ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Bakalářská práce

Návrh a implementace webového informačního systému

Design and implementation of a web-based information system

Jan Tlačil

Plzeň

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma

„Návrh a implementace webového informačního systému“

vypracoval samostatně pod odborným dohledem vedoucího bakalářské práce za použití pramenů uvedených v přiložené bibliografii.

Plzeň dne 20.4.2022 v. r. Jan Tlačil

Poděkování

Rád bych poděkoval vedoucímu práce doc. RNDr. Mikuláši GANGUROVI, Ph.D. za jeho cenné připomínky a rady, které mi během vypracování této práce poskytoval.

Obsah

Úvod

Informačních systémů je v dnešní době opravdu mnoho, mohli bychom i říci, že informační systém již existuje téměř na všechno. V době takového technologického a informačního rozvoje je téměř nemyslitelné, aby někdo s takovým systémem ještě nepřišel do styku. Informační systémy nejčastěji bývají interními systémy podniků, neboť velmi usnadňují práci a správu dat, například v logistickém odvětví jsou informační systémy velmi populární. Ale můžeme se s nimi setkat prakticky kdekoliv a kdykoliv.

Webové informační systémy jsou speciálními případy informačních systémů. Jedná se o jakousi kombinaci informačního systému a webové aplikace. Takový systém je pak přístupný na internetu kdykoliv, kdekoliv a pro kohokoliv. Často se ale liší funkcionalita pro uživatele, kteří jsou v systému registrováni, a kteří jsou pouhými návštěvníky. S takovými systémy se typicky můžeme setkat při on-line rezervování knih, filmů, vstupenek, nebo nemusí ani jít o nějakou rezervaci, například různé podniky často mívají takový systém, kde si zaměstnanci mohou přehledně zjistit stav sortimentu, nebo naplánovat směnu.

Cílem této bakalářské práce je navrhnout a implementovat webový informační systém pro sdílení hudby pro nezávislé interprety. Systém bude umožňovat sdílení vlastních nahrávek a následně přehrávání jak svých, tak cizích. Mezi dílčí cíle se řadí následující:

- **Definice webového informačního systému, jeho vlastností a popsání jeho vrstev**
- **Navrhnutí funkcionalit systému**
- **Definice využitých entit, relačních vazeb a integritních omezeních**
- **Navrhnutí aplikační vrstvy systému**
- **Navrhnutí prezentační vrstvy systému**
- **Zvolení vhodného vývojového prostředí**
- **Ohodnocení funkčnosti systému a jeho užitečnosti**
- **Navrhnutí případných zlepšení**

Bakalářská práce je rozdělena do deseti kapitol, počínaje první, ve které je webový informační systém definován a jsou uvedeny jeho dobré vlastnosti. Následně jsou postupně popsány tři vrstvy systému: databázová, aplikační a prezentační. V databázové části je definován pojem databáze, pojmy důležité pro její strukturu a také jazyk SQL a jeho části.

V rámci aplikační vrstvy se nejdříve zvolí vhodné vývojové prostředí a bude popsán jazyk PHP, pomocí kterého je aplikační vrstva realizována a jaký je rozdíl mezi dynamickými a statickými stránkami. U poslední, prezentační vrstvy, jsou uvedeny programovací jazyky, které slouží k její realizaci. Po definování všech důležitých pojmů je již možné přistoupit ke specifikaci konkrétních požadavků na systém neboli jaké funkcionality má systém mít a jak by měla hudební databáze vypadat. Z těchto konkrétních požadavků se již může přistoupit k samotnému navržení struktury databáze. Následuje návrh aplikační vrstvy, která zajišťuje vlastní funkcionalitu systému a také volba vhodného vývojového prostředí. Vizuální stránka bude zajištěna prezentační vrstvou, u které se nejprve určí používané programovací jazyky a navrhne se samotný design systému. Po navržení všech vrstev následuje samotná implementace systému, ve které budou všechny vrstvy propojeny. Práce je zakončena otestováním, ohodnocením a navržením případných budoucích zlepšení systému.

1 Webový informační systém

Nejprve je nutné definovat pojem „webový informační systém“. Jedná se o webovou aplikaci spojenou s informačním systémem, která je umístěna na internetu a je proto dostupná odkudkoliv a kdykoliv, pokud má uživatel přístupové údaje (Advin).

Definic a vysvětlení pojmu „informační systém“ existuje nesčetně mnoho a nelze říci, které by byly jednoznačně vysvětlující, neboť vždy záleží na pohledu, na který je při definici kladen důraz. Neboli záleží, co bylo autorovým hlavním závěrem při definici (Gangur, 2014).

Zde pro příklad autor uvádí dvě možné definice IS.

„Informační systém je soubor lidí, technických prostředků a metod (programů), zabezpečujících sběr, přenos, zpracování, uchování dat, za účelem prezentace informací pro potřeby uživatelů činných v systémech řízení“ (Wikipedie, 2022).

„Množina prvků ve vzájemných informačních a procesních vztazích (informační procesy). Informační systémy zpracovávají data a zabezpečují komunikaci informací mezi prvky“ (Wikipedie, 2022).

U webové aplikace se opět nelze setkat s jednotnou a jednoznačnou definicí, je vhodné si ale opět alespoň jednu uvést.

„Webová aplikace (anglicky Web Application) je taková aplikace, kterou není nutno instalovat na zařízení uživatele (počítač, tablet, smartphone) a můžete ji spustit z kteréhokoliv zařízení pomocí webového prohlížeče, protože je spuštěna na straně serveru a webový prohlížeč se postará o její zobrazení. Vzhledem k tomu, že je třeba jen prohlížeč se webová aplikace někdy nazývá též jako lehký klient“ (Management Mania).

1.1 Cíl webového informačního systému

Před samotným návrhem a implementací systému je nutné si položit základní otázky, a na základě odpovědí na ně teprve s projektem začít.

- Co je hlavní náplní systému
- Kdo jsou uživatelé systému

- Jaké přínosy bude mít ze systému uživatel IS a jaký provozovatel IS, neboli kolik času se ušetří, jaké prostředky se ušetří a jaké nové služby bude možné realizovat
- Kolik to bude stát a kdo to zaplatí. Kolik bude potřeba investovat do zprovoznění systému a kolik bude potřeba na provoz a údržbu systému.
- Je možné převzít nebo modifikovat již existující řešení?
- Jaká bude kooperace a kompatibilita s jinými systémy?
- Jaké požadavky budou na systém kladeny v budoucnu (Gangur, 2014)?

Jakmile jsou tyto otázky zodpovězeny, může se přistoupit k dalšímu kroku, kterým je popis daného problému a specifikace požadavků. Nejprve musíme znát, jaký je cíl daného projektu. Nejčastěji se můžeme setkat s informačním systémem pro knihovnu a jiné výpůjční služby, ale může se jednat i o něco zcela jiného, například IS pro správu volného času, IS pro nějaký podnik, nebo IS, který zajišťuje rezervace například knih, či film.

1.2 Specifikace požadavků

Dále, abychom byli schopni identifikovat a navrhnout funkčnost systému, musíme znát požadavky na IS, ze strany zadavatele. Obecně můžeme tyto požadavky rozdělit na tři následující oblasti (Gangur, 2014).

- Výstupy – obsahují zejména informační služby neboli odpovědi na dotazy a informační produkty
- Vstupy – jedná se o zdroje, tedy data, informace, požadavky a dotazy
- Procesy – Získávání, uložení, zpracování, přenos a zpřístupnění informací (Gangur, 2014).

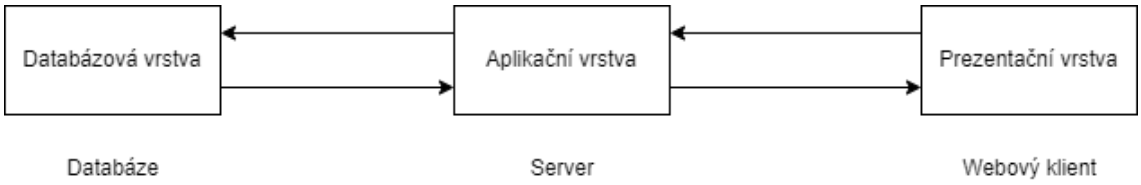
Pokud známe odpovědi na předem zmíněné otázky a známe konkrétní požadavky, můžeme přistoupit k samotnému návrhu prvků systému a vztahů mezi nimi.

1.3 Členění systému

WIS slouží ke komunikaci uživatelů v systému, umožňuje přenos, zpracování a uchování dat a celé to spojuje nějaké uživatelské rozhraní. Na základě tohoto popisu můžeme implementaci WIS rozdělit na 3 vrstvy, které jsou mezi sebou vzájemně propojeny (Gangur, 2014):

- Databázová, která zajišťuje uchování, zpracování a přenos dat. Je realizována samotnou databází a systém pro řízení sběru dat.
- Aplikační vrstva realizuje procesy a metody sběru dat, jejich přenos a zejména jejich zpracování.
- Prezentační vrstva zajišťuje uživatelské rozhraní neboli přehlednou a uživatelsky příjemnou formu (Gangur, 2014).

Obrázek 1 : Vrstvy webového informačního systému



Zdroj: (Gangur, 2014), zpracováno autorem

2 Databázová vrstva systému

Jádrem většiny informačních systémů je databáze a s tím spjatý systém báze řízení dat (SŘBD) (Gangur, 2014). Systémů báze řízení dat je více, avšak pro účely webového informačního systému je nejpoužívanější systém MySQL (Pankrác, 2007). Jako další můžeme uvést například MariaDB, Microsoft Access, Oracle a PostgreSQL. MySQL je tolik používaný právě proto, že se jedná o nejrychlejší, nejlevnější, nejjednodušší a nejspolehlivější SŘBD, který je podporován PHP. MySQL také nabízí převážnou většinu funkcí, které jsou k implementaci informačního systému potřebné (Converse, a další, 2004).

„Databáze je oddělená aplikace, která ukládá kolekci dat. Každá databáze má jedno nebo více API pro tvorbu, přístup, vyhledávání, spravování a replikování dat, která databáze uchovává“ (Converse, a další, 2004 str. 343).

Před samotným návrhem a implementací musí dojít k jistým přípravným krokům. V první řadě je nutné nainstalovat webový databázový server, k čemuž existuje více různých softwarů, ale nejpoužívanější je Xampp, který umožňuje nejen instalaci lokálního webového serveru Apache, ale obsahuje také PHP a MySQL interpreta, který je k implementaci systému nezbytný (Gangur, 2014).

Po instalaci již stačí jen přes Xampp spustit možnost admin, v řádku MySQL a otevře se okno ve webovém pohlížeči s názvem phpMyAdmin, Stejného výsledky lze docílit zadáním localhost/phpmyadmin do webového prohlížeče. PhpMyAdmin umožňuje vytvoření databáze přímo, bez nutnosti instalace jiného softwaru, ale lze tak učinit i jinými způsoby.

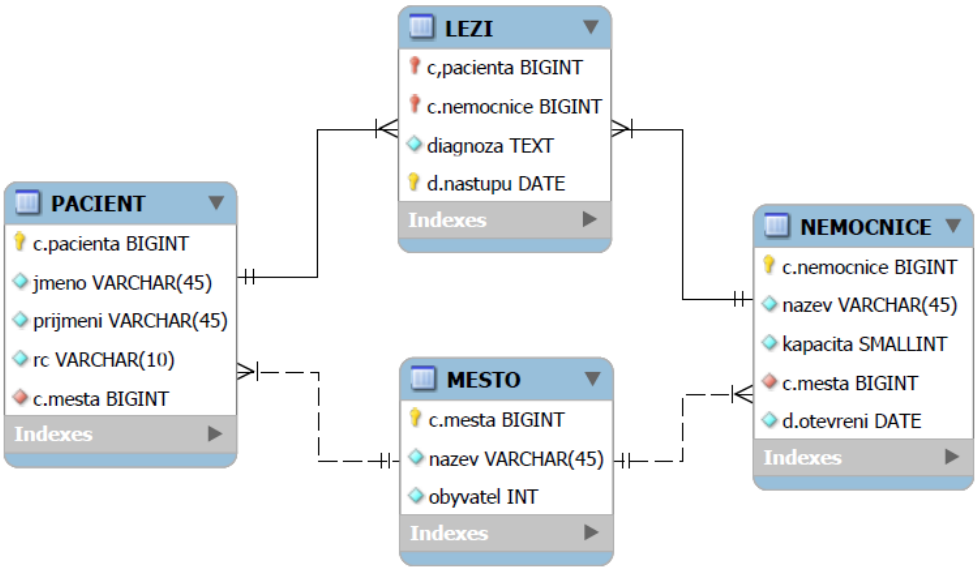
2.1 Návrh databáze

Klíčovým krokem je návrh samotné struktury databáze. Databáze má určitou strukturu, skládající se z entit neboli tabulek, které mají určité atributy a klíče. Mezi těmito entitami existují vazby, které se nazývají relace. Ale před samotným návrhem by bylo vhodné uvést terminologii v databázích užívanou.

- Tabulky – Tabulka obsahuje informace o nějaké entitě, což je model objektu z reálného světa (Gangur, 2014). Typická tabulka se skládá z řádků a sloupců, a tak tomu bude i v tomto případě.
- Sloupce – Každý sloupec má jedinečný název a má přidělený typ dat. Typů dat existuje v MySQL mnoho, ale můžeme uvést několik základních, a to INT, VARCHAR, TEXT a DATE. Každý datový typ má nějaké své omezení, nejčastěji definovaný rozsah, či předem daný tvar. V terminologii databází jsou sloupce často označovány jako atributy.
- Řádky – Každý řádek obsahuje informace o jednom konkrétním objektu té dané entity. V terminologii databází jsou řádky často označovány jako záznamy.
- Hodnoty – Každá buňka, nebo každý řádek v každém sloupci obsahuje hodnotu atributu, která má datový typ, který je definován ve sloupci.
- Klíče – Databáze stojí na funkčnosti takzvaných klíčů, pomocí kterých jsou jednotlivé tabulky propojeny. Takový atribut může mít libovolný datový typ, ale nikdy nesmí být prázdný, jedná se o mandatorní údaj. Klíč se nemusí skládat pouze z jednoho atributu, ale může jich být i více. Klíč je jednoznačný identifikátor tabulky a každá tabulka tento primární klíč mít musí. Tabulky jsou spojovány právě pomocí klíčů a relací. Toto propojení funguje právě na základě primárního klíče v jedné tabulce, a cizího klíče v tabulce jiné (Gangur, 2014).
- Relace – Mezi tabulkami existují relace, či jinak, vztahy. Jak je výše zmíněno, relace fungují na principu primárního a cizího klíče. Můžeme mít tři vztahy, a to: 1:1, 1:N a M:N. Relace nám tedy říkají, kolik údajů z jedné tabulky se vztahuje ke kolika údajům z druhé tabulky. Tyto konkrétní vztahy jsou nazývány kardinalita.
    - 1:1 – Tato vazba říká, že k jednomu záznamu z jedné tabulky existuje právě jeden z jiné.
    - 1:N – K jednomu záznamu z jedné tabulky náleží N záznamů z tabulky druhé.
    - M:N – K jednomu záznamu z jedné tabulky existuje N záznamů z tabulky druhé a k jednomu záznamu z tabulky druhé existuje M záznamů z tabulky první (Gangur, 2014).
- Povinnost výskytu – Zde je již z názvu patrné, že se jedná o to, jestli se nějaký záznam z jedné tabulky musí vyskytovat ve vztahu k jiné tabulce.

Tímto jsme si popsali základní terminologii ohledně databází. Můžeme přejít na návrh takzvaného ERA diagramu. ERA diagram, v angličtině Entity-Relationship diagram, je vlastně model databáze. V modelu jsou vyznačené všechny tabulky, jejich atributy, cizí a primární klíče, relace mezi tabulkami a povinnost výskytu. ERA diagram by mohl v MySQL vypadat například takhle.

Obrázek 2: Příklad ERA diagramu vytvořeného v SŘBD MySQL



Zdroj: Vlastní zpracování

Při návrhu je záhodno dodržovat několik základních principů (Gangur, 2014).

- Které objekty reálného světa budeme modelovat – Uvědomění si, které všechny objekty jsou nutné ke splnění požadavků. Někdy je potřebné definovat i pomocné objekty, které slouží k propojení jiných objektů.
- Neukládat redundantní data – Neboli neukládat data vícekrát, než je opravdu potřeba. V první řadě plýtváme místem, a druhé řadě, když budeme chtít opravit jeden záznam, budeme ho muset upravit ve všech tabulkách/entitách, kde se vyskytuje.
- Používat atomické sloupcové hodnoty – Občas je nutné vytvořit místo jednoho atributu hned novou tabulku, kde budou všechny informace dohledatelné.

- Volba účelného klíče – Pro každou tabulku vytvoříme unikátní primární klíč a pojmenujeme ho tak, aby bylo jednoznačně poznat že se o primární, nebo cizí klíč jedná.
- Promyslet všechny možné dotazy předem – Nad tabulkami budeme chtít provádět dotazy, jejichž náplň je dobré si promyslet předem, neboť potřebujeme vědět, jaké informace má daný dotaz zpracovat.
- Nenavrhovat databázi s mnoha prázdnými poli – U atributů, jejichž hodnota je převážně NULL, je dobré si uvědomit, jestli jsou dané atributy opravdu v tabulce potřebné. Prázdná data totiž komplikují hledání a zároveň zbytečně zabírají místo (Gangur, 2014).

V praxi můžeme databázi v programu navrhnout více způsoby – většina SŘBD umožňuje databázi namodelovat, že postupně přidáváme tabulky na volné plátno. Také je ale možné tabulky vytvořit pomocí jazyka SQL. Ale jelikož máme přístup do phpMyAdmin, není nutné instalovat vlastní SŘBD, protože phpMyAdmin také umožňuje vytvoření databáze a její struktury.

2.2 Jazyk SQL

Jakmile máme databázi navrženou a sestrojenou v SŘBD či v phpMyAdmin, potřebujeme s daty z databáze nějakým způsobem pracovat. K tomu právě slouží jazyk SQL.

