České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Informačný systém pre správu činností prekladateľskej agentúry

Bakalárska práca

Oľga Ostashchuk

Program: Softwarové inženýrství a technologie
Vedúci práce: Ing. Jiří Šebek

Praha, Máj 2022

Thesis Supervisor:
Ing. Jiří Šebek
Department of Telecommunication Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ostashchuk** Jméno: **Oľga** Osobní číslo: **492226**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Informační systém pro správu činnosti překladatelské agentury

Název bakalářské práce anglicky:

Information system for translation agency business functions

Pokyny pro vypracování:

1) Proveďte analýzu podnikových procesů v překladatelské agentuře a na jejím základěupřesněte požadavky, které by měl informační systém splňovat.
2) Na základě provedené analýzy navrhněte doporučené řešení.
3) Navržené řešení implementujte.
a) Implementujte informační systém pro správu činností překladatelskéagentury.
b) Při implementaci použijte technologie: Java, Spring Boot, CI/CD, Redis cache,Docker, JavaScript a případně další vhodné technologie.
c) Implementujte aplikaci tak, aby disponovala následujícími funkcionalitami:
i) evidence a správa objednávekii) evidence a správa zdrojových dokumentů a překladůiii) evidence a správa zákazníkůiv) evidence a správa překladatelův) autentizace uživatelůvi) autorizace uživatelů na základě uživatelských oprávnění
4) Proveďte testování informačního systému.
5) Proveďte nasazení informačního systému na cílový server.6) Proveďte vyhodnocení informačního systému a jeho přínosu pro podnik.

Seznam doporučené literatury:

[1] GAMMA, Erich. HELM, Richard. JOHNSON, Ralph. VLISSIDES, John. Design patterns: elements of reusable object-oriented software. Boston: Addison-Wesley, 1995. ISBN 978-0201633610
[2] BASS, Len. CLEMENTS, Paul. KAZMAN, Rick. Software architecture in practice. 2nd ed. Boston: Addison-Wesley, 2003. ISBN 0321154959
[3] WALLS, Craig. Spring Boot in action. Shelter Island, NY: Manning, 2016. ISBN 978-1-61729-398-6
[4] LARMAN, Craig. Applying UML and patterns: introduction to object-oriented analysis and design and iterative development. 3rd ed. New Jersey: Prentice-Hall, 2005. ISBN 978-0131489066
[5] HANCHETT, Erik. Vue.js in action. Shelter Island: Manning, 2018. ISBN 978-1-61729-524-9
[6] HEROUT, Pavel. Testování pro programátory. České Budějovice: Kopp, 2016. ISBN 978-80-7232-481-1

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studentky

Abstract

The main aim of this bachelor's thesis is to design and implement an information system for managing the activities of the INTERLANG translation agency, which will enable the evidence of order records and all its related processes. When implementing the system, the emphasis should be put on the security of the application in terms of protection of the important information about the company as well as private data of its customers. The resulting system should greatly facilitate administrative work for the translation agency's staff. This bachelor's thesis describes the whole development process from initial studies to its implementation. In the first part of the work, an analysis of business processes was performed, then an analysis of existing possible solutions, which was followed by the definition of functional and non-functional requirements for the new system, was performed. Furthermore, an analysis of available technologies, design of a new information system and its implementation and testing were performed. Thesis also describes the process of deploying the implemented solution on the target server. The final part provides a summary and evaluation of the overall work.

Keywords: Information system, Java, Spring, Vue.js, Vuetify, Vuex, Docker, NGINX, JWT

Abstrakt

Hlavným cieľom tejto bakalárskej práce je navrhnúť a implementovať informačný systém pre správu činností prekladateľskej agentúry INTERLANG, ktorý umožní realizovať evidenciu objednávok a všetkých s tým spojených procesov. Pri implementácii systému by mal byť kladený dôraz v prvom rade na bezpečnosť aplikácie z pohľadu ochrany dôležitých informácií spoločnosti a rovnako i súkromných údajov jej zákazníkov. Výsledný systém by mal zásadne uľahčiť administratívnu prácu zamestnancom prekladateľskej agentúry. Táto bakalárska práca popisuje celý vývojový proces od úvodnej štúdii až po jeho implementáciu. V prvej časti práce bola realizovaná analýza podnikových procesov, následne bola vykonaná analýza existujúcich možných riešení, na ktoré nadväzuje definovanie funkčných a nefunkčných požiadaviek na nový systém. Ďalej v práci bola vykonaná analýza dostupných technológií, návrh nového informačného systému a následne jeho implementácia a testovanie. V rámci práce je taktiež popísaný aj proces nasadenia implementovaného riešenia na cieľový server. V záverečnej časti je uvedené zhrnutie a zhodnotenie celkovej práce.

Klíčová slova: Informačný systém, Java, Spring, Vue.js, Vuetify, Vuex, Docker, NGINX, JWT

Poďakovanie

Rada by som na tomto mieste poďakovala svojmu skvelému vedúcemu pánovi Ing. Jiřímu Šebkovi za pomoc a priateľskú podporu pri vzniku tejto práce.

Čestné prehlásenie

Čestne vyhlasujem, že celú bakalársku prácu, vrátane všetkých jej príloh a obrázkov, som vypracovala samostatne, a to s použitím literatúry uvedenej v priloženom zozname.

Zoznam obrázkov

Obsah

Kapitola 1

Úvod

V súčasnej dobe sa informačné systémy stávajú pre mnohé organizácie esenciálnou potrebou. Sú to dôležité nástroje pre správu, porovnávanie a analýzu veľkého množstva dát, s cieľom získať z nich hodnoverné informácie, ktoré pomáhajú organizáciám pri prijímaní rozličných obchodných rozhodnutí, koordinácii, kontrole a vizualizácii v organizácii. [1]

Rovnakým spôsobom sa rozhodla zautomatizovať svoje podnikové procesy a uľahčiť prácu s veľkými objemami dát aj slovenská prekladateľská agentúra INTERLANG s vyše 10 ročnými skúsenosťami, ktorá doposiaľ žiaden informačný systém nevyužívala. [https://www.interlang.sk/]

Hlavným cieľom tejto bakalárskej práce je tým pádom vytvoriť nový informačný systém pre prekladateľskú agentúru tak, aby sa sprehľadnili a zrýchlili vnútorné procesy a zefektívnil čas zamestnancov strávený pri administratívnych činnostiach, čo by v konečnom dôsledku prispelo k zníženiu nákladov firmy. Pre dosiahnutie tohto cieľa bol projekt rozdelený na niekoľko etáp, ktoré sú ďalej opísané v jednotlivých kapitolách tejto práce.

V kapitole 2 sa budem venovať analýze aktuálnych podnikových procesov v agentúre. Zároveň v tejto časti bude porovnanie existujúcich informačných systémov na trhu, na ktoré nadväzuje definovanie požiadaviek na funkcionalitu, akou by mal poskytnutý informačný systém pre agentúru disponovať. Ďalej v práci nasleduje analýza vnútorného a vonkajšieho okolia agentúry a analýza dostupných technológií pre systém.

V kapitole 3 bol vykonaný návrh nového informačného systému. Súčasťou tejto kapitoly bola navrhnutá architektúra, dátový model, prototyp, typ autentizácie a ďalšie špecifiká systému. Kapitola 4 podrobne zachytáva proces implementácie frontend a bac-

kend aplikácie. Zároveň sa v tejto kapitole opisuje proces nasadenia aplikácií na cieľový server.

Kapitola 5 sa venuje testovacej fáze, opisuje druhy vykonaných testov a výsledky testovania. Kapitola 6, ktorá bude poslednou, sa venuje celkovému zhodnoteniu vykonanej práce a zhodnoteniu vytvoreného systému a jeho prínosu pre prekladateľskú agentúru.

Kapitola 2

Analýza

2.1 Aktuálne podnikové procesy

Hlavná činnosť prekladateľskej agentúry je zameraná na vyhotovenie prekladov ku zdrojovým dokumentom. V rámci jednej objednávky od zákazníka môže byť viac zdrojových dokumentov, ktoré sa môžu líšiť v jazyku, v ktorom sú napísané, a taktiež aj jazykom, do ktorého majú byť preložené.

2.1.1 Prekladateľ

Prekladateľská agentúra spolupracuje s prekladateľmi, ktorí poskytujú preklad zdrojového dokumentu do cieľového jazyka. U každého prekladateľa sú určené dvojice jazykov, na ktoré sa orientujú. Niektorí prekladatelia vyhotovujú v určitej dvojici jazykov iba neúradné preklady a niektorí vyhotovujú úradné a neúradné preklady. Zároveň je u každého prekladateľa určený jeho materinský jazyk, ktorý zohráva rolu pri výbere prekladateľa k prekladu konkrétneho zdrojového dokumentu. Okrem toho sa o prekladateľoch uchovávajú ich kontaktné údaje a IBAN. Kontaktné údaje zahŕňajú: meno, priezvisko, e-mail a telefónne číslo.

2.1.2 Zamestnanec

U zamestnancov firmy sa zaznamenávajú ich kontaktné údaje a IBAN.

2.1.3 Zákazník

Pri zákazníkovi je taktiež dôležité mať uvedené jeho kontaktné údaje.

Zákazníkom môže byť fyzická aj právnická osoba. V prípade právnickej osoby je potrebné uviesť kontaktné údaje fyzickej osoby, ktorú bude prekladateľská agentúra kontak-

tovať, keď sa bude chcieť obrátiť na právnickú osobu.

Zároveň je potrebné o zákazníkovi uchovávať jeho fakturačné údaje, ak sa jedná o právnickú osobu alebo o samostatne zárobkovo činnú osobu (SZČO). Fakturačné údaje zahŕňajú: adresu, meno, IČO (Identifikačné číslo organizácie), IČ DPH (Identifikačné číslo pre daň z pridanej hodnoty), DIČ (Daňové identifikačné číslo).

2.1.4 Objednávka

Pri každej objednávke sa uchovávajú informácie o zákazníkovi, ktorý objednávku vytvoril. Následne údaje o prekladateľoch, ktorí na danej objednávke pracovali, dátum vytvorenia, predpokladaný dátum dodania, dátum dodania, stav objednávky, celková cena, celková zľava v eurách, uhradená suma, zdrojové dokumenty, preklady, faktúry a informácie o úhrade.

Faktúry môžu byť dvoch druhov, zálohová alebo vyúčtovacia faktúra. Údaje, ktoré sa u faktúrach evidujú, sú: číslo faktúry, dátum vystavenia, dátum splatnosti, dátum dodania, čiastka k úhrade. V niektorých prípadoch sa vo faktúrach namiesto dátumu dodania uvádza dátum zdaniteľného plnenia.

Zároveň sú pri každej objednávke dôležité informácie o úhradách, a to dátum uhradenia, spôsob a čiastka.

2.1.5 Preklad

Pre lepší prehľad a organizáciu je dôležité si uchovávať aj isté informácie zvlášť pre každý preklad v rámci objednávky. K týmto informáciám patrí napríklad predpokladaný dátum dodania a dátum dodania, ktoré umožňujú zamestnancom agentúry organizovať ďalšie činnosti pre prekladateľov. Okrem toho sa o preklade uchovávajú údaje ako typ prekladu, počet normostrán, stav spracovania prekladu a zľava v percentách.

2.2 Súčasný stav

Momentálne nevyužíva prekladateľská agentúra žiaden informačný systém. Zamestnanci firmy samostatne zaznamenávajú informácie o každej prijatej objednávke v papierovej a digitálnej forme. Zdrojové dokumenty a preklady k nim sa uchovávajú v digitálnej forme na počítači firmy.

Týmto spôsobom musia zamestnanci viac času tráviť na zaznamenávanie a spracovanie každej objednávky. Zároveň sa viac času spotrebuje pri vyhľadávaní informácii z minulých mesiacov alebo rokov, či to už podľa mena zákazníka, prekladateľa, jazyka prekladu, alebo iných kritérií. Z hľadiska softvérov firma využíva balík Microsoft Office.

Všetok čas strávený na administratívnych záležitostiach by firme pomohol skrátiť nový systém, ktorý by zautomatizoval spomínané procesy. Tým pádom by sa zefektívnil čas zamestnancov venovaný na spracovanie objednávky a prispelo by to k zníženiu nákladov firmy.

2.3 Vyhľadávanie existujúcich možných riešení

Na trhu existuje viacero variant, ktoré by mohli do určitej miery spĺňať požiadavky prekladateľskej agentúry. Jednou z variant je softvér eWorkOrders CMMS. Tento softvér je vyvinutý pre platformy Windows, Linux, macOS, Android a iOS. Softvér využíva cloudové úložisko a je prístupný cez internet, pre jeho používanie nie je potrebná žiadna dodatočná inštalácia. Softvér poskytuje funkcie ako: správa dokumentov, správa zamestnancov, správa dodávateľov, správa zákazníkov, správa objednávok, fakturácia, sledovanie nákladov a iné. Cena tohto softvéru je USD 35.00 / mesiac a ďalej by sa vyvíjala od množstva zakúpených modulov. [2]

Ďalšou z variant je softvér Maxpanda CMMS. Tento softvér je vyvinutý pre platformy Windows, Linux, macOS, Android a iOS. Softvér využíva cloudové úložisko a je prístupný cez internet, pre jeho používanie rovnako nie je potrebná žiadna dodatočná inštalácia. Tento softvér umožňuje správu dokumentov, správu zamestnancov, správu dodávateľov, správu zákazníkov, správu objednávok, fakturáciu, sledovanie nákladov, správu zmlúv / licencií, cenové ponuky / odhady a iné. Tento softvér má 16 vopred definovaných užívateľských rolí zahŕňajúc administrátora, zamestnanca, dodávateľa a tiež rozlíšenie týchto a ďalších rolí podľa pridelenej funkcionality. Cena tohto softvéru je USD 99.00 / mesiac. [3]

Softvér UpKeep je vyvinutý pre platformy Windows, Linux, Android a iOS. Softvér UpKeep využíva cloudové úložisko a je prístupný cez internet. Zároveň je možné si softvér nainštalovať na lokálne počítače. Z hľadiska funkcionality poskytuje funkcie ako: správa dokumentov, správa zamestnancov, správa dodávateľov, správa zákazníkov, správa objednávok, fakturácia, sledovanie nákladov, správa zmlúv / licencií, cenové ponuky / odhady a iné. Softvér UpKeep má celkom 7 užívateľských rolí. Užívateľské role, za ktoré

sa pripláca, sú: administrátor, administrátor s obmedzenými funkcionalitami, technik, technik s obmedzenými funkcionalitami. Užívateľské role bez príplatku sú: rola iba s funkciou prezerania, rola iba s funkciou vytvárania žiadosti a rola pre externých dodávateľov alebo zákazníkov. Užívateľ s rolou dodávateľa alebo zákazníka sa nebude môcť prihlásiť do systému, táto rola slúži iba pre sprehľadnenie aktivít vo firme. Cena tohto softvéru je USD 45.00 / mesiac na jedného užívateľa. [4]

Vyhodnotenie

Vcelku dané softvéry nedisponujú funkcionalitami, špecifickými pre potreby prekladateľskej agentúry. U každej varianty sa nájdu funkcionality, ktoré spĺňajú niektoré požiadavky, avšak ani jedno riešenie nedokáže celistvo vyhovieť prekladateľskej agentúre a navyše sú cenovo neoptimálne. Dané softvéry napríklad neumožňujú spravovať prekladateľov a určovať do akých jazykov prekladajú a či vyhotovujú úradne alebo neúradné preklady. Zároveň softvéry neumožňujú prekladateľom priraďovať vlastnosť rodeného hovoriaceho u konkrétnych jazykov alebo určovať vlastnosti prekladov ako jazyk a typ úradnosti. Práve z vyššie spomenutých príčin je potrebné pre prekladateľskú agentúru navrhnúť a naimplementovať vlastné riešenie. Tým pádom nový systém bude prispôsobený výhradne pre potreby danej firmy a kompletne spĺňať všetky požiadavky.

Čo sa týka podobných systémov u iných prekladateľských firmách, tak sa mi nepodarilo zistiť, aké konkrétne informačné systémy používajú a či vôbec nejaké majú. Keďže sa jedná o systém pre využívanie vo vnútornom prostredí firmy, tak túto informáciu zvyčajne nezverejňujú.

2.4 Požiadavky pre systém

V systéme pre prekladateľskú agentúru by mali byť 2 typy užívateľov:

- **Zamestnanec** - predstavuje bežných zamestnancov firmy. Títo užívatelia majú základnú funkcionalitu, ktorá je zhrnutá ďalej vo funkčných požiadavkách pre všetkých užívateľov.
- **Administrátor** - predstavuje zamestnancov, ktorí majú vedúce postavenie. Užívatelia s touto rolou majú rovnakú funkcionalitu ako bežný zamestnanec, avšak dodatočne môžu vykonávať aj iné činnosti, ktoré sú ďalej zhrnuté vo funkčných požiadavkách pre administrátora.

2.4.1 Funkčné požiadavky pre administrátora

FR01: Vytvorenie nového užívateľa

Systém umožní vedúcemu zamestnancovi vytvoriť účet pre nového užívateľa.

FR02: Úprava účtu užívateľa

Systém umožní vedúcemu zamestnancovi zmeniť údaje o ľubovoľnom užívateľskom účte.

FR03: Zrušenie účtu užívateľa

Systém umožní vedúcemu zamestnancovi zrušiť ľubovoľnému užívateľovi účet.

FR04: Zobrazenie údajov užívateľa

Systém umožní vedúcemu zamestnancovi zobraziť údaje o užívateľovi.

FR05: Zobrazenie zoznamu všetkých užívateľov

Systém umožní vedúcemu zamestnancovi zobraziť zoznam všetkých užívateľov.

FR06: Pridanie nového prekladateľa

Systém umožní vedúcemu zamestnancovi vytvoriť a pridať do systému nového prekladateľa.

FR07: Úprava údajov o prekladateľovi

Systém umožní vedúcemu zamestnancovi zmeniť údaje o prekladateľovi.

FR08: Odstránenie prekladateľa

Systém umožní vedúcemu zamestnancovi odstrániť prekladateľa zo systému.

2.4.2 Funkčné požiadavky pre všetkých užívateľov

FR09: Prihlásenie / odhlásenie užívateľa

Systém umožní užívateľovi prihlásiť / odhlásiť sa z vlastného účtu.

FR10: Zobrazenie vlastného účtu

Systém umožní užívateľovi zobraziť údaje o vlastnom účte.

FR11: Úprava vlastného účtu

Systém umožní užívateľovi zmeniť údaje o vlastnom účte.

FR12: Zadanie novej objednávky

Systém umožní užívateľovi zadať novú objednávku, určiť objednávke zákazníka, preklady, úhrady a faktúry.

FR13: Úprava objednávky

Systém umožní užívateľovi zmeniť údaje o objednávke.

FR14: Vymazanie objednávky

Systém umožní užívateľovi zmazať objednávku.

FR15: Vytvorenie prekladu, úhrady a faktúry

Systém umožní užívateľovi vytvoriť nový preklad, úhradu a faktúru.

FR16: Úprava prekladu, úhrady a faktúry

Systém umožní užívateľovi zmeniť údaje o preklade, úhrade a faktúre.

FR17: Vymazanie prekladu, úhrady a faktúry

Systém umožní užívateľovi zmazať preklad, úhradu alebo faktúru.

