MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Portál práce

Bakalářská práce

VITALII BORTSOV

Brno, podzim 2023

MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Portál práce

Bakalářská práce

VITALII BORTSOV

Vedoucí práce: Mgr. Luděk Bártek, Ph.D.

Fakulta Informatiky

Brno, podzim 2023


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vitalii Bortsov

Vedoucí práce: Mgr. Luděk Bártek, Ph.D.

Poděkování

Rád bych vyjádřil své podekování Mgr. Luďkovi Bártkovi, Ph.D. za jeho věnovaný čas zpětnou vazbu během tvorby této bakalářské práce.

Chtěl bych také poděkovat svým kolegům v práci, Peteru Jančušovi a Dominiku Szalaiovi, kteří mi poskytli výjimečnou podporu a vedení při mé profesní cestě. S jejími cennými radami a zkušenostmi v oblasti vývoje webových aplikací jsem získal klíčové znalosti a dovednosti, které mi pomohly dosáhnout úspěchu při tvorbě této bakalářské práce. Rád bych jim vyjádřil svou hlubokou vděčnost za jejich trpělivost a ochotu sdílet své vědomosti a zkušenosti se mnou.

Shrnutí

Tato bakalářská práce se zaměřuje na návrh a implementaci webové aplikace představující portál práce. Hlavním cílem je vytvořit aplikaci, která umožňuje snadnou výměnu uživatelského rozhraní. Navrhovaná aplikace je implementována s využitím platformy Docker, což zajišťuje jednoduchou přenositelnost mezi různými operačními systémy.

Klíčová slova

Webová aplikace, portál práce, API, databáze, Java, Spring Boot, Docker

Obsah

Seznam obrázků

Úvod

Pracovní portály jsou webové aplikace, které umožňují nabízet a hledat volná pracovní místa online, čímž šetří čas jak společnostem, tak i zájemcům o práci. Nejpopulárnější představitelé portálů práce v České republice jsou Jobs.cz[1], Prace.cz[2] a Indeed[3][1].

Hlavním cílem této práce je navrhnout a implementovat webovou aplikaci Portál práce s důrazem na flexibilitu uživatelského rozhraní. Klíčovým prvkem je vytvoření dobře navrženého a zdokumentovaného API, což umožní snadnou výměnu uživatelského rozhraní. Dále se snažím zajistit univerzálnost aplikace prostřednictvím platformy Docker, což umožňuje její spuštění na různých operačních systémech nezávisle na jejich specifikách.

V první části mé práce zkoumám existující řešení a analyzuji poskytované služby. Následně poskytuji přehled různých typů aplikačních programových rozhraní a vyhodnocuji jejich vhodnost pro účely této práce. V další kapitole se věnuji analýze funkčních a nefunkčních požadavků, navrhuji diagramy případů užití, tříd a entitně-relační diagram (ERD). Popisuji také klíčové nástroje, které byly použity při implementaci. V závěrečné části práce prezentuji vytvořenou aplikaci a konkrétní případy užití definované v rámci analýzy. Práci zakončuji popisem procesu nasazení a uvádím několik návrhů na možná budoucí vylepšení.

---

1. Jobs.cz: `https://www.jobs.cz`
2. Prace.cz: `https://www.prace.cz`
3. Indeed: `https://www.indeed.com`

1 Portály práce

Tato kapitola se zaměřuje na analýzu problematiky spojené s pracovními portály, představení již existujících řešení a jejich služeb. Analýza a zkoumání této problematiky jsou klíčové při návrhu a implementaci konkrétních uživatelských scénářů a sestavení datového modelu pro aplikaci. Cílem této kapitoly je identifikace a porozumění klíčovým aspektům pracovních portálů a získání přehledu o jejich službách.

1.1 Služby pracovních portálů

Pracovní portály slouží především k tomu, aby zjednodušily procesy hledání zaměstnání a náboru nových zaměstnanců.

Uchazeči o zaměstnání prostřednictvím pracovních portálů mohou nastavit různé filtry při vyhledávání, které pomáhají zefektivnit proces hledání vhodné pracovní nabídky. Typicky se jedná o filtry jako lokalita, minimální platové požadavky, požadovaná úroveň vzdělání a počet hodin práce týdně.

Zaměstnavatelům a firmám inzerování na pracovním portálu může přinášet značné výhody, neboť tato forma propagace nabízí významnější dosah a cílovou skupinu tvoří potenciální uchazeči o zaměstnání[2].

Většina pracovních portálů slouží také k jednoduššímu propojení uchazeče o zaměstnání se zaměstnavateli, prostřednictvím notifikací se zvyšuje efektivita komunikace mezi oběma stranami. Tyto notifikace umožňují uchazečům rychle reagovat na nové pracovní nabídky, což zkracuje dobu, po kterou je pozice volná, a firmám poskytuje okamžitou zpětnou vazbu od potenciálních kandidátů. To vše přispívá k efektivnímu procesu hledání a nabízení pracovních příležitostí.

1.2 Jobs.cz

Doména jednoho z prvních českých pracovních portálů Jobs.cz byla zaregistrovaná v roce 1996 Liborem Malým ze společnosti LMC. Dodnes je jedním z nejpoužívanějších webů pro hledání práce v České republice[3].

Jobs.cz nabízí návštěvníkům svých webových stránek nejen přehled volných míst, ale i různé užitečné články, kalkulačky a generátor životopisů. Články většinou obsahují rady na to, jak postupovat při hledání zaměstnání, pohovoru nebo mimořádných pracovních situacích. Kalkulačky pomohou spočítat například čistou mzdu neboli výši dávky po dobu nemoci. Životopis lze vygenerovat jenom přihlášeným uživatelům bezplatně. Po vyplnění formuláře je vygenerován životopis a portál ho nabízí publikovat, aby uchazeče mohli najít a oslovit budoucí zaměstnavatelé[4].

1.3 Prace.cz

Prace.cz je také jedním z nejpoužívanějších pracovních portálů v České republice[3]. I přestože Prace.cz a Jobs.cz jsou součástí téže společnosti, tyto dva portály se výrazně odlišují svým zaměřením na specifické skupiny pracovníků[5].

Na Prace.cz je možné vyhledávat práci nejen podle regionu, ale také podle konkrétního města. Tento pracovní portál je vhodný pro lidi hledající práci v oblasti nižších kancelářských pozic, manuální práce nebo řemeslných oborů[5]. Tento portál cílí na ty, kdo mají praktické zkušenosti a dovednosti, ale nemusí mít vysokoškolské vzdělání[5].

Naopak, Jobs.cz je specializován na vysoko kvalifikované pozice, které obvykle vyžadují vysokoškolské vzdělání a další kvalifikace[5].

Podobně jako Jobs.cz poskytuje i Prace.cz sekci „poradna“, kde uchazeči mohou nalézt užitečné tipy a články.

1.4 Indeed

Indeed je jedním z nejnavštěvovanějších a nejoblíbenějších pracovních portálů na internetu. Tato platforma získala popularitu díky svému jednoduchému designu, snadnému použití a efektivním vyhledávacím nástrojům, což z ní činí oblíbenou jak mezi uchazeči o práci, tak mezi zaměstnavateli[6].

Zaměstnavatelé najdou na platformě Indeed mnoho atraktivních výhod. Díky rozumné cenové politice a širokému dosahu je Indeed vhodným nástrojem pro vyhledávání potenciálních kandidátů. Indeed

poskytuje užitečné nástroje pro náborové týmy a také umožňuje výběr uchazečů a plánování pohovorů[6].

Jedním z klíčových rysů Indeed je možnost rychlého a snadného zveřejnění pracovní pozice, a to i zdarma. Nicméně pro dosažení dostatečně velkého počtu uchazečů bude pravděpodobně třeba sponzorovat pracovní inzeráty na Indeed, což může být finančně nákladné[6].

Indeed nabízí také užitečné funkce, jako jsou dovednostní hodnocení, možnost pozvání konkrétních uchazečů k přihlášení a upozornění na pracovní nabídky pro uchazeče[6].

I přestože Indeed nabízí pracovní pozice z různých odvětví a úrovní, je obzvláště populární mezi uchazeči o práci na nižších úrovních a začátečnických (entry-level) pozicích[6].

2 Aplikační programové rozhraní

Aplikační programové rozhraní (dále jen API) je určitý souhrn pravidel umožňujících komunikaci mezi několika různými aplikacemi[7]. V této kapitole poskytuji přehled základních typů API a následně hodnotím, který z nich je vhodnější pro účely této práce.

2.1 Webová API

Pro webová API se typicky používají protokoly HTTP nebo HTTPS[1]. Komunikací se zde rozumí přenos a zpracování dat mezi softwarovými systémy[7, 8].

Mnoho služeb nabízí svoje API veřejně. Tato veřejná API slouží k rozšíření funkcionality aplikace pomocí již naprogramovaného řešení[9].

Například Google poskytuje API k většině svých služeb. Díky tomu lze do vlastního softwaru integrovat třeba přihlášení pomocí Google[2][10]. Toto umožní vývojářům využít existující řešení bez nutnosti vytvářet vlastní systém přihlášení.

2.1.1 Koncové body

Koncový bod[3] (anglicky „endpoint“) je digitální místo, kde API přijímá požadavky týkající se konkrétního prostředku na svém serveru. V API je koncový bod obvykle reprezentován pomocí jednotné adresy zdroje (URL), jež ukazuje, kde se prostředek na serveru nachází.

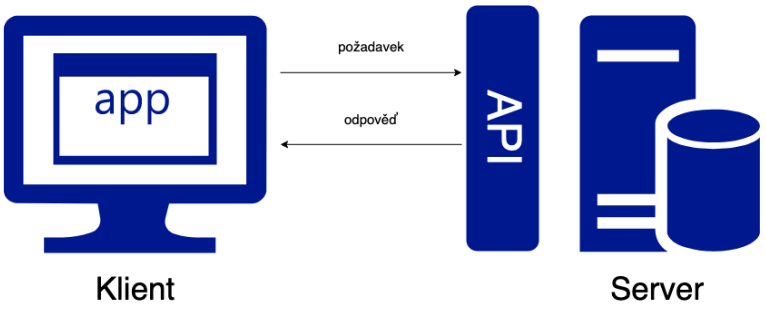
2.1.2 Klient a server

Server je aplikace, která poskytuje určité služby nebo funkcionality. Klient je druhá aplikace vystupující v roli uživatele nebo jiného systému, který využívá služby nebo data poskytovaná serverem. Klient

---

1. protokol HTTP/HTTPS: `https://www.ssls.cz/https.html`
2. Using OAuth 2.0 to Access Google APIs: `https://developers.google.com/identity/protocols/oauth2`
3. `https://blog.hubspot.com/website/api-endpoint`

posílá požadavky na koncové body serveru s cílem získat přístup k určitým datům nebo provést operace. Server na tyto požadavky reaguje a klient následně zpracovává odpovědi.



Obrázek 2.1: Komunikace mezi klientem a serverem

2.2 Typy API

V této části se věnuji popisu různých typů API, konkrétně REST[4], SOAP[5] a GraphQL. Na konci zhodnocuji, který z nich je nejvhodnější pro účely této práce.

2.2.1 REST

RESTful API (Representational State Transfer API) je rozhraní pro programování aplikací, které pro manipulaci s daty využívá HTTP metody (GET, PUT, POST, DELETE atd.). REST API je založeno na konceptu „zdrojů“ (anglicky „resources“), které jsou identifikovány pomocí URL. Každý zdroj má vlastní URL a lze s ním provádět různé operace pomocí HTTP metod. Pro přenos dat v REST je běžně používán formát JSON[6][11].

Každá HTTP metoda v REST API je využívána k provádění konkrétní operace:

- **GET (čtení)**: používá se k získání dat ze zdroje.

4. Representational state transfer
5. Simple Object Access Protocol
6. JavaScript Object Notation

- **PUT** (**aktualizace**): slouží k aktualizaci existujícího zdroje.
- **POST** (**vytvoření**): používá se k vytváření nového zdroje.
- **DELETE** (**smazání**): slouží k odstranění zdroje ze serveru.
- **PATCH** (**částečné úpravy**): slouží k částečné aktualizaci zdroje. Může být použita k modifikaci existujícího zdroje bez nutnosti odeslat celý zdroj.

REST API je založeno na principech jednoduchosti, konzistence a stálosti. Každý požadavek klienta na server by měl být nezávislý a server by měl reagovat na požadavky klienta bez uchovávání stavu mezi jednotlivými požadavky[11].

Tato architektura umožňuje efektivní a jednoduchou komunikaci mezi klientem a serverem a je často používána pro vývoj webových aplikací a webových služeb[11].

2.2.2 SOAP

SOAP je webový komunikační protokol, který byl navržen společností Microsoft v roce 1998. Používá se pro poskytování webových služeb a přenos dat pomocí protokolů HTTP/HTTPS, ale jeho využití není omezeno pouze na ně. SOAP podporuje pouze formát dat ve formě XML[7] a pevně dodržuje předem stanovené standardy, jako je struktura zpráv[12].

Požadavky a odpovědi v protokolu SOAP jsou ve formě obalovaných zpráv, z nichž každá obsahuje čtyři specifické části. Tyto části jsou označeny jako obálka, hlavička, tělo a chyba. Každá z nich má svou určitou úlohu a funkci při komunikaci v rámci SOAP protokolu[12].

Obálka (anglicky „envelope“) je jádrem každé zprávy a je nezbytným prvkem, který zprávu začíná a ukončuje svými značkami, tím ji obaluje, odtud název[12].

Hlavička je část, která definuje specifické charakteristiky a další požadavky zprávy jako například autentizaci[12].

Tělo obsahuje požadavek nebo odpověď[12].

7. Extensible Markup Language

Chyba slouží k detailnímu zaznamenání všech informací o případných chybách, které by mohly v průběhu komunikace v rámci API vzniknout[12].

2.2.3 GraphQL

Na rozdíl od tradičních API, jako jsou REST nebo SOAP, GraphQL umožňuje klientu přesně specifikovat, jaká data potřebuje, a získává tak pouze ty konkrétní informace, které ho zajímají. To znamená, že GraphQL API nabízí klientům větší flexibilitu a umožňuje snadněji získat potřebná data bez nadměrného načítání nebo přenášení nepotřebných informací. Pro získání dat se v GraphQL využívá dotaz a pro provádění změn se používá mutace[13].

Dotazy jsou strukturovány podle toho, jaká data jsou potřebná, a výsledek odpovídá struktuře dotazu. Dotazy umožňují získat přesně definované informace bez nadbytečných dat[13].

Mutace v GraphQL mají podobnou strukturu jako dotazy, ale slouží k provádění změn na serveru, jako jsou vytváření, aktualizace nebo mazání dat[13].

GraphQL poskytuje jednotný bod vstupu pro dotazy a mutace, což znamená, že vývojáři nemusí pracovat s různými koncovými body pro různé požadavky[13].

GraphQL je široce využíván v mnoha aplikacích a systémech, včetně populárních online platforem a služeb, jako jsou Facebook, GitHub, Shopify a Twitter[13].

2.3 Výběr

Pro výslednou aplikaci bylo zvoleno RESTful API, a to z důvodu jeho efektivity, flexibility a snadné implementace. Tyto charakteristiky jsou klíčové pro úspěšnou realizaci CRUD operací, které má výsledná aplikace poskytovat.

GraphQL je sice nejrychlejším typem ze zmíněných[14], ale je vhodný pro rozsáhlejší a náročnější aplikace, což není případ aplikace, která je cílem této práce.