Jazyk SQL neboli Structured Query Language – strukturovaný dotazovací jazyk, je vlastně jedinou možností, jak po databázi požadovat jakoukoliv operaci s daty, ať se jedná o vkládání, mazání, vrácení, setřízení či filtrování dat (Pankrác, 2007). Takové dotazy jsou závislé na databázi, nad které chceme dotaz vykonat (Gangur, 2014). PhpMyAdmin tyto dotazy aktivně podporuje, lze je tedy zadávat přímo, jen musí být zvolená správná databáze.

Jazyk SQL se rozděluje na čtyři části, DDL (Data Definition Language), DML (Data manipulation language), DCL (Data Control Language) a TCC (Transaction Control

Commands) (Lacko, 2005). Definice těchto částí lze vydedukovat z anglické terminologie.

2.2.1 DDL

Jedná se o jazyk, který definuje data neboli pomocí něj vytváříme strukturu databáze. Tedy vytváříme samotnou databázi a její obsah, což jsou tabulky, pohledy, indexy a tak dále. Takovou strukturu ovšem nemusíme vykonávat pomocí příkazů, jak phpMyAdmin, tak samotné MySQL umožňuje databázi vybudovat i bez nich. Mezi dotazy tohoto jazyka patří zejména CREATE – vytváří objekty, například TABLE, INDEX, VIEW a DATABASE. Následně je umožňuje pomocí dotazu DROP také zrušit – smazat. Strukturu lze i obměňovat pomocí ALTER (Lacko, 2005).

Obrázek 3: příklad dotazu CREATE v MySQL

```
CREATE TABLE jazyk
(
    id_jazyka VARCHAR2(20)  NOT NULL,
    nazev_jazyka     VARCHAR2(20)  NOT NULL
)
;
ALTER TABLE jazyk ADD CONSTRAINT jazyk_pk PRIMARY KEY ( id_jazyka );
```

Zdroj: vlastní zpracování v MySQL

2.2.2 DML

Jazyk, který umožňuje samotnou práci s daty, tedy jejich výběr, vkládání, mazání a aktualizování. Mezi tyto příkazy patří hlavní čtyři – SELECT, INSERT, UPDATE a DELETE. Zejména SELECT je velmi využívaný, neboť právě ten umožňuje zvolit data podle toho, jaké jsou zadány podmínky výběru. Tyto dotazy lze vykonávat nad jednou i více tabulkami (Lacko, 2005).

Obrázek 4: příklad dotazu SELECT v MySQL

```
SELECT f.nazev, j.nazev_jazyka, za.nazev_zanru
FROM FILM f
JOIN DABING d
ON f.id_film = d.film_id_film
JOIN ZANRY z
ON f.id_film = z.film_id_film
JOIN ZANR za
ON z.zanr_id_zanru = za.id_zanru
JOIN JAZYK j
ON j.id_jazyka = d.jazyk_id_jazyka WHERE j.nazev_jazyka ='CZ'
ORDER BY f.nazev ASC;
```

Zdroj: Vlastní zpracování v MySQL

2.2.3 DCL

Tento jazyk obsahuje příkazy pro správu uživatelů a jejich přístupu. Jedná se zejména o příkazy CREATE a DROP USER, ALTER USER, GRANT a REVOKE. Jazyk umožňuje přidání, upravení a odebrání uživatele a v neposlední řadě úpravu uživatelských práv (Lacko, 2005).

2.2.4 TCC

Poslední, neméně důležitý jazyk, se zabývá příkazy pro řízení transakcí. Transakce je činnost, která převádí databázi z jednoho konzistentního stavu do jiného. Pro lepší vysvětlení si uvedeme základní příkazy, kterými jsou ROLLBACK, COMMIT, SET TRANSACTION a SAVEPOINT. Z názvu je patrné, že tyto transakce umožňují například vytvoření nějakého bodu obnovení, do kterého se v případě neočekávané a nežádané události můžeme vrátit (Lacko, 2005).

3 Aplikační vrstva

Aplikační vrstva realizuje nad danou databází tu logickou část celé aplikace pomocí programovacího jazyka PHP. Naopak prezentační část tvoří grafické rozhraní systému. Prezentační část je tvořena s využitím jazyka HTML a s tím spjatého CSS a Javascriptu, případně je možné využít různé nadstavby, například bootstrap. Tyto vrstvy řeší několik základních úloh (Gangur, 2014):

- Připojení k databázovému serveru MySQL podle zadaných přístupových parametrů.
- Nastavení aktuální databáze.
- Tvorba uživatelsky přívětivého uživatelského prostředí a předání Webovému serveru.
- Vytvoření uživatelsky příjemných formulářů pro vstup do samotného systému.
- Sestavení SQL dotazů dle vložených šablon s proměnlivými parametry v závislosti na volbách uživatelů informačního systému.
- Převzetí odpovědí na zadané SQL dotazy a interpretace výstupních dat.
- Formátování a příprava dat k prezentaci v prohlížeči webového klienta.
- Odeslání naformátovánch výstupů webovému serveru, který je odešle webovému prohlížeči (Gangur, 2014).

Databázová vrstva je jakási vnitřní architektura databáze, v této části se budeme zabývat tou vnější částí. Vnější architektura je založena na vztahu klient – server. Klient, konkrétně webový prohlížeč, zasílá požadavky na webový server, který požadavek zpracuje a zašle zpět odpověď. Databázová vrstva v tomto vztahu hraje klíčovou roli, neboť zní se čerpají požadované informace a data (Gangur, 2014).

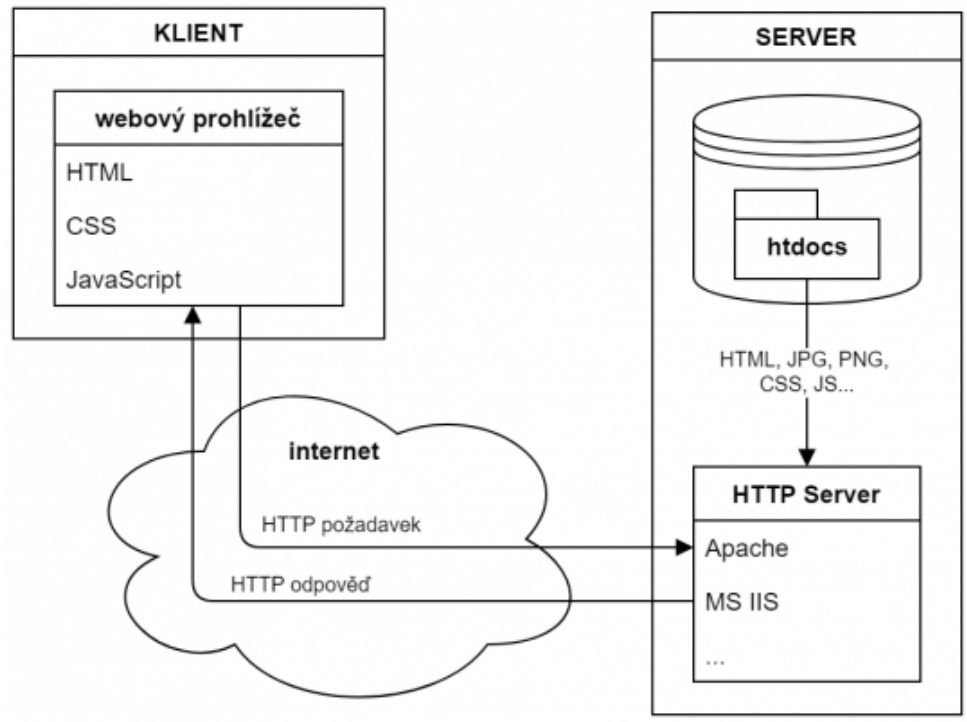
Požadavek na webového klienta bychom mohli shrnout v následujících šesti krocích (Gangur, 2014).

- Server převezme od klienta požadavek, který zpravidla žádá o provedení určitého PHP skriptu, a vyvolá PHP interpret, kterému skript předá.
- PHP interpret provede následující příkazy – Spojí se s databázovým serverem MySQL a zvolí odpovídající databázi a odešle požadavek databázovému serveru

na zaslání požadovaných dat z databáze pomocí SQL příkazu, který se nachází v PHP skriptu,

- Určený databázový server zpracuje přijatý SQL dotaz a vrátí jeho výsledek PHP interpretu.
- PHP interpret tato data převezme a podle PHP skriptu je zpracuje, připraví v požadovaném formátu a odešle webovému serveru.
- Webový server data převezme a odešle je zpět k webovému klientovi.
- V poslední fáze jsou data ve webovém prohlížeči zobrazena (Gangur, 2014).

Obrázek 5: Grafické znázornění průběhu požadavku



Zdroj: (Bubílek, 2019)

3.1 Vývojové prostředí

Před psaním kódu v jakémkoliv programovacím jazyce si musíme zvolit vývojovém prostředí, ve kterém bude psaní probíhat. V dnešní době existuje mnoho takových prostředí, každé má některé své výhody a nevýhody a každé zpravidla bývá vhodné pro jiné jazyky. Je nutné si uvědomit, že ne každé vývojové prostředí je i compiler neboli že v něm lze napsaný kód spustit, některá jsou pouze editační nástroje. Pro příklad si můžeme několik, v rámci tohoto tématu nejužívanějších, uvést.

- Eclipse – Pravděpodobně nejznámější a nejvyužívanější vývojové prostředí pro mnoho programovacích jazyků. Eclipse nabízí více než 350 open-source projektů. (The Eclipse Foundation).
- PSpad – Univerzální textový editor, který slouží k psaní kódu jak v PHP, tak HTML, CSS, JavaScript a mnoha dalších jazyků. Umožňuje i nastavení obarvení textu, což velmi usnadňuje orientaci v kódu a případné nalezení a odstranění chyb (Gangur, 2014).
- Visual Code Studio – Další z velmi rozšířených vývojových prostředí, které je zdarma. Opět podporuje mnoho jazyků (Visual Studio Code).
- SublimeText – Sofistikovaný textový editor s jednoduchým a příjemným uživatelským prostředím, který podporuje mnoho programovacích jazyků (Sublime Text).

Pro tuto práci bylo zvoleno vývojové prostředí SublimeText. Jedná se sice o komerční editor, ale je možné jej zdarma vyzkoušet na neomezenou dobu. Je velmi rychlý, přehledný a podporuje mnoho jazyků. Zde je právě významný rozdíl například v porovnání s velmi rozšířeným prostředím Eclipse, které se načítá mnohem pomaleji a má mnoho funkcí které nejsou pro tento systém využitelné. Další velkou výhodou SublimeText je jeho přehlednost a příjemné uživatelské prostředí, například mapa kódu je velmi užitečným nástrojem, který usnadňuje orientaci v kódu. Software umožňuje instalaci pluginů a má mnoho klávesových zkratek, které velmi urychlují psaní kódu, příkladem může být plugin Emmet. Program také dokáže upozorňovat na syntaxové chyby.

Po zvolení vhodného vývojového prostředí je ještě dobré si uvědomit, že napsané PHP skripty musí být ve složce softwaru Xampp, který implicitně PHP skripty ukládá do složky xampp/htdocs. Tuto konfiguraci je samozřejmě možné v programu změnit, což je mnohokrát žádoucí. Je přehlednější si vytvořit vlastní složku ve složce htdocs, kam budeme skripty ukládat, například „Muj_prvni_WIS“. Vlastní systém se pak spustí přes webový prohlížeč, kam je zadána adresa http://localhost/Muj_prvni_WIS/ (Gangur, 2014).

3.2 Jazyk PHP

„PHP je serverový skriptovací jazyk, který umožňuje tvořit dynamické webové stránky s ohromnými možnostmi“ (Pankrác, 2007 str. 19).

„Zkratka PHP je rekurzivní akronym pro PHP: Hypertextový Preprocesor. PHP je velmi rozšířený a pro všeobecné užívání určený open source skriptovací jazyk, který je vhodný zvláště pro programování na webu a může být vnořen do HTML“ (Leiss, a další, 2010 str. 13).

Je možné HTML zakomponovat do PHP skriptu, avšak jako přehlednější a pohodlnější se jeví oba dva jazyky oddělit. PHP kód totiž potřebujeme pouze když od systému chceme, aby dynamicky reagoval neboli aby se choval podle toho, kdo je do systému přihlášený a jaký je aktuální stav prostředí. Pro statickou část si vystačíme s jazykem HTML a jeho komponenty (Leiss, a další, 2010).

Velkou výhodou PHP je možnost opětovného využití kódu. Nejenže tím šetříme místo, zabraňujeme nastání chyb a udržujeme jednotnost daného informačního systému, ale hlavně si tím šetříme čas, který bychom při opakovaném psaní kódu museli vynaložit.

Výchozí syntaxe oddělení kódu jazyka PHP se dá popsat velmi krátce, a to značkami pro začátek - <?php a konec kódu - ?>. Pro usnadnění lze použít i kratší verzi <? ?>. Z historického hlediska bylo pro některé editory problémové tyto značky rozeznat, z toho důvodu bylo do jazyka PHP začleněno značkování <script> (Gilmore, 2005).

Proměnné v PHP jsou velmi podobné proměnným v jiných programovacích jazycích. Na tento jazyk se často referuje jako „loosely-typed“, což znamená, že jedna proměnná může obsahovat více datových typů, například číslo, text a další typy. Při definování proměnné se vždy začíná znakem $. Pro ukázku zavedeme jednu proměnnou $testVariable = 3; která má v tomto případě číselnou hodnotu, ale mohla by mít i například textovou hodnotu: $testVariable = ‘Three‘; (Butler, 2012).

Po pochopení základních principů a syntaxe jazyka PHP se můžeme podívat na klíčovou část návrhu informačního systému, a to na formulář (Gangur, 2014).

Formulář se opět sestává ze dvou částí, PHP a HTML. Je klíčový z toho důvodu, že umožňuje uživatelům se do daného systému zaregistrovat a následně ho nějakým způsobem využívat.

Struktura formuláře v PHP je následující, základně je tvořena tagem form.

Obrázek 6: Ukázka formuláře v PHP

```
<form action="a.php" method="get">
Jméno: <input type="text" name="jmeno">
věk: <input type="text" name="vek">
<input type="submit" value="odeslat">
</form>
```

Zdroj: vlastní zpracování

Prvek password se vypisuje skrytě. Proměnná typu radio umožňuje zvolit z několika možností – to by se dalo nejčastěji využít u volby pohlaví. Proměnná typu checkbox implikuje že můžeme zvolit více možností a zaškrtnout je. Tuto proměnnou bychom mohli využít například při volbě jazyka, kdy uživatel může být schopen dorozumívat se ve více jazycích. Proměnných tohoto typu existuje více, například select, která umožňuje nastavit viditelnost možnosti. Textarea, jak už název vypovídá, umožňuje uživateli zadat větší množství textu (Gangur, 2014)

PHP musí být bezpečnostně ošetřen, neboť existuje mnoho způsobů, jak takový IS napadnout.

Následuje samotné připojení k databázi MySQL pomocí PHP, což můžeme rozdělit do dvou kroků, a to: Samotné připojení k databázovému serveru a volbu databáze, a vlastní získávání dat z databáze pomocí SQL dotazů (Gangur, 2014)

Obrázek 7: ukázka připojení k databázi pomocí PHP

```
<?php
	$conn = mysqli_connect("localhost", "root", "heslo", "databaze");

	if(!$conn) {
		die("Connection failed: ".mysqli_connect_error());
	}
?>
```

Zdroj: vlastní zpracování.

3.3 Typy stránek

3.3.1 Dynamické stránky

Jak je patrné z názvu, statické webové stránky jsou neměnné a vždy budou ve stejné podobě, v jaké je autor vytvořil. Ke tvorbě takových stránek nepotřebujeme PHP, ale vystačíme si s HTML, CSS a podobnými jazyky (Pankrác, 2007).

3.3.2 Statické stránky

Jak je zmíněno výše, statické webové stránky se nijak nikdy nemění, pokud autor explicitně neprovede nějakou změnu. Dynamické tedy logicky určitými změnami procházejí. Zde právě vstupuje jazyk PHP, který takové změny umožňuje. V praxi dynamičnost webových stránek znamená, že se stránky mění podle toho, kdo je právě do systému přihlášen (Pankrác, 2007).

4 Prezentační vrstva

„Prezentační vrstva: nebo také vrstva uživatelského rozhraní je odpovědná za interakci s obsluhou – sběr/zobrazení informací v uživatelsky čitelné a přívětivé podobě“ (Wi - Webový integrátor, 2013).

Vrstva tedy zajišťuje uživatelské rozhraní pro určené verze cílových zařízení a prohlížečů, typicky bývá v podobě výstupního kódu HTML, CSS a JavaScript. S databázovou vrstvou ji spojuje právě vrstva aplikační (Wi - Webový integrátor, 2013).

4.1 HTML

HTML je pravděpodobně nejvyužívanější jazyk sloužící k tvorbě webových stránek. Jazyk je založen na používání značek neboli z angličtiny hypertextový značkovací jazyk – HyperText Markup Language. Lze vidět, že název jazyka se skládá ze dvou částí, a to HyperText a Markup. HyperText, v češtině hypertextový odkaz, dokáže propojit texty právě na základě odkazů. Používá se k označení dokumentů, které se mohou měnit. Kdežto Markup dává jednotlivým slovům nějaký význam právě pomocí značek. Tyto značky se nazývají tagy a elementy. Značky v HTML fungují tak, aby prohlížeč byl schopen je správně přečíst a interpretovat. Například u tagu bude prohlížeč jasně vědět, že se jedná o seznam, a bude k tomu podle toho přistupovat. Tagů a elementů existuje mnoho, ale cílem této bakalářské práce není popsat jazyk HTML (Schafer, 2009).

Důležité je zmínit, že v dnešní době již HTML neurčuje grafickou stránku webu, ale pouze navrhuje jeho strukturu. Samozřejmě má grafické a stylizační funkce, které je možné využít, ale jedná se o opravdu základní, například ztučnění či převedení textu do kurzívy. A tyto funkce se již v HTML nevyužívají, jelikož k samotnému grafickému provedení stránky v dnešní době slouží jazyk CSS. V provedení stylu již v HTML nám pochopitelně nic nebrání, ale je mnohem jednodušší a přehlednější mít tyto věci v separátním souboru, či souborech.

