Systém pro správu a poskytování lokalizačních textů aplikacím

Bc. Lukáš Šiška

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Lukáš Šiška**
Osobní číslo:	**A21179**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Systém pro správu a poskytování lokalizačních textů aplikacím**
Téma práce anglicky:	**System for Managing and Providing Localization Texts for Applications**

Zásady pro vypracování

1. Proveďte rešerši existujících řešení a zdůvodněte potřebu vytvořit vlastní systém.
2. Popište vývojový framework Phalcon a porovnejte jej s alternativními frameworky.
3. Přibližte problematiku lokalizačních textů používaných pro lokalizaci aplikací.
4. Navrhněte systém pro řešení vybraného problému.
5. Implementujte systém dle návrhu řešení a vytvořte dokumentaci.
6. Popište proces nasazení implementovaného systému a ověřte funkčnost testovacím provozem.
7. Vypracujte závěr a navrhněte doporučení pro budoucí rozvoj systému.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Phalcon Documentation [online]. Phalcon Team, 2022 [cit. 2022-11-26]. Dostupné z: https://docs.phalcon.io.
2. HINZ, Yurek K. Exploring Open Source Software Localization Methods. Lambert Academic Publishing, 2011. ISBN 978--3844399738.
3. ZANDSTRA, Matt. PHP 8 Objects, Patterns, and Practice. 6th ed. New York: Apress, 2021. ISBN 978-1484267905.
4. KRUG, Steve. Don't make me think, revisited. 3rd ed. San Francisco: New Riders, 2013. ISBN 978-0321965516.
5. UNHELKAR, Bhuvan. Software Engineering with UML. Boca Raton: Auerbach Publications, 2017. ISBN 978--0367657383.
6. GKATZIOURAS, Emmanouil. A Developer's Essential Guide to Docker Compose. Birmingham: Packt Publishing, 2022. ISBN 978-1803234366.

Vedoucí diplomové práce: **Ing. Bc. Pavel Vařacha, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 12.05.2023

Lukáš Šiška, v. r.
podpis studenta

ABSTRAKT

Diplomová práce se zabývá návrhem a implementací systému pro správu a poskytování lokalizačních textů aplikacím. Systém je implementován primárně pomocí jazyků PHP a TypeScript za použití frameworků Phalcon a Angular. V rámci teoretické části je provedena analýza existujících řešení a přiblížena problematika lokalizace aplikací. Dále je část věnována samotnému frameworku Phalcon a srovnání s alternativními PHP frameworky. Cílem praktické části je vytvoření návrhu systému a následná implementace, včetně popisu možností nasazení vzniklého systému.

Klíčová slova: Lokalizace aplikací, překlady, webová aplikace, PHP framework, Phalcon, Docker

ABSTRACT

The diploma thesis deals with the design and implementation of a system for managing and providing localization texts for applications. The system is implemented primarily using PHP and TypeScript languages using the Phalcon and Angular frameworks. In the theoretical part, the existing solutions are analyzed, as well as an explanation of application localization. Furthermore, the section is dedicated to the Phalcon framework itself and comparison with alternative PHP frameworks. The aim of the practical part is to create a system design and its implementation, including a description of the possibilities of its deployment.

Keywords: Application localization, translations, web application, PHP framework, Phalcon, Docker

Tímto bych rád bych poděkoval mému vedoucímu Ing. Pavlu Vařachovi, Ph.D. za odborné vedení, rady a vstřícnost při konzultacích.

„SIC PARVIS MAGNA – Veliké věci přicházejí z těch malých“.

Sir Francis Drake

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Hlavním tématem diplomové práce je návrh a implementace systému pro správu a poskytování lokalizačních textů aplikacím.

V dnešní době, kdy většina aplikací, jak už webových, mobilních či i systémových, podporuje několik jazykových mutací, je využívání lokalizačních textů klíčovou součástí vývoje aplikací. Proces tvorby těchto textů a jejich překladů může být rozšířen o spolupráci s externím subjektem, poskytující překlad textů do dalších jazyků. Problémem této spolupráce může být nedostatek znalostí o překládaném systému, respektive v jakém kontextu se daný překlad bude používat, čímž mohou vznikat nevalidní překlady.

Jedním z požadavků klientů, pro něž se aplikace vytvářejí, může být také poskytnutí možnosti úpravy překladů bez potřeby zásahu vývojáře dané aplikace, ideálně i v reálném čase. K dosažení tohoto požadavku by systém měl poskytovat přívětivé uživatelské prostředí, v rámci kterého lze jednoduše rozpoznat, v jakém kontextu je konkrétní překlad využíván.

Počátečním úkolem této práce je analýza již existujících řešení, na jejímž základě je zdůvodněna potřeba vytvoření vlastního řešení. Součástí je též přiblížení problematiky lokalizace aplikací, doplněné o popis PHP frameworku Phalcon, jež má být základním stavebním kamenem vytvářeného systému.

V návaznosti na získané teoretické znalosti je proveden návrh systému a jeho následná implementace, včetně vytvoření uživatelské dokumentace. Důležitou součástí je také návrh a popis procesu nasazení výsledného systému s důrazem na jednoduchost.

Hlavním cílem je tedy vytvoření základní verze funkčního a lehce doručitelného systému, jež může být v budoucnu rozšiřován o další funkcionality, přispívající uživatelskému komfortu, čemuž je v závěru práce věnována samostatná kapitola.

I. TEORETICKÁ ČÁST

1 ANALÝZA

Kapitola je primárně zaměřena na analýzu a následné zhodnocení již existujících řešení, řešící problematiku správy a poskytování lokalizačních textů aplikacím a jiným systémům. Pro zhodnocení řešení jsou vytipovány oblasti pokrývající potřebné funkcionality, co se týče práce s překlady a jejich následné distribuce.

Závěrem kapitoly je na základě získaných poznatků z analýzy existujících řešení a vlastních uvážení, vytvořeno shrnutí, popisující potřebu vytvoření vlastního systému pro řešení daného problému.

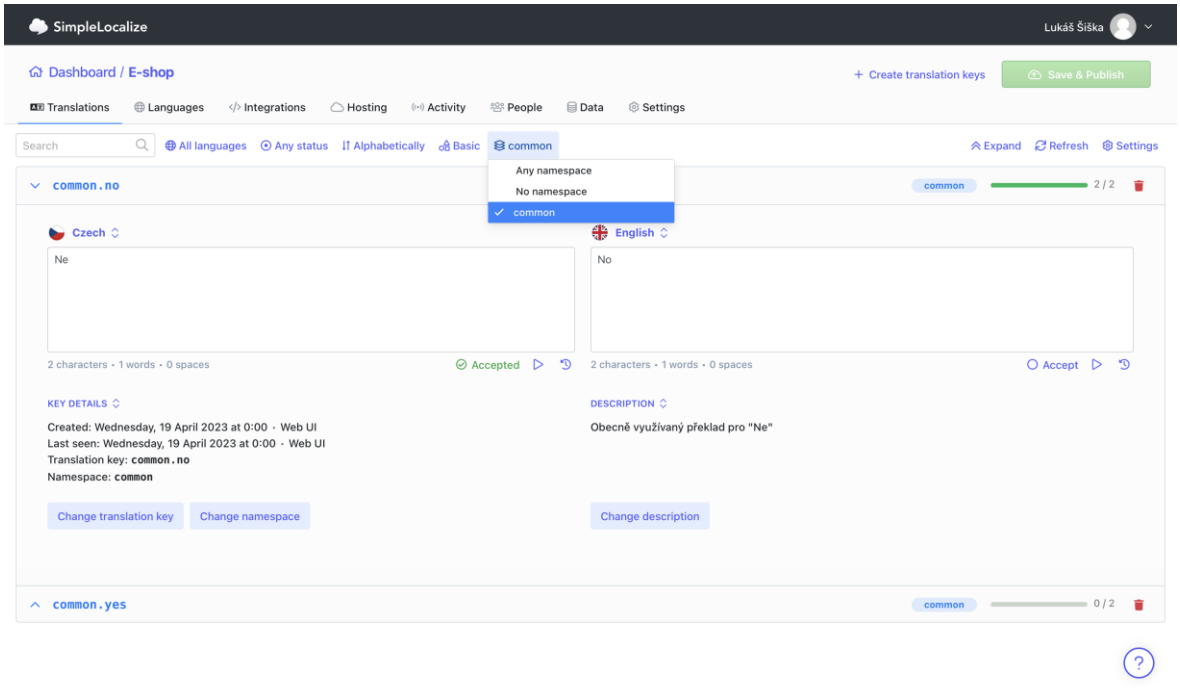
1.1 Analýza existujících řešení

V rámci této podkapitoly je provedena analýza nejpoužívanějších existujících řešení, sestávající z popisu konkrétního řešení, vyzdvihnutí potencionálních výhod či nevýhod a finálního zhodnocení v následujících oblastech:

1. **Strukturovanost** – vytváření složitějších struktur pro definování překladů (například rozdělení lokalizačních klíčů do logicky souvisejících skupin)
2. **Podpora pluralizace** – definování rozdílných překladů na základě určitého počtu (například „*V nákupním košíku nemáte žádnou položku*“, „*V nákupním košíku máte jednu položku.*“, „*V nákupním košíku máte více položek.*“)
3. **Verzování překladů** – zobrazení veškeré historie překladů konkrétního klíče a možnost obnovení vybrané původní verze
4. **Dodatečné informace** – ukládání dodatečných informací k překladům, jako je popis, případně i obrázky, například s ukázkou přesného místa, kde je daný překlad nebo skupina překladů použita
5. **Uživatelské rozhraní** – uživatelská zkušenost při práci s překlady a samotným systémem
6. **Distribuce překladů** – finální distribuce vytvořených překladů aplikacím a jiným systémům
7. **Cena a omezení** – možnosti předplatného, které vybrané řešení nabízí a jak se mění v závislosti s rostoucím počtem aplikací/překladů + omezování funkcionality systému (maximální počet aplikací/projektů, překladů, exkluzivní funkcionalita atd.) v kombinaci s cenou konkrétního předplatného

1.1.1 SimpleLocalize

Webová aplikace *SimpleLocalize*, dostupná na adrese *simplelocalize.io* [27], umožňuje udržovat překlady mezi členy týmu a jejich softwarovými produkty. Pro distribuci překladů aplikace nabízí využití mimo klasický export do souboru různých formátů, také jimi hostovaný CDN pro konkrétní jazyky daného projektu. Uživatelé mohou na projektech kolaborovat ve třech rolí, konkrétně *Vlastník*, *Správce* a *Překladatel*. Aplikace nabízí mnoho zajímavých doplňujících funkcionalit, jako je například automatický překlad chybějících překladů pomocí nástrojů *DeepL* nebo *Google Translate*, integrace nejen s vývojovými prostředími a také různé možnosti nastavení uživatelského rozhraní při práci s překlady pro lepší uživatelskou zkušenost. Mimo to je dostupná také funkcionalita importu překladů z vloženého souboru podporovaného formátu. Jako jedna z mála aplikací umožňuje lehce definovat specifické překlady pro konkrétní zákazníky.



Obrázek 1 Ukázka aplikace SimpleLocalize [27]

Následující část je věnována zhodnocení této analyzované aplikace v oblastech uvedených v úvodu kapitoly *1.1 Analýza existujících řešení*.

Strukturovanost

Lokalizační klíče lze přiřazovat do tzv. jmenných skupin (*namespace*), které jsou reprezentovány pouhým textem. Na základě těchto skupin lze po výběru přesně jedné skupiny provádět filtrování všech klíčů konkrétního projektu. Tyto skupiny působí jako pouhé označení klíče a nedochází tak k jeho automatickému prefixování na základě konkrétní skupiny či možnosti hlubšího logického zanoření v rámci více skupin. K dosažení zanoření klíče pod určitý logický celek ve výsledném exportu, je potřeba mezi název tohoto celku a klíče vložit tečku. Jakmile systém při exportu do formátu JSON narazí na takový klíč, dojde automaticky k jeho zanoření pod uvedený logický celek.

Podpora pluralizace

Aplikace nenabízí rozhraní pro definování rozdílných překladů na základě číselné hodnoty. Tuto funkcionalitu lze simulovat například vytvořením více lokalizačních klíčů anebo uvedením specifického formátu v textové podobě hodnoty překladu.

Verzování překladů

Při provádění změn překladů se uchovává jejich kompletní historie s možností obnovení některé z původních verzí. Mimo verze překladů sleduje aplikace veškeré změny, které nastaly v rámci daného projektu, a je tak možné sledovat prováděné akce včetně autorů těchto akcí.

Dodatečné informace

K lokalizačním klíčům lze přikládat pouze textový popis. Tento popis může obsahovat odkazy na uložené obrázky, což z části simuluje možnost přímého přikládání obrázků. Mimo to editor zobrazuje tzv. popis kódu, který je automaticky přiložen v případě jeho prezence v importovaném souboru. Popis kódu nelze v editoru upravovat.

Uživatelské rozhraní

Aplikace nabízí propracované uživatelské rozhraní, které je intuitivní a umožňuje tak pohodlnou správu překladů i pro uživatele bez větších technických znalostí. Rozhraní aplikace si může každý uživatel přizpůsobit pomocí několika předdefinovaných nastavení na základě vlastní preference. Těmito nastaveními může být například rozložení samotného rozhraní (tabulkový pohled, sloupcový pohled atd.), velikost polí pro úpravu překladů, zobrazování dodatečných informací o překladu a další.

Distribuce překladů

Pro distribuci překladů nabízí aplikace mnoho možností. Jednou z nich je využití klasického exportu překladů v konkrétním jazyku do souboru vybraného formátu. Zajímavější je možnost využití CDN hostingu, díky němuž není třeba „ručně“ nahrazovat soubor s překlady v koncových aplikacích, ale stačí jej pouze konzumovat z příslušné adresy vždy v nejnovější nasazené verzi. Po změně jakéhokoliv překladu je třeba explicitně nasadit novou verzi CDN souborů. Problémem může být fakt, že veškeré soubory, poskytované ve formě CDN, jsou veřejné. K distribuci překladů lze využít také zabudované REST API či tzv. webhooky, kterými lze reagovat na určité eventy, vyvolané v rámci této aplikace.

Cena a omezení

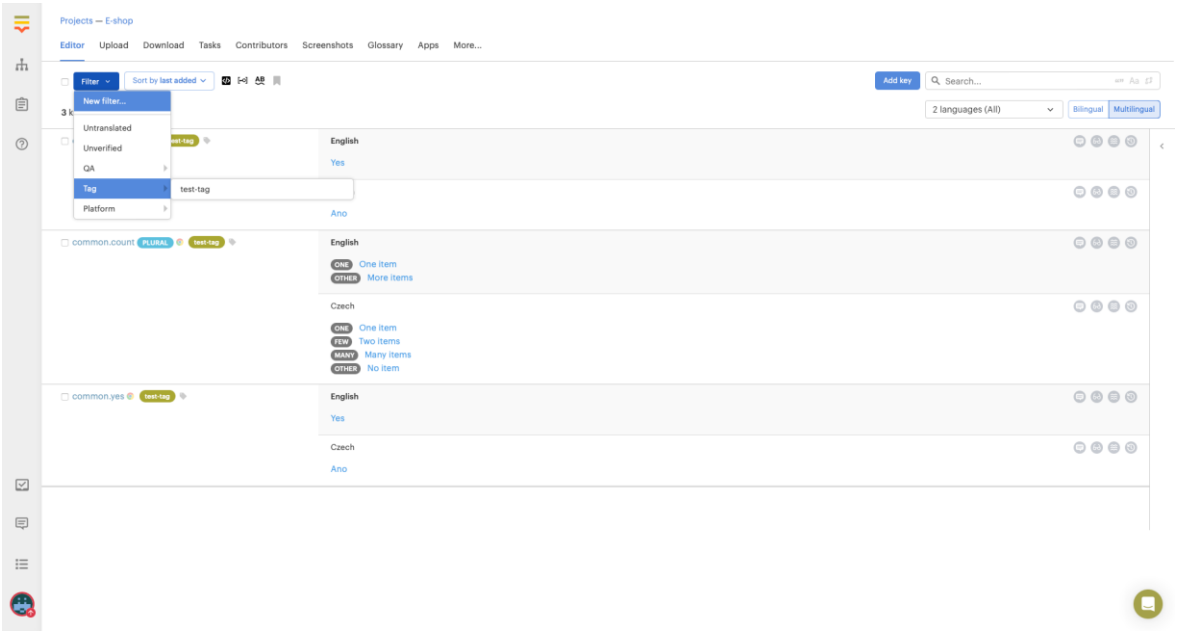
Jsou dostupné 4 verze předplatného, obsahující i plán, který je zcela zdarma. Každé z těchto předplatných nabízí rozdílný maximální počet lokalizačních klíčů, které mohou být v rámci všech projektů pod danou organizací vytvořeny. Aktuálně nejvyšší předplatné v ceně 88 dolarů měsíčně nabízí 12 tisíc lokalizačních klíčů, což může být pro softwarovou firmu spravující větší projekty, omezující. Maximální počet těchto klíčů lze dále navyšovat za předpokladu zvýšené měsíční platby za předplatné. Mimo omezení, co se týče lokalizačních klíčů, je také časově omezena historie prováděných změn či maximální počet uživatelů, kteří mohou do projektů přistupovat.

1.1.2 Lokalise

Webová aplikace *Lokalise*, dostupná na adrese *lokalise.com* [28], umožňuje, obdobně jako aplikace *SimpleLocalize*, udržovat překlady mezi členy týmu a jejich softwarovými produkty. Pro distribuci překladů aplikace nabízí klasický export do souborů různých formátů, případně také využití reakcí na prováděné akce pomocí webhooků. Uživatelé mohou na projektech kolaborovat v rámci nastavených oprávnění pro jednotlivé jazyky. Co se týče automatického překládání textů, je dostupný strojový překlad, případně i zakázkový překlad na základě objednávky. Dostupná je také funkcionalita importu překladů z vloženého souboru podporovaného formátu.

Projekty lze propojit s mnoha dalšími aplikacemi, jak už pro výměnu lokalizačních klíčů a jejich překladů, získávání zdrojových dat (snímky obrazovky, zdrojové soubory s kódem atd.), či využívání různých automatizací.

U projektů lze zadávat uživatelům v daném projektu úkoly, kterými lze v jednoduché formě řídit překlady jednotlivých jazyků, případně kontrolu těchto překladů.



Obrázek 2 Ukázka aplikace Lokalise [28]

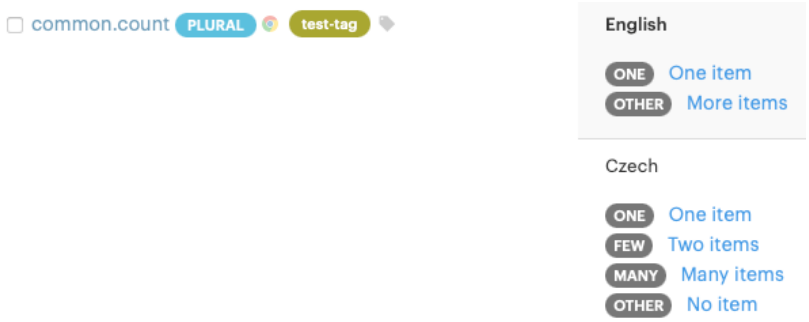
Následující část je věnována zhodnocení této analyzované aplikace v oblastech uvedených v úvodu kapitoly *1.1 Analýza existujících řešení*.

Strukturovanost

Lokalizační klíče lze lépe strukturovat pouze použitím speciální kombinace znaků v názvu klíče. K dosažení zanoření klíče, pod určitý logický celek, je potřeba mezi název tohoto celku a klíče vložit dvě dvojtečky. Jakmile systém při exportu do JSON formátu narazí na takový klíč, dojde automaticky k jeho zanoření pod uvedený celek. Nevýhodou tohoto přístupu je fakt, že takové chování podporuje pouze formát JSON, nikoliv ostatní nabízené formáty. Uživatelské rozhraní není přizpůsobeno pro definování logických celků, ani tomuto formátu.

Podpora pluralizace

Jak lze vidět na obrázku č. 3, aplikace podporuje ukládání překladů pro různé tvary množného čísla v rámci jediného překladového klíče. Aplikace v základu nabízí předem definované tvary (například *One*, *Other* v anglickém jazyku) pro každý jazyk, ale lze si je nadefinovat i samostatně dle vlastních potřeb.



Obrázek 3 Ukázka podpory pluralizace v aplikaci Lokalise [28]

Verzování překladů

Při provádění změn překladů se uchovává jejich kompletní historie s možností obnovení některé z původních verzí. Mimo verze překladů sleduje aplikace veškeré změny, které nastaly v rámci daného projektu a je tak možné sledovat prováděné akce včetně autorů těchto akcí.

Dodatečné informace

K lokalizačním klíčům i samotným překladům lze přidávat komentáře, které případně mohou nahrazovat formu popisu dané jednotky. Hlavní funkcionalitou, co se týče uvádění dodatečných informací, je možnost přidávání snímků obrazovek, podporující interaktivní označování míst, kde se daný překlad bude objevovat, a toto místo následně přiřadit příslušnému lokalizačnímu klíči.

Uživatelské rozhraní

Aplikace nabízí velmi jednoduchý editor pro správu překladů, který lze obsluhovat ve dvou hlavních režimech. Jedním z režimů je zobrazení překladů ve všech jazycích projektu najednou (případně pouze vybrané), včetně možnosti úpravy a dalších doplňujících akcí. Druhým režimem je zobrazení vždy pouze překladu hlavního jazyku (jež se určuje v nastavení projektu) a jednoho vybraného ze všech dostupných v daném projektu.