SOAP je pomalejší, méně flexibilní a komplexnější. I když SOAP může být teoreticky použit v různých síťových protokolech než HTTP

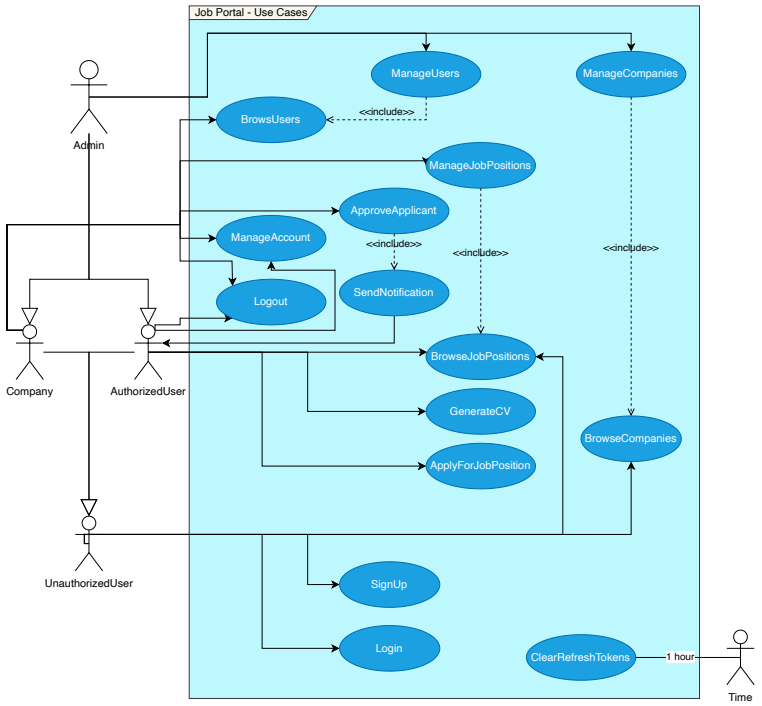
a HTTPS, pro účely této práce to není relevantní, neboť výsledná webová aplikace využívá zmiňované protokoly HTTP nebo HTTPS.

3 Analýza a návrh

Před implementací aplikace Portál práce byla provedena analýza, v jejímž rámci byly specifikovány jak funkční, tak nefunkční požadavky na vyvíjený systém. Následně byl vytvořen návrh databáze a diagram tříd.

3.1 Funkční požadavky

Funkční požadavky se zaměřují na konkrétní funkcionality a operace, které systém poskytuje, a popisují očekávané chování systému ve vztahu k uživatelským požadavkům. Příklady mohou zahrnovat uživatelské operace, výpočty, zpracování dat nebo interakce s externími systémy.



Obrázek 3.1: Diagram případů užití

3.1.1 Uživatelské role

Uživatelské role představují sady oprávnění přiřazené jednotlivým uživatelům v aplikaci. Každý uživatel má v dané aplikaci právě jednu roli, která definuje soubor oprávnění pro jeho interakci se systémem. V aplikaci existují tři typy účtů: běžný, firemní a administrátorský.

Uživatelé s běžným účtem v aplikaci mají přiřazenou roli, která je v kódu označena jako `REGULAR_USER`. Tito uživatelé jsou potenciální uchazeči o zaměstnání, a proto mají sadu oprávnění, která jim umožňují jakoukoliv interakci se systémem spojenou s hledáním práce a účastí v procesu zaměstnávání. V rámci správy svého účtu mohou uživatelé s firemním účtem přidávat své pracovní zkušenosti. Na základě těchto údajů si pak mohou generovat svůj životopis.

Uživatelé s firemním účtem jsou označeni jako `COMPANY` a mají odlišný rozsah funkcí v systému. Jsou to firmy a společnosti, kterým portál umožňuje provádět zjednodušený proces náboru a spravovat svoje pracovní místa.

Nepřihlášeným uživatelům systém umožňuje procházet nabídku pracovních pozic a registrovaných společností bez možnosti podávat žádosti o zaměstnání. Všichni přihlášení uživatelé mohou spravovat svůj účet, což zahrnuje možnost provádět úpravy, odstranění účtu a změnu nastavení. Administrátoři mají plný přístup ke všem funkcionalitám systému.

3.1.2 Registrace

Aplikace umožňuje registraci do systému pomocí e-mailové adresy, přičemž jedna adresa může být použita pouze s jedním účtem. Při registraci je možné vyplnit jméno a příjmení. Pokud uživatel chce registrovat firmu, může vyplnit také základní informace o firmě. Vyplněné informace lze následně editovat s výjimkou e-mailové adresy.

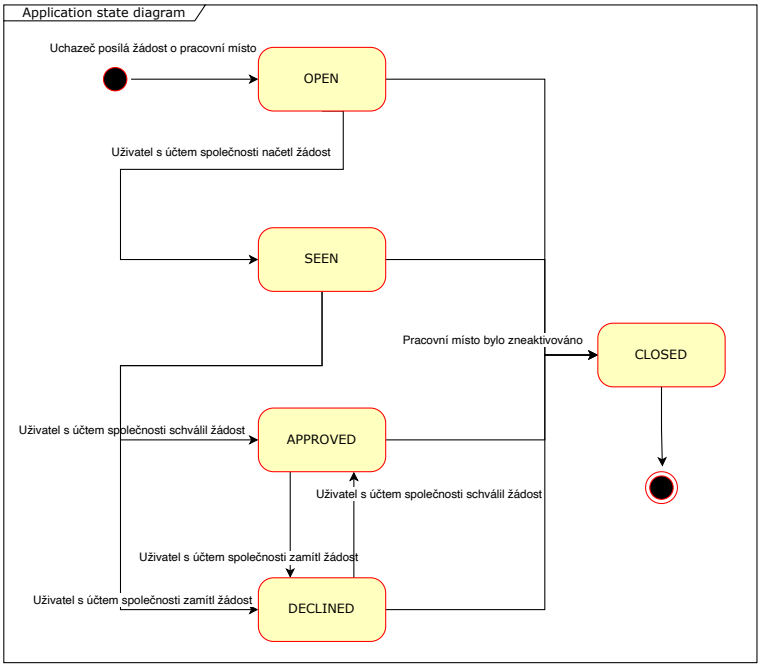
3.1.3 Společnosti a pracovní nabídky

V rámci portálu je možné prohledávat registrované společnosti, které nabízejí pracovní příležitosti. Uživatelé mohou filtrovat a hledat pracovní pozice a společnosti pomocí různých kritérií a dotazů. Ti, kteří aktivně hledají zaměstnání, mohou přidávat zajímavé pracovní pozice do svého seznamu pro snadný přístup. Pokud se rozhodnou ucházet

se o konkrétní pracovní pozici, mohou podat žádost, která se poté zobrazí u zaměstnavatele. Zaměstnavatel má možnost dočasně zneaktivnit zveřejněné pracovní místo, což omezuje přijímání nových žádostí. Když se rozhodne znovu zveřejnit pozici, může ji upravit a opět aktivovat.

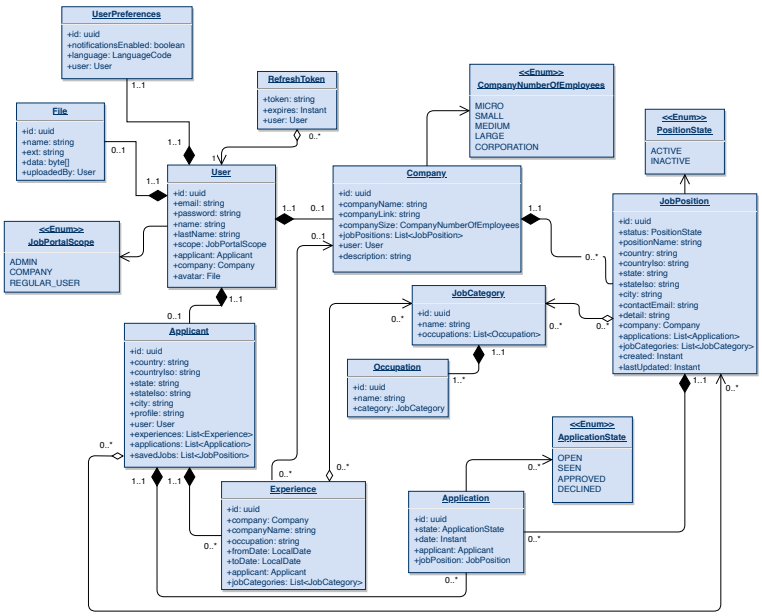
3.1.4 Žádosti o zaměstnání

Žádost o zaměstnání může mít 5 různých stavů: otevřená (`OPEN`), zobrazená (`SEEN`), schválená (`APPROVED`), odmítnutá (`DECLINED`) a uzavřená (`CLOSED`).



Obrázek 3.2: Stavový diagram žádosti o zaměstnání

Pokud má uživatel zapnutá oznámení, dostává e-maily o schválení a odmítnutí podaných žádostí. Uživatel s účtem společnosti může procházet podané žádosti na jeho pracovní nabídky a má možnost žádost zamítnout nebo schválit.



Obrázek 3.3: Diagram tříd

3.2 Nefunkční požadavky

Nefunkční požadavky specifikují další nezbytné vlastnosti nebo omezující podmínky vyvíjeného systému. Mezi nefunkční požadavky patří například jeho výkonnost, spolehlivost nebo bezpečnost systému.

Výsledná aplikace musí zajistit vysokou úroveň bezpečnosti, chránit data uživatelů a zabránit neoprávněnému přístupu.

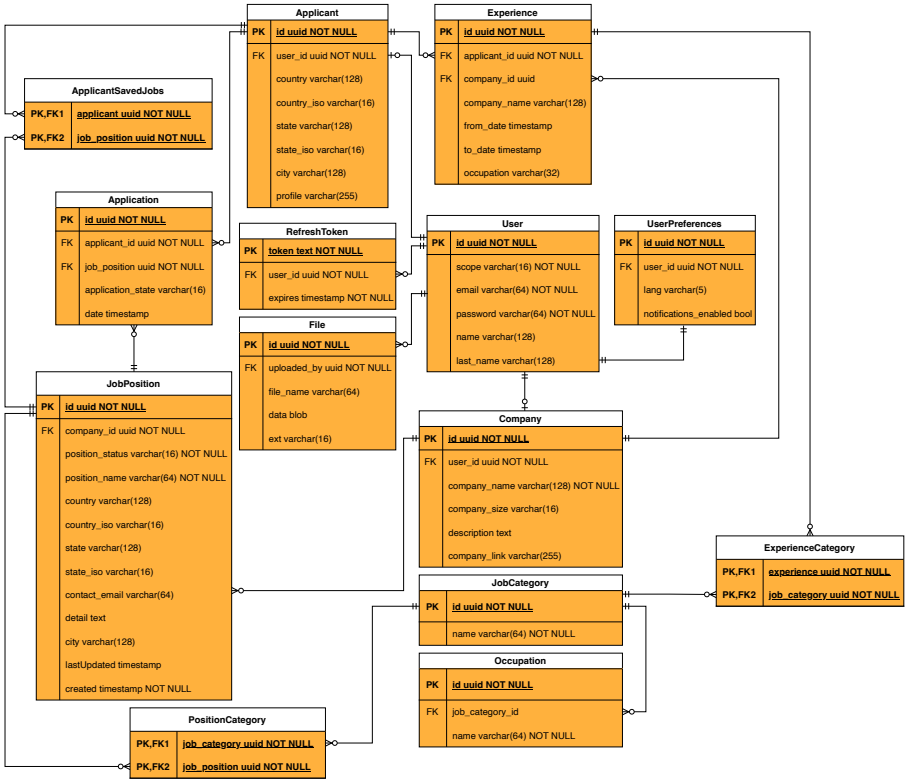
Aplikace by měla být navržena tak, aby byla snadno nasaditelná a spustitelná v různých prostředích pomocí kontejnerizační platformy Docker.

Výsledná aplikace by měla podporovat lokalizaci pro různé jazyky a případně by měla být rozšiřitelná o další jazyky.

Aplikace musí poskytovat dobře navržené a zdokumentované API, které umožní snadnou integraci a vytváření dalších uživatelských rozhraní.

3.3 Návrh databáze

Aplikace vyžaduje efektivní ukládání a správu velkého množství dat. Pro řešení této potřeby byl vytvořen entitně-relační diagram, který slouží k detailnímu popisu struktury databáze a vztahů mezi jednotlivými entitami.



Obrázek 3.4: Entitně-relační diagram

V navrhované databázi jsou definovány vztahy mezi entitami s ohledem na potřeby uchazečů a pracovních nabídek.

Každá pracovní nabídka může přilákat více uchazečů, což je reflektováno v relaci "**1:N**"(jedna ku více) mezi entitami Pracovní pozice a Žádost. To znamená, že jedna pracovní nabídka může mít více příchozích žádostí od různých uchazečů.

Naopak každý uchazeč může podat žádost o účast ve více výběrových řízeních, což je reprezentováno vztahem "**1:N**"(jedna ku více) mezi entitami Uchazeč a Žádost. Tímto způsobem je umožněno, aby jeden uchazeč podával žádosti o více pracovních pozic, přičemž každá z těchto žádostí se vztahuje pouze k jedné konkrétní pracovní nabídce.

4 Použité technologie

Při vývoji webových aplikací existuje nepřeberné množství technologií, které mohou být využity. Klíčem k úspěšnému projektu je výběr těch správných technologií, které nejen splňují požadavky, ale také umožňují dosáhnout optimálního výsledku. Tato kapitola se zaměřuje na představení technologií, které byly zvoleny pro vývoj webové aplikace Portál práce.

4.1 Programovací jazyky a aplikační rámce

V této sekci jsou představeny programovací jazyky a aplikační rámce neboli frameworky[1], které byly zvoleny pro tvorbu aplikace.

4.1.1 Java

Java je široce používaný programovací jazyk pro tvorbu webových aplikací. Java je víceplatformový, objektově orientovaný a zaměřený na síťové technologie. Jedná se o rychlý, bezpečný a spolehlivý programovací jazyk vhodný pro tvorbu mobilních aplikací, podnikového softwaru, aplikací zpracovávajících velká data a technologií na straně serveru[15].

Programovací jazyk Java se vyznačuje svou jednoduchostí, což umožňuje programátorům začít psát kód bez potřeby rozsáhlého školení[16].

Java je od svého základu koncipována jako objektově orientovaný jazyk. Potřeby distribuovaných klient-server systémů dokonale korespondují s principy objektově orientovaného softwaru, zejména pokud jde o zapouzdření a předávání zpráv. Díky této objektové povaze lze vývoj aplikací v Javě provádět efektivně a účinně[16].

Dalším důležitým aspektem Java je možnost programátorů přistupovat k již existujícím knihovnám ověřených objektů. Tyto knihovny nabízejí širokou škálu funkcí od základních datových typů přes rozhraní I/O (vstupů/výstupů) až po grafické uživatelské rozhraní. Na-

1. Rozsáhlá sada vývojářských nástrojů, která pomáhá a určuje, jak má programátor aplikaci vyvíjet.

víc je možné tyto knihovny rozšiřovat, což umožňuje vytvářet novou funkcionalitu[16].

Programovací jazyk Java je navržen pro tvorbu vysoce spolehlivého softwaru. Poskytuje rozsáhlé kontroly v době kompilace, následované druhou úrovní kontrol za běhu[16].

Správa paměti v Javě je mimořádně jednoduchá: objekty se vytvářejí pomocí operátoru `new`. Zde neexistují žádné explicitně definované datové typy ukazatelů ani aritmetika ukazatelů. Tento jednoduchý model správy paměti eliminuje celou řadu chyb, které programátory tíží při používání jazyků jako C a C++[16].

Java byla koncipována s ohledem na její použití v distribuovaném prostředí, což znamená, že bezpečnost má nejvyšší prioritu. Díky integrovaným bezpečnostním funkcím, které jsou součástí jazyka a systému, Java umožňuje vytvářet aplikace, které jsou odolné proti externím hrozbám. Aplikace napsané v programovacím jazyce Java jsou v síťovém prostředí chráněny před neoprávněným vniknutím kódu, který by se pokoušel proniknout za pozadí a vytvářet viry nebo pronikat do souborových systémů[16].

