ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro správu elektronických verzí literárních děl
Student:	Martin Melichar
Vedoucí:	Ing. Karel Klouda, Ph.D.
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vytvořit systém pro správu a snadné vytváření elektronických verzí literárních děl.
1. Seznamte se se standardními formáty (zejm. TEI) používanými pro uchovávání elektronických verzí literárních děl.
2. Pro vybraný formát vytvořte webový redakční systém splňující požadavky, které sesbíráte a dohodnete s pracovníky UČL AV.
   a) Redakční systém musí umožňovat správu dokumentů (tj. souborů ve formátu XML) a přidružených souborů (zejména fotografií).
   b) Systém musí také umožňovat snadné vyhledávání a filtrování v seznamu sbírek.
   c) V systému by měla být možná autentifikace a základní správa uživatelů a jejich rolí.
3. Všechny části systému řádně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. října 2017

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

Bakalářská práce

Systém pro správu elektronických verzí literárních děl

Martin Melichar

Vedoucí práce: Ing. Karel Klouda, Ph.D.

12. května 2018

Poděkování

V prvé řadě bych rád poděkoval panu Ing. Karlu Kloudovi, Ph.D. za pomoc, trpělivost a odborné rady v průběhu psaní této bakalářské práce a za možnost podílet se na reálném projektu, který se bude pravděpodobně v praxi používat. Dále bych chtěl poděkovat své přítelkyni a rodině za trpělivost a podporu v průběhu studia na ČVUT.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona 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, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2018 .....

Odkaz na tuto práci

Melichar, Martin. *Systém pro správu elektronických verzí literárních děl.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem této bakalářské práce je ulehčit převod literárních děl do elektronické podoby. Vytvořené řešení dovoluje upravovat, mazat nebo přidávat dílčí součásti knihy např. autora díla a také spravovat přidružené přílohy k jednotlivým dílům. Systém je napsán v PHP pomocí microframeworku Slim. Hlavním výsledkem je zrychlení vytváření a snadné upravování elektronických verzí literárních děl.

Klíčová slova webová aplikace, redakční systém, návrh a implementace, správa eletronických literárních děl, Slim, PHP

Abstract

The aim of this thesis is to facilitate the conversion of literary works into electronic form.

Keywords web application, management system, design and implementation, digitalized literary works, Slim, PHP

Obsah

Seznam obrázků

Úvod

Elektronická literární díla stále rozšiřují pole své působnosti, ať už mluvíme o nakupování nebo zobrazování knih na počítači či o jejich snadném čtení v e-čtečkách. V Austrálii se rozdíl prodeje elektronických knížek v roce 2008 oproti roku 2009 rapidně zvýšil, a to o více než 100 %. Ve Spojených státech amerických se v lednu roku 2012 zvýšil prodej e-knih pro dospělé o 49,4 % a e-knih pro děti a mládež o 475,1 %, ve srovnáním s lednem 2011 [1]. Z prudkého nárůstu prodejů e-knih můžeme usoudit, že rapidně roste oblíbenost elektronických děl. Zobrazení a následné čtení e-knih na našich elektronických zařízení je pro mnohé nejenom snazší, ale i pohodlnější oproti zapůjčování či koupi papírových knih.

Výsledek této práce je přednostně určen pro zaměstnance Ústavu české literatury Akademie věd České republiky (UČL AV). Pracovníci ústavu získávají, ať formou darů či koupí, postupně více a více literárních děl. Tato díla následně naskenují a převedou do elektronické podoby. Nicméně často to bývají historická díla, a proto se stává, že výsledek přesně neodpovídá tištěné podobě knihy. Proto je nezbytné, aby pracovníci ústavu ručně opravovali chyby a měli možnost doplňovat chybějící části v elektronické podobě díla. Zde přichází na řadu aplikace, která bude umožňovat rychlejší a efektivnější úpravu elektronicé podoby díla.

Cíl práce

Prvním cílem je rešerše existujících aplikací pro tvorbu a správu literárních děl a dále porovnání frameworků a výběr toho nejvhodnějšího pro tvorbu aplikace. S tím úzce souvisí výběr samotného jazyka pro implementaci. Poté přijde na řadu výběr nejvhodnějšího textového formátu pro e-knihy. Dalším krokem je shromáždění poznámek a připomínek od pracovníků pro vytvoření samotné aplikace.

Cílem praktické části práce je navrhnout a implementovat redakční systém

pro správu a snadné vytváření elektronických verzí literárních děl, otestovat systém na reálných datech a uživatelích a řádně jej zdokumentovat. Systém bude umožňovat správu nejen samotných dokumentů, ale i souborů k nim přidruženým, zejména fotografií a obrázků. Ve výsledné aplikaci bude implementován jednoduchý filtr a snadné vyhledávání v seznamu sbírek. Aplikace bude umožňovat autentifikaci a bude zároveň poskytovat správu uživatelů a jejich rolí. V aplikaci bude administrátorovi umožněna registrace nového uživatele, který si následně upraví heslo podle své potřeby.

Ukázka současného stavu

V současnosti mají pracovníci UČL AV okolo 1700 literárních děl uložených v databázi ve formátu XML. Díla jsou rozdělena do dvou částí, jak je vidět na příkladu díla „Vytržené listy“ na obrázku A.2. První část je identifikována tagem hlavička a druhá tagem text. Hlavička obsahuje metadata jako je titul díla nebo rok vydání a ve druhé části je samotný text rozčleněn například do tagů sbírka a báseň.

K některým dílům mají pracovnící k dipozici jejich oskenované stránky. Tyto skeny jsou uložené ve formátu JPG a slouží výhradně ke kontrole převodu elektronické podoby. Na obrázku A.8 je vidět srovnání skenu a xml kódu díla „Cikánčina smrt“ strana 9. Stránka je podrobně rozdělena do nestandardních tagů strofa a v.

Kapitola 1

Analýza

1.1 Existující řešení

Před návrhem a implementací aplikace bylo potřeba řádně prozkoumat existující řešení problému, technologie k tvorbě a elektronické formáty, které se užívají pro uchování elektronických děl. Vzhledem k roztoucí poptávce po elektronických dílech na úkor papírových se podle očekávání objevilo mnoho aplikací pro editaci těchto děl. Dle pracovníku UČL AV je výběr omezen na aplikace pracující s formátem XML.

Následují informace o pěti editorech, které uvádí server [2]:

- Emeditor,
- Editix,
- EditPad Pro,
- Essential XML Editor,
- Exchanger XML Editor.

1.1.1 Emeditor

Emeditor[1] patří podle [2] mezi nejlepší XML editory. Tento software je hlavním produktem americké firmy Emurasoft, Inc. sídlící v Redmondu ve Washingtonu. Firma se nadále stará o podporu i vývoj. Nicméně autorem editoru je Yutaka Emura. Samotná aplikace již vyhrála 24 mezinárodních cen v kategoriích nejlepší webový nástroj nebo nejlepší aplikace roku 2008.

Výhody:

- poslední release v17.5.0 vyšla 27. února 2018, aplikace je tedy pravidelně aktualizována,

[1] domovská stránka: `https://www.emeditor.com/`

- podpora velkých souborů,
- použití více jader při větší zátěži,
- kódování UTF-8,
- konfigurovatelná kontrola pravopisu.

Nevýhody:

- aplikace je placená, ale nabízí trial verzi na 30 dní,
- existuje free verze, nicméně v ní chybí zásadní funkce,
- editor je pouze pro Windows,
- chybí přehledný průvodce základních funkcí po prvním spuštění,
- aplikace funguje jako editor a proto není možná, aby více uživatelů současně pracovali se stejnými daty.

1.1.2 Editix

Editix[2] je produktem francouzské společnosti JAPISoft SARL. Editor vytvořil Alexandre Brillant. Systém je velice přehledný a intuitivní. Na trhu je systém od roku 2004 a nejnovější verze je EditiX XML Editor 2017 v15. Zákazníci, kteří využívají tento software, jsou převážně vzdělávací instituty od University of Oxford po University of Arizona.

Výhody:

- přehledný program,
- aplikace je pro platformy Windows, Linux, MacOS,
- existuje EditiX Community Edition, která je zadarmo,
- mnoho užitečných funkcí např. Find and Replace,
- obsahuje inteligentní našeptávač, který pomáhá uživatelům.

Nevýhody:

- verze Pro je placená, ale nabízí trial verzi na 30 dní,
- existuje Lite verze, které chybí mnoho funkcí,
- zaplacení licence se vztahuje pouze na jednoho uživatele,
- první update je zdarma, další se musí zaplatit.
- aplikace funguje jako editor a proto není možná, aby více uživatelů současně pracovali se stejnými daty.

[2]domovská stránka: `http://www.editix.com/index.html/`

1.1.3 EditPad Pro

EditPad Pro[3] je výhradně textový editor, který lze použít například pro HTML, Javascript nebo XML. Software podporuje změnu jazyka například do francouzštiny, němčiny, polštiny nebo švédštiny. Projeck Just Great Software, pod kterým byla vyvynuta tato aplikace, vznikl v roce 1996. Autorem projektu je Jan Goyvaerts, který je zároveň hlavním ředitelem vývojářu projektu.

Výhody:

- program je obecný textový editor,
- obarvená syntaxe,
- dobře pracuje s velkými soubory,
- podpora UTF-8,
- existuje EditPad Lite verze, která je zdarma.

Nevýhody:

- nemá explicitní podporu pro XML,
- aplikace je pouze pro Windows,
- chybí vyhledávání v souborech,
- verze Pro je placená, ale existuje verze Lite.

1.1.4 Essential XML Editor

Essential XML Editor[4] je jednoduchý XML editor. Jeho klíčovou vlastnostní je vestavěný XML validátor. Vývojáři dříve pojmenovali program Open XML Editor, ale po zavedení poplatku za některé funkce projekt přejmenovali. Autorem je Dieter Köhler.

Výhody:

- program pracuje jako textový editor,
- možnost rychle zjistit zda je soubor validní,
- trial verze není časově omezená,
- vstupní soubor může být v různém kódování,
- klávesová zkratka pro každý příkaz.

[3] domovská stránka: `http://www.editpadpro.com/`
[4] domovská stránka: `http://www.philo.de/xmledit/`

Nevýhody:

- výstupní soubor pouze v UTF-8 kódování,
- aplikace je pouze pro Windows,
- pro zpřístupnění některých funkcí nutnost zakoupit klíč,
- poměrně zastaralý desing aplikace.

