraví.
- **List<UserEntity> getAllUsers()** – Vráti všetkých užívateľov z databázy.
- **void deleteUser(UserEntity)** – Vymaže užívateľa z databázy.

5.2.3 Repozitáre

Repozitáre, sa budú starať o komunikáciu s databázovou tabuľkou. Budeme potrebovať metódy na získanie, uloženie, pridanie a vymazanie záznamov z tabuľky. Pri implementácii repozitára nám zas a znova pomôže Spring Boot. Spring Boot obsahuje už naimplementované repozitáre, ktoré môžeme použiť v našom projekte.

Pre každú entitnú triedu budeme mať jeden repozitár. Repozitár si vytvoríme ako rozhranie, ktoré označíme anotáciou `@Repository` a rozšírime ho o rozhranie `CrudRepository`, ktoré nám poskytne Spring Boot. Keď budeme chcieť použiť náš repozitár, stačí nám pri deklarácií repozitára použiť anotáciu `@Autowired`, ktorá zabezpečí nájdenie vhodnej implementácie rozhrania repozitára.

V demonštračnej aplikácii budú použité tri repozitáre:

- **PlaylistRepository** – Repozitár pre prácu s tabuľkou `playlists`.
- **SongRepository** – Repozitár pre prácu s tabuľkou `songs`.
- **UserRepository** – Repozitár pre prácu s tabuľkou `users`.

5.2.4 Entitné triedy

Entitné triedy budú reprezentovať jeden riadok z tabuľky a na základe nich sa vytvorí tabuľka, ktorej záznam reprezentujú. Tieto triedy sa budú nachádzať v balíčku `sk.jakubmajzlik.printreport.entity`. Aby sme označili danú triedu, že je entita, použijeme anotáciu `@Entity`. Názov tabuľky bude odvodený od názvu triedy, no ak si chceme prispôsobiť tento názov použijeme anotáciu `@Table` s parametrom `name`.

Atribúty budú reprezentovať jednotlivé stĺpce tabuľky a názvy stĺpcov budú odvodené od názvov týchto atribútov. Ak by sme si chceli prispôsobiť názov stĺpca, použijeme anotáciu `@Column` s parametrom `name`.

Pre označenie primárneho kľúča použijeme anotáciu `@Id`. Chceme, aby sa nám primárny kľúč sám inkrementoval pri vytvorení nového záznamu, tak navyše použijeme na primárny kľúč anotáciu `@GeneratedValue` s parametrom `strategy = GenerationType.IDENTITY`.

Každá entitná trieda bude mať vytvorené gettery, settery pre jednotlivé atribúty a prepísanú metódu `toString()`.

PlaylistEntity

Táto entitná trieda bude reprezentovať záznam z tabuľky `playlists`. Bude obsahovať nasledujúce atribúty:

- **id** – Primárny kľúč typu `Integer`.
- **name** – Meno zoznamu s hudbou typu `String`.
- **description** – Popis zoznamu s hudbou typu `String`.
- **lastUpdated** – Dátum poslednej úpravy typu `Date`. Tento atribút bude označený anotáciou `@UpdateTimestamp`, ktorá zabezpečí, že keď sa upraví záznam o zozname s hudbou, tak sa do tohto atribútu uloží aktuálny čas a dátum.
- **owner** – Vlastník zoznamu s hudbou typu `UserEntity`. Tento atribút bude označený anotáciou `@ManyToOne`, ktorá označí vzťah aktuálnej entity a entity `UserEntity` ako Many-To-One.
- **listOfSongs** – Zoznam pesničiek typu `List<SongEntity>`. Tento atribút bude označený anotáciou `@ManyToMany`, ktorá označí vzťah aktuálnej entity a entity `SongEntity` ako Many-To-Many.

SongEntity

Nasledujúca entita bude reprezentovať záznam z tabuľky `songs`. Bude obsahovať nasledujúce atribúty:

- **id** – Primárny kľúč typu `Integer`.
- **songName** – Meno pesničky typu `String`.
- **artistName** – Meno interpreta pesničky typu `String`.
- **albumName** – Meno albumu typu `String`.
- **duration** – Dĺžka pesničky typu `String`[6].

UserEntity

Nasledujúca entita bude reprezentovať záznam z tabuľky `users`. Bude obsahovať nasledujúce atribúty:

- **id** – Primárny kľúč typu `Integer`.
- **firstName** – Meno používateľa `String`.

[6]Typ `String` by bolo ideálne zameniť za `Integer`, aby sa lepšie pracovalo s týmto časom, no v demonštračnej aplikácii nebudeme s ním nič robiť a pre zjednodušenie použijeme práve tento typ.

- **lastName** – Priezvisko používateľa typu `String`.
- **email** – Email používateľa typu `String`.
- **userPlaylists** – Zoznam playlistov, ktoré používateľ vlastní. Tento atribút bude označený anotáciou `@OneToMany`, ktorá označí vzťah aktuálnej entity a entity `PlaylistEntity` ako One-To-Many

5.2.5 Modely

Modely sú jednoduché POJO triedy z balíčka `sk.jakubmajzlik.printreport.model`, ktoré budú slúžiť na prenos dát z formulárov, na vyplnenie šablón stránky, alebo tlačovej zostavy. Niektoré atribúty modelov budú označené anotáciami, ktoré určia aké hodnoty sú validné pre daný atribút. Ak v kontroléri zistíme, že model neprešiel validáciou, môžeme sa vrátiť naspäť na formulár a opraviť chybné údaje.

Následne si popíšeme jednotlivé použité modely v projekte `printreportis`.

PlaylistAddEditModel

Model obsahujúci dáta o zozname s pesničkami používaný pri úprave a pridaní zoznamu.

