FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Web pěveckého sboru Gaudium Praha
Student:	Veronika Vlková
Vedoucí:	Ing. Jan Horáček
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webovou aplikaci pro pěvecký sbor Gaudium Praha. Součástí aplikace bude rozhraní pro prezentaci své činnosti pro veřejnost a rozhraní pro interní potřeby sboru. Interní systém bude umožňovat sboru správu uživatelů, událostí, komunikaci se členy a snadnou editaci prezentační části aplikace.

Cíl práce:
- Analyzujte aktuální web sboru Gaudium (gaudiumpraha.org).
- Analyzujte alespoň tři weby s podobnou tématikou.
- Na základě komunikace s členy sboru analyzujte jejich nároky na web.
- Navrhněte architekturu a funkce webu tak, aby splňovala požadavky klienta.
- Aplikaci naimplementujte (frontend i backend).
- Proveďte nasazení a uživatelské testování webu.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 19. ledna 2022 v Praze.

Bakalářská práce

WEB PĚVECKÉHO SBORU GAUDIUM PRAHA

Veronika Vlková

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Horáček
11. ledna 2024

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěla bych poděkovat svému vedoucímu práce Ing. Janu Horáčkovi, který mi trpělivě odpovídal na mé dotazy a provázel mě vytvářením závěrečné práce. Dále bych chtěla poděkovat svým kamarádům Tomáši Hegerovi, Petru Adámkovi, Ing. Davidu Bernhauerovi, Ph.D., Mariusovi Solbergovi, Mon, Ghormoonovi, Honzovi a Glurakovi za přínosné rady během psaní práce a za podporu během celého studia. Chtěla bych také poděkovat především svým rodičům, na které se vždy mohu spolehnout, a v neposlední řadě mé poděkování patří i mému bratru Martinovi, širší rodině a dalším přátelům, kteří mi byli v průběhu studia oporou, případně se také zúčastnili uživatelského testování.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.
Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. ledna 2024 .....

Abstrakt

Cílem této práce je vytvoření webové aplikace pro pěvecký sbor. Zaměřila jsem se především na co největší jednoduchost používání, což ocení především starší generace. K nalezení řešení byla použita analýza několika webových stránek jiných sborů. Řešení obsahuje použití frameworku Symfony, šablonovacího jazyka Twig a knihovny Doctrine pro přístup k databázi. Vytvořené řešení poskytuje jak možnost prezentace sboru široké veřejnosti, tak přihlášení uživatelů (členů sboru). Po přihlášení je možné zobrazení neveřejných informací, včetně rozšiřujících informací k akcím viditelným pro veřejnost (např. čas srazu členů sboru). Práce také řeší jednoduché vkládání a základní úpravu příspěvků na veřejnou i neveřejnou webovou stránku.

Klíčová slova webová aplikace pro sbor, kalendář akcí, neveřejné informace, PHP, Symfony

Abstract

The aim of this thesis is to create a web application for a choir. I focused mainly on making it as easy to use as possible, which will be appreciated especially by the older generation. An analysis of several websites of other choirs was used to find a solution. The solution involves the use of the Symfony framework, the Twig templating language and the Doctrine library for database access. The developed solution provides the possibility to present the choir to the general public as well as allow users (choir members) to log in. After logging in, it is possible to display non-public information, including additional information on events visible to the public (e.g. time of the choir members' meetings). The thesis also addresses simple insertion and basic editing of posts to the public and non-public web pages.

Keywords web application for choir, calendar of events, non-public information, PHP, Symfony

Seznam zkratek

alt	atribut v HTML, označuje alternativní text pro obrázek
API	Application Programming Interface
CSS	Cascading Style Sheets
ctrl+f	klávesová zkratka používaná pro vyhledání obsahu v souboru/na stránce
DDoS	Distributed Denial of Service
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JSP	JavaServer Pages
MIDI	Musical Instrument Digital Interface
MVC	Model, View, Controller
ORM	Object-Relational Mapping
PDF	Portable Document Format
PHP	Hypertext Preprocesor (původně Personal Home Page)
W3C	World Wide Web Consortium
WYSIWYG	What You See Is What You Get

Kapitola 1
Úvod

Sborový zpěv je poměrně rozšířená zájmová činnost. Lidé zpívají profesionálně, ale i jen tak pro radost. Ovšem i při zpívání pro radost, pokud se jedná o trochu organizovanou a ustálenou skupinu, je potřeba také určitá administrativa kolem.

Takový sbor postupem času potřebuje celkem dost věcí – například sdělení sbormistra členům, záznamy o plánovaných událostech, úložiště not nebo přehled toho, kdo a kdy se může jaké akce zúčastnit. Řešení je mnoho – posílání e-mailů, různá úložiště jako např. Google disk a sdílené tabulky... Jenže do sboru nechodí jen mladí lidé, ale moc ráda si zazpívá i starší generace. A pro tu ovládání počítače může být leckdy náročné, nehledě na to, že se jim občas povede něco omylem smazat, přepsat, či provést další nemilé drobnosti. Nemluvě o tom, když ze starší generace jsou i sbormistři a přidávání obsahu na stránky pro ně může být někdy komplikované.

Jedním takovým sborem je i smíšený pěvecký sbor Gaudium Praha. Jejich aktuální řešení je značně náchylné k nechtěným chybám a problémům, které jim akorát kazí radost z toho, co dělají. Cílem této práce je tento problém vyřešit – vytvořit webovou aplikaci, která by byla především jednoduchá a intuitivní i pro starší generaci, co nejvíce automatická a zároveň by ve velké míře zamezovala vzniku nezamýšlených chyb, ať už neopatrným ovládáním, či špatným podíváním se, který řádek tabulky má člověk upravit, když je mu umožněno zasahovat do všech...

1.1 Cíle práce

Jak již bylo zmíněno výše, cílem práce je vytvoření a nasazení webové aplikace pro pěvecký sbor, kterému bude předcházet analýza aktuálního řešení webových stránek a jejich nedostatků a analýza webových stránek několika jiných pěveckých sborů. Cílem aplikace je poskytovat jak prezentační část pro veřejnost, tak neveřejnou část pro členy sboru, která zabírá většinu práce. Aplikace cílí především na jednoduchou ovladatelnost a intuitivnost i pro starší generaci. Dalším cílem je co nejvíce věcí zautomatizovat – provádět bez zásahu uživatelů (například využívat již jednou zadané informace na více místech – oznámení o koncertě jak ve veřejné části, tak v neveřejné části doplněné o další informace).

Po nasazení aplikace je dalším cílem její otestování na reálných uživatelích a případně její drobné upravení na základě jejich zpětné vazby.

1.2 Struktura práce

První kapitola seznamuje čtenáře s obsahem práce a uvádí jej do problematiky; vysvětluje, proč a čím je práce přínosná. Druhá kapitola analyzuje jak aktuální řešení sboru Gaudium Praha, tak weby dalších sborů, a čtenář zde také nalezne vymezení funkčních a nefunkčních požadavků. Třetí kapitola popisuje návrh a implementaci aplikace a pojednává i o některých problémech, které při tvoření aplikace nastaly. Čtvrtá kapitola obsahuje popis testování prováděného jak průběžně při vývoji aplikace tak na závěr s reálnými uživateli. Obsahuje i jejich zpětnou vazbu. Pátá kapitola pak hodnotí závěrečné splnění cílů a naznačuje možná rozšíření aplikace.

Kapitola 2

Analýza

Tato kapitola se zaměřuje na analýzu současných webových stránek a jejich nedostatků. Analyzuje také webové stránky dalších sborů, minimálně jejich veřejnou část.

2.1 Analýza současného řešení

Současné řešení webu ([1]) vypadá jako z období kolem přelomu tisíciletí, působí zastarale a ani pro osoby starající se o obsah není toto řešení nikterak pohodlné.

2.1.1 Vzhled

Jak jsem již zmínila výše, vizuální stránka nepůsobí příliš moderně, skoro až evokuje otázku, zda jsou webové stránky stále funkční a pravidelně aktualizované (obr. 2.1). Přestože se grafickou stránkou aplikace primárně nezabývám, pro celkový dojem a pozitivní uživatelskou zkušenost je důležitá. Struktura webových stránek je celkem jednoduchá a i celkový obsah není nijak zvlášť obsáhlý, proto jsou i ve stávající podobě celkem přehledné. Na druhou stranu např. umístění navigačního menu ve sloupci vedle hlavního obsahu stránky sice přispívá k přehlednosti, ale zároveň ubírá prostor pro zobrazení užitečných informací. Navíc toto menu není plovoucí, takže při posunu stránky se již nezobrazuje, ale vyhrazený prostor stále zabírá.

Obrázek 2.1 Hlavní stránka webu Gaudium Praha, zdroj: [1]

Další příležitostí k vylepšení je forma zobrazení stránek např. na chytrém telefonu. Ve stávajícím řešení se při prohlížení stránek na mobilním telefonu nic nezmění. Přitom, jak uvádí i webová stránka Active24 ([2]), od roku 2016 prohlíží stránky přes mobilní telefon (či tablet) více lidí než na běžném počítači (desktopu). Stránky by tedy zasloužily změnu na něco, co se případnému prohlížení v mobilu přizpůsobí. To je možné například vyvinutím proprietární aplikace pro mobilní zařízení nebo vytvořením separátní verze webu pouze pro mobilní zařízení. Já jsem zvolila možnost úpravy stránek na tzv. responzivní web.

Responzivní web

> „Za responzivní můžeme označit takové webové stránky, které jsou navrženy a realizovány s ohledem na použitelnost a přizpůsobitelnost různým rozlišením v různých zařízeních (stolní počítače, notebooky, netbooky, tablety nebo mobilní zařízení). Pojem Responzivní web design (RWD) představil již v květnu 2010 americký programátor Ethan Marcotte na blogu A LIST Apart, první implementace se objevila již v následujícím roce.“ [3]

Ethan Marcotte popisuje responzivní design jako cestu vpřed umožňující zobrazení na různých zařízeních a zdůrazňuje důležitost přizpůsobení se. („Now more than ever, we're designing work meant to be viewed along a gradient of different experiences. Responsive web design offers us a way forward, finally allowing us to “design for the ebb and flow of things.”“ [4])

2.1.2 Jazykové mutace

Přestože Gaudium není výdělečná organizace, aplikace má sloužit i k prezentaci sboru. Stránky by tedy měly zaujmout, například v souvislosti s účastí sboru na koncertech či na festivalech. Dle článku na blogu Weglot přispívá možnost lokalizace k zaujetí větší části návštěvníků webu a zároveň napomáhá udržení jejich pozornosti. Lidé oceňují možnost přečíst si obsah stránek v jejich rodném jazyce, případně alespoň v jazyce, kterému rozumějí (např. jedná-li se o cizince). [5]

Současné řešení sice zdánlivě nabízí možnost přepnutí do jiného jazyka, ale při bližším prozkoumání se nejedná o lokalizaci, ale o hypertextové odkazy na jiné stránky. Takové řešení by nevadilo, nicméně je náchylné k chybám. Několik chyb tam opravdu najdeme – např. z anglické verze se při pokusu o prohlédnutí stránky se sborovými CD dostaneme zpět na českou verzi této stránky, a i celý zbytek webových stránek se přepne zpět do češtiny, včetně bočního menu, které bylo o krok zpět již přeložené. Uživatel je tak nucen znovu a znovu přepínat stránky do svého zvoleného jazyka, což ho po pár cyklech tohoto procesu omrzí a raději přestane stránce věnovat pozornost.

2.1.3 Interní informace

Sbor potřebuje místo pro sdílení interních informací, ať už jsou to podrobnosti ke koncertu (např. čas srazu a oblečení), nebo jaké si vzít noty na další zkoušku. Toto by širší veřejnost vidět neměla. Některé informace je pak třeba chránit ze zákona, ať už se jedná o osobní informace nebo třeba noty a některé nahrávky pořízené pro potřeby sboru (tam je zákaz šíření např. z důvodu autorského zákona).

V současném řešení je uživatelská sekce chráněna jménem a heslem. Jméno i heslo je ale pro všechny stejné (sdílené), navíc se moc často nemění. Znamená to, že kdokoliv byl jednou členem sboru, má i nadále přístup do neveřejné části (dokud se heslo nezmění, ale jak je již napsáno výše, to se neděje moc často). Jak zmiňuje CISO z University of Washington, sdílení účtu mezi více uživateli vyvolává hned několik bezpečnostních rizik, např. horší možnost zjištění a prokázání, kdo z daných uživatelů vykonal danou akci nebo si zobrazil chráněná data. Navíc je zde i vyšší šance na kompromitování hesla a, jak už bylo zmíněno výše, složitější vyřazení určité osoby z možnosti přístupu k datům.

Dalším problémem zabezpečení interní sekce u současné verze stránek je fakt, že některé odkazy jsou přístupné i pro nepřihlášené uživatele. Sice se takový uživatel nedostane přes stránku s referencí na chráněná data, ale pokud takovou referenci (odkaz) nějak získal (např. mu ji někdo poslal), může získat i tato chráněná data bez znalosti přístupových údajů k nim.[6]

Důležitým prvkem v organizaci sboru je vědět, s jakými členy lze kdy počítat. Proto je dobré mít možnost tyto informace zaznamenávat a umožnit tak členům sboru přihlašování na jednotlivé akce. Některé sbory to řeší např. napsáním se na papír. To ale přináší několik nevýhod, zejména nemožnost registrace při absenci na zkoušce a nemožnost aktualizace v případě potřeby změny.

V Gaudiu řeší tuto agendu sdílenou Google tabulkou ([7]). Tím se sice vyřeší dva výše zmíněné problémy, ale nastanou problémy jiné. Za zmínku stojí zejména možnost upravovat jakýkoliv řádek tabulky, tedy nikoli jen ten s vlastním jménem (to je sice nějakým způsobem řešitelné, ale vyžaduje to např. individuální Google účet pro každého, což je opět složitější). Chyba tedy může nastat ať už záměrně (ač nepravděpodobné, tak možnost existuje) nebo omylem (obzvlášť při vyplňování na mobilu či lidmi se zhoršeným zrakem se člověk snadno strefí o řádek vedle).

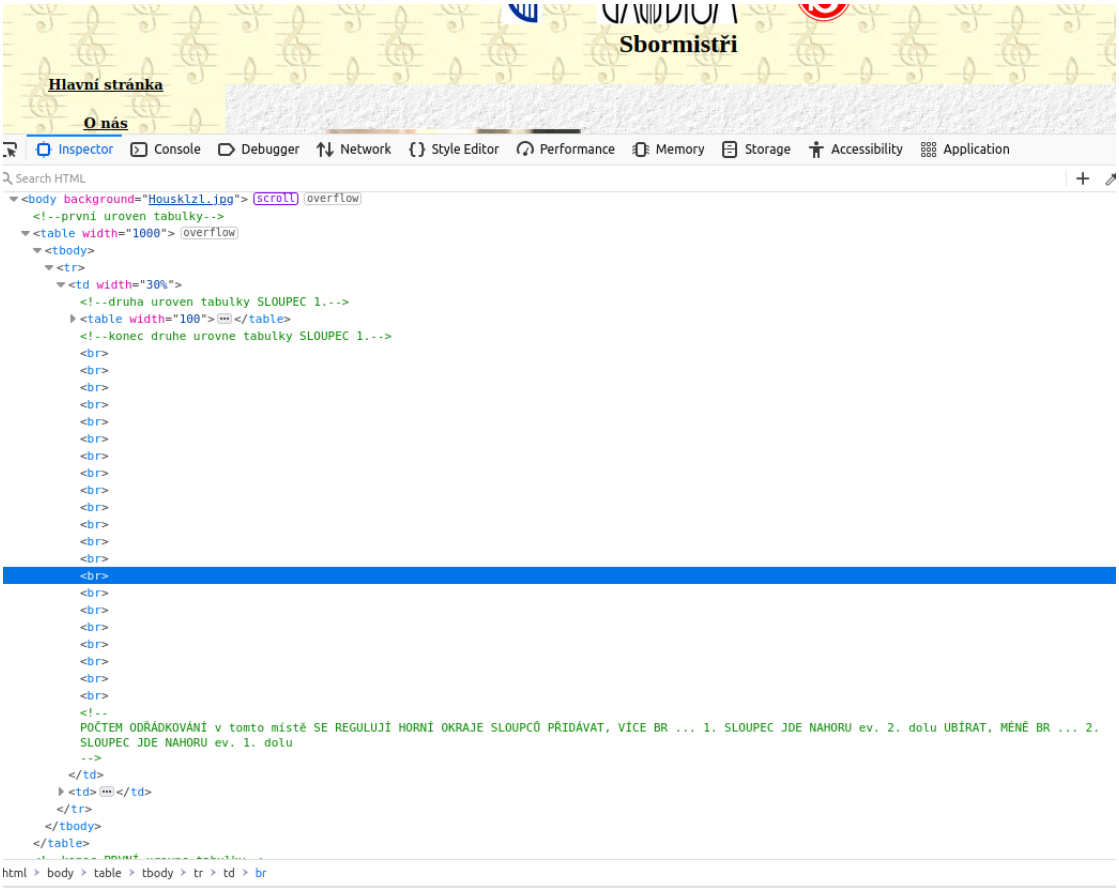
2.1.4 Přidávání a úprava obsahu webových stránek

Aby prezentační i soukromá část plnily své poslání, je potřeba umožnit snadné přidávání obsahu. Ideálně, aby se člověk nemusel zdlouhavě zdržovat formátováním a aby se mu přidáním jedné fotky nerozsypal celý zbytek. Dle rozhovoru se současnou web-editorkou to ale současné řešení neumožňuje. Obsah je zarovnáván ručně, nikoliv automaticky, takže při změně obsahu (např. přidání nové aktuality či odebrání zastaralých informací) se často stane, že je potřeba

přeformátovat půlku stránky, aby to alespoň trochu nějak vypadalo a stránka nebyla úplně rozhozená.

Při bližším zkoumání stránek si můžeme všimnout, že část formátování je jednoduše řešena novým řádkem. Dokonce najdeme i „návod“, jak s daným „formátováním“ pracovat. (obr. 2.2)

Co se týká designu dalších stránek, pro formátování byl například použit Microsoft Word, který byl pak nahrán na web (pomocí Save As možnosti). Toto řešení není příliš vhodné například proto, že když se něco na stránce náhodou pokazí, najít ve změti kódu věc, která je špatně, je téměř nemožné. Word totiž přidává spoustu vlastních tagů a informací, které nejsou ani potřebné, a soubor je pak zbytečně nepřehledný. Microsoft uvádí, že při tomto postupu může výsledek ztratit některé vlastnosti, mezi které patří například formátování znaků a odstavců, rozložení stránky či formátování grafiky a tabulek.[8]



Obrázek 2.2 Část stránky v inspektoru – formátování pomocí prázdných řádků, zdroj: [1]

2.2 Analýza webových stránek dalších sborů

V rámci teoretické části práce se zabývám také analýzou webových stránek/aplikací několika jiných pěveckých sborů, především jejich veřejnou částí. Zaměřuji se zejména na jejich charakteristiky srovnatelné s obsahem předchozí analýzy současného webu zadavatele.

