TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Systém pro vizualizaci informací o rodinných firmách a analýzu jejich vitality

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Jan Pilař**
Vedoucí práce: Ing. Marián Lamr, Ph.D.
Ústav mechatroniky a technické informatiky

Konzultant práce: Ing. Přemysl Svoboda
Ústav mechatroniky a technické informatiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Systém pro vizualizaci informací o rodinných firmách a analýzu jejich vitality

Jméno a příjmení: **Jan Pilař**
Osobní číslo: M16000048
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Proveďte analýzu se zadavatelem a navrhněte systém pro administraci a vizualizaci informací o rodinných firmách a analýzu a vyhodnocení jejich vitality.
2. Vytvořte administrační část pro správu informací o firmách.
3. Naprogramujte část pro vizualizaci informací jako aplikaci pro systém Android.
4. Naprogramujte webovou aplikaci pro analýzu a vyhodnocení vitality rodinných firem.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] CONNOLLY, Thomas M. a Carolyn E. BEG. Database systems: a practical approach to design, implementation, and management. Sixth edition. Boston: Pearson, [2015]. ISBN 978-0132943260.

[2] HEROUT, Pavel. Učebnice jazyka Java. 3., rozš. vyd. České Budějovice: Kopp, 2007. ISBN 978-8072323234.

[3] GRIFFITHS, Dawn a David GRIFFITHS. Head first Android development: a brain-friendly guide. 2nd edition. Sebastopol, CA: O'Reilly Media, 2017. Head first series. ISBN 978-1491974056.

Vedoucí práce:	Ing. Marián Lamr, Ph.D. Ústav mechatroniky a technické informatiky
Konzultant práce:	Ing. Přemysl Svoboda Ústav mechatroniky a technické informatiky
Datum zadání práce:	9. října 2020
Předpokládaný termín odevzdání:	17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 9. října 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

21. dubna 2021 Jan Pilař

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Mariánu Lamrovi, Ph.D. za cenné rady, ochotu a trpělivost, kterou mi v průběhu zpracování bakalářské práce věnoval.

Systém pro vizualizaci informací o rodinných firmách a analýzu jejich vitality

Abstrakt

Tato práce se zabývá vytvořením softwaru pro podporu rodinných firem. Cílem práce je vytvořit systém, který umožní vytvářet testovací moduly určené pro rodinné firmy a usnadní rodinným firmám mezi sebou navazovat kontakty na základě lokace a činnosti. Testovací moduly slouží k testování rodinných firem. Pomocí těchto modulů lze vyhodnotit ukazatele rodinných firem, jako je například jejich vitalita.

V této práci je vytvořena webová aplikace pro NodeJS za využití frameworku Express. Aplikace je rozčleněná na administrační část pro správu testovacích modulů a uživatelskou část, ve které mohou rodinné firmy testovací moduly využít. K ukládání dat se využívá relační databáze PostgreSQL. Také je vytvořena mobilní aplikace pro systém Android, která komunikuje se serverem přes RESTful API. Tato aplikace umožňuje otestovaným rodinným firmám navazovat mezi sebou kontakty.

Klíčová slova: NodeJS, Express, RESTful API, PostgreSQL, Android, rodinná firma, JavaScript

System designed to visualise data regarding family enterprises and analyse their vitality

Abstract

This work is focused on creating software to support family enterprises. The goal of this work is to develop a system designed for creating testing modules intended for family enterprises and help family enterprises make contacts with each other based on location and activity. Testing modules are used to test family enterprises. These modules allow family enterprises to evaluate their indicators, such as their vitality.

This work contains creation of a web application for NodeJS using framework Express. Application is divided into administration part for management of testing modules and user part in which family enterprises can use the testing modules to test themselves. To store data the system utilizes relational database PostgreSQL. It also contains creation of mobile application for Android system that uses a RESTful API to communicate with the server. This application allows family enterprises that tested themselves to make contacts with each other.

Keywords: NodeJS, Express, RESTful API, PostgreSQL, Android, family enterprise, JavaScript

Obsah

1 Úvod **11**

2 Programování webových aplikací **13**
2.1 Webové aplikace a nativní aplikace . . . 13
2.2 Typy webových aplikací . . . 14
2.2.1 Single-page aplikace . . . 14
2.2.2 Multi-page aplikace . . . 14
2.3 Frameworky . . . 14
2.3.1 Klientské frameworky . . . 14
2.3.2 Serverové frameworky . . . 15
2.4 Programování webové aplikace v NodeJS . . . 15
2.5 Express . . . 16
2.6 Objektově relační mapování . . . 17
2.7 Bootstrap . . . 17
2.8 Zabezpečení přístupu přihlášeného uživatele do webové aplikace . . . 17
2.8.1 Session . . . 17
2.8.2 OAuth . . . 18
2.8.3 Porovnání . . . 18
2.9 RESTful API . . . 18
2.9.1 Jakým způsobem se RESTful API používá . . . 18
2.9.2 Struktura URL v RESTful API . . . 19

3 Získávání základních dat z veřejných rejstříků a výběr databáze **20**
3.1 Administrativní registr ekonomických subjektů . . . 20
3.2 Geokódování . . . 20
3.2.1 Google Geocoding API . . . 20
3.2.2 Nominatim . . . 21
3.3 Výběr databáze . . . 21
3.3.1 Firebase . . . 21
3.3.2 PostgreSQL . . . 21
3.3.3 PostGIS . . . 21

4 Programování mobilních aplikací **22**
4.1 Programování aplikací pro systém Android . . . 22
4.2 Aktivity . . . 23

4.3 Zdroje aplikace . . . 24
4.3.1 Drawable . . . 24
4.3.2 Layouty . . . 24
4.3.3 Values . . . 25

5 Systém pro testování vitality a vizualizaci dat o firmách 26
5.1 Požadavky . . . 27
5.2 Návrh řešení . . . 27
5.3 Návrh databáze . . . 28
5.3.1 Rozbor ukládaných dat . . . 28
5.3.2 Schéma . . . 29

6 Webová aplikace 31
6.1 Administrační část . . . 31
6.1.1 Moduly . . . 32
6.1.2 Firmy . . . 33
6.1.3 Uživatelé . . . 35
6.1.4 Blacklist . . . 35
6.1.5 Správa správců . . . 36
6.1.6 Vyplnit data za uživatele . . . 36
6.2 Uživatelská část . . . 36
6.2.1 Moduly . . . 37
6.2.2 Android . . . 37
6.3 Implementace REST API . . . 38
6.3.1 Zabezpečení s OAuth . . . 38
6.3.2 Ověřování identity . . . 38
6.3.3 Vytvoření endpointů pro potřebné URL . . . 39
6.4 Hosting . . . 39

7 Mobilní aplikace 40
7.1 Úvod do aplikace . . . 40
7.2 Hlavní menu . . . 40
7.3 Filtrace . . . 41
7.4 Provozovny na mapě . . . 41
7.5 Detail provozovny . . . 41
7.6 Detail firmy . . . 42

8 Závěr 43

Seznam odborné literatury 44

Přílohy 46

Seznam obrázků

2.1 Npmtrends.com: Počet týdenních stažení NodeJS frameworků během posledních dvou let. 16

4.1 Developer.android.com: Životní cyklus aktivity. 23

5.1 ER diagram testovacích modulů. 30

6.1 Moduly. 32
6.2 Responzivní zobrazení modulů na malém displeji. 32
6.3 Dotazníky. 33
6.4 Otázky. 33
6.5 Firmy. 34
6.6 Provozovna. 35
6.7 Blacklist. 35
6.8 Správa administrátorů. 36
6.9 Vyplňování dotazníku modulu. 37
6.10 Stahování Android aplikace. 38

7.1 Zobrazení provozoven na mapě. 41
7.2 Detail firmy. 42

Seznam zkratek

ARES	Administrativní registr ekonomických subjektů
REST	Representational state transfer je architektura navržená pro efektivní předávání dat mezi serverem a klientem
API	Aplikační programovací rozhraní, definuje funkce, které lze využívat z jiného programu
JWT	JSON Web Token, identifikátor využívaný pro ověření identity
SPA	Single-page aplikace
MPA	Multi-page aplikace
ORM	Objektově relační mapování
DOM	Objektový model HTML dokumentu
NPM	Node package manager
JVM	Java virtual machine
SP	Scalable Pixels
DP	Density-independent Pixels

1 Úvod

Tato práce vznikla na základě podnětu z Ekonomické fakulty Technické univerzity v Liberci. Ekonomická fakulta Technické univerzity v Liberci a Asociace malých a středních podniků a živnostníků ČR ve spolupráci s Masarykovou univerzitou v rámci řešení projektu Technologické agentury České republiky vyvinuli původně webovou aplikaci pro hodnocení vitality rodinných firem. Ekonomická fakulta Technické univerzity v Liberci chtěla původní řešení rozšířit o mobilní aplikaci, kterou by si po otestování mohly rodinné firmy stáhnout. V této aplikaci by bylo možné seznámit se s ostatními otestovanými firmami a také by jim pomáhala v navázání kontaktu. Mobilní aplikace by však potřebovala ke své funkci data, která se v původním řešení systému nenachází. Z tohoto důvodu se došlo k závěru, že by bylo nejlepší vytvořit celý systém znovu. Nové řešení by pak mohlo být oproti původnímu flexibilnější a umožňovalo by využívat administrační část pro správu testovacích modulů. Zde by pak bylo možné vytvářet testovací moduly pro testování nejen vitality rodinné firmy, ale i dalších ukazatelů rodinné firmy.