Distribuce překladů

Distribuci překladů lze zajistit mnoha způsoby. Tím nejpropracovanějším je klasický export překladů do souboru ve vybraném formátu. Tento způsob distribuce nabízí mnoho parametrů k nastavení, které dokážou výstup ovlivnit. Jedná se například o výběr exportovaných překladů na základě v editoru přiřazovaných štítků, možnost rozhodnutí, zda se mají do výstupu vložit i prázdné překlady, formát parametrů v textu nebo množného čísla či samotný

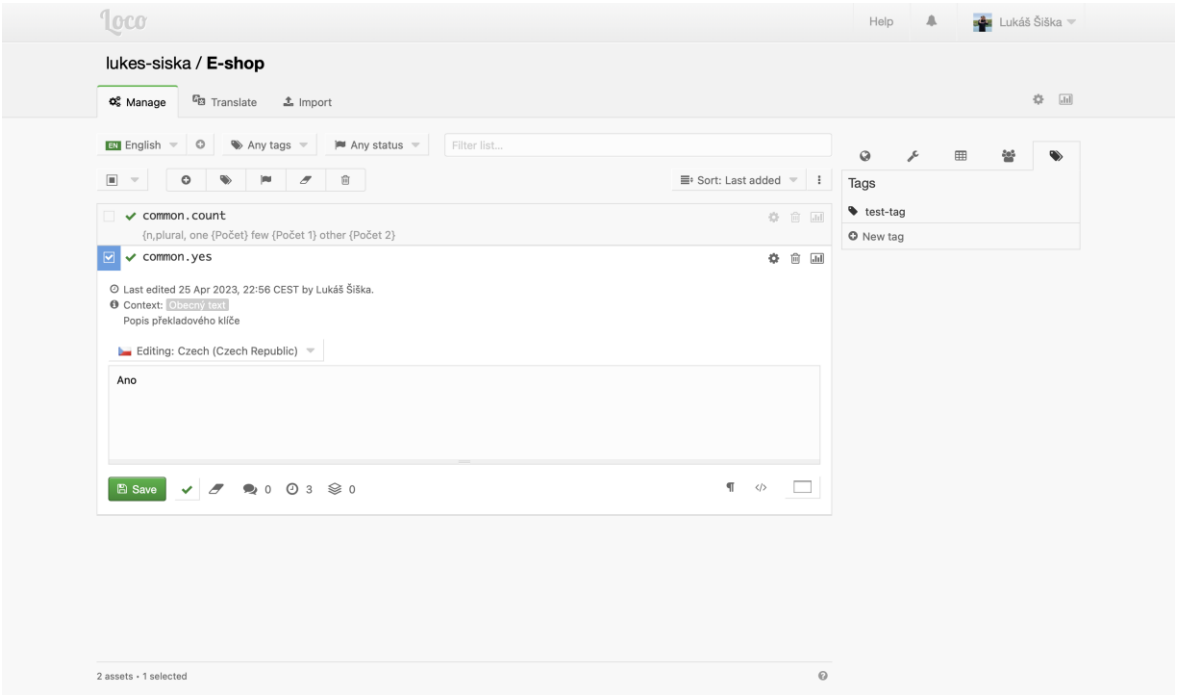
název exportovaného souboru. K distribuci překladů lze využít i tzv. webhooky, kterými lze reagovat na určité eventy, vyvolané v rámci této aplikace.

Cena a omezení

V základu jsou dostupné 4 verze předplatného, obsahující i bezplatný plán. Tento plán je velmi omezen, co se týče jak maximálního počtu lokalizačních klíčů, tak i dostupných funkcionalit (například verzování překladů, přidávání dodatečných informací formou snímků obrazovky atd.). Nejvyšší verze povoluje vytvoření až 30 tisíc lokalizačních klíčů v kombinaci s 15 místy pro kolaborující uživatele v ceně 990 dolarů měsíčně. Tyto limity lze za různé příplatkové ceny navyšovat. Pro větší projekty je dostupný individuální plán s variabilní cenou.

1.1.3 Loco

Webová aplikace *Loco*, dostupná na adrese *lokalise.biz* [29], umožňuje, obdobně jako aplikace *SimpleLocalize* a *Localise*, udržovat překlady mezi členy týmu a jejich softwarovými produkty. Pro distribuci překladů aplikace nabízí klasický export do souborů různých formátů, formou stažení nebo dostupného REST API endpointu. Uživatelé mohou na projektech kolaborovat v rámci nastavených oprávnění pro konkrétní projekt (v bezplatném plánu je dostupná pouze role pro překladatele). Dostupná je také funkcionalita importu překladů z vloženého souboru podporovaného formátu.



Obrázek 4 Ukázka aplikace Loco [29]

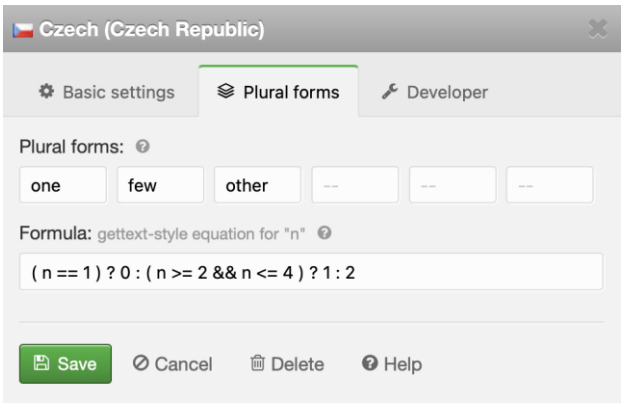
Následující část je věnována zhodnocení této analyzované aplikace v oblastech uvedených v úvodu kapitoly *1.1 Analýza existujících řešení*.

Strukturovanost

Obdobně jako u aplikace Lokalise, lze lokalizační klíče strukturovat použitím speciální kombinace znaků v názvu. K dosažení zanoření klíče, pod určitý logický celek, je potřeba mezi název tohoto celku a klíče vložit tečku. Jakmile systém při exportu do jednoho z formátů, podporující zanoření klíčů, narazí na takový klíč, dojde automaticky k jeho zanoření pod uvedený celek. Klíče je možné rozlišovat také pomocí tagů, dle kterých lze seznam i filtrovat. Uživatelské rozhraní není přizpůsobeno pro definování logických celků, ani tomuto formátu.

Podpora pluralizace

Aplikace umožňuje nastavit přesné zadávání potřebných tvarů množného čísla na úrovni jazyku projektu. Pro všechny jazyky jsou dostupná již předvyplněná nastavení, která lze jednoduše měnit a přizpůsobovat potřebám konkrétního projektu. Na obrázku č. 5 lze vidět základní nastavení pro český jazyk, jež speciální formulí definuje použití jednotlivých tvarů na základě číselné proměnné *n*.



Obrázek 5 Nastavení pluralizace pro konkrétní jazyk v aplikaci Loco [29]

Editor poskytuje dva způsoby definování konkrétních překladů k různým tvarům množného čísla. Jedním z nich je využití ICU syntaxe, kterou je třeba jako první uvést ve zdrojovém jazyku projektu, čímž se pro ostatní přiřazené jazyky zpřístupní speciální formulář k vyplnění konkrétních překladů. Tato syntaxe je následně uložena jako čistý text v hlavním poli překladu klíče. Druhou možností je přiřazování již existujících lokalizačních klíčů k jednotlivým tvarům.

Verzování překladů

Při provádění změn překladů se uchovává jejich kompletní historie s možností obnovení některé z původních verzí. Zde je třeba počítat s omezením maximálního počtu původních verzí k obnovení, jež se odvíjí od aktivního předplatného. Například u předplatného zdarma jsou dostupné pouze dvě nejnovější verze. Mimo verze překladů sleduje aplikace veškeré změny, které nastaly v rámci daného projektu, a je tak možné sledovat prováděné akce včetně autorů těchto akcí.

Dodatečné informace

U lokalizačních klíčů lze ukládat jejich popis, kontext, upřesňující použití klíče v rámci projektu a případně je také provazovat s konkrétními soubory, ve kterých se tento klíč vyskytuje. Mimo to mohou všichni uživatelé, přiřazeni do projektu, přidávat komentáře k jednotlivým překladům. Aplikace neumožňuje přikládání doplňujících obrázků, jako například snímky obrazovek s vyznačeným použitím lokalizačních klíčů.

Uživatelské rozhraní

Správu klíčů a jejich překladů je možné řešit ve dvou hlavních zobrazení. První z nich se zaměřuje primárně na správu klíčů, kde je možné klíče vytvářet, upravovat či mazat. V tomto zobrazení lze také upravovat překlady pro jednotlivé jazyky projektu a případně spravovat i jejich verze. Druhé zobrazení je zaměřeno pouze na překládání veškerých klíčů, kdy je možné vybrat dvojici jazyků – jeden zdrojový a druhý cílový.

Distribuce překladů

K distribuci překladů se zde využívá pouze export do souboru vybraného formátu, který lze po vygenerování stáhnout, případně i konzumovat pomocí dostupného REST API endpointu. Export, mimo široký výběr formátů, nabízí možnost výběru exportovaného jazyku (případně i více v rámci jednoho souboru), filtrování dle tagů, které lze uvádět u lokalizačních klíčů či výběr hodnoty, jež bude považována za klíč ve výsledném souboru. Takovou hodnotou může být zpravidla název překladového klíče nebo samotný překlad, uváděný ve zdrojovém jazyku.

Cena a omezení

V základu jsou dostupné 4 verze předplatného, obsahující i bezplatný plán. Tento plán je velmi omezen, co se týče jak maximálního počtu klíčů a konkrétních překladů, tak i dostupných funkcionalit (například maximální počet verzí překladů k obnovení, omezení

počtu jazyků pro projekt či i počet samotných projektů). Nejvyšší verze povoluje vytvoření až 125 tisíc již konkrétních překladů v ceně 30 dolarů měsíčně. Tento limit lze za různé příplatkové ceny navyšovat.

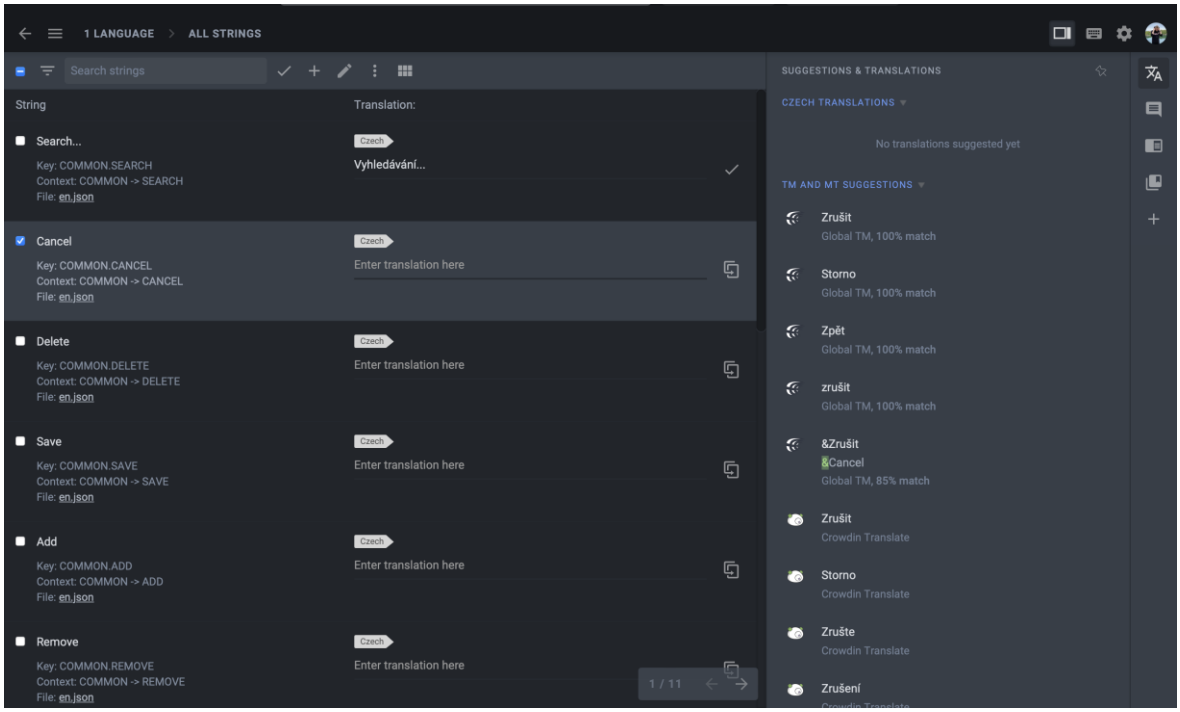
1.1.4 Crowdin

Webová aplikace *Crowdin*, dostupná na adrese *crowdin.com* [30], umožňuje, obdobně jako všechny předchozí aplikace, udržovat překlady mezi členy týmu a jejich softwarovými produkty. V případě této aplikace se při zakládání projektů definuje pevný zdrojový jazyk, ze kterého následně vycházejí všechny další zvolené jazyky. Aplikace působí jako vylepšený grafický editor souboru konkrétního formátu, obsahující překlady. Při prvotním vytvoření projektu je tedy potřeba nahrát, případně vytvořit, soubor se zdrojovými texty.

Pro distribuci překladů aplikace nabízí klasický export, pouze do totožného formátu souboru, ze kterého tyto překlady vycházejí. Případně lze také využít hostované CDN či reakce na prováděné akce pomocí webhooků. Uživatelé mohou na projektech kolaborovat v rámci nastavených oprávnění pro konkrétní projekt. Součástí editoru překladů jsou dostupné různé verze strojových překladů pro právě překládaný text.

Projekty lze propojit s mnoha dalšími aplikacemi, jak už pro výměnu lokalizačních klíčů a jejich překladů, získávání zdrojových dat (snímky obrazovky, zdrojové soubory s kódem atd.), či využívání různých automatizací.

U projektů lze zadávat přiřazeným uživatelům úkoly, kterými lze v jednoduché formě řídit překlady jednotlivých jazyků, případně kontrolu těchto překladů.



Obrázek 6 Ukázka aplikace Crowdin [30]

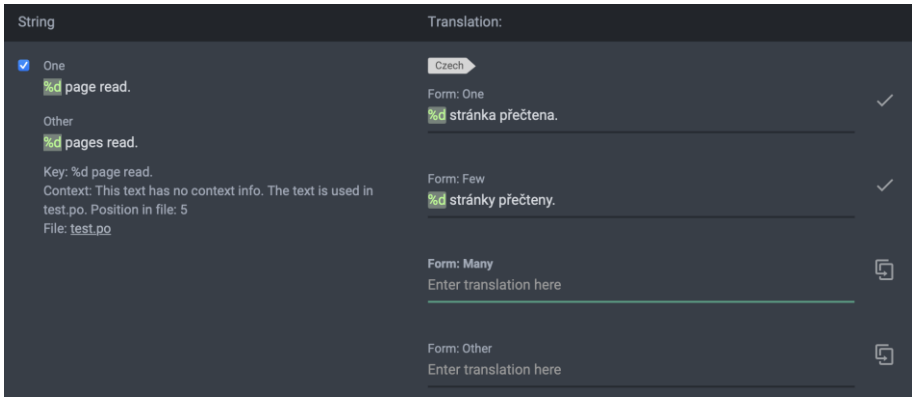
Následující část je věnována zhodnocení této analyzované aplikace v oblastech uvedených v úvodu kapitoly *1.1 Analýza existujících řešení*.

Strukturovanost

K dosažení zanoření lokalizačního klíče, pod určitý logický celek, je potřeba mezi název tohoto celku a klíče vložit tečku. Jakmile systém při exportu do jednoho z dostupných formátů, podporující zanoření klíčů, narazí na takový klíč, dojde automaticky k jeho zanoření pod uvedený logický celek. Klíče je možné rozlišovat také pomocí tagů, dle kterých lze seznam i filtrovat. Uživatelské rozhraní není přizpůsobeno pro definování logických celků, ani tomuto formátu.

Podpora pluralizace

Aplikace v základu nabízí předem definované tvary množného čísla (například *One*, *Other* v anglickém jazyku) pro každý jazyk, ke kterým lze následně přidávat překlady. Tato podpora je dostupná pouze pro formáty přímo podporující pluralizaci (například Gettext). Toto omezení znemožňuje použití u ostatních formátů, jakým je například často využívaný JSON.



Obrázek 7 Ukázka podpory pluralizace v aplikaci Crowdin [30]

Verzování překladů

Při provádění změn překladů se uchovává jejich kompletní historie s možností obnovení některé z původních verzí. Mimo verze překladů sleduje aplikace veškeré změny, které nastaly v rámci daného projektu, a je tak možné sledovat prováděné akce včetně autorů těchto akcí.

Dodatečné informace

U lokalizačních klíčů je možné uvádět kontext, upřesňující použití klíče v rámci projektu. Mimo to, mohou všichni uživatelé, přiřazení do projektu, přidávat komentáře k jednotlivým překladům. Hlavní funkcionalitou, co se týče uvádění dodatečných informací, je možnost přidávání snímků obrazovek s podporou interaktivního označování míst, kde se daný překlad bude objevovat, a toto místo následně přiřadit příslušnému klíči.

Uživatelské rozhraní

Součástí aplikace je propracovaný editor, zaměřující se pouze na překlady klíčů ze zdrojového do cílového jazyku. Mimo zadávání konkrétních překladů nabízí také jejich návrhy z různých zdrojů. Pro správu klíčů je dostupné samotné rozhraní, ve kterém lze tyto klíče přidávat, upravovat, mazat, pouze v kontextu prvně nahraného zdrojového souboru.

Distribuce překladů

Pro distribuci překladů nabízí aplikace několik možností. Jednou z nich je využití klasického exportu překladů v jazyku do souboru totožného formátu s původním zdrojovým souborem. Zajímavější je možnost využití CDN hostingu, díky němuž není třeba „ručně“ nahrazovat soubor s překlady v koncových aplikacích, ale stačí jej pouze konzumovat z příslušné adresy vždy v nejnovější nasazené verzi. Po změně jakéhokoliv překladu je potřeba explicitně nasadit novou verzi CDN souborů. K distribuci překladů lze využít také zabudované REST

API či tzv. webhooky, kterými lze reagovat na určité eventy, vyvolené v rámci této aplikace. Využití CDN a webhooků je dostupné až od placených plánů.

Cena a omezení

V základu jsou dostupné 4 verze předplatného, obsahující i bezplatný plán. Tento plán je velmi omezen, co se týče jak maximálního počtu přeložených slov napříč všemi jazyky (kromě hlavního, resp. zdrojového), tak i dostupných doplňujících funkcionalit či omezení počtu projektů. Nejvyšší verze povoluje v základu překlad až 500 tisíc slov v ceně 450 dolarů měsíčně. Tento limit lze za různé příplatkové ceny navyšovat, čímž dokáže cena překročit i hranici tisíce dolarů. Pro větší projekty je dostupný individuální plán s variabilní cenou.

1.2 Shrnutí

Tato kapitola shrnuje veškeré poznatky, získané v rámci analýzy existujících řešení, dělené do totožných kategorií, jež byly definovány v úvodu kapitoly *1.1 Analýza existujících řešení*. Shrnutí zároveň obsahuje i úvahu nad potřebou vytvoření vlastního řešení, vždy v kontextu zkoumané kategorie.

Strukturovanost

Každá z analyzovaných aplikací řeší možnost zanoření lokalizačních klíčů primárně na úrovni exportu do některého z podporovaných formátů. K rozdělení do skupin se u většiny těchto aplikací používá pouze textových označení, v podobě tagů, díky kterým lze klíče filtrovat.

Pro uživatele příjemnější by mohlo být rozhraní, podobající se vzhledem i chováním adresářové, respektive stromové struktuře.

Podpora pluralizace

Každá ze zmíněných aplikací řeší podporu pluralizace, neboli definování různých tvarů množného čísla, rozdílným způsobem. Většina přizpůsobuje své rozhraní těmto potřebám, ale jsou i takové aplikace, které podporu pluralizace neřeší. Ve výsledku lze ale definování různých tvarů zajistit i u aplikací bez jakékoliv podpory v rámci uživatelského rozhraní, čistě skrze textovou hodnotu. V takovém případě je nutná znalosti určité syntaxe, využívané cílovou aplikací, což může být pro běžné uživatele překážkou.

Ideální by bylo mít pro každý jazyk nadefinované všechny podporované tvary, s možností úpravy jejich množiny. Důležité je také zobrazení těchto tvarů a jejich správa v uživatelském rozhraní aplikace a podpora exportu do všech podporovaných formátů, jež jsou dostupné v dané aplikaci.

Verzování překladů

Všechny analyzované aplikace podporují verzování překladů lokalizačních klíčů. Mimo verzování překladů dokážou některé z těchto aplikací také zaznamenávat historii provedených akcí v rámci konkrétního překládaného projektu.

Tato funkcionalita je často aplikacemi omezována skrze předplatné, jak už na úrovni maximálního počtu udržovaných původních verzí, tak i povolení samotné funkcionality.

Dodatečné informace

U většiny aplikací je možnost přidávat u lokalizačních klíčů pouze textový popis. Tím, že žádná z aplikací nativně nepodporuje dělení klíčů do skupin, není možné přidávat jakékoliv informace právě k těmto logickým celkům. V některých případech je dostupná možnost nahrávání obrázků s podporou interaktivního vyznačování použití klíčů, což z části tyto dodatečné informace supluje.

Pro zlepšení přesnosti překladů může napomoct přidávání dodatečných informací, jak ke klíčům, tak i k jejich skupinám. To může právě vést k většímu pochopení kontextu, v jakém jsou či budou dané klíče využívány.

Uživatelské rozhraní

Každému uživateli může více vyhovovat různé rozhraní aplikace. Ve výsledku by mělo být ale primárně zaměřené na jednoduchost správy překladů pro jednotlivé jazyky. Výrazným zlepšením by mohlo být i dodržení pravidel pro tvorbu tzv. bezbariérového webu.

Distribuce překladů

Nejstěžejnější funkcionalitou všech analyzovaných aplikací je samotná distribuce vytvořených překladů. Mimo export do souboru vybraného formátu jsou poskytovány i další možnosti distribuce, jakými jsou například webhooky, hostování souborů ve formě CDN hostingu nebo integrace s aplikacemi třetích stran.

V základu si většina uživatelů vystačí pouze s exportem do souboru, případně hostovaným souborem formou CDN. U propracovanější integrace, jakou může kupříkladu být vynucení

obnovení webové aplikace v reálném čase po změně překladu, je vhodné poskytovat i další možnosti distribuce – například pomocí dříve zmíněných webhooků.

Cena a omezení

Všechny ze zmíněných aplikací nabízejí různé varianty předplatného. Na základě těchto předplatných jsou omezovány určité funkcionality aplikace, primárně se ale jedná o omezení maximálního počtu možných klíčů, respektive jejich překladů. Ceny těchto předplatných se můžou u některých zkoumaných aplikací vyšplhat až ke stovkám dolarů za měsíc, což může být pro určité skupiny uživatelů či organizací, nepřijatelné. Na trhu existuje i několik bezplatných řešení s velmi omezenou funkcionalitou.

Motivaci k vytvoření vlastního systému podpořil i fakt, že každé existujících řešení je dostupné primárně jako služba. V rámci této služby jsou tak uloženy i všechna data. Někteří uživatelé nemusí být s takovým přístupem ztotožněni, skrze obsažené, potencionálně citlivé, informace v překladech (například konkrétní a podrobné popisy technologických postupů ve výrobních procesech).

Díky vlastnímu systému si tak může každý z uživatelů určovat, na jakém místě budou překlady ukládány, případně kdo k nim a jakým způsobem bude mít přístup.

Systém by mohl být provozován jak samostatně na serverech uživatelů, tak i jako služba, například v SaaS modelu.

2 PHALCON FRAMEWORK

Phalcon je open source framework pro vývoj nejen webových aplikací v jazyku PHP. Na rozdíl od většiny PHP frameworků je Phalcon implementován jako rozšíření, psané v jazyku C, díky čemuž je rychlejší a efektivnější než ostatní frameworky.