Obrázek 8: ukázka struktury HTML dokumentu

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title></title>
</head>
<body>
</body>
</html>
```

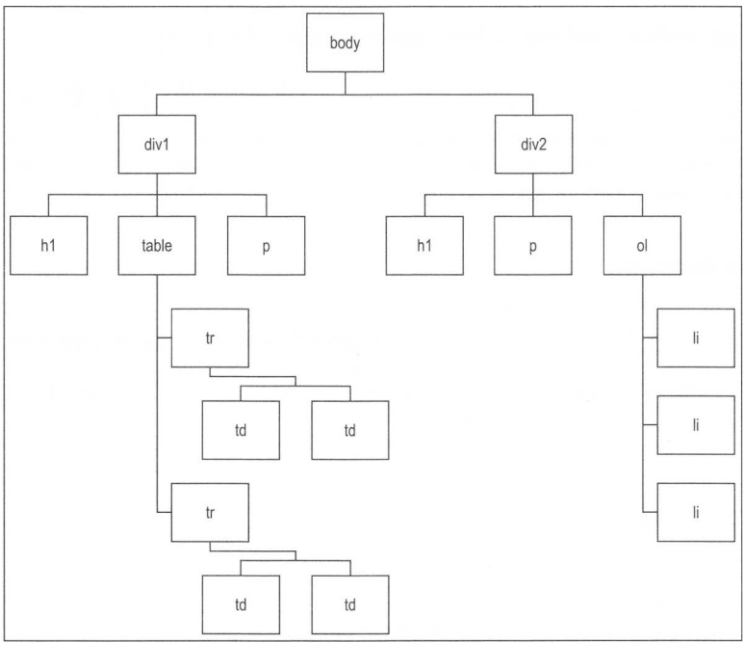
Zdroj:vlastní zpracování

4.2 CSS

Jak bylo již zmíněno výše, jazyk HTML obstarává pouze strukturální část webové stránky a na grafické záležitosti máme jazyk CSS. Z anglického názvu Cascading Style Sheets si rovnou můžeme odvodit význam, a to kaskádové styly. Kaskádové jsou nazývány proto, že je možné jednotlivé styly na sebe vrstvit (Janovský);

V první řadě musíme v HTML souboru odkazovat na zvolený CSS soubor, přes skript. V CSS teprve řešíme rozložení jednotlivých elementů, jejich velikost, barvu pozadí a tak dále. Tento jazyk je velmi užitečný a úsporný v tom, že nám umožňuje nastavit pro více elementů najednou, což ušetří spoustu práce. Je tedy možné nastavit například pro všechny nadpisy, aby byly tučné, a to pouze v jediné řádce kódu (Schafer, 2009).

Obrázek 9: Diagram hierarchie CSS dokumentu



Zdroj: (Schafer, 2009 str. 380)

Kaskádové styly jsou tedy proto, že je možné jednotlivé styly skládat na sebe, tím pádem některé mohou nad jinými převažovat (Schafer, 2009). Pravděpodobná situace při návrhu CSS stylu je taková, že pro určitý element budeme chtít, aby převážně vypadal určitým způsobem, ale víme, že element bude provázet mnoho obměn, které se budou lišit podle aktuálního prostředí. Například chceme, aby nadpisy byly tučné pro každý element nadpis, ale zároveň by se nám líbilo, aby se barva nadpisů lišila v jednotlivých záložkách. Taková věc se dá zařídit s pomocí takzvaných tříd, kdy pro konkrétní nadpis definujeme třídu a následně v té třídě upravíme barvu fontu.

Obrázek 10: ukázka struktury CSS dokumentu

```
.navItem {
    padding: 10px 0;
    font-size: 14px;
    font-weight: 700;
    display: block;
    letter-spacing: 1px;
    position: relative;
}
```

Zdroj: vlastní zpracování

4.3 JavaScript

Javascript je dalším ze základních jazyků pro vývoj webových aplikací. Jedná se objektově orientovaný programovací jazyk. Objektově orientovaný je vlastně způsob, jak nad takovým kódem přemýšlet a jak ho psát (Remetei, 2016). V praxi to znamená, že se co nejvíce snažíme kód neopakovat a využívat již předem napsaný kód vícekrát. Dále je dobré zmínit, že se jedná spíše o komplementární programovací jazyk, tedy jazyk, který je používán jako doplněk a není moc běžné, aby celá aplikace byla napsána pouze v něm (Suehring, 2008).

HTML dodává webové aplikaci struktura, CSS se pak zaměřuje na stylizaci a design a JavaScript stránce dodává požadovanou interaktivitu. S pomocí JavaScriptu je například možné vytvořit dynamické menu a různé hover funkce – neboli co daný kontejner udělá, když po něm uživatel přejede myší. Důležité je si uvědomit, že JavaScript, na rozdíl od jazyka PHP, není vykonáván na serveru, ale až ve webovém prohlížeči. Je tedy vlastně limitován tím, co se může udělat v rámci platformy, na které běží, což je právě ten určitý webový prohlížeč (Suehring, 2008).

Jak uvádí autor v knize JavaScript pro webové vývojáře, javaScript je skriptovací jazyk navržený pro interakci s webovými stránkami a skládá se z následujících tří odlišných částí:

- Jazyk ECMAScript, který je definován ve standardu ECMA-262 a který poskytuje základní funkční prvky.
- Objektový model dokumentu (DOM), který poskytuje metody...<t rozhraní pro práci s obsahem webové stránky.
- Objektový model prohlížeče (BOM), který poskytuje metody a rozhraní pro interakci s prohlížečem (Zakas, 2009).

Obrázek 11: Příklad jednoduché funkce v JavaScriptu

```
function myFunction(p1, p2){
          return p1*p2;
}
```

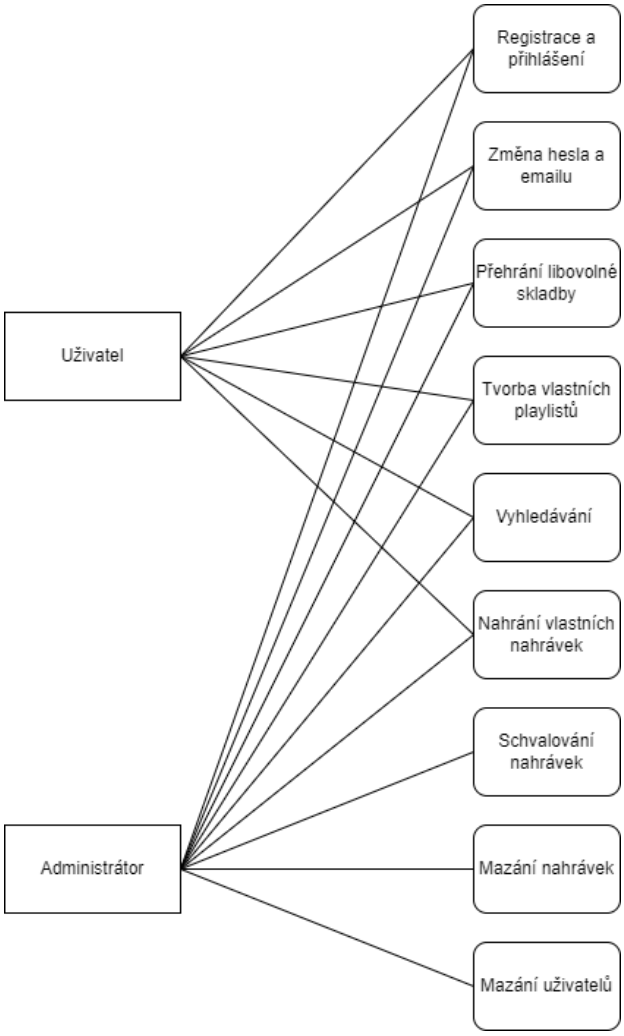
Zdroj: vlastní zpracování

5 Specifikace požadavků na systém

Jak bylo již autorem zmiňováno v první kapitole – Webový informační systém, základem každého takového systému je jasná specifikace požadavků a funkcionalit.

Tento webový informační systém bude přístupný pouze pro přihlášené uživatele. V systému budou tedy přihlášeny jen dvě kategorie – uživatelé a administrátoři. Uživatel bude mít klasická práva, která mu budou umožňovat pouze operace, které nijak nemění systém a jeho data – kromě vlastních uživatelských dat, konkrétně email a heslo. Administrátor bude mít stejné funkce a k tomu nějaké navíc, které budou již s daty určitým způsobem manipulovat.

Obrázek 12: Funkcionality systému



Zdroj: vlastní zpracování

5.1 Uživatel

- Možnost registrace a přihlášení
- Možnost změny hesla či emailu
- Možnost přehrání libovolné skladby či alba
- Možnost vytvoření vlastních playlistů
- Možnost vyhledávání podle skladby, alba a interpreta
- Možnost nahrání vlastních nahrávek

5.2 Administrátor

- Administrátor bude mít v samotném systému stejné funkcionality jako uživatel a k tomu některé navíc
- Schválení či neschválení uživatelských nahrávek
- Možnost smazání jakékoliv skladby
- Administrátor bude mít samozřejmě přístup do databáze, kde bude také moci dělat jakékoliv úpravy

5.3 Uživatelská práva

Práva budou řešena přidáním atributu „rightsId“ do entity Users a vytvoření entity Rights, kde budou názvy těchto práv, jedná se o pouhý číselník. Práva, jak autor zmiňuje na začátku, budou jen dvě – uživatelská a administrátorská. Pro id == 0 se bude jednat o klasického uživatele a toho id mu bude přiděleno automaticky při registraci, ale samozřejmě bude možné přidělit práva administrátora později. Pro id == 1 se bude tedy jednat o administrátora, který bude mít jinou pravomoc. V systému budou před funkcionalitami, ke kterým by měly mít přístup pouze administrátoři podmínky, které funkcionality zobrazí jen administrátorům. Například viditelnost tlačítka pro vymazání skladby z databáze.

Obrázek 13: Příklad zobrazení obsahu administrátorem

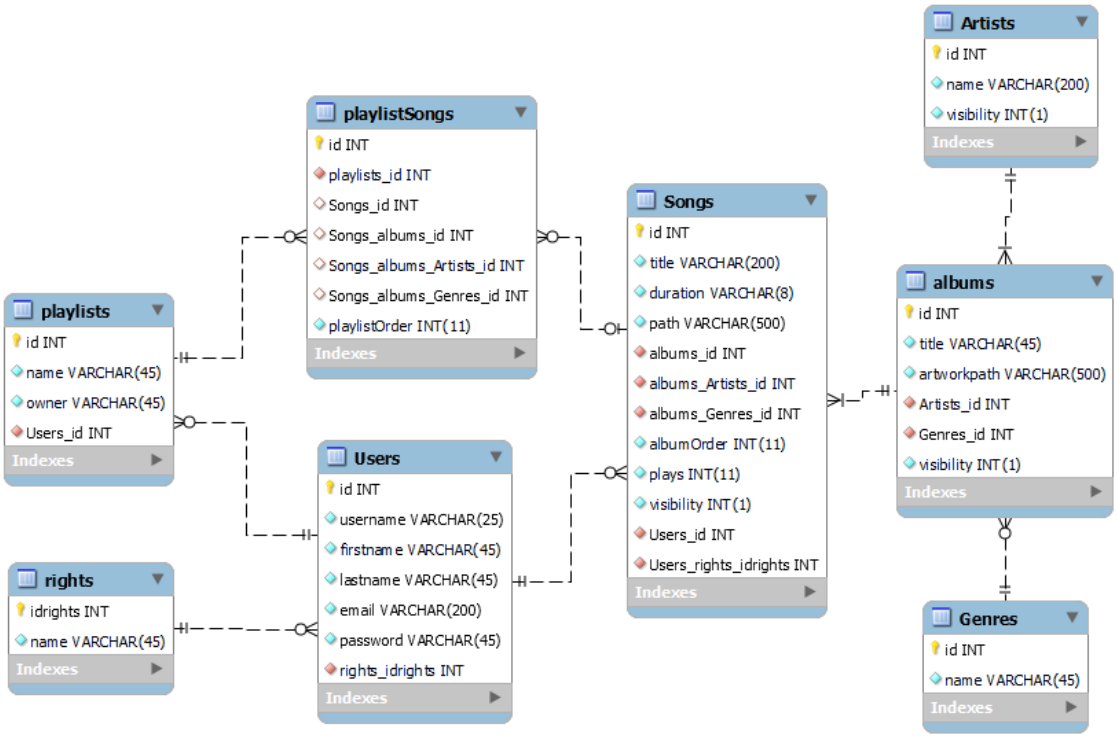
```
<?php     if($userLoggedIn->getRights()==1) { ?>
<div class="item" onclick="removeSong(this,'<?php echo $albumId;?>')">Delete song</div>
<?php
          } ?>
```

Zdroj: vlastní zpracování

6 Databázová vrstva

6.1 ERA model

Obrázek 14: ERA model tohoto webového informačního systému



Zdroj: vlastní zpracování

Návrh databáze byl nejprve uskutečněn v SŘBD MySQL, aby si autor udělal představu o nutných entitách, vazbách a integritních omezeních. Po tomto teoretickém úkonu následovala implementace databáze v prostředí localhost/phpmyadmin. Nejprve byla vytvořena databáze určená pro tento systém a postupně byla naplněna entitami z výše znázorněného ERA diagramu a byla rovnou naplněna zkušebními daty. Tvorba neprobíhala manuálně, ale pomocí SQL dotazů, například:

Obrázek 15: Příklad vytvoření entity

```
CREATE TABLE IF NOT EXISTS `albums` (
  `id` int(11) NOT NULL,
  `title` varchar(250) NOT NULL,
  `artist` int(11) NOT NULL,
  `genre` int(11) NOT NULL,
  `artworkPath` varchar(500) NOT NULL
);
```

Zdroj: vlastní zpracování

6.2 Popis entit a jejich atributů

Entity jsou v tomto systému následující – Uploads, Users, Playlists, PlaylistSongs, Songs, Albums, Artists, Rights a Genres a každá entita obsahuje svůj primární klíč, který je pro jednoduchost všude nazván „id“.

- Users – uživatelé, bude obsahovat základní info o uživatelých, to znamená jejich celá jména, uživatelská jména, hesla, emaily a práva. Uživatel s právem, které bude mít hodnotu ‘0‘, bude pouhým uživatelem, kdežto s právem, které bude mít hodnotu ‘1‘ se bude jednat o administrátora. Při registraci dostane bude každému uživateli automaticky přiděleno právo s hodnotou ‘0‘.
- Genres – Číselníková tabulka, která obsahuje pouze názvy žánrů.
- Artists – Obsahuje pouze název daného Interpreta.
- Albums – Album již obsahuje více atributů, a to název a artworkPath neboli cestu k obalu alba. Tato entita obsahuje dva cizí klíče – artistId a genreId.
- Songs – Entita nahrávek se skládá z názvu, délky stopy, pořadí na albu, počet přehrání a path – cestě k souboru. Cizí klíče jsou v tomto případě albumId, artistId a genreId.
- Playlists – Playlist bude mít jako vlastní atribut pouze název. Jako cizí klíč zde bude působit userId, tedy id uživatele, který playlist vytvořil.
- PlaylistSongs – Jelikož mezi entitami Songs a Playlists figuruje vazba M:N, musí vzniknou třetí, pomocná entita, která vazbu M:N rozloží na 1:M a 1:N. A bude si držet cizí klíče genreId, playlistId, albumId, artistId a songId.
- Rights – Tato entita obsahuje názvy uživatelských práv.

6.3 Popis vazeb a integritních omezení

Mezi výše vyjmenovanými entitami existují vazby, v tomto případě se jedná o vazby 1:N a M:N. Vazba 1:1 se zde nevyskytuje, neboť většinou je jednodušší atribut zkrátka zahrnout do entity, než zbytečně vytvářet novou entitu, která si ponese cizí klíč té první. Avšak v systému se vyskytuje jedna vazba M:N, která není možná namodelovat a musí se řešit takzvaným rozkladem. Z vazby M:N tedy vzniknou vazby 1:M a 1:N a jedna

pomocná entita. V tomto případě ERA model říká, že v M playlistech může být N nahrávek, a N nahrávek může být v M playlistech. Jako pomocná entita vzniká entita s názvem playlistSongs. Ostatní vazby jsou 1:N.

Interpret může mít libovolné množství alb, na albu může být libovolné množství nahrávek. Stejný žánr může mít libovolné množství alb. Každá nahrávka může být v libovolném množství playlistů a každý playlist může mít libovolné množství nahrávek. A uživatel může mít libovolné množství playlistů a nahraných písní.

Poslední záležitost se týká integritních omezeních neboli povinnosti výskytu. Například v databázi nebude interpret, který nemá žádné album. Nebude zde ani žádné album které nemá ani jednu nahrávku. Album musí mít určený žánr. Ale naopak, playlist nemusí mít žádné skladby, a žádná skladba nemusí být součástí nějakého playlistu. Zde je tedy povinnost výskytu dobrovolná.

Vyskytovat se zde budou i jiná integritní omezení, zejména omezení na rozsah. Každý atribut má definovaný maximální rozsah, tedy maximálně kolika znaků může nabývat. Každý atribut také má svůj definovaný typ, například INT či VARCHAR, jedná se o doménové integritní omezení.

6.4 Přístup do databáze

S daty z databáze se bude pracovat SQL dotazy, které budou obsaženy v PHP kódu. Nyní zbývá zpřístupnit databázi systému pomocí jazyka PHP. Přístup bude v souboru config.inc, který budou ostatní stránky obsahovat, pomocí include. Soubor je formátu inc, aby jej nešlo spustit, aby si nikdo nemohl zobrazit přístupové údaje. Níže znázorněná proměnná $con bude následně využívána pro připojení k databázi a práci s jejími daty.

Obrázek 16: Vytvoření přístupu do databáze v odděleném souboru.