1.1.5 Exchanger XML Editor

Exchanger XML Editor[5] je určen pro snadnou editaci, prohlížení, správu a konverzi XML souborů. Exchanger pomáhá svojí širokou nabídkou funkcí XML autorům a vývojářům. Software je produktem firmy Cladonia, která se zaměřuje na vývoj XML aplikací.

Výhody:

- nabízí plnou verzi na 30 dní,
- aplikace je dotupná na všech platformách,
- možnost zobrazení základního náhledu,
- poskytuje podporu pro XML formou stáhnutí balíčku,
- automatická kontrola, jestli je soubor validní.

Nevýhody:

- při instalaci nutnost najít cestu k JRE manuálně,
- zastaralý software,
- časově neomezenou verzi je nutno zakoupit,
- poslední update proběhl v roce 2010.

1.1.6 Vyhodnocení

Z potenciálních konkurentů můžeme vyškrtnout Emeditor, protože je placený. Existuje neplacená verze Emeditoru, ale zároveň v ní chybí například možnost vložení tagu před a za označený text. Aplikace Editix je také zpoplatněna. Velkou nevýhodou je placení každého dalšího updatu editoru zvlášť. EditPad Pro je také placený a ve volně dostupné verzi není možnost vyhledávání v souborech. Navíc explicitně nepodporuje XML. Z názvu editoru Essential XML Editor lze odvodit, že je určený pro XML, ale ihned po spuštění aplikace je zřejmá dlouhodobá neaktualizace designu. Podobný problém s designem má Exchanger XML Editor, u kterého navíc poslední update softwaru proběhl v roce 2010.

[5] domovská stránka: `http://www.exchangerxml.com/editor/`

1.2 Technologie

Mezi první otázky patřilo, v jaké technologii se bude aplikace psát. Po domluvě s pracovníky UČL AV jsem měl vytvořit webovou aplikaci, která má mít přívětivé uživatelské rozhraní pro snadnou a rychlou správu elektronických děl.

Další duležitou otázkou bylo, ve kterém elektronickém formátu se budou díla tvořit a uchovávat. Existuje celá řada formátů, proto bylo pro budoucí možné rozšíření duležité domluvit se s pracovníky na správném formátu.

1.2.1 Jazyk

U jednoduchých webových aplikací postačí, když klientská strana pošle požadavek na serverovou část, ta jej vyhodnotí a pošle odpověď zpět. Pro implementaci tohoto případu se nejčastěji používá architektura client/server [3]. Jedna z výhod této architektury je, že klientská strana programu je oddělena od serverové části. Výpočty jsou prováděny na straně serveru, proto nejsou požadovány vysoké nároky na výpočetní techniku počítače, na kterém běží klientská část. Tato architektura také chrání data jejich uložením na server, což je jedna z nejlepších metod ochrany.

Nevýhodou může být vysoká cena zařízení pro provoz případně je nutné mít pro správu serveru systémového administrátora.
Srovnání jazyků a informace pro webové aplikace uvedené na [3].

- PHP – jeden z nejrozšířenějsích jazyků, který podporoje většina poskytovatelů webhostingu. Tento jazyk je široce využíván mezi uživateli a obsahuje mnoho standardních knihovních funkcí. Hodí se pro malé nebo středně velké projekty.

- Ruby – navzdory tomu, že Ruby je mladý jazyk, těší se velké oblíbenosti mezi webovými vývojáři. Nejznámější framework Ruby on Rails umožňuje rychle vytvářet vzorové nebo malé projekty. Pro nové uživatele je Ruby on Rails poměrně složitý. Neznalý uživatel bude mít na začátku problém i s implementací jednoduchých funkcí.

- Python – díky nástupu velkých frameworků (například Django), je možné použit Python ve velkých webových projektech. Syntaxe kódu je podobná Ruby, ale hlavní rozdíl činí ideologie psaní kódu.

- C# ASP.NET – pro správné fungování jazyka je třeba zařídit ISS server, který je nezbytný pro mnoho komerčních projektů od firmy Microsoft. Jiným řešením může být použití serveru Mono, který ale není stabilní a může obsahovat mnoho chyb.

- JAVA – tento široce využívaný jazyk lze aplikovat na velké projekty. Je poměrně rychlý a obsahuje spoustu již vyřešených složitých problémů,

které nemusí uživatel znovu řešit. Pro začátek je vyžadováno poměrné velké množství znalostí, aby fungovala základní kostra programu, což se považuje za významnou nevýhodu.

Pro tento projekt postačí menší a jednodušší aplikace, kterou zaměstnanci UČL AV snadno zprovozní na svých serverech. Pracovníci nemají k dispozici server ISS, proto vypadává z výběru C# ASP.NET. S Ruby nemám žádné zkušenosti a Python jsem dříve používal jen okrajově. PHP i JAVA jsou velice rozšířené jazyky ve webovém inženýrství, nicméně k PHP mám kladnější vztah. Velkou výhodou pro PHP je, že PHP aplikace už na serverech UČL AV běží. Znalosti PHP jsem využil hned v několika předmětech při studiu. Na základě této analýzy byl vybrán jazyk PHP.

1.2.2 Databáze

Srovnání několika open-source databázových řešení podle [4].

- MariaDB[6] – byla vytvořena původními vývojáři MySQL. MariaDB využívají dnes největší společnosti jako jsou Google nebo Facebook. Ochrana dat je na špičkové úrovni. Systém působí na trhu přes 20 let. Jako nevýhoda se dá počítat chybějící rozhraní pro mezipamět.

- MongoDB[7] – byla vytvořena v roce 2007 jako řešení pro větší projekty. Díky svým sponzorům a podporovatelům si tato NoSQL databáze uchovává myšlenku být jednoduchá a efektivní. Dokáže zpracovat poměrně rychle velké množství dat. Po zaznamenání systémové chyby následuje rychlé obnovení dat. V nerelační databázi se poměrně složitě opravuje návrh databáze. Nevýhodou pro MongoDB je, že pracovnící mají zkušenosti s relační databází.

- MySQL[8] – funguje již od roku 1995. Dnes je využívána jako standardní databáze pro menší i větší projekty. Běží na všech známých operačních systémech a funguje i při výpadku internetu. Velkou výhodou je oddělený server od vývojového prostředí a navíc MySQL používají pracovnící ústavu na svých serverech. Naopak nevýhodou je delší prodleva mezi novými aktualizacemi.

- PostgreSQL[9] – má za sebou 15 let aktivního vývoje a patří mezi databáze, které běží na všech hlavních operačních systémech. S použitím PostgreSQL může uživatel vytvořit vlastní metody nebo nestandardní datové typy. Mnoho věstavěných procedur lze spouštět pomocí mnoha

[6]domovská stránka: `https://mariadb.org/`
[7]domovská stránka: `https://www.mongodb.com/`
[8]domovská stránka: `https://www.mysql.com/`
[9]domovská stránka: `https://www.postgresql.org/`

programovacích jazyků, jako je Java, Perl, Python nebo C/C++. Vývoj je řízen pouze komunitou.

- SQLite[10] – se podle [4] považuje za nejpoužívanější databázi na světě. Vývoj začal v roce 2000 a používají jej významné firmy jako je Facebook, Apple nebo Microsoft. Po každé aktualizaci, vývojáři zveřejní podrobný výpis změn. K dispozici je kvalitní podpora a knihovna, která na úkor velikosti paměti pracuje velmi rychle. SQLite se nedoporučuje pro obsáhlé webové aplikace a velké množství dat.

Výsledná aplikace bude obsahovat méně než 10 tabulek. Aplikace využije jednoduché dotazy a pro chod aplikace nebude potřeba posílání několika dotazů současně. Do budoucna se počítá s rozšířením aplikace, proto je výhodou, když je databáze urdžována a aktualizována. Na základě analýzy, a protože tento projekt bude patřit spíše k těm menším, byla vybrána databáze SQLite.

1.2.3 Elektronický formát

„*Mezi formáty, v nichž můžeme číst elektronickou literaturu, jsou jednak ty, které byly pro tento účel přímo vytvořené, ale také ty, v nichž se e-knihy publikovaly prostě proto, že nebylo mnoho jiných alternativ. Toto se týká zejména stavu v 90. letech, kdy vznikaly kopie (převážně papírových) knih převedené do formátů jako jsou TXT, HTML či RTF.*“[5]

Elektronickému formátu je třeba věnovat zvláštní pozornost, protože se do buducna počítá s rozšířením aplikace o část, která se bude věnovat zobrazením těchto děl pro veřejnost.

Seznam vybraných formátů pro e-knihy uvedené v [5].

- Archos Diffusion – je formát vytvořen franouzskou firmou ArchosDiffusion. Koncovka názvů souborů je .aeh. Formát byl vytvořen pro uchovávání literatury v elektronické podobě a patří do skupiny formátů založených na XML. Otevírat soubory lze v programu Archos Player nebo ve volně dostupné aplikaci Visual Vision EbooksReader. Postupem času uvadá zájem o tento formát.

- AZW – formát vyvinuli vývojáři Amazonu a používá koncovku .azw. Tento formát byl v zhotoven pro uchování elektronických děl v internetovém knihkupectví společnosti Amazon. Knihy lze číst ve všech dostupných verzích čteček Kindle. Díky oblíbenosti čteček Kindle patří AZW mezi nejrozšířenějších formáty na světě. Velkou nevýhodou je jeho uzavřenost, protože knihy ve formátu AZW prakticky nelze číst v jiných čtečkách než jsou Kindle.

[10] domovská stránka: `https://www.sqlite.org/index.html`

- EPUB – patří mezi nejpopulárnější formáty na světe i v České republice. Vytvořilo jej sdružení International Digital Publishing Forum a je založený na XML. Knihy ve formátu EPUB lze číst na většině čtecích zařízeních s výjimkou čteček Kindle, nicméně existuje možnost převodu formátu EPUB do jiného, který dokáží číst i čtečky Kindle. Tento otevřený formát podporuje Digital rights management (DRM) ochranu a proto si získal oblibu i u nakladatelů.

- Hypertext Markup Language (HTML) – se primárně používá pro tvorbu webovýh stránek. V 90. letech, kdy se začaly poprvé vytvářet elektronické podoby knih, nebyl ještě vyvinut žádný formát pro jejich zobrazení, a proto se z nutnosti používal také HTML. Převod probíhal nejprve naskenováním díla a poté nahráním do aplikace Optical Character Recognition (OCR). Výstupem aplikace bylo dílo ve formátu HTML. Ke čtení postačil webový prohlížeč. Nicméně s nástupem formátů navržených pro elektronické knihy se přestal HTML používat.