Atribúty triedy:

- **id** – Primárny kľúč pesničky typu `int`.
- **songsId** – Pole primárnych kľúčov pesničiek, ktoré obsahuje daný zoznam. Keďže komponent checkbox pri zaškrtnutí vracia id ako reťazec, typ pola musí byť pole reťazcov.
- **name** – Názov zoznamu s pesničkami typu `String`. Pri tomto atribúte sa bude kontrolovať dĺžka reťazca, ktorá musí pozostávať aspoň z jedného znaku. Kontrola sa bude vykonávať s pomocou anotácie `@NotBlank`.
- **description** – Popis zoznamu s hudbou typu `String`. Nasledujúci atribút je voliteľný a nevykonávajú sa na ňom žiadné kontroly.
- **ownerId** – Primárny kľúč vlastníka zoznamu s hudbou typu `int`.

PlaylistModel

Model obsahujúci dáta o zozname s pesničkami používaný na doplnenie dát do šablóny.

Atribúty triedy:

- **playlistName** – Názov zoznamu s hudbou typu `String`.
- **ownerName** – Meno vlastníka zoznamu s hudbou typu `String`.
- **description** – Popis zoznamu s hudbou typu `String`.
- **songs** – Zoznam pesničiek, ktoré obsahuje aktuálny zoznam s hudbou typu `List<SongModel>`.
- **lastUpdated** – Dátum a čas poslednej úpravy typu `String`.

PrintModel

Tento model bude obsahovať potrebné dáta pre vyplnenie šablóny.

Atribúty triedy:

- **templateName** – Názov šablóny typu `String`. Tento atribút je povinný a bude sa kontrolovať s pomocou anotácie `@NotBlank`.
- **IDs** – Pole primárnych kľúčov, pre ktoré sa má vytvoriť zostava. Keďže komponent checkbox pri zaškrtnutí vracia id ako reťazec, typ pola musí byť pole reťazcov.

SongModel

Model obsahujúci dáta o pesničke. tento model sa bude využívať vo formulároch aj pre vyplnenie šablón.

Všetky atribúty modelu sú povinné a budú sa kontrolovať s pomocou anotácie `@NotBlank`.

Atribúty triedy:

- **id** – Primárny kľúč pesničky typu `int`.
- **songName** – Názov pesničky typu `String`.
- **artistName** – Názov umelca, alebo umelcov typu `String`.
- **albumName** – Názov albumu typu `String`.
- **duration** – Dĺžka pesničky typu `String`.

TemplateModel

Model bude obsahovať dáta o šablóne.

Atribúty triedy:

- **name** – Názov šablóny typu `String`. Tento údaj je povinný a bude sa kontrolovať anotáciou `@NotBlank`.
- **content** – HTML kód šablóny typu `String`. Nasledujúci atribút je nepovinný.
- **category** – Kategória, do ktorej patrí šablóna typu `String`. Tento údaj je povinný a bude sa kontrolovať anotáciou `@NotBlank`.
- **style** – CSS kód pre šablónu typu `String`. Tento atribút je nepovinný.

UserAddEditModel

Model bude obsahovať dáta o používateľovi pri pridaní a úprave údajov užívateľa.

Atribúty triedy:

- **id** – Primárny kľúč užívateľa typu `int`.
- **firstName** – Meno užívateľa typu `String`. Nasledujúci údaj je povinný a bude sa kontrolovať s pomocou anotácie `@NotBlank`.
- **lastName** – Priezvisko užívateľa typu `String`. Nasledujúci údaj je povinný a bude sa kontrolovať s pomocou anotácie `@NotBlank`.
- **email** – Email používateľa typu `String`. Nasledujúci údaj je povinný a bude sa kontrolovať s pomocou anotácie `@NotBlank` a anotáciou `@Email`.

UserModel

Model bude obsahovať dáta o používateľovi pre vyplnenie šablóny.

Atribúty triedy:

- **firstName** – Meno užívateľa typu `String`.
- **lastName** – Priezvisko užívateľa typu `String`.
- **email** – Email užívateľa typu `String`.
- **playlists** – Zoznam zoznamov s hudbou, ktoré užívateľ vlastní, typu `List<PlaylistModel>`.

5.2.6 Integrácia filtra

Filter bude v demonštračnej aplikácií integrovaný s pomocou anotácie `@WebFilter`. V projekte bude vytvorená nová trieda, na ktorú použijeme túto anotáciu. Trieda bude dediť triedu niektorého z implementovaných filtrov z knižnice `printreportlib`. Nastavíme parameter anotácie `urlPatterns`, ktorý určí aké adresy sa majú filtrovať.

Parameter bude nastavený na nasledujúce hodnoty:

- **/user/make** – Adresa, ktorá bude vracať HTML kód vyplnenej šablóny z kategórie `user`.
- **/playlist/make** – Adresa, ktorá bude vracať HTML kód vyplnenej šablóny z kategórie `playlist`.
- **/song/make** – Adresa, ktorá bude vracať HTML kód vyplnenej šablóny z kategórie `song`.

5.2.7 Konfigurácia

Väčšina nastavení demonštračnej aplikácie sa nachádza v súbore `application.properties`[7]. V nasledujúcom zozname budú popísané nastavenia z tohto súboru:

- **spring.mvc.view.prefix** – Nastavenie prefixu pre pohľady. Hodnota kľúča bude nastavená na hodnotu „/WEB-INF/view/“.
- **spring.mvc.view.suffix** – Nastavenie suffixu pre pohľady. Hodnota kľúča bude nastavená na hodnotu „.jsp“.
- **spring.jpa.hibernate.ddl-auto** – Nastavenie akcie pri spustení aplikácie súvisiacej s databázou. Hodnota kľúča bude nastavená na hodnotu „update“, čo nám zaručí pri spustení aplikácie, že sa vygenerujú, alebo aktualizujú tabuľky v databáze na základe entitných tried.
- **spring.datasource.tomcat.connection-properties** – Nastavenie vlastnosti zabudovaného Tomcat serveru. Hodnota kľúča bude nastavená na hodnotu „useUnicode=true;characterEncoding=utf-8;“.