Obrázek 8 Logo Phalcon frameworku [6]

2.1 Framework

Framework je předem stanovená struktura pro řešení určitého typu problému, poskytující sadu nástrojů, knihoven a postupů pro vytváření aplikací. Slouží jako šablona aplikace, nabízející standardní architekturu a sadu konvencí, které lze dodržovat při vývoji aplikací. Použití frameworku může urychlit dobu vývoje a zajistit konzistenci a spolehlivost, jelikož poskytuje osvědčený přístup k řešení rutinních úkolů. [7]

2.2 Historie

Zakladatel frameworku Phalcon, Andres Gutierrez, přišel v roce 2011 s myšlenkou nového PHP frameworku se zaměřením na použitelnost, funkce, a hlavně na výkon. První verze Phalconu byla vydána v roce 2012 a rychle si získala popularitu právě díky své jedinečné architektuře a vysokému výkonu. V průběhu let se Phalcon neustále vyvíjel a zdokonaloval díky pravidelným vydáním a aktualizacím. Dnes je Phalcon široce používán vývojáři pro vytváření výkonných webových aplikací. [3]

Aktuálně se mimo komunitu přispěvatelů, snažící se framework udržovat a posouvat dále, o Phalcon starají Nikolaos Dimopoulos a Anton Vasiliev. [3]

2.2.1 Verze

Tabulka 1 Verze frameworku Phalcon [2]

Verze	Rok vydání	Poznámka
Phalcon 0.x	2012	Postupné přidávání stěžejní funkcionality jako ORM implementované v jazyku C, dialekt pro SQL – PHQL či template engine Volt.
Phalcon 1.x	2013	
Phalcon 2.x	2015	Většina komponent byla přepsána z jazyka C do Zephiru.
Phalcon 3.x	2016	První LTS verze s podporou PHP 7.
Phalcon 4.x	2019	Zároveň s podporou nových verzí PHP (7.2, 7.3, 7.5) byly zavedeny striktnější rozhraní.
Phalcon 5.x	2022	Podpora pro PHP 7.4 + PHP 8.0 a vyšší.
Phalcon 6.x	nevydáno	Implementace kompletně v čistém PHP pro verze PHP 8.0 a vyšší.

Vývoj verze Phalcon 5 a podpory verzí PHP 8.0 a vyšší byl zpočátku ohrožen, jelikož došlo k odchodu jednoho z hlavních vývojářů Phalconu a jazyku Zephir. Zephir byl vyvinut jako pomocník při přispívání do Phalconu – mnohem jednoduššího jazyka, podobného JS/PHP, který se může každý snadno naučit a přispívat do něj. Bohužel si tento jazyk nezískal popularitu, a proto do něj nepřispívalo mnoho lidí. Jazyk začal stagnovat a bylo obtížné ho udržovat. Bylo objeveno mnoho chyb, které by bylo velmi obtížné opravit, pokud by nebyl celý jazyk předělán. [1]

I přes všechny tyto problémy se podařilo Phalcon týmu, nově v čele s Nikolaosem Dimopoulosem, vydat verzi Phalcon 5 i s podporou nejnovějších verzí PHP 8. V plánu je také vydání verze Phalcon 6, která bude na úkor výkonu napsána kompletně v čistém PHP (obdobně jako ostatní frameworky) s tím, že bude možné pro určité komponenty využít části z rozšíření jazyka C pro dosažení většího výkonu. [1]

2.3 Architektura

Phalcon byl původně vyvinut jako rozšíření pro PHP, napsané v jazyku C, které ve srovnání s jinými PHP frameworky přinášelo značné výhody, co se týče výkonu. Na druhou stranu měl vždy malou vývojářskou komunitu a ve srovnání s jinými PHP frameworky neměl tolik příspěvků do kódu, nabízející opravy chyb a případná vylepšení. To bylo způsobeno především tím, že většina vývojářů, využívajících tento framework, neuměla jazyk C. Z tohoto důvodu vznikl nový programovací jazyk Zephir, mající velmi podobnou syntaxi jako PHP nebo JavaScript a nevyžaduje tak rozsáhlé znalosti jazyku C. Ihned po jeho vydání tak došlo k přepsání Phalcon frameworku do tohoto jazyku. [10]

Kód frameworku v jazyku Zephir je zkompilován do jazyku C, kdy se vygenerují všechny soubory (hlavičkové, zdrojové, konfigurační) potřebné k sestavení daného rozšíření. Kód v C, vygenerovaný v předchozím kroku, je následně zkompilován do rozšíření PHP pomocí vývojových nástrojů. Výsledkem tohoto kroku je soubor sdílené knihovny (*.so* pro Linux a *.dll* pro Windows), který může být načten interpretem PHP. Zkompilované rozšíření se načte do jazyku PHP přidáním *extension=phalcon.so* (záleží na pojmenování výsledné knihovny a cílovém systému) do konfiguračního souboru PHP (*php.ini*) nebo pomocí příslušných konfiguračních nastavení pro používaný webový server. Po načtení rozšíření jsou k dispozici veškeré funkce dostupné ve zkompilované verzi Phalcon frameworku. Tento přístup zachovává výkonnostní výhody související s rozšířením vytvořené v jazyku C a zároveň udržuje proces vývoje přístupnějším pro PHP vývojáře. [5][8][10]

2.3.1 Zephir

Zephir jako takový je napsán v jazyku PHP a z tohoto důvodu potřebuje mít na cílovém systému nainstalovanou relevantní verzi PHP. Samotná instalace Zephiru vyžaduje splnění prerekvizit, v rámci systému:

- Zephir parser
- C kompilátor (např. *gcc* nebo *clang*)
- re2c
- PHP vývojové nástroje (*php-dev*)

Pro linuxové systémy jsou vyžadovány dodatečné nástroje:

- GNU make
- autoconf

- automake
- libpcre3
- build-essential

Před instalací, respektive spuštěním Zephiru je nutno zkontrolovat a případně povolit rozšíření Zephir parser (přidání *extension=zephir_parser.so* do konfiguračního souboru *php.ini*, dle pojmenování rozšíření a typu dle cílového systému). [5]

Způsobů, jakými lze Zephir nainstalovat, existuje více. Doporučený, oficiálně podporovaný a nejjednodušší způsob instalace je pouhé stažení konkrétní verze PHAR souboru z GitHubu do požadovaného adresáře. [5]

Po nainstalování všech potřebných závislostí a stažení PHAR souboru je možné již plně využívat funkcionality, jež Zephir nabízí. Všechny dostupné příkazy jsou vypsány po spuštění příkazu *zephir help*, jehož výstup lze vidět na obrázku č. 9.

```
 _____                  __   _
/__  /  ___  ____  ____/ /_ (_)____
  / /  / _ \/ __ \/ __ \/ __/ / ___/
 / /__/  __/ /_/ / / / / /_/ / /
/____/\___/ .___/_/ /_/_/\__/_/
         /_/

Zephir 0.12.2 by Andres Gutierrez and Serghei Iakovlev (4ab69bd9)

Usage:
  command [options] [arguments]

Options:
      --dumpversion  Print the version of the compiler and don't do
anything else
  -h, --help         Print this help message
      --no-ansi      Disable ANSI output
  -V, --version      Print compiler version information and quit

Available commands:
  api        Generates a HTML API based on the classes exposed in the
             extension
  build      Generates/Compiles/Installs a Zephir extension
  clean      Cleans any object files created by the extension
  compile    Compile a Zephir extension
  fullclean  Cleans any object files created by the extension (including
             files generated by phpize)
  generate   Generates C code from the Zephir code without compiling it
  help       Displays help for a command
  init       Initializes a Zephir extension
  install    Installs the extension in the extension directory (may require
             root password)
  stubs      Generates stubs that can be used in a PHP IDE
```

Obrázek 9 Dostupné příkazy v Zephir

Kód v jazyku Zephir musí být vždy rozdělen do tříd. Jazyk je určen k vytváření objektově orientovaných knihoven/frameworků, tím pádem kód mimo třídy povolen není. Kromě toho je také vyžadován jmenný prostor (*namespace*). [8]

```
namespace Test;

/**
 * This is a sample class
 */
class Hello
{
    /**
     * This is a sample method
     */
    public function say()
    {
        echo "Hello World!";
    }
}
```

Obrázek 10 Třída Hello v Zephir [8]

Na obrázku č. 10 je uveden příklad velmi jednoduché třídy *Hello* s jednou metodou *say*, pouze vracející textový řetězec. Po úspěšném zkompilování této třídy je vyprodukován kód, vyobrazený na obrázku č. 11, jež je transparentně zkompilován pomocí příslušného C kompilátoru, jako je *gcc*, *clang* nebo *vc++*. [8]

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include "php.h"
#include "php_test.h"
#include "test.h"

#include "kernel/main.h"

/**
 * This is a sample class
 */
ZEPHIR_INIT_CLASS(Test_Hello) {
    ZEPHIR_REGISTER_CLASS(Test, Hello, hello, test_hello_method_entry, 0);
    return SUCCESS;
}

/**
 * This is a sample method
 */
PHP_METHOD(Test_Hello, say) {
    php_printf("%s", "Hello World!");
}
```

Obrázek 11 Třída Hello zkompilovaná do jazyku C [8]

2.4 Phalcon IDE Stubs

Na rozdíl od ostatních PHP frameworků nelze v moderních editorech při vývoji nabízet automatické doplňování kódu (např. třídy, konstanty, atributy, metody tříd) samotného Phalcon frameworku. [4]

Pro potřeby vývoje nabízí Phalcon knihovnu pro jazyk PHP, obsahující veškeré třídy, metody, atributy a konstanty, které samotný framework na pozadí implementuje. To znamená, že všechen kód je pouze šablona reálné funkcionality, sloužící jednak k automatickému doplňování kódu při vývoji a jednak pro statickou analýzu kódu. [4]

```
Adds a route to the router that only match if the HTTP method is DELETE
Parameters: array|string $paths – = [ 'module' => '', 'controller' => '', 'action' => '',
            'namespace' => '' ]
            string $pattern
            mixed $position
Returns:    RouteInterface
public function addDelete(string $pattern, $paths = null, $position = Router::POSITION_LAST): RouteInterface
{
}

Adds a route to the router that only match if the HTTP method is GET
Parameters: array|string $paths – = [ 'module' => '', 'controller' => '', 'action' => '',
            'namespace' => '' ]
            string $pattern
            mixed $position
Returns:    RouteInterface
public function addGet(string $pattern, $paths = null, $position = Router::POSITION_LAST): RouteInterface
{
}
```

Obrázek 12 Část třídy Router v knihovně Phalcon IDE Stubs

2.5 Phalcon Devtools

Nástroj Phalcon Devtools napomáhá v generování základní kostry kódu, vytváření migrací databáze a celkově urychluje vývoj. Pomocí jednoduchého příkazu lze vygenerovat základní součásti aplikace, což umožňuje snadno vyvíjet aplikace za použití Phalcon frameworku. [9]

```
Phalcon DevTools (4.2.0)

Available commands:
  info            (alias of: i)
  commands        (alias of: list, enumerate)
  controller      (alias of: create-controller)
  module          (alias of: create-module)
  model           (alias of: create-model)
  all-models      (alias of: create-all-models)
  project         (alias of: create-project)
  scaffold        (alias of: create-scaffold)
  migration       (alias of: create-migration)
  webtools        (alias of: create-webtools)
  serve           (alias of: server)
  console         (alias of: shell, psysh)
```

Obrázek 13 Dostupné příkazy nástroje Phalcon Devtools

Jak lze vidět na obrázku č. 13, nástroj poskytuje mimo vytvoření základní kostry kódu také možnosti pro vygenerování například databázových migrací, databázových modelů nebo kontrolerů. Vedle generování menších částí kódů dokáže také vygenerovat kompletní strukturu například pro CRUD určené entity (kontrolery, modely, pohledy či i databázové migrace).

2.6 Základní komponenty

2.6.1 Application

Hlavní komponentou frameworku je *Application*, zapouzdřující všechny složitější operace spojené s vytvářením všech ostatních komponent, potřebných k běhu MVC nebo CLI aplikace. Použití této komponenty není povinné a lze ji nahradit ručním sestavováním dílčích komponent (DI kontejner, Dispatcher, Router atd.). [11]

Použití spočívá ve vytvoření objektu *Application*, jemuž je volitelně v konstruktoru předán vytvořený DI kontejner. Následně lze do aplikace postupně registrovat jednotlivé moduly, pokud jimi vytvářená aplikace disponuje. Pro spuštění cyklu zpracování požadavku slouží metoda *handle*, očekávající hodnotu URI (např. hodnota *REQUEST_URI* obsažená v super globální proměnné *$_SERVER*).

```php
<?php

use Phalcon\Di\FactoryDefault;
use Phalcon\Mvc\Application;

$container = new FactoryDefault();
$application = new Application($container);

try {
    $response = $application->handle($_SERVER["REQUEST_URI"]);

    $response->send();
} catch (\Exception $e) {
    echo $e->getMessage();
}
```

Obrázek 14 Ukázka použití MVC aplikace

2.6.2 Config

Téměř všechny aplikace vyžadují ke správnému fungování nastavení potřebných parametrů. Konfigurace může obsahovat mimo parametry pro samotnou aplikační logiku také parametry základních částí aplikace, jako je nastavení logování informací (cesty k souborům, formát logované události atd.) anebo také hodnoty připojení k databázi. [12]

Konfigurační komponenta je navržena tak, aby konfigurační data ukládala objektově orientovaným způsobem. Tato data lze načítat pomocí dostupných adaptérů z různých formátů, jako je multidimenzionální pole v nativním PHP, JSON, YAML a INI. Samozřejmostí je také možnost implementace vlastního adaptéru pro podporu dalších formátů. [12]

```php
<?php

return [
    'postgres' => [
        'host' => '',
        'username' => '',
        'password' => '',
        'database' => '',
    ],
    'logger' => [
        'path' => '/var/www/logs',
        'format' => '[%date%] [%level%] %message%',
        'dateFormat' => 'd.m.Y H:i:s',
        'logLevel' => 'debug',
        'filename' => 'application.log',
    ]
];
```

Obrázek 15 Ukázka konfiguračních parametrů v PHP formátu

Inicializace konfigurační komponenty spočívá v pouhém vytvoření objektu adaptéru dle výběru a předání potřebných parametrů tomuto adaptéru. V případě PHP adaptéru je nutné předat pouze cestu k souboru **.php*, jež obsahuje konfigurační parametry v multidimenzionálním poli.

```php
<?php

use Phalcon\Config\Adapter\Php;

$fileName = '/var/www/config/application.php';
$config = new Php($fileName);

echo $config
    ->get('logger')
    ->get('logLevel');

echo $config
    ->path('logger.logLevel');

echo $config->logger->logLevel ?? '';
```

Obrázek 16 Ukázka použití konfigurační komponenty

Na obrázku č. 16 lze vidět ukázku vytvoření konfigurační komponenty na základě definice uvedené na obrázku č. 15 a také několik možností získání zanořeného parametru *logger.logLevel*. Všechny uvedené možnosti vypisují stejný výstup, a to *debug*.

Objekt konfigurace lze také zaregistrovat do DI kontejneru, jednak pod vlastním klíčem (např. FQCN) a jednak pod standardním klíčem *config*, který je následně dostupný v každé třídě, jež vychází ze třídy *Injectable* (více v kapitole *2.6.5 DI Container*). [12]

2.6.3 Router

Phalcon nabízí primárně dva způsoby definování cest (*route*) k prostředkům. Prostředkem může být v tomto případě třída kontroleru a metoda v něm nebo například i anonymní funkce (často se vyskytuje v případě Phalcon Micro aplikací). [13]

Prvním způsobem definice rout je využití metod na samotném objektu routeru, díky nímž lze jednoduše propojit cestu s prostředkem pro konkrétní HTTP metodu. Těmi základními metodami jsou například *addGet*, *addPost*, *addPut*, *addPatch*, *addDelete* a další pro ostatní HTTP metody. [13]

```php
<?php

use Localizer\Modules\Languages\Controllers\LanguagesController;
use Phalcon\Mvc\Router;

$router = new Router();

$router->addGet('/api/v1/languages', [
    'controller' => LanguagesController::class,
    'action' => 'list',
]);

$router->addPost('/api/v1/languages', [
    'controller' => LanguagesController::class,
    'action' => 'create',
]);
```

Obrázek 17 Ukázka použití router komponenty

Phalcon nabízí také možnost využití speciálního routeru, kdy v kombinaci s *Annotation* komponentou dokáže naparsovat anotace tříd a metod kontrolerů a nasměrovat tak do nich požadované requesty. [13]

```php
<?php

use Localizer\Lib\Responses\ResponseCreated;
use Localizer\Lib\Responses\ResponseOk;
use Phalcon\Mvc\Controller;

/**
 * @RoutePrefix('/api/v1/languages')
 */
class LanguagesController extends Controller
{
    /**
     * @Get('/')
     */
    public function listAction(): ResponseOk
    {
        return new ResponseOk([]);
    }

    /**
     * @Post('/')
     */
    public function createAction(): ResponseCreated
    {
        return new ResponseCreated([]);
    }
}
```

Obrázek 18 Ukázka použití router anotací

Router poskytuje mnoho dalších funkcionalit, jako je definování rout do skupin nebo možnost stanovit si dodatečné podmínky pro ovlivňování přiřazování rout v rámci requestu pomocí callbacku *beforeMatch*. Důležité je také zmínit, že samotné chování routeru závisí jak na jeho počátečním nastavení, tak i na nastavení *Dispatcher* komponenty, primárně skrze používaný sufix kontroleru (*Controller*) a metody akce (*Action*). [13]

2.6.4 Dispatcher

Dispatcher zodpovídá za vytváření kontrolerů a provádění požadovaných akcí dle příchozího requestu. Takzvaný dispatching je proces, při kterém se z requestu extrahuje název modulu, název kontroleru, název akce, volitelné parametry, a následně dojde k vytvoření právě samotného kontroleru a provedení příslušné akce. [14]

Proces dispatchingu lze rozšiřovat pomocí reakce na vyvolávané standardní eventy, za použití *Event Manager* komponenty (více v kapitole *2.6.6 Event Manager*). Příkladem takových reakcí může být provedení autentizace či autorizace po vytvoření kontroleru a před provedením samotné akce (*dispatch:beforeExecuteRoute*) nebo přesměrování na chybovou stránku v případě chyby vzniklé v rámci tohoto procesu (*dispatch:beforeException*). Upravovat lze také samotné parametry dispatcheru, jako je například ovlivnění cílového kontroleru, akce a také parametrů předávaných dané akci. [14]

2.6.5 DI Container

Velmi důležitou součástí frameworku je právě DI kontejner, spravující závislosti napříč aplikací. Při registraci služeb do kontejneru lze uvést, zda mají být sdílené pro celou aplikaci nebo bude vždy vytvořena nová instance při předávání závislostí.

Mimo manuální sestrojení DI kontejneru, obsahující základní služby, nabízí Phalcon již předpřipravený kontejner (třída *FactoryDefault*), obsahující všechny tyto služby potřebné pro běh aplikace. Není tak nutné definovat veškeré služby manuálně. [15]

Zajímavou součástí je třída *Injectable*, kterou může jakákoliv třída rozšiřovat a mít tak k dispozici přímý přístup k DI kontejneru i mimo závislosti předávané konstruktoru. Díky této třídě lze pracovat s tzv. dynamickou vlastností třídy, kdy se po jejím použití hledá služba v DI kontejneru a není tak nutné získávat instanci kontejneru a volat na něm příslušné metody. [15]

Určitou nevýhodou může být absence autowiring funkcionality (implicitní předávání závislostí objektům bez dodatečné definice). Aktuálně lze tedy tuto funkcionalitu doimplementovat pouze manuálně, nikoliv s využitím Phalcon frameworku.

Jedním z omezení plnohodnotného využití DI kontejneru je konstruktor v kontroleru, který je označen jako *final* a tudíž jej nelze přetížit ve třídě potomka. Na druhou stranu Phalcon umožňuje implementaci bezparametrové metody *onConstruct*, jež je automaticky spuštěna při inicializaci konkrétního kontroleru, ale v žádném případě nenahrazuje princip konstruktoru. [16]

2.6.6 Event Manager

Účelem této komponenty je umožnění zachycení spuštění různých komponent ve frameworku vytvořením tzv. hooků. Tyto hooky umožňují získávat informace o stavu dané komponenty, manipulovat s daty nebo měnit průběh samotného procesu komponenty. Komponenta se skládá primárně ze správce událostí, třídy *Manager*, starající se jednak o provazování událostí se službami pro obsluhu událostí (metoda *attach*) a jednak o samotné vyvolávání těchto událostí (metoda *fire*). Hlavním nositelem informací, vztažených k danému eventu, je třída *Event*, obsahující informace o svém zdroji a případně také další uživatelská data. Posloupnost zpracování událostí lze ovlivnit nastavením příslušných priorit v podobě číselných hodnot. [17]

```php
<?php

use Phalcon\Events\Event;
use Phalcon\Events\Manager;

$eventManager = new Manager();
$eventManager->attach('component:event', function (Event $event) {
    $data = $event->getData();
});
$eventManager->fire('component:event', $this, ['some' => 'data']);
```

Obrázek 19 Ukázka použití event manageru

2.6.7 Validation

Validační komponenta slouží k validaci libovolných dat. V základu poskytuje framework desítky validačních pravidel pro textové řetězce, číselné hodnoty, soubory a mnoho dalšího jako je kontrola správnosti emailové adresy, kreditní karty nebo také IP adresy. Mimo tyto základní pravidla je umožněno vytvářet si pravidla vlastní, a to vytvořením třídy implementující příslušné rozhraní. [18]

Pro spuštění validace je využívána metoda *validate*, které jsou předána libovolná data (pole, objekt atd.).