- Portable Document Format (PDF) – vyvinutý společností Adobe v roce 1993 byl primárně určen pro uchování souborů pro tisk. Patří mezi formáty, které nebyly vytvořeny pro elektronickou literaturu, ale narozdíl od ostatních, se tak používá dodnes. Velkou předností je nezávislost na platformě, protože s PDF lze pracovat téměř na všech operačních systémech.

- Plain text (TXT) – patří mezí formáty, které nebyly určeny pro uchovávání e-knih. V 90. letech se používal pro zobrazení elektronické literatury, protože nebylo tolik jiných možností. Díky malé datové velikosti souborů a možnosti čtení souborů neomezeně na platformě se stále používá. TXT nedovoluje formátování a nepodporuje vložení obrázků, videí či zvukových stop.

- Text Encoding Initiative (TEI) – byl vytvořen TEI konsorciem primárně pro elektronickou literaturu. Formát je využíván ve výukových projektech i knihovnách po celém světě. Řadí se do skupiny formátů, které jsou založeny na XML. TEI se označuje jako nastavitelný, protože uživatel může podle své vůle přidat, předefinovat nebo přejmenovat tagy a jejich atributy. Formát se dá použít pro různé druhy textů.

Drtivá většina děl dostupná pracovníkům UČL AV je právě ve formátu XML. Avšak tento formát není primárně určen pro uchovávání elektronických verzí literárních děl, proto byl pro mou práci po diskuzi s pracovníky UČL AV vybrán textový formát TEI.

1.3 Software

1.3.1 Microframework vs framework

V následujícím textu vycházím z [6].

Framework poskytuje skoro vše, co programátor potřebuje, od obsluhy webových požadavků po komunikaci s databází. Obsahuje i komponenty, které vývojář nemusí nikdy použít, nicméně z hlediska rozšiřitelnosti jsou výhodné.

Microframework je označením pro framework, který obsahuje pouze nejnutnější komponenty k vývoji webové aplikace. Microframeworky bývají přizpůsobeny menším aplikacím nebo aplikacím s velmi konkrétním účelem. Pro rozšíření funkčnosti je potřeba přidat dané komponenty.

Reálně je microframework sbírka nejnutnějších potřebných komponent pro potřebu webových aplikací, obvykle výtvory na základě architektury MVC. Obsluha microframeworku dostane HTTP požadavek, který zpracuje daný kontroler a ten pošle odpověď obvykle ve formátu HTML zpět. Některé microframeworky obsahují další nástroje pro manipulaci s HTTP požadavky. Mnoho vývojářů používá raději velké frameworky, jako jsou Laravel nebo Symfony. Tyto frameworky disponují mnoha již vyřešenými problémy a mají velikou programátorskou základnu. Tyto frameworky potřebují ale více času pro pochopení a porozumění prostředí.

Jsou projekty, u kterých se vyplatí použít frameworky, například velké e-shopy. V případě projektů, kde není potřeba tolik funkcí a nevyužila by se na plno síla frameworků, je vhodné použít microframework. Tento projekt je spíše menší a nebude obsahovat složité požadavky. Aplikace nebude mít mnoho stránek a proto ideláním řešením bude microframework.

Pět nejlepších microframeworků podle [7].

- Slim[11] – je považován za jeden z nejlepších PHP microframeworků. Umožnuje snadno vytvořit kvalitní webovou aplikaci. Díky nastavitelné a modulární architektuře poskytuje vývojářům přesně to, co potřebují. Slim dovoluje vkládat závislosti, proto jej lze použít společně s externími nástroji.

- Silex[12] – byl vyvinut z frameworku Symfony, aby byl co nejmenší a zároveň poskytoval základní funkčnost. Nakonec vznikly dvě verze. Fat verze má v sobě komponenty ze Symfony, Twig, šablonovací systém a jiné. Druhá slim verze obsahuje základní systém routování a několik procedur.

[11] domovská stránka: `https://www.slimframework.com/`
[12] domovská stránka: `https://silex.symfony.com/`

- Wave[13] – používá architekturu MVC. Neobsahuje doplňkové knihovny a klade důraz na rychlost a optimalizaci. Wave podporuje Apache i Nginx servery.
- Limonade[14] – se zaměřuje obdobně jako Wave na jednoduchost. Limonade je velmi rychlý a snadně se v něm vyvíjí aplikace. Nicméně je až extrémně malý a nelze jej rozšířit o složitější funkčnost. Spoléhá pouze na globální funkce.
- Lumen[15] – je microframework odvozený z asi nejrozšířenějšího PHP frameworku Laravel. Pokud si programátor není jistý velikostí svého projektu, je výhodné použít Lumen, protože stačí veškerý kód z Lumenu převést do Laravelu a vše bude fungovat jak má.

Po prostudování a podrobné analýze současně dostupných microframeworků byl vybrán Slim. Zároveň bude použit šablonovací systém TWIG, který Slim podporuje.

[13] domovská stránka: `https://www.waveframework.com/`
[14] domovská stránka: `https://limonade-php.github.io/`
[15] domovská stránka: `https://lumen.laravel.com/`

Kapitola 2

Návrh aplikace

Návrh aplikace je velice důležitou fází projektu, ve které by mělo dojít ke sjednocení požadavků zadavatelů a reálného provedení. Vytvořením dobrého návrhu se zamezí případným kolizím a zároveň proběhne první interakce mezi zákazníkem a dodavatelem. Ze strany UČL byl vznesen požadavek, aby v aplikaci byl použit Bootstrap[16], protože jsou na něj zvyklí. Databázové schéma bylo čistě na mém rozhodnutí.

2.1 Wireframy

Jedním z nejdůležitějších úkolů bylo navrhnout, jak budou jednotlivé stránky vypadat a kolik jich aplikace bude obsahovat. Základní rozložení stránek bylo na mém rozhodnutí. Design měl být jednoduchý, uživatelsky přívětivý a moderní. Tyto návrhy musely být a byly schváleny pracovníky UČL AV.

Grafika měla být jednoduchá, bez složitých a obsáhlých prvků. Pro snadnější stylování aplikace byl použit Bootstrap. Jedná se o volně dostupnou knihovnu, která dovoluje stylovat vzhled aplikace pouze přídáním určitých tříd k elementům v HTML. Při dodržování pravidel Bootstrapu není potřeba vkládat obsáhlé CSS styly. Pomocí Bootstrapu lze poměrně snadno vytvořit responzivní aplikaci.

Pro návrh GUI jsem se rozhodl použít webovou aplikaci Moqups[17]. Její bezplatná verze nabízí velké množství šablon pro grafické návrhy. Od základních prvků jako je tlačítko nebo nadpis, po mírně složitější formuláře. Velikou výhodou je, že Moqups má v nabídce prvky Bootstrapu.

2.1.1 Login

Pro úvodní přihlašovací stránku jsem vybral jednoduchý formulář, ve kterém je email a heslo, viz obrázek A.1. Pozadí, které by se dalo v budoucím rozšíření

[16] domovská stránka: `https://v4-alpha.getbootstrap.com/`
[17] domovská stránka: `https://moqups.com/`

měnit, je jako na všech ostatních stránkách bílé.

2.1.2 Hlavní stránka

Po úspěšném přihlášení do aplikace se zobrazí hlavní stránka, která je rozdělena do tří částí, jak je vidět na obrázku A.3. V hlavičce se nachází společně s nadpisem odkaz na seznam autorů a vydavatelů děl a vysouvací menu s možnostmi uživatele. Dále je zde filtr aplikovatelný na seznam zobrazených děl. Požadavek od UČL AV byla možnost filtrovat elektronickou literaturu podle autora, roku vydání, textu obsaženého v díle a statusu. Třetí část obsahuje samotný seznam děl. Tento seznam je ve tvaru tabulky se sloupci (*název díla, autor, rok vydání, status*), odkaz na přílohy a nezbytné akce a tabulka se dá seřadit podle vybraných sloupců. U seznamu děl lze nastavit počet zobrazených děl a obsahuje rychlý vyhledávač v textu zobrazeným v tabulce.

2.1.3 Metadata

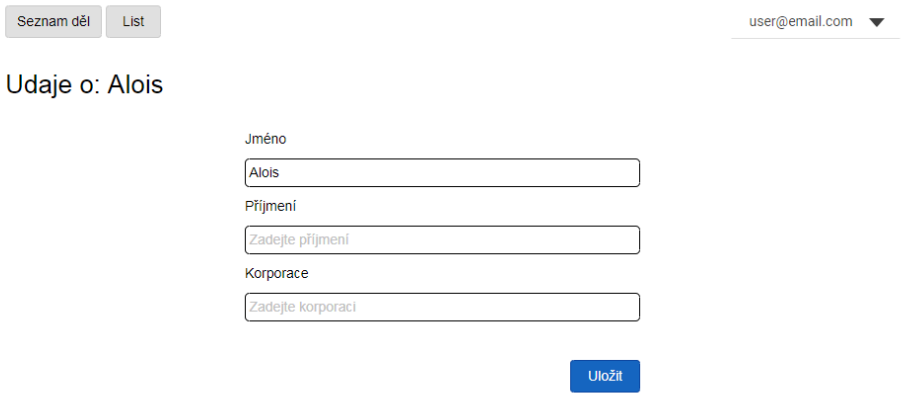
Na obrázku A.4 je vidět rozložení stránky pro úpravu základních údajů o dílu. Opět se skládá ze tří částí. První obsahuje navigaci, dropdown pro přihlášeného uživatele a nadpis. Druhá část je věnována autorům a vydavatelům. V tomto oddílu bude umožněno přidávat a odebírat autora nebo vydavatele. Poslední sekce obsahuje formulář pro úpravu metadat literárního díla. Společně s pracovníky UČL AV byly vybrány atributy, které přímo nesouvisí s obsahem díla, nýbrž popisují samotnou publikaci.

2.1.4 Přílohy

Součástí zadání práce je správa příloh, zejména scanů stránek. Přílohám se věnuje právě tento segment. Obsahem wireframu A.5 jsou naskenované jednotlivé stránky daného literárního díla. V horní části se opět objevuje navigace, uživatelské funkce a nadpis. Následuje sekce věnovaná hromadnému uploadu skenů do níže zobrazené fotogalerie.

2.1.5 Autoři a vydavatelé

Pro správu autorů a vydavatelů byl použit návrh zobrazený v příloze A.6. V horní části se vyskytuje navigace, nezbytné funkce a nadpis. Dále je zde umístěna tabulka záznamů, ve kterých lze snadno vyhledávat. Každý záznam lze upravit a smazat. Pro upravení záznamu byl navržen wireframe z obrázku 2.1, který obsahuje navigaci, funkce pro uživatele a zjednodušený formulář. Formuláře pro upravení vydavatele nebo autora jsou totožné.