2.2.1 Notre Dame

„Smíšený pěvecký sbor Notre Dame[1] vznikl v září 2004 při chrámu Matky Boží před Týnem v Praze na Staroměstském náměstí. Matka Boží (Naše Paní) byla také inspirací pro název sboru. Sbor se zaměřuje na interpretaci liturgické hudby od renesance po současnost, se zvláštním zaměřením na vokálně instrumentální díla baroka a klasicismu. Své místo spatřuje Notre Dame především při zkrášlování liturgie v Týnském chrámu i dalších kostelích po celé republice. Příležitostně vystupuje i koncertně. Sbor od založení vede Leona Stříteská, odborná asistentka na Pedagogické fakultě Univerzity Karlovy v Praze.“ ([9])

Vzhled

Sborové stránky tohoto hudebního tělesa vypadají na první pohled hezky a působí moderním dojmem. (obr. 2.3) Co se přizpůsobení např. pro mobilní telefony týká, na první pohled se stránka menšímu displeji přizpůsobí (zúžením nepodstatných okrajů), ovšem při delším prohlížení stránek narazí uživatel na „okno v okně“. Při menší výšce obrazovky se klasicky na straně objeví posuvník, ovšem samotný obsah je zabalen do JSP[2] kontejneru, což způsobí vytvoření vlastního posuvníku pro posouvání se v daném kontejneru. V rámci toho také velmi špatně funguje funkce vyhledávání, jelikož v obsahu kontejneru sice text vyhledá, ale už na něj „nepřeskočí“, jako normálně, kdy lze pomocí šipek skákat mezi jednotlivými výskyty textu. (obr. 2.4)

Obrázek 2.3 Hlavní stránka webu Notre Dame, zdroj: [9]

Jazykové mutace

Možnost přepnutí do jiného jazyka úplně chybí.

Interní informace

Do interní sekce se členové dostávají pomocí emailu a hesla, každý svého. V případě zapomenutí hesla (či přihlašovacích údajů) se ale stránka sice tváří nápomocná, nicméně po kliknutí na odkazy se vizuálně nic nestane (proběhne jakési přesměrování, ovšem výsledek chybí). Celý interní obsah

[1] `http://www.sbornotredame.cz/` ([9])

[2] technologie umožňující vkládání Java kódu do HTML stránek; Jedná se o serverový skriptovací jazyk, který umožňuje generování dynamického obsahu na webu. Soubory JSP obsahují kombinaci HTML a Java kódu.[10]

je ale zobrazován opět jako „okno v okně“ a samotný obsah je zřejmě také přidáván ručně, jelikož např. adresář členů sboru je seřazený jen částečně – část seřazená je, část je nejspíše dopsána později, někteří členové chybí úplně. Tento problém by mohlo vyřešit například automatické generování stránky na základě informací přímo z profilu uživatele – jakmile se jednou do systému přihlásí a vyplní své údaje (které stejně vedení sboru poskytuje).

2.2.2 Pueri gaudentes

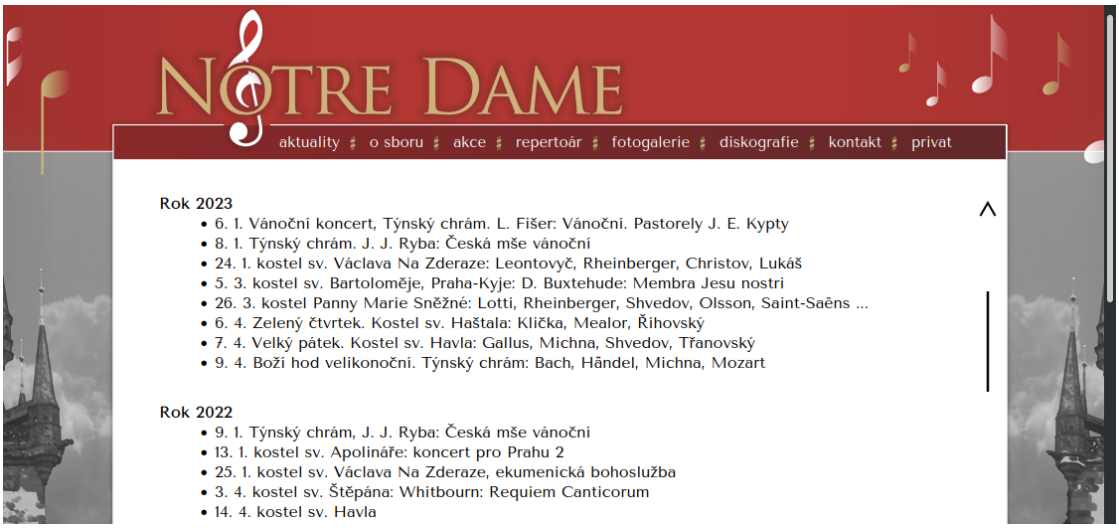
„Pueri gaudentes[3], sbor Základní umělecké školy v Praze 7, je jeden z mála českých chlapeckých sborů. Ve své historii, která sahá do roku 1990, navštívil řadu českých a evropských měst a také Japonsko. Sbor získal vysoká ocenění na prestižních festivalech v Neerpeltu, Lindenholzhausenu, Gorizii a reprezentoval na mezinárodním festivalu Pražské jaro. Pravidelně účinkuje ve Státní opeře Praha. Osmdesát mladých hlasů pravidelně láká stovky posluchačů na své koncerty v krásných historických prostředích, která dokonale podkreslují žánrovou pestrost repertoáru Pueri gaudentes.“ ([11])

Vzhled

Webové stránky tohoto sboru působí na první pohled celkem hravě (obr. 2.5). Barevné schéma je velmi pestré, styl písma navigačních prvků má připomínat dětský „škrabopis“ a i styl grafiky spíše připomíná dětskou kresbu. Při přístupu z mobilního telefonu se sice stránka zbaví přebytečných okrajů, ale jiné přizpůsobení se nekoná, což příliš prohlížení neusnadňuje. Pro dostání se k dalším informacím o sboru, jako například k jednotlivým oddělením, je třeba kliknout na „Sbor“, což způsobí objevení se dalšího menu. Každý na toto řešení může mít jiný názor, nicméně osobně to nepovažuji za příliš šťastné řešení, protože to zbytečně ztěžuje orientaci na stránce. Na druhou stranu stránka obsahuje navigaci i v zápatí, což je velmi užitečné, především když se menu neposouvá spolu s obsahem, a tedy není vždy vidět, když uživatel dojde na konec stránky.

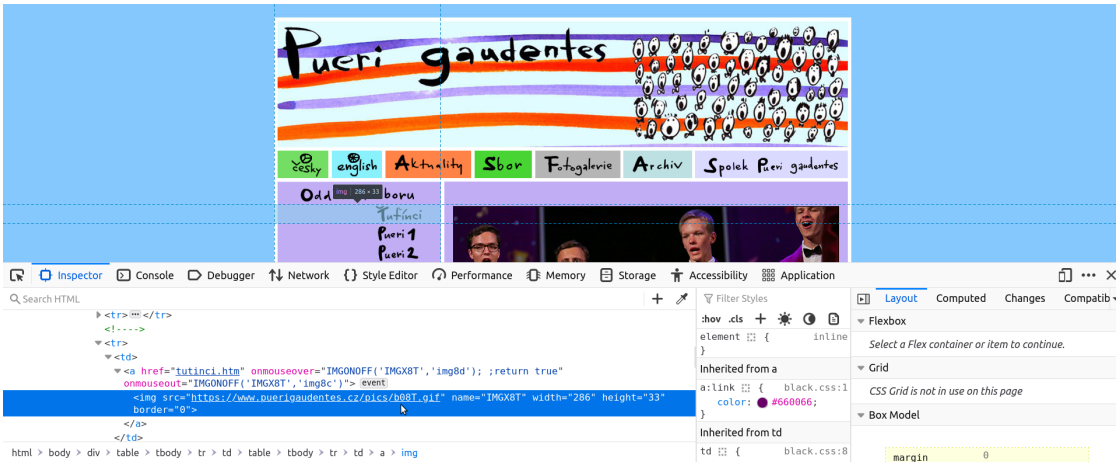
Dalším prvkem k zamyšlení je způsob, jakým je realizováno menu. Přestože sbor není povinným subjektem, který má za povinnost mít přístupné webové stránky (viz např. Zákony pro lidi, [12]), většinu zřejmě napadne, že mít web alespoň trochu přívětivý pro osoby se zhoršeným zrakem, je dobrý nápad, už jen proto, že není neobvyklé, aby se mezi rodiči nacházel např. nevidomý, který potřebuje web pro přístup k informacím, či na stránky koukají prarodiče, kteří mohou také nějaké pomůcky pro orientaci na stránce používat. Současné řešení ovšem používá místo textu v menu obrázky, bez jakéhokoliv návodného jména, chybí dokonce i popisky obrázků (HTML atribut alt) (obr. 2.6). Podle webové stránky Poslepu.cz ([13]) by přístupný web měl obsahovat ke každému netextovému obsahu textovou variantu, což zde splněno není, a ztěžuje to tak porozumění webu zrakově slabším.

[3] `https://www.puerigaudentes.cz/uvod.htm` ([11])



Obrázek 2.4 Posuvník na okraji stránky a na kraji boxu s obsahem, zdroj: [9]

Obrázek 2.5 Hlavní stránka sboru Pueri gaudentes, zdroj: [11]



Obrázek 2.6 Menu jako obrázky na stránkách Pueri, zdroj: [11]

Jazykové mutace

Co se nastavení jazyka týká, na webových stránkách sice existuje možnost přepnutí do anglického jazyka, ale opět pouze pomocí jiných odkazů. Odkazy ale nejsou úplně funkční – opět se stane, že při kliknutí na některou možnost (např. „Photo Gallery“) se stránka (menu) přepne zpět do češtiny. V anglické verzi také úplně zmizí možnost dostat se na podrobnější informace ohledně sboru (jako jsou informace ohledně přípravných oddělení či plánovaných vystoupeních). Osobně si myslím, že sekce plánovaných vystoupení je jedna z nejdůležitějších částí stránek, a měla by být tedy přístupná i v jiném jazyce.

Interní informace

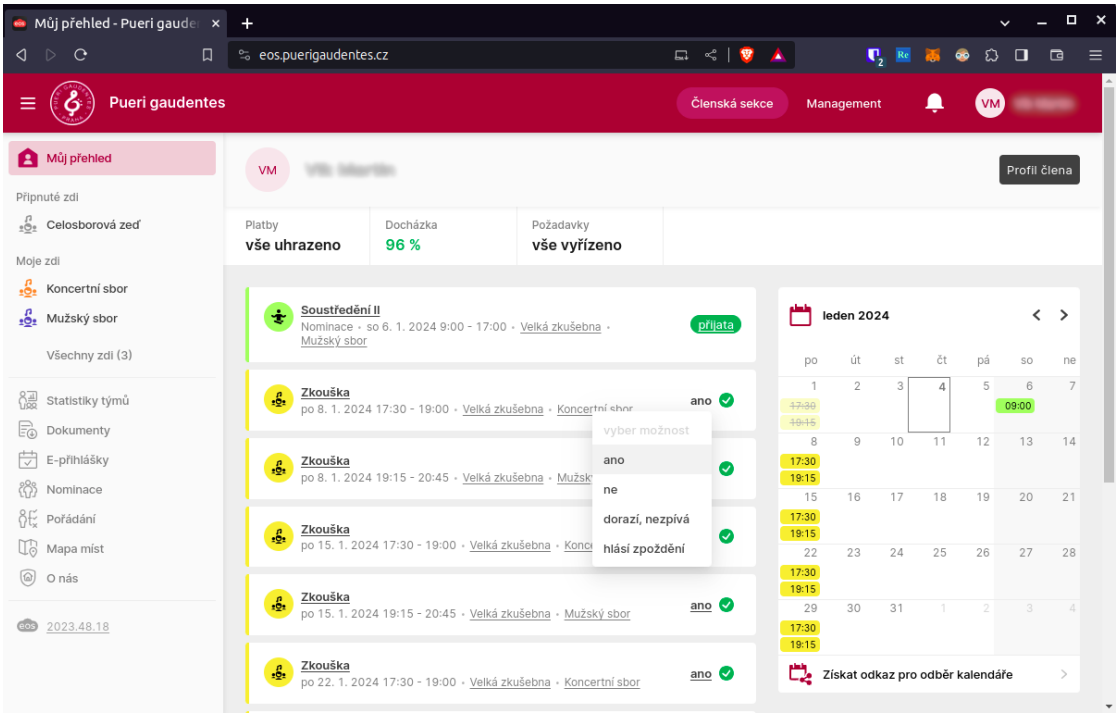
Ohledně interních informací pro členy sboru (jejich rodiče) panuje na stránkách tohoto sboru jakási lehká rozpolcenost.

Sice je stále k dispozici přístup do této sekce navigací přes položky v menu (sekce „Pro rodiče“) přístupná pomocí uživatelského jména a hesla, ale tyto informace nejsou aktualizované již více než dva roky (bez jakéhokoli zaznamenatelného upozornění - možná je to součást zabezpečení).

Na druhou stranu je členům sboru k dispozici zcela nová neveřejná sekce (dokonce tak neveřejná, že na ni z veřejných stránek nevede žádný odkaz), a ta je implementována s využitím komerční aplikace EOS ([14]). EOS je dostupný přes samotné webové stránky a je k dispozici i jako samostatná aplikace pro mobilní zařízení. EOS na první pohled vypadá hezky a působí přehledně. Vcelku žertovnou okolností je fakt, že EOS je primárně určen ke zpracování agendy sportovních klubů, čemuž odpovídá i terminologie. Tak například účast členů sboru na koncertech se registruje jako „nominace“ a sbormistři byli ještě do nedávna prezentováni jako „trenéři“ (po aktualizaci již opraveno). Navzdory některým úsměvným označením však EOS bez problémů agendu pěveckého sboru zvládne. Každý člen sboru má k dispozici vlastní sekci (nástěnku) s přehledem (kalendářem) akcí sboru a s možností doplnit, případně aktualizovat, informaci o své účasti na nich (obr. 2.7). EOS dále nabízí přehled členů sboru, evidenci jejich docházky, kalendář koncertů i pravidelných akcí (např. zkoušek), mapu s přehledem míst významných pro fungování sboru (zkušebny, koncertní sály, shromaždiště, ...), rozdělení na hlasy a mnoho dalších užitečných informací. Významným pomocníkem pro sbormistry je např. také propojení rodičů a dětí pro snadnou identifikaci členů sboru a jejich zákonných zástupců. Velmi užitečnými jsou i možnost výběru členů sboru pro účast na konkrétních akcích, správa omluvenek nebo evidence

pozdních příchodů. Nechybí ani sekce pro možnost přidávání komentářů nebo zasílání zpráv vedení sboru. Příjemným doplňkem je i možnost přidělení různých oprávnění pro použití EOSu.

Dalo by se říct, že EOS je velmi podobný systému, který je předmětem této práce (a ano, v mnohém byl pro mě inspirací) a nabízí se tak otázka, proč ho nevyužít i pro potřeby zadavatele. Odpověď je překvapivě snadná. Cena. Verze, která by licenčně pokryla potřeby sboru, by vyšla na více než 10.000 Kč ročně, což je mimo rámec možností zadavatele.



Obrázek 2.7 Nástěnka s přehledem účastí člena a s dalšími informacemi

2.2.3 Pražská kantiléna

„Pražská kantiléna[4]vznikla při ZUŠ Jižní Město v Praze 4. V současnosti sbor pracuje ve třech přípravných a dvou koncertních odděleních. Všechna sborová oddělení se věnují interpretaci skladeb široké škály hudebních stylů a žánrů. Sbor účinkuje každoročně na desítkách koncertů po celé České republice. (...) Pečlivá práce všech členů sboru od nejmenších po největší sbírá ocenění nejen v České republice, ale i v Evropě.“ ([15])

Vzhled

Prezentační část webových stránek Pražské kantilény dýchá moderním dojmem (obr. 2.8). Webové stránky jsou navrženy tak, aby byly přístupné i na mobilních telefonech. Při zobrazení na malých obrazovkách se horní menu změní na takzvané hamburgerové menu a obrázky reprezentující pěvecké sekce jsou zobrazeny pod sebou (obr. 2.9). Každý obrázek obsahuje také textový popis, což usnadňuje přístupnost a použitelnost stránek. Také text na stránkách je skutečný text, nikoliv text vytvořený jako součást obrázků. Při vývoji těchto stránek bylo pravděpodobně využito platformy WordPress. Tento redakční systém nabízí několik výhod, včetně nízkých nákladů

[4] `https://prazskakantilena.cz/` ([15])

na tvorbu a provoz stránek, velké flexibility při přizpůsobení designu a relativně jednoduchého použití i pro uživatele bez technických znalostí. ([16])

Obrázek 2.8 Zobrazení úvodní stránky Pražské kantilény na počítači, zdroj: [15]

Jazykové mutace

Jednou z věcí, které na jinak pěkných stránkách Pražské kantilény chybí, je možnost změny jazyka. Stránky jsou přístupné jen v českém jazyce, což mi pro stránky moderního sboru, účastnícího se i zahraničních vystoupení, přijde jako velká škoda.

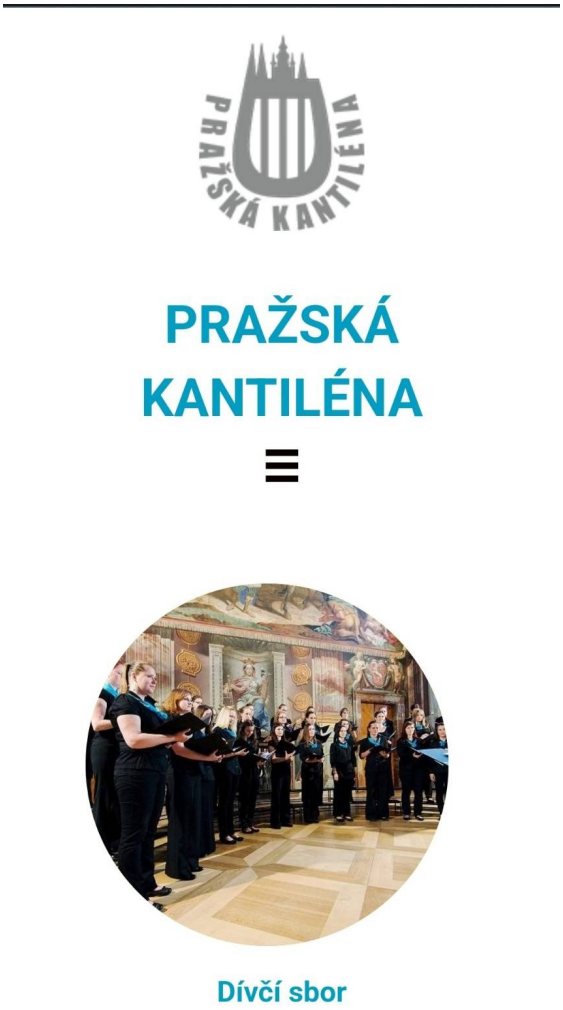
Interní informace

Interní sekce je umístěna za stránkou pro zadání hesla – interní stránka je nejspíše rozdělena na tři kategorie – dětský sbor, dívčí sbor a přípravná oddělení. Ovšem při jednoduchém pokusu o zadání slova „info“ za lomítko do url adresy jsem se dostala na možnost vyhledávání, což zní jako dobrá vlastnost při nenalezení požadované stránky, ale po zadání klíčového slova „informace“ vyhledávací query nalezne i články patřící do interní části. Po kliknutí na „Číst dále“ se tak uživatel dostane na stránku s neveřejnými informacemi, jako jsou nejen například pokyny ke koncertům, ale i třeba jmenný seznam dětí jedoucích na soustředění, či odkaz na pdf soubor not (který by, jakožto dílo s autorskými právy, neměl být sdílen veřejně). Tato zranitelnost byla sboru nahlášena.

