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

Bakalářská práce

Prototyp serverové části pro komunitní překlad popisků z muzeí

Plzeň 2022 Pavel Čácha

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Pavel ČÁCHA**
Osobní číslo:	**A19B0022P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Prototyp serverové části pro komunitní překlad popisků z muzeí**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s problematikou komunitního překladu a s návrhem systému pro překlad popisků.
2. Seznamte se se současnými technologiemi pro implementaci serveru a vyberte vhodnou sadu technologií pro realizaci bakalářské práce.
3. Navrhněte serverovou část pro komunitní překlad textů a dalších dat z muzeí.
4. Implementujte navržený server.
5. Otestujte navržený systém s ohledem na bezpečnost a funkčnost.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Richard Lipka, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 4. května 2022

Pavel Čácha

Abstract

The main goal of this thesis is to create a server for community translation of museum labels, which can also be used by exhibitions, galleries or similar cultural institutions. A web interface for translators and institution administrators is prepared to manage the server. At the start of the project, it was necessary to get acquainted with the available software solutions for community translations that could be used in the project and to analyze the available technologies for the creation of modern servers and web interfaces. Then the whole system was designed and implemented. The work also includes a set of automatic tests.

Abstrakt

Hlavním cílem práce je vytvořit server pro komunitní překlad dat z muzeí, který ale mohou používat například také výstavy, galerie nebo podobné kulturní instituce. K jeho správě je připraveno webové rozhraní určené pro překladatele a správce institucí. Nejprve bylo zapotřebí seznámit se s dostupnými softwarovými řešeními pro komunitní překlad, které by mohly být v projektu využity, a také provést analýzu dostupných technologií pro tvorbu moderních serverů a webových rozhraní. Poté došlo k návrhu a implementaci celého systému. Součástí práce je i sada automatických testů.

Poděkování

Rád bych tímto poděkoval Ing. Richardu Lipkovi, Ph.D. za cenné rady, vstřícný přístup a vedení mé bakalářské práce.

Obsah

1 Úvod **5**

2 Problematika komunitního překladu **6**
2.1 Komunitní překlad 6
2.2 Přístup k formátu překladu 7
2.2.1 Komunitní překlad s validací od jednoho správce nebo skupiny správců 7
2.2.2 Komunitní překlad s validací od skupiny uživatelů . . 7
2.2.3 Shrnutí 8

3 Ucelená softwarová řešení podporující komunitní překlad **9**
3.1 Crowdin 9
3.1.1 Výhody 10
3.1.2 Nevýhody 10
3.2 Pootle 11
3.2.1 Výhody 11
3.2.2 Nevýhody 11
3.3 Translate Toolkit 12
3.3.1 Výhody 12
3.3.2 Nevýhody 12
3.4 MediaWiki v kombinaci s rozšířením pro překlad 12
3.4.1 Výhody 12
3.4.2 Nevýhody 13
3.5 Weblate 13
3.5.1 Výhody 13
3.5.2 Nevýhody 14
3.6 Shrnutí 14

4 Existující API architektury **15**
4.1 SOAP 15
4.1.1 Struktura SOAP zprávy 15
4.2 REST 16
4.2.1 Principy RESTu 16
4.2.2 Vlastnosti RESTu 17
4.3 GraphQL 17
4.3.1 Principy GraphQL 17

4.3.2 Vlastnosti GraphQL 18
4.4 Shrnutí 18

5 Analýza dostupných technologií **19**
5.1 Technologie pro implementaci RESTového serveru 19
5.1.1 Django 19
5.1.2 Express.js 20
5.1.3 Spring Boot 21
5.1.4 Rails 22
5.1.5 Srovnání popularity 24
5.1.6 Shrnutí 25
5.2 Technologie pro implementaci webového rozhraní 25
5.2.1 Angular 26
5.2.2 React 27
5.2.3 Vue 28
5.2.4 Srovnání popularity 29
5.2.5 Shrnutí 30

6 Návrh systému **31**
6.1 Uživatelské role 31
6.1.1 Nepřihlášený uživatel 31
6.1.2 Správce kulturní instituce 31
6.1.3 Překladatel 32
6.1.4 Administrátor 32
6.1.5 Návštěvník kulturní instituce 32
6.2 Případy užití 33
6.3 Databázový model 34
6.3.1 Tabulka user 35
6.3.2 Tabulka role 35
6.3.3 Tabulka institution 35
6.3.4 Tabulka exhibit 36
6.3.5 Tabulka translation 36
6.3.6 Tabulka language 37
6.3.7 Tabulka building 37
6.3.8 Tabulka room 38
6.3.9 Tabulka showcase 38
6.3.10 Ostatní tabulky 38
6.4 Koncové body (endpointy) serveru 39
6.4.1 Koncové body související s uživateli 39
6.4.2 Koncové body související s institucemi 40

6.4.3 Koncové body související s exponáty 42
6.4.4 Koncové body související s překlady 44
6.4.5 Koncové body související s administrací 45
6.4.6 Koncové body spojené s umístěním exponátu 47

7 Popis implementace **50**
7.1 Implementace serveru 50
7.1.1 Databáze napojená na server 50
7.1.2 Obecný průběh zpracování požadavku 50
7.1.3 Struktura a popis projektu 52
7.2 Implementace webového rozhraní 59
7.2.1 Použité technologie 59
7.2.2 Struktura a popis projektu 60

8 Ověřování kvality **65**
8.1 Programové testy 65
8.1.1 Testy servisních tříd 65
8.1.2 Integrační testy 66
8.1.3 Bezpečnostní testy 66
8.1.4 Testy pomocí Selenia 66
8.2 Ověřování odolnosti proti útokům 66
8.2.1 SQL injekce 67
8.2.2 XSS 67
8.3 Uživatelské testy 68
8.3.1 Provedené opravy 68
8.3.2 Návrhy na rozšíření systému 69

9 Závěr **70**

Přehled zkratek **72**

Literatura **73**

A Instalace a spuštění programu **79**

B Uživatelská příručka **80**
B.1 Překladatel 80
B.2 Správce instituce 81
B.3 Administrátor systému 81
B.4 Ukázky obrazovek 82
B.4.1 Založení kulturní instituce 82
B.4.2 Založení exponátu 83

B.4.3 Vytvoření překladu 84
B.4.4 Schválení překladu 85
B.4.5 Správa verzí překladu 86
B.4.6 Správa uživatelů 87

C Popis adresářové struktury odevzdávaného souboru 88

D Zprávy z testování 89
D.1 Zpráva testera A 89
D.2 Zpráva testera B 89
D.3 Zpráva testera C 90
D.3.1 Vizuální stránka 90
D.3.2 Funkcionální stránka 90
D.3.3 Shrnutí 90
D.4 Zpráva testera D 91
D.5 Zpráva testera E 91
D.6 Zpráva testera F 92

1 Úvod

Cílem této bakalářské práce je vytvořit prototyp webového serveru, který bude sloužit pro vytváření překladů popisků muzejních exponátů a jejich snadné získávání za účelem zobrazení uživateli, tedy návštěvníkovi kulturní instituce. Součástí práce je také tvorba webového rozhraní, které bude komunikovat se serverem a bude umožňovat aktivním členům překladatelské komunity překládat jednotlivé muzejní texty na základě podnětů od návštěvníků ve formě fotografií informačních panelů nebo štítků.

V současné době probíhá proces takzvaného zmenšování světa, který se projevuje rychlým a efektivním transportem osob, materiálu a informací. Mnoho lidí si tak zvyklo na globální styl života a s tím jde ruku v ruce cestování. Při svých cestách pak tito lidé rádi navštěvují kulturní zařízení všeho druhu. V institucích, kde se klade důraz na předání informační hodnoty (např. muzea), se pak dostávají do pasti jazykové bariéry. Stává se to z toho důvodu, že texty jsou psané jen v omezeném množství jazyků a lidé neovládající ani jeden z nich poté nemají z návštěvy plnohodnotný zážitek.

Tento problém lze řešit mobilní aplikací, která by sloužila jako průvodce a umožňovala by zobrazování popisků v různých jazycích. Kulturním institucím ale často scházejí peníze potřebné na vývoj softwaru a také na honorář profesionálních překladatelů. Praktická část této bakalářské práce si tedy klade za cíl vyřešit problém s financováním části (server a webové rozhraní) softwarového produktu, protože by byl přístupný všem a zadarmo. Problém s financováním překladatelské části je eliminován využitím komunitního překladu.

Bakalářská práce přímo navazuje na analýzu sepsanou v dokumentu *Analýza existujících mobilních průvodců po muzeích a památkách*. Z tohoto textu jasně vyplývá, že kulturní instituce o takový typ aplikace zájem mají. Mezi těmi nejznámějšími jsou například Západočeské muzeum a Západočeská galerie v Plzni. Instituce, které projevily zájem, byly pak dotazovány na preferovaný přístup ke komunitnímu překladu. Ze všech možných přístupů byl projeven největší zájem o *komunitní překlad s validací od skupiny uživatelů* a o *komunitní překlad s validací od jednoho správce nebo skupiny správců*. Tyto dvě možnosti se liší zejména v různém pojetí zajišťování kvality překladů. Analýza také představuje možné technologie pro tvorbu interaktivního průvodce a existující ucelená řešení pro komunitní překlad, ze kterých tato práce vychází [55].

2 Problematika komunitního překladu

Tato kapitola si klade za cíl představit komunitní překlad a problémy s ním spjaté. Dále jsou zde uvedeny jednotlivé modely jeho fungování, které mohou být realizovány.

2.1 Komunitní překlad

S rozmachem internetu došlo k tomu, že se v on-line prostoru mohou velmi snadno sdružovat velké skupiny lidí se společnými zájmy a to bez ohledu na to odkud pocházejí. Tento fenomén umožnil prudký rozmach komunitního překladu. Takový typ překladatelských prací se, jak již název napovídá, opírá o silnou komunitu tvůrců. Tito lidé jsou vázáni nějakou společnou zálibou a věnují se překladu obvykle zcela zdarma na dobrovolnické bázi ve svém volném čase. Tím je produkováno velké množství odvedené práce, které by za předpokladu využití služeb profesionálů stálo nemalé finanční prostředky.

Této formy překladu nevyužívají pouze nekomerční projekty menšího rozsahu, ale také mezinárodní firmy s obrovskými rozpočty. Příkladem může být sociální síť Twitter, která spustila překladatelské centrum pro dobrovolníky, pomocí kterých došlo k přeložení rozhraní Twitteru do nizozemštiny a indonéštiny. Oficiální stránky společnosti Twitter uvádí, že překladatelská komunita má přes dvě stě tisíc členů, což rozhodně není zanedbatelné číslo [44]. Podobný systém podporující komunitní překlad má také společnost Meta pro svou světoznámou sociální síť Facebook. Velmi silnou základnu překladatelů má také otevřená internetová encyklopedie Wikipedia.

Neblahým důsledkem zapojení neprofesionálů do procesu překladu je nižší kvalita výsledného produktu. Může docházet k nepřesným vyjádřením způsobených chybou nebo nedostatečnými znalostmi daného jazyka autorem překladu. Další možným nechtěným jevem je úmyslné překládání s chybami, chybná úprava již existujících přeložených textů a jiné podobné škodlivé aktivity. V rámci boje s podobnými událostmi zavádějí jednotlivé překladatelské platformy nejrůznější mechanismy sloužící ke kontrole kvality. Ty mohou mít podobu například v hlasování členů komunity, který z dostupných překladů je nejpřesnější, nebo je do překladatelského systému zavedena hierarchie uživatelů, kde členové s vyššími právy mohou schvalovat, který z překladů bude

použit. V krajních případech můžou také dočasně zablokovat škodící překladatele a znemožnit jim tím tak přístup k celému systému nebo alespoň k jeho části.

2.2 Přístup k formátu překladu

V úvodu zmiňovaná práce, na kterou tato bakalářská práce navazuje, *Analýza existujících mobilních průvodců po muzeích a památkách* [55] představuje v páté kapitole čtyři modely způsobu překladu. Jsou jimi modely:

- komunitní překlad s validací od jednoho správce nebo skupiny správců,
- Wiki přístup,
- komunitní překlad s validací od skupiny uživatelů a
- kategorizace uživatelů na překladatele a normální uživatele.

Z reakcí kulturních institucí vyplývá, že nejpřijatelnějším řešením by byl model se správcem překladů nebo validací od skupiny uživatelů. Naopak Wiki přístup by nebyl vhodný z důvodu možného výskytu vandalismu [55].

2.2.1 Komunitní překlad s validací od jednoho správce nebo skupiny správců

Tento model je postaven na aktivní účasti kulturních institucí. Každá instituce by měla jednoho nebo více správců, kteří by měli za úkol kontrolovat kvalitu a zda přeložené texty neobsahují závadný obsah. Prováděli by to schvalováním překladů a případně jejich korekcí [55].

2.2.2 Komunitní překlad s validací od skupiny uživatelů

Tento model staví na aktivitě komunity překladatelů a počítá s rozdělením uživatelů na překladatele a moderátory překladů, kde by každý moderátor byl zároveň překladatelem. Moderátoři by měli možnost hlasovat o nejlepším překladu a popřípadě nad nimi diskutovat. Tím by došlo k vybrání nejlepšího překladu, který by byl následně publikován [55].

2.2.3 Shrnutí

Výsledný model byl zvolen jako kombinace výše zmíněných modelů. Vychází z předpokladu, že každá instituce bude spravována jedním nebo více správci. Ti mají možnost vybírat přeložené texty, které budou publikovány. Tím je dosaženo toho, že instituce budou mít publikované texty pod svojí kontrolou a omezí se tím případný vandalismus. Dále v systému bude fungovat systém hlasování, kde bude moci každý překladatel hlasovat pro jeden nebo více překladů, které se mu zdají nejpřesněji přeložené. Tím dají správcům zpětnou vazbu a pomohou jim tak s výběrem textů k publikaci u jazyků, které správci sami neovládají.

3 Ucelená softwarová řešení podporující komunitní překlad

Na trhu se nachází softwarové produkty, které se snaží poskytnout softwarovou podporu překladatelům ochotným se aktivně zapojit do překladu textů vázaných na určitý projekt. Tématem projektu může být například počítačová hra, film, dokumentace zdrojového kódu, sociální síť nebo informační systém. Populární jsou zejména počítačové hry, protože mají často rozsáhlou komunitu věrných fanoušků žijících v různých částech světa.

Tyto softwarové produkty jsou placené nebo dostupné zdarma. Obvykle je alespoň omezená část funkcionality poskytnuta bez jakéhokoliv poplatku. Je však koncipovaná takovým způsobem, aby neumožňovala používat software pro komerční účely. Cílem takového přístupu je umožnit existenci neziskových projektů a zároveň firmám poskytnou možnost testovat software pro své potřeby před zakoupením produktu.

Mezi ty nejznámější ucelená řešení se řadí Crowdin, Pootle, Translation Toolkit, MediaWiki nebo český projekt Weblate [55]. Tato kapitola se bude zabývat rozborem možných výhod a nevýhod použití zmíněných produktů s vyvíjeným serverem pro komunitní překlad.

3.1 Crowdin

Prvním analyzovaným řešením je produkt od společnosti Crowdin. Tento lokalizační nástroj je postavený na technologii cloud a v rámci distribuce dodržuje přístup `software as a service` [6]. To znamená, že aplikace je hostována a spravována společností Crowdin. Přístup k ní je řešen pomocí internetu a data jí předávaná jsou uložena v databázích patřících Crowdinu. Platforma umožňuje vytvářet dva typy projektů. Jsou to veřejné a privátní projekty, které se liší zejména tím, jací uživatelé mají právo si zobrazit překládané texty [9]. Pro potřeby institucí by veřejné projekty nepředstavovaly problém. Z několika typů dostupných licencí, které se liší dostupnou funkcionalitou a počtem privátních projektů na uživatele [9], by za předpokladu používání veřejných projektů bylo možné vybrat licenci poskytovanou zdarma. To plně vyhovuje nekomerční povaze vytvářeného serveru.

Crowdin kromě klasického grafického rozhraní pro uživatele poskytuje REST API pro programové používání platformy. To obsahuje akce pro vyvolávání naprosté většiny funkcí dostupných z grafického rozhraní. Díky této možnosti lze některé akce zautomatizovat pomocí počítačového programu. Data v požadavcích a odpovědích jsou zapsána ve formátu JSON [7].

3.1.1 Výhody

Mezi výhody Crowdinu se řadí především fakt, že se jedná o známý a již delší dobu používaný software. Mezi klienty jsou celosvětově úspěšné společnosti jako například GitHub, GitLab, Raspberry Pi Foundation nebo Avast Software [6]. Tato fakta naznačují, že se jedná o otestovaný a důvěryhodný software, který bude na trhu figurovat i v budoucnosti.

Další výhodou je velmi dobře zdokumentované API obsahující ukázky požadavků i možné odpovědi serveru. Toto celé je zabalené do přehledného a uživatelsky přívětivého grafického rozhraní.

Crowdin poskytuje mnoho funkcionalit. Lze přidělovat různá práva přístupu uživatelů k určitému projektu, generovat souhrnné zprávy obsahující například žebříček nejaktivnějších členů, provádět automatický překlad nebo ho verzovat. Je možné také vytvářet požadavky na překlad a přiřadit ho uživatelům spolu s mezním termínem pro vyhotovení [8].

3.1.2 Nevýhody

Jako nevýhodu použití Crowdinu s vytvářeným serverem lze označit skutečnost, že by se data zasílala na servery třetí strany. V licenčních podmínkách Crowdinu totiž stojí, že překlady pro projekty vedené pod licencí poskytovanou zdarma jsou darovány bance překladů patřící Crowdinu [9]. Dalším úskalím cloudového řešení je to, že vytvářený server by po síti komunikoval s dalším serverem, což zpomaluje celou uživatelem vyvolanou akci.

Další problém se pojí s nekomerční povahou realizovaného projektu. Pokud by v budoucnu došlo k úpravám licenčních podmínek a případnému kompletnímu zrušení bezplatné licence, znamenalo by to nutnost přejít na jiný software podobného typu nebo doimplementovat potřebnou část programového kódu vlastním úsilím. Tento stav by pak byl velmi nepříjemný a mohl by také znamenat konec celého projektu serveru pro komunitní překlad.

V neposlední řadě API Crowdinu podporuje skoro celou dostupnou funkcionalitu. Z té by ale pro potřeby vytvářeného projektu nebylo využito zdaleka všechno a to by způsobovalo, že by část z atributů posílaných jako tělo požadavku nebyla vůbec použita. V rámci některých požadavků s velmi roz-

sáhlým tělem zprávy (například zakládání nového projektu) by tedy bylo nutné vypisovat mnoho nerelevantních atributů, což by zpomalovalo a znepřehledňovalo proces vývoje.

3.2 Pootle

Pootle je nástroj pro správu překladů napsaný v jazyce Python s využitím frameworku Django a sady nástrojů Translate Toolkit. Primárně se s ním pracuje přes integrované webové rozhraní [27].

3.2.1 Výhody

Největší výhodou Pootlu je bezesporu fakt, že je distribuovaný pod `obecnou veřejnou licencí GNU` [29]. Díky tomu lze tento kus softwaru libovolně zdarma používat, kopírovat, distribuovat a měnit [19]. Nehrozí zde tedy, že by došlo ke změně podmínek užití a za využívání softwaru by se muselo začít platit.

Dále Pootle nabízí možnost hostovat program na svém vlastním serveru. Tím máme na rozdíl od cloudových služeb plnou kontrolu nad překládanými texty.

Pootle disponuje celou řadou zajímavých funkcionalit, které by se daly využít. Je to zejména verzování překladů, podpora více projektů, podpora obrovského množství světových jazyků včetně minoritních a automatický překlad od Googlu nebo Apertia. Neméně zajímavou funkcionalitou je též uchovávání statistik o projektech, uživatelích, jazycích a překladech [28].

3.2.2 Nevýhody

Možnost vlastního hostování Pootlu se zároveň jeví jako nevýhoda, protože by pro správné fungování aplikace musely současně běžet dva servery. Byl by jím Pootle sloužící pro správu překladů a pak server, který by spravoval instituce a exponáty. Zkomplikovalo by to tudíž nasazení aplikace.

Vůbec největší nevýhoda je velmi slabá dokumentace k API. Hlavním cílem použití externího nástroje, jako je Pootle, je usnadnění vývoje aplikace díky užívání již existujícího kusu programu. Za takovýchto podmínek ale úsilí vynaložené na integraci do projektu převýší přidanou hodnotu získanou integrací.

3.3 Translate Toolkit

Translate Toolkit je sada nástrojů určená pro práci se soubory v lokalizačních formátech. Sada je napsána v jazyce Python a poskytuje API, pomocí kterého podporuje vývoj dalších lokalizačních nástrojů.

3.3.1 Výhody

Velkou výhodou spojenou se softwarem Translate Toolkit je jeho licence. Je totiž distribuovaný pod `obecnou veřejnou licencí GNU`. Z toho vyplývá, že jeho užití je kompletně zdarma [43].

Sada nástrojů obsahuje zajímavé funkce. Jsou jimi například kontroly textů validující velká písmena na začátku vět nebo mezery vyskytující se v textu navíc. K dispozici je také funkce počítající slova v textu [40].

3.3.2 Nevýhody

Mezi negativní vlastnosti se řadí zejména fakt, že nakládání s překlady a zdrojovými texty je realizováno s pomocí souborů ve standardních lokalizačních formátech založených obvykle na XML [41]. Počítačové programy, jako například hry, s těmito soubory pracují, a proto je snadné je použít jako zdrojový text. Primární zdroj textů určených pro lokalizaci má ale vytvářený server v jiném formátu. Tímto formátem jsou fotografie štítků nebo panelů vystavovaných exponátů.

Použití nástroje Translate Toolkit by dále přineslo omezení na programovací jazyk zvolený pro implementaci serveru. Knihovna je napsaná v jazyce Python, a proto by server musel být napsán také v Pythonu [42].

3.4 MediaWiki v kombinaci s rozšířením pro překlad

MediaWiki je velmi známý software napsaný v jazyce PHP určený pro sběr a správu znalostí za účelem sdílení mezi lidmi. Nejznámější projekt, který je na této technologii postavený, je Wikipedie - Otevřená encyklopedie. S pomocí rozšíření pro překlad podporuje práci s více jazyky [23, 26].