```
<?php
          ob_start();
          session_start();

          $timezone = date_default_timezone_set("Europe/London");
          $con = mysqli_connect("localhost", "root", "heslo", "database");
          if(mysqli_connect_errno()) {
                    echo "Failed to connect: " . mysqli_connect_errno();
          }
?>
```

Zdroj: vlastní zpracování

Je důležité zmínit, že do databáze nebudou nahrávány žádné soubory, ať už se jedná o audio či obrázky. Vše bude uloženo na cloudovém úložišti a v databázi budou uloženy pouze cesty k daným souborům. Cloudových služeb v této době existuje velké množství, ale můžeme si představit pouze některé nejvýznamnější, které jsou DropBox, iCloud, Google Drive, Microsoft One Drive, Mega, Box, pCloud a AWS.

Jako cloudové úložiště bylo autorem zvoleno AWS – Amazon Web Services (Webové služby Amazonu). Toto úložiště bylo zvoleno z několika důvodů, ať už osobních, či obecných. Některé výhody AWS nebude autor zmiňovat, jelikož pro účely tohoto systému nejsou využitelné. Nejvýznamnější vlastnost AWS je jeho flexibilita, umožňuje nastavit si vlastní operační systém, programovací jazyk, databázi, API (Application Programming Interface) a ostatní služby které mohou být k užitku. AWS je plně zabezpečené a obsahuje multiregionální zálohování, tím pádem je téměř nemožné o nahraná data přijít, či si je nechat odcizit. Jako téměř každé cloudové úložiště, i AWS je do určité míry nezpoplatněné. Výše plateb se pak derivují od zabíraného místa v úložišti, ale to je problém, který v této práci nebude potřeba řešit, jelikož velikost dat potřebná pro demonstraci funkčnosti zdaleka nedosáhne placené hranice. Jak autor výše zmiňuje, AWS bylo zvoleno i z osobních důvodů, a tím je autorova náročnost na design, kterou AWS splňuje.

7 Prezentační vrstva

Prezentační vrstva tvoří design a strukturu celého systému z pohledu uživatele.

7.1 Využité jazyky, techniky a vývojová prostředí

V prezentační vrstvě bylo také využito několik jazyků, a to zejména HTML a CSS. HTML sloužilo k vytvoření struktury systému a CSS naopak ke stylizaci. Vývojové prostředí, jak autor zmiňuje v x-té kapitole, bylo zvoleno SublimeText. Je velmi přehledné, uživatelsky příjemné a snadno se v něm orientuje, zejména díky mapě kódu. Software dokáže pracovat z mnoha jazyky a na základě toho i barví kód, což velmi napomáhá přehlednosti. SublimeText je také schopný upozorňovat na syntaxové chyby či pomocí pluginu vhodně doplňovat aktuálně psaný kód, což ušetří mnoho času.

Struktura bude tedy definována v HTML, jak je níže v kódu vidět, veškeré prvky jsou obalené v takzvaných kontejnerech. Div tag definuje nějakou sekci, či část. Všechny kontejnery a jiné prvky budou mít vlastní třídy, případně id. Pomocí tříd či id se k nim pak bude přistupovat v CSS, kde bude probíhat stylizace.

Obrázek 17: HTML struktura textu úvodní stránky

```
<div id="loginText">
<h1>Get great music, right now</h1>
          <h2>Listen to loads of songs for free</h2>
          <ul>
                    <li>Discover music you'll fall in love with</li>
                    <li>Create your own playlists</li>
                    <li>Follow artists to keep up to date</li>
          </ul>
</div>
```

Zdroj: vlastní zpracování

CSS může k prvkům přistupovat pomocí tříd nebo id, ale také se může obecně zaměřit na nějaký typ tagu, což ukazují následující ukázky kódu.

Obrázek 18: CSS stylizace jednoho určitého tagu

```
body {
    background-color: #181818;
    font-size: 14px;
    min-width: 720px;
}
```

Zdroj: vlastní zpracování

Následující kód znázorňuje stylizaci podle třídy a id.

Obrázek 19: CSS stylizace třídy, id a jejich elementů

```
#nowPlayingBarContainer {
          width: 100%;
    background-color: #282828;
    bottom: 0;
    position: fixed;
    min-width: 620px;
}
.controlButton {
          background-color: transparent;
    border: none;
    vertical-align: middle;
}

.controlButton.play img,
.controlButton.pause img {
          width: 32px;
          height: 32px;
}
```

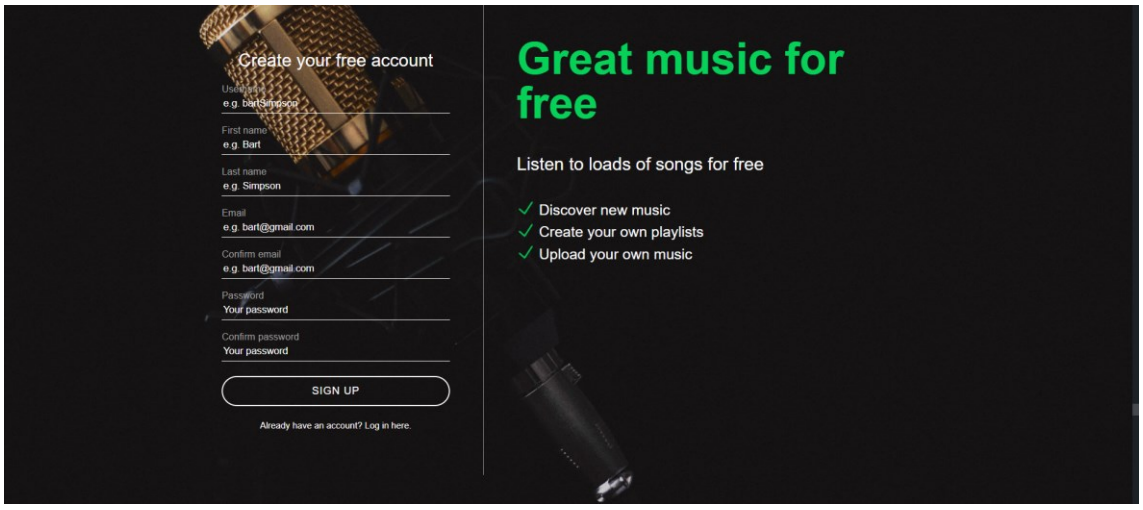
Zdroj: vlastní zpracování

7.2 Struktura úvodní stránky

Úvodní stránka systému bude rozdělena do levé a pravé části. V levé části se bude vyskytovat úvodní formulář, registrační nebo přihlašovací, podle volby uživatele. V pravé pak bude text představující tento systém.

Formulář má klasickou strukturu formuláře, tedy přihlašovací žádá uživatelské jméno a heslo a registrační pak celé jméno, uživatelské jméno, email a heslo. S tím, že email a heslo se zadává pro kontrolu dvakrát.

Obrázek 20: Design titulní stránky



Zdroj: vlastní zpracování

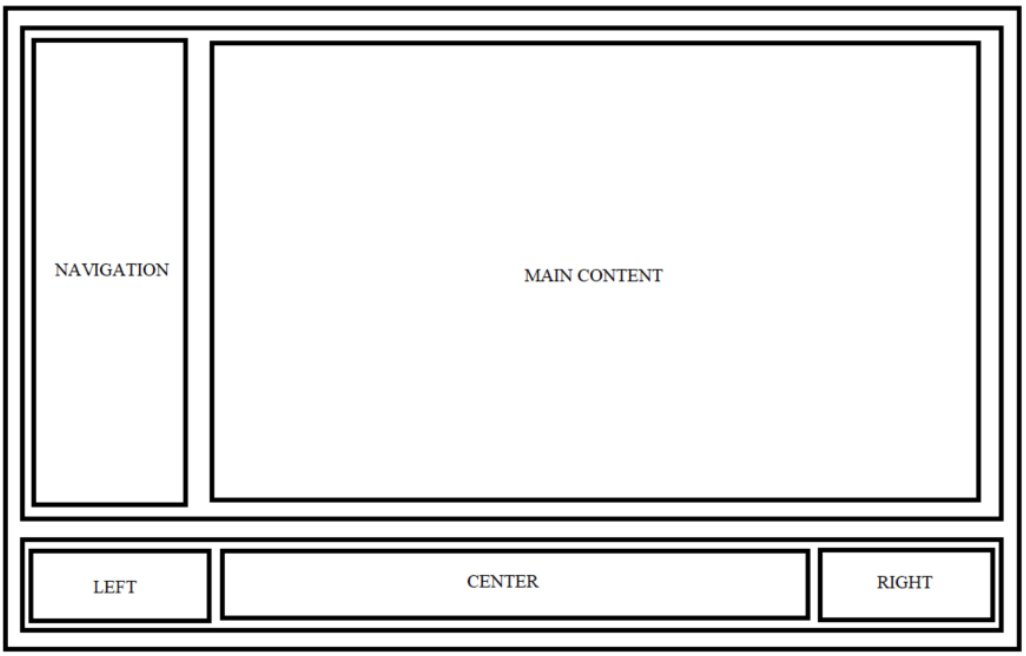
Stylizace bude probíhat velmi podobně jako u levé části s formuláři, avšak v pravé části budou zapotřebí ikony. Ikony je samozřejmě možné vytvořit nové, nebo stáhnout náhodně z internetu, v tomto případě však autorovi přišlo nejvýhodnější využít software Icons8. Software má pro tento systém využitelné funkcionality zdarma. Mezi funkcionality patří volba barvy a velikosti, která je ovšem omezená, ale jak je výše zmíněno, v tomto případě to nevadí.

7.3 Struktura a design systému z pohledu uživatele

Po dokončení výše popsané přihlašovací stránky je uživatel přesměrován na takzvanou indexovou stránku neboli výchozí stránku tohoto systému. Stránku index.php totiž prohlížeč automaticky bere jako stránku úvodní a automaticky se na ni přesměruje, pokud nedefinujeme, na jakou stránku má přejít. Název index.php ale i tak není určitě povinný.

Struktura prezentační stránku tohoto webového informačního systému nebude závažně odlišná od struktury ostatních hudebních platforem. A to z jednoduchého důvodu, struktura je přehledná, funkční a uživatelsky intuitivní a příjemná. Následující obrázek znázorňuje hrubou strukturu těchto stránek, která bude po celou dobu stejná.

Obrázek 21: Struktura systému



Zdroj: vlastní zpracování

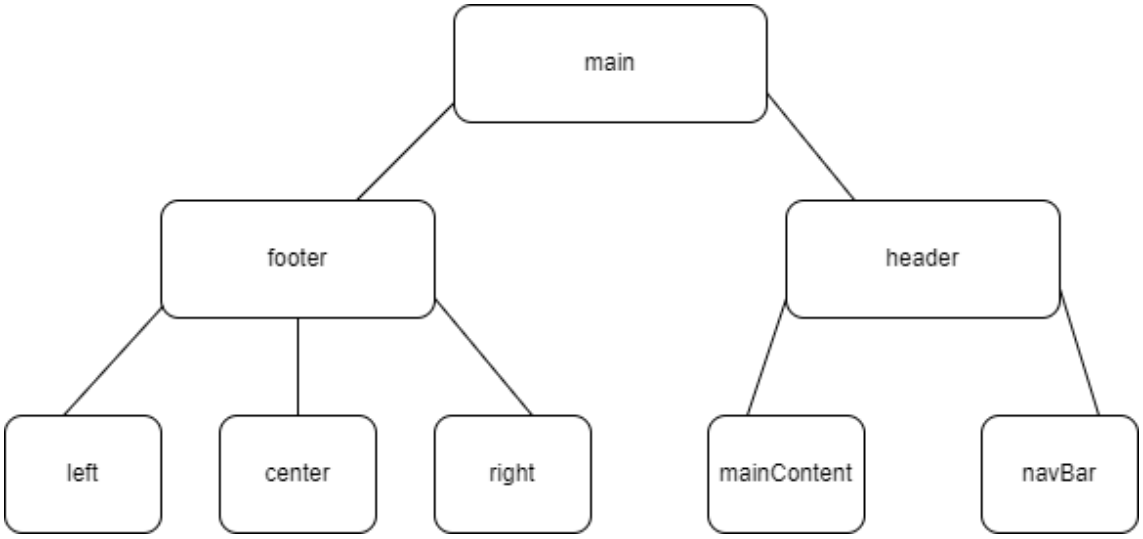
Z výše znázorněné struktury vyplývá, že by se stránka dala rozdělit různými způsoby,

následující diagram upřesňuje defaultní kostru. Hlavní obsah se jistě bude měnit podle toho, na jaké stránce se uživatel bude vyskytovat, ale kostra zůstává stále stejná.

Nejčastěji se stránky dělí na header, mainContent a footer. Jak je již patrné z výše/níže komentovaného obrázku, footer se dělí na tři sekce, pravou, střední a levou. Footer bude v tomto případě sloužit jako hudební přehrávač a je takto rozdělen z následujících důvodů

- Left – Obsahuje informace o právě hrané skladbě
- Center – Hudební přehrávač
- Righ – Ovládání hlasitosti

Obrázek 22: Diagram struktury



Zdroj: vlastní zpracování

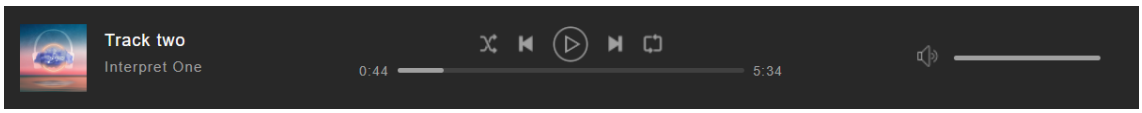
NavBar a footer po celou dobu nebudou měnit svůj obsah, pomineme-li hudební přehrávač – nowPlayingBar. Měnit se bude pouze ten hlavní obsah. Stránky si tedy budou „předávat“ NavBar a footer pomocí include. NavBar a footer bude rozdělen do souborů, aby se s nimi snadněji a rychleji manipulovalo.

Po definování základní struktury systému přichází na řadu stylizace jednotlivých částí. Stylizace zahrnuje nejen barevné prvky a ikony, ale také samotné chování těchto strukturálních prvků a jejich velikost. Pro celý dokument je nastaven výchozí font a barva fontu, která se ovšem dále specifikuje.

Footer neboli v tomto případech hudební přehrávač je tedy rozdělen do tří částí, v levé části se nachází informace o aktuálně hrající skladbě: název skladby, název interpreta a obal daného alba. V prostřední sekci pak budou ikonky pro ovládání přehrávače, tedy play, pause, next, previous, shuffle a repeat. Klíčové v této části pak bude naprogramovat

dynamické střídání ikon pause a play, shuffle-active a shuffle, repeat-active a repeat. Pod ikonami se bude nacházet průběh dané skladby a čas. V pravé sekci pak bude jen ovládání hlasitosti, také s progress barem. Opět je nutné nastavit dynamickou změnu ikony mute a mute-active. Oba dva progress bary budou umožňovat změnu po kliknutí či přetáhnutí myši.

Obrázek 23: Design hudebního přehrávače



Zdroj: vlastní zpracování

Po dokončení stylizace footeru následuje stylizace headeru – navigačního baru. Navigační bar se bude nacházet po levé straně, nikoliv na vrcholu stránky, jak bývá zvykem. Je to z důvodu větší přehlednosti a také proto, že kdyby byl header i footer v horizontální poloze, mohlo by se zdát, že je hlavní obsah úzký a nezobrazuje žádané množství informací.

V navigačním baru, jak už název vypovídá, budou hlavní orientační a navigační prvky systému. Uživatel zde bude mít možnost vyhledávat, procházet, tvořit playlisty, spravovat svůj profil a nahrávat nové skladby. Administrátor pak bude mít funkci navíc, a to schvalování či smazání nově nahraných skladeb. Také zde bude přítomno logo tohoto systému, které uživatele navede na indexovou stránku – v tomto případě je indexová stránka položka Browse – procházet. Sekcí bude v navigačním baru několik a všechny tyto sekce jsou rozděleny do souborů, právě kvůli snazší manipulaci a orientaci v kódu. Footer a header bude přítomen na každé stránce systému, autor by tedy musel tento kód kopírovat pro každou stránku, což by bylo velmi časově a kapacitně neefektivní.

Obrázek 24: Obsažení footeru a headeru