2.3 Hodnocení analýzy

Základní dojmy

Po prohlédnutí a vyzkoušení práce s webovými stránkami několika různých pěveckých sborů, včetně současné verze stránek zadavatele, jsem získala vcelku jasnou představu o tom, jak lze prezentaci hudebních těles tohoto typu pojmout, jak ji lze technicky zpracovat, jaké informace



Obrázek 2.9 Zobrazení úvodní stránky Pražské kantilény na mobilním telefonu, zdroj: [15]

a v jakých strukturách na nich zveřejnit, a to jak anonymně (pro kohokoli), tak adresně (pro konkrétní návštěvníky stránek / členy sborů), jak se poprat s jazykovými mutacemi nebo jak zvládnout zobrazení obsahu na různých typech zařízení disponujících různými vlastnostmi (ovládacími prvky) i různými formáty (rozlišením) displeje. Dalším přínosem a zdrojem hodnotných informací pro návrh a implementaci mé vlastní webové aplikace byl i fakt, že sama jsem členem dvou těchto sborů a můj bratr je členem třetího, a tak jsem měla mnoho příležitostí osobně hovořit s dalšími členy těchto uskupení a při různých příležitostech získat i mnoho neformálních vstupů (referencí na uživatelské zkušenosti) pro mou práci.

Celkově lze říci, že žádný z analyzovaných webů není zpracovaný na zcela profesionální úrovni. Každý obsahuje nějaké chyby, nedotahuje některé detaily technické implementace a určitě by zasloužil nějaké opravy a vylepšení. Na druhou stranu žádný ze sborů, jimž analyzované weby patří, není profesionálním hudebním tělesem, a tak pracují s omezeným rozpočtem bránícím investicím do nákupu profesionální prezentace od renomovaných designových studií. Přesto všechny weby víceméně splňují účel, ke kterému byly vytvořeny a s větším či menším úsilím je lze úspěšně používat i v jejich současných podobách. To samozřejmě neznamená, že by mezi jejich uživateli panovala stoprocentní spokojenost s nimi a nebyl zde prostor pro mnohá vylepšení. Každopádně i pro mě byly mnohé designové i funkční prvky inspirací a podnětem pro tuto mou práci a pro vývoj mé vlastní webové aplikace.

Výstupy

Jednoznačným pozitivním dojmem, který jsem se snažila ve své práci zachovat, je jednoduchost a přehlednost prezentovaných informací a to zejména v interních sekcích určených členům sborů a jejich vedení. Přestože by bylo možné vytvořit spoustu různých grafických ozdob pro zpestření prezentovaného obsahu, z praktického hlediska je takový koncept spíše na závadu zejména v situacích, kdy, jak již bylo uvedeno (web Active24, [2]), většina návštěvníků webových stránek dnes používá k jejich prohlížení mobilní zařízení, tedy taková, jejichž obrazovky sice mnohdy mají rozlišení srovnatelná s monitory stolních počítačů, ale jejich fyzické rozměry jsou velmi omezené a čitelnost zobrazených informací je proto silně limitovaná. V takových případech je tak přítomnost zbytečných ozdobných prvků na stránkách jednoznačnou přítěží. Navíc některé pěvecké sbory jsou uskupeními tvořenými i členy z řad seniorů, kteří jednak často čelí významným limitům v jejich technických znalostech, schopnostech a dovednostech, a jednak se potýkají i s omezeními ve fyzické schopnosti tato zařízení ovládat (třes rukou, slabozrakost, ...). Zejména pro ně je tak jednoduchý a přehledný design absolutní prioritou.

Výše uvedené postřehy a poznatky z provedené analýzy včetně mnoha dalších jsou použity a dále diskutovány ve specifikacích funkčních a nefunkčních požadavků níže.

2.4 Funkční a nefunkční požadavky

Při analýze požadavků na budoucí aplikaci je důležité si správně stanovit cíle. Nároky na budoucí systém dělíme na požadavky funkční a nefunkční, přičemž porozumění jejich rozdílům výrazně pomáhá k úspěšnému dokončení projektu a naplnění očekávání ([17]). Velmi stručně lze hlavní rozdíl mezi nimi specifikovat tak, že „funkční požadavky popisují, co by měl systém dělat, zatímco nefunkční požadavky popisují, co systém potřebuje k tomu, aby byly funkční požadavky bezpečně, spolehlivě a efektivně splněny“ [18].

2.4.1 Funkční požadavky

Po rozhovoru s odpovědnými členy sboru a zadavateli požadavků na novou webovou aplikaci jsem identifikovala následující funkční požadavky:

Rozdělení veřejné a neveřejné části

Nejspíš nejdůležitějším požadavkem na novou webovou aplikaci je rozdělení stránek na veřejnou a neveřejnou část. Veřejná část slouží k prezentaci samotného hudebního tělesa veřejnosti. Součástí je prostor k představení sboru jako takového, jeho sbormistrů a dalších důležitých osob (klavíristka, hlasový korepetitor...) i např. nabídka vydaných CD. Důležitou součástí prezentace veřejnosti je také seznam koncertů a dalších významných událostí.

Neveřejná část naopak slouží k vystavení informací, které mají být veřejnosti skryté, avšak pro hladké fungování sboru jsou potřebné. Jedná se například o informace o čase srazu před koncertem, jaké noty si vzít na zkoušku, či jiná sdělení sbormistrů.

Jazykové mutace

Dalším funkčním požadavkem na web jsou jazykové mutace. Sbor se účastní tuzemských i zahraničních soutěží a festivalů a pro takové účely je dobré, aby si mohli i účastníci či porotci z jiných zemí něco o sboru přečíst. Překlad do anglického jazyka je pro takové účely nejvhodnějším základem, protože se v současné době jedná o statisticky nejpoužívanější „cizí“ jazyk. Požadavek však cílí i na nekomplikovanou možnost případného přidání dalších jazyků v budoucnu. Volba jazyka by měla při procházení stránek nějakou dobu vydržet, aby návštěvník stránek nestrávil půlku času jejich neustálým přepínáním do svého preferovaného jazyka.

Možnost přípravy a nasazení vlastních jazykových mutací jsem volila zejména proto, že i když v současné době fungují nástroje pro instantní automatický překlad (k dispozici např. v Google Chrome nebo v Microsoft Edge) a tato funkcionalita se neustále vylepšuje, přesto ještě není dostatečně dobrá a výsledky takového automatického překladu tak mohou být občas poněkud zavádějící.

Vzhledové sjednocení stránek

Aktuální stránky nemají (v době psaní této práce) jednotné formátování, ne u všech stránek se podařilo stejné nastavení, a tak některé stránky na pohled „odskakují“. Jeden z požadavků tedy cílí na toto. S tímto požadavkem také souvisí požadavek na možnost snadno prohlížet stránky z mobilu, jelikož v dnešní době již většina uživatelů prohlíží webové stránky ze svého mobilního telefonu.

Automatičnost a intuitivnost

Jedním z dalších funkčních požadavků je důraz na intuitivnost a co nejvíce automatických procesů. Velkou část členů sboru tvoří starší generace, pro kterou je často práce s počítačem a změna systému velký boj. Část práce také ušetří některé automatické činnosti, jako je například propisování jedné informace na více míst (místo nutnosti danou informaci psát znova – např. pro veřejnost je potřeba uvést čas začátku akce do sekce koncertů, a členové sboru tuto informaci také potřebují, avšak do sekce koncertů obvykle nekoukají, protože potřebují především informaci o čase srazu, takže je pro ně přívětivější mít obě informace na stejném místě).

Přihlašování na akce

Důležitým funkčním požadavkem je také možnost zaznamenávání plánované účasti na nadcházejících akcích. Pro úspěšné zvládnutí koncertního vystoupení je nutné mít dostatek zpěváků. Ale jelikož ne každý má vždy čas, pro sbormistra je klíčové mít přehled o tom, kdo se kdy zúčastní, ideálně i z jakého je hlasu. Vznikl tak požadavek na snadné zaznamenávání (ne)účasti na akcích tak, aby každý člen sboru mohl vyplnit informaci jen za sebe a omylem nezměnil údaje o někom jiném.

Více práv určité skupině členů

Přestože aplikace cílí na snadnou použitelnost a dostupnost pro všechny, i tak se může stát, že někdo není z různých důvodů schopný údaje samostatně vyplnit, a je třeba, aby je za něj vyplnil někdo jiný. Tím vznikl další požadavek, a to pro vybrané členy nastavit rozšířené oprávnění, které např. kromě možnosti vyplňování informací za jiné členy umožní i přidávání obsahu do neveřejné části, jako např. události, na které je třeba se přihlásit, či noty skladeb.

Adresář s kontakty

K výše zmíněnému požadavku se částečně vztahuje další funkční požadavek, a to prostor na uložení kontaktních informací členů. Jedná se o osobní údaje, a proto by je mělo vidět co nejméně lidí. Požadavkem je tedy skrytí kontaktních údajů běžným členům, avšak snadná dostupnost pro pověřené členy.

Informace pro členy

Speciálním požadavkem byl prostor pro sdělování informací od sbormistra členům sboru. Tento koncept funguje v současném řešení a požadavkem je toto zachovat. Sbormistr (či další oprávněné osoby) by měl mít možnost publikovat sdělení s libovolným obsahem (například odkaz na video z koncertu) ostatním členům, tedy zabezpečené před očima veřejnosti.

2.4.2 Nefunkční požadavky

Dle Pavla Gorbachenka z Enkonix ([17]) nefunkční požadavky se zaměřují na vlastnosti systému. Patří mezi ně například požadavky na:

- rychlost,
- dostupnost (kdy je systém dostupný – například jestli celý den, nebo jen v pracovní dobu),
- kapacitu (jaké jsou limity a co je systém schopný zvládnout),
- spolehlivost,
- technologii,
- vzhled (GUI),
- uživatelskou přívětivost (jak je pro koncového uživatele složité systém používat).

Vzhledem k tomu, že zadavatelé požadavků mají opravdu malé zkušenosti se systémy a s požadavky na jejich funkci, praktických nefunkčních požadavků jsem od zadavatelů získala opravdu jen velmi málo. Jejich identifikace, specifikace a definice tak byly ponechány na mně. Já jsem se nakonec rozhodla takto:

Základním východiskem pro stanovení nefunkčních požadavků pro výslednou aplikaci byl fakt, že se jedná o prezentaci relativně malého zájmového sdružení, které, s ohledem na počet svých členů a potenciálních příznivců a zájemců o informace o sboru, neočekává návštěvnost výsledných webových stránek nijak masivní. Není důvod se domnívat, že by výsledná webová prezentace

sboru čelila náporu tisíců návštěvníků za hodinu nebo vyšší. Ani v případě účasti sboru na akcích typu hudební festival, kdy o získání informací o vystupujících sborech mají zájem ostatní účastníci akce nebo její návštěvníci, není pravděpodobná enormní návštěvnost stránek iniciovaná reálnými uživateli Internetu. Zároveň je předmět zájmu sdružení velmi vzdálen finančnímu sektoru nebo jinému komerčnímu či mocenskému odvětví vzbuzující zájem hackerů, a proto i pravděpodobnost DDoS útoků je vcelku zanedbatelná. Přehnané nároky na výkon, propustnost, dostupnost či zabezpečení tedy v tomto případě nejsou na místě.

V důsledku předchozích analýz a navazujících úvah jsem se proto rozhodla stanovit výchozí limity pro nefunkční požadavky jako velmi nízké, protože i tak budou v naprosté většině reálných situací naprosto dostatečné. Neopominutelným aspektem pro stanovení těchto limitů je i cena nezbytná k dosažení takových limitů, protože sbor, jako nevýdělečná organizace, je z hlediska případných nákladů na provoz výsledného řešení velmi limitován. Abych dokázala alespoň zhruba kvantifikovat tyto výchozí nízké limity právě s ohledem na cenovou dostupnost finální implementace, pokusila jsem se získat parametry nabídky varianty nejlevnějšího webhostingu největšího českého internetového registrátora Internet CZ, a.s., (Forpsi) a jeho služby Webhosting Easy ([19]). Nicméně, jak se ukázalo, přestože popis služby poskytnutý operátorem (poskytovatelem) zahrnuje poměrně podrobný seznam technických parametrů typu kapacita úložiště, použité servery, instalované verze platforem, či počet dostupných databází, jakékoli parametry vypovídající o výkonu (počtu simultánních přístupů), propustnosti (datovém toku), či dostupnosti (garantované) zcela chybí. Tato informace nebyla k dispozici ani na infolince poskytovatele.

Kapitola 3

Návrh aplikace a její implementace

Následující kapitola pojednává o výběru systémů použitých k vytvoření webové aplikace, o jejím návrhu s ohledem na hlavní požadavky a o implementaci.

3.1 Zvolení systémů

Před začátkem každého projektu si musí vývojář promyslet a rozhodnout, jaké systémy bude ve svém projektu používat. V dnešní době je výběr obrovský. V rámci přípravy jsem proto čerpala z mnoha různých informačních zdrojů, například z titulů „PHP, MySQL, JavaScript & HTML5 all-in-one for dummies“ [20], „Building applications with Symfony, CakePHP, and Zend Frameworks“ [21], „Extending symfony2 web application framework: optimize, audit, and customize web applications with symfony“ [22], ale i z mnoha dalších (zmiňovaných průběžně v textu celé této práce a shrnutých v kompletním seznamu na jejím konci).

Pro realizaci cílů této bakalářské práce jsem v první řadě potřebovala identifikovat jednotlivé fáze pro její vytvoření a následné nasazení. Vycházela jsem z počáteční vize, tedy přinést zadavateli nástroj, který jako webová aplikace splní jeho požadavky spojené s vedením agendy pěveckého sboru. S ohledem na předpokládaný objem ukládaných dat, jejich strukturu, očekávaný potřebný procesorový čas a datový tok při práci s nimi (plynoucími z analýz popsaných výše) jsem dále přihlížela i k odhadovaným pořizovacím a provozním nákladům na výslednou aplikaci (licence, hosting).

3.1.1 Hosting

S ohledem na výše uvedené jsem vycházela při volbě systémů vybíraných pro realizaci výsledné webové aplikace z takových, které jsou běžně dostupné u levných poskytovatelů webhostingu. Provoz současných webových stránek pro zadavatele zajišťuje Savana webhosting ([23]), nicméně výběr tohoto poskytovatele je dán historicky zapojením sponzora, který pravděpodobně v blízké budoucnosti přestane sbor podporovat a změna poskytovatele webhostingu je tak vcelku velmi pravděpodobná a zadavatelem předem schválená.

Pro mé další úvahy tedy nebylo nutné omezit výběr použitých technologií na takové, které by byly v nabídce výše zmíněného konkrétního poskytovatele, a zaměřila jsem se proto na takové, se kterými už mám nějaké osobní praktické zkušenosti a které jsou cenově dostupné.

3.1.2 Databázový systém

Jak jsem již zmínila výše, vycházela jsem při výběru použitých technologií z takových, které jsou pro mě již známé. U výběru databázového systému (databáze) jsem tak měla na výběr celkem ze tří.

Oracle Database je jednoznačně nejlepší robustní nástroj s kvalitní technickou podporou a širokou škálou implementací po celém světě dokazující kvalitu a spolehlivost tohoto nástroje, Bohužel, při jeho využití by výsledná cena aplikace byla mimo reálné možnosti sboru, a proto jsem jej musela z mého výběru vyjmout.

MySQL je pravděpodobně nejrozšířenějším nástrojem dostupným na poli českého webhostingu. Nabízí ho většina levných poskytovatelů a pravděpodobně by byl pro výslednou implementaci dostačujícím nástrojem. Bohužel, v tomto případě jsem se již v počátcích setkala s technickými obtížemi při jeho instalaci do mého vývojářského prostředí připravovaného pro vývoj aplikace a nechtěla jsem riskovat jejich eskalaci v průběhu jejího vývoje. Navíc framework zvolený pro samotnou aplikaci (Symfony - diskutováno níže) umožňuje připojit více typů databází, a tak by v případě potřeby bylo možné snadno provést dodatečně i migraci na MySQL.

PostgreSQL je dalším rozšířeným dostupným databázovým nástrojem a jeho použití pro mě bylo od počátku mnohem spolehlivější. Vzhledem k tomu, že všechny předpokládané technické nároky na vývoj a budoucí provoz vyvíjené webové aplikace splňoval, zvolila jsem pro svou práci právě tento nástroj.

3.1.3 Programovací jazyk

Programovacích jazyků existuje mnoho. Ale samozřejmě ne všechny se hodí pro všechno, proto je dobré se nad výběrem programovacího jazyka zamyslet a podívat se alespoň na některé výhody a nevýhody, které nabízí.

Já jsem pro svou aplikaci hledala programovací jazyk vhodný pro použití na straně serveru (back-end) a podporující integraci s databázovým systémem. Využití pouze front-end programovacích jazyků by pro takový účel nebylo dostatečné.

Před výsledným výběrem back-end programovacího jazyka jsem proto provedla následující úvahu. Např. C++ jsem z výběru předem vyloučila, protože nemá přímou podporu pro webové aplikace, a navíc je výrazně systémově orientovaný, což sice přináší vývojáři značnou svobodu, ale na druhou stranu vyžaduje důsledné ošetření chybových stavů a jeho použití je tak potenciálně riskantnější z hlediska bezchybnosti a spolehlivosti výsledného kódu. Ani možnost využití např. TreeFrog frameworku pro C++ Web development ([24]) nebylo dostatečnou výhodou pro využití právě tohoto programovacího jazyka. Do výběru jsem tedy zahrnula jazyky Java, Python a PHP. Na základě předchozích analýz víceméně všechny splnily očekávané budoucí nároky na robustnost výsledného řešení. Nicméně u poskytovatelů webhostingu je v případě Javy webhosting statisticky dražší zejména z důvody vyšší konzumace zdrojů na straně serveru a psaní kódu je zbytečně složitější. Do užšího výběru jsem tedy zahrnula Python a PHP, které jsou z hlediska výkonu, nároku na zdroje i z hlediska složitosti psaní kódu srovnatelné. Nicméně, u poskytovatelů webhostingu je PHP výrazně rozšířenější a navíc i moje osobní předchozí zkušenosti byly v případě PHP, který byl jedním z předmětů studia mého bakalářského oboru, významně rozsáhlejší, a proto padla finální volba právě na **PHP**.