3.4.1 Výhody

Software je dostupný zdarma, protože je šířený pod `obecnou veřejnou licencí GNU` [24].

WikiMedia obsahuje přehledné webové rozhraní, na které jsou uživatelé zvyklí z mnoha jiných webových stránek. Z open-source povahy celého projektu také vyplývá, že server může hostovat každý podle své potřeby [23].

V neposlední řadě je software vybaven mnoha užitečnými funkcemi. Lze vybírat z několika možných způsobů grafického návrhu stránky. Dále pak lze automaticky generovat obsah stránky. Nechybí ani přehled autorů podílejících se na tvorbě stránky nebo historie změn [25].

3.4.2 Nevýhody

Na stránkách MediaWiki je uvedeno, že server je optimalizovaný na vysokou návštěvnost a nemusí být vhodný pro malé projekty, kterým realizovaný projekt bezpochyby je [26].

Velký problém je spojený se samotnou filozofií MediaWiki. Kulturní instituce dali po dotázání jasně najevo, že jim wiki přístup komunitního překladu nevyhovuje a to zejména z důvodu možného vandalismu a malé kontroly nad vznikajícími překlady [55].

Dále by při integraci s MediaWiki nastal problém se zobrazováním fotografie popisku exponátu určeného k přeložení. MediaWiki totiž primárně není software určený pro komunitní překlad, ale pro sdílení vědomostí.

3.5 Weblate

Posledním analyzovaným uceleným řešením je produkt Weblate. Jedná se o český software vytvořený Michalem Čihařem. Nástroj napsaný v jazyce Python má integrovaný verzovací systém a poskytuje uživatelské webové rozhraní. Program je postaven na frameworku Django [52].

3.5.1 Výhody

První nespornou výhodou Weblate je jeho licence. Software je jako mnohá další podobná řešení distribuovaný pod `obecnou veřejnou licencí GNU` [52]. Případné využití by tedy bylo zcela zdarma.

Z licenčních podmínek tedy vyplývá, že server si může hostovat na rozdíl od Crowdinu každý samostatně. Weblate dále nabízí možnost podpory vlastního hostování. Tato služba zahrnuje instalaci serveru a přednostní emailovou podporu, ale je placená [50].

Mezi dalšími benefity Weblate jsou vlastnosti jako nastavování přístupových práv, možnost vytvářet více větví překladu nebo RESTové API pro integraci s dalšími projekty [51].

3.5.2 Nevýhody

Při využití bezplatného přístupu založeného na vlastním hostování Weblate by byly třeba podobně jak v případě Pootlu dva servery. Jeden by byl tvořen softwarem Weblate a druhý by spravoval instituce a další data specifická zadání realizovaného projektu. To by komplikovalo proces nasazení.

Dále by podobně jako v případě Crowdinu byla použita jen část rozsáhlé funkcionality Weblate. Tím by se stalo mnoho vlastností objektů posílaných v rámci těl HTTP požadavků nevyužitých.

3.6 Shrnutí

V kapitole byla představena jednotlivá ucelená řešení problému komunitního překladu. Na internetu jich je k nalezení poměrně hodně, jsou většinou typu open-source a vystavují API umožňující software integrovat do vlastních řešení. Každé řešení bylo zanalyzováno, představeno a následně byly vyjmenovány jeho výhody a nevýhody. Po zvážení všech možností bylo přistoupeno k závěru, že žádný z výše uvedených softwarů nebude použit a to zejména z důvodu velmi specifického zdroje textu k přeložení ve formě fotografie, na který jednotlivá řešení nejsou dostatečně uzpůsobená. Samotná infrastruktura pro komunitní překlad tedy bude implementována v rámci bakalářské práce.

4 Existující API architektury

V této kapitole jsou rozebrány tři nejvýznamnější způsoby, kterými lze přistupovat ke tvorbě API. Jejich popisy spolu s výhodami a nevýhodami jsou seřazeny chronologicky podle data jejich vzniku. Ve shrnutí je pak následně uvedeno, jaká architektura byla vybrána a proč.

4.1 SOAP

Simple Object Access Protocol, zkráceně SOAP, je protokol, který byl navržen firmou Microsoft. Výměna dat tímto protokolem probíhá v textové podobě a to výhradně pomocí značkovacího jazyka XML. Z důvodu mnoha textových řetězců uvozujících přenášená data a dalších prvků jazyka XML jsou po síti přenášené požadavky a odpovědi robustní, což může mít vliv na plynulost provozu aplikací. SOAP primárně využívá protokol HTTP, ale lze ho používat i s jinými transportními protokoly. Jsou jimi například protokoly SMTP nebo FTP [56].

SOAP je využíván spolu s Web Services Description Language (WSDL), což je jazyk založený na XML, který slouží pro popis funkcionality webových služeb založených na SOAP. Definuje tedy formát vstupu a výstupu webové služby jako například, které elementy jsou povinné nebo které se mohou uvést ve větším počtu [57].

Výhodou SOAPu je fakt, že obsahuje podporu pro situace, kdy se vyskytne chyba při zpracování zprávy na straně příjemce. Pokud tedy dojde k chybě, odpověď obsahuje informační zprávu obsahující popis problému. Díky tomuto mechanismu mohou třetí strany, které používají vystavenou webovou službu, snadněji opravovat případné chyby a tím urychlit celý proces vývoje [56].

4.1.1 Struktura SOAP zprávy

SOAP zpráva je XML dokument, který obsahuje velké množství značek, které slouží pro popis struktury zprávy nebo pro popis přenášených dat. Mezi nejdůležitější značky patří následující čtyři:

- `<Envelope>` je kořenová značka obalující celou zprávu. Obsahuje povinně element `<Body>` a volitelně pak také element `<Header>`.

- Element `<Header>` není povinný a obsahuje aplikačně specifické informace (např. autentizaci).
- `<Body>` je povinný element. Obsahuje klientem vyžádaná data, která jsou klíčovou součástí zprávy.
- Element `<Body>` dále volitelně obsahuje element `<Fault>`, který je zodpovědný za přenášení informací o případných chybách [38].

4.2 REST

Representational State Transfer, zkráceně REST, je softwarový architektonický styl, který byl vytvořen, aby řídil vývoj architektury World Wide Webu. REST byl navržen Royem Fieldingem, který architekturu poprvé představil jako součást své disertační práce v roce 2000 [54]. V současnosti je REST velmi rozšířený a oblíbený.

4.2.1 Principy RESTu

REST je postaven na šesti základních principech:

- `Klient-Server`
  Návrhový vzor Klient-Server slouží k rozdělení odpovědnosti a to konkrétně na uživatelské rozhraní a ukládání dat. Tím se dosáhne zlepšení přenositelnosti uživatelského rozhraní a jednotlivé části nesoucí odpovědnost se mohou vyvíjet odděleně.
- `Bezstavovost`
  Princip bezstavovosti se týká komunikace Klient-Server a spočívá ve faktu, že požadavek na server nesmí spoléhat na to, že server drží informace o proběhlé komunikaci. Každý požadavek na server tedy musí nést všechny informace důležité pro to, aby server požadavku plně porozuměl a mohl ho vykonat.
- `Cache`
  Každý požadavek na server je označen za cacheovatelný nebo necacheovatelný. Pokud je požadavek cacheovatelný, server může využít odpověď uloženou v cache a tím zvýšit efektivnost komunikace.
- `Jednotné rozhraní`
  Jednotné rozhraní zjednodušuje a odděluje jednotlivé komponenty. Tím jim umožňuje, aby se samostatně vyvíjely.

- `Vrstevnatost`
  Princip vrstevnatosti spočívá v rozdělením architektury do hierarchických vrstev tak, že každá vrstva může vědět pouze o vrstvě, se kterou bezprostředně komunikuje.

- `Code-On-Demand`
  Code-On-Demand umožňuje, aby byla rozšířena funkcionalita klienta tím způsobem, že stáhne ze serveru kód ve formě appletů nebo skriptů a následně ho spustí [54].

4.2.2 Vlastnosti RESTu

REST využívá jako transportní protokol výhradně HTTP. Díky tomu zde může být použita HTTP cache a tím zvýšena efektivita. Formát pro výměnu dat není pevně stanovený. Nejčastěji se používá JSON, XML nebo CSV. Zprávy zaslané pomocí RESTu jsou samopopisné a obecně nejsou tak robustní jako při použití protokolu SOAP [57].

4.3 GraphQL

Graph Query Language, zkráceně GraphQL, je open-source dotazovací jazyk pro API vyvinutý společností Facebook a uveřejněný v roce 2015 především jako alternativa k RESTu [22]. GraphQL je tedy z trojice vybraných přístupů k architektuře API nejmladší a nejrychleji se rozvíjející.

4.3.1 Principy GraphQL

GraphQL je postaven na pěti základních principech [22]:

- `Zaměření na produkt`
  GraphQL vychází z požadavků tvůrců uživatelských rozhraní, kteří webovou službu pro svůj produkt používají.

- `Hierarchičnost`
  Tento princip spočívá v hierarchickém strukturování požadavku tak, aby byl v souladu s hierarchicky strukturovaným uživatelským rozhraním. Získaná odpověď je následně strukturovaná tím stejným způsobem.

- `Silný typový systém`
  Každá webová služba postavená na GraphQL definuje svůj typový sys-

tém, který pak musí být dodržovaný v rámci komunikace. Díky tomu lze snadno ověřit validitu zpráv.

- `Klientem specifikované odpovědi`
  V běžných webových službách určuje tvar odpovědi server. GraphQL k tomuto problému přistupuje z jiného pohledu a nechává klienta, aby definoval tvar zprávy, kterou má server vrátit. Tím je docíleno zvýšení efektivnosti komunikace, protože se nepřenáší žádná zbytečná data navíc.

- `Introspektivnost`
  Princip introspektivnosti spočívá v tom, že typový systém dané služby postavené na GraphQL lze dotazovat samotným jazykem GraphQL.

4.3.2 Vlastnosti GraphQL

GraphQL jako transportní protokol používá HTTP a jako formát pro výměnu dat JSON. API postavené na GraphQL nepoužívá verzování a v čase se vyvíjí. Cacheování není podporované a komunikace může trpět výkonnostními problémy, pokud požadavky na server obsahují komplexní dotazy využívající více databázových tabulek. V neposlední řadě GraphQL zavádí mnoho nových myšlenek, a proto se musí programátoři před jeho použitím dostatečně informovat [21].

4.4 Shrnutí

V kapitole byly představeny tři hlavní API architektury. Po několik let dominovaly technologie REST a SOAP, ke kterým se následně přidala novinka GraphQL. Ta přistupuje k problematice z nového úhlu a zvyšuje efektivitu komunikace posíláním pouze skutečně potřebných dat.

Ze všech možností byla pro implementaci serveru pro komunitní překlad určena technologie REST. Bylo to zejména z důvodů snadné a přímočaré implementace, která umožňuje přenášet data ve formátu JSON, což je nativní formát pro JavaScriptem řízeného webového klienta.

5 Analýza dostupných technologií

Tato kapitola se zabývá popisem výsledků získaných provedením analýzy dostupných technologií pro tvorbu RESTových serverů a moderních webových rozhraní.

5.1 Technologie pro implementaci RESTového serveru

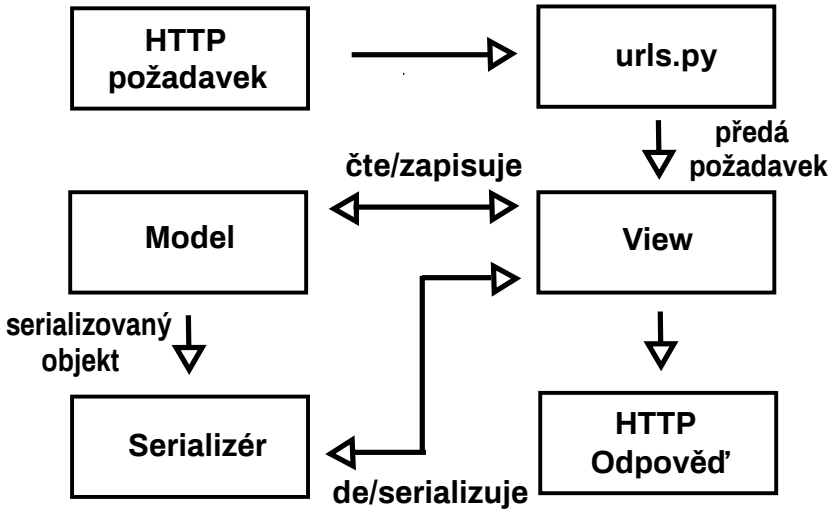
Následující text se zaměřuje na analýzu serverových frameworků (podpůrných softwarových řešení), pomocí kterých lze vytvořit RESTový server. Softwarová architektura REST nevyžaduje pro své správné fungování žádný konkrétní programovací jazyk, takže výběr dostupných frameworků je značně rozsáhlý. V dnešní době naprostá většina vysokoúrovňových jazyků poskytuje alespoň jeden nebo více frameworků, bude tudíž popsán pouze průřez z těch klíčových a největších. Závěrem bude zmíněno, který framework byl zvolen jako základ implementovaného serveru pro komunitní překlad informačních popisků.

5.1.1 Django

Framework Django je vytvořený v programovacím jazyce Python [12]. Python je v současné době velmi používaný vysokoúrovňový jazyk s jednoduchou syntaxí a dynamickým typovým systémem. Framework je díky své open-source povaze zcela zdarma k použití. Byl vytvářen s cílem umožnit programátorům rychlý vývoj a nechat je soustředit se především na logiku konkrétní aplikace poskytnutím softwarové podpory pro často se opakující úkoly v rámci webového vývoje [14].

Django obsahuje podporu pro autentizaci a dále pomáhá programátorům vyvarovat se základním bezpečnostním chybám. Těmi jsou například SQL injekce, cross-site scripting (XSS) nebo cross-site request forgery (CSRF) [14]. Framework podporuje návrhový vzor *Model-View-Template (MVT)*, který se velmi podobá na webu populárnímu vzoru *Model-View-Controller (MVC)*. V rámci vytváření RESTových služeb není využívána HTML šablona, ale pracuje se se strukturami ve formátu JSON. Proto je také význam šablon

upozaděn a naopak se využije serializér provádějící konverzi mezi objekty v Pythonu a JSON strukturami. Diagram zobrazující používanou architekturu lze vidět na obrázku 5.1.



Obrázek 5.1: Architektura využívaná v frameworku Django

Možným úskalím ve využití tohoto frameworku se jeví samotná filosofie jazyka Python umožňující použití více různých paradigmat, kterými jsou například funkcionální a objektově orientované paradigma. Spolu s poměrně benevolentní syntaxí nevyžadující například uzavírání bloků kódu do složených závorek, jako je to časté ve valné většině jazyků, by výsledný kód mohl být nepřehledný.

Django framework je podporován nezávislou nadací Django Software Foundation [11] a mezi nejznámější projekty využívající Django se řadí například Instagram [13] nebo Bitbucket [10].

5.1.2 Express.js

Express.js je framework vytvořený pro JavaScriptové běhové prostředí Node.js a spravovaný společností OpenJS Foundation [15]. S frameworkem lze použít JavaScript nebo nadstavbu JavaScriptu jménem TypeScript, která byla vytvořena společností Microsoft a přináší zejména statické typování do jinak dynamicky typovaného JavaScriptu [45]. Podobně jako Django je Express.js také open-source projekt publikovaný pod MIT licencí [18]. Příklad způsobu jeho použití lze vidět v ukázce kódu 5.2.

```
// ukazka namapovani HTTP pozadavku typu GET s prislusnou URL na
    obsluzny kod
app.get('/', (req, res) => {
  res.send('GET method')
})

// ukazka namapovani HTTP pozadavku typu POST s prislusnou URL na
    obsluzny kod
app.post('/', (req, res) => {
  res.send('POST method')
})
```

Kód 5.2: Ukázka použití frameworku Express.js

Express.js je součástí sad vývojových technologií označovaných zkratkami *MERN* a *MEAN* sloužícími pro tvorbu moderních webových aplikací. Písmena zkratek jsou prvními písmeny softwarových řešení a tyto dvě sady se liší pouze ve vybraném frameworku pro tvorbu uživatelských rozhraní (React nebo Angular). Význam zkratek je tedy následující:

- *M* - MongoDB,
- *E* - Express.js,
- *R* - React nebo *A* - Angular,
- *N* - Node.js.

Tvůrci frameworku ho označují za minimalistický. To znamená, že sám o sobě poskytuje pouze minimální funkcionalitu, ale všechny další významné části můžou být přidány v podobě kompatibilních doplňků. Například pro šablonovací systém je na výběr mezi 14 různými řešeními [17].

Nevýhoda spojená s využitím tohoto přístupu spočívá v obecně vyšší rychlosti vývoje JavaScriptových frameworků. V JavaSriptovém prostředí dochází k častějším změnám verzí, které se oproti ostatním jazykům vyznačují nižší zpětnou kompatibilitou.

Mezi světově známé společnosti používající Express.js se řadí například IBM, Uber nebo PayPal [16].

5.1.3 Spring Boot

Spring Boot je nadstavba frameworku Spring, který je populární zejména v prostředí velkých nadnárodních korporací. Pro svoji funkčnost využívá

programovací jazyk Java, který umožňuje psaní velmi přehledného objektově orientovaného kódu. Podobně jako ostatní výše jmenované frameworky je Spring Boot také open-source publikovaný pod *Apache 2.0 licencí* [4].

Hlavní výhodou frameworku Spring Boot oproti použití samotného Springu je, že obsahuje integrovaný webový server. Integrovat lze servery Tomcat, Jetty nebo Undertow a tím se zbavit nutnosti nasazování aplikace pomocí WAR souborů. Dále jsou dostupné takzvané startovací závislosti usnadňující konfiguraci sestavení projektu. V neposlední řadě pak také není potřeba vytvářet konfigurační XML soubory. Konfigurace je řešena pomocí anotací, což jsou speciální prvky jazyka Java a v rámci frameworku jsou velmi silně používány. Mimo to lze konfiguraci provádět také programovým kódem [5]. Příklad způsobu použití frameworku lze vidět v ukázce kódu 5.3.

```
@RestController
public class ExampleController {
    // ukazka namapovani HTTP pozadavku typu GET s prislusnou URL
        na obsluzny kod
   @GetMapping("/")
   public String getExample() {
      return "GET method";
   }
}
```

Kód 5.3: Ukázka použití frameworku Spring Boot

Obrovskou výhodou je podpora vkládání závislostí (dependency injection), která také tvoří základní pilíř frameworku. Vkládání provádí Spring kontejner, který dovede automaticky vložit objekt do jiných objektů. Tím je dosaženo volného propojení komponent, které vede k lepší udržitelnosti psaného programu [3].

Spring framework je spravovaný společností VMWare a mezi společnosti, které ho využívají se řadí taková jména jako Amazon, Google nebo Microsoft [39].

5.1.4 Rails

Framework Rails (někdy též Ruby on Rails) je postavený na programovacím jazyce Ruby. Ruby je plně objektový dynamicky typovaný programovací jazyk, který byl původně vytvořen jako alternativa k jazykům Perl a Python [37]. Vyznačuje se jednoduchou syntaxí, kterou se lze snadno naučit. I tento

framework je stejně jako všechny uvedené volně k použití díky své licenci typu *MIT* [30]. Příklad jeho použití lze vidět v ukázce kódu 5.4.

Při vytváření programu v Rails je dodržován návrhový vzor *Model-View-Controller (MVC)*. Tento vzor dělí aplikaci do tří vrstev, kde každá má svůj specifický úkol.

- Vrstva *Modelů* reprezentuje doménový model aplikace a logiku s ním spojenou. Jednotlivé modely mohou být přímo mapovány na databázové tabulky nebo jsou jejich zdrojem Ruby třídy.
- Vrstva *Pohledů (View)* poskytuje reprezentace zdrojů. Může nabývat více formátů, ale nejobvyklejší je HTML (nebo JSON při tvorbě RESTového serveru). Tyto reprezentace jsou generovány jako odpověď kontroléru na HTTP požadavek.
- *Kontroléry* jsou zodpovědné za zpracování příchozího HTTP požadavku. Manipulují s modely a připravují pohledy za účelem poskytnutí patřičné odpovědi [32].

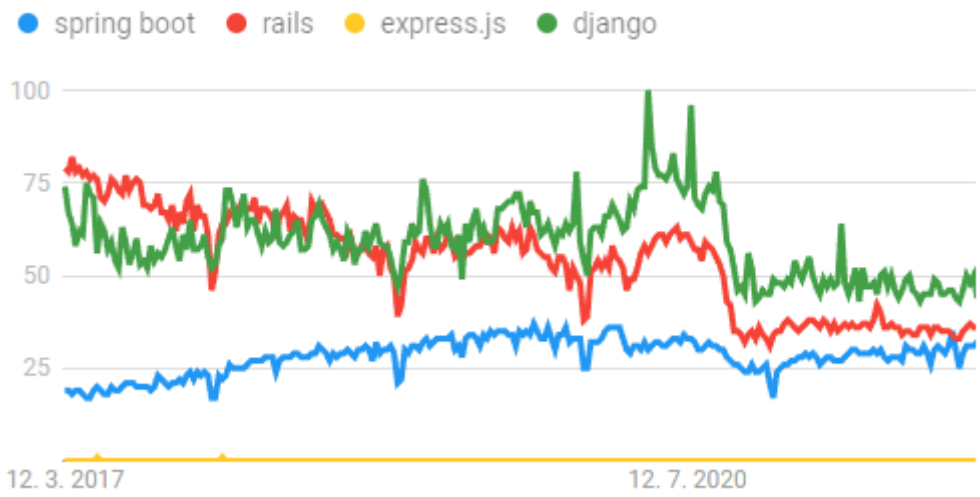
```
// ukazka kodu kontroleru
class ArticlesController < ApplicationController
  def index
    @articles = Article.recent
  end

  def show
    @article = Article.find(params[:id])
    fresh_when etag: @article
  end

  def create
    article = Article.create!(article_params)
    redirect_to article
  end

  private
    def article_params
      params.require(:article).permit(:title, :content)
    end
end
```

Kód 5.4: Ukázka použití frameworku Rails [31]

Správu frameworku, jeho udržování a rozšiřování má na starosti komunita složená z dobrovolníků. Mezi nejznámější společnosti využívající tento software se řadí GitHub, Twitch nebo směnárna kryptoměn Coinbase [31].