Java byla koncipována s ohledem na její použití v heterogenních síťových prostředích. V takových prostředích musí být aplikace schopny provádět na různých hardwarových architekturách a operovat na různých operačních systémech, přičemž musí spolupracovat s různými programovacími jazykovými rozhraními. Aby se vyrovnala s tímto rozmanitým prostředím, kompilátor generuje bytekódy, což je architektonicky neutrální formát, určený pro efektivní přenos kódu mezi různými hardwarovými a softwarovými platformami[16].

Architektonická neutralita je pouze jedním aspektem skutečně přenosného systému. Java jde ještě dál tím, že přesně definuje základní jazykové vlastnosti, jako jsou velikosti datových typů a chování aritmetických operátorů. To znamená, že programy budou na všech platformách jednotné a nedojde k nesrovnalostem v datových typech napříč hardwarovými a softwarovými architekturami[16].

4.1.2 TypeScript

TypeScript byl navržen tak, aby uspokojil potřeby programátorů pracujících s JavaScriptem, kteří vyvíjejí a udržují složité JavaScriptové aplikace, zejména webové aplikace. TypeScript usnadňuje definici

rozhraní mezi softwarovými komponentami a poskytuje vhled do chování existujících knihoven JavaScriptu[17].

Jednou z výrazných vlastností TypeScriptu je jeho nepovinný typový systém, který umožňuje vývojářům v jazyku JavaScript využívat výkonné nástroje a postupy pro programování, jako jsou statické kontroly, symbolická navigace, automatické dokončení kódu a refaktorizace. TypeScript je v podstatě syntaktické vylepšení JavaScriptu a je s ním plně kompatibilní. Jeho syntaxe tvoří nadmnožinu ECMAScriptu 5 (ES5), což znamená, že každý JavaScriptový program může být také považován za TypeScriptový program[17].

Kompilátor TypeScriptu provádí především lokální transformace nad TypeScriptovými programy, aniž by přeuspořádal proměnné deklarované v TypeScriptu. Tím se zajistí, že výstupní kód v JavaScriptu těsně odpovídá vstupnímu kódu v TypeScriptu. Jména proměnných zůstávají nezměněna, což usnadňuje přímé ladění generovaného JavaScriptu. TypeScript volitelně poskytuje zdrojové mapy, což umožňuje ladění na úrovni zdroje[17].

V syntaxi TypeScriptu existuje několik prvků navržených pro ECMAScript 6 (ES6), například třídy a moduly. Třídy umožňují programátorům vyjádřit běžné objektově orientované vzory standardizovaným způsobem, což zlepšuje čitelnost kódu a jeho schopnost spolupracovat s jiným kódem, zejména pokud se jedná o dědičnost. Moduly umožňují organizovat kód do samostatných částí a zároveň eliminovat problémy s konflikty v pojmenování. Kompilátor TypeScriptu nabízí různé možnosti generování kódu modulů, což podporuje jak statické načítání modulů, tak dynamické načítání jejich obsahu[17].

TypeScript přináší také systém nepovinných typových anotací, jež jsou začleněny přímo do syntaxe jazyka, což usnadňuje čtení kódu a snižuje náklady na údržbu synchronizace mezi typovými anotacemi a odpovídajícími proměnnými[17].

4.1.3 Spring Boot

Spring Boot je aplikační rámec s otevřeným zdrojovým kódem pro vývoj aplikací v jazyce Java. Jedná se o nástavbu rámce Spring, která přináší mnoho vylepšení a standardizovaných konvencí, což umožňuje

vytvářet rychle spustitelné, produkčně nasaditelné aplikace s minimálním úsilím v oblasti konfigurace[18].

Spring Boot je založen na konceptech Inversion of Control (obrácené řízení) a Dependency Injection (vkládání závislostí), což jsou klíčové principy rámce Spring[18].

- **Inversion of Control (IoC)**: Spring Boot implementuje princip IoC, což znamená, že ovládání toku programu je předáno rámci (v tomto případě Spring Boot) a není pevně zakořeněno v kódu aplikace. Toto umožňuje vytvářet aplikace, které jsou více modulární a snáze testovatelné. Základními stavebními kameny aplikace jsou tzv. „beany“, což jsou různé komponenty a služby, které tvoří funkcionalitu webové aplikace. Spring Boot využívá kontejner IoC k řízení životního cyklu beanů (komponent, služeb) a jejich závislostí[18].

- **Dependency Injection (DI)**: Spring Boot také používá princip Dependency Injection, což je způsob, jakým jsou komponenty aplikace vkládány do dalších komponent. Tímto způsobem lze snadno spravovat závislosti mezi komponentami a aplikace se stává mnohem flexibilnější a snáze rozšiřitelnou. Spring Boot automaticky provádí DI na základě konfigurace beanů a zjednodušuje tak vytváření komplexních grafů závislostí[18].

4.1.4 Angular

Angular je robustní vývojová platforma postavená na TypeScriptu. Nabízí rámec založený na komponentách pro tvorbu škálovatelných webových aplikací, komplexní soubor dobře integrovaných knihoven pro funkce jako routování, správa formulářů a komunikace mezi klientem a serverem. Angular je navržen tak, aby byl schopen zvládnout projekty různých velikostí od jednoho vývojáře až po podnikové projekty. Kromě toho má komunita této platformy více než 1,7 milionu vývojářů, autorů knihoven a tvůrců obsahu, což zaručuje rozmanitý a podpůrný ekosystém[19].

Komponenty jsou základními stavebními bloky v Angular aplikaci. Komponent zahrnuje třídu napsanou v TypeScriptu ozdobenou deko-

ratérem `@Component()`, HTML[2] šablonu a styly (CSS[3]). Komponenty nabízejí silnou izolaci, intuitivní strukturu aplikace a jednoduché testování jednotlivých částí. HTML šablony definují, jak komponenty renderují obsah, a mohou být buď vložené přímo do komponentů, nebo uloženy v externích souborech. Angular rozšiřuje HTML syntaxi pro dynamické vazby, jako jsou zobrazení dynamického textu a vazby na vlastnosti pro nastavení hodnot a atributů HTML prvků. Posluchači událostí mohou být snadno definovány v šablonách a umožní aplikaci reagovat na akce uživatele[19].

Zásadní koncept v Angular je Dependency Injection. Tento koncept umožňuje deklarovat závislosti tříd napsaných v TypeScriptu a zajišťuje, že se o jejich instanciování nemusí starat programátor. Stejně jako v případě rámce Spring Boot i v Angular se Dependency Injection stává důležitým nástrojem, který zvyšuje testovatelnost a flexibilitu kódu[19].

4.2 Správa dat

K správě dat v aplikaci byly využity nástroje PostgreSQL a Liquibase. Volba databáze PostgreSQL vychází z důrazu na integritu, výkon a bezpečnost dat v celém systému. PostgreSQL a Liquibase hrají klíčovou roli v zajištění stability a konzistence dat v aplikaci a umožňují efektivní správu změn v průběhu vývoje a provozu aplikace.

4.2.1 PostgreSQL

PostgreSQL je výkonný objektově-relační databázový systém s otevřeným zdrojovým kódem. Hlavními výhodami PostgreSQL jsou jeho ověřená architektura, spolehlivost, zajištění integrity dat a rozsáhlá nabídka funkcí. Je to multiplatformní systém, který podporuje všechny hlavní operační systémy[20].

4.2.2 Migrace dat (Liquibase)

Liquibase je nástroj pro správu a kontrolu verzí databázových schémat. Používá se zejména při vývoji softwaru a správě databází, aby se zajiš-

2. The HyperText Markup Language (hypertextový značkovací jazyk)
3. Cascading Style Sheets (kaskádové styly)

ťovala konzistence a sledovatelnost změn v databázovém schématu během vývoje aplikace[21].

Liquibase umožňuje definovat databázové změny pomocí kódu, což usnadňuje verzování těchto změn a spolupráci mezi vývojáři. Změny v databázovém schématu se definují jako XML, YAML nebo SQL soubory[21].

Liquibase také umožňuje automatické nasazení změn na cílovou databázi, což zjednodušuje proces aktualizace databázového schématu v různých prostředích. Tento nástroj je nezávislý na konkrétním databázovém systému a podporuje mnoho populárních databází, včetně PostgreSQL, MySQL, Oracle, SQL Server a dalších[21].

Celkově představuje Liquibase komplexní řešení pro řízení změn v databázi, což vývojářům i správcům databází usnadňuje udržování konzistence a sledování verzí databázového schématu.

4.3 Autentizace

Webová aplikace Portál práce zahrnuje případy užití, které vyžadují specifická oprávnění. Z tohoto důvodu je nezbytné zavést autentizaci. Autentizace je klíčový proces, který zajišťuje ověření identity uživatele v systému a jeho oprávnění k provádění určitých akcí.

Pro autentizaci ve webových aplikacích jsou využívány různé protokoly, jako jsou například JSON Web Token, OAuth 2.0 a OpenID. Pro OAuth 2.0 a OpenID se často využívají přihlašovací služby třetích stran, což usnadňuje implementaci tím, že není nutné vyvíjet vlastní přihlašovací službu. I přestože tyto protokoly mohou být použity pro autentizaci, byla zvolena technologie JSON Web Token (JWT). Tato volba je zdůvodněna tím, že aplikace by měla disponovat vlastní přihlašovací službou, což umožňuje úplnou kontrolu nad procesem autentizace uživatelů.

4.3.1 JSON Web Token

JWT poskytuje způsob, jak uživatele ověřit a vytvořit zabezpečený způsob přenosu informací o autentizaci. Tato volba zajišťuje, že pouze oprávnění uživatelé mají přístup k určitým funkcím a datům.

V rámci aplikace je autentizace provedena za použití zašifrovaného JSON objektu, který slouží jako autentizační žeton. K dosažení této funkcionality byl využit pár RSA klíčů pro šifrování a dešifrování dat v rámci procesu autentizace.

4.4 Vyhledávání

Vyhledávání v aplikaci Portál práce je klíčovou funkcí a tvoří významnou část serverové funkcionality. Aplikace poskytuje dvě hlavní metody vyhledávání: fulltextové vyhledávání založené na klíčových slovech a filtrované vyhledávání na základě různých kritérií. Tato sekce se zaměřuje na technologie a nástroje, které byly použity pro implementaci těchto vyhledávacích funkcí.

4.4.1 Hibernate Search

Hibernate Search provádí indexaci doménového modelu pomocí několika anotací, zajistí synchronizaci mezi databází a indexem a umožňuje přímo pracovat se standardními spravovanými objekty v rámci volných textových dotazů[22].

Hibernate Search využívá k implementaci fulltextového vyhledávání buď Apache Lucene nebo Elasticsearch. Jelikož Elasticsearch interně využívá Lucene, sdílí mnoho charakteristik a obecný přístup k fulltextovému vyhledávání[22].

Tyto vyhledávací nástroje jsou založeny na konceptu invertovaných indexů: slovníku, kde klíčem je klíčové slovo nalezené v dokumentu, a hodnotou je seznam identifikátorů každého dokumentu obsahujícího toto klíčové slovo[22].

Vyhledávání dokumentů, kdy jsou všechny indexovány, probíhá ve třech krocích: extrahování klíčových slov z dotazu, hledání těchto klíčových slov v indexu k nalezení odpovídajících dokumentů a agregace výsledků vyhledávání pro vytvoření seznamu odpovídajících dokumentů[22].

4.4.2 Textové analyzátory

Extrahování klíčových slov obvykle probíhá pomocí analyzátoru. Analyzátor provádí textovou analýzu, která zahrnuje několik kroků.

Nejdříve je vstupní text rozdělen na klíčová slova na základě určených pravidel a věty jsou rozděleny na jednotlivá slova a interpunkci[23].

Následuje fáze filtrace, kdy jsou aplikovány různé filtry. To může zahrnovat odstranění předložek a spojek, převedení slov na malá písmena a odstranění diakritiky[23].

Posledním krokem je lemmatizace, kdy jsou slova převedena do svého základního tvaru. Tím se zajišťuje, že různé tvary slov budou správně vyhledány[23].

Nakonec je výsledkem tohoto procesu seznam klíčových slov, který se použije pro vyhledávání ve fulltextovém indexu[23].

4.5 Swagger

Swagger[4] poskytuje možnost detailně popsat strukturu API tak, aby byla strojově čitelná. V Javě lze popis API snadno vytvořit pomocí anotací. Tento popis rozhraní umožňuje automatickou generaci přehledné a interaktivní dokumentace. Kromě toho Swagger umožňuje automatické generování knihovny klientů pro dané API v různých programovacích jazycích.

4.6 Docker

Docker je kontejnerová virtualizační platforma s otevřeným zdrojovým kódem, která umožňuje izolovaný provoz a správu aplikací a jejich závislostí[24].

4.6.1 Obrazy

Docker obrazy jsou základním stavebním kámenem pro vytváření kontejnerů. Docker obraz funguje jako šablona, ze které lze vytvářet běžící kontejnery. Obraz obsahuje vše, co je potřeba pro běh aplikace, a izoluje ji od hostitelského systému, což zajišťuje konzistenci chování aplikace bez ohledu na hostitelský systém[24].

4. `https://swagger.io`

4.6.2 Kontejnery

Docker umožňuje balení a spuštění aplikace v kontejnerech. Kontejner je spustitelnou instancí obrazu. Kontejner je definován svým obrázkem a konfiguračními možnostmi, které se mu předávají při jeho vytvoření nebo spuštění[24].

4.6.3 Dockerfile

Dockerfile je soubor obsahující instrukce a konfiguraci potřebné k vytvoření Docker obrázku. Tyto instrukce definují postup, jakým má být vytvořen kontejnerový obrázek. Tento soubor může zahrnovat informace o základním obrázku, instalaci softwaru, nastavení konfigurace a dalších operacích, které jsou potřebné k vytvoření prostředí, ve kterém bude kontejner spuštěn[24].

4.6.4 Docker Compose

Docker Compose je nástroj, který pomáhá definovat a sdílet vícekontejnerové aplikace. Je to soubor ve formátu YAML, ve kterém se definují služby, a jediným příkazem lze všechno spustit, nebo zastavit. Hlavní výhodou použití Docker Compose je možnost definovat celou svou aplikaci v jediném souboru, který lze uchovávat v kořeni repozitáře projektu[24].

4.7 Testování

Během vývoje aplikace byly implementovány testy na úrovni jednotek a také integrační testy.

4.7.1 Jednotkové testy

Jednotkové testy jsou testy, které se zaměřují na jednotlivé jednotky kódu, často na úrovni funkcí nebo metod. Cílem testů jednotek je ověřit správné chování jednotlivých komponent a kontrolovat, zda se chovají tak, jak by měly na základě specifikace. Tyto testy jsou obvykle automatizované a provedené během vývoje softwaru. Jsou rychlé a izolované, což znamená, že se nezabývají interakcí s jinými částmi systému.

4.7.2 Integrační testy

Na rozdíl od jednotkových testů se integrační testy zaměřují na ověření, zda různé části systému spolu správně komunikují a interagují. Integrační testy jsou určeny k testování interakcí mezi komponentami nebo moduly systému. Hlavním cílem těchto testů je zabezpečit, že jednotlivé části systému spolupracují v souladu s očekáváním a že komunikace mezi nimi probíhá korektně.

Integrační testy mohou zahrnovat různé aspekty, jako jsou testy aplikačního rozhraní (API), testy spojení s databází, testy komunikace se systémy třetích stran a další. Obvykle se provádí později během vývoje a mohou testovat větší části aplikace než jednotkové testy. Jejich úkolem je zajistit, že všechny komponenty systému spolupracují harmonicky a že dosahují požadované funkcionality.

4.7.3 JUnit

Pro tvorbu jednotkových testů byl využit nástroj JUnit[5], což je testovací rámec určený pro Java aplikace. Tento rámec poskytuje různé funkce a anotace, které umožňují definovat a provádět testovací scénáře, kontrolovat očekávané výsledky a vyhodnocovat, zda kód funguje správně.