[7] Adresa súboru: „src/main/resources/application.properties“

- **spring.datasource.url** – Nastavenie adresy pripojenia k databáze. Hodnota kľúča bude nastavená na hodnotu „jdbc:mysql://localhost:3306/printreport?useSSL=false &useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false &allowPublicKeyRetrieval=true&serverTimezone=UTC“.
- **spring.datasource.sqlScriptEncoding** – Nastavenie kódovania sql scriptov. Hodnota kľúča bude nastavená na hodnotu „UTF-8“.
- **spring.datasource.username** – Užívateľské meno pre prihlásenie do databázy. Hodnota kľúča bude nastavená na hodnotu „pruser“.
- **spring.datasource.password** – Heslo pre prihlásenie do databázy. Hodnota kľúča bude nastavená na hodnotu „prpassword“.

Nastavenia, ktoré určia, kde sa nachádzajú statické súbory použité v aplikácií a šablónach tlačových zostáv sa bude nachádzať v konfiguračnej triede `ResourcesConfig.java`[8].

Táto trieda bude označená anotáciou `@Configuration` a zároveň bude rozširovať triedu `WebMvcConfigurerAdapter`. V tejto triede prepíšeme metódu `addResourceHandlers`, kde nastavíme lokácie statických súborov použitých vo webovej stránke a lokácie statických súborov použitých pri tvorbe šablóny tlačovej zostavy.

5.2.8 Spustenie aplikácie

Pred spustením je potrebné vytvoriť MySQL databázu a používateľa. Poslúži nám na to SQL script `db_init.sql`[9]. Testovacie dáta môžeme vložiť do databázy za pomoci scriptu `test_data.sql`[10].

Na spustenie demonštračenj aplikácie použijeme nástroj Maven. Z koreňovej zložky projektu `printreportis` spustíme z konzoly príkaz `mvn spring-boot:run`. Aplikácia sa preloží, spustí a budeme môcť pristúpiť k nej z webového prehliadača na adrese `http://localhost:8080/`.

Ak by bolo číslo portu už obsadené, môžeme ho zmeniť pridaním `server.port=[číslo portu]` do súboru `application.properties`[11].

[8] Adresa triedy: „src/main/java/sk/jakubmajzlik/printreport/config/Resources.java“
[9] Adresa scriptu: „src/main/sql/db_init.sql“
[10] Adresa scriptu: „src/main/sql/test_data.sql“
[11] Adresa súboru: „src/main/resources/application.properties“

Kapitola 6

Testovanie

Nasledujúca kapitola sa bude zaoberať testovaním demonštračnej aplikácie a porovnaním PDF dokumentov vygenerovanými knižnicami CSSBoxPDF a OpenHTMLToPDF.

Testy som vykonával ručne, pri niektorých prípadoch sa to ani inak nedalo, napríklad generovanie PDF dokumentov.

Testoval som:

- Funkčnosť pridávania, úpravy a vymazania záznamu zo systému
- Validáciu formulárov
- Generovanie PDF dokumentov

6.1 Pridávanie, úprava a mazanie záznamov zo systému

V demonštračnej aplikácii sa nachádzajú tri druhy záznamov, ktoré treba pridávať, upravovať, alebo mazať. Patria medzi ne záznamy o užívateľoch, záznamy o pesničkách a záznamy o zoznamoch s hudbou. Testoval som funkčnosť spomenutých operácií cez užívateľské rozhranie, pridávanie šablón a validáciu formulárov. V nasledujúcom zozname sa nachádza postup ako som testoval jednotlivé operácie[1]:

- Pokus o pridanie prázdneho záznamu
- Pokus o pridanie záznamu s niektorými prázdnymi povinnými poliami
- Pridanie správne vyplneného záznamu
- Pokus o úpravu záznamu s tým, že boli vymazané niektoré povinné údaje
- Úprava záznamu so správnymi údajmi
- Vymazanie záznamu

6.2 Generovanie PDF dokumentov

Generovanie PDF dokumentov som testoval na troch vytvorených šablónach[2]. Postupne som skúšal vygenerovať dokument pre jeden záznam, pre viacero záznamov a pre všetky

[1] Pri šablónach som testoval aj ručné pridávanie.

[2] Šablóny je možné nájsť na adrese: „src/main/webapp/WEB-INF/print-report-templates“

záznamy z danej tabuľky. Následne som zmenil implementáciu filtra a postup som zopakoval znovu.

Prvá šablóna, „users-playlists“ je z kategórie „user“. V dokumente vygenerovanom na základe tejto šablóny sa nachádzajú informácie o tom, aké zoznamy s hudbou daný užívateľ vlastní. V šablóne sa nachádza meno užívateľa a postupne vykreslené tabuľky so všetkými zoznamami s hudbou.

Druhá šablóna, „playlist-view“, je z kategórie „playlist“. Dokument vygenerovaný za pomoci tejto šablóny obsahuje informácie o danom zozname s hudbou, ako názov, popis, dátum poslednej úpravy a pesničky, ktoré daný zoznam obsahuje. Šablóna pozostáva z tmavého pozadia, vycentrovaného názvu a popisu zoznamu s hudbou. Pod popisom sa z ľavej strany nachádza autor a dátum poslednej úpravy a vycentrovaná tabuľka s pesničkami, ktoré obsahuje daný zoznam.