5.1.5 Srovnání popularity

Popularitu frameworků lze jen velmi těžce měřit, ale dobrou představu nám udává nástroj Google Trends, který provádí analýzu frekvence vyhledávání určitých výrazů v internetovém vyhledávači Google. Získaná data tedy neudávají skutečné rozšíření frameworků, ale jejich oblíbenost ve vyhledávání. Do systému bylo zadáno období posledních pěti let a jako region byl zvolen celý svět. Získaný graf lze vidět na obrázku 5.5.



Obrázek 5.5: Popularita frameworků podle Google Trends [20]

Z grafu lze vidět, že nejpopulárnější je Django, poté následuje Rails a na třetí pozici je Spring Boot. Mimo to se také ukazuje, že popularita byla v minulosti více rozptýlená a v posledních letech dochází k jejímu postupnému vyrovnávání.

Z provedené analýzy bohužel úplně vypadl framework Express.js. S největší pravděpodobností tento stav není způsobený nezájmem programátorů o tento framework, ale pouhým faktem, že uživatelé při vyhledávání nezadávají celý výraz *express.js* ale jen *express*. Provedení analýzy tak, že pro tento framework bude použit výraz *express*, bohužel situaci neřeší. Problém spočívá v přílišné obecnosti tohoto slova, které používá i mnoho uživatelů nevyhledávajících informace týkající se softwaru.

5.1.6 Shrnutí

Všechny uvedené frameworky jsou v podstatě vyrovnané. V každém z nich bylo vytvořeno dostatečné množství projektů a jsou využívány napříč programátorskou komunitou. Tato popularita značí, že frameworky jsou funkční, vyzkoušené a otestované. Vytvářený server pro komunitní překlad navíc nemá žádné speciální nároky. Vzhledem k předpokládané návštěvnosti jsou případné odchylky v rychlosti jednotlivých frameworků zanedbatelné, protože rychlost bude jednoznačně dostatečná.

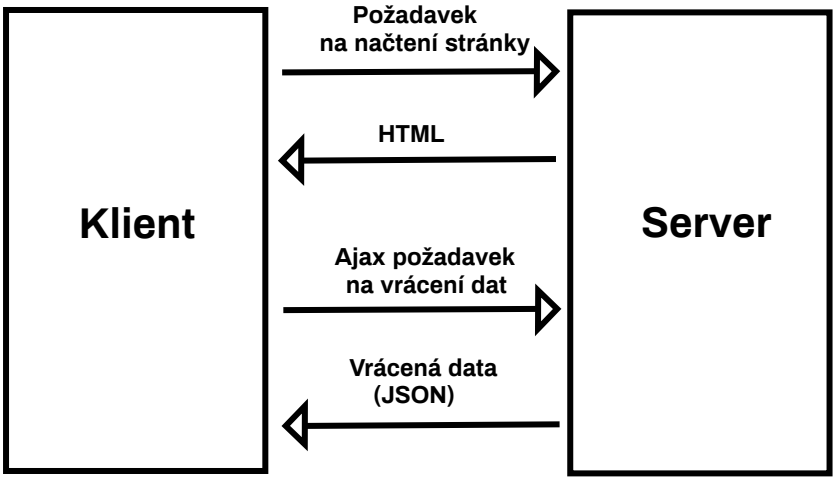
Jeden z hlavních rozdílů v použití jednotlivých frameworků pramení z programovacích jazyků, na kterých jsou postavené. Jazyky Java a TypeScript jsou staticky typované, což může vést k větší přehlednosti psaného kódu a tím usnadnit možný budoucí rozvoj projektu a spolupráci více programátorů.

Vzhledem k výborné dokumentaci, aktivitě rozsáhlé komunity, integrovanému webovému serveru a možnosti psaní přehledného do tříd strukturovaného kódu byl vybrán framework Spring Boot. V zásadě by ale každá z dostupných možností vedla ke zdařilému a použitelnému výsledku.

5.2 Technologie pro implementaci webového rozhraní

V této kapitole jsou rozebrány možné nástroje pro tvorbu webového rozhraní, které bude čerpat data z RESTového serveru. Rozhraní bude sloužit správcům kulturních institucí k snadné editaci jednotlivých exponátů a především bude umožňovat komunitě překladatelů vytvářet překlady jednotlivých informačních popisků. Jeho implementace bude provedena za využití populárního *jednostránkového (single-page application - SPA)* přístupu.

Jednostránkové aplikace představují moderní přístup k vytváření softwaru. Při prvotním načtení stránky dojde ke stažení veškerých dat definujících strukturu rozhraní. Při pohybu na stránce dochází k dynamickému načítání dat ze serveru za pomocí technologie Ajax. Tento způsob, který je zobrazený na obrázku 5.6, je odlišný oproti klasickému přístupu, kdy dochází k načítání celých stránek i s definicí struktury. Při používání SPA nikdy nedochází k obnovení stránky. Tento přístup má velmi rychlé odezvy, které zvyšují celkový uživatelský komfort. Nejznámější aplikace využívající jednostránkový přístup jsou Gmail, Facebook, YouTube nebo Twitter.



Obrázek 5.6: Životní cyklus jednostránkové aplikace

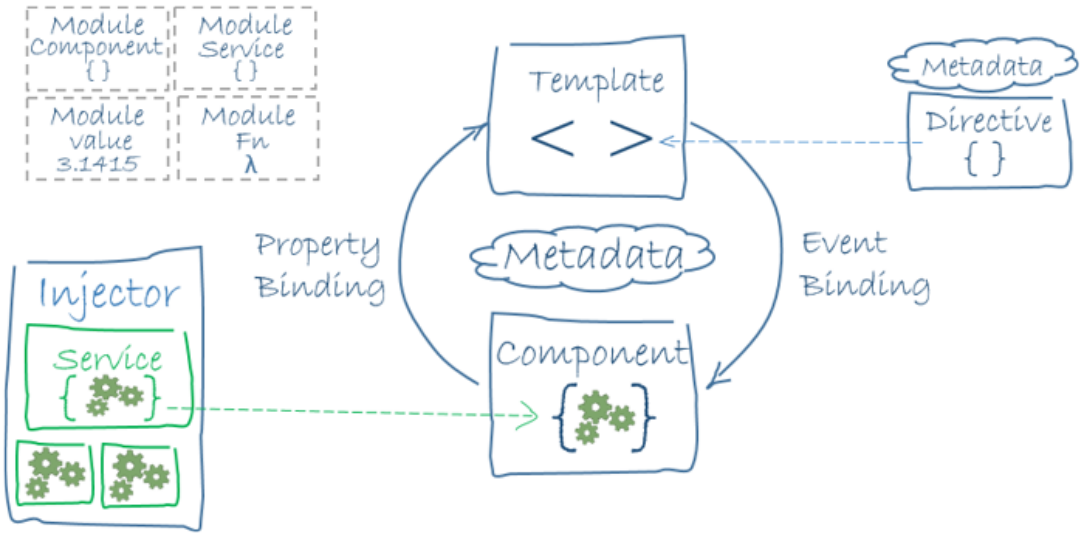
Podobně jako u RESTových frameworků je na výběr mezi velkým množstvím technologií. Proto byly byly k analýze vybrány tři zástupci, kteří se řadí k těm nejpoužívanějším. Dva z nich jsou frameworky a jeden lze označit jako knihovnu.

5.2.1 Angular

Framework Angular je postavený na jazyce TypeScript. Jeho správu provádí společnost Google a komunita složená z přispěvatelů [53]. Projekt je publikovaný pod *MIT licencí*, takže jeho použití je zcela zdarma [2].

Angular je robustní framework založený na komponentách, které jsou představovány programovými třídami. Jeho architekturu lze vidět na obrázku 5.7. Zmíněná robustnost spočívá v tom, že obsahuje podporu pro velké množství různých situací, kterým by jeho uživatel mohl čelit. Tato podpora je tedy dostupná bez přidávání dalších závislostí na externí knihovny a mezi zajímavé funkce se řadí například směrování, správa formulářů nebo řešení pro komunikaci mezi klientem a serverem [53].

Výhodou tohoto frameworku je podpora automatického vkládání závislostí. Té je dosaženo pomocí příkazu *@Injectable* nad definicí třídy, která bude sloužit jako vzor pro vkládání, a následného přidání parametru typu vkládaného objektu do konstruktoru třídy, která automatické vložení požaduje. Díky této funkci je kód flexibilnější a snadněji testovatelný [53].



Obrázek 5.7: Architektura frameworku Angular [1]

5.2.2 React

React (nebo také ReactJS/React.js) je JavaScriptová knihovna pro tvorbu uživatelských rozhraní. Obdobně jako u frameworku Angular, vývoj této knihovny má na svědomí také velká IT společnost a to tentokrát Meta. Na vývoji se ale podílela také komunita nezávislých vývojářů [34]. Projekt je open-source povahy a je publikovaný pod *MIT* licencí [36].

Architektura Reactu je postavena na znovupoužitelných komponentách. Tyto komponenty lze definovat pomocí tříd nebo funkcí. V rámci flexibility lze každé vykreslované komponentě předat parametry, díky kterým je dosaženo variabilní funkcionality, vzhledu nebo zobrazovaných dat [34]. Příklad definice komponenty spolu s jejím vykreslením je vidět v ukázce kódu 5.8.

Při tvorbě rozhraní s použitím Reactu je doporučeno používat *JSX*. JSX je rozšíření jazyka JavaScript, díky kterému lze přehledněji zapisovat strukturu vykreslování komponent. Svým vzhledem připomíná HTML. Rozšíření je pouze volitelné, protože vše, co lze zapsat v JSX, lze zapsat i v čistém JavaScriptu [35].

Aplikace používající React jsou velmi výkonné. Je to způsobeno zejména použitím virtuálního DOMu. Ten se mění a jeho obsah se porovnává se skutečným DOMem. Ke změnám na obrazovce dochází pouze tehdy, když je to na základě výsledku porovnání potřeba a nepřekresluje se tím pádem pokaždé celá obrazovka [33].

```
// ukazka kodu, ktery vykresluje jednoduchou komponentu
class ExampleComponent extends React.Component {
  render() {
    return (
      <div>
        Example component text
      </div>
    );
  }
}

ReactDOM.render(
  <ExampleComponent />,
  document.getElementById('example')
);
```

Kód 5.8: Ukázka použití knihovny React

5.2.3 Vue

Vue (nebo také VueJS/Vue.js) je JavaScriptový framework pro tvorbu webových uživatelských rozhraní [48]. Založil ho v roce 2014 jako osobní vedlejší projekt Evan You a nyní ho společně s komunitou spravuje. Framework nijak nevynucuje volbu mezi JavaScriptem a Typescriptem [46]. Rozhodnutí je ponecháno na zvážení programátorovi. Projekt je obdobně jako ostatní publikován pod MIT licencí, takže jeho použití je zdarma [49]. Mezi velké společnosti využívající Vue se řadí například NASA, Apple, Google nebo Microsoft [46].

Framework využívá standardního HTML, CSS a JavaScriptu. Jeho architektura je postavena na deklarativním přístupu založeném na komponentách. Deklarativní vykreslování spočívá v rozšíření HTML o šablonovací syntaxi, která umožňuje popsat výstup v závislosti na stavu definovaném v JavaScriptu. Vue automaticky detekuje změnu stavu a následně efektivně aktualizuje DOM [47].

Vue se označuje jako *progresivní* framework. Vyznačuje se to tím, že ho lze postupně adaptovat do projektu. Lze začít s jedním skriptem v HTML kódu a následně ho rozšířit do celého projektu bez nutnosti přepisování kódu od základů [47]. Příklad použití frameworku lze vidět v ukázce kódu 5.9.

```
// js kod s vyuzitim funkci poskytovanych Vue
import { createApp } from 'vue'

createApp({
  data() {
    return {
      count: 0
    }
  }
}).mount('#app')

// sablona
<div id="app">
  <button @click="count++">
    Count is: {{ count }}
  </button>
</div>
```

Kód 5.9: Ukázka použití frameworku Vue [47]

5.2.4 Srovnání popularity

Jako v případě analýzy serverových frameworků byl pro srovnání popularity vybrán nástroj Google Trends. Do systému bylo opět zadáno období posledních pěti let a jako region byl zvolen celý svět. Získaný graf je k dispozici na obrázk 5.10.

Obrázek 5.10: Popularita SPA technologií podle Google Trends [20]

Z grafu lze vidět, že zprvu dominovalo Vue, ale následně se na první příčku dostal React. V poslední době se popularita obou rivalů vyrovnává. Angular zaujímá významnou část z celého období na posledním místě. Jeho popularita však není nijak extrémně nižší oproti zbylým dvěma porovnávaným.

5.2.5 Shrnutí

Situace je velmi obdobná jako u analýzy frameworků pro tvorbu serveru. Všechny technologie mají silnou komunitu a jsou velmi dobře použitelné pro tvorbu jednostránkového webového rozhraní pro překladatele.

Nakonec byla hlavně vzhledem k faktu, že je přívětivá pro nové uživatele a lze se ji poměrně rychle naučit, zvolena knihovna React. Mezi její další klady, které vedly k tomuto rozhodnutí, se řadí její vysoká výkonnost a možnost tvorby znovupoužitelných komponent, které zpřehledňují kód aplikace.

6 Návrh systému

Tato kapitola se zabývá popisem návrhu systému pro komunitní překlad. Bude zde rozebrán databázový model a jednotlivé role, které mohou uživatelé zastávat. Následně budou popsány všechny koncové body, pomocí kterých lze se serverem komunikovat.

6.1 Uživatelské role

Uživatel, který se se systémem setká poprvé, nemá ještě vytvořený svůj účet. Tuto situaci pokrývá role *nepřihlášeného uživatele*, která umožňuje registraci do systému. Po zaregistrování si každý uživatel může vytvořit kulturní instituci a stát se tak *správcem instituce*. Mimo to dostane při vytvoření účtu také práva *překladatele*, díky kterým se může stát aktivním účastníkem komunitního překladu. Nejvyšší autoritou systému je pak *administrátor*, kterému náleží právo na správu samotných uživatelů. Poslední rolí je pak *návštěvník kulturní instituce*, který si chce přečíst přeložený informační popisek a pro kterého je celý systém vytvářený.

6.1.1 Nepřihlášený uživatel

Nepřihlášený uživatel je ve svých akcích velmi limitovaný. V rámci webového rozhraní může vidět pouze úvodní stránku poskytující základní informace o účelu systému, informační stránku, přihlašovací formulář a registrační formulář. Pro registraci musí jako povinné atributy vyplnit své jméno, e-mailovou adresu a heslo.

6.1.2 Správce kulturní instituce

Každý uživatel se může stát správcem jedné kulturní instituce. Taková situace může nastat dvěma způsoby. Ten první nastane, pokud ho již existující správce instituce přizve, aby se také podílel na správě. Může to udělat z webového rozhraní pomocí speciální funkce, která vytvoří nový účet správce a jeho údaje zašle na zadanou e-mailovou adresu. Druhou možností je vytvořit si svou vlastní instituci, protože k tomu má právo každý registrovaný uživatel.

Správce instituce může libovolně přidávat povolené jazyky pro překlad. Dále pak přidává exponáty, jejichž klíčovou součástí je fotografie informač-

ního popisku. Samozřejmě má právo také na jejich zobrazování, editaci a mazání. Pokud je na to jeho instituce přizpůsobená, může také stahovat obrázky s QR kódy pro jednotlivé exponáty a následně je umístit v rámci expozice. Systém mu také umožňuje celou svou kulturní instituci smazat i se všemi souvisejícími daty.

Jako správce má možnosti ovlivnit, jaké překlady se budou zobrazovat uživatelům mobilní aplikace. Provádí to schvalováním překladu, který bude zobrazován pro daný jazyk a exponát. Pokud jazyk neovládá, může se řídit počítadlem palců nahoru, které dávají ostatní překladatelé, pokud se jim překlad líbí.

6.1.3 Překladatel

Primárním úkolem každého překladatele je vybrat si jazyk a kulturní instituci, která je mu nejbližší. Poté si může zobrazit všechny její exponáty a začít vytvářet překlady. Předtím si ale může zobrazit všechny již dostupné překlady a pokud najde takový, který se mu dostatečně líbí, může mu dát palec nahoru.

Všechny své překlady si může zobrazit. K dispozici má i jednoduchý verzovací systém, který mu umožňuje vracet se k předchozím verzím nebo vytvářet nové. Disponuje také právem celou historii verzí překladu smazat.

6.1.4 Administrátor

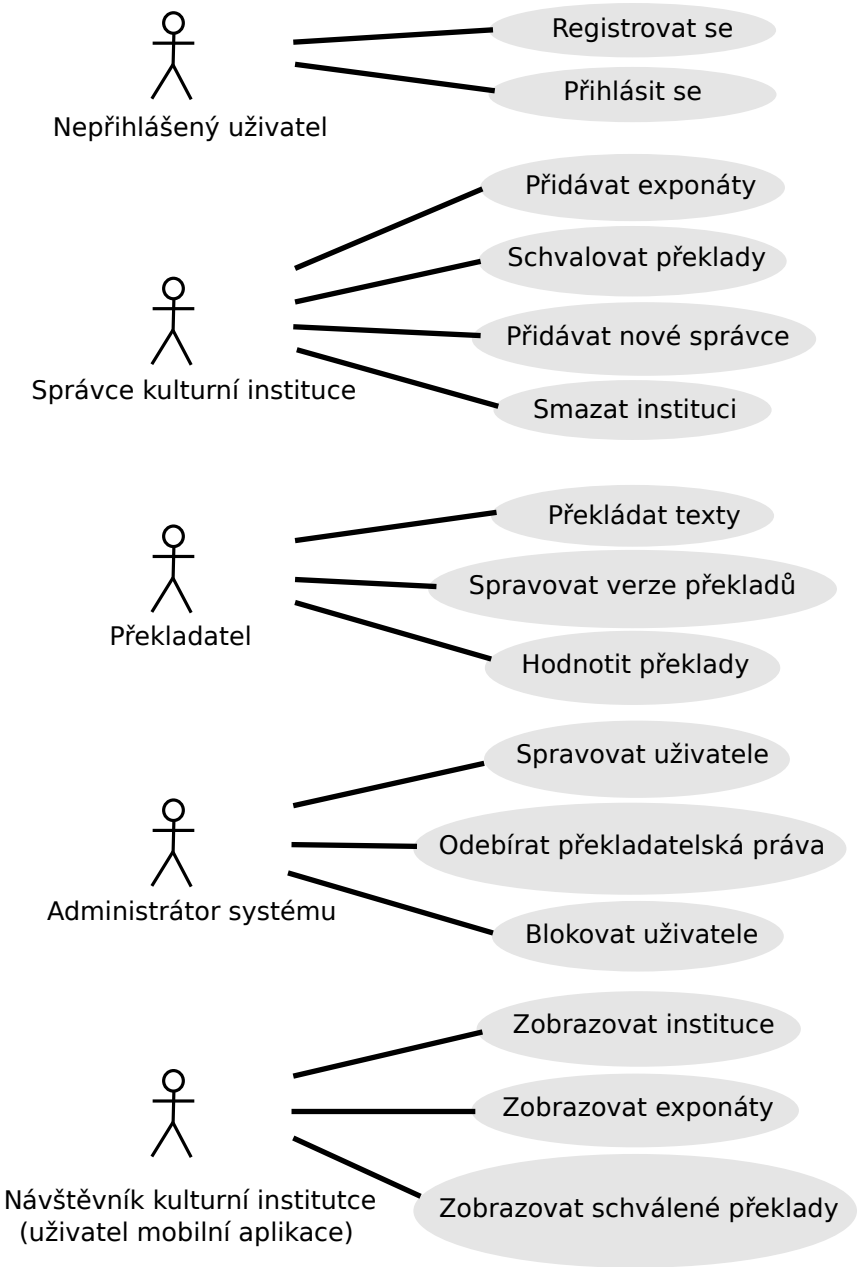
Administrátor systému má na starosti správu uživatelů. Může si je nechat všechny zobrazit a dále má možnost vidět jejich bližší informace. Má právo měnit uživatelům jména a hesla. To zejména proto, aby bylo možné řešit problém ztráty přihlašovacích údajů. Heslo nemění v rámci bezpečnosti administrátor přímo, ale dochází k zaslání nového náhodného hesla na e-mail uživatele. Dále má možnost zamezit nebo obnovit uživatelova práva na překlad a pokud je uživatel správcem instituce, může mu být odebrána. V krajním případě pak může být uživateli úplně zakázán přístup do aplikace.

6.1.5 Návštěvník kulturní instituce

Návštěvník kulturní instituce využívá pro zobrazování překladů mobilní aplikaci, která komunikuje se serverem. Proto je také třeba ho brát v úvahu. Návštěvník si může nechat zobrazit veškeré dostupné kulturní instituce a následně i jejich exponáty. Na základě jím definovaného jazyka si pak může prohlížet i správcem schválené překlady. Nechybí mu ani možnost přidávat nové exponáty.

