Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Parsování CV

Bakalářská práce

Autor: Daniel Vondra
Studijní obor: Aplikovaná Informatika

Vedoucí práce: doc. RNDr. Petra Poulová Ph.D.
KIKM

Hradec Králové ……..

Prohlášení:

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

……..

V Hradci Králové dne 28.4.2019 Daniel Vondra

Poděkování:

Děkuji vedoucí bakalářské práce, doc. RNDr. Petře Poulové Ph.D., za metodické vedení práce a své rodině, přátelům a celému akademickému sboru, kteří mi při bakalářské práci velice pomohli a podporovali mě.

Anotace

Cílem práce je vytvořit webovou aplikaci, která bude schopna parsovat životopisy uživatelů, napsané v různých stylech textu a ve formátech typu DOCX, DOC, PDF a RTF. Aplikace je schopna komunikovat s databází, díky níž bude moci rozpoznávat určité prvky v textu, získaném z životopisu, které jsou nezbytné pro parsování. Dále je schopna ukládat tyto vyparsované prvky do databáze. Databáze udržuje také informace o souborech a o chybách nastalých v aplikaci. Parsovací proces je schopen rozpoznat email, telefon, jméno, příjmení a popřípadě i titul z daného životopisu. Nezbytnou součástí této aplikace jsou uživatelské přístupy, které nám umožní vytvořit částečné zabezpečení. To nedovolí zobrazovat životopisy každému, kdo by si zobrazil aplikaci.

Annotation

Title: Parsing CV

The aim of this work is to create a web application that will be able to parse users' CV, written in various text styles and formats. For example, DOCX, DOC, PDF and RTF. The application can communicate with the database, allowing it to recognize certain elements in the text, obtained from the CV, that are necessary for parsing. Furthermore, it can store these parsed elements into a database. The database also holds information about files and error information in the application. The parsing process can recognize email, phone, name, surname, and possibly the title of user in the CV. An essential part of this application are user approaches that allow us to create partial security. This will not allow CVs to be displayed to anyone who views the app.

Obsah

Seznam obrázků

1 Úvod

Parsování životopisů pomocí aplikace bylo jako téma bakalářské práce zvoleno hlavně z důvodu četných žádostí o pracovní místa. Tato webová stránka, nebo program, by mohl urychlit komunikaci a třídění životopisů uchazeč. Tato aplikace by měla personalistům zejména ulehčit práci v oblasti získávání kontaktních informací, jako je email nebo telefon, pro samotnou komunikaci s žadatelem.

Základním problémem vytvářené aplikace je volba programovacího jazyka, neboť tento program musí být přizpůsobivý k různým velikostem zařízení, aplikace musí být schopna pracovat s vícero platformami, například Windows a MacOS. Program také musí být schopný měnit zvolené soubory na samotný text tak, aby v něm mohl vyhledávat potřebné informace.

Dále je nutno komunikovat s databázovým prostředím, aby bylo možno ukládat, zobrazovat a porovnávat informace ze životopisů. Tudíž musí mít jazyk možnost propojenosti s databází, prostřednictvím dotazovacího jazyka, za pomoci určitých dotazů. To je umožněno v jazyce SQL. Toto je však pouze jedna z možností. Program by také mohl komunikovat se samotným souborovým systémem a získávat informace přímo z texu daného životopisu, prostřednictvím textových editorů. Pro zvolený účel však bude snazší a přehlednější využít databázového systému, který se postará jak o celkovou evidenci životopisů, tak i o zpracovaná, nebo získaná data.

Dalším problémem je umístění, na kterém by aplikace mohla běžet a fungovat. Je nutné, aby její uživatel mohl s aplikací komunikovat odkudkoli, z jakéhokoli rozhraní a mohl používat libovolné zařízení s přístupem k internetu a se souborovým systémem. Tyto vlastnosti, krom počítačů, v dnešní době můžeme nalézt také u tabletů. Proto je nutné, aby byl program schopný s těmito zařízeními komunikovat.

V našem případě není potřeba aplikaci přizpůsobovat pro telefony, protože při větším objemu životopisů pro parsování, by telefon nebyl použitelný. Proces by trval dlouho a uživatel by nemohl využívat jiný, krom prohlížeče jiný program.

Server, kde bude aplikace nahrána, musí podporovat zvolený jazyk, skriptovací jazyk, kaskádové styly pro design aplikace a také databázi. Dále je nutno, aby místo, kde bude webový program nahrán, umožnovalo práci se soubory, které se budou parsovat. Proto je potřebný i větší paměťový a datový prostor. Musíme myslet na to, že převádíme souborové typy mezi sebou. Proto by mělo místo, kde aplikace bude nahraná, umožňovat převod prostřednictvím zvoleného jazyka. Souborový převod je v tomto případě nutný pro sjednocení vícero druhů editorů textu, které uživatel může použít pro tvorbu životopisu. Soubory, které se mohou nahrávat, nebo přidávat prostřednictvím aplikace do databáze, tedy mohou mít několik přípon. V našem případě se jedná o soubory typu PDF, DOC, DOCX nebo RTF.

Velkým problémem je také samotné vyhledávání v textu. Je nutno správně vybírat podstatné informace. Nejsložitějšími prvky pro zjišťování jsou jméno a příjmení uživatele. Ty se mohou nacházet kdekoli v hlavičce životopisu. Jména se musí vyhledávat jiným způsobem, než třeba telefon a email. Jejich struktura je jasná, a proto je lze snadno vyhledat. Zvolený text pro parsování by měl obsahovat všechny hledané informace, a proto musí být dostatečně velký výňatek z textu, nemusí se tedy brát v potaz celý text životopisu. Tím se urychlí samotné vyhledávání.

Dalším krokem je nutné zvolit grafické rozhraní samotné aplikace a jazyk, ve kterém bude toto prostředí navrženo. Aplikace by měla splňovat takový stupeň ovladatelnosti, aby uživatel dovedl s aplikací pracovat bez delšího přemýšlení, nebo studování návodu. Grafické prostředí také musí být přizpůsobivé pro různé velikosti displejů zařízení, jako jsou tablety, notebooky, počítače, popřípadě televizory. Uživatel tak může pracovat odkudkoli, ať už z domova nebo z kanceláře. Zde může pracovat bez ohledu na to, které zařízení využívá.

Jako programovací jazyk bylo zvoleno PHP. Tento jazyk je jedním z nejvíce rozšířených skriptovacích programovacích jazyků k vytváření webových aplikací. Používá se na straně serveru a slouží ke generování HTML/XHTML kódu stránky, jenž pak server odesílá do prohlížeče. [1] PHP podporuje spoustu užitečných funkcí pro práci s textem a vyhledávání specifikovaných výrazů v textu. Jednou z těchto funkcí je REGEX. Jedná se o regulární výrazy, díky kterým se dá snadno vyhledávat, za pomoci zkratek. Pokud víme, co by měl text, nebo kód obsahovat, regex s vyhledáváním pomůže. [2]

Jako systém pro řízení báze dat, bylo zvoleno MySQL, hlavně pro svou jednoduchou funkci propojení mezi PHP a samotným databázovým systémem. Má také dobrou kompatibilitu s aplikacemi, běžícími nad skriptovacím jazykem PHP, což zaručuje snadné používání příkazů pro získávání, nebo správu dat. Také je jednou z nejrozšířenějších databázových prostředí, která jsou spojena s webovými aplikacemi. MySQL umožní ovládat veškeré operace spojené s databází v jednoduchém a snadno pochopitelném webovém prostředí phpMyAdmin. Těmito úkony je myšleno vytváření tabulek a celková správa dat, ať už se jedná o vkládání, získávání, mazání nebo upravování záznamů.

Pro umístění naší aplikace, byl zvolen server, který běží na Linuxu. Server umožní přistupovat k aplikaci odkudkoli, pouze za podmínky internetového připojení. Tento server umožňuje, mimo jiné, komunikovat s námi zvoleným databázovým systémem. Také je zde možnost ukládání do databáze výše zvoleného typu. Operační systém Linux oproti Windows podporuje funkce nutné pro převádění jednotlivých typů souborů, ve kterých se může životopis nacházet. Proto je výhodné využití serveru s operačním systémem Linux. [3]

Pro vyhledávání v textu, jak již bylo zmíněno, bude použit REGEX, což je zkratka z anglického slova regular expression, překlad je regulární výraz. REGEX umožňuje za pomoci jednoduchých, předem stanovených výrazů vyhledávat, nebo libovolně manipulovat s textem tak, jak potřebujeme. Regulární výrazy využívají pro vyhledávání v textu metaznaky. Pomocí těchto znaků lze jednoznačně určit, co je

požadováno ze zadaného textu získat. Kromě vlastních metaznaků lze využít i předdefinované skupiny znaků, které jsou definovány pomocí specifických symbolů, například pro bílé znaky je definován zástupný symbol `\w`.

Veškeré metaznaky a symboly se dají kombinovat do požadovaného výrazu. Ten pak může vypadat např. takto, `(19|20)\d{2}`. Pomocí tohoto výrazu je možné získat letopočty od roku devatenáct set do dvou tisíc (1900–2000).

Grafické rozhraní webové aplikace je napsáno v jazycích Cascading Style Sheets, HyperText Markup Language a Java Script. Tyto jazyky, nebo styly, umožní pracovat s daty, které bude vracet databáze do aplikace, a vytvoří požadovaný výstup. Výsledkem jsou přehledné tabulky, které obsahují informace o nahraných životopisech, získaných datech z nahraných souborů, ale i o chybách, které nastaly při procesech, ať už se jednalo o nahrávání nebo parsování životopisů.

2 Cíl práce

Úkolem zvolené bakalářské práce je vytvořit aplikaci, která bude urychlovat práci personálním agenturám, soukromým firmám a dalším institucím, spolupracujícími s uchazeči o práci. Aplikace napomáhá k urychlení práce se zpracováním životopisů. Měla by být schopna sestavit tabulku se základními informacemi z životopisu tak, aby každý životopis nemusel být procházen podrobně, nebo ručně personalisty.

Hlavními prvky pro výstup z životopisu jsou jméno, příjmení, popřípadě titul, pokud jej osoba vlastní, email a telefon. K těmto údajům bude přiložen odkaz, který automaticky stáhne zpracovaný životopis do počítače, pro možnou kontrolu dat.

Urychlení nebude pouze v zobrazení a získávání informací. Velkou výhodou bude schopnost pracovat s různými formáty a typy souborů. Vzhledem k tomu, že se životopis může nacházet v různých souborech, aplikace bude schopna jednotlivé formáty rozeznat a na základě toho se souborem pracovat dále. Těmito typy budou například DOC, DOCX, RTF a PDF.

Kromě celkového urychlení zpracování životopisů umožní aplikace převádět různé typy souborů do jiného typu. Převede zastaralý wordový typ, DOC, na nový DOCX. Toto urychlí celkové zobrazování a zpracování životopisů pomocí lidských zdrojů, pokud to bude potřeba.

Účelem práce je hlavně pomoci při zpracovávání pracovních žádostí, kde je přiložen životopis. Tato aplikace umožní informace zobrazit a umístit do přehledné tabulky a uživatel si je nebudete muset vypisovat ručně, nebo pamatovat.

3 Dostupná řešení

V této části si ukážeme existující řešení, popis jejich fungování a ohodnocení jejich poskytovaných služeb.

3.1 Textkernel

Jedná se o mezinárodní společnost v oblasti umělé inteligence, strojového učení a sémantické technologii, pro hledání pracovních pozic žadatelů o místo.

TextKernel umožňuje zlepšit, zdokonalit, urychlit a zefektivnit personální řízení a celkový náborový proces. Tento proces je vázán na agentury, zabývající se přesunem, přesídlením a získáváním zaměstnanců.

TextKernel je schopen využívat pokročilé syntaktické analyzátory a zkušenosti, získaných z dříve parsovaných životopisů, pro nalezení informací ze životopisu. Tento proces může nastat automatizovaně, nebo na základě lidské interakce. Tímto procesem se zabývá aplikace s názvem „Extract!“. TextKernel nabízí spoustu dalších aplikací, například Jobfeed, který za pomoci Search! a Match!, je schopen shromáždit informace z miliónů pracovních míst na webu a správně je propojit s žadateli o zaměstnání. [14]

Výhoda TextKernelu je strojové učení. Toto učení je využito především u rozsáhlých a podrobnějších vyhledávácích modelů. Dalším důležitým prvkem je sémantické vyhledávání. Jedná se o práci s úrovněmi významu, z různých struktur jazyka. Sémantické vyhledávání rozhoduje o významu informace v dokumentu. Tento význam je založen na sofistikovaném vyhledávácím modelu. [14]

3.1.1 Extract!

Extract! je schopný získávat přesnější informace díky využití umělé inteligence a techniky učení. Využívá podporovaných vlastností, jako je například automatická transformace každého zpracovávaného životopisu, nebo nespočet podporovaných jazyků, včetně češtiny. Dále využívá profily ze sociálních médií, nebo ze stránek pro hledání zaměstnání. [15]

Extract! využívá proces nazvaný Analýza životopisů. Tento postup převádí nestrukturovanou formu textu ze životopisů, nebo profilů ze sociálních médií na strukturovaný formovaný text. Tento převod umožňuje integraci do jakéhokoli softwarového systému. Díky tomu je formátovaný text upraven a připraven pro vyhledávání. Tato analýza dále umožňuje eliminaci ručního zadávání dat do systému. To přináší lepší výsledky vyhledávání a přidání uchazečů na libovolném zařízení. [15]

Využívá vícenásobné extrakce textu, díky které je schopen analyzovat každý životopis, který je zpracováván v rozdělených detailech, dle odstavců, nebo částí v daném životopisu. Díky tomu je přesně umožněno identifikovat všechny informace, od jména, přes pracovní zkušenosti až po koníčky. [15][16]