FR18: Pridelenie prekladov, úhrad a faktúr k objednávke

Systém umožní užívateľovi prideliť k objednávke preklady, úhrady a faktúry.

FR19: Zobrazenie údajov objednávky

Systém umožní užívateľovi zobraziť údaje o objednávke.

FR20: Zobrazenie zoznamu všetkých objednávok

Systém umožní užívateľovi zobraziť zoznam všetkých objednávok.

FR21: Zobrazenie zoznamu objednávok podľa vlastností

Systém umožní užívateľovi zobraziť zoznam objednávok podľa konkrétneho zákazníka, dátumu vytvorenia, dátumu dodania, celkovej ceny, stavu úhrady.

FR22: Zobrazenie údajov prekladateľa

Systém umožní užívateľovi zobraziť údaje o prekladateľovi.

FR23: Zobrazenie zoznamu všetkých prekladateľov

Systém umožní užívateľovi zobraziť zoznam všetkých prekladateľov.

FR24: Zobrazenie údajov zákazníka

Systém umožní užívateľovi zobraziť údaje o zákazníkovi.

FR25: Zobrazenie zoznamu všetkých zákazníkov

Systém umožní užívateľovi zobraziť zoznam všetkých zákazníkov.

FR26: Pridanie nového zákazníka

Systém umožní užívateľovi vytvoriť a pridať do systému nového zákazníka.

FR27: Úprava údajov o zákazníkovi

Systém umožní užívateľovi zmeniť údaje o zákazníkovi.

FR28: Odstránenie zákazníka

Systém umožní užívateľovi odstrániť zákazníka zo systému.

FR29: Odosielanie notifikačných e-mailov

Systém bude umožňovať odosielať užívateľom notifikačný e-mail o vytvorení účtu.

2.4.3 Nefunkčné požiadavky

NFR01: Kompatibilita s najpoužívanejšími webovými prehliadačmi

Webová aplikácia bude fungovať a správne sa zobrazovať na aktuálnych verziách webových prehliadačov Firefox a Google Chrome.

NFR02: Obrana proti bezpečnostným útokom

Webová aplikácia bude zabezpečená proti rôznym útokom, ktoré by narušili jej správne fungovanie (XSS (Cross-site scripting), CSRF (Cross-site scripting), SQL injection).

NFR03: Bezpečnosť

Webová aplikácia bude zaručovať bezpečnosť citlivých údajov, ku ktorým budú mať prístup iba oprávnení užívatelia.

NFR04: Rozšíriteľnosť

Webová aplikácia bude mať čitateľný kód a bude navrhnutá a naimplementovaná tak, aby sa v budúcnosti mohla prípadne rozširovať o nové funkcionality.

2.5 SWOT, PEST a 5F analýza

Ako bolo zhodnotené v predošlej časti, k naplneniu funkčných požiadaviek systému je potrebné poskytnúť nové riešenie. Avšak pred jeho samotným návrhom a implementáciou bude užitočné predovšetkým vykonať analýzu vnútorného a vonkajšieho okolia.

K tomu použijem nasledujúce metódy: SWOT analýzu, PEST analýzu a 5F analýzu.

2.5.1 Swot analýza

SWOT analýza sa riadi medzi základné metódy strategickej analýzy a zameriava sa na silné a slabé stránky z vnútorného prostredia a na príležitosti a hrozby z vonkajšieho prostredia. SWOT analýza je navrhnutá tak, aby umožnila realistický a na faktoch založený pohľad a musí sa udržiavať presnou tým, že sa vyhýba vopred vytvoreným presvedčeniam alebo šedým oblastiam. [5]

SILNÉ STRÁNKY (S)	SLABÉ STRÁNKY (W)
• Zvýšenie efektivity • Dostatok zdrojov na vývoj • Nízke výrobné náklady na systém	• Komplexný systém náročný na vývoj • Možná slabá motivácia personálu na používanie systému
PRÍLEŽITOSTI (O)	**HROZBY (T)**
• Možnosť predaja systému po dokončení interného vývoja	• Demotivácia zamestnancov a ich prechod ku konkurencii

Tabuľka 2.1: SWOT analýza

2.5.2 Analýza 5F

Analýza 5F alebo Porterov model piatich síl patrí k základným a zároveň najvýznamnejším nástrojom pre analýzu konkurenčného prostredia firmy a jej strategického riadenia. Hlavnou úlohou Porterovho modelu je nájsť pozíciu pre spoločnosť, v ktorej sa môže najlepšie brániť proti konkurenčným silám. Porterova analýza je vhodná pre vyhodnotenie strategických príležitostí a hrozieb konkurencie. Analýza skúma oblasti, ktoré určujú chovanie konkurentov, a to: riziko vstupu potenciálnych konkurentov, rivalitu stávajúcich konkurentov, vyjednávaciu silu dodávateľov a odberateľov a hrozbu substitučných produktov. [6]

Túto analýzu som spracovala pre tento projekt nasledujúcim spôsobom:

- **Konkurencia** - Existuje veľké množstvo konkurenčných firiem, ktoré môžu poskytnúť obdobnú funkcionalitu, avšak ani jedna z nich neposkytne kompletne požadovanú funkcionalitu.
- **Sila dodávateľov** - Vytvorený systém nebude závislý na dodávateľoch.
- **Sila odberateľov** - Pri prípadnom prechode k inej spoločnosti sa môže vyskytnúť problém s prevodom dát, ktorý by bol časovo aj finančne náročný.
- **Substitúti** - Momentálne nie je žiadny systém, ktorý by mohol nahradiť funkcionalitu požadovaného informačného systému.

- **Novo prichádzajúci** - Existuje možnosť ohrozenia v prípade príchodu novej firmy implementujúcej podobné softvérové riešenia a môže to priviesť k strate záujmu prekladateľskej agentúry o daný systém.

2.5.3 Analýza PEST(E)

Analýza PEST je merací nástroj, ktorý sa používa na hodnotenie trhu pre konkrétny produkt alebo podnik v danom časovom rámci. PEST predstavuje politické, ekonomické, spoločenské a technologické faktory. Zaraďujú sa sem aj ekologické faktory. Analýza PEST sa zameriava na vonkajšie okolie, teda na makrookolie. Anlýza týchto faktorov pomáha organizáciám prijímať lepšie strategické obchodné rozhodnutia. [7]

Túto analýzu som spracovala pre tento projekt nasledujúcim spôsobom:

- **Politické** - Situácia je stabilná z hľadiska právnických či politických faktorov.
- **Ekonomické** - Ekonomická situácia Slovenskej republiky tiež nebráni realizácii projektu.Situácia je stabilná.
- **Spoločenské** - Spoločenská situácia Slovenskej republiky nebráni realizácii obdobného projektu. Situácia je stabilná.
- **Technologické** - Technologické štandardy Slovenskej republiky sú na dostatočnej úrovni, aby bolo možné bez problémov realizovať projekt z technologického hľadiska.
- **Ekologické** - Realizácia riešenia nijako nesúvisí s prírodnými zdrojmi.

2.6 Analýza backendových technológií

Informačný systém bude implementovaný vo forme webovej aplikácie, ktorá bude pozostávať z backendovej a frontendovej aplikácie. V tejto časti budem ďalej porovnávať a analyzovať technológie, ktoré sa využijú pri implementácii backendovej aplikácie.

Existuje niekoľko optimálnych možností výberu frameworku pre implementovanie backendovej časti. Patria k nim napríklad framework Spring, ASP.NET, Flask, Django alebo technológie Node.js.

2.6.1 Programovacie jazyky a frameworky

Flask

Flask je open-source mikroframework v programovacom jazyku Python, ktorý sa využíva pri budovaní webových aplikácií. Flask webové aplikácie sú multiplatformové a môžu bežať na rôznych operačných systémoch. Flask je microframework, čo znamená, že poskytuje iba základné funkcionality webovej aplikácie, ale je ľahko rozšíriteľný o ďalšie funkcionality pythonovských knižníc. [8] [9]

S Flaskom je jednoduché začať vytvárať nový projekt a tiež je jeho veľkou výhodou flexibilnosť. Umožňuje robiť technologické zmeny počas budovania, a tým pádom sa neviazať na rozhodnutia prijaté na začiatku budovania projektu. Flask sa hlavne hodí pre projekty, v ktorých sa experimentuje s architektúrou, knižnicami, skúšajú sa nové technológie, tiež pre projekty s neznámym počtom vysoko špecializovaných microservices a pre projekty s veľkým množstvom rôznych, nesúrodých prvkov.

K nevýhodám budovania webovej aplikácie za použitím Flasku patrí väčšie riziko s bezpečnostnými problémami, vyššie náklady na údržbu zložitejších systémov a zložitejšia údržba pri väčších implementáciách. [10]

Django

Django je open-source framework napísaný tiež v programovacom jazyku Python a slúži pre vytváranie webových aplikácií. Django webové aplikácie sú multiplatformové a môžu bežať na rôznych operačných systémoch. [11]

Dodržuje konzistentné princípy a poskytuje veľké množstvo funkcionalít a pripravených riešení, ktoré sa môžu hneď využiť. V porovnaní s Flask je Django bohatší a obsiahlejší, a tým pádom umožňuje Django vytvárať kompletný software.

Tiež Django pomáha vývojárom vyhnúť sa mnohým bezpečnostným chybám ako napríklad XSS, CSRF, SQL injection a považuje sa teda za bezpečnejšie riešenie. [12]

Django používa architektúru založenú na komponentoch, kde každá časť architektúry je nezávislá od ostatných, a preto ju možno v prípade potreby nahradiť alebo zmeniť. Jasné oddelenie medzi rôznymi časťami znamená, že sa môže škálovať pre zvýšenú prevádzku pridaním hardvéru na akejkoľvek úrovni: cache servery, databázové servery alebo aplikačné servery.

Na druhej strane sa tento framework slávi svojím monolitizmom, čo znamená, že uprednostňuje konvenciu pred konfiguráciou a pre vývojárov je modifikácia šablón preto náročná. Zároveň k jeho nevýhodám patrí nízka flexibilnosť, nižšia kompatibilita s najnovšími technológiami a tiež je budovanie webovej aplikácie v Django zložitejšie v porovnaní s Flask. [10]

ASP.NET Core

ASP.NET Core je open-source framework vyvinutý spoločnosťou Microsoft. Je to všeobecný framework pre vývoj softvéru. Umožňuje vývojárom vytvárať rôzne druhy softvéru, vrátane webových aplikácií.

Narozdiel od iných frameworkov, ASP.NET Core nie je limitovaný pre jeden konkrétny programovací jazyk, ale podporuje a umožňuje využívanie viacerých programovacích jazykov ako: C#, VB.NET, F#, XAML a TypeScript. ASP.NET Core nie je viazaný na operačný systém Windows, ako napríklad predchádzajúci framework ASP.NET. Je možné vyvíjať a spúšťať ASP.NET Core aplikácie aj na operačných systémoch Linux a macOS. ASP.NET Core ponúka pokročilé a rozsiahle knižnice tried, API a jednou z najväčších výhod používania ASP.NET Core je vysoký výkon.

Na druhej strane sa môžu vyskytnúť problémy s dokumentáciou k tomuto frameworku. ASP.NET Core dokumentácia nie je výstižná, hlavne v porovnaní s inými vývojovými technológiami. Spoločnosť Microsoft síce aktívne dopĺňa všetky existujúce medzery, ale samotný framework sa často aktualizuje, a preto často uvedené riešenie v dokumentácii nemusí presne fungovať. [13] [14] [15]

Node.js

Node.js je open-source softvérový systém a je to prostredie, ktoré umožňuje spúšťať kód v programovacom jazyku JavaScript. Slúži pre tvorbu a spúšťanie serverovej časti webovej aplikácie a iných aplikácií. Node.js je multiplatformový a môže bežať na rôznych operačných systémoch. [16]

Výhodou využitia Node.js je, že sa môže použiť programovací jazyk JavaScript nielen pre tvorbu frontendu, ale aj pre tvorbu backendu. Silnou stránkou Node.js je aj jeho vysoký výkon. Node.js je dobrá voľba pre aplikácie s microservices architektúrou.

Node.js nepodporuje viacvláknové programovanie, a preto sa nehodí pre komplexnejšie aplikácie. Taktiež využíva model asynchrónneho programovania a sťažuje sa tým celkové chápanie a udržiavanie kódu. Nevýhodou sú aj časté zmeny v Node.js API, ktoré sú spätne nekompatibilné a kvôli ktorým je nutné meniť aj kód aplikácie, aby sa synchronizoval s najnovšou verziou Node.js API. [17]

Spring

Spring je open-source framework, ktorý poskytuje komplexnú podporu infraštruktúry pri vývoji aplikácií v programovacom jazyku Java.

Spring framework využíva Inversion of control, čo znamená, že obsahuje kontajner, ktorý je zodpovedný za vytváranie inštancií, konfiguráciu a zostavovanie objektov, ako aj za správu ich životných cyklov. Na zostavenie tých objektov Spring používa konfiguračné metadáta, ktoré môžu byť vo forme konfigurácie XML alebo anotácií.

Ďalšou vlastnosťou frameworku Spring je Dependency injection - vkladanie závislostí. Dependency injection je základným aspektom Spring frameworku, cez ktorý Spring kontajner nainjektuje objekty do iných objektov. Umožňuje teda voľné spojenie komponentov a presúva zodpovednosť za správu komponentov na kontajner. Tým pádom framework Spring dosahuje loose coupling - voľné spojenie prostredníctvom Dependency injection a programovania založeného na rozhraniach.

Zároveň umožňuje využitie POJO - Plain old java object - objektov, poskytuje konzistentné rozhranie na správu transakcií, podporuje konfiguráciu cez XML a cez anotácie a tiež Spring framework prevzal osvedčené postupy, ktoré boli rokmi overované v rôznych aplikáciách a formalizované ako návrhové vzory.

Na druhej strane je práca s frameworkom Spring zložitejšia. Vyžaduje si to veľa odborných znalostí, preto je potrebné sa ho najprv naučiť. Zároveň Spring poskytuje vývojárom viacero možností. Tieto možnosti vytvárajú zmätok, že ktorú funkciu použiť a ktorú nie a nesprávne rozhodnutia môžu viesť k značným oneskoreniam v dokončení projektu. [18] [19]

Zhrnutie

Po analyzovaní vyššie spomenutých frameworkov a ďalších technológií a zvážení ich pozitív a negatív som prišla k záveru, že z hľadiska technológií bude backendová časť systému naimplementovaná v programovacom jazyku Java s použitím frameworku Spring

a nástrojov Spring Boot. Rozhodla som sa vykonať implementáciu s použitím týchto technológií z viacerých dôvodov. Frameworku Spring a nástrojom Spring Boot bol v rámci výuky venovaný veľký priestor a dôkladne sa preberali jednotlivé detaily. Získala som prax z rozličných úspešne ukončených projektov a doposiaľ to predstavuje pre mňa najoptimálnejšiu variantu pre implementáciu webovej aplikácie.

2.6.2 Databázové systémy

Súčasťou využitých technológií pre backendovú časť bude aj databázový systém. Existuje viacero druhov databázových systémov. V tejto práci sa sústredím na výber medzi relačnými a NoSQL databázami, ktoré sú k danej práci najviac relevantné.

NoSQL

NoSQL databázy sú databázy, ktoré na ukladanie a spracovanie dát používajú iné prostriedky ako tradičné relačné databázy, ktoré využívajú tabuľkové schémy.

K silným stránkam NoSQL databáz patrí škálovateľnosť. Na rozdiel od relačných databáz, ktoré sú dobre škálovateľné iba vertikálne, NoSQL databázy sú škálovateľné aj horizontálne. Škálovateľnosť je jeden z hlavných dôvodov, prečo sa tento typ databázy používa na ukladanie veľkých dát. Vďaka možnosti rozšírenia na stovky rôznych strojov v závislosti od rozsahu databázy, podporuje masívne paralelné spracovanie. To znamená, že môže využívať mnoho procesorov na prácu na rovnakej sade výpočtov súčasne.

Databázy NoSQL sú vysoko flexibilné, pretože môžu ukladať a kombinovať akýkoľvek typ dát, štruktúrovaných aj neštruktúrovaných, na rozdiel od relačných databáz, ktoré môžu ukladať dáta iba štruktúrovaným spôsobom.

Databázy NoSQL umožňujú dynamicky aktualizovať schému tak, aby sa vyvíjala s meniacimi sa požiadavkami, a zároveň zabezpečiť, že to nespôsobí žiadne prerušenie alebo výpadok bežiacej aplikácie.

Na druhej strane, u NoSQL databáz neexistuje žiaden štandard, ktorý by definoval presné pravidlá a úlohy NoSQL databáz. Jazyky dizajnu a dotazovania databáz NoSQL sa medzi rôznymi produktmi NoSQL značne líšia – oveľa viac ako medzi tradičnými relačnými databázami.

NoSQL databáza kladie na prvé miesto škálovateľnosť a výkon, ale pokiaľ ide o kon-

zistenciu údajov, tak na to NoSQL databáza neberie veľký ohľad. Preto sa v nej môžu vyskytnúť aj duplicity, zatiaľ čo relačná databáza zaisťuje, že sa do nej nedostanú žiadne duplicitné riadky. [20]

Taktiež je na NoSQL databázy jednoduchšie vykonať bezpečnostný útok v porovnaní s relačnými databázami.[21]

Existujú štyri hlavné typy NoSQL databáz: databáza párov kľúč - hodnota, stĺpcová databáza, dokumentová databáza a grafová databáza. Každý typ rieši problém, ktorý nie je možné vyriešiť pomocou relačných databáz. [22]

- **Databáza párov kľúč-hodnota** - Tento špecifický typ databázy NoSQL používa metódu kľúč – hodnota a predstavuje kolekciu párov kľúč – hodnota, ktoré sú uložené ako jednotlivé záznamy a nemajú vopred definovanú dátovú štruktúru. Kľúče sú jedinečné identifikátory hodnôt. Hodnoty môžu byť ľubovoľný typ objektu a môže byť aj iný pár kľúč-hodnota, avšak, v takom prípade sa štruktúra databázy stáva zložitejšou. Typickými zástupcami sú: Redis a Riak. Silnou stránkou tohto typu NoSQL databázy sú Jednoduché príkazy a absencia dátových typov. Vďaka tomu môžu dáta v prípade potreby nadobudnúť akýkoľvek typ alebo dokonca viacero typov. Vďaka svojej jednoduchosti dokáže databáza aj rýchlo reagovať. Tento typ databázy slúži najmä na zaplnenie medzier v relačných databázach.Databáza párov kľúč-hodnota nie je vhodná pre aplikácie vyžadujúce časté aktualizácie dát, zložité dotazy zahŕňajúce špecifické hodnoty dát alebo pre dáta s viacerými primárnymi kľúčmi. [23]

- **Stĺpcová databáza** - Stĺpcová databáza organizuje a ukladá dáta nie podľa riadkov, ale podľa stĺpcov. Ku každému kľúču je možné uložiť viac hodnôt, ktoré zodpovedajú tomu istému stĺpcu a tiež každý kľúč môže mať vyplnené hodnoty iných stĺpcov. Typickými zástupcami stĺpcových databáz sú: Apache Hadoop a Apache Cassandra. Vďaka svojmu modelu sú stĺpcové databázy veľmi flexibilné. Stĺpcová databáza dokáže efektívne skladovať dáta, lebo má dobrú kompresiu dát a tým pádom ušetrí pamäť na disku. Medzi výhody patrí tiež efektívnejší prístup k dátam pri dotazovaní iba na podmnožinu stĺpcov - odstránením potreby čítať stĺpce, ktoré nie sú relevantné. Stĺpcová databáza však nie je efektívna v inkrementálnom načítaní dát ani pri online transakčnom spracovaní. [24]

- **Dokumentová databáza** - Dokumentová databáza ukladá informácie do dokumentov vo formátoch CML, YAML, JSON alebo v binárnych dokumentoch, ako je

BSON. Na usporiadanie týchto dokumentov do jedného celku je ku každému dokumentu priradený špecifický kľúč. Vďaka tejto vlastnosti sú dokumentové databázy podobné databázam párov kľúč – hodnota. Dokumentová databáza nemá jednotnú schému a každý objekt je zvlášť uložený v jednom dokumente. Každý dokument môže mať odlišnú štruktúru. Vďaka tomu sa dokumentové databázy označujú za flexibilné. Zároveň sú dokumentové databázy zamerané skôr na dáta než na vzťahy. Vzťahy sú v nich reprezentované prostredníctvom vnorených dát, nie cudzích kľúčov. Teda akýkoľvek dokument môže obsahovať ľubovoľný počet ďalších vnorených dokumentov. [25]

- **Grafová databáza** - Do databázy sa ukladajú uzly a ich vlastnosti a tiež hrany medzi týmito uzlami. Hlavným prínosom je vyhľadávanie príslušných uzlov v rozsiahlych grafoch na základe implementovaných grafických algoritmov, ktoré sú neporovnateľne rýchlejšie, než v bežných relačných databázach. Grafová databáza sa zameriava na vzťahy medzi dátami rovnako ako na dáta samotné. Použitie grafového dátového modelu umožňuje vizualizovať dáta, a teda ich rýchlejšie pochopiť. Využíva sa často pre ukladanie veľkých dát. Nedostatkom týchto databáz je škálovateľnosť. Keďže sú navrhnuté pre jednovrstvovú architektúru, je ťažké ich škálovať na viac serverov. [26]

Relačná databáza

Relačná databáza je databáza založená na relačnom modeli. Relačný databázový model združuje dáta do tzv. relácií - tabuliek. Tabuľky sa skladajú zo stĺpcov - atribútov a riadkov - záznamov. Tabuľky tvoria základ relačnej databázy.