V případě front-end programovacího jazyka byla volba celkem snadná, protože z běžně dostupných jazyků byly na výběr kromě základního **HTML** ještě **CSS** a **JavaScript**. V tomto případě jsem se rozhodla pro všechny, protože s vhodnou kombinací jejich využití jsem mohla na straně uživatele použít interaktivní prvky (např. rozbalení menu nebo zobrazení okna s žádostí o potvrzení požadované operace) bez nutnosti interakce i se serverem (tedy načítat takto mírně upravený zobrazený obsah ze serveru znovu). Navíc i použití některých pluginů (diskutováno níže) využití skriptovacího jazyka na straně klienta (uživatele) vyžaduje.

3.1.4 Framework

Využití frameworku při vývoji webové aplikace přináší hned několik výhod. K nejvýznamnějším patří zejména úspora času samotného vývoje aplikace, protože framework zajišťuje implementaci významné části výsledné funkcionality. Mnoho frameworků využívá MVC přístup, což usnadňuje dodržování „dobrých programovacích návyků“. Nepochybnou výhodou je také jistá míra zabezpečení, které je v dnešní době velmi důležité. Většina frameworků totiž umí odvrátit cross-site scripting útok, či SQL injection. ([25])

MVC je jeden z typů třívrstvé architektury, která se ve webovém vývojářství používá. Jedná se o oddělení dat od logiky a od prezentace uživateli, přičemž každá vrstva může komunikovat pouze s vrstvou o jedna výše či níže. Modelová vrstva se stará o přístup k datům a k databázi a komunikuje s controllerem, což je jakýsi prostředník, který ovládá většinu logických operací v aplikaci. Controller pak předává výstup vieweru, který pak např. vykreslí webovou stránku a zobrazí na ní data. ([26])

Při volbě preferovaného frameworku pro vývoj mé aplikace byl můj výběr o něco snadnější, protože již dle předchozích kritérií jsem si jako back-end programovací jazyk zvolila PHP. Zbývalo tedy vybrat vhodný nástroj, který by mi pomohl s přípravou základní sady kódů, protože nemělo smysl vyvíjet vše od úplného začátku. Naštěstí i tady bylo rozhodování vcelku snadné.

Bulma je k dispozici jako CSS front-end framework ([27]) umožňující snadné formátování výsledného vzhledu webové aplikace s využitím předdefinovaných CSS stylů. Již v základu obsahuje podporu responzivního chování na základě velikosti (rozměrů) zobrazované stránky a typu použitého zařízení. Velmi tak usnadňuje vytváření webových aplikací, které jsou primárně určeny pro zobrazování na různých typech (formátech) displejů. Přestože je k dispozici zdarma (jako open-source), je vybaven velmi kvalitní dokumentací mnohdy doplněnou i o praktické příklady implementace, např. v případech, kdy je nezbytné použít doplňkový kód v JavaScriptu.

jQuery je jedna z knihoven vytvořených jako nadstavba JavaScriptu. Já jsem se jQuery rozhodla využít pro jeho snadné použití, které mi v mnoha případech ulehčilo psaní kódu a umožnilo využít i další knihovny, které jQuery také využívají.

Symfony je dostupná jako **Symfony Web Application framework** ([28]). Jedná se o back-end framework vytvořený jako sada PHP komponent. Byla další součástí mého předchozího studia a vyhovovala všem požadavkům, které jsem na základě předchozí analýzy vyhodnotila jako klíčové. Navíc splňovala i kritérium cenové dostupnosti pro zadavatele, protože je distribuována pod MIT License ([29]), tedy při dodržení licenčních podmínek zcela zdarma. Mému výběru také významně pomohl fakt, že součástí Symfony je Doctrine, což je sada PHP knihoven speciálně vytvořená pro integraci s různými relačními databázemi včetně PostgreSQL (můj výběr) a MySQL (nejpravděpodobnější alternativa), tedy v případě potřeby by byla možnost migrace na jinou databázi možná. V neposlední řadě pro mě byly dalšími významnými výhodami i obsáhlá dokumentace produktu včetně ilustrativních příkladů a široká uživatelská komunita s diskusními

fóry nabízejícími dostatek příležitostí pro získání pomoci v případě potřeby. Množství výhod ještě navíc doplňuje i uspokojivé řešení obsluhy základních úkolů, jako např. URL routing nebo session management, a zajištění bezpečnosti (databáze mé aplikace obsahuje i některá osobní data uživatelů). CSRF ochrana je poskytována automaticky u všech formulářů a Symfony také zabraňuje například SQL injection (díky používání Doctrine ORM se nikde v aplikaci nemusí posílat přímý SQL kód). Symfony se vyznačuje i vysokým výkonem bez přehnaných nároků na systémové zdroje a také schopností efektivní podpory vývoje složitých aplikací včetně snadné údržby. Nabízí i podporu správy uživatelů včetně jejich přihlašování (autentizace) a správy rolí (přístupových oprávnění k různým sekcím aplikace - autorizace).

3.1.5 Doplňky, nadstavby a rozšíření

Velkou výhodou využití masově rozšířených programovacích jazyků je existence množství již hotových komponent, fragmentů kódu a různých šikovných nápadů dostupných v mnoha diskusních fórech a vývojářských komunitách. To mi umožnilo se v mé práci soustředit na její hlavní cíle a nemusela jsem se rozptylovat a zatěžovat vývojem rutinních procedur, které pouze zajišťují podpůrnou funkcionalitu bez jakékoli kreativní přidané hodnoty.

Hlavními doplňky, které jsem při vývoji své aplikace využila a některé začlenila i do výsledného kódu jsou tyto.

TinyMCE je jednoduchý, přehledný a intuitivní WYSIWYG editor se snadnou integrací do webových stránek a se širokou škálou nabízených funkcionalit včetně možnosti zobrazení a přímé editace zdrojového HTML kódu stránky. Je k dispozici i v české jazykové mutaci, což je zejména pro potřeby zadavatele významná výhoda. Je také uživatelsky (autorsky) přizpůsobitelný, což umožňuje začlenit do výsledné aplikace pouze ty jeho části, které byly užitečné pro účely použití právě v mé aplikaci. Navíc skladba využitých komponent a předpokládaná frekvence a rozsah použití umožňuje volbu licenčního modelu s nulovými náklady. Zajímavou a v mém případě užitečnou vlastností tohoto editoru je i možnost volby verze mezi plnou integrací do výsledné aplikace (self-hosted) a verzí provozovanou autorskou firmou (cloud) a volanou pouze přes API, což pro účely mé aplikace plně postačuje.([30])

Selectize je knihovna (plugin) vytvořená s použitím jQuery a kombinující textbox a selectbox. Na webové stránce lze využít např. pro implementaci výběrového seznamu s možností textového vyhledávání v rámci jeho položek, což značně usnadňuje výběr položek u dlouhých seznamů. Licenčně je jeho využití zdarma, a to v případě potřeby i pro komerční účely. ([31])

Twig je druh skriptovacího jazyka umožňující přípravu šablon pro vytváření zobrazovaného obsahu webové aplikace. Obsahuje prvky procedurálního programovacího jazyka pro řízení běhu programu (cyklus, rozhodování, ...), což přináší širokou škálu možností jeho využití při tvorbě dynamického obsahu doplňovaného např. z propojeného databázového systému. Je součástí frameworku Symfony. ([32])

Font Awesome je šikovný doplněk pro vylepšení vzhledu výsledné webové aplikace s využitím vektorových obrázků (ikon), které tak mohou být snadno přizpůsobeny autorovým požadavkům na velikost, barvu, stínování, či cokoli, co poskytuje CSS. K dispozici je zdarma (jako open-source). ([33])

Drawio (Diagrams.net) je aplikace pro vytváření diagramů. S její pomocí jsem vytvořila například diagramy aktivit či diagramy případů užití. ([34])

Balsamiq je modelovací nástroj pro snadné vytváření náčrtků (skic) grafických návrhů (designů) aplikací, webových stránek a jiných uživatelských rozhraní. S výhodou lze využít pro vytváření wireframů navrhovaných webových stránek před jejich finalizací s využitím plné grafiky. ([35])

PHPStorm je nástroj pro snazší psaní zdrojového kódu v PHP. Obsahuje integrované odkazy na dokumentaci, krátké anotace jsou k dispozici přímo při psaní kódu, obsahuje debugger, kontroluje syntaxi, v rámci které syntakticky navrhuje (doplňuje) text kódu. ([36])

Docker je moderní systém, který významně usnadňuje nasazování (instalaci) aplikací. Pro mě byl přínosem pro možnost jednou sestavit instalační předpis a umožnit tak následnou automatickou instalaci ať už do cílového prostředí nebo pro testovací a laboratorní využití kdekoli jinde. ([37])

GIT je verzovací systém užitečný k evidenci a sledování verzí vyvíjené aplikace. V případě potřeby umožňuje snadné využití předchozích verzí kódu a zvládá i vývoj v týmu (tedy provádění změn více vývojáři současně) a to včetně větvení verzí, tedy sledování vzniku různých verzí (větví) aplikace s různými rysy (funkcemi). Pravděpodobně nejrozšířenější implementací je veřejně dostupný github ([38]); já jsem pro svou práci použila školní gitlab ([39]). ([40])

3.2 Návrh aplikace

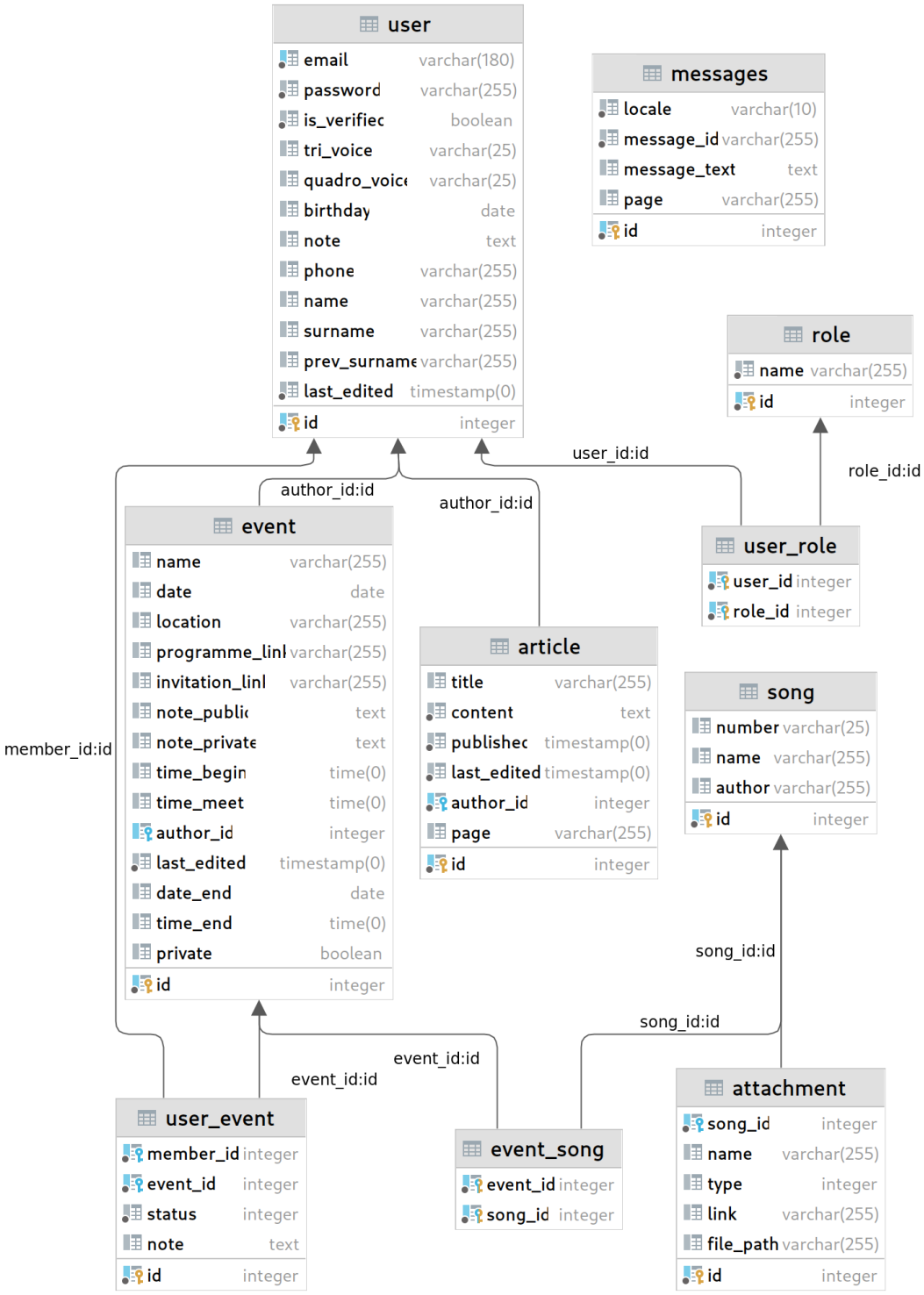
Po provedení a vyhodnocení analýzy současných webových stránek zadavatele a několika dalších pěveckých sborů a dále po zvážení možností a výběru vhodných nástrojů a systémů pro vytvoření a implementaci mé webové aplikace jsem pokračovala návrhem základní struktury a jejím postupným rozšiřováním.

3.2.1 Databázový model

V první řadě jsem si na základě úvodní analýzy a specifikace funkčních požadavků rozpracovala strukturu dat, se kterými chci pracovat. Promyslela jsem základní údaje (databázové entity), které chci v aplikaci ukládat, a doplnila vazby (relace) mezi nimi. Tím jsem vytvořila diagram databáze 3.1 a v návaznosti na něj jsem pak pokračovala v definicích dalších vstupů.

User je jednou ze tří hlavních entit. Reprezentuje skutečného uživatele aplikace, a kromě systémových atributů (stav ověření e-mailu nebo datum poslední změny) definuje zejména základní osobní údaje (jméno, příjmení, e-mail, ...) a dále pak doplňující charakteristiky důležité pro člena sboru, tedy zařazení do konkrétního hlasu ve vícehlasých skladbách. Toto rozdělení uchovává rozdělení u trojhlasů a čtyřhlasů, protože vícehlasy jsou v repertoáru sboru zcela výjimečné a dvojhlasy se odvozují ze čtyřhlasů, a není proto přínosné ukládat je zvlášť. Ohledně množiny atributů uživatele je ještě namístě podotknout, že s ohledem na potenciální citlivost ukládaných osobních údajů jsem se snažila maximálně zredukovat množství uchovávaných informací pouze na takové, které jsou pro zajištění funkce agendy pěveckého sboru nezbytně nutné. Neshromažďuji proto např. rodná čísla nebo poštovní adresy.

Song je další z hlavních entit. Reprezentuje skladbu z repertoáru sboru. V zásadě obsahuje pouze autora a název a dále pak pořadové číslo v repertoáru sboru. Nicméně, na skladbu mohou být navázané přílohy v podobě souborů s doplňujícími daty. Takové přílohy jsou reprezentovány pomocnou entitou `Attachment`.



Obrázek 3.1 Diagram databáze aplikace

Event je poslední z hlavních entit. Reprezentuje událost obvykle typu koncert (ale třeba i soustředění nebo mimořádná zkouška) a definuje pro ni atributy pro záznam všech důležitých informací o ní. Kromě základních informací typu název, místo konání nebo čas začátku a konce samotné události a čas srazu před ní jsou zde i další důležité informace, jako doplňující poznámky (rozdělené na interní a veřejné), odkaz na program nebo na pozvánku, informace o tom, zda je událost soukromá, a samozřejmě, i kdo ji vytvořil a kdy byla naposledy aktualizovaná.

Attachment je pomocná entita doplňující hlavní entitu skladby (`Song`). Umožňuje přidat ke skladbě datový soubor (přílohu) např. s notovým zápisem skladby nebo nahrávkou. Soubor může být přidán buď fyzicky s uložením v aplikaci nebo jen jako reference na externí zdroj. Množina atributů pak obsahuje jednak typ přílohy (noty nebo audio), název umožňující blíže specifikovat její obsah (např. MIDI nahrávka druhého hlasu nebo notový zápis celé partitury) a dále systém doplní cestu k uložení samotné přílohy (souboru).

User_Event je pomocná entita využívající metody dekompozice vztahu M:N k reprezentaci přiřazení uživatelů k událostem (tedy evidence účasti na jednotlivých událostech). Kromě samotné vazby obsahuje navíc i status (bez odpovědi, přijde, ještě neví, nepřijde) a možnost přidat poznámku.

Event_Song je pomocná entita využívající metody dekompozice vztahu M:N k reprezentaci přiřazení skladeb k událostem (tedy repertoár / playlist).

Role je pomocná entita obsahující názvy uživatelských rolí (přístupová práva). Tato entita není uživatelsky přístupná přímo (pouze referencí přes uživatelské rozhraní), což vylučuje riziko neodborného zásahu do obsahu tabulky, protože to by mohlo vést ke ztrátě funkcionality správy přístupových práv jako takové. Hierarchie rolí je diskutována v samostatné sekci níže.

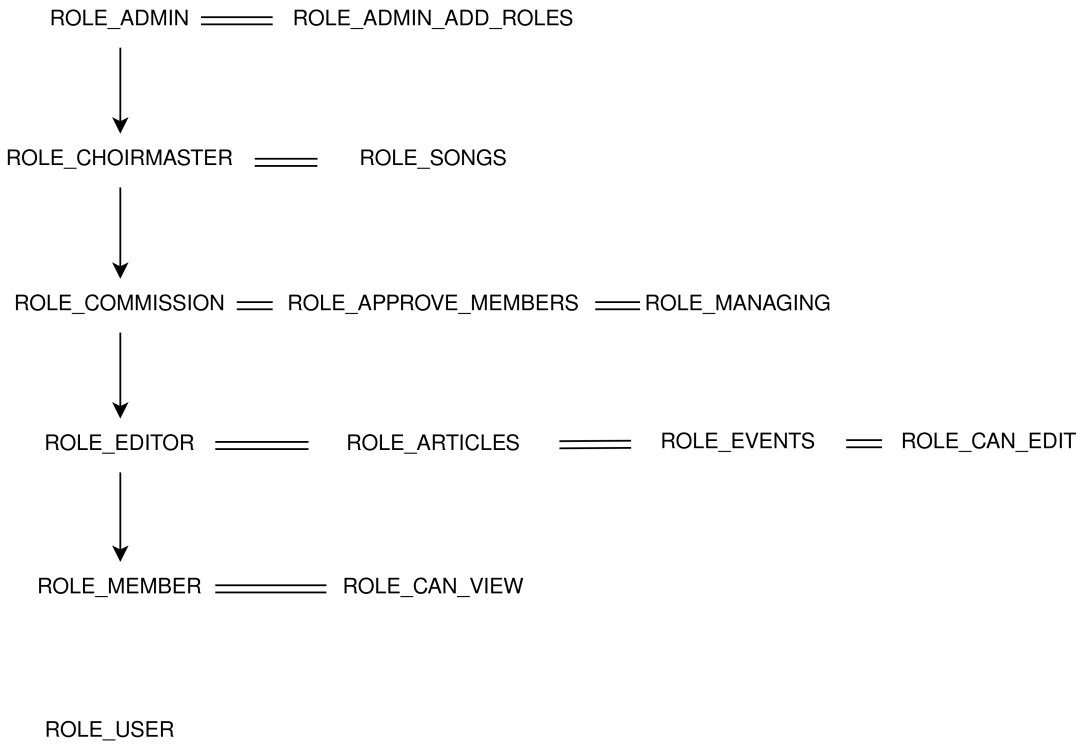
User_Role je pomocná entita využívající metody dekompozice vztahu M:N k reprezentaci přiřazení rolí (přístupových práv) jednotlivým uživatelům.