Obrázek 2.1: Úprava autora

2.1.6 Text

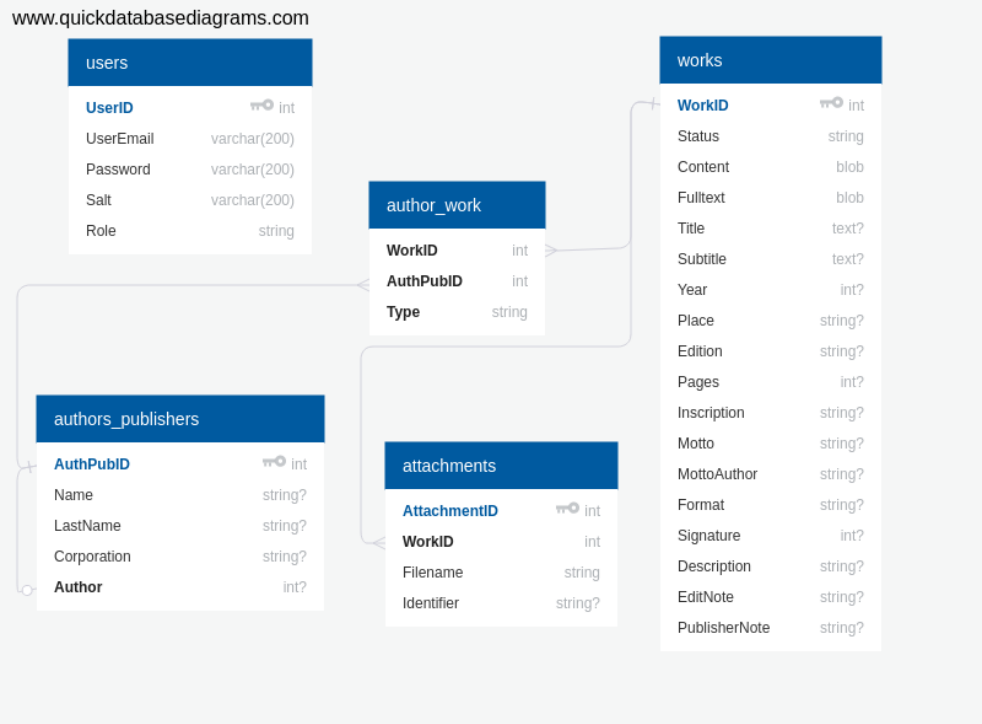
Nejdůležitějším prvkem aplikace byla bezesporu možnost upravovat text literárního díla. Pro tuto funkci byla navržena stránka z obrázku A.7. Jako u předešlých wireframů, horní část se věnuje navigaci, funkcím uživatele a nadpisu. Dále je vidět rozložení stránky na dva oddíly. Jeden pro text knihy a druhý pro možnost vkládání značek. Menu vpravo bylo navrženo vzhledem k nadměrné velikosti první části tak, aby bylo viditelné, když se uživatel posune níž.

2.2 Databázové schéma

Neméně duležitou součástí projektu byla databáze. Ze strany UČL AV nebyly vzneseny žádné požadavky na podobu databáze, proto se mohlo schéma přizpůsobit dle potřeby aplikace.

Pro model databáze byla použita webová aplikace[18]. Schéma je navrženo minimalisticky, ale tak aby zároveň pokrývalo veškeré potřeby aplikace. Výsledný návrh databáze na obrázku 2.2, obsahuje 5 tabulek s jednoduchými vazbami.

[18]domovská stránka: `https://www.quickdatabasediagrams.com/`



Obrázek 2.2: Databázové schéma

V tabulce uživatelů je uloženo (*UserID*), (*UserEmail*) a heslo rozdělené do dvou sloupečků viz sekce 2.3. Uživatelské role jsou identifikovány pomocí sloupce Role.

Duležitá tabulka je (*author_work*). Obsahuje reference na tabulky (*works*) a (*authors_publishers*). Třetím atributem je typ spojení. Tento typ určuje, jestli je spojen autor nebo vydavatel díla. Každý záznam v tabulce propojuje dílo a autora nebo vydavatele. Primárním klíčem je celá trojice.

(*AuthPubID*), (*Name*), (*LastName*), (*Corporation*) a (z*Author*) jsou atributy tabulky (*authors_publishers*). Záznamem může být reálná osoba, její pseudonym nebo společnost. V případě pseudonymu odkazuje záznam na reálnou osobu. Tuto referenci obsahuje sloupec (*Author*), ve kterém je (*AuthPubID*) reálné osoby zaznamenané v (*authors_publishers*) nebo je prázdný.

Nejobsáhlejší tabulka se nazývá (*works*). Obsahuje všechny základní informace o dílu: název díla (*Title*), podtitul (*Subtitle*), datum publikace (*Year*), místo vydání (*Place*), pořadí vydání (*Edition*), počet stran sbírky (*Pages*), věnování autora (*Inscription*), motto (*Motto*), autor motta (*MottoAuthor*), formát naskenovaného díla (*Format*), podpis (*Signature*), popis díla (*Description*) a poznámka k vydání (*EditNote*). Ve (*works*) je atribut (*Status*), který indikuje momentální stav díla. (*Status*) může nabývat hodnot domluvených s UČL AV Nové, Rozpracováno, Zkontrolováno a Hotovo. Celý text včetně tagů

je ve sloupci (*Content*). Dále tabulka (*works*) obsahuje sloupec (*Fulltext*), ve kterém je text díla bez tagů.

Poslední tabulka (*attachments*) je pro přílohy přidružené k dílům. Obsahuje (*AttachmentID*), referenci na dílo (*WorkID*), (*Filename*) a poznámku k obrázku (*Identifier*).

2.3 Autentizace

Součástí zadání je požadavek na možnost autentizace. Ze strany ústavu nebyl vznesem žádný požadavek na ochranu hesel, proto jsem si způsob zabezpečení mohl vybrat. Heslo je zahashované pomocí sha256. Pro větší bezpečnost hesla jsem přidal metodou solení hesla[8] tzn., že se k uživatelskému heslu přidává náhodný text.

Kapitola 3

Implementace

Při vytváření kódu bylo nutné dodržovat domluvené návrhy. Kód musel být přehledný pro případné rozšíření aplikace jiným vývojářem. Díky vybranému softwaru a jeho přívětivé příručce byla poměrně rychle vytvořena aplikace v základní podobě. Slim má obrovskou komunitu uživatelů, což je velice nápomocné při hledání řešení problému.

3.1 Import

Ještě před implementací se musely do databáze aplikace zpracovat sbírky z UČL AV. Podle očekávání byla data dodána ve formátu XML. Import nedoprovázely žádné komplikace zejména díky snadné ovladatelnosti a standardním knihovnám scriptovacího jazyka Python[19]. Současně s importem sbírek proběhlo zpracování příloh k nim přidružených.

3.1.1 Stávající sbírka děl

Stávající díla mají pracovníci z UČL AV v databázi. Na úpravě této sbírky děl se v současné době pracovat nedá, protože pracovníkům chybí vhodný nástroj pro úpravě děl. Dříve byla tato díla upravována pomocí kancelářského balíčku Microsoft Word[20]. Word je nástroj vhodný pro editaci jednotlivých souborů, nicméně nedovoluje spravovat soubory jako celek.

Import literárních děl proběhl otevřením každého souboru a zpracováním skriptem. Skript získal informace o dílu, vytvořil nové nebo navázal na již existující autory či vydavatele a vytvořil záznamy v tabulkách. Pro práci s XML byla použita standardní knihovna ElementTree [21]. Pro připojení do databáze se použila knihovna *sqlite*.

[19] domovská stránka: `https://www.python.org//`
[20] domovská stránka: `https://products.office.com/cs-cz/word`
[21] domovská stránka: `https://docs.python.org/2/library/xml.etree.elementtree`

V ElementTree se na začátku zavolá funkce getroot. Tato funkce inicializuje proměnnou, ve které je obsah souboru a dá se v ní snadno přistupovat k jednotlivým tagům. Tag *hlavicka* indikuje údaje o dílu a obsah díla patří pod tag *text*. Funkce *find* slouží k nalezení a vrácení tagu podle jeho názvu.

Fulltextové vyhledávání v aplikaci vyžaduje text bez tagů. K tomu slouží funkce itertext, která ignoruje tagy a vrátí celý text. Naopak funkce pro získání obsahu včetně tagů není v ElementTree vestavěna. Nicméně potřebný text lze dostat spojením jiných funkcí. Obsah této funkce byl inspirován [9]. Některá díla mají více autorů. V takovém případě jsou všechna jména zapsána v tagu *author*. Tato jména jsou ohraničena množinovými závorkami a oddělena středníkem. Mnoho autorů používalo pseudonym. Pseudonym autora je indikován znakem „=“ v pořadí vlevo pseudonym a vpravo skutečné jméno například „Folklor, Č. = Machar, Josef Svatopluk“. Vlevo od znaku je pseudonym a vpravo je pravé jméno autora. Funkce doAuthors vrací id autora nebo pole id autorů v závislosti na vstupním parametru authors. Tento parametr je typu string a pokud začíná znakem „{“, má dílo více autorů. Pro tento případ funguje funkce jako rekurze. Druhým parametrem funkce je indikátor rekurze (recursion). Pro ilustraci je uveden začátek funkce zpracování autorů díla.

doAuthors

```
def doAuthors(authors, recursion):
    #there is no author
    if (authors == 'neznamy'):
        return -1
    #there are more authors
    if (authors[0] == '{'):
        authors = authors.replace('{', '')
        authors = authors.replace('}', '')
        authArr = authors.split(';')
        authArr[0] = ' ' + authArr[0]
        out = []
        for authorName in authArr:
            out.append(doAuthors(authorName, 1))
        return out
    #there is only one author
    elif (authors.find('=') == -1):
        authors = authors.replace('(', '')
        authors = authors.replace(')', '')
        lastName, sep, name = authors.partition(',')
        name = name[1:]
        if (recursion == 1):
            lastName = lastName[1:]
        #create or find and select author id from DB
        return getAuthorId(name, lastName)
```

Pro obsluhu vydavatelů byla použita funkce doPublisher, která pracuje podobně jako funkce pro autory. U vydavatele se zde nevyskytují pseudonymy.