Relačný model prináša celý rad výhod, najmä často prirodzenú reprezentáciu spracovávaných dát, možnosť jednoduchého definovania a spracovania väzieb.

Aj keď je relačná databáza z hľadiska výkonu slabšia, jej rýchlosť je vďaka jej jednoduchosti podstatne vysoká. Zároveň relačnej databáze zvyšujú rýchlosť aj rôzne optimalizácie, ktoré sú jej súčasťou. Preto aplikácie pri použití relačnej databázy budú bežať s primeranou rýchlosťou.

Relačná databáza kladie veľký dôraz na zachovanie integrity dát. Integrita entity zaisťuje, že primárny kľúč v tabuľke je jedinečný a hodnota nie je nastavená na prázdnu. Referenčná integrita vyžaduje, aby sa každá hodnota v stĺpci cudzieho kľúča nachádzala v stĺpci primárneho kľúča tabuľky, z ktorej pochádza.

Viacerí užívatelia môžu pristupovať k relačnej databáze súčasne a získavať informácie dokonca aj počas toho, ako sa dáta aktualizujú.

Relačná databáza kladie dôraz aj na bezpečnosť. Dáta sú zabezpečené, pretože systém správy relačných databáz umožňuje priamy prístup k dátam iba oprávneným užívateľom. Žiaden neoprávnený užívateľ prístup k informáciám nemá.

Relačné databázy nie sú ideálne na spracovanie veľkého množstva neštruktúrovaných údajov. Údaje, ktoré sú prevažne kvalitatívne, nie sú ľahko definované alebo sú dynamické, nie sú pre relačné databázy optimálne, pretože, ako sa údaje menia alebo vyvíjajú, musí sa s nimi vyvíjať aj schéma, čo si vyžaduje čas.

Relačné databázy nie sú dobre horizontálne škálovateľné naprieč fyzickými úložnými štruktúrami s viacerými servermi. Je ťažké spracovávať relačné databázy na viacerých serveroch, pretože, ako sa zväčšuje a distribuuje množina dát, tak sa narúša ich štruktúra a používanie viacerých serverov vplýva aj na výkon, napríklad na zvyšujúce sa časy odozvy aplikácií a na menšiu dostupnosť. [27] [28] [29]

Zhrnutie

NoSQL databázy sú obvykle lepšou voľbou pre aplikácie, ktoré majú zložitejšie a neustále sa meniace dáta a ich štruktúru. Tieto dáta si vyžadujú flexibilný dátový model, ktorý nie je potrebné okamžite definovať.

Ďalším prípadom využitia NoSQL databáz sú napr. aplikácie, kde nie je potrebné udržiavať relácie medzi jednotlivými dátovými entitami. Typickým príkladom tohto sú aplikácie pracujúce s časovými radmi. Pre tieto účely existuje celá podskupina databázových riešení a spoločne sú často označované ako tzv. "time series"databázy. [30]

Taktiež sa NoSQL databázy využívajú v prípade, kde je potrebné pracovať v tzv. distribuovanom režime, a teda tieto databázy umožňujú nielen ukladať dáta, ale aj spúšťať výpočty nad nimi nie iba na jednom počítači, ale na celom clusteri počítačov. [31]

Vzhľadom k tomu, že informačný systém pre prekladateľskú agentúru nebude narábať s dátami, ktoré majú dynamicky sa meniacu štruktúru, ale naopak s dátami, ktoré majú presne definovanú a stabilnú štruktúru, bude pri implementácii využitá relačná databáza. Taktiež relačná databáza bola zvolená, kvôli tomu, že je vhodná na zachytávanie relácií

medzi dátovými entitami.

2.6.3 Výber relačnej databázy

Existujú rôzne implementácie relačných databáz, napr.: PostgreSQL, MariaDB, MySQL, Oracle a ďalšie.

PostgreSQL

PostgreSQL je open - source, objektovo - relačný databázový systém. Databáza je silná z hľadiska spoľahlivosti, správnosti a integrity údajov. Používa sa na bezpečné ukladanie údajov. Umožňuje komplexné nastavovanie užívateľských práv a veľkou výhodou je, že Microsoft Azure cloudové služby umožňujú využitie u nich PostgreSQL databázy ako platformnej služby. [32]

MySQL

MySQL je open - source systém na správu relačných databáz. Je veľmi podobný MariaDB, keďže MariaDB vznikla zo základov MySQL. Líšia sa v licenčných modeloch a MySQL neponúka takú širokú škálu dátových typov ako PostgreSQL. [32] [33]

MariaDB

MariaDB je open - source relačná databáza a spolupracuje s mnohými spoločnosťami ako Facebook, Alibaba a Google, ktoré sú užitočné pri vývoji nových funkcionalít a vylepšení pre celú komunitu MariaDB. Zároveň MariaDB nahrádza MySQL v mnohých aspektoch, disponuje vylepšenými funkcionalitami, má viac úložných mechanizmov a lepší výkon.

Veľkou výhodou MariaDB je, že sa zameriava na vysokú úroveň zabezpečenia, komunita MariaDB neustále hľadá a opravuje existujúce problémy. [33]

Oracle

Oracle je databázový systém, ktorý nie je open - source. Poskytuje kvalitné služby s veľmi pokročilými možnosťami spracovania dát, vysokým výkonom a jednoduchou škálovateľnosťou, avšak ceny licencií Oracle Database Standard Edition 2 sa pohybujú medzi 700.00−17,500.00. Z tohto dôvodu prekladateľská agentúra zamieta túto možnosť. [34]

Zhrnutie

Pre implementovanie informačného systému som sa rozhodla využiť PostgreSQL databázový systém z viacerých dôvodov. Jednak sme s ním pracovali v rámci výuky na semestrálnych prácach. Tiež je výhodou, že je open - source, ponúka vysoko kvalitné funkcionality a je možné túto databázu použiť u Microsoft Azure.

2.6.4 Cache systém

Taktiež sa v backendovej časti využijú technológie softvérovej cache pre zvýšenie výkonnosti informačného systému. K možným variantám patrí napríklad Redis cache alebo Hazelcast cache. Hazelcast a Redis sú si na prvý pohľad veľmi podobné. Môžu riešiť podobné prípady použitia, a preto môže byť ťažké sa rozhodnúť, ktorú variantu použiť. Spring Boot je dobre integrovaný s oboma cachami. Pri implementácii informačného systému som sa rozhodla využiť Redis cache z dôvodu, že som už s touto technológiou mala skúsenosti v rámci výuky v semestrálnom projekte. [35]

2.6.5 E-mail SMTP server

Zároveň sa v backendovej časti využije E-mailový SMTP server. Spring boot umožňuje ľahko definovať nastavenia SMTP serveru a následne posielať e-maily priamo z aplikácie.

2.7 Analýza frontendových technológií

Pri implementácii frontendovej časti sa bude využívať JavaScriptový framework. V tejto časti ďalej analyzujem a opíšem výber frameworku a ďalších technológií potrebných pre realizovanie frontendovej časti aplikácie.

V dnešnej dobe sa v mnohých frontendových aplikáciách prevažne používajú Angular, Vue.js frameworky alebo React knižnica. [36]

2.7.1 Angular

Angular je open-source framework dizajnu aplikácií a vývojová platforma, postavená na programovacom jazyku TypeScript, ktorá slúži na vytváranie efektívnych a sofistikovaných single-page aplikácií. Angular je platforma, ktorá sa dá škálovať od projektov jedného vývojára až po aplikácie na podnikovej úrovni. Angular je sponzorovaný a udržiavaný spoločnosťou Google a bol u nich použitý v niektorých ich veľkých a kom-

plexných webových aplikáciách.

Angular zaisťuje obojsmernú dátovú väzbu pre okamžitú synchronizáciu medzi modelom a zobrazením, takže akákoľvek zmena v zobrazení sa okamžite prejaví v modeli a naopak. Funkcie Angular tiež umožňujú vývojárom naprogramovať špeciálne správanie pre Document Object Model (DOM) a tým pádom vytvárať bohatý a dynamický obsah HTML.

Ako platforma Angular zahŕňa komponentovo založený framework na vytváranie škálovateľných webových aplikácií, zbierku dobre integrovaných knižníc, ktoré pokrývajú širokú škálu funkcií vrátane smerovania, správy formulárov, komunikácie klient-server a ďalších funkcií a zahŕňa sadu vývojárskych nástrojov, ktoré umožňujú vyvinúť, zostaviť, testovať a aktualizovať kód aplikácie. [37] [38]

2.7.2 React

React je open-source JavaScriptová knižnica na vytváranie užívateľských rozhraní. Spravuje ho Meta (predtým Facebook) a komunita jednotlivých vývojárov a spoločností.

Kľúčovou vlastnosťou Reactu je virtuálny Document Object Model (DOM) s jednosmernou dátovou väzbou. Vďaka funkcii virtuálneho DOM je React chválený za svoj dobrý výkon a je považovaný za jeden z jednoduchších frameworkov na učenie. Keďže React je knižnica a nie framework, tak na rozdiel od iných frameworkov nedisponuje niektorými dôležitými funkciami. Kvôli tomu je potrebné, aby React spolupracoval s inými knižnicami, ktoré tieto funkcionality, ako napríklad správa stavu, smerovanie a interakcia s API, nahradia.

React využíva syntax JSX, ktorá môže byť vstupnou prekážkou pre niektorých vývojárov, ktorí nechcú investovať čas do jej učenia. [38]

2.7.3 Vue.js

Vue.js je open-source, progresívny, prístupný, výkonný a všestranný JavaScriptový framework na vytváranie webových užívateľských rozhraní. Stavia na štandarde HTML, CSS a JavaScript a poskytuje deklaratívny a komponentovo-založený programovací model, ktorý umožňuje efektívne vyvíjať užívateľské rozhrania, či už jednoduché alebo zložité. [39]

Hlavnou myšlienkou vývoja Vue je poskytnúť oveľa jednoduchší koncept než má Angu-

lar. Preto je Vue.js považovaný za jeden z najprijateľnejších frameworkov pre začiatočníkov, ktorý prichádza s dobre prepracovanou dokumentáciou a podpornou komunitou. Vue.js má široký výber nástrojov, ako sú komplexné testovacie nástroje, systémy pre inštaláciu doplnkov, nástroje na ladenie webového prehliadača, renderovanie servera, správa stavu a ďalšie. [38]

2.7.4 Zhodnotenie výhod a nevýhod

Angular

Výhody:

- Komponentovo-založená architektúra
- Obojsmerná dátová väzba
- Vysoko testovateľné / znovupoužiteľné / spravovateľné aplikácie
- Silná komunita, dobré cvičné materiály atď.
- Podporované spoločnosťou Google

Nevýhody:

- Náročné pre začiatočníkov a príliš komplexné pre menšie tímy
- Obmedzené možnosti SEO
- Veľké množstvo kódu

[38]

React

Výhody:

- Podporované spoločnosťou Meta (Facebook)
- Pravidelne aktualizované
- Virtuálny DOM – konzistentný a bezproblémový výkon
- Možno kombinovať s mnohými ďalšími JS knižnicami
- Znovupoužitelné komponenty kódu

Nevýhody:

- Nedostatok dobre spracovanej dokumentácie
- Zložité učenie sa syntaxe JSX

[38]

Vue.js

Výhody:

- Rýchle a rozmerovo malé
- Prívetivé pre začiatočníkov
- Výborná a podrobná dokumentácia
- Jednoduchá syntax
- Obojsmerná dátová väzba
- Má pozitívny vplyv na SEO

Nevýhody:

- Relatívne malá komunita (ale stále rastie)
- Celkom nové a vyvinuté súkromnými osobami
- Obmedzená použiteľnosť pre väčšie projekty

[38]

Kapitola 3

Návrh

3.1 Výber architektúry

Informačný systém bude naimplementovaný vo forme klient - server aplikácie.

Pre serverovú aplikáciu som sa rozhodla zvoliť viacvrstevnatú architektúru, s ktorou som už mala skúsenosti v rámci výuky na univerzite pri tvorbe iných webových aplikácií. Viacvrstevnatá architektúra delí aplikáciu do viacerých vrstiev, kde každá vrstva je zodpovedná za konkrétnu prácu. Jednotlivé komponenty vo vrstvách komunikujú buď s komponentami vo svojej vrstve alebo s komponentami nižších vrstiev. V prípade viacvrstevnatej architektúry so striktnou interakciou komunikujú komponenty okrem vlastnej vrstvy nie so všetkými nižšími vrstvami, ale len s priamo susediacou nižšou vrstvou. [40]

Aplikácia bude pozostávať zo 4 hlavných vrstiev: perzistentná vrstva, biznis logika, REST API, aplikačná vrstva. Jednotlivé vrstvy sú nižšie zobrazené v diagrame komponent na 3.1, ktorý slúži na zachytávanie softvérovej architektúry.

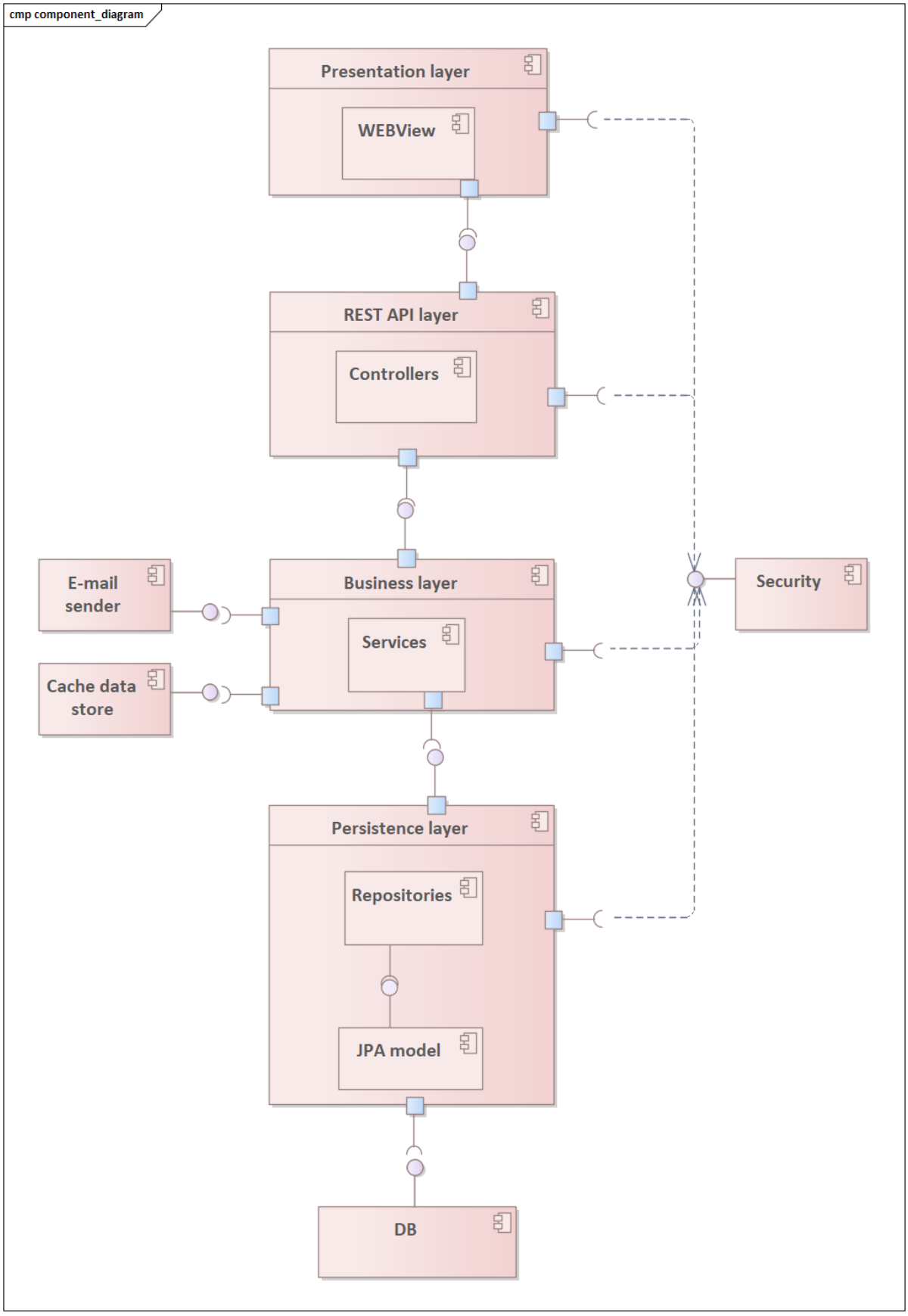
Najnižšia perzistentná vrstva bude napojená na PostgreSQL databázu a vo vnútri obsahuje komponenty JPA modelu - jednotlivé entity, s ktorými aplikácia pracuje. Tiež bude táto vrstva obsahovať tzv. rozhrania repositories - je to návrhový vzor, implementovaný Spring frameworkom a umožňuje vykonávať CRUD (Create, Read, Update, Delete) operácie nad entitami.