Extract! je schopen mapovat extrahované hodnoty do taxonomií. Pro společnost, která chce Extract! využívat, je možnost využít standartní taxonomie, nebo si nechat vytvořit specifickou taxonomii. Veškeré práce s taxonomiemi je možné provádět automaticky, a proto software, který společnost získá, je vždy shodný s funkcemi v backendu společnosti. S tímto procesem je také spojena schopnost fuzzy vyhledávání v textu životopisu. Toto vyhledávání probíhá na základě určených domén, nebo seznamu dovedností, které jsou specifické pro daný typ průmyslu. [15] [16]

Extract! dále nabízí ochranu osobních údajů, která zamezuje využití dat třetí stranou. Extract! se zaměřuje na extrakci dat ze souborů typu DOC, DOCX, PDF, RTF,

HTML, TIFF, TXT, XML a EML, dále je možné jej aplikovat na sociální média, jako je třeba Linkedln, Xing, nebo Viadeo. Výstupní data lze získat v libovolném formátu. TextKernel, jako celek, podporuje všechny standartní formáty včetně HR-XML. [16]

3.2 Daxtra

Společnost Daxtra se zabývá zjednodušením pracovního postupu, softwarem pro správu databází, aplikačním softwarem a softwarem pro správu kandidátů. Především jde o vyhledávání a analýzu životopisů, volných pracovních míst a kandidátských řešení. Hlavní náplní jejich aplikací je automatizovat žádosti o zaměstnání a pomoci náborovým firmám k získání uchazeče, o jimi nabízenou práci. [17]

Proces je započat načtením dat z poštovních stránek nebo VMS serverů. Tato získaná data nahraje přímo do databáze, kde proběhne proces eliminace duplicit. V další fázi vyhledá ve všech kandidátních zdrojích a poskytne nejlepší výsledky. [17]

3.2.1 Daxtra Parser

Díky této aplikaci je sníženo procento manuálního zadávání dat prakticky na nulu. Naopak je zvýšeno procento kandidátů a žadatelů o nabízené pozice ve firmě. Daxtra Parser v sobě nese funkce pro vyhledávání v textu, hledání shod, analýzu a celkové reportování. Je schopen extrahovat informace z vícejazyčných dokumentů. [18]

Jeho největší výhodou je celková přesnost pro více různých jazyků, podporuje až čtyřicet jedna (41) různých jazyků. Vytváří přehledná strukturovaná data. Také přidává možnost využít flexibilní a rozšiřitelná schémata přímo na zakázku, s využitím unikátních taxonomií, zlepšuje vyhledávání, analýzu a reportování a v poslední řadě také usnadňuje integraci vývojářů. [17][18]

Podpora je poskytována pro vícero metod integrace. Je podporováno například API REST, nebo SOAP. Má velkou řadu integračních skriptů a mnoho výstupních

formátů. Výstup je možno vypsat přehledně, třeba i ve strukturovaném formátu, jako je XML, nebo JSON. Využívané taxonomie zahrnují spektrum možností od IT komunikace, přes zdravotnictví, až po stavebnictví. [18]

Daxtra Parser podporuje také analyzování profilových účtů na sociálních sítích, sociálních životopisů a také firemních profilů. Také je schopen získávat kontakty elektronických komunikátorů, jako kontaktní informace o Twitteru, Skype, Xingu a dalších. [18]

Daxtra Parser každý měsíc vydává aktualizace, které upravují nativní přesnosti ve všech podporovaných jazycích, rozšiřují taxonomie. S tím je samozřejmě spojeno přidávání podpory pro různé styly a formáty dokumentů, včetně sledování nejnovějších trendů a inovací v náborovém průmyslu. [18]

4 Vývoj aplikace

Ještě před samotným vytvářením aplikace bylo nezbytné rozplánovat, jak bude aplikace fungovat a navrhnout řešení, které bude pokrývat veškeré nezbytné funkční části. S tím je samozřejmě také spojené vybrání prostředku, nebo jazyka, ve kterém bude aplikace vytvářena

Nejprve bylo zapotřebí naprogramovat elementární části aplikace, které jsou však velice podstatné. Tento základ musel obsahovat několik stavebních prvků. Možnost nahrávat soubory předem definovaného typu, kterými jsou RTF, PDF, DOC a DOCX. Velice podstatnou částí, při práci se soubory, je jejich správa. Ta se týká především mazání souborů. V našem případě se bude jednat o mazání automatické nebo ruční. Nedílnou součástí těchto základů také bylo vybrat vhodný server, kde by aplikace mohla pracovat.

Po dokončení základních prvků se pokročí ke složitějším částem. Hlavním předpokladem pro fungování aplikace, byla schopnost parsovat životopisy. Proto bylo nejlepší vybrat formáty, které se dají nejlépe parsovat. Pro parsování byly vybrány dva, ze čtyř předdefinovaných dokumentů. Samozřejmostí je převádět DOC na novější verzi DOCX, dále je pro převod zvolen i RTF, přestože je parsování RTF podobné DOCX. Převedení tohoto typu bylo především kvůli sjednocení dokumentů. Těmi zvolenými soubory nakonec zůstaly PDF a DOCX. Po vybrání dokumentů je zapotřebí vytvořit algoritmus, který převede ostatní dokumenty do sjednocené formy.

Dalším bodem je již samotné parsování, respektive vymyšlení nejlepší cesty, pro dosažení nejadekvátnějších výsledků. Proto je rozhodnuto o využívání veřejně přístupné české státní databáze, se všemi zaevidovanými příjmeními a jmény České republiky. Nejdůležitějšími body, výstupy z aplikace, chceme mít jméno, příjmení, popřípadě titul, email a telefon. Dále se tyto prvky zpracují a připraví pro výstup, respektive zobrazení na obrazovce.

Jedním z nezbytných prvků aplikace je vytvoření srozumitelného chybového výstupu, který bude evidovat veškeré chybné pokusy o vyparsování, nebo nahrávání souborů. Zároveň bude i zobrazovat pokusy, kde nebyl nalezen titul, pro případ, že by se jednalo o chybu. Tyto chybné záznamy bude zapisovat do databáze a špatně vyparsovaná data bude ukládat do připravených pozic pro výstup tak, aby měl uživatel aplikace přesný přehled a mohl případné chybné soubory sám opravit. Mezi chybové výstupy se také budou řadit neúspěšné pokusy o přihlášení. Ty budou evidovány pro přehlednost chyb ze strany uživatelů, nebo z důvodu zachycení útoku na stránku. Útok by se mohl projevit náhodným zadáváním hesel a jmen.

Bezpodmínečnou částí aplikace bude její zabezpečení. Protože se zde pracuje se soubory, které v sobě uchovávají citlivé, nebo soukromé informace, je nezbytné, aby k nim neměl přístup kdokoli. Proto jako přidaná hodnota bude přihlašovací stránka. Tato stránka dovolí přístup pouze oprávněným osobám, které budou mít přihlašovací údaje. Těmito údaji budou jméno a heslo. K nim se také vztahuje úroveň oprávnění, která bude určovat stupeň pohyblivosti po stránce. Pro účel této aplikace dostačují pouze dvě úrovně. Prvním stupněm oprávněním, který bude zaveden jako 0, bude administrátorské oprávnění. To krom práce s životopisy, včetně jejich nahrávání, mazání nebo parsování, bude umožňovat přidávat a spravovat uživatelské účty. Administrátor tedy bude schopen vytvořit nový uživatelský účet, změnit oprávnění libovolnému účtu, smazat jej, nebo mu pouze dočasně zamezit přístup do aplikace. Druhou úrovní, která bude evidována jako 1, bude uživatelská. Tento přístup umožní danému vlastníkovi, aby se mohl pohybovat po stránce a mohl pracovat se soubory. Bude tedy moci soubory nahrávat, parsovat, nebo mazat. Nemá však žádný přístup do uživatelské části, kde by mohl jakkoli spravovat, nebo zobrazovat, popřípadě ani přidávat uživatelské účty. Každý nově vytvořený uživatel dostane jako základní oprávnění uživatelský přístup s číslem 1. Kromě těchto dvou přístupů, která jsou pro každého oprávněného uživatele, zde bude oprávnění správce. Toto oprávnění bude pouze pro jednoho uživatele a bude označeno jako -1. Správce bude schopen, kromě výše zmíněných možností, také uživatelům měnit hesla.

Pro dokonalé fungování a co nejlepší využití všech výše zmíněných bodů bylo stanoveno, že pro vývoj bude jako backend použito PHP s propojením na phpMyAdmin, které bude využívat SQL. Pro frontend bude využito HTML, CSS a Java Script. Aby bylo možno převádět dokumenty, například převod z DOC na DOCX, server, kde bude aplikace uložena, poběží na Linuxu. Na tomto serveru se bude samozřejmě také nacházet databázový server, který bude uchovávat data, získaná z aplikace. Ať už se bude jednat o data o souborech, uživatelích, nebo o chybách.

4.1 Návrh aplikace

Fungování aplikace bylo navřeno na jednoduchém principu. Základní komunikaci zprostředkovává rozhraní navržené tak, aby bylo snadno ovladatelné a bylo jasné, co má uživatel dělat. To znamená že je aplikace velice intuitivní. Grafické rozhraní aplikace bylo napsáno pomocí jazyka CSS a HTML, které tvoří viditelnou část. Zobrazované prvky jsou navrženy tak, aby byl schopen uživatel pracovat na stránce delší dobu. Java Script vytváří podpůrný systém této aplikace a stará se o informování uživatele o krocích, které on sám provedl, nebo je vykonala aplikace. PHP se stará o ukládání, načítání, parsování a celkovou správu všech dat, se kterými aplikace pracuje.

4.1.1 HTML, CSS a Javascript

HTML je zkratka pro hyper text markup language, což je zkráceně řečeno hypertextový značkovací jazyk. Používá se nejčastěji v kombinaci s CSS, pro vytváření vizuální stránky webové aplikace. HTML značky, nebo také tagy, lze rozdělit dle několika kategorií. Těmi jsou strukturální, popisné a stylistické tagy. Tyto tagy mohou být jak párové, tak i nepárové. Párové jsou ty, které jsou rozděleny na dvě části, například `<a href=“google.com“> Odkaz </a>`. Nepárové mají pouze jednu uzavřenou část, jako je třeba `<br />`. [4][5]

CSS je zkratka slov Cascading Style Sheets. Jsou to kaskádové styly, jedná se o druh kódu, který v sobě drží celkový vzhled aplikace, přilepený na jednotlivých prvcích HTML. Pro připojení stylu na HTML tag se nejčastěji vytváří třídy, na které pak navazují různé vzhledové prvky, od barvy písma až po dekoraci celé stránky. [5][6]

Javascript je jazyk používaný při tvorbě webových stránek. Jeho použití je přímo v HTML kódu v párovém tagu javascript, nebo je možnost jej vepsat do samostatného souboru. Jedná se o klientský script, což znamená že je odeslán do

prohlížeče, kde je také zpracováván. JavaScript běží přímo na zařízení uživatele. [5][8]

4.1.2 PHP

PHP je Hypertextový preprocesor, původně Personal Home Page, je skriptovací programovací jazyk. Určen je především pro programování dynamických internetových stránek a webových aplikací. Nejčastěji se váže na HTML a CSS, pro zobrazování informací a také na databázi, odkud bere data, které dále zpracovává. [5][7]

4.1.3 MySQL

Databázový systém relačního typu DBMS (database management system). Jedná se o multiplatformní databázi, která využívá jazyka SQL. Nejčastěji je spojena s prostředím phpMyAdmin, které bude také využito v naší aplikaci. Díky komunikaci přes SQL je možné vytvářet tabulky a pracovat se všemi daty, která budou vytvářet vstup a výstup aplikace. [5][9]

4.2 Základní fungování aplikace

Na začátku vyvíjení aplikace bylo potřeba vyřešit základní fungování aplikace, které spočívá v nahrávání a celkové správě souborů.

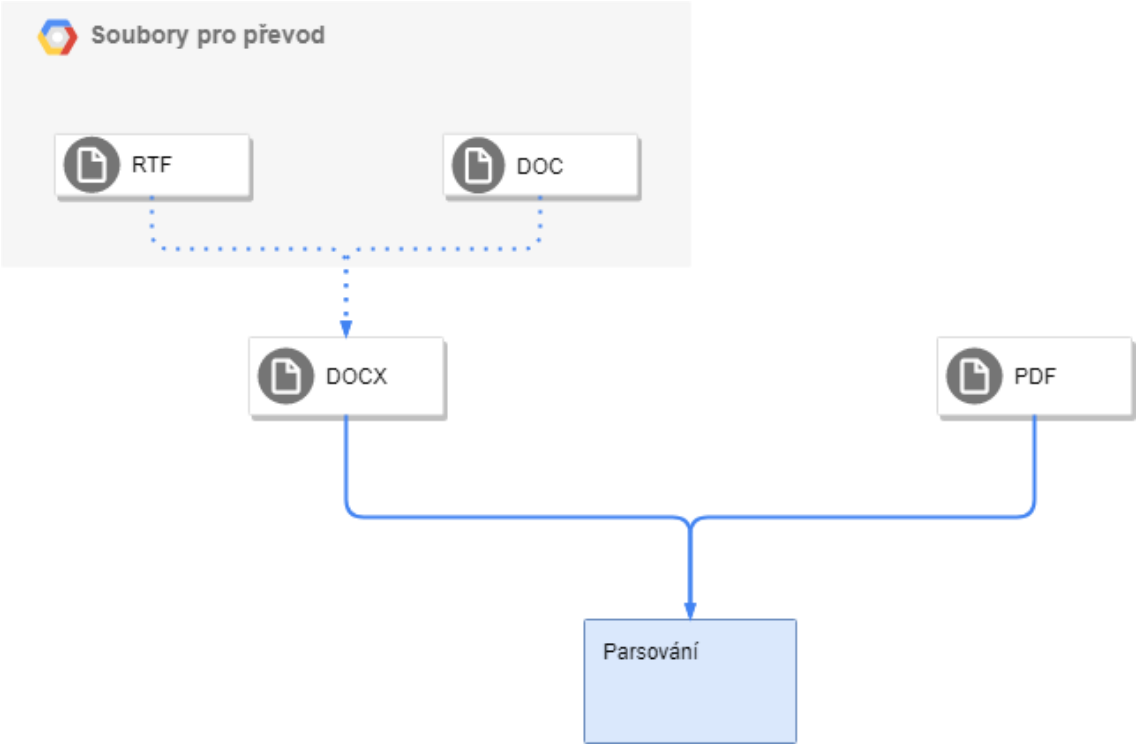
Veškeré úkony, jako je nahrávání, pojmenovávání, mazání nebo parsování, bude probíhat na straně serveru, za pomocí PHP. Soubor bude vybrán uživatelem, následně bude validován. Validace z naší strany spočívá pouze v kontrole přípon souborů, které jsme schopni zpracovat. Pokud se bude jednat o soubor, který by se dal sjednotit, DOC nebo RTF soubor, bude provedena jeho úprava a sjednocení na jednotný typ souboru. Byl zvolen právě DOCX. Pojmenování souboru bude probíhat automaticky podle posledního záznamu v databázi tak, aby byl název vždy unikátní. Tudíž nám vznikne i unikátní identifikátor pro možné mazání nebo parsování souborů.