Article je pomocná entita obsahující příspěvky (aktuality) od vedení sboru určené členům sboru (interní informace) nebo i návštěvníkům webu (veřejná sekce). Kromě samotného obsahu příspěvku a jeho názvu entita obsahuje autora a datum vytvoření a poslední aktualizace.

Messages je pomocná entita obsahující samotné texty stránek webové aplikace. Kromě samotného textu stránky obsahuje i specifikaci jazykové mutace (tedy jazyk, ve kterém je text napsaný) a referenci na stránku, kde je text zveřejněný.

3.2.2 Hierarchie rolí / Přístupová práva

Názvy rolí se používají k implementaci chování aplikace jejich interpretací v kódu aplikace. Základní princip použitý v mé práci k definici rolí spočívá v odvozování jednotlivých rolí od rolí definovaných již dříve, a to jejich kombinací. Nově definovaná role tak získá (zdědí) všechny role použité pro její definici. Hierarchie rolí je znázorněná na obr. 3.2, kde šipka ukazuje na roli, ze které je odvozena role nová (nadřazená) a dále jsou dvojitou čárou vedle příslušné role připojeny i role elementární, které daná role k roli rodičovské přidává.



Obrázek 3.2 Hierarchie rolí

ROLE_USER je základní role, která se automaticky přiděluje registrovaným uživatelům. Z hlediska přístupových práv má takto registrovaný uživatel stejná práva, jako neregistrovaný, ale navíc má právo upravovat některé atributy svého vlastního profilu. To umožňuje provádět registrace nových uživatelů bez nutnosti součinnosti některého ze správců. Jakmile je uživatel zaregistrovaný (může se zaregistrovat sám) a ověřený (tedy potvrdil přijetí ověřovací e-mailové zprávy zaslané systémem na e-mailovou adresu zadanou při registraci), zaznamená se tato skutečnost do záznamu o uživateli a po ztotožnění se skutečnou osobou už pak správce může jen povýšit uživatele na člena.

ROLE_CAN_VIEW je elementární role umožňující zobrazení interního (neveřejného) obsahu webové aplikace.

ROLE_ARTICLES je elementární role umožňující přidávání a mazání aktualit ve veřejné / interní části.

ROLE_EVENTS je elementární role umožňující přidávání a mazání událostí.

ROLE_CAN_EDIT je elementární role umožňující úpravy ve veřejné části stránek a úpravy dalších interních informací (práce s přílohami, úprava skladeb, událostí, aktualit, ...)

ROLE_APPROVE_MEMBERS je elementární role umožňující schvalování nových členů (tedy povyšovat uživatele na členy: `ROLE_USER` → `ROLE_MEMBER`.

ROLE_MANAGING je elementární role umožňující správu uživatelů.

ROLE_SONGS je elementární role umožňující přidávání a mazání skladeb.

ROLE_ADMIN_ADD_ROLES je elementární role umožňující přidávání a odebírání rolí (přístupových práv).

ROLE_MEMBER je základní role pro ověřené uživatelské účty ztotožněné s jejich vlastníky (členy sboru). Tato role umožňuje přístup (čtení) k interním informacím, k seznamu skladeb vč. jejich příloh, k podrobnějším detailům událostí (např. čas srazu) i k neveřejným (soukromým) událostem a detailům o nich.

ROLE_EDITOR je role rozšiřující roli člena a umožňující upravovat existující obsah stránek aplikace (zejména veřejné části). Tato role neumožňuje přidávat nový nebo zcela odstraňovat existující obsah.

ROLE_COMMISSION je role pro členy výboru sboru rozšiřující roli editora. Přidává více možností v úpravách obsahu stránek, umožňuje schvalovat členy a také vyplňovat (upravovat) jejich údaje (např. úprava profilu nebo aktualizace účasti při událostech).

ROLE_CHOIRMASTER je role pro sbormistra rozšiřující roli člena výboru sboru. Sbormistr má navíc právo přidávat a mazat skladby.

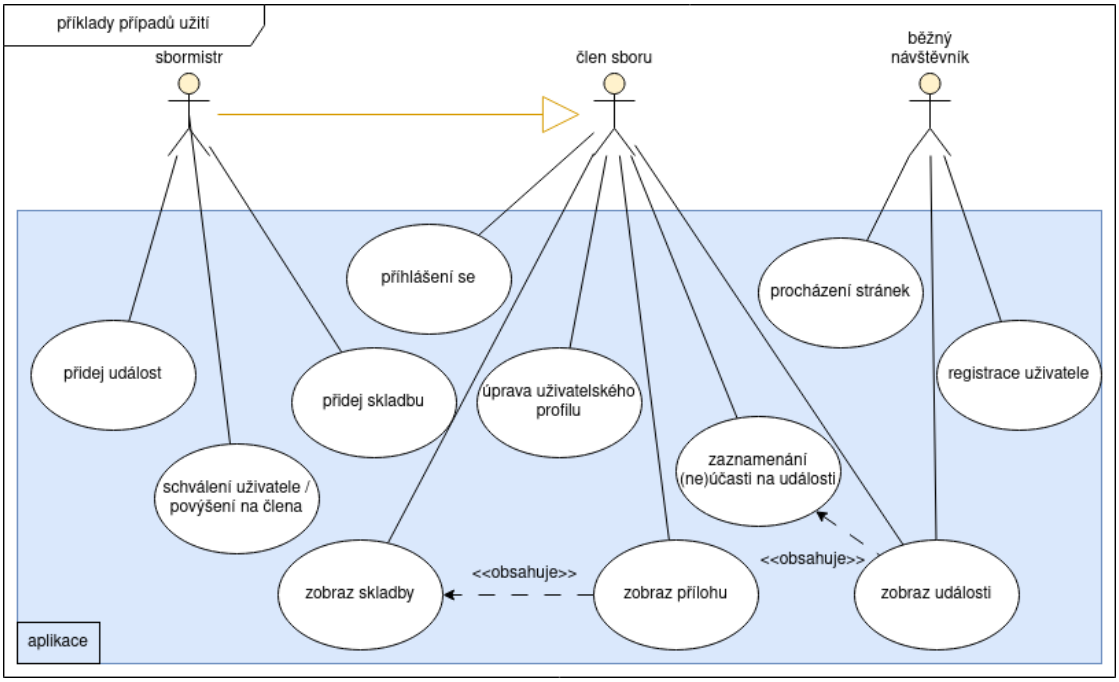
ROLE_ADMIN je role pro správce rozšiřující roli sbormistra o právo přidělovat práva ostatním uživatelům.

IS_AUTHENTICATED je speciální typ role, která postrádá klasickou specifikaci používanou pro role tradiční, ale její chování je roli velmi podobné. V mé aplikaci tuto roli používám k identifikaci přihlášených uživatelů v situacích, kdy je žádoucí, aby se webová aplikace chovala ve stejných situacích (na stejných stránkách) různě podle toho, zda je její uživatel již identifikován (přihlášený), či nikoli.

3.2.3 Aktivity a případy užití

Dalším krokem při vytváření návrhu aplikace bylo rozmyšlení aktivit, které budou při práci s aplikací využívané. I tady byla hlavním zdrojem specifikace funkčních požadavků, tedy jeden z hlavních výstupů úvodní analýzy. V rámci těchto úvah jsem identifikovala tyto základní aktivity, jejichž nasazení bude třeba realizovat. Ilustraci základních případů užití dokumentuje obrázek 3.3.

Registrace uživatele je aktivita, která umožňuje založit (registrovat) nový uživatelský účet a doplnit hodnoty do základní sady jeho atributů. Registrace bude dostupná pomocí odkazu z hlavní stránky webové aplikace přes registrační stránku s interaktivním formulářem pro vyplnění základních profilových informací. Součástí procesu registrace může, ale nemusí, být kompletní vyplnění uživatelského profilu a ověření uživatelovy e-mailové adresy. Pro přihlášení uživatele tyto informace nejsou vyžadovány, nicméně mohou být podmínkou ze strany správce (člena výboru sboru) pro schválení žádosti o povýšení uživatele na člena sboru. Ověření e-mailové adresy proběhne pomocí automatického zaslání ověřovací zprávy na e-mailovou adresu uvedenou v rámci registrace. Tato zpráva bude obsahovat unikátní odkaz s omezenou platností, kterým uživatel svou e-mailovou adresu potvrdí, a webová aplikace na základě tohoto potvrzení označí v uživatelském profilu tento údaj za ověřený.



Obrázek 3.3 Návrh případů užití

Schválení uživatele / povýšení na člena je aktivita určená pro uživatele s oprávněním (rolí) člena výboru sboru (nebo vyšším), která umožňuje schválit registrovaného uživatele jako člena (tedy povýšit oprávnění z `ROLE_USER` na `ROLE_MEMBER`). Technicky pro tento krok nejsou kladeny žádné požadavky, nicméně administrativně je na zvážení schvalovatele, jakou míru detailu vyplnění uživatelského profilu (včetně ověření e-mailové adresy) bude vyžadovat před provedením akce. Aktivita je znázorněna na obrázku 3.4.

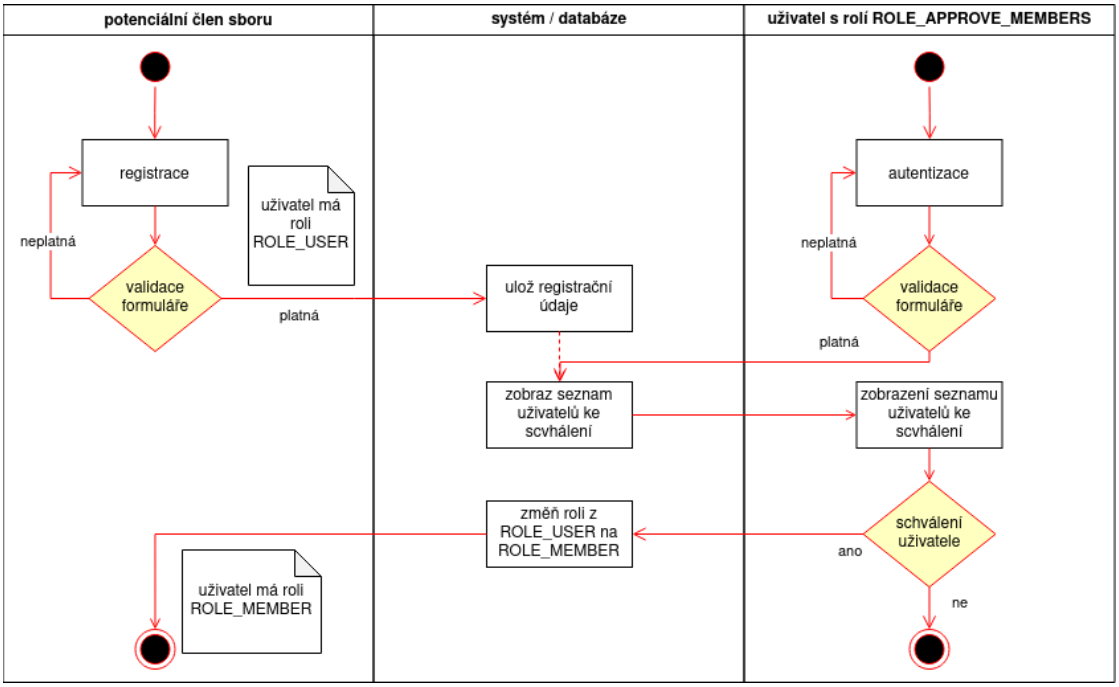
Úprava uživatelského profilu je aktivita určená pro samotné uživatele k úpravě některých atributů jejich vlastního profilu a dále pro oprávněné uživatele k úpravě všech atributů profilu kteréhokoli uživatele.

Přidání události je aktivita určená pro oprávněné uživatele k přidání nové události a specifikaci jejich atributů včetně možnosti zadání seznamu skladeb. Systém automaticky doplní některé atributy, jako např. autora události nebo datum vytvoření.

Zaznamenání účasti na události je aktivita určená pro členy sboru, která umožňuje jejich vlastní registraci na danou událost. Podobně lze iniciovat i aktivitu určenou k aktualizaci účasti. Oprávnění uživatelé mohou tyto aktivity provést i za jiné členy sboru. Ukázku implementace této aktivity ve výsledné aplikaci ilustruje obrázek 3.5.

Přidání skladby je aktivita určená sbormistrům (příp. správcům), která umožňuje dělat změny v seznamu skladeb (repertoáru sboru).

Procházení stránek je aktivita určená uživatelům webové aplikace, která umožňuje číst, případně upravovat, její obsah na základě příslušných oprávnění jejich uživatelů. Rozlišuje se tedy dostupnost obsahu podle identity uživatele (anonymní / přihlášený) i podle stupně přiděleného oprávnění (role).



Obrázek 3.4 Návrh aktivity schvalování členství

3.2.4 Návrh uživatelského rozhraní (GUI)

Vzhledem ke skutečnosti, že členskou základnu sboru zadavatele tvoří z velké části senioři, kteří mají mnohdy problém s akceptací změn, snažila jsem se při návrhu uživatelského rozhraní maximálně zachovat styl současného webu, avšak s citlivým posunem k modernímu designu a s maximálním respektem ke specifikaci funkčních požadavků, které jsem získala jako součást úvodní analýzy. Nejprve jsme základní funkce diskutovali ústně v čistě teoretické rovině a následně jsem s využitím modelovacího nástroje Balsamiq vytvořila náčrtky (wireframy) základů stránek budoucí webové aplikace a ty znovu se zadavatelem prodiskutovala. Výsledkem těchto postupných iterací pak byl finální design webové aplikace v plné grafice s tím, že výsledná webová aplikace je uzpůsobená k tomu, aby některé sekce mohly být ponechány k závěrečným úpravám plně na zadavateli.

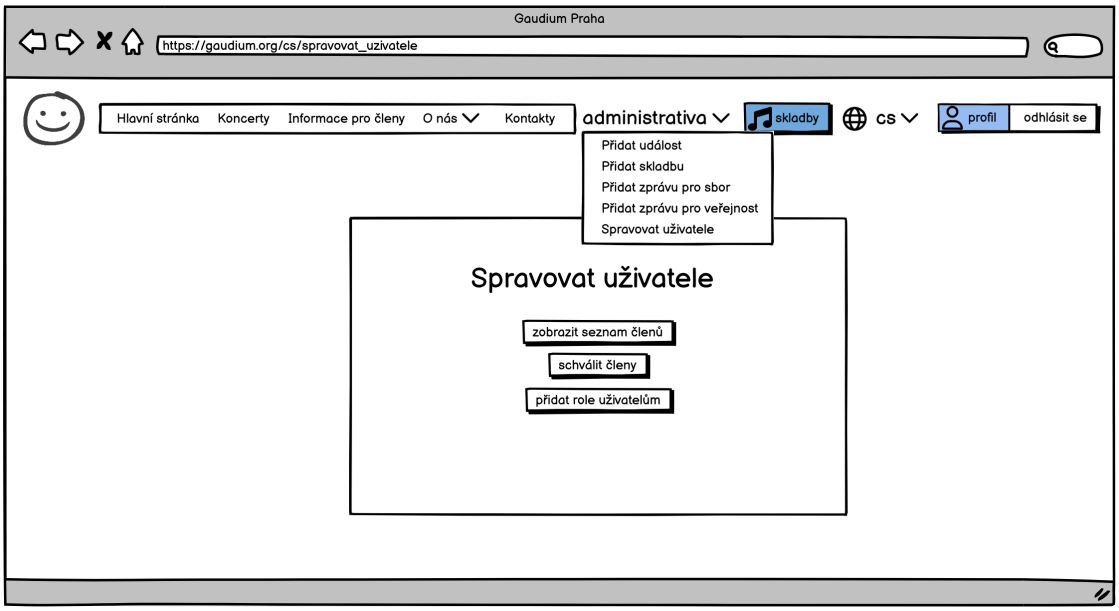
Jako příklady diskutovaných náčrtků uvádím obrázek 3.6, který navrhuje vzhled budoucí administrátorské sekce pro správu uživatelů a rozbalovací nabídky (menu) dalších administrátorských akcí, jako je například přidání nové události či skladby. Pro srovnání uvádím i snímek obrazovky aplikace po implemntaci připraveného návrhu na obrázku 3.8. Zároveň je zde i návrh menu zobrazovaného přihlášeným členům sboru, tedy např. tlačítko pro přístup k seznamu skladeb. Dalším příkladem je obrázek 3.7 s návrhem kalendáře s plánovanými událostmi tak, jak bude k dispozici i pro nepřihlášeného uživatele (návštěvníka webu).