Nové řešení systému tedy bude zahrnovat administrační část pro vytváření nových testovacích modulů a správu dat rodinných firem. Dále bude obsahovat uživatelskou část pro rodinné firmy, ve které se rodinná firma bude moci otestovat využitím vytvořených testovacích modulů. Také bude součástí systému již zmíněná mobilní aplikace pro otestované rodinné firmy. Ta bude využívat informace o firmách vedených v systému a k přehledné vizualizaci těchto dat bude využívat mapu.

Aby bylo testování firem co nejdostupnější a nejpohodlnější, bude vhodné realizovat testování pomocí webových stránek. Celá část pro administraci i testování by tak mohla být vyřešena pomocí webové aplikace.

Jelikož budou potřeba základní data o firmách pro zobrazení na mapě v mobilní aplikaci, bude vhodné data z části získávat strojově z veřejně dostupných zdrojů. Usnadní se tak proces zaregistrování nové firmy pro uživatele a zároveň se i ověří existence této firmy.

Pro účely této práce je tak potřeba se seznámit s problematikou programování webových aplikací a mobilních aplikací pro Android. Bude také potřeba použít databázi k ukládání informací o firmách. Jelikož mobilní aplikace bude využívat stejná data z databáze, bude nutné pro ni vytvořit zabezpečený přístup. Při přímé komunikaci mezi klientskou aplikací a databází jinak hrozí mnohé nebezpečí. Velkou část z nich lze vyřešit použitím webového API pro komunikaci mezi mobilní aplikací a serverem, kterým se zpřístupní data z databáze pro mobilní aplikaci.

V druhé kapitole je rozebrána problematika programování webových aplikací. Je zde popsán rozdíl mezi single-page aplikací a multi-page aplikací a jakým způsobem taková aplikace ovlivňuje požadavky na klientskou část a serverovou část. Poté následuje bližší

seznámení se serverovou technologií NodeJS a u této technologie často využívaným frameworkem Express. Je probráno jakým způsobem se aplikuje architektonický styl REST na webové API, aby mohlo být nazváno RESTful API. V třetí kapitole je rozebráno jaká data lze strojově získat o firmách z veřejných rejstříků a proč je nutné využívat geokódování. Popisuje se rozdíl mezi NoSQL databází Firebase a SQL databází PostgreSQL. Čtvrtá kapitola se zabývá výhodami mobilních aplikací a popisuje způsob, jakým se aplikace programují pro systém Android.

V páté kapitole je provedena analýza zadaného problému, ve které se dle požadavků navrhne systém pro vizualizaci dat o rodinných firmách a analýzu jejich vitality. Pro navržený systém je pak pro vybranou databázi PostgreSQL popsáno, jakým způsobem se v ní budou strukturovat ukládaná data. Implementaci samotné webové aplikace se věnuje šestá kapitola. Je v ní popsáno rozčlenění aplikace na administrační část a uživatelskou část za využití NodeJS a Express na serveru a frameworku Bootstrap na klientu. Dále se popisuje implementace RESTful API a v poslední části kapitoly je rozebrán hosting webové aplikace a co přináší použití Nginx jako reverse proxy. V následující kapitole je implementována mobilní aplikace pro Android využívající mapu a komunikující se serverem pomocí RESTful API.

2 Programování webových aplikací

V dnešní době rychlého internetu a bezdrátových sítí s velkým plošným rozsahem je snadné se na internet připojit téměř kdekoliv. Nejen z tohoto důvodu jsou webové aplikace čím dál víc využívané a oblíbenější. K využití webové aplikace totiž postačí webový prohlížeč, který je dostupný na obrovském rozsahu různých typů zařízení s různými operačními systémy. Jsou to především klasické stolní počítače a notebooky, ale také mobilní zařízení jako chytré telefony a tablety. Jelikož aplikace běží ve webovém prohlížeči, nezáleží na tom, jestli zařízení běží pod systémem Windows, Linux, macOS, Android či iOS. Uživatelé tak mohou používat webovou aplikaci nehledě na jejich operačním systému.

2.1 Webové aplikace a nativní aplikace

Webové aplikace mají oproti nativním aplikacím mnohé výhody i nevýhody. [1] Nativní aplikace, které jsou vytvořeny pro každý operační systém zvlášť je nutné nainstalovat. Po nainstalování aplikace zabírá místo na disku. Oproti tomu se webová aplikace pouze dočasně nahraje přes internet do prohlížeče a na disku se tak vůbec nenachází. To ocení obzvláště uživatelé, kteří používají stejnou aplikaci na více zařízeních. Nejsou tak limitovaní konkrétním operačním systémem a aplikaci mohou používat i na zařízeních, kde nemají oprávnění provést instalaci.

Oproti zřejmé výhodě v dostupnosti webových aplikací je zde však i několik nevýhod. Nativní aplikace jsou nainstalované na zařízení. Lze je spustit a využívat i bez internetového připojení, pokud ho ke svému chodu nepotřebují. Jsou také úzce propojeny s funkcemi operačního systému a mohou tak poskytovat více možností specifických pro daný operační systém.

Webové aplikace se skládají z klientské a serverové části. Za klientskou část se považují webové stránky, které reagují na vstup od uživatele a na kterých je uživateli umožněno provádět požadované akce. Serverová část pak zpracovává tyto požadavky a ukládá data do databáze. Díky serverové části a využití databáze budou změny, které uživatel provedl ve webové aplikaci dostupné i později a z jiného zařízení.

2.2 Typy webových aplikací

Webové aplikace lze primárně rozlišovat jako single-page aplikace (SPA) a multi-page aplikace (MPA). [2]

2.2.1 Single-page aplikace

SPA je aplikace, která se načte ze serveru pouze jednou a nepotřebuje pro svůj chod znovu načítat stránku. Se serverem si pak vyměňuje pouze data. Použitím SPA lze u webové aplikace napodobit chování aplikace nativní. Žádné načítaní nových stránek, pouze dat. Uživatelé tak nemusejí zbytečně čekat během využívání aplikace.

Ze serveru se po načtení aplikace dále posílají jen data pro použití v aplikaci. Renderování stránek takto nemusí probíhat na serveru, ale na klientu. Tímto způsobem pak lze použít stejný kód na serveru i pro vytvoření nativních aplikací.

SPA také mohou efektivně ukládat data v lokálním úložišti. S těmito daty pak mohou pracovat i později a případně tak mohou fungovat i bez připojení k internetu.

Nevýhodou SPA je, že mají delší počáteční načtení aplikace. Pro chod aplikace je totiž nutné stáhnout na klienta celý klientský framework. SPA využívají pro obstarání svých funkcí JavaScript. Ten však může být v prohlížeči uživatelem zakázán a aplikace tak nebude fungovat.

2.2.2 Multi-page aplikace

V multi-page aplikacích je při každé změně nutné vyrenderovat novou stránku ze serveru. To však znamená neustálé načítání téměř stejného obsahu. Proto se v dnešní době v MPA běžně používá i asynchronní JavaScript pro aktualizování části webové stránky bez nutnosti jejího kompletního nového načtení. Ve výsledku se tak využívá členění částí aplikace do samostatných stránek jako v multi-page aplikaci a dále na těchto stránkách se využívá asynchronní komunikace se serverem bez nutnosti dalšího načtení stránky, stejně jako single-page aplikace.

2.3 Frameworky

K tvorbě webových stránek se používá HTML, CSS, JavaScript. Zatímco klientská část webové aplikace je tedy limitována na technologie použité v prohlížeči, pro serverovou část lze vybírat z mnoha jazyků, např. PHP, JavaScript, Java, Python, Ruby.

K usnadnění vytváření webových aplikací vznikly mnohé frameworky jak pro klienta, tak i pro server.

2.3.1 Klientské frameworky

Frameworky pro klientskou část jsou navrženy pro přehledné uchování stavu dat na klientovi a usnadnění manipulace s objektovým modelem HTML dokumentu (DOM). Takové frameworky jsou například React, Angular nebo Vue.