Import do tabulky *works* proběhl ve funkci *doWorks*. Funkce má v parametrech název díla (*title*), rok vydání (*year*), (*status*), proměnnou, která obsahuje ostatní informace (*meta*), (*fulltext*), text (*content*) a poznámku k vydání (*note*). Funkce vrací id vytvořeného díla. Příklad zjištění počtu stránek díla z parametru *meta*:

```
pages = ''.join(meta.find('stran').itertext())
```

Záznamů do tabulky *author_work* byly vloženy pomocí funkce doConnection. Tato funkce má parametry workId, indexId a typeOfConn. První je id vytvořeného díla a indexId je pole id autorů nebo vydavatelů, které může obsahovat jeden prvek. TypeOfConn je typ propojení, kterým je buď author nebo publisher. Obsah funkce je pouze databázový dotaz typu insert.

3.1.2 Přílohy

Zaměstnanci UČL AV dodali společně s některými díly i jejich naskenované stránky. Adresář *scan* obsahoval podadresáře, kde byly hlavní obrázky a zmenšené obrázky. Jednotlivá díla jsou nazvána číslem například 0001.xml. Tato čísla byla zároveň jménem adresáře obsahující obrázky. Podle čísla se zjistilo, ke kterému dílu obrázky patří. V číselných podadresářích byly uloženy obrázky ve formátu jpg a u hlavních obrázků byl soubor pages.csv, ve kterém byly informace o jednotlivých stránkách. Přílohy byly uloženy s číselným názvem. Příklad záznamu v souboru pages: 002.jpg;Strana [1]. Tuto informaci bylo nutné zpracovat a vložit do tabulky jako poznámku k příloze.

V aplikaci se přílohy přidávají do adresáře *images* a podadresáře nazvaném podle id díla. Aby bylo id tvořeno pěti číslicemi, byly k němu zleva přidány nuly. Názvy příloh jsou u každého díla tvořena číslem začínajícím od 1. Pro snadnější řazení se jméno skládá ze tří znaků: čísla a nezbytných nul na začátku. Zmenšené obrázky mají ke jménu přidanou příponu _small, příklad názvu obrázku: *010_small.jpg*. Vytváření adresářů se provádí pomocí knihovny *os*, čtení souboru zajišťuje knihovna *pandas* a obsluhu souborového systému obstarala knihovna *shutil*. Pro import příloh byla použita následující funkce:

Funkce doAttachments

```
def doAttachments(workID, oldID):
    #create directory if not exists
    os.makedirs('./images/' + '{0:0=5d}'.format(workID),
                exist_ok=True)
    tmp = './scan/scan/img/'+str(oldID)+'/'
    tmpSmall = './scan/scan/thumbs/'+str(oldID)+'/'
    khe = pd.read_csv(tmp + 'pages.csv',
                      encoding='cp1250',
                      sep=';', header=None)
    khe.columns = ['id', 'poznamka']
    l = 1
    for value in khe.id.keys():
        db.execute(insertAttachmentSQL,
                   [workID, khe.poznamka[value],
                   str('{0:0=3d}'.format(l)) + '.jpg'])
        sh.copy(tmp + khe.id[value],
               './images/' +
                str('{0:0=5d}'.format(workID)) + '/' +
                str('{0:0=3d}'.format(l)) + '.jpg')
        sh.copy(tmpSmall + khe.id[value],
               './images/' +
               str('{0:0=5d}'.format(workID)) + '/' +
               str('{0:0=3d}'.format(l)) + '_small.jpg')
        l = l + 1
```

3.2 Aplikace

Aplikace se vyvíjela podle domluvených návrhů s úpravami, které byly dohodnuty s pracovníky UČL AV. V průběhu vývoje pracovnící ústavu změnili některá svá rozhodnutí, například u hromadného nahrávání příloh již neměla být možnost upravovat pořadí obrázků. Do termínu odevzdání této práce se také nevyjasnila sada tagů použitých pro formátování textu.

3.2.1 Kostra aplikace

Vývojáři Slimu dávají k dispozici počáteční aplikaci[22] jako výchozí stav pro vývoj. Po naklonování repozitáře a spuštění composeru podle návodu skeletonu lze spustit testovací server pomocí PHP příkazem:

```
php -S localhost:8080 -t public public/index.php
```

[22] Slim-Skeleton: `https://github.com/slimphp/Slim-Skeleton`

Po zadání příkazu je možné aplikaci spustit v internetovém prohlížeči na adrese *localhost:8080*.

Adresářová struktura byla ponechána podle skeletonu. V kořenovém adresáři jsou podadresáře *logs*, *public*, *src*, *templates* a *tests*. Podadresáře *components* a *vendor* se přidaly automaticky po spuštění composeru a nahrání nezbytných knihoven. Poslední podadresář *db* byl vytvořen pro uložení databáze.

Adresář *logs* slouží pro archivaci logů v aplikaci. Framework Slim dovoluje pomocí *middleware* snadno zapisovat potřebné výpisy.

V adresáři *public* jsou skripty, soubory pro stylování a přílohy k dílům. Podadresáře *js* jsou pro skripty, *css* obsahuje styly a *images* slouží pro naskenované obrázky, které patří k jednotilvým dílům. Dále je zde soubor index.php, ve kterém se spouští samotná aplikace.

Ve složce *src* jsou soubory nezbytné pro fungování Slim aplikace. Soubor *dependencies.php* je určen pro vložení závislostí na externích knihovnách. Slim používá DIC (Dependency Injection Container) systém pro uchovávání těchto závislotí. Úkolem systému je nahrát závislost, uložit a poskytnou ji programátorovi kdykoliv to bude potřebovat. V této aplikaci byla použita databáze SQLite, pro připojení závislosti byl přidán následující kód do dependencies.php:

Přidání databáze

```php
<?php
    $container['db'] = function ($c) {
        $pdo = new PDO("sqlite:./db/ebooks");
        $pdo->setAttribute(PDO::ATTR_ERRMODE,
                           PDO::ERRMODE_EXCEPTION);
        $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
                           PDO::FETCH_ASSOC);
        $pdo->exec( 'PRAGMA foreign_keys = ON;' );
        return $pdo;
    };
?>
```

Díky SQLite je databáze uložena v jednom souboru (ebooks) v adresáři *db*. V souboru *middleware.php* lze nastavit kód, který slouží například k ovládání požadavku a odpovědi na server. Zde se v aplikaci nastavila proměnná session, ve které je uložen přihlášený uživatel. Pomocí middleware se kontroluje, zda je uživatel přihlášený a jestli má práva k dané akci. Počáteční konfigurace aplikace je v *settings.php*. Obsluha všech požadavků zaslaných na server je implementována v souboru routes.php. Díky Slimu je na serveru jednoduché zachytit požadavky zaslané z aplikace. Metoda POST funguje obdobně jako GET. Následující funkce se spustí po zaslání požadavku GET na stránku login:

Obsluha požadavku GET na login

```
<?php
    $app->get('/login', function (Request $req, Response $res,
                                  array $args) {
        $tmp = false;
        if ($req->getParam('sessionError')) {
            $tmp = true;
        }
        return $this->view->render($res, 'login.twig', [
            'sessionError' => $tmp
        ]);
    })->setName('login');
?>
```

Adresář *templates* obsahuje html šablony pro zobrazení na webu. Podle návrhu měla aplikace využívat šablonovací systém TWIG, proto adresář obsahuje soubory s koncovku .twig a výchozí soubor home.phtml. Navíc pomocí TWIGu lze zobrazit stejný formulář pro různá data, například v aplikaci se pro úpravu autora nebo vydavatele používá stejná šablona authorPublisher.twig. Ve výchozím souboru se načítají potřebná metadata, css a skripty. Tento soubor se používá jako základ html stránky pro zobrazení všech šablon. Následující příkaz zajišťuje vložení TWIG šablony do výchozího html souboru.

```
{% block content %}{% endblock %}
```

Adresář *tests* obsahuje soubory pro automatické testování aplikace. Ve skeletonu jsou připraveny soubory pro jednotkové testování. Obsahují předvyplněné základní testy, nicméně pro testováná této aplikace museli být přizpůsobeny konfiguraci aplikace.

3.2.2 Login

Na stránce *login* byl podle návrhu naimplementován jednoduchý přihlašovací formulář pro email a heslo. Pokud je email nebo heslo špatně zadané, zobrazí se uživateli formulář se zprávou o nesprávně zadaném obsahu. Po úspěšném přihlášení se stránka přesměruje na seznam děl *content*.

3.2.3 Seznam děl

U seznamu děl a všech dalších stránek je narozdíl od návrhu horní část stránky vyhrazena pro navigační lištu aplikace. Do této části jsou vlevo vložené odkazy na ostatní stránky a vpravo možnosti přihlášeného uživatele. Tato lišta je implementována pomocí Bootstrap komponenty *navbar*.

Drobné úpravy oproti návrhu jsou vidět i ve filtru děl. Pro větší přehlednost byly jednotlivé složky filtru rozděleny po řádcích. Výběr autorů zajištuje ex-

terní knihovna Selectivity[23]. Tato knihovna obsahuje několik možností výběru prvků, ve filtru byl použit vícenásobný výběr společně s možností prázdného výběru. Ostatní prvky filtru patří ke klasickým Bootstrap elementům. Dropdown prvky pro výběr roku vydání, textová pole pro fulltextové vyhledávání a checkboxy, které značí jaký typ díla má být zobrazen.

Tabulka literárních děl se shoduje s návrhem a je implementována pomocí knihovny DataTables[24]. Mezi využité výhody této knihovny patří vestavěné stránkování, okamžité vyhledávání v datech a možnost vicenásobného řazení podle zvoleného sloupce.

3.2.4 Seznam autorů a vydavatelů

Pro zobrazení seznamu autorů a vydavatelů byla použita stejně jako u seznamu děl knihovna DataTable. Vzhledem k návrhu přibyl v tabulce sloupec s počtem výskytu daného záznamu u díla. Možnost přidat nového autora nebo vydavatele zde chyběla, proto byla doplněna formou odkazu na příslušnou stránku.

3.2.5 Metadata

Stránka *metadata* dovoluje uživateli spravovat údaje o dílu, které přímo nesouvisí s obsahem samotné knihy. K implementaci byli navíc oproti návrhu přidaní vedle autorů vydavatelé. Obě skupiny lze přidávat a odebírat pomocí knihovny Selectivity a mohou obsahovat jeden i více záznamů.

Metadata lze upravovat pomocí jednoduchých Bootstrap textových inputů nebo textarea formulářových polí.

3.2.6 Přidání nového autora nebo vydavatele

Přidat nového autora nebo vydavatele lze provést dvěma způsoby. První je přímo na stránce metadata díla, kde se společně se záznamem vytvoří spojení v tabulce *author_work*, a druhá je v seznamu autorů a vydavatelů, kde se vytvoří pouze samotný záznam v tabulce *authors_publishers*.