6.2 Případy užití

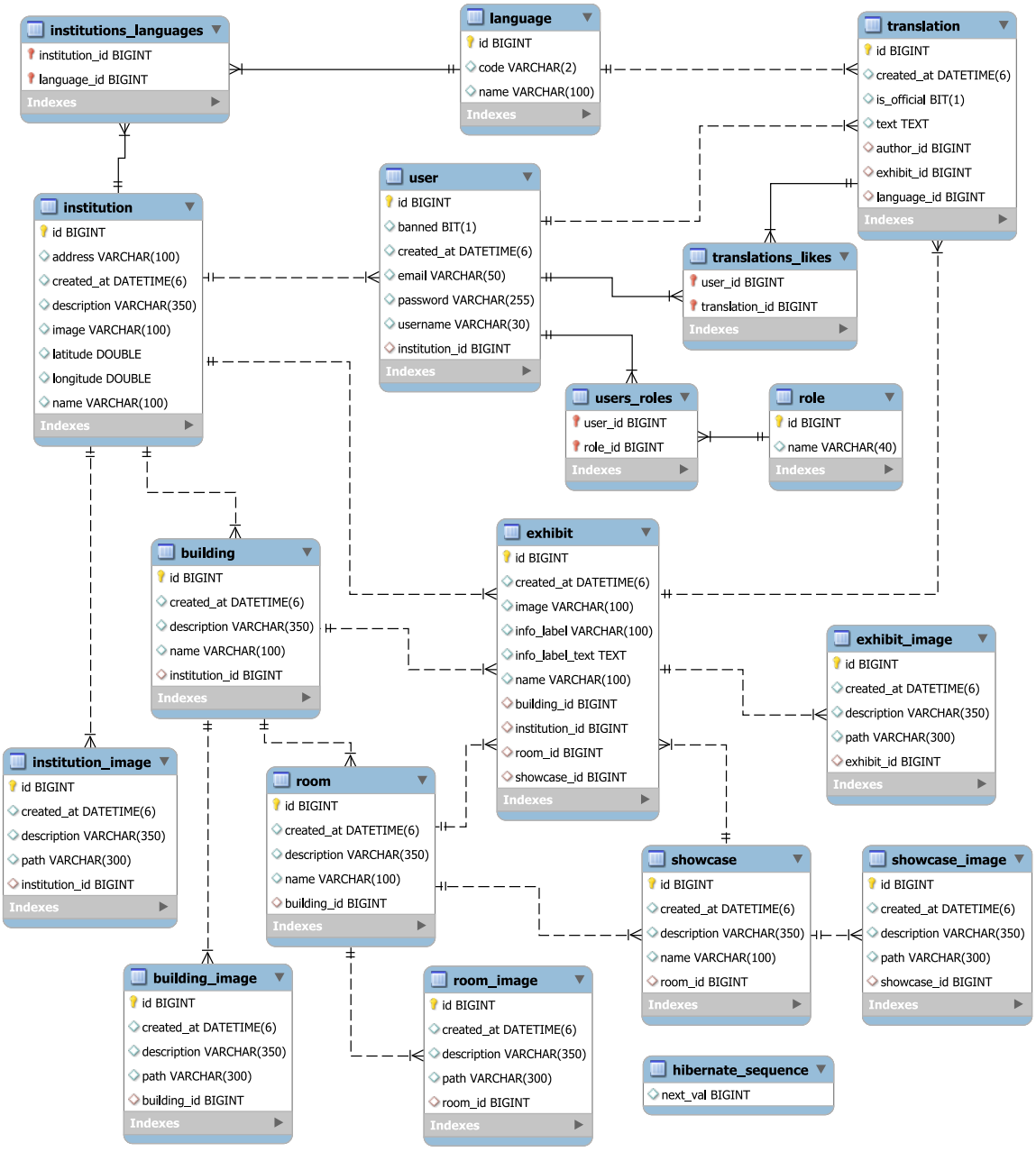
Následující diagram případů užití (6.1) zobrazuje funkcionalitu systému pro komunitní překlad spjatou s jednotlivými rolemi uživatelů. Jsou zde uvedeny nejdůležitější vykonatelné akce za účelem ohraničit projekt a přiblížit jeho rozsah.



Obrázek 6.1: Diagram případů užití

6.3 Databázový model

Na obrázku 6.2 lze vidět diagram vztahů jednotlivých entit navrženého databázového modelu. V této podkapitole budou popsány klíčové tabulky a jejich atributy. Všechny primární klíče tabulek jsou celočíselné a generované pomocí databáze.



Obrázek 6.2: Databázový model

6.3.1 Tabulka user

Tato tabulka reprezentuje uživatele systému. Slouží především k ověřování totožnosti při přihlašování a ověření práv k vykonání akce na základě role nebo autorství. Obsahuje následující atributy.

- Atribut `id` je primárním klíčem uživatele.
- Atribut `banned` je příznak, zda došlo k zablokování uživatele.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k registraci uživatele.
- Atribut `email` uchovává e-mailovou adresu.
- Atribut `password` obsahuje výstup hashovací funkce s parametrem hesla, které uživatel zadal při registraci.
- Atribut `username` uchovává uživatelské jméno.
- Atribut `institution_id` je cizí klíč pojící uživatele se správou dané instituce.

6.3.2 Tabulka role

Tabulka reprezentuje roli, která může náležet uživatelům systému. Role jsou zavedeny, aby bylo možné vytvořit akce, které můžou provádět pouze někteří uživatelé. Jedná se například o roli administrátora, který spravuje uživatele.

- Atribut `id` je primárním klíčem role.
- Atribut `name` obsahuje název role.

6.3.3 Tabulka institution

Účelem této tabulky je nadefinovat reprezentaci kulturní instituce. Ty jsou zakládány uživateli, kteří se tím stávají jejich správci.

- Atribut `id` je primárním klíčem instituce.
- Atribut `address` uchovává adresu, kde instituce sídlí.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k její registraci.
- Atribut `description` slouží pro uchování krátkého popisu kulturní instituce.

- Atribut `image` obsahuje název uložené fotografie zobrazující instituci.
- Atribut `latitude` zaznamenává zeměpisnou šířku umístění instituce.
- Atribut `longitude` zaznamenává zeměpisnou délku umístění instituce.
- Atribut `name` obsahuje celý název instituce.

6.3.4 Tabulka exhibit

Tabulka představuje reprezentaci exponátu nacházejícího se v kulturní instituci. Exponáty jsou přidávány správci a návštěvníky institucí.

- Atribut `id` je primárním klíčem exponátu.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k jeho vytvoření.
- Atribut `image` obsahuje název uložené fotografie zobrazující exponát.
- Atribut `info_label` uchovává název uložené fotografie s informačním popiskem.
- Atribut `info_label_text` slouží pro uchování přepsaného textu informačního popisku.
- Atribut `name` je název exponátu.
- Atribut `building_id` obsahuje cizí klíč určující budovu, kde se exponát nachází.
- Atribut `institution_id` je cizí klíč pojící exponát s institucí, která ho vlastní.
- Atribut `room_id` je cizím klíčem určujícím místnost s exponátem.
- Atribut `showcase_id` je také cizí klíč označující vitrínu s exponátem.

6.3.5 Tabulka translation

Toto je jedna z nejklíčovějších tabulek, protože obsahuje reprezentaci překladu informačního popisku. Ty jsou psány komunitou překladatelů a následně jsou po schválení správcem instituce zobrazovány v mobilní aplikaci.

- Atribut `id` je primárním klíčem překladu.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k jeho vytvoření.

- Atribut `is_official` je příznak značící, zda byl překlad správcem instituce označen pro zobrazování v mobilní aplikaci.
- Atribut `text` obsahuje text vytvořeného překladu.
- Atribut `author_id` je cizí klíč pojící překlad s jeho autorem.
- Atribut `exhibit_id` je také cizí klíč, který překlad váže na exponát.
- Atribut `language_id` je cizí klíč určující, ve kterém jazyku je překlad napsán.

6.3.6 Tabulka language

Tabulka reprezentuje jazyk, který můžou správci přidávat ke své instituci. Překladatelé se poté řídí dostupnými jazyky z výběru.

- Atribut `id` je primárním klíčem jazyku.
- Atribut `code` obsahuje dvoumístný standardní kód jazyka.
- Atribut `name` uchovává anglický název jazyka.

6.3.7 Tabulka building

Tabulka představuje budovu kulturní instituce. S budovami mohou být spjaté exponáty a usnadňovat tak návštěvníkům orientaci.

- Atribut `id` je primárním klíčem budovy.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k jejímu vytvoření.
- Atribut `description` uchovává popis budovy.
- Atribut `name` obsahuje název budovy.
- Atribut `institution_id` je cizí klíč pojící budovu s kulturní institucí, kterou je vlastněna.

6.3.8 Tabulka room

Tato tabulka reprezentuje místnost nacházející se v budově kulturní instituce. Místnost je dalším prvkem v hierarchii lokalizace určené pro snazší orientaci návštěvníků.

- Atribut `id` je primárním klíčem místnosti.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k jejímu vytvoření.
- Atribut `description` uchovává popis místnosti.
- Atribut `name` obsahuje název místnosti.
- Atribut `building_id` je cizí klíč pojící místnost s budovou, uvnitř které se nachází.

6.3.9 Tabulka showcase

Tabulka reprezentuje vitrínu a je posledním prvkem v hierarchii lokalizace exponátu.

- Atribut `id` je primárním klíčem vitríny.
- Atribut `created_at` obsahuje datum a čas, kdy došlo k jejímu vytvoření.
- Atribut `description` uchovává popis vitríny.
- Atribut `name` obsahuje název vitríny.
- Atribut `room_id` je cizí klíč pojící vitrínu s místností, uvnitř které se nachází.

6.3.10 Ostatní tabulky

Tabulky označené `institution_image`, `exhibit_image`, `building_image`, `room_image` a `showcase_image` umožňují přidávat větší množství obrázků k institucím, exponátům, budovám, místnostem a vitrínám. Nejsou výše popsány, protože se v databázi vyskytují pouze pro možné budoucí rozšíření projektu a nejsou systémem v této verzi používány. Zbylé dosud nezmiňované tabulky jsou vazebního typu a slouží k podpoře vztahů *M:N*.

6.4 Koncové body (endpointy) serveru

Tato podkapitola se zabývá návrhem RESTového API, které bude vystaveno serverem. Na základě jeho definice se bude řídit komunikace mezi serverem a webovým rozhraním nebo mobilní aplikací. Pro přidávání, získávání nebo manipulaci s daty budou na serveru přístupné koncové body poskytující potřebnou funkcionalitu.

Popis koncových bodů bude rozdělen do několika částí obsahujících funkcionalitu, která spolu nějakým způsobem souvisí. Jednotlivé body budou popsány ve tvaru:

<role>: <HTTP metoda>: <URL adresa zkrácená o svůj kořen>.

Kde *role* označuje roli uživatele volajícího koncový bod a může nabývat následujících hodnot:

- N - nepřihlášený uživatel,
- U - přihlášený uživatel,
- S - správce instituce,
- P - překladatel,
- A - administrátor systému.

Ve všech případech kromě koncových bodů označených N (nepřihlášený uživatel) tedy musí být při volání dodán také platný autentizační token ve hlavičce HTTP požadavku.

Obsah této podkapitoly může sloužit i jako uživatelská dokumentace v případě vytváření nové klientské aplikace.

6.4.1 Koncové body související s uživateli

Tato sada koncových bodů se zabývá podporou uživatelských účtů. Jedná se zejména o vytváření nových uživatelů, jejich přihlašování nebo aktualizování osobních údajů.

N: POST: /users/register

Tento koncový bod slouží pro registraci nových uživatelů. V rámci těla požadavku očekává údaje o novém uživateli. Mezi tyto údaje patří uživatelské jméno, e-mail a heslo. Pokud vše proběhne v pořádku, vrací informační hlášku indikující úspěšnou registraci v systému.

N: POST: /users/login

Tento bod slouží pro přihlašování do systému. V těle požadavku musí být přítomny údaje o jménu a heslu přihlašovaného. Při úspěšném přihlášení dojde k vrácení tokenu, který má omezenou platnost a lze se ním prokazovat svou totožnost v rámci systému.

U: GET: /users/token

Pomocí tohoto bodu si může přihlášený uživatel, jehož token se již blíží expiraci, nechat vygenerovat token nový. Ten bude vrácen v těle odpovědi.

U: PUT: /users/updateUser

Cílem bodu je poskytnout podporu pro aktualizaci osobních údajů. V těle požadavku je očekávána nová hodnota uživatelského jména a e-mailové adresy. Po úspěšné aktualizaci je vrácena informační zpráva.

U: PUT: /users/updatePassword

Tento koncový bod slouží pro aktualizaci hesla. V těle příchozího požadavku tedy očekává novou hodnotu hesla. Po úspěšném provedení aktualizace je vrácena kladná informační zpráva.

6.4.2 Koncové body související s institucemi

Tato sada obsahuje koncové body pro práci s institucemi. Obsahuje podporu pro jejich vytváření, získávání nebo aktualizování údajů. Velkou většinu zde uvedených mohou úspěšně volat pouze správci institucí.

N: GET: /institutions

Tento koncový bod slouží k získávání všech vytvořených kulturních institucí. Pro svou funkčnost nepotřebuje žádný vstup a jeho produktem je seznam všech institucí vrácených v těle odpovědi.

N: GET: /institutions/ordered

I tento bod vrací seznam všech kulturních institucí, ale v tomto případě seřazených na základě vzdálenosti od uživatele. Za tímto účelem je tedy v rámci těla požadavku potřeba předat uživatelovu zeměpisnou šířku a délku (např. 49.745).

U: POST: /institutions/myInstitution

Účelem tohoto bodu je umožnit vytváření nových kulturních institucí. Proto je potřeba v těle požadavku poskytnout nezbytné údaje. Těmi jsou název, adresa a souřadnice. Případně lze také dodat fotografii instituce. Odpovědí na požadavek je kladná informační zpráva v případě, že došlo k vytvoření.

S: GET: /institutions/myInstitution/languages

Tento koncový bod nepotřebuje žádné dodatečné parametry. Slouží k získání seznamu jazyků, které byly přidány k instituci, a seznamu jazyků, které ještě lze přidat.

S: POST: /institutions/myInstitution/languages/<languageId>

Cílem tohoto bodu je podpora pro přidávání jazyků k institucím. Pokud je jazyk k instituci přidán, překladatelé do něho mohou překládat. Bod potřebuje v rámci URL adresy předat identifikátor jazyku jako parametr. Po úspěšném přidání vrací informační zprávu.

S: PUT: /institutions/myInstitution/updateImage

Tento bod slouží k aktualizaci fotografie instituce. Za tímto účelem je jako parametr v těle požadavku předán obrázek zakódovaný pomocí *Base64* kódování. Po úspěšném provedení vrací informační zprávu.

S: PUT: /institutions/myInstitution

Cílem tohoto bodu je umožnit aktualizaci základních údajů o instituci. V těle požadavku tedy musí být dodány aktualizované hodnoty jména, adresy a zeměpisných souřadnic. Po aktualizování je vrácena informační zpráva.

U: GET: /institutions/myInstitution

Pomocí tohoto bodu může správce kulturní instituce získat všechny její základní údaje. Není třeba předávat žádné parametry.

S: POST: /institutions/myInstitution/addManager

Díky tomuto bodu je správcům instituce umožněno přidat dalšího správce pomocí zaslání nového správcovského účtu na e-mail. Je tedy nutné v těle požadavku dodat e-mailovou adresu, která patří budoucímu správci. Po úspěšném zaslání e-mailu je vrácena informační zpráva.

S: DELETE: /institutions/myInstitution

Použitím tohoto bodu může správce instituce smazat instituci, kterou spravuje. Není potřeba předávat parametry a po úspěšném smazání je opět vrácena informační zpráva.

6.4.3 Koncové body související s exponáty

Pomocí této sady koncových bodů lze manipulovat s exponáty. Obsahuje podporu pro jejich přidávání, upravování, získávání a mazání. Podobně jako u předchozí snahy může většinu z bodů volat pouze správce instituce.

N: GET: /exhibits/all/<institutionId>

Díky tomuto koncovému bodu lze získávat seznam všech exponátů vyskytujících se v kulturní instituci. Ta je specifikována svým identifikátorem předaným v rámci URL adresy. Žádné další parametry nejsou potřeba.

S: GET: /exhibits/all

Pomocí tohoto bodu lze také získat seznam všech exponátů, ale tentokrát není potřeba předávat vůbec žádné parametry, protože instituce je vybrána ta, kterou spravuje přihlášený uživatel.

S: GET: /exhibits/<exhibitId>

Použitím tohoto bodu lze získat údaje o exponátu. Exponát je definován identifikátorem předaným v rámci URL a žádné další parametry nejsou potřeba.

S: GET: /exhibits/<exhibitId>/qrcode

Díky tomuto bodu lze získávat QR kódy pro jednotlivé exponáty. Exponát je definovaný identifikátorem zadaným v URL adrese. V těle odpovědi je dodán požadovaný obrázek s QR kódem zakódovaný pomocí Base64 kódování.

S: DELETE: /exhibits/<exhibitId>

Pomocí tohoto koncového bodu může správce instituce smazat exponát definovaný identifikátorem předaným v URL adrese. Po úspěšném smazání je mu vrácena informační zpráva.

S: POST: /exhibits

Tento bod slouží k tomu, aby mohl správce instituce přidávat nové exponáty. V těle požadavku tedy musí být předány informace o vytvářeném exponátu. Jsou jimi název exponátu a identifikace budovy, místnosti a vitríny, kde je umístěn. Dalšími parametry jsou přepsaný obsah informačního popisku a zakódované fotografie popisku a exponátu. Po úspěšném vytvoření je jako odpověď zaslána informační zpráva.

N: POST: /exhibits/<institutionId>

Tento bod je velmi podobný předchozímu, pouze ho nepoužívají správci instituce, ale obyčejní návštěvníci. V rámci URL adresy je tedy specifikováno, k jaké instituci je exponát přidáván pomocí identifikátoru instituce. Po přidání je opět zaslána informační zpráva.

S: PUT: /exhibits/<exhibitId>/updateImage

Díky tomuto bodu lze aktualizovat fotografii exponátu. Exponát je určen pomocí jeho identifikátoru zadaného v URL adrese a Base64 zakódovaná nová fotografie je dodána v těle požadavku. Po provedení aktualizace se vrátí informační zpráva.

S: PUT: /exhibits/<exhibitId>/updateInfoLabel

Tento bod je velmi podobný předchozímu. Také slouží pro aktualizaci fotografie, ale tentokrát informačního popisku. Identifikátor exponátu i aktualizovaná fotografie jsou předány stejnou cestou. Po aktualizaci se opět posílá informační zpráva.

S: PUT: /exhibits/<exhibitId>

Díky tomuto bodu je umožněno aktualizovat informace o exponátu. Exponát je definován pomocí identifikátoru v URL adrese. V těle požadavku jsou předány aktualizované informace jako je název, umístění v rámci instituce nebo přepis informačního popisku. Po aktualizaci je vrácena informační zpráva.

S: GET: /exhibits/approveTranslations

Pomocí tohoto bodu mohou správci institucí jednoduše získávat přehled o exponátech a povolených jazycích. Není třeba dodávat žádné parametry. Vráceny jsou všechny exponáty instituce, kterou uživatel spravuje, a také všechny podporované jazyky.

P: GET: /exhibits/translate/<institutionId>

Účelem tohoto bodu je poskytnout překladatelům přehled exponátů a podporovaných jazyků určené instituce. Instituce je definována identifikátorem v URL adrese. Vráceny jsou všechny její exponáty a seznam podporovaných jazyků.

6.4.4 Koncové body související s překlady

Tato sada koncových bodů umožňuje překladatelům vytvářet a hodnotit překlady. Dále také umožňuje spravovat verze překladů včetně vracení se k předchozím pokusům nebo mazání celých sekvencí.

P: GET: /translations/sequences

Pomocí tohoto bodu lze získat všechny sekvence překladů přihlášeného uživatele. Není třeba předávat žádný parametr. Výstupem je seznam sekvencí v těle odpovědi.

P: DELETE: /translations/sequences/<exhibitId>/<languageId>

Účelem tohoto bodu je umožnit překladateli smazat celou svou sekvenci překladů spojených s daným exponátem a jazykem. Tyto parametry jsou definovány pomocí identifikátorů v URL adrese. Po dokončení mazání je navrácena informační zpráva.

P: GET: /translations/sequence/<exhibitId>/<languageId>

Tento bod je určený pro získávání všech verzí překladu týkajících se daného exponátu a jazyka. To znamená celé překladové sekvence vytvořené překladatelem. Identifikátory exponátu a jazyka jsou opět součástí URL adresy.

P: DELETE: /translations/sequence/<translationId>

Pomocí tohoto bodu se překladatel může vracet k předchozím verzím překladu. Je to provedeno smazáním všech verzí, které jsou novější, než verze specifikována identifikátorem v URL adrese. Po provedení akce je vrácena informační zpráva.

P: GET: /translations/new/<exhibitId>/<languageId>

Tento koncový bod slouží pro získávání informací důležitých k vytvoření překladu. Jazyk a exponát, pro které je překlad určen, jsou definovány po-

mocí identifikátorů v URL adrese. Mezi informace důležité pro překlad se řadí například název exponátu nebo název uložené fotografie informačního popisku.

P: POST: /translations/new/<exhibitId>/<languageId>

Díky tomuto bodu je umožněno překladatelům vytvářet nové překlady. Exponát a jazyk vázaný na překlad jsou určeny identifikátory v URL adrese. V těle požadavku musí být předán také samotný přeložený text. Po uložení překladu je vrácena kladná informační zpráva.

P, S: GET: /translations/rate/<exhibitId>/<languageId>

Pomocí tohoto bodu lze získat všechny překlady spojené s daným exponátem a jazykem a další důležité informace potřebné k hodnocení překladů. Identifikátory exponátu a jazyka jsou opět součástí URL adresy.

S: PUT: /translations/official/<translationId>

Díky tomuto bodu může správce instituce označit překlad, který se má zobrazovat v mobilní aplikaci. Popřípadě může odznačením překladu svou volbu vzít zpět. Překlad je definován identifikátorem v URL adrese. Naopak hodnota, zda se překlad má zobrazovat nebo ne, je předána v těle požadavku. Po změně stavu je vrácena informační zpráva.

N: GET: /translations/official/<exhibitId>/<languageCode>

Pomocí tohoto bodu mohou návštěvníci kulturní instituce získávat schválené překlady pro exponáty a jazyky definované v rámci URL adresy.

P, S: PUT: /translations/like/<translationId>

Za použití tohoto bodu mohou překladatelé (nebo správci institucí při schvalování oficiálních překladů) označovat překlady, které se jim líbí, a případně vzít svou volbu zpět. Hodnocený překlad je definovaný v URL adrese a hodnota, zda se překlad má označit nebo odznačit jako oblíbený, je předaná v těle požadavku. Po zanesení do systému je vrácena informační zpráva.

6.4.5 Koncové body související s administrací

Díky této skupině koncových bodů mohou administrátoři systému provádět správu uživatelů. Mohou jim například odebírat nebo obnovovat práva na

překlad nebo na přístup do celého systému. K funkcionalitě uvedené v této podkapitole tedy běžní uživatelé systému nemají přístup.