```php
<?php

use Phalcon\Filter\Validation;
use Phalcon\Filter\Validation\Validator\Email;

$validation = new Validation();
$validation->add(
    'email',
    new Email([
        'message' => 'User\'s email must be a valid email address.'
    ])
);

$validationMessages = $validation->validate(['email' => 'invalid-email']);
```

Obrázek 20 Ukázka použití validační komponenty

2.6.8 Request

Request komponenta zapouzdřuje skutečný HTTP požadavek, obvykle iniciovaný webovým prohlížečem a následně využíván v samotné aplikaci.

PHP automaticky vyplňuje superglobální proměnné jako *$_GET*, *$_POST* a *$_REQUEST* v závislosti na typu požadavku. Tyto proměnné obsahují hodnoty obsažené například v odeslaných formulářích či parametry uvedené přímo v URL adrese v podobě tzv. query parametrů. Tyto hodnoty nejsou implicitně sanitovány a mohou tak obsahovat nepovolené znaky či dokonce i škodlivý kód, což může vést k SQL injection nebo XSS útokům. [20]

Phalcon umožňuje díky této komponentě přistupovat ke zmíněným superglobálním proměnným a zároveň hodnoty v nich sanitovat či filtrovat pomocí své vlastní *Filter* komponenty. [21]

```php
<?php

use Phalcon\Filter\Filter;
use Phalcon\Http\Request;

$request = new Request();
echo $request->getQuery('page', Filter::FILTER_INT);
```

Obrázek 21 Ukázka použití request komponenty s použitím filtru

2.6.9 Response

Response komponenta zapouzdřuje skutečnou HTTP odpověď od aplikace směrovanou zpátky k uživateli. Mimo sestrojení odpovědi působí komponenta také jako HTTP klient, který samotnou odpověď odesílá. V odpovědi lze jednak posílat potřebný obsah a jednak nastavovat hodnoty hlavičky, jako je typ vrácené odpovědi (*Content-Type*) a další, uvedené v RFC 9110 [23]. Mimo to umožňuje komponenta také pracovat s cookies. [22]

```php
<?php

use Phalcon\Http\Response;

$response = new Response();
$contents = file_get_contents('file.pdf');

$response
    ->setContent($contents)
    ->setContentType('application/pdf')
    ->setHeader('Content-Disposition', "attachment; filename='file.pdf'")
    ->send();
```

Obrázek 22 Ukázka použití response komponenty

2.6.10 Frontend

Framework nabízí mnoho komponent pro usnadnění implementace frontendové části aplikace. Seznam těch nejvyužívanějších s krátkým popisem je uveden v tabulce č. 2.

Tabulka 2 Seznam komponent pro implementaci frontendové části [24]

Komponenta	Popis
Assets	Umožňuje spravovat statické prostředky, jako jsou soubory stylů CSS nebo JavaScript knihovny ve webové aplikaci.

Flash Messenger	Flash zprávy upozorňují uživatele na stav akcí, které provedl, nebo také k jednoduchému zobrazení informací uživatelům.
Forms	Pomáhá vytvářet a spravovat formuláře, které lze použít k vykreslování prvků HTML ve webové aplikaci, ale také k provádění validace vstupů z těchto prvků.
Image	Poskytuje adaptéry, nabízející funkce pro manipulaci s obrázky.
Tag	Komponenta dokáže generovat značky HTML. Pro maximální výkon je každá třída načítána lazy způsobem.
View	Pohledy představují uživatelské rozhraní aplikace, obsahující HTML kód s podporou vkládání PHP kódu, které provádějí úlohy související výhradně s prezentací dat.

K vytváření pohledů poskytuje Phalcon šablonovací engine Volt, jenž je velmi rychlý díky své implementaci v jazyku C pro následné použití v jazyku PHP. Volt dokáže velmi dobře spolupracovat s ostatními komponentami Phalconu, ale lze jej použít i jako samostatnou komponentu. Mimo to nabízí také sadu pomocných nástrojů pro snadné psaní pohledů. [25]

```
{% for user in users %}
<div class='row'>
    <div>
        ID: {{ user.userId }}
    </div>
    <div>
        {%- if user.isActive === true -%}
        Active
        {%- else -%}
        Inactive
        {%- endif -%}
    </div>
    <div>
        {{ user.fullName }}
    </div>
    <div>
        {{ user.email }}
    </div>
</div>
{% endfor %}
```

Obrázek 23 Ukázka použití Volt engine

2.6.11 Database

Pro práci s databází, nabízí Phalcon několik možností, od volání čistých SQL dotazů až po vytváření modelů konkrétních entit pomocí objektového přístupu. V základu poskytuje

ovladače pro relační databáze jako jsou *MySQL*, *PostgreSQL* a *Sqlite*. Ve verzi 4.0 a výš byla odstraněna podpora dokumentových databází v zastoupení *MongoDB*, z důvodu již léta nepodporovaného PHP ovladače (návrat podpory je plánován do budoucích verzí frameworku). [24]

Mimo základní práci s databází, jako je správa tabulek a dat v nich, poskytuje Phalcon možnost reagovat na vyvolané události v určitých částí procesu – například je možnost před každým voláním SQL dotazu logovat jeho spuštění a po jeho dokončení naopak logovat trvání provedení samotného dotazu. [26]

2.7 Phalcon Micro

Phalcon framework také nabízí velmi „tenkou“ a rychlou aplikaci, zvanou Phalcon Micro, díky níž lze vytvářet mikroaplikace s minimem režií a kódu. Phalcon Micro je vhodný pro menší aplikace, jakými mohou být například jednoduché API nebo také různé prototypy aplikací. [19]

```php
<?php

use Phalcon\Mvc\Micro;

$app = new Micro();
$app->get('/hello-world', fn() => 'Hello World!');
$app->handle($_SERVER["REQUEST_URI"]);
```

Obrázek 24 Ukázka použití Phalcon Micro

I v aplikaci tohoto typu lze využívat součásti plnohodnotného frameworku, jakými jsou router, dependency injection, event manager, database drivery a další. [19]

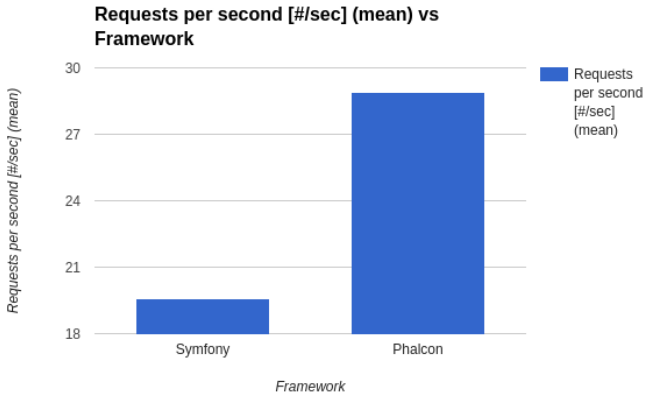
2.8 Alternativní frameworky

Na trhu existuje mnoho alternativních PHP frameworků, jakými jsou například nejpoužívanější Laravel [42], Symfony [43], Nette [44] ale i další, méně známější. Mimo tyto robustní frameworky, nabízející širokou škálu funkcionalit, existuje i několikero tzv. micro frameworků, jakým je například Slim [45].

Primární rozdíl mezi Phalcon a ostatními PHP frameworky je v jejich architektuře. Zatímco Phalcon je implementován jako rozšíření PHP v jazyku C, ostatní jsou tvořené čistě v PHP. Z tohoto důvodu není možné ladit kód samotného frameworku společně s vlastním aplikačním kódem, což může způsobovat větší časovou náročnost při implementaci s využitím komponent samotného frameworku.

Nevýhodou oproti ostatním frameworkům může být také malá komunita jak uživatelů, tak i samotných přispěvatelů.

Velkou výhodou Phalcon je jeho rychlost, což může být v mnoha ohledech důležitým parametrem.



Obrázek 25 Graf počtu požadavků za sekundu – Phalcon vs. Symfony [46]

Na obrázku č. 25 lze vidět graf, vyjadřující porovnání počtu odbavených požadavků na primitivním REST API za sekundu při 1000 požadavcích a 100 souběžných připojení mezi frameworky Phalcon a Symfony. S příchodem novějších verzí jazyku PHP se můžou čistě PHP frameworky postupně přibližovat rychlosti Phalcon.

Phalcon nemá oproti většině ostatních frameworků pevně definovanou strukturu, ani jakým způsobem má být výsledná aplikace sestavena. Tato variabilita může být pro začátečníky spíše překážkou.

Výběr vhodného frameworku se z velké části odvíjí od povahy projektu, na němž bude využíván. Pro případ implementace webové aplikace, využívající pohledy a další funkcionality s nimi spjaté, se můžou jevit Laravel, Symfony či Nette jako vhodnější. Pokud je projekt implementován z velké části jako REST API, využívající minimum funkcionalit frameworku, může být vhodnější využití právě Phalcon frameworku a jeho rychlosti.

Důležité je zdůraznit fakt, že celková rychlost jak webové, tak REST API aplikace, se z velké části odvíjí od její samotné implementace, nikoliv použitého frameworku.

3 LOKALIZACE

Softwarová lokalizace je proces přizpůsobování mobilních a webových aplikací, případně i jiného softwarového produktu, specifickému jazyku, kulturnímu prostředí a jejich zvyklostem. Cílem lokalizace je zajistit, aby měli všichni koncoví uživatelé napříč celým světem, využívající daný produkt, přístup ke stejným informacím.

Ne všechny jazyky jsou psány zleva doprava. Některé jazyky, jako je hebrejština a arabština, jsou psány zprava doleva. To může mít vliv na rozložení textu a umístění grafických prvků v aplikaci. Při lokalizaci aplikací je tedy nutné zohlednit i tyto specifické vlastnosti daného jazyku.

Obsah této kapitoly vychází z informací, primárně čerpaných ze zdrojů [39], [40], [41] a z části i z analýzy existujících řešení, provedené v kapitole *1.1 Analýza existujících řešení*.

3.1 Lokalizační texty

Lokalizační text je jakýkoliv text, přeložený do konkrétního jazyku s důrazem na zvyklosti v daném kulturním prostředí.

Důležitou součástí lokalizace textů je styl samotné lokalizace. Každý jazyk má své specifika, co se týče gramatiky, skloňování, jazykových obratů, tvarů množného čísla či jiných jazykových prvků. Ve výsledných textech by měly být také zohledněny kulturní rozdíly v konkrétních oblastech, používající daný jazyk.

3.1.1 Parametrizace

K dynamickému vkládání hodnot do výsledných překladů je možné využít parametrů. Formát těchto parametrů může záležet na použité knihovně v aplikaci, jež je využívána pro zpracování těchto překladů. Formát pro parametr *name* může být například *{{name}}*, *{name}*, *%name%* či i obecnější *%s* nebo *%0*, pro konkrétní pořadí parametrů.

3.1.2 Pluralizace

Jazyky jsou rozdílné v tom, jak zacházejí s množným číslem podstatných jmen nebo jednotkových výrazů („hodina“ vs. „hodiny“ apod.). Některé jazyky mají dva tvary (například angličtina), některé pouze jeden tvar a některé i více tvarů.

Unicode Common Locale Data Repository [39] definuje pro tyto kategorie krátké, mnemotechnické značky:

- zero
- one
- two
- few
- many
- other (požadovaný obecný tvar množného čísla – používá se také v případě, kdy má jazyk pouze jediný tvar)

Například Gettext využívá pro definici těchto kategorií číselné hodnoty, počínaje nulou.

Způsob, jakým jsou tvary množného čísla definovány, závisí nejen na použitém formátu uchovávání lokalizačních textů (více v kapitole *3.1.3 Výstupní formát*), ale také i na možnostech použité knihovny pro zpracování těchto textů.

V následujících podkapitolách jsou uvedeny příklady definice tvarů, primárně pro formát JSON.

3.1.2.1 ICU MessageFormat

Speciální formát, definující různé tvary množného čísla, včetně konkrétního názvu proměnné (*count*), na jejímž základě je použit příslušný tvar (*one*, *other*). Kompletní popis parametrů a jejich použití je uveden v dokumentaci [40].

```
{
  "common.count": "{count, plural, one {Jeden} other {Více}}"
}
```

Obrázek 26 Definice pluralizace ve formátu ICU MessageFormat

3.1.2.2 Array

Tvary množného čísla jsou zanořené v asociativním poli, kdy zanořeným klíčem je textová (případně i číselná) podoba konkrétního tvaru (*one*, *other*).

```
{
  "common.count": {
    "one": "Jeden",
    "other": "Více"
  }
}
```

Obrázek 27 Definice pluralizace ve formátu Array

3.1.2.3 JSON String

Jedná se o formát, strukturou totožný formátu Array (uvedený v kapitole *3.1.2.2 Array*), s tím rozdílem, že je uchována formou textového řetězce místo zakomponování do samotné struktury.

```
{
  "common.count": "{\"one\":\"Jeden\",\"other\":\"Více\"}"
}
```

Obrázek 28 Definice pluralizace ve formátu JSON String

3.1.2.4 i18next

Výsledný klíč je složen z názvu samotného klíče (*common.count*), doplněn o identifikátor tvaru v číselné podobě (*0*, *1*).

```
{
  "common.count_0": "Jeden",
  "common.count_1": "Více"
}
```

Obrázek 29 Definice pluralizace ve formátu i18next

3.1.2.5 i18next V4

Výsledný klíč je složen z názvu samotného klíče (*common.count*), doplněn o identifikátor tvaru v textové podobě (*one*, *other*).

```
{
  "common.count_one": "Jeden",
  "common.count_other": "Více"
}
```

Obrázek 30 Definice pluralizace ve formátu i18next V4

3.1.3 Výstupní formát

V následujících podkapitolách jsou uvedeny příklady využívaných formátů pro uchovávání lokalizačních textů. Každý ze zde uvedených formátů obsahuje příklad překladu klasického textového řetězce a řetězce, obsahujícího dva tvary množného čísla (*one* a *other*). Pro klíč je zvoleno označení kódem, ale obecně lze za klíč považovat i text ve zdrojovém jazyku.

Formátů existuje mnoho – zde jsou uvedeny ty známější, jež jsou využívány při implementaci webových či mobilních aplikací, nezávisle na platformě.

Výsledný tvar struktury může záležet na možnostech použité knihovny pro práci s lokalizačními texty v aplikaci.

3.1.3.1 JSON

Možností, které poskytuje formát JSON pro uchovávání lokalizačních textů je více. Na obrázku č. 31 je uveden příklad struktury, podporující zanoření klíčů. Využít lze také čistě plochou strukturu s tím, že všechny klíče budou uvedeny na první úrovni s uvedenou kompletní cestou (například *common.countNested.one*). Definování tvarů množného čísla lze docílit využitím zanoření klíčů, případně i použitím ICU nebo jiného formátu.

```
{
  "common": {
    "info": "Informace",
    "count": "{count, plural, one {Jeden} other {Více}}",
    "countNested": {
      "one": "Jeden",
      "other": "Více"
    }
  }
}
```

Obrázek 31 Lokalizační texty ve formátu JSON

3.1.3.2 Gettext

Formát Gettext, v podobě PO souboru, obsahuje pro každý lokalizační klíč samostatný blok. V základu tento blok sestává z identifikátoru překladu (*msgid*) a textu překladu (*msgstr*). Pro definování tvarů množného čísla se kromě identifikátoru překladu (*msgid*) uvádí i identifikátor množného čísla (*msgid_plural*). Text překladu jednotlivých tvarů je uváděn pod klíčem *msgstr[X]*, kde *X* supluje číselnou hodnotu, počínaje v nule. Konkrétní ukázka definice je uvedena na obrázku č. 32. Veškeré podporované klíče, respektive parametry, jsou dostupné v dokumentaci [36].

```
msgid "common.info"
msgstr "Informace"

msgid "common.count"
msgid_plural "common.count"
msgstr[0] "Jeden"
msgstr[1] "Více"
```

Obrázek 32 Lokalizační texty ve formátu Gettext

3.1.3.3 YAML

V případě formátu YAML jsou možnosti totožné s formátem JSON (uvedené v kapitole *3.1.2.1 JSON*) pouze s jiným zápisem.

```yaml
common:
  info: "Informace"
  count: "{count, plural, one {Jeden} other {Více}}"
  countNested:
    one: "Jeden"
    other: "Více"
```

Obrázek 33 Lokalizační texty ve formátu YAML

3.1.3.4 Android Resources

Primárně pro aplikace nad systémem Android je dostupný formát Android Resources, vycházející z formátu XML. Značka *string* a její atribut *name* definují překlad běžného klíče. K definici tvarů množného čísla je využívána značka *plurals* s atributem *name*, obsahující pod sebou další značky *item*. Samotný tvar je určen atributem *quantity*. Konkrétní příklad je uveden na obrázku č. 34, vycházející z dokumentace [37].

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="common.info">Informace</string>
    <plurals name="common.count">
        <item quantity="one">Jeden</item>
        <item quantity="other">Více</item>
    </plurals>
</resources>
```

Obrázek 34 Lokalizační texty ve formátu Android XML

3.1.3.5 Apple Stringsdict

Pro platformu Apple je možné využívat formát Apple Stringsdict (na principu slovníku – klíč, hodnota), jehož příklad je uveden na obrázku č. 35, případně i Apple Strings či Apple XLIFF. Výsledný použitý tvar množného čísla se řídí hodnotou v klíči *NSStringLocalizedFormatKey*, jehož hodnota se skládá z předpony *%#@*, názvu proměnné, nesoucí hodnotu a přípony @. Veškeré podporované klíče a značky, včetně podrobnějšího popisu, jsou dostupné v dokumentaci [38].