4.7.4 Mockito

Mockito[6] je knihovna pro Java aplikace, která umožňuje vytvářet a používat tzv. mock objekty při psaní jednotkových testů. Mock objekty jsou zjednodušené náhražky reálných objektů a slouží k simulaci chování a vlastností těchto objektů. Mockito umožňuje vývojářům vytvářet tyto mock objekty, definovat, jak by se měly chovat (např. jak mají reagovat na volání metod), a následně je používat při testování.

Tímto způsobem mohou vývojáři izolovat jednotlivé části kódu (komponenty) a provádět testy na těchto komponentách nezávisle, bez nutnosti pracovat s reálnými objekty a jejich závislostmi. Mockito je často používáno ve spojení s testovacím rámcem JUnit a je užitečné při provádění jednotkových testů a testování chování kódu.

5. `https://junit.org`
6. `https://site.mockito.org`

4.7.5 Testcontainers

Testcontainers[7] je knihovna pro testování, která poskytuje jednoduchá a lehká API pro spuštění integračních testů s reálnými službami v Docker kontejnerech.

V průběhu typického integračního testu s Testcontainers se před testy spustí požadované služby v Docker kontejnerech, testy proběhnou s využitím těchto kontejnerizovaných služeb a po skončení se kontejnery zničí bez ohledu na výsledek testů.

4.8 Generování kódu

Lombok a MapStruct, knihovny pro generování kódu, byly použity k odstranění opakujícího se kódu, zlepšení jeho čitelnosti a udržitelnosti.

4.8.1 Lombok

Lombok[8] je knihovna, která umožňuje snadnou generaci opakujícího se kódu pomocí anotací. Tato knihovna je často využívána k odstranění konstruktorů a metod, jako jsou gettery a settery. Lombok umožňuje psát méně kódu, zlepšuje jeho čitelnost a udržovatelnost.

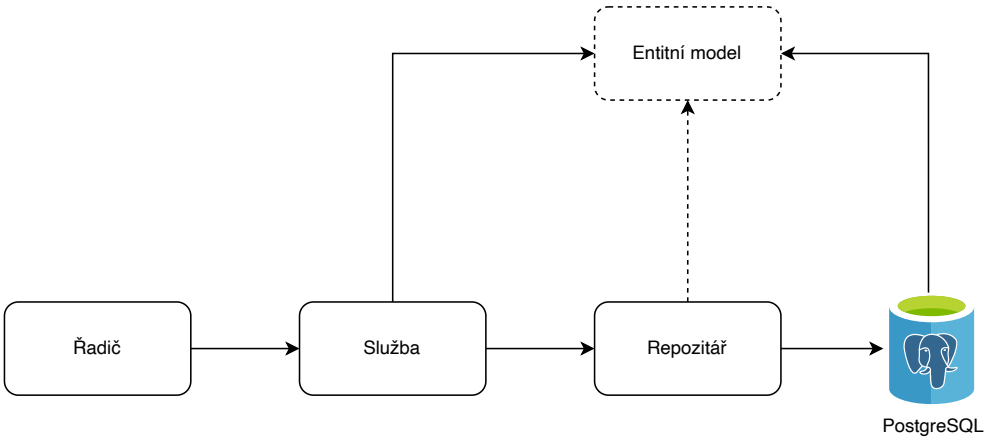
4.8.2 MapStruct

MapStruct[9] je knihovna pro mapování objektů. Tento nástroj umožňuje vytvářet efektivní mapovací kódy mezi různými datovými objekty, což je užitečné zejména při převodu dat mezi různými vrstvami aplikace, jako jsou doménové objekty, DTO[10] a další.

7. `https://testcontainers.com`
8. `https://projectlombok.org`
9. `https://mapstruct.org`
10. Data Transfer Object (objekt pro přenos dat)

5 Implementace

S cílem zvýšit údržbu a rozšiřitelnost byla zvolena vrstvená architektura, což je architektonický design, který organizuje aplikaci do logických vrstev. Každá vrstva má specifickou odpovědnost a interaguje s okolními vrstvami. Tento přístup pomáhá oddělit různé aspekty softwarového systému, což usnadňuje údržbu a rozšiřitelnost aplikace.



Obrázek 5.1: Architektura aplikace

5.1 Vrstva řadičů

Na vrstvě řadičů jsou definovány koncové body API. Hlavním účelem této vrstvy je poskytovat služby aplikace klientům a zpracovávat jejich požadavky. Tato vrstva neobsahuje žádnou aplikační logiku kromě zpracování požadavků a posílání odpovědí klientům. Využívá objekty ve formě DTO[1] z důvodu oddělení od datové struktury aplikace.

5.2 Vrstva služeb

Aplikační logika je umístěna ve vrstvě služeb (anglicky „Service Layer“), která zpracovává data získaná od řadičů a připravuje je pro

1. Data Transfer Object (objekt pro přenos dat)

datovou vrstvu. Jejím hlavním cílem je vykonávat operace, jako jsou výpočty, validace a manipulace s daty. Na rozdíl od vrstvy řadičů, která se zaměřuje na přijímání a odesílání dat klientům, vrstva služeb se stará o interní operace potřebné pro správný běh aplikace.

5.2.1 Mapování objektů

Pro převod mezi objekty DTO a entitami se využívají tzv. mappery (viz ukázka kódu B.4). Mappery v aplikaci jsou definovány pomocí nástroje MapStruct.

5.3 Datová vrstva

Datová vrstva se skládá z repozitářů, které provádějí operace do databáze. Díky repozitářům není potřeba vytvářet objekty pro přístup k datům (anglicky „Data Access Object“ — DAO). Repozitáře slouží jako rozhraní mezi aplikační logikou a databází a umožňují jednoduchý a standardizovaný přístup k datům. Příklad repozitáře je uveden v ukázce kódu B.3.

5.4 Entitní modely

Entitní model je reprezentace tabulky z databáze. Pro správu objektově relačního mapování (ORM) je využíván Hibernate. ORM je technika, která umožňuje mapovat objekty v programovacím jazyce na záznamy v relační databázi a naopak. Příklad entitního modelu je uveden v následujícím kódu: B.2.

5.5 Návrh API

Při návrhu a testování API byl využíván nástroj Swagger, který s pomocí anotací umožňuje snadno vytvořit specifikaci OpenAPI, jež slouží k popisu a dokumentaci API. Swagger umožňuje interaktivní zobrazení celého API prostřednictvím uživatelsky přívětivého rozhraní. Dokumentace API je k dispozici v příloze C této práce.

5.6 Autentizace

Všechny zabezpečené koncové body aplikace vyžadují autentizaci, pro niž je nezbytné v hlavičce požadavku poslat přístupový žeton.

```
curl -X GET \
  -H "Authorization: Bearer pristupovy_token" \
  https://api.example.com/applicants
```

Ukázka kódu 5.1: Autentizovaný požadavek

Použité přístupové a obnovovací žetony v aplikaci jsou ve formátu JWT, který je možné dekódovat. V dekódované struktuře žetonu jsou obsaženy základní informace o jeho držiteli.

- **sub** Identifikátor uživatele, pro kterého byl žeton vydán (ID účtu).
- **nui** ID uchazeče nebo společnosti.
- **scope** Uživatelská role.
- **exp** Unixový čas, kdy žeton vyprší.
- **iat** Unixový čas, kdy byl žeton vydán.
- **lang** Jazyk uživatele.
- **jti** ID žetonu.
- **email** Emailová adresa uživatele.

```
{
  "sub": "c7abf5c8-f533-4a96-8e8a-fc6329af1885",
  "nui": "55c9e3e0-ebab-4f4b-8aa1-191750c63df7",
  "scope": "REGULAR_USER",
  "exp": 1700696769,
  "lang": "cs",
  "iat": 1600380000,
  "jti": "1c62e03c-9fed-4464-bb20-bb6a0b7ddaa4",
  "email": "vitalii.bortsov@jobportal.cz"
}
```

Ukázka kódu 5.2: Dekódovaný přístupový žeton

5.7 Inicializace databáze

Pro inicializaci databáze byl využíván nástroj Liquibase. Konfigurace pomocí souborů ve formátu XML umožňuje snadno provádět změny v databázovém schématu. S jejich přispěním bylo inicializováno databázové schéma a přidány některé záznamy, jako například účet administrátora.

5.8 Stránkování a řazení

Většina koncových bodů, které slouží k získání seznamu objektů, podporuje stránkování a řazení. Při dotazování určitého seznamu je možné předat URL parametry `page` (číslo stránky), `size` (velikost stránky) a `sort` (řazení). Každý z těchto koncových bodů také podporuje fulltextové vyhledávání (parametr `q`).

5.9 Domovská stránka

Na obrázku A.1 je ukázána domovská stránka aplikace z pohledu nepřihlášeného uživatele. Pro návrat na domovskou stránku slouží logo v levém horním rohu. V horní části je nástrojová lišta (anglicky „toolbar“) s odkazy na vyhledávání společností a pracovních míst.

Vyhledávání pracovních míst je možné začít přidáním vyhledávacího dotazu přímo z domovské stránky, jak je vidět na obrázku A.19.

5.10 Přihlášení a registrace

Přihlášení a registrace jsou provedeny prostřednictvím dialogového okna, které se objeví po kliknutí na tlačítko v pravém horním rohu. Pro přihlášení je nutné zadat e-mailovou adresu a heslo, jak je ukázáno na obrázku A.2. Při registraci lze vybrat mezi registrací jako firma (viz obrázek A.4), nebo jako běžný uživatel (viz obrázek A.3).

5.11 Uživatelský účet

Po přihlášení se na místě tlačítka pro přihlášení zobrazí rozbalovací uživatelské menu, které nabízí možnosti přejít do profilu, do nastavení nebo se odhlásit (viz obrázek A.5).

Uživatelské profily jsou zobrazeny na obrázcích A.7 a A.13.

Společnosti ve svém účtu mohou spravovat pracovní místa, jak je vidět na obrázku A.14.

Uchazeči mohou spravovat své podané žádosti o zaměstnání (viz obrázek A.10).

5.12 Životopis

Uchazeči ve svém profilu mohou vygenerovat životopis na základě vyplněných informací ve svém účtu. K tomu slouží tlačítko zobrazené na obrázku A.8. Vygenerovaný soubor je možné stáhnout ve formátu PDF. Příklad toho, jak může životopis vypadat, je zobrazen na obrázku A.9.

5.13 Uložená pracovní místa

Při procházení nabídek práce si uchazeči mohou uložit pozice do seznamu oblíbených. Tyto uložené pozice jsou pak k dispozici v rozbalovacím menu na nástrojové liště (viz obrázek A.11) nebo v uživatelském profilu, jak je znázorněno na obrázku A.12.

5.14 Nastavení

V nastaveních mají uživatelé možnost zapnout, nebo vypnout oznámení, změnit heslo nebo smazat účet (viz obrázek A.18).

5.15 Lokalizace

Aplikace je dostupná ve čtyřech jazycích: češtině, slovinštině, angličtině a ruštině. Změnu jazyka lze provést v rozbalovacím jazykovém menu na nástrojové liště, které je zobrazeno na obrázku A.6. Vybraný jazyk se automaticky ukládá do nastavení účtu a při opětovném přihlášení bude zachován.

5.16 Vyhledávání práce

Při vyhledávání pracovních nabídek lze výsledky filtrovat podle společnosti a vybraných kategorií. Výsledky jsou zobrazeny v stránkovacím komponentu, kde je možné nastavit velikost a číslo stránky (viz obrázek A.20). Po kliknutí na pracovní pozici se zobrazí její detail (viz obrázek A.21) s podrobným popisem pozice. Dole je tlačítko pro podání žádosti a nahoře je uveden počet uchazečů.

5.17 Vytvoření pracovní nabídky

Při vytváření nové pracovní nabídky se zobrazí dialogové okno A.15. Popis pozice se zadává pomocí WYSIWYG editoru, což poskytuje možnost formátování textu.

5.18 Přehled uchazečů

Společnosti mohou u svých pracovních nabídek zobrazit přehled uchazečů A.22, kteří o tuto pozici projevili zájem. Po kliknutí na jméno uchazeče se zobrazí jeho profil se základními informacemi a pracovními zkušenostmi. Lze rychle přepínat mezi popisem pozice a profilem vybraného uchazeče, což usnadňuje posouzení žádosti a shody s požadavky pozice.

5.19 Schvalování žádosti

Schválit, nebo odmítnout žádost je možné pomocí tlačítek v přehledu uchazečů. Uchazeči s aktivovanými oznámeními obdrží e-mailovou zprávu o schválení/odmítnutí žádosti. Tato zpráva obsahuje název pozice, název společnosti a odkaz na aplikaci (viz obrázek A.23).

5.20 Seznam uživatelů aplikace

Při přihlášení jako administrátor se na nástrojové liště zobrazí také odkaz na seznam uživatelů A.24. Tento seznam je dostupný pouze pro administrátory, kteří mají oprávnění upravovat a mazat uživatelské účty.

5.21 Druhé uživatelské rozhraní

Pro demonstraci změny uživatelského rozhraní bylo vytvořeno dodatečné uživatelské rozhraní A.25, které využívá stejné API. Toto rozhraní má jednodušší design a je určeno pouze pro administrátory a správu dat v aplikaci. Díky tomu, že administrátoři mají přístup ke všemu, mohou pomocí tohoto rozhraní snadno spravovat celou aplikaci.

6 Nasazení aplikace

Aplikace je připravena k jednoduchému nasazení a lze ji nasadit pomocí platformy Docker. Každá služba má svůj vlastní soubor Dockerfile, který definuje kroky pro vytvoření Docker obrazu této služby. V kořenovém adresáři je také připraven soubor Docker Compose, který usnadňuje spuštění aplikace s potřebnými konfiguracemi. Pro nasazení je nutné mít nainstalovaný Docker a stáhnout nebo naklonovat aplikaci Portál práce.

6.1 Konfigurace

V souboru `docker-compose.yml` (ukázka kódu B.1) jsou uvedeny všechny služby a jejich příslušné konfigurace. Tento soubor obsahuje proměnné prostředí, které slouží k oddělení konfigurace spustitelného souboru od samotné aplikace.

Proměnné serveru jsou specifikovány v části `services.job-portal.environment`.

- `BASE_URI` představuje identifikátor zdroje (URI) pro klienta, zejména využívaný pro odkazy v e-mailových zprávách.
- `ACCESS_TOKEN_DURATION` určuje dobu platnosti přístupového žetonu.
- `REFRESH_TOKEN_DURATION` určuje dobu platnosti obnovovacího žetonu.
- `SPRING_DATASOURCE_URL` obsahuje URL adresu databáze.
- `SPRING_DATASOURCE_USERNAME` definuje uživatelské jméno pro přístup k databázi.
- `SPRING_DATASOURCE_PASSWORD` zastupuje heslo pro uživatele databáze.
- `EMAIL_STMP_HOST` určuje SMTP[1] hostitele pro emailovou službu, který je využíván příslušnou e-mailovou adresou specifikovanou v proměnné `EMAIL_NOTIFICATION_APP_USERNAME`.

1. `https://www.strafelda.cz/smtp`

- `EMAIL_NOTIFICATION_APP_USERNAME` představuje e-mailovou adresu používanou pro odesílání oznámení uživatelům.
- `EMAIL_NOTIFICATION_APP_PASSWORD` zahrnuje heslo aplikace, které umožňuje přístup k e-mailovému účtu[2].
- `NOTIFICATIONS_ENABLED` je nastaveno na hodnotu `true`, pokud je vyžadováno aktivovat oznámení. V opačném případě je nastaveno na `false`.

Proměnné databáze jsou specifikovány v části `services.db.environment` a jsou pouze dvě: `POSTGRES_USER` a `POSTGRES_PASSWORD`. Tyto proměnné by měly odpovídat hodnotám uvedeným v `SPRING_DATASOURCE_USERNAME` a `SPRING_DATASOURCE_PASSWORD`.

6.2 Spuštění aplikace

Pokud jsou všechna proměnná prostředí vyplněná, aplikace je připravena ke spuštění. Stačí použít příkaz `docker compose up -d --build`[3].