Přídat je možné osobu se jménem a přijmením, pseudonym osoby nebo korporaci. Při přidávání pseudonymu je nutno vybrat z ostatních záznamů, aby byla zajištěna reference na reálné jméno autora. Tuto možnost lze vybrat pomocí knihovny Selectivity, výběr je zde zúžen na jeden nebo žádný záznam.

3.2.7 Upravení autora nebo vydavatele

Autora nebo vydavatele lze upravit pomocí stejné TWIG šablony jako pro přidání nového údaje. Jediný rozdíl je v nadpisu stránky, který pro úpravu

[23] domovská stránka: `https://arendjr.github.io/selectivity/`
[24] domovská stránka: `https://datatables.net/`

záznamu zobrazí jméno, přijmení a korporaci a pro nový záznam zobrazí text „Nový záznam“.

3.2.8 Přidání nového uživatele

Nového uživatele může přidat pouze uživatel, který je v roli admina. Tato funkce se nachází v pravé části navigace po výběru možnosti přidat uživatele. Šablona *addUser.twig*, která je zobrazena při této akci, obsahuje jednoduchý formulář pro vložení emailu a hesla nového uživatele. Heslo je pro kontrolu nutno zadat dvakrát. Při zaslání požadavku přidání uživatele na server se kontroluje, zda email již není registrován a zda souhlasí obě zadaná hesla. V kladném případě se založí nový uživatel, který se může do aplikace ihned přihlásit.

3.2.9 Změna hesla uživatele

Změnu hesla může provést každý uživatel v pravé části navigátoru. Stránka změna hesla obsahuje obdobně jako u přidání nového uživatele jednoduchý formulář, kde se pro jistotu musí zvolit staré a poté dvakrát nové heslo.

3.2.10 Seznam uživatelů

Seznam uživatelů byl přidán do aplikace pro snadnou správu všech uživatelů v databázi. Seznam lze zobrazit v menu po kliknutí na dropdown a zvolení možnosti „Seznam uživatelů“. Obsahem této stránky je tabulka DataTables, kde se zobrazuje email, role a možnost smazat uživatele. Smazání uživatele je podmíněno nadřazenou rolí přihlášeného uživatele. Obdobně jako v seznamu děl nebo seznamu autorů a vydavatelů je zde možnost vyhledávat v zobrazených záznamech.

3.2.11 Přílohy

Při vytváření stránky přílohy docházelo ke změnám názoru ústavu a musely se opakovaně přepisovat funkce. Nicméně změny nebyly rozsáhlé. Podle návrhu byla implementována možnost hromadného uploadu scanů pomocí Bootstrap custom file input tagu. Smazání všech příloh se provede po potvrzení akce.

Nahrané přílohy se nejprve seřadí abecedně podle jména a poté se uloží na server do patřičného adresáře. Název adresáře je WorkID díla, pokud id není pětimístné jsou k němu zprava přidány nuly. Jméno přílohy se určuje počtem již nahraných obrázků k dílu a pokud počet není třímístný přidají se ke jménu zleva nuly. Automaticky se také vytvoří miniatury obrázků pro zobrazení na stránce, kterým se přidá přípona _small.

Příklad vytvoření jména adresáře příloh díla:

Vytvoření adresáře pro přílohy k dílu

```php
<?php
    $path = __DIR__.'/../public/images/' .
            str_pad($args['workId'], 5, '0', STR_PAD_LEFT).'/';
    if (!file_exists($path)) {
        mkdir($path, 0777, true);
    }
?>
```

3.2.12 Text

Změna textu je nejdůležitější částí aplikace. Zde se upravuje nebo vytváří elektronické verze literárních děl. Uživatel zde může mazat části díla, přidávat text nebo opravovat chyby z naskenování.

Změna textu se téměř shoduje s jejím návrhem. V horní části je menu, a spodní část se věnuje obsahuj díla. stránka je rozdělena do dvou částí: vlevo textové pole pro editaci elektronické verze a vpravo jsou možnosti přidání speciálních tagů. Oproti návrhu, přibyla možnost výběru statusu, ve kterém se dílo nachází.

V možnostech v pravé části stránky je vložení určitého tagu na předem označenou pozici. Pomocí Javascriptu se zjistí uživatelem označený text v dílu. Poté se tag vloží před text a za něj se vloží tag uzavírací. Pro vložení jiného tagu se jednoduše přidá další akce, ve které se zavolá Javascriptová funkce s parametrem názvu tagu. Díky této funkčnosti je aplikace prakticky nezávislá na zvoleném formátu elektronických verzí děl.

Snadnější přehled v elektronické podobě díla zajišťuje knihovna *highlight.js*[25]. Tato knihovna od sebe odděluje tagy, atributy a samotný text díla pomocí zvýraznění. Uživatel poté dokáže snadno oddělit prvky od sebe a může se soustředit pouze na svou práci.

Úprava textu probíhá buď psaním ručně do editoru nebo vkládáním tagů pomocí označení textu. Problém nastává při zobrazení velmi obsáhlého díla. Úprava textu se zpomalí, což může uživatelovi ztížit práci.

[25] domovská stránka: `https://highlightjs.org/`

Kapitola 4

Testování

Testování aplikace je důležitou součástí projektu, protože se ještě před vydáním aplikace zjistí, zda funguje vše, jak má. Díky testování se dá předejít případným fatálním chybám systému. Pro testování z hlediska kódu byla vybrána metoda jednotkových testů a pro testování, zda je prostředí uživatelsky přívětivé, se použila metoda kognitivního průchodu aplikace.

4.1 Uživatelské testování

Uživatelské testování probíhalo formou kognitivních průchodů, protože jsem se s nimy setkal v průbehu studia. Kognitivní průchod je prediktivní metoda testování aplikace. Díky této metodě se zjistí, jestli je aplikace přehledná a zda se dá ovládat intuitivně. K provedení testu jsou potřeba alespoň tři respondeti, testovací scénáře a testovací artefakt.

Relevantní výsledky testu zajistí výběr osob z různých věkových katogorií a odlišného zaměstnání. Prvním subjektem je studentka VŠ ve věku 23 let, která před rokem dosáhla bakalářského stupně vzdělání. Počítač používá denně okolo 6 hodin zejména k aktivitám na sociálních sítích. Druhým testerem je pětatřicetiletý dělník, který má základní vzdělání a denně stráví na počítači okolo 2 hodin. Poslední účastník testu je prodavač, který dovršil 53 let. Jeho dosažené vzdělání je střední s maturitou a denně stráví méně jak 3 hodiny na počítači.

4.1.1 První scénář

Prvním úkolem pro účastníky bylo přidání pseudonymu k autorovi Petr Bezruč. Na tomto scénáři se testovalo, zda uživatelé pochopí, že musí založit nového autora a přiřadit referenci k reálné osobě.

Při plnění zadání respondentům nejdéle trvalo najít, kde se přidává nový záznam autora. Poté se stalo, že testeři hledali samotný záznam Petr Bezruč

a zkoušeli jej upravit. Nicméně po menší radě se povedlo všem přidat pseudonym.

Díky připomínkám osob se zvýraznil odkaz v menu na seznam autorů a vydavatelů a změnila se pozice tlačítka pro založení nového záznamu. Nově je tlačítko viditelné ihned po přesměrování na seznam autorů a vydavatelů.

4.1.2 Druhý scénář

Jako druhá akce, kterou měli testeři projít, bylo nahrazení poslední přílohy u díla České zpěvy. Tento úkol měl ověřit, jak rychle dokážou uživatelé nalézt příslušné dílo, jestli dokážou smazat poslední přílohu a zda nahrají novou přílohu bez problémů.

Pro vyhledání díla použili respondenti filtr v horní části stránky i vestavěný filtr v tabulce děl. Intuitivně ihned nalezli poslední přílohu. Nicméně ze začátku zkoušeli nahradit původní přílohu novou. Až po chvíli pochopili, že musí nejprve přílohu smazat a poté nahrát novou.

Výsledkem testu je posunutí prvku pro nahrání příloh výš, aby bylo lépe vidět, a vložení nápovědy. Tato nápověda se zobrazí po najetí kurzoru na otazníček v kruhu.

4.1.3 Třetí scénář

Posledním scénářem bylo upravení ediční poznámky díla jejímž autorem je Ferdinand Tomek. Zde se kontrolovalo vyhledávání v dílech a zda je formulář pro změnu údajů o dílech snadno pochopitelný.

V průběhu testování využili osoby zkušenosti s vyhledáváním z předešlých scénářů a neměli žádný problém s úpravou záznamu.

Po opakovaném použití aplikace respondetni pochvalují jednoduchost a snadné ovládání aplikace. Po zpětné vazbě byly opraveny nejasné pasáže, zvýrazněny hlavní funkce a přidány krátké nápovědy.

4.2 Jednotkové testy

Jednotkové testy prověřují funkčnost jednotlivých částí programu, v tomto případě funkcí v souboru routes.php. Byly otestovány všechny hlavní stránky od zobrazení literárních děl v tabulce po úpravu jednotlivých autorů.

Framework Slim dovoluje provádět testování pomocí knihovny PHPUnit. Pomocí této knihovny lze simulovat server aplikace a posílat na něj požadavky. V odpovědích se testuje, zda odpověď obsahuje očekávaný html status a jestli se renderuje správná šablona. Test správnosti v odpovědi se provádí pomocí assertů. Příklad jednoho testu včetně assertu.

Test změny názvu díla s WorkID 52

```php
<?php
public function testPostMetadata()
    {
        $response = $this->runApp('POST', '/metadata/52',
                                  array('title' => 'test' ));
        $this->assertEquals(302, $response->getStatusCode());
    }
?>
```

Díky jednotkovým testům byly zjištěny menší chyby, které vznikly při aktualizaci databáze, nicméně byly rychle opraveny. Ostatní testy proběhly bez chyb, tudíž se aplikace považuje za otestovanou.

4.3 Testování ústavem

Kapitola 5

Budoucí rozšíření

5.1 Responzivita

Další výhoda Bootstrapu je rychlé a snadné nastavení mobilního vzhledu. Požadovaného výsledku je možné dosáhnout použitím specialních tříd do prvků html. Tyto třídy určují, jak se má prvek zobrazit na menší obrazovce. V případě rozhodnutí vytvořit responzivní aplikaci ze strany ústavu, postačí přidat výše uvedené třídy a základní mobilní vzhled je hotový.