```
include("includes/header.php");
include("includes/footer.php");
```

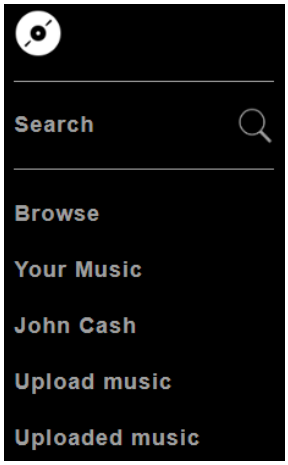
Zdroj: vlastní zpracování

Mezi tyto dvě obsažené stránky bude psán kód toho hlavního obsahu. Stránky v navigačním baru jsou následující:

- Logo systému – Po kliknutí uživatele přesměruje na úvodní, indexovou stránku.
- Search – Přesměruje na stránku search, která umožňuje vyhledávání v systému.
- Browse – Stejné jako indexová stránka, po kliknutí zobrazí deset náhodných alb.

- Your music – Zavede uživatele do sekce jeho playlistů, kde je může vytvářet, měnit jejich obsah a mazat je.
- User profile – Stránka pojmenovaná podle křestního jména a příjmení přihlášeného uživatele. Bude umožňovat odhlášení a změnu údajů.
- Upload music – Možnost nahrání vlastních nahrávek.
- Uploaded music – Položka pro administrátora kde může schvalovat nahrané skladby.

Obrázek 25: Design navigačního baru



Zdroj: vlastní zpracování

7.4 Struktura jednotlivých stránek

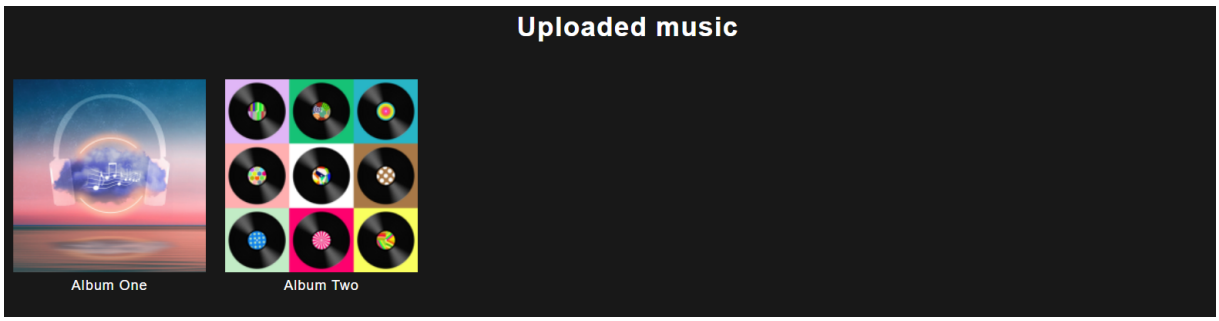
Výše zmíněná struktura hovoří pouze o obecném rozložení a designu celého systému. Jednotlivé stránky pak musí mít specifičtější design uzpůsobený pro jejich obsah. Ale v tomto případě budou nastávat situace, kde různé stránky budou mít stejný nebo velmi podobný design, bude tedy použit již předem napsaný CSS kód. Kupříkladu sekce album a playlist bude mít identickou strukturu, jelikož v obou dvou případech bude potřebný seznam skladeb. Analogicky, v sekci browse neboli index autor zamýšlí přehled alb, a stejný design je potřebný v sekci uploaded, kde administrátor schvaluje nově nahrané nahrávky. Stejnou strukturu jako přehled alb bude mít ale také stránka playlist, kde bude také potřeba přehled playlistů. V posledních dvou sekcích – User profile a Upload music bude již duplicita menší, ale některé věci se budou opakovat, jelikož v obou sekcích bude potřebný formulář a tlačítka. V celém systému jsou tedy potřebné nastylizovat čtyři části,

a to přehled alb, přehled nahrávek, formuláře a tlačítka. A kombinace všech těchto návrhů bude stránka s vyhledáváním.

7.4.1 Alba

U alb, jelikož jsou čtvercového charakteru, byla zvolena takzvaná mřížková struktura. Struktura byla opět napsána v HTML a stylizována v CSS. Alba budou v tomto přehledu ukazovat pouze svůj název, ostatní informace se zobrazí po rozkliknutí. Výsledek je následující:

Obrázek 26: Přehled alb



Zdroj: vlastní zpracování

Požadované mřížkové struktury bude dosaženo pomocí jednoduchého kaskádového stylu. Následující kód říká, že se položky v náhledu budou zobrazovat vedle sebe, budou mít určitou výšku a šířku a mezi nimi budou určité mezery. Margin dělá to, že zvětšuje mezeru mimo ten požadovaný objekt. To je důležité si uvědomit, neboť padding funguje přesným opakem, tedy že tvoří vnitřní mezeru.

Obrázek 27: Mřížková struktura

```
.gridViewItem {
    display: inline-block;
    margin-right: 20px;
    max-width: 200px;
    min-width: 150px;
    margin-bottom: 20px;
}
```

Zdroj: vlastní zpracování

7.4.2 Seznamy skladeb

Seznam skladeb po rozkliknutí alba bude potřebný téměř na každé stránce. Bude realizován logicky seznamem. Využité tagy budou tedy ul-seznam a li–položka seznamu. Každá položka seznamu neboli každá nahrávka v seznamu má zobrazit své pořadí na

albu, svůj název, interpreta a dobu trvání. Po přejetí myši se dodatečně zobrazí ikona play a ikona možnosti.

Obrázek 28: Design tracklistu

Zdroj: vlastní zpracování

7.4.3 Formuláře a tlačítka

Mimo registrační a přihlašovací formuláře, které jsou popsány v úvodní stránce, budou v systému využity dva další formuláře, a to formulář na změnu přihlašovacích údajů a formulář pro nahrání skladeb. U každého formuláře bude nutná přítomnost tlačítka sloužícího k odeslání formuláře. Tyto tlačítka budou stylizována obecně pro celý systém. Formuláře budou vytvořeny stejně jako předchozí formuláře, jak je níže vyobrazeno.

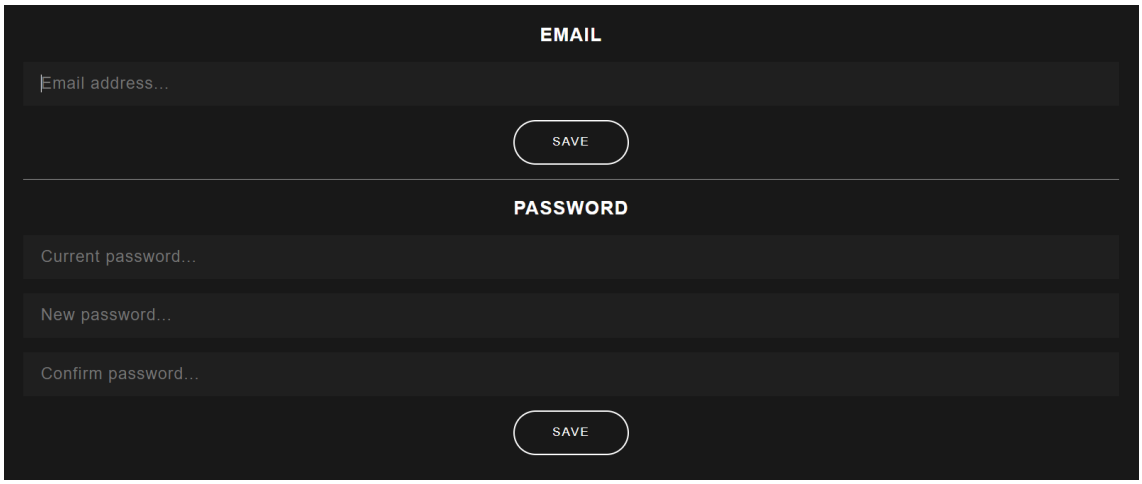
Obrázek 29: Formulář pro změnu emailu a hesla

```
<div class="userDetails">
          <div class="container borderBottom">
                    <h2>EMAIL</h2>
                    <input type="email" class="email" name="email" placeholder="Email
address..." value="<?php echo $userLoggedIn->getEmail(); ?>">
                    <span class="message"></span>
                    <button class="button" onclick="updateEmail('email')">SAVE</button>
          </div>

          <div class="container">
                    <h2>PASSWORD</h2>
                    <input type="password" class="oldPassword" name="oldPassword"
placeholder="Current password...">
                    <input type="password" class="newPassword1" name="newPassword1"
placeholder="New password...">
                    <input type="password" class="newPassword2" name="newPassword2"
placeholder="Confirm password...">
                    <span class="message"></span>
                    <button class="button"
onclick="updatePassword('oldPassword','newPassword1','newPassword2')">SAVE</button>
          </div>
</div>
```

Zdroj: vlastní zpracování

Obrázek 30: Design formuláře



Zdroj: vlastní zpracování

Obrázek 31: CSS kód pro design tlačítka

```
.button {
    color: #fff;
    cursor: pointer;
    margin-bottom: 10px;
    background-color: transparent;
    font-weight: 500;
    letter-spacing: 2px;
    border: 2px solid #fff;
    border-radius: 500px;
    padding: 15px;
    min-width: 130px;
}
```

Zdroj: vlastní zpracování

7.4.4 Vyhledávácí stránka

Jak autor zmiňuje v úvodu této podkapitoly, vyhledávací stránka kombinuje téměř všechny prvky využité v předchozích. Základem je textový vstup, kam uživatel zadává, co chce po systému vyhledat. Výsledky vyhledávání budou vyobrazeny ve třech kategoriích – Interpret, nahrávka a album. Tyto sekce již ale byly stylizovány v předchozích podkapitolách, je tedy možné využít již předem napsaný kód a modifikovat ho pro tuto vyhledávácí stránku.

8 Aplikační vrstva

8.1 Zvolené vývojové prostředí

Vývojové prostředí bylo zvolené stejně jako pro prezentační část, a to SublimeText. Důvody pro zvolení tohoto prostředí jsou identické s důvody, které autor zmiňuje v kapitole Prezentační vrstva.

8.2 Využité programovací jazyky a techniky

Při implementaci tohoto systému bylo využito mnoho jazyků, technik a konceptů. Základní funkčnost obstarával jazyk PHP, který právě umožňuje komunikaci mezi databázovou a prezentační vrstvou. PHP skript má ale jednu nevýhodu, a to že takový skript se spouští, když se stránka načítá. Tato vlastnost by jistě nevadila a nevadí u jiných systémů, ale v tomto systému je žádoucí co nejméně znovunačtení. PHP v tomto systému slouží zejména pro komunikaci s databází a práci s formuláři. Bezpochyby je také klíčovým prvkem u přihlašování a udržení přihlášení. Ovšem pro dynamické či asynchronní změny nelze PHP použít. Z toho důvodu byl velmi hojně využíván JavaScript, jeho knihovny, koncepty a techniky.

- JavaScript
- JSon
- JQuery
- Ajax calls

8.3 Úvodní formuláře

Titulní stránka tohoto systému se skládá z přihlašovacího/registračního formuláře. Mezi formuláři lze dynamicky přepínat pomocí JavaScriptu, spíše je to pomocí knihovny JQuery, která umožňuje bez znovunačtení změnit obsah stránky.

Po vytvoření těchto dvou formulářů, kdy je v přihlašovacím uživatel tázán na přihlašovací jméno a heslo, a v registračním na přihlašovací jméno, křestní jméno a příjmení, heslo a email.

Ve formuláři musí existovat validace a sanitace zadaných údajů, ať už z důvodu bezpečnosti, ale i z důvodu, aby uživatel nezadal něco, co nebylo v jeho úmyslu. Nejprve

bylo přistoupeno k sanitaci formuláře, jednalo se zejména o eliminaci možnosti zadání takzvaných nebezpečných znaků neboli znaků, pomocí kterých by šikovný uživatel mohl do formuláře zadat nějaký nebezpečný skript. Dále se bude jednat o funkci, která když uživatel omylem zadá prázdný znak, nahradí žádným znakem. Poslední sanitační funkce bude jen u křestního jména a příjmení, kdy bude žádané, aby první písmeno bylo velké.

Následně se přistupuje k validaci vstupů, které uživatel zadal. Nejprve je vytvořena třída Account, která nese informace o objektu uživatele a poskytuje právě funkce na validaci vstupů, ale také na přihlášení a registraci, tedy samotné nahrání uživatelských dat do databáze.

Validace obnáší zejména kontrolu délky vstupů, kupříkladu aby nějaký uživatel neměl uživatelské jméno dlouhé jeden znak, anebo v opačném extrému. Délka vstupu bude kontrolována u všeho, kromě emailu. U emailu se bude jednat o kontrolu správného formátu pomocí zabudované funkce FILTER_VALIDATE_EMAIL. Email se kvůli bezpečnosti zadává dvakrát, bude tedy nutné ověřit shodu a v poslední řadě také jedinečnost v databázi. U hesla, jak už tomu obvykle bývá, bude primárně kontrola typu znaků, a provedena bude pomocí takzvaného regulérního výrazu [^A-Za-z0-9], který právě říká, že v hesle smí být pouze písmena a čísla. Na závěr opět proběhne test na shodu dvou hesel.

S tím souvisí vypisování chybových zpráv, aby uživatel věděl, ve kterém vstupu má chybu. Bude toho docíleno pomocí pole errorArray(), do které se v případě splnění podmínky na chybu uloží chybová zpráva. Pomocí funkce getError se pak tato zpráva bude v poli hledat, pokud narazí na shodu textového řetězce, vypíše ho vedle toho chybně zadaného vstupu.

Obrázek 32: Vypsání chybové zprávy

```
<?php echo $account->getError(Constants::$usernameTaken); ?>
```

Zdroj: vlastní zpracování

Pro usnadnění je vytvořena nová třída Constants, kde budou tyto zprávy ukládány. Je to velmi praktické, neboť pokud bude chtít autor, nebo kdokoliv jiný změnit znění chybové zprávy, stačí mu změnu udělat na jednom místě.

Při většině registrací dochází k chybám poměrně často, z tohoto důvodu považuje autor za vhodné vložit funkci, která si bude pamatovat předtím zadané správné vstupy, aby je uživatel nemusel vypisovat znovu.

Přihlašovací formulář má následující strukturu, jen dvě políčka pro uživatelské jméno a heslo.

Obrázek 33: Přihlašovací formulář

```
function getInputValue($name) {
                    if(isset($_POST[$name])) {
                              echo $_POST[$name];
                    }
          }
<form id="loginForm" action="register.php" method="POST">
                              <h2>Login to your account</h2>
                              <p>
                    <label for="loginUsername">Username</label>
                    <input id="loginUsername" name="loginUsername" type="text"
placeholder="e.g. JohnSmith" value="<?php getInputValue('loginUsername') ?>" required>
                                                  </p>
                                                  <p>
                              <label for="loginPassword">Password</label>
          <input id="loginPassword" name="loginPassword" type="password" placeholder="Your
password" required> </p>

                    <button type="submit" name="loginButton">LOG IN</button>
                    <div class="hasAccountText">
                    <span id="hideLogin">Don't have an account yet? Signup here.</span>
                                                  </div>
                                        </form>
```

Zdroj: vlastní zpracování

U registračního formuláře je struktura nepatrně rozsáhlejší, neboť, jak je zmíněno výše, musí probíhat důkladná validace a sanitace.

Jak je výše zmíněno autorem, formuláře se budou dynamicky střídat, podle potřeby uživatele systému. Aby bylo dosaženo střídání bez znovunačtení, je nutné využít knihovnu JavaScriptu – JQuery.

Před využitím JQuery je ale nutné tuto knihovnu importovat, a to pomocí následujícího skriptu. JQuery je zdarma poskytována googlem, stačí tedy pouze zkopírovat volně dostupný odkaz a vložit do souboru jako skript. Důvod použití JQuery je, že samotný JavaScript není schopný poznat, jestli bylo PHP tlačítko stisknuto.

Obrázek 34: Dynamická změna formulářů

```
1.  if(isset($_POST['registerButton'])) {
2.              echo '<script>
3.              $(document).ready(function() {
4.                        $("#loginForm").hide();
5.                        $("#registerForm").show(); });
6.                        </script>';
7.    }
8.    else {
9.              echo '<script>
10.             $(document).ready(function() {
11.                       $("#loginForm").show();
12.                       $("#registerForm").hide(); });
13.                       </script>';
14.   }
15.   ?>
```

Zdroj: vlastní zpracování

Po všech předchozích procesech, tj. zejména validaci a sanitaci vstupních dat, je možné přistoupit k samotnému nahrání dat do databáze. Funkce to je insertUserDetails, a opět se bude vyskytovat ve třídě Account.php a přistupovat se k ní bude vytvořeným objektem uživatele.

Obrázek 35: Funkce vkládající uživatelské údaje do databáze

```
private function insertUserDetails($un, $fn, $ln, $em, $pw) {
          $date = date("Y-m-d");
          $result = mysqli_query($this->con, "INSERT INTO users VALUES ('', '$un', '$fn',
'$ln', '$em', '$pw', '$date')");

                    return $result;
                    }
```

Zdroj: vlastní zpracování

Když je uživatel registrován, bude mu umožněno se přihlásit. Podobně jako byl výše kód pro manipulaci s registračním formulářem, zde se bude jednat o manipulaci s tím přihlašovacím. V login-handler ve své podstatě jen autor dodává username a password skrze globální proměnnou $_POST. Tyto vstupy jsou nejprve zkontrolovány sanitační funkcí. Následně je pomocí objektu $account volána funkce login ze třídy Account.php, která ověří, zda se zadané uživatelské jméno a heslo shoduje s daty v databázi, a následně vrátí výsledek v podobě true či false.

Obrázek 36: Ověření přihlášení

```
$query = mysqli_query($this->con, "SELECT * FROM users WHERE username='$un' AND
password='$pw'");
```

Zdroj: vlastní zpracování

Kód, který vezme data z databáze a vrátí počet shod, v případě true by shoda měla být

jedna, jelikož by měl existovat pouze jeden uživatel s daným uživatelským jménem.

Pokud funkce najde právě jednu shodu, jednoho uživatele, nastartuje takzvanou SESSION, s parametrem userLoggedIn a uživatele přihlásí – přesměruje na úvodní stránku samotného systému.

Obrázek 37: Přihlášení uživatele

```
<?php
if(isset($_POST['loginButton'])) {
          //Login button was pressed
          $username = $_POST['loginUsername'];
          $password = $_POST['loginPassword'];
          $result = $account->login($username, $password);
          if($result == true) {
                    $_SESSION['userLoggedIn'] = $username;
                    header("Location: index.php");
          }
}
?>
```

Zdroj: vlastní zpracování

Přihlášení a registrace je nyní plně funkční a dynamické, je tedy záhodno přistoupit ke stylizaci této úvodní stránky s těmito dvěma formuláři. Stylizace, jak je zmíněno v kapitole prezentační část, bude probíhat v jazyce CSS – v kaskádových stylech.

8.4 Obecné funkcionality systému

8.4.1 Dynamické přepínání stránek

Jak autor již několikrát výše zmiňoval, jediný obsah, který se v systému mění, je ten prostřední – mainContent. Například při kliknutí na jinou stránku v navigačním baru je žádoucí, aby aktuálně hrající nahrávka nepřestala hrát, či se dokonce nezměnila. Tohoto dynamického charakteru bude opět dosaženo pomocí JQuery. JQuery má zabudované funkce Ajax. Ajax bez toho, že by se stránka změnila nebo dokonce znovunačetla, získá obsah té žádané stránky, na kterou bylo kliknuto, a následně jsou obsahy vyměněny pomocí JQuery. Vymění se tedy pouze hlavní obsah.

S tím souvisí funkce openPage(). Kdy presměrování na jinou stránku bude zajištěno právě pomocí této funkce. Pokud by byl ponechán jako odkaz tag „a href“, stránka by se znovu načítala, což je nežádoucí.

Obrázek 38: OpenPage funkce