2.3.2 Serverové frameworky

Tyto frameworky pomáhají zjednodušit nadefinování URL cest a jejich obsluhu, interakci s databází, autorizaci uživatele či formátování výstupu. [3]

- Django je framework pro Python. Poskytuje ze základu téměř vše, co by mohl vývojář potřebovat. Díky tomu vše funguje dohromady jak má.
- Express je rychlý, flexibilní a minimalistický framework pro NodeJS. Využívá JavaScript.
- Ruby on Rails je webový framework vytvořen pro jazyk Ruby. Podobně jako Django obsahuje souhrn standardních mechanismů.
- Laravel je PHP framework s elegantní a expresivní syntaxí. Obsahuje jednoduché a rychlé routování či intuitivní databázové ORM.
- Spring Boot je framework pro jazyk Java, který je navržen pro využití knihoven třetích stran. Zároveň však umožňuje použití s minimální konfigurací.

2.4 Programování webové aplikace v NodeJS

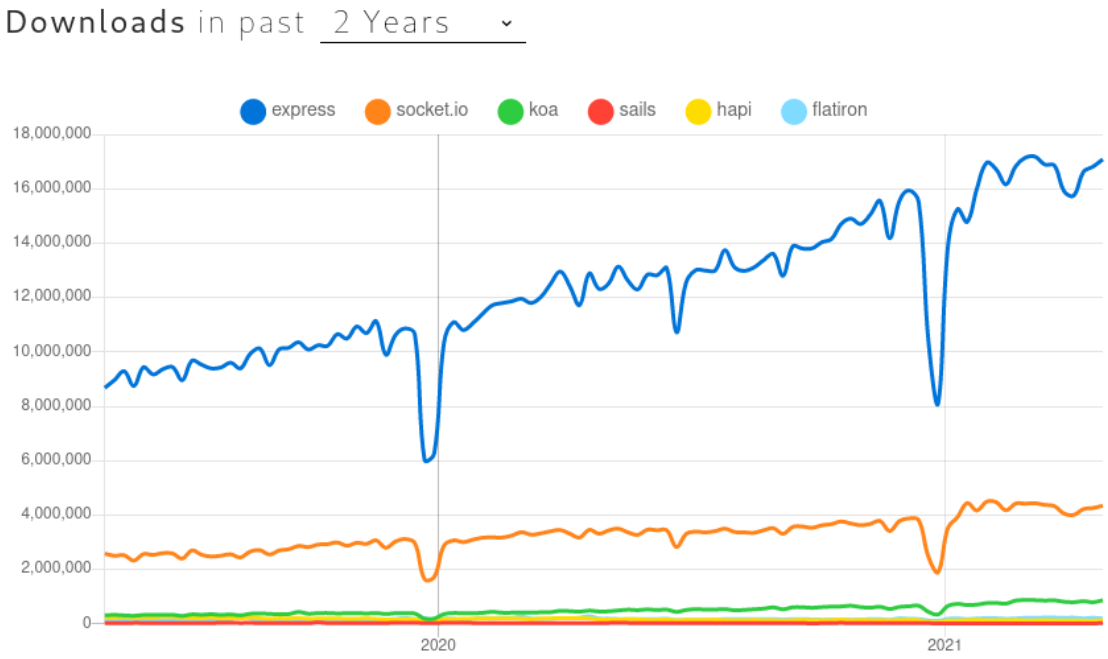
NodeJS je asynchronní prostředí řízené událostmi pro spouštění JavaScript aplikací na serveru bez použití webového prohlížeče. [4] Využitím NodeJS tak lze programovat jak serverovou část, tak i klientskou část ve formě webových stránek za použití stejného programovacího jazyka JavaScript.

NodeJS je navržen pro vytváření škálovatelných webových aplikací. Využívá event loop stejně jako JavaScript ve webovém prohlížeči. NodeJS aplikace tak běží jako jediný proces, bez nutnosti vytváření nového vlákna pro každý request. Při spuštění asynchronní operace nezablokuje vlákno a pokračuje v zpracování teprve až asynchronní operace skončí. Díky tomu zvládne zpracovávat velké množství současných připojení bez nutnosti správy více vláken.

V NodeJS lze používat nové verze ECMAScript standardů, jelikož verze běžícího NodeJS na serveru není limitovaná webovými prohlížeči.

NodeJS využívá node package manager (npm). Npm je balíčkovací systém, na kterém se nachází více jak milión open source JavaScript knihoven pro NodeJS. Základním kamenem pro vytváření webových aplikací je http modul. Pomocí něho se vytvoří webový server. Při každém requestu na server jsou pro zpracování zpřístupněny objekty request a response. Na objektu request se nachází informace o requestu, hlavičky a data. Objekt response je využíván pro definování hlaviček a dat, které se mají odeslat zpět.

Podle počtu stažení jednotlivých frameworků z npm je zřejmé, že nejvíc používaný framework v NodeJS je právě Express a jeho běžný počet stažení v průběhu času stále značně roste.



Obrázek 2.1: Npmtrends.com: Počet týdenních stažení NodeJS frameworků během posledních dvou let.

2.5 Express

Express je minimalistický, flexibilní a nejpopulárnější webový framework pro NodeJS. [5] Umožňuje vytvářet funkce, kterými se budou zpracovávat requesty s různými HTTP metodami a URL cestami, neboli routy. Umožňuje použít view template engine, který se používá k renderování HTML souborů s konkrétními daty.

Express využívá middleware. Middleware je funkce, která je zařazena do řetězce funkcí, které request zpracovávají. Každý middleware může request zkontrolovat, upravit, případně i ukončit. Pokud request v daném middlewaru neskončí, je nutné aby ho middleware poslal další funkci v řetězci zavoláním funkce next(). V případě, že nastala nějaká chyba při zpracování v middlewaru, lze ji delegovat na další middlewary tím, že se informace o chybě přidá jako parametr funkce next(). Díky tomu, že Express využívá middleware, je pro Express vytvořena spousta knihoven řešících dílčí úkoly formou middleware funkcí.

Express je takzvaně unopinionated. To znamená, že neurčuje jakým způsobem se má strukturovat. Jedním z oblíbených způsobů je architektura Model-View-Controller (MVC). [6] Zde je kladen důraz na oddělení logiky aplikace od zobrazení. Tímto rozdělením se napomáhá k lehčí údržbě kódu aplikace.

MVC architektura se skládá ze tří částí. Model je část, ve které je definována struktura dat a logika aplikace. View je část, která se stará o uživatelské rozhraní, jakým způsobem se zobrazí data aplikace. Controller je část, ve které je ovládací logika, která určuje jak jsou navzájem části View a Model propojené.

2.6 Objektově relační mapování

Objektově relační mapování (ORM) se využívá k usnadnění práce s SQL databází tak, že namísto komplikovaného SQL dotazu se dotaz vytváří pomocí objektů daného programovacího jazyka. [7] Pomocí ORM je vytvořena abstrakce nad databází. Pokud ORM podporuje dialekty pro více různých databází, lze v případě nutnosti zaměnit databázi bez velkých potíží. ORM také může podporovat i pokročilé funkce jako jsou transakce, migrace či connection pooling. Použitím ORM tak lze vcelku jednoduše vytvářet i komplexní dotazy na databázi. Oproti tomu však většina dotazů tvořených přes ORM bude pomalejší, než by byl správně sestrojený SQL dotaz. Také je nutné pro ORM nejdříve nadefinovat modely tabulek a jejich vzájemné relace.

2.7 Bootstrap

Klientský CSS framework Bootstrap slouží pro vytváření responzivních webových stránek. [8] Obsahuje HTML a CSS šablony pro vytváření běžných komponent uživatelského rozhraní jako jsou tlačítka, navigace, formuláře či popisky.

Bootstrap používá předdefinované styly tříd. Pomocí těchto tříd lze nadefinovat jak má rozvržení webu vypadat. Pomocí identifikátorů pro velikost displeje zařízení lze snadno nadefinovat různé zobrazení pro každou kategorii. Všechny komponenty používají v celé aplikaci stejné styly, tím je zaručena konsistence vzhledu webových stránek.

Bootstrap je kompatibilní se všemi moderními webovými prohlížeči. Aby každé webové stránky používající Bootstrap nevypadaly stejně, je k dispozici mnoho variant námětů.

2.8 Zabezpečení přístupu přihlášeného uživatele do webové aplikace

Poté co se prokáže identita uživatele přihlašovacími údaji, je uživateli na čas přidělen přístup na web server. Toho lze docílit použitím session nebo OAuth. [9]

2.8.1 Session

Po úspěšném ověření přihlašovacích údajů je pro uživatele na serveru vytvořena session. Veškerá data o vytvořeném spojení jsou uložena do databáze a uživatel dostane zpět jen identifikátor této session, kterým se bude po dobu trvajícího připojení prokazovat. Do session se průběžně ukládají data potřebná k chodu aplikace pro uživatele. Session jsou tedy stavové.