Tretia šablóna, „message“, je z kategórie „user“. Vo vygenerovanom dokumente sa bude nachádzať správa pre užívateľa. Text správy bude zarovnaný do bloku a na spodnej časti stránky sa bude nachádza priestor pre podpis a dátum vytvorenia dokumentu. Na spodnej časti dokumentu sa bude nachádzať aj obrázok noty.

6.3 Porovnanie vygenerovaných dokumentov

Pri vygenerovaných dokumentoch som si na prvý pohľad všimol rozdiel pri veľkosti fontov. Knižnica CSSBoxPDF vykresluje znaky menšie. Pri použití OpenHTMLtoPDF je veľkosť znakov približne rovnaká ako pri náhľade HTML kódu šablóny.

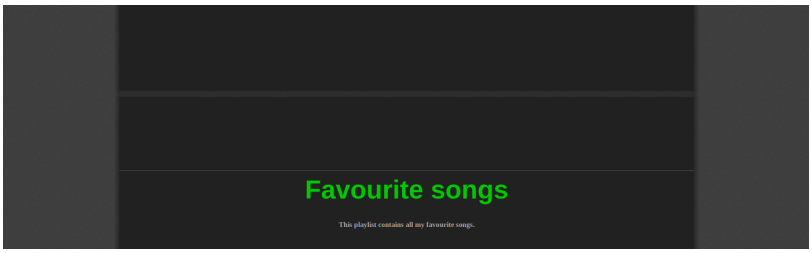
Keď som porovnal vygenerované PDF dokumenty z prvej šablóny (príloha A.1 a A.2) tak som zistil, že jediný rozdiel je vo veľkosti písma. Obe knižnice bez problémov vedia vykresliť tabuľky, vycentrovať ich a nastaviť okraje.

Podobne vyzerajú aj vygenerované dokumenty z druhej šablóny (príloha A.3 a A.4). Rovnako ako pri prvej šablóne je rozdiel vo veľkosti fontov, čo zapríčiňuje aj rozdielnú veľkosť tabuliek. V CSS súbore šablóny nie je určená šírka tabuľky, len je vycentrovaná na stred dokumentu, tak isto nie je určená ani veľkosť písma. Otestoval som aj ako sa budú chovať jednotlivé implementácie filtrov, keď počet riadkov tabuľky bude prevyšovať veľkosť strany (príloha A.5 a A.6). Pri oboch implemenáciach tabuľka z ďalšej strany nadväzuje na tabuľku z predchádzajúcej. Pri použití knižnice CSSBoxPDF sa na ďalšej strane nachádza okraj a až potom tabuľka, pri použití knižnice OpenHTMLtoPDF je ďalšia strana bez horného okraja.

Vygenerované dokumenty z tretej šablóny (príloha A.7 a A.8) vyzerajú pri použití oboch knižníc rovnako, až na veľkosť písma. Text je zarovnaný do bloku, nadpis vycentrovaný, obrázok vykreslený. Jediným problémom, ktorý môžeme vidieť je, že knižnica OpenHTMLtoPDF nevie vykresliť písmena s mäkčeňom, dĺžne mu nerobia problém. Pri tejto šablóne som musel nastaviť výšku a šírku stránky, aby som mohol napozíciovať miesto pre podpis a obrázok. Veľkosť stránky som nastavil v CSS na veľkosť A4 s využitím jednotky `mm`. Knižnica OpenHTMLtoPDF sa s tým vysporiadala v poriadku, no pri vygenerovaní dokumentu s pomocou knižnice CSSBoxPDF bola výsledná stránka menšia ako stránka vo formáte A4. Z tohoto dôvodu som musel prispôsobiť veľkosť stránky pre túto knižnicu.

6.4 Zistené nedostatky

Pri použití filtra, ktorý používal knižnicu CSSBoxPDF sa vyskytol problém pri použití CSS vlastnosti `page-break-before`, alebo `page-break-after`. Posledný element z predchádzajúcej stránky sa roztiahol až na začiatok ďalšej stránky (obrázok 6.1). Ak som nepoužil zlom strany, tak stránka vyzerala normálne. Pravdepodobne sa jedná o chybu knižnice.



Obr. 6.1: Pretečenie predchádzajúcej stránky pri použití zalomenia stránky

Väčšinu problémov s diakritikou som vyriešil pridaním tagu s metadátami o kódovaní znakov `<meta charset="UTF-8">` do šablón, ale knižnica OpenHTMLtoPDF má aj naďalej problémy s niektorými znakmi. Vyriešil som to pridaním fontu, ktorý obsahuje dané znaky, cez CSS súbor s pomocou `@font-face`. Riešením by mohlo byť aj nahradenie problémových znakov príslušnými HTML entitami.

6.5 Zhodnotenie výsledkov

Obe knižnice, až na pár nedostatkov, generujú testovacie dokumenty v poriadku. Všimol som si rozdiel vo veľkostiach či už písma, alebo elementov dokumentu, čo môžem vyriešiť prispôsobením šablóny pre danú knižnicu. Obe knižnice vykreslia obrázky, tabuľky a naformátujú text v poriadku. Aktuálne, knižnica CSSBoxPDF má problém s vynútenými zalomeniami strán, z tohto dôvodu je pre viac stranové zostavy lepšie použiť filter s knižnicou OpenHTMLtoPDF.