6.3 Změna uživatelského rozhraní

Pro změnu uživatelského rozhraní je nutné vytvořit Dockerfile pro nové uživatelské rozhraní. Po vytvoření Dockerfile existují dva způsoby, jak integrovat novou službu do souboru `docker-compose.yml`.

První způsob spočívá v tom, že se vytvoří Docker obraz z aplikace obsahující nové uživatelské rozhraní. Pro vytvoření obrazu pomocí Dockerfile je potřeba použít příkaz `docker build`[4]. Vytvořený Docker obraz bude mít svůj název a verzi. Následně je třeba upravit soubor `docker-compose.yml` tak, aby část `services.frontend.image` odkazovala na nově vygenerovaný Docker obraz. Taktéž je potřeba odstranit část `services.frontend.build`, neboť používáme již existující Docker obraz.

2. `https://support.google.com/accounts/answer/185833?hl=cs`
3. `https://docs.docker.com/engine/reference/commandline/compose_up`
4. https://docs.docker.com/engine/reference/commandline/build

Druhý způsob nevyžaduje vytvoření nového Docker obrazu. Stačí upravit soubor `docker-compose.yml` tak, aby v části `services.frontend.build.context` byla specifikována cesta k Dockerfile pro nové uživatelské rozhraní.

7 Návrhy na zlepšení

7.1 Zlepšení propojení s e-mailem

Uživatelské účty v aplikaci nejsou pevně spojeny s e-mailovými adresami, na které jsou registrovány. Toto bylo zavedeno za účelem umožnění spuštění a používání aplikace bez nutnosti existujícího e-mailu nebo přístupu k internetu. Nicméně by bylo rozumné posílit vazbu mezi uživatelskými účty a e-maily, což je standardní praxe u většiny internetových služeb s možností registrace účtů.

V současné podobě aplikace chybí ověření uvedených e-mailových adres, což umožňuje libovolnému uživateli registrovat účet na cizí e-mailovou adresu. Přidání ověření e-mailu by znamenalo, že neověření uživatelé by měli omezené možnosti interakce s aplikací. Například by stále mohli upravovat svůj účet, ale nemohli by se ucházet o pracovní místa.

Další vylepšení by mohlo zahrnovat možnost obnovení zapomenutého hesla prostřednictvím e-mailové adresy. V současné době neexistuje mechanismus pro obnovu zapomenutého hesla, což může být překážkou pro uživatele, kteří zapomněli své přihlašovací údaje.

7.2 OAuth 2.0

Aplikace momentálně podporuje pouze klasické přihlášení pomocí e-mailu a hesla. Možnost přihlášení lze však rozšířit o další varianty, například o OAuth 2.0[1]. Tento typ přihlášení je běžný v mnoha internetových službách a umožňuje uživatelům přihlásit se pomocí jiných služeb, jako jsou Google, Facebook nebo GitHub. Také by bylo možné zahrnout přihlašování pomocí účtu z Informačního systému Masarykovy univerzity[2].

---

1. `https://oauth.net/2`
2. Jednotné přihlášení MUNI: `https://it.muni.cz/sluzby/jednotne-prihlaseni-na-muni`

7.3 Soukromé zprávy

Komunikace mezi uchazečem a zaměstnavatelem v aplikaci je omezena pouze na oznámení o schválení, nebo odmítnutí žádosti. Bylo by však vhodné rozšířit tuto komunikaci o možnost posílání zpráv. Tyto zprávy by mohly být odesílány buď prostřednictvím e-mailu, nebo by bylo možné implementovat systém soukromých zpráv, který by fungoval prostřednictvím protokolu Websocket.

Závěr

V této bakalářské práci jsem se zaměřil na návrh a implementaci webové aplikace s dobře definovaným API, které umožňuje snadnou úpravu uživatelského rozhraní.

Práce započala analýzou existujících řešení, na jejichž základě jsem identifikoval nezbytné služby, které by měla aplikace poskytovat.

Následně jsem se věnoval definování služeb, stanovování funkčních a nefunkčních požadavků a navrhl datový model. Pro zajištění úspěšnosti výsledné aplikace jsem vybral potřebné nástroje a provedl jejich analýzu.

Implementace aplikace proběhla úspěšně, přičemž výsledná aplikace plně splňuje stanovené požadavky a návrh.

Pro zajištění přenositelnosti a jednoduchého nasazení aplikace jsem využil kontejnerizaci pomocí platformy Docker. Tímto způsobem lze spouštět aplikaci na různých operačních systémech, které podporují Docker.

Celkově lze konstatovat, že cílů této bakalářské práce bylo úspěšně dosaženo. Webová aplikace Portál práce, kterou jsem navrhl a implementoval, plně splňuje všechny stanovené požadavky. API této aplikace nabízí flexibilitu pro vývoj alternativních rozhraní a díky použití kontejnerizace je aplikace snadno přenosná a nasaditelná v různých prostředích.

Bibliografie

1. KOKEŠOVÁ, Gabriela. *Přehled nejlepších pracovních portálů v ČR* [online]. Laba [cit. 2023-11-29]. Dostupné z: https://l-a-b-a.cz/blog/705-prehled-nejlepsich-pracovnich-portalu-v-cr.
2. REICHLOVÁ, DENISA. *Proces a metody získávání pracovníků ve vybrané společnosti*. 2021. Dostupné také z: https://theses.cz/id/gc8tgy. Bakalářská práce. Vysoká škola ekonomie a managementu, o.p.s.Praha.
3. *Co vlastně to LMC dělá?* [online]. LMC s.r.o. [cit. 2023-10-15]. Dostupné z: https://www.lmc.eu/co-vlastne-to-lmc-dela.
4. *Jobs.cz* [online]. LMC s.r.o. [cit. 2023-10-15]. Dostupné z: https://www.jobs.cz.
5. *Prace.cz nebo Jobs.cz?* [online]. LMC s.r.o. [cit. 2023-10-15]. Dostupné z: https://magazin.lmc.eu/clanky/prace-cz-nebo-jobs-cz.
6. JEFF WHITE, CASSIE BOTTORFF, ROB WATTS. *Indeed Review 2023: Features, Pros and Cons* [online]. Forbes Media LLC [cit. 2023-10-15]. Dostupné z: https://www.forbes.com/advisor/business/software/indeed-review-for-employers.
7. *What is an API (application programming interface)?* [online]. IBM [cit. 2023-03-13]. Dostupné z: https://www.ibm.com/topics/api.
8. *API Types and Protocols. What type of API do you need, and what protocols and standards should it use?* [online]. Stoplight [cit. 2023-03-13]. Dostupné z: https://stoplight.io/api-types.
9. *What is open API (public API)?* [online]. TechTarget [cit. 2023-03-15]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/open-API-public-API.
10. *Google APIs Explorer* [online]. Google [cit. 2023-03-15]. Dostupné z: https://developers.google.com/apis-explorer.
11. GILLIS, Alexander S. *REST API (RESTful API)* [online]. TechTarget [cit. 2023-11-08]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API.

12. *What is SOAP API: Formats, Protocols, and Architecture* [online]. AltexSoft Inc. [cit. 2023-11-08]. Dostupné z: https://www.altexsoft.com/blog/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/.

13. *GraphQL | A query language for your API* [online]. The GraphQL Foundation [cit. 2023-11-08]. Dostupné z: https://graphql.org/.

14. SAYAGO HEREDIA, Jaime; FLORES-GARCÍA, Evelin; SOLANO, Andres Recalde. Comparative Analysis Between Standards Oriented to Web Services: SOAP, REST and GRAPHQL. In: BOTTO-TOBAR, Miguel; ZAMBRANO VIZUETE, Marcelo; TORRES-CARRIÓN, Pablo; MONTES LEÓN, Sergio; PIZARRO VÁSQUEZ, Guillermo; DURAKOVIC, Benjamin (ed.). *Applied Technologies*. Cham: Springer International Publishing, 2020, s. 286–300. ISBN 978-3-030-42517-3.

15. *What is Java?* [online]. Amazon Web Services [cit. 2023-10-16]. Dostupné z: https://aws.amazon.com/what-is/java.

16. GOSLING, J.; MCGILTON, H. *The Java Language Environment: A White Paper*. Sun Microsystems Computer Company, 1995. Dostupné také z: https://books.google.cz/books?id=pUJ_GwAACAAJ.

17. *TypeScript Language Specification*. Microsoft Corporation, 2015. Dostupné také z: https://web.archive.org/web/20131117065339/http://www.typescriptlang.org/Content/TypeScript%20Language%20Specification.pdf.

18. *Spring* [online]. VMware, Inc. [cit. 2023-10-22]. Dostupné z: https://spring.io.

19. *Angular* [online]. Google [cit. 2023-10-22]. Dostupné z: https://angular.io.

20. *PostgreSQL: The World's Most Advanced Open Source Relational Database* [online]. The PostgreSQL Global Development Group [cit. 2023-10-22]. Dostupné z: https://www.postgresql.org.

21. *Liquibase Documentation* [online]. Liquibase Inc. [cit. 2023-10-22]. Dostupné z: https://docs.liquibase.com/home.html.

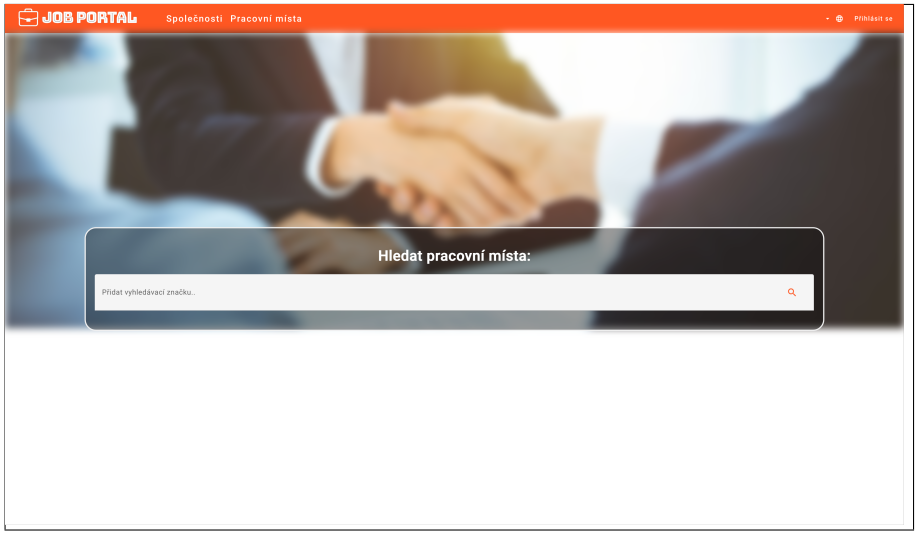
22. *Hibernate Search 6.2.2.Final: Reference Documentation* [online]. [cit. 2023-10-22]. Dostupné z: `https://docs.jboss.org/hibernate/stable/search/reference/en-US/html_single`.

23. *Text Analysis within a Full-Text Search Engine* [online]. Abhinav Dangeti [cit. 2023-10-22]. Dostupné z: `https://www.couchbase.com/blog/full-text_search_text_analysis`.

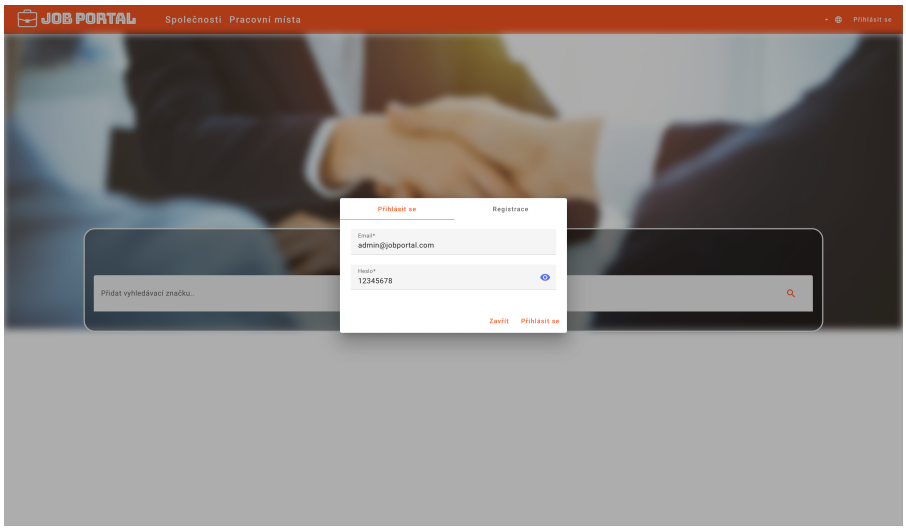
24. *Docker Docs* [online]. Docker Inc. [cit. 2023-10-28]. Dostupné z: `https://docs.docker.com/`.

A Obrázky

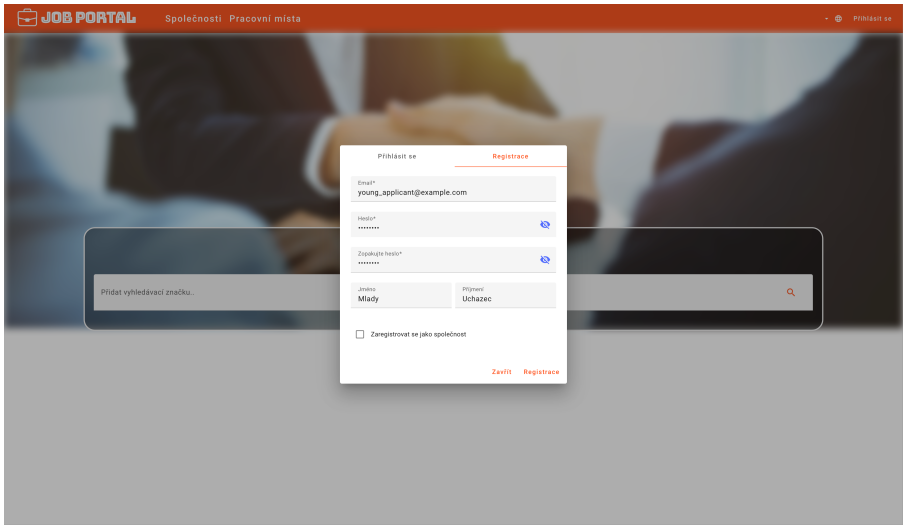
Tato příloha obsahuje obrázky, na které se odkazuji v hlavním textu práce.