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
    <dict>
        <key>common.info</key>
        <dict>
            <key>NSStringLocalizedFormatKey</key>
            <string>%@</string>
            <key>common.info.value</key>
            <string>Informace</string>
        </dict>
        <key>common.count</key>
        <dict>
            <key>NSStringLocalizedFormatKey</key>
            <string>%#@count@</string>
            <key>common.count.value</key>
            <dict>
                <key>NSStringFormatSpecTypeKey</key>
                <string>NSStringPluralRuleType</string>
                <key>one</key>
                <string>Jeden</string>
                <key>other</key>
                <string>Více</string>
            </dict>
        </dict>
    </dict>
</plist>
```

Obrázek 35 Lokalizační texty ve formátu Apple Stringsdict

3.2 Lokalizace aplikací

Existuje několik způsobů, jakými lze aplikace lokalizovat. V této kapitole si popíšeme používané metody pro zavádění a poskytování lokalizačních textů těmto aplikacím.

Většina aplikací využívá pro zavádění textů knihovny, podporující různé formáty, jež tyto texty uchovávají (více v kapitole *3.1.3 Výstupní formát*).

Samotný překlad lokalizačních textů lze provádět pomocí mnoha pomocných nástrojů třetích stran, odvíjející se opět od konkrétního formátu.

3.2.1 Lokální soubory

Často využívaným způsobem je použití lokálních souborů v konkrétním formátu, jež integrovaná knihovna podporuje. Knihovna si dle nastaveného jazyku uživatele načte konkrétní soubor a následně zobrazí texty v něm obsažené. Tyto soubory jsou běžně uloženy společně se zdrojovým kódem aplikace, tudíž jejich změna vyžaduje případně vydání i nové verze.

3.2.2 Databáze

Lokalizační texty lze ukládat i do databáze, což umožňuje jejich jednodušší správu bez nutnosti vydávání nových verzí. V takovém případě je ale vhodné řešit optimalizaci z hlediska výkonu (například využitím cache), jelikož není nutné pokaždé načítat aktuální data z databáze. Benefitem ukládání textů v databázi může být i možnost implementace vlastního nástroje pro jejich správu.

3.2.3 Externí služby

Využívat lze také externí služby pro správu lokalizačních textů, nabízející mimo přizpůsobené rozhraní pro jejich definování, i nástroje pro následnou distribuci. Taková distribuce může být řešena například způsobem CDN hostingu, poskytnutého REST API, webhooků a případně i exportu do souboru vybraného formátu pro následné lokální použití.

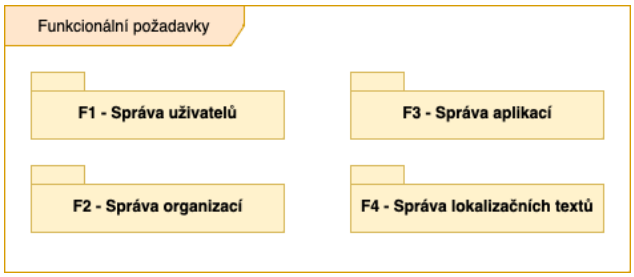
II. PRAKTICKÁ ČÁST

4 POŽADAVKY NA SYSTÉM

Systém bude primárně určen pro správu a poskytování lokalizačních textů aplikacím. Jedním z úkolů je také zaměření se na pohodlí uživatele při správě těchto textů. Většina požadavků vyplynula z analýzy již existujících řešení v teoretické části práce a obecně potřebných funkcionalit pro dosažení zabezpečeného a jednoduše doručitelného systému nejen v interní síti.

4.1 Funkcionální požadavky

V této kapitole jsou uvedeny funkcionální požadavky na systém, definující nezbytné úkoly či akce, jež musí aplikace umožňovat. Veškeré požadavky jsou logicky rozděleny do několika balíků funkcionalit a podrobněji rozepsány v jednotlivých podkapitolách. Každý požadavek je označen kódem *Fxy*, kde *x* je číslo balíku funkcionalit a *y* číslo požadavku.



Obrázek 36 Funkcionální požadavky rozdělené do balíků

4.1.1 F1 – Správa uživatelů

F1001 – Systém musí administrátorovi umožnit zobrazení seznamu všech uživatelských účtů, ve kterém lze vyhledávat podle jména, příjmení a emailové adresy

F1002 – Systém musí administrátorovi umožnit vytvoření uživatelského účtu

F1003 – Systém musí administrátorovi umožnit upravení uživatelského účtu

F1004 – Systém musí administrátorovi umožnit smazání uživatelského účtu

F1005 – Systém musí před smazáním uživatelského účtu vyžádat potvrzení smazání

F1006 – Systém musí evidovat tyto informace o uživateli: jméno, příjmení, emailová adresa, heslo, role

F1007 – Systém musí kontrolovat správný tvar a unikátnost emailové adresy

F1008 – Systém musí kontrolovat sílu hesla: alespoň 8 znaků, alespoň 1 malé písmeno, alespoň 1 velké písmeno, alespoň 1 číslice

F1009 – Systém musí rozlišovat uživatele dle rolí: administrátor a klasický uživatel

F1010 – Systém musí umožnit přihlášení uživatele pomocí emailové adresy a hesla

F1011 – Systém musí umožnit odhlášení uživatele

F1012 – Systém musí při prvotní instalaci vytvořit jeden uživatelský účet s rolí administrátor

F1013 – Systém nesmí umožnit smazání hlavního administrátorského účtu

4.1.2 F2 – Správa organizací

F2001 – Systém musí administrátorovi umožnit zobrazení seznamu všech organizací, ve kterém lze vyhledávat podle názvu

F2002 – Systém musí administrátorovi umožnit vytvoření organizace

F2003 – Systém musí administrátorovi umožnit upravení organizace

F2004 – Systém musí administrátorovi umožnit smazání organizace

F2005 – Systém musí před smazáním organizace vyžádat potvrzení smazání

F2006 – Systém musí evidovat tyto informace o organizaci: název

F2007 – Systém musí kontrolovat unikátnost názvu organizace

F2008 – Systém musí administrátorovi umožnit zobrazení seznamu všech uživatelů, kteří jsou přidáni do vybrané organizace

F2009 – Systém musí administrátorovi umožnit přidání uživatele do organizace

F2010 – Systém musí administrátorovi umožnit odstranění uživatele z organizace

F2011 – Systém nesmí povolit smazání organizace v případě, kdy je pod ní vytvořena alespoň jedna aplikace

4.1.3 F3 – Správa aplikací

F3001 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit zobrazení seznamu všech aplikací pod vybranou organizací, ve kterém lze vyhledávat podle názvu

F3002 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit vytvoření aplikace pod vybranou organizací

F3003 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit upravení aplikace

F3004 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit smazání aplikace

F3005 – Systém musí před smazáním aplikace vyžádat potvrzení smazání

F3006 – Systém musí evidovat tyto informace o aplikaci: název

F3007 – Systém musí kontrolovat unikátnost názvu aplikace v kontextu dané organizace

F3008 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit přidání jazyku aplikace

F3009 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit upravení jazyku aplikace

F3010 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit smazání jazyku aplikace

F3011 – Systém musí před smazáním jazyku aplikace vyžádat potvrzení smazání

4.1.4 F4 – Správa lokalizačních textů

F4001 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit přidání skupiny lokalizačních klíčů

F4002 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit upravení skupiny lokalizačních klíčů

F4003 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit duplikaci skupiny lokalizačních klíčů

F4004 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit smazání skupiny lokalizačních klíčů

F4005 – Systém musí před smazáním skupiny lokalizačních klíčů vyžádat potvrzení smazání

F4006 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit přidání lokalizačního klíče

F4007 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit upravení lokalizačního klíče

F4008 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit smazání lokalizačního klíče

F4009 – Systém musí před smazáním lokalizačního klíče vyžádat potvrzení smazání

F4010 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit překlad lokalizačního klíče do jakéhokoliv jazyku

F4011 – Systém musí zajistit verzování překladů lokalizačního klíče

F4012 – Systém musí umožnit obnovení původní verze překladu lokalizačního klíče

F4013 – Systém musí umožnit přidat popis a obrázek ke skupině lokalizačních klíčů

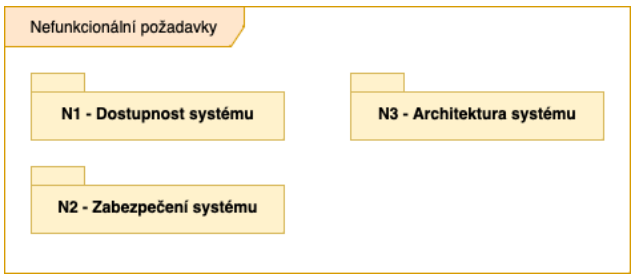
F4014 – Systém musí umožnit přidat popis a obrázek k lokalizačnímu klíči

F4015 – Systém musí poskytnout rozhraní pro vytvoření překladu včetně možnosti skloňování

F4016 – Systém musí administrátorovi a uživateli ve vybrané organizaci umožnit export lokalizačních textů do souboru vybraného formátu

4.2 Nefunkcionální požadavky

V této kapitole jsou uvedeny nefunkcionální požadavky na systém, definující požadavky na architekturu systému, uživatelské rozhraní či prostředí pro běh systému. Veškeré požadavky jsou logicky rozděleny do několika balíků požadavků a podrobněji rozepsány v jednotlivých podkapitolách. Každý požadavek je označen kódem *Nxy*, kde *x* je číslo balíku požadavků a *y* číslo požadavku.



Obrázek 37 Nefunkcionální požadavky rozdělené do balíků

4.2.1 N1 – Dostupnost systému

N1001 – Systém musí být dostupný jako webová aplikace přes webový prohlížeč

N1002 – Systém musí být lokalizován do českého a anglického jazyka

N1003 – Systém musí poskytovat REST API pro veškeré operace

N1004 – Systém musí poskytovat API endpoint pro získání všech lokalizačních textů pro konkrétní aplikaci a jazyk

4.2.2 N2 – Zabezpečení systému

N2001 – Systém musí mít zabezpečené veškeré API endpointy, vyjma přihlášení

4.2.3 N3 – Architektura systému

N3001 – Systém musí být poskytován ve formě Docker images

N3002 – Systém lze jednoduše nasadit do interní sítě

N3003 – Systém nesmí přímým způsobem ukládat soubory na systémový disk

4.3 Případy užití

Pro lepší pochopení požadavků na systém jsou sestaveny případy užití, v nichž vystupují aktéři, vykonávající určitý scénář. V jednotlivých podkapitolách jsou přiblíženi jak aktéři, tak i samotné scénáře případů užití.

4.3.1 Aktéři

Anonymní uživatel

Aktér, představující aktuálně nepřihlášeného uživatele při prvotním přístupu do webové aplikace. Pro anonymního uživatele je dostupný pouze formulář pro přihlášení.

Klasický uživatel

Aktér, představující autentizovaného uživatele s rolí „klasický uživatel“. Uživatel může následně spravovat aplikace, lokalizační klíče a jejich překlady v organizacích, do kterých je přiřazen administrátorem.

Administrátor

Aktér, představující autentizovaného uživatele s rolí „administrátor“. Administrátor disponuje stejnými možnostmi, jako klasický uživatel, s tím rozdílem, že není omezen přiřazením do konkrétních organizací. Mimo to může spravovat všechny uživatele a organizace v systému. Při prvotní instalaci systému je vždy vytvořen jeden hlavní

administrátorský účet. Pro tento jediný hlavní administrátorský účet je vyhrazena speciální role „systém“.

4.3.2 Scénáře případů užití

Scénář případu užití je tvořen posloupností kroků, popisující interakci mezi aktéry a systémem, vedoucí ke splnění určitého požadavku.

V podkapitolách jsou popsány jednotlivé scénáře, stěžejní pro chod celého systému, označené kódem *UCx*, kde *x* je číslo konkrétního scénáře. U každého ze scénářů jsou uvedeni aktéři, předpoklad pro splnění všech kroků a samotná posloupnost kroků.

4.3.2.1 UC01 Přihlášení

Aktéři: Anonymní uživatel

Předpoklad: Otevřená webová aplikace ve webovém prohlížeči

Scénář:

1. Webová aplikace nabídne formulář pro přihlášení s poli pro zadání emailové adresy a hesla
2. Uživatel vyplní emailovou adresu a heslo
3. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů vůči databázi
4. Pokud jsou údaje správné, dojde k přesměrování na hlavní stránku aplikace
5. Pokud nejsou údaje správné, je uživateli zobrazena chybová hláška

4.3.2.2 UC02 Vytvoření uživatele

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Uživatelé“
2. Webová aplikace zobrazí seznam všech uživatelů v systému
3. Administrátor klikne na tlačítko „Vytvořit uživatele“
4. Webová aplikace nabídne formulář pro vytvoření uživatele s poli pro zadání jména, příjmení, emailové adresy, hesla a role
5. Administrátor vyplní všechny požadované informace

6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna pole vyplněna, korektní formát a unikátnost emailové adresy, síla hesla)
7. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a administrátorovi je zobrazena hláška „Uživatel byl úspěšně vytvořen“
8. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou

4.3.2.3 UC03 Upravení uživatele

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Uživatelé“
2. Webová aplikace zobrazí seznam všech uživatelů v systému
3. Administrátor vybere konkrétního uživatele a na řádku klikne na tlačítko „Upravit uživatele“
4. Webová aplikace nabídne předvyplněný formulář pro upravení uživatele s poli pro zadání jména, příjmení, emailové adresy, hesla a role
5. Administrátor změní všechny požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna pole vyplněna, korektní formát a unikátnost emailové adresy, síla hesla)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a administrátorovi je zobrazena hláška „Uživatel byl úspěšně upraven“
8. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou

4.3.2.4 UC04 Smazání uživatele

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Uživatelé“
2. Webová aplikace zobrazí seznam všech uživatelů v systému

3. Administrátor vybere konkrétního uživatele a na řádku klikne na tlačítko „Smazat uživatele“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce
5. V případě kliknutí na tlačítko „Smazat“ je uživatel smazán z databáze a administrátorovi je zobrazena hláška „Uživatel byl úspěšně smazán“
6. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zpět“ není uživatel smazán

4.3.2.5 UC05 Vytvoření organizace

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Organizace“
2. Webová aplikace zobrazí seznam všech organizací v systému
3. Administrátor klikne na tlačítko „Vytvořit organizaci“
4. Webová aplikace nabídne formulář pro vytvoření organizace s polem pro zadání názvu
5. Administrátor vyplní název organizace
6. Systém po odeslání formuláře zkontroluje správnost vyplněného názvu (zda je vyplněn, unikátnost názvu)
7. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a administrátorovi je zobrazena hláška „Organizace byla úspěšně vytvořena“
8. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou

4.3.2.6 UC06 Upravení organizace

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Organizace“

2. Webová aplikace zobrazí seznam všech organizací v systému
3. Administrátor vybere konkrétní organizaci a na řádku klikne na tlačítko „Upravit organizaci“
4. Webová aplikace nabídne předvyplněný formulář pro upravení organizace s polem pro zadání názvu
5. Administrátor změní název organizace
6. Systém po odeslání formuláře zkontroluje správnost vyplněného názvu (zda je vyplněn, unikátnost názvu)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a administrátorovi je zobrazena hláška „Organizace byla úspěšně upravena“
8. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou

4.3.2.7 UC07 Smazání organizace

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Organizace“
2. Webová aplikace zobrazí seznam všech organizací v systému
3. Administrátor vybere konkrétní organizaci a na řádku klikne na tlačítko „Smazat organizaci“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce
5. V případě kliknutí na tlačítko „Smazat“ je organizace smazána z databáze a administrátorovi je zobrazena hláška „Organizace byla úspěšně smazána“
6. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zpět“ není organizace smazána

4.3.2.8 UC08 Přidání uživatele do organizace

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Organizace“
2. Webová aplikace zobrazí seznam všech organizací v systému
3. Administrátor vybere konkrétní organizaci a na řádku klikne na tlačítko „Uživatelé v organizaci“
4. Webová aplikace zobrazí seznam všech přiřazených uživatelů v organizaci
5. Administrátor klikne na tlačítko „Přidat uživatele“
6. Webová aplikace nabídne výběr všech uživatelů, kteří nejsou v organizaci přiřazeni
7. Administrátor vybere uživatele z nabídnutého seznamu
8. Systém po potvrzení výběru přidá vybraného uživatele do organizace
9. V případě úspěšného přidání uživatele je administrátorovi zobrazena hláška „Uživatel byl úspěšně přidán do organizace“
10. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou

4.3.2.9 UC09 Odstranění uživatele z organizace

Aktéři: Administrátor

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“ nebo „systém“

Scénář:

1. Administrátor klikne v menu na položku „Organizace“
2. Webová aplikace zobrazí seznam všech organizací v systému
3. Administrátor vybere konkrétní organizaci a na řádku klikne na tlačítko „Uživatelé v organizaci“
4. Webová aplikace zobrazí seznam všech přiřazených uživatelů v organizaci
5. Administrátor vybere uživatele ze seznamu a klikne na tlačítko „Odebrat uživatele“
6. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení odebrání a storno akce
7. V případě kliknutí na tlačítko „Odebrat“ je vybraný uživatel odebrán z organizace a administrátorovi je zobrazena hláška „Uživatel byl úspěšně odebrán z organizace“
8. V případě jakékoliv chyby je administrátorovi zobrazena chybová hláška s konkrétní chybou
9. V případě kliknutí na tlačítko „Zpět“ není uživatel z organizace odebrán

4.3.2.10 UC10 Vytvoření aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel klikne na tlačítko „Vytvořit aplikaci“
3. Webová aplikace nabídne formulář pro vytvoření aplikace s polem pro zadání názvu
4. Uživatel vyplní název aplikace
5. Systém po odeslání formuláře zkontroluje správnost vyplněného názvu (zda je vyplněn, unikátnost názvu v rámci organizace)
6. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a uživateli je zobrazena hláška „Aplikace byla úspěšně vytvořena“
7. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.11 UC11 Přejmenování aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Nastavení“
3. Uživatel klikne na tlačítko „Přejmenovat aplikaci“
4. Webová aplikace nabídne předvyplněný formulář pro přejmenování aplikace s polem pro zadání názvu
5. Uživatel vyplní název aplikace
6. Systém po odeslání formuláře zkontroluje správnost vyplněného názvu (zda je vyplněn, unikátnost názvu v rámci organizace)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a uživateli je zobrazena hláška „Aplikace byla úspěšně přejmenována“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.12 UC12 Smazání aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Nastavení“
3. Uživatel klikne na tlačítko „Smazat aplikaci“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce
5. V případě kliknutí na tlačítko „Smazat“ je aplikace smazána z databáze a uživateli je zobrazena hláška „Aplikace byla úspěšně smazána“
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zpět“ není aplikace smazána

4.3.2.13 UC13 Přidání jazyku aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Jazyky“
3. Uživatel klikne na tlačítko „Přidat jazyk“
4. Webová aplikace nabídne formulář pro přidání jazyku aplikace s poli pro zadání názvu a kódu jazyku
5. Uživatel vyplní název aplikace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna pole vyplněna, unikátnost kódu jazyku)
7. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a uživateli je zobrazena hláška „Jazyk aplikace byl úspěšně přidán“

8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.14 UC14 Upravení jazyku aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Jazyky“
3. Uživatel vybere konkrétní jazyk aplikace a klikne na tlačítko „Upravit jazyk“
4. Webová aplikace nabídne předvyplněný formulář pro upravení jazyku s poli pro zadání názvu a kódu jazyku
5. Uživatel změní všechny požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna pole vyplněna, unikátnost kódu jazyku)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a uživateli je zobrazena hláška „Jazyk aplikace byl úspěšně upraven“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.15 UC15 Smazání jazyku aplikace

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Jazyky“
3. Uživatel vybere konkrétní jazyk aplikace a klikne na tlačítko „Smazat jazyk“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce

5. V případě kliknutí na tlačítko „Smazat“ je jazyk aplikace smazán z databáze včetně všech souvisejících překladů a uživateli je zobrazena hláška „Jazyk aplikace byl úspěšně smazán“
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zpět“ není jazyk smazán

4.3.2.16 UC16 Přidání skupiny lokalizačních klíčů

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel klikne na tlačítko „Přidat skupinu“
4. Webová aplikace nabídne formulář pro přidání skupiny s poli pro zadání názvu a volitelného popisu
5. Uživatel vyplní požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna povinná pole vyplněna, unikátnost názvu skupiny v kontextu nadřazené skupiny)
7. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a uživateli je zobrazena hláška „Skupina lokalizačních klíčů byla úspěšně přidána“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.17 UC17 Upravení skupiny lokalizačních klíčů

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací

2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní skupinu lokalizačních klíčů a klikne na tlačítko „Upravit skupinu“
4. Webová aplikace nabídne předvyplněný formulář pro upravení skupiny s poli pro zadání názvu a volitelného popisu
5. Uživatel změní všechny požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna povinná pole vyplněna, unikátnost názvu skupiny v kontextu nadřazené skupiny)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a uživateli je zobrazena hláška „Skupina lokalizačních klíčů byla úspěšně upravena“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.18 UC18 Duplikování skupiny lokalizačních klíčů

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní skupinu lokalizačních klíčů a klikne na tlačítko „Duplikovat skupinu“
4. Systém vytvoří novou skupinu s názvem duplikované skupiny a příponou „*_DUPLICATED*“ včetně všech lokalizačních klíčů v duplikované skupině
5. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.19 UC19 Smazání skupiny lokalizačních klíčů

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní skupinu lokalizačních klíčů a klikne na tlačítko „Smazat skupinu“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce
5. V případě kliknutí na tlačítko „Smazat“ je skupina smazána z databáze včetně všech obsažených lokalizačních klíčů či skupin a uživateli je zobrazena hláška „Skupina lokalizačních klíčů byla úspěšně smazána“
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zpět“ není skupina smazána

4.3.2.20 UC20 Přidání lokalizačního klíče

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel klikne na tlačítko „Přidat klíč“
4. Webová aplikace nabídne formulář pro přidání lokalizačního klíče s poli pro zadání názvu, volitelného popisu a příznaku, zda má být dostupná podpora pluralizace
5. Uživatel vyplní požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna povinná pole vyplněna, unikátnost názvu klíče v kontextu skupiny)
7. Pokud formulář projde úspěšně kontrolou, je v databázi vytvořen nový záznam a uživateli je zobrazena hláška „Lokalizační klíč byl úspěšně přidá“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.21 UC21 Upravení lokalizačního klíče

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní lokalizační klíč a klikne na tlačítko „Upravit klíč“
4. Webová aplikace nabídne předvyplněný formulář pro upravení skupiny s poli pro zadání názvu a volitelného popisu
5. Uživatel změní všechny požadované informace
6. Systém po odeslání formuláře zkontroluje správnost vyplněných údajů (zda jsou všechna povinná pole vyplněna, unikátnost názvu klíče v kontextu skupiny)
7. Pokud formulář projde úspěšně kontrolou, je v databázi upraven existující záznam a uživateli je zobrazena hláška „Lokalizační klíč byl úspěšně upraven“
8. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

4.3.2.22 UC22 Smazání skupiny lokalizačních klíčů

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní lokalizační klíč a klikne na tlačítko „Smazat klíč“
4. Webová aplikace zobrazí dialogové okno s tlačítky pro potvrzení smazání a storno akce
5. V případě kliknutí na tlačítko „Smazat“ je klíč smazán z databáze a uživateli je zobrazena hláška „Lokalizační klíč byl úspěšně smazán“
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

7. V případě kliknutí na tlačítko „Zpět“ není klíč smazán

4.3.2.23 UC23 Upravení překladu lokalizačního klíče

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní lokalizační klíč a klikne na text jednoho z jazyků
4. Webová aplikace zobrazí pole pro vyplnění překladu klíče, předvyplněné poslední verzí překladu
5. Po kliknutí na tlačítko „Potvrdit“ systém vytvoří novou verzi překladu a v případě úspěšného provedení je uživateli zobrazena hláška „Překlad byl úspěšně uložen“
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou
7. V případě kliknutí na tlačítko „Zrušit“ není překlad uložen

4.3.2.24 UC24 Obnovení původní verze překladu

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Překlady“
3. Uživatel vybere konkrétní lokalizační klíč a klikne na tlačítko „Verze překladu“ jednoho z jazyků
4. Webová aplikace zobrazí seznam všech verzí vybraného překladu
5. Uživatel vybere konkrétní verzi překladu a klikne na tlačítko „Obnovit verzi“

6. Systém vytvoří novou verzi překladu na základě vybrané původní verze a v případě úspěchu je uživateli zobrazena hláška „Vybraná předchozí verze překladu byla obnovena“
7. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou
8. V případě kliknutí na tlačítko „Zrušit“ není žádná z verzí překladu obnovena

4.3.2.25 UC25 Export lokalizačních textů do souboru

Aktéři: Administrátor, Klasický uživatel

Předpoklad: Uživatel je přihlášen pod účtem s rolí „administrátor“, „systém“ nebo „klasický uživatel“ s tím, že je přiřazen do vybrané organizace

Scénář:

1. Webová aplikace zobrazí seznam všech aplikací pod vybranou organizací
2. Uživatel vybere konkrétní aplikaci a kliknutím přejde na její detail do záložky „Export“
3. Uživatel vybere jazyk aplikace a formát výstupního souboru
4. Uživatel klikne na tlačítko „Exportovat“
5. Po kliknutí na tlačítko „Exportovat“ jsou všechny překlady lokalizačních klíčů z aktuální aplikace exportovány do souboru vybraného formátu a následně zahájeno jeho stahování
6. V případě jakékoliv chyby je uživateli zobrazena chybová hláška s konkrétní chybou

5 NÁVRH SYSTÉMU

V této kapitole je na základě funkcionálních a nefunkcionálních požadavků, včetně navazujících případů užití, vytvořen návrh systému, sestávající z výběru vhodných technologií a namodelování databázového modelu.

5.1 Výběr technologií

Systém bude rozdělen primárně na dvě části – backendovou, poskytující REST API a webovou, napojující se právě na toto API a poskytující tak uživatelské rozhraní.

5.1.1 REST API

V rámci teoretické části je popsán vývojový PHP framework Phalcon, použitý pro implementaci REST API části systému. Tento framework byl vybrán na základě několika faktorů, jako je jeho rychlost napříč všemi PHP frameworky (díky jeho speciální architektuře), také fakt, že systém bude rozdělen na webovou část a API část (tudíž není potřeba využití nástrojů frameworku pro vytváření pohledů aplikace) a z části dané také autorovou zkušeností s daným frameworkem v profesním životě.

K zachování rychlosti poskytované vybraným frameworkem je vhodné využívat co nejvíce jeho nativních komponent.

5.1.2 Webová aplikace

Pro webovou část systému bude použita platforma Angular, založená na jazyku TypeScript. Tato platforma nabízí více možností než samotný PHP framework Phalcon, co se týče tvorby webového rozhraní. Výhodou rozdělení systému na API a web je fakt, že lze lehce vyměnit pouze webovou část beze změny API části. V takovém případě je možné vytvořit pro systém i mobilní aplikaci, opět bez zásahu do části API.

5.1.3 Databáze

K ukládání veškerých dat systému je vybrána relační databáze PostgreSQL. Do této databáze se bude připojovat jak pomocí ovladače poskytovaného frameworkem Phalcon, tak i přes ORM framework Doctrine.

Databázové migrace budou řešeny také pomocí Doctrine, respektive Doctrine Migration nástroje.

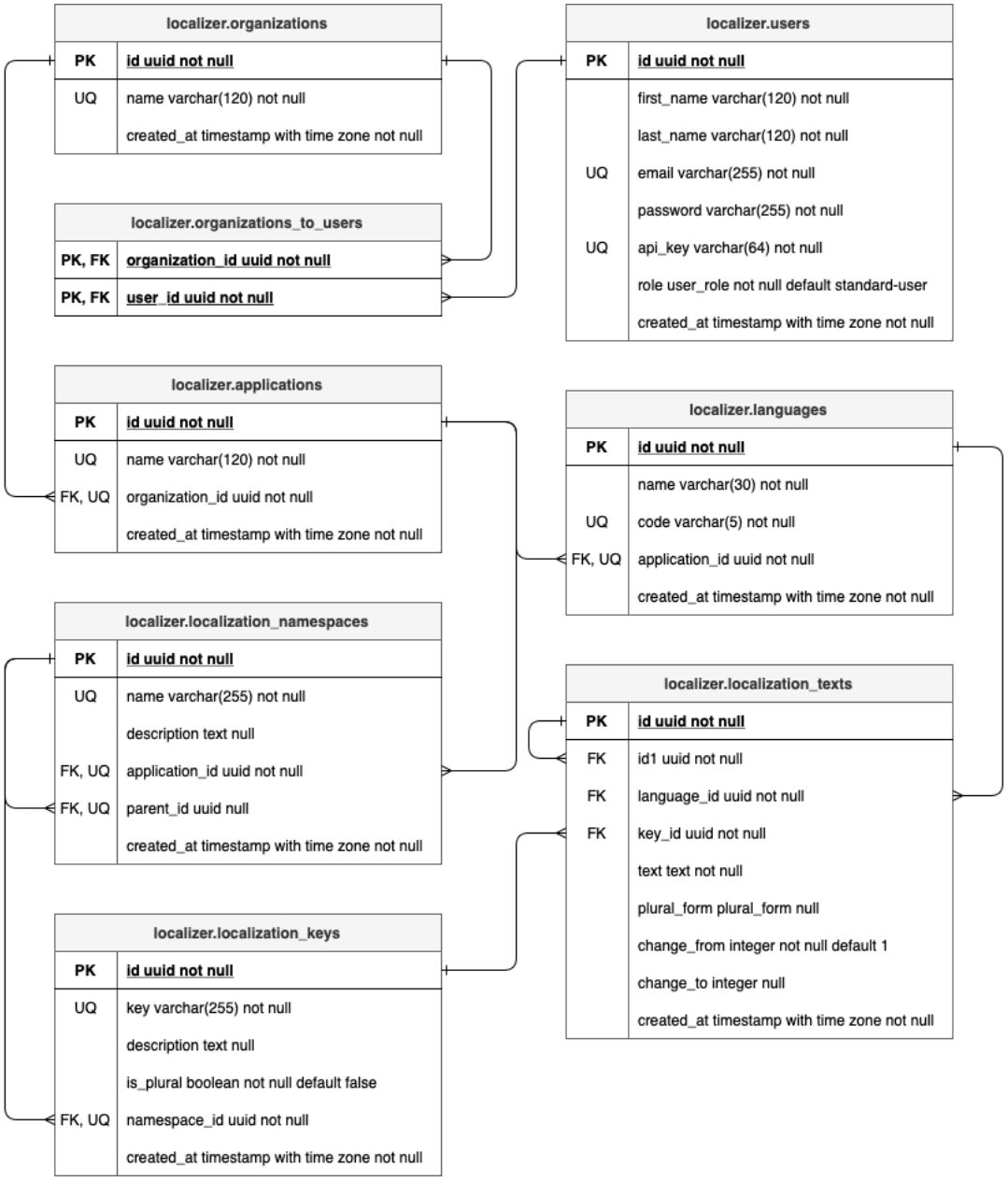
5.1.4 Uložiště souborů

Jedním z nefunkcionálních požadavků je potřeba ukládat soubory jinam než na disk samotného serveru. K dosažení tohoto požadavku je vybráno použití dokumentové databáze MongoDB a systému GridFS.

GridFS rozděluje soubor na části neboli chunky a každý chunk ukládá jako samostatný dokument. Jako výchozí velikost chunků se používá 255 kB – to znamená, že systém GridFS rozdělí soubor na chunky o velikosti 255 kB s výjimkou posledního chunku. Poslední chunk je pouze tak velký, jak je to nutné. Stejně tak soubory, které nejsou větší než velikost chunku, mají pouze poslední chunk a využívají pouze tolik místa, kolik je potřeba. [31]

Systém GridFS používá k ukládání souborů dvě kolekce. Jedna kolekce ukládá právě zmiňované chunky souboru a druhá informace o souboru, jimiž jsou název souboru, velikost souboru a případně i vlastní metadata. [31]

5.2 Databázový model



Obrázek 38 Návrh databázového modelu systému

V podkapitolách jsou popsány jednotlivé tabulky, používané v rámci systému, včetně detailu jejich struktury v grafické podobě.

5.2.1 Tabulka users

Tabulka *users* ukládá údaje o uživatelích, jako je jejich jméno, příjmení, e-mailová adresa a přiřazená role. Role je reprezentována výčtovým typem, obsahující tři možné hodnoty: *standard-user*, *administrator* a *system*. Mimo tyto základní údaje je ukládáno heslo v zashasované podobě a API klíč, obsahující náhodný řetězec o 64 znacích.

localizer.users	
PK	id uuid not null
	first_name varchar(120) not null
	last_name varchar(120) not null
UQ	email varchar(255) not null
	password varchar(255) not null
UQ	api_key varchar(64) not null
	role user_role not null default standard-user
	created_at timestamp with time zone not null

Obrázek 39 Tabulka users

5.2.2 Tabulka organizations

Informace o organizacích se ukládají do tabulky *organizations*. Primárně se jedná o název organizace, který musí být unikátní napříč všemi organizacemi.

localizer.organizations	
PK	id uuid not null
UQ	name varchar(120) not null
	created_at timestamp with time zone not null

Obrázek 40 Tabulka organizations

5.2.3 Tabulka organizations_to_users

Pro přiřazování uživatelů k organizacím je použita vazební tabulka *organizations_to_users*. Každý uživatel může být přiřazen v *N* organizacích a každá organizace může mít přiřazeno *M* uživatelů.

localizer.organizations_to_users	
PK, FK	organization_id uuid not null
PK, FK	user_id uuid not null

Obrázek 41 Tabulka organizations_to_users

5.2.4 Tabulka applications

Ke každé organizaci lze vytvořit *N* aplikací, jež jsou následně uloženy do tabulky *applications* včetně jejich názvu. Tento název musí být unikátní napříč všemi aplikacemi v jedné stejné organizaci.

localizer.applications	
PK	id uuid not null
UQ	name varchar(120) not null
FK, UQ	organization_id uuid not null
	created_at timestamp with time zone not null

Obrázek 42 Tabulka applications

5.2.5 Tabulka languages

Každá aplikace může být překládána do *N* jazyků, reprezentované názvem a kódem. Pro jednu stejnou aplikaci musí být kód jazyku vždy unikátní. Tyto jazyky jsou ukládány do tabulky *languages*.

localizer.languages	
PK	id uuid not null
	name varchar(30) not null
UQ	code varchar(5) not null
FK, UQ	application_id uuid not null
	created_at timestamp with time zone not null

Obrázek 43 Tabulka languages

5.2.6 Tabulka localization_namespaces

Skupiny lokalizačních klíčů jsou ukládány do tabulky *localization_namespaces*. Tyto skupiny jsou vždy vázány na jednu konkrétní aplikaci a lze u nich evidovat název, popis a případně nadřazenou skupinu. Název skupiny musí být unikátní napříč skupinami na stejné úrovni a pro danou aplikaci.

localizer.localization_namespaces	
PK	**id uuid not null**
UQ	name varchar(255) not null
	description text null
FK, UQ	application_id uuid not null
FK, UQ	parent_id uuid null
	created_at timestamp with time zone not null

Obrázek 44 Tabulka localization_namespaces

5.2.7 Tabulka localization_keys

U lokalizačních klíčů se do tabulky *localization_keys* ukládá jeho název, popis, v jaké skupině je klíč používán a příznak, zda má klíč podporovat pluralizaci. Název klíče musí být unikátní v rámci jedné stejné skupiny.

localizer.localization_keys	
PK	**id uuid not null**
UQ	key varchar(255) not null
	description text null
	is_plural boolean not null default false
FK, UQ	namespace_id uuid not null
	created_at timestamp with time zone not null

Obrázek 45 Tabulka localization_keys

5.2.8 Tabulka localization_texts

Veškeré překlady lokalizačních klíčů jsou ukládány do tabulky *localization_texts*. Mimo vazbu na lokalizační klíč, jazyk aplikace a samotný text překladu se eviduje i konkrétní tvar množného čísla (pokud překládaný lokalizační klíč podporuje pluralizaci), reprezentován výčtovým typem, obsahující hodnoty *zero*, *one* a *other*.

Pro možnost verzování jednotlivých překladů jsou použity tyto sloupce:

- **id1** – ukládá identifikátor první verze překladu
- **change_from** – ukládá číslo předchozí verze (číslo 1 pro případ první verze)
- **change_to** – ukládá číslo následující verze (bez hodnoty pro případ poslední verze)

localizer.localization_texts	
PK	**id uuid not null**
FK	id1 uuid not null
FK	language_id uuid not null
FK	key_id uuid not null
	text text not null
	plural_form plural_form null
	change_from integer not null default 1
	change_to integer null
	created_at timestamp with time zone not null

Obrázek 46 Tabulka localization_texts

6 IMPLEMENTACE SYSTÉMU

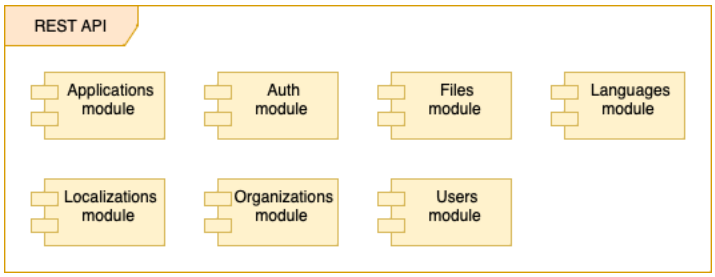
6.1 REST API

Pro implementaci REST API systému byly použity primárně následující technologie v uvedených verzích:

- PHP 8.2
- Phalcon 5.2.1
- Doctrine ORM 2.14.1
- PostgreSQL 15.2
- Redis 7 (poskytuje cache pro Doctrine ORM, pouze v produkčním prostředí)
- MongoDB 4 (starší verze skrze chybějící AVX instrukce na procesoru testovacího serveru [33])

Kód backendové části je rozdělen do dvou hlavních adresářů – *Lib* a *Modules*. V adresáři *Lib* se nachází obecná funkcionalita, nezávislá na doménové logice (komunikace s databází, inicializace potřebných služeb Phalcon frameworku atd.). Adresář *Modules* obsahuje veškeré moduly, využívané v rámci API.

Implementace také využívá získaných poznatků jedné ze knih doporučené literatury, co se týče vývoje v jazyku PHP 8 [34].



Obrázek 47 Dostupné moduly v rámci REST API

6.1.1 Controllers

V adresáři *Controllers* se nachází třídy obsahující veškeré kontrolery, používané v daném modulu. Každý z kontrolerů obsahuje jednotlivé akce dostupné v API. U těchto akcí lze pomocí atributu *Acl* definovat role, jež mají k akci přístup. Jedná se o tenkou část aplikace, mající na starost zpracování požadavku a jeho předání příslušné službě.