Další součástí základního fungování je práce s uživatelskými účty. Zde se budou uživatelé přidávat, upravovat a posléze se také bude validovat jejich přihlašování. Zde budeme kontrolovat správnost jména vůči zadanému uživatelskému textu a samozřejmě správnost hesla, které je s tímto jménem spojeno. Krom těchto dvou prvků bude prováděna validace toho, zdali a jaký má uživatel povolen přístup do aplikace. Po přihlášení proběhne porovnání s úrovní uživatele a na základě toho mu bude zobrazena jeho část aplikace.

Veškeré tyto informace se budou ukládat do databáze, aby bylo možné s nimi pracovat. Každý úkon bude mít vlastní tabulku pro přehlednost. Nahrané i vyparsované soubory budou navzájem propojeny pomocí jednoznačných identifikátorů pro získávání podstatných informací.

4.2.1 Dokumenty

Základními dokumenty, které budeme parsovat, budou dokumenty typu DOCX a PDF. Tyto dokumenty byly zvoleny pro jejich práci s textem, kde oba používají jasně specifikované formátování textu. Zbylé dokumenty, které se budou do databáze moci nahrát, jako je DOC a RTF budou převáděny právě na DOCX, který má jasně specifikované XML. Krom této specifikace bylo DOCX zvoleno pro dostupnost funkcí pro převod souborů právě na tento typ. Open XML, ve kterém je DOCX zapsán, je velice užitečný právě proto, že se dají snadno odstranit tagy obsahující pouze stylování textu. Díky tomu dostaneme pouze informace, které uživatel do životopisu zapsal ručně.



Obr. 1 Návrh struktury dokumentů

4.2.2 XML

XML je, podobně jako HTML, značkovacím jazykem. Jedná se o Extensible Markup Language, alias Rozšiřitelný značkovací jazyk. Zejména je doporučenou specifikací pro W3C, jako všeobecně použitelný značkovací jazyk. Toto značí, že XML není předdefinováno, a proto je nutné, aby byly značky specifikovány neboli si musíte vytvořit vlastní. Nejčastěji se používají pro sdílení dat, například na internetu ve FEED, nebo v DOCX dokumentech. [10]

4.3 Převod dokumentů na DOCX

Pro převod mezi soubory, pro náš případ převod souborů typu RTF a DOC na DOCX, byla zvolena aplikace LibreOffice. LibreOffice je velice podobná Microsoft Office, s výhodou multiplatformní instalace. Ta nám umožní používat tento program jak na Windows, tak i na Linux serverech, nebo desktopových zařízeních. Stejně jako Microsoft Office má prezentace (Impress), tabulky (Calc) a také i textové editory (Writer). LibreOffice má pro nás však přidanou hodnotu v podobě Linuxových příkazů, které dovolují vytvářet, libovolně editovat, nebo také převádět souborové typy na jiné.

Právě pro tento převod souboru je vytvořen příkaz `soffice --headless --convert-to docx.` Tento příkaz nám umožňuje vzít námi dva zvolené typy dokumentů, RTF a DOC, a převede nám je na DOCX. [11]

4.3.1 Linux

Linux je jádro operačního systému, které spolu s ostatními programy tvoří operační systém. Jeho obrovskou výhodou je zdarma sdílená licence, která umožňuje libovolně stahovat, distribuovat a upravovat. Upravování je možné pouze dle určitých kritérii.

Pro tuto aplikaci je Linux užit na serveru jako operační systém, na který je aplikace nahrána. Hlavním důvodem, je jíž výše zmíněný, převod souborů přes Linuxové příkazy. [12]

4.4 Parsování

Parsování dokumentů probíhá pomocí regexových příkazů v PHP. Ze souboru je vytažen celý text, který je v něm uložen. Tento text dostaneme pomocí již naprogramovaných tříd, které jsou volně ke stažení. Pro parsování je nadále použita pouze první část celého životopisu. Tou je hlavička, ve které jsou specifikované veškeré prvky, které má aplikace zjišťovat. Tato část je definovaná prostřednictvím počtu znaků, který je stanoven dle maximální délky textu, ve které se mohou nacházet hledané prvky. Tato délka nám ořízne vytažený text tak, že nám zůstane pouze krátký text, který obsahuje námi hledané prvky. To urychlí samotné vyhledávání a práci s textem.

Výsledný text je procházen a za pomoci několika regexových příkazů a databázových informací, jsou data získána a připravena pro následné zpracování. Každý takový příkaz je přesně definován právě pro jeden hledaný prvek. Proto máme pět regexových příkazů, které jsou specifické pro každou hledanou informaci, tedy pro titul, jméno, příjmení, telefon a email.

4.4.1 Regex

Regex, neboli regulární výraz. Jedná se o speciální řetězec znaků, který představuje určitý vzor pro posloupnost znaků. [2][13]

Tyto řetězce se nejčastěji používají pro vyhledávání, rozpoznávání nebo dekódování dat zadávaných ve formuláři, nebo v našem případě pro získání určité znakové posloupnosti, které stanovují potřebné prvky, email, jméno a tak dále.

Regex je možno použít téměř ve všech programovacích jazycích, ale nevztahuje se pouze na programovací jazyky, je možné je najít i ve spoustě aplikací, například v aplikacích pro Linux, nebo Unix. [2][13]

Regex má několik druhů. Nejznámějšími z nich jsou například Perl-compatible regulární výrazy nebo POSIX regulární výrazy. Většina regulárních výrazů je odvozena od POSIX. [13]

4.5 Tvorba aplikace

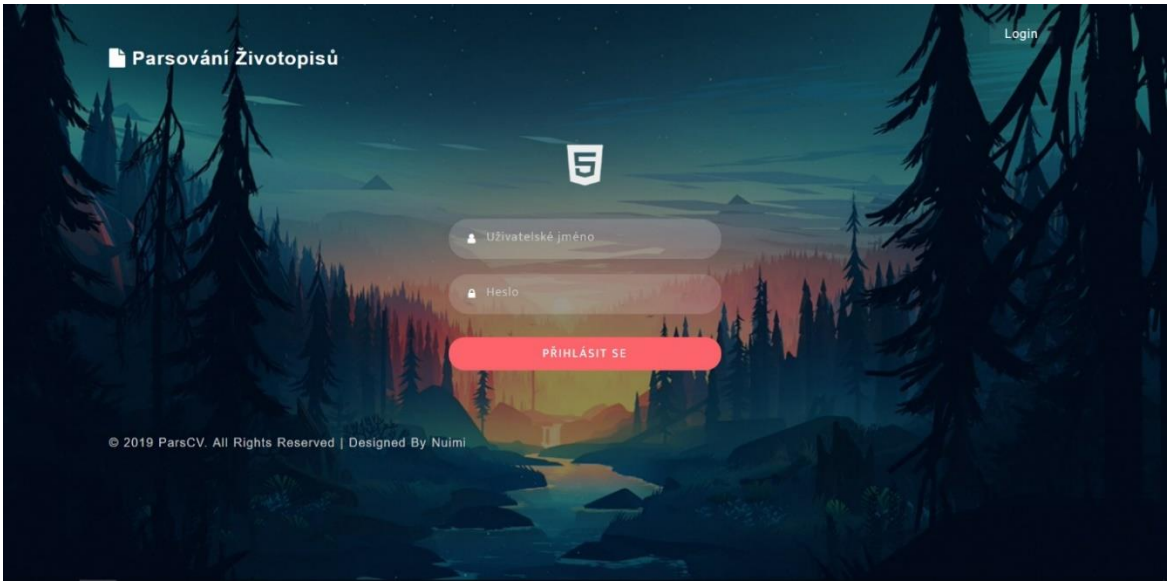
V této části bude přesně popsáno, jak byla aplikace vyvíjena, chyby při tvorbě, použité knihovny pro práci s dokumenty, vytvořené rozhraní, komunikace s databází a samozřejmě výstupy, kterých se snažíme dosáhnout.

Jak už to většinou bývá u webových aplikací, i tato je rozdělena na backend a frontend. Frontend má v našem případě přidanou funkci, která zajišťuje komunikaci s uživatelem a informuje ho o stavu probíhajících prací. Tudíž je velice spjat s backendem a využívá jeho informací pro informování uživatele. Samozřejmostí pro frontend je vykreslování informací získaných z backendu a celkové vykreslení samotné webové aplikace. Hlavní náplní backendu je pracovat s obdrženými životopisy. Doprovází je po celou dobu jejich životního cyklu. Ať už se jedná o nahrání životopisu, jeho vyparsování, nebo samotné uživatelské, nebo automatické smazání. Kromě práce s životopisy se bude backend starat o uživatelské účty. Hlavní náplní bude přidávání uživatelů a kontrola přihlašovacích údajů. Backend se dále bude starat o upravování samotných uživatelských účtů, kde administrátor může změnit jejich úroveň oprávnění, nebo uživatelům zablokovat přístup do aplikace. Pro správce je zde ještě přidaná hodnota a tou je změna hesla. Samozřejmostí je také komunikace s databází, díky ní je backend schopen reprezentovat veškeré potřebné informace, požadované uživatelem. Ať už se jedná o získávání informací o souborech, nebo o dotazování se na data o uživatelích.

4.5.1 Frontend

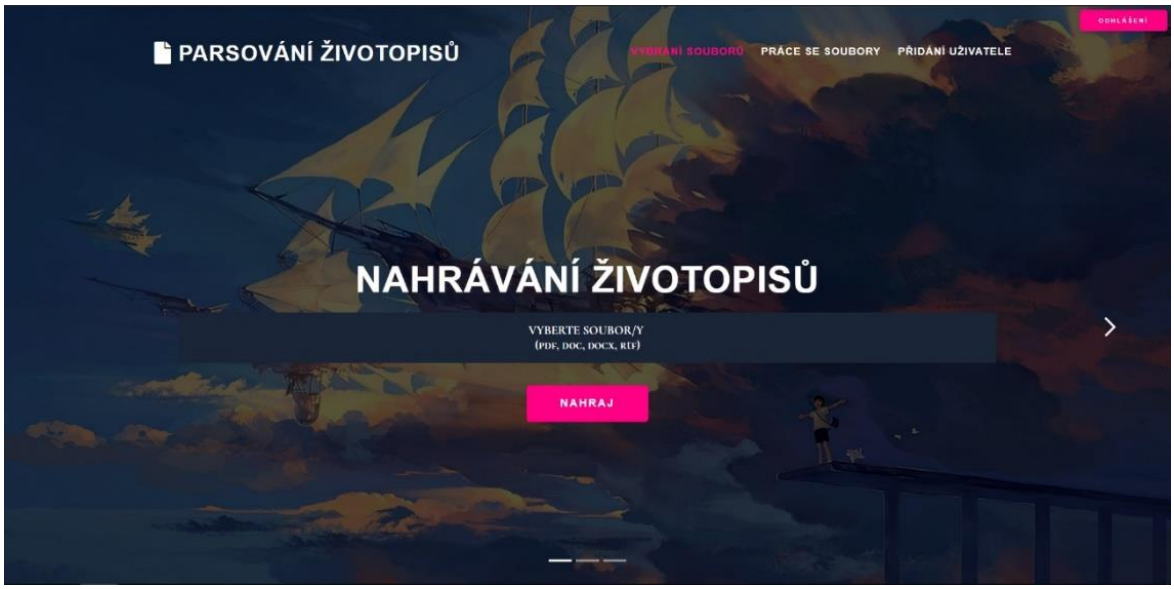
Celkový vzhled aplikace, jak už bylo výše zmíněno, je vytvořen pomocí HTML a CSS. Tyto dva prvky tvoří celkovou strukturu aplikace. Pro vylepšení vzhledu je využita CSS knihovna s názvem Bootstrap, která se stará a modernější vzhled aplikace.

První stránku, kterou uživatel uvidí při načtení aplikace, je přihlašovací stránka. V našem případě se jedná o obyčejnou stránku s formulářem, která umožňuje uživateli přihlásit se do aplikace. Krom základního formuláře jsou zde samozřejmě informační prvky, které upozorňují uživatele. Toto upozornění se zobrazí v několika okamžicích. Upozornění se zobrazí, pokud uživatel zadá neexistující jméno, zadá chybné heslo, nebo má zamítnutý přístup do aplikace.



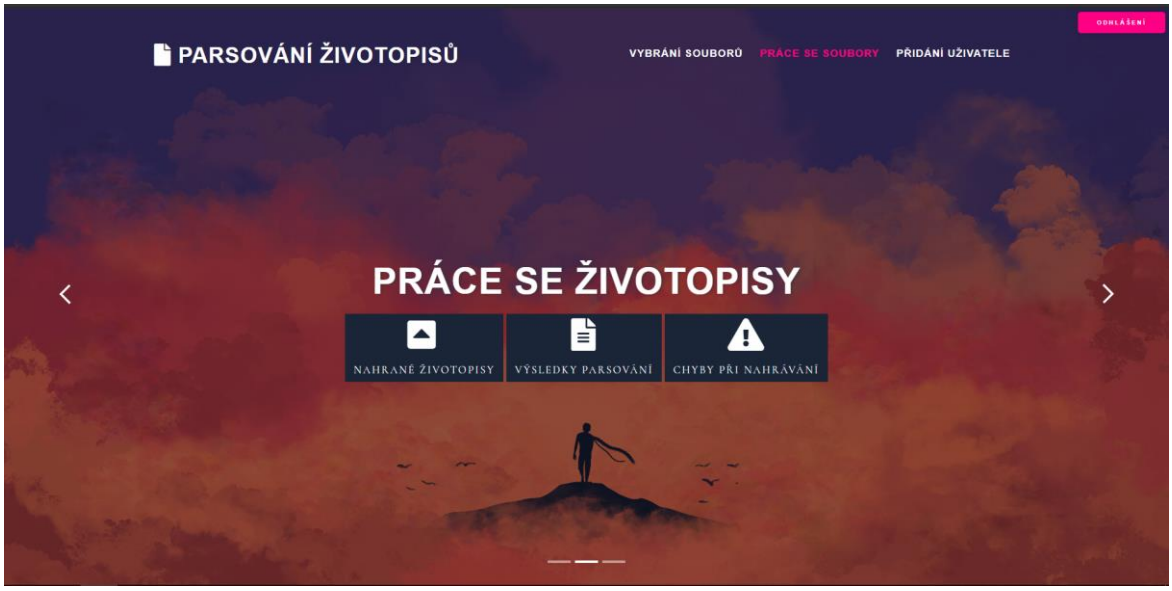
Obr. 2 Přihlašovací stránka aplikace

Po přihlášení do aplikace se uživateli zobrazí úvodní obrazovka. Ta pracuje pouze s jedním oknem, kde se nachází formulář s tlačítky pro nahrání souborů. Další částí této stránky jsou modaly, vyskakovací okna, která se zobrazí pomoci JavaScriptových knihoven, přesněji pomocí knihovny jQuery a bootstrap.js, které napomáhají stránce fungovat bez obnovy. Na této stránce je uživatel schopen vybírat soubory, které chce nahrát, případně pouze prohlédnout a další funkcí je přímo nahrání do aplikace.



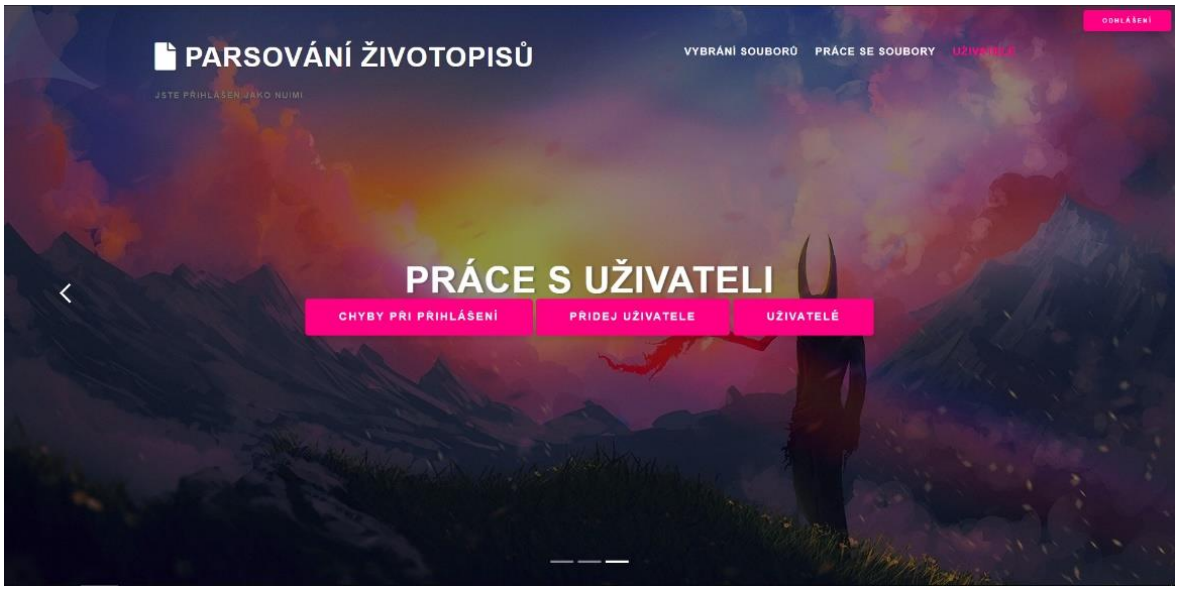
Obr. 3 Titulní stránka aplikace.

Po přesunutí na další okno se dostaneme do zobrazení několika tlačítek, které nám zobrazí výše zmíněné modaly. V těchto modalech jsou zobrazeny veškeré informace o souborech. Jsou tu soubory, které by mohly být použity pro parsování, nebo také výstupy z již vyparsovaných souborů. Uživatel přehledně vidí, co je již zpracováno, co je potřeba zpracovat, popřípadě se může podívat na chyby, které mohly nastat při parsování. Také jsou zde informace z logování chyb, ať už se jedná o chyby při nahrávání souborů, nebo při samotném parsování. Veškeré tyto informace jsou přehledně uloženy v tabulkách tak, aby se uživatel orientoval v každém záznamu.



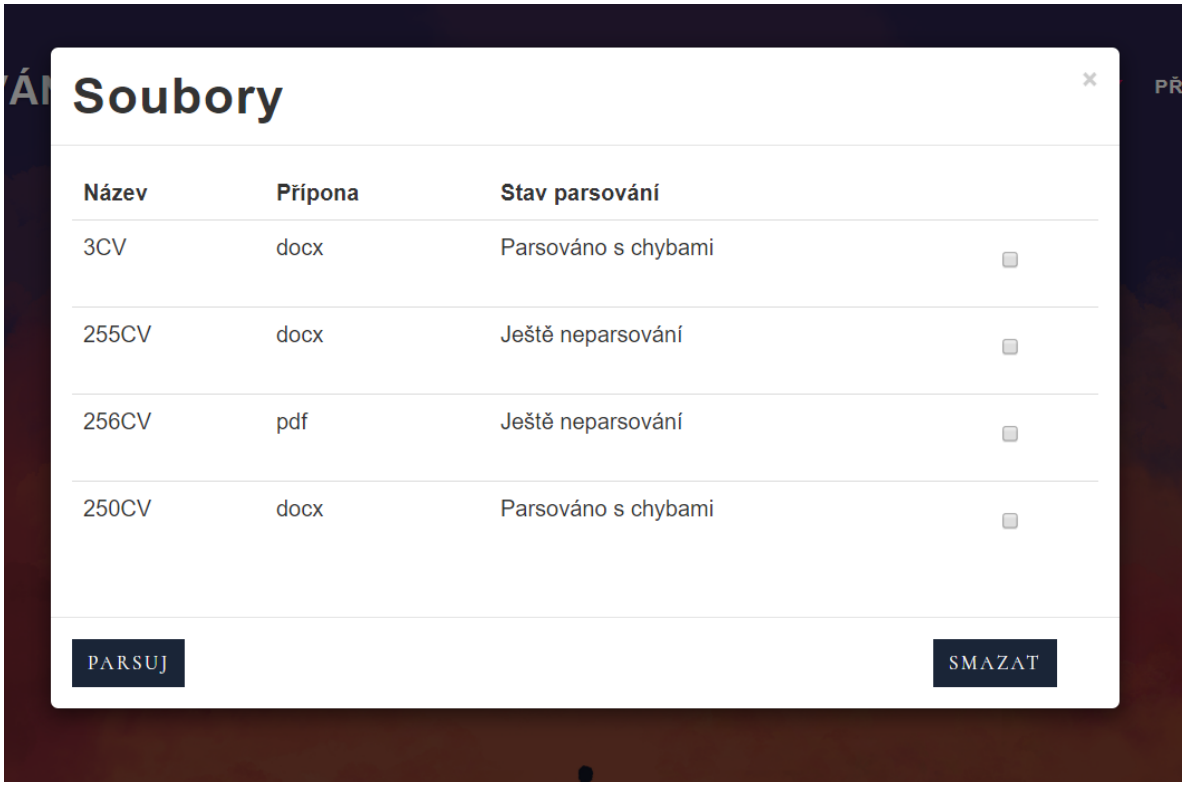
Obr. 4 Stránka pro práci s životopisy

Do tohoto posledního okna se může dostat pouze uživatel s oprávněním správce, nebo administrátor. Je zde několik tlačítek, které nám umožňuji pracovat s uživateli. Správce nebo administrátor je schopen uživatele přidávat, nebo evidovat. Také zde najdeme informace o chybných přihlášeních. V tomto modalu administrátor, nebo správce, může vidět přihlášení, které se nepovedlo. Může se jednat buď o chybně zadané jméno nebo heslo při přihlašování.



Obr. 5 Stránka na přidávání uživatelů

Celkový vzhled je koncipován tak, aby uživateli bylo jasné, co má kde dělat a vše bylo přehledné, tedy User Friedly. Aplikace je v každém okamžiku, kdy něco provádí ať už uživatel nebo aplikace samotná, schopna komunikovat. Uživatel je prostřednictvím hlášek informován o samotném průběhu činnosti. Ku příkladu o prováděném parsování, o nastalých chybách, nebo o počtu úspěšně nahraných souborů. Veškeré tyto prvky jsou zobrazeny v modalech za pomoci Javascriptu.



Obr. 6 Zobrazení informací, nebo tabulek v modalu

4.5.2 Backend

Backendová část aplikace je napsána v jazyce PHP. Pro držení veškerých informací a dat je využito databázové aplikace phpMyAdmin, která komunikuje prostřednictvím SQL příkazů s daty v databázi. Díky těmto dotazům je PHP schopno se všemi uloženými daty, které dotaz navrátí, pracovat, zobrazovat je uživateli a

upravovat jejich hodnoty. PHP je také za pomoci své funkcionality schopno prostřednictvím specifického příkazu exec, volat příkazy z příkazového řádku serveru, na kterém aplikace běží. To znamená, že cokoli napíšeme do funkce exec nám vytváří systémový příkaz, ze kterého posléze dostaneme výstup. Ten jsme pak schopni zobrazit uživateli, nebo jej libovolně reprezentovat. V našem případě jde o specifický příkaz v Linuxu. Ten nám zajistí komunikaci s aplikací LibreOffice, která nám bude nápomocna při převádění formátu DOC a RTF na formát DOCX. Tuto PHP funkci využijeme pouze s jedením parametrem a tím je samotný příkaz, který chceme nechat proběhnout. Tento příkaz je zapsán ve formátu textového řetězce. V našem případě bude použit tento dotaz, který je specifikován v LibreOffice: `soffice --headless --convert-to docx –outdir soubory/ funkcni/parsovani/soubory/jméno.doc`. Ten nám říká, že každý soubor, který projde podmínkou pro nahrání a spadá do souborů pro možné převedení typu, bude konvertován na DOCX a jeho výsledná souborová struktura bude uložena ve složce soubory. Soubor pro převod mezi typy je uveden na konci dotazu, je jím tedy jmeno.doc. K souboru se musí uvádět celá cesta od souboru kde je zapsán a spuštěn dotaz. V této ukázce je tou cestou funkcni/parsovani/soubory/. Na serveru se však používá relativní cesta, které nás odkáže na dočasné umístění souboru.

4.5.2.1 Databáze aplikace

Databáze je v tomto projektu velice zásadní, ať už se jedná o samotné ukládání informací, které se mohou týkat souborů, uživatelů a chyb, tak je i nedílnou součástí při parsování. Jak bude zmíněno níže, při několika postupech parsování, v našem případě při parsování jména, příjmení nebo titulu, bude využito databáze jako samostatného prvku pro porovnávání textu s databázovými záznamy při vyhledávání.

Při parsování jednotlivých prvků jsou potřeba tabulky, které drží záznamy o existujících jménech, příjmeních anebo titulech. Tyto záznamy jsou získány a

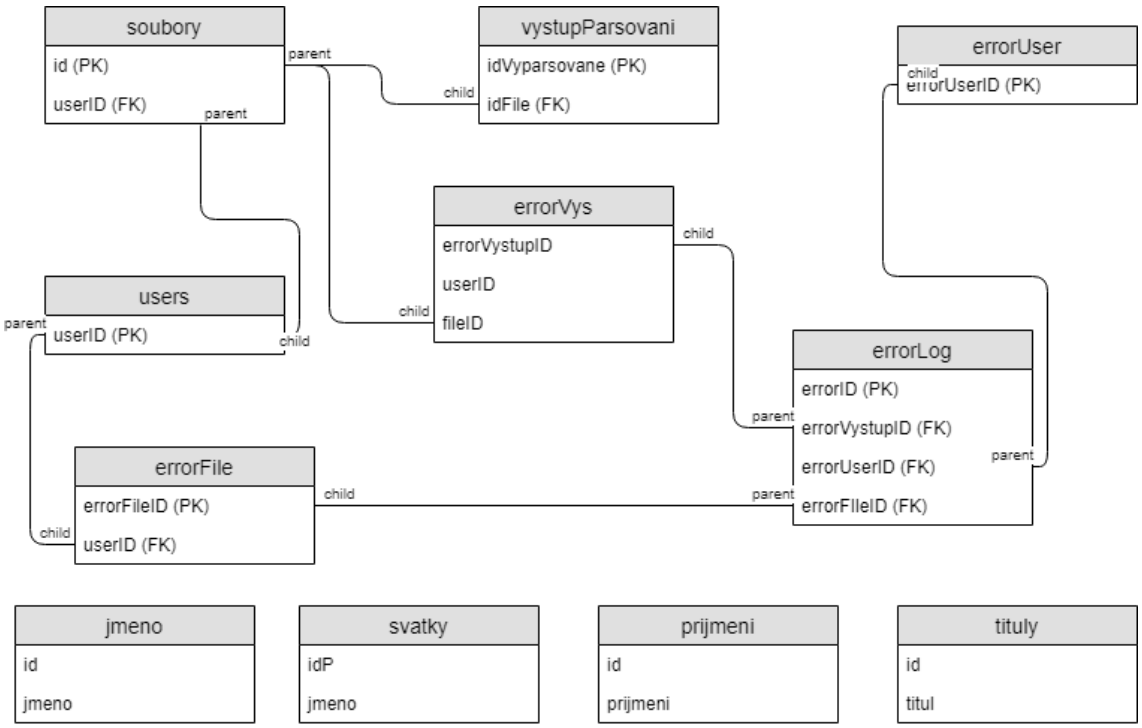
zaznamenány na území České republiky. Naším zdrojem pro záznamy o jménech budou dvě databáze. Tou první je klasická databáze jmen, která je obsažená v kalendářích. To nám poskytne prvotní nástin a nejrychlejší vyhledávání jmen v našem zvoleném textu. V našem případě se jedná o tabulku s tří sta sedmdesáti pěti (375) záznamy, kde jsou obsažena veškerá česká jména, která můžeme nalézt v kalendáři, rozdělených do jednotlivých řádků. Pokud se žádné z těchto využitých jmen neshoduje, tak se postupuje na další možnost hledání jmen. Jako další je tedy využita tabulka pro jména, která jsou čerpána z databáze jmen přístupné z veřejného archivu Ministerstva Vnitra České republiky. [20] První tabulka tedy určuje den svátku každého běžného jména na území České republiky. V druhém případě se jedná o všechna jména, používaná u nás, včetně těch běžných. V našem případě tedy jde o tabulku o velikosti šedesát devět tisíc pět set třicet šest (69 536) řádků, kde každý řádek obsahuje jedno jméno. Další tabulka je spjata s příjmeními. Stejně, jako tomu je v druhém případě se jmény, je využito databáze pro příjmení z archivu Ministerstva Vnitra České republiky. [20] Protože pro příjmení, jak bude zmíněno níže, máme snazší postup vyhledávání. Jedná se o způsob, kdy se hledá další slovo po jméně. Hledání prostřednictvím tabulky je až jako druhá možnost pro hledání příjmení. Důvodem je její o něco delší doba hledání a větší paměťová náročnost. Důvodem větší časové náročnosti je počet záznamů v tabulce, který je dvě stě osmdesát pět tisíc tři sta padesát osm (285 358). Poslední podstatnou tabulkou pro nás je tabulka titulů, která je převzata z veřejně dostupných informací o titulech, kterých je člověk schopen dosáhnout. Její velikost je třicet tři (33) záznamů, kde je každý řádek koncipován jako jeden titul.

Databáze je v zásadě tedy rozdělena do tří částí. První, již zmíněnou částí, jsou tabulky pro parsování. Tyto tabulky obsahují pouze záznamy, potřebné pro parsování životopisů a není tedy potřeba jejich jakéhokoli propojení. Ať už mezi sebou, nebo s jinou tabulkou.

Druhou tabulkovou částí jsou datové tabulky. Tyto tabulky drží informace o souborech, které jsou nahrané. Dále také o jejich umístění a samozřejmě o typu souboru. K těmto tabulkám je také přidána informace o uživateli, který je nahrál,

nebo parsoval. Tento záznam je přidán toho z důvodu, aby uživatel viděl pouze životopisy, které nahrál, nebo se kterými pracoval. Díky tomu je omezena možnost krádeže citlivých informací. Také jsou zde tabulky, které drží informace o parsování. Je zde přesně vidět, kolik informací bylo získáno u daného životopisu. Bezprostřední součástí datových tabulek je tabulka s uživateli. Zde můžeme najít uživatelská jména, hesla, úroveň oprávnění a samozřejmě také, zdali má daný uživatel přístup do aplikace. Tato tabulka, jak již bylo zmíněno, je spjata se soubory pro kontrolu, který uživatel co nahrál. Tím nám však může vzniknout redundance dat, protože vícero uživatelů bude moci nahrávat stejné soubory. To nám však nevadí, protože každý uživatel uvidí pouze svá data a své nahrané životopisy.

Třetí částí, kterou bereme v potaz v databázi, je logovací část pro zaznamenávání chyb. Tyto tabulky obsahují informace o nastalých chybách v aplikaci. Tyto errory se týkají především nahrávání souborů, kde můžeme vidět, proč soubor nebyl nahrán. Také jsou zde chyby, které nastaly při parsování. Ty zobrazují, které informace při parsování byly, nebo právě nebyly, nalezeny. Tato data jsou přímo spojena s daným souborem, aby bylo vidět, pro který soubor chyba nastala. Součástí těchto logovacích tabulek je také hlášení o chybných přihlášeních uživatele. V této tabulce se vyskytují data o tom, jaká chyba nastala, zdali šlo o špatné heslo, nebo o chybně zadané jméno. Všechny tyto chybové údaje jsou uvedeny s datumem, aby bylo přesně dané, kdy se udály. Každá chybová hláška týkající se souboru, je spjata pomocí identifikátoru s uživatelem, tak aby každý viděl pouze svá chybová hlášení.



Obr. 7 Struktura Databáze

Úplná struktura databáze viz Příloha č. 1

4.5.2.2 Postup parsování

Hlavní náplní aplikace je parsování životopisů, o které se bude starat příkaz pracující s regulárními výrazy. Tyto výrazy byly vysvětleny výše. Pro jejich používání má PHP vlastní funkci, tou je `preg_match_all`. Tato funkce je v našem případě použita jako čtyř parametrická. Pro čtvrtý parametr, který je nepovinný, použijeme „flag“, pro získání pouze prvku, který si žádáme. Tento čtvrtý parametr může nabývat několika hodnot, my z nich použijeme `PREG_SET_ORDER`, který nám upraví pole výsledků tak, jak jej budeme následně potřebovat pro výstup do databáze.

Funkce `preg_match_all` potřebuje jeden, pro nás velice důležitý parametr a tím je samotný text, ve kterém má vyhledávat. Tento text se získává dvěma způsoby, které jsou rozděleny dle typu souboru, který parsujeme. V našem případě se jedná pouze o dva typy souborů, které lze parsovat, neboť ostatní jsou převáděny na jeden

z těchto typů. Prvním typem, o který se zajímáme je DOCX. Z tohoto wordovského souboru získáváme text pomocí vytvořené, volně stažitelné, třídy jménem DocumentParser.

DocumentParser [21] nám bezpečně umožní získat text z několika typů souborů. Pro nás je však výhodnější typy mezi sebou převádět, pro lepší výsledky a celkovou ucelenost formátů.

Z DOCX se v této třídě stane ZIP soubor, který v sobě ukrývá veškeré potřebné soubory a v jednom z nich se nachází požadovaný text. Z tohoto ZIP souboru je však možno získat i profilovou fotku, pokud by byla přiložena, anebo by pro nás byla zásadní. Při procházení tohoto ZIP souboru hledáme dokument s názvem „document.xml“. Tento dokument v sobě drží vše, co uživatel napsal, nebo upravil ve Wordu, včetně předdefinovaných stylů záhlaví a dalších informací. Jak nám název napovídá, soubor je strukturován pomocí XML tagů. Tyto tagy v sobě drží informace o vzhledu, o umístění textu, o jeho struktuře a několika dalších informací o textu. Proto se pomocí jednoduchých pravidel zbavíme všech tagů, které patří k těmto stylům. Díky tomu nám zůstane čistý text bez vzhledu a struktury. Text tedy bude pouze v jednom odstavci, nebo řádku. Záleží na způsobu, jakým uživatel psal životopis. Z tohoto textu si vezmeme část nutnou pro parsování, tu určíme maximální možnou délkou, kterou chceme získat. To provedeme pomocí PHP funkce `substr`. Ta pro své fungování, v našem případě, potřebuje tři parametry. Tím prvním je text, který chceme upravit, text získaný z životopisu. Dalšími dvěma prvky jsou čísla, která určují začátek a konec výsledného textu. My budeme pro začátek používat nulu a jako konec výsledného textu bude osmistý znak. S textem, který získáme, dále pracujeme.

Dalším typem, o který se zajímáme je PDF. Pro získání textu z tohoto typu souboru využíváme volně dostupný software, v našem případě se jedná o PHP třídu s názvem PdfParser od Smalot [22]. Tato třída se nám pomocí specifických funkcí postará o získání stejného holého textu, jako je tomu v případě DOCX dokumentu. U PDF se však nedá zbavit úplně všech tagů. Proto se někdy stane, že námi zvolený text

je stále obalen, proto na to musí být při vyhledávání brán zřetel. V tomto případě se nemusí dočasně měnit typ dokumentu, jako tomu bylo u DOCX, zde se pouze pomocí transformací a vyhledávacích funkcí uchopí celý text, který je obsažen v PDF souboru a podobně jako tomu bylo u DOCX se zbaví veškerých vzhledových vlastností a zůstane nám pouze text, se kterým dále můžeme pracovat a parsovat jej. Opět odřízneme horní část, stejným způsobem jako tomu je i DOCX a pošleme ji k dalšímu zpracování, kterým je samotné parsování.

Parametr, který se krom textu posílá do naší parsovací funkce, je samotný regulární výraz. Tento regulární výraz je tvořen sekvencí znaků, symbolů nebo textových řetězců, které chceme získat. Při správném sestavení regulárního výrazu tak můžeme libovolně hledat v textu. Aby nám byla ulehčena práce, tak regulární výraz má speciální zástupné znaky. Každý tento znak je využíván pomocí zpětného lomítka. Toto lomítko se také používá, pokud chceme hledat nějaký znak z předem definované stupnice. Pokud bychom měli nějaký dlouhý text a v něm měli tečku, nemůžeme regulární výraz napsat takto `/./`. Důvodem je to, že symbol tečky je v regulárním výrazu znakem pro jakýkoli znak, číslici nebo symbol. Pokud bychom chtěli tedy najít pouze tečku, celý výraz by vypadal takto `/\./`. Díky tomuto zápisu by nám regex vrátil veškeré tečky ve vloženém textu. V regulárním výrazu značí lomítko začátek nebo konec hledaného výrazu. Za ukončovací lomítko je možnost doplnit regex vlajky, které nám mohou pomoci ke změně chápání dotazu, nebo k samotnému hledání v textu. Základní vlajkou, která je vždy předdefinovaná je vlajka `g`, která je zkratkou pro výraz global. Ta nám říká, že se bude hledat vše a že se nevrátí pouze první shoda. Regex má velké množství vlajek, například `i`, insensitive, která nám říká, že výraz není citlivý vůči velkým a malým písmenům, také `x`, extended, ta ignoruje veškeré bílé znaky, kterými jsou třeba mezery nebo zalomení řádku, nebo vlajka `m`, multi line, ta nám přidává znak ^ pro začátek a `$` pro konec řádku u hledaného výrazu. Ale, jak již bylo zmíněno, velice důležité a podstatné jsou předdefinované skupiny znaků. Tyto skupiny nám právě specifikují podstatné informace tak, abychom je nemuseli vymýšlet sami. Jedou takovou skupinou je třeba skupina `\d`, která bere v potaz pouze veškeré číslice, dále je zde třeba `\w`, ta bere do hledání veškeré znaky, tudíž je ekvivalentem k regulárnímu

zápisu `/[a-zA-Z0-9_]/`. Veškeré znaky napsané v hranatých závorkách, tak jak bylo znázorněno, nám vytváří vlastní skupiny znaků pro hledání. Kromě těchto skupin nebo metaznaků, máme také kvantifikátory. Ty nám značí, kolikrát se může hledaný prvek opakovat. Hvězdička představuje neomezený počet opakování, nebo `{m,n}`, takto zapsaný výraz nám značí, že hledaná věc se může opakovat za sebou minimálně m krát a maximálně n krát.

Kromě dvou výše zmíněných parametrů, kterými jsou text, ve kterém se bude hledat a regulární výraz, je velmi podstatným parametrem proměnná. Tato proměnná musí být typu pole, to proto, aby se do ní mohl ukládat celkový počet nalezených výrazů. Z tohoto pole si pak dále můžeme vyfiltrovat, nebo pouze získat, námi požadovaný výstup. U PHP nemusíme specifikovat datový typ, proto není potřeba nastavovat proměnou jako pole, ale ničemu se tím ani neuškodí.

Parsovat životopisy budeme dle dvou kritérií. Prvním kritériem je, zdali se jedná o hodnotu, kterou lze někde vyčíst, nebo získat. Toto kritérium se vztahuje zejména k vyhledávání hodnot typu jméno, titul nebo příjmení. To z toho důvodu, že tyto hodnoty se kvůli rozmanitosti, spoustě různých druhů životopisů a použitých jazycích ve jménech, nedají najít dle jasných a hodnot, které lze přesně specifikovat regulárním výrazem. Proto pro postup tohoto vyhledávání jsou vytvořeny dvě možnosti, jak je vyhledávat.

První možností, která se vztahuje na jméno a příjmení je možnost vzájemného vyhledávání za pomoci regulárních výrazů. To znamená, že pokud je nalezeno ve vybraném textu pro parsování slovo, hledané jako jméno, hledají se slova hned za tímto nalezeným textem. Toto hledání nepřestane, dokud nenastane zalamovací část, respektive se jedná o bílý znak, který nám naznačuje další řádek textu. Pokud se jedná o PDF, je možnost zastavit hledání dalšího slova pomocí tagu, který zůstane v textu po vytažení textu. Následně jsou tato slova, dle počtu, rozdělena na jméno a příjmení. Toto je však až poslední z variant, která je aplikována na vyhledávání příjmení. Přednější variantou, která probíhá jako první je databázové vyhledávání. Pomocí tabulky jmen je vyhledáno jméno ve vybraném textu pro parsování a to

velice jednoduchým regexem, `/\s?'.$jmeno.'\s/`. Tento příkaz nám zajistí, že se projdou všechna jména, která dostaneme z databáze a vyhledají se v textu. Toto procházení je zajištěno pomocí cyklu, který probíhá tak dlouho, dokud neprojde každý záznam z databáze, který vrátil dotaz. Pokud je nalezena shoda, začne automaticky probíhat výše zmíněný proces. Pokud další slovo neexistuje, nebo není slovo nalezeno, nastává stejné hledání, avšak za pomoci tabulky příjmení, pro hledání příjmení vlastníka životopisu. Toto databázové vyhledávání je stejně aplikováno i na vyhledávání titulů, pouze se změnou tabulky. Regulární výrazy pro tento typ hledání mají specifické dvě hodnoty. Tou první je znak pro předdefinovanou skupinu bílých znaků, kterým je `\s`. Ten nám tedy říká, že se za hledaným slovem musí vyskytovat mezera, nebo zalomení řádku, popřípadě jiný bílý znak. Před jménem je však tento zástupný symbol spjat s kvantifikátorem, otazníkem. Ten nám říká že se zde může, ale také nemusí, vyskytovat bílý znak. Otazník je tedy naše druhá specifická hodnota. Samozřejmostí je prvek jméno, příjmení, anebo titul, který se mění dle dat z databáze. Pro nás to však není specifický prvek, nýbrž podstatný prvek pro parsování.

Druhým kritériem pro získávání hodnot je přesně stanovený regulární výraz. Toto kritérium nám určuje, že hodnota se musí přesně vztahovat k jednomu přesně stanovenému typu. Tento typ je neměnný a váže se na hodnoty, které jsou žádány. Tyto neměnné hodnoty, v našem případě, jsou celorepublikově uznávány, a proto je lze aplikovat. Toto kritérium se především vztahuje na emaily, které mají prakticky vždy zcela shodný typ, se kterým se pracuje. Tím je myšleno, že struktura emailu, ať už jste kdekoli je následující: „jakýkoli počet znaků vytvářející jméno účtu“ následuje „zavináč“ a v poslední řadě je zde „doména serveru“ kde se nachází náš email. Celková struktura emailu pak může vypadat zhruba takto „daniel.vondra258@seznam.cz“. Proto je snadné za pomoci tohoto regulárního výrazu `/(\s?[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-z]+\s?)/` přesně vyhledat email v získaném textu z životopisu pro parsování. Další hodnotou, která spadá do tohoto kritéria je telefonní číslo. Samozřejmě vezmeme-li v potaz to, že akceptujeme pouze telefonní čísla určena pro Českou republiku. Zde tedy počítáme s použitím předvolby +420, která je volitelná, po které následuje devět číslic.

Těchto devět číslic mezi sobou může, nebo nemusí mít mezery. Proto je pro určení telefonního čísla snadné určit regulární výraz, který vypadá takto `/\s?(\+420)?\s?[1-9][0-9]{2}\s?[0-9]{3}\s?[0-9]{3}/`. Díky tomuto výrazu jsme schopni získat telefonní číslo ze získaného textu pro parsování.

Po projití textu a nalezení, či nenalezení potřebných informací se začne vyhodnocovat chybovost. Chybovostí je myšlen proces, který zkoumá získané prvky a dle předdefinovaných informací určí, jak se má pokračovat.

První z možností, jak proces pokračuje, je situace, kdy máme nalezeny všechny potřebné informace. Tento stav nastává, pokud máme nalezené jméno, příjmení, telefonní číslo a email. Samozřejmě je zde možnost nalezení titulu, který sem také zapadá, ale pro vyhodnocení není podstatný, protože mohou být lidé bez titulu. Pokud nastane tato situace, dokument je vyhodnocen jako správně vyparsovaný. Stav je poté nastaven na sedmidenní destrukční dobu, kdy čeká, než bude automaticky smazán. Během tohoto nastavení se nalezené hodnoty uloží a zobrazí se pro uživatele. Tento proces neukládá žádná data do error logu, jedinou výjimkou je částečné uložení do tohoto chybového hlášení. To nastane v okamžik, kdy je vše nalezeno, krom titulu. Toto uložené hlášení nás tedy bude pouze upozorňovat na to, že u daného souboru nastal okamžik, kdy chybí titul. V tento okamžik je tedy potřeba ruční kontrola, pouze pro jistotu, aby vlastník životopisu nebyl ošizen o data o svém studiu.

Dalším stavem, který může nastat, je moment, kdy některá z hodnot není nalezena, nebo jakákoli kombinace prvků, kdy alespoň jedna z podstatných informací chybí. Opět jsou jako podstatné informace jméno, příjmení, telefonní číslo a email, stejně jako u předchozího. I zde titul nemusí být vyhodnocen, proto není zařazen jako podstatný prvek. Pokud toto nastane, je životopis nastaven do stavu možná. To znamená, že informace jsou zobrazeny uživateli, ale dokument není nastaven na sedmidenní destrukční dobu. Místo toho je navrácen zpět pro možnosti, kdy by mohl být znovu vyparsován, nebo ponechán pro ruční kontrolu. Následujícím krokem v tomto stavu je zapsání informace o tom, která hodnota nebyla nalezena a

jakého dokumentu se to týká. To proto, aby mohla být snazší ruční kontrola. Včetně změny stavu a uložení nalezených informací, proběhne uložení do error logu. Zde je nahlášena úplná chyba, kde se přesně zobrazí, u kterého životopisu nebyl nalezen prvek tak, aby uživatel mohl projít životopis ručně, nebo jej mohl zkusit vyparsovat znovu.

Posledním z možných nastalých stavů je úplná chyba. Tento okamžik nastane velice zřídka. Jedná se o stav, kdy nebyla nalezena ani jedna z povinných hodnot, kterými jsou, jak bylo stanoveno již výše, jméno, příjmení, telefonní číslo a email. Ani u tohoto stavu není titul brán jako povinná hodnota, a proto jeho nalezení, nebo nenalezení, nijak neovlivní vyhodnocování. Při nastoupení tohoto stavu je dokument okamžitě nataven zpátky, do stavu pro možnost jej znovu parsovat. Uživatel je v informačním okně výpisu obeznámen o plně neúspěšném pokusu o parsování. Dále je zapsán do error logu stav chyby, kde nebyla nalezena žádná informace tak, aby měl uživatel přehled o stavu daného dokumentu, kdyby se chyba opakovala. Dokument se tedy nezmění, přidá se pouze záznam do error logu.

Ať se jedná o jakýkoli z předešlých stavů, není v našem případě možné zaručit stoprocentní přesnost údajů z důvodu velké rozmanitosti jmen, příjmení, variant životopisů a stylů jejich psaní. Proto je zde stále v některých případech, při neshodě jmen, nebo při jejich divném znění nutnost ruční kontroly daného dokumentu. Jako příklad můžeme znázornit tento nastalý stav: Může nastat okamžik, kdy ve výběru bude zahrnuta škola, kterou majitel životopisu vystudoval a bude zde tedy nadpis Škola nebo Školy. Pokud jeho jméno nebo příjmení bude až po písmeni Š je možné, že slovo škola bude považováno jako příjmení, neboť jsou lidé s tímto příjmením evidováni v naší databázi. Největší pravděpodobnost správného nalezení je u vyhledávání emailů a telefonů, díky jejich předdefinovaným stylům, jak je zapisovat.

4.5.3 Uživatelský přístup

Velice podstatným prvkem v aplikace v každé webové aplikaci, která obsahuje citlivé informace, nebo chce oddělit pohyblivost po stránce, jsou uživatelské přístupy. Tato oprávnění oddělují běžného uživatele od uživatele s větší možností pohybu po stránce. Uživatelé se do těchto oddělených částí dostávají pomocí přihlášení, kde má každý uživatel své heslo a přihlašovací jméno a velice často také skrytý prvek, kterým je úroveň oprávnění. Ta mu určuje, co vše je schopen v aplikaci dělat, na co má oprávnění, či naopak, na co nemá oprávnění.

Naše aplikace obsahuje dvě úrovně zabezpečení proto, že jich v nynějším stavu není více zapotřebí. První takovou úrovní je běžný uživatel. V našem případě se jedná o uživatele, který má přihlašovací údaje do aplikace. Přihlašovací stránku si může načíst každý uživatel internetu, ale pouze určitá skupina lidí má přístup k datům, právě prostřednictvím přihlašovacích údajů. Každý přihlášený uživatel, dále jen uživatel, se může pohybovat po stránce tam, kde má povolen přístup. Každý jeho krok v aplikaci je doprovázen pomocí jeho identifikátoru, který je přímo spjat s jeho uživatelskými přístupy. Díky tomuto jednoznačnému prvku se dozví pouze o úkonech, které dělal on sám. Pokud tedy bude nahrávat, nebo parsovat soubor, bude vždy přiloženo jeho identifikační číslo. Tento identifikátor je také spjat s erorovými chybami, které mohou nastat během práce se soubory. Stejně jak tomu bylo u parsování a u nahrávání, i zde se tedy díky identifikačnímu číslu uživateli zobrazí pouze jeho chybové hlášky. Uživatel tedy uvidí pouze prvky, se kterými pracoval on sám. Tato úroveň je v aplikaci uvedena pod číslem jedna.

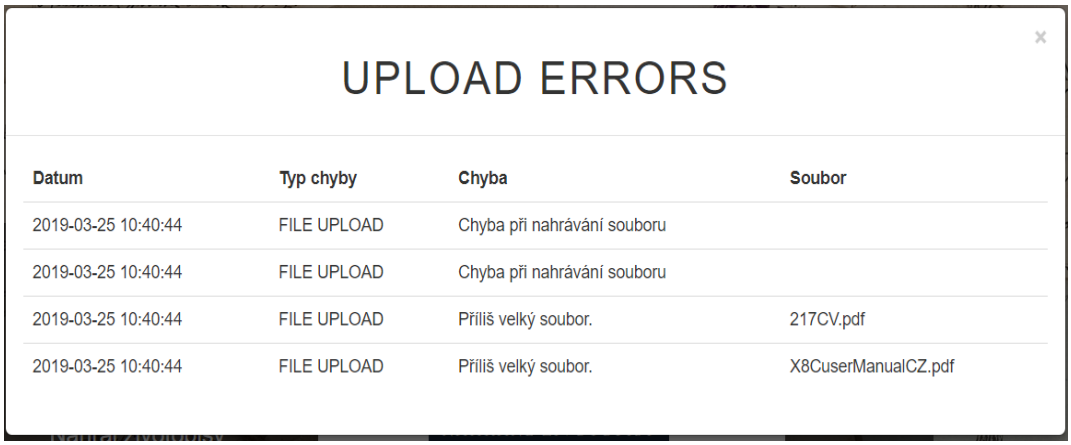
Druhou nezbytnou úrovní v naší aplikaci, je úroveň administrátorská. Toto oprávnění, krom samotného parsování, nahrávání a všech výsad, které má uživatel, také dovoluje administrátorskému účtu, dále jen admin, pracovat s účty. Admin je schopen přidávat uživatele, pokud by to bylo nezbytné, je také schopen měnit jejich oprávnění tak, že může být více uživatelů s oprávněním Admin. Jednou z výhod admina je, že může zamezit přístup libovolnému uživateli, nebo mu jej naopak přidělit. Dále uvidí chybové hlášení týkající se přístupu do aplikace, kde nastala

jakákoli z námi evidovaných chyb. Jak už bylo uvedeno, admin bude v databázi evidován pod číslem nula. Aby bylo možné oddělit správce, který může měnit i oprávnění adminům, bude mít tento správce oprávnění mínus jedna. Díky tomu budeme schopni je vzájemně odlišit, aniž bychom museli rozšiřovat správcovská oprávnění. Správce, krom jiného, bude také schopen měnit hesla uživatelům.

4.5.4 Logování chyb

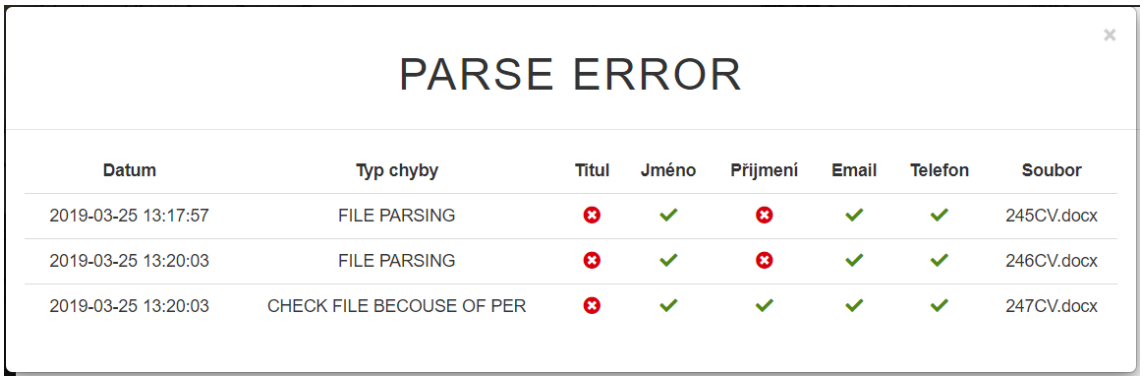
Logování chyb v aplikaci, v našem případě, probíhá vždy na straně serveru, kde se provádí kontrola prováděného úkonu. Každá chyba, která je v aplikaci provedena, je uložena do databáze s požadovaným popisem chyby a datumem jejího vzniku. Díky tomu je uživatel schopen ohodnotit stav, který chybně nastal a opravit potřebné prvky tak, aby se chybě v dalším kroku vyvaroval.

Aplikace je schopna rozeznat několik druhů chyb, které mohou v aplikaci nastat. První druh může nastat v okamžiku nahrávání souborů. Při nahrávání souborů evidujeme několik možných nastalých chyb, například velikost souboru, nebo typu formátu. Chyba nastane v okamžiku, kdy se uživatel snaží nahrát soubor, který je příliš velký, tedy větší než rozsah, který máme stanovený pro jednotlivé soubory. Velikost je akceptovatelná do dvě stě megabytů (200 MB) na každý nahrávaný soubor. Další chyba se týká formátu souboru, tato chyba je zde zavedena hlavně pro případ, kdyby si uživatel ve výběru zvolil možnost nahrát všechny soubory a pokusil se nahrát jiný formát, než je jeden z námi zvolených typů (PDF, DOC, DOCX, RTF). Chyba, které také v tento okamžik může nastat, je při nahrávání. Informace o souboru se nepodaří nahrát do databáze, ať je důvod jakýkoli. Toto jsou chyby, které se objevují nejčastěji, a proto jsou evidovány jmenovitě, dále však také evidujeme obecných chyby, které při nahrávání mohou nastat, ale tyto evidujeme pouze obecně. Tyto jsou nahrány jako chyba s číslem a názvem souboru. Všechny výše zmíněné chyby jsou uživateli zobrazeny po nahrávní souborů s popisem nastalé chyby. Zde uživatel vidí počet úspěšně nahraných souborů, počet neúspěšně nahraných a samozřejmě chybové hlášky, které jsou uchovány i v error logu.



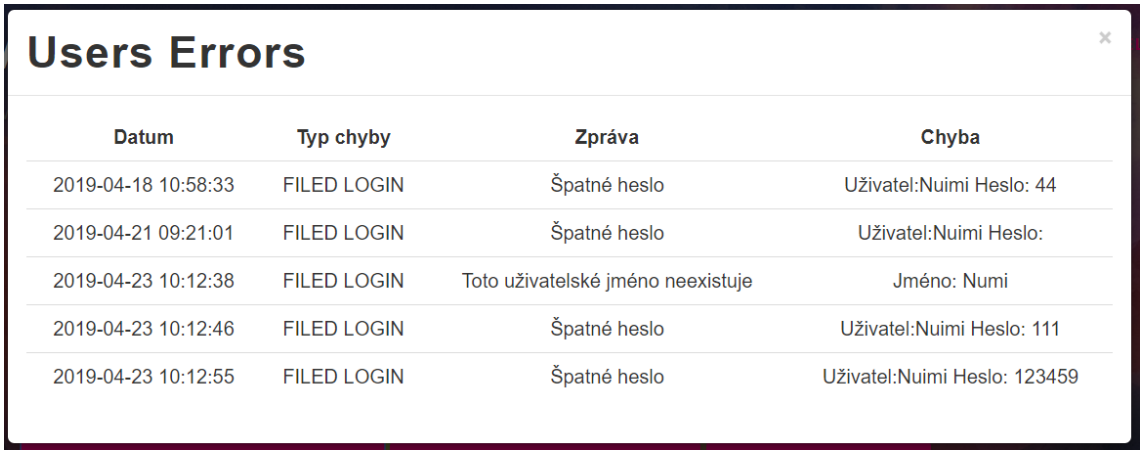
Obr. 8 Chybové hlášky při nahrávání souborů

Dalším druhem chyb, které evidujeme, je chyba při parsování souborů. Tyto nastávají v okamžik, kdy parsujeme soubor, kde se nepodařilo nalézt nějaký prvek z těch, které jsme si předdefinovali (titul, jméno, příjmení, email, telefon). Pokud chybí jakýkoli z těchto prvků je zaevidován error. Ten nám oznamuje soubor, ve kterém chyba nastala, které z prvků byly nalezeny a které chybí. Tato chyba nastává i v případě, že nebyl nalezen titul, a to pro případ, že by uživatel titul měl, ale aplikace jej nebyla schopna rozeznat. U každého errorového výstupu je možnost nahlédnout na parsovaný text, popřípadě si zobrazit samotný životopis pro kontrolu údajů. V procesu parsování může samozřejmě nastat okamžik kdy nebyl získán text, tato chyba se eviduje prostřednictvím výpisu, který informuje uživatele o tom, že nebyl nalezen žádný prvek.



Obr. 9 Výstup při chybách v parsování

Jednou z velmi významných errorových hlášek, kterou aplikace bude zaznamenávat, je chyba, která může nastat při logování uživatele. Tyto chyby mohou nastat v několika okamžicích a podle toho s nimi bude zacházeno. Všechny tyto typy však spadají pod uživatelské chyby. Prvním jsou chyby při přihlášení. Můžeme dostat chyby o neúspěšných pokusech při přihlášení, kde nebylo nalezeno jméno v databázi, nebo kde uživatel zadal špatné heslo. Tato část chyb bude evidována hlavně kvůli případnému napadení aplikace. Útočník by se mohl snažit hádat hesla a jména pro získání přístupu do aplikace. Tyto chyby budou vyhodnocovány a bude podle toho hlášen stav, zda se dle počtu pokusů jednalo o útok, nebo o špatné zadání hesla. Dále budeme evidovat chyby při práci s uživateli v samotné aplikaci. Pouze pro případ, kdyby se opakovaly chyby při ukládání dat, nebo nastávaly opakované chyby při upravování uživatelů.



Obr. 10 Výstup při chybách u přihlášení

4.6 Ajax

Ajax je prvek Java scriptu, který nám umožňuje komunikovat se serverem tak, že nemusíme čekat, než přijde odpověď a můžeme se stránkou pracovat. Jedná se o asynchronní funkci, která neblokuje činnost stránky. V okamžik, kdy přijde odpověď, obnoví se pouze část stránky, pro kterou jsme dostali informace a stránka se nemusí celá obnovovat, tomu se říká collback. Díky tomuto lze komunikovat s databází, nebo s PHP scripty, aniž bychom obtěžovali uživatele stále se opakujícím obnovováním stránky. Ajax pochází ze zkratky Asynchronous Javascript And XML. XML se zde nachází proto, že Ajax pro datovou komunikaci může používat kupříkladu XML, dalšími možnostmi jsou ale také JSON, text, nebo binární data. [18]

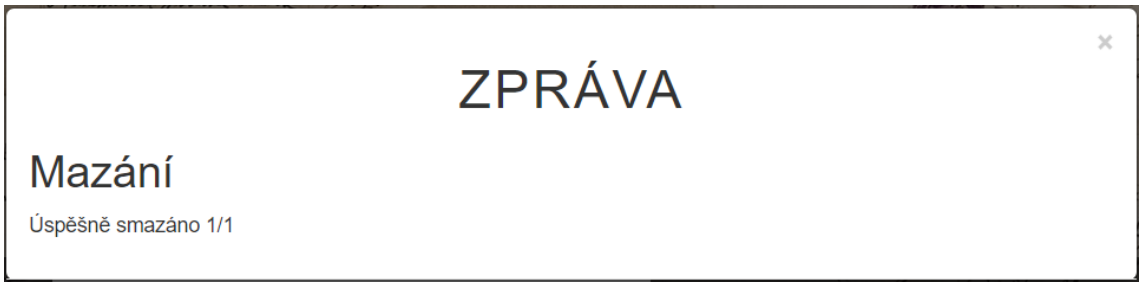
Ajax v naší aplikaci tvoří nejpodstatnější část, hraje obrovskou roli při parsování. Jeho funkce nám umožňuje posílat soubory jeden po druhém pro parsování, informuje uživatele o průběhu parsování a samozřejmě zobrazuje výsledek.

Uživatel nejprve zvolí soubory, které chce parsovat a odešle je pro parsování. Ajax si tyto soubory přebere a sestaví z nich pole, kde definuje pouze id, které jej spojuje s informacemi z databáze. Následně tyto id posílá PHP souboru, který je zpracovává. V ten samý okamžik, kdy odešle pole, zobrazí uživateli informaci, že se zahájilo parsování. Tím jeho práce nekončí. Po celý průběh parsování komunikuje s PHP scriptem, který tuto činnost vykonává a informuje uživatele o stavu. V okamžik dokončeného parsování uživateli oznámí, že práce byla dokončena a obnoví data, která PHP script provedl. Zde již uživatel vidí provedené úkony a získané prvky z životopisů.

4.7 Výstup z aplikace

Výstup z naší aplikace se dá rozdělit do dvou částí, informační a výstupová. Oba tyto typy se zobrazují v modalech. Jedná se o prvek, vytvořený přes Bootstrap, který nám může zobrazovat informace. Blok, který se zobrazí nad stránkou a pozadí se ztmaví, pro lepší přehlednost.

Informačními výstupy jsou zprávy, které se zobrazují uživateli. Každý úkon, který uživatel provede, je následován informační zprávou, která uživateli zobrazí text, který mu oznamuje, co se provedlo. Nejvíce těchto zpráv je zobrazeno v okamžik, kdy se nahrávají soubory. Uživatel je informován o stavu jeho nahráváních souborů. Získá informaci o počtu souborů, který se povedl nahrát, popřípadě se zobrazí i chyby, které nastaly při nahrávání. Dále se zobrazují při mazání souborů. Uživatel vidí počet souborů, které se smazaly. V poslední řadě se zobrazují zprávy při parsování, kde uživatel vidí postup při parsování souborů.



Obr. 11 Ukázka informační zprávy

Výstupová část zobrazování ukazuje uživateli prvky, které obsahuje aplikace. Nejpodstatnějšími informacemi jsou soubory, které uživatel nahrál. Zde uživatel vidí název souboru a jeho typ. Dále to jsou informace získané po parsování, kde uživatel vidí veškerá získaná data, jako titul, jméno, příjmení, email a telefon. Také je zde vidět soubor samotný, znázorněný v obrázkovém odkazu. Uživatel si jej může znovu stáhnout pro kontrolu. Z tohoto informačního bloku se lze podívat do error logu. Také odtud se dá přesunout do chybného výstupu, který se moc neliší. Jedinou změnou je, že jsou zde zobrazeny pouze soubory, při kterých se nenalezly nějaké

podstatné prvky (jméno, příjmení, email nebo telefon). Mezi výstupovou část se také počítá errorový výstup, kde můžeme vidět nastalé chyby, ať už se jedná o chyby z parsování, nebo nahrávání souborů.

Obr. 12 Ukázka výstup

5 Shrnutí výsledků

Při tvorbě této webové aplikace se vyskytovaly časté problémy, vázající se na získávání jmen a příjmení, z parsovaných životopisů. Když zanedbáme pravděpodobnost, je možnost, kdy nám aplikace vrátí chybně nalezený prvek. Kvůli chybovosti, která mohla nastat, jak bylo již zmíněno výše, se jednalo o chyby textového prvku. Kvůli velké škále typů životopisů, jejich struktury nebo celkovému stylu zápisu se mohlo stát, že se pro jeden životopis psaný jinak, vrátilo několik různých chybných výsledků.

Ku příkladu při hledání příjmení podle jména, kdy je nalezeno jméno a dle jeho pozice sledujeme, zdali existuje za tímto slovem prvek, který je oddělen pouze mezerou. Takto hledané slovo by mohlo být příjmením nejpravděpodobněji, pokud by bylo jméno nalezeno správně. Chybovost však v tomto okamžiku nastávala ve dvou možnostech. Tou první byl okamžik, kdy se nám správně vrátilo příjmení, ale pokud se jednalo o ženu a její příjmení končilo na „á“, tak nám regulární výraz místo tohoto znaku vrátil sekvenci symbolů, která by tento znak měla reprezentovat. Jenže kvůli zpětnému převodu jsme nemohli zpátky získat korektní znak. Proto byl regulární výraz přepsán. Tím nám však vznikla nová chyba.

Tato druhá chyba vznikla ohraničením výběru. Pokud by bylo příjmení tedy zapsáno za jménem, tak bychom ho v tento okamžik již vrátili správně. Teď však šlo o strukturu životopisu. Pokud jsme vzali ten samý životopis a udělali ho v několika formách, včetně změny dokumentu, ku příkladu na PDF, tak nám v několika okamžicích byl vrácen špatný text. Toto vrácení vzniklo na základě omezeného výběru, který se především sledoval, zdali za příjmením existuje styl nebo mezera. Protože každý životopis může mít jinou strukturu, získávali jsme občas jinak dlouhé hodnoty. Tento problém byl odstraněn zvolením vícero možností, jak ukončit vyhledávání pomocí regulárního výrazu.

6 Závěry a doporučení

Aplikace splňuje předpoklady, které byly stanoveny. Je tedy schopna získávat informace z vícero životopisů a získává je správně. Největší chybovost byla odstraněna a aplikace funguje s velikou pravděpodobností správných informací. Největší chybovost však stále zůstává při získávání informací o jméně a příjmení vlastníka životopisu. Tato chybovost by se dala do budoucna úplně odstranit, přidáním umělé inteligence, která by se sama učila, z již vyparsovaných dokumentů. Přidání této umělé inteligence je tedy možným rozšířením.

V této práci jsou i přidané prvky, které v požadavcích nebyly. Jedním z těchto prvků je ku příkladu cron, který je schopen kontrolovat stáří souborů, zdali jsou vyparsované správně a na základě těchto informací je schopen tyto soubory mazat tak, aby zde nezůstávaly soubory, které už nejsou potřeba. Hlavně z důvodu udržení soukromí vlastníků životopisu. Velice podstatná přidaná hodnota je uživatelské rozhraní. Aplikaci rozděluje pro uživatele, návštěvníky a samozřejmě správce. Díky tomu máme aplikaci zabezpečenou a můžeme tak lépe chránit osobní údaje uživatelů. Tento prvek by se dal do budoucna rozšířit tak, aby zde mohlo pracovat vícero organizací, které by na sobě nebyly nijak závislé a nevěděly by vzájemně o uživatelích, nebo o souborech.

Krom přidání umělé inteligence, jež by získávala informace z životopisu, je zde možné rozšířit tuto webovou aplikaci tak, aby byla schopna získávat více informací, včetně fotky uživatele, pokud jej životopis obsahuje, nebo také jeho schopnosti, či doporučení, která jsou uvedena v životopisu.

7 Seznam použité literatury

[1] PHP. Adaptic [online]. Praha: adaptic, 2015 [cit. 2018-08-20]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/php/

[2] Regulární výrazy. Mergado [online]. Brno: Mergado technologies, 2018 [cit. 2018-08-20]. Dostupné z: https://www.mergado.cz/tema/regularni-vyrazy

[3] Základní linuxové funkce I. Sally [online]. Praha: Sally, 2016 [cit. 2018-08-20]. Dostupné z: https://www.sallyx.org/sally/c/linux/zaklady-i

[4] Co je to HTML?. *Principy značkovacích jazyků a příklady použití jazyka HTML a XML – Wikisofia* [online]. Česká republika: Creative Commons Uveďte autora 3.0 Česko, 2013 [cit. 2018-11-06]. Dostupné z: https://wikisofia.cz/wiki/Principy_zna%C4%8Dkovac%C3%ADch_jazyk%C5%AF_a_p%C5%99%C3%ADklady_pou%C5%BEit%C3%AD_jazyka_HTML_a_XML

[5] NIXON, Robin. *Learning PHP, MySQL & JavaScript WITH JQUERY, CSS & HTML5*. United States of America.: O'Reilly Media, Inc., 2015, s. 5-11. ISBN 978-1-491-91866-1.

[6] Co je CSS. In: *Pěstujme web* [online]. Česká republika: Pěstujme web, 2011 [cit. 2018-11-06]. Dostupné z: http://www.pestujemeweb.cz/obsah/css/co-je-css.php

[7] Co je PHP. *Adaptic* [online]. Česká Republika: Adapric, 2015 [cit. 2018-11-06]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/php/

[8] Úvod do JavaScriptu. *Jak psát web* [online]. Česká republika, 2005 [cit. 2018-11-19]. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html

[9] Co je to databáze MySQL?. *Artic Studio* [online]. Česká republika, 2005 [cit. 2018-11-19]. Dostupné z: https://www.artic-studio.net/slovnicek-pojmu/databaze-mysql/

[10] XML introduction. *MDN web docs* [online]. USA: Mozilla, 2015 [cit. 2018-12-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/XML_introduction

[11] LibreOffice 6 Review. *Linux and Ubuntu* [online]. USA: Ubuntu, 2018 [cit. 2018-12-03]. Dostupné z: http://www.linuxandubuntu.com/home/libreoffice-6-review

[12] Co je Linux?. *Linux Expres* [online]. Czech Republic: LinuxExpres, 2018 [cit. 2018-12-03]. Dostupné z: https://www.linuxexpres.cz/co-je-linux

[13] Regulární výrazy. *Regular Expressions* [online]. Česká Republika: Miroslav Pecka, 2005 [cit. 2019-01-31]. Dostupné z: https://www.regularnivyrazy.info/#gref

[14] About TextKernel. *TextKernel* [online]. Rakousko: TextKernel B.V., 2019 [cit. 2019-01-31]. Dostupné z: https://www.textkernel.com/company/about-textkernel/

[15] Extract!. *TextKernel* [online]. Rakousko: TextKernel B.V., 2019 [cit. 2019-01-31]. Dostupné z: https://www.textkernel.com/hr-software/extract-cv-parsing/#benefits-link

[16] Extract!. *TextKernel* [online]. Rakousko: TextKernel B.V., 2019 [cit. 2019-01-31]. Dostupné z: https://www.textkernel.com/hr-software/extract-cv-parsing/#features-link

[17] Daxtra overview. *Daxtra* [online]. Veklá Británie: Dextra, 2002 [cit. 2019-01-31]. Dostupné z: https://www.daxtra.com/candidate-management-software/

[18] Dextra Parser. *Dextra* [online]. Veklá Británie: Dextra, 2002 [cit. 2019-01-31]. Dostupné z: https://www.daxtra.com/resume-database-software/resume-parsing-software/

[19] Lekce 1 - Úvod do AJAXu. *ITnetwork.cz Přihlásit seRegistrovat* [online]. Česká republika: ITnetwork.cz Přihlásit seRegistrovat [cit. 2019-03-24]. Dostupné z: https://www.itnetwork.cz/javascript/ajax/uvod-do-ajaxu

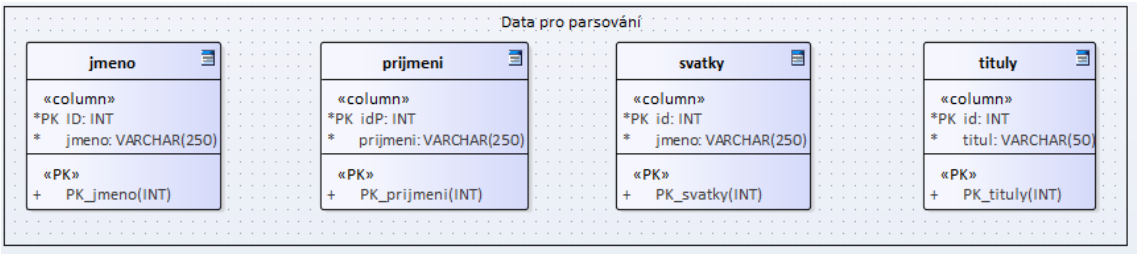
[20] Četnost jmen a příjmení. *Ministerstvo vnitra České Republiky* [online]. Česká Republika [cit. 2019-04-18]. Dostupné z: https://www.mvcr.cz/clanek/cetnost-jmen-a-prijmeni.aspx

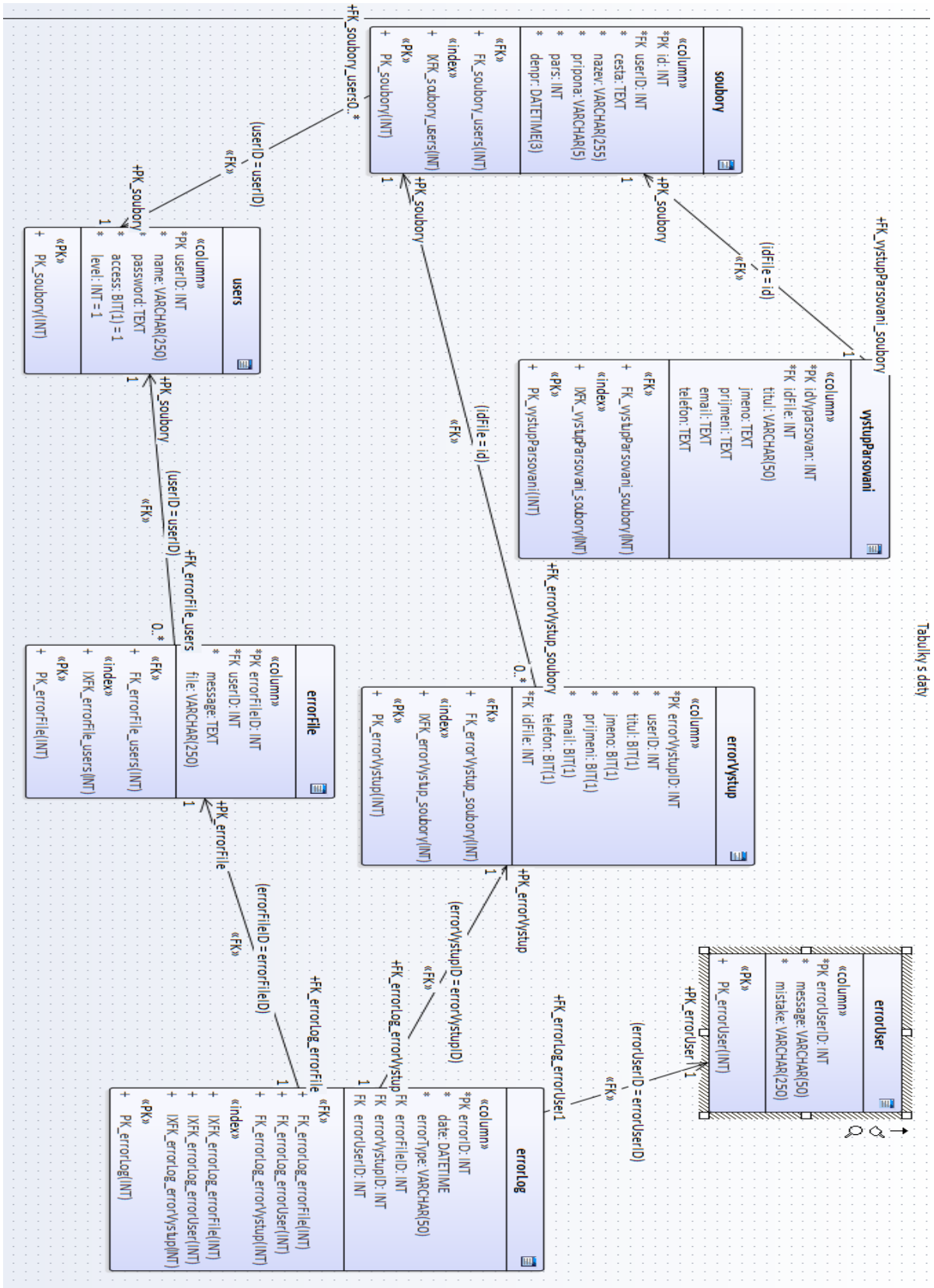
[21] PHPDocumentParser. *GitHub* [online]. GitHub, 2019 [cit. 2019-04-18]. Dostupné z: https://github.com/LukeMadhanga/PHPDocumentParser

[22] PdfParser. *GitHub* [online]. GitHub, 2019 [cit. 2019-04-18]. Dostupné z: https://github.com/smalot/pdfparser

8 Přílohy

1) Úplná struktura databáze


Data pro parsování
jmeno
«column»
*PK ID: INT
* jmeno: VARCHAR(250)
«PK»
+ PK_jmeno(INT)
prijmeni
«column»
*PK idP: INT
* prijmeni: VARCHAR(250)
«PK»
+ PK_prijmeni(INT)
svatky
«column»
*PK id: INT
* jmeno: VARCHAR(250)
«PK»
+ PK_svatky(INT)
tituly
«column»
*PK id: INT
* titul: VARCHAR(50)
«PK»
+ PK_tituly(INT)


Tabulky s daty
errorUser
«column»
*PK errorUserID: INT
* message: VARCHAR(50)
* mistake: VARCHAR(250)
«PK»
PK_errorUser(INT)
+PK_errorUser
(errorUserID = errorUserID)
«FK»
+FK_errorLog_errorUser1
errorLog
«column»
*PK errorID: INT
* date: DATETIME
* errorType: VARCHAR(50)
FK errorFileID: INT
FK errorVystupID: INT
FK errorUserID: INT
«FK»
FK_errorLog_errorFile(INT)
FK_errorLog_errorUser(INT)
FK_errorLog_errorVystup(INT)
«index»
IXFK_errorLog_errorFile(INT)
IXFK_errorLog_errorUser(INT)
IXFK_errorLog_errorVystup(INT)
«PK»
PK_errorLog(INT)
+FK_errorLog_errorFile
+FK_errorLog_errorVystup
(errorFileID = errorFileID)
(errorVystupID = errorVystupID)
errorVystup
«column»
*PK errorVystupID: INT
* userID: INT
* titul: BIT(1)
* jmeno: BIT(1)
* prijmeni: BIT(1)
* email: BIT(1)
telefon: BIT(1)
«FK»
FK_errorVystup_soubory(INT)
«index»
IXFK_errorVystup_soubory(INT)
«PK»
PK_errorVystup(INT)
+FK_errorVystup_soubory
+PK_errorVystup
(idFile = id)
0..*
errorFile
«column»
*PK errorFileID: INT
*FK userID: INT
* message: TEXT
file: VARCHAR(250)
«FK»
FK_errorFile_users(INT)
«index»
IXFK_errorFile_users(INT)
«PK»
PK_errorFile(INT)
+FK_errorFile_users
+PK_errorFile
(userID = userID)
users
«column»
*PK userID: INT
* name: VARCHAR(250)
password: TEXT
* access: BIT(1) = 1
* level: INT = 1
«FK»
FK_soubory_users(INT)
«index»
IXFK_soubory_users(INT)
«PK»
PK_soubory(INT)
+FK_soubory_users
+PK_soubory
soubory
«column»
*PK id: INT
*FK userID: INT
* cesta: TEXT
* nazev: VARCHAR(255)
* pripona: VARCHAR(5)
* pars: INT
* denpr: DATETIME(3)
«FK»
FK_vystupParsovani_soubory(INT)
«index»
IXFK_vystupParsovani_soubory(INT)
«PK»
PK_vystupParsovani(INT)
+FK_vystupParsovani_soubory
+PK_soubory
vystupParsovani
«column»
*PK idVyparsovan: INT
*FK idFile: INT
titul: VARCHAR(50)
jmeno: TEXT
prijmeni: TEXT
email: TEXT
telefon: TEXT
«PK»
PK_vystupParsovani(INT)
(idFile = id)

Oskenované zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Vondra Daniel	Doubrava 103, Loukov	I1600632

TÉMA ČESKY:

Parsování CV

TÉMA ANGLICKY:

CV parsing

VEDOUCÍ PRÁCE:

doc. RNDr. Petra Poulová, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Základem práce bude vytvoření aplikace pro parsování životopisů ve formátu doc, docx, rtf a pdf.
Aplikace bude schopná detekovat jméno, příjmení, telefonního číslo, email popřípadě titulu dané osoby z CV.
Osnova:
1. Úvod
2. Teorie - a. životopisy (struktura xml)
   b. technologie
3. Praxe - a. vytvoření aplikace
   b. testování
4. Výstupy z aplikace
5. Záver

SEZNAM DOPORUČENÉ LITERATURY:

PHP and MySQL for Dynamic Web Sites (Fourth Edition) - Larry Ullman
PHP and MySQL? web development (Fourth Edition) - Luke Welling, Laura Thomson
Learning PHP, MySQL & JavaScript, With jQuery, CSS & HTML5 - Robin Nixon

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....