3.3 Implementace aplikace

3.3.1 Vývojové prostředí

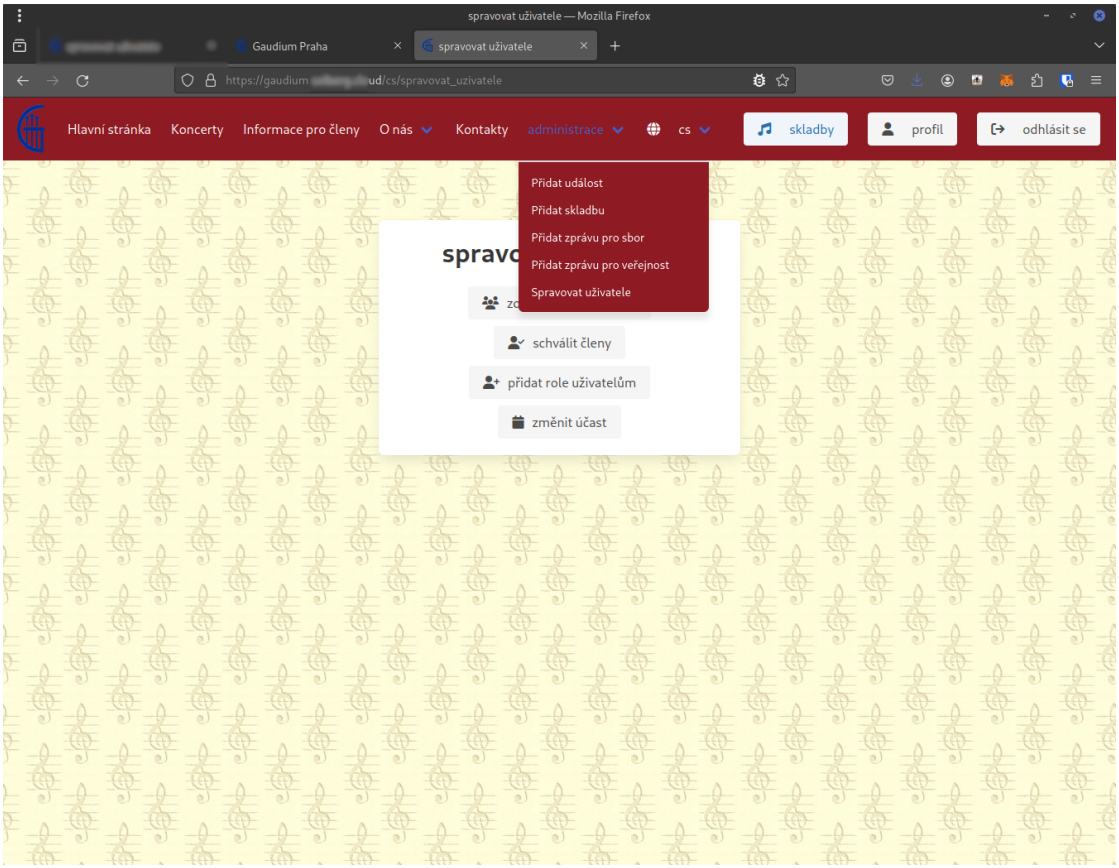
PHPStorm Jako hlavní předpoklad pro úspěšné zahájení vývoje mé webové aplikace jsem si připravila vývojové prostředí, které mi umožnilo oprostit se od zbytečných rutinních operací

Obrázek 3.5 Kalendář zobrazený členem sboru na mobilním zařízení



Obrázek 3.6 Grafický návrh zobrazení administrátorské sekce

Obrázek 3.7 Grafický návrh zobrazení kalendáře pro běžného návštěvníka webu



Obrázek 3.8 Zobrazení menu uživatele se správcovskou rolí

a soustředit se na design a vývoj aplikace samotné. PHPStorm je integrované vývojové prostředí (IDE) obsahující všechny pro mě důležité komponenty potřebné k hladkému, efektivnímu a úspěšnému vývoji. Jedná se zejména o editor se zvýrazněním syntaxe a syntaktickou nápovědou (návrhy) při psaní kódu, debugger s funkcemi breakpointů a inspektorem obsahu proměnných a v neposlední řadě i s náhledem do obsahu propojené databáze. Užitečnost těchto komponent je dále popsána v sekci testování dále v textu.

Docker aplikaci jsem použila pro spuštění a otestování vytvořených souborů ke konfiguraci dockeru, což pak bylo použito k snadnému nasazení aplikace. Více o implementaci dále.

Systémové aplikace typu SSH klient nebo FTP klient jsem používala pro přenos obsahu webové aplikace (programového kódu) do cílového hostingového prostředí.

3.3.2 Vývoj aplikace

Jakmile jsem měla zkompletovanou většinu vstupních podkladů získanou jako výsledek úvodní analýzy a definic funkčních a nefunkčních požadavků, vhodně zvolené systémy a nástroje pro samotnou tvorbu aplikace, připravené vývojové prostředí a vytvořené základní diagramy a schémata pro práci s daty a designem výsledné aplikace, přistoupila jsem k zahájení samotného vývoje.

Vývoj jsem zahájila sestavením základní struktury a jejím naprogramováním. Postupně jsem pak přidávala jednotlivé komponenty s příslušnými funkcionalitami a každou takovou fázi jsem doplnila otestováním a opravou případných chyb (popsáno též v sekci testování). Průběžně jsem také vylepšovala celkový design aplikace s důrazem na zachování přehlednosti a ergonomie. V zásadě jsem během vývoje každé jednotlivé fáze použila velmi zjednodušenou a zkrácenou formu life-cycle managementu a jeho aplikací jsem se přes tyto postupné vývojové iterace propracovala až k cíli.

Výsledná struktura aplikace je tedy tvořena několika složkami se soubory, které obsahují:

- zdrojové kódy v PHP
- konfigurační soubory v yaml
  - např. překlady nebo definice zabezpečení (hierarchie rolí, cesta pro Login a Logout)
- soubory CSS s definicemi stylů
  - vzhledem k použití Bulmy stačí jediný CSS soubor pouze s drobnostmi
- soubory s výchozími daty obsahu databáze
  - získané exportem z databáze v PHPStormu ve formátu SQL

Zdrojový kód v PHP je rozdělen do několika souborů členěných do částečně samostatných logických celků. Základní členění je rozdělené podle jednotlivých entit (viz databázový model) a každý z těchto celků obsahuje kód pro vytvoření dané entity a práci s ní. Prakticky je zde využito knihovny Doctrine, což je součást Symfony určená pro práci s databázemi. Její funkce `make:entity` vytvoří základní kostru entity spolu s vytvořením *repository* s definicemi pomocných tříd pro práci s daty v databázi. U některých jsem pak navíc definovala vlastní rozšíření s využitím funkce `createQueryBuilder`.

Dalším datovým typem použitým při vývoji aplikace byla třída `AbstractController` (součást Symfony), ze které jsem odvodila vlastní *controllery* využívané pro řízení a implementaci chování aplikace při jednotlivých operacích. Hlavní funkcionalita, kterou tato třída definuje, je sada metod a cest k nim. Při implementaci je tak možné vlastní metody pojmenovat vlastními interními aliasy a následně volat na základě veřejného odkazu použitého v URL adrese zadané uživatelem

(„kliknuté“ při používání aplikace). Tato služba je implementovaná s využitím přesměrování reakce web serveru na „handle HTTP request“ a následně „return HTTP response“.

- `BaseController.php` – základní třída pro usnadnění vývoje některých dalších controllerů; obsahuje funkci showError, která vrací chybovou stránku
- `EditController.php` – třída odvozená z BaseControlleru pro vytváření a úpravu (editaci) textů a dat zadávaných do aplikace
- `ViewController.php` – třída odvozená z BaseControlleru pro zobrazování (view) textů a dat z aplikace
- `RegistrationController.php` - třída odvozená přímo z AbstractControlleru pro registraci nového uživatele a odesílání e-mailů
- `LoginController.php` – třída odvozená přímo z AbstractControlleru pro přihlášení (login) registrovaného uživatele
- `TestController.php` – třída odvozená z BaseControlleru pro testování („výrobě“) chybových stavů a možnosti následného vývoje reakce na chybový stav

Ve složce *Form* jsou speciální třídy, které jsem vytvořila odvozením z `AbstractType` (součást Symfony). Použitím metody `buildForm` jsem tyto třídy následně využila pro vytvoření formulářů pro různé účely (např. pro registraci uživatele, přidání skladby, úprava příspěvku apod.).

Templates je složka se soubory, které jsem vytvořila jako šablony s využitím syntaxe skriptovacího jazyka Twig. Následně se příslušné šablony využívají v jednotlivých controllerech pro generování výsledných stránek a jejich dynamického naplnění aktuálními daty. Jednou z užitečných vlastností, které šablony ve Twigu mají, je možnost jejich dynamického rozšiřování a vzájemného vkládání, což umožňuje využívat jeden blok kódu na více místech podobně, jako např. volání funkce. Tím, že Twig obsahuje i prvky pro řízení zpracování kódu typu for cyklus, je možné v rámci dynamického vytváření stránek aplikace používat i relativně jednoduché šablony pro vytvoření stránek s opakujícím se obsahem, např. seznamem událostí (koncertů). Dalším přínosem je možnost využívat definici bloků kódů se specifikací jejich umístění, což jednak zaručí správné umístění výsledného obsahu na správné místo na stránce a jednak to umožňuje i nahrazení příslušného bloku jiným, případně jeho spojením s obsahem rodičovské šablony.

Do složky *public* je možné umístit soubory dostupné veřejně mimo kontrolu aplikace. Např. zdrojové kódy (např. vstupní bod aplikace index.php), css styly nebo obrázky.

Složka *private_uploads* je privátní složka spravovaná aplikací a využívaná pro umístění uživatelských souborů (např. notových zápisů nebo médií).

Složka *translations* obsahuje soubory ve formátu yaml sloužící k definování slovníku používaného pro překlady (jazykové mutace) vybraného typu obsahu stránek aplikace (např. formulářů, menu nebo tlačítek). Uživatelské texty typu informace o sboru nebo kroniku tímto způsobem překládat nelze a je v takových situacích nutné vložit překlad celé stránky.

Složka *config* obsahuje soubory ve formátu yaml sloužící k nastavení parametrů Symfony včetně např. definice rolí uživatelů (typů přístupových oprávnění).

Složka *data* slouží pouze pro úvodní naplnění databáze po instalaci buď jako demo (na médiu přiloženém k této práci) nebo např. po migraci k jinému poskytovateli hostingu; pro samotnou webovou aplikaci a její funkci nejsou tyto soubory nutné, pokud uživatel začíná s prázdnou aplikací a přidává pouze data vlastní.

Složka *migrations* je určená k migracím databáze dostupným jako funkce Doctrine, která slouží k úpravám struktury databáze např. při záměrných změnách (v mém případě jsem v rámci vývoje aplikace postupně přidávala jednotlivé entity definované v databázovém modelu), případně lze využít i k úpravám struktury při migraci na jiný typ databázového systému, který některé struktury použité ve zdrojové databázi nepodporuje.

Dalšími důležitými soubory potřebnými k vývoji a k následné instalaci a běhu aplikace jsou:

- `.env` - nastavení klíčových parametrů prostředí
- `.gitignore` - soubor s definicí obsahu neukládaného do gitu
- `Caddyfile` - soubor s nastavením parametrů Caddy serveru
- `composer.json` - soubor se specifikacemi potřebnými pro `Composer` (součást Symfony)
- `composer.lock` - soubor se specifikací aktuální verze (buildu) aplikace
- `docker-compose.yml` - soubor s nastavením parametrů pro Docker
- `Dockerfile` - soubor s nastavením parametrů pro vytvářený image Dockeru
- `symfony.lock` - soubor se specifikací aktuální verze (buildu) Symfony

3.3.3 Vývoj klíčových částí aplikace

Jak už jsem popsala v předchozí části, v rámci základního postupu vývoje aplikace jsem se držela výstupů z předchozí analýzy a schémat zahrnutých do návrhu aplikace. Postupně jsem tak doplňovala jednotlivé komponenty tvořící výslednou aplikaci. Některé části pak vyžadovaly dodatečnou pozornost.

Implementace rolí umožňuje nastavit chování aplikace tak, aby mohla ve stejných situacích (při přístupu na stejné stránky) různými uživateli reagovat různě podle stupně oprávnění (rolí) přidělených těmto uživatelům. Klíčem k jejich použití je funkce `isGranted`, která je v Symfony k dispozici jako nástroj ke zjištění, zda má uživatel přidělená konkrétní přístupová práva. Jako parametr funkce `isGranted` je použit název role. Přitom samotná role je definovaná pomocí souboru `security.yaml`, který obsahuje definici všech použitých rolí. Velmi přínosná je metoda interpretace těchto definic, protože ta zahrnuje i možnost dědění definic jednotlivých rolí (tedy respektuje hierarchii a dává tak možnost zahrnout do definice nové role jednu nebo více již definovaných jiných rolí), což významně zjednodušuje a zpřehledňuje jejich praktické použití. (To bylo i vidět na obr. 3.2.)

Na druhou stranu Symfony vyhodnocuje přiřazení zděděných rolí pouze interně a není tak k dispozici kompletní seznam všech držitelů konkrétní zděděné role. Proto např. k vyhledání všech členů sboru (tedy de facto odfiltrování uživatelů jen zaregistrovaných) bylo nutné napsat vlastní kód (viz ukázka kódu 3.1), protože vyhledání uživatelů s přiřazenou rolí ROLE_MEMBER vrací pouze uživatele s přímým přiřazením této role, ale např. správce (s rolí ROLE_ADMIN) nevrátí, přestože i ten roli člena zdědil.

Výpis kódu 3.1 PHP funkce v UserRepository k získání všech členů sboru

```
    public function findAllExceptRoleUser()
{
    return $this->createQueryBuilder('u')
        ->leftJoin('u.groups', 'r')
        ->where('r.name != :role')
        ->setParameter('role', 'ROLE_USER')
        ->getQuery()
        ->getResult();
}
```

IS_AUTHENTICATED je speciální typ role, která postrádá klasickou specifikaci používanou pro role tradiční, ale její chování je roli velmi podobné. Symfony používá tuto roli k identifikaci přihlášených uživatelů. To je užitečné v situacích, kdy je žádoucí, aby se webová aplikace chovala ve stejných situacích (na stejných stránkách) různě podle toho, zda je její uživatel již identifikován (přihlášený), či nikoli. Symfony nabízí ještě detailnější rozšíření této role podle toho, zda proběhlo skutečné přihlášení (`IS_AUTHENTICATED_FULLY`) anebo bylo použito z historie (`IS_AUTHENTICATED_REMEMBERED`), ale s ohledem na velmi nízkou citlivost dat dostupných ve webové aplikaci jsem se rozhodla toto rozlišení zanedbat.

Lokalizace aplikace je metoda jejího překladu do jiných jazyků (přizpůsobení pro budoucí možnost přidání dalších překladů), která byla další celkem zajímavou částí vývoje.

Základním záměrem bylo vyhnout se nutnosti vytváření samostatných stránek a ovládacích prvků pro každou jazykovou mutaci zvlášť. Proto jsem zvolila možnost dostupnou v rámci komponenty `Translation` (součást Symfony), která implementuje využití systému definice klíčových slov a jejich zaznamenání do souboru se slovníkem vytvořeného ve formátu yaml spolu s jejich překladem do alternativního jazyka. Tento slovník je pak následně automaticky využíván přímo Symfony prostřednictvím služby `EventListener`, která po kliknutí na odkaz pro změnu jazyka umožňuje přenastavení lokalizace celé aplikace na jiný jazyk. To má za následek automatické použití definovaného překladového slovníku pro generování cílového obsahu ve zvoleném jazyce kdykoli je v aplikaci požadováno zobrazení obsahu s klíčovými slovy. Samozřejmě kód aplikace musí být pro tento systém použití předem připraven a klíčová slova musí být používána jako výhradní způsob definice obsahu zobrazovaných stránek.

Z výše uvedeného popisu vyplývá, že tento systém je určen pouze pro automatický překlad systémového obsahu a nelze jej využít pro překlady přirozeného jazyka, tedy uživatelského obsahu stránek vytvořených správcem.

Jazykové mutace v podobě podpory překladů i uživatelského obsahu jsou aplikací podporovány s využitím několika triků. První podmínkou je, že schéma stránek je pevně dané a uživatelé (správci aplikace) jej nemohou měnit, tedy přidávat nové stránky nebo odstraňovat stávající. Změna schématu je samozřejmě možná také, ale vyžaduje specifický formát u nově přidávaných stránek tak, aby vyhovoval dalším podmínkám, tedy zejména struktuře stránky samotné, kdy prakticky každá stránka s uživatelským obsahem je vytvořena jako prázdná pouze s funkcí Twigu `{{ message.getMessageText() | raw }}`, která s využitím nastavených parametrů před jejím zobrazením doplní z databáze její obsah podle aktuálně nastavené (uživatelem zvolené) preference zobrazovaného jazyka. Uživatelské překlady jsou pak díky tomu implementovány velmi jednoduše tak, že uživatel s rolí ROLE_EDITOR prostě vybere preferovaný jazyk, zobrazí požadovanou stránku, ta se v tu chvíli už zobrazuje ve zvolené jazykové mutaci, a pokud editor následně její obsah změní, provede se tato změna právě v dané jazykové mutaci a uloží do databáze pod správnými identifikátory. V případě budoucího požadavku na rozšíření sady jazykových mutací pak stačí pouze upravit systémové menu aplikace tak, aby obsahovalo položku pro doplňovanou jazykovou mutaci, a do složky *translations* přidat soubor (slovník) pro nový jazyk pojmenovaný podle definované jmenné konvence.

Uživatelské úpravy stránek jsou v aplikaci realizovány s využitím komponenty TinyMCE. Ta vloží jednoduchý WYSIWYG editor přímo do upravované stránky, což následně uživateli umožní provést požadované změny případně vytvořit obsah nový. Technicky je samotná implementace provedena vložením pomocného skriptu v jQuery přímo do kódu stránky. Ten pak na stránce vyhledá textová pole (`input`) se specifickým id (nebo třídou) a ty pak upravuje (formátuje) podle uživatelem nastavených parametrů editoru (tedy např. tučné nebo kurzívní písmo,

zarovnání textu, vložení seznamu apod.). Dané textové pole je standardní součástí formuláře, data jsou standardní textový typ a po ukončení úprav se uloží do databáze ve formátu HTML stejným způsobem, jako data z jiných běžných textových polí formuláře.

Content Delivery Network (CDN) je systém umožňující využívat v aplikacích (nejen webových) obsah poskytnutý třetí stranou. Takový obsah je k dispozici na serverech na Internetu a klientům je poskytován až v okamžiku jeho použití. Komponenta TinyMCE je příkladem takového obsahu. Tím, že jsou tyto servery začleněny do CDN (a nejsou jen náhodně rozmístěny na Internetu jako samostatné entity), je umožněna optimalizace při doručování poskytovaného obsahu na místo určení výběrem topologicky nejbližšího serveru nebo serveru s momentálně nejnižším vytížením. Přínosem pro uživatele je snížení zátěže jeho vlastního serveru, který se tak uložením a poskytováním daného obsahu nemusí zabývat, a dále je zajištěna i optimalizace datového toku při doručování obsahu (např. s využitím datové komprese), vysoká dostupnost (požadovaný obsah je k dispozici z více zdojů) nebo např. automatická aktualizace.

3.3.4 Nasazení aplikace

Docker je implementační systém tvořený z několika logických částí, které obsahují definice jednotlivých komponent (kontejnerů) výsledné aplikace, způsob jejich kombinování a nastavení. Ke každé komponentě je vytvořen soubor `Dockerfile` se specifikací parametrů vlastních (gaudium-web) nebo získaných z již dříve vytvořených šablon, vzorů a jiných předpřipravených kontejnerů. Hlavním konfiguračním předpisem je soubor `docker-compose.yaml`, který z nově vytvořených či stažených šablon (image) vytvoří kontejnery a následně je spustí. V parametrech nastavení je např. i možnost definice portu, na kterém bude výsledná aplikace následně dostupná, nebo třeba jestli (a kam) se budou perzistentně ukládat její data. Já jsem svou webovou aplikaci chtěla připravit právě na možnost nasazení prostřednictvím Dockeru a vytvořit pro tento systém příslušný instalační předpis. Ten jsem potřebovala následně otestovat právě s využitím lokálně nainstalovaného Dockeru.