Session může mít pevně daný čas vypršení, nebo tuto dobu lze obnovovat při využívání aplikace, aby nedošlo k nucenému odhlášení, když uživatel aplikaci stále používá. K zabezpečení session na klientovi se identifikátor session posílá jako HttpOnly cookie. HttpOnly cookie jsou pouze posílány webovým prohlížečem zpět na server a žádný Javascript kód je nemůže přečíst a zjistit tak identifikátor session.

2.8.2 OAuth

OAuth je protokol, který definuje bezstavový způsob zajištění přístupu k datům pomocí tokenů. Po ověření přihlašovacích údajů jsou pro uživatele vytvořeny 2 tokeny, přístupový token a obnovovací token. Na serveru se uloží obnovovací token, a oba tokeny jsou poslány zpět uživateli. Pro přístup k datům na serveru nyní bude používán přístupový token, dokud je platný. Dle protokolu se přístupový token posílá při dotazu na server v hlavičce Authorization. Jakmile přístupovému tokenu vyprší platnost, použije se obnovovací token k zažádání o nový přístupový token. Obnovovací token je dle protokolu posílán v těle dotazu. Na serveru je obnovovací token uživatele porovnán s tím uloženým a pokud se shodují, server vygeneruje nový přístupový token pro uživatele.

Obnovovací token může aplikace využívat k získání přístupu k datům a uživatel se tak nemusí sám znovu přihlašovat. Toto řešení je ideální, pokud není potřeba udržovat stav na serveru.

2.8.3 Porovnání

Hlavní rozdíl mezi těmito způsoby udržení přístupu k webserveru je stavovost session a bezstavovovst tokenů. Session je tedy vhodná k použití s webovými stránkami. Na serveru se tak může ukládat stav aplikace pro uživatele a dokud je uživatel aktivní, session se bude obnovovat. Po delším nevyužíváním aplikace session expiruje a uživatel se znovu přihlásí.

Ovšem mobilní aplikace bude pouze občas potřebovat přístup k datům. K tomu je zbytečné, aby musel být na serveru vytvářen dočasný záznam, jelikož není potřeba uchovávat stav pro aplikaci na serveru. V mobilní aplikaci se tedy bude používat protokol OAuth s tokeny, navíc se díky tomu nebude muset uživatel skoro nikdy znovu přihlašovat.

2.9 RESTful API

Aby měla mobilní aplikace přístup k datům systému, bude potřeba umožnit komunikaci mezi webovým serverem a mobilní aplikací. K tomu se běžně využívají RESTful API.

Za RESTful API je považováno takové aplikační rozhraní, které je navrženo podle architektury REST, dlouze Representational state transfer. Tato architektura je navržena pro rychlost, škálovatelnost a spolehlivost webových služeb. Je využíváno k bezstavovému přenosu dat mezi serverem a klientskou aplikací. [10]

2.9.1 Jakým způsobem se RESTful API používá

Webový server může využívat RESTful API pro vytvoření přístupu k datům v databázi pro klientskou aplikaci. Tímto způsobem klientská aplikace nemá přímý přístup do databáze a může využívat jen předem definované funkce, které webový server přes RESTful API poskytuje. V API jsou konkrétní data přístupné na předem definovaných URL. Přístup je tak realizován pomocí http dotazů z klienta na server. Klient v dotazu musí zvolit specifické URL, kterým definuje nad kterými daty se má operace provést a http metodu dotazu, která určuje typ operace.

U RESTful API je běžné posílat data ve formátech XML nebo JSON.

Data záznamu, která server odešle na požadavek klientovi, nemusí mít stejnou strukturu pod jakou jsou ukládána v databázi. Server tak může připravit data o záznamu do přehlednějšího formátu a na klientovi pak nebude odhalena vnitřní struktura databáze.

2.9.2 Struktura URL v RESTful API

URL se skládá ze segmentů, parametrů a případně i dotazu. Segment, většinou slovní název, určuje o který typ dat se jedná. Parametr je unikátní identifikátor, většinou číslo nebo náhodný textový řetězec, kterým lze vybrat právě jeden konkrétní záznam tohoto typu. Údaje v dotazu většinou slouží jen ke změně v zobrazení výsledku, například změna v jejich seřazení. Dají se však použít i pro jejich filtrování.

Konkrétní URL by pak mohla být vytvořena například v následujícím formátu: webovy.server/firmy/123456/provozovny/1. Podle tohoto URL požadavku webový server pozná, že se jedná právě o provozovnu identifikovanou číslem 1, která patří pod firmu identifikovanou číslem 123456. Tímto způsobem lze jednoznačně určit, pro který záznam je vyžadována akce. Pokud by nebyl uveden poslední parametr identifikující provozovnu, jedná se o kolekci všech provozoven dané firmy.

Běžně se používají tyto http metody:

- GET pro získání dat o záznamu nebo kolekci záznamů
- PUT pro aktualizaci existujícího záznamu
- POST pro vytvoření nového záznamu do kolekce
- DELETE pro smazání záznamu

Metody GET, PUT a DELETE se používají v kombinaci s URL pro konkrétní záznam. Pro URL kolekci se běžně používá jen GET a POST.

K informování o tom, zda zpracování dotazu na serveru proběhlo v pořádku nebo nastal nějaký problém se používají http stavové kódy.

3 Získávání základních dat z veřejných rejstříků a výběr databáze

3.1 Administrativní registr ekonomických subjektů

Jako hlavní zdroj doplňujících informací lze využít administrativní registr ekonomických subjektů ARES. Za využití ARES API lze podle IČO podniku ověřit, zda vůbec takový podnik existuje. Dále jsou pak k dispozici užitečné základní informace o podniku, zejména jeho název, statistická právní forma, datum vzniku, adresa sídla, počet zaměstnanců a klasifikace ekonomických činností NACE. Přes ARES API však lze zjistit nejen informace z registru ekonomických subjektů, ale i z obchodního rejstříku. V něm můžeme získat další informace o podniku, například jeho kapitál.

V rámci rodinných podniků bude převážně zadaná adresa sídla z ARES také jediným místem, kde podnik působí. Lze však předpokládat, že podnik bude mít i více provozoven a na této žádná z nich být nemusí. Pokud by tedy na adrese nebyla provozovna, bude možné pro ni nastavit, aby se nezobrazovala na mapě a místo ní se vytvoří nová provozovna se správnou adresou.

3.2 Geokódování

Abychom mohli provozovny zobrazit na mapě, bude také potřeba znát jejich zeměpisnou délku a šířku. Abychom zjistili tyto koordináty, využijeme k tomu geokódovací API, které nám podle zadané adresy určí koordináty.

3.2.1 Google Geocoding API

Jako první kandidát se jeví Google Geocoding API, jelikož se bude používat také přihlašování s identitou od Google. Nicméně oproti využívání neplacené funkce přihlašování, kde stačí zřídit API klíč, je pro Google Geocoding API nutné zřídit i způsob platby. [11] Google sice dává každý měsíc zdarma kredit k využití těchto služeb a v rámci tohoto systému je nepravděpodobné, že by byl kredit vyčerpán a muselo se tak za API platit, nicméně tomuto riziku se kompletně vyhneme využitím jiného geokódovacího API. Konkrétně Nominatim.

3.2.2 Nominatim

Nominatim je nástroj k vyhledání lokací podle názvu a adresy z dat OpenStreetMap. K využití Nominatim API dokonce není ani potřeba zřídit API klíč. Pokud se podaří nalézt lokaci, můžeme využít mimo koordinát také jejich přehledně rozčleněnou adresu. Stává se totiž, že adresa z ARES není členěna a ještě navíc jsou části zkráceny.

3.3 Výběr databáze

3.3.1 Firebase

Firebase je NoSQL databáze běžící v cloudu. Umí se synchronizovat s klienty v reálném čase a zůstane na klientu dostupná i když zrovna není funkční připojení k internetu. [12] Tyto vlastnosti jsou skvělé obzvláště pro mobilní aplikaci, kde připojení k internetu může být nestabilní. Která data jsou pro uživatele dostupná lze nastavit ve Firebase pomocí pravidel. Může tak mít přístup jen ke konkrétním informacím vedených o podnicích a zároveň pouze výsledky modulů, které sám vyplnil.

Jako NoSQL databáze však strádá, když je potřeba provést komplexní dotaz přes několik sloupců, ve kterém musí být hodnoty v určitém rozmezí. Takové dotazy se ale budou využívat při filtraci podniků na mapě. Firebase tedy pro účely tohoto systému nevyhovuje a vymění se za SQL databázi.