```
function openPage(url) {
          if(url.indexOf("?") == -1) {
                    url = url + "?";
          }
          var encodedUrl = encodeURI(url + "&userLoggedIn=" + userLoggedIn);
          console.log(encodedUrl);
          $("#mainContent").load(encodedUrl);
          $("body").scrollTop(0);
          history.pushState(null,null,url);
          if(timer!=null){
                    clearTimeout(timer);
          }
}
```

Zdroj: vlastní zpracování

Na kliknutí se zavolá funkce openPage, kde parametr bude právě stránka, kterou chce uživatel otevřít.

Teď ale nastávají dvě situace načtení stránku, první možností je pomocí Ajax volání, a druhou je klasický reload stránky pomocí ikony v prohlížeči. Je nutné ošetřit obě dvě možnosti. Pokud je změna provedena pomocí Ajax, stačí zahrnout pouze header.php a footer.php. Ale pokud se jedná o klasický reload, stránka znovu načítá celý obsah, tedy vlastně potřebuje zahrnout všechny ostatní třídy. Tohoto procesu je dosaženo velmi jednoduše, vytvořením souboru includedFiles.php, kde budou tyto možnosti ošetřeny.

Obrázek 39: Included files

```
<?php

if(isset($_SERVER['HTTP_X_REQUESTED_WITH'])) {
          include("includes/config.inc");
          include("includes/classes/User.php");
          include("includes/classes/Artist.php");
          include("includes/classes/Album.php");
          include("includes/classes/Song.php");
          include("includes/classes/Playlist.php");
          include("includes/classes/Uploaded.php");
          if(isset($_GET['userLoggedIn'])) {
$userLoggedIn = new User($con, $_GET['userLoggedIn']);
          }
          else {
echo "Username variable was not passed into page. Check the openPage js function.";

          }
}
else {
          include("includes/header.php");
          include("includes/footer.php");
          $url = $_SERVER['REQUEST_URI'];
          echo "<script>openPage('$url')</script>";
          exit();
}

?>
```

Zdroj: vlastní zpracování

8.4.2 Možnosti nahrávek

Každá nahrávka bude mít ikonu možností, která bude mít velmi využívanou podobu tří teček. Možností bude několik a budou se lišit podle toho, na jaké stránce budou možnosti využívány a také podle toho, jestli uživatel má administrátorská práva, či nikoliv.

Uživatelské možnosti nahrávky jsou následující:

- Přidání nahrávky do libovolného playlistu – Tato možnost bude funkční na všech stránkách, například na stránce alba a vyhledávání. Ale bude figurovat i na stránce playlistu, neboť je možné že uživatel bude mít potřebu nahrávky přidat do více playlistů.
- Odebrání nahrávky z playlistu – Tato možnost bude logicky dostupná pouze na stránce playlistu.

Administrátor má několik možností navíc, ale sdílí i ty uživatelské.

- Smazání nahrávky – Kdekoliv na stránce má administrátor možnost smazat jakoukoliv skladbu. Skladba se kompletně vymaže z databáze.
- Schválení či neschválení nahrané skladby – Když uživatel nahraje skladbu, či skladby, musí být nejprve zkontrolovány administrátorem. Možnost bude viditelná jen na stránce s nahranými skladbami.

Obrázek 40: Možnosti skladby

```
function showOptionsMenu(button) {
          var songId = $(button).prevAll(".songId").val();
          var menu = $(".optionsMenu");
          var menuWidth = menu.width();
          menu.find(".songId").val(songId);

          var scrollTop = $(window).scrollTop();
          var elementOffset = $(button).offset().top;
          var top = elementOffset - scrollTop;
          var left= $(button).position().left;
          menu.css({"top": top+"px", "left":left - menuWidth+"px", "display": "inline" });
}
```

Zdroj: vlastní zpracování

Funkčnost těchto možností je implementována pomocí JavaScriptu a konceptu Ajax. Autor zde požaduje, aby se výše zmíněné činnosti jako je například přidání skladby do playlistu vykonávaly bez nutnosti znovunačtení stránky. JavaScriptová funkce bude volat php kód, který zajišťuje funkcionalitu těchto možností.

Obrázek 41: Funkce smazat skladbu

```
function removeSong(button, albumId){
          var songId = $(button).prevAll(".songId").val();
          $.post("includes/handlers/ajax/removeSong.php", { albumId: albumId, songId:
songId}).done(function(error){
          if(error!=""){
                    alert(error);
                    return;
                    }
          openPage("album.php?id="+albumId);
          })
}
```

Zdroj: vlastní zpracování

Výše je znázorněna JavaScriptová funkce, která zjistí id nahrávky a následně zavolá PHP funkci, která zajistí funkčnost.

Obrázek 42: Ajax skript pro smazání skladby

```
<?php

include("../../config.inc");

if(isset($_POST['albumId']) && isset($_POST['songId'])){
          $albumId = $_POST['albumId'];
          $songId = $_POST['songId'];
          $sql = "DELETE FROM songs WHERE album='$albumId' AND id = '$songId'";
          $query = mysqli_query($con, $sql);

          $sql = "DELETE FROM playlistSongs WHERE songId='$songId'";
          $query = mysqli_query($con, $sql);
}
else {
          echo "PLaylistId or songId parameters not passed into removeFromPlaylist.php";
}
?>
```

Zdroj: vlastní zpracování

Funkce klasicky pracuje s databází, jako v předchozích řešeních, jen bylo nutné tuto operace provést asynchronně. Po vykonání PHP kódu vrátí JavaScriptová funkce uživatele na stejnou stránku.

8.5 Funkcionality jednotlivých stránek a tříd

Hlavní obsah bude označován jako mainContent, který, jak autor výše zmiňuje, se bude měnit. Hlavní obsah bude mít několik stránek, a to budou následující: album, artist, browse, index, playlist, search, settings, updateDetails, upload, uploaded a yourMusic.

Jedná se o stránky, nikoliv o třídy, proto názvy začínají malým písmenem.

Stránka index pouze referuje na stránku browse, jelikož autor chtěl, aby výchozí stránka

hned ukazovala určitý počet alb. Alba budou náhodně vybírána z databáze a budou omezena určitým limitem, aby uživatel nebyl přehlcen. Do databáze se přistupuje přes proměnnou $con, která byla definována v souboru config, který je zahrnut ve všech ostatních souborech (stránkách). Zbývá tedy vytvořit dotaz, který alba z databáze převezme, bude vypadat následovně.

Obrázek 43: Převzetí dat z databáze

```
<?php
$albumQuery = mysqli_query($con, "SELECT * FROM albums ORDER BY RAND() LIMIT 20");
          while($row = mysqli_fetch_array($albumQuery)) {
          echo "<div class='gridViewItem'>
          <span role='link' tabindex='0'
onclick='openPage(\"album.php?id=".$row['id']."\")'>
                    <img src='".$row['artworkPath']."'>
                    <div class='gridViewInfo'>". $row['title'] ."</div>
                                        </span>
                              </div>";
          }
?>
```

Zdroj: vlastní zpracování

Z dotazu je patrné, že alba jsou z databáze vybrána vždy náhodně a uživatel tedy vždy může objevit něco nového.

Po dokončení funkčního kódu této stránky musí být opět stylizována, aby se alba zobrazovala ve správném formátu a pozici. Název třídy je gridViewItem, bude se tedy jednat o jakési mřížkové zobrazení.

Je žádoucí, aby alba byla navázána na odkaz, který zobrazí album a všechny skladby v něm. Nejprve je ale nutné mít vytvořenou stránku, na které se tyto informace budou zobrazovat, stránka bude jednoduše pojmenována album.php. První, co bude muset stránka zjistit, je Id alba, na které bylo kliknuto. Id je předáváno pomocí URL adresy, je tedy postačující jedna podmínka:

Obrázek 44: Podmínka pro id alba

```
if(isset($_GET['id'])) {
          $albumId = $_GET['id'];
}
else {
          header("Location: index.php"); }
```

Zdroj: vlastní zpracování

Informace na stránce zobrazované budou získávány pomocí takzvaného objektu. Objekt je instance nějaké třídy, kdy třída umožňuje s daným objektem pracovat. Třída má funkci konstruktor, který vytvoří požadovaný objekt, v tomto případě objekt alba. Objekt bude mít pouze dva parametry, a to proměnnou připojení k databázi $con a $id.

Obrázek 45: Třída alba

```
class Album {

                    private $con;
                    private $id;
                    private $title;
                    private $artistId;
                    private $genre;
                    private $artworkPath;

                    public function __construct($con, $id) {
                              $this->con = $con;
                              $this->id = $id;
                              $query = mysqli_query($this->con, "SELECT * FROM albums WHERE
id='$this->id'");
                              $album = mysqli_fetch_array($query);
                              $this->title = $album['title'];
                              $this->artistId = $album['artist'];
                              $this->genre = $album['genre'];
                              $this->artworkPath = $album['artworkPath'];
                    }
```

Zdroj: vlastní zpracování

Referenční proměnná this ukazuje, že další proměnná je součástí právě tohoto (this) objektu/instance. Důvod, proč to autor řeší přes objekty je ten, že je to mnohem jednodušší, přehlednější a časově a výpočetně efektivnější. K informacím o albu je přistupováno pomocí funkcí v třídě. Funkce mohou vypadat například tímto způsobem:

Obrázek 46: Funkce třídy Album

```
public function getGenre() {
          return $this->genre;
}
```

Zdroj: vlastní zpracování

Na samotné stránce alba je pak k těmto informacím přistupováno pomocí vytvořeného objektu, který je vytvořen a volá se pomocí něj následovně:

Obrázek 47: Práce s objektem alba

```
$album = new Album($con, $albumId);
$artist = $album->getArtist();
```

Zdroj: vlastní zpracování

K nahrávkám přítomným v albu se přistupuje stejně, metoda getSongIds() vrátí pole se všemi skladbami a následně jsou pomocí cyklu vypsány. Každá nahrávka bude mít informaci o názvu, názvu interpreta, pořadí na daném albu a také délku trvání. Když uživatel najede na jakoukoliv nahrávku, zobrazí se mu dodatečné možnosti, zejména ikona play a v pravém rohu ikona s možnostmi. Všechno je opět stylizované v CSS.

Kdekoliv na stránce bude možné kliknout na obal alba či název interpreta a uživatel bude přesměrován na požadovanou stránku.

Obrázek 48: Dynamický odkaz na stránku

```
<span role='link' tabindex='0' onclick='openPage(\"album.php?id=".$row['id']."\")'>
```

Zdroj: vlastní zpracování

Obrázek je uzavřen ve spanu, který funguje jako odkaz a po kliknutí se odkáže na funkci openPage, která otevře stránku alba, na které uživatel klikne. Id je předáváno přes URL.

Následuje vytvoření třídy Artist.php, která, podobně jako třída Album.php, uchovává informace objektu interpreta. Třída opět obsahuje konstruktor a metody getName(), getId() a getSongIds(). V metodě getSongIds() jde o to, aby se u interpreta vždy zobrazovalo několik nejhranějších nahrávek. Počet přehrání autor řeší v následující kapitole.

Stránka artist.php bude zobrazovat informace o interpretovi, tedy název, seznam deseti nejpřehrávanějších skladeb a seznam alb.

Obrázek 49: Funkce vracející id nahrávek

```
public function getSongIds() {
          $query = mysqli_query($this->con, "SELECT id FROM songs WHERE artist='$this->id'
ORDER BY plays ASC");
          $array = array();
          while($row = mysqli_fetch_array($query)) {
                    array_push($array, $row['id']);
                              }
          return $array;
                    }
```

Zdroj: vlastní zpracování

Po třídách Album.php a Artist.php bude logicky navazovat třída Song.php, která bude uchovávat informace o určité skladbě. Třída bude mít velmi podobnou strukturu jako dvě zmíněné předtím. Funkce budou stejného charakteru, jen u nahrávky bude autor zjišťovat také duration a path.

8.5.1 Album

Na stránce album jsou klíčové následující informace – Název a obal alba, počet skladeb na albu, název interpreta a seznam skladeb na albu. Funkcionalita bude zajištěna dotazem, který podle albumId vrátí všechny skladby s tímto cizím klíčem. Id alba bude předáno přes URL stránky. Tyto dotazy ale nebudou muset být napsány znovu na této stránce, protože jsou již obsaženy ve třídě Album. Postačující bude tedy tvorba objektu alba, s předaným parametrem $albumId. Pomocí objektu alba pak systém vrátí id patřící nahrávkám tohoto alba a následně je vypíše.

8.5.2 Interpret

Po kliknutí na název interpreta, kdekoliv na stránce, se zobrazí stránka artist.php, která bude zobrazovat název interpreta, bude mít možnost play, zobrazí deset nejhranějších skladeb a seznam všech alb daného interpreta. Id intepreta bude opět zjištěno z URL. Informace budou opět získány skrze vytvoření objektu interpreta, přes který budou volány metody na získání dalších informací. Metody ve třídě jsou stejného charakteru jako u Alba.

Obrázek 50: Funkce ve třídě Artist

```
public function getName() {
                                $artistQuery = mysqli_query($this->con, "SELECT name FROM
artists WHERE id='$this->id'");
                                $artist = mysqli_fetch_array($artistQuery);
                                return $artist['name'];
                    }
```

Zdroj: vlastní zpracování

8.5.3 Nahrávka

Nahrávka nebude mít vlastní stránku, ale určitě bude mít vlastní metody. Analogicky jako u alba a interpreta, u nahrávky je vytvořena třída s názvem Song.php. Vytvořena bude pouze třída, nikoliv stránka. Třída umožní získávat informace o zvolené nahrávce stejným způsobem, jako ve dvou předchozích kapitolách.

Obrázek 51: Funkce ve třídě Song

```
public function getTitle() {
                                return $this->title;
                    }
```

Zdroj: vlastní zpracování

8.5.4 Nahrávání

Nahrávání je funkcionalita, která by měla tvořit přidanou hodnotu tohoto systému. Tato stránka se bude tedy velmi lišit od ostatních, jelikož u ostatních se obsah pouze zobrazoval, zde se bude obsah nahrávat. Je vytvořena stránka upload.php, která nejenže obsahuje vlastní formulář, ale také JavaScript s částečným řešením funkcionality. Skript není v odděleném souboru jako tomu bylo doposud, jelikož pro autora této práce je to tak přehlednější.

Cílem této stránky je možnost nahrát vlastní nahrávky, čehož je docíleno formulářem. Formulář nabízí tři textové vstupy, pro název interpreta, alba a názvu. Dále je zde

umístěna kolonka pro nahrání obalu alba. Jako poslední je řádek, ve kterém uživatel nahraje soubor a zároveň vedle do textového vstupu zadá jeho název. Nejníže se bude vyskytovat tlačítko, které v případě uživatelovi potřeby přidá další vstupní pole pro nahrávku. Tyto pole je samozřejmě možné dynamicky mazat. V závěru je přítomno tlačítko Upload, tedy nahrát.

Prvním krokem po napsání struktury a formuláře bylo zajištění dynamického přidávání a odebírání polí pro nahrání nahrávky. Funkce addMore přidává další vstup.

Obrázek 52: Funkce přidávající další pole formuláře pro nahrávání

```
function addMore(){
    var html = '<div class="input-group" style="margin-bottom:10px;"><input type="file"
style="width:300px;" name="file[]"><a href="javascript:void(0);" onclick="delfile(this);"
style="border-top-left-radius:0px;border-bottom-left-radius:0px; >Remove</a><input
type="text" class ="uploadTrack" name="track" placeholder="Name of the current
track"></div>';
    $("#uploaddiv").append(html);
  }
```

Zdroj: vlastní zpracování

Obrázek 53: Funkce, která další vstup odebírá

```
function delfile(id) {
    $(id).parent().remove();
  }
```

Zdroj: vlastní zpracování

A právě toho dynamické přidávání a odebírání vstupů je dosaženo pomocí JQuery a Ajax volání, protože jak autor několikrát zmiňoval, Ajax umožňuje provádět asynchronní změny bez nutnosti znovunačtení stránky.

Obrázek 54: JQuery které které tyto změny umožňuje provádět dynamicky

```
  $(document).ready(function() {
    $("#upload").click(function(e) {
      e.preventDefault();
      var formData = new FormData($(this).parents('form')[0]);
      $.ajax({
        url:'upload2.php',
        type:'POST',
        success: function (data){
          alert("Data uploaded: "+data); },
        data:formData,
        cache:false,
        contentType: false,
        processData: false
      });
      return false;
    }) });
```

Zdroj: vlastní zpracování

Následuje implementaci kódu, který nahrávky již skutečně nahrává do databáze, respektive informace a cestu. Jelikož se jedná o formuláře, musí být opět provedena

sanitace a validace zadaných dat, ale i souborů. U souborů je kontrolována velikost a typ. Po těchto mandatorních kontrolách musí následovat i logické kontroly, a to zejména zjištění, jestli už uživatelem zadané informace v databázi nejsou. Například autor nechce mít v systému dva interprety se stejným jménem. U názvu alba a nahrávky je už ale shoda povolena. Funkce také musí zajistit to, aby se v případě negativního výsledku duplicity nový údaj nahrál do databáze. To bude opět provedeno dotazem. Po kontrole jedinečnosti interpreta nastává kontrola žánru. Pokud již žánr s tímto názvem existuje, není znovu nahrán záznam, ale je pouze uchováno id žánru, které je následně předáno dotazu, který nahrává nahrávky do databáze. V opačném případě je nový žánr nahrán a dotazu je předáno id nového žánru. Po těchto krocích následuje přidání alba do databáze, a zde je pouze třeba zkontrolovat, jestli dané album už u daného interpreta není. Pokud ne, jsou mu předány id interpreta, id žánru a obal alba a je nahráno do databáze. Po učinění těchto přípravných kroků je možné přistoupit k nahrání samotných skladeb do databáze.

Jak je zmiňováno v předchozích kapitolách, nahrané soubory budou ukládány na cloud AWS – Amazon Web Services. AWS nabízí mnoho služeb, pro tuto práci bude muset být využit pouze S3 – Simple Storage Service. V S3 se vytvoří takzvaný bucket, což je to vlastní úložiště, pojmenován bude bakalarskaprace a do něj se budou nahrávat nahrávky a obaly alb. Pro nahrávání bude ještě nutná instalace AWS SDK for PHP. Nainstalována bude do stejné složky v Xamppu, kde se nachází celý projekt. Na stránce upload pak bude zahrnován skrytý soubor config.inc, který v sobě ponese informace o připojení k autorovu cloudovému úložišti, k tomu slouží dva klíče, veřejný a soukromý. V tomto souboru bude zahrnut i soubor aws-autoloader.php, který poskytuje metody nutné k nahrávání a získávání dat z cloudu. Popis těchto metod a funkcí lze nalézt na oficiálních stránkách aws. Do databáze tedy bude ukládána cesta do cloudového úložiště a úložiště vygeneruje url pro získání požadovaného souboru. K souborům se tedy bude přistupovat stejně, jako kdyby byly nahrané na lokálním úložišti.

Obrázek 55: Funkce nahrávání souborů s využitím S3

