FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Egidio – platforma pro řešení katastrof
Student:	Filip Ballek
Vedoucí:	Ing. Marek Sušický
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Vlivem globálního oteplování se na Zemi děje stále více katastrof. Často se řeší, jak nabízet pomoc, jak ji přijmout, jak o ní napsat.
Cílem je navrhnout a implementovat platformu, ve které bude možné uživatelsky přívětivě nabídnout pomoc, napsat si o ní a také ji organizovat. Bude možné vytvořit „katastrofu" (projekt) a povolit určité typy pomoci. Bude možné zápisy sdílet na sociálních sítích a nabízející budou moci řídit úroveň zveřejněných osobních údajů. U „běžných" katastrof bude dostupná šablona toho, co je běžně potřeba. Například povodně – vysoušeče, ubytování, peníze, čerpadla…
Postup řešení:
Rešerše stávajících řešení.
Navrhněte vhodnou škálovatelnou architekturu řešení, diskutujte a vyberte vhodné technologie.
Implementujte a otestujte vzniklý webový portál.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 21. listopadu 2022 v Praze.

Bakalářská práce

EDIGIO – PLATFORMA PRO ŘEŠENÍ KATASTROF

Filip Ballek

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Marek Sušický
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat své rodině za podporu nejen při tvorbě této práce, ale během celého studia. Dále bych rád poděkoval vedoucímu své práce, Ing. Marku Sušickému, a také Adamovi Szabó, za pravidelné konzultace a další pomoc při tvorbě této práce. Zároveň bych chtěl poděkovat svým kolegům z Profinitu, kteří mi mnohokrát, při rozhovorech u šálku dobré kávy, poradili při řešení různých technický problémů. V neposlední řadě bych chtěl poděkovat Profinitu, přes který jsem se k práci dostal, a který mi jako zaměstnavatel časově vyšel vstříc tak, abych byl schopen práci včas dokončit.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. května 2023 .....

Abstrakt

Důvodem k vytvoření práce byla snaha pomoci lidem, kteří chtějí nabídnout či poptat pomoc během jedné z mnoha katastrof, které se na planetě Zemi každý rok dějí. V práci byla vytvořena platforma, která má potenciál učinit poskytování a poptávání pomoci efektivnější, než tak činí jiná řešení. Byl proveden průzkum podobných řešení, které jsou na internetu dostupné. Platforma byla následně navržena, implementována a otestována. Jako součást testování vzniklo 23 testovacích scénářů.

Klíčová slova webová platforma, backend, frontend, zprostředkování pomoci, katastrofa, OpenDataLab, Kotlin, Spring Boot, Angular, Docker, PostgreSQL

Abstract

The reason to create this solution was an effort to help people who want to offer their help or ask for a help during one of many catastrophes that happen on Earth every year. Platform, that has potential to make offering a help and asking for a help more efficient than other solutions do, was created. Research of similiar solutions that are available on the Internet was done. Then the platform was designed, implemented and tested. As a part of test process, 23 test scenarios were created.

Keywords web platform, backend, frontend, help mediation, catastrophe, OpenDataLab, Kotlin, Spring Boot, Angular, Docker, PostgreSQL

Seznam zkratek a pojmů

API	Aplication Programming Interface – Aplikační programové rozhraní
BE	Backend – Aplikace běžící na serveru
CORS	Cross-Origin Resource Sharing – Mechanizmus kontroly sdílení zdrojů
CSRF	Cross-Site Request Forgery - Typ útoku na uživatele
DTO	Data Transfer Object – Objekt pro přesun dat
FAQ	Frequently Asked Questions – Často kladené dotazy
FE	Frontend – Uživatelské rozhraní/webová stránka
HTTP	Hypertext Transfer Protocol – Protokol pro přenost hypertextu
IDE	Vývojové prostředí
JWT	JSON Web Token
MPSV ČR	Ministerstvo práce a sociálních věcí ČR
PP	Privacy policy – Zpracování osobních údajů
PPA	PP Agreement – Souhlas se zpracováním osobních údajů
Proxy	Prostředník
SSL	Secure Socket Layer – Zabezpečený protokol pro přenos dat
ToS	Terms of Services – Podmínky užívání služby
ToSA	ToS Agreement – Souhlas s ToS
UI	User Interface – Uživatelské rozhraní
UX	User Experience – Zážitek uživatele
XSS	Cross Site Scripting – Typ útoku na uživatele

Kapitola 1
Úvod

V posledních několika letech zasáhlo nejen Českou republiku, ale i celý svět mnoho katastrof. Příkladem budiž tornádo na Moravě, pandemie koronaviru, jejíž následky doznívají ještě dnes, nebo válka na Ukrajině. Při všech těchto událostech se mezi lidmi zvedla obrovská vlna solidarity, která bohužel nebyla nijak centrálně koordinována. Výsledkem bylo mnoho stránek či skupin na sociálních sítí, kde lidé pomoc nabízeli. To mohlo (a nejspíše mělo) za následek, že někde určitá pomoc scházela a jinde zase přebývala. Pokud by se podařilo, aby byly nabídky a poptávky nejen v rámci krajů, ale v rámci celé republiky více centrálně koordinované, bylo by možné dosáhnout efektivnějšího využití dostupných prostředků. To bylo hlavním důvodem k výběru tohoto tématu. Zaplnění tohoto místa na trhu by mohlo přinést vyšší efektivitu využití dostupných zdrojů a zvýšit šance, aby se pomoc dostala tam, kde je potřeba.

Cílem bakalářské práce je návrh a implementace platformy, na které budou moci lidé nabízet svou pomoc při řešení různých, nejen přírodních, katastrof a zasažení lidé si o ni budou moci požádat.

Bude zjištěno, jaká jsou existující řešení. U nich bude provedena analýza jejich vlastností a možností, jak se u nich inspirovat.

Následně budou analyzovány požadavky, které jsou na platformu kladeny ze strany zadavatele (OpenDataLabu). Některé z požadavků také vyplynou z analýzy stávajících řešení.

Na těchto požadavcích budou poté stavět další modely. Půjde například o model případů užití nebo o doménový model.

Následně bude navržena architektura aplikace tak, aby splnila vše, na co se přišlo v předchozích fázích, a zároveň byla snadno škálovatelná.

Řešení bude zahrnovat jak aplikaci na straně serveru (backend), tak aplikaci na straně uživatele (frontend). Obě tyto části budou implementovány a budou připraveny pro nasazení provozovatelem. Výplň některých částí, jako je ToS či PP budou ponechány pro doplnění provozovatelem.

Vyvíjená aplikace bude otestována, a to jak automatickými, tak manuálními testy. Výsledky testů budou následně okomentovány.

Nakonec budou diskutována možná vylepšení do budoucna.

Kapitola 2

Rešerše existujících řešení

2.1 Shrnutí

Prvním krokem k úspěšnému řešení každé práce je rešerše stávajících řešení. Bylo zjištěno, jaká řešení existují, jaké jsou jejich přednosti a naopak, jaké mají nedostatky. Cílem průzkumu byly především webové či mobilní aplikace, ale bylo nahlédnuto i do skupin existujících na sociálních sítích. Prozkoumána byla jak řešení v ČR, tak mezinárodní řešení.

Výsledkem tohoto průzkumu je, že pro řešení krizí a nabízení pomoci existuje mnoho prostorů. Většinou však nejsou nijak sjednocené pro více krizových událostí najednou. Pro jednu událost navíc existuje několik platforem či skupin na sociálních sítích, které o sobě nemusí nutně vědět. Výsledkem je, že není možné efektivně distribuovat pomoc na úrovni oblastí. Tato bakalářská práce tedy má potenciál, při dobré adaptaci veřejností, tyto díry zalepit a zefektivnit pomoc. Rozhodně by však bylo dobré uživatele upozornit i na další, již existující řešení, které mohou plnit komplementární funkci. Jednou z možností je například napojení na krizové centrum Facebooku. Díky tomu by měl uživatel aplikace možnost se spojit s ostatními lidmi zasaženými touto katastrofou, a třeba si navzájem vypomoci.

2.2 Pomáhej Ukrajině

Z existujících řešení mě nejvíce zaujala platforma Pomáhej Ukrajině, dostupná na adrese `https://www.pomahejukrajine.cz/`. Jde o platformu, která by měla sdružovat nabízení a využití pomoci Ukrajincům zasažených invazí Ruska na Ukrajinu. Na úvodní stránce je rozcestník kde si mohu vybrat, jestli pomoc nabízím, nebo jestli ji potřebuji.[1]

2.2.1 Nabídka

Nabídku lze zadat pomocí formuláře, který očekává informace jako je kontakt na inzerenta, kdy může být kontaktován, jakými jazyky hovoří, co nabízí nebo jaká je jeho odborná způsobilost. Část informací je povinná, některé však lze vynechat. Následně je ještě potřeba ověřit zadaný email, poté by nabídka měla být zveřejněna. [2, 3]

Dostupné jsou také **neveřejné nabídky**. *„Tyto nabídky jsou přístupné pouze registrovaným organizacím a institucím. Jsou ověřovány a jejich využití je koordinováno. Důvodem ověřování nabídek je bezpečnost příchozích“* [4].

2.2.2 Poptávka

Poptávka probíhá na stránce organizované formou jakéhosi dashboardu. Nabídky jsou formou karet, kde vidím informace k tomu, co je nabízeno. Na nabídku lze reagovat stiskem tlačítka „potřebuji tuto pomoc“, která se nalézá na spodní straně karty. Filtry nabízejí několik možností jak najít přesně to, co uživatel potřebuje, nebo alespoň tu nabídku, která je tomu nejblíže. [5]

2.2.3 Uživatelský účet

Na stránce je možné přihlásit se uživatelským účtem. Autor této bakalářské práce se domnívá, že jediným možným způsobem registrace je buďto přes tlačítko pro registraci organizace u neveřejné nabídky pomoci, nebo zasláním nabídky pomoci a ověřením emailu. Nikde nebyla nalezena jiná možnost, jak se zaregistrovat. K tomu, že se ověřením emailu vytvoří účet autor dospěl z toho, že dle FAQ si lze po ověření emailu a zveřejnění nabídky procházet a upravovat nabídky uživatele na svém uživatelském profilu [3].

2.3 Mapy na Mapotic

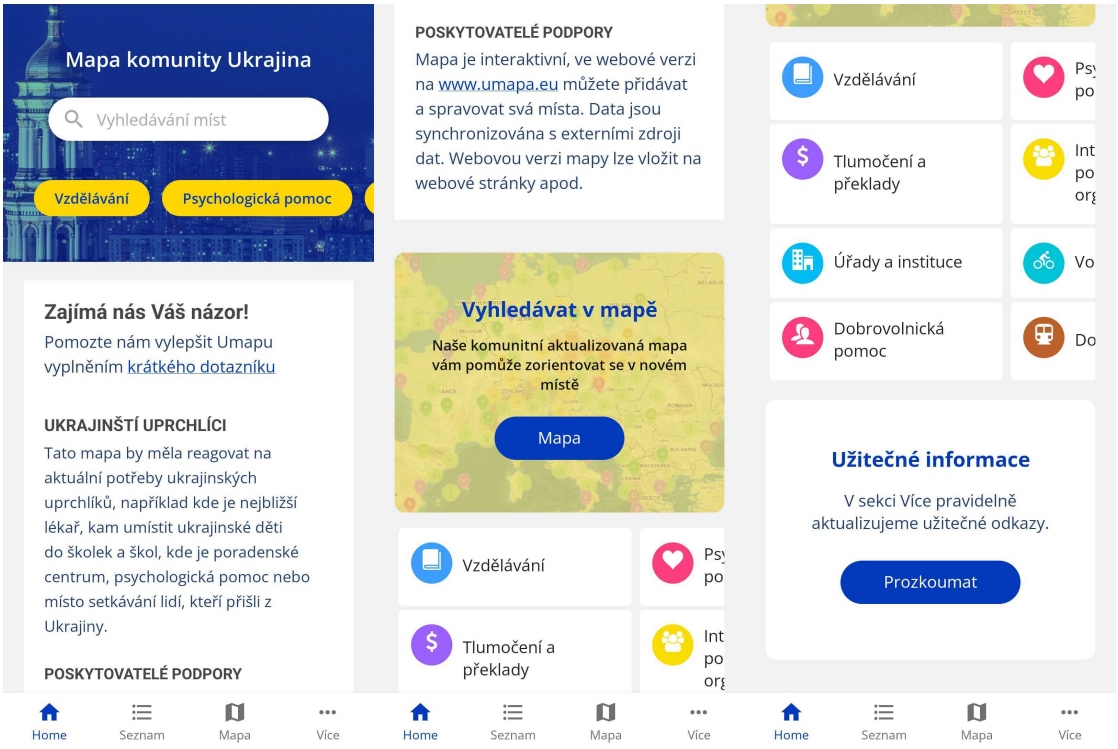
Mapotic je platforma s adresou `https://www.mapotic.com/`, která umožňuje vytvářet komunitní mapy s informacemi relevantními pro danou komunitu. [6] Mimo jiné je zde několik map vytvořených pro pomoc při různých krizových situacích (viz. podsekce 2.3.1 a podsekce 2.3.2).

2.3.1 Umapa

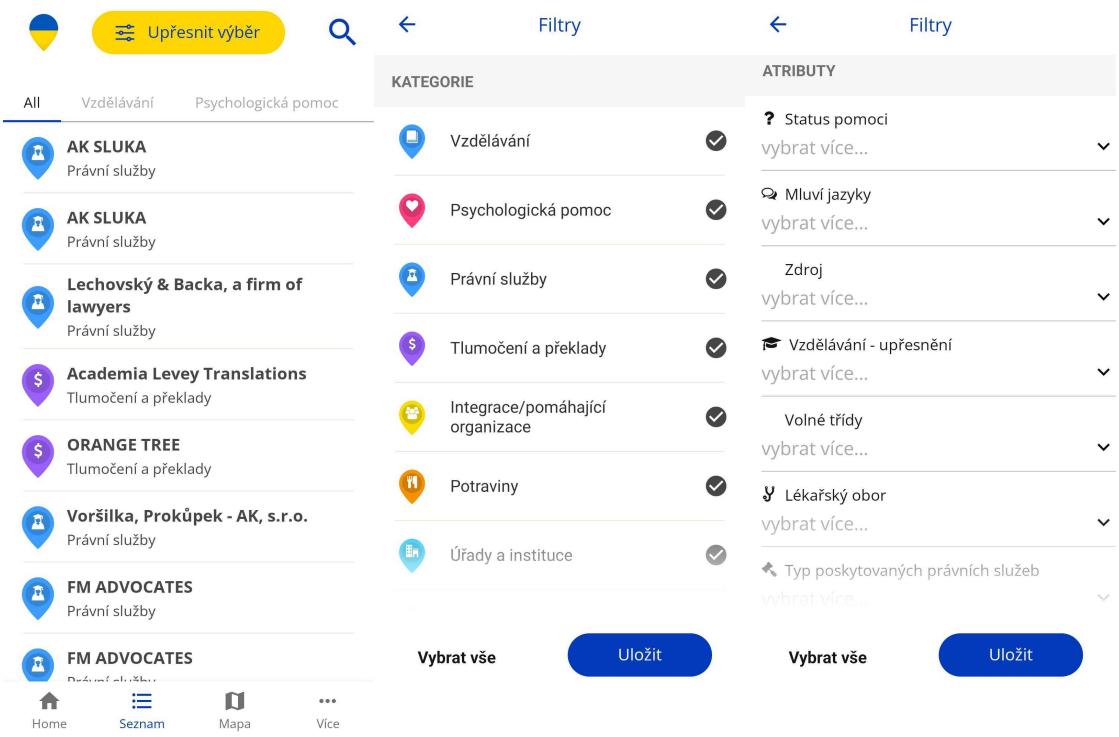
Umapa je portál pro pomoc Ukrajincům, kteří do České republiky prchají před válkou na Ukrajině. Jedná se o jednu z uživatelských map na platformě Mapotic. Vyvíjena je v rámci Cesko.digital. Nalézt ji lze přes odkaz `https://www.umapa.eu/`. Uživatelé mohou do této platformy volně zadávat místa, která by mohla být pro nově příchozí nějak zajímavá, nebo mohou upravovat existující místa. Jde například o možnosti právních služeb, místa kde mohou požádat o pomoc dobrovolníků, nebo potravinových bank, ve kterých si mohou zažádat o potraviny. Kde se jednotlivá místa nachází je zobrazeno na mapě. Zároveň je zde aktivní skupina správců a dobrovolníků, kteří se o platformu starají a postupně se ji snaží sami aktivně rozšiřovat o další data Při dostatečném oddálení se místa v určité oblasti zobrazí jako ikonka s počtem míst pro danou oblast. Během přibližování se rozpadají počty míst v dané oblasti na počty míst v podoblastech. Od určité chvíle už se pak začnou zobrazovat jednotlivá místa. Po rozkliknutí místa se zobrazí sekce s různými doplňujícími informacemi, včetně možnosti ho ohodnotit. Vytvořili si i svoji vlastní mobilní aplikaci, ve které si lze trochu přehledněji procházet jednotlivé položky. V aplikaci si je možné zobrazit nějaký rychlý úvod, kde se lze dozvědět o co v aplikaci jde (viz. obrázek 2.1). Dále jsou zde seznamy s nabídkami rozřazenými do kategorií (obrázek 2.2). Kromě toho je tu ještě samostatná mapa, která funguje zhruba stejně, jako webová aplikace (obrázek 2.3). [7, 8]

2.3.2 Mapa dobrovolníků

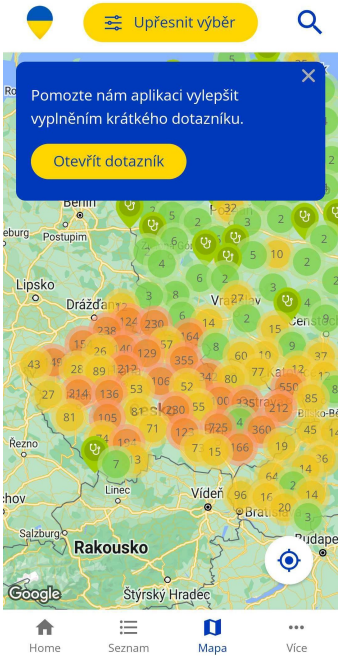
Mapa je věnována pomoci lidem zasaženým koronavirem. Dostupná je na adrese `https://www.mapotic.com/mapa-dobrovolniku`. Většina nabídek je na rozvoz, dovoz roušek, právní pomoc a tak podobně. Komunita již dle všeho není aktivní. V době psaní této rešerše byl poslední příspěvek přidaný tři roky zpátky. [9]



Obrázek 2.1 Mobilní aplikace Umapa – úvodní stránka [8]



Obrázek 2.2 Mobilní aplikace Umapa – seznam [8]



Obrázek 2.3 Mobilní aplikace Umapa – mapa [8]

2.3.3 FandiMat

Dle slov tvůrců *„FandiMat propojuje lidi v nouzi i charitativní organizace s lidmi a firmami, které mohou materiální pomoc poskytnout“* [10]. Adresa této platformy je `https://www.mapotic.com/fandimat`.
Nabídky a poptávky na této platformě jsou různorodé. Od venčení psů, asistence seniorům až po poptávku hraček pro děti, které by jinak nedostaly nic pod stromeček, jelikož se jejich matka samoživitelka ocitla v těžké finanční situaci. [10, 11]

2.4 Portál pomoci

Portál nabízející pomoc lidem zasaženým koronavirem. Je zde také možnost využití *chatu* – dopisování si s někým, nejspíše s člověkem, na druhé straně. Portál je hostovaný na adrese `https://portalpomoci.cz/`. Mají mnoho forem pomoci. Jmenovat lze například sousedskou pomoc, výpomoc s domácností nebo péči o zvířata. Inzerují také různé možnosti nákupu poukazů - zde jde především o pomoc subjektům, jejichž podnikání/živobytí bylo zasaženo koronavirem. Autorovým názorem je, že nákup poukazu pomůže danému subjektu a ještě tím lze udělat radost někomu dalšímu. Mimo této inzerce je zde ještě nabízena hmotná pomoc zdarma, která je buď formou daru, nebo výpůjček. Portál se již zdá být neaktivní, poslední příspěvek je zde z roku 2020. [12]

2.5 Naši Ukrajinci

Stránka s URL `https://www.nasiukrajinci.cz/cs` poskytující informace pro prchající Ukrajince a lidi mající zájem těmto lidem pomáhat. Lze zde dohledat například informace o tom co, kdy a kde je potřeba udělat, případně co zařídit. Informace o tom, kde najít práci a jak řešit víza či pobyty je zde možné rovněž najít. Pro uprchlíky může být pro prvotní orientaci také zajímavý

výpis zpravodajství v ukrajinštině. Pro lidi nabízející pomoc Ukrajincům stránky nabízejí rozcestník z něhož se mohou dostat na stránku Pomáhej Ukrajině. Pro pomoc většího rozsahu jsou zde kontakty na krajská asistenční centra, kam se lidé mohou obracet a nabídnout pomoc. [13]

2.6 Smart Migration

Aplikace provozovaná MPSV ČR, která obsahuje souhrn nejdůležitějších informací týkajících se života cizinců na území ČR. Její charakter je čistě informativní. [14] V aplikaci je dostupný chat. Podle všeho to ale vypadá, že odpovídá pouze robot. [15]

2.7 Jobs4UA

Stránka dostupná na `https://www.jobs4ua.cz/` na které probíhá nabídka práce Ukrajincům. Inzerenti jsou státem pověření zaměstnavatelé. Ve FAQ jsou mimo jiné dohledatelné informace o tom, co se kde musí hlásit, na co má kdo nárok a jak u nás fungují některé procesy, jako je poptávání práce, nebo založení živnosti. [16]

2.8 Dobrovolnik.cz

Web, provozovaný centrem pro dobrovolnictví, s adresou `https://www.dobrovolnik.cz/`. Na tomto webu se, mimo jiné, nachází stránka s nabídkami dobrovolnictví. Na ní jsou dostupné různé příležitosti, při kterých lze vypomáhat. Nabídky jsou různorodé. Dobrovolník může vypomáhat seniorům, ve zdravotnictví, nebo s fundraisingem a benefičními akcemi. [17] Jednou z nabídek je například pomoc dětem ze sociálně znevýhodněného prostředí. V detailu je motivace, proč příležitost využít a s čím dobrovolník, který ji využije, pomůže. Pro zájemce je k dispozici tlačítko využít příležitost, které uživatele přesměruje na přihlašovací stránku. [18, 19]. Registrovat se však autor nezkoušel.

2.9 Donio

Portál hostovaný na adrese `https://www.donio.cz/`, na kterém probíhají sbírky za různými účely. Část sbírek je komerčních a pomáhá lidem splnit si nějaký cíl, založit podnikání a tak podobně. Jsou zde ale i dobročinné sbírky zaměřené na pomoc lidem. Zaměření sbírek je primárně na jednotlivce. Najdou se ale i výjimky. Jako příklad lze použít sbírku na nákup pomůcek pro řidiče vozící zásoby na Ukrajinu [20], nebo nákup termo-dronu pro mediky na Ukrajině [21].

2.10 Ukraine Now

Stránka je dostupná na adrese `https://www.ukrainenow.org/`. Organizuje se zde mezinárodní pomoc Ukrajincům. Pro nabídku a poptávku jsou na stránce dostupné formuláře. U formulářů si uživatel může vybrat, jestli chce ukrajinskou nebo anglickou variantu. Ještě je zde v hlavičce stránky tlačítko pro přispění, přičemž dle autorů stránky jsou všechny příspěvky použity na evakuaci lidí z Ukrajiny, pomoc uprchlíkům a zajištění humanitární pomoci. [22, 23, 24, 25] Ještě je možné se zde dočíst, jaké jsou možnosti pomoci. [26]

2.11 Unified Volunteers Platform

Jde o nabídku pomoci dobrovolníků OSN. Pomáhají napříč celým světem. Platformu lze nalézt na adrese `https://app.unv.org/`. Při vstupu na hlavní stránku aplikace se zobrazí přehled projektů, do kterých se lze zapojit. Filtrovat nabídky lze například podle titulku, země, kde pomoc probíhá, kategorie dobrovolníka (dle OSN) nebo potřebné expertízy. [27] U každého projektu se po rozkliknutí zobrazí stránka s detaily. Detaily obsahují například to, co je cílem pomoci, jaké jsou požadované zkušenosti, věk nebo doba po kterou je pomoc potřeba. V patičce je odkaz, který v překladu říká „Staň se OSN dobrovolníkem“ [27]. Při rozkliknutí je uživatel přesměrován na přihlašovací stránku s možností prokliku na registraci [27, 28]. K tomu, co se děje po registraci, se však autorovi informaci dohledat nepodařilo.

2.12 Volunteer World

Web s vypsanými možnostmi dobrovolnických prací v různých koutech světa. Web lze nalézt na adrese `https://www.volunteerworld.com/en`. Jde o dlouhodobé projekty. Projekty jsou zaštiťovány různými organizacemi. Každý projekt má detailní popis toho, co od něj očekávat a na jaká úskalí se připravit. Uchazeč si může od lidí, co již projekt absolvovali, přečíst hodnocení. Zároveň si lze vybírat, od kdy a do kdy chce člověk projekt absolvovat. Od dobrovolníků vybírají organizace, které mají daný projekt na starosti, nějaké poplatky. V nabídkách bývá vypsané, co je součástí poplatků. Zároveň tam i bývá vypsané, co součástí poplatků není. Typicky je v něm zahrnuté jídlo (do určité míry), ubytování a občas i doprava z letiště. Naopak v něm nebývá zahrnutá letenka a cestovní pojištění, to si musím zajistit sám. Mimo to jde část poplatků na finanční pomoc dané organizaci, která většinou bývá nezisková a toto jí pomáhá uhradit náklady spojené s jejím provozováním (administrativa, IT atd.). [29, 30]

2.13 Skupiny na Facebooku

Pro lidi, kteří mají zájem pomoci, je na Facebooku mnoho skupin. Bohužel však velká část těchto skupin vypadá, jako by nebyla nijak, nebo jen mizerně, moderovaná. Seriózní příspěvky bývají proložené SPAMem, případně jinými nabídkami/poptávkami, které vypadají podezřele (dojem autora). Jedna ze skupin, která vypadá seriózně a udržovaně, je skupina Pomoc Ukrajině na adrese `https://www.facebook.com/groups/468032188137410`. Lidé si zde sdílejí nabídky pro prchající Ukrajince, případně žádosti o nějakou pomoc pro ně. Zároveň je zde mnoho žádostí o dovoz pomoci na Ukrajinu. I diskuze autorovi po pročtení připadá, že je většinově slušná. V době psaní této rešerše je skupina aktivní – poslední příspěvek je z předchozího dne. [31]

2.14 Krizové centrum na Facebooku

Služba Facebook Crisis Response (v češtině krizové centrum) má pomoci ve chvíli, kdy jsou lidé zasažení nějakou katastrofou. URL, na které se nachází, je `https://www.facebook.com/crisisresponse/?source=facebook_bookmark`. Dle Facebooku: *„V krizovém centru najdou lidé zasažení nějakou katastrofou možnost dát přátelům vědět, že jsou v pořádku, hledat nebo nabízet svoji pomoc a najdou v něm i informace o aktuálním dění“* [32]. Jednotlivé události jsou v krizovém centru založeny po tom, co je Facebook na krizovou situaci upozorněn některou z globálních agentur. Pokud se při některé z krizových událostí vyskytuje v zasaženém místě, je šance, že mu vyskočí upozornění, aby označil, že je v bezpečí. V nápovědě ke krizovému centru se ještě lze dočíst, jak se chovat, pokud uživatel reaguje na nějakou nabídku pomoci tak, aby se nevystavil většímu nebezpečí, než je nezbytně nutné. [33]

Kapitola 3
Analýza požadavků

Po zjištění, jaká řešení nyní existují, byl dalším krokem sběr požadavků, které jsou na aplikaci kladeny a jejich analýza. To probíhalo formou rozhovorů při společných schůzkách s vedoucím práce a s Adamem Szabó z OpenDataLabu[1]. Byla použita standardní kategorizace na funkční a nefunkční požadavky. Následovalo vytvoření modelu případů užití, ve kterém nejdříve byly identifikováni jednotlivý aktéři a následně byly vytvořeny jednotlivé případy užití, které rozvíjely funkční požadavky. Nakonec bylo zkontrolováno, že pro každý funkční model existuje nějaký případ užití, a zároveň že není případ užití, pro který by neexistoval funkční požadavek (viz. tabulka 3.1)

3.1 Funkční požadavky

Funkční požadavky vyjadřují, jaké funkce jsou od aplikace očekávány.

FR1 – Projekt pro každou katastrofu Pro každou katastrofu bude možné vytvořit jeden projekt. K dispozici bude jméno projektu, které by mělo co nejtrefněji vyjadřovat, jakou katastrofu zaštiťuje. Kromě toho bude evidován popis projektu lokalizovaný do podporovaných jazyků, datum jeho vytvoření a datum poslední úpravy. Projekt bude možné vytvořit před tím, než bude dostupný veřejnosti. Dále ho bude možné archivovat. Po archivaci už by projekt neměl být dále dostupný pro veřejnost. Pro vytváření projektu bude postačovat mít vytvořené API rozhraní, na které budou moci administrátoři a koordinátoři odesílat žádosti k vytvoření nového projektu. Zveřejněné projekty bude moci kdokoliv prohlížet a zobrazené projekty si bude moci filtrovat.

FR2 – Užitečné informace Ke každému projektu bude dostupný seznam užitečných informací. Užitečné informace bude stejně jako projekt možné zadat administrátorem nebo koordinátorem pomocí API.

FR3 – Evidence položek k vypsání V systému budou evidované položky, ze kterých se bude vybírat při přidávání položek do inzerátu či odpovědi. Položky budou mít vícejazyčný titulek a popis. Tyto informace půjde zobrazit společně s informacemi ke konkrétní vypsané položce u inzerátu.

FR4 – Šablony pro výčet položek inzerátu Pro každou katastrofu bude k dispozici seznam položek k nejčastěji vypisovaným typům inzerátů. Stejně jako pro inzerát bude postačovat, pokud bude k dispozici API endpoint, přes který bude možné seznam vytvářet.

[1] OpenDataLab - https://opendatalab.cz/

FR5 – Evidence inzerátů V systému bude evidována nabídka a poptávka pomoci (dále jen inzerát). Inzerát bude moci vytvořit kdokoliv, kdo přijde na stránku. Před zveřejněním bude nutné schválení administrátorem nebo koordinátorem. Evidovány budou informace o tom, kdo inzerát vytvořil, kdy ho vytvořil, kdo ho schválil a k jakému místu se inzerát vztahuje. Obsahem samotného inzerátu pak bude nějaký trefný titulek, popisek a požadované či poptávané položky. Titulek a popisek budou lokalizovatelné do jazyků, do kterých je přeložené uživatelské rozhraní. Položky v inzerátu budou vybírané ze zdrojových položek, které budou v systému předvyplněné. K dispozici bude popis položek, který bude taktéž lokalizovatelný do jazyků, do kterých je lokalizováno rozhraní aplikace. Zadavatel i administrátor musí mít možnost inzerát archivovat. Archivovaný inzerát už nesmí být zobrazován mezi vypsanými inzeráty, ale může být zobrazitelný přes přímý link, pokud to administrátor či koordinátor povolí. Archivované inzeráty již nebude možné znovu aktivovat. Zveřejněnými inzeráty bude možné listovat. List inzerátů bude možné filtrovat a upřesnit tím požadované vlastnosti hledaných inzerátů.

FR6 – Sdílení inzerátů a projektů Inzeráty a projekty bude možné sdílet emailem a na Facebooku. Při sdílení bude ve zprávě předvyplněný odkaz na projekt. Text si uživatel vyplní sám. Stejně tak si uživatel vyplní předmět emailu a cílovou adresu. Samotné odeslání emailu či vytvoření příspěvku nebude probíhat na platformě, ale bude probíhat přes externí službu (emailový klient, stránka pro sdílení příspěvků na FB…).

FR7 – Odpověď na inzerát Na inzerát bude opět moci zareagovat kdokoliv. Ve výchozím stavu by měly být předvyplněné přesně ty položky, na které se inzerát dotazuje (či které nabízí). Položky bude možné upravit, a to jak odebráním či snížením množství, tak přidáním nových položek či zvýšením množství. Popis položek v inzerátu, pokud bude zadán, bude jednojazyčný. Ten kdo vypsal inzerát, na nějž se odpověď vztahuje, bude moci nabídku/poptávku přijmout, nebo ji odmítnout. Po přijmutí by mělo dojít k uzavření inzerátu.

FR8 – Kontaktní formulář Uživatel bude mít možnost kontaktovat správce platformy pomocí kontaktní formuláře. Kontaktní formulář bude vyžadovat jméno uživatele a email. Dále bude moci uživatel zadat své příjmení a telefonní číslo.

FR9 – Lokalizace aplikace Rozhraní aplikace a některé její dynamické části musí být lokalizovatelné do více jazyků. Statické části stránky musejí být přeložené do češtiny a angličtiny, ale musí být možné snadno přidat i překlad do dalších jazyků.

FR10 – Evidence uživatelů Systém bude dělit uživatele do několika skupin. Každá skupina bude mít jiné možnosti a jiná oprávnění. Uživatelé se budou moci registrovat na naší stránku pomocí registračního formuláře. Při registraci povinně zadají uživatelské jméno (username), jméno, příjmení a email. Naopak nepovinné musí být telefonní číslo. Při registraci musí uživatel udělit svůj souhlas se zpracováním osobních údajů (PPA) a s podmínkami užívání služby (ToSA). Uživatel se bude moci přihlásit pomocí přihlašovacího formuláře. Bude si také moci resetovat heslo.

3.2 Nefunkční požadavky

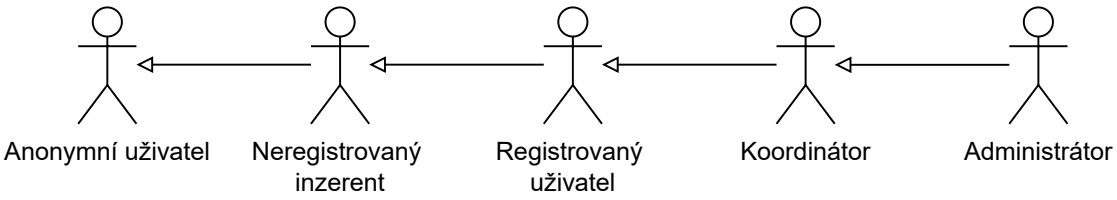
Nefunkční požadavky jsou takové, které jsou kladeny na vlastnosti aplikace.

NFR1 – Prezentace dat uživateli Uživateli budou data prezentována na webové stránce.

NFR2 – Responzivita stránky Stránka musí být responzivní, musí se dobře užívat jak na velkých monitorech, tak na malých mobilních zařízeních.

NFR3 – GDPR Aplikace bude dodržovat obecné nařízení o ochraně osobních údajů.

NFR4 – Šifrování komunikace Komunikace bude probíhat pouze přes šifrovaný protokol.



Obrázek 3.1 Aktéři v modelu

3.3 Model případů užití

Model případů užití slouží pro detailní popis funkčních požadavků. Skládá se z textů a diagramu, přičemž texty slouží pro popis jednotlivých případů užití a diagram pak pro přehledný náhled na to jaký aktér může vykonávat jednotlivé případ užití. Diagramy byly pro jednoduchost rozděleny do menších diagramů tak, že je jeden diagram pro každou podsekci. Zároveň jsou z diagramů vynecháni Ti aktéři, kteří přímo nevykonávají žádný případ užití. Důvodem bylo zpřehlednění diagramů.

3.3.1 Aktéři

Aktéři jsou entity, které systém nějakým způsobem užívají. V systému se vyskytuje pět aktérů tvořící hierarchii. Nabízelo se ještě vytvořit šestého aktéra, kterým by byl „neregistrovaný respondent“. Jednalo by se o neregistrovaného uživatele, který odpověděl na inzerát. V tuto chvíli však neexistují akce uvnitř našeho systému, které by byly přiřaditelné respondentovi (odeslání odpovědi je přiřazeno anonymnímu uživateli).

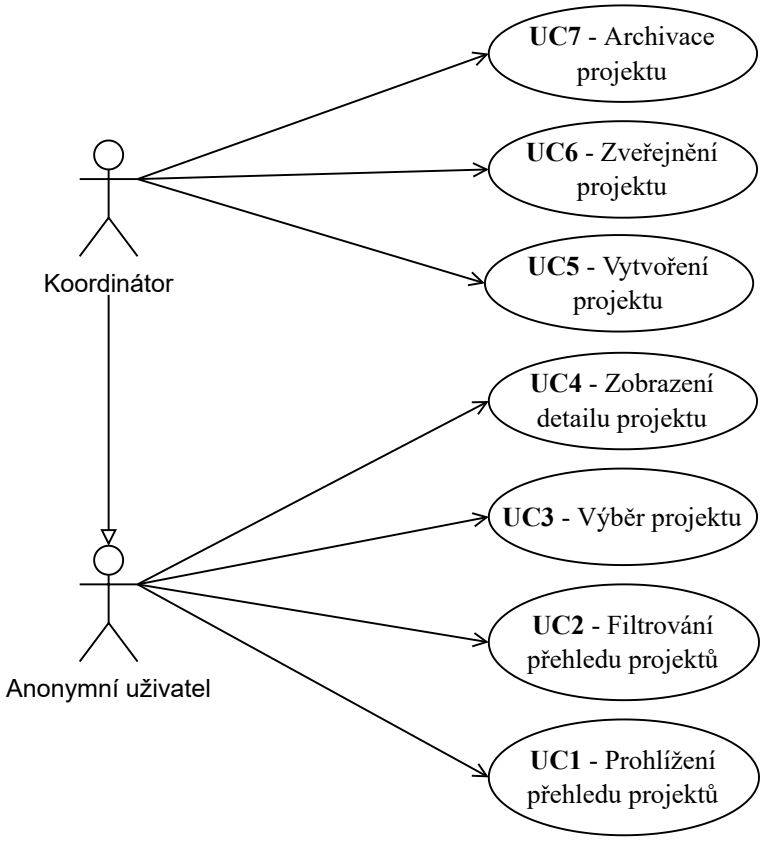
Anonymní uživatel Anonymním uživatelem se stává kdokoliv, kdo navštíví platformu a zároveň není přihlášený. O takovémto uživateli nemáme žádné údaje, a možnosti jeho aktivit jsou nejomezenější ze všech typů uživatelů. Může se registrovat, přihlásit se či si požádat o zaslání zapomenutého hesla od nějakého účtu. Dále může procházet zveřejněné projekty, zveřejněné inzeráty, na které může rovnou i odpovědět. Kromě toho může uživatel kontaktovat provozovatele přes kontaktní formulář.

Neregistrovaný inzerent Takovýto uživatelský status je typicky provázaný s právě jedním inzerátem, a to tím, který uživatel vytvořil (jehož je autorem). Má stejná práva jako anonymní uživatel, a zároveň ještě může archivovat inzerát a reagovat na odpovědi, které na inzerát přišly. Jednou z jeho aktivit je i zadání potvrzujícího kódu, kterým verifikuje svůj email.

Registrovaný uživatel Registrovaný uživatel má stejné možnosti, jako neregistrovaný inzerent a neregistrovaný respondent najednou. Dále má možnost samostatně upravovat své údaje a uživatelské nastavení.

Koordinátor Koordinátor je registrovaný uživatel, který zároveň může schvalovat a zamítat inzeráty, případně je archivovat. Navíc může přidávat projekty, zveřejňovat je či je archivovat. Kromě toho ještě může přidávat, upravovat či odebírat dostupné šablony a věci k zahrnutí do inzerátu. V neposlední řadě má možnost zablokovat uživatele.

Administrátor Administrátor je nejvýše postavený aktér. Může dělat vše, co mohou dělat předchozí zmínění aktéři. Od koordinátora se v tuto chvíli liší tím, že má možnost zablokovat uživatele a měnit jeho roli.



Obrázek 3.2 Diagram případů užití vztahujících se k projektům

3.3.2 Případy užití vztahující se k projektům

UC1 – Prohlížení přehledu projektů Uživatel přišel na stránku a prohlíží si, které inzeráty by ho zajímaly.

UC2 – Filtrování přehledu projektů Uživatel nechtěl prohlížet všechny projekty, ale rozhodl se počet zobrazených projektů redukovat nastavením vhodných filtrů.

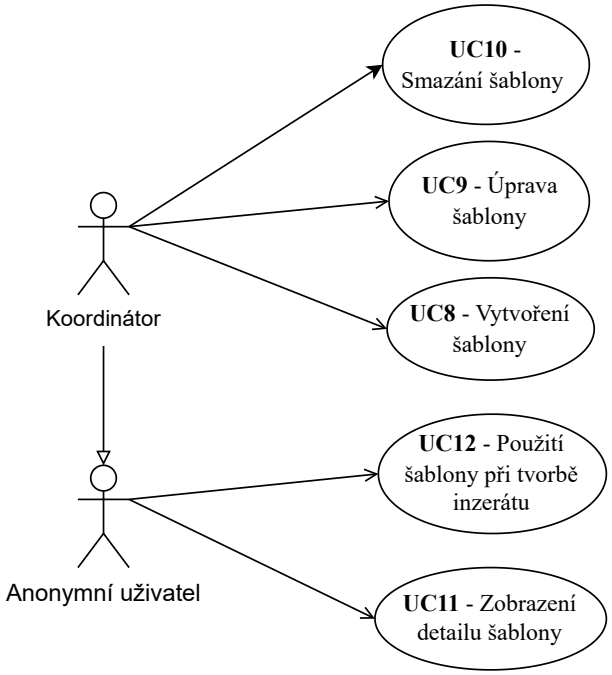
UC3 – Výběr projektu Uživatel si vybral projekt, se kterým chce dále pracovat. Nyní může procházet i sekce platformy, které jsou omezeny nutností mít vybraný projekt.

UC4 – Zobrazení detailu projektu Případ užití vyjadřuje situaci, kdy si uživatel vybral projekt, o kterém by se chtěl dozvědět více, a nyní si prohlíží stránku s detaily a užitečnými informacemi.

UC5 – Vytvoření projektu Koordinátor došel k závěru, že existuje vhodná katastrofa, pro kterou na stránce neexistuje projekt. Rozhodl se tedy projekt založit voláním API.

UC6 – Zveřejnění projektu Koordinátor chce zveřejnit projekt. Zašle požadavek na API a projekt bude nově dostupný veřejnosti.

UC7 – Archivace projektu Projekt již není relevantní. Koordinátor ho chce deaktivovat zasláním požadavku na API.



Obrázek 3.3 Diagram případů užití vztahujících se k šablonám

3.3.3 Případy užití vztahující se k šablonám inzerátů

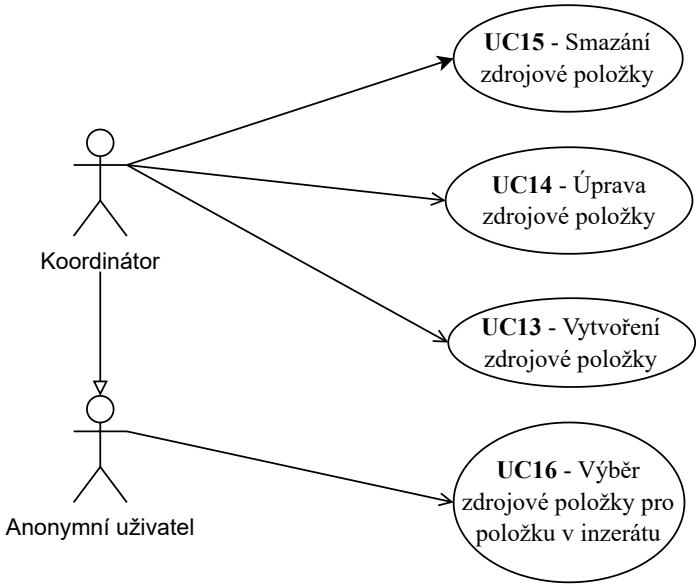
UC8 – Vytvoření šablony Koordinátor se rozhodl doplnit šablonu pro tvorbu inzerátů. Zavolal API s potřebnými daty a šablona byla vytvořena.

UC9 – Úprava šablony K nějaké šabloně jsou nepřesné detaily, je potřeba přidat do ní další položky, případně ji jinak upravit. Koordinátor toto provede voláním API.

UC10 – Smazání šablony Šablona se prokázala být nepotřebnou, nebo je nějaký jiný důvod k jejímu smazání. Koordinátor šablonu smaže voláním API.

UC11 – Zobrazení detailu šablony Uživatel se chce o šabloně dozvědět více, takže si rozklikne detail, kde se k ní dozví více informací.

UC12 – Použití šablony při tvorbě projektu Uživatel se rozhodl, že daná šablona je vhodná pro použití k předvyplnění položek zahrnutých v inzerátu. Nechá si tedy předvyplnit položky, které se k ní vztahují.



Obrázek 3.4 Diagram případů užití vztahujících se ke zdrojovým položkám

3.3.4 Případy užití vztahující se ke zdrojovým položkám

UC13 – Vytvoření zdrojové položky Administrátor či koordinátor došel k názoru, že na stránce chybí nějaká položka. Přidává položku voláním API.

UC14 – Úprava zdrojové položky Některému zdroji chybí nějaká informace, nebo je nepřesná. Administrátor nebo koordinátor se tedy rozhodl zdroj upravit.

UC15 – Smazání zdrojové položky Administrátor či koordinátor chce smazat nějaký zdroj pro položky. Pokud zdroj není použitý v žádné šabloně, inzerátu, ani odpovědi, je mu to umožněno.

UC16 – Výběr zdrojové položky pro položku v inzerátu Uživatel při přidávání položky do inzerátu vybral vhodnou zdrojovou položku.

3.3.5 Případy užití vztahující se k inzerátům

UC17 – Sdílení inzerátu Uživatel chce sdílet příspěvek emailem nebo přes Facebook.

UC18 – Vytvoření inzerátu Případ užití umožňuje uživateli vytvořit inzerát, ve kterém by si rád požádal o pomoc, nebo pomoc nabídl. Inzerát vytváří pro jeden vybraný projekt. V inzerátu vyplňuje všechny potřebné informace, aplikuje šablonu a vybírá zahrnuté položky.

UC19 – Archivace inzerátu Uživatel či koordinátor (případně administrátor) se rozhodl inzerát archivovat. Důvodem může být například to, že inzerátu bylo již vyhověno, nebo již není relevantní. Při archivaci jsou všechny aktivní odpovědi označené jako zamítnuté.

UC20 – Schválení inzerátu Administrátor nebo koordinátor se rozhodl, že je obsah inzerátu adekvátní a je možné ho schválit.

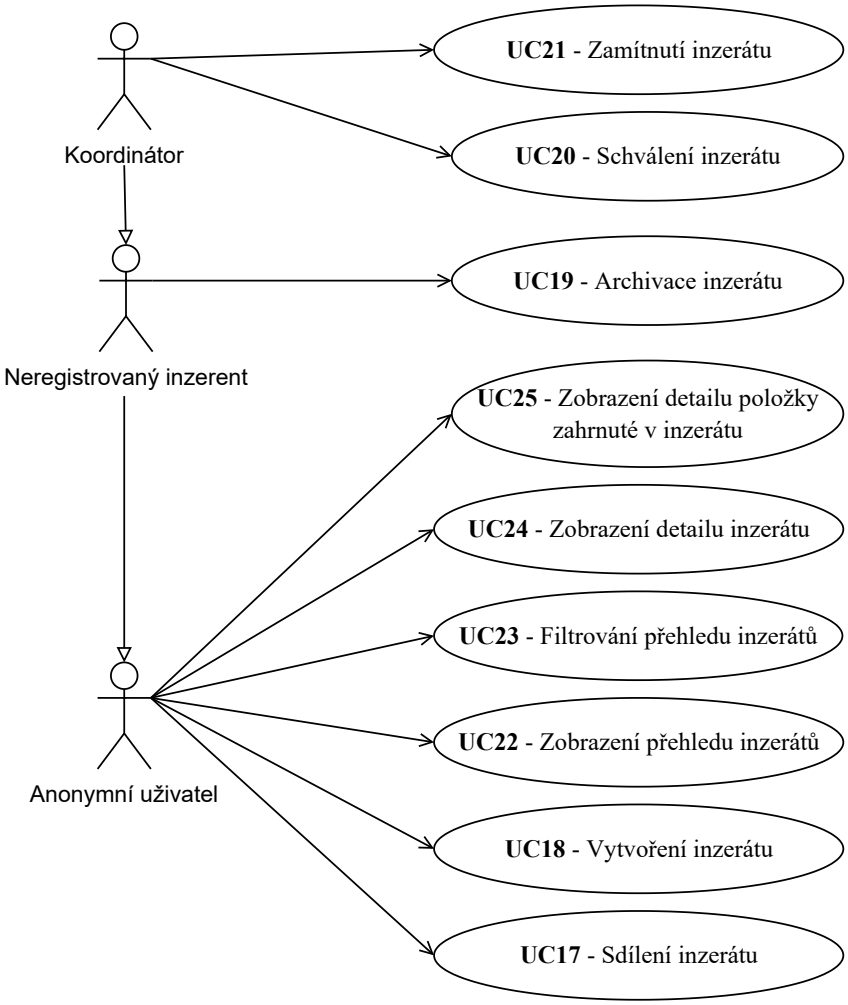
UC21 – Zamítnutí inzerátu Inzerát z nějakého důvodu není vhodné pustit na veřejnost a umožnit reakce na něj. Administrátor či koordinátor se rozhodl, že inzerát zamítne.

UC22 – Zobrazení přehledu inzerátů Uživatel se rozhodl, že si chce prohlídnout dostupné inzeráty.

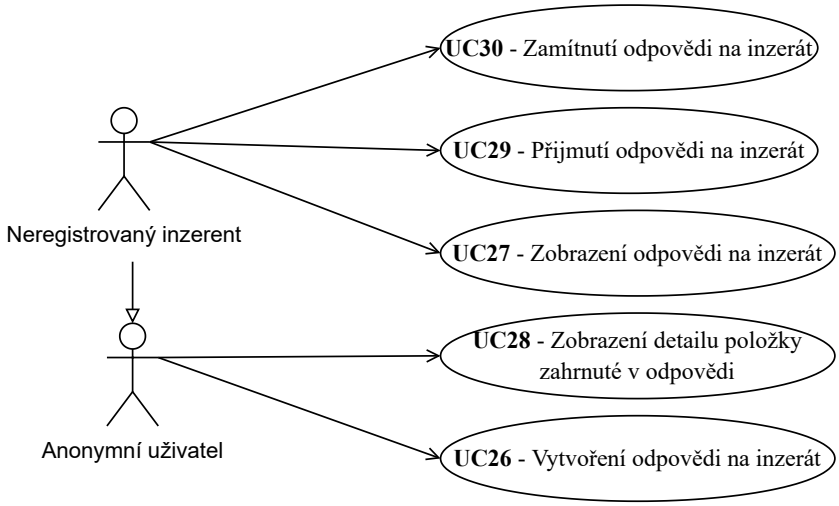
UC23 – Filtrování přehledu inzerátů Při prohlížení inzerátů si uživatel nastavil filtry, podle kterých by chtěl redukovat počet dostupných inzerátů.

UC24 – Zobrazení detailu inzerátu Uživatel si z nabídky inzerátů, nebo z jiného místa, vybral nějaký inzerát, o kterém by chtěl zjistit více. Proklikl se tedy na stránku kde se mu zobrazí detailní informace inzerátu

UC25 – Zobrazení detailu položky zahrnuté v inzerátu Uživatele při prohlížení detailu inzerátu, nebo při vytváření inzerátu zajímá více informací o vypsané položce. Rozklikl si ji aby se dozvěděl více nejenom o té konkrétní zahrnuté instanci dané věci, ale i o tom, jaký zdroj obaluje.



Obrázek 3.5 Diagram případů užití vztahujících se k inzerátům



Obrázek 3.6 Diagram případů užití vztahujících se k odpovědi na inzerát

3.3.6 Případy užití týkající se odpovědi na inzerát

UC26 – Vytvoření odpovědi na inzerát Uživatele nějaký inzerát zaujal natolik, že se rozhodl na něj odpovědět. Vyplní tedy formulář, který už byl částečně předvyplněný, a odešle ho. Nakonec ještě potvrdí svůj email kódem v emailové zprávě.

UC27 – Zobrazení odpovědi na inzerát Přišla odpověď na inzerát. Uživatel byl notifikován emailem a přes link se proklikl na stránku s náhledem odpovědí a možností ji přijmout či odmítnout.

UC28 – Zobrazení detailu položky zahrnuté v odpovědi Při prohlížení odpovědi se inzerent rozhodl, že by se chtěl dozvědet více o nějaké zahrnuté položce. Rozklikl si ji a zobrazil si detail. Další možností je, že si detail prohlíží respondent před odesláním odpovědi.

UC29 – Přijmutí odpovědi na inzerát Uživateli došla odpověď, kterou se rozhodl přijmout. Při přijmutí mohl uvést poznámku o tom, jak se s respondentem dále spojí.

UC30 – Zamítnutí odpovědi na inzerát Uživateli došla odpověď na inzerát, kterou se ale rozhodl neakceptovat. Při odeslání reakce mohl vyplnit poznámku o tom, proč se odpověď rozhodl zamítnout.

3.3.7 Případy užití vztahující se k uživateli

UC31 – Přepnutí jazyka UI Uživateli nevyhovuje aktuálně vybraný jazyk. Přepnul si tedy jazyk rozhraní a textů do jednoho z nabízených jazyků, který mu vyhovuje více.

UC32 – Registrace uživatele Uživatel se chce stát registrovaným uživatelem. Vyplní registrační formulář, potvrdí email a stane se registrovaným uživatelem.

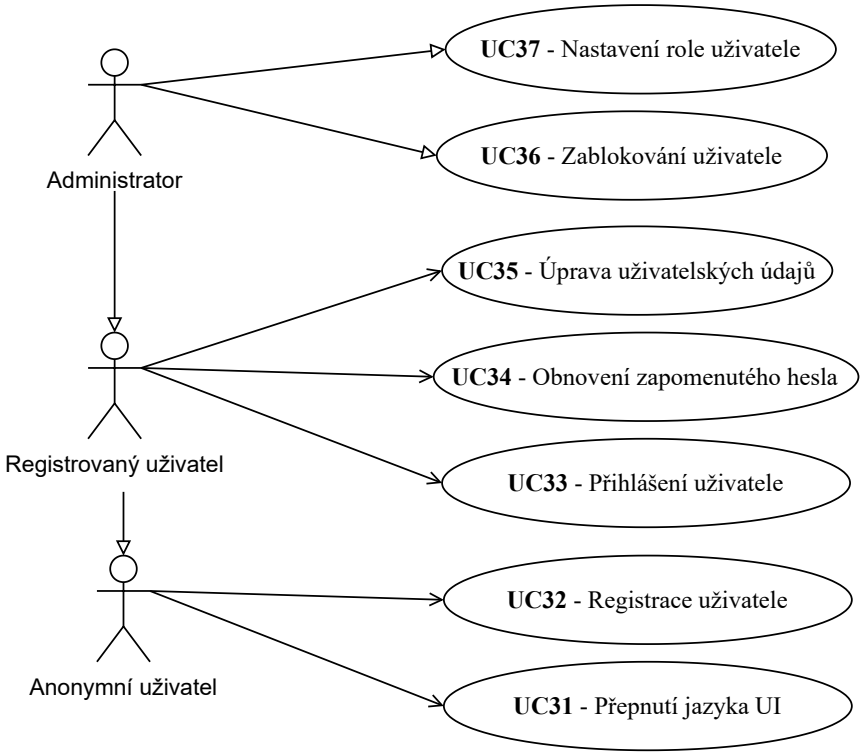
UC33 – Přihlášení uživatele Uživatel je registrovaný a chce se přihlásit. Vyplní přihlašovací údaje a odešle je na server.

UC34 – Obnovení zapomenutého hesla Uživatel zapomněl své heslo, případně chce heslo resetovat (jelikož v této fázi nemáme implementovanou samostatnou změnu hesla). Vyplnil svůj email a došla mu zpráva s odkazem na stránku pro vyplnění hesla.

UC35 – Úprava uživatelských údajů Registrovaný uživatel chce upravit nějakou část svých uživatelských údajů. Vyplní formulář na stránce, provede potřebná potvrzení a odešle.

UC36 – Zablokování uživatele Nějaký uživatel na stránce dělá problémy. Administrátor se rozhodl, že uživateli zablokuje přístup na stránku.

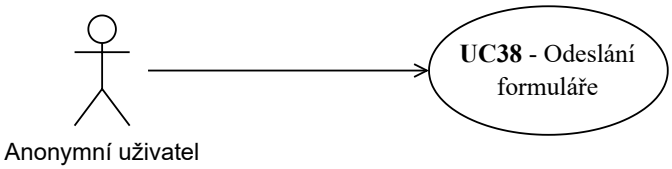
UC37 – Nastavení role uživatele Případ užití, který umožňuje administrátorovi změnit roli uživatele v systému na více či méně privilegovanou. Tuto změnu provede příslušným voláním API.



Obrázek 3.7 Diagram případů užití vztahujících se k uživateli

3.3.8 Případy užití pro kontaktní formulář

UC38 – Odeslání formuláře Uživatel vyplnil kontaktní formulář a odeslal ho.



Obrázek 3.8 Diagram případů užití vztahujících se ke kontaktnímu formuláři

Tabulka 3.1 Tabulka realizace funkčních požadavků jednotlivými případy užití

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10
UC1	✓									
UC2	✓									
UC3	✓									
UC4	✓	✓								
UC5	✓									
UC6	✓									
UC7	✓									
UC8				✓						
UC9				✓						
UC10				✓						
UC11				✓						
UC12				✓	✓					
UC13			✓							
UC14			✓							
UC15			✓							
UC16			✓							
UC17						✓				
UC18					✓					
UC19					✓					
UC20					✓					
UC21					✓					
UC22					✓					
UC23					✓					
UC24					✓					
UC25			✓		✓					
UC26							✓			
UC27				✓			✓			
UC28			✓				✓			
UC29							✓			
UC30							✓			
UC31									✓	
UC32										✓
UC33										✓
UC34										✓
UC35										✓
UC36										✓
UC37										✓
UC38								✓		

Kapitola 4
Návrh

4.1 Doménový model

4.1.1 Shrnutí

Byl vytvořen model, který zachycuje všechny doménové objekty projektu. Přítomné objekty vyplynuly především z analýzy funkčních požadavků. Z tohoto modelu následně vychází databázový model a částečně také členění částí aplikace. Model, a tudíž i názvy doménových objektů, jsou dělané v angličtině, aby bylo možné tento model přímo namapovat dále na kód a databázový model. Rychlý přehled lze vidět na obrázku 4.5.

4.1.2 Advertisement

Jedná se o entitu, která představuje jednu nabídku/poptávku (inzerát). Má svůj titulek a popis, přičemž ve výsledné aplikaci budou oba vícejazyčné. Dále má datum vytvoření a datum poslední úpravy. Součástí entity jsou i dva tokeny, přičemž jeden slouží pro označení inzerátu za vyřešený a druhý pro uzavření inzerátu (viz. sekce s expirujícími tokeny). Každý inzerát má svůj stav, přičemž přechody z jednoho stavu do druhého jsou znázorny v diagramu 4.1. Možné stavy inzerátu jsou následující:

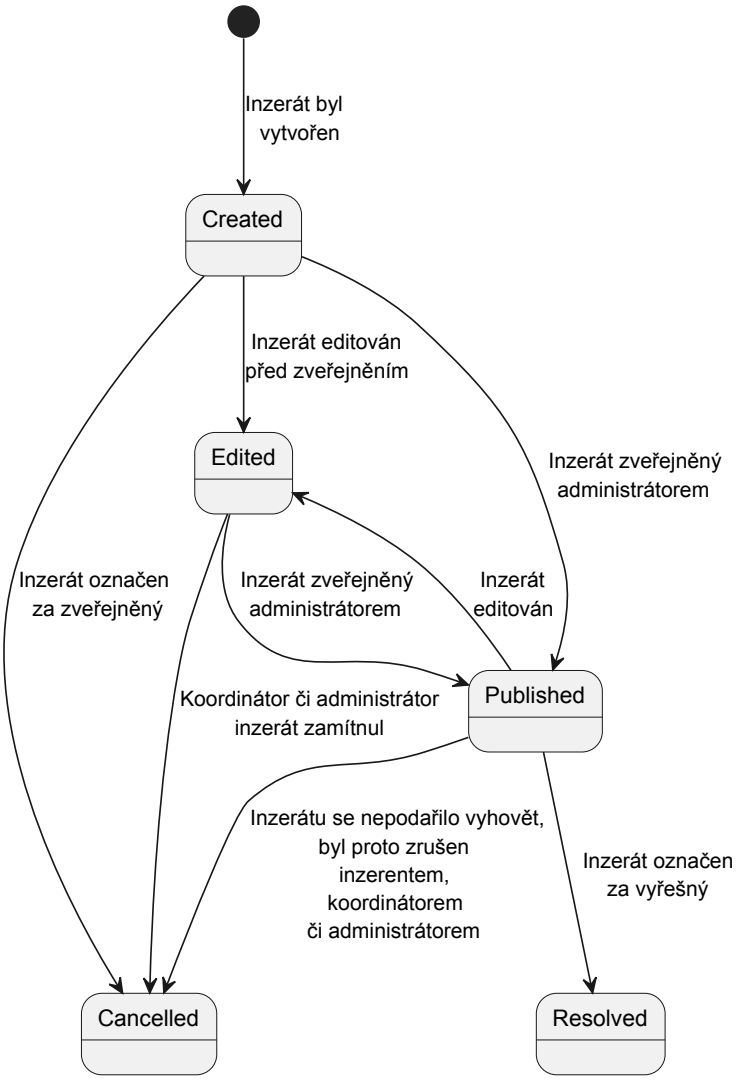
Created Inzerát byl vytvořen, ale ještě nebyl zveřejněn a jeho obsah nebyl editován

Edited Inzerát byl upraven, ale ještě nebyl uzavřen ani vyřešen. Mohl však již být před tím zveřejněný.

Published Autor inzerátu má potvrzený kontakt. Inzerát byl navíc schválený koordinátorem či administrátorem. Nyní čeká na vyřešení či uzavření.

Closed Inzerát byl uzavřený. Jednou z příčin může být rozhodnutí koordinátora či administrátora, že inzerát není možné schválit. Další možností je rozhodnutí autora, že si nakonec nepřeje inzerát vyřešit.

Resolved Autor uznal, že inzerát je již vyřešený. Kromě toho je ještě k dispozici informace o tom, jestli se jedná o nabídku či poptávku a také o tom, jaký typ pomoci je v inzerátu obsažený (materiální pomoc, psychologická pomoc…).



Obrázek 4.1 Diagram přechodů mezi stavy inzerátu

4.1.3 Location

Pro každý inzerát je nastavena právě jedna lokace, které se inzerát týká. Lokace se skládá ze státu, kraje, města, ulice, čísla domu a poštovního směrovacího čísla. Důvod pro existenci lokace jako samostatné doménové entity je takový, že jedním z budoucích vylepšení výsledků této práce by měla být i možnost znovupoužití jednotlivých adres nejen pro inzeráty.

4.1.4 Resource

Zdroj pro položky v inzerátech a odpovědích. Může jít jak o nějaký materiál, tak například o lidskou práci. Má pouze svoje jméno a popis, přičemž oboje je ve více jazycích.

4.1.5 Advertisement Item

Jedna položka v inzerátu. Obaluje nějaký zdroj. Je u ní uváděno množství (může se tedy jednat o více kusů obalovaného zdroje). Ještě má svůj popis. Ten by měl více upřesňovat, co přesně položka vyjadřuje. I tento popis je ve více jazycích. Takovouto položkou může být například nákladová dodávka s množstvím dva a popisem vyjadřujícím potřebnou kapacitu jedné dodávky.

4.1.6 Advertisement Response

Odpoveď na inzerát je reprezentována touto entitou. Obsahuje poznámky inzerenta a respondenta, přičemž obě budou pouze v jednom jazyce. Důvodem je to, že narozdíl od inzerátu zde není široké množství lidí, pro které jsou tyto poznámky mířené. Lokalizace by tedy v tomto případě úplně postrádala smysl. Dále má políčka pro nastavení data vytvoření a vyřešení. Za vyřešení se považuje buďto přijmutí nabídky/poptávky v odpovědi, nebo její zamítnutí. Pro přístup budou podobně jako u inzerátů uložené dva tokeny. *Preview token*, který slouží pro náhled (neumožňuje však vyřešení), a *resolve token*, který umožňuje i vyřešení odpovědi, a který by měl mít k dispozici pouze autor inzerátu pro který byla odpověď vytvořena. Je zde ještě stav odpovědi, který může nabývat následujících hodnot (diagram přechodu mezi těmito stavy je na obrázku 4.2):

Waiting For Contact Confirmation Odpověď čeká na to, až bude mít zadavatel potvrzený kontakt. V tuto chvíli ještě není dostupný ani inzerentovi ani respondentovi.

Waiting For Resolve Odpověď čeká na rozhodnutí inzerenta či koordinátora/administrátora, jak si ji přeje vyřešit.

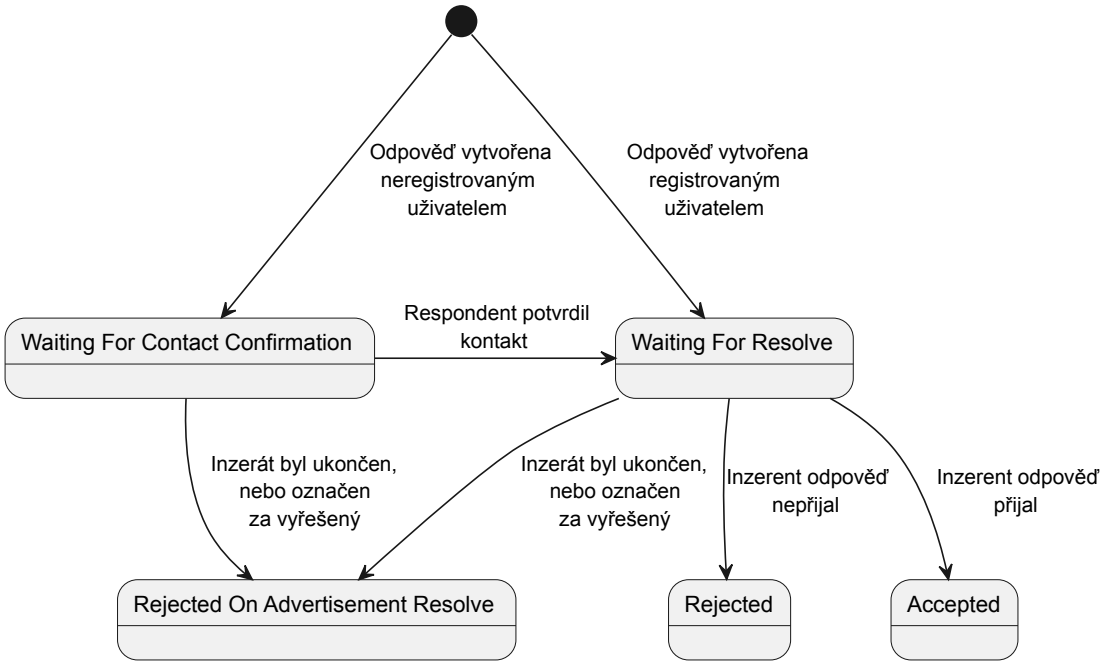
Rejected Inzerent, koordinátor nebo administrátor odpověď zamítl.

Accepted Inzerent, koordinátor nebo administrátor odpověď přijmul.

Rejected On Advertisement Resolve Inzerát byl vyřešen, ale odpověď byla v tu chvíli nevyřešená. Při vyřešení inzerátu došlo k automatickému odmítnutí odpovědi.

4.1.7 Response Item

Položka v odpovědi na inzerát. Stejně jako *Advertisement item*, i tato entita obaluje nějaký zdroj a skládá se z množství a popisu. Rozdílem je, že popis je ukládán v pouze jednom jazyce. Zdůvodnění je stejné jako u odpovědi na inzerát.



Obrázek 4.2 Diagram přechodů mezi stavy odpovědi na inzerát

4.1.8 Advertisement Template

Dle požadavků má být pro každou katastrofu k dispozici vzor, podle kterého budou moci být vytvářeny nové inzeráty. Tato entita zaštiťuje takovýto vzor. Má vícejazyčný text a popis. Vzor v tuto chvíli říká pouze to, které zdroje jsou využívané (bez množství). Ke vzoru je informace o tom, pro jaké typy katastrof, pomoci a inzerátu může být použitý. Dále je k dispozici konkrétní projekt, pro který může být vzor použitý.

4.1.9 User

Každý uživatel má jméno a příjmení. Kontakt na uživatele se skládá z emailu a telefonního čísla. U emailu je navíc uváděno, jestli byl již potvrzený. Dále je pro každého uživatele k dispozici doba, kdy byl registrován (v případě neregistrovaného uživatele doba, kdy byl vytvořen) a kdy byl email potvrzený.

Každý uživatel má nějakou roli, přičemž u neregistrovaného uživatele je role vždy *Non-registered user*. Další role jsou:

User běžný registrovaný uživatel.

Coordinator uživatel, který navíc může koordinovat pomoc - spravovat inzeráty, projekty...

Administrator správce aplikace. Má práva na všechny akce, které jsou v doméně dostupné.

V rámci této práce jsou práva administrátora a koordinátora stejné. Do budoucna se však počítá s tím, že se budou alespoň mírně lišit, a tak byly raději rozděleny rovnou.

Dalším atributem jsou jazyky, které uživatel ovládá. Ty slouží především k tomu, aby ostatní uživatelé věděli, jakým jazykem mohou s uživatelem komunikovat (vhodné především při odpovědi na inzerát).

Jedním z požadavků byla také možnost uživatele určit si, jaké detaily jsou zveřejněné. I to je součástí této doménové entity. Konkrétně má ukazatele toho, zda je zveřejněné jméno, příjmení, email a telefonní číslo.

Pro registrovaného uživatele je navíc ještě k dispozici jeho uživatelské jméno a to, jestli byl jeho účet zamknutý.

4.1.10 Email Change Request

Změna emailové adresy musí být vždy potvrzena uživatelem. Navíc je nutné potvrdit vlastnictví nové emailové adresy. Pro tento účel byl vytvořen koncept žádosti o změnu. Žádost obsahuje dva konfirmační tokeny. Jeden slouží pro potvrzení že si žádost vyžádal uživatel a bude odesílán na původní emailovou adresu. Druhý slouží pro potvrzení vlastnictví nové emailové adresy a bude tedy odeslán na tuto adresu. Dále obsahuje čas, kdy byla žádost vytvořena a kdy byla ukončena. S tím souvisí stavy žádosti. Možné stavy jsou vypsány v následujícím výčtu. Diagram stavů žádosti lze vidět na obrázku 4.3.

Active Žádost o změnu byla vytvořena a čeká na konfirmaci.

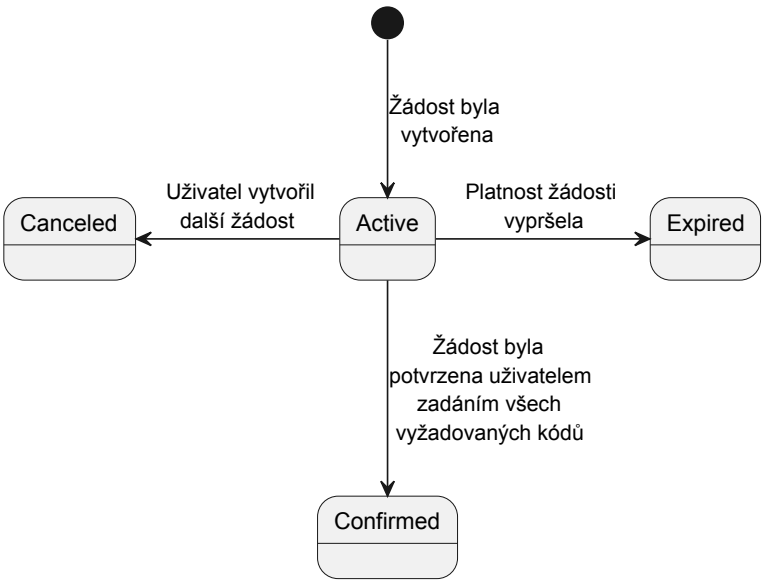
Confirmed Žádost o změnu byla potvrzena.

Expired Žádost o změnu expirovala.

Canceled Žádost o změnu byla zrušena.

4.1.11 Telephone Number Change Request

Podobně jako u změny emailové adresy je i při změně emailu nutné potvrdit žádost uživatelem. Není nutné potvrzovat vlastnictví nového telefonního čisla. I zde je tedy využit koncept žádosti o změnu, tentokrát však pouze s jedním konfirmačním tokenem odesílaným na emailovou adresu uživatele. Vše ostatní zůstává stejné jako u žádosti o změnu emailové adresy.



Obrázek 4.3 Diagram přechodů mezi stavy žádosti o změnu

4.1.12 Project

Projekt organizovaný pro nějakou katastrofu. Je k dispozici titulek a popis projektu, přičemž stejně jako u inzerátu jsou i zde oba texty zadávané ve více jazycích. Dále je jeho součástí typ katastrofy (viz. tabulka 4.1), ke které se projekt vztahuje, doba, kdy byl vytvořený, upravený, publikovaný a archivovaný. K projektu je vždy přiřazený uživatel, který ho vytvořil. Volitelně dále může být přiřazen uživatele, který ho upravil, zveřejnil, nebo archivoval. Každý projekt má navíc stav, ve kterém se zrovna nachází. Stavy jsou rozepsané níže. Rychlý přehled lze vidět na obrázku 4.4.

Prepared Projekt je připraven, ale zatím ještě není veřejně dostupný. Může to být například z toho důvodu, že se čeká na doplnění nějakých dalších informací.

Published Projekt je zveřejněn, mohou na něj být vytvářené nové inzeráty a na ně mohou být vytvářeny odpovědi.

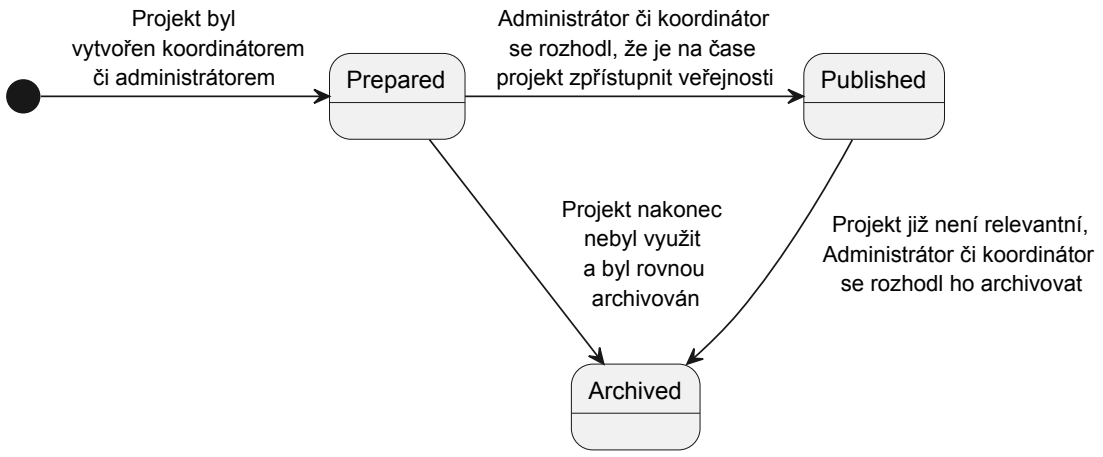
Archived Projekt je archivován. Jde tedy především o historické projekty.

Projekt může mít přiřazenou jednu nebo více informací, které mohou být důležité pro někoho zasaženého katastrofou, nebo někoho, kdo chce o katastrofě zjistit více (viz. podsekce 4.1.13).

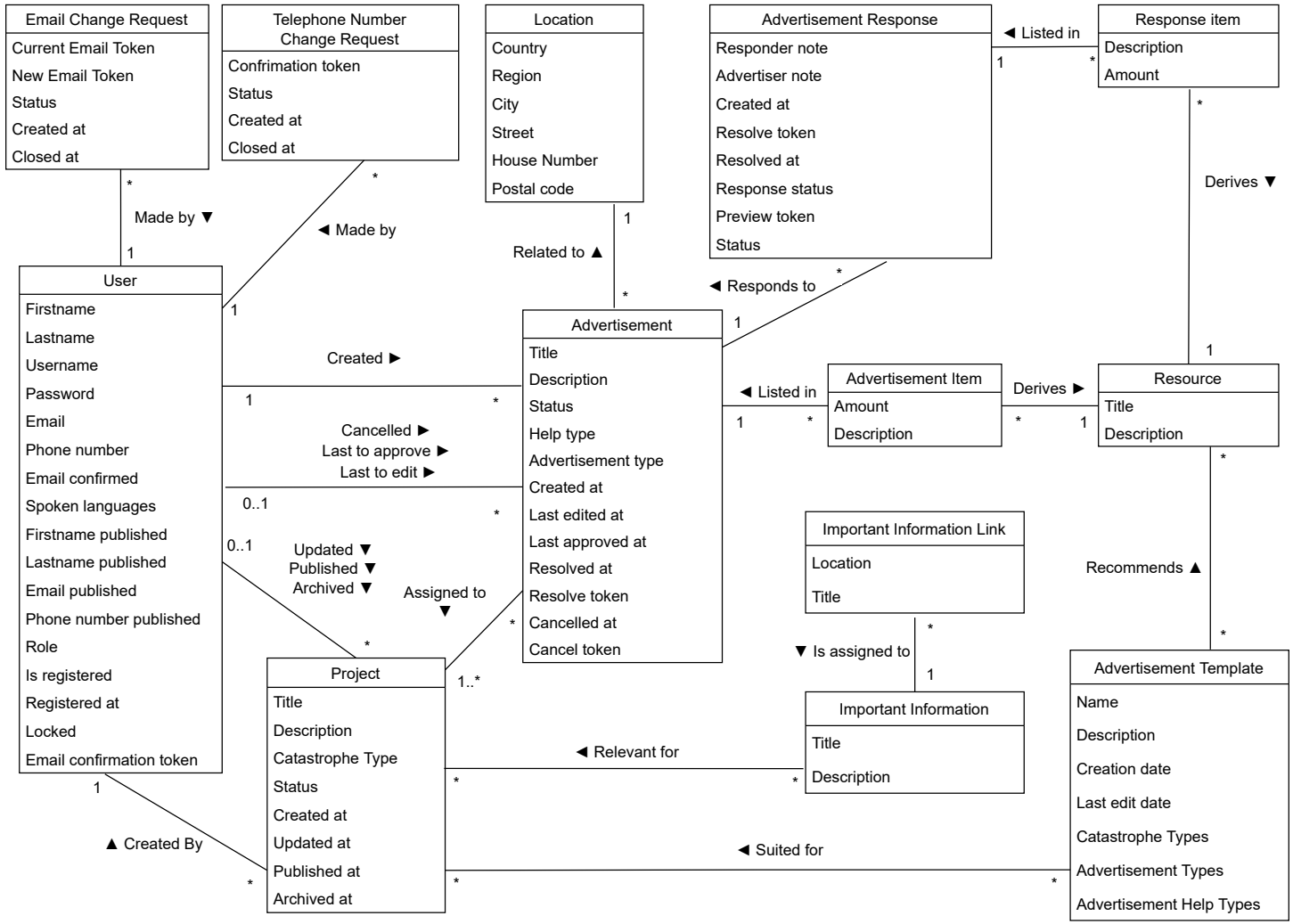
4.1.13 Important information

Jak bylo zmíněno výše, každý projekt může mít přiřazeno libovolné množství informací, které jsou pro někoho, kdo byl zasažen katastrofou, nebo chce o katastrofě zjistit více, důležité. Například pro válku na Ukrajině by takovouto důležitou informací mohl být odkaz na web `www.seznam.cz`.

Pro každou informaci je titulek a text ve více jazycích. Zároveň může mít více odkazů, které se k ní vztahují, což má využití například pro odkaz na českou a anglickou variantu webu (jeden odkaz pro českou variantu a druhý pro anglickou).



Obrázek 4.4 Diagram přechodů mezi stavy projektu



Obrázek 4.5 Diagram modelu domény

4.2 Výběr předvyplněných (enumerovaných) hodnot

Pro některé členy doménových hodnot se počítá s tím, že budou vybírané ze seznamu dostupných hodnot. Takovéto seznamy bylo nutné vytvořit tak, aby pokud možno pokryly v rozumné míře všechny potenciálně potřebné hodnoty, a zároveň nebyl jejich počet zbytečně rozsáhlý.

4.2.1 Typy katastrofy

Při výběru vhodných typů katastrof byl výchozím bodem dokument zveřejněný hasičským záchranným sborem ČR [34]. Hodnoty z tabulky č.1 ve zmiňovaném dokumentu byly shrnuty tak, jak je to prezentováno v tabulce 4.1. Důvodem shrnutí byla snaha nemít projekty zbytečně rozředěné do mnoha kategorií s příliš nízkou hustotou. Zároveň by takto mělo být možné většinu inzerátu jednoduše zařadit do jedné z vytvořených specifických kategorií a minimalizovat využití kategorie „ostatní“.

4.2.2 Typy pomoci

Výběr potřebných typů pomoci vycházel z typů pomoci dostupných na stránce s nabídkami na webu Pomáhej Ukrajině [35] a statistik na stránkách konsorcia nevládních organizací pracujících s migranty [36]. Byly vybrány některé z nejčastěji využívaných typů pomoci. Zároveň byly přidány některé další typy pomoci, které vycházely čistě z úvahy autora.

Ubytování Ubytování pro lidi zasažené katastrofou.

Materiál Nejobecnější typ materiální výpomoci. Měla by být použita, pokud žádná jiná kategorie nevyhovuje. Může být použita také ve chvíli, kdy nabídka obsahuje více typů materiální pomoci.

Jídlo a voda Nabídka/poptávka vody, konzerv s trvanlivými potravinami…

Zdravotnické potřeby Náplasti, dezinfekce, léky…

Dobrovolnictví Dobrovolnická výpomoc od lidí, kteří třeba nemají v ničem expertízu, ale chtějí pomoct.

Specialista Pomoc od člověka, který má pro něco expertízu. Může se jednat například o pomoc od IT specialisty.

Řemeslník Výpomoc řemeslníka.

Odvoz Nabídka/poptávka řidiče, automobilu, dodávky…

Zdravotnická výpomoc Výpomoc od zdravotníků, zdravotních sester, doktorů…

Psychologická pomoc Nejčastěji pomoc nabízená psychologem, případně poptávání pomoci psychologa.

Administrativa Pomoc s potřebnou administrativou - vyplňování úředních listin, chození po úřadech…

Ostatní Vše co nespadá do žádné z ostatních kategorií.

Tabulka 4.1 Typy katastrof

Hodnota na platformě	Hodnoty v dokumentu
Sucho	Dlouhodobé sucho
Vysoké teploty	Extrémně vysoké teploty
Vítr	Extrémní vítr
Vydatné srážky	Vydatné srážky
Povodeň	Povodeň Přívalová povodeň Zvláštní povodeň
Výpadek jídla a vody	Narušení dodávek potravin velkého rozsahu Narušení dodávek pitné vody velkého rozsahu
Biotické hrozby	Epidemie - hromadné nákazy osob Epifytie - hromadné nákazy polních kultur Epizootie – hromadné nákazy zvířat
Chemické hrozby	Únik nebezpečné chemické látky ze stacionárního zařízení Radiační havárie
Výpadek energií	Narušení dodávek plynu velkého rozsahu Narušení dodávek ropy a ropných produktů velkého rozsahu Narušení dodávek elektrické energie velkého rozsahu
Migrace	Migrační vlny velkého rozsahu
Kriminalita	Narušování zákonnosti velkého rozsahu (včetně terorismu)
Ostatní	Narušení funkčnosti významných systémů elektronických komunikací Narušení bezpečnosti informací kritické informační infrastruktury Narušení finančního a devizového hospodářství státu velkého rozsahu

4.3 Databázový model

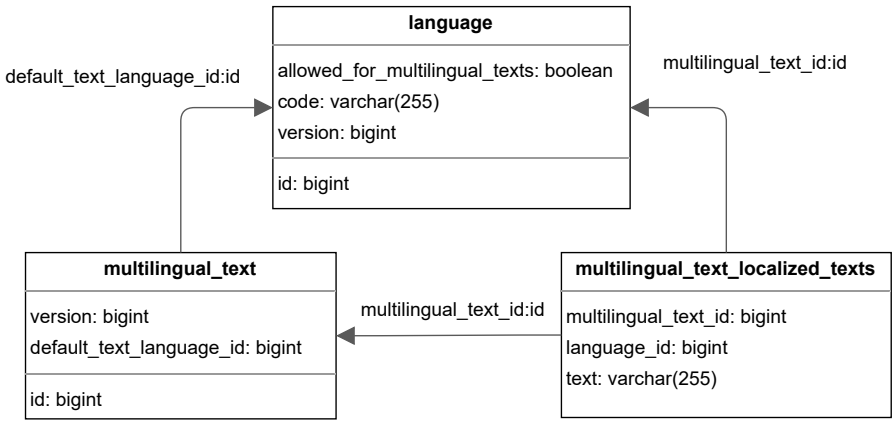
Model, který vyjadřuje strukturu databáze, vychází z doménového modelu popsaného v sekci 4.1. Jelikož tento model vychází z doménového modelu, budou v této sekci popsány pouze věci, které doménovým modelem nebyly zachyceny. U entit, které mají svou identitu, a jsou nějakým způsobem referencovány z vnější, přibyl k internímu identifikátoru ještě veřejný identifikátor. Tento identifikátor slouží pro identifikaci mimo aplikaci. Toto má několik výhod. První výhodou takovéhoto identifikátoru je, že ztěžuje uhodnutí identifikátoru entity, která není veřejně přístupná (rozhodně se to ale nedá považovat za ochranu). Hlavní výhoda přichází ve chvíli, kdy je jako identifikátor použit tzv. *slug*, což je v této aplikaci lidsky čitelný řetězec, skládající se z časové značky vytvoření a části titulku (nebo podobné textové hodnoty) ve výchozím jazyce (viz. podsekce 4.3.1). Z odkazu, ve kterém je takovýto slug zakomponován, je pak zřejmě lépe vidět, co přesně lze na této adrese očekávat.

4.3.1 Vícejazyčný text

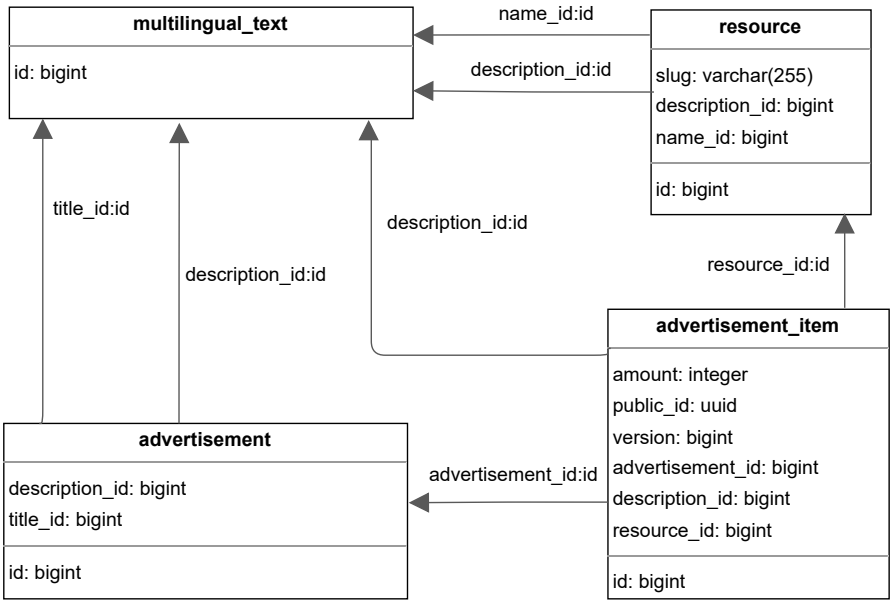
Jak vyplnulo z požadavků, je nutné aby byly lokalizované nejen statické texty, ale i texty které může zadávat uživatel. Takovýto text bude v databázi uložen, jako samostatná entita. Bude mít svojí identitu, kterou budou přebírat texty v jednotlivých jazycích. Jak již bylo předesláno předchozí větou, vícejazyčný text se skládá z několika textů, které mají zároveň určený svůj jazyk. Zároveň bude určený výchozí jazyk textu. Pokud nebude dostupný text v jazyce, který má nastavený uživatel, bude použit text ve výchozím jazyce. Text pro výchozí jazyk musí být vždy přítomný. Diagram těchto dvou tabulek, zahrnující i tabulku s jazykem, lze vidět na obrázku 4.6.

4.3.2 Věci v inzerátu a odpovědi

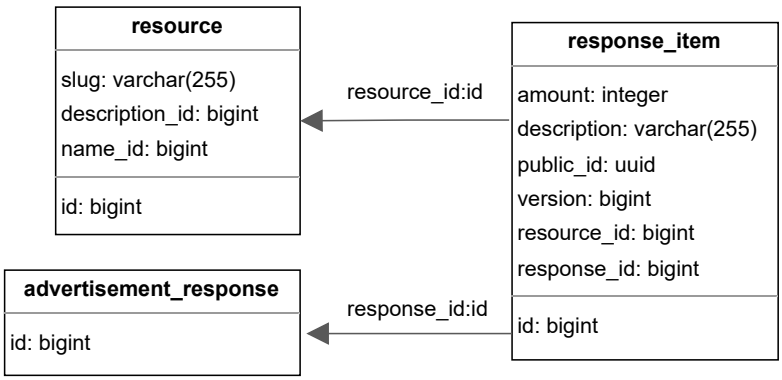
Věci v inzerátu a odpovědi mají podobnou strukturu, takže budou shrnuty v jedné sekci. Liší se tím, že u věci v inzerátu je navíc ještě napojení na vícejazyčný text kvůli jeho popisu. U odpovědi je toto pole jednoduchý řetězec (*varchar*). V obou případech je napojení na zdrojový prvek, které je povinné a má aritu M:1. Na obrázku 4.7 je model pro věc v inzerátu a na obrázku 4.8 pro věc v odpovědi na inzerát.



Obrázek 4.6 Databázový model vícejazyčného textu



Obrázek 4.7 Databázový model věci v inzerátu



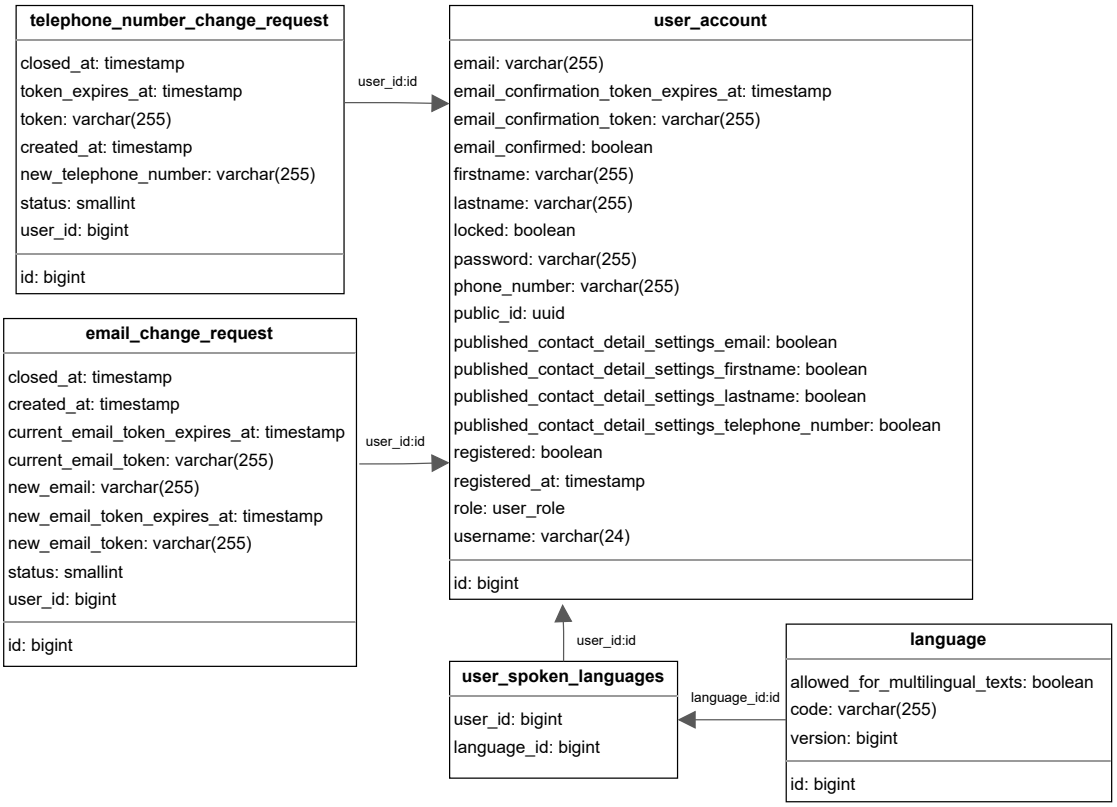
Obrázek 4.8 Databázový model věci v odpovědi na inzerát

4.3.3 Uživatel

Tabulka uživatele byla pojmenována *user_account*. Důvodem, proč bylo toto jméno zvoleno, je to, že *user* bývá v databázích rezervovaný výraz. Tabulku lze vidět na obrázku 4.9.

Nastavení zveřejněných detailů je uloženo v tabulce uživatelů. Jde o 4 sloupce typu *boolean*, kde každý nastavuje status zveřejnění jednoho z detailů. Jedná se o typický návrh vazby 1:1, kde obě strany jsou povinné.

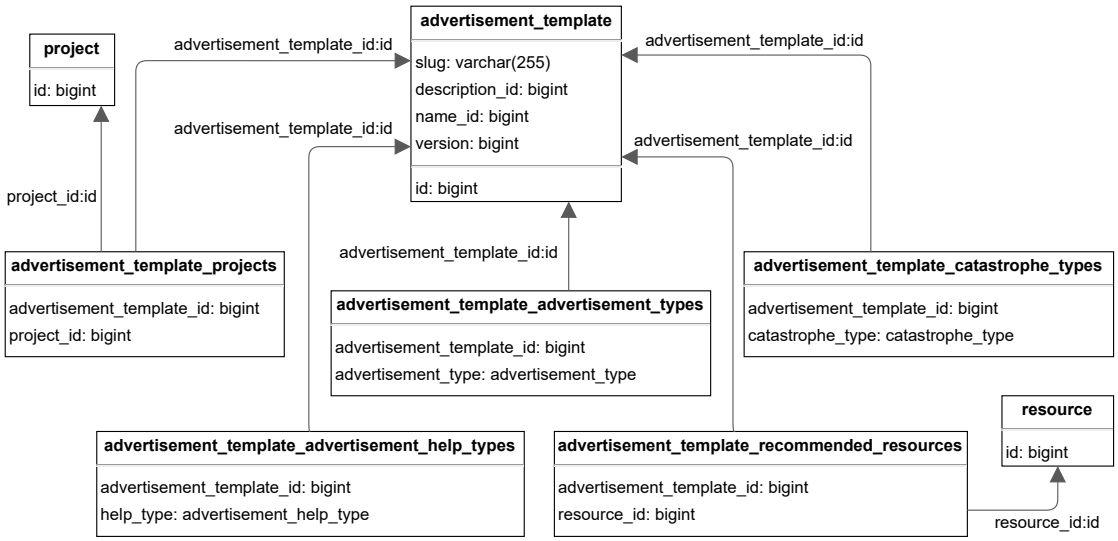
Jazyky, kterými uživatel mluví, jsou reprezentovány vazební tabulkou, která realizuje vazbu M:N mezi tabulkou uživatele a tabulkou jazyků.



Obrázek 4.9 Databázový model uživatele

4.3.4 Šablona inzerátu

Pro šablonu inzerátu vzniklo celkem šest tabulek. Kromě tabulky pro samotnou šablonu ještě vznikla jedna tabulka pro každý atribut, který přijímá více položek. Dvě z nich jsou tabulky pro projekt a doporučené zdroje zahrnutých položek, sloužící jako vazební tabulka pro vytvoření vazby M:N (obě strany vztahu mohou mít více jedinců). Zbylé tři tabulky slouží pro přiřazení hodnot z výčtů (typ katastrofy, typy inzerátů a typy pomoci). Záznamy v těchto třech tabulkách nemají vlastní identitu, místo toho ji přebírají od jiné spřízněné entity (v tomto případě od šablony). Bylo tak učiněno z toho důvodu, že se jedná pouze o vícenásobné atributy entit (každý pro právě jednu entitu) a bez existence těchto entit nemá smysl existence těchto atributů. Diagram s těmito tabulkami a zkrácenými tabulkami, se kterými mají šablony M:N vztah, lze vidět na obrázku 4.10.



Obrázek 4.10 Databázový model šablony inzerátu

4.4 Návrh API

Na serveru bude implementováno rozhraní (dále jen API), na které se bude moci frontend a koordinátoři či administrátoři dotazovat. V této sekci je toto rozhraní navržené pro implementaci v dalších fázích této práce. API se skládá z koncových bodů (anglicky *endopointů*), které jsou pro dotazy dostupné. Koncové body rozhraní byly rozdělené do několik skupin. URL koncových bodů ve skupině mají společnou předponu. V následujícím výčtu jsou předpony skupin s jejich popisem.

/project Koncové body pracující s projekty. Mimo jiné sem patří jejich vytvoření, získání, nebo archivace.

/advertisement Jak již název napovídá, seskupují se zde ty koncové body, které se vztahují k inzerátům. Je zde koncový bod pro vytvoření inzerátu, jeho schválení a uzavření či vyřešení.

/advertisement-response Skupina pro koncové body, které mají co do činění s odpovědí na inzerát (například její vytvoření, zamítnutí či schválení).

/advertisement-template Koncové body, sloužící pro správu šablon inzerátů. Patří sem například jejich vytvoření, nebo získání.

/important-information Skupina koncových bodů pro důležité informace k projektu. V tuto chvíli obsahuje pouze jeden koncový bod, který slouží pro vytvoření záznamu koordinátorem či administrátorem.

/resource Seskupení koncových bodů, které nějakým způsobem pracují se zdroji pro položky v inzerátech a odpovědích na ně.

/user Koncové body pro práci s uživatelem. Patří sem získání informací o uživateli či potvrzení kontaktu. Pro aktuálně přihlášeného uživatele je místo *slugu* v cestě použito „me“.

/auth Skupina skládající se ze dvou koncových bodů, jednoho pro přihlášení a druhého pro odhlášení.

Cesty k jednotlivým koncovým bodům byly sestavené tak, aby jejich účel bylo možné pochopit co nejvíce bez toho, že by bylo nutné nahlížet do dokumentace. Například cesta /advertisement/20230409142115803-převoz-leku-na-ukrajinu/detail pro HTTP metodu *GET* vede ke koncovému bodu pro získání detailu inzerátu, který má jako veřejný identifikátor nastavený řetězec „20230409142115803-převoz-leku-na-ukrajinu“. Dokumentaci jednotlivých koncových bodů lze nalézt na přiloženém médiu. Tato dokumentace byla vygenerována z již hotového kódu, jako OpenAPI[1] specifikace. Typy cest, které bylo nutné pro koncové body vytvořit, se dají zobecnit do několika málo skupin. Konvence pro vytváření těchto cest je v následujícím výčtu.

Není specifikována instance zdroje ani akce Cesta vede ke kořenu skupiny, která zaštiťuje koncové body vztahující se k entitě. Obecný tvar je *{cesta-ke-skupině}/*. Příkladem z API je `/advertisement/`, který na POST vytváří nový inzerát.

Není specifikována instance zdroje, ale je specifikována akce Podobně jako u předchozího bodu, počátek cesty je kořen zaštiťující skupiny. Za něj je ještě připnutá akce. Obecný tvar je *{cesta-ke-skupině}/{akce}*. Do této kategorie spadá například `/advertisement/filtered-page`, což vede na koncový bod, který při požadavku s *POST* metodou vrací stránku s inzeráty odpovídajícími filtru předanému v těle metody (důvod proč byl použit POST, a ne například GET, je rozebíraný v sekci 7.7.3).

Je specifikována instance zdroje, ale akce specifikována není Cesta vede ke kořenu skupiny a po něm následuje veřejný identifikátor zdroje (typicky buďto *slug*, nebo *publicId*). Drobnou výjimkou je uživatel, kde se v případě, kdy se dotaz vztahuje k aktuálně příhlášenému uživateli, používá místo veřejného identifikátoru výraz *me* (anglický výraz pro „já“). Tvar této cesty je tedy zobecněně *{cesta-ke-skupině}/{veřejný identifikátor}*. Příkladem může být `/resource/20230406154534009-konzerva-s-trvanlivym-jidlem`, který získá zdrojový prvek se *slugem* „20230406154534009-konzerva-s-trvanlivym-jidlem“.

Je specifikována jak instance zdroje tak akce Tato cesta je vytvořená tak, že se vezme cesta pro situaci, kdy je specifikována instance zdroje, ale akce specifikována není. Za ní se připojí akce a případně token, pokud je potřeba. Tvar, který cesty spadající do této skupiny dodržují, je tedy *{cesta-ke-skupině}/{veřejný identifikátor}/{akce}(/{token})?*

[1] Specifikace OpenAPI: `https://swagger.io/specification/`

Kapitola 5

Technologie a zabezpečení platformy

5.1 Úvod

Pro úspěšný vývoj aplikace je prvně potřeba rozumně zvolit kombinaci technologií tak, aby spolu vše hezky spolupracovalo. Při výběru je nutné se zaobírat více kritérii. Prvně bude nutné vyřešit licencování. Je nutné si dát pozor na to, aby se dané řešení dalo zakomponovat do aplikace, která je vydaná pod Apache 2.0 licencí[1]. Stejně tak nelze opomenout kompatibilitu jednotlivých technologií. Výběr dvou řešení, které spolu nejsou vzájemně kompatibilní, může zbytečně zvýšit náročnost celého vývoje (či ho učinit nemožným). Co hůře, pro zkombinování takovýchto technologií jsou často potřeba různá obcházení zamýšleného způsobu užívaní. Důležitou součástí pro autora budou dostupné materiály k dané technologii. Zároveň je ale důležité myslet na to, že novější se ne vždy rovná lepší. Bylo zvažováno jak to, na kolik autor danou technologii zná, tak i to jaké klady a zápory pro něj má její využití oproti jiné technologii.

5.2 Technologie pro aplikaci běžící na straně Serveru

5.2.1 JVM Technologie

Prostředí Javy je dobře známé mezi vývojáři softwaru. Umožňuje vytvářet software, který lze spustit na různých platformách. Jedinou věcí, která je od cílového systému požadována, je aby na něm bylo zprovozněné běhové prostředí. *Bytecode*, který vznikne kompilací původního *Java* kódu, je čtený a interpretovaný virtuálním strojem *JVM*. [37] Multiplatformnost je tedy jednou z velkých výhod všech jazyků *Javovského* prostředí a nebude dále explicitně zmiňována. Kromě samotné, dlouhými roky odzkoušené Javy, existuje také mnoho novějších variant. Tyto varianty jsou vlastními jazyky. V následujících odstavcích je rozebrána jedna z nich, která byla vybrána pro tvorbu backendu.

5.2.1.1 Java

Programovací jazyk Java je jedním z nejrozšířenějších programovacích jazyků na světě. Dle webu statista.com je Java pátý nejpoužívanější jazyk [38] a dle TIOBE Indexu dokonce třetí nejpoužívanější [39]. Jednou z výhod Javy je velká dostupnost vývojářů, kteří danou technologii ovládají.

[1] Apache license, version 2.0: `https://www.apache.org/licenses/LICENSE-2.0`

Zároveň je na webu dostupné široké množství materiálů, ze kterých lze při vývoji aplikací čerpat. Zmínit bych mohl i velké množství knihoven a frameworků, které jsou k dispozici. Jelikož je i druhá zmíněná technologie s Javou zpětně kompatabilní [40], je jedinou výhodou v této oblasti to, že tyto frameworky na Javě běží nativně. Java má oproti ostatním JVM technologiím také jisté neduhy. Na syntaxu Javy se začíná projevovat stáří - v porovnání s jinými JVM jazyky je dle názoru autora trochu neohrabaný.

Oproti jiným jazykům vzniká v Javě mnoho tzv. „boilerplate kódu“. To je kód který řeší nějakou obecnou věc a je nutné ho implementovat stále znovu jen s drobnými, nebo dokonce žádnými rozdíly [41]. Zde se však dá argumentovat tím, že za dobu existence Javy vznikly knihovny, které množství takovéhoto kódu redukují (viz. například projekt Lombok [42]). Z autorovi zkušenosti je však u těchto knihoven problém s tím, že integrace s vývojovými prostředími je dosti problematická. Často se díky nim rozbijí nějaké funkce, jako je například automatický refactoring.

Další nevýhodou je nullabilita typů. Java umožňuje, aby byly všechny typy nastavené na `null` označující jakousi prázdnou hodnotu. Tvůrce této hodnoty v jazyce ALGOL W, ze kterého ji Java přijala, Tony Hoare, ji dokonce označuje za „Billion Dolar Mistake“ - „Chyba za miliardu dolarů“ [43]. I tady existuje validní protiargumentace - v Javě 8 tvůrci přidali třídu Optional, která má spolehlivě označovat *nullable* typy a zjednodušovat práci s nimi. Oproti řešením z jiných jazyků má však neduhy vyplývající z toho, že se jedná o *value-based* třídu [44]. Při pokusech o práci s referencemi pak hrozí výsledek, který nebyl predikován [45].

5.2.1.2 Kotlin

Kotlin je jazyk vyvinutý firmou Jetbrains a přispěvateli z open source komunity [46]. Cílem kotlinu je dle slov tvůrců udělat vývojaře šťastnější. Během vývoje odstranili některé nedostatky, které Java měla, a dále na tomto jazyku pracují. [47] Narozdíl od Javy je u Kotlinu nutné explicitně říct, že je očekávana možná přítomnost *null* hodnoty. [48] Také má mechanizmy, které mají za cíl redukovat množství boilerplate kódu. Mezi ně se řadí například datové třídy, které již mají předpřipravené funkcionality, jako jsou `equals` a `hashCode` podle dat, nebo formátovaný `toString`. [49] Podpora *delegation* vzoru pak také přispívá k redukci boilerplate kódu. [50]. Autor s tímto jazykem, stejně jako s Javou, již má předchozí zkušenost. Z těchto, ale i dalších důvodů, byl jako jazyk pro tvorbu této práce vybrán Kotlin.

5.2.2 Spring

Pro použití na backendu byl zvolen Spring framework a jeho projekty. Jedná se o Java framework, který se soustředí na rychlost, jednoduchost a produktivitu. Používá ho mnoho velkých společností, jako je Amazon či Google. Je flexibilní v tom, co vše s ním lze postavit, Jeho vývojáři se starají o jeho výkon. *Spring boot* navíc umožňuje stavět aplikace s daleko menši námahou, než při použití jiných konkurenčních paradigmat. Jeho součásti, jako jsou zabudovaný web server a autokonfigurace umožňují vývojářům mnohem rychleji iterovat (nové verze aplikace – pozn. autora). Spring se navíc osvědčil svým rychlým a zodpovědným řešením technických problémů. Přispěvatelé při opravách a testování nahlášených chyb spolupracují s odborníky na bezpečnost. Kromě toho probíhá monitoring závislostí třetích stran. Jsou vydávány průběžné aktualizace, které mají data a aplikace zachovat tak bezpečné, jak jen to jde. Projekt Spring Security, který je dnes již de-facto standardem pro bezpečnost aplikací založených na Springu, ulehčuje integraci se standardními průmyslovými bezpečnostními schématy, a doručovat důvěryhodná řešení která jsou již v základu bezpečná. [51, 52] Jak již bylo naznačeno, Spring má několik projektů pro různé účely. Jedním z již zmíněných projektů je Spring Security zabezpečující aplikace. Dalším příkladem projektu, který bude také využit při tvorbě této aplikace, je Spring Session zprostředkovávající rozhraní a implementace pro správu relací uživatele (viz. podsekce 5.2.5 a podsekce 5.2.6). [53] Hlavní důvodem k tomuto výběru bylo široké spektrum nabízených funkcionalit, podpora

Kotlinu [54], a v neposlední řadě zkušenost autora s touto technologií ze školního i produkčního prostředí.

5.2.3 REST API

Pro prezentaci dat ze serveru je nutné vytvořit rozhraní, na které se bude moci webová stránka dotazovat. Rozhodl jsem se vytvořit REST API. REST je architektonický vzor založený na několika principech. Jednou z nejznámějších charakteristik je, že komunikace mezi serverem a klientem je **bezstavová**. To znamená, že jednotlivé dotazy od klienta k serveru jsou na sobě z pohledu serveru nezávislé. Server je zpracovává nezávisle na sobě a je starostí klienta, aby vedl „konverzaci“ správným směrem [55]. K dalším charakteristikám patří například uniformní rozhraní - server vrací ze svých endpointů data v pevně daných formátech. Těch může být více a můžou se lišit od reprezentace v aplikaci. Jaký výstupní formát server zasílá se lze dozvědět z odpovědi. Pokud jsou ke kompletnosti dat potřebné i další zdroje (typicky provázaná entita), měl by server poslat v odpovědi i informace o tom, kde daný zdroj získat (hyperlink). [55]

5.2.4 Autentikace a autorizace

Pro autentikaci a autorizaci uživatele existuje několik možností.

Nejdříve je potřeba vyřešit, jak poprvé zjistit identitu uživatele (registrace a přihlášení). K tomu je několik dobře známých řešení. Pro aplikaci bylo vybráno přihlášení pomocí přihlašovacího formuláře. Uživatel vyplní jméno a heslo, které se poté odešle serveru. Server poté tyto údaje zpracuje a porovná, jestli je možné se s nimi přihlásit jako nějaký uživatel [56]. Nakonec, v případě úspěchu, odešle server klientovi zpátky nějakou informaci, pomocí které se bude v dalších konverzacích prokazovat (o tom v následujících sekcích).

Pro jednorázové akce je možné prokazování se pomocí jednorázových přístupových *tokenů*. *Token* v této aplikaci bude náhodně vygenerovaný, jeho platnost bude omezená právě pro jednu akci (přijetí odpovědi, označení inzerátu za přijatý, potvrzení emailu…), a bude ukládán vždy v tabulce u entity ke které patří. V některých případech, jako je potvrzení žádosti o změnu nastavení, bude navíc platnost *tokenu* omezena časově, což by mělo zúžit možné riziko ukradení platného tokenu. Útočník se k němu bude muset dostat před vypršením jeho platnosti.

Pro tento projekt bude použita autorizace na základě rolí, která podle uživatelské role buďto dovolí, aby uživatel akci provedl, nebo mu to zamítne. Konkrétní přístup bude záviset na vybraném frameworku.

5.2.5 JWT

Jak již bylo řečeno, po autorizaci musí server odeslat něco, čím se bude uživatel při každém requestu prokazovat. Zvažován byl standard JWT (RFC 7519). Ten pracuje s tokeny, ve kterých jsou obsažené informace o uživateli, jako jeho uživatelské jméno nebo oprávnění. Tyto informace server posílá podepsané, takže pokud by se je někdo pokusil změnit, je možné to rozeznat a takovýto token neuznat. [57] Tato možnost ale nakonec zvolena nebyla, jelikož její použití na stránkách má řadu nevýhod. Ta největší je dána jednou z jeho vlastností, a to tím, že dokud je token platný, jsou data v něm považována za platná. Invalidace relace před jejím zamýšleným vypršením (např. kvůli zrušení všech relací uživatele) je zbytečně obtížná a často jde proti tomu, že je JWT bezstavový. Bezstavovost je přitom hlavním důvodem k jeho užití. [58] Z tohoto důvodu nakonec tato varianta nebyla využita.

5.2.6 Session ID

Další možností je využití session ID, kdy u klienta bude uložený pouze nějaký identifikátor, kterým se bude klient při komunikaci se serverem prokazovat. Spring framework toto umí pro přihlášeného uživatele udělat automaticky, přičemž ID je zpět posíláno ve formě cookie. Toto řešení má tu výhodu, že je možné relaci invalidovat (například při změně role či zablokování uživatele). [59, 60] Toto řešení bylo nakonec uznáno za dostatečné a bude tedy využito.

5.3 Technologie pro použití na straně uživatele

5.3.1 Angular Framework

Pro tvorbu uživatelských byl využit Typescriptový framework Angular. Tento framework, se sice netěší takové popularitě, jako např. React, který je dnes jedním z nejpopulárnější Frameworků pro tvorbu webových stránek[61] ale má jiné přednosti. Jeho architektura (MVVM[2]) umožňuje psát kód, který má separovanou logiku od uživatelského rozhraní. Navíc má dostupnout vývojářskou příručku a spoustu dalších vlastností, které usnadňují vývoj a pomáhají psát udržitelný kód. [62] Umí ošetřit různé typy bezpečnostních chyb (viz. sekce 5.5). Stejně jako pro cokoliv *JavaScriptového* platí i pro Angular, že pro jeho rozšíření existuje mnoho knihoven, přičemž existují i katalogy některých těchto knihoven. Má navíc propracovaný systém pro generování různých typů součástí kódu [63]. Je nutné také zmínit podporu ze strany IDE od JetBrains, které bude využíváno k vývoji frontendové i backendové části. V neposlední řadě je to také jediný framework, se kterým má autor alespoň nějaké okrajové zkušenosti z jeho pracovního života, a se kterým mu jsou kolegové schopni poradit v případě, že by se na něčem zasekl.

5.4 Technologie pro nasazení aplikace a ukládání dat

U většiny aplikací je cílem jejich vývoje to, aby je nakonec bylo možné někam nasadit. Nejinak je tomu i v případě této práce. Zvolením vhodných technologií je možné si tuto práci usnadnit a předejít zbytečným chybám. Jelikož lidé čas od času dělají chyby, je snaha co nejvíce věcí automatizovat, aby se co nejvíce předešlo příležitostem k tomu nějakou chybu udělat.

5.4.1 Docker

Docker je open-source platformou využívanou při vývoji a dodávkách aplikací, značně snižující prodlevu mezi psaním kódu a jeho spuštěním na produkci [64]. Ke svému fungování využivá kontainerizaci. Kontejner je odděleným procesem běžícím na hostujícím sytému. Takovýto proces pak může být spuštěn na libovolném systému. [65] Jako vzor pro vytvoření kontejnerů slouží obraz (*image*). Obraz je často založený na jiným obrazu a je dále upravený. Instrukce k úpravám jsou dodávány v souboru, kterému se říká Dockerfile. Ten se může skládat z více instrukcí, a každá instrukce pak vytváří další vrstvu, přičemž při znovuvystavění kontejneru jsou přestavěny jen ty vrstvy, které byly změněny. [66]

5.4.2 PostgreSQL

Pro ukládání dat byl zvolen databázový stroj *PostgreSQL*. Jedná se o mocný, objektově relační databázový systém. Za dobu svého aktivního vývoje, který již trvá více jak 35 let, si získal dobrou reputaci pro svoji spolehlivost, robustní vlastnosti a výkon. [67] Bylo provedeno porovnání s jiným oblíbeným databázovým systémem, *MySQL*, a nakonec z něj vyšel vítězně právě *PostgreSQL*.

[2] `https://www.ackee.cz/blog/glossary/mvvm`

Kromě popularity obou těchto databází byly zvoleny pro porovnání také z toho důvodu, že obě databáze jsou k použití zdarma (jediné co je potřeba je server, na kterém budou hostovány). První rozdíl je v tom, jaký přístup k ukládání dat databázové systémy mají. Zatímco *PostgreSQL* je objektově-relační databáze, *MySQL* je čistě relační databáze. [68] V *MySQL* jde čistě o vztahy mezi daty a *PostgreSQL* kombinuje věci známé jak z objektově orientovaného přístup (modelování dat, jako objektů, které od sebe dědí atd.), tak právě relačního přístupu (relace mezi daty). Oba typy databází mají tabulkovou strukturu, objektově relační databáze ji však mají ještě obohacenou o koncepty objektového programování. [69] *PostgreSQL* je bohatší na funkcionalitu, má například širší podporu typů, kde navíc oproti *MySQL* mimo jiné podporuje enumerace, síťové adresy, či pole a také umožňuje definici vlastních datových typů. *PostgreSQL* by navíc, díky mechanizmu kterému se říká MVCC[3], mělo nabízet rychlejší paralelní přístup k datům, což může mít u webové aplikace, kde je předpoklad, že k těm samým datům bude přistupovat více uživatelů najednou, pozitivní vliv na výkon. [68]

5.5 Bezpečnost aplikace

Bezpečnost aplikace je dnes ve většině případů důležitá. Je několik věcí, které je nutné chránit. Jednou z nejdůležitějších věcí je ochrana dat uživatelů. Jsou chráněny nejenom před neoprávněným čtením, ale i neoprávněnou změnou, vytvořením či smazáním. S tím souvisí další důležitá část - řízení přístupu k jednotlivým částem aplikace.

5.5.1 Ochrana proti downgrade útoku na HTTP/TLS

Komunikace by dnes již vždy měla probíhat přes *HTTP/TLS*. *HTTP/TLS* je varianta *HTTP*, která při komunikaci používá protokol *TLS* [70]. Je zde ale riziko tzv. *downgrade útoku* na *HTTPS*. Ten spočívá v tom, že se útočník odposlouchávající komunikaci (tzv. *Man in the Middle*) pokusí přesvědčit klienta, aby komunikaci zasílal přes čisté *HTTP*. [71] Data jsou pak díky tomu odesílaná v nešifrované podobě a každý přes koho komunikace prochází si je může volně pročítat. Jednoduchým řešením je při první návštěvě stránky odeslat *HSTS* hlavičku. Tou je indikováno, že by komunikace se servery domény měla vždy probíhat přes šifrovaný protokol. Pokud toto prohlížeč podporuje, pak pokaždé, když dojde k pokusu o komunikaci s doménou přes nešifrované *HTTP*, dojde k přesměrování na *HTTPS* variantu. Navíc pokud server hostující aplikaci obdrží *HTTP* požadavek přes nešifrovaný protokol, měl by odeslat odpověď indikující trvalé přesměrování na *HTTPS* variantu *URL*. [72]

5.5.2 Ochrana proti CSRF útoku

Cross Site Request Forgery (dále jen *CSRF*) útok spočívá v tom, že se útočník pokusí nějakým způsobem vynutit vykonání nějaké akce na stránce, na které je oběť zrovna autentifikována. Příkladem může být internetové bankovnictví. Útočník pošle klientovi email, ve kterém ho požádá o nějakou akci. V pozadí bude skrytý formulář, který bude mít pro akci nastavenou adresu pro přeposlání peněz z účtu oběti na jiný účet. Pokud se oběť nechá nachytat a na odkaz klikne, může v případě špatného zabezpečení tohoto bankovnictví přijít o sumu peněz nastavenou ve formuláři. [73] Chránit se lze několika způsoby. Jeden z nich, známý právě z již zmíněného internetového bankovnictví, je dvoufázové ověřování pro akce způsobující změnu na stránce. Je možnost vynutit, že uživatel musí každou změnu údajů či úpravu stránky potvrdit přes nějaký další faktor (např. kód doručený do mailu, nebo telefonu). Toto řešení ale může uživatele začít po chvíli značně obtěžovat. Mnohem lepším řešením je použít alternativy dostupné v použitém frameworku. Angular nabízí jednoduchou možnost, jak se chránit. Se správným nastavením, bude

[3]Multi-Version Concurrency Control: https://www.postgresql.org/docs/7.1/mvcc.html

do každého požadavku na server přidávat *X-XSRF-TOKEN* hlavičku, a zároveň bude posíláno a *XSRF-TOKEN cookie*, které předtím klient obdržel ze serveru [74]. Server musí toto *cookie Angularu* poslat. *Spring boot* však již má mechanizmus, který toto umožňuje. Mimo jiné má výchozího poskytovatele těchto cookies, který se jmenuje `CookieCsrfTokenRepository`, a používá stejná jména pro *cookie* a hlavičku požadavku s *XSRF tokenem*, jako používá *Angular*, což ještě ulehčí implementaci pro tento projekt [75].

5.5.3 Cross-Origin Resource Sharing (CORS)

Jde o mechanizmus založený na *HTTP* hlavičkách, který umožňuje serveru indikovat, odkud by měl prohlížeč povolit načítaní zdrojů (kromě sebe sama). Server může, kromě povoleného původu zdroje (*Access-Control-Allow-Origin*[4]), předat hlavičky upravující povolené *HTTP* metody požadavků (*Access-Control-Allow-Methods*[5]) nebo hlavičky (*Access-Control-Allow-Headers*[6]). Navíc povoluje říct, jestli je má server u takovýchto požadavků povolit přidávání přihlašovacích údajů (*Access-Control-Allow-Creadentials*[7]). [76] Na straně klienta tedy *CORS* řeší prohlížeč. Je ale nutné nastavit server, aby posílal požadované hlavičky. Naštěstí má pro toto *Spring Framework* zabudovanou podporu. Je pouze nutné nastavit, jaké mají mít hlavičky posílané v rámci předletové přípravy hodnotu. [77]

5.5.4 Cross-site scripting (XSS)

Jednomu z typů útoků, který cílí primárně na klientskou část aplikace, se v angličtině říká *Cross-site scripting*, což v přímém překladu znamená „scriptování mezi webovými stránkami“ (autorovi se nepodařilo najít ekvivalentní český výraz, tak se pokusil o překlad sám). Jedná se o situaci, kdy jsou do stránky, která je jinak spolehlivá, vloženy škodlivé kusy kódu (*skripty*). Prohlížeč nemá jak zjistit, že se jedná o kód který je nedůvěryhodný, a tak kód spustí. Tento kód může přistupovat ke všem *cookies* a dalším datům, které jsou dostupné pouze navštívené stránce (například *JWT token* popsaný v sekci 5.2.5). [78] Ochrana proti této zranitelnosti je naštěstí již v tuto chvíli součástí často používaných *frameworků*. Výjimkou není ani *Angular framework* používaný v této aplikaci. Ve výchozím stavu považuje *Angular* všechny hodnoty, jako nedůvěryhodné, a ošetřuje je při vkládání do dokumentu. Pokud by vývojář chtěl použít hodnotu bez ošetření, musí to explicitně určit mechanizmy, které *Angular* pro toto definuje. [79]

5.5.5 Ukládání hesla a jiných přístupových údajů

Při zabezpečení aplikace by se nemělo zapomínat na to, jak jsou ukládána hesla a jiné přístupové údaje. Pokud dojde k úniku dat, nemělo by být možné zjistit jaká je jejich skutečná hodnota Z toho důvodu je dnes již běžnou praxí, že se hesla před uložením do databáze tzv. *hashují*. *Hashování* je jednosměrná funkce, což znamená, že je nemožné z výstupu získat původní vstup této funkce. [80] Pokud je zahashována pouze hodnota hesla, je možné provést útok za použití tzv. *rainbow table* [81]. Tento problém řeší technika zvaná solení, která k zadanému heslu přidává náhodně vygenerovaný řetězec [80]. O konkrétní implementaci použité na této platformě více v sekci 7.2.4.

[4] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin`
[5] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Methods`
[6] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Headers`
[7] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Credentials`

Kapitola 6
Implementace frontendu

6.1 Shrnutí

Jako první byla implementována frontendová část. Hlavním důvodem k tomuto postupu byla možnost vyzkoušet si práci s navrženým modelem. Díky tomu bylo hned zpočátku odchyceno několik chyb a nedotažených částí. Aplikace je virtuálně rozdělena do několika menších vrstev.

6.2 Design aplikace

Pro design aplikace byla zvolena knihovna *Angular Material*. Jak název napovídá, tato knihovna poskytuje komponenty, při jejichž návrhu byl použit návrhový systém Material Design. [82]

Dle slov tvůrců je *Material Design* návrhový systém podporovaný designéry a vývojáři společnosti *Google*. Zahrnuje do hloubky jdoucí návod pro *UX* a implementaci *UI komponent* pro *Android*, *Flutter* a *Web*. [83] Jako hlavní téma bylo vybráno *deeppurple-amber* předpřipravené tvůrci *Angular Material*. Pro toto téma je hlavní barvou odstín fialové (*deep-purple*). Jako sekundární barva, sloužící především pro zvýraznění klíčových prvků rozhraní, se používá odstín žluté (*amber*). Výstražnou barvou je pak tradičně červená. [84]

6.3 Logo aplikace

Pro platformu bylo potřeba vytvořit logo. K vytvoření byl použit program *Paint 3D* dodávaný s operačním systémem *Windows 10*. Logo má průhledné pozadí a skládá se z ohraničení a prvních dvou písmen názvu platformy (E a G). Logo zobrazené v hlavičce stránky bylo zmenšeno na velikost 16x16 a převedeno do formátu *ico* používaného pro ikony. Logo je k nahlédnutí na obrázku 6.1.

Obrázek 6.1 Logo

6.4 Lokalizace aplikace

Pro lokalizaci statických částí aplikace byla využita knihovna *ngx-translate*. Překlady jsou organizované do JSON souborů, kde jeden soubor obsahuje překlady pro jeden jazyk. Překlady v těchto souborech jsou organizované, a je v nich možné snadno vyhledávat.

Překlad dynamických částí je řešen pomocí entitních struktur určených pro vícejazyčný text. Při načtení stránky je jazyk nastavený na angličtinu. Změnit ho je možné v hlavičce stránky, a po jeho změně dojde k překladu všech statických částí a také těch dynamických částí stránky, u kterých je to možné. Pro sjednocení práce s jazykem byly vytvořeny dvě servisní třídy, jedna pro práci s dynamickým vícejazyčným textem (*MultilingualTextService*) a druhá pro práci s jazykem aplikace (*LanguageService*). Jednou z funkcionalit, které třída pro práci s vícejazyčným textem poskytuje, je vytvoření *Observable* emitující hodnotu textu v aktuálně vybraném jazyce při každé změně jazyka.

Třída pro práci s jazykem aplikace mimo jiné poskytuje jazyky, které jsou v aplikaci dostupné. Rozděluje je na jazyky, které jsou aplikaci známé, a ty které jsou použitelné pro jazyk uživatelského rozhraní. Platí, že skupina s jazyky použitelnými pro jazyk UI je podmnožinou známých jazyků.

6.5 Komunikace se serverem

Komunikace se serverem probíhá na servisní vrstvě frontendu. Jednotlivé servisní třídy využívají pro posílání požadavků a získání odpovědi třídu *HttpClient* poskytovanou *Angularem*. Ze serveru je na většině míst získán *DTO*, který je následně v konvertoru namapován na modelovou třídu aplikace. Tento postup byl zvolen z toho důvodu, že *Angular* neumí ze stahovaných dat automaticky vytvořit instance konkrétních tříd [85]. V lepším případě by tedy bez konverze byl výsledkem objekt, který má požadované atributy se správným typem, ale nemá očekávané metody. V horším případě, by ani atributy objektu nebyly v souladu s tím, co bylo deklarováno rozhraním/třídou.

6.6 Práce s datumy

Na několika místech aplikace bylo nutné pracovat s datumy. Základní funkce poskytované jako součást JS jsou však značně omezené. Není možné si například určit, jaký formát mají datumy mít. [86] Pro zjednodušení práce s nimi byla použita knihovna *date-fns*[1]. Důvodem k výběru této knihovny byl dobrý poměr mezi složitostí využívání a rozmanitostí funkcí které nabízí. [87] Výhodou také byla podpora ze strany Angular Material. [86] Pro účely této práce jsou *Angulařímu DateAdapteru* nastavené české *locales*. *Locales* je sada pravidel definujících nějaký jazyk (v tomto případě především formát datumu) [88]. Z předchozích dvou vět plyne, že komponenty zobrazují datumy ve formátech, na které jsou zvyklí lidé v ČR.

6.7 Často používané komponenty

6.7.1 Vstup s vícejazyčným textem

Jelikož aplikace má pomoci co nejširšímu spektru lidí, bylo nezbytně nutné přijít s nějakým konceptem vícejazyčného textu zadávaného uživatelem. Struktura vícejazyčného textu již byla dříve popsána. Bylo potřeba vymyslet komponentu, do které bude možné text ve více jazycích zadávat. Vznikly dvě komponenty které sdílí společnou logiku. Liší se tím z čeho jsou složené.

[1] date-fns - `https://date-fns.org/`

Jedna komponenta je *wrapper* kolem *inputu* s typem *text*, zatímco druhá komponenta je *wrapper* kolem *<textarea>*. Obě komponenty umožňují přepínat aktuálně upravovaný text, který může být získáný z vnějšku, nebo za pomoci *select komponenty* vnořené do *inputu/textarei* vlevo. Nabízí i několik možností pro validaci a úpravy výstupních dat. Můžeme určit, jaké jazyky jsou vyžadované (musejí být neprázdné). Stejně tak je možné určit, jestli chceme text pro jazyk smazat, pokud je text v daném jazyce nevyžadovaný a zároveň je text prázdný, což je věc kterou typicky chceme, protože je pak text rovnou připravený k odeslání na server.

6.7.2 Komponenta pro výběr jazyka

Pro výběr jazyka vznikla komponenta, která pracuje se strukturou vytvořenou pro jeho reprezentaci v aplikaci. V tuto chvíli aplikace podporuje dva jazyky rozhraní, češtinu a angličtinu, lze však snadno přidat další jazyky, nastavení těchto jazyků se provádí jak na serveru, tak na straně frontendu. Názvy jazyků jsou bez překladu přímo v tom jazyce. Angličtina je tedy English a čeština je Čeština. Využití této komponenty lze vidět na obrázku 6.2 jako součást hlavičky stránky.

6.7.3 Náhledový grid

Jak již název napovídá, náhledový grid (či náhledová mřížka) slouží především pro zobrazení náhledů na několik položek. Náhledy jsou zobrazené ve mřížce, jejíž konkrétní nastavení závisí na místě použití a dodaných datech (resp. jejich množství). Typické rozložení je popsané v tabulce 6.1. Přepočet počtu sloupců je prováděn v této komponentě v závislosti na celkové velikosti obrazovky a zadaných parametrech při vytvoření. Z toho nepřímo vyplývá, že ideálním použitím této komponenty je v místech, kde zabírá celou stránku, nebo alespoň její větší část. Položky v gridu je možné sdílet, je k tomu potřeba pouze dodat odkaz, kam má zpráva odkazovat.

Tabulka 6.1 Běžné rozložení náhledové mřížky

Šířka obrazovky	Počet sloupců	Počet řádek
< 900	1	$PocetPolozek$
< 1200	2	$\frac{PocetPolozek}{2}$
≥ 1200	4	$\frac{PocetPolozek}{4}$

6.7.4 Univerzální prohledávatelný list

V rámci projektu jsem si jako jednu z komponent napsal i list, který lze různě prohledávat a vyhledávací dotazy zasílat na server. To, že list stahuje data ze serveru, je znázorněno animovaným kolečkem ve vyhledávacím poli.

6.7.5 Notifikační služba

Na různých stránkách je potřeba nějakým způsobem upozornit uživatele, že se něco děje. K tomu v základu využívám knihovnu Notiflix. Jelikož se jedná o čistě Javascriptovou knihovnu (není ušitá přímo pro *Angular*, ale univerzálně pro jakoukoliv *Javascriptovou* aplikaci) [89], autor brzy narazil na několik problémů. Tím největším byl překlad hlášek. Jelikož JSON s překlady je načítán ze serveru, nemusí být v době vytvoření načítacího *spinneru* „loading“ hláška načtená. Na různých místech byl tedy kód, který byl velice podobný, a který sloužil k obcházení tohoto neduhu. Byla tedy vytvořena servisní třída (*NotificationService*), která je postavená nad touto knihovnou, a která umí řešit i situaci kdy ještě text není načtený.

6.8 Obrazovky

6.8.1 Header - hlavička stránky

Header ve kteroukoliv chvíli obsahuje titulek a navigaci k těm stránkám, které nevyžadují vybraný projekt či přihlášeného uživatele (případně naopak nepřihlášeného uživatele). Jedná se o název stránky (Egidio), který je nejen logem, ale je také prokliknutelný a lze se přes něj dostat na úvodní stránku (Projects). V navigační liště je dostupná položka vedoucím k seznamu projektů. Po výběru projektu se nabídka rozšíří o přehled inzerátů a jejich vytvoření. Dále je zde možnost vybrat jeden z nabízených jazyků a odkaz ve formě ikony, vedoucí na stránku s přihlášením. Po přihlášení uživatele se odkaz změní tak, aby místo na přihlašovací stránku, vedl na hlavní uživatelskou stránku, a vedle něj ještě přibude odhlašovací ikona. Hlavička zobrazená přihlášenému uživateli, s vybraným projektem „Válka na Ukrajině“, je k náhlednutí na obrázku 6.2

Obrázek 6.2 Hlavička s vybraným projektem a otevřenou nabídkou pro výběr jazyka

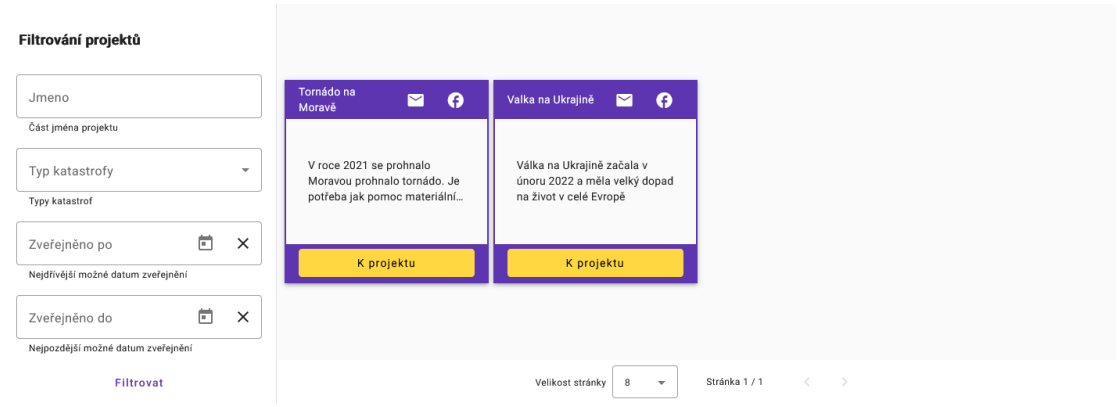
6.8.2 Footer - patka stránky

Footer je pravděpodobně nejjednodušší částí stránky. Obsahuje pouze klauzuli, ve které informuji o spolupráci s OpenDataLabem a ČVUT FIT. Názvy i loga subjektu jsou rozkliknutelné a směrují na stránky jednotlivých subjektů. Kromě ČVUT FIT a OpenDataLabu bylo na footer přidáno logo Profinitu. V dalším odstavci upozorňuji na to, že zadané údaje nejsou prověřovány a jako provozovatel se zříkám zodpovědnosti za škody, které by jejich užitím mohli vzniknout. Prohlédnout si ji je možné na obrázku 6.3

Obrázek 6.3 Patka stránky

6.8.3 Projects - úvodní stránka

Jako vstupní stránku jsem zvolil přehled dostupných projektů. Ty jsou zobrazené formou responzivního gridu. Zároveň ho lze filtrovat podle následujících filtrů:



Obrázek 6.4 Přehled projektů

Část jména - část textu, kterou titulek obsahuje buďto ve vybraném jazyce, nebo jazyce který byl vybrán zadavatelem projektu, jako výchozí pro titulek a popisek

Časové rozmezí publikace - doba, mezi kterou byl projekt zadán, přičemž platí, že počáteční mez je od půlnoci zadaného dne a koncová mez do 23:59:59 daného dne. Je validováno, že zadané hodnoty jsou data, a že první datum je nejpozději stejný den, jako poslední datum. V případě kdy tomu tak není vyskočí chybová hlaška

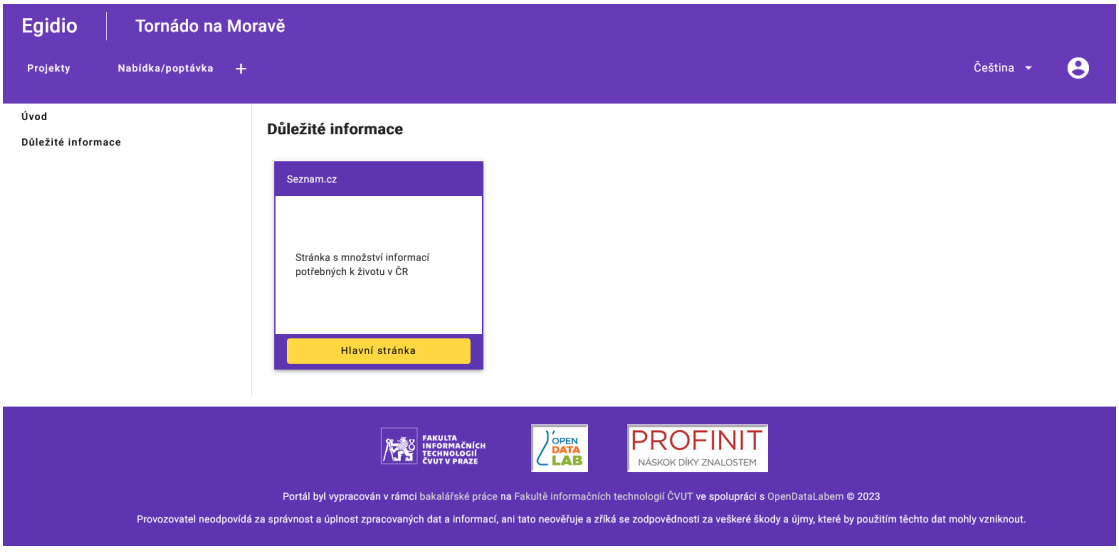
Typ katastrofy - povodně, požár…

Toto je jedna z pěti stránek, která je dostupná bez vybraného projektu. Zádávání projektů v tuto chvíli probíhá pouze přes neveřejné API a uživatelské rozhraní pro zadavání práce ručně bude součástí následného rozšíření.

6.8.4 Project detail

Stránka obsahuje dvě sekce, mezi kterými se uživatel pohybuje pomocí bočního navbaru. Těmito sekcemi je stránka s popisem projektu (viz. obrázek 6.5) a stránka s důležitými odkazy (viz. obrázek 6.6), které se mohou hodit uživatelům zasaženým katastrofou, nebo uživatelům chtějícím pomoci.

Obrázek 6.5 Úvod k projektu



Obrázek 6.6 Důležité informace o projektu

6.8.5 Help list

Stránka s nabídkou/poptávkou pomoci je stejně jako stránka projektů realizována formou gridu. Pro filtrování jsou pak k dispozici následující položky:

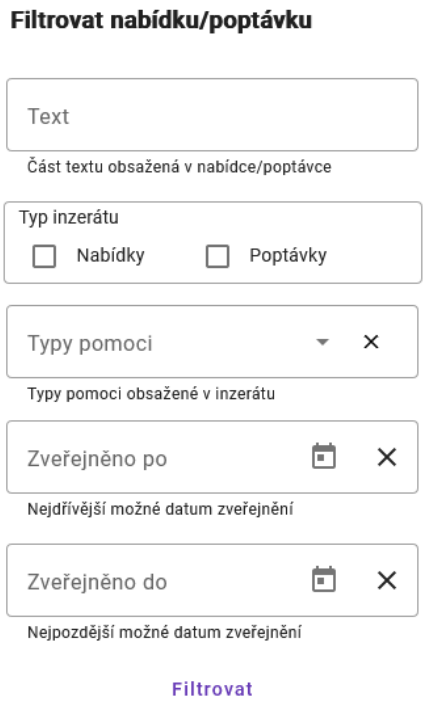
Text - část textu, která je buďto součástí popisu, nebo součástí titulku

Typ inzerátu - nabídka, poptávka, obojí

Typ pomoci - materiální pomoc, odvoz, lékařská pomoc, psychologická pomoc… Je možné vybrat více typů pomoci najednou. Pokud nejsou vybrány žádný, pracuje se s tím jako by byly vybrány všechny.

Časové rozmezí poslední publikace - maximální doba od posledního schválení inzerátu

Náhled filtrovacího formuláře je na obrázku 6.7. Stejně jako na stránce projektů, i zde ukládám filtry a stránkování do url stránky, a tudíž je možné se k nim později vracet. Zároveň, stejně jako u stránky s projekty, je možné inzeráty sdílet emailem a na facebook.

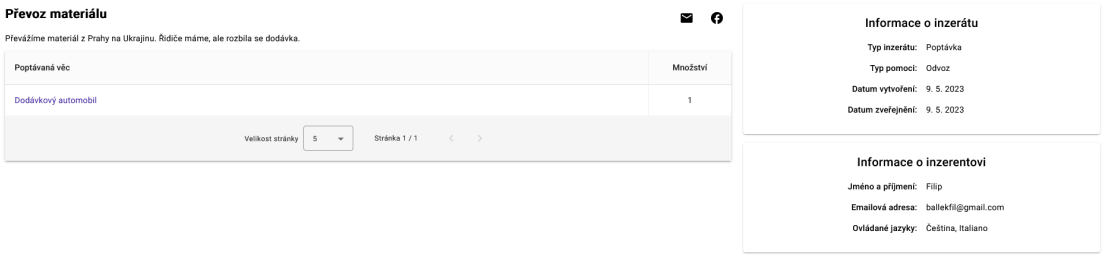


Obrázek 6.7 Filtrování inzerátů

6.8.6 Advertisement detail

Detail inzerátu obsahuje všechny informace, které jsou o něm uživateli dostupné. Plně lokalizovaný titulek a popisek, karta zadavatele s hodnocením a informacemi, které o sobě dovolil zveřejnit a o něco důležitější tabulka s položkami, které jsou v žádosti/nabídce zahrnuty. Přímo v tabulce je viditelný rychlý přehled obsahující název zdroje a poptávané/nabízené množství. Detailnější informace jsou k nahlédnutí v modálním okně po kliknutí na název. Dále je zde jedna

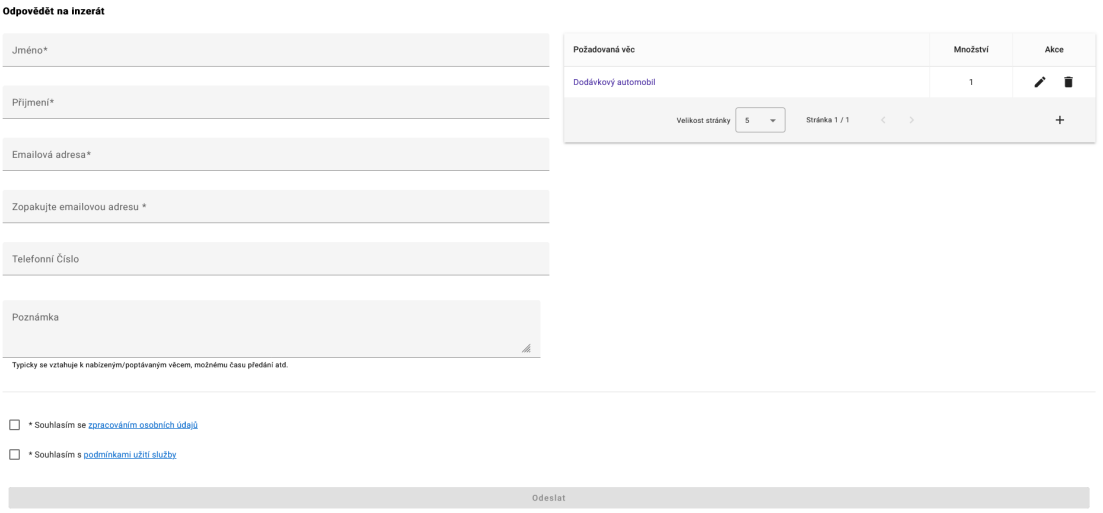
karta se základními informacemi o inzerátu, jako datum publikace, typ inzerátu nebo typ obsažené pomoci. Pod ní se ještě nalézá karta s informacemi o uživateli, který inzerát vytvořil. Detail inzerátu je k nahlédnutí na obrázku 6.8



Obrázek 6.8 Detail inzerátu

6.8.6.1 Odpověď na inzerát

Pod informacemi o inzerátu byl umístěn formulář sloužící k odpovědi na inzerát. Ten si od uživatele vyžaduje osobní údaje, jako je jméno a příjmení, email, a volitelně i telefonní číslo. Zdroje, které chce uživatel využít se zadávají do podobně řešené tabulky, jako je ta u informací o inzerátu. Předvyplněna je tak, aby vyplnila to o co se v inzerátu žádá. Jak již bylo psáno výše, její obsah lze libovolně upravovat. Prvně lze přidat další zdroj. K tomu je určené tlačítko ”plus”na spodku tabulky. Po kliknutí na něj se uživateli zobrazí modální okno. Prvně ve vyhledávacím políčku s listem filtrovaným v databázi uživatel vybere jeden z existujících zdrojů. Vztahuje se k němu omezení, že není možné mít ke jednomu inzerátu vícekrát stejný zdroj. Dále (volitelně) uživatel napíše popis, který se, na rozdíl od obecného popisu zdroje, vztahuje více ke konkrétní nabídce/poptávce. Nakonec ještě zadá množství, které je omezeno pouze tím, že musí být kladné. Před odesláním musí uživatel potvrdit ToSA a PPA (dva samostatné souhlasy). Po odeslání se zobrazí uživateli hláška informující uživatele o úspěšném odeslání a žádající ho o potvrzení emailu odkazem odeslaným na zadanou adresu. Zároveň je vyčištěný formulář pro odeslání odpovědi. Formulář pro tvorbu odpovědi lze vidět na obrázku 6.9



Obrázek 6.9 Formulář pro tvorbu odpovědi na inzerát

6.8.7 Vytvoření inzerátu

Vytvoření inzerátu probíhá pomocí vícekrokového formuláře. Pro přechod mezi kroky je vyžadováno, aby byl předchozí krok validní.

6.8.7.1 Krok 1 – detail inzerátu

Prvně je potřeba vyplnit detail inzerátu. Ten se skládá z typu inzerátu (nabídka/poptávka), typu pomoci v inzerátu (odvoz, manuální práce, psychologická pomoc, …) titulku a popisku. Titulek a popisek lze zadat ve více jazykových variantách. Prvně se vybírá primární jazyk[2]. Z důvodu vybraného způsobu tvorby komponenty pro výběr typu inzerátu nebylo možné přidat zvýraznění chybějící hodnoty v políčku pro typ pomoci, takže na tuto chybějící hodnotu uživatele upozorňuji alespoň U pole s výběrem primárního jazyka je i pole pro výběr aktuálně upravovaného jazyka. Výběr obou jazyků je realizován formou dedikovaných komponent. Před změnou primárního jazyka je uživatel upozorněn, že dojde že tato změna ovlivní i položky vypsané v dalším kroku, a že tyto položky mohu mít prázdný výchozí text. Aktuálně upravovaný jazyk ovlivňuje políčka pouze v tomto kroku. Těmito políčky jsou titulek inzerátu, který vyžaduje aby byl zadaný text alespoň ve výchozím jazyce, a popisek (text inzerátu), který může být prázdný.

6.8.7.2 Krok 2 – zadání nabízených/poptávaných položek

V dalším kroku uživatel vybírá, jaké položky jsou nabízené/poptávané. Je možné vybírat z předpřipravených vzorů. Pro výběr vzorů byla vytvořena univerzálně použitelná komponenta, která umožňuje vyhledávat informace z různých zdrojů (pozor, neplést s komponentou která je např. v komponentě pro výběr zdroje, a která je součástí rozšíření *ngx-mat-select-search*[3]). V tomto případě jsou šablony stahované ze server podle filtru, který se skládá z konkrétního projektu, typu inzerátu, typu pomoci a zadaného textu, který musí být součástí buďto názvu šablony, nebo jejího popisu. Ve posuvném listu s pevnou výškou se pak zobrazí všechny šablony, které vyhovují zadanému vzoru. Po kliknutí na jednu z položek vyskočí modální okno. Okno obsahuje název a popisek vzoru. Na spodu je výrazné upozornění, že při souhlasu s aplikací šablony dojde k přepsání aktuální tabulky s vypsanými věcmi. Po zavření okna mohou nastat dvě varianty:

1. Uživatel klikl na tlačítko se souhlasem
2. Uživatel ukončil modální okno jiným způsobem

V případě kdy uživatel okno ukončí nějakým jiným způsobem, než souhlasem, nestane se nic kromě notifikace, která uživatele informuje, že šablona nebyla použita. Pokud uživatel s použitím tabulky souhlasí, obsah tabulky se přepíše *vylistovanými* věcmi, které mají množství nastavené na jedna a popisek nastavený na prázdný text (který se u *vylistovaných* položek povoluje). Nakonec je ještě uživatel informován hláškou o úspěchu s informací, že byla šablona použita. Na vypsané položky je pouze to omezení, že není možné vypsat nabídku nad dvěma stejnými zdroji. Není tedy ani nutné zadat jakýkoliv prvek.

6.8.7.3 Krok 3 – místo ke kterému se inzerát vztahuje

V tomto kroku uživatel vyplňuje, k jakému místu se inzerát váže. Prvním kolonkou, která je zároveň jako jediná povinná je stát. Dále lze zadat kraj, město, ulice, PSČ a číslo domu. Existence jednotlivých částí adresy není nikde kontrolována, takže je pouze na uživateli, aby je zadal správně.

[2] Jazyk použitý v případě, že není dostupná jazyková varianta kterou má uživatel vybranou
[3] ngx-mat-select-search - https://www.npmjs.com/package/ngx-mat-select-search

6.8.7.4 Krok 4 – Kontakt na zadavatele

Posledním krokem je získání kontaktu na zadavatele. Svůj popis začnu tím, co se vyžaduje po nepřihlášeném uživateli. V takovémto případě je nutné zadat jméno, přímení, email i se zopakováním a volitelně číslo (také se zopakováním). Dále je potřeba zadat detail zadaných údajů, který si přeje zveřejnit. Křestní jméno je zveřejněné vždy. Vybrat lze, jestli si uživatel přeje zveřejnit příjmení, email a telefonní číslo. Tento výběr probíhá formou zaškrtávacích políček. Políčka, které uživatel nezaškrtne, jsou sice uloženy v databázi a přístupné administrátorům, ale nejsou přístupné ostatním uživatelům. Předposlední věcí je výběr jazyků kterými uživatel mluví, ten používá komponentu popsanou dále v sekci 6.8.12.5. Nakonec je ještě požadováno PPA a ToSA. Přihlášenému uživateli se tento krok vůbec nezobrazí.

6.8.7.5 Odeslání inzerátu

Pokud je uživatel příhlášený, probíhá odeslání inzerátu ve třetím kroku. V opačném případě je inzerát odeslán na konci kroku 4. Pro odeslání je nutné, aby všechny vyžadované kroky, byly ve validním stavu. Na konci je uživatel informován o úspěchu či neúspěchu a přesměrován na stránku s projekty.

6.8.8 Odpověď na inzerát

Ve chvíli kdy někdo odešle odpověď na inzerát, inzerentovi přijde email s odkazem na náhled odpovědi a možností přijetí či zamítnutí. Jak inzerentovi, tak respondentovi navíc přijde druhý odkaz který lze využít pouze pro náhled. Náhled obsahuje jednojazyčný text odpovědi, seznam položek zahrnutých v odpovědi, které mohou ale nemusí odpovídat tomu co bylo v inzerátu, a dvě náhledové karty. První karta slouží pro náhled respondenta a je stejná, jako ta u detailu inzerátu ukazující inzerenta (viz. sekce 6.8.6). Druhá náhledová karta obsahuje informace o odpovědi, mezi které patří titulek inzerátu, datum vytvoření odpovědi a stav, ve kterém se odpověď nachází. Pokud je uživatel inzerentem (k otevření použije patřičný odkaz) a odpověď ještě nebyla přijatá ani zamítnutá, jsou pod těmito detaily ještě zobrazeny tlačítka pro zamítnutí a přijmutí. Po kliknutí na libovolné z nich se zobrazí dialog, ve kterém může inzerentem zadat poznámku ke svému rozhodnutí. Po odeslání se zobrazí hláška s výsledkem, a oběma účastníkům (inzerentovi a respodentovi) přijdou emaily informující o provedené změně a poskytující nový odkaz, který nyní slouží pouze k náhledu.

6.8.9 Přihlášení

Na stránce je v tuto chvíli nabízena možnost přihlášení pomocí jména a hesla. Formulář je jednoduchý a nejspíše na něm není nic moc k vysvětlování. Dvě políčka, tlačítko pro přihlášení a odkaz pro přesměrování na stránku s registrací. Při úspěšném přihlášení je uživatel přesměrován na hlavní uživatelskou stránku. Při neúspěšném přihlášení je mu zobrazena hláška popisující co se pokazilo. Z důvodu bezpečnosti však ani ze serveru není sdělováno například to, že bylo správné jméno, ale špatné heslo. Na frontendu je v takovém případě zobrazeno, že bylo špatné jméno nebo heslo.

6.8.10 Registrace

Registrace do aplikace probíhá přes registrační formulář. Po uživateli je vyžadováno zadání uživatelského jména, které smí obsahovat pouze alfa-numerické znaky, emailu který je nutné zadat dvakrát pro ověření, a který musí mít validní formát. Opakovaně je nutné zadat také heslo. K heslu uživateli zobrazuji informaci o tom, jak silné heslo si zvolil. Pro zobrazení síly hesla

je používáno rozšíření @eisberg-labs/ngx-strength-meter[4]. Jediné omezení k heslu je jeho délka, kdy minimum je 8 a maximum 64, a vynucení probíhá jak zde na straně FE, tak poté na straně BE. Své rozhodnutí jsem provedl na základě doporučení agentury OWASP [90]. Z osobních údajů je vyžadováno pouze jméno a příjmení. Zadání telefonního čísla je stejně jako u zadání inzerátů a odpovědí na ně volitelné. Stejné jako už již zmíněné stránky jsou zde i políčka pro ToSA a PPA. Po úspěšném odeslání uživatele přesměruji na stránku s přihlášením a zobrazím hlášku ve kterého informuji o odeslaném emailu na zadanou adresu, ve kterém je odkaz pro potvrzení zadaného emailu, což slouží jako ochrana proti *SPAM* registracím.

6.8.11 Hlavní uživatelská stránka

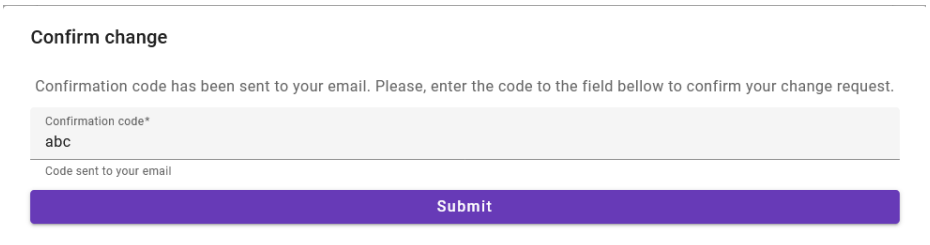
Stránka, která se uživateli zobrazí poté, co se přihlásí přes přihlašovací formulář. Jedná se o rozcestník, který umožňuje uživateli přistupovat k částem stránky, které jsou určeny primárně pro něj. V tuto chvíli jde pouze o stránku s nastavením uživatelských údajů. Do budoucna sem přibude správa inzerátů, které uživatel vypsal a na které odpověděl. Také zde přibude stránka s odpověďmi, které uživatel na své inzeráty obdržel a nebo je naopak na nějaký inzerát odeslal.

6.8.12 Úprava uživatele

Uživatelské rozhraní pro úpravu uživatele sestává z pěti samostatných formulářů. Všechny změny vyžadují nějakou formu konfirmace. K tomu jsou využívány různé konfirmační dialogy. Jeden dialog, je popsán v samostatném odstavci. Druhý dialog pak bude popsán u formuláře který ho využívá. Společným chováním formulářů je, že pokud nejsou změněné, není žádost o změnu odeslána na server a uživateli je pouze zobrazena informační hláška.

6.8.12.1 Dialog s jedním kódem

Dialogem, který je v tuto chvíli využíván pro změnu telefonního čísla, je dialog vyžadující právě jeden kód. V tuto chvíli je tento kód zasílán pouze na email. Typicky to probíhá tak, že se na server odešle žádost o vytvoření „change requestu“ a server odešle zprávu s kódem na emailovou adresu uživatele. Poté klient obdrží odpověď, že žádost byla úspěšně vytvořena (případně nějakou chybovou odpověď, na kterou nějak zareaguje). Uživatel má možnost kód zadat a formulář odeslat. Pokud je kód správný a nedojde k žádné neočekávané chybě, provede se změna a uživateli vyskočí hláška informující ho o úspěchu. V opačném případě dojde k vypsání chybové hlášky informující uživatele o tom, že změna selhala. Další možností je formulář uzavřít bez odeslání. V tom případě se pouze vypíše chybová hláška informující uživatele, že se změna nezdařila, protože formulář s kódem nebyl odeslán. Tento formulář ve stavu, kdy byl zadán kód „abc“, lze vidět na obrázku 6.10

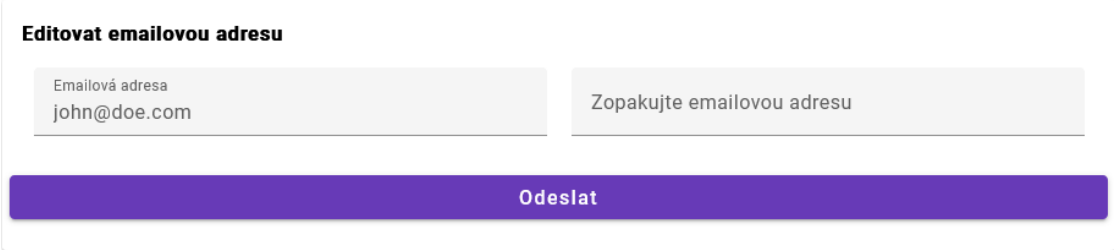


Obrázek 6.10 Konfirmační dialog s jedním kódem

[4] @eisberg-labs/ngx-strength-meter: `https://www.npmjs.com/package/@eisberg-labs/ngx-strength-meter`

6.8.12.2 Formulář pro změnu emailu

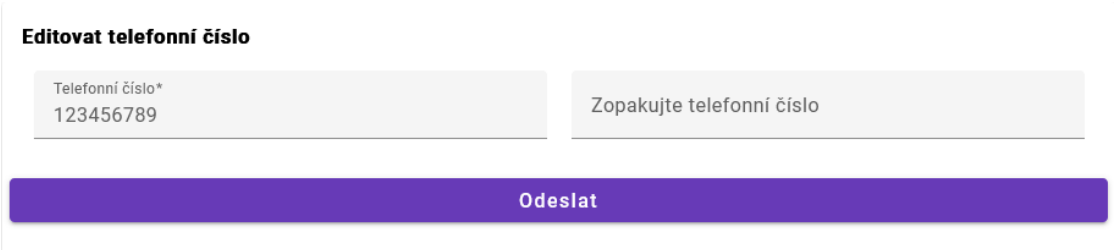
Tento formuář slouží pro změnu emailu. Zadává se druhý email, který je nutné zopakovat. Pokud email není validní, zobrazí se pod políčkem chybová hláška. Podobně pokud se emaily neshodují, zobrazí se chyba u pole pro opakování emailu. Pokud jsou obě hodnoty validní, je možné formulář odeslat. Po odeslání je na server odeslána žádost o vytvoření „change requestu“. Zároveň s tím jsou na původní email uživatele a nově zadaný email odeslané konfirmační kódy. Pokud ze serveru dojde chybová odpověď, je uživateli zobrazena hláška o tom, že se pokus o změnu nezdařil. V opačném případě je uživateli zobrazen modální dialog, ve kterém je informován o nutnosti zadat oba kódy a kde je získat. Formulář může uzavřít bez zadání kódu, v tom případě je uživatel informován o tom, že se změna nezdařila kvůli uzavření bez odeslání. Pokud uživatel kódy zadá a formulář odešle, jsou konfirmační kódy odeslány na server. V případě že žádost selže, je uživateli ukázána náležitá chybová hláška. V opačném případě je uživateli ukázána hláška o úspěchu změny. Formulář ve výchozím stavu, pro uživatele s uloženou emailovou adresou „john@doe.com“, je k nahlédnutí na obrázku 6.11.



Obrázek 6.11 Formulář pro změnu emailu

6.8.12.3 Formulář pro změnu telefonního čísla

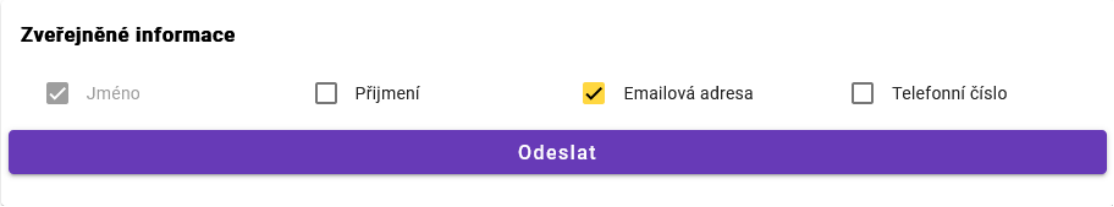
Dalším formulář umožňuje změnu telefonního čísla uživatele. Stejně jako při změně emailu, je i v tomto případě nutné telefonní číslo zopakovat. Analogicky k tomu, co již bylo popsáno u emailu, jsou i validační chyby. Odeslání „change requestu“ a konfirmace jsou v souladu s tím, co bylo popsáno u konfirmačního dialogu s jedním kódem. Odůvodnění proč v tomto případě, narozdíl od změny emailu, stačí jeden kód je to, že v tuto chvíli aplikace neumí konfirmovat telefonní číslo. Proto postačí konfirmace pomocí jednoho kódu zaslaného na email. V budoucnu je ale počítáno s tím, že se zřídí také potvrzení telefonního čísla. Na obrázku 6.12 lze vidět výchozí stav tohoto formuláře, pro uživatele s telefonním číslem „123456789“



Obrázek 6.12 Formulář pro změnu telefonního čísla

6.8.12.4 Formulář pro změnu úrovně zveřejněných informací

Uživatel si může zvolit, jaké údaje o sobě chce zveřejnit. Vyžadováno je vždy jméno. Možnost nezveřejňovat má u příjmení, emailové adresy a telefonního čísla. Tato změna se ihned po odeslání na server uloží, a uživateli je poté zobrazena hláška informujícího o úspěchu. Pokud žádost selže, je přesměrován na chybovou stránku. Na obrázku 6.13 je formulář, který v tomto stavu říká, že si uživatel přeje, aby bylo zveřejněno jeho jméno a emailová adresa. Příjmení a telefonní číslo nemá být veřejné.

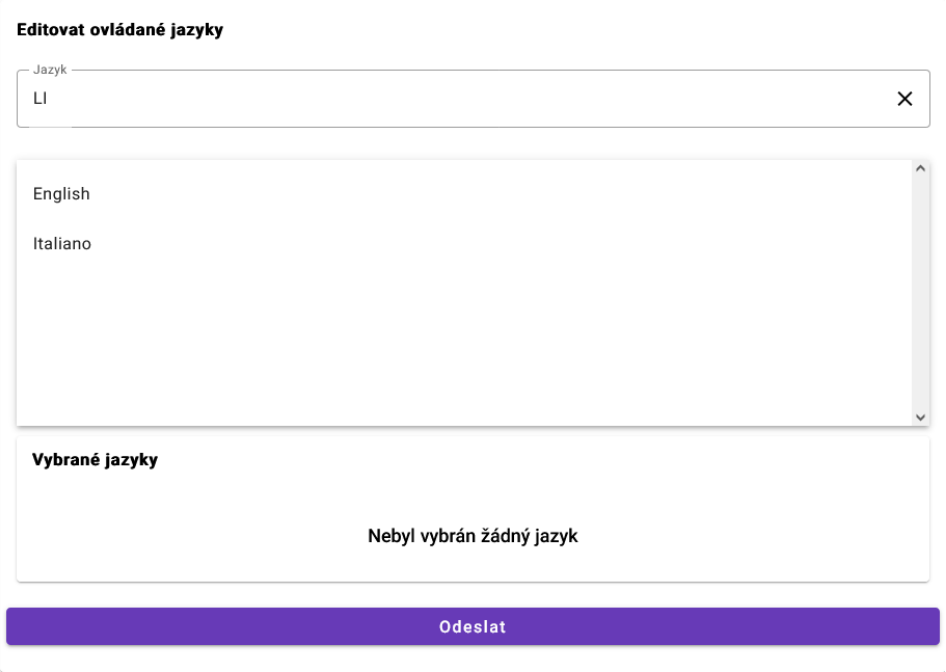


Obrázek 6.13 Formulář pro změnu úrovně zveřejněných detailů

6.8.12.5 Formulář pro jazyky ovládané uživatelem

Formulář umožňuje na svém profilu nastavit, jaké jazyky ovládá. Ty jsou pak zobrazeny u inzerátu. Vybírat lze několika jazyky, které jsou nadmnožinou jazyků nabízených pro rozhraní aplikace a multijazyčné texty. Mezi jazyky lze vyhledávat pomocí vyhledávacího pole, přičemž vyhledávání probíhá lokálně. Zvolené jazyky se pak zobrazí pod listem ve formě „chipsů“[5]. Stejně jako při změně úrovně zveřejněných informací, je změna i v tomto případě ihned po odeslání na server uložena, a uživateli je poté zobrazena hláška informujícího o úspěchu. Pokud změna selže, je přesměrován na chybovou stránku. Na obrázku 6.14 je vidět formulář, ve kterém uživatel nemá vybraný žádný jazyk a ve vyhledávacím poli má zadáno „LI“, což odpovídá právě dvěma jazykům dostupným v aplikaci.

[5] Angular Material Chips - `https://material.angular.io/components/chips/overview`



Obrázek 6.14 Formulář pro změnu jazyků ovládaných uživatelem

Kapitola 7
Implementace backendu

V rámci implementace backendu bylo vytvořeno API, na které se může frontend implementovaný v předchozí kapitole dotazovat. Zároveň bylo vytvořené API, na které se mohou dotazovat koordinátoři a administrátoři. Je využíván návrh API ze sekce 4.4.

7.1 Vytvoření Spring boot projektu

Pro vytvoření projektu byl využit *Spring intializr* (`https://start.spring.io/`). Jedná se o webovou aplikaci, která poskytuje jednoduché rozhraní pro úvodní tvorbu Spring boot projektů. Použité nastavení je možné vidět na obrázku 7.1. Jako jméno skupiny bylo zvoleno doménové jméno, které je pro platformu registrované OpenDataLabem. Jako jméno projektu byl pak zvolen jednoduše „backend“, protože se jedná o backendovou část aplikace. V následujícím výčtu jsou okomentovány některé vybrané závislosti.

Spring Session Rozhraní a a implementace správy uživatelských relací [91]. V aplikaci je využita varianta, která po přihlášení vrací *cookie* s identifikátorem relace, a která ukládá informace o relaci do databáze s použitím (*JDBC HttpSession*) [92].

Spring Security Jedná se o framework, který dodává autentizaci (kontrola identity uživatele), autorizaci (kontrola oprávnění uživatele k danému úkonu) a také ochranu proti známým typům útoků. [93, 94, 95]
Tento framework poskytuje rozsáhlou podporu akcí zmíněných v předchozí větě. Zároveň nabízí integraci s ostatními knihovnami. Mimo jiné ho je možné snadno lokalizovat, což, vzhledem k integraci lokalizace skrz celou aplikaci, může najít své využití do budoucna.

Spring Data JPA *Spring Data* je projekt, jehož cílem je poskytnout známý a konzistentní model pro přístup k datům. Skládá se z několika na sobě nezávislých projektů. [96] Jedním z nich je *Spring Data JPA*, stavějící na *JPA*, resp. jeho implementaci Hibernate. Mezi jeho funkce patří třeba podpora stránkování či validace dotazů v době závadění (startování) aplikace. [97]

REST Repositories Rozšíření, které staví na *Spring Data Repositories* a dává je k dispozici, jako *REST* zdroje [98]. Jednalo se o omyl při úvodním výběru a z aplikace byl odstraněn smazáním z Gradle souboru.

Spring Web Závislost, která byla použita pro stavbu API. Byl ponechán výchozí server.

Quartz Scheduler Tato knihovna umožňuje vytvářet jak jednoduché, tak komplexní plánové akce. [99] Ačkoliv v této fázi nebyla nakonec využita, jsou v aplikaci místa, u kterých je možné

do budoucna spouštět pravidelné akce. Příkladem může být automatické čištění uživatelů, kteří si neaktivovali email. Tito uživatelé plní databázi. To by sám o sobě nemusel být takový problém, protože tyto jejich data nezabírají nějak velké množství místa. Na uživatele jsou ale typicky navázané další věci, jako jsou inzeráty a odpovědi. Mazáním těchto uživatelů po nějakém rozumně stanovém čase by mohlo přinést úsporu místa.

Apache Freemarker Jedná se o knihovnu založenou na Javě, která umožňuje generovat textový výstup založený na šablonách a měnících se datech (tzv. *template engine*) [100]. Tato knihovna byla nakonec v průběhu tvorby projektu nahrazena knihovnou *Thymeleaf*, která se autorovi lépe používala.

7.2 Nastavení aplikace

Pro správnou funkci aplikace bylo po stažení z *Initializru* nutné dodat několik nastavení. Nastavení aplikace bylo prováděno na jednom ze tří míst. Prvním z těchto míst jsou konfigurační třídy. Jedná se o třídy označené pomocí anotace `Configuration` [102]. V těch bylo prováděno především nastavování jednotlivých *Bean*, což jsou objekty spravované *Spring IoC Kontejnerem* a tvořící páteř aplikace[103]. Dalším místem byly *properties* soubory. V nich jsou nastovány některé konfigurační parametry springu, jako je například nastavení připojení k *SMTP serveru* (server se kterým aplikace komunikuje, pokud potřebuje odeslat email), nebo úroveň detailu *logovaných* zpráv.

7.2.1 Připojení k databázi

Jednou z nejdůležitějších věcí pro fungování celé aplikace je to, že může přistupovat k nějaké databázi, do které jsou ukládány data aplikace a zároveň jsou z ní čteny. Nastavení přístupu do database proběhlo v *properties* souboru. Nastaven byl přístup do databáze. Ten zahrnuje její JDBC url, uživatelské jméno a heslo. Kromě toho je nastaven ještě ovladač, který je pro přístup k databázi použit. Jelikož databáze používá databázi *PostgreSQL*, byl vybrán ovladač *org.postgresql.Driver*.

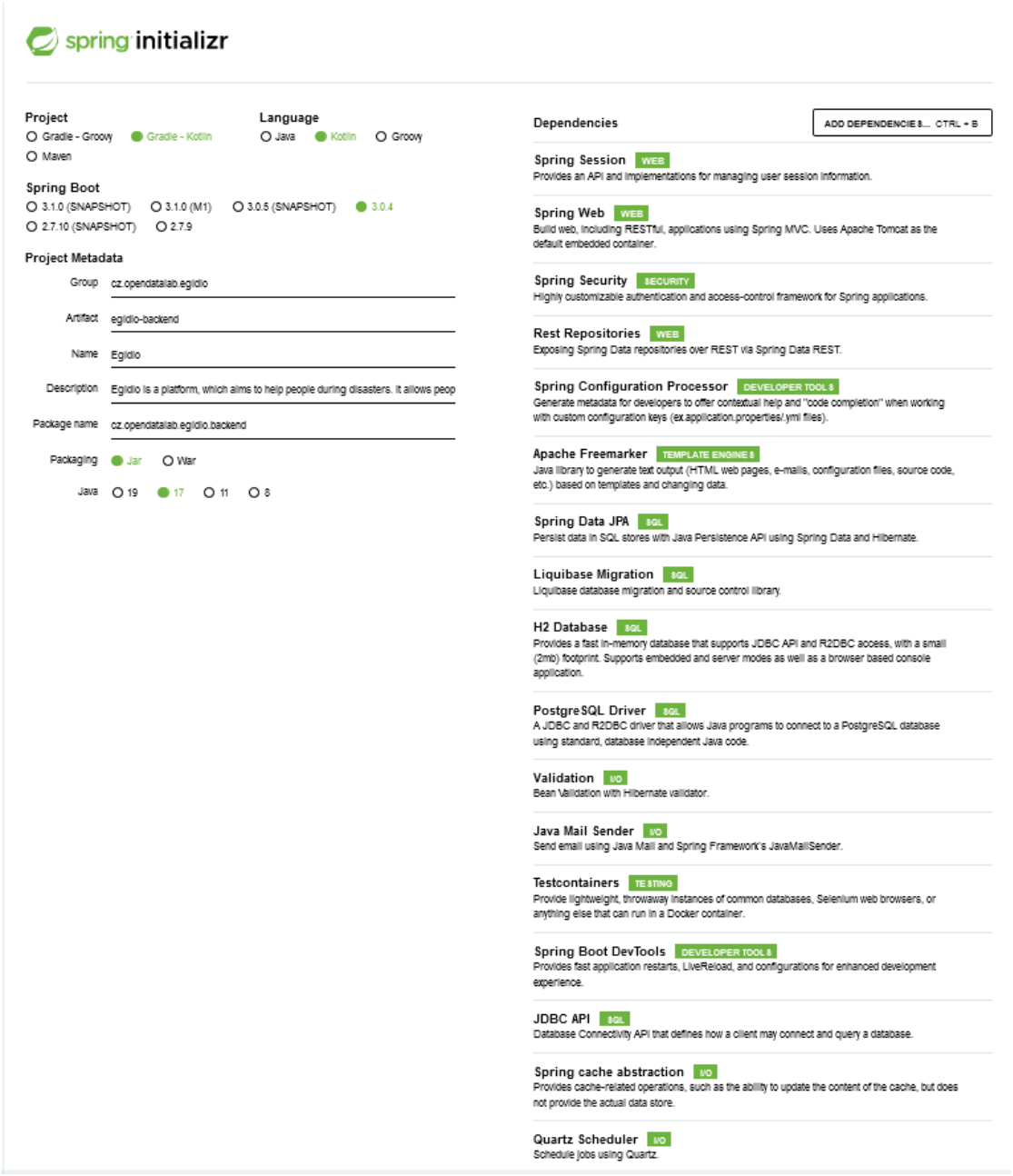
7.2.2 Nastavení času

Pro práci s datem a časem je využíváno API dodávané jako součást *Java 8*. Při získávání času jsou využívány ty varianty času, které jako jeden z argumentů berou třídu *Clock*. To mimo jiné umožňuje použití tohoto API jednoduše testovat, dodáním hodin s pevným časem. Pro produkční použití byly nastaveny hodiny dodávající aktuální systémový čas pro časovou zónu *UTC*[1]. Hodiny jsou nastaveny v konfiguračním souboru, jako *Bean*. Nastavení je prováděno v konfigurační třídě *DateTimeConfiguration*.

7.2.3 Nastavení přístupu k emailovému serveru

Jako první věc, kterou bylo nutné pro odesílání emailů nakonfigurovat, byl přístup k SMTP serveru. Přístupy byly nakonfigurovány v *properties* souboru. Konkrétně bylo nastaveno uživatelské jméno, heslo, adresa serveru, jeho port a protokol. Přístupové údaje jsou předávány pomocí proměnných prostředí (*env variables*). Pro testovací účely autor využil testovací účet na *Google* (přístup k tomuto účtu autor nezveřejnil. Kromě toho byly ještě dodány některé další konfigurační parametry. Jde například o dobu, po kterou má být udržováno spojení se SMTP serverem,

[1] Posun o dvě hodiny dozadu oproti Středoevropskému letnímu času



Obrázek 7.1 Spring initializr - úvodní nastavení projektu [101]

nebo kterou má čekat na odpověď. Tím se mimo jiné předejde zamrznutí aplikace, pokud server neodpovídá.

7.2.4 Nastavení zabezpečení

Jak již bylo zmíněno v první sekci této kapitoly, pro zabezpečení aplikace je použito *Spring Security*. Při přidání tohoto frameworku do projektu je ochrana rovnou aktivována. Základní nastavení však pro platformu nepostačovalo. Pro všechny koncové body požadovalo, aby byl uživatel přihlášený. Přihlášení navíc mělo pouze jednoho, společného, uživatele. Pro konfiguraci zabezpečení dle potřeb aplikace byla vytvořena konfigurační třída `SecurityConfiguration`. V té je nadefinováno několik *Bean*, které pomáhaí se zabezpečením aplikace. Tyto *bean* jsou popsány ve výčtu níže.

stringTokenHasher Instance třídy implementující rozhraní `Hasher<String>` pro hashování textových řetězců. Používa se pro vygenerovaných přístupových tokenů, aby nebyly v databázi uloženy ve své surové formě, a ověřování předaných tokenů s tokeny uloženými v databázi. K hashování je využíván algoritmus *SHA3-256*.

passwordEncoder Instance třídy implementující rozhraní `PasswordEncoder` poskytované *Springem*. Její využití je, podobně jako u stringTokenHasheru, pro hashování hesel uživatele, a ověření pro hashování hesla při jeho ověřování během přihlášení uživatele.

daoAuthenticationProvider *Bean*, kterou Spring využívá pro získávání dat uživatelů. Je využívaná instance třídy `DaoAuthenticationProvider`, která je poskytovaná *Springem*. Před návratem je této instanci nastaven `passwordEncoder`, kterým je *bean* zmíněná výše, a *bean* pro rozhraní `UserDetailsService` poskytované Springem. Touto *bean* je třída UserDetailsServiceImpl, která byla vytvořena pro tuto aplikaci, a která získává uživatelská data za pomoci `UserService`, což je servisní třída pro aplikační entitu User. Přihlásit je možné pouze registrovaného uživatele, přihlášení neregistrovaného uživatele není podporované. Přihlášení takového uživatele by ani nedávalo příliš smysl. K akcím, které jsou mu v tuto chvíli k dispozici, dostává přístup přes token posílaný v URL. Jeho ověření tedy většinou v servisní třídě entity, kde se zadaný token porovnává s tokenem v databázi u konkrétního záznamu.

corsConfigurer Jedním z mechanizmů, které Spring Security nabízí, je regulace *CORS*. O tom co *CORS* je a k čemu je takováto regulace dobrá bylo napsáno více v sekci 5.5.3. V této *bean* je nastaveno, že se při tzv. předletové *CORS* přípravě pro libovolný endpoint vrátí odpověď která říká, že je možné použít libovolnou *HTTP* metodu, s libovolnými hlavičkami, a že je možné v *cross-origin* požadavcích zasílat přihlašovací údaje, což bylo nutné pro správné fungování přihlašování na platformě (bez této možnosti nebyl posílán *session cookie*). Domény, ze kterých by měl prohlížeč požadavky povolovat, jsou nastavené na místní (localhost) adresy z důvodu funkčnosti při vývoji, a na `https://www.egidio.opendatalab.cz`, na které bude platforma hostována.

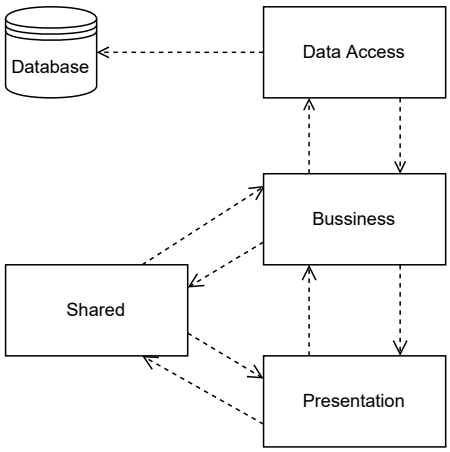
filterChain Část autentifikace a autorizace je řešena pomocí řetězce bezpečnostních filtrů. Třída (`SecurityFilterChain`), jejíž je tato *bean* instancí, je využitá jedním z filtrů k určení, toho jaké další filtry mají být zavolané. Těchto filtrů může být v aplikaci více. [104] Tato konkrétní *bean* určuje, že žádost o přihlášení s přihlašovacím jménem a heslem (*formLogin*) probíhá na adrese */auth/login* relativní vůči adrese, na které se aplikace nachází. Dále určuje, že tyto žádosti o přihlášení mají mít obsah, jehož *MIME* typ je *application/x-www-form-urlencoded* a obsahuje položky *username* s přihlašovacím jménem a *"password"* s heslem. Dále určuje, že po přihlášení ani odhlášení uživatel nemá být nikám přesměrován (využití *bean* `successfulLoginHandler` a `successfulLogoutHandler`). Nakonec ještě povoluje řízení CORS a vypíná CSRF ochranu, z důvodu nefunkčnosti ani při použití reverzní proxy.

7.3 Řízení změn v databázi

Jednotlivé změny ve struktuře databáze jsou prováděny za pomoci Liquibase changelogu. Prvotní struktura databáze byla vytvořena *Hibernate*, podle vytvořených entit (o těchto entitách více v podsekci 7.7.2 níže). Následně byl z databáze za použití nástroje liquibase generate-changelog vygenerovaný SQL skript, který vytváří tabulky a další objekty v databázi. Liquibase změny seskupuje do tzv. *changelogů*, což mohou být například XML schemata, nebo, jako v případě tohoto projektu, YAML schemata [105]. Ve složce se změnami byla vytvořena složka *init*, která obsahuje *changelogy* a další soubory (*SQL*) použité k inicializaci databáze. Mimo jiné je zde prováděna inicializace datových typů pro enumerované hodnoty, a nastavení tzv. *partial unique indexů*, které v tomto případě slouží ke kontrole toho, že jedna emailová adresa není použitá pro dva registrované uživatele (může však být použita pro registrovaného uživatele a jednoho nebo více neregistrovaných uživatelů). Obecná struktura adresáře pro Liquibase je taková, že v kořenu je soubor pojmenovaný *db.changelog-master.yaml*, ten odkazuje na hlavní *changelogy* v adresářích uvnitř kořenového adresáře. Tyto changelogy následně odkazují na soubory s jednotlivými změnami. S takovouto strukturou půjde i v budoucnu seskupovat změny podle toho, jaké verze aplikace se týkají.

7.4 Události v aplikaci

V aplikaci jsou využívány notifikace o událostech. Pro aplikaci bylo vytvořeno několik typů událostí, přičemž všechny jsou organizovány na *bussiness* vrstvě v podsložce *events*. V tuto chvíli jsou vytvořeny události pro tři entity. Těmi jsou *Advertisement, AdvertisementResponse a User*. Událostmi jsou například vytvoření inzerátu, zveřejnění odpovědi či vytvoření žádosti o změnu uživatelského emailu. Pro každý typ události byla navíc vytvořena třída naplnitelná daty, která jsou pro událost důležitá. Typicky se jedná o data vytažená z entity, ke které se událost vztahuje. Například pro událost ke zveřejnění inzerátu třída obsahu *slug*, pomocí kterého se lze na instanci odkázat, titulek inzerátu ve formátu vícejazyčného textu a email inzerenta. Tyto tři kusy dat inzerátu byly vybrány z toho důvodu, že událost je zatím využívána pouze pro posílání emailu, a právě tato tři pole jsou k této akci potřeba. Důvod proč není posílán celý inzerát je ten, že byla snaha minimalizovat množství dat, které se dostane ven mimo dosah metody. Minimalizace množství dat, které jsou posílány dále, se v programování obecně považuje za dobrý postup (viz. také Déméteřin zákon či zákon minimální znalosti [106]). Události jsou publikované za pomoci objektu implementujícího rozhraní `ApplicationEventPublisher`, jehož instance je dodávána *Springem* bez nutnosti cokoliv nastavovat. Jednotlivé události jsou odchytávány v komponentech, které byly nazvány `EventNameListenersGroup`, přičemž *EventName* je v jejich názvech nahrazeno jménem odchytávané události. Existuje tedy například `AdvertisementApprovedEventListenersGroup`, nebo `AdvertisementResolvedEventListenersGroup`. Tyto třída seskupuje posluchače událostí (anglicky *EventListener*) pro právě jednu událost. Posluchačem je vždy metoda v těchto třídách, anotována jednou z dostupných anotací. V tuto chvíli je u všech posluchačů použita pouze anotace `TransactionalEventListener`, označující ty posluchače, kteří jsou volaní podle toho, v jaké fázi se aktuální transakce nachází. U těchto typů posluchačů je nutné dát si pozor na to, že pokud dojde ke zveřejnění události mimo transakci, takto označený posluchač událost zahodí. [107] Možnosti zavolat posluchače v určité fázi transakce je využíváno především k tomu, aby k odeslání informačních emailů došlo až poté, co je změna, ke které se email vztahuje, uložená (byl úspěšně proveden *COMMIT* - fáze *AFTER_COMMIT* [108]). Tím se předejde tomu, že dojde k odeslání emailu o provedení nějaké změny, která pak ale selže při ukládání do databáze.



Obrázek 7.2 Vrtvy backendu a jejich propojení

7.5 Posílání emailových zpráv

Šablony pro emaily jsou uložené ve složce */source/backend/src/main/resources/templates/emails* (dále v textu jen *templatesDir*), respektive v jejích podadresářích. Podadresáře jsou vytvořené tak aby logicky rozdělili šablony podle toho k jaké entitě a adresátovi se vztahují. Například *templatesDir/advertisement_response/advertiser* obsahuje ty šablony, které se vztahují k odpovědi na inzerát, a jsou odesílané inzerentovi (například zpráva o přijetí inzerátu). Pro posílání emailů byla vytvořeno několik servisních tříd. Každá tato třída má jednu nebo více metod, a každá slouží pro odeslání jednoho nebo více typů zpráv. Pro jednotlivé zprávy byly navíc vytvořeny datové třídy, které slouží pro předání dat, které zpráva potřebuje. Tento návrh umožňuje soustředit odesílání zpráv na malé množství míst. Zároveň je použitím datových tříd docíleno toho, že se do servisních tříd nedostává více informací, než kolik jich k odeslání zprávy opravdu potřebují (opět viz. Déméteřin zákon [106]).

7.6 Změna nastavení uživatele

Změna nastavení uživatele probíhá na koncových bodech spadajících pod *"/user"*. Pro každý typ změny byl vytvořen jeden nebo více koncových bodů. Pokud změna nevyžaduje další potvrzení, byl pro ni vytvořen jeden koncový bod (změna ovládaných jazyků a změna publikovaných detailů kontaktu). U změn, které vyžadují zadání jednoho (změna telefonního čísla) nebo více (změna emailové adresy) potvrzujících kódů byly vytvořeny dva koncové body. Jeden pro vytvoření požadavku na změnu a druhý na potvrzení požadavku. Logika změn je ve třídě `UserServiceImpl`, implementující rozhraní `UserServiceImpl`

7.7 Struktura aplikace

Aplikace byla rozdělena na 3 různé vrstvy (rychlý náhled na obrázku 7.2). Jde o jeden z principů návrhů aplikace, jehož následování by mělo vést k lépe udržovatelné aplikaci, než kdyby byla aplikace monolitická. Vazby mezi těmito vrstvami jsou maximálně přes jednu úroveň. Kromě těchto oddělených vrstev je v aplikaci přítomná ještě skupina kódu, která je nazvaná *shared*, a je využitá skrz všemi vrstvami.

7.7.1 Vrstva pro přístup k datům (*Data Access*)

Jedná se o vrstvu, jejímž jediným úkolem je získat data z nějakého místa, na kterém jsou uloženy (v tomto případě z databáze). V tuto chvíli se skládá pouze z repozitářů implementujících rozhraní `JpaRepository<T, I>`, kde T je typ ukládané entity a I je typ primárního klíče. Toto rozhraní je rozšířením rozhraní `CrudRepository<T, I>` a `PagingAndSortingRepository<T, I>`. Mimo jiné tedy poskytuje metody pro ukládání nových a upravených entit, jejich mazání a čtení z místa uložení. Kromě toho ještě přidává schopnost stránkování výsledku. Výhodou je, že není nutné psát logiku těchto metod. Hodně si jich framework umí kompletně vygenerovat sám, a u další části mu lze pouze dát *JPQL* dotaz nezávislý na databázi, a on si z něj pak vygeneruje zbytek logiky. V případě nutnosti je mu dokonce možné dát i SQL dotaz nativní pro databázový stroj. [109]

7.7.2 Business vrstva

Na této vrstvě se nachází **doménové entity**, které reprezentují doménové objekty popsané v sekci 4.1 rozšířené o atributy potřebné pro fungování aplikace, jako je interní ID, verze entity nebo entita či kolekce entit se kterými má daná entita nějaký vztah. Entitám a jejich atributům byly dodány informace k validaci a mapování na databázovou strukturu primárně pomocí anotací dodávaných jako součást *JPA* a *Spring Data*. Ve vyjímečných případech byly použity i anotace z *Hibernate*. Důvodem snahy vyhnout se v co největší možné míře anotacím z Hibernate bylo to, že se tím zvyšuje provázání s konkrétní implementací *JPA* (tedy s *Hibernate*). Pokud by projekt někdy v budoucnu přecházel na jinou implementaci *JPA* (například z toho důvodu, že by byla nalezena knihovna která má lepší podporu *Kotlin*), bude nutné tyto *Hibernate* anotace nahradit buďto vlastními, nebo anotacemi z nové ORM knihovny. *JPA* anotace bude možné ponechat.

K entitám jsou ještě vytvořené **projekce**. Jedná se o částečné pohledy na data [110] a v aplikaci jsou využívané pro případy, kdy není nutně potřeba celá entita, ale jen její část. Výhodou je, že z databáze nejsou vytahována zbytečná data. Příkladem může být například projekce Projektu na datovou třídu s typem katastrofy a stavem projektu. Bez projekce (nebo podobného mechanizmu) by musel být vytažený celý projekt včetně všech jeho textů a navázaných dat, a osekání by proběhlo až v aplikaci. To by vedlo na to, že budou mezi aplikací a databází zbytečně posílána i data, jako je titulek či popis projektu, která budou ihned po osekání výsledku na požadovaný pár zahozena. To by bylo obzvlášť nepříjemné ve chvíli, kdy je aplikace a databáze na jiném stroji či dokonce úplně jiné síti.

Podčástí business vrstvy jsou i **rozhraní servisních tříd a jejich implementace**. Díky tomu, že je odděleno rozhraní od implementace je možné v budoucnu nahradit momentálně používanou implementaci za jinou, nebo vytvářet v testech *mock* místo reálné instance (o *mockovaných* objektech bude napsáno více v kapitole 8) Tyto třídy si poskytují služby navzájem, a zároveň je poskytují prezentační vrstvě. Nachází se v nich většina logiky jednotlivých akcí nad entitami získávanými z repozitářů na perzistentní vrstvě. Takovou akcí je například vytvoření inzerátu nebo jeho schválení. Je zde mimo jiné prováděna i dodatečné kontrola toho, zda lze akci provést. Kontrolováno je zde například to, jestli schvalovaný inzerát již není schválený, nebo uzavřený. Navíc je na této úrovni kontrolováno oprávnění uživatele k akci. Kontrola je prováděna dvěma způsoby. Jedním ze způsobů je kontrola pomocí anotací poskytovaných jako součást Spring Security. Pomocí těch je možné například povolit přístup všem, nepovolit ho nikomu, nebo ho povolit jen uživatelům s určitou rolí (případně třeba jen přihlášeným uživatelům) [111]. Pro přístup uživatelům, kteří mají roli koordinátora, nebo vyšší, byla vytvořena vlastní anotace `PermitCoordinator`, kontrolující zda byla přiřazena alespoň jedna z rolí. Vytvoření a využití této anotace a využití anotace `PermitAll` povolující přístup všem uživatelům lze vidět ve výpisu kódu 1. Další způsob kontroly pracuje s konkrétními entitami. Ověřuje například to, jestli uživatel entitu vlastní, nebo jestli má alespoň dostatečně vysokou roli na provedení akce bez ohledu na vlastnictví.

Zbylé části jsou události a posluchači události ze sekce 7.4, vlastní výjimky, validátory a validační konstanty.

```
@PreAuthorize("hasAnyAuthority('COORDINATOR','ADMIN')")
annotation class PermitCoordinator()

interface AdvertisementService {
    ...
    @PermitAll
    fun getPage(
        filteredPageRequest : CustomFilteredPageRequest<AdvertisementFilter>
    ) : CustomPage<Advertisement>
    ...
    @PermitCoordinator
    fun publishAdvertisement(slug : String)
    ...
}
```

Výpis kódu 1 Využití anotací PermitAll a PermitCoordinator

7.7.3 Prezentační vrstva

Nastavení koncových bodů API je prováděno v **kontrolerech**. Kontrolery jsou anotované pomocí *Spring Web* anotace `RestController`. Každý kontroler má přiřazenou cestu, která začíná od kořene. Příklad takové cesty je */advertisement* pro *AdvertisementController*, tedy kontroler spravující koncové body vztahující se k inzerátům. Každý kontroler v aplikaci obsahuje jeden nebo více koncových bodů. Cesta k jednotlivým koncovým bodům je úvaděna relativně vůči cestě ke kontroleru. Implementaci takového kontroleru lze vidět ve výpisu kódu 2. Definice je, stejně jako v případě servisních tříd, rozdělena na rozhraní a implementaci. Důvod k tomuto rozdělení je stejný, jako v případě zmíněných servisních tříd. Člověka znalého návrhů *REST* rozhraní může překvapit, že je k získání stránky zdrojů využívána *HTTP POST* metoda. Důvodem je, že data k žádosti objektu jsou komplexní objekt (obsahují velikost a číslo stránky společně s filtrem obsahujícím objekty které jsou již složitější, jako je vícejazyčný text). Pro *HTTP GET* by dle specifikace nemělo být posílané tělo požadavku. Vše by mělo být posíláno v URL. Posílat takováto data v URL by však bylo zbytečně složité a navíc nepraktické. Specifikace *HTTP POST* říká, že zdroj zpracuje reprezentaci v požadavku podle jeho vlastní sémantiky. [112] Neříká však, že musí být nutně vytvořen nový zdroj, nebo že musí být upravena nějaká data. Je tedy možné ho využít i k takovémuto získání stránky s objekty. Nevýhodou je složitější cachování výsledku oproti GET requestu (je potřeba explicitně zahrnout informaci o čerstvosti dat a hlavičku *Content-Location*. [112])

Pokud je odpovědí na volání endpointu objekt, je vracen některý z objektů pro přenos dat (dále jen **DTO**[2]). Jedná se o další z možných pohledů na data (dokumentace Spring Data ho dokonce zmiňuje, jako jednu z variant projekce[110]). Výhod které toto řešení přináší je několik. Stejně jako u projekcí lze zmínit snížení množství přenášených dat. Zároveň lze takto skrýt implementační detaily před vnějším světem (což ale vzhledem k tomu, že se jedná o open source projekt, ztrácí v tomto případě význam). Co ale význam má je omezení informací, které ten kdo volá API dostane. Ne vždy je žádoucí, aby každý, kdo si požádá o nějaký zdroj, dostal všechny informace. *DTO* umožňuje nejen určit jaký bude maximální rozsah informací, které půjdou ven,

[2] Zkratka pro data transfer object.

ale zároveň umožňuje snížit i minimální rozsah informací tím, že některé vlastnosti na rozdíl od entity udělá dobrovolné - *nullable*.

7.7.4 Sdílený kód

Pro převod mezi *business entitami* a *DTO* jsou využívány **konvertory**. Metoda pro převod je přidávána vždy, když je potřeba provést převod mezi nějakým párem, který ještě převáděn nebyl. Díky tomu, pokud dojde ke změně struktur jednoho ze dvou účastníků převodu, bude nutné provést úpravu pouze na tomto jednom místě.

```
@RestController
@RequestMapping(
    name = "Resource",
    path = ["/resource"]
)
class ResourceControllerImpl(
    private val resourceService: ResourceService,
    private val resourceConverter: ResourceConverter
) : ResourceController {
    @ResponseStatus(HttpStatus.OK)
    @PostMapping(
        name = "findAllByName",
        path = ["/all"]
    )
    override fun getPage(
        @RequestBody
        customFilteredPageRequest: CustomFilteredPageRequest<ResourceFilter>
    ): CustomPage<ResourceShortDto> {
        return resourceService
            .getFilteredPage(customFilteredPageRequest)
            .map(resourceConverter::convertToShortDto)
    }

    @PostMapping(
        name = "create",
        path = [""]
    )
    override fun create(
        @RequestBody createDto: ResourceCreateDto
    ) : ResponseEntity<String> {
        return ResponseEntity
            .status(HttpStatus.CREATED)
            .body(resourceService.create(createDto).slug)
    }

    @ResponseStatus(HttpStatus.OK)
    @GetMapping(
        name = "getBySlug",
        path = ["/{slug}"]
    )
    override fun getBySlug(
        @PathVariable("slug") slug: String
    ): ResourceDto {
        return resourceService
            .getBySlug(slug)
            .let(resourceConverter::convertToDto)
    }
}
```

Výpis kódu 2 Příklad kontroleru z aplikace

Kapitola 8
Testování

V práci byly implementovány jednotkové testy pro testování backendu, a také byly sepsány testovací scénáře, které jsou využity při uživatelském testování. Tyto scénáře pokrývají většinu funkcionalit. Pokrytí kódu backendu jednotkovými testy není ideální, ale je v souladu s běžnými softwarovými projekty. Do budoucna by bylo vhodné implementovat také integrační testy. V tuto chvíli je však dostatečné otestování aplikace následováním sepsaných scénářů.

8.1 Testování backendu

Pro backend byly vytvořeny jednotkové testy, které pokrývají základní funkcionalitu aplikace.

Jednotkové testy jsou takové testy, které vždy pokrývají co nejmenší jednotku kódu (typicky třídu, metodu, funkci…). Takovéto testy by měly být nezávislé na zbytku aplikace. Zároveň by ideálně měly při každém spuštění vrátit stejný výsledek. Výjimkou mohou být testy s náhodně generovanými vstupními daty.

8.1.1 JUnit

Pro testování kódu v Javě byl využit testovací framework *JUnit*. *JUnit* je dodáván při vytvoření *Spring boot* projektu, a ve světě Javy je již dlouho standardem.

JUnit Framework ve verzi 5 se skládá z několika modulů ze tří různých podprojektů. Jedním z těchto podprojektů je *JUnit Platform*, který slouží jako základ pro spouštění testů na JVM. Druhým podprojektem je *JUnit Jupiter*, který je kombinací dvou různých modelů pro psaní testů a rozšíření v JUnit 5. *JUnit Vintage* je třetím podprojektem. Tento podprojekt poskytuje instanci implementující rozhraní *TestEngine*, usnadňující objevení a spuštění testů pro určitý programovací model.[113, 114]

Důvodem k výběru tohoto frameworku byla především podpora ze strany *Spring bootu*, stejně jako podpora ze strany *IDE*. Jeho funkcionality jsou navíc více než dostatečné pro potřeby této práce.

8.1.2 Mockito

Jednotkové testy by nemohly jednoduše fungovat bez možnosti vytvořit tzv. Mock rozhraní. Jedná o implementaci objektů, která simuluje chování skutečných objektů u těch částí, které jsou potřeba v testech[115]. Díky tomu se autor při psaní testů servisních tříd mohl soustředit na funkčnost samotné servisní třídy, a nemusel řešit fungování ostatních objektů, na kterých je servisní třída závislá.

8.1.3 Výsledek testů

Všechny jednotkové testy prošly. Bylo pokryto 17% procent kódu 45 testy. Toto pokrytí není nejnižší, mohlo by však být vyšší. Pro každý kus kódu by bylo vhodné dopsat nějaký test. Také by bylo dobré mít pokryto více mezních případů. Unit testy navíc kontrolují pouze to jak se jednotlivé části chovají v izolaci. Do budoucna bude tedy vhodné doplnit i integrační testy, které by otestovaly spolupráci mezi komponentami.

8.2 Uživatelské testování

Pro uživatelské testování bylo vytvořeno několik testovacích scénářů. Tyto scénáře popisují, jak se má při testování postupovat a jaké jsou očekávané výsledky. Testovací scénáře byly sepisány tak, aby byly pokryty všechny případy užití vzniklé v sekci 3.3, a pro které je v aplikaci v tuto chvíli implementována funkcionalita. Samotné testovací scénáře jsou z důvodu velkého rozsahu (42 stránek) vložené jako příloha. Jejich struktura je následující:

Motivace proč se test provádí, co je jeho cílem

Předpoklady co musí platit před tím, než začne uživatel test provádět

Postup testování jednotlivé kroky nutné k provedení testu

Očekávaný výsledek jaký by měl být stav na konci testu

Pokrytí jednotlivých případů užití testovacími scénáři bylo vyznačeno v tabulce 8.1 a tabulce 8.2. Důvodem k rozdělení do dvou tabulek bylo zpřehlednění zobrazovaných dat. Z tabulky lze vypozorovat, že nebyly pokryty případy užití UC9 , UC10 , UC14 , UC15 , UC34 , UC36 , UC37 a UC38 . Je to z toho důvodu, že funkcionality vyjádřené v těchto případech užití byly z implementace v rámci této práce nakonec vynechané z důvodu již tak velkého rozsahu. UC36 a UC37 (zablokování a změna role uživatele) však jsou administrátorovi v aplikaci částečně dostupné. Tyto dvě akce lze provést jednoduchou změnou hodnot v příslušných sloupcích v databázi. Funkcionalita v UC38 není nezbytná pro chod platformy, jelikož ke kontaktu lze jednoduše využít email. Chybějící API pro úpravu a mazání šablony (UC9 a UC10) bude mít vliv na pohodlí administrátora, jelikož nyní lze tyto zásahy provádět pouze ručně v databázi. Stejně tak úprava zdrojové položky (UC14). Mazat zdrojové položky (UC15) dotazem nad databází je v tuto chvíli jedinou problematickou operací, jelikož z důvodu zachování konzistence dat je nutné dát pozor na to, aby položka nebyla použita ani v inzerátu ani v odpovědi na inzerát. V nejbližších dnech po nasazení by to však absence této funkcionality neměla představovat větší problém. Jediným chybějícím případem užití, který by bylo vhodné mít v době spuštění aplikace hotový, je obnovení zapomenutého hesla (UC34). Není to jen z toho důvodu, že by uživatel mohl heslo zapomenout, ale především z toho důvodu, že se takto dá resetovat heslo v případě jeho krádeže. Jako dočasnou možnost ochrany uživatelova účtu při krádeži hesla lze využít možnost zablokovat účet změnou hodnoty sloupce v databázi. Jedná se však o první funkcionalitu, která bude implementována v další fázi projektu.

Jednotlivé testy provedeny lokálně na stroji autora. Pro testy byla sestavena a nasazena produkční verze programu pomocí vytvořeného docker-compose souboru, takže samotné prostředí bylo tak blízko produkčnímu prostředí, jak jen to je na lokálním stroji možné. Během testů bylo objeveno několik menších chyb, které byly autorem opraveny. Příkladem jedné z těchto chyb byla například nemožnost přímého použití URL k otevření stránky s vytvářením nových inzerátů. Další chybou která se vyskytla na více místech bylo chybějící volání překladové funkce (případně volání metody servisní třídy pro notifikace bez argumentu vyžadujícího překlad). To mělo za následek, že místo skutečného textu se na daném místě vyskytoval klíč překladu. Po opravě těchto chyb však již nebyly žádné další nalezeny. Po nasazení však bude vhodné provést testy znovu, aby se odladily i případné chyby spojené např. s použitím jiné domény, než je

localhost, nebo s použitím HTTPS. Zároveň by po nasazení bylo přínosem, pokud by testy provedl někdo jiný, než autor, nebo vedoucí práce. Z takovéhoto testování by bylo možné zjistit, jak je aplikace uživatelsky přívětivá pro nezasvěcené uživatele. Podle těchto výsledků by se dalo doladit uživatelské rozhraní a procesy uvnitř platformy.

8.2.1 Testovací scénáře

TS1 – Vytvoření inzerátu anonymním uživatelem Zjišťuje se, jestli funguje vytvoření nového inzerátu anonymním uživatelem. Během testů je vyzkoušeno zadávání vícejazyčného textu, aplikace šablony, úprava a zobrazení zahrnutých věcí, a v neposlední řadě také zasílané emailové zprávy se žádostí o potvrzení emailové adresy a s informací o vytvoření inzerátu. Otestováno je také potvrzení kontaktu anonymního uživatele.

TS2 – Zrušení inzerátu čekajícího na schválení Scénář ověřuje, že inzerent může zrušit inzerát pomocí odkazu, který mu přišel v emailu odeslaném při vytvoření inzerátu. Kontrolován je jak výsledný stav inzerátu a jeho přístupnost v očekávaném stavu, tak odeslání emailu informujícího o úspěšném zrušení inzerátu.

TS3/TS4 – Schválení/zamítnutí inzerátu čekajícího na schválení Cílem je ověřit fungování API, které umožňuje schválení a zamítnutí (zrušení) inzerátu koordinátorem. Zároveň se ověřuje posílání emailových zpráv o zrušení inzerátu.

TS5 – Vytvoření projektu Tento test ověřuje fungování rozhraní pro vytváření nových projektů. Testuje pouze fungování API. Po doplnění uživatelského rozhraní bude doplněn tak, aby testoval vytváření nových projektů přímo z webové stránky.

TS6 – Publikace projektu Ověření, že funguje rozhraní pro publikaci vytvořených projektů, čekajících na schválení. Subjektem testů je pouze API (podobně jako u TS5).

TS7 – Archivace projektu Test který zkoumá fungování rozhraní pro archivaci projektů. Stejně jako u TS5 a TS6 se test zaměřuje pouze na API.

TS8 – Prohlížení a filtrování přehledu projektů Scénář jehož cílem je ověřit, že funguje prohlížení seznamu projektů, včetně filtrování a stránkování.

TS9 – Vytvoření zdroje pro položky v inzerátech a odpovědích Ověření toho, že lze voláním API vytvořit nový zdroj. Očekávaným výsledkem je to, že je zdroj uložený v databázi a je tedy připravený jak pro samostatné přidání do inzerátu či odpovědi, tak pro doporučení v šabloně.

TS10 – Vytvoření šablony pro projekt Testovací scénář který ověřuje, že lze pomocí volání API vytvořit novou šablonu. Na konci je šablona vytvořená a připravená na použití při tvorbě inzerátu.

TS11 – Prohlížení inzerátů Test ve kterém je zkoumána funkčnost prohlížení seznamu dostupných inzerátů, včetně jeho filtrování a stránkování.

TS12 – Sdílení inzerátu Testuje se sdílení inzerátu jak z přehledu, tak z detailu. Ověřuje se, že je možné inzerát sdílet na Facebook a emailem. Na lokálním prostředí lze otestovat pouze sdílení emailem, jelikož sdílení přes Facebook vyžaduje platnou externí URL, což localhost bohužel není.

TS13 – Zobrazení detailu inzerátu Účelem je ověření možnosti prokliknout na zveřejněný inzerát, zkontrolovat správnost zobrazovaných údajů a otestovat funkčnost zobrazení detailu položek. Zároveň je vyzkoušena funkčnost překladů položek.

TS14 – Vytvoření odpovědi na inzerát Cílem je otestovat, že funguje vytváření nové odpovědi na inzerát neregistrovaným uživatelem. Zároveň je ověřeno, že funguje email s výzvou k potvrzení zadaného emailu. Kromě toho jsou kontrolovány další odesílané emailové notifikace. Funkčnost linku na náhled odpovědi a vyřešení odpovědi v přijaté zprávě v rámci tohoto testu ověřována není.

TS15 – Prohlížení detailu projektu V rámci toho scénáře je kontrolována funkčnost stránek s detailem projektu a důležitými informacemi k projektu. Je otestována také funkčnost překladů.

TS16 – Zobrazení odpovědi na inzerát Kontrola, že se u odpovědi, která ještě inzerentem nebyla přijata ani zamítnuta, zobrazují správné hodnoty. Hodnoty jsou kontrolovány proti tomu, co je uloženo v databázi. Je kontrolována jak varianta zobrazená respondentovi, tak varianta zobrazená inzerentovi (měli by být stejné až na tlačítka pro přijmutí/zamítnutí, které jsou zobrazené pouze inzerentovi).

TS17/TS18 – Přijmutí/odmítnutí odpovědi Cílem testů je ověřit, že lze přijmout/zamítnout odpověď, která ještě nebyla inzerentem nijak vyřešena. Ověřována je funkčnost odkazu v původní emailové zprávě notifikující o tom, že byla obdržena odpověď. Dále je vyzkoušeno, že funguje samostatná akce přijmutí/zamítnutí inzerátu na webové stránce. Na konci je zkontrolován výsledný stav inzerátu, zobrazení poznámky inzerenta, invalidace původních odkazů a správné vygenerování a odeslání nových odkazů (a tedy i odeslání očekávaných emailových zpráv).

TS19 – Registrace a přihlášení uživatele Test ověřuje funkčnost registrace nového uživatele. Ověřuje se jak funkčnost samotného formuláře pro registraci, tak odesílání emailových zpráv a správnost odkazů v nich. V rámci ověření fungování procesu aktivace je také odzkoušen přihlašovací formulář.

TS20 – Změna emailové adresy V tomto scénáři je testován proces změny emailové adresy aktuálně přihlášeného uživatele. Je otestován jak samotný formulář a rozhraní na backendu, tak i odesílání emailových zpráv s konfirmačními kódy a potvrzením o úspěšné změně.

TS21 – Změna telefonního čísla Testuje se, že lze provést změnu telefonního čísla z rozhraní pro editaci údajů uživatele. Kontrolována je nejenom funkčnost formulář a rozhraní na backendu, ale i odesílání emailových zpráv s konfirmačním kódem a potvrzením o úspěšné změně.

TS22 – Změna jazyků ovládaných uživatelem a zveřejněných detailů Tento scénář je zaměřen na změnu ovládaných jazyků uživatele a změnu nastavení úrovně zveřejněných jazyků. Je otestována jak stránka, tak emaily potvrzující úspěšnou změnu, odesílané na emailovou adresu uživatele.

TS23 – Označení inzerátu, jako vyřešený Ověření toho, že lze inzerát označit za vyřešený. Zároveň je zjišťováno, jestli jsou nevyřešené odpovědi na inzerát označeny, jako automaticky odmítnuté. Kontrolovány jsou také emailové zprávy, které jak inzerentovi, tak respondentům, jejichž odpovědi byly automaticky odmítnuté, chodí.

Tabulka 8.1 Tabulka pokrytí případů užití testovacími scénáři

	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8	TS9	TS10	TS11	TS12
UC1								✓				
UC2								✓				
UC3							✓					
UC4												
UC5					✓							
UC6						✓						
UC7							✓					✓
UC8										✓		
UC9												
UC10												
UC11	✓											
UC12	✓											
UC13									✓			
UC14												
UC15												
UC16	✓											
UC17												✓
UC18	✓											
UC19		✓										
UC20			✓									
UC21				✓								
UC22											✓	
UC23											✓	
UC24												✓
UC25	✓											
UC26												
UC27												
UC28												
UC29												
UC30												
UC31												
UC32												
UC33												
UC34												
UC35												
UC36												
UC37												
UC38												

Tabulka 8.2 Tabulka pokrytí případů užití testovacími scénáři

	TS13	TS14	TS15	TS16	TS17	TS18	TS19	TS20	TS21	TS22	TS23
UC1			✓								
UC2											
UC3			✓								
UC4			✓								
UC5											
UC6											
UC7											
UC8											
UC9											
UC10											
UC11											
UC12											
UC13											
UC14											
UC15											
UC16											
UC17											
UC18											
UC19											✓
UC20											
UC21											
UC22											
UC23											
UC24	✓	✓									
UC25	✓										
UC26		✓									
UC27				✓	✓	✓					
UC28		✓		✓							
UC29					✓						
UC30						✓					
UC31	✓		✓	✓							
UC32							✓				
UC33							✓				
UC34											
UC35								✓	✓	✓	
UC36											
UC37											
UC38											

Kapitola 9

Možnosti vylepšení do budoucna

Jak už to tak většinou bývá, i zde je co zlepšovat nebo co rozšiřovat. Seznam chyb a možných vylepšení do budoucna je veden v *issue tracking* systému na GitHubu. V této sekci budou zmíněné některé z nich.

Během rozboru možností vylepšení do budoucna bylo identifikováno několik věcí, které bude možné (ne-li přímo nutné) v budoucnu implementovat. K několika z nich již existuje i koncept, jak by mohly vypadat, jelikož byly na počátku zvažovány, ale nakonec byly vynechány z důvodu příliš velkého rozsahu.

Pro platformu se do budoucna počítá s **mobilní aplikací**. Ta bude nejspíše vyvíjena v rámci jiné bakalářské práce. Mobilní aplikace by měla přinést uživatelský zážitek vylepšený díky využití nativního prostředí mobilních zařízení. [116]

Od začátku se počítá s tím, že na stránce v další fázi přibude **mapa**. Mapa bude využívána na více místech. V rámci rešerše byly analyzovány některé mapové platformy. Zároveň byl proveden průzkum možných řešení pro implementaci mapy. Také byl udělán předběžný návrh toho, jak by varianta s mapou mohla vypadat (viz. obrázek 9.1 a obrázek 9.2). Zbytek návrhu není 1:1 s tím, jak vypadá finální verze webu, takže je nutné ho brát s rezervou, ale základní myšlenku to předá.

Projekty by mělo být v budoucnu možné **vytvářet přímo z webu**. Bude zde nutné promyslet, kdo bude moci nové projekty přidávat, upravovat a archivovat.

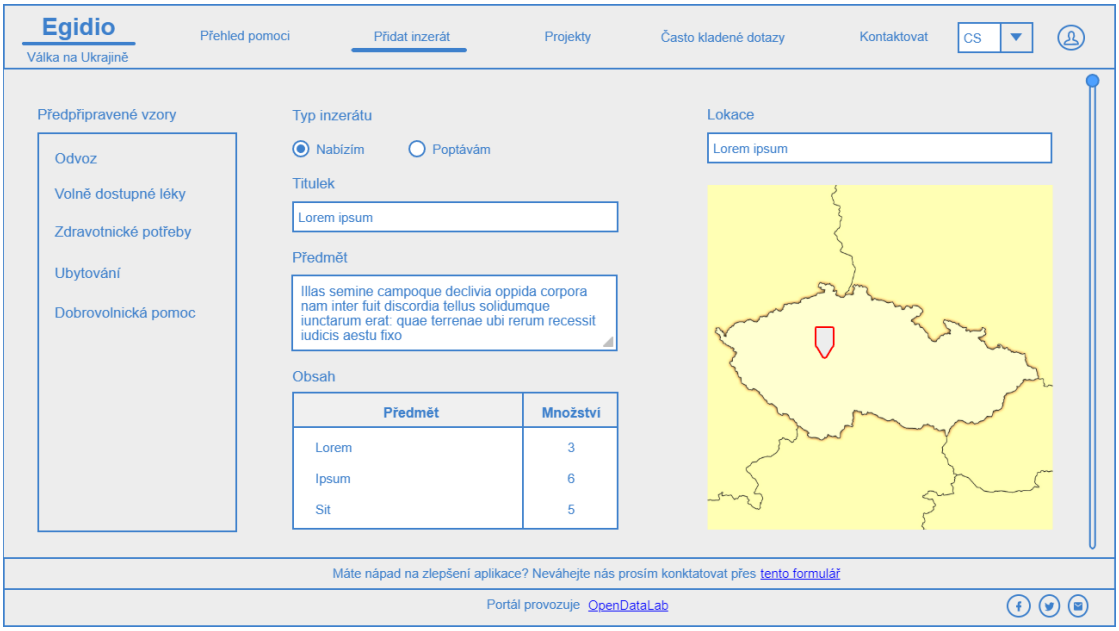
Ve chvíli, kdy budou projekty vytvářené z UI, bude skoro určitě nutné přidat i nějaké **rozhraní pro správu šablon věcí zahrnutých v inzerátech**.

Na stránce s detailem projektu by měly být dostupné **statistiky daného projektu**. Koncept, jak by mohly vypadat, byl již vytvořen v rámci návrhu (viz. obrázek 9.3). Jejich samotná implementace však bude součástí jedné z dalších fází.

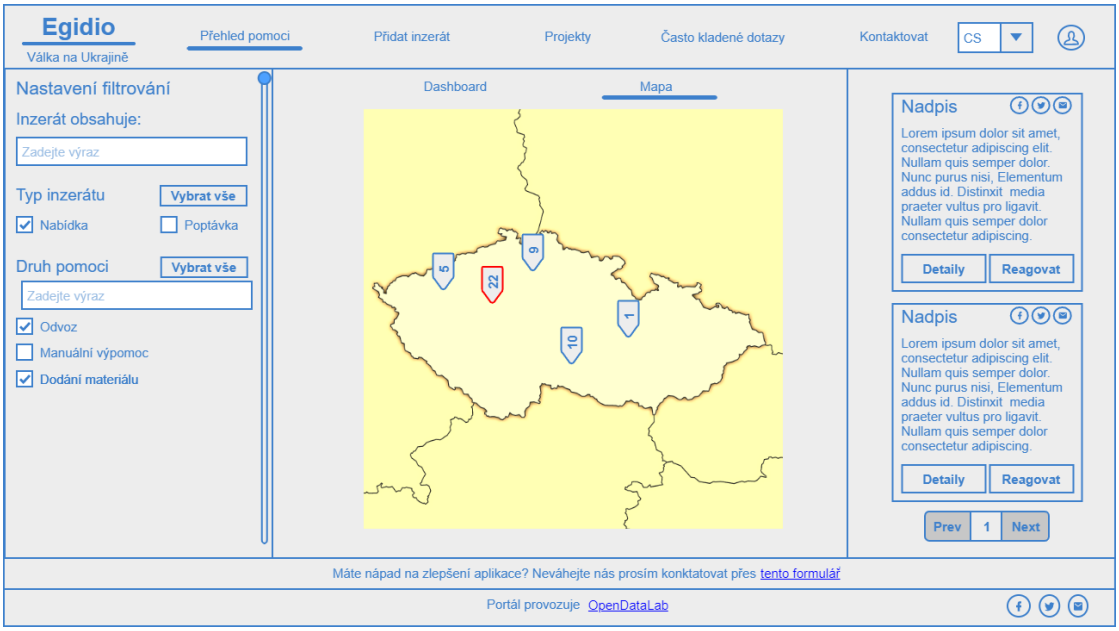
Do budoucna by nebylo od věci umožnit **prohlížení všech zveřejněných inzerátů bez ohledu na to, k jakému projektu jsou vázané**. Případem užití pro tuto funkcionalitu je situace, kdy chce uživatel nabídnout nějakou pomoc v dané lokaci, ale nezáleží mu na tom, k jaké katastrofě je inzerát vázaný.

Mělo by být možné vytvořit **inzerát pro více projektů najednou**. Inzerátům by se takto poteciálně mohl zvýšit jejich dosah. Struktura v databázi to již umožňuje.

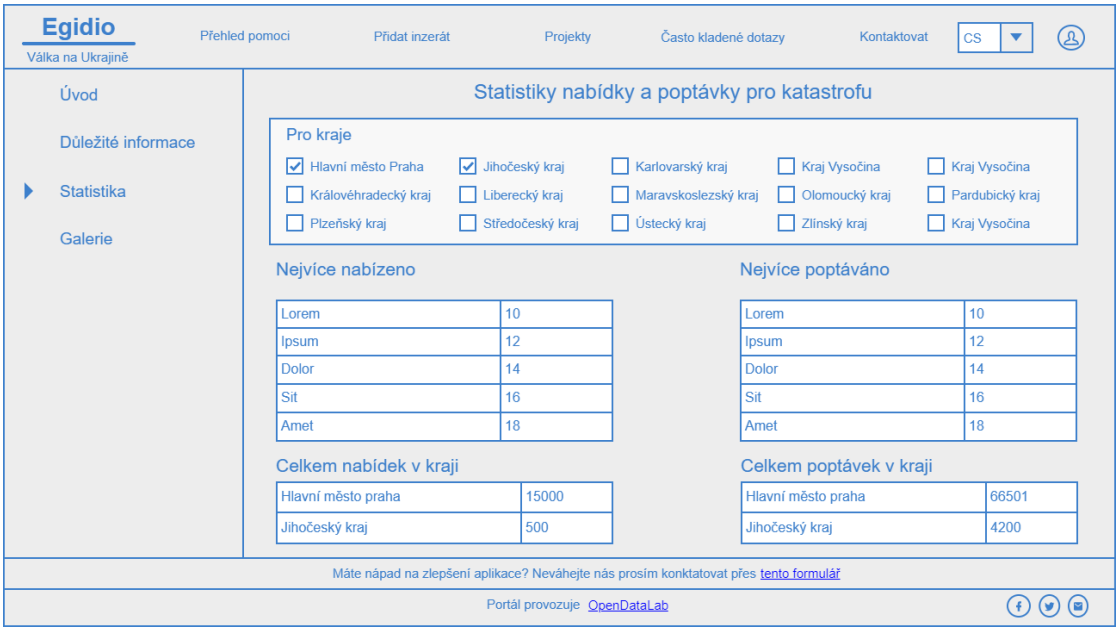
V aktuálním provedení nelze inzerát po vytvoření upravit. Pokud je na inzerátu něco špatně, musí ho smazat a vytvořit nový. Toto chování není optimální, a to jak z pohledu uživatele, tak z pohledu databáze. V databázi takto vznikají inzeráty, které jsou de-facto duplikací předchozích inzerátů. Čím větší bude celkový počet inzerátů, tím více poroste i počet chybných inzerátů. Pro tyto inzeráty budou navíc uloženy všechny překlady jak jejich popisů a titulků, tak konkrétních věcí zahrnutých v inzerátu. Vhodnějším řešením tedy bude přidat **možnost upravovat inzerát jak administrátorem/koordinátorem, tak přímo uživatelem**. U uživatele pak asi ještě bude nutné opět vynutit ověření inzerátu koordinátorem.



Obrázek 9.1 Koncept použití mapy u vytváření inzerátu



Obrázek 9.2 Koncept použití mapy u přehledu inzerátů



Obrázek 9.3 Statistiky projektu

V této chvíli jsou automaticky zveřejněny pouze inzeráty, které odeslal nějaký koordinátor či administrátor. Pro snížení administrativní zátěže by ale bylo vhodné implementovat i další kritéria, která umožní automatické přijetí inzerátů běžnými uživateli. Těmi může být například to, kolik inzerátů uživatele již bylo zveřejněno, nebo jaké má uživatel hodnocení (případně kombinace obojího).

Jednou z věcí, pro kterou v této fázi implementace nezbyl prostor, a která této aplikaci značně chybí, je verifikace zadaných lokací. V případě, kdy bude implementována mapa, bude toto nezbytnou prerekvizitou. I bez ní by ale bylo dobré začít místa co nejdříve verifikovat. Úplně ideálním řešením by pak bylo projet i staré inzeráty, pokusit se lokaci opravit a ty které opravit nepůjdou, pak nastavit na nějakou výchozí hodnotu která bude značit, že lokace není dostupná.

Na stránce nyní chybí jakýkoliv přehled uživatelových inzerátů a jeho odpovědí na jiné inzeráty. Stejně tak chybí přehled o tom, jaké odpovědi přišly na daný inzerát. Implementace těchto věcí by nepochybně měla kladný vliv na uživatelský zážitek z užívání naší aplikace.

Na platformě by mělo být možné hodnotit uživatele. Toto hodnocení by si pak měli být lidé schopni přečíst před tím, než odpoví na inzerát, jehož je uživatel autorem, nebo před přijmutím odpovědi odeslané daným uživatelem. Implementace zobrazení hodnocení uživatele již byla na frontendu z části dokončena u detailu inzerátu. Frontendový kód smazán nebyl, hodnocení se však nezobrazuje, protože do komponentu s náhledem uživatele není předáváno hodnocení uživatele. Logika pro hodnocení na frontendu nakonec nebyla implementována.

Schválení a zamítnutí inzerátů dnes probíhá výhradně přes API. Je ale více než žádoucí, aby bylo přidáno uživatelské rozhraní přímo na web. Správa inzerátů se tím značně sníží. Stejně tak by bylo vhodné přidat na web notifikace pro osoby, které mohou inzeráty schvalovat, aby byly informovány, že je dostupný nějaký inzerát který mohou schválit.

Aplikace nyní nijak neověřuje, že je zadané telefonní číslo platné, nebo že telefonní číslo opravdu patří danému uživateli. To představuje potencionální problém, jelikož nic nezabrání uživateli zadat telefonní číslo, které buďto neexistuje, nebo v horším případě dokonce patří někomu jinému. Řešením by mohlo být ověření čísla pomocí SMS. K tomu ale bude nejdříve potřeba vyřešit, jakým způsobem SMS posílat.

V tuto chvíli je možné se přihlásit pouze zadáním přihlašovacích údajů platných pouze pro

tuto stránku. Přidání podpory přihlášení přes externí služby jako je Google a Facebook by mohlo uživatelům zjednodušit užívání platformy, a mohlo by to přesvědčit i další uživatele, aby se na platformě registrovali, protože už si nebudou muset pamatovat další pár jméno-heslo.

Blokování uživatelů a úprava jejich práv nyní probíhá manuálním voláním API administrátorem. Stejně tak pokud chce administrátor uživateli změnit nějaké nastavení, musí volat API. Vhodnějším řešením by ale bylo přidat pro administrátora na platformu ovládací prvek, který bude umožňovat upravovat uživatele.

Pro zvýšení efektivity využití dostupných zdrojů by bylo dobré implementovat funkcionalitu, díky které by byl uživatel informován o tom, že existuje podobný inzerát (nebo i více inzerátů), ve kterém jsou buďto nabízené některé z poptávaných věcí, nebo poptávané některé z nabízených věcí, a nabídnout mu, jestli raději nechce nejdříve zkusit odpovědět na tyto inzeráty.

Tato funkcionalita by umožňovala zasílat uživatelům emaily (či jiné notifikace), informující o tom, že byl vypsán inzerát, který je podobný něčemu, co již někdy v minulosti inzerovali (případně něčemu na co již odpovídali).

U některých částí aplikace by bylo vhodné, kdyby k nim byla zakomponována galerie. Již bylo odzkoušeno přidání galerie k detailu projektu. Použita byla knihovna ngx-gallery, mezi jejíž přednosti patří to, že je stavěna přímo pro Angular, je responzivní a má více možností druhů zobrazení (carousel, mřížka…). Navíc je pod MIT licencí, takže je možné ji zkombinovat s kódem pod Apache 2.0 licencí, která byla použita pro tuto práci. Nebyl však vytvořen kód na straně backendu a zároveň nebyly provedeny důkladnější testy frontendového kódu. HTML kód byl z detailu projektu nakonec smazán, lze ho však stále dohledat v historii na GitHubu.

Galerie k inzerátům by našla své využití především jako doplněk popisu. Bylo by s její pomocí možné dodatečně ilustrovat to, kam pomoc míří, k čemu by byla použita atd.

Galerie zdrojů by mohla pomoci tím, že se s její pomocí budou moci lidé ujistit, že si pod daným termínem nepředstavují něco jiného, než je jím opravdu myšleno.

Galerie položek v inzerátu/odpovědi by např. mohla ilustrovat stav nabízených věcí.

Uživatelům by mohlo být nápomocné větší napojení na různé sociální sítě. Mohlo by být například možné zobrazovat příspěvky mající nějaký tag. Také by nemuselo být od věci zobrazení příspěvků v krizovém centru pro danou katastrofu na FB (viz. 2.14 Krizové centrum na Facebooku).

Jednou z věcí, která nebyla přidána, protože na ni již nezbyl čas, je vyhledávání podle věcí obsažených inzerátů. Tato funkcionalita by však uživatelům umožnila mnohem podrobněji odfiltrovat inzeráty, které pro ně nejsou relevantní. Pro přidání by mělo být možné použít vyhledávací komponentu podobně, jako je tomu u hledání zdrojů pro položky inzerátu či odpovědi. Bude zde ale ještě nejspíše nutné přidat možnost zobrazit si více informací, případně se spokojit s tím, že uživatel bude hledat jen podle jména, což je asi taky akceptovatelné.

V tuto chvíli ze serveru jsou posílány chybové hlášky v angličtině. Jelikož je ale aplikace lokalizovaná, bylo nutné nějak vyřešit překlad nejčastějších chyb. To mělo za následek nešikovný kód na frontendu, který vyžaduje handling těchto chyb v komponentách. Mnohem správnější by bylo posílat na backend aktuálně vybraný jazyk, a pro něj lokalizovat chybové hlášky. Ty by poté bylo možné odchytávat pomocí Angulařího HttpInterceptoru. Tím by se rapidně snížilo množství boiler plate kódu, který na straně Angularu je, protože o handling by se nestarali komponenty, ale staralo by se o ně jedno centralizované místo.

Validace zpráv ze serveru V aktuálním stavu nejsou na frontendu validována data, která přijdou ze serveru. Frontend spoléhá na to, že mu server zasílá data ve validním formátu. To s sebou do budoucna nese jistá rizika při úpravách API. Dobrým řešením by mohlo být zakomponovat nějakou z validačních knihoven, například Zod[1].

[1] `https://zod.dev/`

Kapitola 10
Závěr

Cílem práce bylo navrhnout a implementovat platformu, ve které je možné uživatelsky přívětivě nabídnout pomoc, napsat si o ní a také ji organizovat. Zápisy mělo být možné sdílet na sociálních sítích a nabízející měli moc řídit úroveň zveřejněných detailů. Mělo být možné vytvořit katastrofu a povolit určité typy pomoci. Pro běžné katastrofy měla být dostupná šablona toho, co je běžně potřeba. Postup řešení měl sestávat z rešerše stávajících řešení, návrhu vhodné škálovatelné architektury řešení, diskuze a výběru vhodné technologie, implementace a otestování vzniklého webového portálu.

Zadání bylo splněno. Prvně byla provedena rešerše stávajících řešení, ze které vzešlo, že i když existují řešení, na kterých je možné nabízet či poptávat pomoc, žádné z nich není centralizované pro více katastrof.

Následně byla navržena a implementována platforma, ve které je možné nabídnout pomoc a napsat si o ní. Těmto nabídkám a poptávkám se v práci souhrnně říká inzeráty a může je zveřejňovat jak návštěvník, který není registrovaný, tak registrovaný uživatel. V obou případech lze řídit úroveň zveřejněných osobních údajů pomocí jednoduchého formuláře. O organizaci pomoci se starají tzv. koordinátoři, kteří v tuto chvíli mohou kontrolovat, jaké nabídky a poptávky jsou zveřejňované. Inzeráty navíc mohou být i dlouhodobějšího rázu, takže není nutné je po první přijaté odpovědi ukončit.

V inzerátu je možné nabídnout nebo poptat různé typy pomoci, jako například psychologickou pomoc, materiální pomoc atd. které jimi mohou být nabízené či poptávané.

Vytvořené API nabízí koordinátorům a administrátorům možnost vytvořit, zveřejnit a archivovat projekty reprezentující jednotlivé katastrofy.

Jak inzeráty, tak projekty, lze přímo z webové platformy sdílet na Facebooku a emailem.

Byl vytvořen systém šablon, které je možné přiřadit nejen přímo ke konkrétnímu projektu, ale i obecně k typům katastrofy, pomoci a inzerátu (nabídce/poptávce). Šablony lze využít při tvorbě nového inzerátu k předvyplnění seznamu poptávaných a nabízených věcí.

Pro backendovou část byly vytvořeny jednotkové testy ověřující funkčnost částí kódu. Pro uživatelské testování byly vytvořeny scénáře, které byly provedeny na lokálním testovacím prostředí, které je téměř shodné s produkčním prostředím.

Laboratoř otevřených dat (OpenDataLab) plánuje výstup této práce dále rozšiřovat. Již nyní je vypsána závěrečná práce, jejímž tématem je návrh a implementace mobilní aplikace. Autor sám by rád v práci pokračoval v implementaci navrhovaných změn z kapitoly 9.

Příloha A

README

Edigio

Platforma pro řešení katastrof. Jejím cílem je zprostředkovat nabídku a poptávku pomoci při různých katastrofách. Jak registrovaní, tak neregistrovaní uživatelé mají možnost vytvořit nabídku či poptávku, a také mají možnost na ni odpovědět. Pro registrované uživatele je proces zjednodušen tím, že nemusejí stále dokola vyplňovat kontaktní informace a souhlas se zpracováním osobním osobních údajů a s podmínkami užití služby. Navíc nemusí při každém vytvoření nabídky, poptávky či odpovědi na ně znovu potvrzovat svoji emailovou adresu. Inzeráty a projekty (katastrofy) je možné prohlížet, filtrovat a sdílet. Při vytvoření inzerátu lze využít jednu ze šablon, které je možné vložit do databáze, a které specifikují běžný obsah různých typů inzerátů pro danou katastrofu. Inzerent může specifikovat, jaká úroveň detailů má být zveřejněná. Aplikace je plně lokalizována (jak její statické části, tak dynamické).

Podmínky služby a zásady zpracování OU

Pro zásady zpracování OU a podmínky užití služby byly vytvořeny dvě komponenty - PrivacyPolicyComponent a TermsOfServicesComponent.
Obsah těchto dokumentů nebyl vytvořen. Je na provozovateli, aby obsah těchto dokumentů vyplnil. Zásady zpracování OU se vyplňují v dokumentu "source/frontend/src/app/components/privacy-policy/privacy-policy.component.html" a podmínky užití služby v dokumentu "source/frontend/src/app/components/terms-of-services/terms-of-services.component.html". Oba dokumenty mají následující strukturu, z obsahu divů je již zřejmé co a kam doplnit.

```
<div class="col-12">
  <div *ngIf="displayEnglishVariant">
    Sem doplňte text v Angličtině
  </div>

  <div *ngIf="displayCzechVariant">
    Sem doplňte text v Češtině
  </div>
</div>
```

Instalace

Pro jednoduchou instalaci byl vytvořen docker-compose soubor. Pro jeho použití stačí mít na svém počítači (či cílovém serveru) zprovozněný Docker a Docker Compose. Je potřeba vyplnit textový soubor '.env', který musí být ve stejné složce, jako docker compose soubor. Jsou očekávány následující proměnné

#Uzivatelske jmeno k emailové adrese
EGIDIO_EMAIL_USERNAME=UZIVATELSKEJMENO

#Heslo k emailové adrese
EGIDIO_EMAIL_PASSWORD=HESLO
#Host emailové schránky
EGIDIO_EMAIL_HOST=smtp.gmail.com
#Port pro email
EGIDIO_EMAIL_PORT=587
#Protokol používaný pro email aplikace
EGIDIO_EMAIL_PROTOCOL=smtp

#Cesta k databázi, ve které jsou data aplikace. Pokud používáte nezměněný docker-compose, můžete ponechat tak jak je.
EGIDIO_DATASOURCE_URL=jdbc:postgresql://egidio-database:5432/postgres
#Heslo k databázi, ve které jsou data aplikace.
EGIDIO_DATASOURCE_PASSWORD=postgres
#Uživatelské jméno k databázi, ve které jsou data aplikace.
EGIDIO_DATASOURCE_USERNAME=postgres

#Port na kterém je z vnější dostupná databáze
POSTGRES_EXTERNAL_PORT=5433

#Externí URL frontendu využívaná backendem pro tvorbu linků
EGIDIO_FRONTEND_URL="localhost"

Během verzování si dávejte pozor, abyste tento soubor neodeslali do repozitáře. Mohlo by dojít k úniku Vašich informací!
Je nutné, aby byl volný port 80, se kterým aplikace počítá!

Pokud používáte jiné než výchozí nastavení docker compose, je ještě nutné v souboru 'source/frontend/src/app/api-config/common-api-config.ts' vyplnit adresu URL, která je v tuto chvíli relativní vůči adrese, na které klient stránku otevřel. To se může hodit například i při lokálním spuštění frontendu mimo docker. Ještě je potřeba upravit nastavení CORS. To se provadí v backendovém kódu ve třídě cz.opendatalab.egidio.backend.configs.SecurityConfiguration (resp. její metodě corsConfigurer). Je potřeba změnit seznam ve volané metodě registry#allowedOrigins podle toho, z jakých adres bude frontend dostupný.

Dále je ještě potřeba, aby uživatel, který bude spouštět docker compose, měl práva pro spuštění gradlew. Nastavení práv může na Linuxu (pokud jsme v kořenové složce projektu) vypadat například takto:

```
chmod +x ./gradlew
```

V tuto chvíli by již vše mělo být připraveno ke spuštění. Stačí zavolat příkaz docker compose up ve složce 'source'. Po sestavení jednotlivých obrazů by již aplikace měla být spuštěná s parametry, které byly zadány.

Bibliografie

1. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *Pomáhej Ukrajině* [online]. Pomáhej Ukrajině. [cit. 2022-11-05]. Dostupné z: `https://www.pomahejukrajine.cz/`.
2. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *Pomáhej Ukrajině* [online]. Pomáhej Ukrajině. [cit. 2022-11-05]. Dostupné z: `https://www.pomahejukrajine.cz/nabidka`.
3. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *FAQ* [online]. Pomáhej Ukrajině. [cit. 2022-11-06]. Dostupné z: `https://www.pomahejukrajine.cz/faq`.
4. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *Neveřejné nabídky pomoci* [online]. Pomáhej Ukrajině. [cit. 2022-11-05]. Dostupné z: `https://www.pomahejukrajine.cz/neverejne-nabidky/3a0731d3-2102-4a7b-a5c4-096e0876f10b`.
5. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *Veřejné nabídky pomoci* [online]. Pomáhej Ukrajině. [cit. 2022-11-05]. Dostupné z: `https://www.pomahejukrajine.cz/nabidky/`.
6. MAPOTIC. *Community Maps* [online]. Mapotic, ©2023. [cit. 2023-05-10]. Dostupné z: `https://www.mapotic.com/solutions/community-maps/`.
7. MAPOTIC; ČESKO.DIGITAL. *Umapa - Ukrainian Community Map* [online]. Mapotic, ©2023. [cit. 2023-05-08]. Dostupné z: `https://www.umapa.eu`.
8. *Umapa: Mapa pro Ukrajince v ČR* [soft.]. Pod Hájkem 2204/1, 180 00 Praha: Mapotic, Česko.Digital, 2023. Ver. 1.0.26 [cit. 2023-05-08]. Dostupné z: `https://play.google.com/store/apps/details?id=com.mapotic.ukrainecommunity`. Požadavky na systém: Android 5.0 nebo novější.
9. ZAMAZAL, Pavel. *Mapa Dobrovolníků* [online]. Mapotic, ©2023. [cit. 2023-05-09]. Dostupné z: `https://www.mapotic.com/mapa-dobrovolniku`.
10. *FandiMat* [online]. FandiMat, ©2022. [cit. 2022-11-05]. Dostupné z: `https://www.mapotic.com/fandimat`.
11. *Klidně Použité Hračky Nebo Knížky Jen Ať Mají Kluci Něco Pod Stromečkem* [online]. FandiMat, ©2022. [cit. 2022-11-05]. Dostupné z: `https://mapa.fandimat.cz/1460754-klidne-pouzite-hracky-nebo-knizky-jen-at-maji-kluci-neco-pod-stromeckem`.
12. KODIRNA.CZ. *Portál pomoci - pomoc poškozeným* [online]. Portál pomoci, ©2023. [cit. 2023-05-08]. Dostupné z: `https://portalpomoci.cz/`.

13. MINISTERSTVO VNITRA. *Informace o pomoci Ukrajincům | nasiukrajinci.cz* [online]. nasiukrajinci.cz, ©2023. [cit. 2023-05-08]. Dostupné z: https://www.nasiukrajinci.cz/cs.
14. MINISTERSTVO PRÁCE A SOCIÁLNÍCH VĚCÍ. *Smart Migration – Aplikace na Google Play* [online]. Google Play, 2023-01-12. [cit. 2022-11-05]. Dostupné z: https://play.google.com/store/apps/details?id=cz.mpsv.smartmigration&hl=cs&gl=US.
15. *Smart Migration* [online]. Ministerstvo práce a sociálních věcí, 2023. Ver. 1.0.4 [cit. 2022-11-05]. Dostupné z: https://play.google.com/store/apps/details?id=cz.mpsv.smartmigration&hl=cs&gl=US. Požadavky na systém: Android 6.0 nebo novější.
16. CZECHINVEST; NULA S.R.O. *Czechinvest - Jobs 4 UA* [online]. Jobs4UA. [cit. 2023-04-23]. Dostupné z: https://www.jobs4ua.cz/.
17. HESTIA - CENTRUM PRO DOBROVOLNICTVÍ, Z.Ú. *Příležitosti* [online]. Dobrovolnik.cz. [cit. 2023-05-08]. Dostupné z: https://www.dobrovolnik.cz/prilezitosti.
18. BULÁKOVÁ, Jana. *Pomáhej dětem ze sociálně znevýhodněného prostředí* [online]. Dobrovolnik.cz, 2020-03-30. [cit. 2023-05-10]. Dostupné z: https://www.dobrovolnik.cz/prilezitost/pomahej-detem-ze-socialne-znevyhodneneho-prostredi.
19. HESTIA - CENTRUM PRO DOBROVOLNICTVÍ, Z.Ú. *Vytvořte svůj uživatelský účet, zvolte možnosti* [online]. Dobrovolnik.cz. [cit. 2023-05-10]. Dostupné z: https://www.dobrovolnik.cz/user.
20. *Přispěli Jste Na Ochranné Prostředky pro Řidiče Na Ukrajinu* [online]. Donio, ©2022. [cit. 2023-04-23]. Dostupné z: https://www.donio.cz/ochranne-prostredky-pro-ridice.
21. *Dofinancovali Jste Dron s Termokamerou a Zachránili Jste Tím Ještě Více Životů* [online]. Donio, 2022-09-07. [cit. 2023-04-23]. Dostupné z: https://www.donio.cz/dron-pro-mediky.
22. UKRAINENOW. *Accommodation Abroad* [online]. UkraineNow, ©2022. [cit. 2023-05-10]. Dostupné z: https://www.ukrainenow.org/looking-for-help-abroad.
23. UKRAINENOW. *I'm Abroad and I Can Host* [online]. UkraineNow, ©2022. [cit. 2023-05-10]. Dostupné z: https://www.ukrainenow.org/i-can-host.
24. UKRAINENOW. *I Can Help with Resources and Supply* [online]. UkraineNow, ©2022. [cit. 2023-05-10]. Dostupné z: https://www.ukrainenow.org/i-can-help-with-resources-and-supply.
25. UKRAINENOW; PAYPAL. *Přispějte příjemci UkraineNow* [online]. PayPal, ©1999–2023. [cit. 2023-05-10]. Dostupné z: https://paypal.com/donate/?hosted_button_id=KPSDU25RLLAS4.
26. UKRAINENOW. *How to Help List* [online]. UkraineNow, ©2022. [cit. 2023-05-10]. Dostupné z: https://www.ukrainenow.org/how-to-help-list.
27. UNITED NATIONS VOLUNTEERS. *UVP - Unified Volunteer Platform* [online]. ©2021. [cit. 2023-05-08]. Dostupné z: https://app.unv.org/.
28. UNITED NATIONS VOLUNTEERS. *Signin* [online]. UNV, ©2020. [cit. 2023-05-10]. Dostupné z: https://unvprodb2c.b2clogin.com/unvprodb2c.onmicrosoft.com/b2c_1a_uvp/oauth2/v2.0/authorize?response_type=id_token&scope=openid%20ca9dfccc-7bfe-4b88-89a2-20d4646a0613%20openid%20profile&client_id=ca9dfccc-7bfe-4b88-89a2-20d4646a0613&redirect_uri=https%3A%2F%2Fapp.unv.org&state=eyJpZCI6IjMwNzRmMThkLTMxZTctNDI5NS04NDg4LWIzODJjZjk5NDE4NCIsInRzIjoxNjgzNzAyNDk4LCJtZXRob2QiOiJyZWRpcmVjdEludGVyYWN0aW9uIn0%3D%7Chttps%3A%2F%2Fapp.unv.org%2F%3FfromLoginBtn%3D1&nonce=7459b948-123e-4aee-bc15-41888d91a102&client_info=1&x-client-SKU=MSAL.JS&x-client-Ver=1.4.16&

client-request-id=d4189c6e-71ab-49d3-b4a2-ca535efe8389&response_mode=fragment.

29. *Why Do I Have to Pay to Volunteer Abroad?* [online]. Volunteer World, 2022-11-06. [cit. 2022-11-06]. Dostupné z: https://intercom.help/volunteerworld/en/articles/376451-why-do-i-have-to-pay-to-volunteer-abroad.
30. VOLUNTEER WORLD. *Top 30 Volunteer Programs on Volunteer World* [online]. Volunteer World. [cit. 2023-05-10]. Dostupné z: https://www.volunteerworld.com/en/filter.
31. PEDRO FIT; KOMUNITA. *Pomoc UKRAJINĚ* [online]. Facebook, 2022-11-05. [cit. 2022-11-06]. Dostupné z: https://www.facebook.com/groups/468032188137410/.
32. META PLATFORMS, INC. *Krizové centrum* [online]. Facebook, ©2023. [cit. 2022-11-06]. Dostupné z: https://www.facebook.com/crisisresponse/.
33. META PLATFORMS, INC. *Krizové centrum | Centrum nápovědy* [online]. Facebook, ©2023. [cit. 2022-11-06]. Dostupné z: https://www.facebook.com/help/141874516227713.
34. PAULUS, František et al. *Analýza hrozeb pro Českou republiku* [online]. Praha, 2015 [cit. 2023-04-23]. Závěrečná zpráva. Dostupné z: https://www.hzscr.cz/soubor/analyza-hrozeb-zprava-pdf.aspx.
35. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY V ČR, Z.S. *Veřejné nabídky pomoci* [online]. Pomáhej Ukrajině. [cit. 2022-05-11]. Dostupné z: https://www.pomahejukrajine.cz/nabidky/.
36. KONSORCIUM NEVLÁDNÍCH ORGANIZACÍ PRACUJÍCÍCH S MIGRANTY. *Pomáhej Ukrajině v grafech* [online]. Konsorcium nevládních organizací pracujících s migranty, 2023-05-01. [cit. 2023-05-11]. Dostupné z: https://migracnikonsorcium.cz/cs/data-statistiky-a-analyzy/pomahej-ukrajine-v-grafech/.
37. SIMPLILEARN SOLUTIONS. *Why and How Is Java Platform Independent?* [online]. Simplilearn.com, 2022-09-20. [cit. 2023-05-01]. Dostupné z: https://www.simplilearn.com/why-and-how-is-java-platform-independent-article.
38. *Most Used Languages among Software Developers Globally 2022* [online]. Statista, 2022-08-09. [cit. 2022-11-19]. Dostupné z: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.
39. TIOBE SOFTWARE BV. *TIOBE Index for April 2023* [online]. TIOBE, ©2023. [cit. 2023-05-05]. Dostupné z: https://www.tiobe.com/tiobe-index/.
40. JETBRAINS S.R.O. *FAQ* [online]. Kotlin Help, 2023-05-10. [cit. 2023-05-11]. Dostupné z: https://kotlinlang.org/docs/faq.html.
41. PŘISPĚVATEL TECHTARGETU. *What Is Boilerplate?* [online]. WhatIs.com, 2023-10. [cit. 2022-11-19]. Dostupné z: https://www.techtarget.com/whatis/definition/boilerplate.
42. AUTOŘI PROJEKTU LOMBOK. *Lombok Features* [online]. Project Lombok, ©2009-2022. [cit. 2022-11-19]. Dostupné z: https://projectlombok.org/features/.
43. *Null References The Billion Dollar Mistake* [online]. 2019. [cit. 2022-11-19]. Dostupné z: https://www.youtube.com/watch?v=YYkOWzrO3xg.
44. ORACLE. A/NEBO JEJÍ PŘIDRUŽENÉ SPOLEČNOSTI. *Optional* [online]. Java™ Platform, Standard Edition 8 API Specification, ©1993, 2023. [cit. 2022-11-19]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/index.html?java/util/Optional.html.
45. ORACLE AMERICA, INC. *Value-Based Classes* [online]. Java™ Platform, Standard Edition 8 API Specification, 2022-11-19. [cit. 2022-11-19]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/lang/doc-files/ValueBased.html.

46. JETBRAINS S.R.O. *Kotlin Programming Language* [online]. Kotlin, 2022-11-19. [cit. 2022-11-19]. Dostupné z: https://kotlinlang.org/.
47. JETBRAINS S.R.O. *Get Started with Kotlin* [online]. Kotlin Help, 2022-11-16. [cit. 2022-11-19]. Dostupné z: https://kotlinlang.org/docs/getting-started.html.
48. JETBRAINS S.R.O. *Null Safety* [online]. Kotlin Help, 2023-05-10. [cit. 2023-05-10]. Dostupné z: https://kotlinlang.org/docs/null-safety.html.
49. JETBRAINS S.R.O. *Data Classes* [online]. Kotlin Help, 2023-05-10. [cit. 2023-05-10]. Dostupné z: https://kotlinlang.org/docs/data-classes.html.
50. JETBRAINS S.R.O. *Delegation* [online]. Kotlin Help, 2023-05-10. [cit. 2023-05-10]. Dostupné z: https://kotlinlang.org/docs/delegation.html.
51. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Why Spring* [online]. Spring, 2023-05-07. [cit. 2023-05-07]. Dostupné z: https://spring.io/why-spring.
52. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Security* [online]. Spring Security, ©2023. [cit. 2023-05-07]. Dostupné z: https://spring.io/projects/spring-security.
53. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Projects* [online]. Spring, ©2023. [cit. 2023-05-07]. Dostupné z: https://spring.io/projects.
54. JOHNSON, Rod et al. *Language Support* [online]. Spring Framework Documentation, 2023-04-13. Dostupné také z: https://docs.spring.io/spring-framework/docs/current/reference/html/languages.html#kotlin.
55. AMAZON WEB SERVICES, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *What Is RESTful API? - RESTful API Beginner's Guide - AWS* [online]. Amazon Web Services, ©2022. [cit. 2022-11-26]. Dostupné z: https://aws.amazon.com/what-is/restful-api/.
56. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Form Login* [online]. Spring Security, ©2023. [cit. 2023-04-29]. Dostupné z: https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/form.html.
57. PARNISARI, Maria Ines et al. *Introduction to JSON Web Tokens* [online]. JSON Web Tokens, 2022-03-10. [cit. 2023-04-11]. Dostupné z: https://jwt.io/introduction.
58. RAO, Raja. *JSON Web Tokens (JWT) Are Dangerous for User Sessions—Here's a Solution* [online]. Redis, 2021-06-24. [cit. 2023-05-10]. Dostupné z: https://redis.com/blog/json-web-tokens-jwt-are-dangerous-for-user-sessions/.
59. *CookieHttpSessionIdResolver* [online]. spring-session-docs 3.0.1 API, 2023-05-01. [cit. 2023-05-01]. Dostupné z: https://docs.spring.io/spring-session/docs/current/api/org/springframework/session/web/http/CookieHttpSessionIdResolver.html.
60. WOLTER, Jan. *3. Do-It-Yourself Authentication Options* [online]. A Guide to Web Authentication Alternatives, 2003-10. [cit. 2023-05-01]. Dostupné z: https://www.unixpapa.com/auth.
61. *Most Used Web Frameworks among Developers 2022* [online]. Statista, 2022-09-08. [cit. 2023-05-11]. Dostupné z: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.
62. S, Sakthi. *Top 10 Benefits Of Angular You Should Be Aware Of In 2022* [online]. Calibraint, 2022-01-19. [cit. 2023-05-01]. Dostupné z: https://www.calibraint.com/blog/benefits-of-angular-web-application-development.
63. GOOGLE. *Ng Generate* [online]. Angular, ©2010-2023. [cit. 2023-05-11]. Dostupné z: https://angular.io/cli/generate.
64. DOCKER INC. *Get Docker* [online]. Docker Documentation, 2023-04-24. [cit. 2023-04-24]. Dostupné z: https://docs.docker.com/get-docker/.

65. DOCKER INC. *Get Started - Overview* [online]. Docker Documentation, 2023-04-24. [cit. 2023-04-24]. Dostupné z: https://docs.docker.com/get-started/.

66. DOCKER INC. *Docker Overview* [online]. Docker Documentation, 2023-04-24. [cit. 2023-04-24]. Dostupné z: https://docs.docker.com/get-started/overview/.

67. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL* [online]. PostgreSQL, 2023-05-02. [cit. 2023-05-02]. Dostupné z: https://www.postgresql.org/.

68. AMAZON WEB SERVICES, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *MySQL vs. PostgreSQL - Comparing Relational Database Management Systems (RDBMS) - AWS* [online]. Amazon Web Services, Inc., ©2023. [cit. 2023-05-02]. Dostupné z: https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/.

69. ITSKAWAL2000. *Difference between RDBMS and ORDBMS* [online]. GeeksforGeeks, 2020-10-25. [cit. 2023-05-02]. Dostupné z: https://www.geeksforgeeks.org/difference-between-rdbms-and-ordbms/.

70. RESCORLA, Eric. *HTTP Over TLS* [online]. 2000-05. [cit. 2023-04-23]. Request for Comments, RFC 2818. Internet Engineering Task Force. Dostupné z DOI: 10.17487/RFC2818.

71. AO KASPERSKY LAB. *Downgrade Attack* [online]. Kaspersky IT Encyclopedia, ©2022. [cit. 2022-11-26]. Dostupné z: https://encyclopedia.kaspersky.com/glossary/downgrade-attack/.

72. HODGES, Jeff; JACKSON, Collin; BARTH, Adam. *HTTP Strict Transport Security (HSTS)* [online]. 2012-11. [cit. 2023-04-29]. Request for Comments, RFC 6797. Internet Engineering Task Force. Dostupné z DOI: 10.17487/RFC6797.

73. KIRSTENS. *Cross Site Request Forgery (CSRF)* [online]. In collab. with WICHERS, Dave et al. OWASP Foundation, ©2023. [cit. 2023-04-29]. Dostupné z: https://owasp.org/www-community/attacks/csrf.

74. GOOGLE LLC. *HTTP Client - Security: Cross-Site Request Forgery (XSRF) Protection* [online]. Angular, 2022-11-14. [cit. 2023-04-29]. Dostupné z: https://angular.io/guide/http-security-xsrf-protection.

75. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Cross Site Request Forgery (CSRF) for Servlet Environments* [online]. Spring Security, ©2023. [cit. 2023-04-29]. Dostupné z: https://docs.spring.io/spring-security/reference/servlet/exploits/csrf.html#servlet-csrf.

76. MDN PŘISPĚVATELÉ. *MDN* [online]. MDN, 2023-04-26. [cit. 2023-04-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

77. JOHNSON, Rod et al. *Web on Servlet Stack* [online]. Spring Framework Documentation, 2023-04-13. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html.

78. OWASP FOUNDATION. *Cross-Site Scripting (XSS)* [online]. OWASP Foundation Wiki, 2018-06-05. [cit. 2023-04-29]. Dostupné z: https://wiki.owasp.org/index.php/Cross-site_Scripting_(XSS).

79. GOOGLE LLC. *Security* [online]. Angular, 2022-02-28. [cit. 2023-04-30]. Dostupné z: https://angular.io/guide/security.

80. CHEATSHEETS SERIES TEAM. *Password Storage* [online]. OWASP Cheat Sheet Series, ©2021. [cit. 2023-05-08]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.

81. BEYOND IDENTITY ™. *What Is a Rainbow Table Attack?* [online]. Beyond Identity, ©2023. [cit. 2023-05-08]. Dostupné z: https://www.beyondidentity.com/glossary/rainbow-table-attack.

82. TEAM, Angular Components. *Angular Material* [online]. Angular Material, ©2010-2023. [cit. 2023-05-11]. Dostupné z: https://material.angular.io/.

83. *Get Started* [online]. Material Design, 2023-04-23. [cit. 2023-04-23]. Dostupné z: https://m3.material.io/get-started.

84. ANGULAR COMPONENTS TEAM. *Theming Angular Material* [online]. Angular Material, ©2010-2023. [cit. 2023-04-24]. Dostupné z: https://material.angular.io/guide/theming.

85. TOHAR, Doron. *A Simple Type Safe HTTP Client Wrapper* [online]. Medium, 2021-11-03. [cit. 2023-05-11]. Dostupné z: https://medium.com/@dorontohar/a-simple-type-safe-http-client-wrapper-edb7df9317db.

86. ANGULAR COMPONENTS TEAM. *Datepicker* [online]. Angular Material, ©2010-2023. [cit. 2023-05-10]. Dostupné z: https://material.angular.io/components/datepicker/overview.

87. *Getting Started* [online]. date-fns. [cit. 2023-05-10]. Dostupné z: https://date-fns.org/docs/Getting-Started.

88. THOMAS KOENIG; MICHAEL KERRISK. *Locale(7)* [online]. Linux manual page, 2022-12-18. [cit. 2023-04-23]. Dostupné z: https://man7.org/linux/man-pages/man7/locale.7.html.

89. *Notiflix* [online]. npm, 2023-01-11. [cit. 2023-05-11]. Dostupné z: https://www.npmjs.com/package/notiflix.

90. CHEATSHEETS SERIES TEAM. *Authentication* [online]. OWASP Cheat Sheet Series, ©2021. [cit. 2023-04-29]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

91. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Session* [online]. Spring Session, ©2023. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-session/reference/index.html.

92. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *HttpSession Integration* [online]. Spring Session, ©2023. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-session/reference/http-session.html#httpsession.

93. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Security* [online]. Spring Security, ©2023. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-security/reference/index.html.

94. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Authentication* [online]. Spring Security, ©2023. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-security/reference/features/authentication/index.html.

95. OKTA, INC. *What Is Authorization? - Examples and Definition* [online]. Auth0, ©2023. [cit. 2023-04-26]. Dostupné z: https://auth0.com/intro-to-iam/what-is-authorization.

96. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Data* [online]. Spring Data, ©2023. [cit. 2023-05-10]. Dostupné z: https://spring.io/projects/spring-data#overview.

97. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Data JPA* [online]. Spring, ©2023. [cit. 2023-04-27]. Dostupné z: https://spring.io/projects/spring-data-jpa.

98. BRISBIN, Jon et al. *Introduction* [online]. Spring Data REST Reference Guide, 2023-04-14. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-data/rest/docs/current/reference/html/#intro-chapter.

99. TERRACOTTA, INC. *What Is the Quartz Job Scheduling Library?* [online]. Quartz Enterprise Job Scheduler, 2023-04. [cit. 2023-04-26]. Dostupné z: https://www.quartz-scheduler.org/.

100. *What Is Apache FreeMarker™?* [online]. FreeMarker, 2023-01-15. [cit. 2023-05-08]. Dostupné z: https://freemarker.apache.org/.

101. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Spring Initializr* [online]. Spring Initializr, ©2023. [cit. 2023-03-22]. Dostupné z: https://start.spring.io/.

102. WEBB, Phillip et al. *Developing with Spring Boot* [online]. Spring Boot Reference Documentation, 2023-04-20. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-boot/docs/3.0.6/reference/html/using.html.

103. JOHNSON, Rod et al. *Core Technologies - The IoC Container - Introduction to the Spring IoC Container and Beans* [online]. Spring Framework Documentation, 2023-04-13. [cit. 2023-04-26]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-definition.

104. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Architecture* [online]. Spring Security, ©2023. [cit. 2023-04-29]. Dostupné z: https://docs.spring.io/spring-security/reference/servlet/architecture.html.

105. LIQUIBASE INC. *Changelog* [online]. Liquibase, ©2023. [cit. 2023-05-09]. Dostupné z: https://docs.liquibase.com/concepts/changelogs/home.html.

106. LIENBERHERR, K. J. Formulations and Benefits of the Law of Demeter. *SIGPLAN Not.* [online]. 1989, roč. 24, č. 3, s. 67–78 [cit. 2023-08-05]. ISSN 0362-1340. Dostupné z DOI: 10.1145/66083.66089.

107. NICOLL, Stephane; BRANNEN, Sam; DROTBOHM, Oliver. *Annotation Interface TransactionalEventListener* [online]. Spring Framework 6.0.8 AP, 2023-04-28. [cit. 2023-04-28]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/event/TransactionalEventListener.html.

108. NICOLL, Stephane; HOELLER, Juergen; BRANNEN, Sam. *Enum Class TransactionPhase* [online]. Spring Framework 6.0.8 API, 2023-04-28. [cit. 2023-04-28]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/event/TransactionPhase.html.

109. GIERKE, Oliver et al. *Working with Spring Data Repositories* [online]. Spring Data JPA - Reference Documentation, 2023-03-03. [cit. 2023-04-27]. Dostupné z: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.

110. GIERKE, Oliver et al. *Reference Documentation* [online]. Spring Data JPA - Reference Documentation, 2023-03-03. [cit. 2023-04-27]. Dostupné z: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#projections.

111. VMWARE, INC. NEBO PŘIDRUŽENÉ SPOLEČNOSTI. *Expression-Based Access Control* [online]. Spring Security, ©2023. [cit. 2023-05-11]. Dostupné z: https://docs.spring.io/spring-security/reference/servlet/authorization/expression-based.html#_access_control_using_preauthorize_and_postauthorize.

112. FIELDING, Roy T.; NOTTINGHAM, Mark; RESCHKE, Julian. *HTTP Semantics* [online]. 2022-06. [cit. 2023-04-27]. Request for Comments, RFC 9110. Internet Engineering Task Force. Dostupné z DOI: 10.17487/RFC9110.

113. BECHTOLD, Stefan et al. *Overview* [online]. JUnit 5 User Guide, 2023-04-22. [cit. 2023-04-24]. Dostupné z: https://junit.org/junit5/docs/current/user-guide/#overview.

114. *TestEngine* [online]. JUnit 5.0.1 API, 2023-04-24. [cit. 2023-04-24]. Dostupné z: https://junit.org/junit5/docs/5.0.1/api/org/junit/platform/engine/TestEngine.html.

115. MACKINNON, Tim; FREEMAN, Steve; CRAIG, Philip. Endo-Testing: Unit Testing with Mock Objects [online]. 2001 [cit. 2023-05-05]. Dostupné z: https://www2.ccs.neu.edu/research/demeter/related-work/extreme-programming/MockObjectsFinal.PDF.

116. PROFINIT EU, S.R.O. *Mobilní Aplikace pro Egidio – Platforma pro Řešení Katastrof* [online]. Jsme Profinit, ©2023. [cit. 2023-04-23]. Dostupné z: https://jsmeprofinit.eu/zaverecne-prace/mobilni-aplikace-pro-egidio-platforma-pro-reseni-katastrof/.

Obsah přiloženého archivu

- `readme.txt` — Stručný popis obsahu archivu
- `main` — Aplikace
  - `README.md` — Zdrojový kód instalační příručky
  - `README.pdf` — Instalační příručka
  - `LICENSE` — Licenční soubor zdrojových kódu přiložených v práci
  - `source` — Jednotlivé podčásti aplikace
    - `frontend` — Adresář obsahující frontendovou část aplikace
      - `src` — Složka se zdrojovými kódy frontendu
      - `Dockerfile` — Dockerfile použitý pro sestavení FE v docker compose
      - `nginx.conf` — Nastavení Nginx serveru
      - `package.json` — Nastavení NPM
      - `thirdparty.txt` — Licence knihoven třetích stahovaných při sestavení aplikace
    - `backend` — Adresář s backendovou částí aplikace
      - `src` — Zdrojové kódy backendu
      - `Dockerfile` — Dockerfile pro sestavení backendu
      - `build.gradle.kts` — Hlavní nastavení balíčkovacího systému Gradle
      - `thirdparty.txt` — Licence knihoven třetích stahovaných při sestavení aplikace
    - `proxy` — Adresář obsahující soubory potřebné pro sestavení proxy serveru
      - `Dockerfile` — Dockerfile pro sestavení
      - `nginx.conf` — Konfigurace proxy serveru
    - `openapi.html` — Dokumentace API rozhraní
    - `.env` — Environmentální proměnné používáné docker-compose
    - `docker-compose.yml` — Docker Compose soubor používaný k sestavení aplikace
- `misc` — Dodatečné materiály vzniklé či použité v procesu tvorby práce
  - `graphics` — Grafické prvky použité ve náčrtech obrazovek (nevytvořené autorem)
  - `models` — Modely různých věcí v aplikaci. Některé již nemusí být aktuální.
  - `screens` — Fotky některých obrazovek v aplikaci
  - `wireframes` — Náčrty některých obrazovek. Většina již není aktuální
  - `LICENSE` — Licenční soubor
- `scenarios` — Složka s testovacími scénáři
  - `test-scenarios.pdf` — PDF dokument se scénáři
  - `source` — Zdrojové soubory testovacích scénářů
- `thesis` — Bakalářské práce
  - `source` — Zdrojové kódy bakalářské práce
  - `thesis.pdf` — Text bakalářské práce