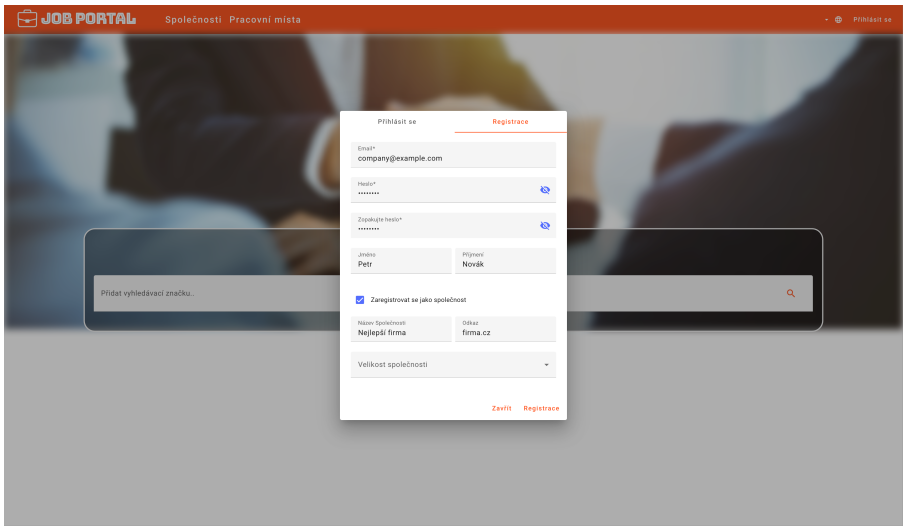
Obrázek A.1: Domovská stránka aplikace Portál práce



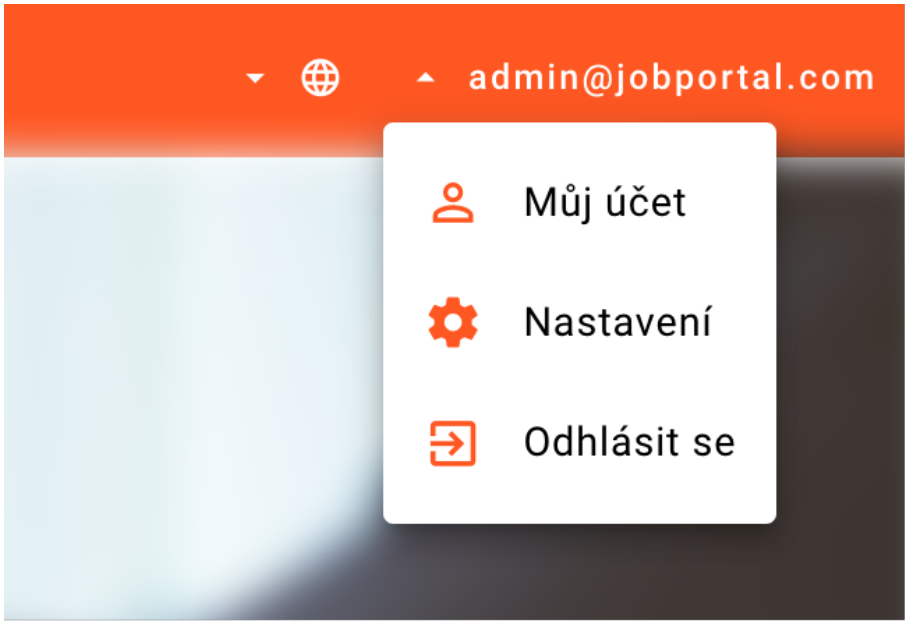
Obrázek A.2: Přihlášení



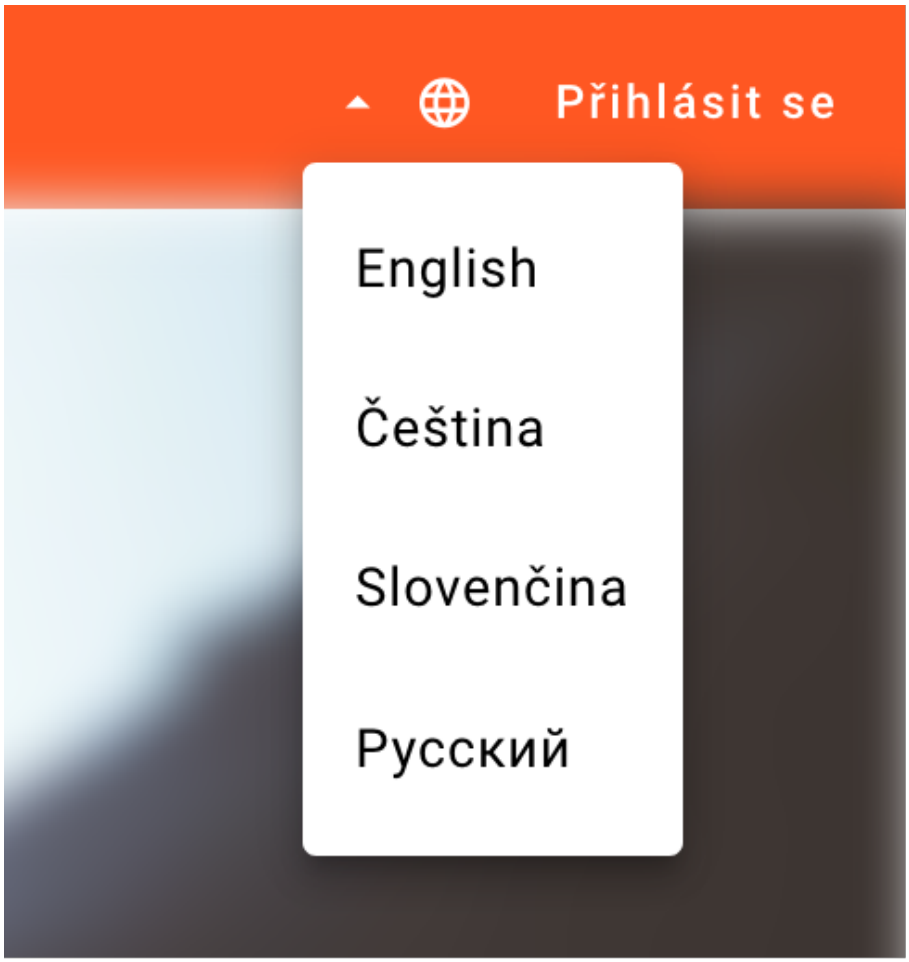
Obrázek A.3: Registrace účtu



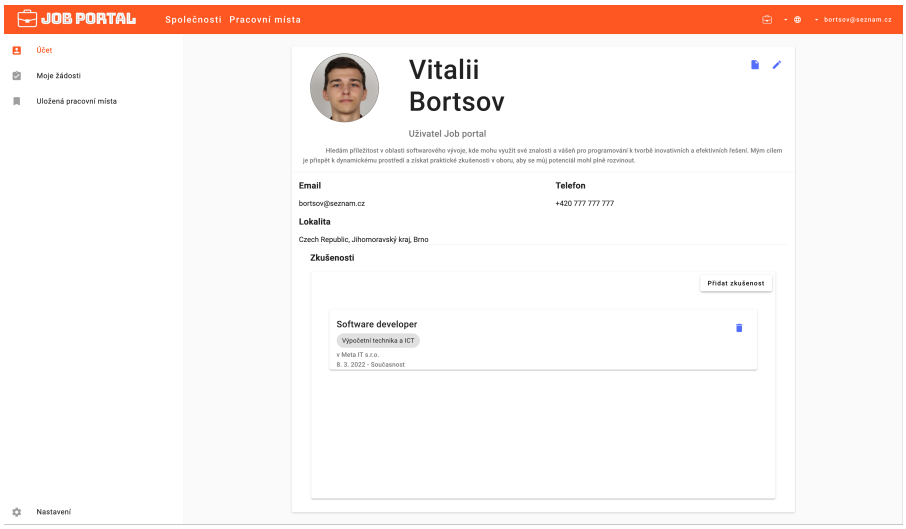
Obrázek A.4: Registrace účtu společnosti



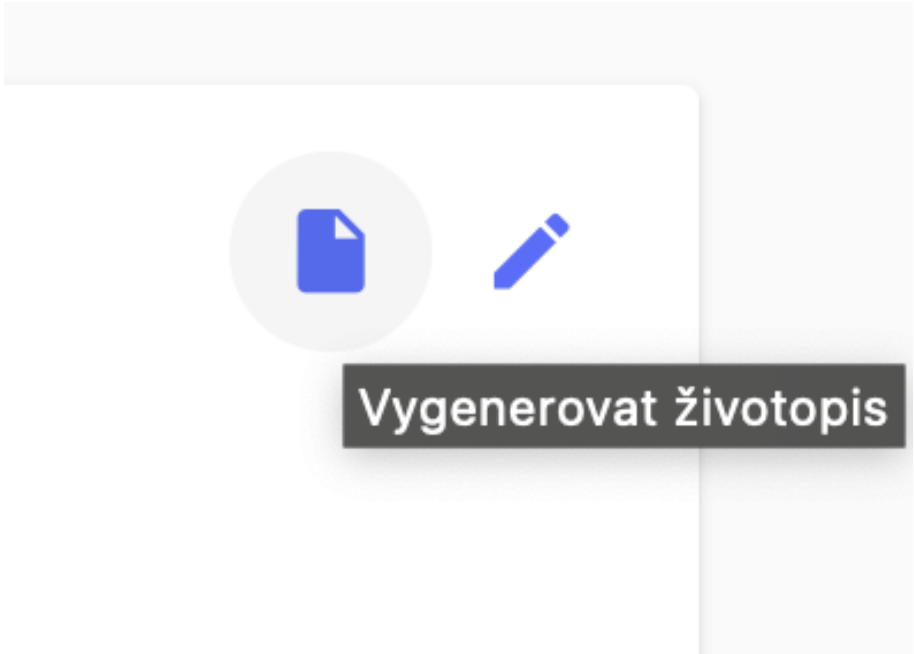
Obrázek A.5: Uživatelské menu v pravém horním rohu



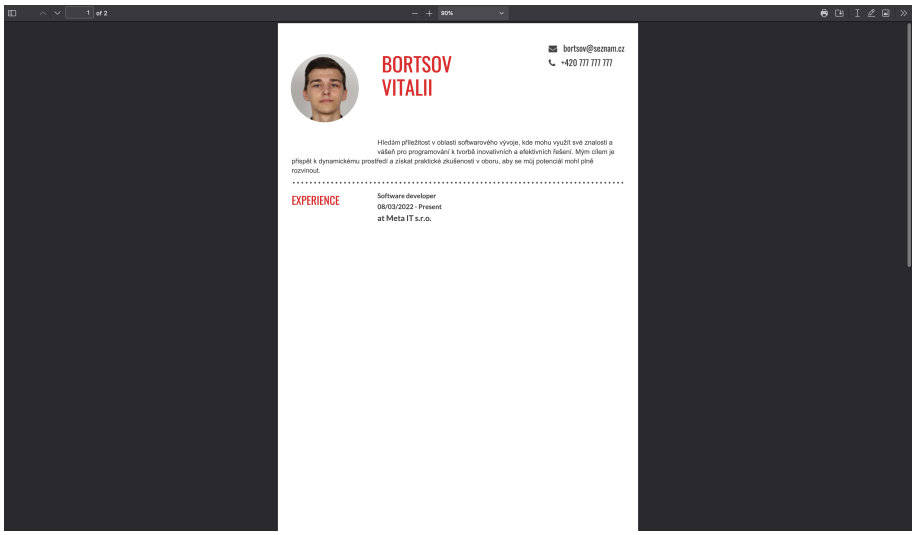
Obrázek A.6: Menu s jazykovými možnostmi v pravém horním rohu



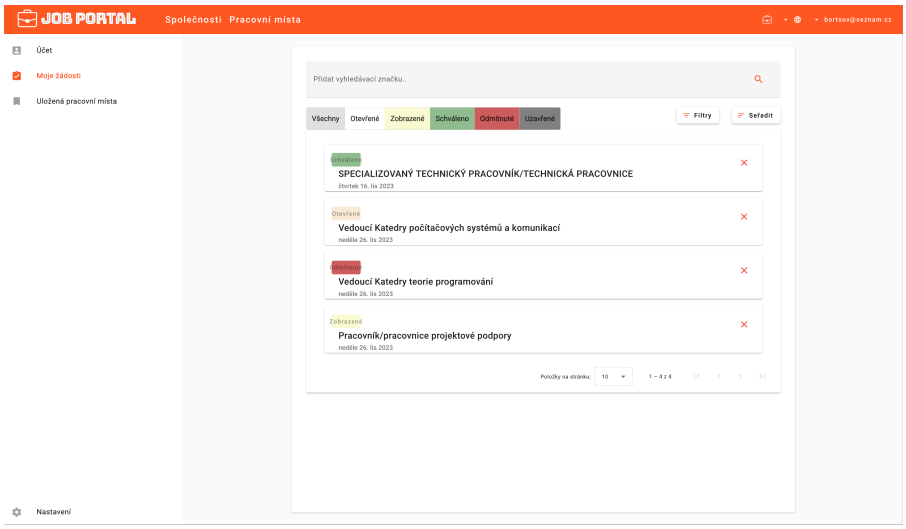
Obrázek A.7: Uživatelský profil



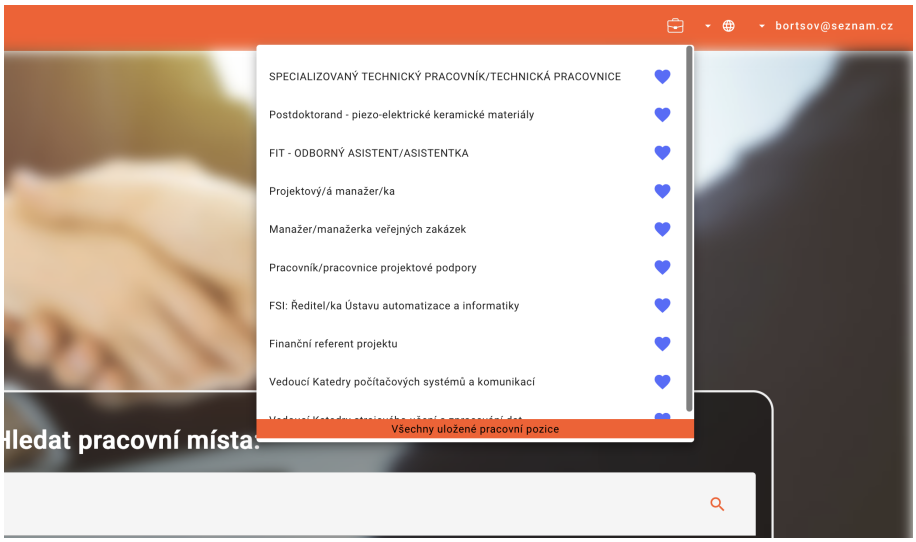
Obrázek A.8: Tlačítko pro generování životopisu



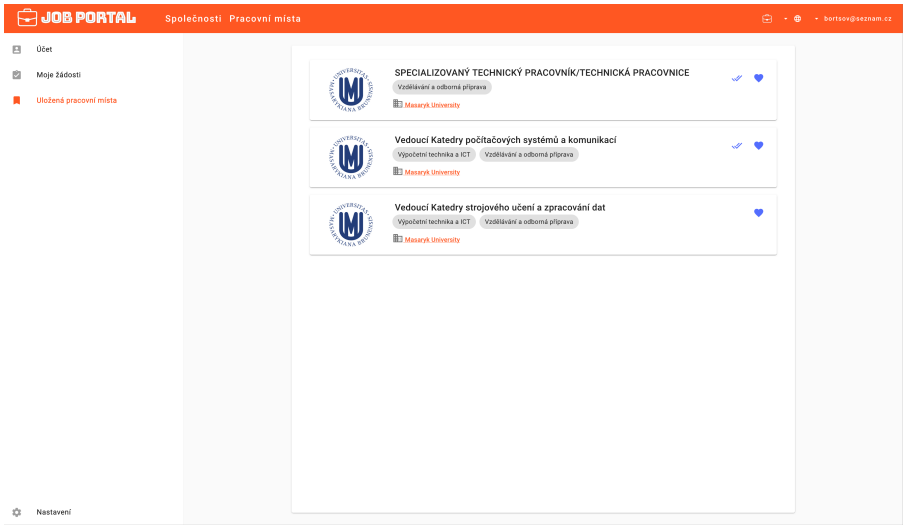
Obrázek A.9: Vygenerovaný životopis



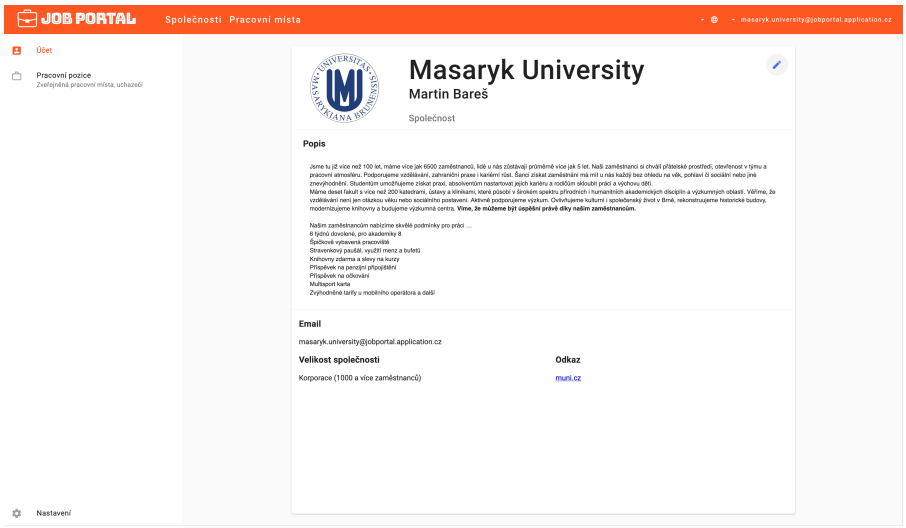
Obrázek A.10: Seznam podaných žádostí



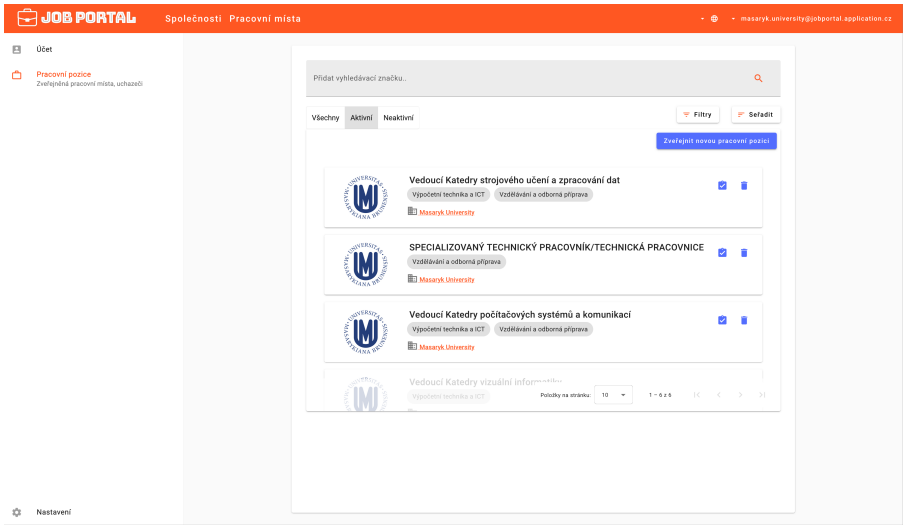
Obrázek A.11: Menu s uloženými pracovními pozicemi