Kapitola 7

Záver

Zámerom tejto práce bolo navrhnúť riešenie tvorby PDF dokumentov na základe šablóny v HTML a CSS, ktoré by bolo vhodné na použitie vo webových aplikáciách postavených na platforme Java EE. Základom riešenia sa stala technológia filtrov, ktoré slúžia na manipuláciu komunikácie medzi klientom a serverom na určitých adresách. Použitie tejto technológie nám splní podmienku zo zadania, a to jednoduchú integráciu do informačného systému, keďže na integráciu nám stačí len pár riadkov kódu. Tento filter zachytí odpoveď s HTML kódom vyplnenej šablóny zo servera, vygeneruje z nej PDF dokument s pomocou knižnice na generovanie PDF dokumentov a odošle ho klientovi. To aká knižnica sa použije závisí od filtra, ktorý bol použitý pri integrácií.

Nie všetky knižnice sú vhodné na generovanie všetkých dokumentov, preto je možné pri integrácii vybrať implementáciu filtra. V knižnici sa nachádzajú dve implementácie filtrov, a to CSSBoxPDF a OpenHTMLtoPDF implementácia. Poprípade je možne si naimplementovať vlastný filter s inou knižnicou na generovanie PDF dokumentov.

Vo vývoji knižnice môžeme pokračovať pridaním ďalších implementácií filtrov, prípadne pridaním podpory ďalších formátov výstupu.

Literatúra

[1] Big Faceless Organization. *Java Report Generator - Convert XML to PDF with BFO* [online]. 2020 [cit. 2020-3-28]. Dostupné z: https://bfo.com/products/report/.

[2] Burget, R. *CSSBox - Java HTML rendering engine* [online]. 2019 [cit. 2019-12-1]. Dostupné z: http://cssbox.sourceforge.net/.

[3] Chung, K. man. *JavaServer Pages^{TM} Specification* [online]. JSR-000245. Java Community Proccess, máj 2013. Maintenance Release 3. Dostupné z: https://jcp.org/en/jsr/detail?id=245.

[4] Horstmann, C. S. a Cornell, G. *Core Java Volume I–Fundamentals*. 9. vyd. Prentice Hall, 2013. 23 s. ISBN 0-13-708189-8.

[5] Jaspersoft. *JasperReports® Library | Jaspersoft Community* [online]. 2020 [cit. 2020-3-19]. Dostupné z: https://community.jaspersoft.com/project/jasperreports-library.

[6] Jendrock, Eric and Cervera-Navarro, Ricardo and Evans, Ian and Haase, Kim and Markito, William. *The Java EE tutorial: release 7* [online]. 7. vyd. September 2014. 17-1;17-7 s. Dostupné z: https://docs.oracle.com/javaee/7/JEETT.pdf.

[7] Oracle Corporation. *Java Platform, Micro Edition (Java ME)* [online]. 2019 [cit. 2019-11-02]. Dostupné z: https://www.oracle.com/java/technologies/javameoverview.html.

[8] Oracle Corporation. *Java^{TM} EE at a Glance* [online]. 2019 [cit. 2019-11-02]. Dostupné z: https://www.oracle.com/java/technologies/java-ee-glance.html.

[9] Pivotal Software. *Embedded Web Servers* [online]. 2020 [cit. 2020-3-05]. Dostupné z: https://docs.spring.io/spring-boot/docs/2.1.9.RELEASE/reference/html/howto-embedded-web-servers.html.

[10] Stack Exchange Inc. *Most Popular Technologies* [online]. 2020 [cit. 2020-3-19]. Dostupné z: https://insights.stackoverflow.com/survey/2019#technology.

[11] W3C ®. *Simple selectors* [online]. 2019 [cit. 2019-11-02]. Dostupné z: https://www.w3.org/TR/2018/REC-selectors-3-20181106/#simple-selectors.

[12] Wright, P. et al. *Open HTML to PDF* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://github.com/danfickle/openhtmltopdf.

Príloha A

Porovnanie vygenerovaných dokumenntov

Jakub Majzlik's playlists

My playlist

Description: Playlist with my favourite songs

Title	Artist	Album	Duration
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56

WIP Playlist

Description: empty playlist

Title	Artist	Album	Duration

Obr. A.1: Dokument vygenerovaný kinižnicou OpenHTMLtoPDF na základe šablóny „users-playlists“

Jakub Majzlik's playlists

My playlist

Description: Playlist with my favourite songs

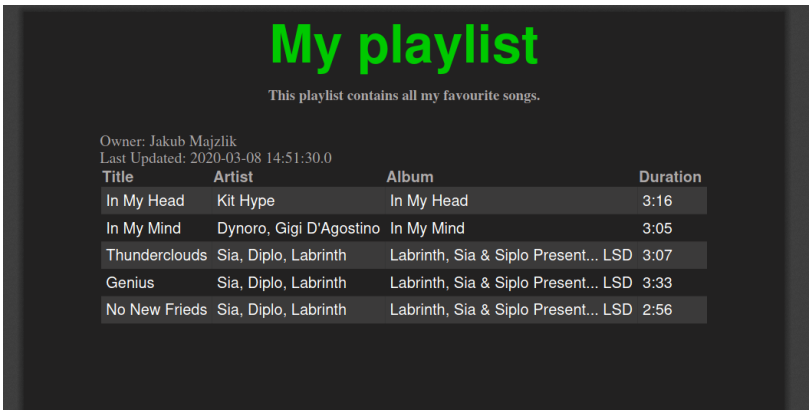
Title	Artist	Album	Duration
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56