Pro budoucí automatickou instalaci (nasazení) mé aplikace prostřednictvím Dockeru jsem se nechala inspirovat příkladem Dockerfilu dostupného pro instalaci Symfony ([41]). Navíc jsem hledala i doplňující informace s využitím vyhledávače Google a služby chatGPT. Následně jsem vytvořila jeden vlastní image (Dockerfile) a využila dva další image, které při zpracování Dockerem vytvoří po dokončení instalace celkem čtyři kontejnery nezbytné pro spuštění výsledné aplikace:

- gaudium-web – z image gaudium-web
  – hlavní část aplikace
- gaudium-web-migration – z image gaudium-web
  – předpis pro migraci schématu databáze na aktuální verzi využívanou v aplikaci
- caddy – z image caddy:latest
  – webový server, na kterém aplikace běží; konfigurace pomocí souboru `Caddyfile`
- db – z image postgres:15
  – instance SQL databáze pro zprostředkování přístupu k datům aplikace

Součástí instalačního předpisu je i cílové nastavení dvou perzistentních úložišť – pro databázi a složku private_uploads, do které v rámci používání aplikace nahrávají uživatelé svůj vlastní obsah.

Reálné nasazení na dočasném (testovacím) serveru do doby, než výbor sboru schválí nového poskytovatele hostingu, proběhlo pomocí ssh přístupu, kdy s využitím konfiguračního balíčku připraveného v Dockeru stačilo jen soubory zkopírovat na server, ověřit přítomnost Dockeru

a pomocí příkazů `sudo docker-compose build --pull` a `sudo docker-compose up -d` uvést Docker do provozu. Posledním krokem bylo nahrání předpřipravených dat do databáze a kontrola, že vše funguje dle očekávání.

3.3.5 Zálohování aplikace

Možnost vytváření zálohy uživatelských dat přímo z prostředí aplikace není v současné verzi k dispozici, ale zálohy lze vytvářet i tak. Struktura ukládaných uživatelských dat je rozdělena pouze do tří částí

- kód samotné instalované aplikace (k dispozici na instalačních médiích)
- databáze obsahující veškerá data zadávaná uživately (správci)
- souborové úložiště používané k ukládání souborů příloh (noty, média)

Zkopírováním dat z těchto tří zdrojů s využitím nástrojů poskytovatele hostingu získá uživatel kompletní zálohu webové aplikace včetně uživatelských dat.

3.3.6 Řešené problémy

Upgrade frameworku byl jedním z problémů, se kterým jsem se během práce na vývoji aplikace setkala. Z důvodu ukončení podpory verze Symfony, se kterou jsem pracovala, jsem musela provést migraci na verzi novější (podporovanou). Tím jsem si zároveň vyzkoušela tuto proceduru pro budoucí využití v rámci příštích aktualizací.

Chybějící obsah stránky byl problém, který se objevoval, i když byla aplikace spuštěna. Tento problém vznikl při přechodu z vývojového prostředí PHPStormu na testovací produkční prostředí Caddy web serveru. Byl způsoben různými způsoby, jakým obě prostředí implementovaly úpravu (přepis) URL adres podle metody „Clean URLs“, kdy staticky ukládané soubory jsou adresovány přímo, ale URL adresy dynamicky generovaného obsahu se přepisují do čitelnějších a přehlednějších podob. Oba systémy používají jednotný vstupní bod do aplikace (single entry aplication) prostřednictvím `index.php` (to umožňuje například lepší zabezpečení), ale tím, že Caddy web server nemá implementovanou logiku pro rozlišení obou výše zmíněných typů obsahu automaticky, bylo nutné upravit `Caddyfile` tak, aby se přepis adres týkal jen dotčených odkazů.

Chybějící obsah aplikace byl problém, kdy se při pokusu o zobrazení stránky stránka nezobrazila vůbec. Důvodem bylo nedostatečné nastavení přístupových práv do složky `public`, ve které je mj. uložený i soubor `index.php` nezbytný pro elementární fungování aplikace i pro nepřihlášené uživatele.

Mizející uživatelské soubory byl problém, kdy se při restartu Dockeru smazal obsah složky s uživatelskými soubory. Řešením bylo vytvoření nového trvalého svazku (persistent volume). Při řešení tohoto problému jsem se inspirovala článkem [42].

Přizpůsobení chybových stránek zobrazovaných v reakci na mimořádné stavy aplikace byl úkol, k jehož vyřešení jsem si připravila speciální třídu `TestController`. To mi umožnilo generovat různé chybové stavy „na žádost“ a upravit jejich zobrazení v souladu s designem aplikace.

Zabezpečené stahování souborů bylo dalším úskalím při implementaci uživatelských rolí. Jádro problému spočívalo ve faktu, že v případě umístění souboru do složky s přímým přístupem ze strany webového prohlížeče klienta je přístup k takovému souboru realizován přímo prohlížečem a neprochází tedy jakýmkoli zabezpečením nastaveným v rámci aplikace. Přitom využití definovaných rolí pro nastavení přístupu přímo do jednotlivých složek web serveru nemusí být vždy podporováno (respektováno), a proto jsem jako řešení této situace zvolila úpravu chování aplikace tak, aby uživatelské soubory nebyly dostupné přímo ze složek, ale jako výstup funkce, kde se soubor získává přes query u URL adresy a ne přímo jako cesta v URL adrese.

Odstraňování jiné aktuality byla záludnost, kterou jsem zaznamenala při implementaci funkce mazání aktualit (article) ze seznamu. Smazání se uživatelsky provede kliknutím na tlačítko zobrazené u dané aktuality. Nicméně, při rutinním použití cyklů pro generování seznamů v Twigu se ke každé aktualitě přidá „stejné“ tlačítko (všechna tlačítka mají stejné id), což znemožňuje následně rozlišit, na které tlačítko (u které aktuality) uživatel klikl. Výsledkem bylo, že po stisku jakéhokoli tlačítka u jakékoli aktuality v seznamu došlo ke smazání té poslední. K vyřešení tohoto problému bylo proto nutné upravit kód generující seznamy (nejen u aktualit, ale třeba i u seznamu událostí nebo skladeb) tak, aby u každé položky v seznamu mělo příslušné tlačítko vlastní id, podle kterého by byla identifikace dané položky jednoznačná. Klíčová část řešení je zobrazená v ukázce kódu 3.2.

Výpis kódu 3.2 Renderování aktualit v šabloně Twig (výňatek)

```
{% for article in articles %}
<div class="box is-darkred p-2 mb-2">
        {{ article.getTitle() | raw }}
        <div class="columns is-mobile is-vcentered">
            {% if is_granted('ROLE_CAN_VIEW') %}
                {% if article.getAuthor() %}
                    {{ article.getAuthor().getFullName() }}
                {% else %}
                    {{ 'deleted_user'|trans }}
                {% endif %}
            {% endif %}
            <span class="is-pulled-right m-1 has-text-white">
                {{ article.getLastEditedString()}}</span>
            {% if is_granted('ROLE_ARTICLES') %}
                {% if article.getAuthor() is same as (user) %}
                    <button class="button" id="edit-article" type="button"
                    onclick="location.href='{{ path('editArticle',
                        {'id': article.getId}) }}'">
                        {{ 'edit' | trans }}
                    </button>
                {% endif %}
                <button class="js-modal-trigger button is-danger"
                    data-target="modal-js-remove-{{ article.getId }}">
                    {{ 'delete' | trans }}
                </button>
            {% endif %}
        </div>
</div>
<div class="box has-background-link-light">
    {{ article.getContent() | raw }}
</div>

<div id="modal-js-remove-{{ article.getId }}" class="modal">
```

```
    <div class="modal-background"></div>
    <div class="modal-card">
        <header class="modal-card-head">
            <p class="modal-card-title">{{ 'delete_article' | trans }}</p>
            <button class="delete" aria-label="close"></button>
        </header>
        <footer class="modal-card-foot">
            <button class="button is-danger"
                onclick="location.href='{{ path('deleteArticle',
                    {'id': article.getId}) }}'">
                {{ 'delete' | trans }}
            </button>
            <button class="button">{{ 'cancel' | trans }}</button>
        </footer>
    </div>
</div>
{% endfor %}

<script> // handling confirmation window
    $(document).ready(function() {
        // functions to open and close a modal
        function openModal($el) {
            $el.addClass('is-active');
        }
        function closeModal($el) {
            $el.removeClass('is-active');
        }

        // add a click event on buttons to open a specific modal
        $('.js-modal-trigger').each(function() {
            var $trigger = $(this);
            var modal = $trigger.data('target');
            var $target = $('#' + modal);

            $trigger.click(function() {
                openModal($target);
            });
        });
    });
</script>
```

Kapitola 4
Testování

Následující kapitola obsahuje výběr a definici scénářů pro otestování funkčnosti aplikace a dále pojednává i o testování uživatelské přívětivosti jejího použití běžnými uživateli včetně těch, kteří nedisponují jakoukoli IT odborností.

4.1 Průběžné testy

Základní testovací scénář je v zásadě velmi prostý. Je to scénář používaný pro testy základních funkcionalit ještě během vývoje samotné aplikace. Prakticky každou funkcionalitu je dobré otestovat co nejdříve po jejím nasazení, protože tento postup usnadňuje detekci potenciálních chyb a umožňuje jejich včasnou identifikaci a eliminaci, což brání jejich kumulaci a nutnosti náročných oprav po implementaci rozsáhlejších celků.

Jedním z prvních testů, které jsem provedla hned v první fázi vývoje aplikace byl test zobrazení prázdné (úvodní) webové stránky, kterou jsem vytvořila jako základ mé budoucí webové aplikace ihned po nainstalování Symfony.

Následovaly další jednoduché testy, které např. ověřovaly, zda formuláře ukládají zadaná data do připojené databáze. K těmto testům jsem využívala PHPStorm (IDE), který jsem použila jako hlavní vývojové prostředí v průběhu celého vývoje mé aplikace. Toto prostředí integruje mnoho užitečných nástrojů včetně např. nástroje pro inspekci obsahu připojené databáze, což bylo využito právě v případě těchto typů testů. Dalším velmi užitečným nástrojem dostupným jako součást prostředí PHPStorm je integrovaný PHP built-in web server, čehož jsem s výhodou využívala pro podrobnější ladění psaného kódu, protože v případě použití tohoto nástroje odpadá nutnost interakce s externím hostingovým prostředím a eliminují se tak možné chyby způsobené nikoli samotným kódem, ale např. chybou právě na straně poskytovatele této služby nebo chybou vzniklou během datové komunikace s ním. Nedocenitelným pomocníkem při práci s tímto integrovaným web serverem je pak přítomnost debuggeru, který poskytuje jak samotné krokování spuštěného kódu, tak i nastavení breakpointů pro ladění kódů reagujících na události, a z tohoto důvodu velmi obtížně krokovatelné přímým spuštěním. Neopominutelnou pomůckou je i možnost nahlížet během krokování do obsahu proměnných, a to i do složitějších struktur.

Výše uvedeným postupem jsem otestovala všechny postupně přidávané funkcionality včetně vzhledu jednotlivých stránek prezentujících zadávaná nebo již uložená data. Jednalo se zejména o tyto scénáře:

- vytvoření základní sady stránek

- vytvoření základní navigační lišty (menu)
- registrace uživatele
- zobrazení uživatelů
- úprava profilu uživatele
- přidání skladby
- přidání přílohy ke skladbě
- přidání události
- registrace účastníka události
- aktualizace záznamu o účasti na události
- přidání článku (article)
- úprava obsahu webové sttránky
- úprava přístupových práv uživatele (změna role)
- předchozí scénáře s použitím uživatelů s různými rolemi

4.2 Průběžné testy s uživateli

Již v průběhu vývoje aplikace jsem se snažila získat alespoň základní zpětnou vazbu od budoucích uživatelů aplikace, a proto jsem domlouvala jejich účast na testech ucelenějších částí vyvíjené aplikace prakticky po každém významnějším pokroku. V rámci zajištění co možná nejobjektivnější zpětné vazby jsem do testování zapojovala uživatele z různých věkových kategorií i na různých úrovních orientace ve světě IT. Některé ze získaných připomínek jsem použila pro následnou úpravu, některé vyhodnotila jako nepodstatné a odložila a některé začlenila do seznamu plánovaných vylepšení v budoucnu v rámci běžného užívání výsledné aplikace a jejích úprav jako součást jejího životního cyklu.

Příkladem zapracované zpětné vazby byl postřeh jednoho z testerů, který poukázal na zcela odlišný vhled a chování formulářů na registraci nového uživatele a přihlášení stávajícího. Nově jsou tyto formuláře sjednocené.

Příkladem zapracované připomínky k odhalené chybě je úprava vstupního řádku pro zadání hesla, kdy v původní implementaci se po zadání hesla a stisknutí klávesy `Enter` zobrazilo zadané heslo namísto odeslání formuláře k vyhodnocení požadavku. Ukázalo se, že příčinou byl aktivní prvek („očičko“) přítomný ve formuláři, který na kliknutí zobrazoval zadané heslo a který se aktivoval také právě po stisku klávesy `Enter` namísto tlačítka `přihlásit se` pro odeslání formuláře. Nově je již chování formuláře dle očekávání.

Příkladem zapracované připomínky, která měla za následek i rozšíření původního seznamu funkčních požadavků, je přání mít možnost zadávat i soukromé události, tedy takové, které se nebudou automaticky zobrazovat ve veřejné sekci.

Nezapracovanými připomínkami pak jsou např. návrh na přidání možnosti specifikovat rozdělení do hlasů u každé skladby, anebo doplnění více jazykových mutací (to systém umožňuje již v základu, ale další jazykovou mutaci samotnou by musel někdo vytvořit).

Příkladem připomínek přidaných do plánu budoucích úprav (vylepšení) aplikace je např. možnost definice pořadí skladby v detailu události anebo možnost automatického zasílání informačního e-mailu v případě úpravy registrace na událost v termínu kratším než týden před konáním události nebo v případě zveřejnění nové nebo úpravy existující zprávy (aktuality).

4.3 Závěrečné testy s uživateli

Testování prototypu webové aplikace probíhalo ve dvou základních režimech.

4.3.1 Testovací scénář

Někteří uživatelé obdrželi před provedením samotného testování obecný testovací scénář, který měli následovat. Tento scénář nebyl sestaven z přímých pokynů typu „klikni sem“ nebo „klikni tam“, ale z obecných úkolů, které specifikovaly informace, které měl tester z webové aplikace získat, anebo úlohy, které měl splnit. Pro účely provedení a vyhodnocení těchto testů jsem si vyžádala výslovný souhlas s pořízením audio/video záznamů od jednotlivých testerů a s jejich zveřejněním jako součástmi dokumentace mé webové aplikace. Některé takto provedené testy jsou tedy zaznamenány a uloženy na přiloženém médiu. Záznamy obsahují jak záznam obrazovky (tedy, co uživatel při testu dělal), tak některé i nahrávku tváře uživatele, tedy výrazu a reakcí na prováděné operace. To hraje vcelku významnou roli při vyhodnocování výsledků testů, protože to umožňuje nejen vyhodnotit splnění zadaného úkolu, ale i posoudit uživatelskou zkušenost a celkovou úroveň spokojenosti při použití webové aplikace.

Obsah testovacího scénáře pro běžného návštěvníka

1. Zaujal tě tento sbor a chceš jít na nejbližší možný koncert.
2. Podívej se na pozvánku a program na tento koncert.
3. Bohužel se ti tento termín nehodí, pokus se najít nějaké další možnosti.
4. Sbor tě zaujal a chceš se o něm dozvědět více.
5. Chceš se dozvědět něco víc o sbormistrech.
6. Rád/a bys sboru něco sdělil/a. Pokus se najít nějaký způsob.
7. Zajímá tě, jakých akcí se sbor nedávno účastnil.
8. Pro pokročilé: Máš anglicky mluvícího kamaráda a chceš mu sbor představit.

Test 19.11.2023 provedl externí návštěvník webu, věk 20-30 let, IT specialista. Test probíhal interaktivně dle mých instrukcí předávaných přímo testerovi.

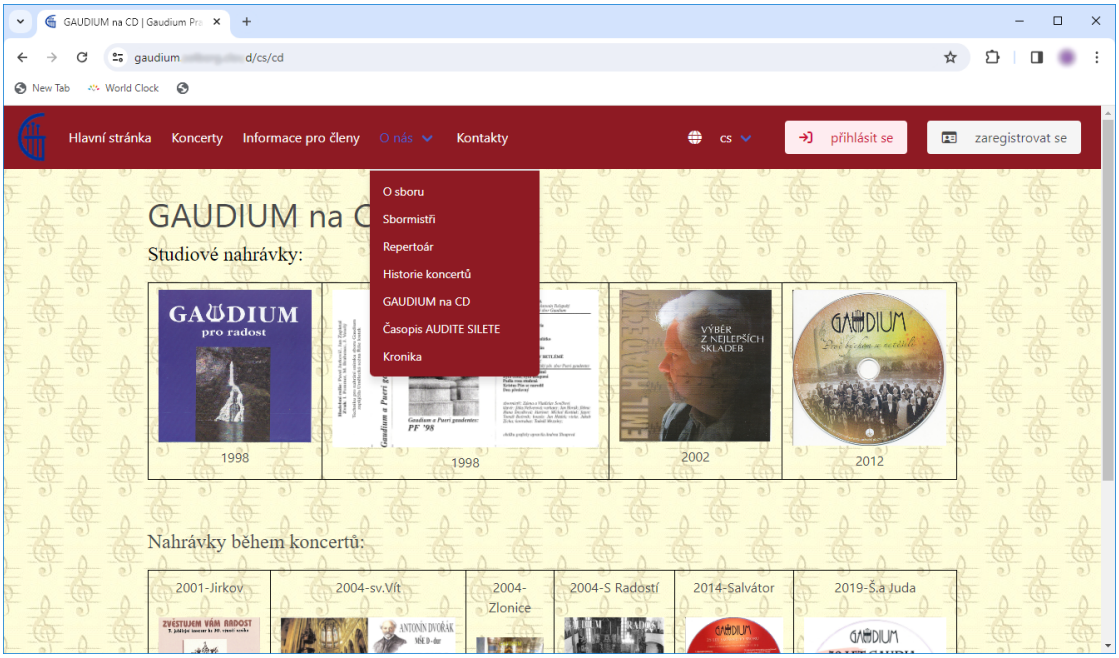
Výsledkem testu je zjištění, že základní schéma webové aplikace je přehledné, orientace intuitivní a relativně snadná, avšak srozumitelnost detailů u některých zobrazovaných informací není úplná. Například identifikace místa konání koncertu nebyla jednoznačná, ale to bylo zapříčiněno zápisem informace o koncertu, nikoli chováním aplikace samotné. Dle výrazu a reakcí testera v průběhu testu a dle adekvátnosti reakcí na podávané instrukce hodnotím celkovou uživatelskou zkušenost jako výbornou.

V reakci na drobné zaváhání testera při požadavku na vyhledání informací o dalších koncertech nezobrazených na úvodní stránce bylo ve finální verzi webové aplikace k sekci s informací o nejbližší událostí přidáno tlačítko s odkazem na seznam všech událostí (koncertů).

Test 2.1.2024 provedl externí návštěvník webu, věk 70-80 let, bez IT odbornosti. Test probíhal interaktivně dle mých instrukcí předávaných přímo testerovi.