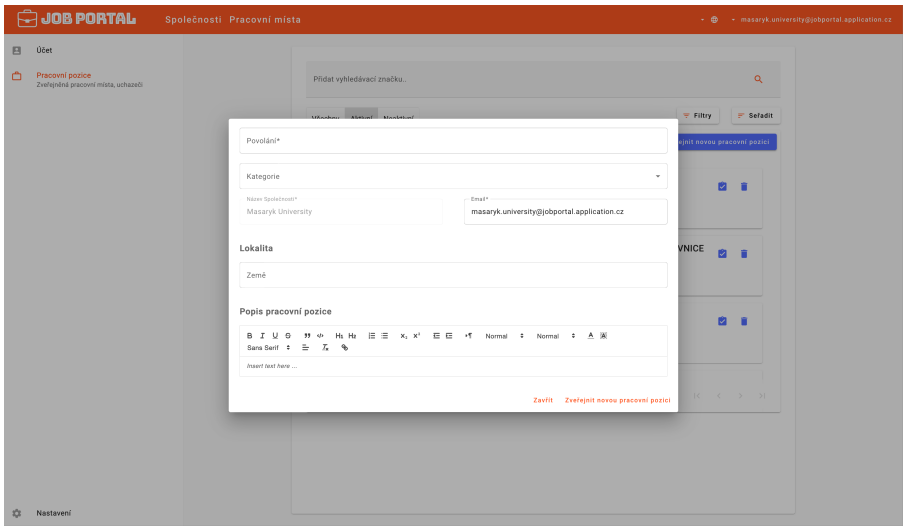
Obrázek A.12: Seznam uložených pracovních pozic



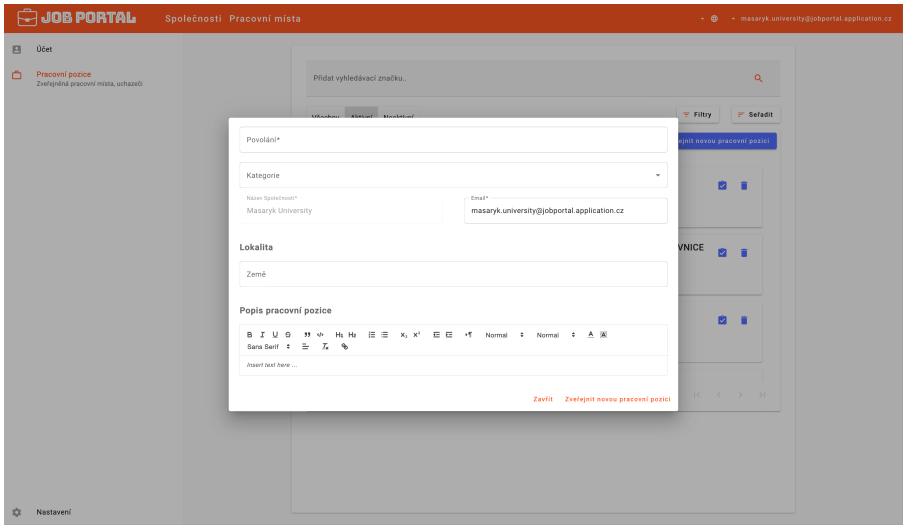
Obrázek A.13: Uživatelský profil společnosti



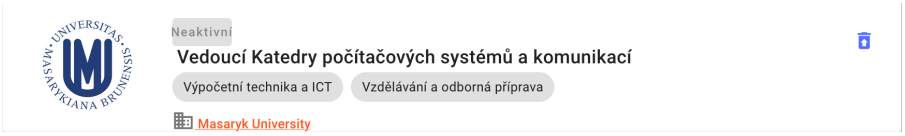
Obrázek A.14: Seznam zveřejněných pracovních míst



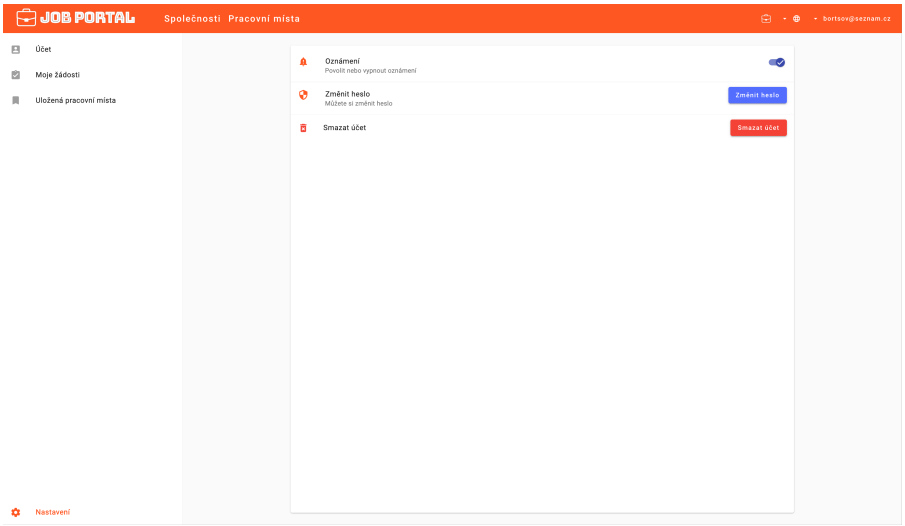
Obrázek A.15: Vytvoření nové pozice



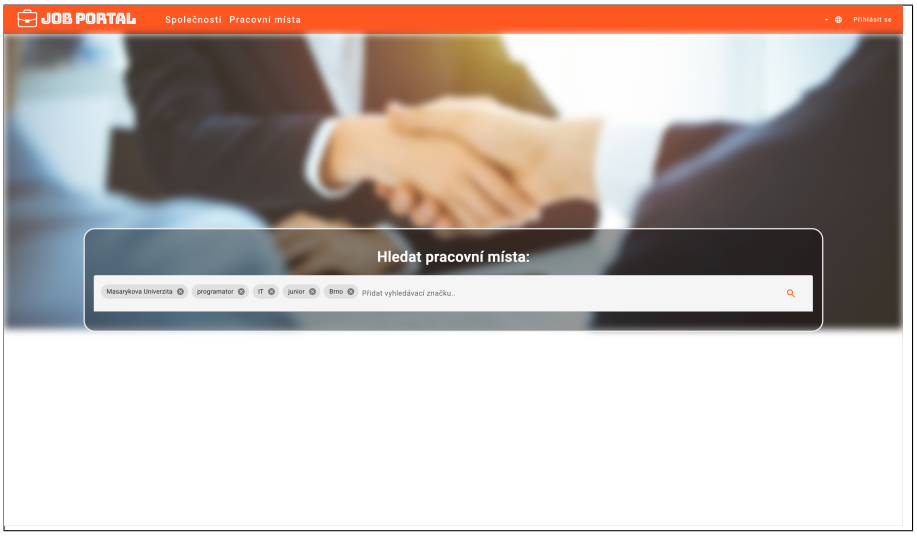
Obrázek A.16: Deaktivace pracovní nabídky



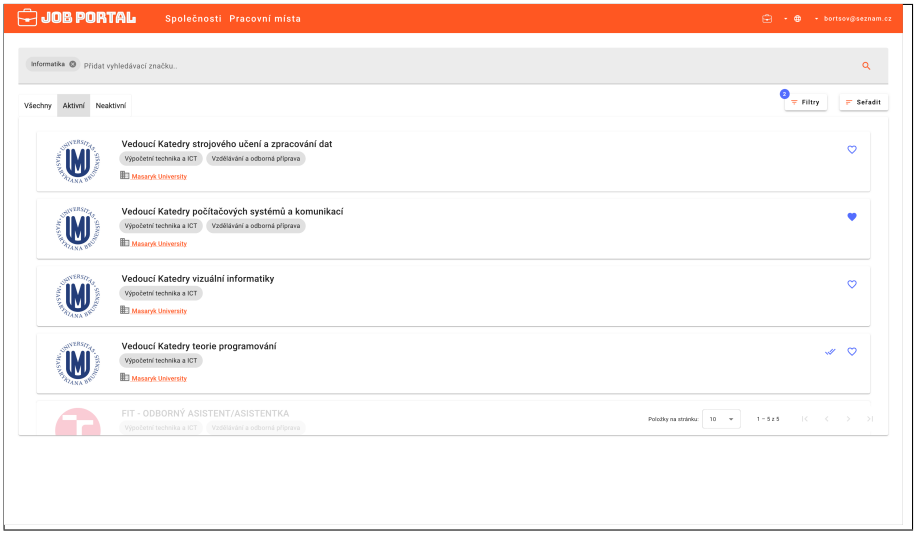
Obrázek A.17: Uzavřená pozice



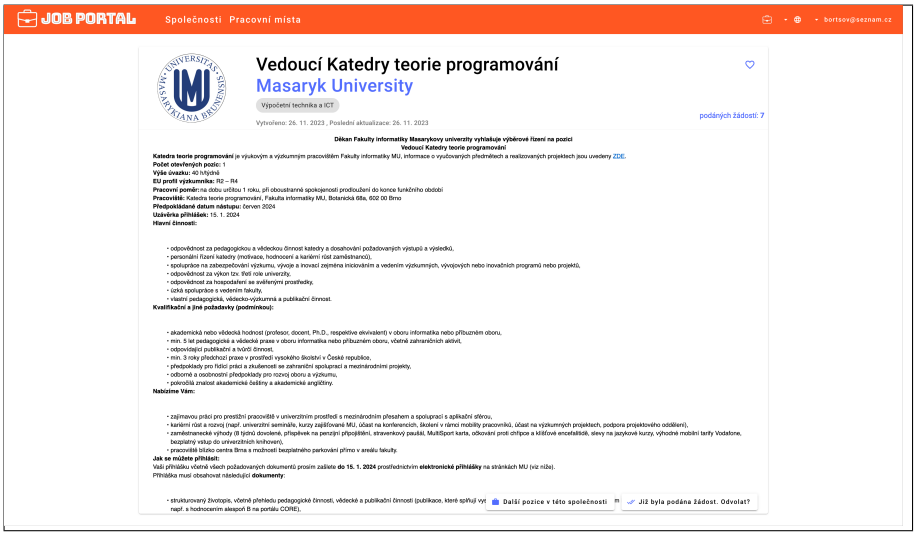
Obrázek A.18: Uživatelská nastavení



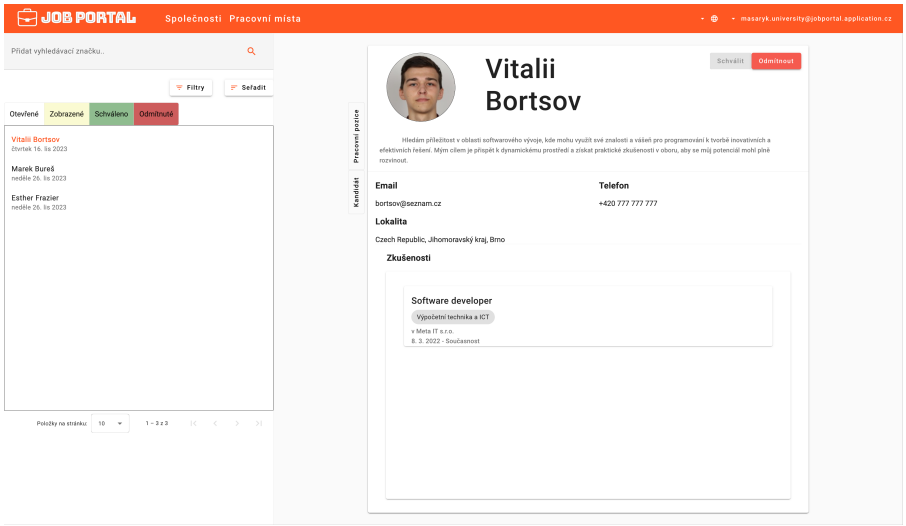
Obrázek A.19: Zadávání vyhledávacích dotazů



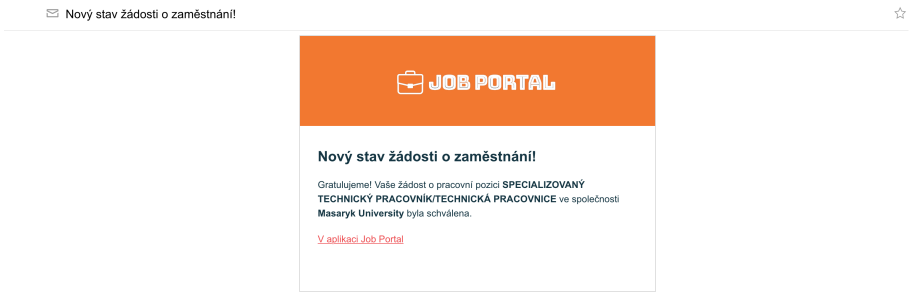
Obrázek A.20: Vyhledávání pracovních pozic