WIP Playlist

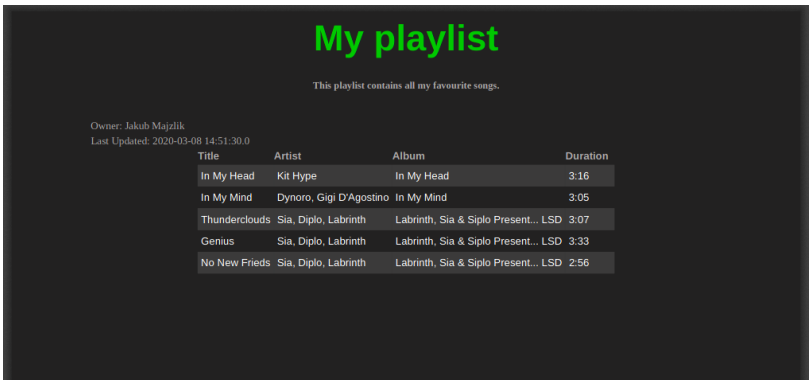
Description: empty playlist

Title	Artist	Album	Duration

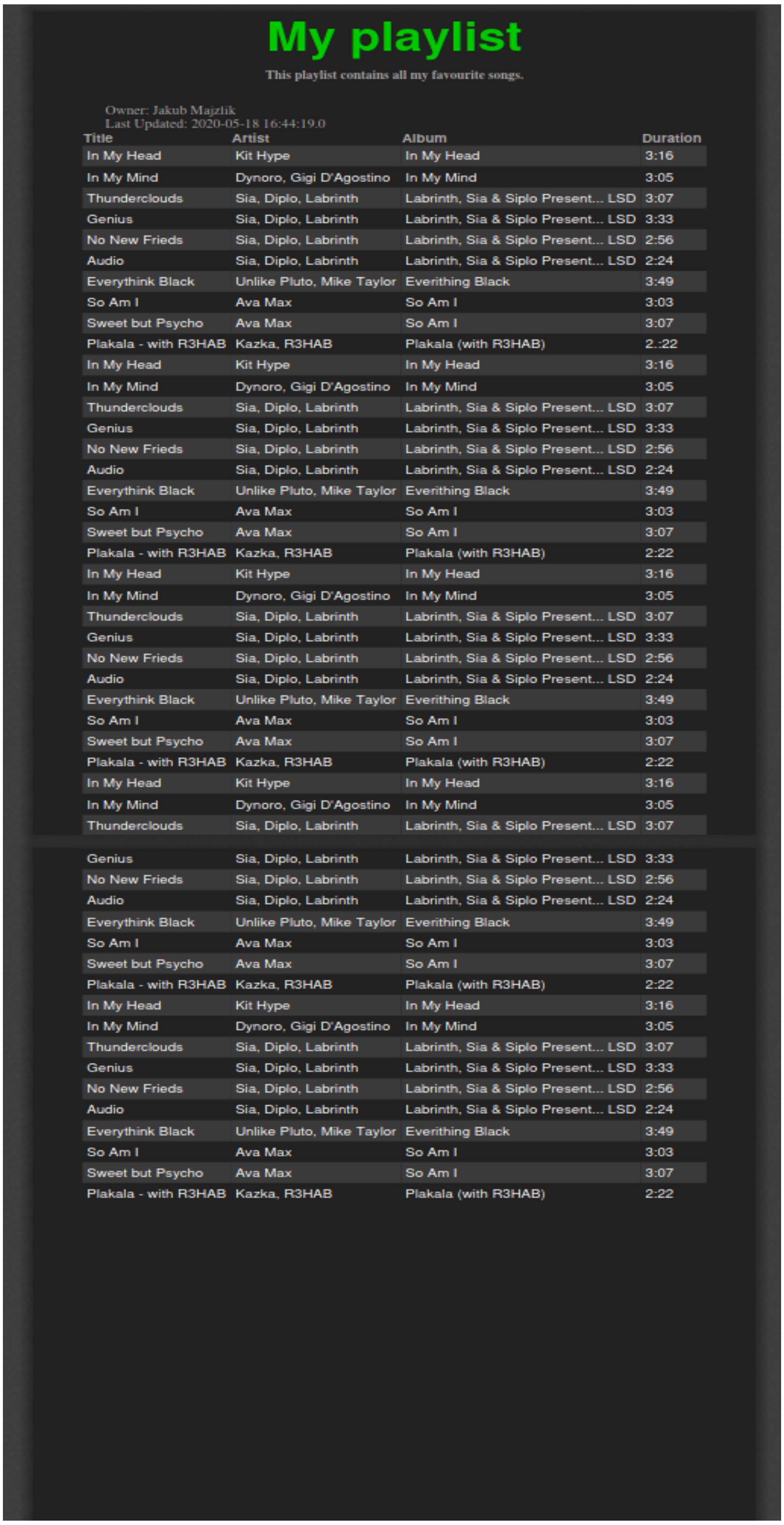
Obr. A.2: Dokument vygenerovaný kinižnicou CSSBoxPDFF na základe šablóny „users-playlists“



Obr. A.3: Dokument vygenerovaný kinižnicou OpenHTMLtoPDF na základe šablóny „playlist-view“



Obr. A.4: Dokument vygenerovaný kinižnicou CSSBoxPDF na základe šablóny „playlist-view“