```
function uploadFile($file, $category)
{
          global $aws_key, $aws_secret_key;
          $client = new Aws\S3\S3Client([
                    "version" => "latest",
                    "region" => "eu-west-1",
                    "credentials" => [
                    "key" => $aws_key,
                    "secret" => $aws_secret_key
                    ]
          ]);
          $path = "assets/$category/$file[name]";
```

```
        $uploader = new Aws\S3\MultipartUploader($client, $file["tmp_name"], [
                "Bucket" => "bakalarskaprace",
                "Key" => $path
        ]);
        $result = $uploader->upload();
        if ($result["@metadata"]["statusCode"] !== 200) exit("File $file[name] failed to
upload to cloud storage.");
        return $path;
}
```

Zdroj: vlastní zpracování

Jelikož se jedná o formulář, i zde se vyskytuje kontrola vstupů, analogicky stejně jako u ostatních formulářů. Kontrola souborů probíhá ověřením přípony souboru, jelikož jsou povolené pouze určité typy.

8.5.5 Nahrané nahrávky

V sekci nahraných nahrávek – uploaded.php, bude administrátor schvalovat uživatelské nahrávky. Struktura stránky bude stejná jako struktura browse, a po rozkliknutí alba se zobrazí jeho tracklist. Tyto funkcionality již byly implementovány dříve, autorovi tudíž stačilo je pouze zkopírovat s lehkými úpravami. Změna nastává u možností každé nahrávky, administrátor bude mít na výběr dvě možnosti – Approve song a Disapprove song, tedy schválit či neschválit. Tyto metody budou analogicky stejné jako RemoveSong.php a AddSongToPlaylist.php. V případě neschválení budou skladby rovnou smazány a v případě schválení jim bude atribut visibility změněn na 0, což znamená že se přesunou do viditelné sekce pro všechny uživatele.

8.5.6 Playlist

Další klíčovou a uživatelsky velmi vítanou funkcionalitou je tvorba vlastních playlistů a přidávání nahrávek do playlistů. Uživatel si tak může vytvořit vlastní hudební mix. Další stránkou tedy bude stránka playlist.php. Struktura této stránky bude téměř identická se stránkou album.php, neboť zde opět autor chce mřížkové uspořádání playlistů.

Logickou funkcionalitou je zde tlačítko pro přidání nového playlistu a jeho pojmenování. Playlist bude možné následně smazat. Po kliknutí na playlist se zobrazí seznam skladeb v daném playlistu, opět to bude mít stejnou podobu jako seznam skladeb po rozkliknutí alba.

Stejně jako u předchozích stránek, i zde bude vytvořena třída, která bude umožňovat vytvoření objektu playlist a použití jeho metod.

S tím souvisí možnost přidání nahrávek do playlistu. Každá píseň bude mít v levém rohu

ikonu možnosti, kde právě bude tato funkcionalita zveřejněna. Možnost bude mít název Add to playlist, která po rozkliknutí zobrazí seznam vytvořených playlistů. Funkce vypadá následovně:

Obrázek 56: Funkce které přidává playlisty do možnosti AddToPlaylist

```
public static function getPlaylistDropdown($con, $username) {
                              $dropDown = '<select class="item playlist">
                              <option value="">Add to playlist</option>';
                    $sql = "SELECT id, name FROM playlists WHERE owner='$username'";

                              $query = mysqli_query($con, $sql);
                              while($row=mysqli_fetch_array($query)){
                                        $id = $row['id'];
                                        $name = $row['name'];
                    $dropDown = $dropDown. "<option value='$id'>$name</option>";
                              }
                              return $dropDown."</select>"; }
```

Zdroj: vlastní zpracování

8.5.7 Uživatel

Další důležitou a potřebnou stránkou bude stránka s uživatelskými informacemi a možnosti jejich změny. Pro uživatelské účely je postačující možnost změnit heslo a e-mail. Stejně jako u všech předchozích stránek, PHP kód zjistí pomocí proměnné $userLoggedIn, kdo je přihlášen a podle toho se zobrazí uživatelovo jméno a příjmení. Na této stránce bude kromě změny některých údajů také funkce odhlásit se. User details budou navrženy formou formuláře, stejně jako přihlašovací stránka.

Změnu přihlašovacích údajů je nutné opět provádět asynchronně, tím pádem bude využit stejný koncept jako například u možnostech nahrávky, tedy Ajax. JavaScriptová funkce bude volat PHP kód, který změnu údajů vykoná. Jelikož se jedná o formulář, bude nutná kontrola zadaného vstupu, stejně jako u přihlašovacího a registračního formuláře. Uživatelská budou v databázi nahrazena pomocí dotazů.

Obrázek 57: Funkce odhlášení

```
<?php
session_start();
session_destroy();
?>
```

Zdroj: vlastní zpracování

Uživatelská práva budou řešena přidáním atributu „rightsId“ do entity Users a vytvoření entity Rights, kde budou názvy těchto práv, jedná se o pouhý číselník. Práva, jak autor zmiňuje na začátku, budou jen dvě – uživatelská a administrátorská. Pro id==0 se bude jednat o klasického uživatele a toho id mu bude přiděleno automaticky při registraci, ale

samozřejmě bude možné přidělit práva administrátora později. Pro id==1 se bude tedy jednat o administrátora, který bude mít jiné pravomoci. V systému budou před funkcionalitami, ke kterým by měly mít přístup pouze administrátoři podmínky, které funkcionality zobrazí jen administrátorům. Například viditelnost tlačítka pro vymazání skladby z databáze.

Obrázek 58: Ověření, že se jedná o administrátora

```
<?php
          if($userLoggedIn->getRights()==1) {
?>
<div class="item" onclick="removeSong(this,'<?php echo $albumId;?>')">Delete song</div>
          <?php } ?>
```

Zdroj: vlastní zpracování

8.5.8 Vyhledávání

Po finalizaci přehrávače a všech funkcionalit, jako jsou dynamické přepínání stránek následuje stránka search.php. Uživatel pravděpodobně nemá zájem o manuální vyhledávání nahrávek či alb, které by při vyšším množství dat mohlo být velmi časově náročné. Z tohoto důvodu bude systém nabízet funkcionalitu vyhledávání. Vyhledávání bude rozděleno do tří částí – dle interpreta, alba a nahrávky. Uživatel bude moci vyhledávat pomocí klasického textového pole. Avšak toto pole musí být určitým způsobem upraveno, protože výchozí nastavení funguje tak, že když uživatel přestane psát, tak se kurzor ocitne na začátku textového pole. Tímto způsobem by uživatel musel pokaždé manuálně přeskakovat na konec pole, což není žádoucí.

Další funkcionalitou je, že samotné vyhledávání začne až po vteřině, co uživatel přestane psát. Je to z toho důvodu, že jinak by vyhledávání probíhalo neustále, což postrádá význam. Dále je potřebné ošetřit situaci, kdy uživatel nezadá vůbec nic, ve výchozím nastavení by totiž systém vyhledal všechno. Vyřešeno je to jednoduchou podmínkou, která říká že když je „term“ prázdný String, tak skript ukončí.

Obrázek 59: Skript, řešící vyhledávací pole

```
<input type="text" class = "searchInput" value="<?php echo $term?>" placeholder="Start
typing..." onfocus ="this.value = this.value">
<script>
          $(function() {
                    $(".searchInput").keyup(function() {
                              clearTimeout(timer);
                              timer = setTimeout(function (){
                                        var val = $(".searchInput").val();
                                        openPage("search.php?term="+val);},1000) }); });
</script>
```

Zdroj: vlastní zpracování

Vyhledávání je dosaženo pomocí parametru „term“ v URL stránky. Proměnná timer pak říká, po jaké době po skončení psaní má vyhledávání započít. Timer je ale nutné ukončit, jinak běží i na jiné stránce a může tedy zpětně přepnout na stránku search.php

Search stránka má tedy zobrazit výsledky pro interprety, alba a nahrávky. Je žádané, aby výsledky měli stejnou podobu a strukturu jako na předchozích stránkách. Neboli aby výsledky vyhledávání pro alba vypadali stejně jako na stránce browse.php. To samé je žádoucí u seznamu vyhledaných skladeb, kde bude podoba stejná jako na stránce artist.php. Seznam vyhledaných interpretů bude jako jediný nový, a bude to klasický seznam. Velká výhoda je, že většina těchto požadavků již byla naprogramována na předchozích stránkách, stačí tedy pouze kód okopírovat a lehce poupravit.

8.6 Hudební přehrávač

Patrně nejkomplikovanější částí celého systému je samotný hudební přehrávač. Při jeho implementaci musel být využit JavaScript. JavaScript byl využit proto, že PHP se načítá ihned po načtení stránky, v tomto systému je ale žádoucí, aby změny probíhaly bez nuceného znovunačtení.

Dále bude využita technika JSON, nebo JavaScript Object Notation, což je technika na ukládání dat, aby tomu rozuměl každý jazyk. Konkrétně, v tomto systému bude potřeba využít PHP pole v JavaScriptu, což by bez převedení JSonem nebylo možné. JSon je zabudovaná (built-in) funkce, není tedy třeba nic importovat.

Využito bude muset být více nástrojů, dalším takovým je JQuery. Jedná se o JavaScriptovou knihovnu, která nabízí různé užitečné funkce. Tyto funkce lze naprogramovat v JavaScriptu, ale je to velmi časově náročné, JQuery je o mnoho rychlejší a na mnohem méně řádek kódu. JQuery umožňuje Ajax volání, což jsou taková volání, která vykonají PHP kód bez nutnosti znovunačtení stránky. JQuery bylo již využito v kapitole Registrace a přihlašování, kdy bylo potřeba dynamicky měnit typ formuláře podle toho, co si uživatel vybral a bez znovunačtení stránky.

První krok v programování přehrávače bude střídání ikon play a pause, bez znovunačtení stránky. Postup je stejný jako u registračního a přihlašovacího formuláře.

Obrázek 60: Dynamická změna ikon pause a play

```
function playSong() {
          if(audioElement.audio.currentTime == 0) {
          $.post("includes/handlers/ajax/updatePlays.php", { songId:
audioElement.currentlyPlaying.id });
          }
          $(".controlButton.play").hide();
          $(".controlButton.pause").show();
          audioElement.play();
}

function pauseSong() {
          $(".controlButton.play").show();
          $(".controlButton.pause").hide();
          audioElement.pause();
}
```

Zdroj: vlastní zpracování

Následujícím krokem bylo nastavení právě hrající nahrávky.

Obrázek 61: Funkce setTrack

```
function setTrack(trackId, newPlaylist, play) {
          if(newPlaylist != currentPlaylist) {
                    currentPlaylist = newPlaylist;
                    shufflePlaylist = currentPlaylist.slice();
                    shuffleArray(shufflePlaylist);
          }
          if(shuffle == true) {
                    currentIndex = shufflePlaylist.indexOf(trackId);
          }
          else {
                    currentIndex = currentPlaylist.indexOf(trackId);
          }
          pauseSong();

          $.post("includes/handlers/ajax/getSongJson.php", { songId: trackId },
function(data) {
                    var track = JSON.parse(data);
                    $(".trackName span").text(track.title);
                    $.post("includes/handlers/ajax/getArtistJson.php", { artistId:
track.artist }, function(data) {
                              var artist = JSON.parse(data);
                              $(".trackInfo .artistName span").text(artist.name);
                              $(".trackInfo .artistName
span").attr("onclick","openPage('artist.php?id="+artist.id+"')");
                    });
                    $.post("includes/handlers/ajax/getAlbumJson.php", { albumId:
track.album }, function(data) {
                              var album = JSON.parse(data);
                              $(".content .albumLink img").attr("src", album.artworkPath);
                              $(".content .albumLink img").attr("onclick",
"openPage('album.php?id="+album.id+"')" );
                              $(".trackInfo .trackName span").attr("onclick",
"openPage('album.php?id="+album.id+"')" );
                    });

                    audioElement.setTrack(track);
                    if(play == true) {
                    playSong();
          }
          });
}
```

Zdroj: vlastní zpracování

Ajax volání jsou zde využita, aby autor mohl vykonávat PHP kód skrze JavaScript, bez znovunačtení stránky. Příkaz $.post() v kódu říká, jaký PHP skript má funkce volat. Tento PHP skript je uložen v separátním souboru a obsahuje jednu krátkou funkci.

Obrázek 62: Skript, který vrací aktuálně zvolenou nahrávku

```
<?php
include("../../config.inc");
if(isset($_POST['songId'])) {
          $songId = $_POST['songId'];
          $query = mysqli_query($con, "SELECT * FROM songs WHERE id='$songId'");
          $resultArray = mysqli_fetch_array($query);
          echo json_encode($resultArray);
}
?>
```

Zdroj: vlastní zpracování

Stejně jsou získány ostatní informace o nahrávce, opět pomocí Ajax volání a převedení dat skrze JSON. Skrze Ajax volání bude také realizován počet přehrání.

Když aktuálně hrající nahrávka skončí, automaticky se přesune na další v aktuálním playlistu. V této části bude poprvé využito takzvaného addEventListener. AddEventListener, jak již název vypovídá, je funkce, která „přidá posluchače akce“.

Obrázek 63: AddEventListener pro automatické přehrání další nahrávky

```
this.audio.addEventListener("ended", function() {
                    nextSong();
          });
```

Zdroj: vlastní zpracování

EventListener říká, že když nastane akce „ended“, tedy když nahrávka skončí, zavolá funkci nextSong, která přejde na další nahrávku.

8.6.1 Progress bar

Obrázek 64: AddEventListener pro zjištění délky trvání

```
this.audio.addEventListener("canplay", function() {
                    var duration = formatTime(this.duration);
                    $(".progressTime.remaining").text(duration);});
```

Zdroj: vlastní zpracování

Tento EventListener říká, že když započne akce „canplay“, na objektu „audio“ vykoná kód který následuje po function(). Vytvoří JQuery objekt, který má jako parametr třídu progressBaru. Doba trvání z funkce duration je ale třeba formátovat, neboť výchozí jednotkou jsou sekundy, pomocí jednoduché funkce je čas formátován na minuty a sekundy.

Následně je žádoucí, aby se ProgressBar vyplňoval podle toho, v jaké části se přehrávání nachází. Opět bude využito AddEventListener a následně JavaScriptové funkce.

Obrázek 65: Funkce která dynamicky updatuje průběh nahrávky

```
function updateTimeProgressBar(audio) {
          $(".progressTime.current").text(formatTime(audio.currentTime));
          $(".progressTime.remaining").text(formatTime(audio.duration -
audio.currentTime));

          var progress = audio.currentTime / audio.duration * 100;
          $(".playbackBar .progress").css("width", progress + "%");
}
```

Zdroj: vlastní zpracování

Po zprovoznění této funkcionality je potřebné ProgressBar nastavit tak, aby bylo po kliknutí či přetažení myši možné nahrávky posunout do uživatelem požadované polohy. Této funkcionality bude dosaženo s JQuery.

Obrázek 66: JQuery umožňující měnit pozici v nahrávce

```
$(".playbackBar .progressBar").mousedown(function() {
                    mouseDown = true;
          });

$(".playbackBar .progressBar").mousemove(function(e) {
                    if(mouseDown == true) {
                              timeFromOffset(e, this);
                    }
          });

$(".playbackBar .progressBar").mouseup(function(e) {
                    timeFromOffset(e, this);
          });
```

Zdroj: vlastní zpracování

Funkce timeFromOffset zjišťuje, jak daleko se myš v progressBaru nachází, a na základě toho zjistí požadovaný čas.

Obrázek 67: TimeFromOffset nastaví nahrávku na požadovaný čas

```
function timeFromOffset(mouse, progressBar) {
          var percentage = mouse.offsetX / $(progressBar).width() * 100;
          var seconds = audioElement.audio.duration * (percentage / 100);
          audioElement.setTime(seconds);
}
```

Zdroj: vlastní zpracování

Analogicky bude napsána funkcionalita VolumeBar, která umožní uživateli měnit volume kliknutím či přetažením myši.

8.6.2 Tlačítka hudebního přehrávače

Hudební přehrávač má několik tlačítek – ikon, kterým je potřeba dodat vlastní funkcionalitu. Tlačítka play a pause již byla zprovozněna na začátku této kapitoly, zbývají tedy shuffle, next, previous a repeat.

Prvním tlačítkem bude next – další nahrávka. Systém si bude držet index aktuálně přehrávané nahrávky a ve funkci setTrack se tento index předá do aktuálního playlistu a ke tlačítku next bude přiřazena funkce nextSong.

Obrázek 68: Funkce pro tlačítko next

```
function nextSong() {
          if(repeat == true) {
                    audioElement.setTime(0);
                    playSong();
                    return;
          }
          if(currentIndex == currentPlaylist.length - 1) {
                    currentIndex = 0;
          }
          else {
                    currentIndex++;
          }

          var trackToPlay = shuffle ? shufflePlaylist[currentIndex] :
currentPlaylist[currentIndex];
          setTrack(trackToPlay, currentPlaylist, true);
}
```

Zdroj: vlastní zpracování

Následuje tlačítko repeat – opakovat jednu nahrávku. Jako u play a pause, i zde se budou střídat dvě ikony – repeat a repeat-active. Ve výše znázorněné funkci je zajištěna funkcionalita tlačítka, když je repeat aktivní, čas aktuálně přehrávané skladby se nastaví na nulu a nahrávka začne hrát odznova. Dolní funkce znázorňuje změnu ikon.

Obrázek 69: Funkce pro tlačítko repeat