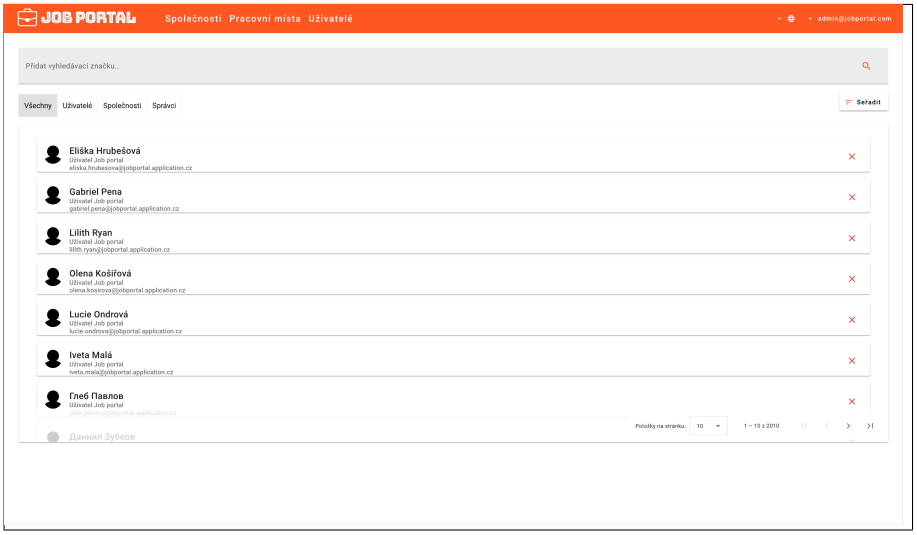
Obrázek A.21: Popis pracovní nabídky



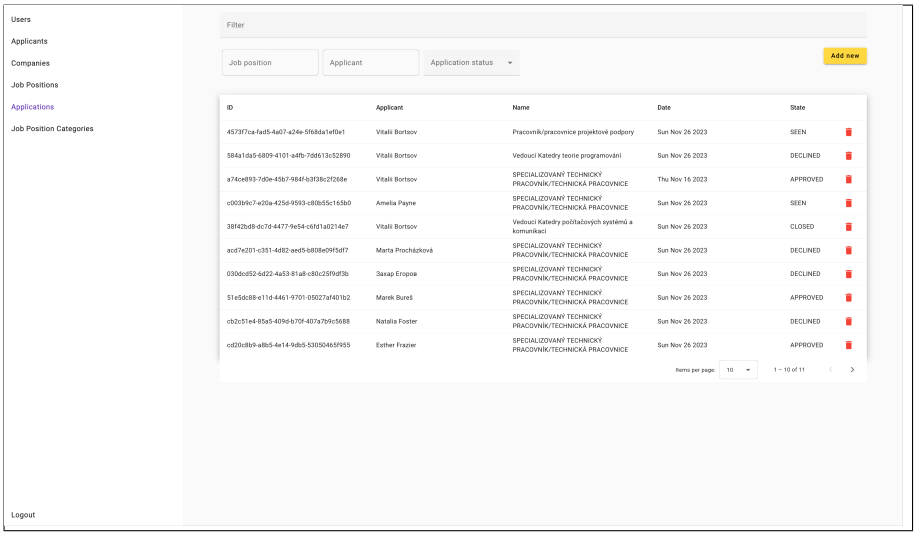
Obrázek A.22: Přehled kandidátů



Obrázek A.23: Oznámení o schválení žádosti



Obrázek A.24: Seznam uživatelů aplikace



Obrázek A.25: Uživatelské rozhraní pro správu aplikace

B Ukázky kódu

Tato příloha obsahuje ukázky kódu, ke kterým se odkazuji v hlavním textu práce.

```yaml
version: '3.9'

services:
  job-portal:
    image: job-portal
    build:
      context: ./job-portal-be
    container_name: job-portal
    ports:
      - "8080:8080"
    depends_on:
      - db
    environment:
      - BASE_URI=http://localhost:4200
      - ACCESS_TOKEN_DURATION=1H
      - REFRESH_TOKEN_DURATION=1D
      - SPRING_DATASOURCE_URL=jdbc:postgresql://db:5432
    ↪ /jobportal
      - SPRING_DATASOURCE_USERNAME=jobportal
      - SPRING_DATASOURCE_PASSWORD=jobportal
      - SPRING_JPA_HIBERNATE_DDL_AUTO=validate
      - EMAIL_STMP_HOST=smtp.gmail.com
      - EMAIL_PORT=587
      -
    ↪ EMAIL_NOTIFICATION_APP_USERNAME=example@gmail.com
      - EMAIL_NOTIFICATION_APP_PASSWORD=examplepssword
      - NOTIFICATIONS_ENABLED=true
    volumes:
      - app-data:/tmp

  frontend:
    image: job-portal-fe
```

```
31      container_name: job-portal-fe
32      build:
33        context: ./job-portal-fe
34      ports:
35        - "4200:4200"
36      depends_on:
37        - job-portal
38
39    db:
40      image: 'postgres:13.1-alpine'
41      container_name: db
42      ports:
43        - "5432:5432"
44      environment:
45        - POSTGRES_USER=jobportal
46        - POSTGRES_PASSWORD=jobportal
47      volumes:
48        - postgres-data:/var/lib/postgresql/data
49  volumes:
50    postgres-data:
51    app-data:
```

Ukázka kódu B.1: Soubor `docker-compose.yml`

```java
@Getter
@Setter
@Entity
@Table(name = "jp_users")
@EqualsAndHashCode(of = "id")
public class User {

  @Id
  @GeneratedValue
  private UUID id;

  @Column(name = "email", unique = true)
  private String email;
  private String password;
  private String name;
  private String lastName;
  @Enumerated(EnumType.STRING)
  private JobPortalScope scope;
  @OneToOne(mappedBy = "user", cascade =
   ↪  {CascadeType.PERSIST, CascadeType.REMOVE})
  @JoinColumn(name = "applicant_id")
  private Applicant applicant;
  @OneToOne(mappedBy = "user", cascade =
   ↪  {CascadeType.PERSIST, CascadeType.REMOVE})
  @JoinColumn(name = "company_id")
  private Company company;
}
```

Ukázka kódu B.2: Příklad entitního modelu uživatele

```java
public interface UserRepository extends
 ↪  JpaRepository<User, UUID> {

  @Query("SELECT u FROM User u WHERE u.email = :email")
  Optional<User> findByEmail(String email);

  @Query("SELECT CASE WHEN COUNT(u) > 0 THEN TRUE ELSE
   ↪  FALSE END FROM User u WHERE u.email = :email")
  boolean existsByEmail(String email);
}
```


Ukázka kódu B.3: Příklad repozitáře

```java
@Mapper
public interface CompanyMapper {

  @Mapping(target = "user", ignore = true)
  Company map(CompanyCreateDto request);

  @Mapping(target = "email", source = "user.email")
  @Mapping(target = "userId", source = "user.id")
  CompanyDetailDto map(Company source);

  @Mapping(target = "userId", source = "user.id")
  CompanyDto mapDto(Company source);

  @Mapping(source = "name", target = "user.name")
  @Mapping(source = "lastName", target =
   ↪  "user.lastName")
  Company update(@MappingTarget Company target,
   ↪  CompanyUpdateDto source);

  @Mapping(target = "user", source = "created.id")
  CompanyCreateDto map(UserCreateDto request, User
   ↪  created);

}
```

Ukázka kódu B.4: Třída pro mapování objektu

C Dokumentace API

Tato příloha obsahuje popis všech dostupných koncových bodů. V elektronické příloze D je rovněž k dispozici specifikace OpenAPI 3 a vygenerovaný HTML klient, který obsahuje detailní popis koncových bodů, těl požadavku, těl odpovědí a parametrů.

Autorizace

`POST /auth/login`

Koncový bod pro přihlášení uživatele.

`POST /auth/register`

Koncový bod pro registraci uživatele.

`POST /auth/refresh`

Koncový bod pro obnovení přístupového žetonu.

Uchazeči

Všechny koncové body týkající se uživatelů s běžným účtem jsou zabezpečené a vyžadují autentizaci.

`GET /applicants`

Koncový bod pro získání všech uživatelů.
URL parametry:

- Stránkování (`page`, `size`, `sort`)
- Vyhledávací dotazy (`q`)
- ID pracovního místa (`jobPosition`)

Vlastnosti pro řazení uchazečů	
Atribut	Název vlastnosti pro řazení
Jméno uchazeče	`name`
Název pracovní pozice	`jobPosition`
Lokalita	`country`, `state`, `city`

Koncový bod je bez omezení dostupný pouze pro administrátory. Uživatelé s firemním účtem mohou vyhledávat pouze uchazeče o pracovní místo ve své společnosti.

`GET /applicants/:ID`

Koncový bod pro získání informací o uchazeči na základě jeho ID. K němu mají přístup uchazeči s odpovídajícím ID, administrátoři a společnosti.

`GET /applicants/:ID/generate-cv`

Koncový bod pro generování životopisu uchazeče s daným ID. Přístupný pro uchazeče s odpovídajícím ID a administrátory.

`PUT /applicants/:ID`

Koncový bod pro aktualizaci informací o uchazeči na základě jeho ID. Je přístupný pro uchazeče s odpovídajícím ID a administrátory.

`POST /applicants/:ID/experience`

Koncový bod pro vytvoření nové pracovní zkušenosti uchazeče na základě jeho ID. Je přístupný pro uchazeče s odpovídajícím ID a administrátory.

`DELETE /applicants/:ID/experience/:ID`

Koncový bod pro odstranění pracovní zkušenosti uchazeče na základě ID uchazeče a ID zkušenosti. Má přístup uchazeč s odpovídajícím ID a administrátoři.

Žádosti o zaměstnání

Všechny koncové body týkající se žádostí vyžadují autentizaci.

`GET /applications`

Koncový bod pro získání všech žádostí.
URL parametry:

- Stránkování (`page`, `size`, `sort`)
- Vyhledávací dotazy (`q`)
- ID uchazeče (`applicant`)
- ID pracovního místa (`jobPosition`)
- ID společnosti (`company`)
- Stav žádosti (`status`)
- Datum podání žádosti od a do (`dateFrom`, `dateTo`)

Vlastnosti pro řazení žádostí	
Atribut	Název vlastnosti pro řazení
Název společnosti	`company`
Název pracovní nabídky	`jobPosition`
Jméno uchazeče	`applicant`
Stav žádosti	`status`
Datum žádosti	`date`

Koncový bod je plně přístupný pouze pro administrátory. Uživatelé s firemním účtem mohou vyhledávat pouze žádosti o pracovní místo ve své společnosti, zatímco uživatelé s běžným účtem mají přístup pouze ke svým vlastním žádostem.

`GET /applications/:ID`

Koncový bod pro získání informací o žádosti na základě jejího ID. Přístupová práva jsou stejná jako u `GET /applications`.

`DELETE /applications/:ID`

Koncový bod pro odstranění žádosti o pracovní místo. Operace je přístupná pouze administrátorům a vlastníkům žádosti.

`PUT /applications/:ID/state`

Koncový bod pro změnu stavu žádosti. Oprávnění k provádění této operace mají administrátoři a uživatelé s firemním účtem, kteří mají přístup k dané žádosti.

Společnosti

Všechny koncové body pro společnosti používající HTTP metodu `GET` nevyžadují autentizaci.

`GET /companies`

Koncový bod pro získání všech společností.
URL parametry:

- Stránkování (`page`, `size`, `sort`)
- Vyhledávací dotazy (`q`)
- Velikost společnosti (`companySize`)

Vlastnosti pro řazení společností	
Atribut	Název vlastnosti pro řazení
Název společnosti	`name`
Popis společnosti	`description`
Velikost společnosti	`companySize`

`GET /companies/:ID`

Koncový bod pro získání informací o společnosti na základě ID.

`PUT /companies/:ID`

Koncový bod pro aktualizaci informací o společnosti na základě ID. Aktualizaci mohou provést administrátoři nebo samotná společnost.

Uživatelské účty

Všechny koncové body v této sekci jsou zabezpečené, s výjimkou `GET /users/:ID/avatar`.

`GET /users`

Koncový bod pro získání všech uživatelů.

URL parametry:

- Stránkování (`page`, `size`, `sort`)
- Vyhledávací dotazy (`q`)
- Role uživatele (`scope`)

Vlastnosti pro řazení uživatelů	
Atribut	Název vlastnosti pro řazení
Jméno uživatele	`name`
Emailová adresa	`email`
Role	`scope`

Přistup k tomuto koncovému bodu má pouze administrátor.

`GET /users/:ID`

Koncový bod pro získání informací o uživateli na základě ID. Informace o ostatních uživatelích, kromě sebe samého, může získat pouze administrátor.

`GET /users/:ID/avatar`

Koncový bod pro získání veřejných obrázků účtů, tedy účtů společností.

`GET /users/:ID/avatar-secure`

Zabezpečený koncový bod pro získání obrázků účtů uchazečů. Administratori mohou získat libovolný obrázek, zatímco společnosti mají přístup pouze k obrázkům těch uživatelů, kteří jsou uchazeči o nějaké pracovní místo v této společnosti. Všichni uživatelé mají oprávnění k obrázku svého účtu.

`GET /users/:ID/preferences`

Koncový bod pro získání nastavení účtu uživatele na základě ID účtu.

`PUT /users/:ID`

Zabezpečený koncový bod pro aktualizaci informací o uživateli na základě ID. Oprávnění k provedení aktualizace mají uživatelé, kteří mají přístup k tomuto účtu.

`PUT /users/:ID/preferences`

Koncový bod pro aktualizaci nastavení účtu uživatele na základě ID účtu.

`POST /users/:ID/avatar`

Nahrávání nového obrázku účtu na základě ID účtu. Oprávnění k provedení této operace mají uživatelé, kteří mají přístup k tomuto účtu.

`DELETE /users/:ID`

Koncový bod pro odstranění uživatele. Tuto operaci může provést samotný uživatel na svém účtu nebo administrátor.

`DELETE /users/:ID/avatar`

Koncový bod pro odstranění obrázku účtu. Tuto operaci může provést vlastník účtu nebo administrátor.

Pracovní místa

Zabezpečené jsou všechny koncové body související s pracovními nabídkami, s výjimkou `GET /positions/:ID` a `GET /positions`.

`GET /positions`

Koncový bod pro získání všech pracovních míst.
URL parametry:

- Stránkování (`page`, `size`, `sort`)
- Vyhledávací dotazy (`q`)
- Stav nabídky (`status`)
- Kategorie (`categories`)
- Společnosti (`companies`)

Vlastnosti pro řazení pracovních míst	
Atribut	Název vlastnosti pro řazení
Název pracovní nabídky	`name`
Název společnosti	`company`
Kategorie	`category`
Popis pozice	`description`
Datum vytvoření	`created`
Lokalita	`country`, `state`, `city`

`GET /positions/:ID`

Koncový bod pro získání podrobností o pracovním místě na základě ID.

`GET /positions/applicants/:ID/favorites`

Koncový bod pro získání uložených pracovních míst uchazeče na základě jeho ID. Je dostupný pouze pro administrátory a uchazeče s odpovídajícím ID.

`PUT /positions/applicants/:ID/favorites/:ID/add`

Přidání pracovního místa do seznamu uložených pracovních míst uchazeče.

`PUT /positions/applicants/:ID/favorites/:ID/remove`

Odstranění pracovního místa ze seznamu uložených pracovních míst uchazeče.

`PUT /positions/:ID`

Koncový bod pro aktualizaci pracovního místa na základě ID. Přístupný pro administrátory a společnosti, které vlastní dané pracovní místo.

`POST /positions`

Vytvoření nového pracovního místa. Nové pracovní místo může vytvořit společnost nebo administrátor.

`POST /positions/applicants/:ID/apply/:ID`

Vytvoření žádosti o pracovní místo na základě ID uchazeče a ID pracovního místa. Podání žádosti je možné pouze o aktivní nabídku.

Kategorie

`GET /job-categories`

Koncový bod pro získání všech kategorií.

`GET /job-categories/occupations`

Koncový bod pro získání všech povolání na základě vyhledávacího dotazu (URL parametr `q`).

`GET /job-categories/:ID`

Koncový bod pro získání všech povolání na základě ID kategorie.

`POST /job-categories/:NAME`

Vytvoření nové kategorie s daným jménem. Pouze pro administrátory.

`DELETE /job-categories/:ID`

Odstranění kategorie na základě jejího ID. Pouze pro administrátory.

D Obsah elektronického archivu

Elektronická příloha v archivu IS MU obsahuje zdrojové kódy aplikace, konfigurační soubor `docker-compose.yml`, soubor s OpenAPI specifikací (`swagger.json`) a vygenerovaný HTML klient `api.html`.