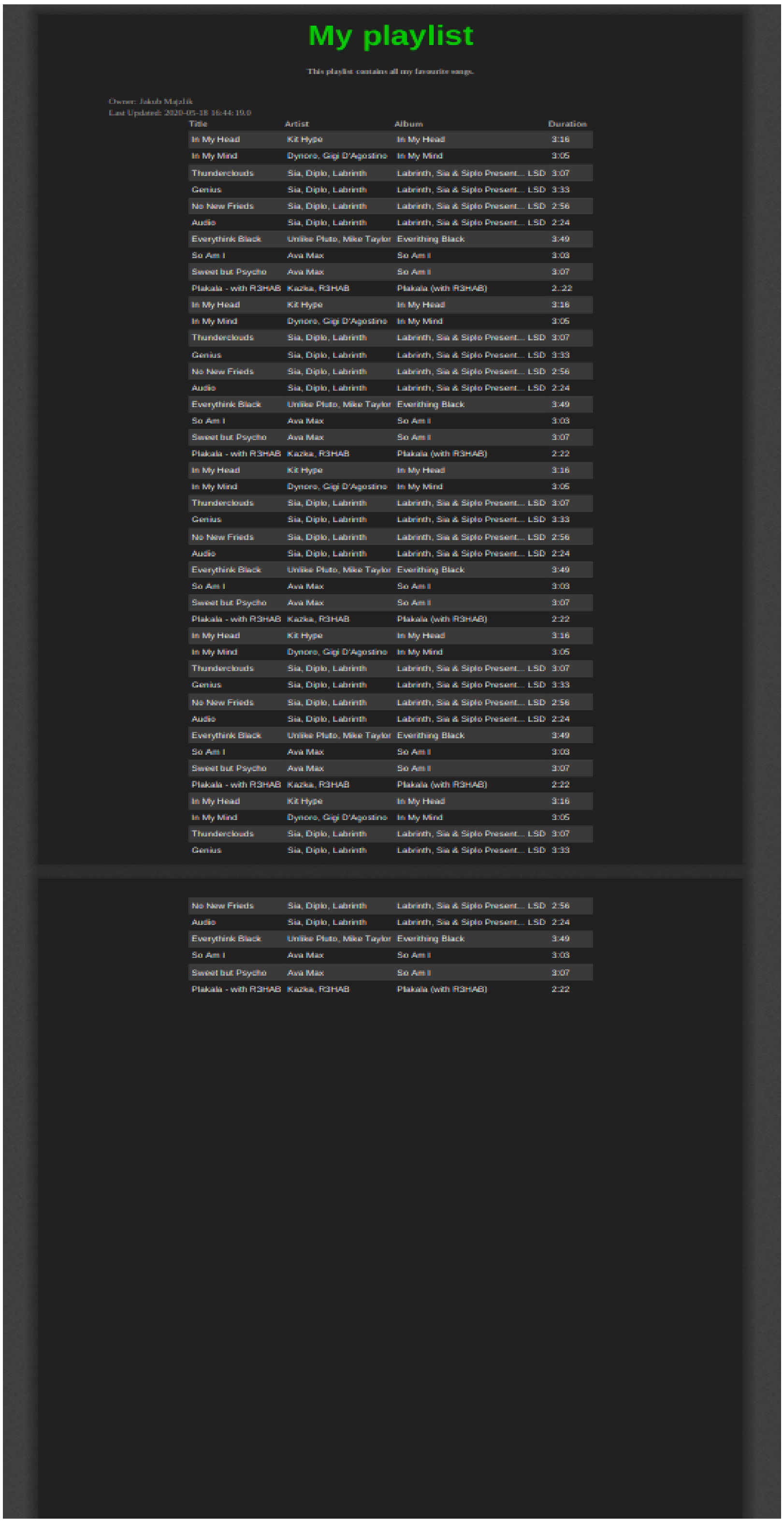
My playlist

This playlist contains all my favourite songs.

Owner: Jakub Majzlik
Last Updated: 2020-05-18 16:44:19.0

Title	Artist	Album	Duration
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everithing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2.:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everithing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everithing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everithing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everithing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22

Obr. A.5: Dokument vygenerovaný kinižnicou OpenHTMLtoPDF na základe šablóny „playlist-view“ s tabuľkou cez dve strany