3.3.2 PostgreSQL

Jako náhrada za Firebase bude použita open source databáze PostgreSQL, objektově-relační databáze využívající jazyk SQL. Tato databáze je považována za spolehlivou, rozšiřitelnou a výkonnou. Za její silné vlastnosti lze považovat například robustní sadu funkcí, schopnost uchovávat integritu dat a dostupnost dalších užitečných rozšíření databáze, jako je například PostGIS. PostgreSQL umožňuje využívat indexování, transakce, uložené funkce či procedury. Podporuje full-text vyhledávání a umožňuje nerozlišovat malá a velká písmena. Její zabezpečení umožňuje robustní nastavení přístupu k datům. Lze nastavit přístup pro jednotlivé tabulky nebo dokonce na úrovni sloupců a řádků. PostgreSQL také podporuje vícefaktorové ověřování přístupu. [13]

Tato relační databáze umožní provádět komplexní dotazy které budou pro chod systému potřeba. Umožní nám například snadno najít a zobrazit pouze ty provozovny podniků, které se nachází v oblasti 50 kilometrů od uživatele a přičemž se podnik zároveň zabývá konkrétní ekonomickou činností.

3.3.3 PostGIS

PostGIS je rozšíření pro PostgreSQL, které přidává funkcionalitu práce s geografickými objekty. Lze tak v databázi používat další datové typy jako jsou body, mnohoúhelníky a geometrické kolekce. Obsahuje funkci ST_Distance_Sphere, která vypočítá vzdálenost mezi dvěma souřadnicemi. Tato funkce při výpočtu bere v potaz kulatost Země. Díky této funkci lze poměrně přesně zjistit, které provozovny jsou v daném okruhu.

4 Programování mobilních aplikací

V dnešní době má mobilní zařízení téměř každý. Mobilní zařízení jsou hojně využívány pro jejich všestranné využití a malé rozměry. Oproti stolním počítačům, které jsou obrovské, těžké a jejich ventilátory často hlučné, je takový lehký telefon snadno přenositelný a jeho chod prakticky neslyšný. A přitom výkon mobilního zařízení, ačkoliv značně nižší než u stolního počítače, bohatě postačí k využívání i náročnějších aplikací. Navíc bezdrátové sítě mají v dnešní době takové pokrytí, že připojení k internetu je takřka všude a telefon tak umožňuje být neustále online. I proto jsou mobilní zařízení tak oblíbené. Nicméně dalším z hlavních důvodů oblíbenosti mobilních zařízení je široké spektrum aplikací, které lze na nich využívat. Může to být aplikace nahrazující funkci kalkulačky nebo digitální kalendář. Takto užitečné aplikace pak můžeme mít kdykoliv k dispozici.

Nejpoužívanější mobilní zařízení jsou telefony a také tablety, které jsou o něco větší. Tato zařízení mohou využívat různé operační systémy. Nejoblíbenější jsou však Android od Google a iOS od Apple. Jelikož se aplikace pro oba systémy vyvíjí jiným způsobem, je při vytváření mobilních aplikací nutné předem vybrat pro jaký operační systém bude určena. Oba systémy využívají svoji verzi obchodního centra s aplikacemi, kde je možné stahovat bezplatné i placené aplikace. K uvedení aplikace do těchto center je však nutné splnit různé podmínky. Oproti iOS, Android umožňuje instalaci aplikací, které pochází z jiného zdroje. Pokud se tedy nechceme zabývat s obchodním centrem a mít aplikaci dostupnou například z vlastních webových stránek, zřetelně vyhrává Android.

4.1 Programování aplikací pro systém Android

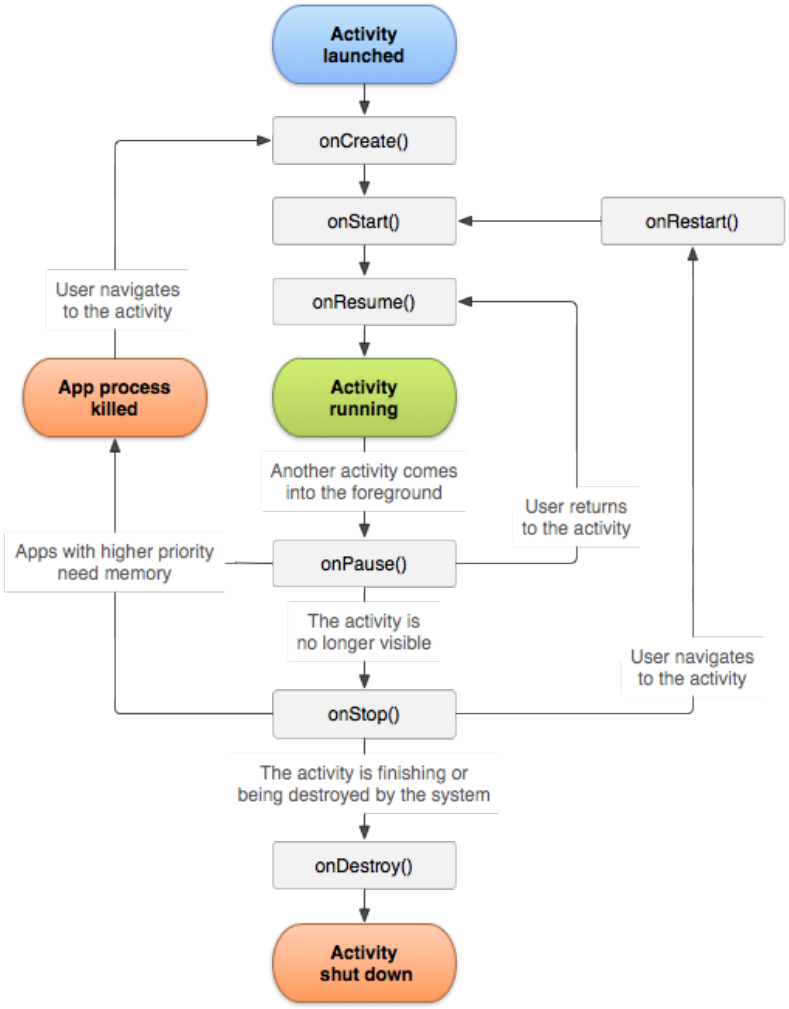
Android je operační systém vyvíjený společností Google pro mobilní zařízení založený na linuxovém jádře. K programování aplikací pro Android je možné využít jazyky Kotlin nebo Java. Kotlin je relativně nový programovací jazyk a je nyní považován za preferovaný jazyk pro vývoj Android aplikací. Java je léty osvědčený jazyk, který se nejprve kompiluje do bajtkódu a následně je spouštěn pomocí interpretru Java virtual machine (JVM), který umožňuje využití stejného zkompilovaného kódu na různých platformách. [14] Jelikož je Kotlin interoperabilní s Javou, lze využívat ve stejném projektu oba jazyky. Při seznamování se s programováním pro Android se tak jeví Java jako lepší volba pro programátora, který už s ní má zkušenosti, jelikož později nebude velký problém případně přejít na Kotlin. Jako hlavní vývojové prostředí se dá považovat Android Studio, postavené nad JetBrains IntelliJ IDEA.

Android Studio využívá balíčkovací nástroj Gradle pro vytváření APK balíčků, které jsou využívány pro distribuci a instalaci Android aplikací. Knihovny, které bude aplikace

využívat, spolu s jejich repozitáři se definují právě v souborech build.gradle. Každá Android aplikace má také svůj AndroidManifest.xml manifest. V něm se definují základní informace o aplikaci. Jeho součástí je definice vstupní aktivity, která se spustí jako první. A také všechny potřebné oprávnění pro funkci aplikace. [15]

4.2 Aktivity

Aplikace se dále dělí na Java kód a zdroje. Pomocí Java kódu jsou tvořeny aktivity, ve kterých je rozdělena logika aplikace. Každá aktivita řeší nějakou dílčí část. Například jedna zobrazí mapu a ovládá její funkčnost, zatímco druhá má na starost zobrazení detailu firmy. Aktivity ovládají grafické rozhraní, které je definováno pomocí xml layoutů.



Obrázek 4.1: Developer.android.com: Životní cyklus aktivity.

Každá aktivita má svůj životní cyklus, který počíná spuštěním a končí ukončením aktivity. Tento cyklus je zde využíván z důvodu limitovaných hardwarových specifikací mobilního zařízení. Primárně jsou to malá operační paměť a pomalý procesor. Při spuštění

aplikace se nejprve spustí hlavní aktivita a vykonají se metody onCreate, onStart a onResume. V momentě, kdy uživatel naviguje na novou aktivitu, je ta původní zastavena metodou onPause a v případě, že nová aktivita zabírá celý displej a z původní tak není nic vidět, se spustí i metoda onStop. Pokud nová aktivita nemá k dispozici dostatek prostředků, může systém Android původní aktivitu úplně zničit a uvolnit tak prostředky pro nynější aktivitu. Pokud byla původní aktivita stále vidět a byla tak pozastavena pouze metodou onPause, při návratu uživatele do této aktivity se spustí metoda onResume. Pokud byla zastavena metodou onStop, spustí se metoda onStart. V případě, že byl systém nucen aktivitu zničit, je celá aktivita znovu spuštěna s novým životním cyklem. Pro uchování pořadí, ve kterém byly aktivity spouštěny je používán activity back stack. V momentě, kdy se nejnovější aktivita ukončí nebo se uživatel bude chtít vrátit zpět si tak systém pamatuje do které aktivity se má vrátit.

4.3 Zdroje aplikace

Zdroje, jako jsou obrázky a texty, které aplikace využívá jsou umístěny ve složce res. V této složce se dál dělí na složky drawable, mipmap, layout, menu a values.

4.3.1 Drawable

Ve složce drawable se ukládají veškeré grafické zdroje. Mohou to být rastrové obrázky typu jpeg a png. Také to mohou být vektorové obrázky nebo definice tvarů pro změnu zobrazení prvků v layoutech ve formátu xml. Rastrové obrázky mohou být vygenerovány i v horší kvalitě pro menší displeje, čímž lze zmenšit celkovou velikost aplikace. Ve složce mipmap se nachází ikony aplikace. I ty mohou být definovány v různých velikostech pro různé velikosti displejů. Oproti ostatním obrázkům však jsou vždy ikony součástí aplikace. Launcher systému si tak může vybrat i lepší rozlišení ikony, než by běžně displej potřeboval. Toho využívá při zvětšeném zobrazení ikonek aplikací.

4.3.2 Layouty

Layouty jsou zdroje ukládáné ve formátu xml, které obsahují rozložení prvků grafického rozhraní. Existují různé druhy layoutů určené pro různá využití. Například LinearLayout se používá pro uspořádání prvků v řadě za sebou, zatímco ConstraintLayout definuje vazby mezi jednotlivými prvky. V těchto layoutech pak lze definovat které další zdroje aplikace se v prvcích využijí a zobrazí. Konkrétně tak v prvku ImageView může být zobrazen specifický obrázek a v prvku TextView zase textová hodnota. Toto přiřazení lze udělat jak v definici layoutu, tak i v kódu aplikace. Pro oddělení uživatelského rozhraní od logiky se však upřednostňuje definice v layoutu, pokud není nutné vybírat mezi použitým zdrojem v prvku za pomoci kódu.

ConstraintLayout je navrhnutý pro vizuální sestavení layoutu v Android Studiu. Oproti ostatním layoutům s ním lze vytvářet stejné rozložení, ke kterému by jinak bylo nutné využít kombinaci několika jiných layoutů. Použitím několika layoutů v sobě se zvyšuje

náročnost na výpočet finálního rozložení prvků a doporučuje se tak používat ConstraintLayout všude, kde je to možné. [16]

4.3.3 Values

Ve zdrojích se dále nachází složka values. Jsou v ní definované hodnoty, které lze využívat jak v layoutech, tak i v kódu aplikace. V souboru arrays.xml se nachází pevně definovaná pole s položkami typu string a integer. V souboru colors.xml se nachází definice barev pomocí modelu rgb v hexadecimální soustavě. V souboru dimens.xml se běžně nachází hodnoty pro velikost textu uváděných v jednotkách SP (Scalable Pixels) a hodnoty pro margin a padding v jednotkách DP (Density-independent Pixels). DP značí velikost nezávislou na rozlišení displeje zařízení. Na každém zařízení tak bude tato hodnota představovat stejnou velikost. Oproti tomu bude text definovaný pomocí SP rozměrově upraven na základě preferencí uživatele.

V souboru strings.xml se nachází textové řetězce. Pro určení textového řetězce jsou používány jednoznačné názvy. Díky tomuto rozčlenění je pak snadné vytvořit nový soubor s modifikátorem v názvu, který určuje lokalizaci a obsahuje překlad do jiného jazyka.

5 Systém pro testování vitality a vizualizaci dat o firmách

Nové řešení systému pro testování vitality rodinných firem by mělo nově umožnit správu testovacích modulů a informací o rodinných firmách přímo z aplikace. Dále by mělo umožnit uchování potřebných informací o rodinných firmách, které budou využívány v mobilní aplikaci. Z mobilní aplikace by pak měl být zřízen chráněný přístup k těmto informacím.

Testovací modul by se v průběhu času mohl změnit. Může se jednat o drobné vylepšení znění některé z otázek, kdy nebude nutné další akce. Může se ale také jednat o kompletní odstranění otázky a nebo rozšíření o nové otázky a bude nutné vytvořit nový modul. Aby se zajistilo, že podnik bude mít stále dostupný výsledek z předešlých testování, bude nutné zachovat i původní modul. Aby všechny tyto funkce s moduly mohly vykonat pověřené osoby, bude vhodné pro ně vytvořit administrační část webových stránek. Zde bude možnost archivovat původní modul a vydat novou verzi. Každou verzi modulu bude mít možnost podnik vyplnit nejvýše jednou.

Moduly budou pro firmy nyní na webových stránkách volně dostupné a automaticky nelze zjistit, jestli modul vyplnil rodinný podnik. Jelikož to bude muset hlídat pověřená osoba, bude administrační část obsahovat také správu firem, které moduly vyplnily. Zde bude možné vymazat vyplněné moduly firem, pro které není tento systém určen. Aby pověřená osoba měla snazší vyhodnotit zda se jedná o rodinný podnik, využije se veřejných rejstříků k získání základních dat o firmě. Firma také může o sobě uvést doplňující informace. Pokud by firma nebyla rodinným podnikem, pověřená osoba vymaže všechna data o firmě a jejích vyplněných modulech a může jí zablokovat přístup do systému i do budoucna.

Rodinné podniky, které využijí tento systém k otestování své vitality, budou moci využít také mobilní aplikaci. Ta je určena především k vzájemnému seznámení rodinných podniků v blízkém okolí. K zobrazení ostatních podniků v blízkosti skvěle poslouží mapa. Na mapě budou přehledně vidět jednotlivé podniky a uživatel si následně bude moci zobrazit detailní informace zadané samotnou firmou i strojově načtená data z veřejně dostupných rejstříků. Dozví se tak ekonomickou činnost dané firmy a potenciálně i adresu webových stránek, což může posloužit jako první krok ke vzájemné spolupráci.

Protože modul může být vyplněn podniky i na papíře, bude potřeba umožnit pověřeným osobám z administrační části vyplnění modulu v aplikaci za podnik. Aby se toto zpětné vyplňování mohlo delegovat i na další osoby, bude žádoucí umožnit vytvoření administračního účtu s omezeným oprávněním pouze na toto vyplňování.

5.1 Požadavky

Podle předcházejícího zadání lze tedy určit tyto specifické požadavky na vytvoření systému:

- Vytvořit webové stránky pro administraci.
  - Zpřístupnit pouze pověřeným osobám z Ekonomické fakulty.
  - Umožnit vytvoření testovacího modulu, dále jeho následnou publikaci a poté archivaci.
  - Umožnit export vyhodnocených modulů.
  - Umožnit správu nad daty podniků v systému.
  - Možnost všechna data o podniku smazat a případně i zamezit budoucímu přístupu do systému.
  - Umožnit nastavení limitovaných oprávnění.
- Vytvořit webové stránky pro rodinné podniky.
  - Při přihlášení do systému uživatel identifikuje svůj podnik pomocí IČO.
  - Do systému načíst informace o podniku z veřejně dostupných zdrojů.
  - Uživatel své údaje ověří a doplní případně další.
  - Umožnit vyplnění modulů a zobrazit jejich vyhodnocení.
- Vytvořit mobilní aplikaci pro rodinné podniky.
  - Umožnit přístup pouze uživatelům, kteří svůj podnik zaregistrovali do systému na webových stránkách.
  - Zobrazit podniky na mapě.
  - Možnost zobrazit podrobné informace o podniku.
  - Umožnit filtrování podniků. Například zobrazit pouze podniky v blízkém okolí nebo s určitou ekonomickou činností.

5.2 Návrh řešení

Administrační část pro správu, stejně jako část pro rodinné podniky, bude řešena formou webových stránek. Obě části tedy mohou být společně realizovány v jedné webové aplikaci. K provozu web serveru bude použit NodeJS, který umožní napsat serverovou část v jazyce JavaScript. Díky tomu lze použít stejný jazyk na serveru jako se bude používat i v klientské části aplikace běžící ve webovém prohlížeči.

Do administrační části bude přístup zabezpečen přihlašovacími údaji jménem a heslem. Budou přítomny dva stupně oprávnění, limitované a neomezené. Uživatelé s neomezeným oprávněním budou moci spravovat celý systém, včetně upravování detailních

informací o podnicích. Uživatelé s limitovaným oprávněním budou moci pouze vyplňovat moduly za podniky.

Jelikož se uživatel za rodinný podnik přihlašuje jak do webových stránek, tak i do mobilní aplikace, bude řešeno jeho přihlašování s pomocí Google identity. Díky tomu nebude muset vymýšlet nové heslo, které by pak také musel přepisovat i na mobilním zařízení. Zároveň již pravděpodobně bude mít tuto identitu na mobilním zařízení spárovanou a nebude tak muset vůbec žádné heslo psát.

Při přihlášení do uživatelské části zadá uživatel IČO svého podniku. Jelikož je IČO unikátní číslo přiřazené podniku, bude se používat k jednoznačné identifikaci daného podniku v tomto systému. IČO bude také stačit k dohledání základních informací z veřejných rejstříků.

Mobilní aplikace bude vytvořena pro systém android a napsána v jazyce Java.

Aby mohl systém správně fungovat, bude nutné využít databázi k trvalému uložení dat. Jelikož bude využito přihlašování Google identitou, naskýtá se možnost zvolit databázi Firebase, která je již připravená na tento způsob přihlašování a lze s ní bezpečně komunikovat přímo z klienta. Jelikož však jako NoSQL databáze není určena pro složité dotazy, pomocí kterých se budou filtrovat výsledky pro mobilní aplikaci, bude použita relační databáze PostgreSQL, která tuto funkcionalitu obstarat dokáže.

5.3 Návrh databáze

5.3.1 Rozbor ukládaných dat

O pověřených osobách, které budou spravovat tento systém se bude uchovávat email, heslo a pravomoce. O samotném uživateli, který se bude se svou firmou přihlašovat do systému, se bude uchovávat jeho email.

O firmách se bude ukládat celá řada informací. Jednak to budou základní informace vedené v ARES, jako jsou IČO, název, právní forma, adresa, datum vzniku, počet zaměstnanců, klasifikace ekonomických činností NACE či kapitál. Ale také doplňující informace, které o své firmě může uživatel sám doplnit. Například adresu webových stránek, facebook, kontaktní telefon či email, nebo také historii podniku, jeho poslání a nabízené služby.

Mimo těchto dat bude také potřeba ukládat informace o jednotlivých provozovnách firem. Jejich adresu, koordináty a otevírací dobu.

Dále se budou ukládat testovací moduly vytvořené pověřenými osobami. U nich bude uveden název, delší popis a stav, který udává jestli mohou uživatelé tento modul vyplňovat. Samotný modul se dále člení na několik dotazníků. Každý dotazník má pak svůj vlastní název a popis. Dotazníky se dále dělí na konkrétní otázky. U otázky je vedeno její textové znění a váha, která určuje jak výrazně bude její odpověď ovlivňovat vyhodnocení za celý dotazník.

Po vyplnění modulu se budou ukládat jednotlivé číselné odpovědi na otázky, procentuální vyhodnocení za každý dotazník a také procentuální vyhodnocení za celý modul.

Pro automatické vyhodnocení jednotlivých dotazníků a následně i modulu bude nutné ke každému z nich ukládat hodnocení. Každé hodnocení má procentuální hranici, podle

této hranice je určeno do kterého hodnocení uživatelův výsledek zapadá. Dále také krátký popis určující stupeň ohodnocení, a následně dlouhý popis, ve kterém je detailně popsáno jakou má uživatel úspěšnost.

Bude také potřeba vést záznamy, které firmy a uživatelé mají trvale zablokovaný přístup do systému.

5.3.2 Schéma

V tabulce firma se budou ukládat informace o firmě. Jako primární klíč je použito IČO firmy. IČO je unikátní identifikátor firmy tvořený číselnou kombinací. V tabulce firma jsou také uloženy cizí klíče pro tabulky pravni_forma a nace. Tabulka pravni_forma má primární klíč tvořen specifickým číselným kódem, dále ukládá název jako text a zda je klasifikována jako právnická osoba pomocí typu boolean. Tabulka pravni_forma má vůči tabulce firma vazbu 1:N. Každá firma má tedy právě jednu právní formu a stejná právní forma může být u více firem.

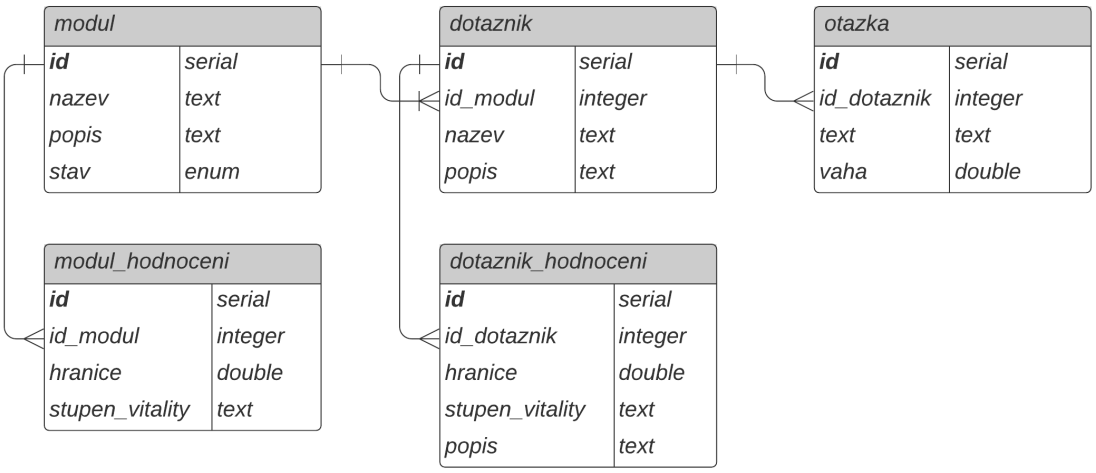
Tabulka nace má primární klíč číselný kód a dále textový atribut pro název. Vůči tabulce firma je zde vazba M:N, totiž stejná NACE může být přiřazena více firmám, ale i firma může mít vícero hodnot NACE. Pro realizaci této vazby je nutné vytvořit další tabulku firma_nace, ve které budou záznamy identifikovány pomocí složeného klíče z primárních klíčů obou tabulek. Zároveň je však v tabulce firma cizí klíč právě jednoho záznamu nace, jelikož se pro účely aplikace považuje první načtená NACE z ARES jako hlavní NACE firmy. Podle právě této NACE je pak na mapě firma zobrazována pomocí specifické ikonky.

V tabulce provozovna budou uloženy záznamy o provozovnách firem, jako primární klíč automaticky inkrementované číslo id. Firma může mít provozoven více, ale každá provozovna patří právě jedné firmě. Proto je u záznamu provozovny veden primární klíč firmy. V tabulce provozovna se dále ukládá adresa a koordináty. Koordináty jsou uloženy jako speciální typ pro PostGIS jako geometrický bod.

O provozovnách se také ukládá jejich otevírací doba. Aby bylo možné definovat různé otevírací doby pro každý den a zároveň mít možnost během otevírací doby dne i několik pauz, ukládá se otevírací doba do tabulky oteviraci_doba. Primární klíč je automaticky inkrementované číslo id, také je přítomen cizí klíč ukazující na provozovnu, které otevírací doba patří. Dále je uložen čas otevření cas_od a čas zavření cas_do, typu time a konkrétní den jako typ enum(’po’, ’út’, ’st’, ’čt’, ’pá’, ’so’, ’ne’). Tímto způsobem lze pro stejný den definovat libovolný počet otevíracích dob a není tak limitován ani počet přestávek.

Jednotlivé testovací moduly jsou ukládány v tabulce modul s primárním klíčem id. Ukládá se o nich jejich název a detailní popis ve sloupcích nazev a popis jako typ text a stav jako typ enum(’editovatelne’, ’publikovano’, ’archivovano’). Každý modul má více hodnocení. Jednotlivá hodnocení pro modul jsou ukládána v tabulce modul_hodnoceni, kde primární klíč id je typu serial, id_modul je cizí klíč pro modul typu integer, kterému hodnocení patří, hranice je typu double a stupen_vitality je typu text. Modul je dále skládán z jednoho nebo více dotazníků. Dotazníky jsou uchovány v tabulce dotaznik, kde primární klíč id je typu serial, id_modul je cizí klíč pro modul typu integer a nazev a popis jsou typu text. Každý dotazník pak má několik hodnocení, které se ukládají v tabulce dotaznik_hodnoceni. Obdobně jako hodnocení pro modul se u hodnocení pro dotazník

ukládá pod stejnými typy id, hranice a stupen_vitality. Dále se ukládá cizí klíč typu integer pro dotazník, kterému hodnocení patří a popis typu text. Každý dotazník se pak skládá z několika otázek. Otázky jsou ukládány v tabulce otazka, kde id je primární klíč typu serial, id_dotaznik je cizí klíč pro dotazník, ke kterému otázka patří, text otázky je typu text a váha je typu double.