A: GET: /admin/users

Tento koncový bod poskytuje funkcionalitu pro získávání všech uživatelů, kteří nemají administrátorská práva. Není potřeba předávat žádný dodatečný parametr.

A: GET: /admin/users/<userId>

Díky tomuto bodu lze získat detaily související s konkrétním uživatelem. Uživatel je definován identifikátorem obsaženým v URL adrese. V těle odpovědi jsou poté obsaženy takové informace, jako například jméno, e-mailová adresa, datum registrace nebo příznak, zda má uživatel povolený přístup do systému.

A: PUT: /admin/users/<userId>/updateUsername

Pomocí tohoto bodu správce systému dovede změnit jméno uživatele. Aktualizované jméno je dodáno v rámci těla požadavku a uživatel je specifikován identifikátorem v rámci URL adresy. Po provedení aktualizace je vrácena informační zpráva.

A: PUT: /admin/users/<userId>/updatePassword

Účelem tohoto bodu je poskytnout administrátorům možnost měnit uživatelům hesla na jejich vyžádání. Je to provedeno zasláním nově vygenerovaného náhodného hesla na uživatelovu e-mailovou adresu. Identifikace uživatele probíhá pomocí identifikátoru obsaženého v URL adrese. Po odesláním e-mailové zprávy je vrácena informační zpráva o kladném průběhu.

A: PUT: /admin/users/<userId>/updateTranslator

Tento bod slouží k odebírání nebo obnovování překladatelských práv. Informace o tom, zda dochází k odebrání nebo obnovení, je obsažena v těle požadavku. Uživatel je specifikován identifikátorem obsaženým v URL adrese. Po provedení aktualizace práv je vrácena informační zpráva.

A: PUT: /admin/users/<userId>/removeInstitution

Pomocí tohoto bodu mohou administrátoři zbavit uživatele správcovských práv vůči instituci. Uživatel je definován pomocí identifikátoru v URL adrese

a po úspěšném odebrání správcovství je vrácena informační zpráva. Pokud nastane stav, že instituce nemá žádného správce, dojde k jejímu odstranění ze systému.

A: PUT: /admin/users/<userId>/updateBan

Závěrečný bod umožňující správu uživatelů je určen k modifikaci přístupu uživatele do systému. Uživatel může mít odepřen nebo opět povolen přístup do systému a aktuální nastavovaná hodnota je přenesena v rámci těla požadavku. Uživatel je specifikován v URL adrese pomocí identifikátoru. Po úspěšném nastavení přístupových práv je vrácena informační zpráva.

6.4.6 Koncové body spojené s umístěním exponátu

Instituce může být v zájmu orientace návštěvníka hierarchicky rozdělena na budovy, místnosti a vitríny, ke kterým lze přidávat exponáty. Účelem této sady koncových bodů je tedy vytvořit podporu pro správu jednotlivých lokací. Může se jednat například o funkcionalitu spojenou s jejich zakládáním, získáváním nebo mazáním.

S: GET: /location/buildings

Díky tomuto koncovému bodu lze získávat seznam všech budov náležící kulturní instituci spravované přihlášeným uživatelem.

N: GET: /location/buildings/all/<institutionId>

Tento bod je velmi podobný předchozímu. Také slouží pro získávání všech budov, ale instituce již není specifikována na základě přihlášeného uživatele. Místo toho je předán její identifikátor v rámci URL adresy.

S: GET: /location/buildings/<buildingId>

Použitím tohoto bodu lze získat údaje o budově. Budova je definována identifikátorem předaným v rámci URL a žádné další parametry nejsou potřeba.

S: POST: /location/buildings

Tento bod slouží k zakládání nových budov. V těle požadavku musí být předány informace o nové budově, kterými jsou její název a popis. Po úspěšném založení dojde k vrácení informační zprávy.

S: PUT: /location/buildings/<buildingId>

Díky tomuto bodu je umožněno aktualizovat informace o budově. Budova je definována identifikátorem dodaným v URL adrese a aktualizované údaje jsou předány v těle požadavku. Po aktualizaci je vrácena informační zpráva.

S: DELETE: /location/buildings/<buildingId>

Pomocí tohoto koncového bodu může správce instituce smazat budovu definovanou identifikátorem v URL adrese. Po smazání dojde k vrácení informační zprávy.

N: GET: /location/rooms/all/<buildingId>

Cílem tohoto bodu je umožnit získávání seznamu všech místností uvnitř budovy specifikované identifikátorem předaným v URL adrese.

S: GET: /location/rooms/<roomId>

Použitím tohoto bodu lze získat údaje o místnosti. Místnost je definována identifikátorem nacházejícím se v URL adrese.

S: POST: /location/rooms/<buildingId>

Tento bod slouží k zakládání nových místností. V těle požadavku musí být tedy předán název a popis místnosti. Budova, ve které se místnost nachází, je definována identifikátorem v URL adrese. Po založení dojde k vrácení informační zprávy.

S: PUT: /location/rooms/<roomId>

Díky tomuto bodu je umožněno aktualizovat informace o místnosti. Místnost je definována identifikátorem dodaným v URL adrese a aktualizované údaje jsou předány v těle požadavku. Po aktualizaci je vrácena informační zpráva.

S: DELETE: /location/rooms/<roomId>

Pomocí tohoto koncového bodu lze smazat místnost definovanou identifikátorem v URL adrese. Po smazání dojde k vrácení informační zprávy.

N: GET: /location/showcases/all/<roomId>

Cílem tohoto bodu je umožnit získávání seznamu všech vitrín uvnitř místnosti specifikované identifikátorem předaným v URL adrese.

S: GET: /location/showcases/<showcaseId>

Použitím tohoto bodu lze získat údaje o vitríně. Vitrína je definována identifikátorem nacházejícím se v URL adrese.

S: POST: /location/showcases/<roomId>

Tento bod slouží k zakládání nových vitrín. Místnost, kde se vitrína nachází, je definována v rámci URL adresy. V těle požadavku musí být také předány informace o jméně a popisu vitríny. Po založení je vrácena informační zpráva.

S: PUT: /location/showcases/<showcaseId>

Díky tomuto bodu je umožněno aktualizovat informace o vitríně. Nové údaje jsou předány v těle požadavku a vitrína je definována identifikátorem v URL adrese. Po aktualizaci je vrácena informační zpráva.

S: DELETE: /location/showcases/<showcaseId>

Pomocí tohoto koncového bodu lze smazat vitrínu definovanou identifikátorem v URL adrese. Po smazání dojde k vrácení informační zprávy.

7 Popis implementace

Cílem této kapitoly je přiblížit implementaci obou částí vytvořeného systému. Nejprve bude popisována serverová část a následně webové rozhraní.

7.1 Implementace serveru

Hlavním úkolem serveru je přijímat a dodávat data mobilní aplikaci a webovému rozhraní. Implementace probíhala na principech softwarového architektonického vzoru REST a jako formát pro výměnu dat byl zvolen JSON.

Jak již bylo zmíněno u analýzy dostupných technologií pro tvorbu RESTového serveru, pro implementaci byl použit populární framework Spring Boot. Pro usnadnění správy závislostí a sestavování byl do projektu zaveden známý nástroj *Maven*. Díky němu pak byly přidány užitečné knihovny, jako například knihovna pro generování QR kódů.

Samotné psaní programu probíhalo na základě návrhu databázového modelu a koncových bodů. Byla vynaložena snaha, aby byl kód co nejjasněji strukturovaný a jeho pochopení usnadňovaly komentáře psané v dostatečném množství.

7.1.1 Databáze napojená na server

Pro tvorbu reálně použitelného softwarového produktu bylo zapotřebí vyřešit situaci, jak trvale uchovávat potřebná data. Za tímto účelem byl použit velmi populární systém řízení báze dat *MySQL*. Tento systém byl vybrán hlavně díky své rozšířenosti a open-sourcové povaze.

Pro správné fungování aplikace tedy musí být spuštěný aplikační server a zároveň databázový server. Databáze nevyžaduje nijak rozsáhlou konfiguraci. Postačuje, aby byla v systému založena (například pomocí nástroje *MySQL Workbench*), protože o všechno ostatní se již postará aplikační server. Ten dovede vytvořit potřebné databázové tabulky a navíc je naplnit výchozími daty, pokud k jejich uložení ještě nedošlo.

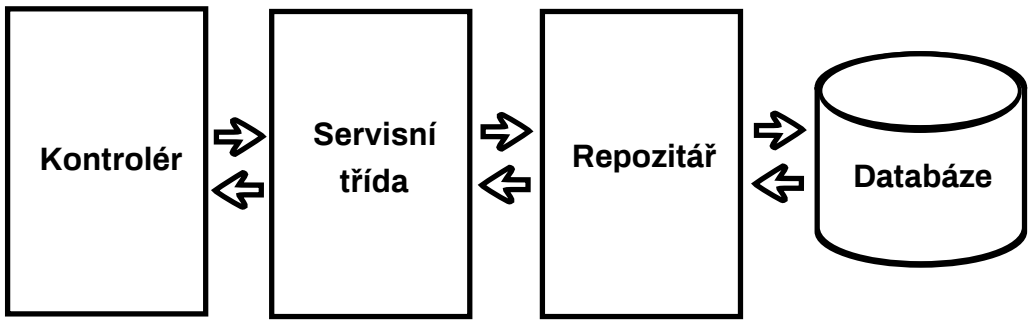
7.1.2 Obecný průběh zpracování požadavku

Cílem této podkapitoly je poskytnout představu o tom, jak probíhá zpracování HTTP požadavku zaslaného na server. Rozbor bude zjednodušený a vysokoúrovňově orientovaný. Obsahem bude obecný seznam kroků, které

jednotlivé požadavky podstupují, doplněný o příklady konkrétních tříd starajících se o danou problematiku.

Požadavky jsou zpracovávány následujícím způsobem:

- Přijetí HTTP požadavku od klienta.
- Pokud daný koncový bod není volně přístupný, dojde k ověření totožnosti uživatele a v případě nutnosti také k ověření, zda uživatel disponuje požadovanou rolí. Definice zabezpečení koncových bodů se nachází ve třídě `SecurityConfiguration` a ověřování totožnosti probíhá ve třídě `JwtAuthenticationFilter`.
- Následně dochází k namapování těla HTTP požadavku ve formátu JSON na instance tříd programovacího jazyka.
- Poté se požadavek předá kontroléru (např. `UserController` nebo `InstitutionController`), který požadavek předá dále servisním třídám a obvykle je nechává vykonat většinu logiky. Toto předání je znázorněno na obrázku 7.1.
- V servisní třídě (např. `AdminService` nebo `ExhibitService`) jsou provedeny operace potřebné pro vykonání uživatelem vyžadované funkcionality.
- Servisní třída obvykle využívá služeb repozitářů, které mají na starosti komunikaci s databází.
- Repozitář (např. `TranslationRepository` nebo `UserRepository`) provede potřebné změny v databázi a případně vrátí servisní třídě získaná data.
- Po provedení logiky se program vrací zase do kontroléru, kterému lze také předat data získaná v nižších vrstvách.
- Pokud jsou dodána nějaká data patřící do těla HTTP odpovědi, je provedena jejich konverze do formátu JSON.
- Odeslání odpovědi iniciátorovi komunikace.



Obrázek 7.1: Nejdůležitější kroky při zpracování požadavku

7.1.3 Struktura a popis projektu

Tato podkapitola se zaměřuje na přiblížení zdrojového kódu a dalších pomocných souborů programu. Uvedeno bude i to, jak jsou jednotlivé třídy děleny do balíků. Výchozí balík, ve kterém jsou všechny dílčí balíky, je `cz.zcu.students.cacha.bp_server`.

Soubor pom.xml

Tento soubor se nachází v kořenové složce se zdrojovým kódem serveru. Obsahuje mimo jiné seznam závislostí na externí knihovny ve formátu XML, a proto je velmi důležitý pro nástroj *Maven*, který na jeho základě provádí sestavování aplikace.

Soubor application.properties

Tento soubor se nachází ve složce `src/main/resources` a obsahuje konfigurační údaje. Údaje jsou načteny při startu a jejich pomocí je provedeno nastavení parametrů aplikace. Formát zápisu dodržovaný v tomto souboru je velmi jednoduchý. Na každé řádce je specifikován jeden konfigurační údaj zapsaný ve tvaru: *<celý název vlastnosti>=<hodnota>*.

Mezi nejdůležitější nastavené parametry se řadí:

- Připojení k databázi je nastaveno ve vlastnosti `spring.datasource.url`.
- Jméno uživatele, pomocí kterého dochází k připojování k databázi, je nastaveno ve vlastnosti `spring.datasource.username`.

- A jeho heslo následně v `spring.datasource.password`.
- Název složky s fotografiemi institucí je uveden ve vlastnosti `cts.paths.institutions_images_folder`. Podobně jako názvy složek s fotografiemi exponátů a informačních popisků, je tato vlastnost vytvořena speciálně pro tento projekt a není využívána žádnými knihovnami.
- Název složky s fotografiemi exponátů je definován v `cts.paths.exhibits_images_folder` a název složky s informačními popisky v `cts.paths.info_labels_images_folder`.

Zbylý obsah složky src/main/resources

Zbylým obsahem této složky jsou dva pomocné soubory. Prvním z nich je obrázek `defualt.png`, který představuje chybějící fotografii a používá se, pokud uživatel nepřidal fotku instituce nebo exponátu. Druhým je soubor `language_codes.csv` obsahující výčet 184 nejpoužívanějších jazyků a jejich standardních dvoumístných kódů. Na základě tohoto souboru je v rámci databáze naplněna tabulka s jazyky.

Obsah složky src/main/resources/static

V této složce se vyskytuje sestavené webové rozhraní optimalizované pro produkční nasazení.

Třída BpServerApplication

Tato třída se nachází v základním balíku `cz.zcu.students.cacha.bp_server`. Je to hlavní vstupní bod programu, protože obsahuje statickou metodu `main(...)` volanou při spuštění aplikace.

Mimo to třída slouží pro vytváření instancí sdílených napříč celým programem. Obsahuje také funkcionalitu pro vkládání rolí a jazyků do databáze, pokud se tam při spuštění programu již nevyskytují.

Balík assets_store_config

Tento balík je určen pro správu zobrazování všech druhů fotografií, které uživatelé nahráli do systému. Při návrhu struktury projektu bylo počítáno s více třídami, ale nakonec balík obsahuje pouze třídu `WebConfiguration`.

Třída načítá z konfiguračního souboru jména složek s fotografiemi institucí, exponátů a informačních popisků. Při startu programu pak zkontroluje,

zda tyto složky existují a pokud tomu tak není, jsou vytvořeny. Navíc do složek s fotografiemi institucí a exponátů vkládá výchozí obrázek zobrazovaný, pokud fotografie není dostupná.

Díky implementaci rozhraní `WebMvcConfigurer` má třída možnost překrýt metodu `addResourceHandlers(...)`. To se také děje a je na základě toho dosaženo namapování URL adres na složky s fotografiemi, které potom mohou být stahovány z mobilní aplikace nebo webového rozhraní.

Balík controllers

V tomto balíku se vyskytuje definice RESTového API. Jednotlivé koncové body jsou poskytované kontroléry, které jsou anotované anotací `@RestController` poskytovanou frameworkem. Kód respektuje návrh rozhraní, a proto jsou jednotlivé body rozděleny do více tříd. Jsou jimi třídy `AdminController`, `ExhibitController`, `InstitutionController`, `TranslationController`, `LocationController` a `UserController`.

Třídy obsažené v tomto balíku slouží k přehlednému oddělení definice rozhraní, a proto jsou jejich metody z pravidla velmi krátké. Provedení potřebných operací je přenecháno servisním třídám, jak lze vidět na ukázce kódu 7.2.

```
@GetMapping
public List<InstitutionVM> getInstitutions() {
    List<InstitutionVM> institutions =
        institutionService.getInstitutions();
    return institutions;
}
```

Kód 7.2: Ukázka koncového bodu pro získávání registrovaných institucí

Balík domain

Na základě tříd soustředěných do tohoto balíku je generována databáze. Namapování tříd na tabulky v databázi se provádí pomocí anotace `@Entity` uvedené nad definicí třídy. Instance takto označených tříd, pokud již došlo k jejich uložení, představují řádku databázové tabulky.

V balíku se vyskytují třídy představující hlavní objekty, se kterými systém pracuje. Nejdůležitějšími třídami jsou `Exhibit`, `Institution`, `Language`, `Role`, `Translation` a `User`. Vztahy entit, pomocí kterých dochází k definici vazebních tabulek, jsou také určovány na základě anotací.

Významnými anotacemi jsou například `@OneToMany` a `@ManyToOne` umožňující vytvářet vztahy *1:N*. Další podstatnou anotací je pak `@ManyToMany` umožňující tvorbu vztahů *M:N*.

Pro udržení přehledného kódu byla využita anotace `@Data` z knihovny *Lombok*. Ta mimo jiné zajišťuje generování metod, které umožňují získávání (getters) a nastavování (setters) soukromých atributů tříd. Kód obsahuje také validační anotace použité ke kontrole vstupu uživatelů, jejichž použití lze vidět v ukázce 7.3. Mezi nejdůležitější validační anotace se řadí `@NotNull` kontrolující, že atribut nenabývá hodnoty `null`, `@Size` ohraničující délku řetězců, `@Pattern` pro ověření, že hodnota odpovídá danému regulárnímu výrazu, a nebo `@Email` ověřující, že hodnota obsahuje e-mailovou adresu.

```
@NotNull(message = "Username can not be blank")
@Size(min = 3, max = 30, message = "Username must be between 3 to
    30 letters long")
@UniqueUsername
@Column(length = 30)
private String username;
```

Kód 7.3: Ukázka atributu s validačními anotacemi

Balík error

V tomto balíku se vyskytují dvě třídy zodpovědné pro řešení chybových stavů. První z nich je třída `ApiError` sloužící jako šablona pro odpověď na chybový stav zasílaná v těle HTTP odpovědi. Mezi její atributy se řadí například časová značka, zpráva s chybou nebo výčet nevalidních vstupů upozorňujících uživatele na chybnou hodnotu ve formuláři.

Druhou třídou je třída `ValidationErrorsHandler` anotovaná pomocí `@RestControllerAdvice`. To jí umožňuje reagovat na výjimky vyskytující se za běhu programu a následně uživateli vrátit příslušné vysvětlující informace obsažené v instanci již zmiňované třídy `ApiError`.

Balík exceptions

Zde jsou soustředěny definice výjimek používaných napříč aplikací.

- Výjimka `CannotPerformActionException` nastává, když požadavek z nějakého důvodu nelze vykonat. Příkladem může být opakované zaslání požadavku na vytvoření kulturní instituce, ale uživatel může spravovat maximálně jednu.

- Výjimka `CannotSaveImageException` se používá pokud dojde k selhání uložení uživatelem nahrané fotografie na disk.
- Výjimka `NotFoundException` se vyskytne pokud uživatel dodá takové parametry, pro které nelze nalézt odpovídající záznamy v databázi.
- Výjimka `UnauthorizedException` nastává pokud uživatel požaduje provést akci s objektem, ke kterému nemá práva.
- Výjimka `ValidationErrorException` je použita, pokud uživatel dodal nevalidní parametry požadavku.

Balík repositories

Tento balík obsahuje repozitáře určené pro manipulaci s databází. Pro každou třídu anotovanou pomocí `@Entity` je připraven jeden repozitář. Ty jsou definovány jako rozhraní anotované pomocí `@Repository`, které navíc rozšiřuje rozhraní `JpaReposiotry` pocházející z *Java Persistence API* (JPA). JPA je implementováno hojně rozšířeným frameworkem *Hibernate*, který umožňuje objektově relační mapování (ORM).

Pokud nedostačují metody převzaté z rozhraní `JpaReposiotry`, existuje možnost vytvoření vlastního SQL dotazu, který bude prováděn. Příklad takového dotazu je vidět v ukázce 7.4. Mimo to lze také podle speciálního formátu definovat hlavičky metod, jejichž implementaci je framework schopný poskytnou automaticky.

```
// vrati uzivatele, kteri nejsou administratori
@Query(
    value = "select * from user u " +
        "where 'ROLE_ADMIN' not in (" +
        "select r.name from role r " +
        "join users_roles ur on r.id = ur.role_id " +
        "where ur.user_id = u.id) " +
        "order by username",
    nativeQuery = true)
List<User> getNonAdminUsers();
```

Kód 7.4: Ukázka vlastního SQL dotazu

Balík responses

Tento jednoduchý pomocný balík obsahuje třídy, jejichž instance se po konverzi do formátu JSON posílají v těle některých HTTP odpovědí. Jsou zde

umístěny dvě třídy. První z nich je `GenericResponse` sloužící jako šablona pro odpovědi informačního typu. Druhou je pak `JWTLoginSuccessResponse` použitá jako odpověď obsahující přihlašovací token zasílaná po úspěšném přihlášení.

Balík security

Balík `security` je velmi důležitý, protože zajišťuje autentizaci uživatelů a autorizaci jejich akcí na základě jimi vlastněných rolí. Přihlašování bylo implementováno pomocí *JSON Web Tokenů (JWT)* v součinnosti s frameworkem *Spring Security*. Pokud tedy uživatel chce provést akci, pro kterou je potřeba ověření, musí v rámci HTTP požadavku poskytnout také svůj platný token.

Třída `JwtTokenProvider` slouží pro generování nových tokenů, které v sobě mají zakódované základní informace o uživateli jako například jeho identifikátor nebo uživatelské jméno. Mimo to také třída obsahuje metodu, která dovede ověřit validitu dodaného tokenu.

Pomocí třídy `JwtAuthenticationFilter` dědící ze třídy `OncePerRequestFilter` lze přidat další filtr do řetězce filtrů spravovaného frameworkem *Spring Secuirty*. To se děje za využití překrytí metody `doFilterInternal(...)`, která obsahuje logiku pro přihlášení uživatele na základě platného tokenu.

Ve třídě `SecurityConfiguration` dochází k nastavování bezpečnostních parametrů. Nejdůležitější je metoda `configure(...)`, kde dochází k definici přístupových omezení pro páry URL adres a typů HTTP metod. Omezení jsou nastavena tak, že vyžadují přihlášení uživatele nebo povolují přístup pouze uživatelům disponujícím určitou rolí.

Balík services

Servisní třídy obsažené v tomto balíku jsou nejobsáhlejší z celého projektu. Vykonává se zde převážná část logiky, a proto tvoří jádro systému. Všechny třídy zde jsou anotované pomocí `@Service` a jsou následující:

- Třída `AdminService` obsahuje funkcionalitu spojenou se správou uživatelů.
- `AuthUserService` slouží pro podporu autentizace uživatelů.
- `EmailService` provádí odesílání e-mailových zpráv.
- Třída `ExhibitService` se stará o operace spojené s exponáty.

- `FileService` je určen pro mazání a ukládání fotografií nahraných uživateli.
- `InstitutionService` je servisní třída starající se o operace týkající se kulturních institucí.
- `LocationService` dodává potřebnou funkcionalitu pro správu budov, místností a vitrín.
- Třída `QRCodeService` generuje QR kódy obsahující identifikátory exponátů.
- `TranslationService` obsahuje funkcionalitu spojenou s překlady.
- `UserService` se stará o operace spojené s uživateli, jako je přihlašování nebo změna osobních údajů.

Balík shared

Balík `shared` má pomocný charakter. Je zde umístěna třída `RolesConstants` obsahující výčet rolí vyskytujících se v systému (role překladatele, správce instituce a administrátora). Dále je zde definována anotace `CurrentUser` umožňující automaticky získávat přihlášeného uživatele.

Balík validators

V tomto balíku se nacházejí speciální validátory na kontrolu uživatelského vstupu. Skládají se ze dvou částí. První z nich je definice anotace a druhou je třída implementující rozhraní `ConstraintValidator` obsahující samotnou logiku pro provádění validace. Vytvořené anotace se umisťují nad atributy, na které se mapuje uživatelský vstup. Validace obsažené v tomto balíku se zaměřují zejména na testování unikátnosti názvů v rámci systému nebo kontrolu, že nahraný soubor je formátu *PNG* nebo *JPEG*.

Balík view_models

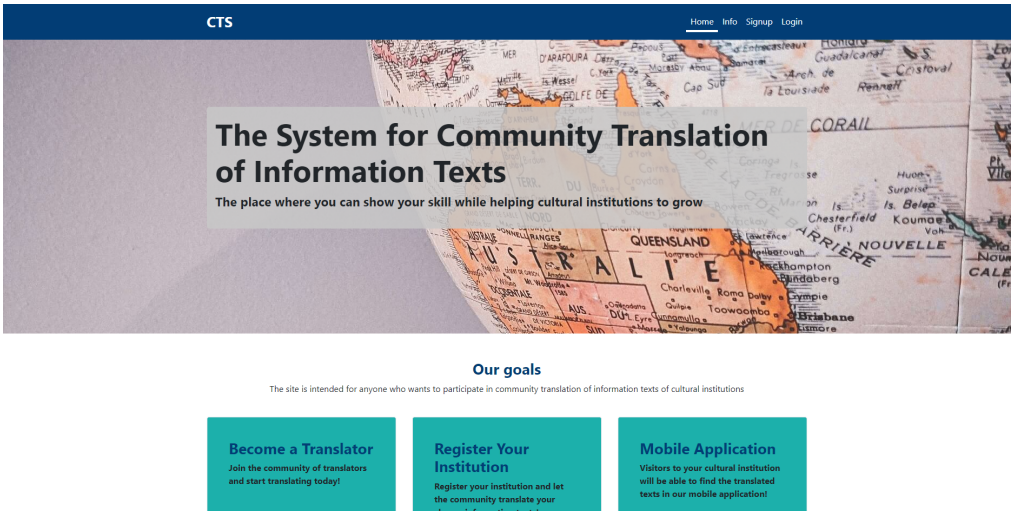
Zde se nachází jednoduché pomocné třídy určené pro komunikaci s webovým rozhraním nebo mobilní aplikací. Třídy obsahují minimální množství logiky a jejich atributy přímo odpovídají požadavkům obou klientů.

7.2 Implementace webového rozhraní

Webové rozhraní systému pro komunitní překlad dat z kulturních institucí bylo vytvořeno na principech moderního jednostránkového přístupu ke tvorbě webových aplikací. Při tvorbě byla vynaložena snaha, aby rozhraní působilo esteticky příjemným dojmem a poskytovalo pro své uživatele co největší komfort. Vzhledem k velkému rozšíření používání mobilních telefonů nebo tabletů při procházení internetových stránek bylo rozhraní uzpůsobeno i pro tato zařízení. Už od samého začátku tvorby bylo počítáno s responzivním chováním.

Jak již bylo zmíněno v kapitole zabývající se analýzou technologií pro implementaci webového rozhraní, pro jeho tvorbu byla použita v současné době velmi oblíbená knihovna React. Pro snazší řešení problémů, které nejsou podporovány knihovnou React, byly do projektu zavedeny závislosti i na další knihovny. Jednou z těch nejdůležitějších je například knihovna Redux umožňující centrálně spravovat stav aplikace.

Účelem rozhraní, jehož úvodní stránka je zobrazená na obrázku 7.5, je umožnit registrovaným uživatelům vytvářet kulturní instituce a přidávat jejich exponáty. Dále pak zajistit překladatelské komunitě platformu pro překlad informačních popisků.



Obrázek 7.5: Úvodní stránka webového rozhraní

7.2.1 Použité technologie

V této podkapitole jsou uvedeny důležité knihovny a frameworky použité při vývoji webového rozhraní.

Font Awesome

Font Awesome[1] je knihovna poskytující vysoké množství ikon. Ty byly použity k dekoraci tlačítek nebo položek menu s cílem zvýšit celkový estetický dojem z aplikace.

Bootstrap

Bootstrap[2] je framework, který byl do projektu přidán kvůli tvorbě vzhledu webu. Naprostá většina kaskádových stylů formujících komponenty je dodána třídami poskytovanými frameworkem. Kromě vzhledu je díky Bootstrapu dosaženo také responzivního zobrazení.

Axios

Axios[3] byl použit k usnadnění asynchronního volání serverové funkcionality. Jeho použití je založeno na moderním přístupu pomocí objektů typu *Promise*.

Redux

Knihovna Redux[4] byla použita pro centrální správu stavu aplikace. Díky tomu se například mezi komponentami nemusí složitě předávat informace o přihlášeném uživateli, ale stačí se napojit na centrální úložiště.

React Router

React Router[5] byl použit pro umožnění změn zobrazované stránky na základě modifikace URL adresy.

7.2.2 Struktura a popis projektu

V této podkapitole bude rozebrána struktura zdrojového kódu (konkrétně obsah složky *src*). Kód je za účelem zvýšení přehlednosti členěn do několika složek. Jsou jimi složky *apiCalls, assets, components, pages, shared a store*. Mimo to se ve složce *src* nachází také tři soubory neumístěné v žádné složce (*index.js, index.css a App.js*).

[1] https://fontawesome.com
[2] https://getbootstrap.com
[3] https://axios-http.com
[4] https://redux.js.org
[5] https://reactrouter.com

Soubor index.js

Tento soubor je pro celé fungování aplikace klíčový a má jediný hlavní účel. Tím je vykreslit vytvořené rozhraní do výchozí HTML stránky, která obsahuje hlavičku a element typu *div* s názvem *root* v těle. Kód vykonávající tuto činnost je v následující ukázce 7.6.

```
ReactDOM.render(
    <React.StrictMode>
        <Provider store={store}>
            <BrowserRouter>
                <App />
            </BrowserRouter>
        </Provider>
    </React.StrictMode>,
    document.getElementById('root')
);
```

Kód 7.6: Výkonný kód souboru *index.js*

Soubor index.css

Tento soubor obsahuje kaskádové styly využité různými elementy napříč celou aplikací. Obsah souboru není příliš dlouhý, protože většina stylování je poskytnuta třídami frameworku Bootstrap.

Soubor App.js

App.js obsahuje definici vysokoúrovňové komponenty určené pro řešení logiky směrování, jejíž jednoduchý příklad je v ukázce 7.7. Když dojde ke změně adresy, je na základě množiny uživatelových práv umožněn nebo odepřen přístup na požadovanou stránku. Ta je navíc obalena základním prvky stránky jako je patička nebo navigace v horní části obrazovky.

```
// prikaz umoznujici vykreslit stranku s uzivateli, pokud je
    prihlaseny uzivatel administratorem systemu
{user.isAdmin &&
    <Route exact path="/users" component={UserManagerPage} />}
```

Kód 7.7: Ukázka směrovacího příkazu

Složka store

V této složce jsou všechny soubory sloužící k manipulaci s centrálním úložištěm stavu aplikace a k jeho konfiguraci. Jeho vytvoření bylo dosaženo s využitím knihovny Redux a obsahuje informace o aktivním uživateli. Výchozí hodnota stavu úložiště je k vidění v ukázce kódu 7.8.

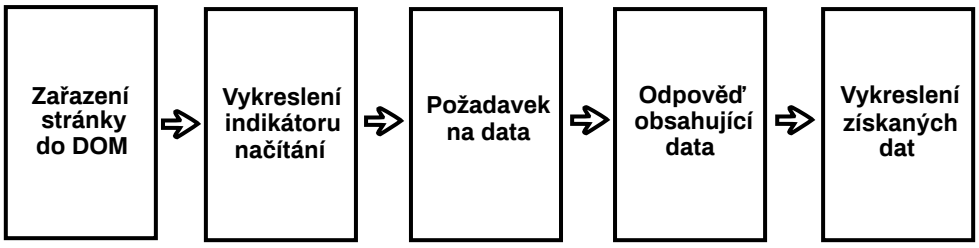
```
const initialState = {
    id: 0,
    username: "",
    email: "",
    createdAt: null,
    isTranslator: false,
    isInstitutionOwner: false,
    isAdmin: false,
    isLoggedIn: false,
    token: "",
    expiredAt: null,
}
```

Kód 7.8: Výchozí stav centrálního úložiště popisující aktivního uživatele

Vytvoření a konfigurace centrálního úložiště je provedena v souboru *store.js*. Jeho aktualizace má na starosti soubor *authReducer.js*, který tak činí na základě vstupu funkcí definovaných v souboru *authActions.js*.

Složka pages

Tato složka je naprosto klíčová a to z toho důvodu, že obsahuje jednotlivé stránky, mezi kterými se může uživatel pohybovat. Naprostá většina těchto stránek si drží svůj vnitřní stav a jsou složeny z dalších komponent, jako je například tlačítko s točivým ukazatelem průběhu. Zobrazování stránek je závislé na aktuálně aktivní URL adrese. Většina stránek neobsahuje statický, ale dynamicky načítaný obsah. Jejich životní cyklus je k vidění na obrázku 7.9.



Obrázek 7.9: Životní cyklus stránky s dynamicky načítaným obsahem

Složka components

Ve složce *components* se vyskytují komponenty, které většinou neuchovávají žádný svůj vnitřní stav. Jejich vyčlenění do jednotlivých souborů bylo provedeno zejména proto, že dochází k jejich opakování ve více úsecích projektu nebo jejich vyjmutí způsobilo zlepšení přehlednosti a udržitelnosti kódu.

Komponenty v této složce jsou v naprosté většině postaveny na bázi funkcí. Příkladem může být například tlačítko s indikací průběhu, generický formulářový vstup nebo animace s točícím se kolečkem naznačující načítání.

Složka apiCalls

V této složce se soustředí veškerý kód určený pro volání služeb serveru. Poskytuje tak rozhraní odstiňující od přesné specifikace serverového API, které obsahuje URL adresy nebo typy HTTP metod, jak lze vidět v ukázce kódu 7.10. Pro provádění volání byla použita knihovna Axios.

```
// asynchroni prihlasovaci HTTP pozadavek na server
export const login = credentials => {
    return axios.post("/users/login", credentials);
}
```

Kód 7.10: Funkce volaná za účelem přihlašování

Složka shared

Složka *shared* obsahuje funkcionalitu sdílenou napříč celým projektem. Jedná se například o konstanty používané v různých úsecích kódu nebo kód reagující na požadavek zaslaný s nedostatečným oprávněním.

Složka assets

Tato složka je určena pro všechny statické zdroje použité v rámci webové aplikace. Jedná se o obrázky, které jsou použity na uvítací a informační obrazovce nebo jako dekorativní pozadí pro podstránky nepřekrývající celou obrazovku.

8 Ověřování kvality

V této kapitole bude rozebráno, jakým způsobem byla ověřována kvalita systému pro komunitní překlad. Došlo na testování serveru i webového rozhraní. Obsah kapitoly je členěn do tří podkapitol, z nichž první pojednává o testování pomocí programových testů, druhá popisuje ověřování proti útokům a poslední popisuje, jak byl systém otestován potencionálními uživateli.

8.1 Programové testy

Tato podkapitola se soustředí na programové testy, kterými byl pokryt kód serverové části projektu a část funkcionality webového rozhraní. Testy byly rozděleny do čtyř balíků na základě jejich typu. Názvy balíků jsou `services_tests`, `integration_tests`, `security_tests` a `selenium_tests`. Mimo to ještě existuje balík `common`, který obsahuje logiku sdílenou napříč testy.

Za pomocí anotací byl testům nastaven nový profil umožňující využít odlišného nastavení od hlavní aplikace. Testy jsou spouštěny proti testovací databázi `museum_guide_db_test`, jejíž struktura je vždy před testy vytvářena automaticky znovu, a proto testy začínají s prázdnou databází.

Pro účely testování byl použit velmi oblíbený framework *JUnit* ve verzi 5. Po vytvoření testů byly všechny spuštěny s volbou sledovat pokrytí zdrojového kódu testy. Jako nástroj pro měření pokrytí kódu byl použit výchozí nástroj vývojového prostředí IntelliJ IDEA. Získané statistiky lze vidět v tabulce 8.1.

Třídy	Metody	Řádky kódu
86%	74%	74%

Tabulka 8.1: Statistiky pokrytí kódu testy

8.1.1 Testy servisních tříd

Servisní třídy tvoří jádro celého programu, a proto byly důkladně podrobeny programovému testování v balíku `services_tests`. Balík je dělen do tříd odpovídajících jednotlivým servisním třídám, kde se vyskytují metody anotované pomocí `@Test` provádějící testování. Všechny tyto testy lze označit za integrační, protože využívají závislostí používaných při běžném spuštění programu a nepoužívají takzvané *mock* objekty.

8.1.2 Integrační testy

Tento balík (`integration_tests`) obsahuje testy velmi podobné těm testujícím servisní třídy. Rozdíl spočívá v tom, že testy obsažené zde pokrývají ještě širší kus programu, protože netestují jednotlivé metody, ale vytvářejí testovací požadavky pomocí třídy `TestRestTemplate`. První metoda (nepocházející z knihoven) vyvolaná testem se tedy vždy nachází v nějakém kontroléru, a proto jsou i třídy tohoto balíku vytvářeny na jejich základě.

8.1.3 Bezpečnostní testy

Testy obsažené v balíku `security_tests` mají za cíl ověřit zabezpečení přístupu k jednotlivým koncovým bodům. Je to provedeno voláním zabezpečených bodů s tím, že se očekává, že návratový stavový HTTP kód bude typu *401 Unauthorized*. Pokud tomu tak není, koncový bod není pravděpodobně dostatečně zabezpečen.

Mimo to je také v balíku obsažen test, který kontroluje, zda se po registraci nového uživatele neukládá do databáze heslo přímo v takovém tvaru, ve kterém bylo zadáno. Test je nastaven tak, aby procházel pouze v případě, že je do databáze uložen výsledek hashovací funkce s parametrem uživatelem zadaného řetězce.

8.1.4 Testy pomocí Selenia

Selenium je nástroj pro automatické testování webových aplikací. Díky tomuto nástroji lze při testování přímo otevřít webový prohlížeč a automaticky vykonávat operace na webovém rozhraní. Tímto způsobem byl proveden test základní funkcionality, kterou je umožnit překladatelům registraci a vytvoření překladu. Test je obsažený v balíku `selenium_tests`.

Testování webového rozhraní probíhá pomocí prohlížeče *Google Chrome* ve verzi 100. Je to dáno jeho navázáním na speciální ovladač *chromedrive.exe*, který je využívaný Seleniem a je součástí programu.

8.2 Ověřování odolnosti proti útokům

Tato podkapitola si klade za cíl ověřit odolnost vytvořeného systému vůči běžným útokům na webové aplikace. Ověřování bylo provedeno vyzkoušením dvou nejběžnějších typů útoků (*SQL injekce* a *XSS*) na běžící aplikaci. Po jejich provedení se neprojevily zamýšlené nežádoucí účinky, a proto lze

konstatovat, že se systém útokům úspěšně ubránil. Dále budou popsány konkrétní příklady útoků a bude stručně představen princip jejich fungování.

8.2.1 SQL injekce

SQL injekce je technika útoku na databázovou vrstvu aplikace, která spočívá v pozměnění vykonaného SQL dotazu pomocí neošetřeného uživatelského vstupu. Takový útočníkem upravený dotaz pak může mazat databázové tabulky nebo vykonávat jinou škodlivou činnost. Framework využívaný v serverové části pro komunikaci s databází (*Hibernate*) naštěstí poskytuje podporu, pomocí které bylo takovým útokům zabráněno.

Útok SQL injekcí

Při registraci se ověřuje, zda uživatel s daným jménem není již v databázi pomocí SQL dotazu vyhledávajícího na základě jména. Toho využívá tento útok, při kterém bylo zadáno uživatelské jméno v následujícím tvaru s cílem smazat tabulku s institucemi:

```
u; DROP TABLE institution
```

8.2.2 XSS

XSS je útok spočívající ve vložení kódu z neošetřeného vstupu do dynamické webové stránky. Vložený kód i škodlivá činnost mohou být velmi různorodé. Knihovna React, jejíž pomocí bylo vytvořeno webové rozhraní, obsahuje obranu proti XSS automatickým ošetřováním vypisovaných řetězců. V rámci vypisování překladů, kde musí být vkládány přímo HTML elementy zvýrazňující text (pocházející z editoru), je navíc vstup ošetřen díky tzv. dezinfekci HTML.

První XSS útok

První útok proběhl vložením škodlivého kódu do vstupu s popisem instituce. Útok se ale nezdařil. Při vypisování popisu se zobrazil pouze text a k provedení skriptu tedy nedošlo. Vkládaný kód XSS útoku byl následující:

```
<script>alert('XSS útok!')</script>
```

Druhý XSS útok

Pro dosažení větší jistoty, že obrana proti XSS funguje, byl útok vložen i do editoru pro vytvoření nového překladu. Po jeho odeslání a zobrazení byl útok opět odražen vypsáním prostého textu. Kód XSS byl stejný jako v předchozím případě.

8.3 Uživatelské testy

Jako další vrstva v ověřování kvality vytvořeného systému bylo zvoleno uživatelské testování. Testeři byli vybíráni tak, aby byli co možná nejvíce různorodí. Výběr tedy obsahoval osoby disponující technickým i netechnickým vzděláním s poměrně širokým věkovým rozptylem. Testerům byla dodána URL adresa webové stránky a jejich úkol spočíval pouze v tom, aby se pokusili se stránkou pracovat jako běžní uživatelé. Nikdo z testerů nedostal scénáře testů, proto lze zvolený přístup označit jako *exploativní testování*. Dobrovolníků testujících aplikaci bylo celkem šest a jejich zprávy z testování jsou k dispozici v příloze D.

Ze zpráv vyplývá, že aplikaci považují testeři za užitečnou a použitelnou. Kritická je však absence podrobného vysvětlení chování systému. Toto je nejcennější informace získaná z testování a byla provedena snaha o její nápravu. Byla přidána samostatná informační stránka a u většiny stránek byl přidán informační popisek o jejich smyslu.

Na základě zbylé zpětné vazby uživatelů byly poté provedeny další akce. Mimo malého počtu nesprávné interpretace chování systému byly návrhy na změny věcné a užitečné. Došlo tedy k jejich rozdělení do dvou skupin podle jejich důležitosti a rozsahu. První skupina obsahuje připomínky, které byly přijaty a byla provedena snaha o jejich nápravu. Druhá skupina se skládá z možných rozšíření systému.

8.3.1 Provedené opravy

- Do webového rozhraní byla zavedena drobečková navigace.
- U rozsáhlých formulářů byl dodán indikátor povinnosti atributu.
- Na serveru byla ošetřena maximální velikost nahrávané fotografie.
- Překládané informační popisky byly vycentrovány.
- Byla přidána informační stránka dostupná z menu a pomocí odkazů u registrace a na uvítací obrazovce.

- Byly dodány informační popisky do stránek, kde nebylo zcela jasné, k čemu slouží.
- Místy byla provedena snaha o vylepšení vzhledu aplikace.
- Kartám na úvodní obrazovce byla dána stejná velikost a byly změněny jejich barvy.
- Patička webové aplikace byla upravena tak, aby nebyla vždy fixně u dolního kraje obrazovky, kde mohla překrývat část stránky. Po úpravě je až za obsahem nebo u dolního kraje (pokud je obsah stránky krátký).
- Zobrazování horního menu bylo upraveno tak, aby byla aktivní záložka podtržena.

8.3.2 Návrhy na rozšíření systému

- Aplikace může umožnit více jazykových mutací rozhraní.
- Informační popisky mohou být rozděleny na více fotografií.
- K přeloženým textům může být zřízena diskuze.
- Přeložené texty mohou být hodnoceny i záporně (palec dolů).
- Vytvořený systém by mohl za účelem zlepšení orientace návštěvníků podporovat také přidávání orientačních plánků institucí.
- Uživatelé mohou přidávat své profilové obrázky.
- Při prohlížení více objektů může být dodáno stránkování a filtr.
- Aplikace může umožňovat volbu velikosti písma.

9 Závěr

V rámci bakalářské práce proběhlo nejprve seznámení s již existujícím návrhem systému pro překlad popisků. Následně proběhla analýza, pomocí jakých technologií by mohl být projekt implementován. Byl proveden průzkum existujících ucelených řešení pro komunitní překlad i moderních technologií pro tvorbu serverů a webových rozhraní. Dalším krokem bylo vytvořit návrh implementace, jehož hlavními body byly především databázový model a struktura uživatelských rolí.

Po dokončení návrhu došlo k implementaci RESTového serveru a nad rámec zadání také ke tvorbě webového rozhraní. Když byl výsledný produkt hotový, došlo na jeho otestování automatizovanými programovými testy a také skupinou uživatelů provádějících manuální testování.

Hlavním cílem práce bylo vytvořit server a webové rozhraní, což se podařilo v takové míře, že by systém mohl být nasazen pro veřejné použití. Mezi nejzákladnější funkce, které mohou uživatelé používat, se řadí například zakládání kulturních institucí, přidávání exponátů nebo vytváření překladů informačních popisků.

Cíle práce byly tedy splněny v celém rozsahu. Komentovaný a přehledně strukturovaný kód také umožňuje případný další rozvoj projektu. Jedním z možných rozšíření by mohlo být například umožnění uživatelům přidávat komentáře k přeloženým textům.

Přehled zkratek

- **API** - Application Programming Interface - rozhraní pro komunikaci aplikací
- **CSRF** - Cross-Site Request Forgery - typ útoku na internetové aplikace
- **CSS** - Cascading Style Sheets - jazyk popisující způsob zobrazování
- **CSV** - Comma-Separated Values - souborový formát
- **DOM** - Document Object Model - rozhraní pracující s XML nebo HTML dokumenty jako se stromovou strukturou
- **FTP** - File Transfer Protocol - protokol pro přenos souborů
- **HTML** - Hypertext Markup Language - značkovací jazyk
- **HTTP** - Hypertext Transfer Protocol - internetový protokol
- **IT** - Information Ttechnology - informační technologie
- **JPEG** - Joint Photographic Experts Group - formát souborů s obrázky nebo metoda ztrátové komprese
- **JS** - JavaScript - skriptovací jazyk
- **JSON** - JavaScript Object Notation - způsob zápisu dat
- **JSX** - JavaScript XML - rozšíření syntaxe JavaScriptu
- **JWT** - JSON Web Tokens - typ tokenů používaných k autentizaci
- **MEAN** - MongoDB-Express.js-Angular-Node.js - populární sada technologií pro vývoj
- **MERN** - MongoDB-Express.js-React-Node.js - populární sada technologií pro vývoj
- **MVC** - Model-View-Controller - softwarová architektura
- **MVT** - Model-View-Template - softwarová architektura
- **PNG** - Portable Network Graphics - formát souborů s obrázky

- **QR Code** - Quick Response Code - čtvercový strojově čitelný kód s uloženými daty
- **REST** - Representational State Transfer - softwarový architektonický styl
- **SMPT** - Simple Mail Transfer Protocol - internetový protokol pro přenos zpráv elektronické pošty
- **SOAP** - Simple Object Access Protocol - protokol pro výměnu zpráv založený na XML
- **SPA** - Single-Page Application - jednostránková webová aplikace
- **SQL** - Structured Query Language - jazyk pro komunikaci s databází
- **URL** - Uniform Resource Locator - řetězec specifikující umístění zdrojů na internetu
- **WAR** - Web Application Resource - typ souboru
- **WSDL** - Web Services Description Language - jazyk založený na XML pro popis funkcionality webových služeb
- **XML** - Extensible Markup Language - značkovací jazyk
- **XSS** - Cross-Site Scripting - typ útoku na internetové aplikace

Literatura

[1] *Introduction to Angular concepts* [online]. Google, 2022. [cit. 2022/3/15]. Dostupné z: https://angular.io/guide/architecture.

[2] *Angular Licence* [online]. Angular komunita, 2022. [cit. 2022/3/15]. Dostupné z: https://github.com/angular/angular/blob/master/LICENSE.

[3] *Spring Dependency Injection* [online]. VMWare, 2022. [cit. 2022/3/10]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html.

[4] *Spring Boot Licence* [online]. Spring Boot komunita, 2022. [cit. 2022/3/10]. Dostupné z: https://github.com/spring-projects/spring-boot/blob/main/LICENSE.txt.

[5] *Spring Boot Features* [online]. VMWare, 2022. [cit. 2022/3/10]. Dostupné z: https://spring.io/projects/spring-boot.

[6] *Crowdin* [online]. Crowdin, 2021. [cit. 2021/10/08]. Dostupné z: https://crowdin.com.

[7] *Crowdin REST API* [online]. Crowdin, 2021. [cit. 2021/10/08]. Dostupné z: https://support.crowdin.com/api/v2/.

[8] *Crowdin Features* [online]. Crowdin, 2021. [cit. 2021/10/08]. Dostupné z: https://support.crowdin.com/features/.

[9] *Crowdin Licence* [online]. Crowdin, 2021. [cit. 2021/10/08]. Dostupné z: https://crowdin.com/pricing#annual.

[10] *Why did you choose Django?* [online]. Bitbucket, 2019. [cit. 2022/3/6]. Dostupné z: https://web.archive.org/web/20160420214550/https://code.djangoproject.com/wiki/DjangoSuccessStoryBitbucket.

[11] *About the Django Software Foundation* [online]. Django Software Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://www.djangoproject.com/foundation/.

[12] *Meet Django* [online]. Django Software Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://www.djangoproject.com.

[13] *What Powers Instagram: Hundreds of Instances, Dozens of Technologies* [online]. Instagram Engineering, 2011. [cit. 2022/3/6]. Dostupné z: https://instagram-engineering.com/what-powers-instagram-hundreds-of-instances-dozens-of-technologies-adf2e22da2ad.

[14] *Why Django?* [online]. Django Software Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://www.djangoproject.com/start/overview/.

[15] *Express* [online]. OpenJS Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://expressjs.com.

[16] *Companies using Express in production* [online]. OpenJS Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://expressjs.com/en/resources/companies-using-express.html.

[17] *Express.js Philosophy* [online]. OpenJS Foundation, 2022. [cit. 2022/3/8]. Dostupné z: https://github.com/expressjs/express.

[18] *Express Licence* [online]. OpenJS Foundation, 2022. [cit. 2022/3/6]. Dostupné z: https://github.com/expressjs/express/blob/master/LICENSE.

[19] *What is Free Software?* [online]. Free Software Foundation, 2021. [cit. 2021/10/08]. Dostupné z: https://www.gnu.org/philosophy/free-sw.en.html.

[20] *Google Trends* [online]. Google, 2022. [cit. 2022/3/8]. Dostupné z: https://trends.google.com/trends/?geo=CZ.

[21] *Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC* [online]. AltexSoft, 2020. [cit. 2021/12/14]. Dostupné z: https://www.altexsoft.com/blog/soap-vs-rest-vs-graphql-vs-rpc/.

[22] *GraphQL* [online]. GraphQL Foundation, 2021. [cit. 2021/12/14]. Dostupné z: https://spec.graphql.org/draft/.

[23] *MediaWiki is a collaboration and documentation platform brought to you by a vibrant community.* [online]. MediaWiki, 2021. [cit. 2021/10/09]. Dostupné z: https://www.mediawiki.org/wiki/MediaWiki.

[24] *Copyright* [online]. MediaWiki, 2021. [cit. 2021/10/09]. Dostupné z: https://www.mediawiki.org/wiki/Manual:Copyright.

[25] *MediaWiki feature list* [online]. MediaWiki, 2021. [cit. 2021/10/09]. Dostupné z: https://www.mediawiki.org/wiki/Manual:MediaWiki_feature_list.

[26] *What is MediaWiki?* [online]. MediaWiki, 2021. [cit. 2021/10/09]. Dostupné z: https://www.mediawiki.org/wiki/Manual:What_is_MediaWiki%3F.

[27] *Pootle Documentation* [online]. Pootle, 2021. [cit. 2021/10/08]. Dostupné z: http://docs.translatehouse.org/projects/pootle/en/stable-2.8.x/index.html#.

[28] *Discover Pootle* [online]. Pootle, 2021. [cit. 2021/10/08]. Dostupné z: https://pootle.translatehouse.org/discover.html.

[29] *License* [online]. Pootle, 2021. [cit. 2021/10/08]. Dostupné z: http://docs.translatehouse.org/projects/pootle/en/stable-2.8.x/license.html.

[30] *Rails Licence* [online]. Rails komunita, 2022. [cit. 2022/3/10]. Dostupné z: https://github.com/rails/rails/blob/main/MIT-LICENSE.

[31] *Rails* [online]. Rails komunita, 2022. [cit. 2022/3/10]. Dostupné z: https://rubyonrails.org.

[32] *Welcome to Rails* [online]. Rails komunita, 2022. [cit. 2022/3/10]. Dostupné z: https://api.rubyonrails.org.

[33] *Virtual DOM and Internals* [online]. Meta, 2022. [cit. 2022/3/16]. Dostupné z: https://reactjs.org/docs/faq-internals.html.

[34] *React* [online]. Meta, 2022. [cit. 2022/3/16]. Dostupné z: https://reactjs.org.

[35] *Introducing JSX* [online]. Meta, 2022. [cit. 2022/3/16]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html.

[36] *React Licence* [online]. React komunita, 2022. [cit. 2022/3/16]. Dostupné z: https://github.com/facebook/react/blob/main/LICENSE.

[37] *About Ruby* [online]. Ruby komunita, 2022. [cit. 2022/3/10]. Dostupné z: https://www.ruby-lang.org/en/about/.

[38] *The structure of a SOAP message* [online]. IBM, 2021. [cit. 2021/12/14]. Dostupné z: https://www.ibm.com/docs/en/integration-bus/10.0?topic=soap-structure-message.

[39] *Why Spring?* [online]. VMWare, 2022. [cit. 2022/3/10]. Dostupné z: https://spring.io/why-spring.

[40] *Features* [online]. Translation Toolkit, 2021. [cit. 2021/10/09]. Dostupné z: http://docs.translatehouse.org/projects/translate-toolkit/en/latest/features.html.

[41] *Translation File Formats* [online]. Translation Toolkit, 2021. [cit. 2021/10/09]. Dostupné z: http://docs.translatehouse.org/projects/translate-toolkit/en/latest/formats/index.html.

[42] *Translation Toolkit GitHub* [online]. Translation Toolkit, 2021. [cit. 2021/10/09]. Dostupné z: https://github.com/translate/translate.

[43] *License* [online]. Translation Toolkit, 2021. [cit. 2021/10/09]. Dostupné z: http://docs.translatehouse.org/projects/translate-toolkit/en/latest/license.html.

[44] *It takes a community to translate Twitter* [online]. Twitter, 2011. [cit. 2022/3/6]. Dostupné z: https://blog.twitter.com/en_us/a/2011/it-takes-a-community-to-translate-twitter.

[45] *TypeScript* [online]. Microsoft, 2022. [cit. 2022/3/8]. Dostupné z: https://www.typescriptlang.org.

[46] *Frequently Asked Questions* [online]. Evan You, 2022. [cit. 2022/3/17]. Dostupné z: https://vuejs.org/about/faq.html.

[47] *Vue Dokumentace* [online]. Evan You, 2022. [cit. 2022/3/17]. Dostupné z: https://vuejs.org/guide/introduction.html.

[48] *Vue* [online]. Evan You, 2022. [cit. 2022/3/17]. Dostupné z: https://vuejs.org.

[49] *Vue Licence* [online]. Vue komunita, 2022. [cit. 2022/3/17]. Dostupné z: https://github.com/vuejs/vue/blob/dev/LICENSE.

[50] *Ceník Weblate* [online]. Weblate, 2021. [cit. 2021/10/10]. Dostupné z: https://weblate.org/cs/hosting/.

[51] *Vlastnosti Weblate* [online]. Weblate, 2021. [cit. 2021/10/10]. Dostupné z: https://weblate.org/cs/features/.

[52] *Weblate GitHub* [online]. Weblate, 2021. [cit. 2021/10/10]. Dostupné z: https://github.com/WeblateOrg/weblate.

[53] *What is Angular?* [online]. Google, 2022. [cit. 2022/3/15]. Dostupné z: https://angular.io/guide/what-is-angular.

[54] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures* [online]. University of California, Irvine, 2000. [cit. 2021/12/14]. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[55] Mikešová, A. *Analýza existujících mobilních průvodců po muzeích a památkách* [online]. 2021. [cit. 2021/10/05]. Bakalářská práce. Dostupné z: https://otik.uk.zcu.cz/bitstream/11025/44241/1/BP_Mikesova.pdf.

[56] Nilo Mitra, Y. L. *SOAP Version 1.2 Part 0: Primer (Second Edition)* [online]. W3C, 2007. [cit. 2021/12/14]. Dostupné z: https://www.w3.org/TR/soap12-part0.

[57] Reselman, B. *An Architect's guide to APIs: SOAP, REST, GraphQL, and gRPC* [online]. RedHat, 2020. [cit. 2021/12/14]. Dostupné z: https://www.redhat.com/architect/apis-soap-rest-graphql-grpc.

Přílohy

A Instalace a spuštění programu

Pro spuštění aplikace na místním hostiteli musí být na stroji řádně nainstalovaná *Java 11* a *Apache Maven 3.8.4*. Dále je potřeba mít nainstalovaný a spuštěný *MySQL Community Server* verze 8 spolu s příslušným nástrojem *MySQL Workbench*.

Dále je potřeba přejít do kořenové složky s programem serveru a následně do složky `src/main/resources`. Zde se nachází soubor `application.properties`, kde je definována konfigurace programu. V souboru musí být správně nastaveny následující vlastnosti:

- Vlastnost `spring.datasource.url` definuje připojení k databázi, výchozí hodnota musí být upravena tak, aby odpovídala potřebám databázového serveru (je dobré zkontrolovat správnost portu).
- Vlastnosti definující přihlašovací údaje do databáze jsou `spring.datasource.username` a `spring.datasource.password`, tudíž je také potřeba je příslušně změnit.
- V souboru jsou také definovány uživatelské údaje administrátora systému pro komunitní překlad (uživatelské jméno, e-mailová adresa a heslo), pokud výchozí údaje nejsou vhodné, je možné je změnit. Provede se to změnou vlastností `cts.admin.name`, `cts.admin.email` a `cts.admin.password`.

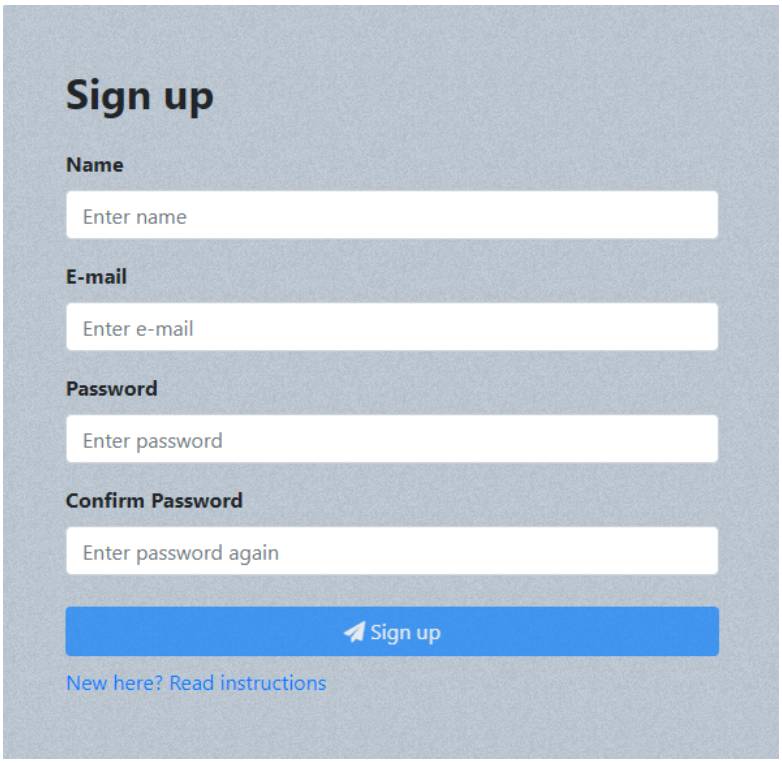
Dále je potřeba přejít do MySQL Workbench a tam na základě uživatele a připojení definovaného v konfiguračním souboru `application.properties` přidat nové schéma pojmenované `museum_guide_db`. Toto schéma musí být založeno s parametry:

- charset: `utf8mb4`,
- collation: `utf8mb4_czech_ci`.

Na závěr stačí v kořenové složce serveru vykonat spouštěcí příkaz `mvn spring-boot:run` a následně si pomocí prohlížeče spustit webovou aplikaci na adrese `http://localhost:8080`.

B Uživatelská příručka

Vše začíná registrací do systému pomocí formuláře zobrazeného na obrázku B.1, na který se lze dostat přes menu v horní části webového rozhraní (záložka *Signup*). Pokud ale uživatel se systémem nikdy předtím nepracoval, měl by si nejprve přečíst informační stránku dostupnou z úvodní stránky nebo z menu (záložka *Info*).



Obrázek B.1: Formulář pro registraci nového uživatele

B.1 Překladatel

Každý zaregistrovaný uživatel se stává překladatelem. Pomocí záložky v menu nazvané *Translate* se může dostat do výběru institucí. Po výběru instituce se zobrazí seznam exponátů. Po výběru jazyka u exponátu může překladatel začít překládat nebo hodnotit ostatní přeložené texty.

Pomocí záložky *My Translations* si překladatel může nechat zobrazit své sekvence překladů. Dále má možnost zobrazit detaily sekvencí, kde může

vidět všechny verze překladu a popřípadě se vracet ke starším verzím, pokud ve svém překladu nalezne chybu nebo se mu z nějakého jiného důvodu bude zdát lepší předchozí verze.

B.2 Správce instituce

Každý uživatel se může stát správcem jedné kulturní instituce. Po kliknutí na záložku *My Institution* může uživatel založit novou instituci. Po odeslání požadavku na její založení může přidat také povolené jazyky, do kterých budou překladatelé překládat.

Detaily instituce jsou po jejím založení opět vidět v záložce *My Institution*. Zde lze změnit její základní údaje, přidat nového správce nebo smazat celou instituci. Mimo to jsou zde také odkazy na přidávání nových jazyků, nových exponátů nebo na zobrazování stávajících exponátů. Po kliknutí na tlačítko pro zobrazení exponátů může být využita důležitá funkce pro vygenerování obrázků s QR kódem.

V menu se pro správce institucí zobrazuje záložka *Approve*, pomocí které si může uživatel nechat zobrazit všechny exponáty své instituce. Po vybrání jazyka může také provést schválení překladu, který se bude zobrazovat v mobilní aplikaci.

B.3 Administrátor systému

Administrátoři systému mají speciální oprávnění spravovat registrované uživatele. Právě proto se jim také zobrazuje v menu záložka *Users*. Poté, co je rozkliknuta, zobrazí se administrátorovi všichni uživatelé bez administrátorské role. Uživatelé se vyskytují v tabulce, která zobrazuje pouze základní údaje. Pro zobrazení dalších je potřeba kliknout na tlačítko označené *Detail*.

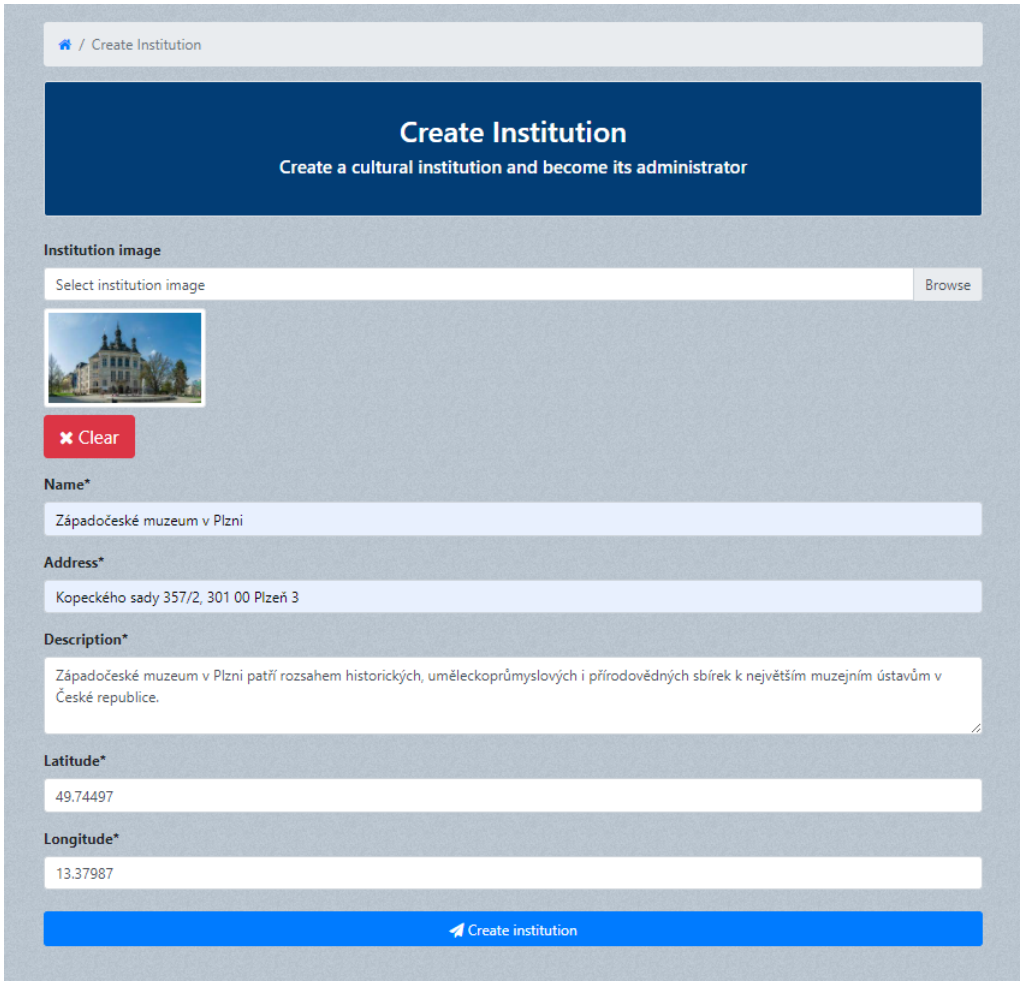
Po zvolení detailů uživatele je administrátorovi zobrazena nová podstránka. Vyskytují se na ní základní údaje o uživateli, z nichž může většinu také měnit. Lze například změnit uživatelské jméno, zaslat uživateli na e-mailovou adresu nově vygenerované heslo, zamezit překladatelská práva nebo rovnou přístup k celému systému. Pokud je prohlížený uživatel správcem instituce, mohou mu být odebrána správcovská práva.

B.4 Ukázky obrazovek

Tato podkapitola si klade za cíl představit ukázky obrazovek webového rozhraní pro klíčové případy užití. Systém byl v době získávání ukázek částečně naplněn testovacími daty a vyobrazené formuláře obsahují možné příklady uživatelského vstupu.

B.4.1 Založení kulturní instituce

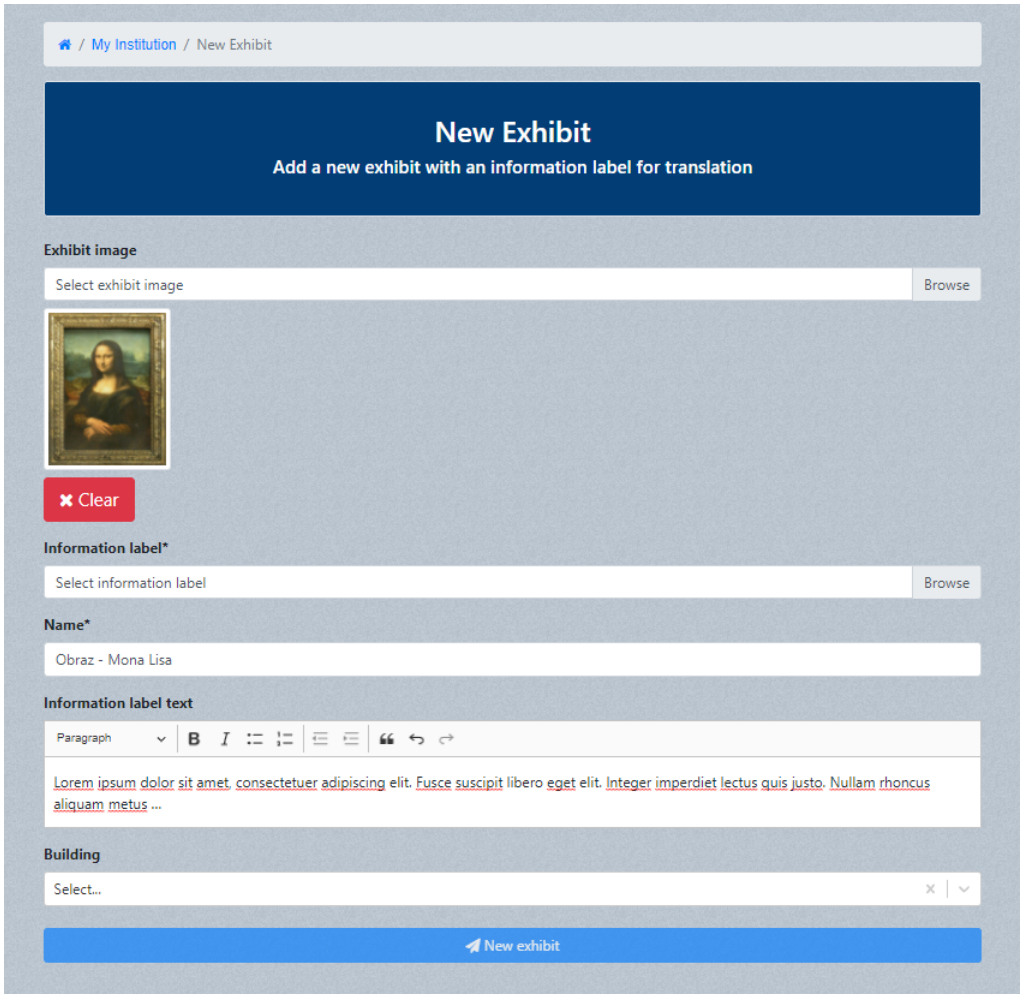
Ukázka formuláře pro založení nové kulturní instituce se vstupy, které by mohly být použity pro Západočeské muzeum v Plzni.



Obrázek B.2: Formulář pro založení kulturní instituce

B.4.2 Založení exponátu

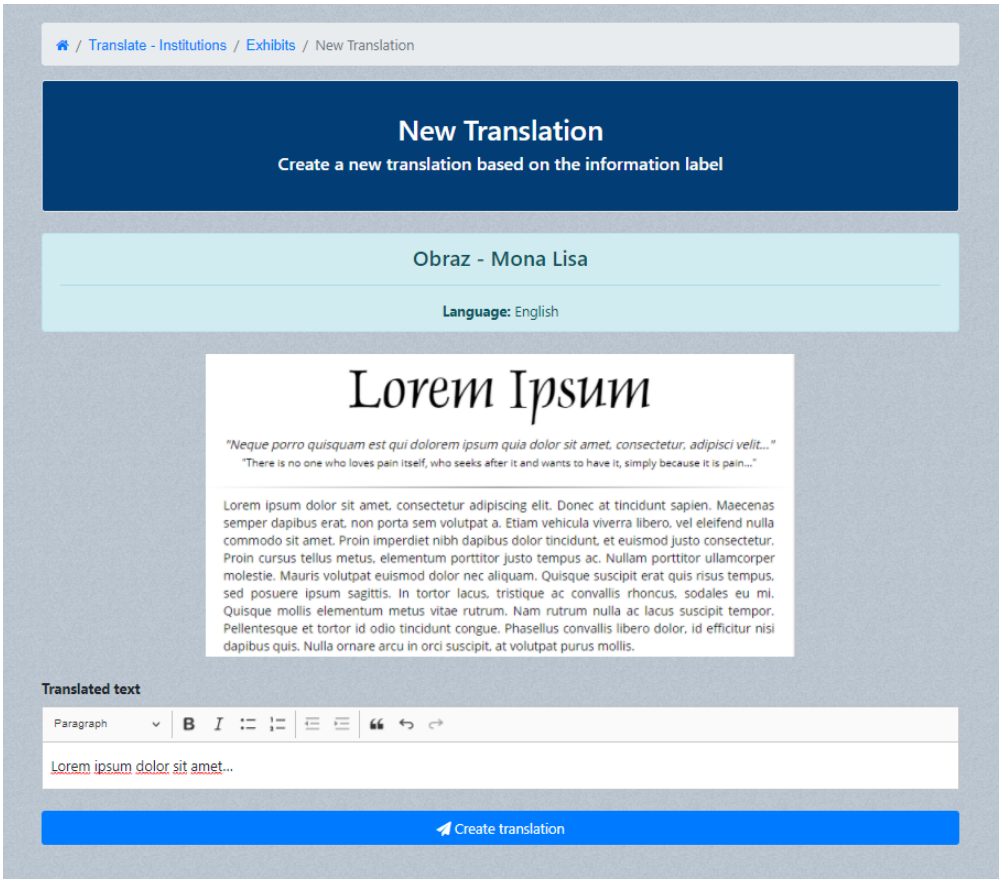
Následující formulář slouží pro založení nového exponátu. Nejdůležitějším atributem tohoto formuláře je fotografie informačního popisku, která ale není uživatelem vybrána, protože se zobrazuje ve větší kvalitě než fotografie exponátu a znepřehledňovala by ukázku. Uživatel může volitelně přidat i umístění exponátu, které slouží pro navigaci návštěvníků.



Obrázek B.3: Formulář pro založení exponátu

B.4.3 Vytvoření překladu

V této ukázce je zobrazen formulář pro vytvoření nového překladu. Překladatel píše do editoru, který mu umožňuje přizpůsobovat text tak, aby co nejvíce odpovídal informačnímu popisku na fotografii. Může například vytvářet odrážky nebo zvýrazňovat text.



Obrázek B.4: Formulář pro vytvoření překladu

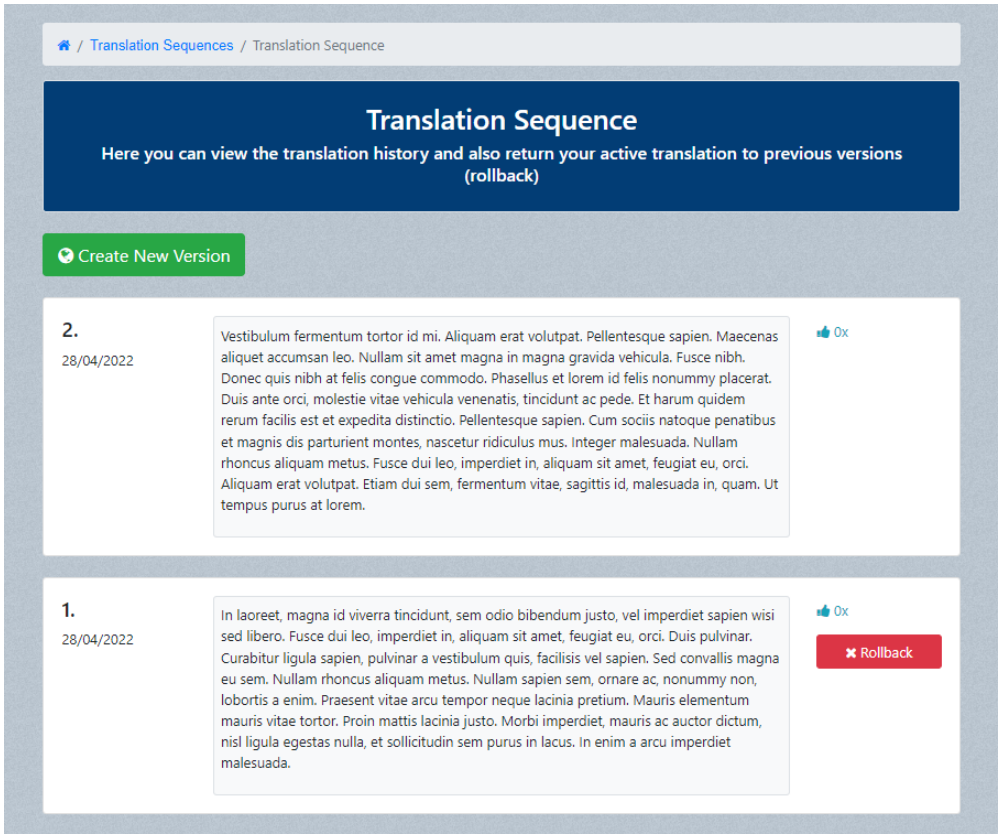
B.4.4 Schválení překladu

V této ukázce je vidět část stránky, kde může správce instituce schvalovat oficiální překlady. Nápovědou mu při této činnosti může být počet překladatelů, kteří označili překlad jako vydařený pomocí tlačítka s palcem nahoru.

Obrázek B.5: Schvalování oficiálních překladů

B.4.5 Správa verzí překladu

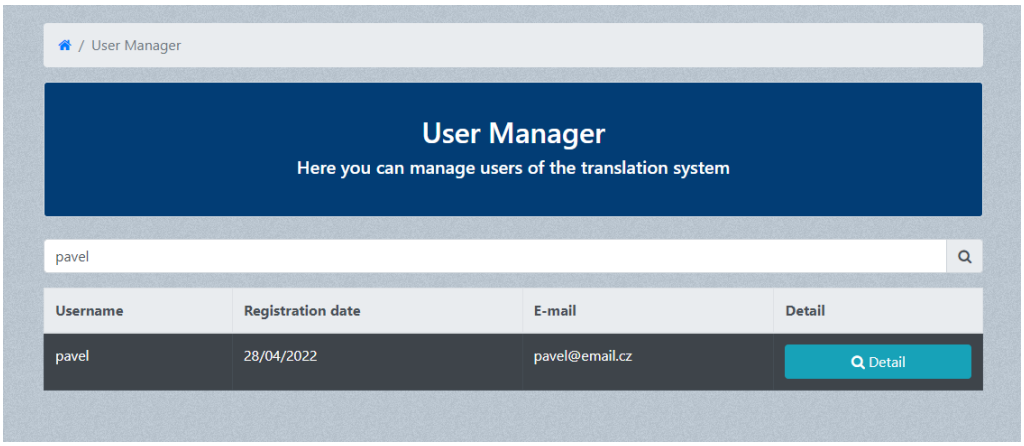
Tato ukázka zobrazuje historii verzí překladu spjatého s jedním exponátem a jazykem. Překladateli je zde umožněno vracet se k předchozím verzím pomocí tlačítka *Rollback*.



Obrázek B.6: Historie verzí překladu

B.4.6 Správa uživatelů

V poslední ukázce je vidět prohledávatelná tabulka, pomocí které může administrátor systému najít uživatele a stisknutím tlačítka *Detail* si nechat zobrazit jeho detaily. V detailech jsou následně zahrnuty akce jako blokování přístupu do systému nebo odebírání překladatelských práv. V tabulce byl pomocí vyhledávacího pole vyhledán uživatel *pavel*.



Obrázek B.7: Tabulka s uživateli určená pro jejich správu administrátorem

C Popis adresářové struktury odevzdávaného souboru

- `Aplikace_a_knihovny`
  - adresář `server`
    - adresář `docs` - dokumentace ke zdrojovému kódu serveru
    - adresář `jar` - obsahuje soubor `server.jar`
    - adresář `src` - obsahuje zdrojové kódy serveru
    - soubor `README.md` - popis spuštění serveru
  - adresář `webove_rozhrani`
    - adresář `docs` - dokumentace ke zdrojovému kódu webového rozhraní
    - adresář `src` - obsahuje zdrojové kódy webového rozhraní
    - soubor `README.md` - popis spuštění webového rozhraní
  - soubor `Readme.txt` - obsahuje popis jednotlivých částí odevzdávané aplikace
- `Text_prace`
  - adresář `src` - všechny zdrojové soubory písemné části BP
  - `A19B0022P_text_prace.pdf` - písemná část BP
- `Readme.txt` - popis adresářové struktury

D Zprávy z testování

D.1 Zpráva testera A

Úvodní strana webového klienta je jednoduchá, přehledná a líbivá. Po přihlášení však nemusí být úplně jasné a intuitivní, jak přidat nový exponát nebo nový překlad. Po prvním průchodu celého webu už ale bylo zřejmé, kde co najít. Při přidání jazyka k instituci mi nevyhovuje vybírání jazyků. Pokud kliknu na vyhledávací pole, nezobrazí se mi ani jeden jazyk. Čekal bych ale, že se zobrazí všechny. Přidávání a odebírání exponátů, budov i místností je díky barevnému odlišení přidání a odebrání přehledné. Další výhodou je stejný styl formulářů. Při výběru překladu je přehledně vidět, který překlad je aktuálně oficiální. Celkově má aplikace jednoduchý design. Nejsou zde žádné rušivé barevné skoky a vybraná modrá barva se zároveň hodí ke zvolenému pozadí. Kromě nedostatku u výběru jazyků, tak v aplikaci funguje vše správně a práce s ní je bez problémů. Svůj účel aplikace plní správně.

D.2 Zpráva testera B

Obsahem této zprávy je mé zhodnocení webové aplikace pro komunitní překlad. Po otevření stránky se mi zobrazila hezká úvodní obrazovka. Následně jsem se bez problémů zaregistroval do systému, ale nebyl jsem si úplně jistý, co mám dělat dále. Postrádal jsem nějaký návod a musel jsem se proto chvíli s aplikací seznamovat. Po prozkoumání aplikace jsem vytvořil několik pokusných překladů. Při jejich vytváření se mi líbila možnost upravovat vzhled textu v editoru. Potom jsem přešel k založení vlastního muzea a přidal jsem si do něj exponáty. Líbilo by se mi mít možnost přidat také plánek muzea a umožnit vkládání více obrázků k informačnímu popisku - občas jsou z prostorových důvodů rozděleny na dvě části.

V celkovém hodnocení se mi ale aplikace zdá užitečná a splňuje svůj účel. Místy se mi nelíbil design (například nevycentrovaný překládaný popisek), ale zato je aplikace dost rozsáhlá.

D.3 Zpráva testera C

D.3.1 Vizuální stránka

1. Karty v dolní části hlavní stránky mají rozdílné velikosti, což působí vizuálně trošku divně. Šlo by je udělat všechny stejně vysoké.

    - Barvy textu silně splývají s barvou pozadí. Mohlo by působit špatně pro barvoslepé lidi.

2. Zvolil bych větší font písma vzhledem k tomu, že uživatelem aplikace mohou být i starší lidé, kteří by mohli hůře vidět.

3. Zafixovaná patička působí agresivně. Je úplně zbytečně stále na očích, ačkoli v sobě nenese žádnou důležitou funkcionalitu pro uživatele a zabírá tak zbytečně místo a částečně překrývá prvky. Rozhodně by bylo vhodné ji nechat na spodku celé stránky a ne fixně v dolní části obrazovky.

D.3.2 Funkcionální stránka

1. Po přístupu na stránku není zcela jasné, co má nový uživatel udělat. Kam má kliknout, co je jeho úkolem, jaká má práva apod.

2. Při vytváření nových objektů není ošetřena velikost nahrávaného obrázku. Při vybrání obrázku o vysoké velikosti mi nahrání trvalo přes 6 vteřin.

3. Dalším nedostatek je chybějící informační prvek indikující aktuální polohu uživatele - na jaké záložce se nachází. Vhodné by bylo využití například drobečkové navigace.

D.3.3 Shrnutí

Stránka působí poměrně normálním dojmem. Velikým nedostatkem je minimální, až žádné množství informativních prvků, které by říkaly uživateli, co má na stránce dělat, což je veliké mínus. Při vytváření webové aplikace je velice důležité brát ohled na to, aby byl "user experience"co nejlepší. Uživatelé nebudou chtít používat aplikaci, která je pro ně neintuitivní a neví, jak se na ní mají chovat. Musí mít naznačené jasné kroky, které mohou vykonávat.

Z pohledu člověka IT znalého bych se na stránce po kratší době naučil orientovat, avšak i tak mám stále malé problémy pochopit, jaký je účel některých stránek aplikace.

Nicméně, z pohledu vývojáře se aplikace zdá být stabilní a dobře ošetřena na nevalidní vstupy. Útok XSS je znemožněn a SQL injekce také. Aplikaci bych hodnotil z vizuální stránky 6ti body z 10. Logika aplikace se zdá být v pořádku a aplikace nejspíše zcela splňuje svůj účel (10 z 10).

D.4 Zpráva testera D

Provedla jsem testování webové aplikace systému pro komunitní překlad. Prvotní orientace v programu byla pro mě velmi náročná. Až po delším pohybu na stránce jsem se dovedla zorientovat. Nejprve jsem vytvořila vlastní instituci a začala přidávat exponáty s popisky k překladu. Potěšila mě možnost přidávání obrázků k muzeím a exponátům. Celou stránku hezky oživují. Následně jsem provedla překlad několika popisků, abych viděla, jak lze vybírat oficiální překlady. Poté, co jsem si vyzkoušela krátkou práci v programu, bylo již snazší se v programu orientovat. Zároveň by to mohlo být jednodušší také tím, že by webové rozhraní bylo i v české verzi. Případně by mohly být přidány i jiné jazykové mutace. Vizuálně působí aplikace velmi příjemně, barevně je dobře nastavená. Grafické zpracování je na vysoké úrovni.

D.5 Zpráva testera E

Webová aplikace poskytuje možnost překládat popisy exponátů muzeím a jiným podobným místům zaměřeným na předání informační hodnoty. Dle obsahu je portál určen především překladatelům a zaměstnancům kulturních institucí. Na domovské stránce je uživatel stručně seznámen s jeho hlavním účelem. Uživatel se pro vstup do aplikace musí registrovat nebo přihlásit. V registračním formuláři jsou vstupní data ošetřena a nelze zadat slabé heslo nebo neplatnou emailovou adresu. Webová stránka je responzivní, tudíž se správně zobrazuje pro různá rozlišení obrazovky a funguje i v mobilních zařízeních. Do aplikace také nelze vložit dva uživatele se stejným uživatelským jménem. Celý proces vytváření instituce není příliš intuitivní a přehledný. Uvítal bych nějaké informační popisy. Po vytvoření instituce má uživatel možnost aktivně ji spravovat. Je možnost členit instituci na budovy, místnosti a vitríny. Exponáty lze do těchto vytvořených lokací přiřazovat. Při editaci místností není možné vrátit se k nadřazené budově. Musí se opět skrze menu nebo tlačítkem zpět v prohlížeči. Tento problém provází celé rozhraní webové aplikace a je velmi nepříjemný. Při nepozornosti tak může uživatel ztratit orientaci, kde přesně se nachází. Správce instituce má možnost překlady schvalovat. Bylo by dobré přidat i záporné hodnocení překladů.

Popisovaná webová aplikace poskytuje ucelený nástroj pro správu exponátů ve vytvořených institucích. Hlavním smyslem je zejména překlad popisů exponátů do cizích jazyků. Do budoucna portál poskytuje řadu dalších možností pro jeho zlepšení, kterými jsou například stránkování nebo filtrování výsledků.

D.6 Zpráva testera F

Hodnocená webová aplikace je určena pro dobrovolné překladatele a správce muzeí. Svůj účel plní dostatečně a je z programátorského hlediska stabilní. Musím ocenit volbu moderního SPA přístupu, protože díky tomu je aplikace velmi rychlá a načítání nových stránek je velmi plynulé. Často používané točivé indikátory průběhu také dodávají stránce dynamiku a jsou uživatelsky přívětivé. Líbila se mi možnost mít pro jedno muzeum více správců. Díky tomu je starost se schvalováním překladů rozdělena mezi více lidí. Co se mi na schvalování překladů ale nelíbilo bylo to, že správce nemá moc informací, jak zjisti, který překlad je nejlepší. Je zde pouze možnost dávat "palec nahoru". Mimo to by bylo dobré například také umožnit komentáře k překladům. V celé aplikaci se vyskytuje mnoho obrázků a to dělalo aplikaci o dost zajímavější. Chyběla mi pouze možnost přidat uživatelský profilový obrázek (ten by se také dobře uplatnil při případném doplňování komentářů). Aplikace podporuje širokou funkcionalitu. U formulářů mi chyběla indikace, které všechny atributy jsou povinné.

Celkově aplikaci mohu doporučit. Její úspěch bude záviset zejména na aktivitě dobrovolných překladatelů a zájmu zaměstnanců muzeí zapojit se do projektu.