```php
#[Acl(roles: [UserRole::ADMINISTRATOR, UserRole::SYSTEM])]
public function createAction(
    CreateUserRequest $createUserRequest
): ResponseCreated {
    return new ResponseCreated(
        $this->userService->createUser($createUserRequest)
    );
}
```

Obrázek 48 Akce kontroleru pro vytvoření uživatele

6.1.2 Dtos

Adresář *Dtos* obsahuje dva typy tříd – DTO objekty a mappery, mapující data z entit na samotné DTO objekty. Tyto objekty definují rozhraní API a na rozdíl od entit neobsahují žádnou dodatečnou logiku.

6.1.3 Entities

Třídy entit, modelující řešenou doménu, jsou obsaženy v adresáři *Entities*. Veškerá doménová logika pracuje pouze s objekty těchto tříd.

6.1.4 Migrations

Součástí jsou veškeré databázové migrace daného modulu, nimiž jsou propagovány změny ve databázové struktury (případně i vkládání základních dat). Migrace zajišťuje nástroj Doctrine Migrations.

6.1.5 Repositories

Veškerá práce s databází, jako je vytváření, upravování a mazání záznamů, se odehrává ve třídách zvaných *Repository*, na základě stejnojmenného návrhového vzoru [35]. Tato třída vytváří abstrakci nad databázovou vrstvou a je primárně zodpovědná za správu všech entit v systému.

Pro získávání entit z databáze je dostupná dvojice tříd – *Criteria* a *CriteriaEvaluator*. Třídy typu *Criteria* jsou pouze nositelem nastavených filtračních parametrů (například text pro vyhledávání, stránkování, řazení atd.). Samotná filtrace probíhá ve třídách typu *CriteriaEvaluator*, kdy je databázový dotaz, před jeho samotným provedením, upraven na základě dodaných filtračních parametrů.

```php
readonly class OrganizationRepositoryCriteriaEvaluator
{
    public function evaluateQueryBuilder(
        OrganizationRepositoryCriteria $criteria,
        QueryBuilder $queryBuilder
    ): QueryBuilder {
        $queryBuilder
            ->setFirstResult(
                ($criteria->getPage() - 1) * $criteria->getLimit()
            )
            ->setMaxResults($criteria->getLimit());

        $queryBuilder
            ->orderBy(OrganizationRepository::QUERY_ALIAS . '.createdAt');

        return $queryBuilder;
    }
}
```

Obrázek 49 Logika stránkování a řazení v seznamu organizací

6.1.6 Requests

Obsahem jsou třídy, zapouzdřující samotný požadavek na API (filtrace záznamů, vytvoření či upravení záznamu). Tato třída je automaticky vytvořena při volání akce v kontroleru (musí být uvedena jako její první argument). Lze zde pracovat s daty, jež jsou v požadavku dostupná (parametry v URL, query parametry, tělo požadavku). Zároveň lze definovat pravidla pro automatickou validaci těla požadavku. Třídy jsou následně předávány službám (kapitola *6.1.7 Services*).

6.1.7 Services

Služby mají za úkol zpracovat příchozí požadavek. V rámci služeb se primárně pracuje s potřebnými repositories (kapitola *6.1.5 Repositories*), díky kterým lze získávat, respektive spravovat, entity v systému a provádět požadovanou logiku. Výstupem těchto služeb jsou vždy příslušné DTO objekty, vytvořené pomocí jejich mapperů (kapitola *6.1.2 Dtos*).

Na obrázku č. 50 je zobrazena metoda, mající na starost získání všech organizací, splňující filtrační parametry z požadavku. Postup je následovný:

1. Vytvoření *Criteria* objektu na základě příchozího požadavku z kontroleru
2. Předání vytvořeného *Criteria* objektu metodě na *Repository*
3. Vytvoření a vrácení DTO objektů pomocí příslušného DTO mapperu

```php
readonly class OrganizationService implements OrganizationServiceInterface
{
    public function findAllOrganizations(
        FilterOrganizationsRequest $filterOrganizationsRequest
    ): array {
        $criteria = (new OrganizationRepositoryCriteria())
            ->setSearch($filterOrganizationsRequest->getSearch())
            ->setPage($filterOrganizationsRequest->getPage())
            ->setLimit($filterOrganizationsRequest->getLimit());

        $organizations = $this->organizationRepository
            ->findAllOrganizations($criteria);

        return $this->organizationDtoMapper->mapMultiple($organizations);
    }
}
```

Obrázek 50 Získání seznamu organizací na základě filtračních parametrů

6.1.8 Validators

Zde jsou obsaženy veškeré speciální validace, které Phalcon framework nenabízí. Typicky se může jednat o kontrolu síly hesla, definované požadavky na systém, nebo také kontrola unikátnosti jednotlivých entit.

6.1.9 ValueObjects

Díky těmto třídám lze zajistit větší typovou bezpečnost. Jedná se primárně o třídy pro identifikátory entit, případně i e-mail atd. V běžném případě je jako identifikátor uváděn pouze primitivní typ (číslo, textový řetězec) a může tak dojít k neplatnému přiřazení, například kvůli nepozornosti. V případě použití těchto tříd by tuto chybu zachytili i moderní editory skrz předávání nesprávného typu.

Dalším z benefitů může být také validace při vytváření těchto objektů (například korektní formát e-mailu).

6.1.10 Module.php

Tato třída zastřešuje celý modul. Dochází zde k provazování URL adres API endpointů s konkrétními akcemi na kontrolerech v rámci příslušných HTTP metod. Důležitou částí je také vytváření definic pro DI kontejner, převážně pro rozhraní a třídy služeb či repositories. Registrace modulu probíhá v souboru *public/index.php*, dostupný z kořenového adresáře projektu.

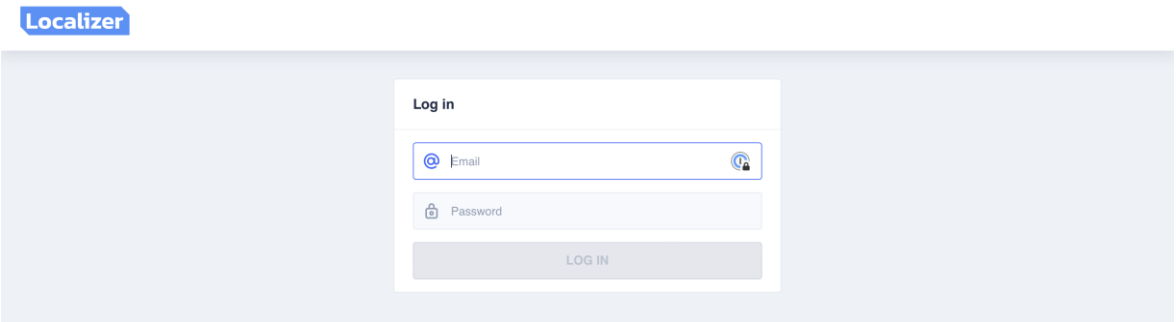
```php
final readonly class Module extends AbstractModule
{
    public function registerModuleServices(DiInterface $container): void
    {
        $container->setShared(AuthServiceInterface::class, [
            'className' => AuthService::class,
            'arguments' => [
                [
                    'type' => 'service',
                    'name' => UserRepositoryInterface::class,
                ],
                [
                    'type' => 'service',
                    'name' => SecurityServiceInterface::class,
                ],
                [
                    'type' => 'service',
                    'name' => LoggedUserDtoMapper::class,
                ]
            ]
        ]);
    }

    public function registerModuleRoutes(RouterInterface $router): void
    {
        $routerGroup = new Group([
            'module' => $this->getModuleName()
        ]);

        $routerGroup->addPost('/api/v1/auth/login', [
            'controller' => AuthController::class,
            'action' => 'login',
        ]);

        $router->mount($routerGroup);
    }
}
```

Obrázek 51 Definice Auth modulu

6.2 Webová aplikace

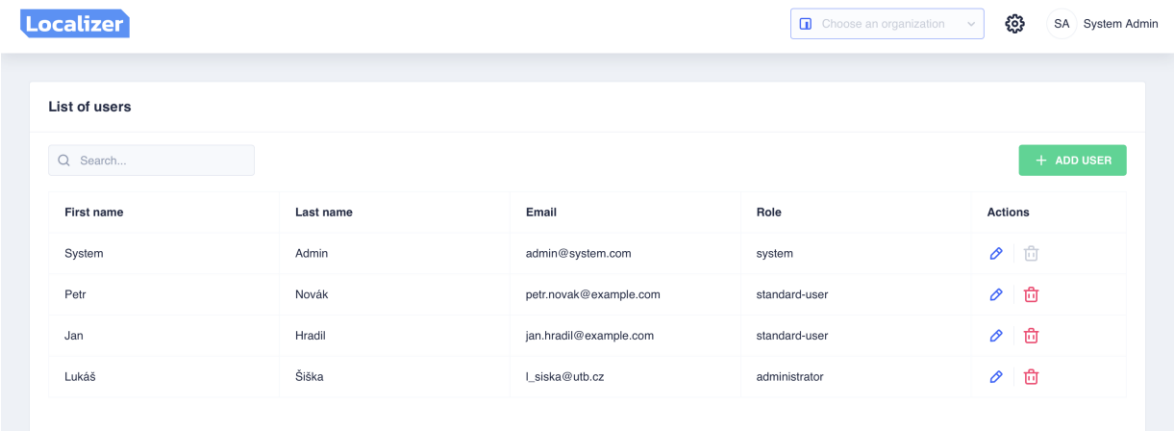
Webová aplikace je vytvořena kompletně za použití frameworku Angular 14 a jazyku TypeScript. Uživatelské rozhraní je z velké části tvořeno pomocí standardních komponent UI knihovny Angularu – Nebular [32].

Pro zpřístupnění veškerých funkcionalit (závislé na uživatelské roli) je vyžadováno přihlášení. Přihlásit se lze pomocí formuláře, jehož podoba je znázorněna na obrázku č. 52.

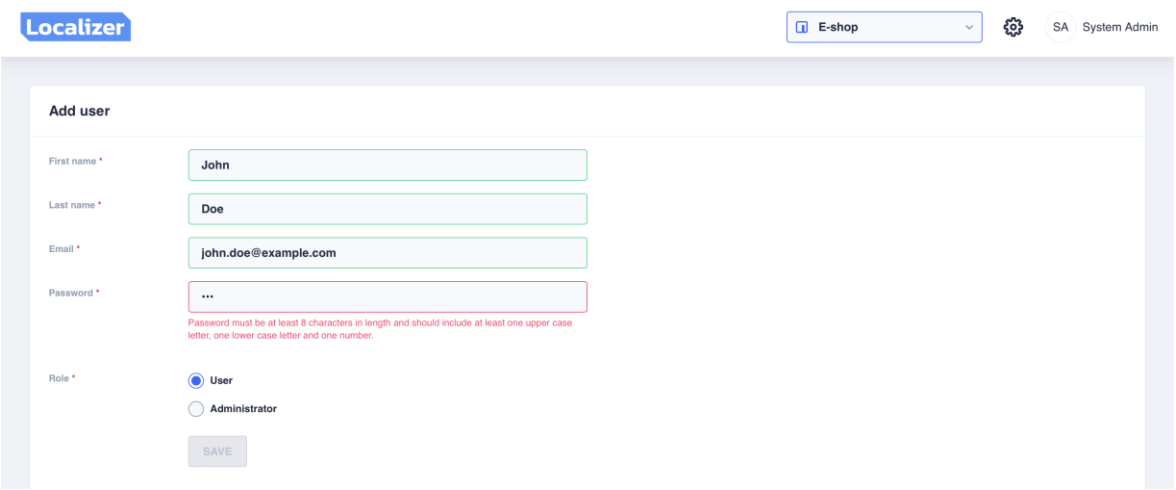


Obrázek 52 Přihlášení do systému

V případě administrátorské role je přístupná správa všech uživatelů v systému. Primární obrazovkou je seznam uživatelů (obrázek č. 53), nabízející vyhledávání dle jména, příjmení a e-mailu. Uživatele lze také přidávat (obrázek č. 54), upravovat a mazat (vyjma systémového uživatele, jež smazat nelze) pomocí dostupných akčních tlačítek. Stejným způsobem i rozhraním je pro administrátora dostupná správa organizací a přiřazování uživatelů do jednotlivých organizací.

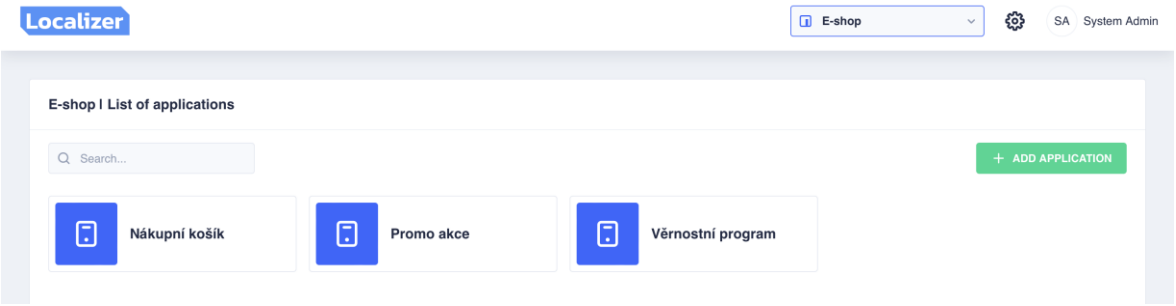


Obrázek 53 Seznam uživatelů v systému



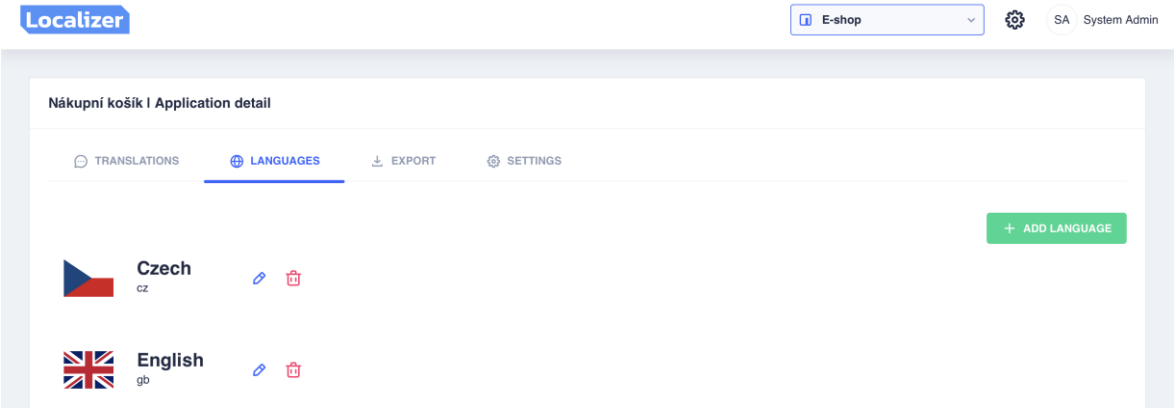
Obrázek 54 Přidání uživatele do systému

V záhlaví každé ze stránek (vyjma přihlášení) je selektor pro výběr jedné z organizací, do kterých je přihlášený uživatel přiřazen. Po kliknutí na některou z organizací se zobrazí seznam všech aplikací pod vybranou organizací (obrázek č. 55). Mimo vyhledávání aplikací dle jejich názvu je dostupné také akční tlačítko pro vytvoření aplikace.



Obrázek 55 Seznam aplikací v organizaci

Po kliknutí na jednu z aplikací v seznamu (obrázek č. 55) je zobrazen detail samotné aplikace. Důležitou součástí je záložka pro správu jazyků aplikace (obrázek č. 56), do kterých bude aplikace následně překládána. V tomto pohledu lze jazyky přidávat, upravovat i mazat. Pokud je kód jazyku obsažen v určité množině kódů, jež jsou spjaté s některým ze států, je zobrazena i vlajka tohoto státu.



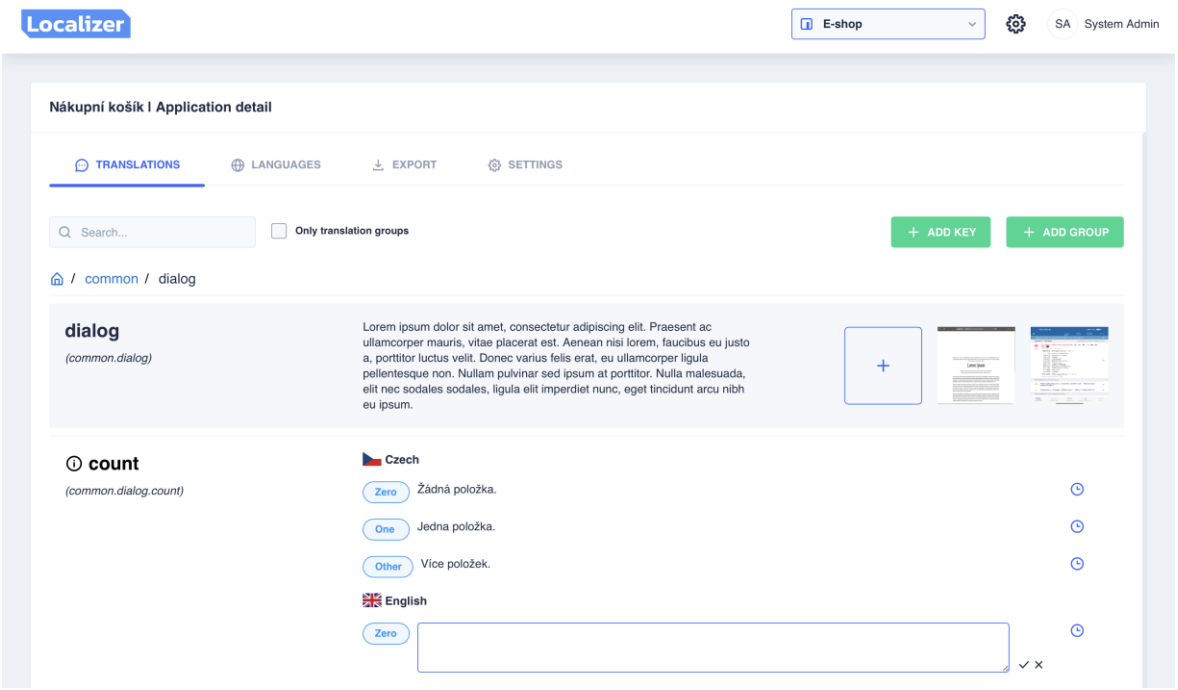
Obrázek 56 Seznam jazyků aplikace

Nejdůležitější částí na detailu vybrané aplikace je záložka s překlady lokalizačních klíčů pro veškeré přiřazené jazyky (obrázek č. 57). Tento pohled je primárně zaměřen na aktuálně vybranou skupinu překladů, v rámci, které lze lokalizační klíče, respektive skupiny lokalizačních klíčů, přidávat, upravovat či mazat. Dostupné je vyhledávání dle názvů klíčů i skupin v kontextu aktuálně vybrané skupiny, případně si lze zobrazit pouze skupiny bez klíčů.

Pro informaci o zanoření aktuálně vybrané skupiny je dostupná tzv. breadcrumb navigace, která zobrazuje kompletní cestu od kořenové skupiny až po aktuální zanoření.

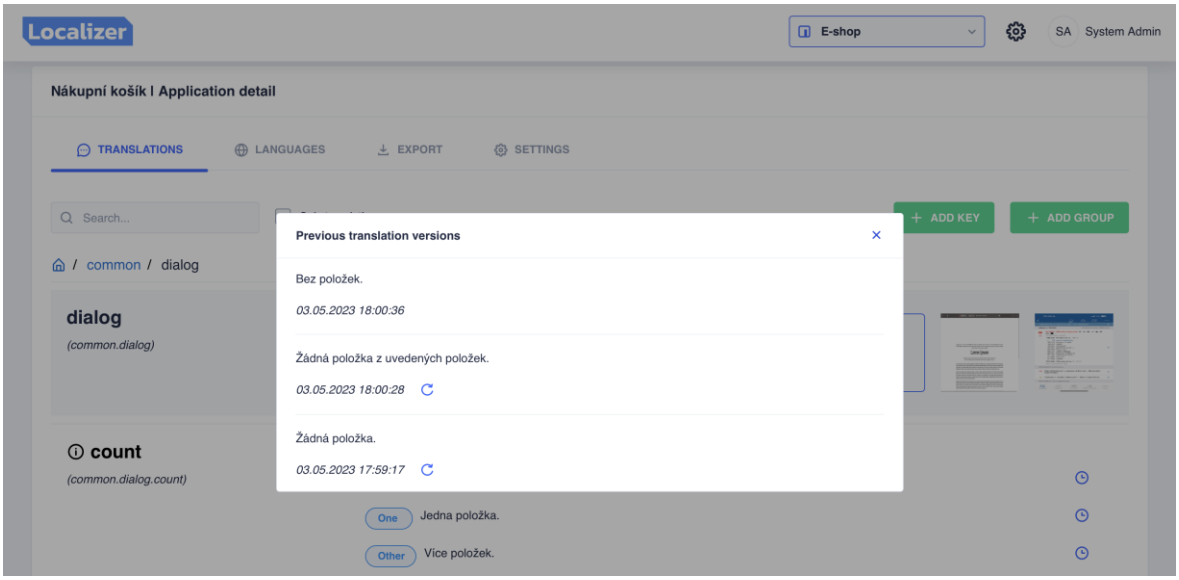
Zobrazení obsahuje vždy informace o aktuálně vybrané skupině, jako je kompletní cesta (sestavená z názvů předchozích skupin, oddělená tečkou), popis skupiny a případně i doplňující obrázky, které lze ke skupině přikládat.

Část zobrazení s lokalizačním klíčem primárně obsahuje vstupní textové pole pro každý jazyk aplikace, či i více polí v případě množného čísla.



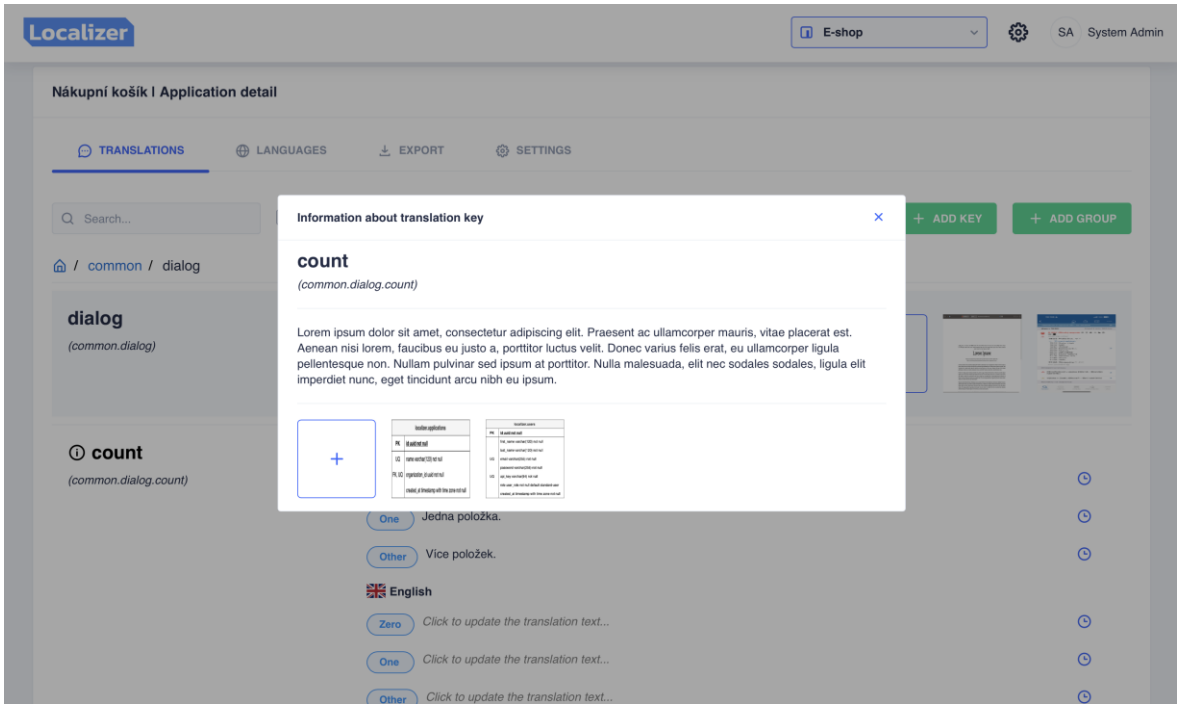
Obrázek 57 Přehled skupiny lokalizačních klíčů

Na každém řádku, obsahující překlad, je dostupné akční tlačítko pro zobrazení dialogového okna se seznamem všech verzí tohoto překladu (obrázek č. 58). V seznamu lze vidět text původní verze a datum jejího vytvoření. Součástí je také akční tlačítko pro obnovení konkrétní verze.



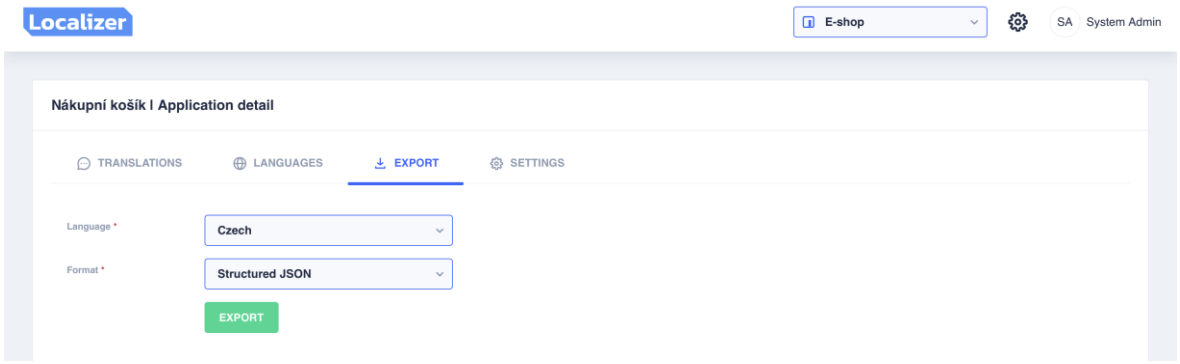
Obrázek 58 Seznam verzí překladu

Kliknutím na název lokalizačního klíče (případně ikonu „i“) je zobrazeno dialogové okno s informacemi o daném klíči (obrázek č. 59), jako je jeho kompletní cesta (sestavená z názvu skupiny, do níž klíč patří, a názvů všech předchozích skupin, oddělené tečkou), informace o klíči a doplňující obrázky, které lze v tomto okně také přikládat.



Obrázek 59 Detail lokalizačního klíče

Na detailu aplikace je dostupná záložka v rámci, které lze exportovat veškeré překlady lokalizačních klíčů aplikace do souboru vybraného formátu (obrázek č. 60).

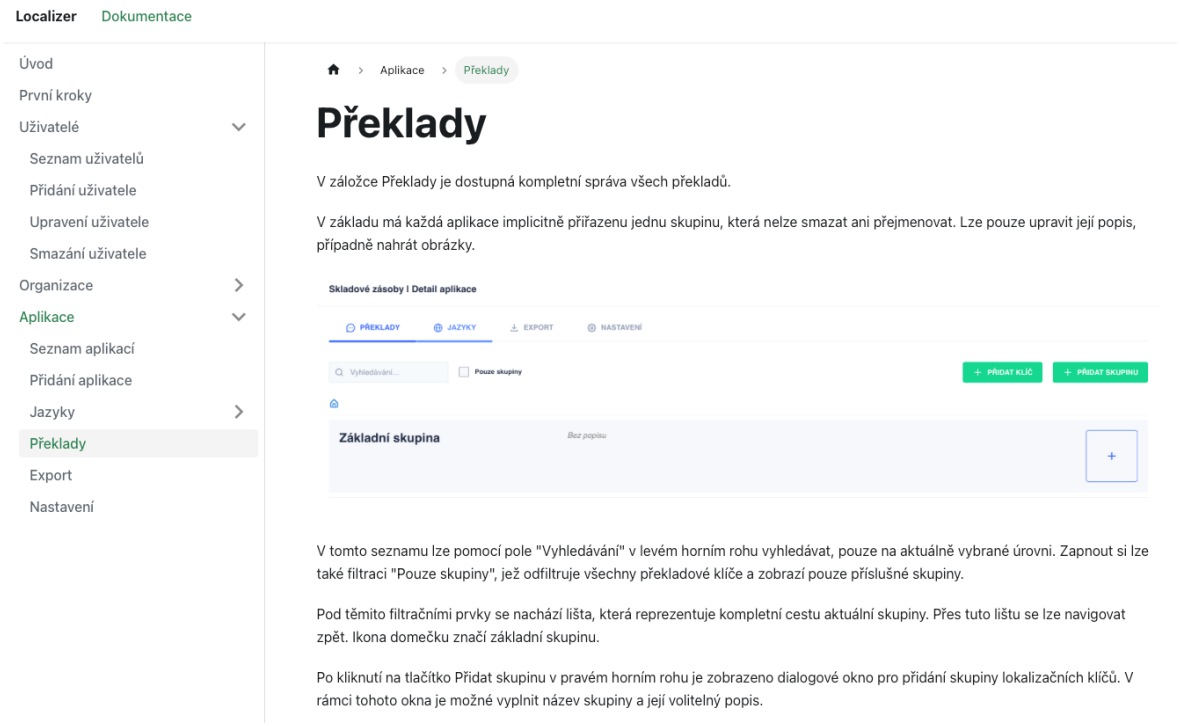


Obrázek 60 Export jazyku do souboru

6.3 Dokumentace

Kompletní popis, včetně snímků obrazovek, je součástí dokumentace systému. Aktuálně je dostupná online na adrese *docs.localizer.siskalukas.com*, případně i lokálně v příloze práce.

K vytvoření dokumentace je využito nástroje Docusausus [47]. Na obrázku č. 61 je ukázka části její online verze.



Obrázek 61 Ukázka online dokumentace systému

7 PROVOZ SYSTÉMU

7.1 REST API

V následujících podkapitolách jsou popsány postupy, jakými lze docílit provoz backendové části systému, jak na vývojovém, tak produkčním prostředí.

7.1.1 Základní Docker image

Základní Docker image vychází z oficiálního image *php:8.2-apache*, obsahující i webový server Apache2. Konfigurace webového serveru je upravena dle potřeb pro provoz API, jako je povolení potřebných HTTP metod, hlaviček a definice umístění vstupního bodu – *index.php*. Následně dochází k instalaci samotného Phalcon frameworku (ukázka instalace je zobrazena na obrázku č. 62) a ovladačů pro připojení a práci s MongoDB a PostgreSQL databází. Důležitá je také instalace nástroje Composer [48], jež poskytuje jednoduchou správu knihoven a zajišťuje tak i automatické načítání jejich tříd v různých částech kódu.

```
ARG PHALCON_VERSION=5.2.1

# Install Phalcon
RUN set -xe && \
    docker-php-source extract && \
    # Install ext-phalcon
    curl -LO
https://github.com/phalcon/cphalcon/archive/v${PHALCON_VERSION}.tar.gz && \
    tar xzf ${PWD}/v${PHALCON_VERSION}.tar.gz && \
    docker-php-ext-install -j $(getconf _NPROCESSORS_ONLN) \
        ${PWD}/cphalcon-${PHALCON_VERSION}/build/phalcon \
    && \
    # Remove all temp files
    rm -r \
        ${PWD}/v${PHALCON_VERSION}.tar.gz \
        ${PWD}/cphalcon-${PHALCON_VERSION} \
    && \
    docker-php-source delete
```

Obrázek 62 Ukázka instalace Phalcon frameworku

7.1.2 Vývojové prostředí

K lokálnímu vývoji celého systému je použit vlastní Docker image (definován v samostatném *Dockerfile* souboru), vycházející ze základního (jehož popis je uveden v kapitole *7.1.1 Základní Docker image*). Tento image přidává nástroj Xdebug [49], umožňující vývojářům postupně krokovat svůj kód, případně provádět různé výkonnostní testy. Pro zajištění nejen typové bezpečnosti je také instalován nástroj, poskytující statickou analýzu kódu – PHPStan [50]. Ukázka instalace těchto dvou nástrojů v rámci vývojového *Dockerfile* souboru je uvedena na obrázku č. 63.

```
ARG PHPSTAN_VERSION=1.10.14

# Install Xdebug
RUN set -xe && \
    pecl install xdebug && \
    echo 'zend_extension=xdebug.so' >> /usr/local/etc/php/php.ini && \
    echo 'xdebug.mode=debug' >> /usr/local/etc/php/php.ini && \
    echo 'xdebug.client_host=host.docker.internal' >>
/usr/local/etc/php/php.ini && \
    echo 'xdebug.client_port=9000' >> /usr/local/etc/php/php.ini && \
    echo 'xdebug.start_with_request=yes' >> /usr/local/etc/php/php.ini

# Install PHPStan
RUN set -xe && \
    wget -O phpstan.phar
https://github.com/phpstan/phpstan/releases/download/${PHPSTAN_VERSION}/php
stan.phar && \
    chmod a+x phpstan.phar && \
    mv phpstan.phar /usr/local/bin/phpstan
```

Obrázek 63 Instalace nástrojů Xdebug a PHPStan

Ke spuštění kontejneru s backendovou částí systému a dalších závislostí, jako je PostgreSQL databáze, Redis či MongoDB, je vytvořena samostatná Docker compose definice. Definice jednotlivých kontejnerů pro závislosti třetích stran jsou vytvořeny na základě jejich oficiální dokumentace.

Vlastní definice pro backendový kontejner primárně určuje, v jaké cestě se nachází soubor *Dockerfile*, ze kterého je kontejner následně sestaven (sekce *build* na obr. č. 64). Chování kontejneru je ovlivňováno proměnnými prostředí (sekce *environment* na obr. č. 64), kterými lze kupříkladu měnit parametry PHP. Pro lokální vývoj jsou do kontejneru zrcadleny veškeré soubory, aby se změny v kódu propisovaly instantně (sekce *volumes* na obr. č. 64). Výsledný kontejner je poté dostupný na lokální síti s adresou *localhost:32000*, kdy je po přístupu na tuto adresu směrována komunikace na port *80* uvnitř tohoto kontejneru (sekce *ports* na obr. č. 64).

Součástí vývojového prostředí je soubor *compose.sh*, usnadňující práci s *docker compose* příkazem. Obsahem tohoto souboru je jednořádkový příkaz *docker compose -p localizer_backend -f ./docker/devel/docker-compose.yml $@*, který dokáže zpracovat jakýkoliv *docker compose* argument.

Před spuštěním backendového kontejneru je potřeba nainstalovat veškeré knihovní závislosti pomocí příkazu *./docker/compose.sh run --rm backend composer install --prefer-source*.

Spuštění kompletního vývojového prostředí je provedeno příkazem v kořenovém adresáři projektu – *./docker/compose.sh up -d*.

Pro aplikování databázových migrací je v rámci nástroje Composer dostupný script, spustitelný příkazem *./docker/compose.sh run --rm backend composer run database-migration* z kořenového adresáře projektu.

```
backend:
  build: .
  restart: "unless-stopped"
  ports:
    - "32000:80"
  environment:
    - APPLICATION_ENV=devel
    - XDEBUG_CONFIG
    - TZ=UTC
    - PHP_MAX_EXECUTION_TIME=120
    - PHP_MEMORY_LIMIT=2048M
    - PHP_POST_MAX_SIZE=128M
    - PHP_UPLOAD_MAX_FILESIZE=32M
  volumes:
    - ${PWD}/app:/var/www/app:rw
    - ${PWD}/bin:/var/www/bin:rw
    - ${PWD}/configs:/var/www/configs:rw
    - ${PWD}/data:/var/www/data:rw
    - ${PWD}/logs:/var/www/logs:rw
    - ${PWD}/public:/var/www/public:rw
    - ${PWD}/tests:/var/www/tests:rw
    - ${PWD}/vendor:/var/www/vendor:rw
    - ${PWD}/composer.json:/var/www/composer.json:rw
    - ${PWD}/composer.lock:/var/www/composer.lock:rw
    - ${PWD}/phpunit.xml:/var/www/phpunit.xml:rw
  depends_on:
    - postgres
    - mongodb
    - redis
  networks:
    local:
```

Obrázek 64 Definice backendového kontejneru

7.1.3 Produkční prostředí

Obdobně jako u vývojového prostředí je použit vlastní Docker image, opět definován v samostatném *Dockerfile* souboru, vycházející ze základního (jehož popis je uveden v kapitole *7.1.1 Základní Docker image*). Na rozdíl od image pro vývojové prostředí se nepřidávají další závislosti, jelikož vše potřebné pro chod systému je již součástí toho základního. Během sestavování tohoto image se nakopírují veškeré zdrojové soubory a adresáře, potřebné pro běh backendové části, a následně je spuštěna instalace knihovních závislostí pomocí nástroje Composer, příkazem *composer install*.

K testovacímu provozu produkčního prostředí byl zvolen nástroj Docker Swarm, sloužící pro orchestraci kontejnerů. Veškeré závislosti, včetně samotného backendové kontejneru, jsou uvedeny v Docker compose definici. Rozdíl v definici oproti vývojovému prostředí je takový, že backendový kontejner nevyžaduje přímé zrcadlení souborů mezi diskem a

kontejnerem. Hlavním rozdílem je ale předávání konfiguračních hodnot, jak už pro backendový, tak i ostatní kontejnery.

Pro předávání citlivých konfiguračních hodnot (např. přístupy k databázím atd.) je v rámci Docker Swarm využito tzv. secrets. Jednotlivé secrets je nutné explicitně přidělovat vybraným kontejnerům, které mají mít k dané citlivé hodnotě přístup. Vytvoření secret lze docílit spuštěním příkazu *printf "localizer" | docker secret create postgres_database -*.

```
services:
  backend:
    image: docker.io/lukas3k11/localizer-backend:production
    environment:
      - REDIS_HOST_FILE=/run/secrets/redis_host
      - REDIS_PORT_FILE=/run/secrets/redis_port
    secrets:
      - source: localizer_redis_host
        target: /run/secrets/redis_host
      - source: localizer_redis_port
        target: /run/secrets/redis_port

secrets:
  localizer_redis_host:
    external: true
  localizer_redis_port:
    external: true
```

Obrázek 65 Použití Docker secrets

Na obrázku č. 65 je uvedena zjednodušená definice backendového kontejneru pro ukázku použití Docker secrets. Hodnoty ze secrets jsou v daném kontejneru dostupné v adresáři */run/secrets/*, kde je pro každý secret vytvořen zvláštní soubor, nesoucí hodnotu v textové podobě. U většiny Docker images třetích stran je dostupná varianta proměnné prostředí pro předávání hodnoty v souboru. Taková proměnná má příponu *_FILE* a je jí v rámci daného kontejneru předána cesta k souboru s hodnotou.

Aby bylo možné dotazovat API pomocí klasické domény, bylo zvoleno využití reverzní proxy Nginx [51] a automatického generování SSL certifikátů od certifikační autority Let's Encrypt [52]. V reálném produkčním prostředí by se měl využít vlastní zakoupený SSL certifikát pro konkrétní doménu.

```
services:
  backend:
    image: docker.io/lukas3k11/localizer-backend:production
    environment:
      - VIRTUAL_HOST=api.localizer.siskalukas.com
      - LETSENCRYPT_HOST=api.localizer.siskalukas.com
    networks:
      nginx-proxy:

networks:
  nginx-proxy:
    external: true
```

Obrázek 66 Zapojení kontejneru do sítě Nginx proxy

Na základě hodnoty proměnné prostředí *VIRTUAL_HOST*, uvedené na konkrétním kontejneru, je schopna Nginx proxy nasměrovat komunikaci ze zadané domény do správného kontejneru. SSL certifikát od certifikační autority Let's Encrypt je automaticky vystavován, případně obnovován, pro doménu uvedenou v proměnné *LETSENCRYPT_HOST*.

Veškeré zde popsané postupy provozování produkčního prostředí, vyjma tvorby Docker image, lze nahradit postupy vlastními. Dodaný image lze kompletně nakonfigurovat pomocí proměnných prostředí (například přístupy k databázím atd.).

7.2 Webová aplikace

V následujících podkapitolách jsou popsány postupy, jakými lze docílit provoz webové části systému, jak na vývojovém, tak produkčním prostředí.

7.2.1 Vývojové prostředí

Pro vývojové prostředí je připraven Docker image, v podobě vlastního *Dockerfile* souboru, vycházející z oficiálního image pro Node.js [53]. Součástí tohoto image je nakopírování potřebných souborů (ne zdrojových souborů), instalace CLI nástroje pro Angular 14 [54] a příprava příkazu, jenž je proveden, jakmile dojde k vytvoření samotného kontejneru.

Jako první je nutné sestavit definovaný image pomocí příkazu *docker build -t localizer-frontend:devel* .. Po úspěšném sestavení je možné spustit kontejner příkazem *docker run -it --rm -v $(pwd):/usr/src/app localizer-frontend:devel bash*. Tímto příkazem se docílilo spuštění konzole uvnitř kontejneru. V konzoli lze ovládat CLI nástroj Angularu, ale především instalovat veškeré závislosti za použití příkazu *npm install*. Jakmile jsou všechny závislosti nainstalovány, je možné spustit další kontejner, tentokrát již s běžící webovou

aplikací, pomocí příkazu *docker run -it --rm -p 4200:4200 -v $(pwd):/usr/src/app localizer-frontend:devel*. Aplikace bude poté dostupná v lokální síti na adrese *localhost:4200*.

Na rozdíl od backendové části není třeba definovat zvláštní Docker compose definici, jelikož webová část nevyžaduje žádné další závislosti, co se kontejnerů týče.

7.2.2 Produkční prostředí

Docker image pro produkční prostředí, respektive jeho definice v *Dockerfile* souboru, je rozdělena na dvě fáze. Během první fáze dochází ke kopírování veškerých souborů (včetně těch zdrojových), instalaci Angular CLI nástroje a finální produkční sestavení se zapnutou Ahead-of-time (AOT) kompilací pomocí příkazu *ng build --configuration production –aot*. Druhá fáze vychází z odlehčené verze Nginx Docker image (alpine) a kopíruje tak již zkompilované soubory z předchozí fáze do adresáře */usr/share/nginx/html*, ze kterého je následně obsah poskytován.

```
services:
  frontend:
    image: docker.io/lukas3k11/localizer-frontend:production
    deploy:
      mode: global
    environment:
      - VIRTUAL_HOST=localizer.siskalukas.com
      - LETSENCRYPT_HOST=localizer.siskalukas.com
      - BACKEND_BASE_URL=https://api.localizer.siskalukas.com
    networks:
      nginx-proxy:
```

Obrázek 67 Definice frontendového kontejneru

Poskytovaný Docker image webové části přijímá pouze jedinou vlastní proměnnou prostředí, a tou je *BACKEND_BASE_URL*, definující adresu, na které se nachází API systému. Postup nasazení pomocí nástroje Docker Swarm a Nginx je totožný s backendovou částí, uvedený v kapitole *7.1.3 Produkční prostředí*.

7.3 Testovací provoz

Otestování hlavních funkcionalit systému – definování překladů dělených do skupin a následný export těchto překladů – bylo provedeno již při zavádění lokalizace, respektive překladu webové aplikace vytvářeného systému do českého a anglického jazyku. Po nadefinování překladů pro jednotlivé části aplikace byl proveden export do formátu strukturovaného JSON dokumentu, načež byly tyto soubory beze změny použity v rámci knihovny ngx-translate pro Angular. Výsledkem je tedy kompletně přeložený vytvářený systém, do českého a anglického jazyku, dostupný na adrese *localizer.siskalukas.com*.

8 NÁVRH BUDOUCÍHO ROZVOJE

Vytvořený systém je možné v budoucnu rozšiřovat o další funkcionality pro zefektivnění jak překládání aplikací, tak i samotné distribuce překladů.

Důležitou součástí systému by měla být možnost importovat již existující překlady ze souborů různých formátů. Touto funkcionalitou lze do procesu překladu aplikací zahrnout i externí subjekty bez přístupu do systému, využívající například vlastní software pro správu překladů. Uživatel systému tak může existující překlady exportovat do souboru, poslat externímu subjektu a následně jej naimportovat zpět do systému.

Systém podporuje export překladů do souboru pouze ve formátu JSON, což aktuálně splňuje základní potřeby většiny aplikací. Možným vylepšením této části je rozšíření o další formáty (například Gettext a jiné).

Mezi další funkcionalitu, vylepšující uživatelské pohodlí, je přidání možnosti kopírování veškerého obsahu jedné aplikace, co se týče lokalizačních klíčů a jejich skupin včetně doplňujících informací, do aplikace druhé v rámci stejné organizace. Pro administrátory by mohla být dostupná tato možnost i pro kopírování aplikací mezi dvěma různými organizacemi.

Vylepšením by mohla projít také podpora pluralizace, jelikož aktuálně jsou vždy dostupné pouze tři pevně definované tvary množného čísla. Uživatel by si mohl tyto tvary odděleně definovat pro každý jazyk aplikace.

Zajímavou funkcionalitou může být podpora automatizovaného překladu textů pomocí různých nástrojů třetích stran.

Aktuálně je distribuce překladů řešena pouze exportem do souboru, jež musí být proveden ručně uživatelem. K zajištění větší míry automatizace se nabízí využití tzv. webhooků, reagující na předem definované události, vyvolávané během práce se systémem. Lze tak rychle reagovat na případné změny v překladech.

Prostor pro zlepšení je také v samotném uživatelském rozhraní webové aplikace, které je aktuálně sestaveno ze standardních komponent a stylů UI knihovny Angularu. V tomto případě je ideální mít vlastní design, reprezentující samotný systém, splňující určité UX zvyklosti a pravidla pro tvorbu tzv. bezbariérového webu.

Jelikož je systém navržen primárně pro nasazení v rámci jedné organizace, není tak v první fázi kladen důraz na přístupová práva, co se přiřazených organizací týče. Tento nedostatek by bylo nutné vyřešit v případě, kdy by systém obsluhoval více než jednu organizaci.

Samozřejmostí je kontinuální oprava případných nedostatků a chyb, postupně získávaných při intenzivnější práci se samotným systémem.

ZÁVĚR

Hlavním cílem diplomové práce bylo vytvoření lehce doručitelného systému pro správu a distribuci lokalizačních textů aplikacím.

Celá práce se skládá ze dvou hlavních částí. V první, teoretické části, byla provedena analýza již existujících řešení, obdobného charakteru, po níž následovala úvaha nad potřebou vytvoření vlastního systému. Další kapitola se podrobněji zabývá vybraným PHP frameworkem, Phalcon, jež je využit v rámci implementace backendové části vytvářeného systému. Součástí této kapitoly je shrnutí historie frameworku, popis jeho speciální architektury v oblasti dostupných PHP frameworků, včetně následného srovnání a seznámení se s jeho důležitými vlastnostmi, které byly později využity při implementaci navrhovaného systému. Závěrem teoretické části je přiblížena problematika lokalizačních klíčů a samotné lokalizace aplikací. Veškeré znalosti, získané z této části, byly využity při následném návrhu a implementaci výsledného systému.

V rámci druhé, praktické části, byl na základě analýzy existujících řešení v části teoretické a vlastních požadavků, vytvořen návrh systému. Návrh primárně sestává ze seznamu funkcionálních a nefunkcionálních požadavků, aktérů a případů užití. Dle těchto požadavků byly vybrány vyhovující technologie a nástroje, sestaven databázový model a naimplementován samotný systém, rozdělený na backendovou a webovou část včetně vytvoření uživatelské dokumentace, použitím vývojových frameworků Phalcon a Angular. V neposlední řadě je popsán návrh a ukázka provedení nasazení systému do produkčního prostředí, čemuž předchází příprava prostředí vývojového.

Výsledný systém obsahuje pouze nejdůležitější funkcionality pro správu a poskytování lokalizačních textů. Poslední kapitola praktické části je tak věnována rozboru možného budoucího rozvoje systému, co se funkcionalit týče.

Výsledkem celé práce je tedy plně soběstačný systém, poskytující rozhraní pro správu aplikací a jejich lokalizací, včetně možnosti logického členění textů do skupin a přikládání dodatečných informací, jakými jsou obrázky a případný popis. Distribuce vzniklých lokalizací je řešena exportem do souboru vybraného formátu. Kompletní systém lze nasadit v podobě dvou kontejnerových aplikací na základě vzniklých Docker images. I přes testování během vývoje a testovacího provozu se v systému můžou objevovat různé chyby a nedostatky, které budou postupně eliminovány v rámci dalšího rozvoje systému.

SEZNAM POUŽITÉ LITERATURY

[1] The Future of Phalcon [online]. Phalcon Team, 2020 [cit. 2023-02-05]. Dostupné z: https://blog.phalcon.io/post/the-future-of-phalcon

[2] Phalcon/cphalcon [online]. GitHub: Phalcon Team [cit. 2023-02-05]. Dostupné z: https://github.com/phalcon/cphalcon

[3] Team - Phalcon Framework [online]. Phalcon Team [cit. 2023-02-05]. Dostupné z: https://phalcon.io/en-us/team

[4] Phalcon/ide-stubs [online]. GitHub: Phalcon Team [cit. 2023-02-05]. Dostupné z: https://github.com/phalcon/ide-stubs

[5] Zephir Documentation - Installation [online]. Zephir Team, c2013-2023 [cit. 2023-03-20]. Dostupné z: https://docs.zephir-lang.com/0.12/en/installation

[6] Phalcon SVG Vector Logos [online]. Vector Logo Zone, 2018 [cit. 2023-02-05]. Dostupné z: https://www.vectorlogo.zone/logos/phalconphp/index.html

[7] What is a Framework? [online]. Medium: Alex Zelinsky, 2022 [cit. 2023-02-05]. Dostupné z: https://medium.com/@alexzelinsky124/what-is-a-framework-31852ff8553b

[8] Zephir Documentation - Introducing Zephir [online]. Zephir Team, c2013-2023 [cit. 2023-03-20]. Dostupné z: https://docs.zephir-lang.com/0.12/en/introduction

[9] Phalcon Documentation - Devtools [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/devtools

[10] Phalcon Documentation - Testing Environment [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/testing-environment

[11] Phalcon Documentation - Application [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/application

[12] Phalcon Documentation - Config [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/config

[13] Phalcon Documentation - Routing [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/routing

[14] Phalcon Documentation - Dispatcher [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/dispatcher

[15] Phalcon Documentation - Dependency Injection / Service Locator [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/di

[16] Phalcon Documentation - Controllers [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/controllers

[17] Phalcon Documentation - Events Manager [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/events

[18] Phalcon Documentation - Validation [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/filter-validation

[19] Phalcon Documentation - Micro Application [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/application-micro

[20] Phalcon Documentation - HTTP Request [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/request

[21] Phalcon Documentation - Filter [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/filter-filter

[22] Phalcon Documentation - HTTP Response [online]. Phalcon Team, c2012-2023 [cit. 2023-03-20]. Dostupné z: https://docs.phalcon.io/5.0/en/response

[23] RFC 9110: HTTP Semantics [online]. R. Fielding, Ed., M. Nottingham, Ed., J. Reschke, Ed., 2022 [cit. 2023-03-20]. Dostupné z: https://www.rfc-editor.org/rfc/rfc9110.html

[24] Phalcon Documentation [online]. Phalcon Team, c2012-2023 [cit. 2023-03-21]. Dostupné z: https://docs.phalcon.io/5.0/en

[25] Phalcon Documentation - Volt: Template Engine [online]. Phalcon Team, c2012-2023 [cit. 2023-03-21]. Dostupné z: https://docs.phalcon.io/5.0/en/volt

[26] Phalcon Documentation - Model Events [online]. Phalcon Team, c2012-2023 [cit. 2023-03-21]. Dostupné z: https://docs.phalcon.io/5.0/en/db-models-events

[27] Translation Management for software projects | SimpleLocalize [online]. Jakub Pomykała, 2023 [cit. 2023-04-01]. Dostupné z: https://simplelocalize.io

[28] A Localization and Translation Software Tool | Lokalise [online]. Lokalise, c2017-2023 [cit. 2023-04-01]. Dostupné z: https://lokalise.com

[29] Loco - Translation Management System [online]. White Interactive, c2023 [cit. 2023-04-15]. Dostupné z: https://localise.biz

[30] Localization Management Platform for agile teams | Crowdin [online]. Crowdin, c2023 [cit. 2023-04-20]. Dostupné z: https://crowdin.com

[31] GridFS - MongoDB Manual [online]. MongoDB, c2023 [cit. 2023-04-26]. Dostupné z: https://www.mongodb.com/docs/manual/core/gridfs

[32] Nebular [online]. Akveo, c2015-2019 [cit. 2023-04-30]. Dostupné z: https://akveo.github.io/nebular

[33] MongoDB 5.0 CPU Intel G4650 compatibility - MongoDB Developer Community Forums [online]. MongoDB, c2021 [cit. 2023-05-01]. Dostupné z: https://www.mongodb.com/community/forums/t/mongodb-5-0-cpu-intel-g4650-compatibility/116610

[34] ZANDSTRA, Matt. PHP 8 objects, patterns, and practice: mastering OO enhancements, design patterns, and essential development tools. Sixth edition. New York, NY, U.S.A.: Apress, [2021]. ISBN 978-1484267905.

[35] Repository Design Pattern | by Per-Erik Bergman | Medium [online]. Per-Erik Bergman, c2017 [cit. 2023-05-01]. Dostupné z: https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30

[36] PO Files (GNU gettext utilities) [online]. Free Software Foundation, c1996-2023 [cit. 2023-05-01]. Dostupné z: https://www.gnu.org/software/gettext/manual/html_node/PO-Files.html

[37] .XML - Android (Strings) - Phrase [online]. Phrase, c2023 [cit. 2023-05-01]. Dostupné z: https://support.phrase.com/hc/en-us/articles/6111362189212--XML-Android-Strings-

[38] Localization | Apple Developer Documentation [online]. Apple, c2023 [cit. 2023-05-01]. Dostupné z: https://developer.apple.com/documentation/xcode/localization

[39] Unicode CLDR - Plural Rules [online]. Unicode, c1991-2023 [cit. 2023-05-01]. Dostupné z: https://cldr.unicode.org/index/cldr-spec/plural-rules

[40] ICU Documentation [online]. Unicode, c2016-2023 [cit. 2023-05-01]. Dostupné z: https://unicode-org.github.io/icu/

[41] HINZ, Yurek K. Exploring Open Source Software Localization Methods. Lambert Academic Publishing, 2011. ISBN 978-3844399738.

[42] Laravel - The PHP Framework For Web Artisans [online]. Laravel LLC., c2011-2023 [cit. 2023-05-01]. Dostupné z: https://laravel.com

[43] Symfony, High Performance PHP Framework for Web Development [online]. Symfony Team, c2023 [cit. 2023-05-01]. Dostupné z: https://symfony.com

[44] Nette - Comfortable and Safe Web Development in PHP [online]. David Grudl, c2008-2023 [cit. 2023-05-01]. Dostupné z: https://nette.org

[45] Slim Framework [online]. Slim Framework Team, c2023 [cit. 2023-05-01]. Dostupné z: https://www.slimframework.com

[46] Symfony Vs Phalcon: Which framework to use for building REST APIS? | Medium [online]. Naukri Engineering, c2017 [cit. 2023-05-01]. Dostupné z: https://medium.com/naukri-engineering/symfony-vs-phalcon-which-framework-to-use-for-building-rest-apis-942120ab7c99

[47] Docusaurus [online]. Meta Platforms, c2023 [cit. 2023-05-08]. Dostupné z: https://docusaurus.io

[48] Composer [online]. Nils Adermann, Jordi Boggiano, c2023 [cit. 2023-05-08]. Dostupné z: https://getcomposer.org

[49] Xdebug - Debugger and Profiler Tool for PHP [online]. Derick Rethans, c2002-2023 [cit. 2023-05-08]. Dostupné z: https://xdebug.org

[50] Find Bugs Without Writing Tests | PHPStan [online]. Ondřej Mirtes, c2016-2023 [cit. 2023-05-08]. Dostupné z: https://phpstan.org

[51] Advanced Load Balancer, Web Server, & Reverse Proxy - NGINX [online]. F5, c2023 [cit. 2023-05-08]. Dostupné z: https://www.nginx.com

[52] Let's Encrypt [online]. Internet Security Research Group, c2023 [cit. 2023-05-08]. Dostupné z: https://letsencrypt.org

[53] Node.js [online]. OpenJS Foundation, c2023 [cit. 2023-05-08]. Dostupné z: https://nodejs.org/en

[54] Angular [online]. Google, c2010-2023 [cit. 2023-05-08]. Dostupné z: https://angular.io

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

LTS	Long Term Support
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
REST	Representational State Transfer
CDN	Content Delivery Network
DI	Dependency Injection
FQCN	Fully-Qualified Class Name
HTTP	Hypertext Transfer Protocol
MVC	Model-View-Controller
CLI	Command Line Interface
PHP	Hypertext Preprocessor
JS	JavaScript
SQL	Structured Query Language
JSON	JavaScript Object Notation
URI	Uniform Resource Identifier
YAML	YAML Aint't Markup Language
INI	Initialization
IP	Internet Protocol
URL	Uniform Resource Locator
XSS	Cross Site Scripting
ICU	International Components for Unicode
ORM	Object Relational Mapper
SSL	Secure Sockets Layer
UI	User Interface

UX	User Experience
SaaS	Software as a Service
DTO	Data Transfer Object
PO	Portable Object
XML	Extensible Markup Language
AOT	Ahead-of-time

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: PŘILOŽENÉ CD

CD přiložené k diplomové práci obsahuje:

- diplomovou práci ve formátu PDF
- zdrojové kódy vytvářeného systému
- dokumentaci k systému
- exportovanou Postman kolekci s API endpointy ve formátu JSON