```
function setRepeat() {
          repeat = !repeat;
          var imageName = repeat ? "repeat-active.png" : "repeat.png";
          $(".controlButton.repeat img").attr("src", "assets/images/icons/" + imageName);
}
```

Zdroj: vlastní zpracování

Tlačítko previous – předchozí song, bude implementováno velmi podobně jako next. Klíčem je posunout index o -1, a také ošetřit aby index nebyl 0, v poli se totiž žádná pozice -1 nevyskytuje.

Obrázek 70: Funkce pro tlačítko previous

```
function prevSong() {
          if(audioElement.audio.currentTime >= 3 || currentIndex == 0) {
                    audioElement.setTime(0);
          }
          else {
                    currentIndex = currentIndex - 1;
                    setTrack(currentPlaylist[currentIndex], currentPlaylist, true);
          }
}
```

Zdroj: vlastní zpracování

Následuje tlačítko mute – ztlumit, opět bude implementováno velmi podobně jako tlačítko repeat.

Obrázek 71: Funkce pro tlačítko mute

```
function setMute() {
          audioElement.audio.muted = !audioElement.audio.muted;
          var imageName = audioElement.audio.muted ? "volume-mute.png" : "volume.png";
          $(".controlButton.volume img").attr("src", "assets/images/icons/" + imageName);
}
```

Zdroj: vlastní zpracování

Lehce obtížnějším bude tlačítko shuffle – zamíchat. Nejprve je nutné si uvědomit, že k funkci zamíchat bude potřebná proměnná druhého, zamíchaného playlistu. Playlisty se nesmí přepisovat, neboť ten původní potřebujeme mít k dispozici. K zamíchání aktuálního playlistu je napsána funkce, která zamíchá pole aktuálního playlistu.

Obrázek 72: Funkce pro tlačítko shuffle

```
function setShuffle() {
          shuffle = !shuffle;
          var imageName = shuffle ? "shuffle-active.png" : "shuffle.png";
          $(".controlButton.shuffle img").attr("src", "assets/images/icons/" + imageName);
          if(shuffle == true) {
                    shuffleArray(shufflePlaylist);
                    currentIndex =
shufflePlaylist.indexOf(audioElement.currentlyPlaying.id);
          }
          else {
currentIndex = currentPlaylist.indexOf(audioElement.currentlyPlaying.id);
          }}
```

Zdroj: vlastní zpracování

9 Testování a návrhy na zlepšení a inovaci

9.1 Testování

Po úspěšné implementaci celého systému bylo přistoupeno k testování všech funkcionalit. Tento systém byl testován na následujících webových prohlížečích pro Windows – Opera, Google Chrome, Mozilla Firefox a Microsoft Edge. Testování probíhalo formou uživatelských scénářů.

Nejprve bylo vyzkoušeno registrování do systému, vypisování správných chybových zpráv a správné ukládání do databáze phpMyAdmin. Následně bylo provedeno přihlášení uživatele, které také proběhlo v pořádku. Systém si správně uchovával uživatelské jméno přihlášeného uživatele. Následovalo otestování obecných funkcionalit systému, prvním v pořadí bylo přesměrovávání na žádané stránky po kliknutí na určitý text, tlačítko či obrázek. Nejprve byly otestovány jednotlivé stránky – Search, browse, your music, user details, upload music a uploaded music, a jejich správné přesměrovávání. Následně bylo učiněno to samé, aby když kdekoliv v systému uživatel klikne na název alba, obal alba nebo název interpreta, byl přesměrován na požadovanou stránku s požadovanými informacemi. Následovalo otestování tlačítek Create new playlist, user details a log out.

Pod tlačítkem user details se skrývá formulář pro změnu emailu a hesla uživatele, bylo tedy uskutečněno změnění těchto údajů, které úspěšně proběhlo a data se v databázi aktualizovala.

Funkce vyhledávání byla otestována pro různé znaky, slova a slovní spojení, aby bylo dosaženo požadovaných výsledků v oblastech alb, interpretů a skladeb. Vyhledávání proběhlo správně.

Další důležitou funkcionalitou je tvorba vlastních playlistů s vlastním názvem, přidávání nahrávek do vlastních playlistů, odebírání nahrávek z playlistů a případné mazání playlistů. Přidání do playlistu bylo požadováno, aby fungovalo jak ze stránky alba, ale i ze stránky interpreta, jiného playlistu a stránky vyhledávání. Všechny funkcionality byly vyzkoušeny a výsledek byl dle očekávání kladný.

Patrně nejdůležitější funkcionalitou je pak samotné přehrávání nahrávek, kdekoliv v systému, to znamená na stránce alba, v playlistu, na stránce interpreta, ve vyhledávání a pro administrátora i v nahraných souborech. Přehrávání fungovalo na všech těchto

stránkách. Otestována musela být ale také všechny tlačítka hudebního přehrávače a progressBar s volumeBar. Uživatel vyzkoušel nahrávku pozastavit a následně přehrát, přeskočit na další nahrávku, a naopak posunout se o nahrávku zpět, zapnout a vypnout tlačítko opakování a zapnout a vypnout tlačítko zamíchání. Následně testoval funkci posunutí se do určitého času v nahrávce, pomocí progress baru, tato funkcionalita byla vyzkoušena jak kliknutím na určitou pozici, tak přetažením myši na určitou pozici. Analogicky stejně bylo vyzkoušeno ovládání hlasitosti a také ztlumení.

Přidanou hodnotou tohoto systému je umožnění nahrávání vlastních nahrávek přímo v systému. Uživatel nejprve vyzkoušel funkčnost formuláře, tedy jestli mu formulář povolí zadat nepovolené znaky či vynechat některé povinné položky. Následně zkusil nahrát album čítající pět skladeb od nového interpreta. Album se správně nahrálo do cloudové služby a odkazy na skladby a obal byly uloženy do databáze v PhpMyAdmin.

Po nahrání vyzkoušel uživatel s administrátorskými právy tyto nahrané skladby schválit na stránce Uploaded music, schvalování proběhlo úspěšně.

Po úspěšném otestování celého systému a všech jeho funkcionalit může autor říci, že všechny požadované funkcionality fungují tak jak bylo zamýšleno.

9.2 Porovnání s podobnými systémy

Hudebních platforem existuje velmi vysoké množství, ale autor se zaměřuje jen na některé nejvyužívanější, které jsou dle autorova názoru Spotify, Apple music, Google music, Soundcloud a Deezer. Z těchto pěti platforem se autorovi nejvíce zamlouvá právě Spotify. Avšak právě Spotify postrádá klíčovou funkcionalitu nahrání vlastních skladeb přímo v platformě. Tento proces musí být proveden přes třetí strany, a to je zdlouhavější řešení, které autor nepovažuje za nejlepší. Naopak Soundcloud umožňuje nahrání přímo v platformě, a dokonce ani nekontroluje co bylo nahráno a nahrávka je ihned zveřejněna. Soundcloud ale bohužel spíše funguje jako přehrávač jednotlivých skladeb než jako celku. Apple music, Google music a Deezer jsou srovnatelné se Spotify v rámci funkcionalit.

Naopak všechny tyto platformy obsahují množství funkcionalit navíc, například zobrazení textu nahrávky, lepší vyhledávání, možnost sledovat co poslouchají uživatelovi přátelé, zobrazují relevantní obsah pro každého uživatele či umožňují filtrovat podle různých kritérií. Autorův záměr je se těmto funkcionalitám postupně přibližovat.

9.3 Návrhy na zlepšení

Návrhů na zlepšení je opravdu velké množství, neboť vždycky je co zlepšovat. Zlepšení se týká všech stránek – aplikační, prezentační a databázové.

Z hlediska databázové stránky se nabízí jedno zlepšení, a to způsob nahrávání dat – nahrávek a obrázků. Nahrávání v tomto systému funguje tak, že se nahrávky ukládají na lokální počítač a do MySQL databáze v phpMyAdmin jsou nahrány pouze odkazy na cestu k daným souborům. V tomto případě by se jako vhodné nabízelo řešit ukládání souborů na cloud.

Některé inovace vycházejí z inspirace z již existujících systémů, zejména možnost filtrování nahrávek podle žánrů je velmi žádaná funkcionalita. Uživatelé pravděpodobně chtějí nahrávky, které jsou nejbližší jejich hudebnímu vkusu, a toho může být docíleno právě takto. Filtrování může být zavedeno i podle země, ze které interpret pochází, nebo například jaké nástroje interpret používá. To by vyžadovalo přidání několika atributů.

Obecně by jako zlepšení mohlo být bráno zavedení personalizovaného obsahu, tedy že by systém, po souhlasu uživatele sledoval, jaké nahrávky a interprety uživatel nejčastěji poslouchá a vyhledává, a na základě těchto statistik by mu zobrazoval pro něj relevantní obsah.

Dále by uživateli bylo umožněno skladby hodnotit, buď pro sebe, nebo viditelně. Skladbám by mohl dávat „to se mi líbí“ a automaticky by takové skladby byly přidávány do defaultního playlistu, zkrátilo by to tedy čas vyhledání či vytvoření správného playlistu.

Inovace se ale netýkají pouze uživatele, ale i administrátora, který samozřejmě může veškeré informace upravovat v databázi, v prostředí PhpMyAdmin, ale stále se jedná o manuální činnost, kterou se autor snažil co nejvíce v systému eliminovat. Zatím může administrátor nahrávky schvalovat či mazat v prostředí webového informačního systému. Žádoucí by ale bylo, aby to samé mohl vykonávat například s uživateli, nebo uživatelům přidělovat práva přímo ze systému. Editace veškerých dat by měla být přístupná přímo v systému, administrátor by tak rovnou mohl přejmenovávat, přesouvat, měnit interprety, přiřazovat žánry a mnoho dalších úprav, které jsou doteď dostupné jen manuálně.

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat webový informační systém pro sdílení a přehrávání hudebních nahrávek. Systém byl vytvořen ve vývojovém prostředí SublimeText a webovém klientu PhpMyAdmin. V rámci implementace bylo použito několik programovacích jazyků – HTML, CSS, SQL, PHP a JavaScript a jeho knihovny a funkce. Základní struktura a design stránky byl napsán v jazyce HTML a CSS. Databázová část byla navržena a implementována v klientu PhpMyAdmin a její následný obsah byl upravován pomocí příkazů SQL. Tyto příkazy byly vykonávány v PHP kódu. PHP kód zajišťoval komunikaci mezi prezentační a databázovou vrstvou a zajišťoval základní funkcionality. Funkcionality, které nejsou v PHP možné, neboť PHP skripty se spouští při načtení stránka, byly vykonány v JavaScriptu. V rámci JavaScriptu byla využita knihovna JQuery, jejíž funkce velmi usnadňují a zkracují kód. Byl využit koncept Ajax, kdy jsou prováděny asynchronní změny. Pro určité funkce k manipulaci s audio soubory bylo využito API s funkcemi AddEventListener.

V počátku byl teoreticky popsán webový informační systém a jeho tři vrstvy – databázová, prezentační a aplikační. Před samotným návrhem a implementací systému byly definovány požadované funkcionality pro přihlášené uživatele a administrátory. Systém umožňuje přehrávat, nahrávat, vyhledávat a přidávat nahrávky do playlistů pro přihlášené uživatele. Pro přihlášené administrátory navíc umožňuje schvalovat nahrané skladby a také je mazat. Všichni mají možnost změny emailu a hesla.

Po navržení funkcionalit bylo již možné přistoupit k jednotlivým vrstvám systému, první byla zpracována databázová vrstva, kde proběhl návrh ERA diagramu, tedy entit, jejich atributů, vazbami mezi nimi a integritními omezeními. Následně byla celá databáze vytvořena pomocí SQL kódu v PhpMyAdmin a stejným způsobem naplněna testovacími daty.

Po databázové vrstvě autor přistoupil k vrstvě prezentační, ve které byl navrhnut a implementovat design a struktura celého systému. Vrstva byla implementována v HTML a CSS. V HTML byla navržena struktura a stylizována byla v CSS.

Aplikační vrstva byla klíčová pro komunikaci mezi dvěma předchozími vrstvami, realizována byla zejména v PHP, JavaScriptu a jeho knihovnami a koncepty. Umožnila dynamické chování celého systému, tedy nebylo potřeba znovunačtení stránky.

V aplikační vrstvě byl implementován i hudební přehrávač a jeho funkcionality, což autor považuje za patrně nejkomplikovanější část celého systému, neboť museli být použity JQuery, JSON, Ajax a JavaScript.

V PHP byly realizovány SQL dotazy, které komunikovaly s databází a přenášeli data do systému. Hlavní přidanou hodnotou tohoto systému byla možnost nahrání vlastních nahrávek přímo v systému, čímž se tento systém odlišuje od jiných, kde je tato funkcionalita poskytována pouze třetími stranami. Nahrávání funguje v souladu s definovaným cílem, tedy že po nahrání nahrávek uživatelem jsou následně schvalovány administrátorem. Další funkcionalitou, která vychází z cílů práce je tvorba a úprava vlastních playlistů. Uživatel si může vytvořit vlastní playlist s vlastním pojmenováním a následně do něj přidávat a z něj odebírat libovolné nahrávky a playlist je možné kdykoliv smazat.

V závěru je systém otestován formou testovacích scénářů, porovnán s ostatními, podobnými systémy a jsou navrženy inovace.

Seznam použitých zdrojů

Advin Webové informační systémy [Online] // Advin. - https://www.advin.cz/webove-informacni-systemy.

Bubílek Michal Objektové programování pro web [Online] // Inovace VOV. - 28. 2 2019. - https://www.vovcr.cz/odz/tech/394/page00.html.

Butler Tom PHP & MySQL: Novice to Ninja [Kniha]. - Victoria, Australia : SitePoint Pty. Ltd., 2012.

Converse Tim, Park Joyce a Morgan Clark PHP5 and MySQL Bible [Kniha]. - New Jersey, USA : Wiley Publishing, Inc., 2004.

Gangur Mikuláš Návrh a implementace webového informačního systému [Kniha]. - Plzeň, Česko : Západočeská univerzita, 2014.

Gilmore W Jason Velká kniha PHP5 a MySQL: kompendium znalostí pro začátečníky a profesionály [Kniha]. - Brno, Česko : Zoner Press, 2005.

Janovský Dušan CSS styly - úvod [Online] // Jak psát web. - https://www.jakpsatweb.cz/css/css-uvod.html.

Lacko Ľuboslav SQL - Kapesní přehled [Kniha]. - Brno, Česko : Computer Press, a.s., 2005.

Leiss Oliver a Schmidt Jasmin PHP v praxi: pro začátečníky a mírně pokročilé [Kniha]. - Praha, Česko : Grada, 2010.

Management Mania Webová aplikace (Web Application) [Online] // Management Mania. - https://managementmania.com/cs/webova-aplikace-web-application.

Management Mania Webová aplikace (Web Application) [Online] // Management Mania. - https://managementmania.com/cs/webova-aplikace-web-application.

Pankrác Miloslav PHP a MySQL bez předchozích znalostí [Kniha]. - Brno, Česko : Computer Press, a.s., 2007.

Remetei Dominik Co je to OOP a proč se to mám učit? [Online] // Engeto. - 2016. - https://engeto.cz/blog/programovani/co-je-to-oop/.

Schafer Steven HTML, XHTML a CSS: bible [pro tvorbu WWW stránek]. [Kniha]. - Praha, Česko : Grada, 2009.

Sublime Text Sublime Text - Text Editing, Done Right [Online] // Sublime Text. - https://www.sublimetext.com.

Suehring Steve JavaScript Krok za krokem [Kniha]. - Brno, Česko : Computer Press, a.s., 2008.

The Eclipse Foundation The Eclipse Foundation - Enabling Open Innovation & Collaboration [Online] // The Eclipse Foundation. - https://www.eclipse.org.

Visual Studio Code Visual Studio Code FAQ [Online] // Visual Studio Code. - https://code.visualstudio.com/docs/supporting/FAQ.

Webové informační systémy na míru | ADVIN.cz [Online] // ADVIN. - Advin base s.r.o.. - 3. 2 2022. - https://www.advin.cz/webove-informacni-systemy.

Wi - Webový integrátor Jak uplatnit principy třívrstvé architektury v rámci web integračního projektu [Online] // Wi - Webový integrátor. - 15. Listopad 2013. - https://web-integrator.cz/webova-integrace/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu.

Wikipedie Informační systém [Online] // Wikipedie. - 2022. - https://cs.wikipedia.org/wiki/Informační_systém.

Zakas Nicholas Z Javascript pro webového vývojáře [Kniha]. - Brno, Česko : Computer Press, a.s., 2009.

Seznam použitých obrázků

Seznam příloh

Příloha A: Zdrojový kód celého systému na přiloženém CD

Abstrakt

Tlačil, J. (2022). *Návrh a implementace webového informačního systému*. (Bakalářská práce), Západočeská univerzita v Plzni, Fakulta ekonomická, Česko.

Klíčová slova: Webový informační systém, databáze, php, sql, html, css, javascript

Bakalářská práce s názvem *Návrh a implementace webového informačního systému* popisuje webový informační systém, jeho dobré vlastnosti a jeho vrstvy a následně takový systém navrhuje a implementuje. V první části je webový informační systém popsán teoreticky, včetně databázové, prezentační a aplikační vrstvy. Pozornost je věnována také programovacím jazykům, které jsou v systémech používány. Po definování pojmů následuje samotný návrh a implementace systému. V práci je popsán postup, využité programovací jazyky, techniky a koncepty. V závěru je implementovaný systém otestován, ohodnocen a jsou navrženy případné inovace.

Abstract

Tlačil, J. (2022). *Design and implementation of a web-based information system*. (Bachelor Thesis), University of West Bohemia, Faculty of Economics, Czech Republic.

Klíčová slova: Web-based information system, database, php, sql, html, css, javascript

Bachelor thesis entitled *Design and implementation of a web-based information system* describes web-based information system, its good features and its layers and then designs and implements such system. In the first part, the web-based information system is described theoretically, including the database, presentation and application layers. Attention is paid to the programming languages used in these systems. The definition of terms if followed by design and implementation of the system. The thesis describes the process, used programming languages, techniques and concepts. In the end, the implemented system is tested, evaluated and possible inovations are proposed.