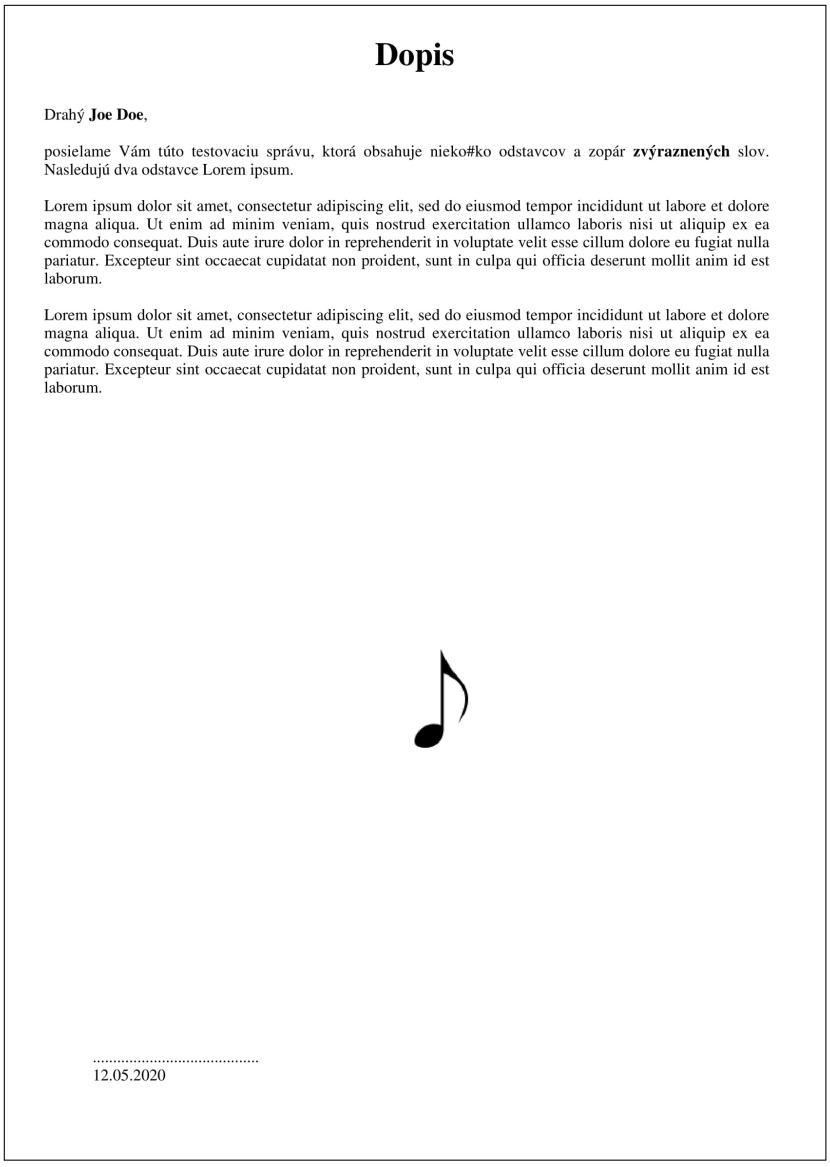
My playlist

This playlist contains all my favourite songs.

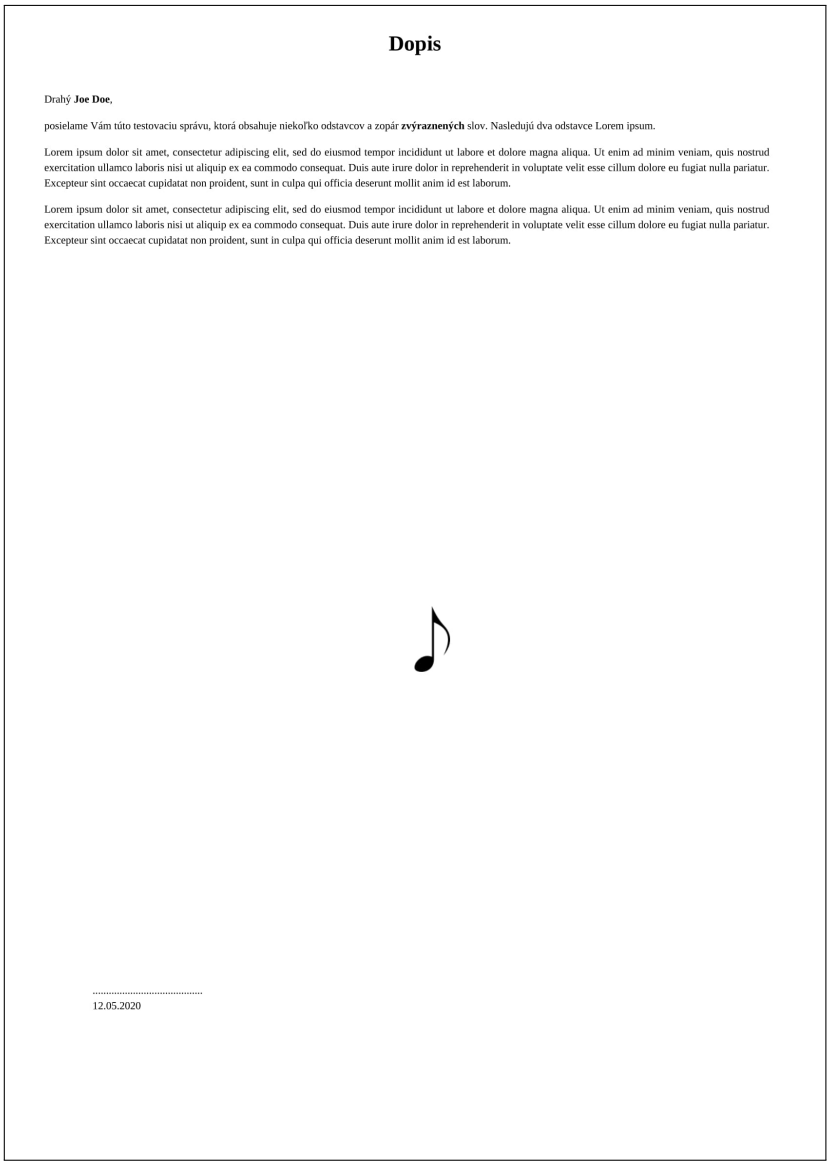
Owner: Jakub Majzlík
Last Updated: 2020-05-18 16:44:19.0

Title	Artist	Album	Duration
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everthing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2..22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everthing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everthing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everthing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22
In My Head	Kit Hype	In My Head	3:16
In My Mind	Dynoro, Gigi D'Agostino	In My Mind	3:05
Thunderclouds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:07
Genius	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	3:33
No New Frieds	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:56
Audio	Sia, Diplo, Labrinth	Labrinth, Sia & Siplo Present... LSD	2:24
Everythink Black	Unlike Pluto, Mike Taylor	Everthing Black	3:49
So Am I	Ava Max	So Am I	3:03
Sweet but Psycho	Ava Max	So Am I	3:07
Plakala - with R3HAB	Kazka, R3HAB	Plakala (with R3HAB)	2:22

Obr. A.6: Dokument vygenerovaný kiničnicou CSSBoxPDF na základe šablóny „playlist-view“ s tabuľkou cez dve strany



Obr. A.7: Dokument vygenerovaný kinižnicou OpenHTMLtoPDF na základe šablóny „message“



Obr. A.8: Dokument vygenerovaný kinižnicou CSSBoxPDF na základe šablóny „message“

Príloha B

Obsah CD

Na priloženom CD sa bude nachádzať:

- Priečinok „printreportlib“ so súbormi projektu `printreportlib`
- Priečinok „printreportis“ so súbormi projektu `printreportis`
- Súbor „readme.txt“ s informáciami na spustenie projektu
- Priečinok „doc“ so zdrojovými súbormi bakalárskej práce