5.2 Zobrazení děl

V průběhu vývoje pracovníci ústavu navrhli doplnit zadání o náhled díla. Jednalo by se o zobrazení díla při jeho úpravě. Byl by to zajímavý prvek aplikace, nicméně tato vlastnost byla diskutována společně s vedoucím a bylo rozhodnuto z časových důvodů nezahrnout požadavek do této bakalářské práce.

Do budoucna se počítá se zpřístupnením děl široké veřejnosti. Pro tento krok, by muselo být implementováno rozhraní, kde se budou elektronická díla zobrazovat tak, jak je zaměstnanci upravili. Buďto by se uživatelé přihlašovali do stejné aplikace jako UČL AV a měli by pomocí uživatelské role odepřeny některé funkce nebo se vyvine úplně nová aplikace, která by se věnovala čistě zobrazení literárních elektronických děl ze stejné databáze jako tato aplikace. V současnosti preferuje UČL AV druhý scénář.

Závěr

Cílem práce bylo vytvoření systému pro správu a snadné vytváření elektronických verzí literárních děl včetně otestování a řádné dokumentace. Vytvořená aplikace dovoluje snadné vložení elektronické verze díla a následnou úpravu díla. V aplikaci je k dispozi správa příloh k jednotlivým dílům. Tyto přílohy jsou naskenovaná původní díla a lze je do aplikace přidat pomocí hromadného uploadu. Instalační příručka je v příloze A.9 nebo v souboru *README.md*.

Součástí vývoje je import dat, která mají pracovnící k dipozici. Jedná se o 1700 sbírek básní ve formátu XML. Import byl ztížený nesourodými daty v databázi. Díla měla různé tagy s ruznými atributy, nicméně data se sjednotila a úspěšně importovala do nové databáze. V přílohách, které bylo nutno taky sjednotit, byly taktéž nesrovnalosti. V souboru, ve kterém byly identifikátory jednotlivých obrázků, se tyto poznámky prezentovaly ruzným způsobem. Nejpre se muselo sjednotit podobu a poté se mohl provést import do nové databáze. Nakonec proběhl převod úspěšně.

Aplikace je vyvinuta na míru pro UČL AV a proto byly všechny požadavky podrobně probírány se zaměstnancemi ústavu. Problém nastával po opakované změně požadavků v průběhu implementace. Nicméně podařilo se vyhovět všem počátečním požadavkům a po krátkých rozpravách i těm změněným. Situace je taková, že pracovníci UČL AV se rozhodli zatím nechat tagy původní, neb si sami musí ujasnit, co by bylo pro ně nejlepší.

Cíl práce se podařilo úspěšně splnit a výsledná aplikace by měla být v brzké době nasazena do praxe.

Literatura

[1] D'Ambra, J.; Wilson, C. S.; Akter, S.: Application of the task-technology fit model to structure and evaluate the adoption of E-books by Academics. *Journal of the Association for Information Science and Technology*, ročník 64, č. 1, 2013: s. 48–64.

[2] TEI Community: Editors [online]. [cit. 2018-03-22]. Dostupné z: `https://wiki.tei-c.org/index.php/Editors`

[3] Parkhomenko, A.; Sokolyanskii, A.; Gladkova, O.; aj.: Investigation of remote lab design technologies. In *2015 XI International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, Sept 2015, s. 92–95, [cit. 2018-03-31].

[4] Jennifer Champagne: The Top 7 Free and Open Source Database Software Solutions [online]. [cit. 2018-04-3]. Dostupné z: `https://blog.capterra.com/free-database-software/`

[5] Pokorný Lukáš: *Formáty elektronických knih: specifika a popularita*. Inflow: information journal [online], 2012, [cit. 2018-04-5]. Dostupné z: `http://www.inflow.cz/formaty-elektronickych-knih-jejich-specifika-popularita`

[6] Vladimir Kazankov: FULL-STACK FRAMEWORK OR MICROFRAMEWORK, LARAVEL OR LUMEN? [online]. [cit. 2018-04-12]. Dostupné z: `https://belitsoft.com/laravel-development-services/full-stack-framework-or-microframework-laravel-or-lumen`

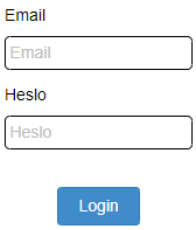
[7] Ankur Kumar: Best 5 Lightweight PHP Frameworks for REST APIs Development [online]. [cit. 2018-04-12]. Dostupné z: `http://findnerd.com/list/view/Best-5-Lightweight-PHP-Frameworks-for-REST-APIs-Development/33833/`

[8] Michal Mencl: Zabezpečení hesla z pohledu programátora [online]. [cit. 2018-05-12]. Dostupné z: http://www.it-joker.cz/Pocitace-weby/93-Zabezpeceni-hesla-z-pohledu-programatorap.2.html#kap11b

[9] Mark Amery: How do I get the full XML or HTML content of an element using ElementTree? [online]. [cit. 2018-04-29]. Dostupné z: https://stackoverflow.com/questions/380603/how-do-i-get-the-full-xml-or-html-content-of-an-element-using-elementtree