Výsledkem testu je zjištění, že základní schéma webové aplikace je přehledné, orientace intuitivní a relativně snadná, avšak potíže s obsluhou polohovacího zařízení (nebyla k dispozici myš, ale jen touchpad) zpomalily interakci testera s webovou aplikací. Díky drobné úpravě formátu a obsahu zobrazovaných informací od verze použité při předchozím testu bylo získání detailu události snazší. Dle výrazu a reakcí testera v průběhu testu a dle adekvátnosti reakcí na podávané instrukce hodnotím celkovou uživatelskou zkušenost jako velmi dobrou.

V reakci na zaváhání testera při požadavku na vyhledání informací o sboru bylo ve finální verzi webové aplikace upraveno rozbalovací menu v sekci „O nás“, kdy v původním návrhu byla další položka „O nás“ součástí podrobnější (rozbalené) nabídky. To působilo zmateně, protože nebylo zřejmé, že je třeba podruhé kliknout sice na jinou, ale stejně pojmenovanou, položku v menu. Po úpravě se tato následná položka již nejmenuje znovu „O nás“, ale „O sboru“, jak ilustruje snímek 4.1 z výsledné aplikace.



Obrázek 4.1 Zobrazení stránky se sborovými CD běžným uživatelem

Obsah testovacího scénáře pro člena sboru

1. Zaregistruj se do aplikace.
2. Přihlaš se do aplikace.
3. Podívej se, zda můžeš do interní sekce.
   (v této chvíli povýšit na člena)
4. Podívej se na svůj profil.
5. Vyplň svou účast na koncertě.
6. Podívej se na bližší info o koncertu.
7. Zjistil jsi, že se nemůžeš zúčastnit, změň svou účast.
8. Získej verzi not na Adeste Fideles.
9. Pro pokročilé: Máš anglicky mluvícího kamaráda, který by se rád stal členem sboru. Představ mu interní sekci v angličtině.

Test 9.1.2024 provedl člen sboru, věk 20-30 let, běžný uživatel IT. Test probíhal interaktivně dle mých instrukcí předávaných přímo testerovi.

Výsledkem testu je zjištění, že základní schéma webové aplikace je přehledné, orientace intuitivní a relativně snadná. Tester mylně vyhodnotil chování aplikace, kdy se domníval, že již

došlo k jeho odhlášení, což se nestalo. Další výtka zazněla v souvislosti se seznamem událostí (koncertů), protože ten je zadáván vedením sboru pouze v jedné jazykové mutaci, a proto např. názvy koncertů nejsou přeloženy. Tento test nebyl doplněn záznamem tváře testera, nicméně z jeho reakcí a intonace hlasu v průběhu testu a dle adekvátnosti reakcí na podávané instrukce hodnotím celkovou uživatelskou zkušenost jako velmi dobrou.

V reakci na výhrady testera k použité grafice, zejména k barevnému schématu a výsledné čitelnosti některých zobrazených navigačních prvků, bylo ve finální verzi aplikace barevné schéma upraveno. Výskyt několika drobností souvisejících s jazykovými mutacemi, kdy některé ovládací prvky a chybové stavy nejsou důsledně převedeny do alternativního jazyka, bude předmětem budoucích úprav.

Obsah testovacího scénáře pro správce

1. Přihlaš se.
2. V sekci repertoár něco drobně uprav.
3. Schval budoucího člena.
4. Zobraz si seznam členů.
5. Druhý ze členů byl přeřazen do jiného hlasu, udělej to za něj.
6. Přidej novou událost.
7. Napiš zprávu pro sbor.
8. Odhlaš se.

Test 10.1.2024 provedl současný člen sboru v roli správce, věk 20-30 let, IT specialista. Test probíhal interaktivně dle mých instrukcí předávaných přímo testerovi.

Výsledkem testu je zjištění, že základní schéma webové aplikace je přehledné, orientace intuitivní a relativně snadná. Nicméně, vzhledem k tomu, že možnosti správce jsou obsáhlejší, než možnosti člena sboru nebo externího návštěvníka stránek, a dostupných aktivit je pro takovou roli výrazně více, nebyly některé instrukce zprvu pochopeny dle záměru scénáře testu. Po upřesnění instrukcí už byla akce provedena správně, i když s drobným zaváháním při hledání správného ovládacího prvku pro provedení požadované operace. To je dané zejména tím, že možností designu je mnoho a není možné implementovat všechny současně. V důsledku toho pak intuice neposkytuje jednoznačné vodítko ve všech případech. To je však běžná situace i u komerčních aplikací, protože i tam je mnohdy nezbytné postupovat podle přiloženého návodu.

Dle výrazu a reakcí testera v průběhu testu a dle adekvátnosti reakcí na podávané instrukce hodnotím celkovou uživatelskou zkušenost jako velmi dobrou.

4.3.2 Volný styl

Někteří uživatelé obdrželi před provedením samotného testování jen velmi vágní instrukci, aby „se na web podívali a zkusili s ním pracovat“. Cílem bylo zjistit, jak je webová aplikace celkově zaujala, jak se v navigaci po ní pohybují, jaké informace vyhledávají, případně, do kterých sekcí se vracejí.

Vzhledem k velmi omezenému rozsahu obsahu webové aplikace během testu (prakticky jen demo verze) nebylo možné plnohodnotně vyhodnotit zaujetí testerů obsahem samotným. Nicméně všichni hodnotili orientace v aplikaci jako intuitivní a relativně snadnou, což bylo v souladu s výsledky testů prováděných podle jednotlivých scénářů.

Kapitola 5
Závěr

Zhodnocení splnění cílů a nástin možností dalšího vylepšení a rozšíření.

5.1 Splnění cílů

Hlavní cíle bakalářské práce byly splněny. Mým hlavním záměrem bylo zejména poskytnout webovou aplikaci, která bude přinášet funkcionalitu požadovanou zadavatelem, avšak nedostupnou v současné verzi webových stránek, a napravovat hlavní nedostatky současné verze. Přestože aplikace ještě není nasazená do rutinního provozu z důvodu absence finálního rozhodnutí zadavatele o výběru poskytovatele hostingu, demoverze aplikace je umístěná u provizorního poskytovatele a je k dispozici na vyzkoušení. Tato implementace byla použita i pro provedení některých závěrečných testů.

5.2 Možnosti rozšíření

Přestože aplikace splnila všechny funkční požadavky zadavatele, během jejího vývoje i následných závěrečných testů došlo k identifikaci některých funkcí, které by bylo užitečné doplnit, ale které už nebylo možné z časových důvodů zapracovat do verze současné. Mezi plánovaná rozšíření patří přidání možností

- definovat pořadí skladeb na koncertě
- rozšířit status v záznamu účasti na akci o „nezobrazeno“
- implementovat více možností pro uživatelský překlad, například aktualit
- dočasně pozastavit členství
  (např. z důvodu mateřské dovolené, zahraničního studia, ...)
- automaticky zasílat emaily v případě změny záznamu o účasti na akci nebo jejího nezobrazení méně než týden před jejím konáním
- automaticky zasílat emaily v případě přidání aktuality nebo změny některé existující
- zaznamenat datum vytvoření některých entit případně i autora provedené změny
  (nejen autora nové)
- definovat pravidelně se opakující akce (např. zkoušky)
- dědit vlastnosti předchozí události (např. seznam skladeb nebo místo konání)
- vyhledávat ve skladbách
- vytvářet skupiny skladeb (např. vánoční, velikonoční, ...)
- vytvářet fotogalerie

- zálohovat data aplikace přímo z aplikace
  (v současné verzi jen s využitím nástrojů poskytovatele hostingu)

5.3 Shrnutí

Práce na webové aplikaci mi přinesla mnoho cenných a užitečných zkušeností. Nejvíce pozitivně vnímám zejména fakt, že se nejednalo o čistě laboratorní akademický projekt, ale o práci, která po finální implementaci do cílového živého prostředí přinese užitek desítkám lidí, členům sboru, a dalším stovkám lidí, kteří jejich webové stránky prezentované mou webovou aplikací v budoucnu navštíví. Neopominutelnou pozitivní zkušeností byla i práce s lidmi, kteří mi pomáhali s testováním aplikace jak už v průběhu jejího vývoje, tak při závěrečných testech. Jejich podněty a připomínky pramenící ze zájmu o vyvíjený produkt a mnohdy i radost z přípravy tohoto nástroje, který jim ulehčí a zpříjemní vedení agendy sboru, byly jednoznačně povzbuzující.

Příloha A

Návod na spuštění a používání aplikace

A.1 Návod na spuštění aplikace

1. Stáhněte a nainstalujte si Docker[1] a spusťte ho.
2. Otevřete příkazovou řádku v adresáři projektu (adresář *app*).
3. Proveďte příkazy `sudo docker-compose build --pull` a `sudo docker-compose up -d`, které rozběhnou dockerovský kontejner.
4. Nahrajte data do databáze: příkazem `sudo docker exec -it gaudium-db /bin/bash` se dostanete dovnitř kontejneru a pomocí `psql -U pguser -d public < /data/data.sql` naplňte databázi testovacími daty.
5. Otevřete webový prohlížeč a na localhostu[2] naleznete aplikaci.

A.2 Návod k použití

Při vývoji a implementaci aplikace byl kladen důraz na intuitivnost a jednoduchost, popisky v aplikaci by tedy měly být samovysvětlující a návodné.
Uživatel se musí nejprve zaregistrovat. Po registraci se již může uživatel přihlašovat do aplikace pomocí tlačítka *přihlásit se*. Po registraci je uživatel pouze obecným uživatelem, další možnosti jsou přístupné až po udělení vyšší role – např. člen. Role může přidávat administrátor, sbormistr či uživatel s rolí výbor.
Člen pak může například:

- zobrazit podrobnější informace k plánovaným akcím (jako např. čas srazu),
- přihlásit se na událost,
- zobrazit si přehled skladeb a jejich případných příloh (např. noty, MIDI nahrávka),
- zobrazit si soukromá sdělení od vedení sboru členům,
- vyplnit svůj profil s kontaktními údaji.

[1] `https://www.docker.com/products/docker-desktop/`
[2] `http://localhost:80/`

Bibliografie

1. *Gaudium Praha* [online]. [cit. 2023-01-20]. Dostupné z: http://gaudiumpraha.org/.
2. *Responzivní web: jak vypadá a proč ho mít* [online]. Active24 [cit. 2023-02-26]. Dostupné z: https://www.active24.cz/jak-na-tvorbu-webu/tvorba-stranek-pokrocila/responzivni-web-jak-vypada-a-proc-ho-mit.
3. KUBÍK, Milan. *Co znamená responzivní web a proč ho mít* [online]. 2021. [cit. 2023-02-26]. Dostupné z: https://www.webnia.cz/deje-se/co-znamena-responzivni-web-a-proc-ho-mit.
4. MARCOTTE, Ethan. *Responsive Web Design* [online]. 2010. [cit. 2023-02-26]. Dostupné z: https://alistapart.com/article/responsive-web-design/.
5. O'HARE, Sean. *Reasons Why a Multi-Language Site Benefits Your Business* [online]. 2023. [cit. 2023-04-20]. Dostupné z: https://weglot.com/blog/reasons-why-a-multi-language-site-benefits-your-business/.
6. (CISO), Chief Information Security Officer. *Shared Accounts* [online]. University of Washington [cit. 2023-01-20]. Dostupné z: https://ciso.uw.edu/education/risk-advisories/shared-accounts/.
7. GOOGLE. *Google Sheets - create and edit spreadsheets online* [online]. 2023. [cit. 2023-11-13]. Dostupné z: https://www.google.com/intl/cs/sheets/about/.
8. *Limitations when you save a Word document as a web page* [online]. Microsoft Corporation, Microsoft [cit. 2023-01-20]. Dostupné z: https://support.microsoft.com/en-us/topic/limitations-when-you-save-a-word-document-as-a-web-page-f361de08-ca4c-bc53-11ef-138c0e405c44.
9. *Sbor Notre Dame* [online]. 2012. [cit. 2023-01-20]. Dostupné z: http://www.sbornotredame.cz/.
10. PULKITAGARWAL03PULKIT. *Difference between JSP and HTML* [online]. GeeksforGeeks [cit. 2023-01-20]. Dostupné z: https://www.geeksforgeeks.org/difference-between-jsp-and-html/.
11. *Pueri Gaudentes* [online]. [cit. 2023-01-20]. Dostupné z: https://www.puerigaudentes.cz/uvod.htm.
12. *Zákon č. 99/2019 Sb., o zpracování osobních údajů a o změně souvisejících zákonů* [online]. Zakonyprolidi.cz, 2019 [cit. 2023-01-20]. Dostupné z: https://www.zakonyprolidi.cz/cs/2019-99.
13. PAVLÍČEK, Radek. *Web Content Accessibility Guidelines (WCAG): seznamte se, prosím* [online]. Poslepu.cz, 2019-02 [cit. 2023-01-20]. Dostupné z: https://poslepu.cz/web-content-accessibility-guidelines-wcag-seznamte-se-prosim/.

14. *EOS Homepage* [online]. [cit. 2023-09-26]. Dostupné z: https://www.eos.cz/.
15. *Pražská kantiléna* [online]. 2023. [cit. 2023-01-20]. Dostupné z: https://prazskakantilena.cz/.
16. NEWCOMER, Colin. WordPress Review. *ThemeIsle Blog* [online]. 2023 [cit. 2023-01-20]. Dostupné z: https://themeisle.com/blog/wordpress-review/#gref.
17. GORBACHENKO, Pavel. *Functional Requirements vs. Non-Functional Requirements: What's the Difference?* [online]. Enkonix [cit. 2023-01-24]. Dostupné z: https://enkonix.com/blog/functional-requirements-vs-non-functional/.
18. MORRISON, Roger. *Rozdíl mezi funkčními a nefunkčními požadavky* [online]. strephonsays [cit. 2023-01-24]. Dostupné z: https://cs.strephonsays.com/functional-and-non-functional-requirements-3325.
19. *Easy Webhosting* [online]. FORPSI. [cit. 2023-10-12]. Dostupné z: https://www.forpsi.com/webhosting/easy/.
20. SUEHRING, Steve; VALADE, Janet. *PHP, MySQL, JavaScript & HTML5 all-in-one for dummies*. Somerset: Wiley, 2013. For dummies. ISBN 9781118213704.
21. PORĘBSKI, Bartosz. *Building applications with Symfony, CakePHP, and Zend Frameworks*. 1st edition. Indianapolis, IN: Wiley, 2011. Wrox programmer to programmer Building PHP applications with Symfony, CakePHP, and Zend Framework. ISBN 1-283-37455-2.
22. ARMAND, Sébastien. *Extending symfony2 web application framework : optimize, audit, and customize web applications with symfony*. Birmingham, England: Packt Publishing Ltd, 2014. Community experience distilled. ISBN 1-78328-720-9.
23. *Savana Webhosting* [online]. [cit. 2023-09-28]. Dostupné z: https://www.savana.cz/webhosting.
24. *TreeFrog Framework* [online]. [cit. 2023-08-12]. Dostupné z: https://www.treefrogframework.org/.
25. M., Will. *Best PHP Frameworks in 2023: Which One Should You Choose?* [online]. Hostinger, 2023. [cit. 2023-02-22]. Dostupné z: https://www.hostinger.in/tutorials/best-php-framework.
26. SVIRCA, Zanfina. *Everything You Need to Know About MVC Architecture* [online]. Towards Data Science, 2020. [cit. 2023-02-22]. Dostupné z: https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1.
27. *Bulma: A Modern CSS Framework*. Bulma. Dostupné také z: https://bulma.io/.
28. *Symfony*. Symfony. Dostupné také z: https://symfony.com/.
29. *Contributing to Symfony - Code License* [online]. [cit. 2024-01-06]. Dostupné z: https://symfony.com/doc/current/contributing/code/license.html.
30. *TinyMCE* [online]. [cit. 2023-11-12]. Dostupné z: https://www.tiny.cloud/.
31. *Selectize* [online]. [cit. 2023-12-12]. Dostupné z: https://selectize.dev/.
32. *Twig – The flexible, fast, and secure template engine for PHP*. Symfony. Dostupné také z: https://twig.symfony.com/.
33. *Font Awesome v4* [online]. [cit. 2023-09-10]. Dostupné z: https://fontawesome.com/v4/.
34. *draw.io* [online]. [cit. 2023-08-28]. Dostupné z: https://www.drawio.com/.
35. *Balsamiq* [online]. [cit. 2023-09-29]. Dostupné z: https://balsamiq.com/.
36. *PhpStorm: The Lightning-Smart PHP IDE* [online]. [cit. 2024-01-08]. Dostupné z: https://www.jetbrains.com/phpstorm/.
37. *Docker Documentation* [online]. [cit. 2023-09-18]. Dostupné z: https://docs.docker.com/.

38. *GitHub* [online]. [cit. 2024-01-05]. Dostupné z: `https://github.com/`.

39. *GitLab na Českém vysokém učení technickém v Praze* [online]. [cit. 2024-01-05]. Dostupné z: `https://gitlab.fit.cvut.cz/`.

40. *Git* [online]. [cit. 2024-01-05]. Dostupné z: `https://git-scm.com/`.

41. DUNGLAS. *Dockerfile in symfony-docker repository* [online]. 2023. [cit. 2023-09-18]. Dostupné z: `https://github.com/dunglas/symfony-docker/blob/main/Dockerfile`.

42. TEAM, Better Stack. *How to preserve data when Docker container exits* [online]. [cit. 2023-11-20]. Dostupné z: `https://betterstack.com/community/questions/how-to-preserve-data-when-docker-container-exits/`.

Obsah přiloženého zip souboru

- `README.md` ... stručné pokyny ke spuštění aplikace
- `license.txt` ... informace o licenci
- `vlkova.pdf` ... text práce ve formátu PDF
- `app` ... adresář se soubory aplikace
  - `bin`
  - `config` ... adresář s konfiguračními soubory
  - `data` ... adresář s připravenými testovacími daty pro nahrání do databáze
  - `migrations` ... adresář s databázovými migracemi
  - `private_uploads` ... adresář s (testovacími) uživatelskými soubory
  - `public`
    - `style` ... adresář s CSS soubory
    - `images` ... adresář s obrázky použitými v aplikaci
    - `favicon.ico` ... ikonka zobrazená v prohlížeči
    - `index.php` ... vstupní bod aplikace
  - `src` ... adresář se zdrojovými kódy
  - `templates` ... adresář s Twig šablonami stránek
  - `translations` ... adresář se soubory používanými pro jazykové přizpůsobení
  - `Caddyfile` ... konfigurační soubor pro Caddy server
  - `composer.json` ... soubor s definicemi závislostí a konfigurací pro Composer
  - `composer.lock` .. soubor obsahující seznam přesných verzí nainstalovaných závislostí pro Composer
  - `docker-compose.yml` ... soubor pro konfiguraci a orchestraci Docker kontejnerů
  - `Dockerfile` ... soubor pro definici Docker image (obrazu)
  - `symfony.lock` ..... soubor obsahující seznam verzí nainstalovaných balíčků pro Symfony
  - `.env` ... soubor s prostředím pro běh aplikace
- `prilohy` ... adresář s dalšími přílohami
  - `testy` ... adresář s video záznamy testů
    - `testovaci_scenar.md` ... testovací scénář
  - `wireframes` ... adresář s návrhy GUI aplikace