Nad perzistentnou vrstvou je biznis logika, táto vrstva sa tiež môže označovať ako aplikačná vrstva. Obsahuje komponenty služieb, ktoré zahŕňajú jadro aplikácie, jej logiku, funkcie, výpočty a spracovanie dát. Nad biznis logikou je vrstva REST API, ktorá vnútri obsahuje kontroléry, ktoré budú zachytávať príchodzie HTTP požiadavky od klienta.

Poslednou, najvyššou vrstvou je aplikačná vrstva, ktorá je zodpovedná za zobrazenie informácií pre užívateľov, väčšinou formou grafického užívateľského rozhrania, môže kontrolovať zadávané vstupy, neobsahuje však spracovanie dát. Táto vrstva bude realizovaná klientskou časťou aplikácie. [41]

Security (bezpečnostná) komponenta nesie zodpovednosť naprieč celou aplikáciou a má na starosti autentizáciu.

E-mail a cache komponenty budú napojené na vrstvu biznis logika.



Obr. 3.1: Diagram komponent

3.2 Technológie vo frontendovej aplikácii

Vue.js

Frontendová časť bude vyvinutá ako single-page aplikácia a bude napísaná v programovacom jazyku JavaScript spolu s frameworkom Vue.js, ktorý dnes patrí k najpopulárnejším open-source JavaScriptovým frameworkom. Keďže som predtým nemala skúsenosti so žiadnym JavaScriptovým frameworkom, moja osobná voľba pre Vue.js je odôvodnená hlavne kvôli jeho podrobnej dokumentácii, vynikajúcemu výkonu, jeho rastúcej popularite a bohatému množstvu doplnkových komponentov. [39]

Vuex

Vuex je vzor pre spravovanie stavu a tiež knižnica pre Vue.js aplikácie. Umožňuje skladovať komponenty v aplikácii a zaručuje isté pravidlá, podľa ktorých sa môže modifikovať stav komponent. [42]

V tejto práci sa Vuex využije pre ukladanie dát o aktuálne prihlásenom užívateľovi a pre ukladanie jeho JWT (JSON Web tokenu).

Vue Router

Vue Router je oficiálny smerovač pre Vue.js aplikácie. Táto knižnica sa integruje s jadrom Vue.js a umožňuje realizovanie navigácie u single - page aplikácií priradením určitých URL adries ku konkrétnym komponentom, ktoré by sa mali vykresliť. [43]

V tejto práci sa Vue Router knižnica využije na prepínanie jednotlivých komponent podľa URL adries.

Vuetify

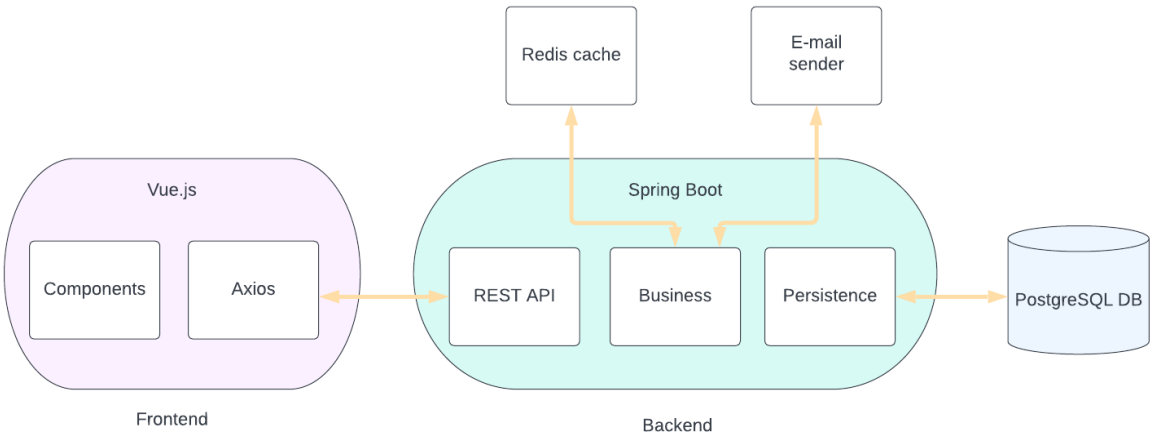
Vuetify je kompletný framework užívateľského rozhrania postavený na Vue.js. Jeho cieľom je poskytnúť vývojárom nástroje, ktoré potrebujú na vytváranie bohatých a responzívnych užívateľských rozhraní. Na rozdiel od iných frameworkov je Vuetify od základov navrhnutý tak, aby sa dal ľahko naučiť a jeho komponenty sú vytvorené podľa špecifikácie Material Design tak, aby boli modulárne. Vuetify kladie veľký dôraz na responzívny dizajn a zabezpečuje, aby aplikácie s jeho použitím správne fungovali, či už ide o telefón, tablet alebo stolný počítač. [44]

V tejto práci využijem Vuetify framework pri vytváraní užívateľského rozhrania, kvôli jeho kvalitne navrhnutým komponentom a udržiavaniu responzívneho dizajnu.

Axios

Axios je knižnica, ktorá sa využije v klientskej aplikácii a umožňuje odosielanie asynchrónnych HTTP požiadaviek na REST API na serverovej aplikácii. [45]

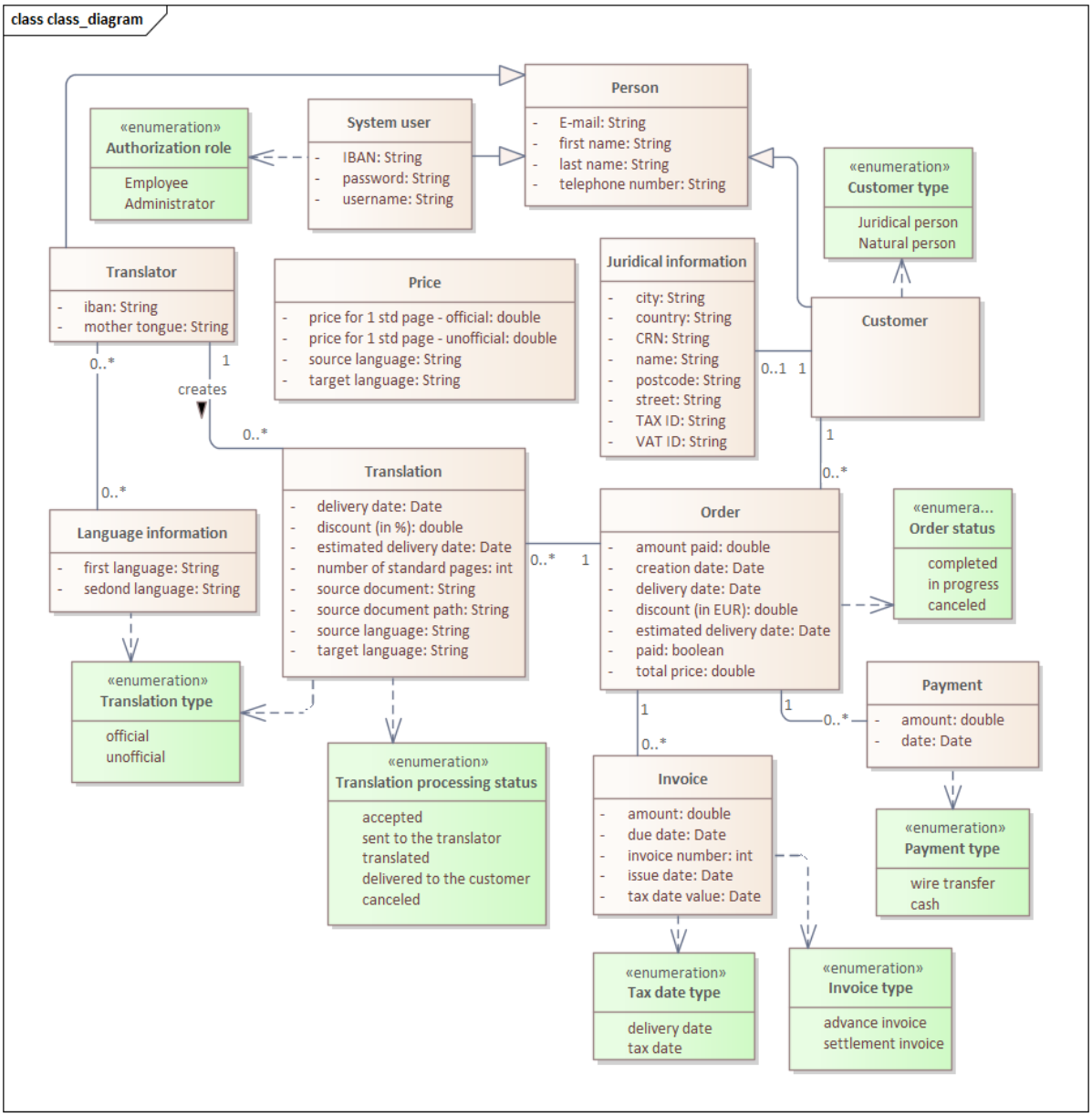
Znázornenie toho, ako bude backendová aplikácia napojená na frontendovú aplikáciu a zachytenie celkovej komunikácie popisuje nasledujúci diagram na 3.2.



Obr. 3.2: Diagram architektúry celého projektu

3.3 Dátový model

Vizualizáciu všetkých podstatných dát, s ktorými agentúra narába, popisuje diagram tried na 3.3. Diagram zobrazuje štruktúru jednotlivých entít projektu, ich atribúty a vzťahy medzi nimi.

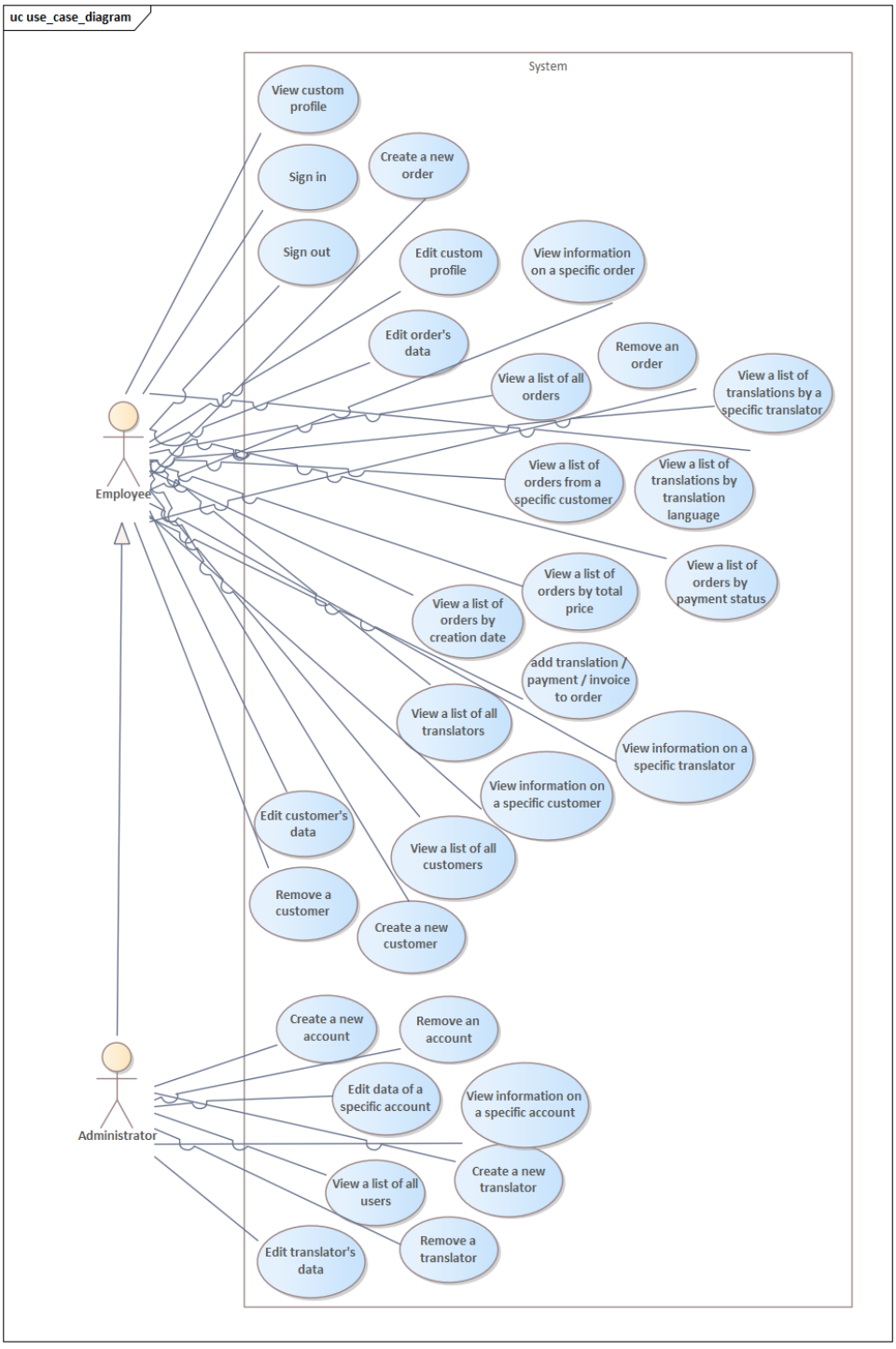


Obr. 3.3: Diagram tried

Entita System user predstavuje zamestnanca, ktorý pracuje v agentúre. Autorizačnú rolu zamestnanca som sa rozhodla zaznamenať pomocou výpočtového typu a nie pomocou dedičnosti, aby sa každému zamestnancovi dala prepnúť rola na inú rolu bez toho, aby sa musela inštancia zmazať a vytvoriť nová. Zároveň sa výpočetný typ hodí, keďže sa nepočíta s pridávaním nových autorizačných rolí za behu aplikácie.

Vizualizáciu jednotlivých funkcionalít pre obe užívateľské role popisuje diagram prípadov použitia na 3.4. Diagram prípadov použitia sa používa na znázornenie dynamického správania systému. Zapuzdruje funkčnosť systému tým, že zahŕňa prípady použitia, aktérov

a ich vzťahy. Modeluje úlohy, služby a funkcie požadované systémom / subsystémom aplikácie. [46]

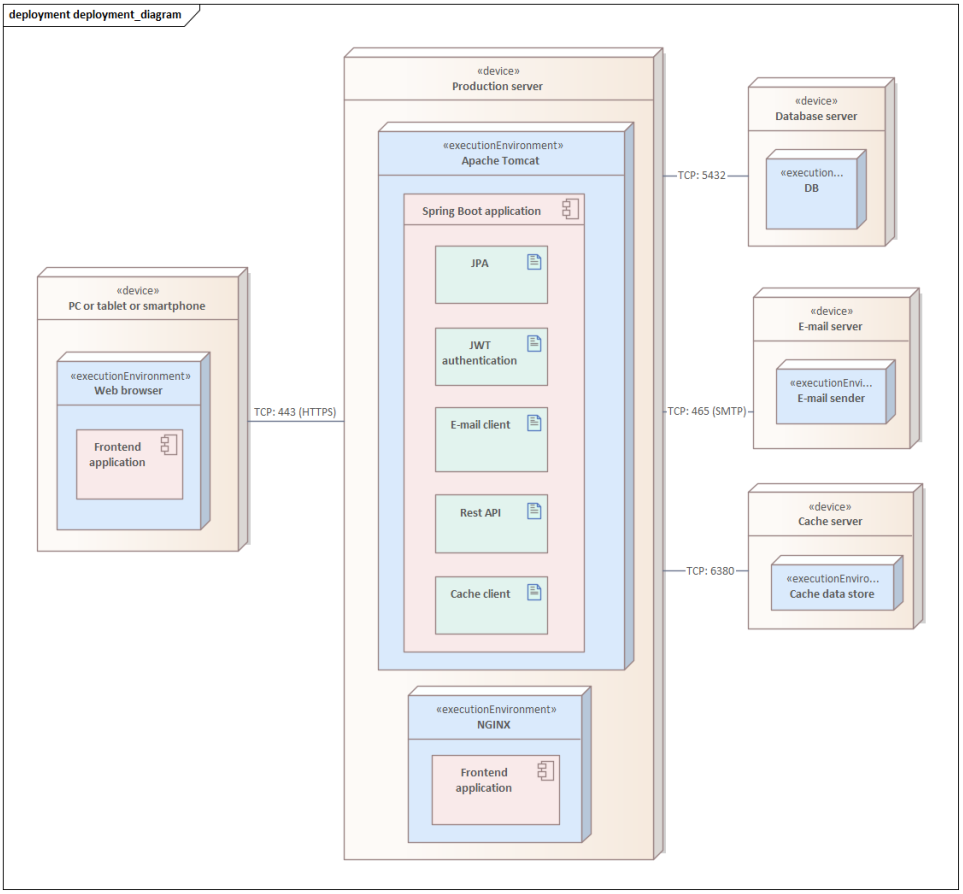


Obr. 3.4: Diagram prípadov použitia

Vizualizáciu topológie fyzických komponentov systému, na ktorých budú nasadené softvérové komponenty popisuje diagram nasadenia na 3.5.

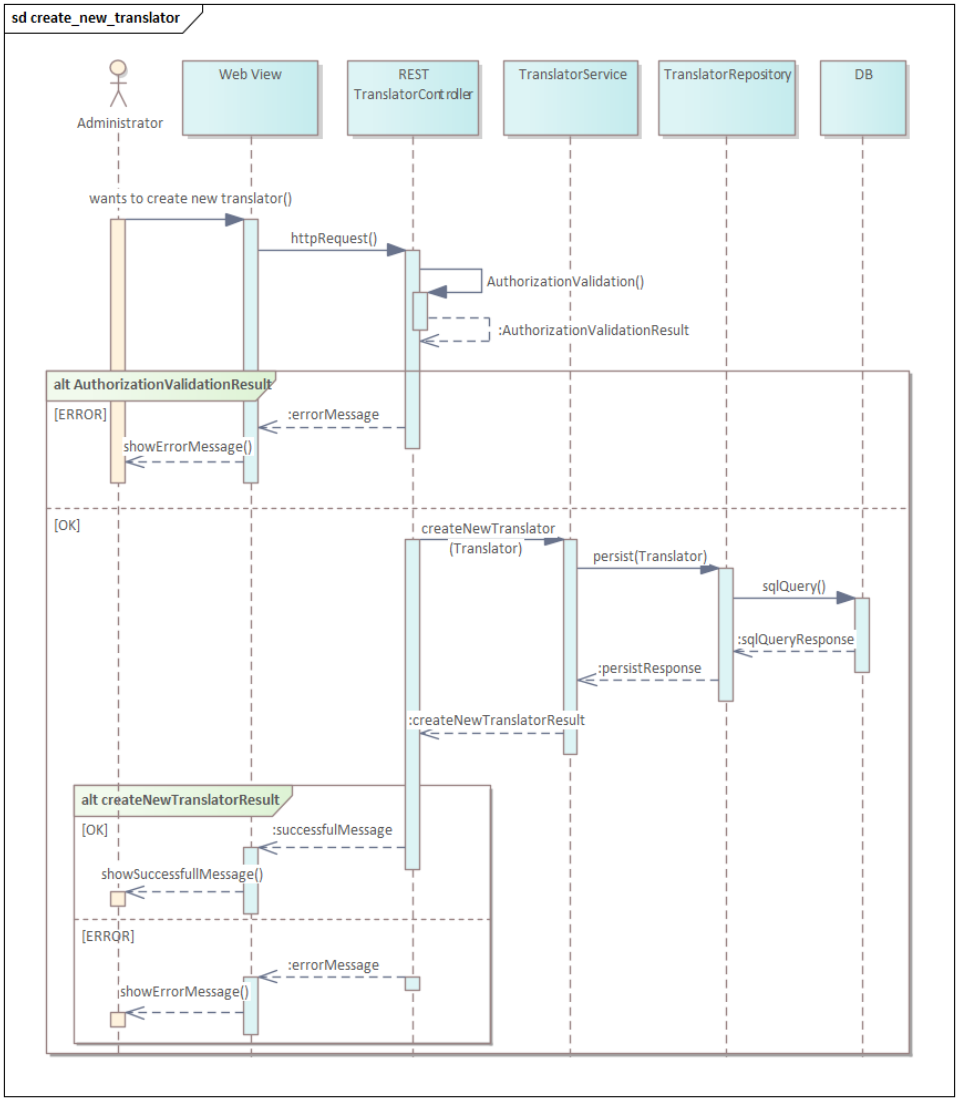
Backendová aplikácia, implementovaná s pomocou Spring Bootu, sa bude spúšťať pomocou zabudovaného serveru Apache Tomcat. Celý projekt bude nasadený na produkčnom serveri zvolenom firmou po dokončení implementácie. Backendová aplikácia bude napojená na PostgreSQL databázový server, E-mail SMTP server a Redis cache server.

Frontendová aplikácia bude poskytovaná webovým serverom Nginx a bude sa spúšťať u klienta vo webovom prehliadači.



Obr. 3.5: Diagram nasadenia

Vizualizáciu sekvencie činností pri procese vytvárania nového prekladateľa popisuje sekvenčný diagram na 3.6, ktorý modeluje interakcie medzi objektmi. Rovnakým princípom sa budú v aplikácií vytvárať aj ostatné entity.



Obr. 3.6: Sekvenčný diagram: Vytvorenie nového prekladateľa

3.4 Autentizácia

V tomto projekte sa využije autentizácia založená na tokenoch, a to s pomocou JSON Web Tokenu (JWT). Klient najprv odošle na serverovú stranu svoje prihlasovacie údaje. Serverová aplikácia overí tieto údaje, a ak sú správne, tak z nich vygeneruje JWT a odošle ich klientovi. Klient si tento JWT uloží a pripojí ho ku každému ďalšiemu požiadavku, ktorý vyžaduje autentizáciu. Server pri ďalšom požiadavku overí JWT od klienta a pošle mu zodpovedajúcu odpoveď.

V porovnaní s autentizáciou založenou na reláciách, kde je potrebné ukladať všetky relácie na strane servera, je veľkou výhodou tokenovo založenej autentizácie to, že tokeny sa ukladajú iba v lokálnom úložisku na strane klienta. [47]

3.5 Autorizácia

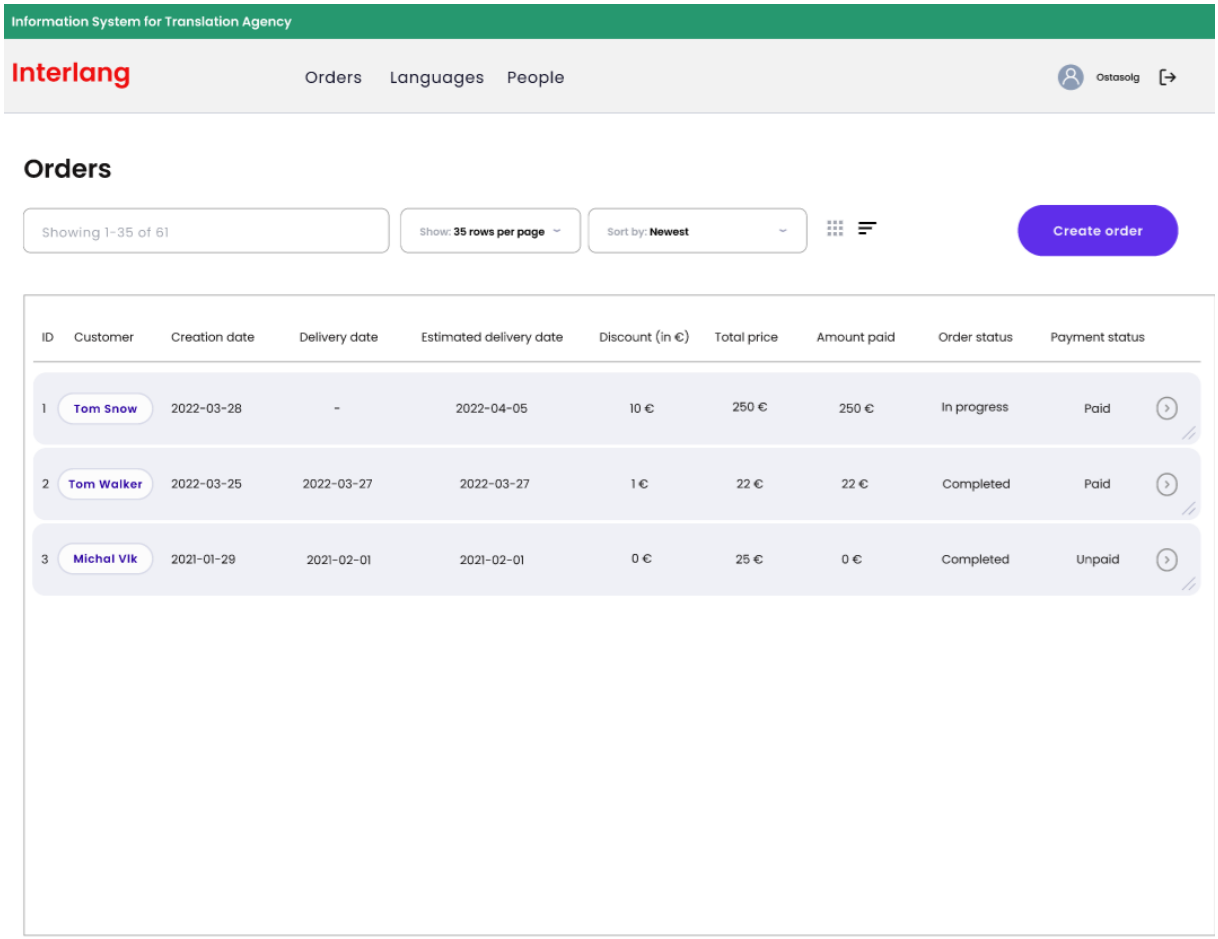
Autorizácia bude prebiehať aj na backendovej, aj na frontendovej strane. V backendovej časti bude autentizácia spolu s autorizáciou nastavená v Spring Security konfiguračných triedach. Vo frontendovej časti bude za autorizáciu zodpovedná komponenta Vue Router, ktorá bude určovať podľa preddefinovaných pravidiel, či sa prihlásenému užívateľovi môže zobraziť požadovaná stránka alebo nie.

3.6 Prototyp

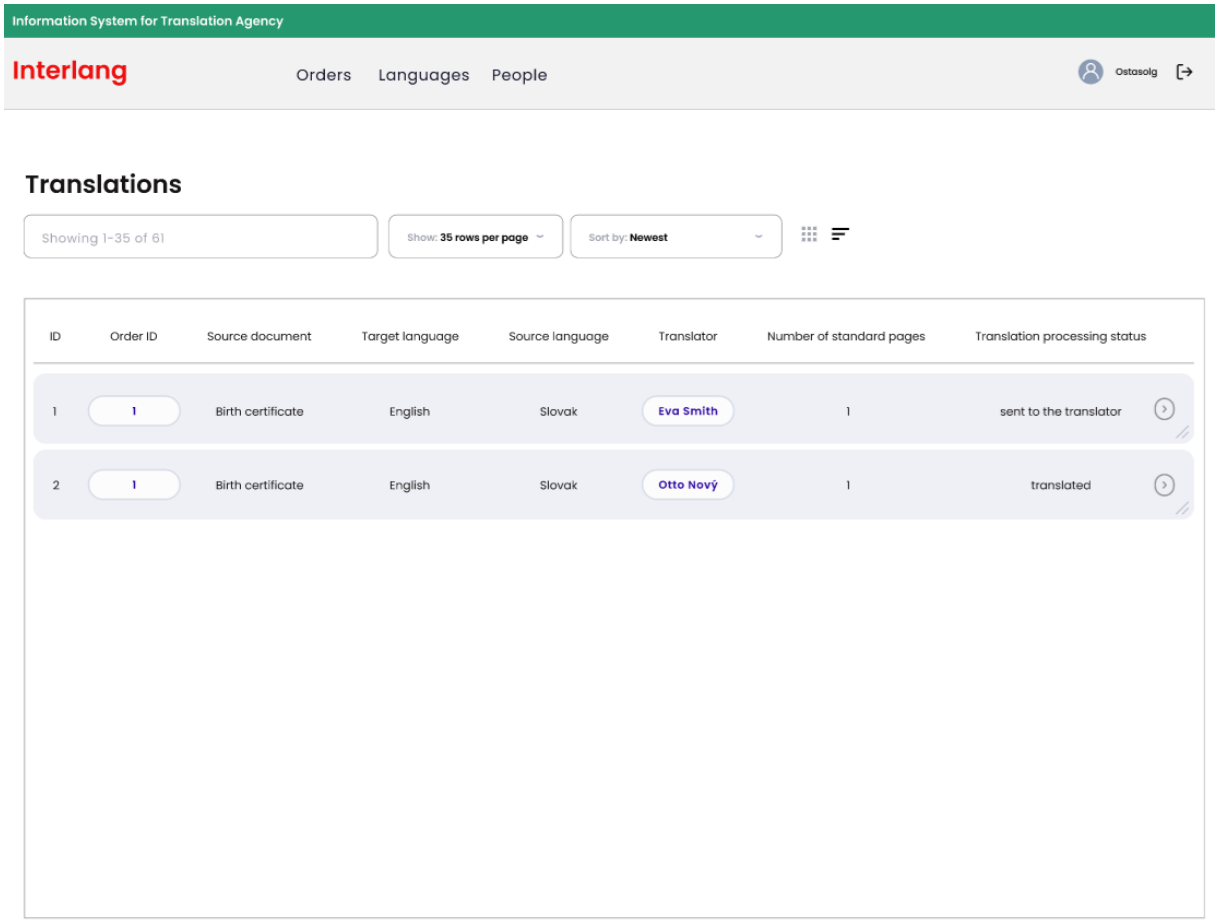
Pre daný projekt bol vytvorený prototyp zobrazujúci základnú funkcionalitu. K vytvoreniu som použila online vektorový grafický editor a nástroj na vytváranie prototypov Figma (https://www.figma.com).

Prototyp bol otestovaný vedúcim zamestnancom prekladateľskej agentúry. Celkový návrh aplikácie bol schválený. Ako pripomienky k danému prototypu vyznelo:

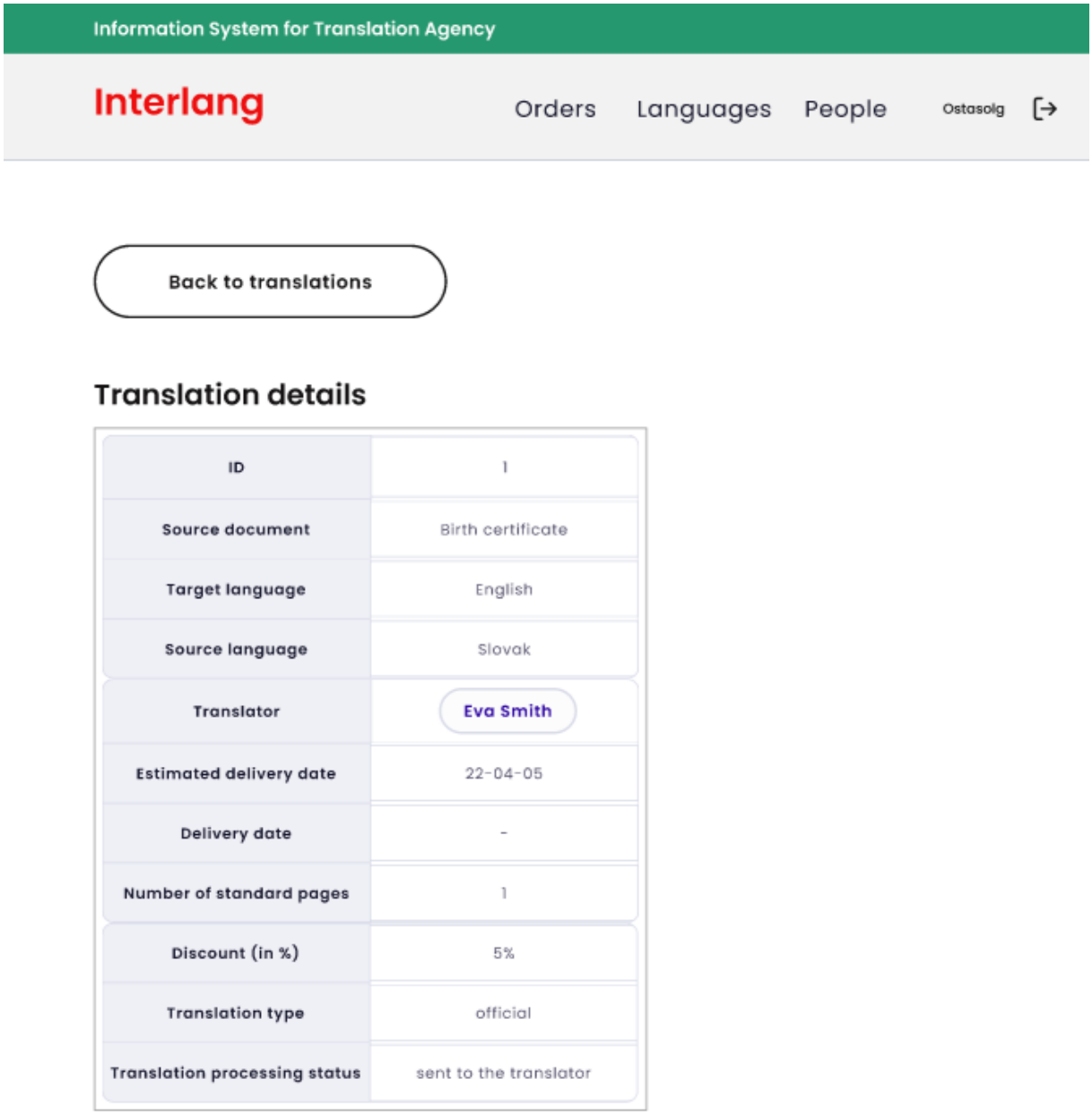
- V tabuľkách je potrebné zväčšiť veľkosť písma, aby bol text viac čitateľný. Obr. 3.7, 3.8, 3.10, 3.11
- Bolo by vhodné v tabuľkách k riadkom pridať tlačidla na editovanie a na odstránenie prvku. Obr. 3.7, 3.8, 3.10, 3.11
- V tabuľke všetkých objednávok netreba uvádzať stĺpce s hodnotami: zľava (v eurách), uhradená čiastka. Tieto hodnoty stačí uviesť, iba pri zobrazení detailov k jednej konkrétnej objednávke. Obr. 3.7
- Naopak v tabuľke všetkých prekladov je požadované zobraziť všetky atribúty entity prekladu. 3.8
- U entity prekladateľ nie je potrebné uchovávať jeho adresu a namiesto tohto stĺpca v tabuľke pridať stĺpec IBAN. Obr. 3.10



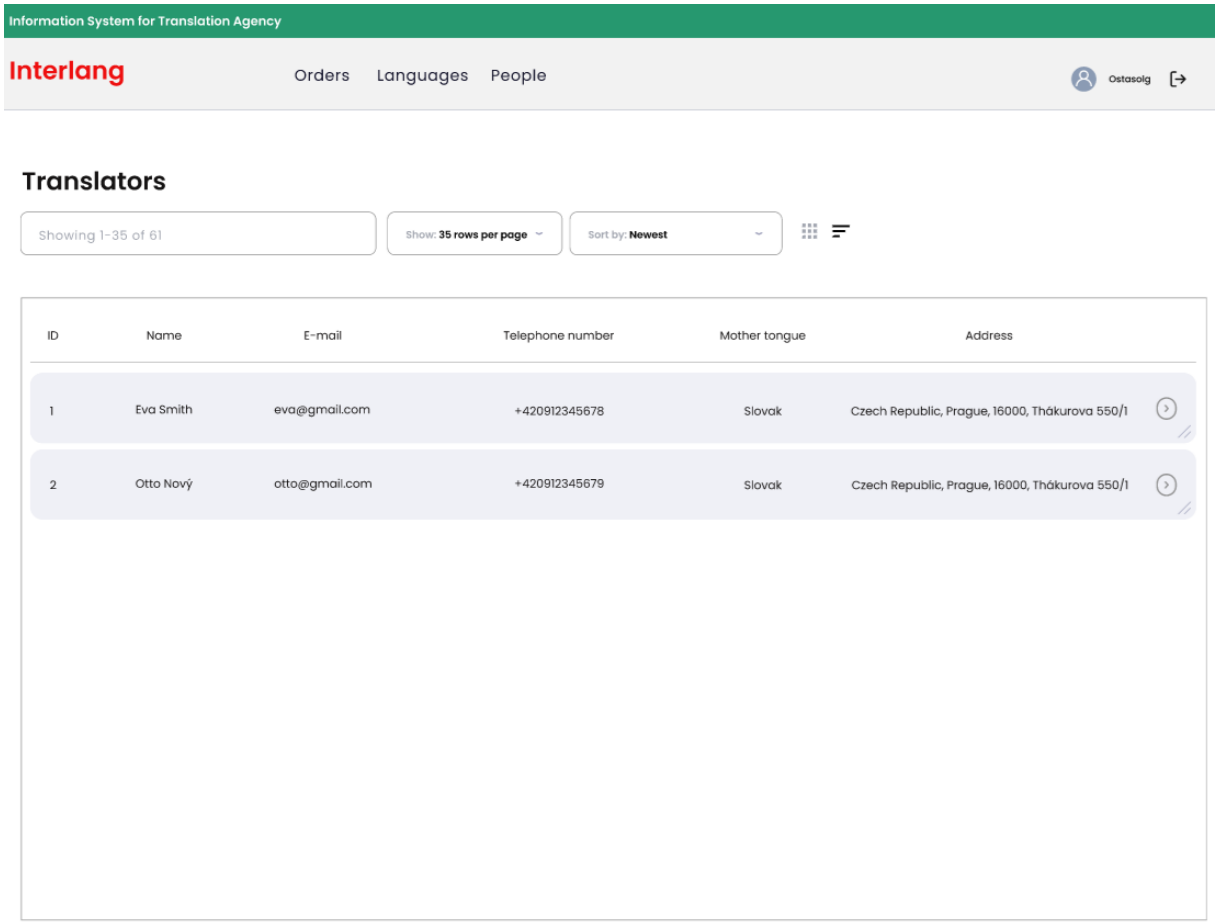
Obr. 3.7: Obrazovka všetkých objednávok



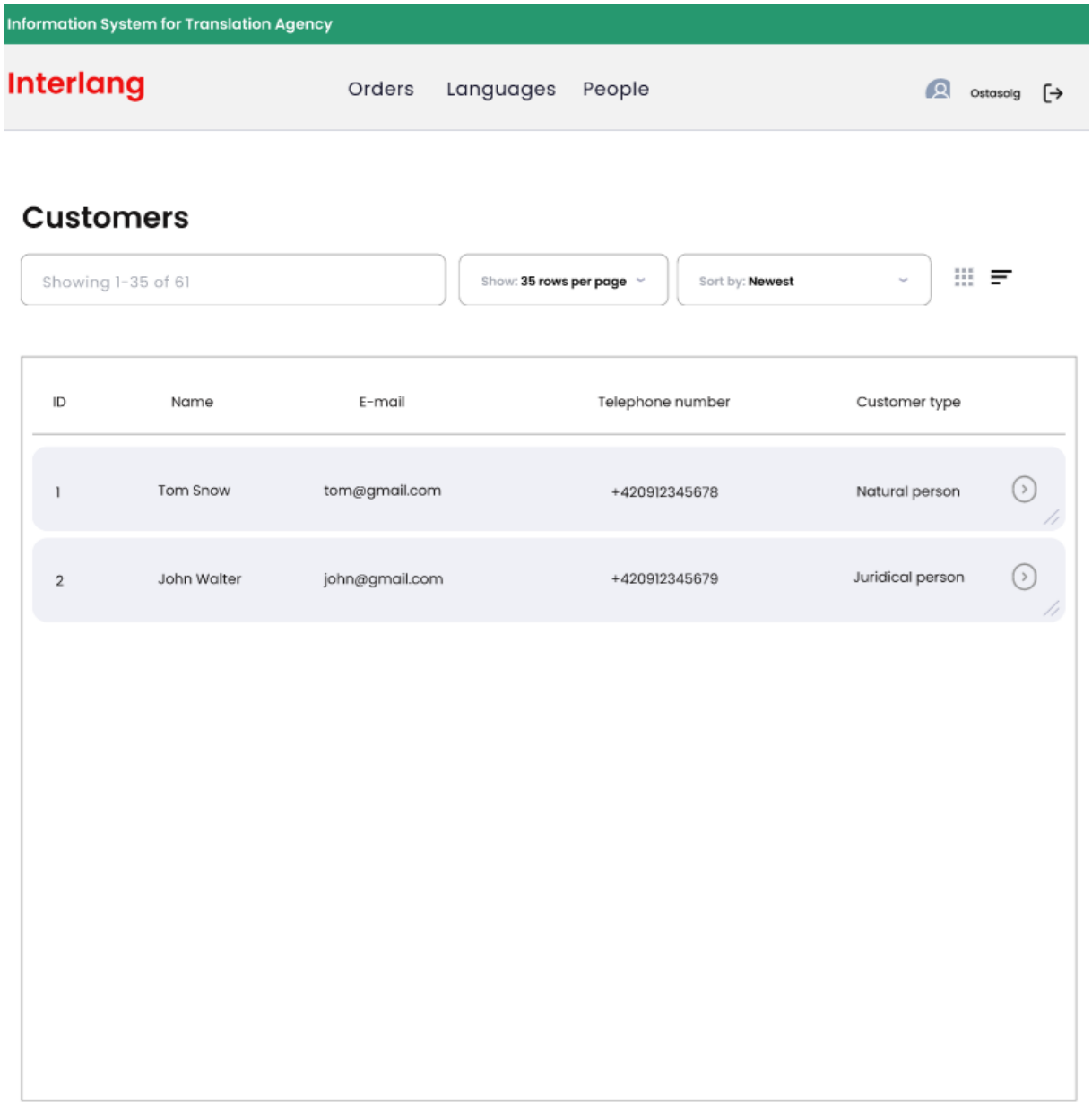
Obr. 3.8: Obrazovka všetkých prekladov



Obr. 3.9: Obrazovka detailov prekladu



Obr. 3.10: Obrazovka všetkých prekladateľov



Obr. 3.11: Obrazovka všetkých zákazníkov

Kapitola 4

Implementácia

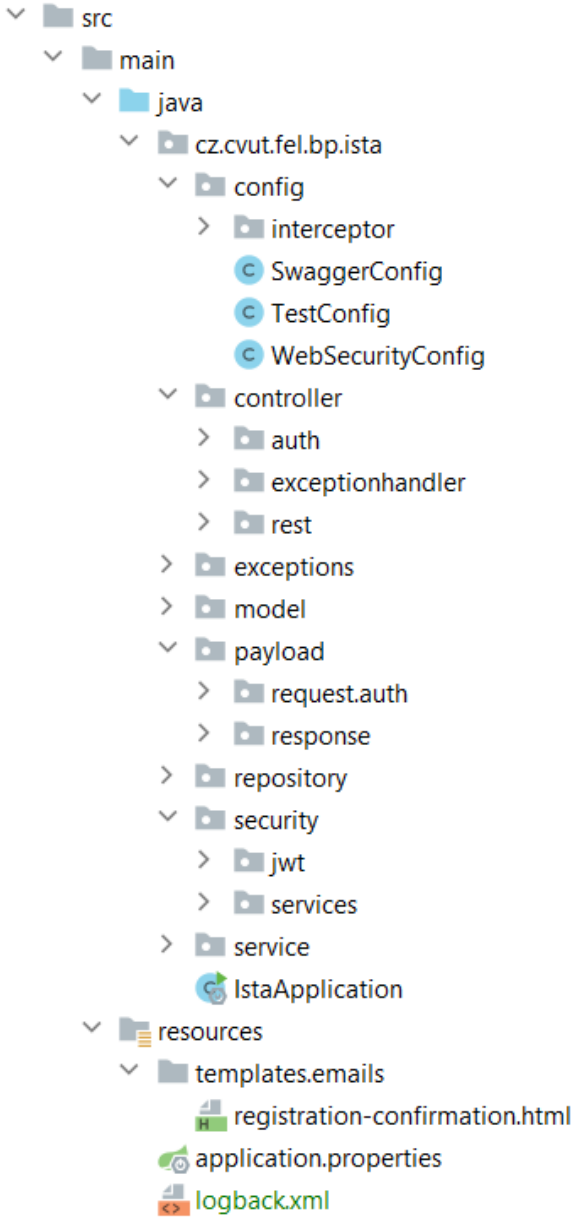
Implementácia informačného systému bola rozdelená do dvoch častí, backend a frontend. V tejto kapitole budem spracovávať postup a využitie technológií zvlášť pre obe tieto časti.

4.1 Backend

Projekt bol vygenerovaný s pomocou nástroja Spring Initializr, ktorý vytvára štruktúru pre Spring Boot projekty.

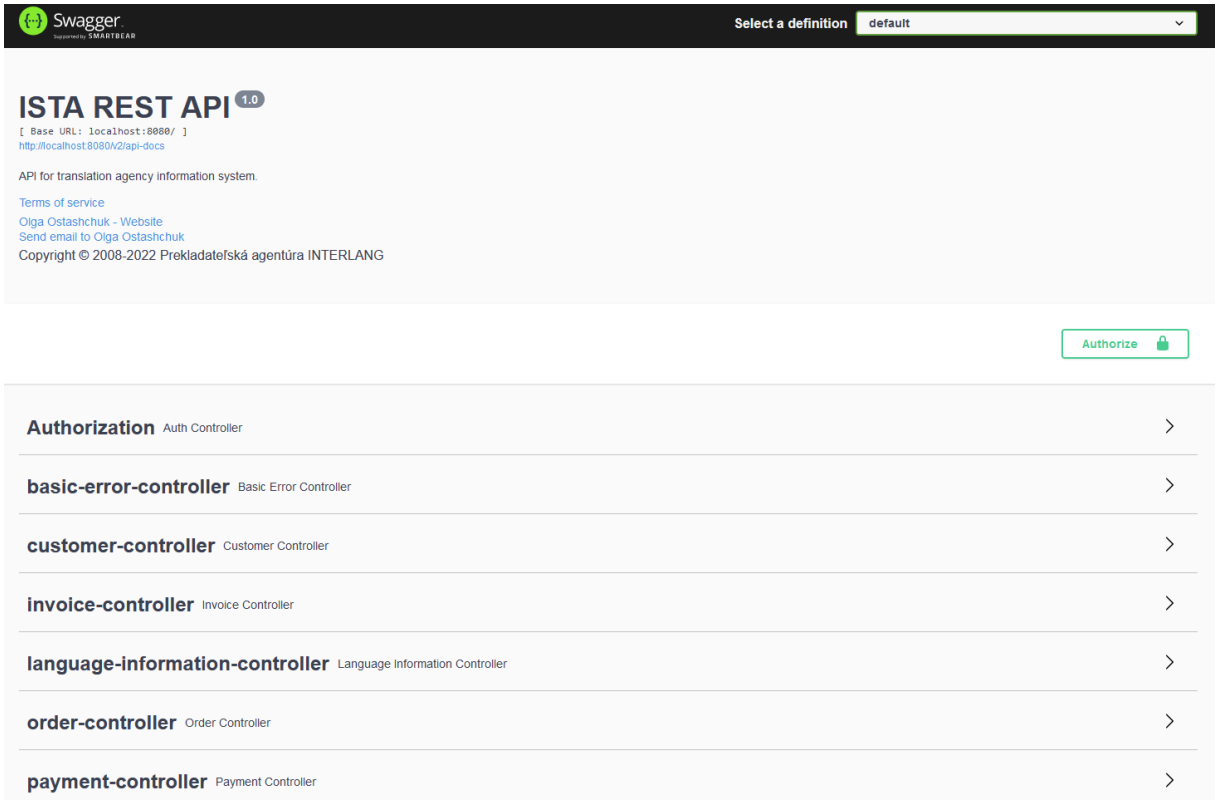
Pri implementácií som používala vývojové prostredie IntelliJ IDEA, ktoré umožňuje tvorbu aplikácií v programovacom jazyku Java, nástroj Maven, ktorý pomáha vybudovať projekt, knižnicu Lombok, ktorá pomáha redukovať štandardný kód a nástroj Swagger UI. Taktiež sa aplikácia pripája na PostgreSQL databázu, využíva Redis Cache a E-mail SMTP server na odosielanie e-mailov novovytvoreným užívateľom. K validácii bol použitý Hibernate Validator, referenčná implementácia Jakarta Bean Validation.

Štruktúra adresárov projektu je zobrazená na Obr. č.4.1.



Obr. 4.1: Štruktúra adresárov backendového projektu

V priečinku - config - sa nachádzajú konfiguračné súbory pre celú aplikáciu. Definujú sa tam aj nastavenia pre Swagger UI, čo je nástroj, ktorý umožňuje vytvoriť dokumentáciu pre REST API, interagovať s ním a rýchlo ho otestovať. [48]



Obr. 4.2: Swagger UI

Zároveň sa v priečinku - config - definujú nastavenia pre Spring Boot interceptory, ktoré umožňujú vykonať požadovanú činnosť pri prijímaní požiadavku od klienta alebo pred odoslaním odpovede klientovi. Daná funkcionalita bola využitá pre účely logovania. [49]

V triedach priečinku - controllers - boli naimplementované REST API kontroléry. Tiež tu boli s pomocou springovských anotácií @PreAuthorize nastavené užívateľské role, ktoré majú povolené pristupovať ku konkrétnym metódam. V tomto priečinku je aj zahrnuté odchytávanie a reagovanie na rozličné výnimky, ktoré môžu za behu aplikácie vzniknúť.

V zložke - exceptions - boli definované vlastné výnimky, ktoré môžu vzniknúť za behu aplikácie.

V priečinku model boli definované entity aplikácie.

V priečinku - payload - sú definované pomocné triedy, ktoré sa využijú pri vytváraní tela požiadavku alebo tela odpovede pre klienta.

V zložke - repository - sú rozhrania dediace od Sprinového JpaRepository, ktoré umožňujú vykonávať CRUD operácie nad entitami.

V priečinku - security - sú triedy, v ktorých sa implementuje generovanie a overovanie JWT.

Zložka - services - obsahuje triedy implementujúce logiku a jadro aplikácie.

Súbor - IstaApplication.java - predstavuje hlavný vstupný bod Spring Boot aplikácie.

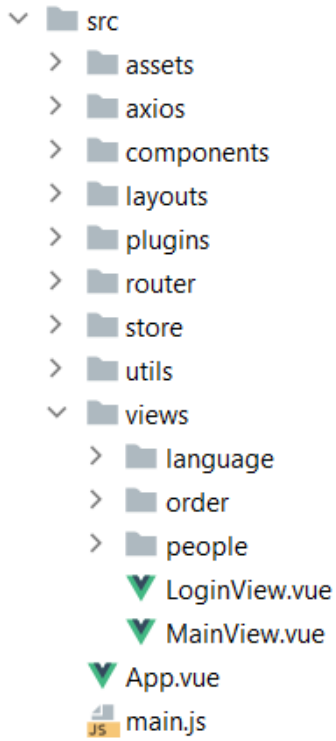
V zložke - resources - sa nachádza súbor - logback.xml - v ktorom je nastavená konfigurácia pre logovanie aplikácie. Logovanie je realizované pomocou nástrojov Logback a SLF4J. Tiež sa tam nachádza súbor pre celkové nastavenia aplikácie - application.properties - a súbor registration-configuration.html - kde sa nastavuje šablóna vo formáte html pre e-mailovú správu.

4.2 Frontend

K vytvoreniu frontendového projektu bolo potrebné spočiatku nainštalovať Node.js - prostredie, ktoré umožňuje spúšťať kód v programovacom jazyku JavaScript, následne nainštalovať npm - Node Package Manager, ktorý umožňuje inštalovať ďalšie potrebné komponenty, ako napríklad Vue CLI, ktorá je požadovaná k inicializácií Vue.js projektu.

Zároveň bolo potrebné nainštalovať všetky požadované komponenty spomenuté v predošlej kapitole a určiť verziu Vue.js. V tomto projekte bola zvolená 2. verzia Vue.js z dôvodu, že ešte nebola vydaná nová verzia Vuetify frameworku, ktorá by podporovala 3. verziu Vue.js.

Štruktúra adresárov projektu je je zobrazená na Obr. č.4.3.



Obr. 4.3: Štruktúra adresárov frontendového projektu

V zložke - assets - sa nachádzajú doplnkové súbory, ako napríklad obrázky loga firmy.

V zložke - axios - je konfigurácia pre Axios komponentu, ktorá umožňuje odosielanie HTTP požiadaviek na server. Zároveň sa v tejto časti určí základná URL adresa servera tak, aby sa vo zvyšných častiach frontendovej aplikácie mohli používať už iba relatívne cesty. Taktiež sa tu využívajú interceptory na vykonanie požadovanej činnosti pri prijatí HTTP požiadavku alebo pred odoslaním HTTP požiadavku. V tomto prípade interceptory reagujú na stav odpovede od servera a v prípade chyby sa užívateľovi zobrazí notifikačná hláška.

V priečinku - components - sú vytvorené UI komponenty.

V priečinku - layouts - sú vytvorené layout komponenty.

V zložke - plugins - je zahrnutý konfiguračný súbor importujúci dodatočné knižnice.

V zložke - router - je konfigurácia pre Vue Router komponentu, ktorá realizuje navigáciu v single-page aplikácii. Priraďujú sa tu určité URL adresy ku konkrétnym komponentom, ktoré by sa mali vykresliť. Vue Router komponenta je tiež zodpovedná za autorizáciu, ktorá určuje podľa preddefinovaných pravidiel, či sa prihlásenému užívateľovi

môže zobraziť požadovaná stránka alebo nie. V prípade, že prihlásený užívateľ na zobrazenie obsahu stránky nemá autorizačné práva, tak sa mu zobrazí notifikačná hláška alebo bude presmerovaný na inú stránku.

V zložke - store - je konfigurácia pre Vuex Store. Táto komponenta je dôležitá pre implementovanie autentizácie s JWT, keďže práve ona ukladá serverom vygenerovaný JWT. Táto komponenta bola v tomto projekte využitá aj pre ukladanie údajov o prihlásenom užívateľovi. Údaje uložené touto komponentou sú prístupné z ľubovoľného iného miesta frontendovej aplikácie.

V priečinku - utils - sú pridané dodatočné podporné funkcie.

V priečinku - views - obsahuje komponenty, ktoré budú vykresľované ako samostatné stránky.

Súbor - App.vue - predstavuje hlavný súbor Vue.js aplikácie.

Súbor - main.js - predstavuje hlavný JavaScriptový súbor, do ktorého je pripojená Vue.js aplikácia.

4.3 Naplnenie požiadaviek

FR01: Vytvorenie nového užívateľa

Splnené v backendovej aplikácií pomocou metódy createSystemUser v triede SystemUserController. Danú metódu môže spustiť iba prihlásený užívateľ s autorizačnou rolou Admin (administrátor). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore SystemUsersNewView.vue. Autorizačné pravidlá sú nastavené v zložke router.

FR02 / FR11: Úprava účtu užívateľa / Úprava vlastného účtu

Splnené v backendovej aplikácií pomocou metódy updateSystemUser v triede SystemUserController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Admin môže editovať údaje hociktorého užívateľa, User môže editovať údaje iba o vlastnom účte. Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore SystemUsersEditView.vue.

FR03: Zrušenie účtu užívateľa

Splnené v backendovej aplikácií pomocou metódy deleteSystemUser v triede SystemUserController. Danú metódu môže spustiť iba prihlásený užívateľ s autorizačnou rolou Admin (administrátor). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore SystemUsersTableView.vue.

FR04 / FR10: Zobrazenie údajov užívateľa / Zobrazenie vlastného účtu

Splnené v backendovej aplikácií pomocou metódy getSystemUserById v triede SystemUserController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Admin si môže zobraziť údaje hociktorého užívateľa, User môže vidieť údaje iba o vlastnom účte. Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore SystemUsersEditView.vue.

FR05: Zobrazenie zoznamu všetkých užívateľov

Splnené v backendovej aplikácií pomocou metódy getAllSystemUsers v triede SystemUserController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore SystemUsersTableView.vue.

FR06: Pridanie nového prekladateľa

Splnené v backendovej aplikácií pomocou metódy createTranslator v triede TranslatorController. Danú metódu môže spustiť iba prihlásený užívateľ s autorizačnou rolou Admin (administrátor). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore TranslatorsNewView.vue.

FR07: Úprava údajov o prekladateľovi

Splnené v backendovej aplikácií pomocou metódy updateTranslator v triede TranslatorController. Danú metódu môže spustiť iba prihlásený užívateľ s autorizačnou rolou Admin (administrátor). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore TranslatorsEditView.vue.

FR08: Odstránenie prekladateľa

Splnené v backendovej aplikácií pomocou metódy deleteTranslator v triede TranslatorController. Danú metódu môže spustiť iba prihlásený užívateľ s autorizačnou rolou Ad-

min (administrátor). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore TranslatorsTableView.vue.

FR09: Prihlásenie / odhlásenie užívateľa

Prihlásenie je splnené v backendovej aplikácií pomocou metódy authenticateSystemUser v triede AuthUserController. Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore LoginView.vue. Odhlásenie je realizované na strane frontendovej aplikácie tým, že sa užívateľovi zmaže z lokálneho úložiska jeho JWT.

FR12: Zadanie novej objednávky

Splnené v backendovej aplikácií pomocou metódy createOrder v triede OrderController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersNewView.vue.

FR13: Úprava objednávky

Splnené v backendovej aplikácií pomocou metódy updateOrder v triede OrderController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersEditView.vue.

FR14: Vymazanie objednávky

Splnené v backendovej aplikácií pomocou metódy deleteOrder v triede OrderController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersTableView.vue.

FR15: Vytvorenie prekladu, úhrady a faktúry

Splnené v backendovej aplikácií pomocou metód createTranslation v TranslationController, createPayment v PaymentController a createInvoice v InvoiceController. Dané metódy môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súboroch TranslationsNewView.vue, PaymentsNewView.vue a InvoicesNewView.vue.

FR16: Úprava prekladu, úhrady a faktúry

Splnené v backendovej aplikácií pomocou metód updateTranslation v TranslationController, updatePayment v PaymentController a updateInvoice v InvoiceController. Dané metódy môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súboroch TranslationsEditView.vue, PaymentsEditView.vue a InvoicesEditView.vue.

FR17: Vymazanie prekladu, úhrady a faktúry

Splnené v backendovej aplikácií pomocou metód deleteTranslation v TranslationController, deletePayment v PaymentController a deleteInvoice v InvoiceController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersTableView.vue.

FR18: Pridelenie prekladov, úhrad a faktúr k objednávke

Splnené v backendovej aplikácií pomocou metód createOrder a updateOrder v triede OrderController. Dané metódy môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súboroch OrdersNewView.vue a OrdersEditView.vue.

FR19: Zobrazenie údajov objednávky

Splnené v backendovej aplikácií pomocou metódy getOrderById v triede OrderController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersEditView.vue.

FR20 / FR21: Zobrazenie zoznamu všetkých objednávok / Zobrazenie zoznamu objednávok podľa vlastností

Splnené v backendovej aplikácií pomocou metódy getAllOrders v triede OrderController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore OrdersTableView.vue, kde je tiež realizované filtrovanie objednávok podľa konkrétnych vlastností.

FR22: Zobrazenie údajov prekladateľa

Splnené v backendovej aplikácií pomocou metódy getAllTranslators v triede TranslatorController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore TranslatorsTableView.vue.

FR23: Zobrazenie zoznamu všetkých prekladateľov

Splnené v backendovej aplikácií pomocou metódy getAllTranslators v triede TranslatorController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore TranslatorsTableView.vue

FR24: Zobrazenie údajov zákazníka

Splnené v backendovej aplikácií pomocou metódy getCustomerById v triede CustomerController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore CustomersEditView.vue.

FR25: Zobrazenie zoznamu všetkých zákazníkov

Splnené v backendovej aplikácií pomocou metódy getAllCustomers v triede CustomerController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore CustomersTableView.vue

FR26: Pridanie nového zákazníka

Splnené v backendovej aplikácií pomocou metódy createCustomer v triede CustomerController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore CustomersNewView.vue.

FR27: Úprava údajov o zákazníkovi

Splnené v backendovej aplikácií pomocou metódy updateCustomer v triede CustomerController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore CustomersEditView.vue.

FR28: Odstránenie zákazníka

Splnené v backendovej aplikácií pomocou metódy deleteCustomer v triede CustomerController. Danú metódu môže spustiť prihlásený užívateľ s autorizačnou rolou Admin (administrátor) a User (zamestnanec). Vo frontendovej aplikácií je tento požiadavok realizovaný v súbore CustomersTableView.vue.

FR29: Odosielanie notifikačných emailov

Splnené v backendovej aplikácií pomocou metódy sendRegistrationConfirmationEmail v triede SystemUserService. Daná metóda sa vykoná po vytvorení nového užívateľa.

Frontend a backend aplikácia boli navrhnuté a naimplementované spôsobom, aby boli splnené aj všetky nefunkčné požiadavky.

Snímky obrazoviek vytvoreného informačného systému sú dodané v prílohe dokumentu.

4.4 Nasadenie

Backend aj frontend aplikácia boli nasadené na Linuxový virtuálny server s využitím Docker kontajnerov. K vytvoreniu Docker kontajnerov bol použitý konfiguračný súbor Dockerfile (4.4, 4.5).

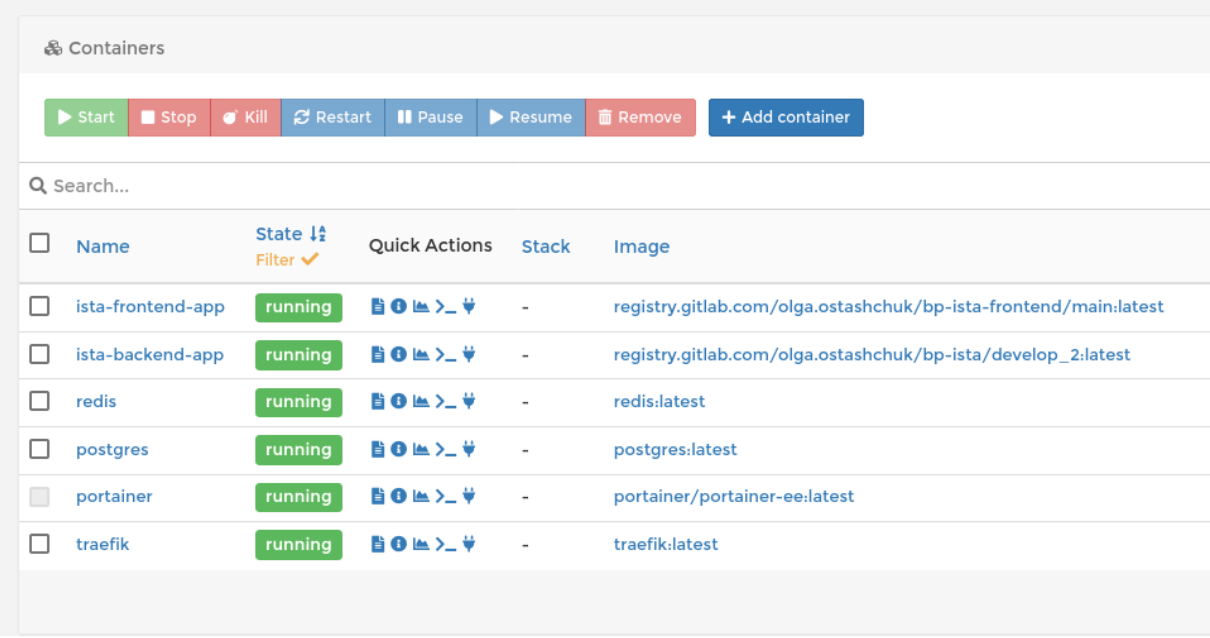
```
1  FROM openjdk:17-jdk-alpine
2
3  # set time zone to Europe/Prague
4  ENV TZ=Europe/Prague
5  RUN apk add tzdata
6  RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
7
8  ADD /target/*.jar ISTA-is-0.0.1.jar
9
10 ENTRYPOINT ["java","-jar","/ISTA-0.0.1.jar"]
```

Obr. 4.4: Dockerfile pre backend aplikáciu

```
FROM node:lts-alpine3.10 as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY ./ .
RUN npm run build

FROM nginx:stable-alpine as production-stage

ENV TZ=Europe/Prague
RUN apk add tzdata
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

RUN mkdir /app
COPY --from=build-stage /app/dist /app
COPY nginx.conf /etc/nginx/nginx.conf
```

Obr. 4.5: Dockerfile pre frontend aplikáciu

Procesy vytvárania backend a frontend aplikácie sú riadené pomocou GitLab CI/CD pipeline. Komunikácia s Docker kontajnermi prebieha cez proxy server.



Obr. 4.6: Zoznam Docker kontajnerov

Pre pohodlnejšiu správu Docker kontajnerov bol využitý nástroj Portainer. Prehľad všetkých kontajnerov je na Obr. 4.6.

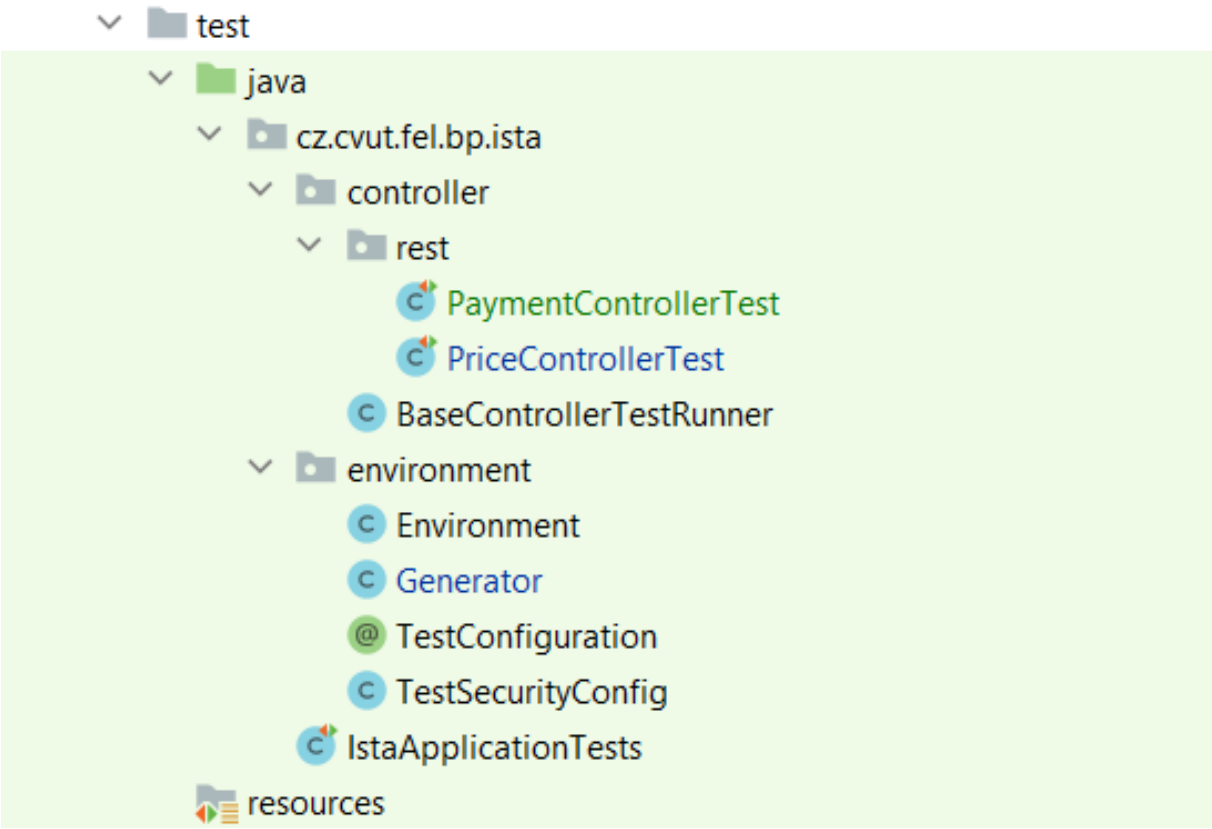
Kapitola 5

Testovanie

Testovanie je proces, ktorý umožňuje odhaliť chyby vytvoreného systému, zistiť jeho obmedzenia a poskytuje možnosť overiť, či daný systém spĺňa požadovanú funkcionalitu. Je to dôležitý krok pri tvorbe ľubovoľného nového softvéru. [50]

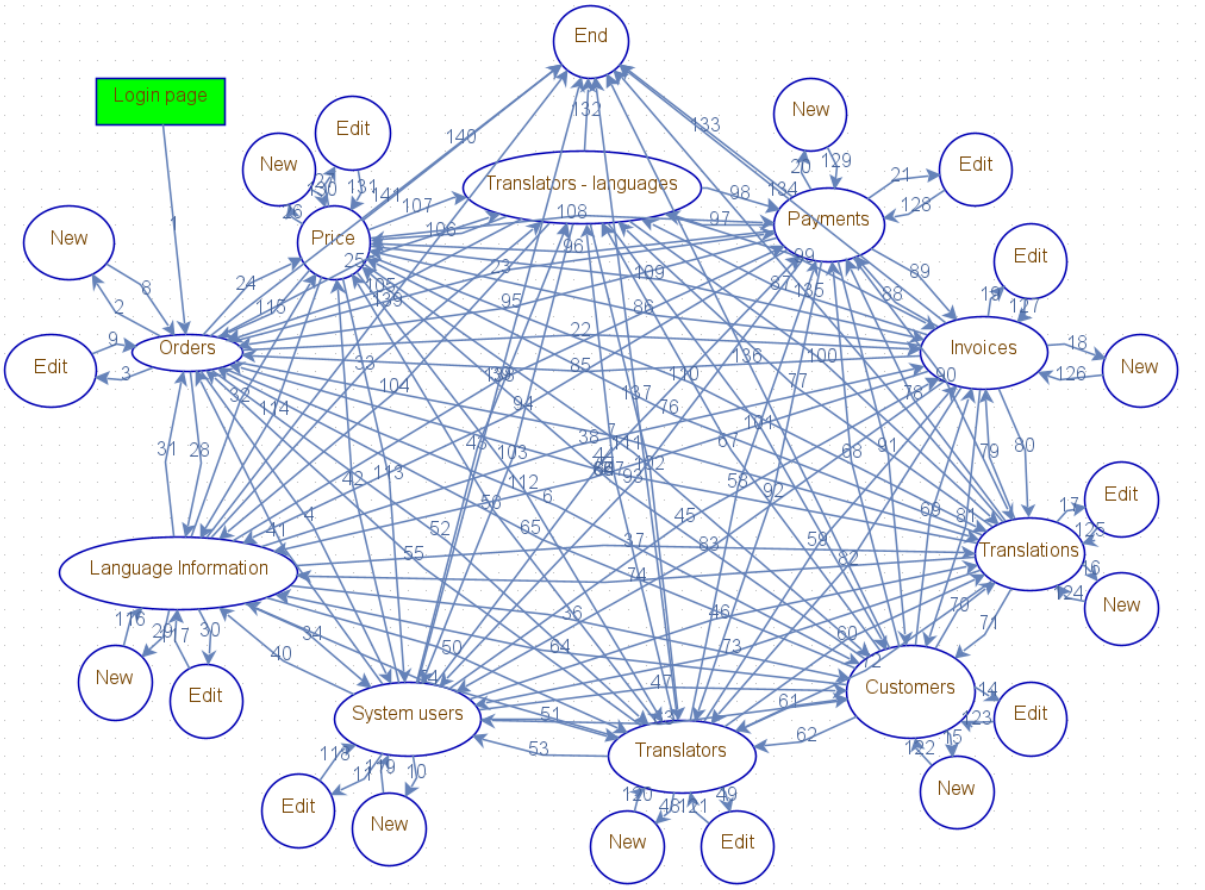
Pre testovanie backendovej aplikácie bol využitý open - source framework Mockito a open - source framework JUnit. S ich pomocou boli vytvorené jednotkové testy pre otestovanie hlavných funkcionalít. [51]

Mockito je framework, knižnica založená na programovacom jazyku Java, ktorá sa používa na efektívne jednotkové testovanie Java aplikácií. Mockito umožňuje simuláciu rozhraní tak, že do simulovaného rozhrania možno pridať fiktívnu funkcionalitu a následne využiť toto rozhranie pri testovaní, akoby sa jednalo o skutočné rozhranie. Je to užitočný nástroj, keď je potrebné otestovať nejakú funkcionalitu v izolovanom prostredí a nie v rámci celej aplikácie s napojenými ďalšími komponentami. [52]



Obr. 5.1: Štruktúra adresárov k testovaniu backend aplikácie

Na otestovanie frontendovej aplikácie a systému ako celku boli pre tento projekt pripravené aj Selenium testy, ktoré vystupujú v roli akýchsi užívateľských akceptačných testov. Tieto testy boli implementované v programovacom jazyku Python s pomocou balíka Selenium a s pomocou webového prehliadača Firefox.



Obr. 5.2: Graf pre tvorbu testovacích scenárov

Pri procese tvorby testovacích scenárov bol vytvorený graf na Obr. č.5.2, ktorý pomocou vrcholov zobrazuje miesta v aplikácii, na ktorých sa užívateľ môže nachádzať. Hrany medzi vrcholmi predstavujú, akými spôsobmi sa užívateľ medzi vrcholmi môže pohybovať.

Následne, na základe vytvoreného grafu boli vytvorené scenáre pre testovanie. Využili sa levely hĺbok testovania 1 a 2. Pri leveli hĺbky testovania 1 sa do testovacích scenárov zahrnuli všetky výstupy z každého vrcholu aspoň raz. Pri leveli hĺbky testovania 2 sa robili kombinácie každého vstupu a výstupu z vrcholu pre všetky vrcholy. Každá vytvorená kombinácia sa rovnako musela v testovacích scenároch vyskytnúť minimálne jedenkrát. [53]

Test situations 3, TDL= 2, ALG= PCT.

Sub-combinations of edges | Test situations

No.	Test sequence
28	1 - 24 - 111 - 136
29	1 - 25 - 101 - 136
30	1 - 52 - 55 - 52 - 61 - 136
31	1 - 7 - 135
32	1 - 4 - 46 - 135
33	1 - 6 - 70 - 135
34	1 - 23 - 90 - 135
35	1 - 22 - 80 - 135
36	1 - 7 - 16 - 124 - 135
37	1 - 7 - 17 - 125 - 135
38	1 - 25 - 104 - 38 - 87 - 104 - 50 - 58 - 94 - 50 - 59 - 88 - 97 - 104 - 37 - 77 - 132
39	1 - 28 - 37 - 135
40	1 - 24 - 110 - 135
41	1 - 25 - 100 - 135
42	1 - 25 - 102 - 60 - 77 - 98 - 20 - 129 - 133
43	1 - 52 - 60 - 135
44	1 - 22 - 134
45	1 - 4 - 45 - 134
46	1 - 6 - 69 - 134
47	1 - 7 - 79 - 134
48	1 - 23 - 89 - 134
49	1 - 22 - 18 - 126 - 134
50	1 - 22 - 19 - 127 - 134
51	1 - 28 - 38 - 134
52	1 - 24 - 109 - 134

Situations: 112, Steps: 1802, High: 0, Medium: 0, Low: 1802, Unique nodes: 29, Unique edges: 137

Obr. 5.3: Ukážka vytvorených kombinácií priechodov aplikáciou s levelom hĺbky testovania 2

Vytvorené kombinácie boli využité pri tvorbe testovacích scenárov a scenáre s najvyššou prioritou boli následne otestované.

Záver testovania

Celkovo bola aplikácia priebežne testovaná a všetky funkčné a nefunkčné požiadavky boli splnené a chyby, ktoré sa počas testovania podarilo odhaliť, boli odstránené.

Kapitola 6

Záver

Hlavným cieľom tejto bakalárskej práce bolo vytvoriť nový informačný systém pre riadenie a prevádzkovanie slovenskej prekladateľskej agentúry INTERLANG.

Pre dosiahnutie tohto cieľa bol projekt rozdelený na nasledovné etapy:

- analýza súčasného stavu prevádzkových podnikových procesov
- zber a analýza požiadaviek na nový informačný systém
- stanovenie rozsahu riešenia, ktoré bude vytvorené v rámci projektu
- návrh architektúry riešenia
- dizajn jednotlivých častí a komponentov
- vývoj
- inštalácia riešenia na produkčný server
- testovanie
- akceptácia a odovzdanie zákazníkovi

Analýza súčasného stavu prevádzkových procesov a zber funkčných a nefunkčných požiadaviek na nový podnikový informačný systém boli uskutočnené formou riadených rozhovorov s vedením spoločnosti.

S ohľadom na dostupný čas pre realizáciu projektu, náročnosť práce, dostupnosť softvérových nástrojov a technických prostriedkov, bol stanovený a dohodnutý rozsah riešenia. Pritom som sa snažila zahrnúť všetky dôležité požiadavky a nechať otvorené možnosti pre neskoršie zdokonalenie a rozšírenie riešenia (napríklad, mimo rozsah tohto

projektu bola vyčlenená integrácia vyvíjaného informačného systému so softvérom na vedenie účtovníctva, obchodné analýzy a integrácia so systémom personálneho oddelenia).

Architektúra riešenia bola navrhnutá na základe požiadaviek s ohľadom na aktuálne trendy v oblasti IT a maximálne využitie open - source produktov a nástrojov.

Užívateľské rozhranie bolo navrhnuté s cieľom zabezpečiť, čo najlepšiu ergonómiu, zrozumiteľnosť a pohodlie užívateľov.

Projekt bol nasadený na produkčný server a je dostupný na nasledujúcich URL:
Frontend aplikácia - https://ista.interlang.cz
Backend aplikácia - https://backend.ista.interlang.cz

S použitím nového informačného systému v testovacom režime sa zefektívnila prevádzka agentúry. Vnútorne procesy sú prehľadnejšie a rýchlejšie. Prijímanie a správa objednávok prebieha teraz vždy štandardným spôsobom. Údaje zákazníkov sú uložené v systéme, a tak už nie je potrebné ich zapisovať pri každej novej objednávke. Ďalšou výhodou je prehľadná evidencia záväzkov a úhrad zákazníkov, možnosť jednoducho vyhľadať informácie aj spätne. Takisto sa zefektívnila práca s prekladateľmi.

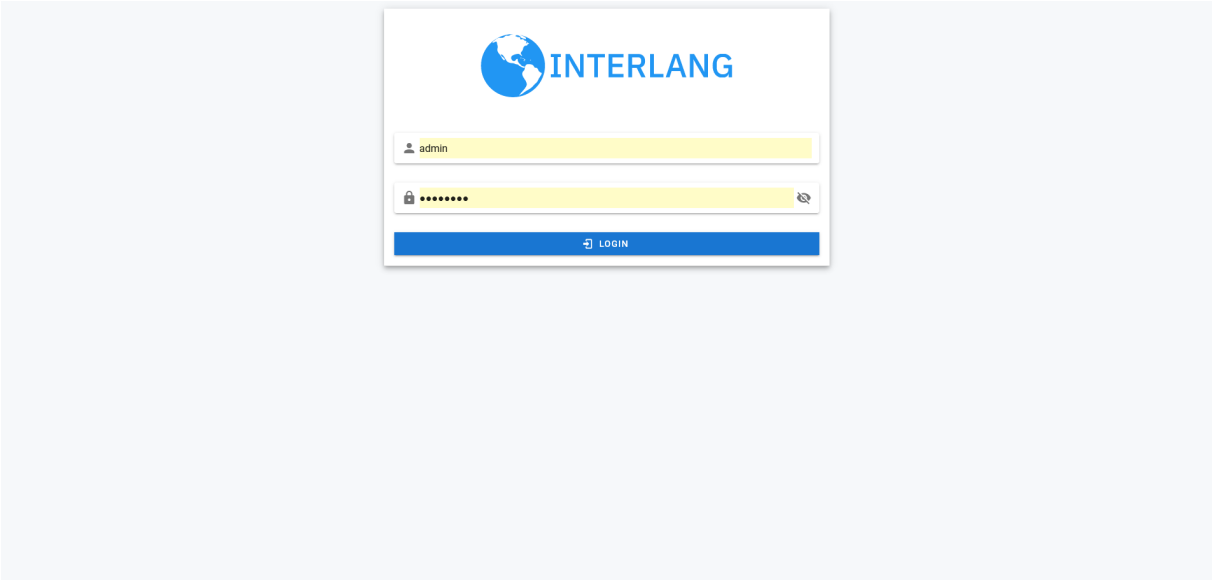
Celkovo sa vo veľkej miere znížila pravdepodobnosť chýb.

Všetky informácie sú uložené na jednom mieste. Perfektne sa vyriešil problém zdieľania informácii medzi zamestnancami, ktorý sa najviac prejavoval pri potrebe zastupovania jedného zamestnanca iným. Veľkou výhodou je aj možnosť vykonávať zálohovanie dát.

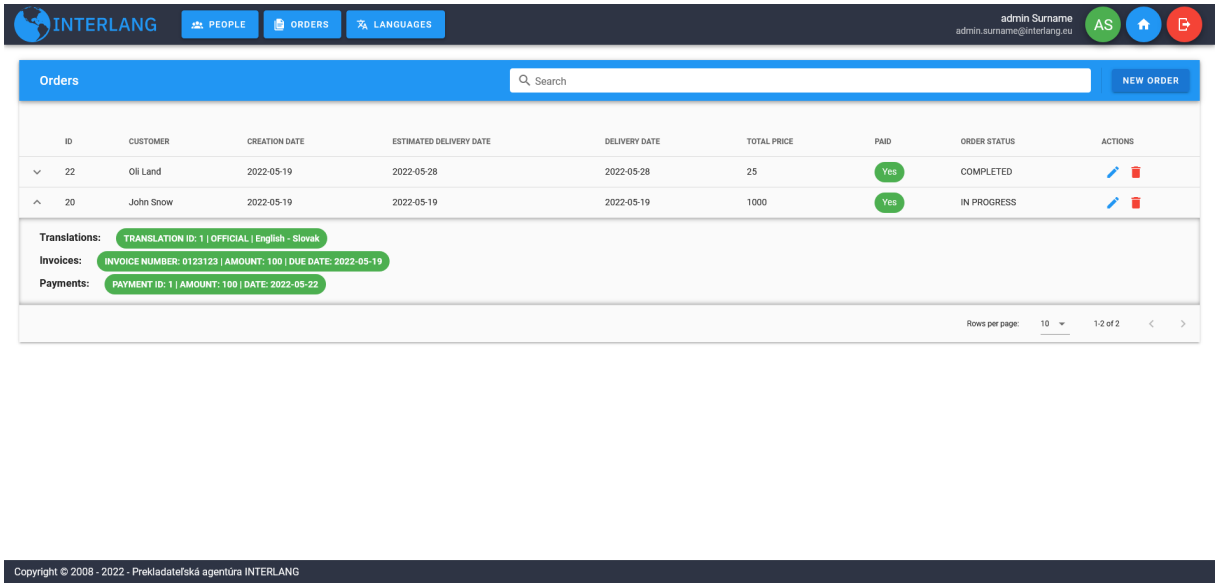
Po určitom čase používania nového informačného systému v testovacom režime, plánuje agentúra jeho zdokonalenie a rozšírenie. Akých vlastností a funkcií sa budú zmeny týkať, to sa upresní neskôr, na základe skúsenosti s novým systémom.

Dodatok A

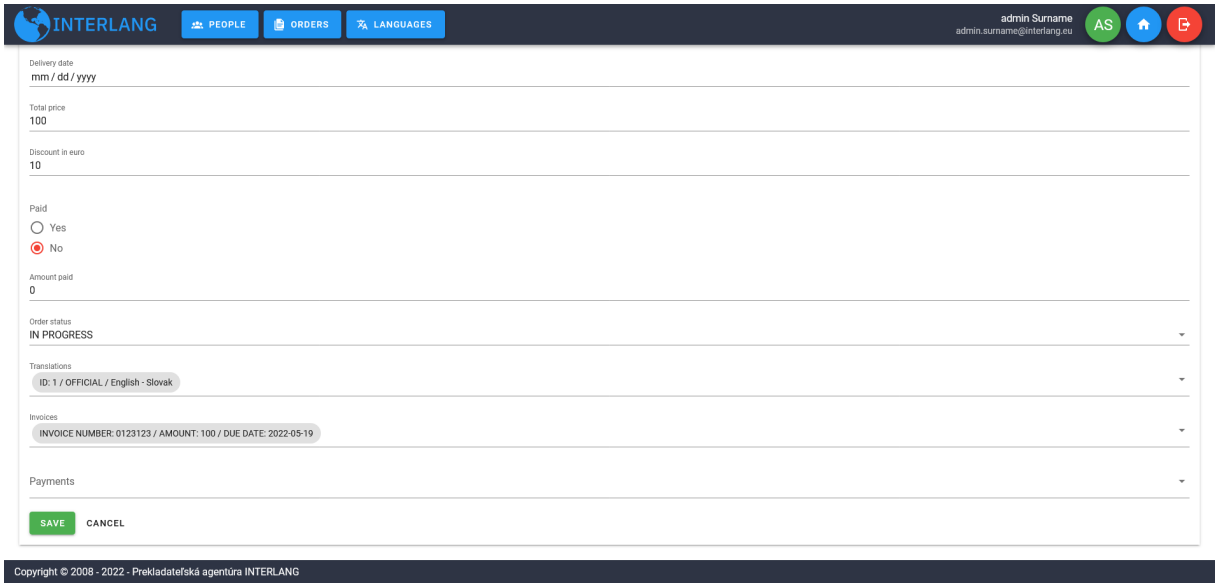
Snímky obrazoviek



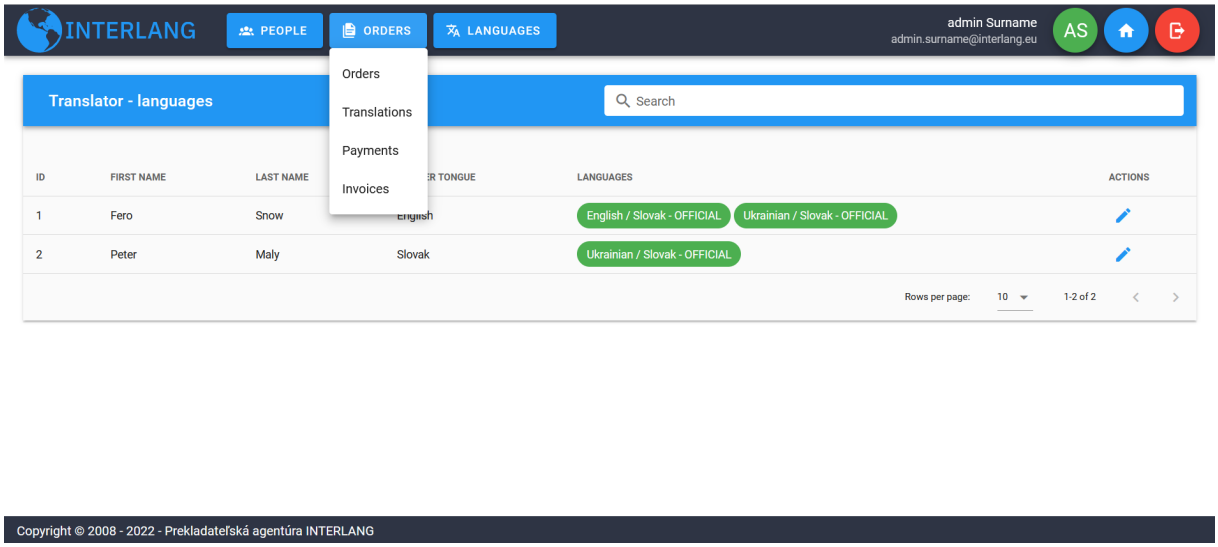
Obr. A.1: Prihlásovacia stránka



Obr. A.2: Stránka so zoznamom všetkých objednávok



Obr. A.3: Stránka pre vytvorenie novej objednávky



Obr. A.4: Stránka prekladateľov a jazykov

Bibliografia

[1] G. Piccoli a F. Pigni, *Information Systems for Managers*. Amsterdam, Netherlands: Amsterdam University Press, 2019.

[2] *eWorkOrders CMMS Pricing, Alternatives and More 2022*. URL: https://www.capterra.com/p/90092/eWorkOrders-CMMS/.

[3] *Maxpanda CMMS Pricing, Alternatives and More 2022*. URL: https://www.capterra.com/p/221240/Maxpanda-CMMS/.

[4] *UpKeep Pricing, Alternatives and More 2022*. URL: https://www.capterra.com/p/225407/UpKeep/.

[5] P. P. Publishing, *SWOT Analysis Notebook: STRENGTHS, WEAKNESSES, OPPORTUNITIES and THREATS Analysis for All-kind-of Projects - Journal for achieving greater results by 360 degree analysis of Office, Business and Home Tasks*. Independently published, 2021.

[6] *Porter's 5 Forces*, feb. 2020. URL: https://www.investopedia.com/terms/p/porter.asp.

[7] G. Blokdyk, *PEST analysis: Standard Requirements*. CreateSpace Independent Publishing Platform, 2018.

[8] *What is Flask Python - Python Tutorial*. URL: https://pythonbasics.org/what-is-flask-python/.

[9] *Welcome to Flask — Flask Documentation (2.1.x)*. URL: https://flask.palletsprojects.com/en/2.1.x/#.

[10] W. Semik, *Flask vs. Django: Which Python Framework Is Better for Your Web Development?*, nov. 2021. URL: https://www.stxnext.com/blog/flask-vs-django-comparison/.

[11] *Django introduction - Learn web development — MDN*, apr. 2022. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

[12] A. Joy, *11 Advantages of Django: Why You Should Use It*, okt. 2021. URL: https://pythonistaplanet.com/advantages-of-django/.

[13] D. Čápka, *Lekce 1 - Úvod do ASP.NET*. URL: https://www.itnetwork.cz/csharp/asp-net-core/zaklady/tutorial-uvod-do-asp-dot-net.

[14] T. Author, *Advantages and Disadvantages of ASP.NET — Software Developer India*, júl 2021. URL: https://www.software-developer-india.com/advantages-and-disadvantages-of-asp-net/.

[15] A. Lock, *ASP.NET Core in Action, Second Edition*. Manning, 2021.

[16] J. Máca, *Lekce 1 - Úvod do Node.js*. URL: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.

[17] T. Kaneriya, *Advantages and Disadvantages of Node.js : Why to Use Node.js?*, máj 2022. URL: https://www.simform.com/blog/nodejs-advantages-disadvantages/#section15.

[18] *Spring Framework Documentation*. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/index.html.

[19] *Java Spring Pros and Cons - Javatpoint*. URL: https://www.javatpoint.com/java-spring-pros-and-cons.

[20] Admin, *Advantages and Disadvantages of NoSQL Databases*, sept. 2021. URL: https://technologypoint.in/advantages-and-disadvantages-of-nosql-databases/.

[21] M. S. Gharajeh, *Security Issues and Privacy Challenges of NoSQL Databases*, júl 2017. URL: https://www.academia.edu/33823008/Security_Issues_and_Privacy_Challenges_of_NoSQL_Databases.

[22] *NoSQL databáze*. URL: https://www.kiv.zcu.cz/studies/predmety/db2/cviceni-nosql.html.

[23] A. Williams, *NoSQL database types explained: Key-value store*, aug. 2021. URL: https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Key-value-store.

[24] ——, *NoSQL database types explained: Column-oriented databases*, sept. 2021. URL: https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases.

[25] ——, *What is a Document Database?*, sept. 2021. URL: https://phoenixnap.com/kb/document-database.

[26] ——, *NoSQL database types explained: Graph*, dec. 2021. URL: https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Graph.

[27] M. Roomi a M. Roomi, *6 Advantages and Disadvantages of Relational Database — Limitations and Benefits of Relational Database*, apr. 2021. URL: https://www.hitechwhizz.com/2021/04/6-advantages-and-disadvantages-limitations-benefits-of-relational-database.html.

[28] B. Lutkevich a J. Biscobing, *relational database*, jún 2021. URL: https://www.techtarget.com/searchdatamanagement/definition/relational-database.

[29] Z. Akhtar, *Relational Database Benefits and Limitations (Advantages and Disadvantages)*, aug. 2021. URL: https://databasetown.com/relational-database-benefits-and-limitations/.

[30] S. Schlothauer, *Why You Should Migrate from SQL to NoSQL for Time Series Data*, okt. 2020. URL: https://jaxenter.com/nosql-time-series-data-172822.html.

[31] A. M. A. Mamun, *NoSQL Databases: When to Use NoSQL vs SQL*, jan. 2022. URL: https://www.serverwatch.com/guides/when-to-use-nosql/.

[32] *PostgreSQL vs. MySQL: What You Need to Know — Blog — Fivetran*. URL: https://www.fivetran.com/blog/postgresql-vs-mysql.

[33] K. Gupta, *MariaDB vs. MySQL vs. PostgreSQL In-Depth Comparison*, júl 2017. URL: https://www.freelancinggig.com/blog/2017/07/22/mariadb-vs-mysql-vs-postgresql-depth-comparison/.

[34] D. Rincon, *Oracle Database Licensing*, feb. 2022. URL: https://cdsiusa.com/hardware-and-licensing/oracle-database-products/.

[35] Hazelcast, *Compare Redis and •*, jan. 2020. URL: https://hazelcast.org/compare-with-redis/.

[36] S. Daityari, *Angular vs React vs Vue: Which Framework to Choose*, dec. 2021. URL: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

[37] *Angular*. URL: https://angular.io/guide/what-is-angular.

[38] D. Karczewski, *What Are The Best Frontend Frameworks To Use In 2022?*, máj 2022. URL: https://www.ideamotive.co/blog/best-frontend-frameworks.

[39] *Introduction — Vue.js*. URL: https://vuejs.org/guide/introduction.html.

[40] M. Richards, *Software Architecture Patterns*. URL: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html.

[41] Baeldung, *Layered Architecture*, nov. 2021. URL: https://www.baeldung.com/cs/layered-architecture.

[42] *What is Vuex? — Vuex*. URL: https://vuex.vuejs.org/.

[43] *Introduction — Vue Router*. URL: https://router.vuejs.org/introduction.html.

[44] *Why you should be using*. URL: https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started.

[45] *Getting Started — Axios Docs*. URL: https://axios-http.com/docs/intro.

[46] N/A, *Applying UML and Patterns - Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd, 05) by [Hardcover (2004)]*. Prentice Hal, Hardcover(2004), 2022.

[47] *What is JWT — JavaInUse*. URL: https://www.javainuse.com/spring/jwt.

[48] *REST API Documentation Tool — Swagger UI*. URL: https://swagger.io/tools/swagger-ui/.

[49] *Spring Boot - Interceptor*. URL: https://www.tutorialspoint.com/spring_boot/spring_boot_interceptor.htm.

[50] *What is Software Testing and How Does it Work? — IBM*. URL: https://www.ibm.com/topics/software-testing.

[51] T. Kaczanowski, *Practical Unit Testing with JUnit and Mockito*. Tomasz Kaczanowski, 2019.

[52] *Mockito framework site*. URL: https://site.mockito.org/.

[53] *Pairwise Testing*. URL: https://www.tutorialspoint.com/software_testing_dictionary/pairwise_testing.htm.