Příloha A

Screenshoty

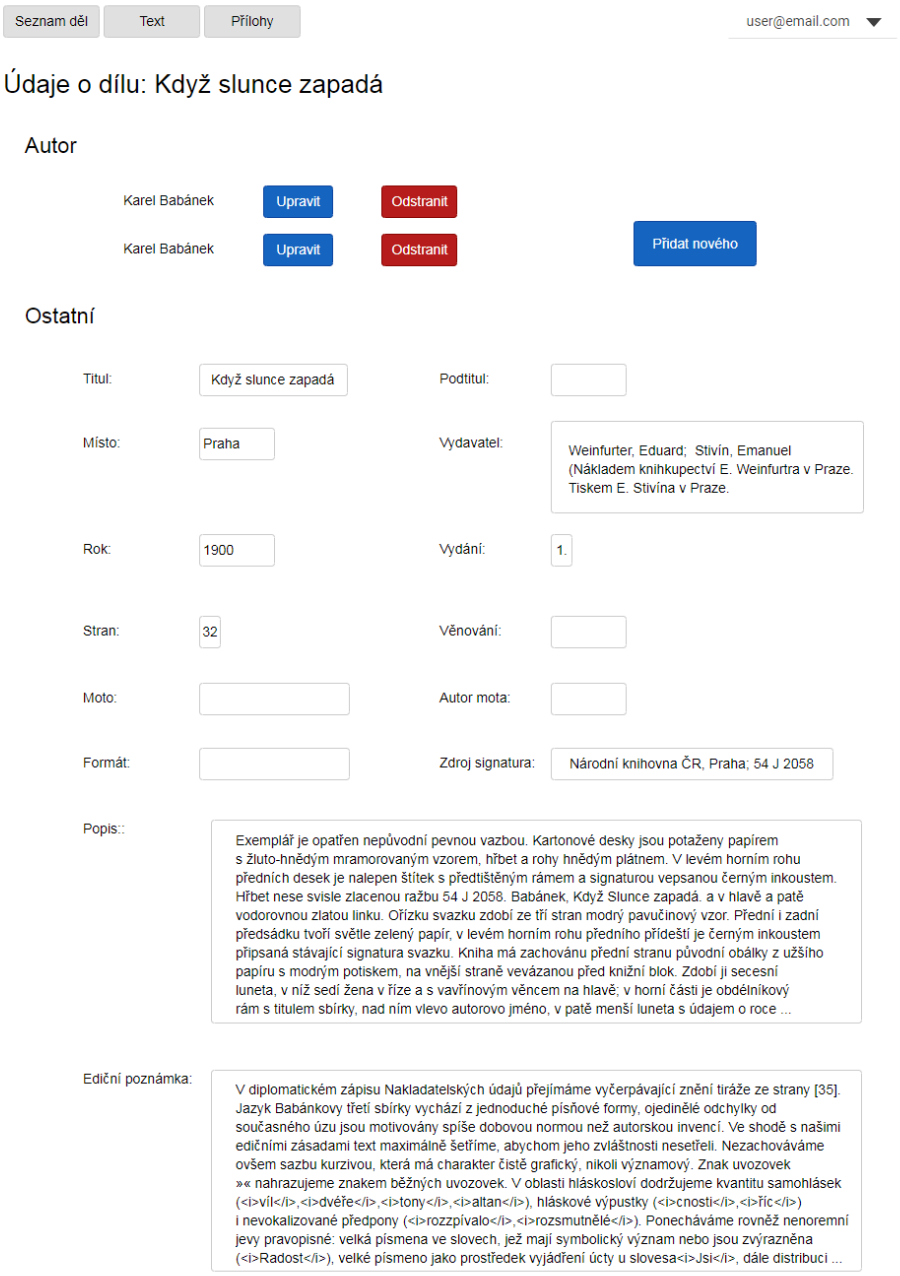


Obrázek A.1: Přihlašování

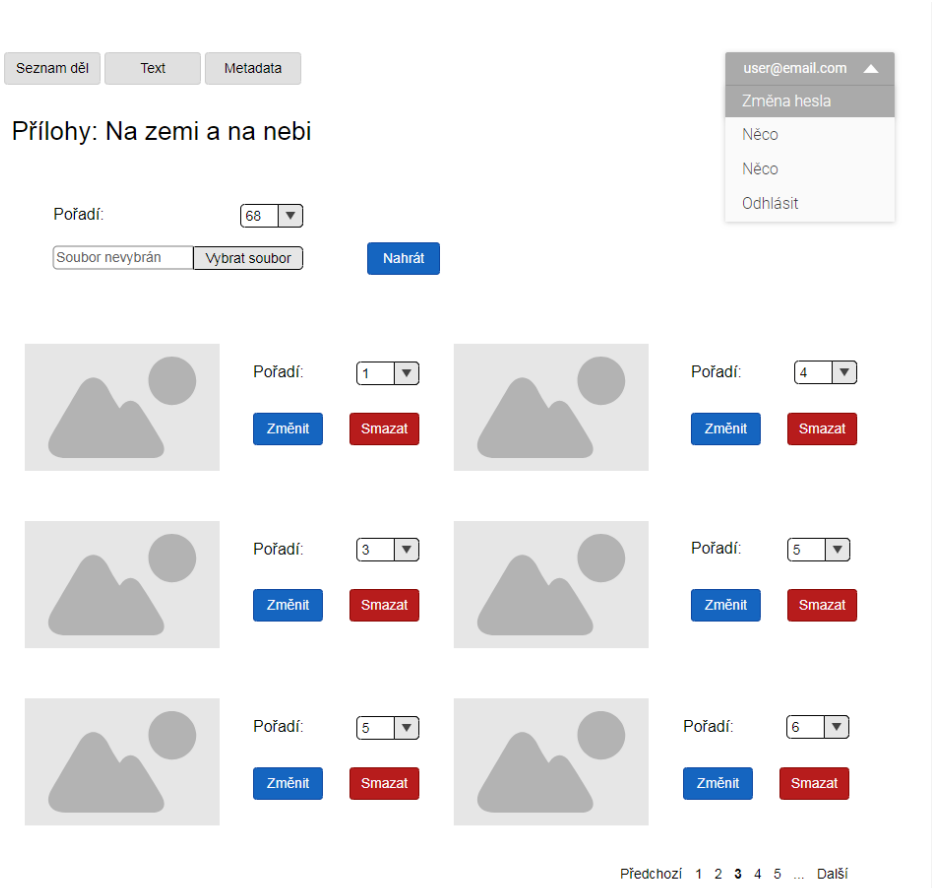
```
▼<dilo>
  ▼<hlavicka>
     <autor>Babánek, Karel</autor>
     <titul>Vytržené listy</titul>
     <podtitul/>
     <misto>Praha</misto>
    ▶<vydavatel>...</vydavatel>
     <rok>1896</rok>
     <vydani>[1.]</vydani>
     <stran>56</stran>
     <venovani/>
     <moto/>
     <autormota/>
     <format>159x119 mm</format>
    ▶<popis>...</popis>
     <zdroj-signatura>ÚČL AV ČR; 246 VIII 5</zdroj-signatura>
    ▶<edicnipoznamka>...</edicnipoznamka>
    ▶<komentare>...</komentare>
   </hlavicka>
  ▼<text>
    ▼<spisovatel>
       <b>Karel Babánek</b>
     </spisovatel>
     <br/>
    ▼<sbirka id="1">
      ▶<nadpis>...</nadpis>
       <br/>
       <strana id="1" netistena="ano">[1]</strana>
      ▶<tiraz>...</tiraz>
       <br/>
       <strana id="2" netistena="ano">[2]</strana>
      ▼<oddil id="1">
        ▼<nadpis>
           <b>Z ULICE</b>
         </nadpis>
         <br/>
         <strana id="3" netistena="ano">[3]</strana>
        ▼<basen id="1">
          ▶<nadpis>...</nadpis>
           <br/>
          ▼<strofa id="1">
             <v id="1">Jde krokem loudavým a ruce v zad</v>
             <v id="2">má skříženy a tupě hledí v před,</v>
             <v id="3">když v šeré ulice tmy závoj pad',</v>
             <v id="4">a plynu žlutavý plá v chodník svit.</v>
           </strofa>
```

Obrázek A.2: Ukázka podoby díla

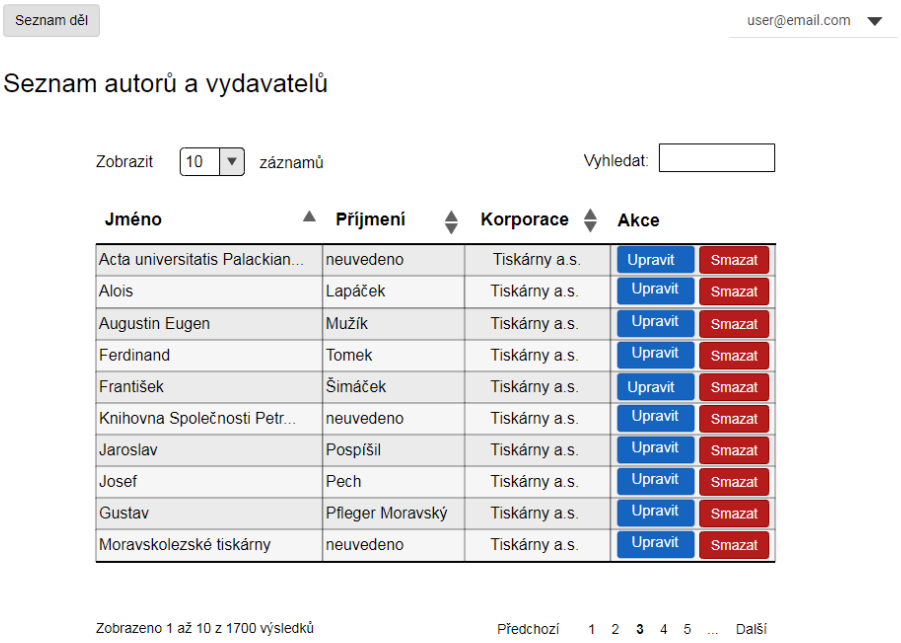
Obrázek A.3: Hlavní stránka



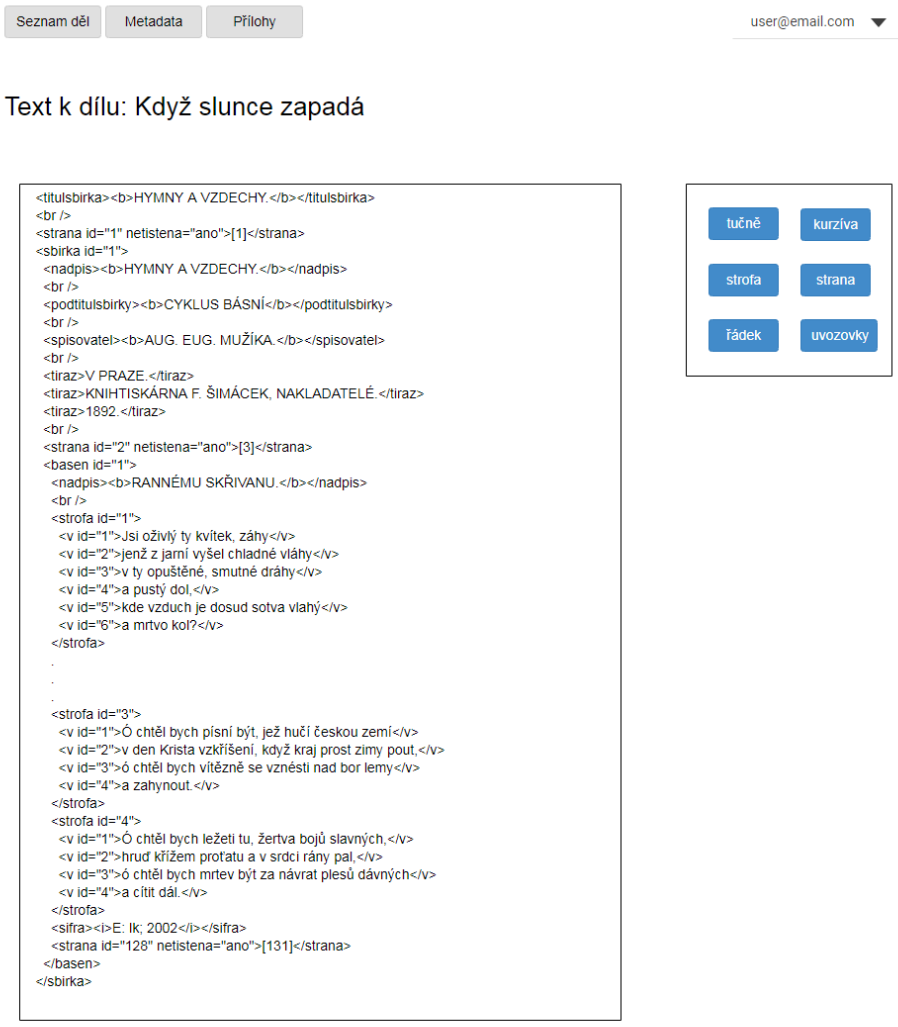
Obrázek A.4: Metadata



Obrázek A.5: Přílohy

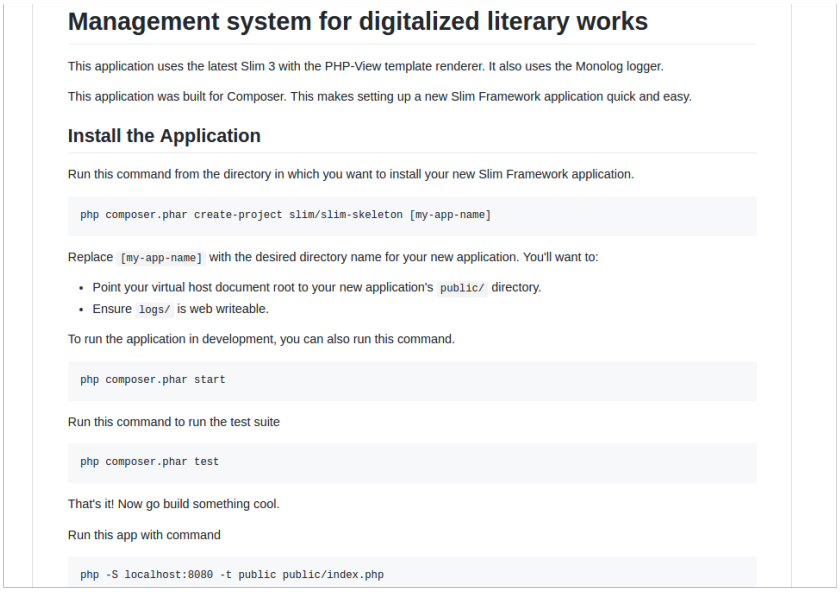


Obrázek A.6: Seznam autorů a vydavatelů



Obrázek A.7: Text díla

Obrázek A.8: Porovnání skenu a elektronické verze



Management system for digitalized literary works

This application uses the latest Slim 3 with the PHP-View template renderer. It also uses the Monolog logger.

This application was built for Composer. This makes setting up a new Slim Framework application quick and easy.

Install the Application

Run this command from the directory in which you want to install your new Slim Framework application.

```
php composer.phar create-project slim/slim-skeleton [my-app-name]
```

Replace `[my-app-name]` with the desired directory name for your new application. You'll want to:

- Point your virtual host document root to your new application's `public/` directory.
- Ensure `logs/` is web writeable.

To run the application in development, you can also run this command.

```
php composer.phar start
```

Run this command to run the test suite

```
php composer.phar test
```

That's it! Now go build something cool.

Run this app with command

```
php -S localhost:8080 -t public public/index.php
```

Obrázek A.9: Instalační příručka

Příloha B

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

TEI Text Encoding Initiative

HTML HyperText Markup Language

JRE Java Runtime Environment

UČL AV Ústavu české literatury Akademie věd České republiky

ISS Internet Information Services

DRM Digital rights management

DIC Dependency Injection Container

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS