Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Elektronický nářeční slovník

Plzeň 2017 Petr Lukašík

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. května 2017

Petr Lukašík

Poděkování

Chtěl bych poděkovat své rodině za podporu při vypracovávání této bakalářské práce a svému vedoucímu Ing. Romanu Moučkovi, Ph.D. za přínosné konzultace. Dále Antonínu Vrbovi za poskytnutí serveru pro testování aplikace.

Abstract

This bachelor's thesis deals with the creation of a web application for administration and presentation of dialects in the Czech Republic. The aim is to preserve the Czech dialect by converting individual words into electronic form with the possibility of administration. For designing a unique web application, an analysis of available electronic dialectical dictionaries and cultural artefact management application is performed. The design of the new database is based on the existing model of the electronic dialect dictionary of the University of West Bohemia. The web application is then implemented, according to the design, with the use of web-based frameworks. At the conclusion, the whole application is tested by designed tests.

Abstrakt

Tato bakalářská práce se zabývá tvorbou webové aplikace pro správu a prezentaci nářečí v České republice. Cílem je uchování českého nářečí převedením jednotlivých hesel do elektronické podoby s možností správy. Pro navrhnutí unikátní webové aplikace je provedena analýza dostupných elektronických nářečních slovníků a aplikace pro správu kulturních artefaktů. Návrh nové databáze vychází z existujícího modelu databáze elektronického nářečního slovníku Západočeské univerzity. Webová aplikace je následně podle návrhu implementována s využitím frameworků pro tvorbu webu. V závěru je celá aplikace otestována navrženými testy.

Obsah

1 Úvod

Čeština je velmi rozmanitý jazyk, ve kterém se formy mluveného slova z relativně jednotného jazykového základu vyvíjely různě v průběhu historie České republiky. Hlavní silou měnící jazykový vývoj je bezesporu jazykový kontakt s cizími jazyky, kterými jsou nejčastěji němčina, polština, slovenština a také různá nářečí na českém území. V průběhu historie se však nářeční rozdíly začaly stírat do základního nespisovného útvaru, který se dodnes rozděluje na čtyři hlavní skupiny (česká, středomoravská, východomoravská, moravskoslezká). Zájmem je tedy zachovat nářeční hesla a s nimi pojené kulturní artefakty, které se v České republice vyskytovaly v průběhu historie.

Existující nářeční slovníky pro Českou republiku se zaměřují převážně na specifická území výskytu nářečního dialektu. Pouze malé procento webových aplikací zaznamenává nářeční hesla z celých Čech. Západočeská univerzita ve spolupráci se Slezskou univerzitou v Opavě spolupracuje na utváření elektronických nářečních slovníků. V současné době využívaný slovník však nebyl dlouho udržován a stále v něm existují chyby. I přes tento nedostatek je webová aplikace využívána a nyní obsahuje tisíce záznamů.

V rámci bakalářské práce je provedena analýza dostupných elektronických nářečních slovníků, o které se zmiňuje kapitola 2 a ve 3. kapitole je představena webová aplikace pro správu kulturních artefaktů. Ve 4. kapitole je následně vypracován návrh nového elektronického nářečního slovníku. Při implementaci byly použity frameworky pro snazší programování webové aplikace, které byly analyzovány v kapitole 5. Navrhnutí modelu databáze je zpracováno v kapitole 6. a v 7. kapitole je představena již samotná implementace systému podle navržených řešení. V 8. kapitole je implementovaná webová aplikace řádně otestována navrženými testy. V 9. a 10. kapitole jsou zhodnoceny všechny dosažené výsledky a splněné požadavky celé práce.

2 Elektronické nářeční slovníky

Český jazyk není jednotvárný, a proto se již od pradávna vytvářely různorodé formy mluveného slova v České republice. Formy mluveného slova jsou obecně nazývány jako nářečí dané oblasti. V průběhu času se tato hesla sepisovala, a tak vznikaly první dokumenty českého nářečí. Dokumenty následně posloužily při vytváření internetových stránek zaměřených na nářečí jednotlivých oblastí a elektronických slovníků. Dostupné elektronické nářeční slovníky jsou v této bakalářské práci analyzovány pro zhodnocení konkurenčních webových aplikací. Výsledky analýzy budou využity při vytváření elektronického slovníku.

2.1 Slovník nářečí českého jazyka - SNČJ

Nářeční slovník spravovaný Ústavem pro jazyk český [1] je nová webová aplikace založená v roce 2016. Design stránky staví na jednoduchosti a přehlednosti. Na první pohled lze vidět navigační lištu, vyhledávač slov a základní informace. Slovník je výjimečný tím, že zpracovává všechna nářečí v České republice. Po bližším prozkoumání bohužel obsahuje pouze hesla začínající na písmena A-C. Tým stojící za tímto slovníkem ale upozorňuje na brzké přidání dalších hesel. Jednotlivá hesla obsahují velmi podrobný popis, synonyma pro jiné oblasti, lokaci výskytu, příklad užití ve větě a případně odkaz na literaturu viz Obr. 2.1. Vyhledávání funguje zadáním hledané fráze či některé její varianty. Pro veřejnost zde neexistuje možnost sdílet své poznatky či nová hesla.

Obrázek 2.1: Podrobný popis hesla na SNČJ [1]

2.2 Nářeční slovník ZČU

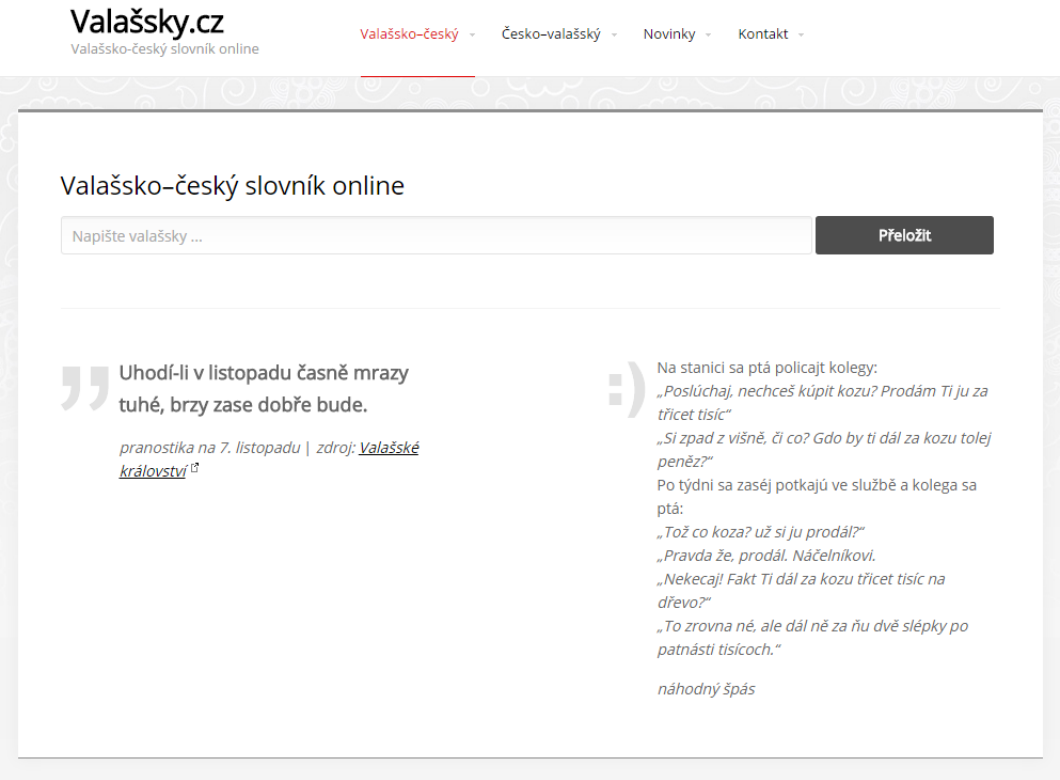
Tento elektronický nářeční slovník [2] byl vytvořen před deseti lety jako součást bakalářské práce Jany Michalicové a Jiřího Bergera. Ve slovníku lze vyhledat hesla z celé České republiky. Vizuální stránka tohoto webu zřejmě nebyla aktualizována od doby vzniku a vývoj se nejspíše zastavil v době testování. Copyright stránky však ukazuje na stále aktivní fungování webu. Množství hesel v databázi je neustále zvyšováno, jelikož každý registrovaný uživatel má možnost přispět novým heslem vyplněním formuláře na webu. Tato hesla jsou samozřejmě následně kontrolována administrátorem. Pro vyhledávání hesel zde existuje vyhledávač s mnoha filtry. Hledat hesla lze podle textového řetězce, frazémů, synonym, rozdělení podle obce nebo mluvčího. Speciálně lze také vyhledávat audio nahrávky. Popis u každého hesla se liší podle informací, které byly dodány uživatelem, který dané heslo přidal. V nejlepším případě se u hesla vyskytuje slovní druh, význam, příznak, výslovnost, oblast užití, příklad ve větě, exemplifikace, užití, synonymy, tezaurus a audio. Hesla však nejsou vždy takto do detailu popsána a jsou ponechávána s menším množství informací.

2.3 Valašsko-český slovník

Populární slovník orientovaný na oblast nejvýchodnější části Moravy [3]. Současná vysoká aktivita lze dohledat přímo na webu díky historii posledních hledaných hesel. Webová aplikace upoutává pozornost, krom hladkého a respozivního designu, hlavně náhodnou pranostikou a vtipem na hlavní stránce viz Obr. 2.2. Vyhledávání hesel funguje podobným způsobem jako překladač, kde můžeme buď zadat české slovo a "přeložit" ho do valašského či naopak. Popis hesla zahrnuje zkratky popsané ve vysvětlivkách a někdy i použití ve větě. Slovní zásobu autor použil z knihy "Slovník valašského nářečí". Jediná možnost pro přidání nových hesel od uživatelů je možná pouze posláním vzkazu na webu nebo na email.

2.4 Slovník valašského nářečí

Velmi zjednodušená forma elektronického slovníku [4], ve kterém nelze ani přesně určit, jak stará webová aplikace je. Celý web je pozicovaný v levém horním rohu a je stylizovaný v ostře žlutých barvách. Pro nalezení hesla zde neexistuje žádný vyhledávač. Hesla lze zobrazit v PDF souborech, které jsou rozděleny podle abecedy. Hledané heslo je pak již třeba vlastnoručně dohle-



Obrázek 2.2: Hlavní strana webové aplikace Valašsky.cz [3]

dat v souboru. Jednotlivá hesla jsou přeložená a některá přiměřeně popsaná. Do slovníku nelze přidávat další hesla z důvodu nemožnosti přístupu a integrace do PDF souborů. Tyto materiály byly s největší pravděpodobností použity i u předchozího Valašsko-českého slovníku, protože se zde nachází naprosto totožně sepsané vysvětlivky.

2.5 Slovník brněnského hantecu

Elektronický slovník nářečí [5] užívaného v Brně, kde se všechna hesla vyskytují pouze na jedné stránce. Webová aplikace není nijak udržovaná, jelikož poslední úprava proběhla v roce 2007, což se také podepsalo na zastaralém designu. K vyhledávání hesel zde slouží abeceda odkazující na níže vypsaná hesla začínající na vybrané písmeno. Hesla jsou pouze vysvětlená a nejsou k nim přidané žádné detaily. Případné přidávání nových hesel není umožněno přes web, ale je možné zaslat náměty a připomínky na autorovu emailovou adresu.

2.6 Slovník po naszi(y)mu

Slovník horalsko-slezkého nářečí [6] používaného na rozmezí České republiky, Slovenska a Polska. Web působí staromódně a zastarale, avšak podle aktualit webu je slovník stále udržován. Také zde je nemožné nalézt vyhledávač, a proto se jednotlivá hesla hledají podle prvních písmen abecedy. Hesla se vypisují pouze v levé části webu a obsahují pouze přímý překlad bez dalších detailů. Jednotlivá fráze a hesla jsou psány podle pravidel polského pravopisu, a proto lze v pravé části webu nalézt návod ke čtení. Pod návodem je zvýrazněna informace o zrušení možnosti přidávat nová hesla, která jsou nyní přidávaná pouze autorem na požádání přes email.

2.7 Shrnutí

V následující tabulce je zaznamenáno celkové shrnutí jednotlivých webů s ohodnocením **1-3**, kde **1** znamená nejlepší a **3** nejhorší. Výsledná známka je vypočítána průměrem součtu udělených známek. Výsledek je v případě hodnoty desetinného čísla 0.5 (1.5, 2.5) zaokrouhlen dolu.

	Design	Množství dat	Bohatost popisu	Možnost přispívání	Výsledná známka
Slovník nářečí českého jazyka - SNČJ	1	3	1	3	2
Nářeční slovník ZČU	2	2	1	1	1
Valašsko-český slovník	1	2	1	2	1
Slovník valašského nářečí	3	2	2	3	2
Slovník brněnského hantecu	3	2	3	3	3
Slovník po naszi(y)mu	2	1	3	2	2

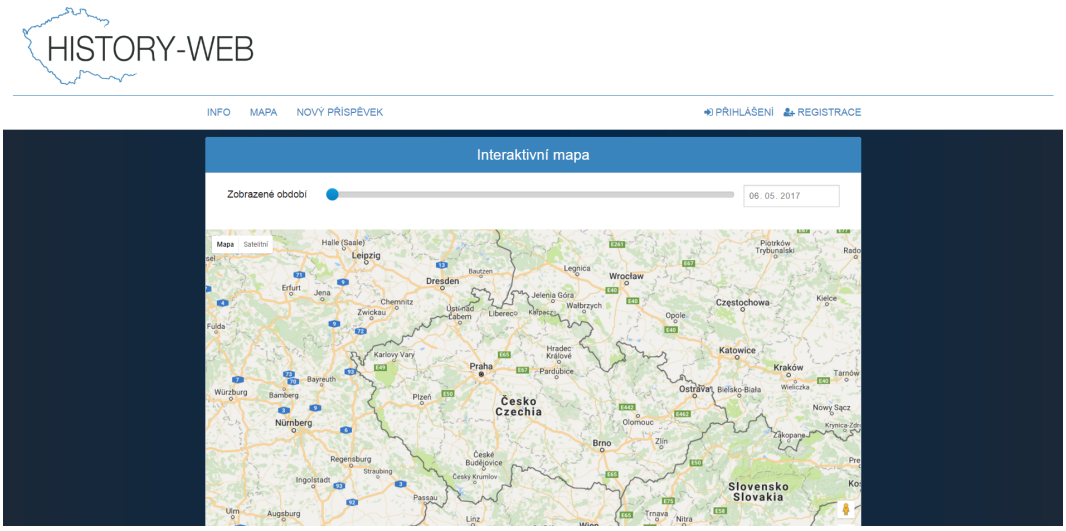
Tabulka 2.1: Shrnutí analýzy elektronických nářečních webů

Analýza těchto webů poukázala na některé slabiny a silné stránky elektronických nářečních slovníků. Celkově vyšly nejlépe weby Nářečního slovníku ZČU a Valašsko-českého slovníku. Tento projekt se bude snažit vyhnout převážně designovým chybám, které se objevovaly nejčastěji.

3 Webový portál pro správu kulturních artefaktů

Týmový projekt vytvářený v předmětu Základy softwarového inženýrství na katedře informatiky Západočeské univerzity. Cílem práce bylo vytvoření webové aplikace spravující kulturní artefakty v České republice viz Obr. 3.1, pro uchování historicky zajímavých dat a souborů. Na projektu se podílel Jan Šedivý, jako grafik, hlavní programátor front-end webu a Google mapy. Dále Jan Palcút, který navrhl model databáze, zajistil metody propojení s použitým frameworkem CakePHP 3 a zpracoval uživatelské dokumentace. Má práce v týmu spočívala v udržení správného směru vykonávané práce, programování back-end aplikace a prezentaci výsledků.

Nejdůležitější částí projektu je prezentace artefaktů v Google mapě v podobě tzv. markerů. Pro přidání nových artefaktů je nutné přihlášení uživatele. Každý nově přidaný artefakt se ihned zobrazí na Google mapě, jako marker označující oblast výskytu artefaktu. Pro zvýraznění některých artefaktů se okolo mapy vyskytují volitelné filtry pro zobrazení v daném časovém úseku nebo zobrazení artefaktů podle typu souboru (textový, obrázkový, audio, video). Markery je možné rozkliknout pro zobrazení podrobných informací o artefaktu. Výsledný projekt však nebyl nasazen do ostrého provozu z důvodu neošetřených chyb při vkládání souboru a nevyhovujícímu hostingu.



Obrázek 3.1: Webový portál pro správu kulturních artefaktů s mapou [7]

4 Specifikace

4.1 Úvod

4.1.1 Předmět specifikace

Cílem této práce je vytvoření webové aplikace pro sběr různých mluvených forem slov neboli nářečí. Tato hesla se liší pro určité oblasti České republiky, avšak jsou mezi oblastmi srozumitelná. Snahou tohoto webu je zachránit české nářečí a uchovávat ho v elektronické formě volně dostupné všem uživatelům. K jednotlivým heslům lze získat detailní popis vzniku, souvislostí a informací pro danou oblast. Web bude používat novodobý front-end a back-end framework pro efektivnější programování. Stejně tak bude upravena databáze, za účelem snížení přebytečných atributů a využití nově získaných dat. Projekt nahradí momentálně funkční nářeční slovník Západočeské univerzity a poslouží k záznamu, zachování a prezentaci vložených dat.

4.1.2 Typografické konvence

Ve specifikaci budou důležité informace vyznačovány **tučným písmem** např. názvy tabulek databáze, proměnné nebo elementy webu. *Kurzívou* budou značeny přesné názvy v případě potřeby poukázání na nějaký problém spojený s daným názvem. Ukázky programovacího kódu budou psány fontem `teletypefont family`.

4.1.3 Cílové publikum, návod ke čtení

Specifikace je hlavně určena pro odborné uživatele a administrátory webové aplikace. Celý projekt byl konzultován s doc. Zbyňkem Holubem, Ph.D., který jakožto zástupce cílového publika poukázal na problémy momentálně používaného elektronického nářečního slovníku ZČU.

4.1.4 Rozsah projektu

V tomto projektu elektronického nářečního slovníku je zahrnuto vytvoření webových stránek pomocí HTML a PHP jazyka s využitím front-end a back-end frameworků pro snadnější práci zejména s designem, serverovým zpracováním dat a komunikací s databází. Databáze poběží na volně dostupném MySQL softwaru, který se snadno implementuje a je podporován na mnoha

operačních systémech. V současné době stále funkční web elektronického nářečního slovníku ZČU, vypracovaný Jiřím Bergerem, poslouží jako výchozí bod při vytváření funkcí webové aplikace. Nový model databáze vychází z původního modelu databáze, který byl vypracován jako součást diplomové práce Jany Michalicové, pro elektronický nářeční slovník ZČU.

4.2 Obecný popis

Elektronický nářeční slovník není na internetu žádným unikátním dílem. Pro pomalu každý stát se již dají nalézt webové aplikace zaměřené na nářečí v různých oblastech. Některé weby jsou ovšem spravovány několikačlennými týmy a z hlediska podrobnosti hesel obsahují detailní a ověřené informace. Cílem této práce je vytvořit webovou aplikaci na úrovni týmových projektů. Taková aplikace bude ostatním konkurovat hlavně po kvalitativní stránce a bude se snažit dosáhnout vyšší návštěvnosti webu. Web proto bude navíc obsahovat seznam naposledy vyhledaných hesel pro znázornění aktivity na webu a oproti ostatním elektronickým nářečním slovníkům bude umožňovat přidávat nová hesla všem registrovaným uživatelům. Jednotlivá hesla bude možné najít v rejstříku nebo vyhledat pomocí vyhledávače na hlavní straně webové aplikace.

4.2.1 Kontext systému

Systém bude vycházet z elektronického nářečního slovníku ZČU. Spolu se zmiňovaným slovníkem je možné v tomto projektu využít části z projektu *Webový portál pro správu kulturních artefaktů*. Nejzajímavějším prvkem z této práce je Google mapa ukazující markery, kde se nacházejí dané artefakty. Tato funkce by mohla fungovat stejně, avšak pro jednotlivá nářeční hesla, která budou poté zobrazena markerem v oblasti výskytu. Mapa obsahuje filtry, které vymezují zobrazení daných artefaktů. Pro naše užití by mohlo zobrazení fungovat například pouze pro vybraný kraj, obec nebo město. Na implementaci mapy však nebude kladen důraz při vypracovávání této práce.

4.2.2 Funkce projektu

Hlavní funkcí tohoto projektu je poskytnout veřejnosti znalosti o nářečí z České republiky. Každý uživatel, který se chce podělit o heslo z určité oblasti České republiky, má po úspěšné registraci možnost vyplnit formulář vyskytující se na webu. Po zkontrolování věrohodnosti a případném doplnění informací administrátorem bude heslo schváleno a prezentováno na webu.

Jednotlivá hesla pak mohou být zobrazena i s detaily všemi uživateli webové aplikace.

4.2.3 Třídy uživatelů

Každý uživatel má možnost nahlédnout do uživatelské dokumentace, která se nachází v příloze B. Práva v systému jsou dále nastavena podle toho, do jaké skupiny daný uživatel patří. Skupiny se rozdělují na:

- Administrátor
  Nemá žádné omezení pro přidávání nových záznamů nebo komentářů. Má přístup ke správě všech oblastí a registrovaných uživatelů. Nově registrovaní uživatelé musí být nejdříve schváleni administrátorem.

- Registrovaný uživatel
  Může přidávat nové obce a hesla, která ovšem nejdříve musí projít kontrolou administrátora nebo správcem pro danou oblast.

- Správce oblasti
  Registrovaný uživatel, kterému byla administrátorem navíc přidělena práva pro editaci, smazání a akceptování hesel pro vybranou oblast.

- Běžný uživatel
  Má přístup pouze k prohlížení webu.

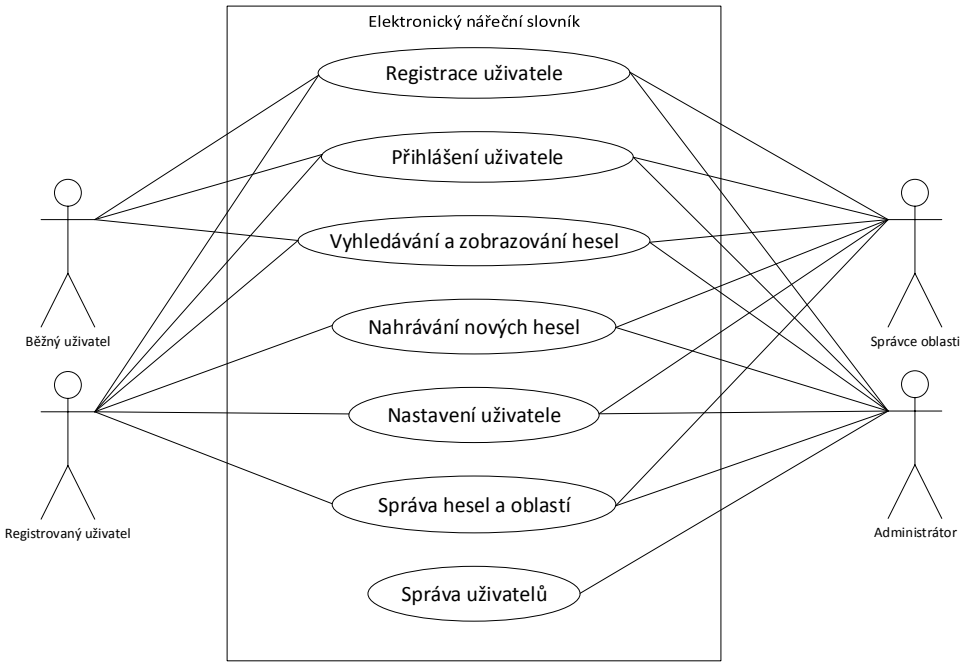
Podrobné zobrazení přístupných funkcí danému uživateli lze vidět na Obr. 4.1. Funkce z daného use case diagramu jsou detailně popsány v sekci 4.3.

4.2.4 Provozní prostředí a omezení návrhu a implementace

Systém poběží na výpočetní technice Západočeské univerzity v Plzni, kde k provozu stačí server, na kterém poběží databáze a webová aplikace.

4.2.5 Předpoklady a závislosti

Projekt závisí na aktivní komunitě a neustálém vývoji využívaných frameworků Material Design Lite a Laravel. Jelikož bude projekt nasazen na techniku Západočeské univerzity v Plzni, počítá se správnou funkčností tohoto vybavení.



Obrázek 4.1: Use case diagram systému

4.3 Funkce systému

V této kapitole jsou popsány funkce webové aplikace případem užití. Každá funkce obsahuje obecný popis, předpoklady na uživatele, výsledný stav, detailní průběh funkce, možné výjimky, prioritu, případné poznámky a uživatele, kteří mohou funkci využít.

4.3.1 Registrace

Uživatelé

Běžný uživatel, registrovaný uživatel, správce oblastí, administrátor.

Popis

Uživatel se registruje pomocí elektronického formuláře, kde musí vyplnit jméno, emailovou adresu, heslo a znovu heslo pro kontrolu. Přihlášení probíhá přes elektronický formulář zadáním emailu a hesla. Kontrola správnosti zadaných údajů funguje pomocí JavaScriptu u klienta a následně PHP validací vstupních dat na serveru. Nově registrovaný uživatel musí počkat na schválení účtu od administrátora.

Předpoklady

Žádné

Výsledný stav

Uživatel je registrovaný v systému.

Průběh

Registrace nového uživatele.

1. Uživatel vyplní registrační formulář a odešle data serveru.
2. Server provede validaci vstupních dat.
3. Server vytvoří záznam uživatele v databázi a nastaví úroveň autorizace na 0.
4. Server přesměruje uživatele zpět s notifikační zprávou.
5. Uživatel musí vyčkat, dokud nebude jeho účet schválen administrátorem.

Výjimky

Validace dat nebude úspěšná (krok 2)

1. Server přesměruje uživatele zpět na registrační formulář (krok 1) s chybovými hláškami.

Priorita

Vysoká

Poznámky

Očekávaná vyšší zátěž v průběhu prvního týdne od spuštění webu. Neošetření velkého počtu registrací (10+) za krátkou časovou dobu (<3min).

4.3.2 Přihlášení

Uživatelé

Běžný uživatel, registrovaný uživatel, správce oblastí, administrátor.

Popis

Přihlášení probíhá přes elektronický formulář zadáním emailu a hesla. JavaScriptem se kontroluje vzor emailové adresy u klienta. Po odeslání je provedena PHP validace vstupních dat na serveru a porovnání hesla na vstupu se zašifrovaným heslem v databázi spojeným s danou emailovou adresou. Pokud uživatel v systému existuje a jeho registrace byla schválena administrátorem, bude přihlášen.

Předpoklady

Registrovaný uživatel v systému a schválený administrátorem.

Výsledný stav

Uživatel je přihlášený v systému.

Průběh

Přihlášení registrovaného uživatele.

1. Uživatel vyplní přihlašovací formulář a odešle data serveru.
2. Server provede validaci vstupních dat.
3. Server zkontroluje, zda je uživatel schválený administrátorem.
4. Server porovná zadané heslo se zašifrovaným heslem v databázi.
5. Server si uloží data o uživateli do relace.
6. Server přesměruje uživatele na hlavní stránku s notifikační zprávou úspěšného přihlášení.

Výjimky

Validace dat nebude úspěšná (krok 2)

1. Server přesměruje uživatele zpět na přihlašovací formulář (krok 1) s chybovými hláškami.

Uživatel není dosud schválen administrátorem (krok 3)

1. Server přesměruje uživatele zpět na přihlašovací formulář (krok 1) s informativní zprávou.

Zadané heslo se neshoduje s heslem v databázi (krok 4)

1. Server přesměruje uživatele zpět na přihlašovací formulář (krok 1) s chybovou hláškou.

Priorita

Vysoká

4.3.3 Vyhledávání a zobrazování hesel

Uživatelé

Běžný uživatel, registrovaný uživatel, správce oblastí, administrátor.

Popis

Zadání hledaného řetězce znaků do prvku vyhledávače a zaškrtnutí jednoho z různých filtrů pro upřesnění hledání se po kliknutí na tlačítko **Hledat** odešle dotaz na server. Pro vyhledání lze vybrat filtr podle názvu hesla, významu hesla nebo obce výskytu hesla. Server vrací hesla obsahující daný řetězec znaků v podobě seznamu. U každého navráceného hesla se vypisuje heslo, výslovnost, význam a použití ve větě v případě existence těchto jednotlivých elementů.

Předpoklady

Žádné

Výsledný stav

Vyhledaná hesla zobrazená v seznamu.

Průběh

Vyhledávání hesla

1. Uživatel zaškrtne příslušný filtr.
2. Server zobrazí vyhledávač nebo seznam obcí pro vybraný filtr.
3. Uživatel zadá řetězec do vyhledávače nebo vybere obec ze seznamu a odešle data serveru.
4. Server provede funkci vyhledání hesel podle vybraného filtru.

5. Server přesměruje uživatele na stránku rejstříku a zobrazí vyhledaný seznam hesel.

Výjimky

Vyhledávané heslo neexistuje (krok 4)

1. Server přesměruje uživatele na stránku rejstříku a zobrazí zprávu oznamující nenalezení žádného hesla.

Priorita

Vysoká

Poznámky

Při zadání řetězce do vyhledávače a následně přepnutí filtru na oblast se vyhledávač schová. Zadaný řetězec ve vyhledávači je při odeslání dat na server ignorován.

4.3.4 Nahrávání nových hesel

Uživatelé

Registrovaný uživatel, správce oblastí, administrátor.

Popis

Registrovaní uživatelé mohou přidávat nová hesla v záložce **Nové heslo** na hlavním panelu. Do formuláře v daném panelu se vypisuje heslo daného nářečí, význam, výslovnost, oblast užití, slovní druh a podrobnější informace např. užití ve větě, příznak, synonymum, audio apod. Po odeslání se validuje vstup, vytvoří se nové heslo v databázi a čeká se na schválení správcem oblasti nebo administrátorem.

Předpoklady

Přihlášený uživatel v systému.

Výsledný stav

Heslo je s podrobnostmi nahrané v databázi.

Průběh

Nahrávání nového hesla.

1. Přihlášený uživatel vyplní formulář nového hesla a odešle data.
2. Server provede validaci vstupních dat a zkontroluje vyplnění povinných polí.
3. Server vytvoří slovní druh v databázi s případnými informacemi o podstatném jménu nebo slovesu.
4. Server uloží přikládaný audio soubor do vymezeného úložného prostoru.
5. Server vytvoří záznam hesla v databázi, připojí id hodnoty uživatele, slovního druhu, obce a vytvořené cesty k audio souboru a nastaví hodnotu schválení na 0.
6. Server přesměruje uživatele zpět s notifikační zprávou o úspěšném přidání hesla.

Výjimky

Validace dat nebude úspěšná (krok 2)

1. Server přesměruje uživatele zpět na formulář nového hesla (krok 1) s chybovými hláškami.

Nahrání audio souboru na server nebude úspěšné (krok 4)

1. Server přesměruje uživatele zpět na formulář nového hesla (krok 1) s chybovou hláškou.

Neexistuje žádný audio soubor ve vstupních datech pro uložení (krok 4)

1. Server přeskočí krok ukládání audio souboru (krok 4).

Priorita

Vysoká

Poznámky

Omezení na velikost audio souboru.

4.3.5 Nastavení uživatele

Uživatelé

Registrovaný uživatel, správce oblastí, administrátor.

Popis

Pro všechny registrované uživatele je v panelu **Nastavení** přidán formulář pro změnu jména, roku narození, původního bydliště a současného bydliště. Z tohoto místa je možné se ještě tlačítkem **Změna emailu a hesla** dostat k formuláři pro změnu přihlašovacích údajů. V případě vyplnění daných políček se kliknutím na tlačítko **Uložit** provede odeslání požadavku na server. Následně proběhne základní validační kontrola měněných prvků na straně serveru a v případě ověření správnosti se změní údaje v databázi. Při nevyplnění polí ve formuláři se ponechají v databázi staré hodnoty.

Předpoklady

Přihlášený uživatel v systému.

Výsledný stav

Změněné údaje uživatele.

Průběh

Změna nastavení osobních údajů.

1. Uživatel vyplní formulář osobního nastavení a odešle data serveru.
2. Server provede validaci vstupních dat.
3. Server uloží změny uživatele v databázi.
4. Server přesměruje uživatele zpět s notifikační zprávou.

4.3.6 Alternativní průběh

Změna nastavení soukromých údajů (krok 1)

1. Uživatel vyplní formulář soukromého nastavení a odešle data serveru.
2. Server provede validaci vstupních dat.
3. Server porovná zadané staré heslo s heslem v databázi.
4. Návrat na krok 3.

Výjimky

Validace dat nebude úspěšná (Změna nastavení osobních údajů - krok 2)

1. Server přesměruje uživatele zpět na formulář nastavení (krok 1) s chybovými hláškami.

Validace dat nebude úspěšná (Změna nastavení soukromých údajů - krok 2)

1. Server přesměruje uživatele zpět na formulář soukromého nastavení (krok 1) s chybovými hláškami.

Zadané heslo se neshoduje s heslem v databázi (Změna nastavení soukromých údajů - krok 3)

1. Server přesměruje uživatele zpět na formulář soukromého nastavení (krok 1) s chybovou hláškou.

Priorita

Střední

4.3.7 Správa hesel

Uživatelé

Správce oblastí, administrátor.

Popis

Administrátor a správce oblasti má přístup ke správě nově přidávaných hesel. Panel správy obsahuje tabulku s nově přidanými hesly, které je nutné zkontrolovat. Pokud bude vše v pořádku, heslo může označit jako akceptovatelné. V opačném případě může heslo zamítnout. Pokud jsou zadány špatné nebo nevyhovující údaje, je možné nové heslo také editovat pro opravení nebo doplnění správných informací. Správci oblasti bude správa hesla zobrazena pouze v případě, že oblast výskytu hesla bude spadat pod jeho spravované oblasti.

Předpoklady

Přihlášený administrátor nebo správce oblasti v systému.

Výsledný stav

1. Schválené heslo
2. Upravené heslo
3. Smazané heslo

Průběh

Schválení hesla

1. Server zobrazí odborným uživatelům neschválená hesla.
2. Uživatel na stránce neschválených hesel zaškrtne přepínač prováděné akce.
3. Uživatel klikne v tabulce na heslo, nad kterým chce akci provést.
4. Server zkontroluje, zda je uživatel autorizovaný k dané akci.
5. Server schválí dané heslo nastavením hodnoty schválení na 1.
6. Server přesměruje uživatele zpět s notifikační zprávou.

Alternativní průběh

Upravení hesla (po kroku 4)

1. Uživatel je přesměrován na formulář editace hesla.
2. Uživatel upraví/doplní informace o heslu a odešle data na server.
3. Server provede validaci vstupních dat.
4. Server smaže záznam slovního druhu, vytvoří nový a uloží změny daného hesla.
5. Návrat na krok 6.

Smazání hesla (po kroku 4)

1. Server zobrazí uživateli dialog okno s potvrzením o smazání hesla.
2. Uživatel potvrdí smazání hesla a odešle požadavek na server.
3. Server smaže propojený záznam slovního druhu a následně i samotné heslo.
4. Návrat na krok 6.

Výjimky

Server neobsahuje žádná hesla k zobrazení (Schválení hesla - krok 1)

1. Server zobrazí uživateli notifikační zprávu o žádných heslech ke schválení (přeskočení všech kroků).

Uživatel nebude autorizovaný k dané akci (Schválení hesla - krok 4)

1. Server přesměruje uživatele na hlavní stránku s chybovou hláškou.

Vstupní data neprojdou validací (Upravení hesla - krok 3)

1. Server přesměruje uživatele zpět na formulář editace hesla (krok 1) s chybovými hláškami.

Uživatel nepotvrdí smazání hesla (Smazání hesla - krok 2)

1. Server zavře dialog okno.

Priorita

Vysoká

4.3.8 Správa uživatelů

Uživatelé

Administrátor.

Popis

Panel **Seznamu uživatelů**, který je přístupný všem registrovaným uživatelům, obsahuje pouze pro administrátora funkce pro správu uživatelů. V případě registrace nového uživatele se zobrazuje nová tabulka s akcemi *Akceptování* nebo *Smazání* uživatele. Všichni registrovaní uživatelé se zobrazují v tabulce níže. Registrovaným uživatelům lze prohlížet profil, editovat osobní údaje (jméno, rok narození, bydliště), přiřazovat správu oblasti nebo je možné je smazat z databáze. Pro smazání uživatele je nutné tuto akci potvrdit heslem administrátora. Akce se provádí zaškrtnutím přepínače a kliknutím na pole s uživatelem v tabulce.

Předpoklady

Přihlášený administrátor v systému.

Výsledný stav

1. Schválený uživatel
2. Uživatel s upravenými osobními údaji
3. Uživatel s právy správy oblastí
4. Smazaný uživatel

Průběh

Schválení nově registrovaného uživatele.

1. Administrátor na stránce seznamu uživatelů zaškrtne přepínač prováděné akce.
2. Administrátor klikne v tabulce na uživatele, nad kterým chce akci provést.
3. Server zkontroluje, zda je přístup k akci autorizovaný.
4. Server nastaví vybranému uživateli úroveň autorizace na 1.
5. Server odešle uvítací email na registrovanou adresu.
6. Server přesměruje administrátora zpět s notifikační zprávou.

Alternativní průběh

Úprava osobních údajů uživatele (po kroku 3)

1. Administrátor je přesměrován na formulář nastavení uživatele.
2. Administrátor upraví/doplní informace o uživateli a odešle data na server.
3. Server provede validaci vstupních dat.
4. Server uloží změny osobních údajů daného uživatele.
5. Návrat na krok 6.

Přidání správy oblastí uživateli (po kroku 3)

1. Server zobrazí administrátorovi formulář s tabulkou oblastí.

2. Administrátor zaškrtne oblasti, které chce uživateli přidat ke správě a případně odškrtne ty, které mu chce odebrat.
3. Administrátor odešle požadavek na server.
4. Server upraví uživateli správu oblasti podle zaškrtnutých oblastí a vytvoří/smaže záznamy propojení uživatele s oblastmi v databázi.
5. Návrat na krok 6.

Smazání uživatele (po kroku 3)

1. Server zobrazí uživateli dialog okno s potvrzením o smazání uživatele.
2. Administrátor potvrdí smazání uživatele zadáním svého hesla a odešle požadavek na server.
3. Server smaže uživatele ze systému.
4. Návrat na krok 6.

Výjimky

Uživatel nebude autorizovaný k dané akci (Schválení nově registrovaného uživatele - krok 3)

1. Server přesměruje uživatele na hlavní stránku s chybovou hláškou.

Vstupní data neprojdou validací (Úprava osobních údajů uživatele - krok 3)

1. Server přesměruje uživatele zpět na formulář nastavení uživatele (krok 1) s chybovými hláškami.

Administrátor nepotvrdí smazání uživatele (Smazání uživatele - krok 2)

1. Server zavře dialog okno.

Zadané heslo se neshoduje s heslem administrátora v databázi (Smazání uživatele - krok 2)

1. Server přesměruje uživatele zpět na dialog (krok 1) s chybovými hláškami.

Priorita

Vysoká

4.3.9 Nahrávání nových oblastí

Uživatelé

Registrovaný uživatel, správce oblastí, administrátor.

Popis

V panelu **Seznam měst** mají všichni registrovaní uživatelé přístup k tlačítku **Přidat novou oblast**, které po stisknutí otevře dialog. V tomto dialogu se vyplňuje název obce, okresu a kraje. Po potvrzení zadaných hodnot se data odešlou na server a provede se validace vstupu a kontrola duplicity oblasti. Chybějící obec se nahraje do databáze a bude přístupná pro všechny funkce, které obce využívají. Administrátor má navíc volbu smazat některou oblast v případě nevyhovující oblasti.

Předpoklady

Přihlášený uživatel v systému.

Výsledný stav

Nový záznam obce v databázi.

Průběh

Přidání nové oblasti

1. Uživatel stiskne tlačítko pro přidání nové oblasti.
2. Server zobrazí dialog okno s formulářem pro přidání nové oblasti.
3. Uživatel vyplní formulář a odešle data na server.
4. Server provede validaci vstupních dat.
5. Server vytvoří nový záznam obce v databázi.
6. Server přesměruje uživatele zpět s notifikační zprávou.

Výjimky

Validace dat nebude úspěšná (krok 4)

1. Server přesměruje uživatele zpět na dialog (krok 2) s chybovými hláškami.

Uživatel zadá údaje již existující údaje v databázi (krok 3)

1. Server přesměruje uživatele zpět na dialog (krok 2) s chybovou hláškou.

Priorita

Střední

4.4 Požadavky na vnější rozhraní

4.4.1 Uživatelská rozhraní

Aplikace bude splňovat webové standardy, které lze ověřit na validační webové stránce od W3Schools [8]. Pro kontrolu mobilních rozlišení, standardně menší šířky rozlišení než 768px, se využije validační webová stránka MobileTest [9] a mobilní telefon typu Samsung Galaxy S4 LTE-A.

4.4.2 Hardwarová rozhraní

Pro databázi se očekává kvalitně vybavená serverovna s HDD nebo SSD disky starými v nejlepším případě do 10 let. Počítá se samozřejmě se síťovým připojením od Západočeské univerzity, kde je služba poskytující internet stabilní, kromě plánovaných výpadků. Z pohledu uživatele je zapotřebí jakékoli zařízení s možností připojení na internet a existencí webového prohlížeče v daném zařízení.

4.4.3 Softwarová rozhraní

Webová aplikace nezaznamenává žádné rozdíly, pokud je spuštěna na různých operačních systémech. Databáze webové aplikace běží na databázovém systému MySQL. Je nutný, aby server splňoval následující požadavky pro správný chod Laravel frameworku [10]:

- PHP >= 5.6.4
- OpenSSL PHP rozšíření
- PDO PHP rozšíření
- Mbstring PHP rozšíření
- Tokenizer PHP rozšíření
- XML PHP rozšíření

Klient využívá k používání webové aplikace webové prohlížeče.

4.4.4 Komunikační rozhraní

Celá komunikace mezi serverem a klientem probíhá přes webový prohlížeč. Systém bude využívat elektronického formuláře pro registraci uživatele a nahrání nového hesla. Při registraci se vyplňuje email, který bude moci být využit pro kontaktování daného uživatele administrátorem.

4.5 Další parametrické požadavky

4.5.1 Výkonnostní požadavky

Webová aplikace zvládne zpracovávat aktivitu v řádech desítek najednou aktivních uživatelů. Jakékoliv zařízení má v současné době s přístupem na internet dostatečný výkon pro načtení webové stránky.

4.5.2 Bezpečnostní požadavky

Webová aplikace bude obsahovat PHP ochrany, aby bylo zabráněno přístupu neoprávněných uživatelů k uživatelským datům jako např. email, která by mohla být následně použita pro jiné účely. Data jsou ukládána do databáze, kde každý přístup je kontrolován v samotném back-endu webové aplikace před různými útoky (CSRF, XSS, apod.). Serverovna samotná, ve které jsou data uchovávána, musí být v čístém bezprašném prostředí a dostatečně klimatizována. Tato místnost musí také dodržovat jiné bezpečnostní požadavky např. hasící přístroje, kvalitní síťová infrastruktura, náhradní zdroje napájení.

4.5.3 Kvalitativní parametry

Kvalita webové aplikace je popisována požadavky:

- Snadnost používání
  Elektronické formuláře jsou popsány hesly s nápovědou v případě potřeby. Administrátor a správce oblastí musí znát základní funkce spravování přidávaných hesel.

- Spolehlivost
  Provoz bude nepřetržitý, a proto je důležité, aby nenastala havárie v serverovně. Je důležité otestování všech nevalidních vstupů či dotazů na databázi pro korektní chování webové aplikace na nevalidní zprávy.

4.6 Testování

Z hlediska bezpečnosti a zajištění plné funkčnosti celé webové aplikace je nutné se věnovat také testování. Testování se bude provádět pomocí metod integrované knihovny **Laravel Dusk** ve frameworku Laravel, která je velmi dobře popsána v dokumentaci. Současná práce však spoléhá na správnou funkcionalitu implementovaných funkcí a při výskytu chyb ve frameworku se mohou chyby promítnout také do testování v bakalářské práci. Reálný odhad času strávený testováním této práce by měl být kolem 20 hodin. Z časového odhadu je 5 hodin vyhrazeno na naučení se práce s výše zmiňovanou knihovnou a zbylých 15 hodin na programování testů.

5 Analýza frameworků

Pro usnadnění práce s programováním webu je možné využít k programování již mnoho existujících populárních open source frameworků. Tyto frameworky mnohdy nahrazují rozsáhlé bloky kódu jednoduchou jednořádkovou syntaxí, čímž ušetří programátorovi drahocenný čas. Provedená analýza je shrnuta do tabulky, ze které se vybere nejvhodnější framework, jak pro front-end tak pro back-end této webové aplikace.

5.1 Front-end frameworky

Open source frameworky zaměřené na front-end velmi ulehčují práci se vzhledem webové aplikace. V současné době se využívá hlavně značkovací jazyk (HTML) s kombinací kaskádových stylů (CSS) při vytváření webových aplikací. Tyto frameworky jsou napsány právě převážně ve formátu CSS. Každý takový framework má zdrojový kód spravovaný na Githubu a při hodnocení bude tedy vycházeno z údajů, které lze z této webové služby vyčíst. Zde jsou vybrány frameworky, které byly vybrány na základě vysoké popularity a analýzy z webu Keycdn [11]:

Material Design Lite (MDL)

Google přišel s poměrně novým frameworkem vydaným v polovině roku 2015 založeném na jejich designovém jazyku Material Design. Framework se soustředí především na rychlost, animace prvků, spolehlivost responzivity a podporu všech aktuálně používaných webových prohlížečů. Skutečnost, že framework je využíván ve webových aplikacích od Google, by mohla vypovídat i o záruce současné kvality a dobré vizi do budoucna. Popularita Material Design Lite nabývá po stovkách nových sledujících uživatelů na Githubu každým měsícem a s novými updaty přibývá ve frameworku mnoho nových funkcí. Populární stránky využívající tento framework jsou např.: Google Wallet, Google Developers a Skillmapped.

Bootstrap

V současné době nejpopulárnější framework pro designování webových stránek pomocí HTML, CSS a JS. Díky statisícové základně fanoušků framework obsahuje velké množství vlastností a funkcí. Při problémech implementace nebývá složité najít existující řešení na internetu do několika minut. Nejsou

však podporovány některé webové prohlížeče, jmenovitě PC verze Safari, Internet Explorer 7 a Microsoft Edge. Toto by ale mělo být napraveno novou verzí Bootstrap 4.0, která je v současné době v testovací alpha verzi. Populární stránky využívající tento framework jsou např.: Spotify, NASA, FIFA, Vevo.

Foundation

První front-end framework profesionálně podporovaný organizací (ZURB) je jako Bootstrap založený na HTML, CSS a JS a obsahuje mnoho unikátních šablon pro rychlý začátek při tvorbě front-end webu. Na rozdíl od Bootstrapu je více přizpůsobitelný v tvorbě webového designu a také má implementované základní CSS styly, tudíž není nutné přidávat třídu ke každému prvku. Populární stránky využívající tento framework jsou např.: Lamborghini, Whirpool, Pixar a Ford.

Semantic UI

Poslední porovnávaný front-end framework je zaměřený spíše na pokročilejší vývojáře. Základní instalace projektu obsahuje mnoho volitelných motivů pro vzhled prvků. Oproti jiným frameworkům se není možné obejít bez základních znalostí Javascriptu, a to z důvodu, že skoro všechny části frameworku bez JS nebudou fungovat správně. Jednotlivé syntaxe tříd se zapisují jako celá slova, čímž je programování více intuitivní a sémantické. Nedávno vydaná verze 2.2 z června 2016 a velká početní podpora ze strany vývojářů [12] naznačuje aktivitu a aktuálnost tohoto frameworku. Populární stránky využívající tento framework jsou např.: Snapchat, Codility a Diigo.

Shrnutí

V následující tabulce 5.1 je rychlé shrnutí výše zmiňovaných front-end frameworků. Všechna data jsou shromažďována z webové služby Github. *Popularita* je reprezentována počtem lidí tzv. **Stargazers**, kteří projektu vyjádřili podporu a zájem přidáním si projektu do svých záložek. Takto uložený projekt v záložce následně upozorňuje dané uživatele na nové aktualizace a novinky. *Počet podílejících vývojářů* je vyčten ze stejného Github projektu, avšak podle počtu **Contributors**, což jsou lidi podílející se na vývoji daného projektu.

Ze souhrnu populárních front-end frameworků bude v tomto projektu použit Material Design Lite, který vychází z vzhledu Material Design využívaným v mobilních zařízeních se systémem android [13]. Přibližný průměr

	Rok vzniku	Popularita	Počet podílejících se vývojářů	Licence	Výsledné pořadí
Material Design Lite	2015	25 862	167	Apache-2.0	1
Bootstrap	2011	106 423	831	MIT	2
Foundation	2011	24 902	909	MIT	4
Semantic UI	2013	31 194	163	MIT	3

Tabulka 5.1: Shrnutí analýzy front-end frameworků ze dne 2.2.2017.

vzrůstu popularity za poslední rok je 13 000 sledujících uživatelů. Oproti Bootstrapu, který čítá přibližný průměrný vrůst popularity 17 500 sledujících uživatelů za rok, však statisícová popularita naznačuje, že mnoho webových aplikací je založeno na tomto frameworku. S poměrně mladým frameworkem Material Design Lite je tedy zakládáno na větší originalitě designu než s často užívanými styly Bootstrapu. Frameworky Foundation a Semantic UI jsou si s MDL přibližně rovny v počtu Popularity, přestože MDL je přibližně o 3 roky mladší.

5.2 Back-end frameworky

Backend webové aplikace se dá napsat v několika programovacích jazycích např. Python, Ruby, Java, PHP. Pro každý takový jazyk existuje samozřejmě i velké množství frameworků. Tato webová aplikace bude vytvořena v jazyce PHP z důvodu podpory velkého množství databází, kompatibility s valnou většinou hostingových služeb a možnosti běhu na více platformách. Společně s těmito výhodami souvisí také větší vlastní zkušenosti při vytváření back-endu webové aplikace s programovacím jazykem PHP oproti jazykům ostatním. A proto se tento projekt na PHP frameworky zaměřuje.

Symfony

Velká reputace, která předchází tento framework, je založena na rychlosti, flexibilitě a jednoduchosti. Funguje již od roku 2005, kdy si ho vývojáři rychle oblíbili, a svoji popularitu si drží dodnes. Symfony využívá vlastní nezávislé komponenty pro vytváření modulárních webových aplikací a stránek. Projekt se stále drží mezi špičkou PHP frameworků hlavně díky tisícovému počtu podílejících se vývojářů, kteří vydávají aktualizace průměrně jednou za měsíc.

Laravel

Framework postavený na back-end frameworku Symfony je rychlý a jednoduchý na ovládání. Využívá MVC architekturu a speciální šablonovací engine

zvaný Blade, který neomezuje v používání prostého PHP ve view komponentách. Elegantní zápis a čitelnost syntaxe je věc, která je velmi chválena mezi programátory webových aplikací. Avšak hlavní předností je skvěle zpracovaná dokumentace, video návody a bezmála třicetitisícová komunita. Tím, že má Laravel v sobě zabudované možnosti pro testování pomocí PHPUnit, Dusk nebo Mock rozšíření, umožňuje testovat bezpečnost bez stahování dalších programů.

CakePHP 3

Specifický framework také založený na MVC architektuře je specifický hlavně díky vlastnímu přístupu propojení databáze s webovou aplikací. Dokumentace je stručná a místy nepřehledná, avšak dostačující. Přesto se naučit využívat všechny funkce tohoto frameworku trvá déle než u jiných, kvůli využívání pokročilejších funkcí v PHP. Takové funkce jsou v jiných frameworcích již implementovány v jednoduchých metodách (např. ověření uživatele). Komunita frameworku je však v řádech tisíců, a proto ne všechny dotazované problémy byly na internetu zodpovězeny.

Shrnutí

Tabulka 5.2 pro shrnutí back-end frameworků je prezentována stejným stylem jako tabulka 5.1 pro front-end frameworky. Data jsou také převzata z webové služby Github. Popularita je znázorněna počtem tzv. **Stargazers** a počet podílejících vývojářů podle počtu **Contributors**.

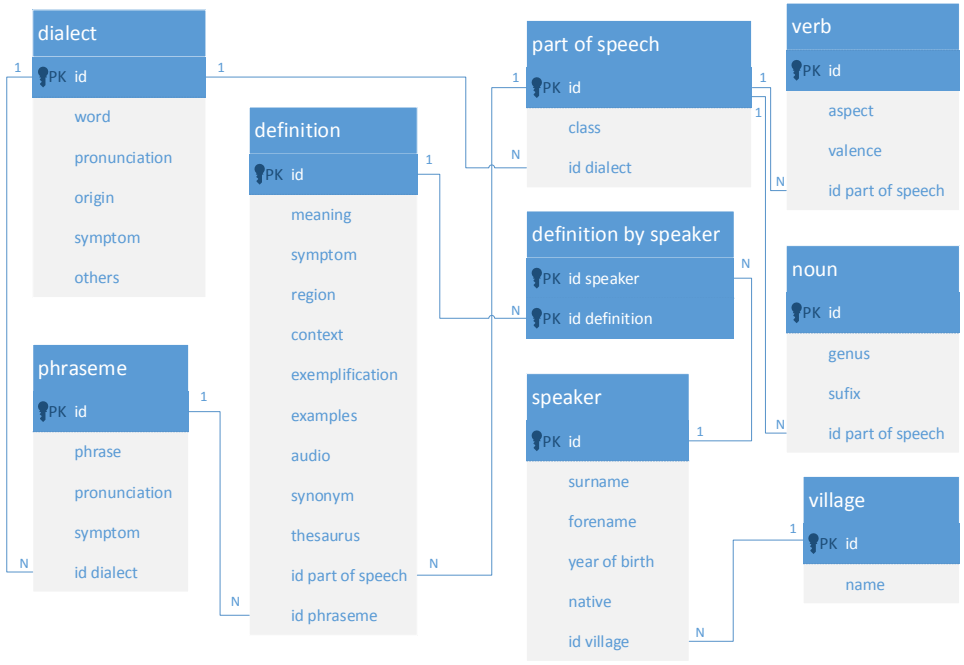
	Rok vzniku	Popularita	Počet podílejících vývojářů	Licence	Výsledné pořadí
Symfony	2010	13 783	1 408	MIT	3
Laravel	2011	29 337	388	MIT	1
CakePHP 3	2005	6 928	455	MIT	2

Tabulka 5.2: Shrnutí analýzy back-end frameworků ze dne 2.2.2017

Pro práci na back-end straně webové stránky bude použit framework Laravel. Hlavním důvodem je více jak dvojnásobná popularita oproti Symfony frameworku s popularitou necelých 14 000 a uživatelsky příjemně propracovaná dokumentace. I přesto, že mám zkušenosti s prací v CakePHP 3 a následná práce s ním by mohla být jednodušší, věřím, že vyhlídky do budoucna jsou lepší s Laravelem. Oproti Symfony je Laravel přívětivější novým vývojářům. K výběru použití Laravelu bylo přihlédnuto také k podpoře testovacích rozšíření frameworku, které umožňují usnadnit jednu z plánovaných částí bakalářské práce.

6 Návrh databázového modelu

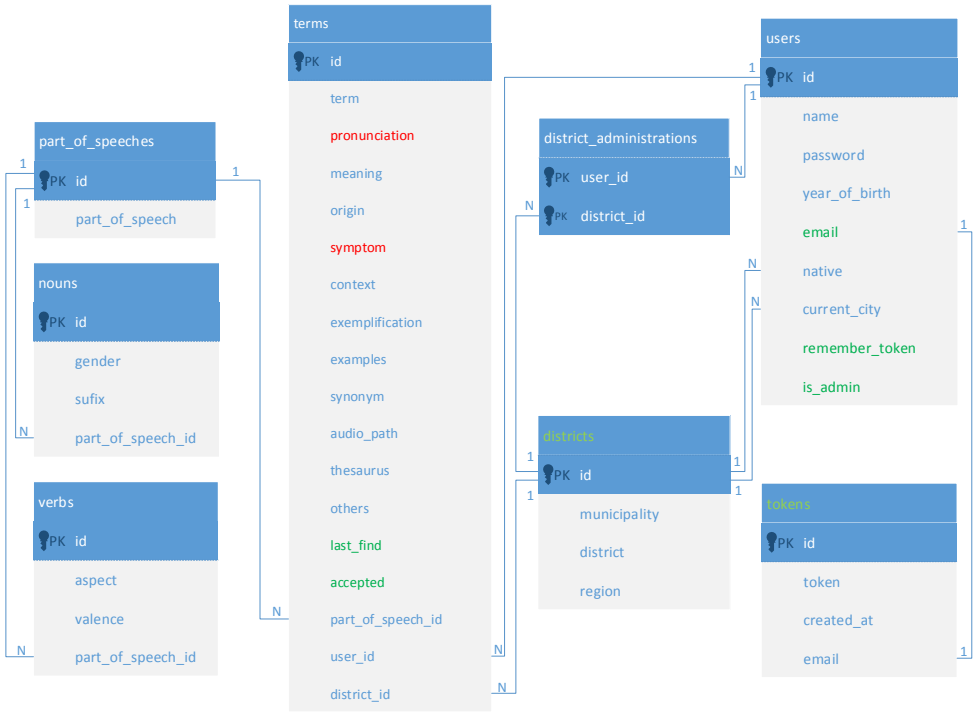
Pro návrh nového databázového modelu je použita momentálně používaná databáze nářečního slovníku ZČU, popsaná v diplomové práci Jany Michalcové [14]. Nová databáze vychází z původního modelu databáze viz Obr. 6.1. Atribut příznaku **symptom** obsahuje zkratkovitý zápis stylově zabarveného



Obrázek 6.1: ERA diagram původní databáze [14]

slova. Atribut výslovnosti **pronunciation** je, jak český název napovídá, výslovnost daného hesla. Zmíněné atributy se nacházejí v tabulkách nářečního hesla **dialect**, fráze **phraseme** a významu **definition**. Data těchto atributů při detailním pohledu do databáze byla stejná nebo zanedbatelná, kdy v databázi existovalo 5 frazémů na 1764 hesel. Při návrhu nové databáze pro tuto webovou aplikaci se tedy tyto duplicitní atributy odebraly a zbylé atributy ve vybraných tabulkách sloučily do jedné tabulky s názvem **terms**, čímž vzniká univerzálnější model viz Obr. 6.2. Na obrázku jsou červenou barvou znázorněny atributy, které vznikly sloučením duplicitních atributů původní

databáze. Zelená barva zase zvýrazňuje tabulky nebo atributy, které v modelu přibyly. Ostatní atributy zůstaly nepozměněny nebo se změnilo pouze jejich přemístění v tabulkách, avšak jejich funkce zůstává stejná. Zanedbatelná data nebyla převedena, avšak v případě potřeby je možné je manuálně převést. Tento databázový model byl také upraven pro vhodnější integraci dat obcí, která jsou převzata z databáze webové aplikace pro správu kulturních artefaktů. Z tabulek nového modelu však nejsou smazány žádné důležité atributy, které by mohly znamenat problém při migraci dat ze staré databáze do nové. Všechny tabulky jsou pojmenovány anglickými výrazy v množném čísle. V této kapitole jsou všechny tabulky a atributy nového modelu databáze popsány. Ke každé tabulce je také napsána vazba na tabulky s ní propojené.



Obrázek 6.2: ERA diagram upravené databáze

6.1 Terms

Tato tabulka uchovává všechna vytvořená hesla s podrobným popisem. Informace o heslu se zobrazují v katalogu hesel a náhledu hesla. Tabulka se váže relací N:1 k tabulkám **part_of_speech**, **users** a **districts**. Níže jsou rozepsány podrobnosti uchovávaných dat této tabulky.

- **term**: Jednoslovný název hesla nebo slovní spojení. Nejdůležitější atribut tabulky.
- **pronunciation**: Výslovnost hesla v mluvené formě, která se může lišit od formy písemné.
- **meaning**: Význam přidávaného nářečního hesla.
- **origin**: Zkratka jazyka, ze kterého heslo pochází.
- **symptom**: Zkratkovitý zápis stylově zabarveného hesla (archaické, lichotné, odborné, apod.).
- **context**: Příklad užití slova ve větě. V případě více vět se záznamy rozdělují podle tečky jako ukončovacího znaku.
- **exemplification**: Vysvětlení souvislosti hesla ve větě. V případě více vět užití se tyto věty rozdělují tečkou.
- **examples**: Reálné užití hesla vztažené k významu.
- **synonym**: Slova, která mají podobný nebo stejný význam jako význam zadávaného hesla.
- **thesaurus**: Odkaz na význam daného hesla vyskytující se v Tezauru.
- **others**: Ostatní informace spojené s heslem.
- **last_find**: Časová značka představující poslední vyhledání hesla.
- **accepted**: Atribut nabývající hodnot pouze 0 a 1. Představuje, zda je dané heslo schválené, nebo není.

6.2 Users

Data o uživatelích, kteří se registrují v této webové aplikaci, se ukládají do této tabulky. Podrobné údaje o uživatelích jsou využívány převážně pro ověření věrohodnosti přispívaných hesel. Tabulka se váže relací 1:N k tabulkám **terms**, **district_administration**, N:1 k tabulce **districts** a relací 1:1 k tabulce **tokens**.

- **name**: Celé jméno nebo přezdívka uživatele.
- **password**: Heslo uživatele zašifrované AES šifrováním.

- **year_of_birth**: Čtyřmístné číslo zobrazující rok narození uživatele.
- **email**: Unikátní email uživatele v databázi, který slouží k přihlašování společně s heslem.
- **native**: Cizí klíč na tabulku měst představující původní bydliště uživatele.
- **current_city**: Cizí klíč na tabulku měst představující současné bydliště uživatele.
- **remember_token**: Náhodně vygenerovaný řetězec znaků a čísel, který se používá k uložení relace uživatele pro zapamatování přihlášení. Tento řetězec se vygeneruje znovu s každým novým přihlášením.
- **auth_level**: Hodnota nabývající čísel 0, 1, 2 poukazující na úroveň pověření ve webové aplikaci.

6.3 Districts

Záznamy obcí z celé České republiky převzaté z webové aplikace pro správu kulturních artefaktů. Záznamy byly upraveny do vlastního formátu z dat poskytnutých Ministerstvem vnitra České republiky [15]. Vázána relacemi 1:N k tabulkám **terms**, **users** a **district_administration**.

- **municipality**: Název obce
- **district**: Název okresu, ve kterém se obec nachází.
- **region**: Název kraje, ve kterém se obec nachází.

6.4 District administration

Potřebné záznamy zobrazující propojení mezi uživateli a oblastmi přiřazující správu hesel nad danou oblastí. Návrh projednává o správě podle jednotlivých obcí. Výsledná implementace se však může lišit. V případě správy podle okresů nebo krajů budou uživatelé ukládáni do tabulky s první nalezenou obcí v daném okresu nebo kraji. Ukládá pouze primární klíče ve vztahu N:1 tabulek **users** a **districts**, čímž vytváří mezi těmito dvěma tabulkami relaci N:M.

6.5 Tokens

Obsahuje vygenerované hash klíče, které se kontrolují porovnáním při potřebě změny hesla uživatele. Existuje zde pouze relace 1:1 k tabulce **users**.

- **token**: Náhodně vygenerovaný řetězec znaků a čísel.
- **created_at**: Časová značka vytvořená v době vzniku nového záznamu.
- **email**: Email uživatele požadujícího změnu hesla.

6.6 Part of speeches

Tabulka obsahující slovní druh, který je podle id přiřazován k heslu. V případě, že je slovní druh podstatné jméno nebo sloveso, je id slovního druhu přiřazeno k odpovídajícímu záznamu v tabulce podstatného jména **nouns** nebo slovesa **verbs**. Relací 1:N vázána k tabulkám **terms**, **nouns** a **verbs**.

- **part_of_speech**: Slovní druh

6.7 Nouns

Pokud je slovní druh vytvářeného hesla podstatné jméno, je vytvořen záznam v této tabulce, který obsahuje doplňující informace hesla. Tabulka se váže relací N:1 k tabulce **part_of_speech**

- **gender**: Rod podstatného jména.
- **sufix**: Koncovka hesla ve 2. pádu.

6.8 Verbs

Podobně jako ve výše zmiňované tabulce **nouns** je záznam v této tabulce vytvořen v případě, že slovní druh je sloveso. Obsahuje doplňující informace pro heslo. Vázána relací N:1 k tabulce **part_of_speech**.

- **aspect**: Slovesný vid hesla
- **valence**: Valence hesla podle možnosti na sebe vázat další větné členy.

7 Implementace

Po analýze všech potřebných dat a návrhu databázového modelu přichází na řadu implementace webové aplikace. Zprvu bylo možné využít již existující řešení některých funkcí z projektu ZSWI[1] webové aplikace pro správu kulturních artefaktů, avšak práce byla vytvářena ve frameworku CakePHP 3. Při výběru back-end frameworku pro tuto práci byl vybrán Laravel viz [5.2] a propojení metod mezi těmito frameworky by nadělalo více problémů než užitku.

Řešení jednotlivých funkcí bylo tedy vytvářeno vlastním způsobem s pomocí dokumentace Laravelu [16]. Z velké části však také dopomohly k lepším zkušenostem programování s Laravelem video návody, které vývojáři Laravelu volně poskytují na jejich webové stránce Laracast [17]. Zkušenosti s front-end frameworkem Material Design Lite, který je využívaný pro tuto webovou aplikaci, byly čerpány z příkladů jednotlivých komponentů frameworku [18].

Níže jsou popsány a zobrazeny části kódu, které byly pro dosažení výsledného implementačního řešení problematické nebo nějakým způsobem zajímavé.

7.1 Migrace databáze

Nejprve je nutné správně přesunout data ze staré databáze do nové. V této práci je zobrazený starý model databáze s tabulkami pojmenovanými anglickými názvy. Anglické pojmenování bylo provedeno mnou, pro rychlejší orientaci mezi obrázky těchto databází popisovaných v kapitole 6. Původně však byly tabulky modelu staré databáze pojmenované českými názvy, a proto se v následujících ukázkách SQL[2] kódu budou vyskytovat správná pojmenování oproti dříve představovanému obrázku starého modelu.

Tabulky podstatných jmen a sloves se nezměnily a data tak byla přesunuta pouze se změnou názvů atributů. Stejně tak to platí pro slovní druh, který navíc obsahuje id hesla, které je použito pouze při přesunu záznamů významu ke vztahovanému heslu. Prvním problémem bylo propojení záznamů měst nové a staré databáze. Obce obsažená v původní databázi se manuálně přidávala přes textové pole přímo na stránkách webu. Jelikož byla takto

[1] Základy softwarového inženýrství

[2] Structured Query Language

zapsaná obec bez určení příslušného okresu a regionu, do kterého spadala, nebylo možné přesně zjistit, o jakou obec se jedná. Proto byla tato tabulka se 24 záznamy smazána a ke vztahujícím se cizím klíčům nastavena nulová hodnota.

Sloučování atributů tří tabulek **heslo**, **vyznam** a **frazem** ze staré databáze do jedné tabulky **terms** představovalo největší komplikace. Jednodušší část vložení atributů *heslo*, *výslovnost*, *původ*, *příznak* a *ostatní* z tabulky **heslo** do tabulky **terms** prošla bez problémů použitím dotazu **INNER JOIN**. Význam hesel v tabulce **vyznam** se však vázal na hesla v tabulce **heslo** přes tabulku slovních druhů **druh**, která obsahuje id hesla. Pro tento případ tak bylo potřeba propojit tabulky **druh** a **vyznam** dotazem **LEFT JOIN**, a dotazem **INNER JOIN** pak propojit tabulky **druh** a **terms** pro aktualizaci atributů v nové tabulce hesel viz Kód 7.1. Před aktualizací atributů bylo zkontrolováno kolem 30 záznamů hesel, které obsahovaly v tabulce **vyznam** vyplněný atribut *priznak*. Při kontrole však nedošlo k žádnému rozporu mezi atributy *priznak* v tabulkách **vyznam** a **heslo**, a proto byl atribut *priznak* v tabulce **terms** aktualizován podle tabulky **vyznam**. V SQL kódu lze zpozorovat vynechání atributu *oblast*, který by se musel do nové databáze převádět na atribut *district_id*. Oblasti ve staré databázi však byly zapisovány zkratkami nebo názvem obce, které pro náš případ nelze převzít. Jedním z hlavních důvodu je také to, že záznamy oblastí v nové databázi mohou obsahovat více stejných názvů obcí, avšak patřících pod jiný okres či kraj. Tímto způsobem není možné dohledat, přesnou obec, ve které se heslo užívalo. Prvek výskytu hesel v obcích se tedy bude muset upravit již ve funkčním systému pomocí webového rozhraní.

```
UPDATE terms
INNER JOIN druh ON terms.id = druh.id_heslo
LEFT JOIN vyznam ON druh.id = vyznam.id_druh
SET
terms.meaning = vyznam.vyznam,
terms.symptom = vyznam.priznak,
terms.context = vyznam.kontext,
terms.exemplification = vyznam.exemplifikace,
terms.examples = vyznam.uziti,
terms.thesaurus = vyznam.tezaurus
```

Kód 7.1: Propojení hesel se slovním druhem

Tabulka frazémů, u které byly po analýze ignorovány atributy výslovnosti a příznaku, obsahovala pouze 5 záznamů, přestože hesel existovalo v databázi 1728. Z tohoto důvodu se došlo k řešení, že atribut *frazem*, obsahující bude

manuálně převeden do nové tabulky hesel na pozici atributu *exemplification*, kde jsou jednotlivé příklady užití ve větě oddělovány tečkou.

Slovní druh je k heslům ze staré databáze propojen cizím klíčem *id_heslo*, obsaženým v tabulce **druh**. Pro migraci toho atributu je tedy třeba propojit tento cizí klíč s primárním klíčem hesla v nové tabulce **terms**. Výsledný SQL dotaz pro aktualizaci cizího klíče slovního druhu lze pak vidět v následujícím kódu 7.2.

```
UPDATE terms
INNER JOIN druh ON terms.id = druh.id_heslo
SET terms.part_of_speech_id = druh.id
```

Kód 7.2: Propojení hesel se slovním druhem

Těmito akcemi jsou nyní přesunuty všechna data ze staré databáze do nové. Kromě problému s výskytem hesel v oblasti se při migraci žádná data neztratila ani nesmazala. Databáze je tedy připravena na provoz ve webové aplikaci.

7.2 Struktura aplikace

Web bude využívat dnes již rozšířeného architektonického vzoru MVC[3]. Hlavním technickým prvkem tohoto vzoru je rozdělení logického zpracovávání od výstupu. MVC architektura se rozděluje na tři komponenty, kde každá část má svoji úlohu, kterou provádí při dotazu uživatele na web. V případě uživatele požadující nějaké informace (chce zobrazit stránku například "users.php") se zpracuje požadavek s parametry do Controlleru. Pro přístup do databáze třída **Controller** posílá dotaz na určitý Model, který získá požadovaná data z databáze a následně navrací výsledek zpět. Získaná data nakonec projdou zpracováním v třídě **Controller** a výsledné hodnoty přeposílá dál do View komponenty, která již prezentuje koncový výstup v jednotlivých šablonách webu.

Laravel pro instalaci a aktualizaci všech svých komponent využívá nástroj **Composer** [19]. Tento volně dostupný nástroj zajišťuje správu používaných komponent pro PHP zapisováním příkazů do terminálu. Po instalaci Laravelu je vytvořena výchozí adresářová struktura. Tuto strukturu není třeba dodržet a je možné si ji předělat podle vlastního uvážení. S pomalu nulovými zkušenostmi s tímto frameworkem jsem však adresářovou strukturu ponechal a neměnil ani v průběhu programování. Zde jsou nyní pojmenovány jednotlivé složky s obecným popisem obsahovaných komponent v nich:

[3]Model, View, Controller

- **app** - Jádro celé aplikace obsahující logickou část webu. Všechny třídy webové aplikace jsou obsaženy v této složce a zpracovávají požadavky posílané na server.
- **bootstrap** - Obsahuje soubory načítané pro spuštění frameworku, nastavení cest k třídám a mnou nevyužívaného front-end frameworku bootstrap. Bootstrap framework je automaticky zahrnut při stažení Laravelu, který ho doporučuje k použití, avšak nevyžaduje jej.
- **config** - Uvnitř této složky jsou konfigurační soubory pro různá nastavení serveru. Tato práce vyžadovala úpravy pouze v souborech **app.php**, **auth.php**, **database.php** a **mail.php**. Ostatní kofigurační soubory zůstaly nezměněny.
- **database** - Adresář obsahující migrační a seed soubory Laravelu. V těchto souborech (třídách) jsou definovány funkce, spouštěné pomocí příkazů zadávaných přes příkazovou řádku. Migrační soubory obsahují funkce které vytváří, aktualizují nebo mažou tabulky v připojené databázi. Seed soubory obsahují funkce pro vytvoření testovacích dat pro tabulky připojené databáze. Tato možnost však nebyla využívána, aby se předešlo komplikacím, protože je optimalizována hlavně pro virtuální systém Laravel Homestead, který nebyl v této práci použit.
- **public** - Složka obsahující **index.php** soubor, který představuje vstupní bod pro všechny požadavky přicházející od klienta. Obsahuje složky **css**, **js** a **images**, v nichž existují důležité soubory spojené se zobrazením designu a funkcí na klientské straně webu.
- **resources** - HTML soubory webové aplikace, které zobrazují výstup na klientskou část. Součástí jsou také soubory s českou lokalizací automaticky generovaných chybových hlášek při validaci vstupu.
- **routes** - Soubory definující práci serveru se vstupem od klienta. Soubor **web.php** zpracovává url cestu na webové stránce. Ke každé definované url cestě je odkazována určitá funkce třídy v **app** adresáři, které jsou přeposlány vstupní parametry. Ostatní soubory byly ponechány ve výchozím nastavení, protože nebyly využity v této webové aplikaci.
- **storage** - Adresář uchovávající vygenerované šablony webu, ukládané soubory z webové aplikace či soubory generované frameworkem. Obsahuje také logy všeho, co se na webu zpracovává, které je možné si prohlédnout pro nalezení případné chyby webu.

- **tests** - Složka obsahující automatizované testy.
- **vendor** - Obsah této složky se skládá ze všech využívaných komponentů spravovaných nástrojem **Composer**.

Mimo jiné se v kořenové složce nachází důležitý soubor **.env**, ve kterém se ukládají přihlašovací údaje a nastavení pro spojení s MySQL databází a nastavení protokolu smtp pro odesílání emailových zpráv.

7.3 Klientská část webu

Pro prezentaci výstupu používá Laravel vlastní šablonovací systém nazývaný Blade [20]. Všechny šablony vytvořené tímto systémem jsou ukládány s koncovkou **.blade.php**. Tento systém umožňuje využití rozšířených funkcí, jako je například rozdělení na sekce, dědění, apod. jednotlivých šablon pro usnadnění a lepší přehlednost v kódu. Zápis rozšířených funkcí **include** a **yield** lze vidět v ukázce kódu souboru **master.blade.php** (viz Kód 7.3), ze kterého všechny ostatní šablony, kromě záhlaví **header.blade.php**, dědí. Zmiňovaná šablona záhlaví je vložená na začátku těla souboru **master.blade.php** funkcí include. Vložený parametr představuje název šablony bez koncovky spolu s názvy adresářů, ve kterých je šablona obsažena. Funkce yield v souboru pak označuje sekci *content*. Děděné šablony následně používají funkci **section('content')** pro vkládání HTML kódu do sekce definované v děděné třídě. Jedna z nejčastěji používaných funkcí je však vypisování výstupu proměnných pomocí dvou složených závorek, které nahrazují php funkci **echo**.

```
<body>
    ...
    @include('layout.header')
    <main class="mdl-layout__content␣mdl-color--grey-100">
        @yield('content')
    </main>
    ...
</body>
```

Kód 7.3: Hlavní obsah všech šablon

Tato kapitola se bude věnovat ukázkám práce s JavaScript a JQuery jazykem a frameworkem Material Design Lite (dále jen MDL). Framework využívá pro uspořádání komponent kombinace tříd definovaných v příslušných CSS[4] souborech pro klasické prvky webu. Tyto soubory musí být odkazovány v html souboru, v tomto případě v hlavičce všech existujících html souborů.

[4]Cascading Style Sheets

MDL rozkládá stránku na *grid* neboli mřížky. Mřížka je dynamicky přizpůsobující se plocha pro různá rozlišení. Plocha na počítači se rozkládá na 12 panelů, 8 panelů pro tablety a 4 panely pro mobilní telefony, kde panely mají v každém rozlišení jinak předdefinované odsazení. Takto lze přidělovat pro HTML značky (div, table, p, input, atd.) pouze určitou část plochy v dané mřížce.

Hlavní zájem frameworku směřuje samozřejmě na to, jak si poradí s vytvářením formulářů. Všechny formuláře na webové stránce jsou vytvářeny stejným způsobem. Zde bude tedy ukázán pouze příklad stránky s registrací nového uživatele. Všechny formuláře jsou v mřížce rozděleny na štítek zabírající 4 panely plochy a textové pole zabírající 8 panelů plochy. Textová pole jsou definována CSS třídou *mdl-textfield mdl-js-textfield*. Jednotlivá pole jsou pak validována JavaScriptem, kdy se v případě chybného zadání zobrazí chybová hláška pod textovým polem. V MDL se takové pole definuje CSS třídou *mdl-textfield__error*. Jak jsou zapsány vstupní pole, znázorňuje ukázka kódu vstupního pole **email** viz Kód 7.4.

```
<label class="mdl-cell␣mdl-cell--4-col␣textLabel">Email<
   ↪ span style="color:␣red">*</span>:</label>
<div class="mdl-textfield␣mdl-js-textfield␣mdl-cell--8-col
   ↪ ␣mdl-cell">
    <input class="mdl-textfield__input" type="email" id="
       ↪ email" name="email" value="{{␣old('email')␣}}">
    <label class="mdl-textfield__label" for="email">jan.
       ↪ novy@email.cz</label>
    <span class="mdl-textfield__error">Neni validni email
       ↪ </span>
    <span class="error">{{$errors->first('email')}}</span>
</div>
```

Kód 7.4: Ukázka jedné kolonky formuláře

Validace vstupu u klienta však není zcela bezpečná a každý vstup se tak kontroluje i na serveru, kde se v případě chybného vstupu zapisují chyby do pole s názvem **$errors**. Při chybné validaci je uživatel přesměrován zpět na stránku s formulářem spolu s chybovými hláškami a globální proměnnou **old**, ve které se nachází zapamatovaná vstupní data, aby je uživatel nemusel zadávat do formuláře znovu.

Webová aplikace obsahuje tabulky v šablonách **members.blade.php** a **districts.blade.php** pro snadné zobrazení a správu uživatelů a oblastí administrátorem. Jelikož jsou šablony skoro identické, budou ukázky zaměřeny pouze na šablonu uživatelů **members.blade.php**. Vytvoření ta-

bulky v MDL je zapisováno CSS třídou *mdl-data-table mdl-js-data-table*. Tabulka je pak naplněna uživateli, kteří jsou posíláni serverem do této šablony v proměnné **$users**. Sloupce jsou formátovány MDL třídou *mdl-data-table__cell–non-numeric* podle toho, zda je sloupec plněn numerickými daty nebo textem. Níže je zobrazena ukázka kódu 7.5 vytvoření těla tabulky. Cyklem se vytvoří řádek v tabulce pro každého uživatele, který obsahuje jediný číselný sloupec roku narození. Pokud uživatel není administrátor je po kliknutí na záznam v tabulce pomocí JS metody **onclick** přesměrován na profil daného uživatele. V opačném případě se akce na uživatele vybírá podle přepínače.

```
<tbody>
@foreach($users as $user)
    <tr id="{{$user->id}}" class="clickable" about="{{
       ↪ $user->name}}" @if(!auth()->user()->isAdmin())
       ↪ onclick="location.href = '/profile/{{$user->id}}'
       ↪ " @endif>
        <td class="mdl-data-table__cell--non-numeric">{{
           ↪ $user->name}}</td>
        ...
        <td>@if(isset($user->year_of_birth)){{$user->
           ↪ year_of_birth}} @else neuvedeno @endif</td>
    </tr>
@endforeach
</tbody>
```

Kód 7.5: Vytvoření těla tabulky uživatelů

Pro řádky s uživateli byla vytvořena vlastní CSS třída *clickable*, která po najetí myší zvýrazňuje danou řádku změnou barvy textu a změnou typu ukazatele. Pro provedení akce administrátora po kliknutí na uživatele v tabulce je však zapotřebí JavaScript a JQuery, který zpracovává vybraný prvek přepínače. MDL přepínač nenabízí jednoduché vyjmutí právě zaškrtnutého prvku, a proto je nutné procházet všechny prvky přepínače, dokud nenarazí na zaškrtnutý prvek akce vybrané administrátorem (viz Kód 7.6). Při nalezení zaškrtnutého prvku je podle hodnoty *value* atributu provedena vybraná akce.

Při výběru akce **Smazat** uživatele se používá MDL komponenta **Dialog**. Dialog je vyskakující okno používané na upozornění nebo jednoduché formuláře. Sami vývojáři však upozorňují, že tato komponenta má v současné době velmi malou podporu prohlížečů a doporučují použití pluginu **dialog polyfill** [21], který v MDL není obsažen, nebo vytvoření vlastního

konceptu vyskakovacího okénka. V této práci jsem tedy využil plugin podporující všechny prohlížeče.

```
var id = $(this).attr('id');
jQuery("input[name='options']").each(function () {
    if (this.checked) {
        switch (parseInt(this.value)) {
            case 1:
                location.href = "/profile/" + id;
                break;
            case 2:
            ...
        }
    }
});
```

Kód 7.6: JQuery procházení zaškrtnutého prvku přepínače

Většina prováděných akcí a odesílaných dat ve webové aplikaci je provázena takzvanými *Flash* zprávami. Při poslání dat na server se data zpracují a přesměrují uživatele na vybranou stránku spolu s flash zprávou a případnými daty. Tyto zprávy se ukládají do globální proměnné **session** pod názvem *success*, *info* nebo *error*.

7.4 Serverová část webu

Všechny HTTP[5] požadavky zasílané uživatelem jsou přesměrovány přes soubor **routes.php** na příslušnou funkci v takzvaných **Controller** třídách. Požadavek HTTP protokolu zpracovávaný souborem **routes.php** v této práci nabývá 4 metod: GET, POST, PATCH a DELETE. Při snaze přístupu uživatele na stránku se tedy podle zadané cesty v URL adrese, která obecně následuje za doménou serveru, a poslané HTTP metody určí, ve které třídě Controlleru se má provést určitá funkce. Pro lepší představu je zde přiložen Kód 7.7, který představuje zápis jednotlivých přesměrování. V kódu lze vidět cestu **'/term/delete/id'** u metody DELETE, která obsahuje prvek id ve složených závorkách. Složka ve složených závorkách je parametr, který je přeposílán do funkce v **TermController**. Stručně zde budou popsány jednotlivé využité metody [22]. Nejzákladnější požadavek je vysílán s metodou GET. Tato metoda je vyslána pokaždé, když se uživatel snaží dostat k nějaké stránce na webu. Metoda POST slouží při posílání dat z formuláře na webu. Laravel pro každý formulář s POST, PATCH a DELETE metodou

[5]Hypertext tranfer protocol

```
Route::get('/members', 'PageController@showMembers');
Route::post('/login', 'SessionController@login');
Route::patch('/profile/settings', 'UserController@editUser
    ↪ ');
Route::delete('/term/delete/{id}', '
    ↪ TermController@deleteTerm');
```

Kód 7.7: Zápis přesměrování HTTP požadavků

musí obsahovat CSRF[6] token, který slouží jako ochrana proti CSRF útoku. Metoda POST se určuje u každého formuláře, u kterého chceme, aby data posílaná na server nebyla zobrazována v url adrese na výstupu. HTML standart povoluje nastavení metody formuláře pouze na GET a POST, a proto Laravel pro ostatní metody využívá vložení skrytého vstupu ve formuláři (viz Kód 7.8). Tento vstup je pak jednoduše zpracován frameworkem, který již nahrazuje metodu požadavku.

```
<input type="hidden" name="_method" value="PATCH">
```

Kód 7.8: Formulář se skrytým vstupem pro HTTP metodu PATCH

Metoda PATCH se využívá pro operace editace či aktualizace jednotlivých záznamů v databázi a DELETE metoda je zase využívána na mazání záznamů z databáze.

7.4.1 Modely

Laravel pro práci s databází využívá vlastní Eloquent ORM[7], který zjednodušuje práci s tabulkami pomocí odpovídajících tříd modelu v kořenu složky **"app/"**. Každý model je v aplikaci pojmenován názvem tabulky v jednotném čísle. Pokud tedy existuje v databázi tabulka **users**, model se bude nazývat **User**. Pojmenované tabulky databáze v množném čísle jsou z hlediska databázového zápisu nekorektní, avšak z důvodu přístupu Laravel frameworku k tabulkám se ponechá takto výchozí nastavení vyhovující frameworku.

Všechny modely Laravelu ve výchozím nastavení dědí z rodičovské třídy **"*/Eloquent/Model.php"**, která zajišťuje propojení s tabulkou v databázi pomocí předdefinovaných metod. Mimo jiné předpokládá, že primární klíč bude v tabulce nazýván *id* a v tabulce existují kolonky **created_at** a **updated_at** pro časové značky vzniku a úpravy daného záznamu. Kolonky časových značek však v této databázi nejsou využívány, tak bylo nutné tuto

[6] Cross-site request forgery
[7] Object-relational mapping

funkci vypnout pro všechny modely. Z tohoto důvodu byla vytvořena třída **"app/Model.php"**, která je potomkem Eloquent modelu, avšak rodičem pro všechny modely tabulek databáze této bakalářské práce. Ve vytvořené třídě je nastavena proměnná **$timestamps** na hodnotu **false**, čímž je vypnuto automatické vytváření atributů výše zmíněných časových značek.

Pro práci s modely je výhodné mít možnost získávat data z propojených tabulek. Laravel pro vztahy mezi tabulkami využívá metody **belongsTo**, **hasMany**, **belongsToMany** viz příklad Kód 7.9, které v případě nadefinování v modelu fungují stejně jako **JOIN** metody v jazyce SQL. Propojení primárního klíče s cizím klíčem v druhé tabulce se provádí automaticky, dokud jsou cizí klíče zapisovány názvem tabulky s koncovkou **_id**. V opačném případě se musí cizí klíč manuálně nadefinovat.

```
public function user(){
    return $this->belongsTo(User::class);
}
public function terms(){
        return $this->hasMany(Term::class);
}
```

Kód 7.9: Příklad metody **user** vztahu N:1 a **terms** vztahu 1:N

7.4.2 Registrace a přihlášení uživatele

Po kliknutí na odkaz registrace či přihlášení, který se nachází v záhlaví webové aplikace (viz Obr. 7.1) je poslán dotaz na specifický **MainController.php**. Tento soubor zpracovává hlavně požadavky, které mohou využít všichni uživatelé webové aplikace bez omezení. Požadované dokumenty jsou tedy vraceny metodami této třídy. Při návratu dokumentu je možné připojit funkcí **compact** proměnné definované v metodě, které je pak následně možné využít při výpisu dat v šabloně.

Obrázek 7.1: Záhlaví webové aplikace

Po odeslání formuláře na stránce registrace se nejdříve provede validace dat. Validace se provádí kontrolou získaných dat z formuláře v třídě **ValidatesRequest**. V případě úspěšné kontroly se bude vykonávat zbylý kód metody, avšak pokud alespoň jedna validace neprojde, je vyhozena výjimka a uživatel je přesměrován zpět na stránku s formulářem a chybo-

vými hláškami. Při registraci se samozřejmě vytváří nový záznam do tabulky **Users** databáze. Zavoláním metody **create** nad vytvořeným modelem uživatele se vytvoří nový záznam do tabulky s příslušnými atributy převzatými ze vstupních dat. Pro šifrování hesla se využívá implementovaná šifrovací funkce Bcrypt. Vytvořený uživatel má v databázi výchozí hodnotu atributu *auth_level* na 0. To značí, že registrovaný uživatel ještě nebyl potvrzen od administrátora, který tak musí učinit. Pokud by se tak nově registrovaný uživatel chtěl přihlásit, zobrazí se mu informativní flash zpráva o vyčkání na potvrzení od administrátora. V ukázce kódu 7.10 je zobrazena jak validace, tak vytvoření nového záznamu do tabulky. V poslední řadě je navraceno přesměrování na domovskou adresu s flash zprávou *success*, která informuje uživatele o úspěšné registraci.

```
public function registration(){
    $this->validate(request(), [
        'name' => 'required|min:3|max:30',
        'email' => 'required|unique:users|email',
        'password' => 'required|min:6|max:30|confirmed',
        'password_confirmation' => 'required|min:6|max:30'
    ]);

    User::create([
        'name' => request('name'),
        'email' => request('email'),
        'password' => Hash::make(request('password'))
    ]);

    return redirect()->home()->with('success', '<strong>
       ↪ Registrace␣uspesna!</strong><br>Nyni␣vyckejte␣na␣
       ↪ potvrzeni␣administratorem.');
}
```

Kód 7.10: Registrace nového uživatele

Přihlašování uživatele probíhá nejprve validací vstupu. Po validaci je získán uživatel z databáze dotazem **find** na eloquent model, který hledá zadaný email použitý při přihlášení. V případě neexistence emailu v databázi je uživatel přesměrován zpět s chybovou hláškou. Pokud však záznam v databázi existuje, je nejprve testována podmínkou úroveň autorizace, která musí být nastavena na hodnotu 1. Následně je volána metoda porovnávající zašifrované heslo s heslem zadaným viz Kód 7.11. Do metody přihlašování je ještě vkládána proměnná **$remember**, která představuje boolean hodnotu nabývající hodnot 1 nebo 0. Pokud tedy uživatel zaškrtl tlačítko ve formu-

láři přihlášení, pro zapamatování údajů bude uživatel přihlášen do doby, než se sám odhlásí. Tato funkce funguje na uložení vygenerovaného tokenu uživatele v cookies prohlížeče. Po úspěšném přihlášení jsou informace o uživateli uloženy v relaci webu. Akce registrace, přihlášení a odhlášení uživatele se provádí ve třídě **SessionController.php**.

```
if (auth()->attempt(request(['email', 'password']),
   ↪ $remember)) {
    return redirect()->home()->with('success', '<strong>
       ↪ Uspesne prihlasen!</strong>');
}
```

Kód 7.11: Pokus o přihlášení uživatele

7.4.3 Vyhledávání a zobrazování hesel

Na hlavní stránce webové aplikace se nachází pole pro vyhledávání hesel. Při zadání řetězce a vybrání filtru se pošle dotaz do třídy **MainController**. Všechna schválená hesla obsahující vybraný řetězec jsou pak vybrána z databáze a uloženy do proměnné **$terms**. Přepínačem se samozřejmě určuje, podle kterého filtru se vyhledávání řídí. Ukázka kódu 7.12 zobrazuje hesla se zadaným řetězcem podle významu. V ukázce je také ukázáno, že proměnné s daty jsou do šablon přeposílány pomocí metody **compact**, která figuruje jako druhý nepovinný parametr **view** metody.

```
public function searchTerms(){
...
$terms = Term::where('meaning', 'LIKE', '%' . request('
   ↪ searchTerm') . '%')->where('accepted', '1')->get();
...
$alphabet = $this->getAlphabet();
return view('main.catalog', compact('alphabet', 'terms'));
}
```

Kód 7.12: Vybrání hesel obsahující řetězec podle významu

Prezentace těchto hesel se pak provádí v šabloně **catalog.blade.php**. Šablona katalogu obsahuje v záhlaví abecedně seřazené záložky písmen, na která začínají hesla obsažena v databázi. Pro zjištění abecedy byla vytvořena vlastní metoda **getAlphabet** procházející všechna hesla v databázi, která následně ukládá unikátní počáteční písmena do pole. Bohužel nebylo možné vybírat pouze unikátní záznamy hesel začínající na dané písmeno, protože SQL dotazy mají problém s českou diakritikou. Získaná data jsou následně

prezentována postupně po třech heslech na řádku stránky viz Obr. 2. Na zobrazená hesla lze kliknout, čímž se uživatel dostane na detail daného hesla. Při kliknutí na detail hesla se aktualizuje atribut *last_find* značící časovou značku posledního vyhledání. Tato časová značka je následně využita pro vybrání posledních pěti vyhledaných hesel, které jsou zobrazeny na hlavní stránce.

7.4.4 Nahrávání nových hesel

Nové heslo je se přidává v šabloně **newTerm.blade.php**, kde je již zobrazen obsáhlý formulář pro zadání všech detailů hesla. Pro vyplňování oblasti výskytu hesla je využit JQuery plugin **Select2**, který spojuje vyhledávací okénko spolu s polem dat. Tento plugin pak nabízí zobrazovat vyhledávané obce v reálném čase. Pro zobrazení hledané obce je však nutné zadat alespoň 2 znaky, obsažené v celém fulltextu záznamu, které již značně omezují seznam vypisovaných dat a vyhledávání je tak plynulé. Po odeslání do třídy **TermController.php** dochází k validaci nutných polí na serveru. Poté probíhá kontrola vkládaného audio souboru, který musí dodržet podmínky maximální velikost 10MB a přípona souboru je povolena pouze *.mp3* a *.wav*. Vytváření nového hesla pak probíhá nejprve vytvořením záznamu slovního druhu. Pokud je slovní druh podstatné jméno nebo sloveso, zapisuje se id slovního druhu do vytvořeného záznamu v tabulce **nouns** nebo **verbs**. Poté probíhá uložení audio záznamu na lokální disk do adresáře **storage/app/** pomocí Laravel třídy **Storage**. Tato třída obsahuje metody pro jednoduchou správu souborů nahrávanými na server. V tomto případě je použita metoda **putFile**, která obsahuje jako první parametr název podsložky, do které se audio uloží, a druhý parametr je Symfony třída **File**, která obsahuje parametr cesty k nahrávanému souboru (viz Kód 7.13). Do nově vzniklé proměnné **$filePath** se ukládá cesta k souboru na lokálním disku, která je předávána do atributu *audio_path* hesla.

```
$file = request('fileUp');
isset($file) ? $filePath = Storage::putFile('audio', new
   ↪ File($file)) : $filePath=null;
```

Kód 7.13: Ukládání audio souboru na lokální disk

Se zbylými daty se vytvoří nový záznam hesla. Vytvořený záznam obsahuje atribut *accepted*, který je při vytvoření hesla nastaven na hodnotu 0. Takto označené heslo musí projít kontrolou správcem oblasti nebo administrátorem jako nové heslo čekající na schválení.

7.4.5 Nahrávání nových oblastí

Šablona **districts.blade.php** pro každého uživatele nabízí tlačítko pro přidání nové oblasti nacházející se v levém horním rohu hlavního obsahu stránky. Při odeslání dat proběhne validace zadaných polí obec, okres a kraj, jelikož žádné z nich nesmí být prázdné. Cyklus následně prochází všechny obce, které porovnávají atributy *municipality*, *district*, *region* z tabulky **districts** s vloženými daty. V případě jmenné shody všech atributů u záznamu s vloženými daty je uživatel přesměrován zpět s chybovou hláškou. V opačném případě je nová oblast přidána do databáze a je možné ji vyhledat při každé práci s obcemi ve webové aplikaci.

7.4.6 Nastavení uživatele

Pro samotné správce elektronického slovníku je důležité vědět o uživatelích doplňující informace. Proto je v šabloně **settings.blade.php** formulář pro změnu osobních údajů. Každé odeslání formuláře je validováno na serveru ve třídě **UserController**. Tato validace využívá integrovaného rozšíření **Carbon** pro validaci roku narození viz Kód 7.14. Rozšíření usnadňuje práci s daty a časem v PHP. Validace vstupu pro původní a současné bydliště je identická, kdy v případě zadání bydliště se kontroluje existence záznamu obce v tabulce **districts**. Při změně údajů je podmínkou kontrolováno, zdali je pole vyplněné nebo jestli se pole nerovná stávajícímu údaji v databázi. V případě že je jedna z těchto podmínek pravdivá, se daný atribut v databázi neaktualizuje a zůstává stejný.

```
'year' => 'nullable|numeric|digits:4|min:' . (Carbon::now
    ↪ ()->year-150) . '|max:' . Carbon::now()->year
```

Kód 7.14: Ukázka kódu validace pro rok narození uživatele

Administrátor webu má přístup ke změně nastavení těchto údajů každého uživatele. Tato akce je samozřejmě kontrolována podmínkou, kdy při změně nastavení konkrétního uživatele je vyhodnoceno zda je přihlášený uživatel administrátorem. V opačném případě se vždy mění nastavení právě přihlášeného uživatele. V nastavení je možné si pod zobrazovaným profilem všimnout tlačítka pro změnu hesla a emailu. Tato akce je již přístupná pouze přihlášenému uživateli pro změnu přihlašovacích údajů. Šablona **setnsitiveSettings.blade.php** obsahuje podobný formulář, jako při registraci nového uživatele. Odeslání dat do třídy **UserController** je kontrolováno stejnou validací jak u registrace uživatele (viz Kapitola 7.4.2) a změna údajů je vždy nejdříve kontrolována potvrzením starého hesla.

7.4.7 Správa uživatelů

Při každé nové registraci je uživateli zobrazena zpráva, která oznamuje, že uživatel musí počkat na potvrzení registrace administrátorem. Při schvalování registrace má administrátor možnosti schválit, nebo smazat uživatele. Akce probíhá kliknutím na záznam v tabulce spolu s vybraným zaškrtnutým filtrem přepínače v levém poli. Všechny funkce administrátora se provádí ve třídě **AdministrationController**. Při schvalování registrace uživatele je uživateli nastavena hodnota atributu **auth_level** na 1 a poslán email s úspěšnou registrací viz Kód 7.15. Metoda **findOrFail** Eloquent ORM modelu vyhledává v databázi záznam uživatele podle id. V případě nenalezení tohoto uživatele je vyhozena HTTP výjimka 404 a přesměruje uživatele na šablonu **404.blade.php**. Posílaný email je vytvořená šablona **welcome.blade.php**, která obsahuje uvítání uživatele na webové stránce. Email je posílán přes metodu **send** v Laravel třídě **Mail**.

```
public function authUser($id)
{
    $user = User::findOrFail($id);
    $user->auth_level = 1;
    $user->save();
    Mail::to($user)->send(new Welcome($user));
    return redirect()->back()->with('success', 'Uzivateli␣
       ↪ "' . $user->name . '"␣byla␣schvalena␣registrace␣a
       ↪ ~poslan␣email.');
}
```

Kód 7.15: Schválení registrace

Společně s těmito akcemi pro nově registrované uživatele se administrátorovi v oddělené tabulce nabízí možnosti editovat základní údaje, přidělit správu oblasti nebo smazat již schválené uživatele v systému. Smazání uživatele ze systému je kontrolováno potvrzením v okně dialogu, kde administrátor vidí jméno uživatele a akci musí potvrdit zadáním svého hesla. Změna základního nastavení funguje stejně jako ve výše zmiňované sekci 7.4.6, kde probíhá kontrola při změně údajů, zda k metodě přistupuje uživatel s právy administrátora. Změna údajů uživatele se pak zobrazí v seznamu uživatelů ihned. Důležitý prvkem je však přiřazování správy oblastí pro uživatele.

Administrátor je po akci přidání nové oblasti uživateli odkazován na šablonu **distSettings.blade.php**, která obsahuje zaškrtávací seznam 14 krajů České republiky. Zaškrtané kraje jsou odeslány v textovém řetězci, ve kterém jsou odděleny čárkou. Na serveru ve třídě **AdministrationController** jsou následně rozděleny metodou **explode** do pole **$selected**. Upravovaný

uživatel však může již některé oblasti spravovat, a proto je nutné vyfiltrovat oblasti, které již spravuje a zapamatovat si oblasti, které mu budou odebrány (viz ukázka Kód 7.16). Metoda nejprve cyklem vybere z databáze všechny spravované kraje uživatelem a uloží je do pole **$checked**. Jednotlivá pole jsou mezi sebou následně porovnávána a v případě výskytu kraje, který byl zaškrtnut, v poli již spravovaných krajů se z obou polí tyto prvky odeberou. Pole se musí následně znovu seřadit metodou **array_values**, jelikož metoda **unset** pouze odebere záznam z pole, který by v následném cyklu procházení mohl vyhodit chybu neexistujícího indexu. Tato metoda ve výsledku zajistí, že všechny zaškrtnuté kraje, které uživatel nespravoval, zůstanou v poli **$selected** a kraje, které mají být uživateli odebrány, se nachází v poli **$checked**. Cyklem se pak prochází všechny složky těchto polí a vytváří se k nim záznamy v tabulce **district_administration**, kde

```
$selected = explode(',', request('region_name'));
$user = User::findOrFail(request('user_id'));
$checked = array();

foreach ($user->districtAdmin() as $district) {
    $checked[] = $district->region;
}

if (count($checked) > 0) {
    for ($i = 0; $i < count($selected); $i++) {
        for ($j = 0; $j < count($checked); $j++) {
            if ($checked[$j] === $selected[$i]) {
                unset($selected[$i]);
                unset($checked[$j]);
                $selected = array_values($selected);
                $checked = array_values($checked);
                $i--;
                break;
            } elseif ($selected[$i] === '') {
                unset($selected[$i]);
                $i--;
                break;
            }
        }
    }
}
```

Kód 7.16: Rozdělení přidělovaných a odebíraných krajů do polí

atribut *user_id* nabývá hodnoty id upravovaného uživatele a *district_id* nabývá hodnoty první nalezené obce ve vybraném kraji.

Vybírat pro správu hesel celé kraje je kompromisem řešení, kdy správa hesel nad jednotlivými obcemi či okresy, by nebyla ve stávajícím návrhu webové aplikace dostatečně využívána, jelikož je zaměření oblasti příliš konkrétní. Správa jednotlivých oblastí je následně prezentována v seznamu uživatelů a na profilu samotného uživatele.

7.4.8 Správa hesel

Jako správa uživatelů je na stejném principu vytvořena správa jednotlivých hesel. Při editaci hesla je uživateli zobrazena šablona **editTerm.blade.php**, která je vizuálně stejná jako šablona pro přidání nového hesla. Jediným rozdílem jsou vyplněné kolonky nově přidávaného hesla od autora, které jsou do šablony poslány v proměnné **$term**. Proměnná je naplněna daty, které byly uloženy v databázi při přidání nového hesla. Uživatel s autorizací následně může doplnit chybějící nebo opravit špatné informace o heslu. Podmínky validované při provádění metody editace jsou identické jako při přidávání hesla. Jediná změna v kódu se liší odebráním záznamu slovního druhu a doplňujících informací o podstatném jménu či slovesu, v případě změny tohoto údaje. Pro každou akci nad heslem, tedy editace, mazání a schválení je kontrolováno, zda je daný uživatel oprávněn k provedení akce metodou **isTermViable()** v modelu **User** (viz Kód 7.17). Takto je předcházeno neoprávněným přístupům, které by chtěly poškodit záznamy v databázi. Schválením hesla se nastavuje v záznamu hodnota atributy *accepted* na 1 a aktualizuje se v databázi. Schválená hesla je pak ihned možné najít v rejstříku hesel pod začínajícím heslem nebo je možné ho vyhledat vyhledávačem na hlavní stránce.

```
$term = Term::findOrFail($id);
if(!auth()->user()->isTermViable($term->district->region))
   ↪ {
    return redirect()->to('/term/waiting')->with('info', '
       ↪ K␣takove␣operaci␣nemate␣pristup');
}
```

Kód 7.17: Kontrola oprávnění uživatele pro správu hesla

8 Testování

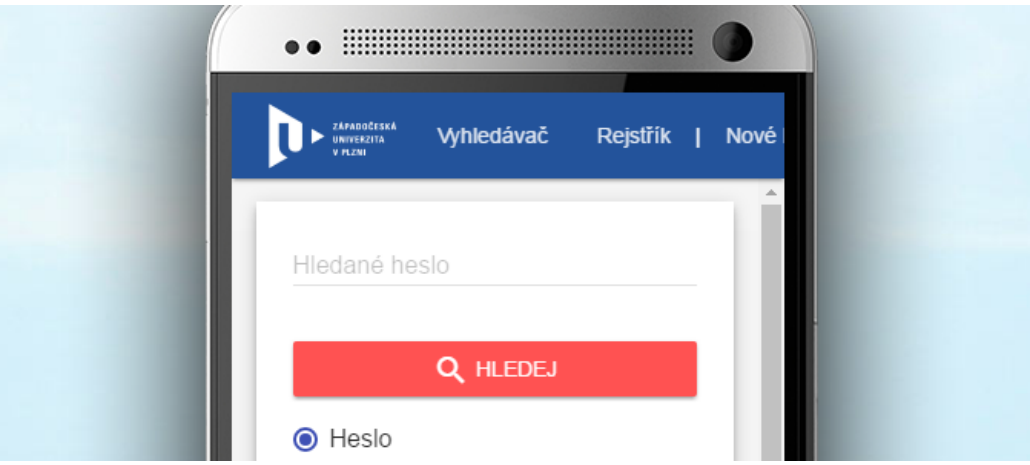
Při vývoji aplikace bylo vše spouštěné na lokálním Apache web serveru verze 2.4.25. Ke zprovoznění serveru na operačním systému Windows byl použit nástroj XAMPP. Nástroj mimo jiné obsahuje i phpMyAdmin software verze 4.6.5.2 pro správu MySQL databáze, která byla využívaná pro ukládání dat. Pro podporu nejnovějších funkcí PHP 7.1.1 byl stažen v současné době nejnovější instalátor XAMPP aplikace (v. 7.1.1), podporující tuto verzi PHP.

Pro otestování správného zápisu HTML elementů byla využita volně dostupná internetová stránka W3C validator [8]. Testování probíhá přístupem na jednotlivé stránky této webové aplikace, která následně zkontroluje zdrojové kódy šablon. V případě problematických částí kódu vypíše chybovou hlášku nebo varování společně s možným řešením. Chyba také přesně zobrazuje číslo řádky a vypisuje chybný kód viz Obr. 8.1. Chyb vyskytujících

Obrázek 8.1: Chybové hlášky stránky https://validator.w3.org/ [8]

se v prezentaci výstupu této webové aplikace bylo kolem 15 a pro všechny otestované šablony byly opraveny.

Stránka MobileTest [9] nabízí napodobení zobrazení webové aplikace na různých rozlišeních. Na stránce je k dispozici náhled webové aplikace na zařízeních Apple Iphone 5, HTC ONE, Nokia Lumia 920, Samsung Galaxy Y, Google Nexus 7 a Apple iPad Mini. Pro všechna tato zařízení byl při testování zpozorován problém, kdy záhlaví webové aplikace nebylo dynamické a pravá část lišty vždy zmizela z obrazovky (viz Obr. 8.2). Tímto byly nepřístupné všechny funkce spojené s uživatelem a správy hesel a uživatelů. Pro optimalizaci webové aplikace i v mobilních zařízeních muselo být upraveno záhlaví webu. Nyní je tedy naprogramováno, že v případě přechodu rozlišení do menšího, než je standardní rozlišení 786px, jsou odkazy v záhlaví přemístěny do bočního menu. Pokračujícím testováním na emulátoru byly zjištěny



Obrázek 8.2: Problém zobrazení v mobilních zařízeních [9]

problémy zalamování textu u přepínačů v panelech akcí na uživatele či hesla. Jelikož se tato chyba vykytuje pouze u administrátora a nemá přímý vliv na chod funkčnosti aplikace, byla prozatím vynechána. V budoucnosti je však tento problém jedním z plánovaných bodů na vylepšení webu.

Implementovaná serverová část webové aplikace je otestována automatickými testy. Jak již bylo zmíněno v sekci 4.6 pro automatické testování bylo užito integrované rozšíření **Laravel Dusk**. Při vytváření projektu Laravel automaticky vygeneruje složku **tests/Browser** v kořenu aplikace, která vytvořené testy obsahuje. Příkazem **php artisan dusk** zadaným do konzole se spustí testy sepsané ve třídách obsažených ve výše zmiňovaném adresáři. Při spuštění se zapíná obsažený driver **ChromeDriver** pro automatizované testování stránek. Jádro celé aplikace stojí na bezproblémové funkčnosti metod v Controller třídách, jejichž metody zpracovávají všechny požadavky uživatele. Pro každou metodu je tedy vytvořen odpovídající test v testovací třídě. Browser testy, jak jsou Laravelem nazývány, jsou psány stylem přístupu uživatele na web, který provádí jednotlivé akce. Výstup výsledku testů se zobrazuje přímo do konzole, kde v případě chyb nebo varování je popsáno, v jaké testovací třídě byla chyba nalezena, která výjimka chybu způsobila a popis definující problém (viz ukázka 8.1).

```
There was 1 error:
1) Tests\Browser\UserTest::testRegisterAndLogin
InvalidArgumentException: Unable to locate button [
    ↪ Prihlasit].
```

Kód 8.1: Chybová hláška nenalezení tlačítka na stránce

Testování využívá falešná data, která představují reálné záznamy uklá-

dané do databáze. Tato data se vytváří pomocí **factory()** metody, která vygeneruje náhodná falešná validní data pro model. Hodnoty factory metody se definují ve třídě **ModelFactory.php**, která s použitím předdefinovaných metod třídy **Faker/Generator** vytváří náhodná data (viz ukázka Kódu 8.2) pro vygenerování uživatele.

```
$factory->define(App\User::class, function (Faker\
   ↪ Generator $faker) {
    return [
        'name' => $faker->firstName.'␣'.$faker->lastName,
        'email' => $faker->unique()->safeEmail,
        'password' => $password ?: $password = bcrypt('
           ↪ secret'),
        'remember_token' => str_random(10),
        'auth_level' => 0
    ];
});
```

Kód 8.2: Falešný záznam uživatele pro testování

Pro testování registrace uživatele (viz Kód 8.3), je možné vidět otestování registračního formuláře. Testy jako tento jsou tvořeny hlavně na kontrolu vstupu nevalidních dat při odesílání formulářů a správné odpovědi serveru. Psaní testů je však práce převážně rutinní a často se opakuje, proto se většina kódu v testech velmi podobá. V ukázce kódu lze vidět průběh vykonávaných akcí v pořadí: navštívení domovské stránky, kliknutí na odkaz *Registrovat*, vyčkání na text na stránce, zadávání nevalidních údajů, vyčkání na chybu vyhozenou JavaScriptem, zadávání jiných údajů, odeslání dat na server a čekání na chybovou hlášku ze serveru.

```
$this->browse(function (Browser $browser) {
    $browser->visit('/')
        ->clickLink('Registrovat')
        ->waitForText('Registrace␣noveho␣uzivatele')
        ->type('name', 'Petr␣Lukasik')
        ->type('email', 'plukas@cz...91872')
        ->waitForText('Neni␣validni␣email')
        ->type('email', 'admin@admin.cz')
        ->press('create')
        ->waitForText('Tento␣email␣byl␣jiz␣zabran.')
       .....
});
```

Kód 8.3: Testování registračního formuláře a odpovědí serveru

Pro účely této webové aplikace bylo vytvořeno 15 testů. Při vytváření testů byly nalezeny v průměru 2 chyby na test. Testováním se odhalili důležité problémy jako neodeslání emailu, chybějící údaje v detailu hesla nebo nahrávání duplicitních záznamů do tabulky obcí. Všechny chyby byly následně opraveny přímo v Controller třídách. Zajímavé metody Laravel Dusk rozšíření jsou oproti jiným testovacím nástrojům třeba možnost nasimulovat kurzor myši přejíždějící přes nějaký element ve webové aplikaci nebo simulace psaní textu držením klávesy SHIFT. Pro testování metod vyžadujících autorizovaného uživatele je také umožněno použít metodu **loginAs**, která jako parametr požaduje id uživatele existujícího v databázi. Tímto se lze vyhnout neustálým opakováním vyplňování formuláře pro přihlášení při každém spuštěném testu.

9 Dosažené výsledky

Vybrané technologie k vypracování webové aplikace splnily všechna očekávání. Programování back-endu webové aplikace použitím Laravel frameworku nebylo obtížné, především díky mnohačetných video návodů pro začátečníky s Laravelem. Při práci s front-end frameworkem MDL bylo očekáváno, že všechny komponenty ještě nebudou zcela bezproblémové, když je MDL na poli front-end frameworků přibližně dva roky. Webová aplikace obsahuje ještě drobné chyby v zalamování textů a algoritmicky náročnější zpracování zaškrtnutých filtrů, avšak tyto problémy neznemožňují správnou funkčnost webu.

Funkce výsledného produktu byly implementovány podle návrhu a splňují všechny kladené požadavky. Z původní databáze elektronického nářečního slovníku ZČU byla přesunuta data do nové databáze s minimální ztrátou informací, kdy nebylo přesunuto pět frazémů. Při testování nebyly nalezeny chyby, které by ohrožovaly funkčnost celé webové aplikace, a aplikace je připravena nahradit doposud funkční elektronický nářeční slovník vytvořený na Západočeské univerzitě.

10 Závěr

V rámci této bakalářské práce bylo analyzováno několik existujících českých elektronických nářečních slovníků. Výsledky této analýzy poukázaly na uživatelské nedostatky všech webů. Elektronický nářeční slovník vytvořený na Západočeské univerzitě obsahoval chyby ve funkcionalitě a správa hesel či uživatelů byla velmi problémová. Vzhledem ke zjištěným nedostatkům došlo k navrhnutí nového řešení nahrazující původní systém a konkurující existujícím elektronickým nářečním slovníkům.

Z analýzy dostupných technologií pro tvorbu webu byly vybrány vhodné frameworky. Pomocí frameworků byl vytvořen desing webové aplikace a hlavní funkce, které jsou dostupné uživatelům v závislosti na úrovni pověření. Prohlížení hesel je umožněno všem návštěvníkům webu, pro vkládání nových hesel je pak nutná registrace. Správa hesel je omezená pouze na specifický výběr lidí, kteří k těmto akcím dostanou práva od administrátora. Administrátor jako jediný může spravovat ostatní uživatele, kterým může přidělovat oblasti pro správu hesel.

V úvodu 4. kapitoly bylo stanoveno, že vytvořená webová aplikace by měla nahradit stávající elektronický nářeční slovník ZČU. V současné době ještě aplikace tento slovník nenahrazuje, avšak je dostupná z dočasného hostingu na stránce `https://lukado.hornet-cz.com/`. Do budoucna je pro práci připraveno ostré nasazení do provozu a vyladění designu na mobilních zařízeních, které při testování měly problém se správným zobrazením formátování textu. Plánované je také převedení vybraných hesel do textového dokumentu, který bude vhodný pro tisk.

Literatura

[1] *Slovník nářečí českého jazyka* [online]. dialektologické oddělení Ústavu pro jazyk český AV ČR, 2016. [cit. 2016/10/22]. Dostupné z: http://sncj.ujc.cas.cz.

[2] Jiří Berger, J. M. R. M. Z. H. *Nářeční slovník Západočeské univerzity v Plzni* [online]. Katedra informatiky a výpočetní techniky, Fakulta aplikovaných věd, Západočeská univerzita v Plzni a Ústav bohemistiky a knihovnictví, Filozoficko-přírodovědecká fakulta, Slezská univerzita v Opavě, 2006. [cit. 2016/10/22]. Dostupné z: http://narecnislovnik.kiv.zcu.cz.

[3] *Valašsko-český slovník* [online]. Folta Josef, 2012. [cit. 2016/10/22]. Dostupné z: https://www.valassky.cz.

[4] *Slovník valašského nářečí* [online]. neuvedeno, 2011. [cit. 2016/10/22]. Dostupné z: http://www.jurkovic.cz.

[5] *Slovník brněnského hantecu* [online]. Bronislav Marek, 2007. [cit. 2016/10/22]. Dostupné z: http://www.hantec.cz/hantec/slovnik/slovnik.htm.

[6] *Slovník po naszi(y)mu* [online]. Klus Jan, 2001. [cit. 2016/10/22]. Dostupné z: http://www.blaf.cz/index.php?body=slovnik.

[7] Petr Lukašík, J. P. J. S. *Webová aplikace pro správu kulturních artefaktů* [online]. 2016. [cit. 2017/05/01].

[8] *Markup Validation Service* [online]. W3C, 1994. [cit. 2016/11/28]. Dostupné z: https://validator.w3.org.

[9] *Mobile test* [online]. Vangelis Bibakis, 2014. [cit. 2016/11/28]. Dostupné z: http://mobiletest.me.

[10] *Laravel - Server requirements* [online]. Laravel, 2017. [cit. 2017/04/21]. Dostupné z: https://laravel.com/docs/5.4#server-requirements.

[11] Arsenault, C. *Top 10 Front-End Frameworks of 2016* [online]. KeyCDN, 2016. [cit. 2017/02/02]. Dostupné z: https://www.keycdn.com/blog/front-end-frameworks.

[12] *Semantic UI - RELEASE NOTES* [online]. Semantic-UI, 2016. [cit. 2017/02/02]. Dostupné z: https://github.com/Semantic-Org/Semantic-UI/blob/master/RELEASE-NOTES.md#version-220---june-26-2016.

[13] *Material Design for Android* [online]. Google, 2016. [cit. 2017/02/02]. Dostupné z: https://developer.android.com/design/material/index.html.

[14] MICHALICOVÁ, J. Databázový model a WWW rozhraní výkladového slovníku. Diplomová práce, Západočeská univerzita, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2006.

[15] *Databáze adres v ČR a číselníky územních celků* [online]. Ministerstvo vnitra České republiky, 2011. [cit. 2017/04/27]. Dostupné z: http://www.mvcr.cz/clanek/databaze-adres-v-cr-a-ciselniky-uzemnich-celku.aspx.

[16] *Laravel - Documentation* [online]. Laravel, 2017. [cit. 2017/04/20]. Dostupné z: https://laravel.com/docs/5.4.

[17] *Laracast - Laravel* [online]. Laracasts, 2017. [cit. 2017/04/20]. Dostupné z: https://laracasts.com/skills/laravel.

[18] *Material Design Lite - Components* [online]. Google Design, 2015. [cit. 2017/04/27]. Dostupné z: https://getmdl.io/components.

[19] *Laravel - Installation* [online]. Laravel, 2017. [cit. 2017/04/27]. Dostupné z: https://laravel.com/docs/master/installation#installing-laravel.

[20] *Laravel - Blade Templates* [online]. Laravel, 2017. [cit. 2017/04/27]. Dostupné z: https://laravel.com/docs/5.4/blade.

[21] *Material Design Lite - Dialogs* [online]. Google Design, 2015. [cit. 2017/04/27]. Dostupné z: https://getmdl.io/components/index.html#dialog-section.

[22] KOSEK, J. *PHP tvorba interaktivních internetových aplikací.* Grada Publishing, 1999. ISBN 80-7169-373-1.

Seznam zkratek

- **PDF** - Portable Document Format - univerzální souborový formát pro ukládání dokumentů
- **HTML** - Hypertext Markup Language - značkovací jazyk
- **PHP** - Hypertext Preprocessor (původně Personal Home Page) - skriptovací programovací jazyk
- **MDL** - Material Design Lite - front-end framework pro webové stránky
- **JS** - JavaScript - objektově orientovaný skriptovací jazyk
- **CSRF** - Cross-site request forgery - druh útoku do webových aplikací využitím chyb webu
- **XSS** - Cross-site scripting - útok do webových aplikací využitím chyb u skriptů
- **CSS** - Cascading style sheets - jazyk pro zobrazení elementů napsaných v HTML, XHTML nebo XML
- **AES** - Advanced Encryption Standart - symetrická bloková šifra
- **ERA** - Entity, Relation, Attribute - specifikace struktury databáze
- **SQL** - Structured Query Language - standartizovaný strukturovaný dotazovací jazyk
- **MVC** - Model, View, Controller - softwarová architektura
- **HTTP** - Hypertext transfer protocol - internetový protokol pro výměnu HTML dokumentů
- **URL** - Uniform resource locator - definovaná struktura odkazující na přesné umístění dokumentu na internetu
- **ORM** - Object-relational mapping - programovací technika

Přílohy

A Obsah přiloženého DVD

- Dokumentace

  src - zdrojové soubory LaTeX a obrázky

  out - vygenerovaný soubor PDF s textem bakalářské práce

  diag - zdrojové soubory diagramů

- Webová aplikace

  dictionary - zdrojové soubory webové aplikace

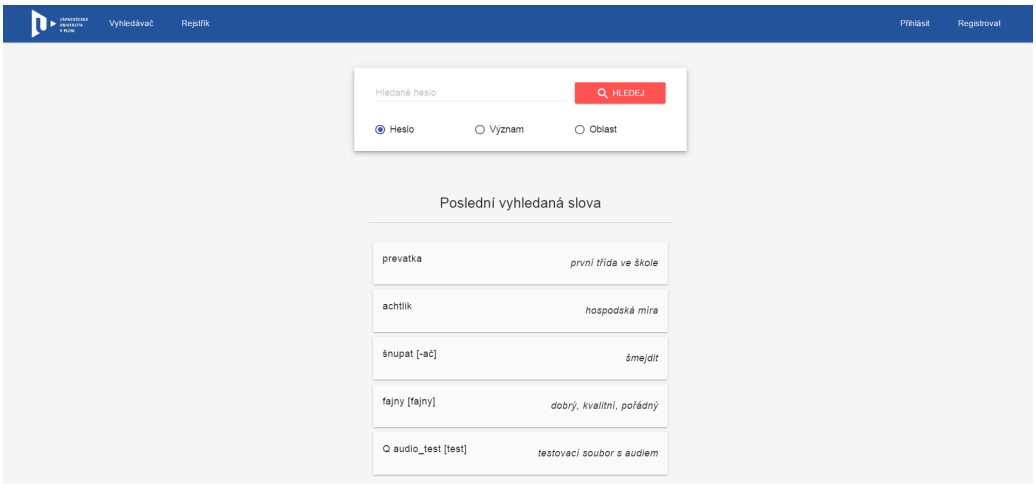
  tests - testovací třídy a výsledky testů

  database - SQL soubory původní a nové databáze

B Uživatelská dokumentace

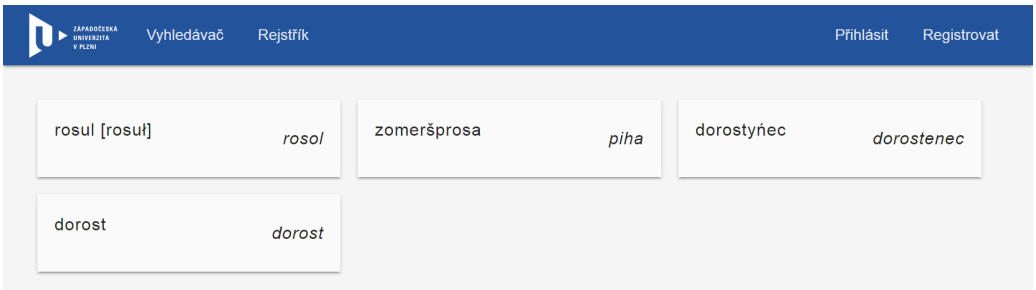
Hlavní strana a vyhledávání

Na hlavní straně se nachází vyhledávač a seznam naposledy vyhledaných hesel (viz Obr 1). Do vyhledávače se zadávají textové řetězce, který je po odeslání hledán v heslu. Hesla lze vyhledávat podle jejich názvu, významu



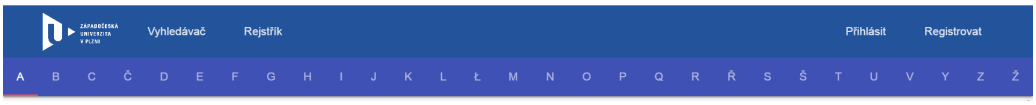
Obrázek 1: Hlavní strana webové aplikace

nebo oblasti výskytu. V případě nenalezení žádného záznamu je v šabloně katalogu zobrazena zpráva o nenalezení žádného hesla. V seznamu naposledy vyhledaných hesel lze na jednotlivá hesla kliknout pro zobrazení detailu daného hesla. Při nalezení hesel obsahující vyhledávaný textový řetězec jsou uživateli zobrazeny všechna hesla v seznamu (viz Obr. 2). Jednotlivá hesla lze rozkliknout pro zobrazení detailu hesla. Pro vyhledání hesla podle pís-



Obrázek 2: Katalog zobrazovaných hesel obsahující řetězec "ros"

mena na které začíná se nachází na webu rejstřík. Na stránce s rejstříkem se zobrazují v záhlaví písmena abecedy, na která hesla začínají (viz Obr. 3). Všechna hesla začínající na jedno ze zobrazovaných počátečních písmen



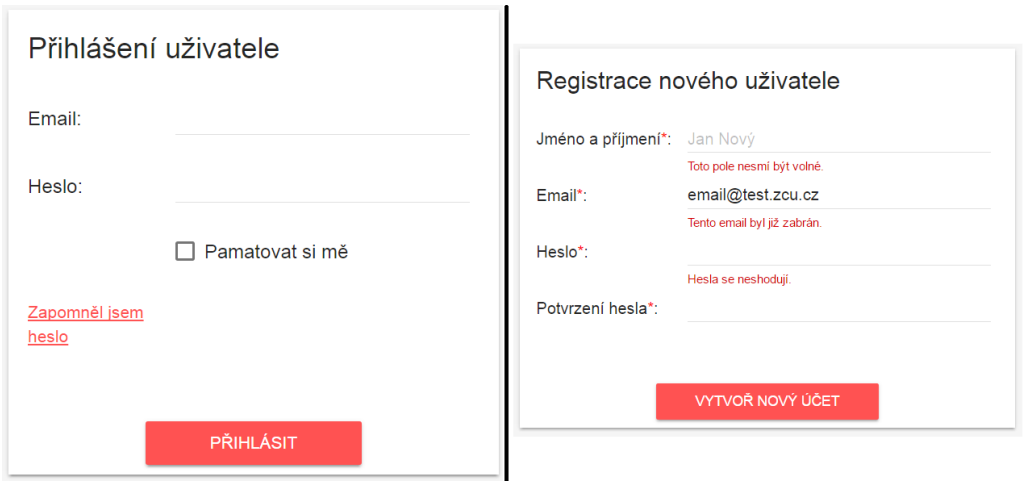
Obrázek 3: Záhlaví webu s rejstříkem

mohou být zobrazena kliknutím na záložku daného písmena. Jestliže neexistuje žádný záznam hesla v databázi začínající na písmeno W, nebude tato záložka zobrazena. V detailu jsou zobrazeny všechny informace o heslu ukládané v databázi. Pokud k heslu existuje audio soubor, je možné ho kliknutím na tlačítko stažení uložit na svém zařízení.

Registrace a přihlášení uživatele

Uživatel se může registrovat do systému vyplněním formuláře registrace. Nachází se zde pole **Nový email**, **Nové heslo**, **Potvrzení nového hesla** a **Staré heslo**. Každé pole je kontrolováno validací podle vybraných pravidel. Ve formuláři registrace i přihlášení je každý vstupní parametr povinný. Podmínkou pro pole jména a příjmení je délka, která musí být v rozmezí 3 až 30 znaků. Email, který slouží jako identifikátor při přihlašování spolu s heslem, musí být unikátní a musí splňovat vzor adresy emailu. U hesla je zase kontrolována minimální délka 6 znaků a maximální délka 30 znaků. Heslo se také musí shodovat s polem pro potvrzení hesla. Pokud uživatel pošle nevalidní data, vzniká-li tedy chybný vstup u některého textového pole, server vrátí naplněnou proměnnou předdefinovanými chybovými hláškami, které se zobrazují pod textovými poli. Tato situace je zobrazena na Obr. 4. Vrácení uživatele na předchozí stránku s chybami by však znamenalo, že by všechna pole musel zadávat znova, jelikož si klient zadané hodnoty nepamatuje. Server si však vstupní data pamatuje a spolu s navrácením na předchozí stránku, v případě chybné validace, je pošle uživateli.

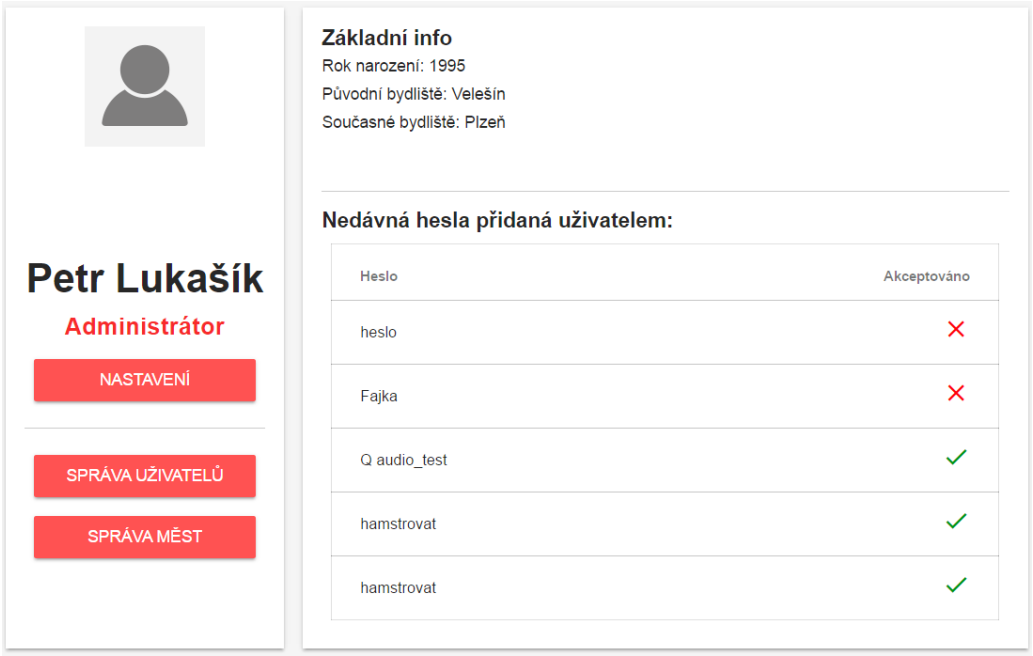
Přihlášení uživatele se provádí kliknutím na odkaz **Přihlásit**. Formulář přihlášení požaduje zadání registrované emailové adresy a hesla (viz Obr. 4). Volitelné pole pro zapamatování údajů lze zaškrtnout v případě, že uživatel chce být zapamatován, aby se nemusel při příští návštěvě webu znovu přihlašovat. V případě zapomenutého hesla lze kliknout na odkaz **Zapomněl jsem heslo**. Objeví se nový formulář pro zadání emailové adresy použité při registraci, na kterou budou odeslány pokyny pro obnovení hesla.



Obrázek 4: Formulář přihlášení a formulář registrace nového uživatele s chybovými hláškami

Funkce registrovaného uživatele

Každý přihlášený uživatel má možnost si zobrazit svůj profil, kliknutím na své jméno v záhlaví webu. Profil obsahuje v levém panelu základní informace *Jméno*, *Pověření* a tlačítka pro správu účtu. Administrátorovi se navíc zobrazují tlačítka pro správu uživatelů a hesel (viz Obr. 5). V panelu uprostřed

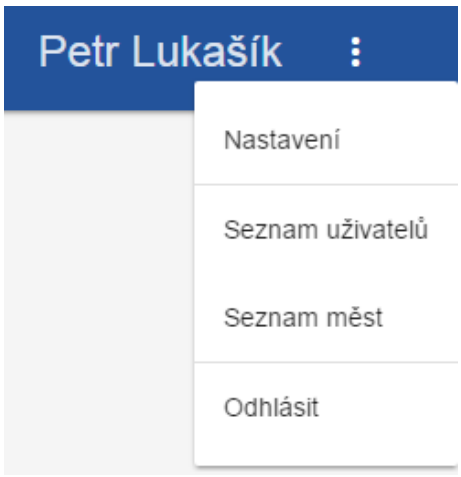


Obrázek 5: Profil uživatele s informacemi

obrazovky se zobrazují obecné údaje o uživateli a případně kraje, pro které

daný uživatel spravuje hesla. Pod těmito údaji se nachází seznam nedávno přidaných hesel tímto uživatelem. U jednotlivých hesel je zobrazeno jestli již byla schválena intuitivními ikonkami.

Přihlášený uživatel si v pravém horním rohu může rozkliknout nabídku obsahující odkazy na nastavení a odhlášení uživatele (viz Obr. 6). Nabídka

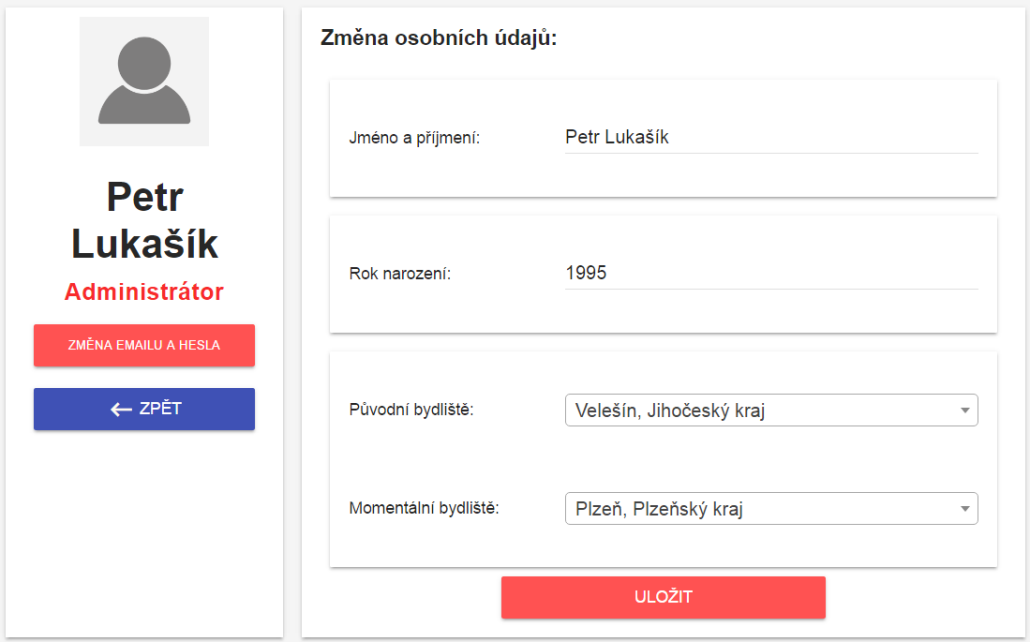


Obrázek 6: Vyskakovací menu v pravém horním rohu záhlaví

také obsahuje odkazy na seznam uživatelů a seznam oblastí. Při kliknutí na odkaz nastavení se uživateli zobrazí formulář s textovými poli **Jméno a příjmení**, **Rok narození** a dva vyhledávače pro **Původní bydliště** a **Současné bydliště** se seznamem obcí pro nastavení osobních údajů (viz Obr. 7). Poskytnuté informace od uživatele lze využít při potvrzování věrohodnosti původu hesel. Všechna tato pole jsou nepovinná pro vyplnění a v případě ponechání všech polí prázdných se záznam v databázi neaktualizuje. Pole pro rok narození je na klientské straně omezeno na 4 číselné znaky, které v případě nečíselné hodnoty upozorňují chybou na nevalidní vstup. Pro pole jména a příjmení se provádí kontrola pro rozsah mezi 3 až 30 znaky. Vstup pro rok narození musí být číselný, čtyřmístný a musí být v rozsahu dnešního roku do 150 let před dnešním rokem.

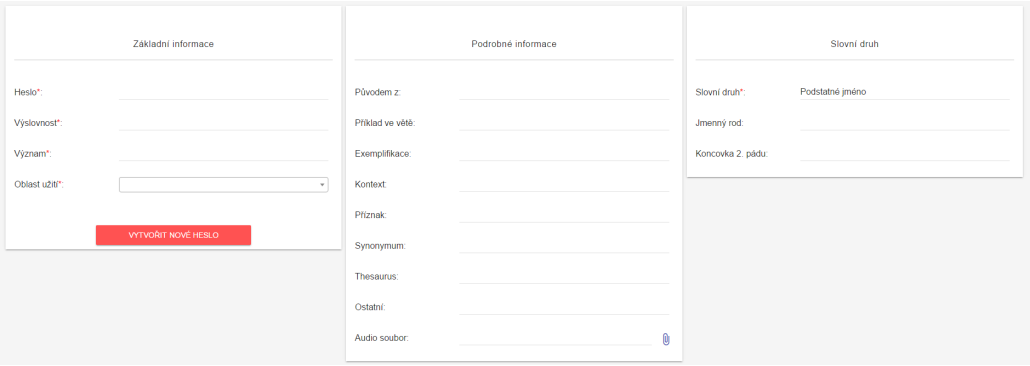
Přes stránku nastavení se lze dostat kliknutím na tlačítko **Změna emailu a hesla** v levém panelu dostat na formulář pro změnu soukromého nastavení. Formulář je velice podobný registračnímu formuláři a validační pravidla platí také stejná. Navíc je pouze přidané pole *Potvrzení starého hesla*, do kterého se musí zadat stávající heslo pro potvrzení změny v emailu nebo heslu.

Pro možnost nahrávání nových hesel do systému musí být uživatel přihlášen. Účel tohoto rozhodnutí pochází z obavy proti záškodníkům či robotům, kteří by mohly naplňovat databázi zbytečnými daty. Jakmile je uživatel přihlášen, zobrazí se mu nový odkaz **Nové heslo** v záhlaví vpravo od rejstříku



Obrázek 7: Formulář změny nastavení osobních údajů uživatele

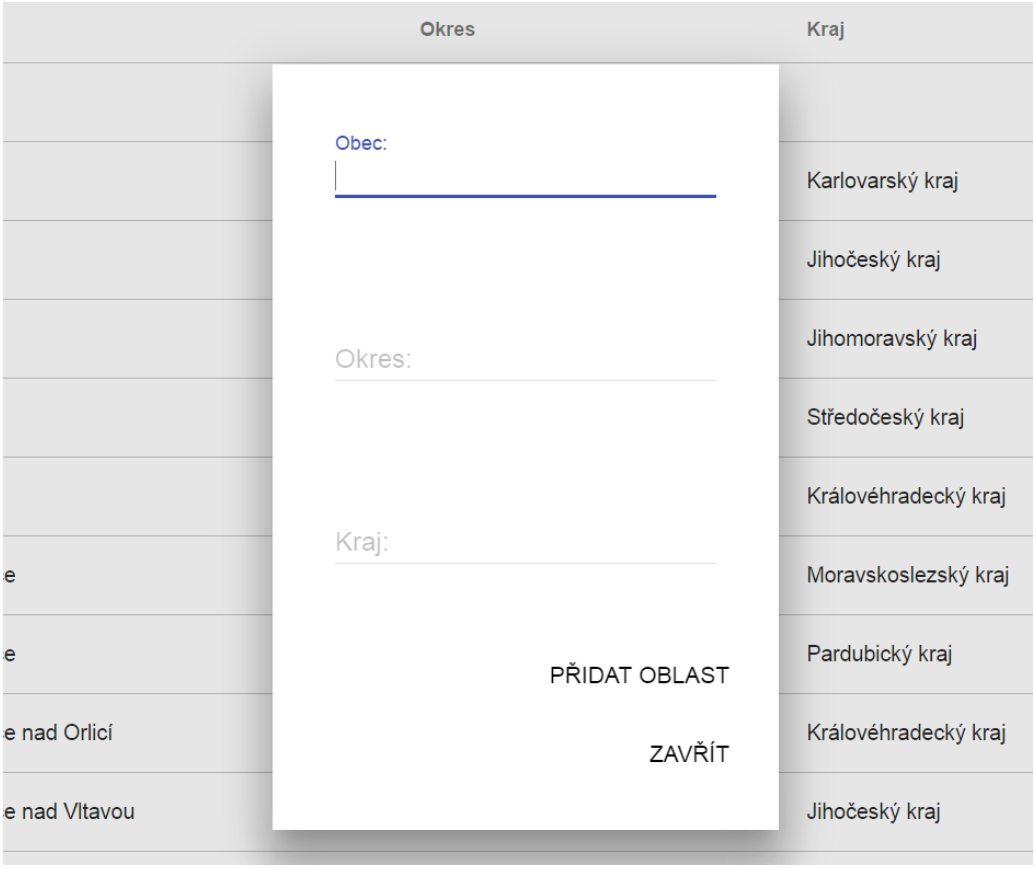
hesel. Při vyplňování hesla je nutné vyplnit pole *heslo*, *výslovnost*, *význam*, *oblast užití* a *slovní druh* (viz Obr. 8). Všechna ostatní pole jsou nepo-



Obrázek 8: Formulář pro přidání nového hesla

vinná. Obce se zobrazují ve formátu "Obec, Okres, Kraj", tudíž by neměl být problém s vyhledáním požadované oblasti. Při výběru slovního druhu ze seznamu se v případě výběru podstatného jména nebo slovesa objeví ještě nepovinné kolonky pro dodatečné informace k těmto slovním druhům. Všechna data se při odeslání na server proberou validací a v případě chyby se uživatel přesměruje zpět a pod daným polem zobrazí chybová zpráva.

Uživatel v seznamu měst (viz Obr. 9) má možnost kliknutím na tlačítko **Přidat novou oblast** přidat novou oblast. Uživateli se zobrazí dialog okno

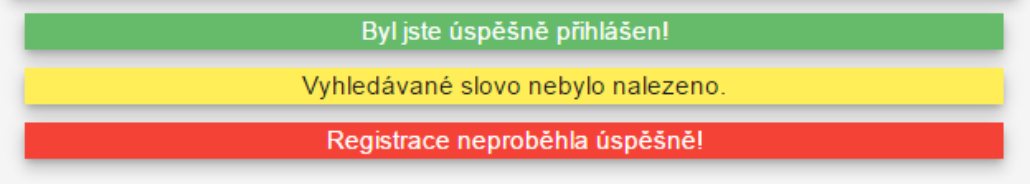


Obrázek 9: Dialog přidání nové oblasti

s formulářem pro přidání nového města (viz Obr. 9). Odeslání formuláře musí být provedeno stisknutím tlačítka **Přidat oblast**. V případě duplicitního záznamu nebo špatně zadaných polí se zobrazí chybová notifikační zpráva na spodní části webové stránky seznamu měst.

Na stránkách se vyskytují mnou navržené koncepty kontejnerů na zpracování a prezentaci notifikačních zpráv. Každý typ zprávy je zobrazen v jinak barevném pozadí kontejneru a spojen s určitou informací. Pro úspěšnou zprávu je přidělena zelená barva a vyznačuje nějaký úspěch, chybová zpráva je zobrazena v červeném kontejneru a označuje neúspěch a informativní zpráva je zvýrazněna žlutou barvou a slouží pouze k výpisu věty informativního rázu viz Obr. 10.

Prozatím je v seznamu měst a uživatelů zobrazeno pouze 15 záznamů. Existuje-li více než 15 záznamů v databázi je ve spodní části tabulky vytvořeno stránkování (viz Obr. 11). Na obrázku lze i vidět, že stránka, na které se nacházíme, je zablokovaná. Po najetí myší na číslo se dané číslo zvýrazní. Pohybovat se po stránkách je možné kliknutím na přesné číslo v liště nebo



Obrázek 10: Kontejnery flash zpráv

jednokrokovým posuvem kliknutím na postranní šipky.

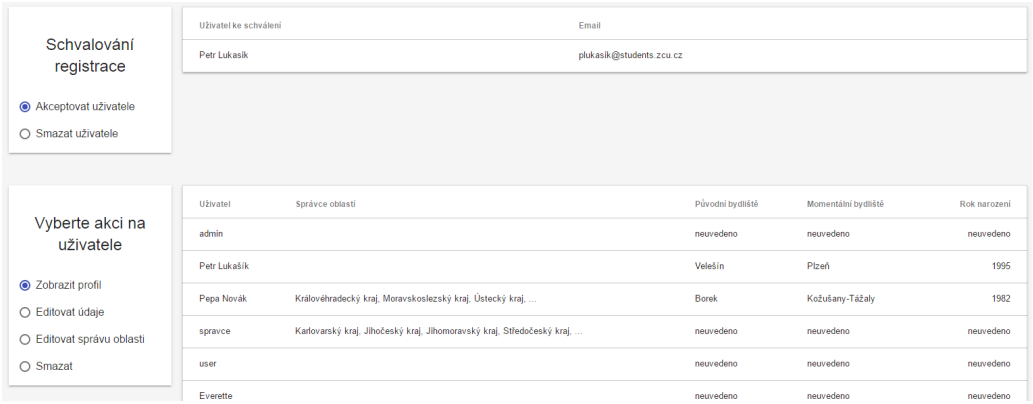
Obrázek 11: Element stránkování seznamu

Funkce administrátora

Kromě všech výše zmíněných akcí má administrátor funkce navíc. Administrátor webu má jako jediný uživatel práva pro schvalování nebo zamítání nových registrací. Důležité notifikace, jako jsou nově registrovaní uživatelé nebo nová přidaná hesla, se zobrazují administrátorovi v záhlaví webu vedle jména (viz Obr. 12). Na notifikace nových uživatelů lze kliknout jako

Obrázek 12: Část horní lišty s notifikacemi pro administrátora

na odkaz, který přesměruje administrátora na stránku seznamu uživatelů. Seznam uživatelů, který si mohou prohlídnout všichni registrovaní uživatelé webu, obsahuje pouze pro administrátora ovládací panel pro správu uživatelů (viz Obr. 13. Seznam uživatelů obsahuje sloupce *Jméno uživatele*, *Správce oblastí*, *Původní bydliště*, *Současné bydliště* a *Rok narození*. V každé řádce je uvedeno id uživatele a jeho jméno k umožnění práce s těmito daty po vybrání akce. V případě neuvedených osobních údajů se zobrazuje v kolonce text **neuvedeno**. V horní tabulce nových uživatelů lze vybrat z filtrů pouze možnost schválení nebo zamítnutí uživatele. Po vybrání daného filtru se akce provede kliknutím na uživatele horní tabulky. Schválení automaticky také posílá uvítací email uživateli, akce tedy někdy trvá trochu déle. Při použití akce **Smazat** na uživatele vyskočí okno se jménem uživatele a potvrzením

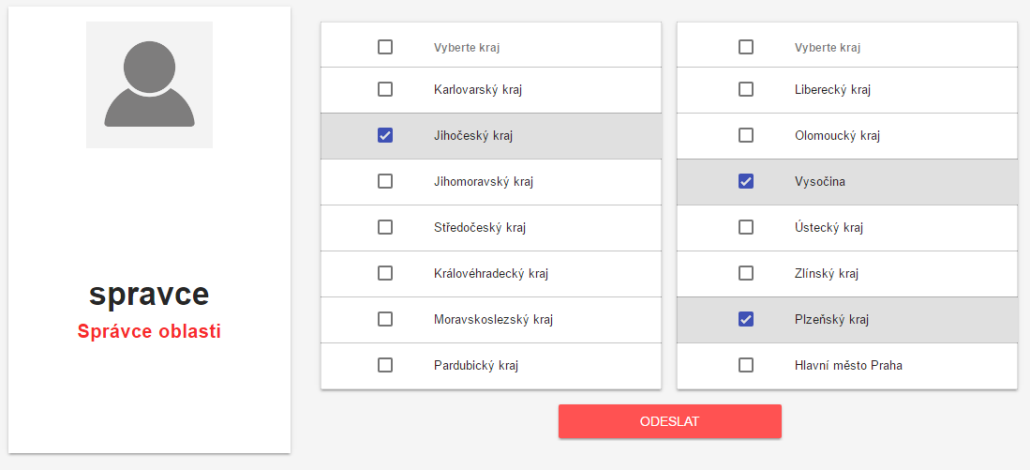


Obrázek 13: Seznam uživatelů s panely pro správu uživatelů

o smazání zadáním hesla administrátora. Tato kontrola by měla zabránit smazání uživatele, který byl omylem vybrán administrátorem se zaškrtnutou akcí smazání.

Ke spodní tabulce se nachází po levé straně panel s filtry **Zobrazit profil**, **Editovat údaje**, **Editovat správu oblasti** a **Smazat**. Akce se také provádí kliknutím na uživatele v tabulce. Zobrazení profilu přesměruje administrátora na profil daného uživatele. Editování údajů přesměruje administrátora na stejný formulář pro nastavení osobních údajů o uživateli, který obsahuje stejná pravidla při vyplňování polí. Akce smazání uživatele se provádí stejně jako u tabulky nových uživatelů. Jediná neznámá akce zde je tedy úprava správy oblastí pro daného uživatele. Každý uživatel může mít přiřazenou správu oblasti podle kraje. Pokud uživatel spravuje například „Jihočeský kraj“, všechna nově přidaná hesla nacházející se v tomto kraji, budou zobrazena notifikací u daného uživatele, který je následně může upravovat, schvalovat nebo mazat. Administrátor je po akci přidání nové oblasti uživateli odkazován na šablonu **distSettings.blade.php**, která obsahuje zaškrtávací seznam 14 krajů České republiky viz Obr. 14. Zaškrtáváním jednotlivých krajů vybírá, které budou přidány uživateli pro správu hesel. Po odeslání seznamu je v seznamu uživatelů zobrazeno u záznamu daného uživatele, které oblasti spravuje.

Stejná notifikace jako pro nově registrované uživatele, je vytvořena v záhlaví webu pro nově přidané heslo přidané od schválených registrovaných uživatelů. Výjimkou je však, že tato notifikace se nezobrazuje pouze administrátorovi, avšak i všem správcům spravující oblast, ve které se heslo užívá. Hesla zobrazená na stránce připomínají stejnou tabulku jako pro seznam uživatelů nebo měst. Hesla jsou v tabulce popsaná *názvem*, *výslovností*, *významem*, *oblastí užití* a *jméno uživatele*, který heslo přidal. Po levé straně



Obrázek 14: Formulář výběru oblastí

hlavního obsahu mohou správci a administrátor znovu nalézt panel s přepínačem možných akcí nad hesly. Stejně jako u správy uživatele se zde nachází editace, smazání a schválení, tentokrát však pro heslo. Schválená hesla lze ihned vyhledat přes vyhledávač na hlavní stránce a také lze takové heslo najít v rejstříku. Při editaci hesla je uživatel přesměrován na formulář, který je vizuálně stejný jako šablona pro přidání nového hesla. Jediným rozdílem jsou vyplněné kolonky nově přidávaného hesla od autora. Administrátor a správci oblastí při zobrazení detailu schváleného hesla pod spravovanou oblastí vidí oproti ostatním navíc tlačítka editace a smazání. Takto mohou upravit či smazat přesně vyhledané heslo v zobrazení detailu.