Obrázek 5.1: ER diagram testovacích modulů.

Pro zablokování uživatelů a firem jsou vytvořeny tabulky blacklist_firma a blacklist_uzivatel. Tabulka blacklist_uzivatel si uloží záznam o zablokovaném uživateli pod stejným primárním klíčem jako je veden v tabulce uzivatel. Tímto způsobem lze zajistit zablokování přístupu jak pro stávajícího uživatele, který ještě všechna data v systému má, tak i pro znemožnění nové registrace pro uživatele, jehož všechna data byla smazána. Obdobně je to tak i pro firmu.

6 Webová aplikace

Webová aplikace se skládá ze serverové a klientské části. Klientská část je tvořena formou webových stránek tvořených pomocí HTML, CSS, JavaScript a frameworku Bootstrap pro vytvoření responzivního zobrazení webu.

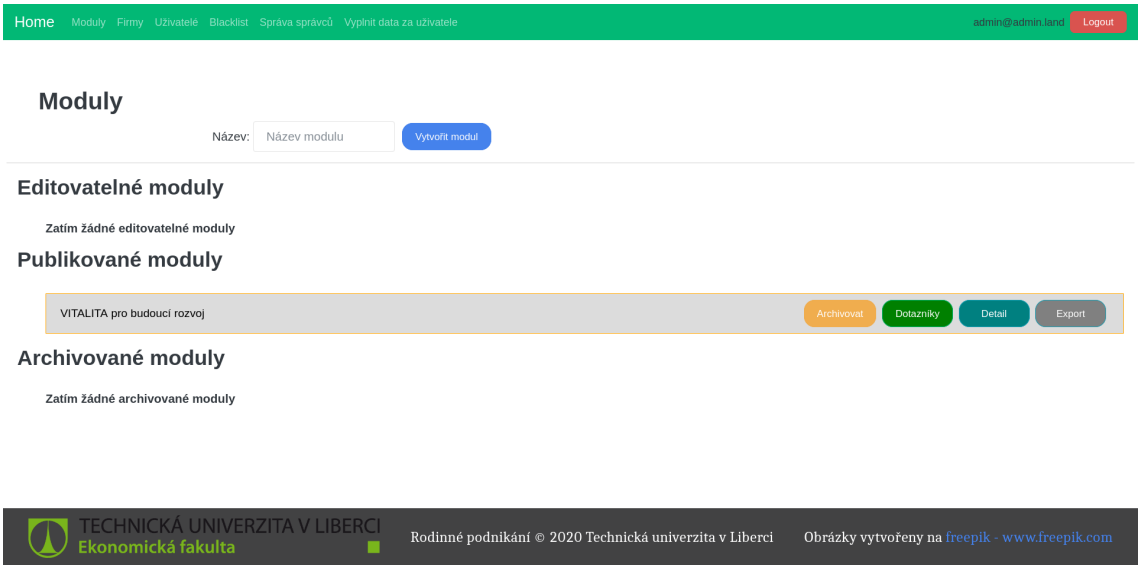
Na serveru je použit NodeJS web server psaný v jazyce JavaScript. Veškeré moduly, které jsou používány jsou vedeny v souboru package.json. Dále je využíván minimalistický framework Express. Jako view template engine je použit Pug. Struktura souborů je rozdělena podle MVC architektury. Ve složce models se nachází definice dat z databáze. Ve views jsou uloženy šablony pro html stránky. Ve složce controllers se nachází logika, která ovládá jak se budou data měnit na základě vstupu od uživatele. Ve složce routes jsou definovány URL, napojené na konkrétní kontroler.

Pro identifikaci uživatele webových stránek se využívá modulu ExpressSession, který ukládá Session do databáze a při requestu v middleware přibalí informace o uživateli z existující Session do objektu request.

Pro spojení s PostgreSQL databází je využit ORM Sequelize. Pro Sequelize jsou nadefinovány modely, které reprezentují data uložené v tabulkách databáze a jejich vzájemné relace.

6.1 Administrační část

Celá administrační část webu je oddělena přidáním do cesty URL /admin. Přístup k webovým stránkám je chráněn pomocí uživatelského jména a hesla. K hashování a ověření správnosti hesla je použit bcrypt, což je knihovna pro hashování hesel. [17] Po přihlášení se uživatel dostane na dashboard, kde se nachází uvítací stránka, odkud se může dostat do zbylých částí aplikace.

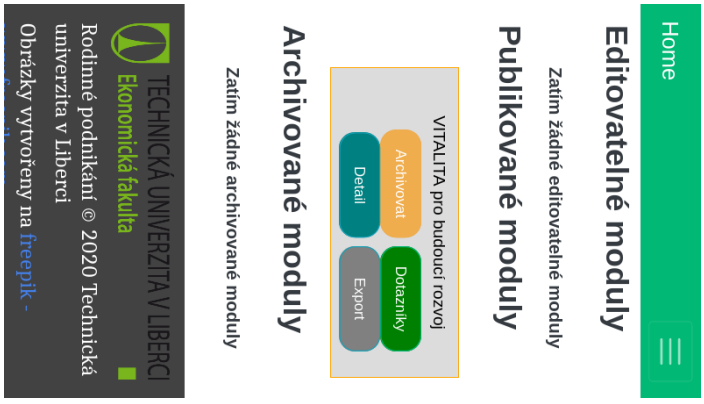


Obrázek 6.1: Moduly.

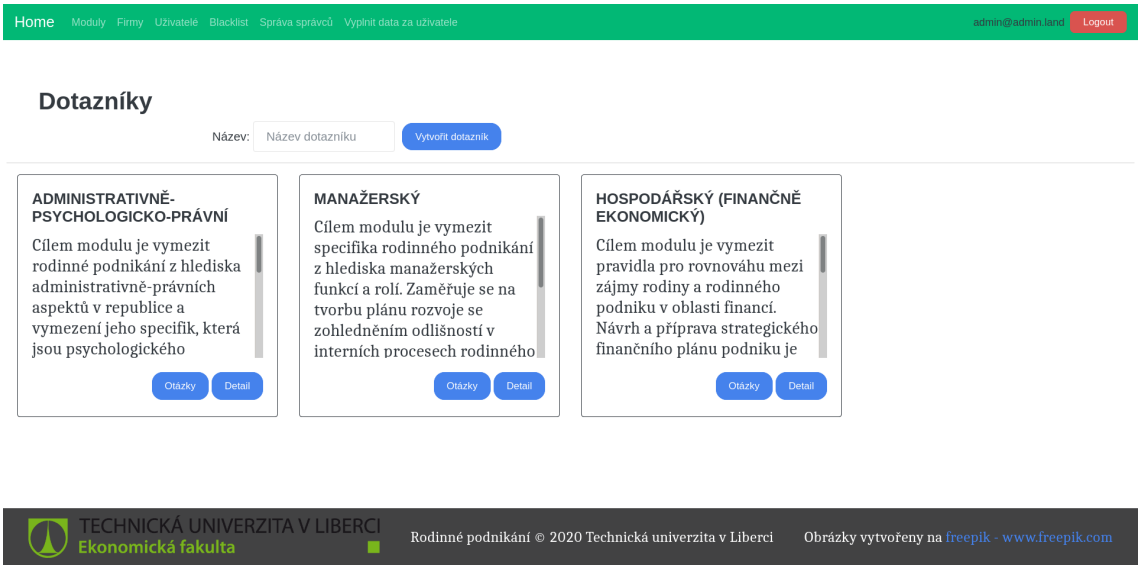
6.1.1 Moduly

Na stránce /admin/moduly se nachází správa modulů. Zde lze vytvářet nové moduly. Při vytvoření nového modulu je jeho stav nejprve považován za editovatelný. Dokud je v tomto stavu, nezobrazuje se uživatelům k vyplnění a lze jej libovolně měnit a případně i smazat. Ve chvíli, kdy se rozhodne že je modul připraven k vyplňování, lze modul publikovat.

Publikovaný modul již bude vidět v uživatelské části a uživatelé ho budou moci vyplnit. U publikovaných modulů je stále umožněno provádět změny, aby se mohlo opravit případné překlepy. Nicméně už by nemělo být zasahováno do struktury modulů. Poté co uplyne doba, kdy má být modul přístupný uživatelům, lze modul archivovat. Po archivaci už jej nemohou uživatelé vyplňovat. Pro publikované a archivované moduly je možnost exportu. Exportem se vytvoří excelová tabulka ve formátu xlsx, ve které je vidět kdo a jak modul vyplňoval. K tvorbě xlsx souboru je využit modul zvaný xlsx.

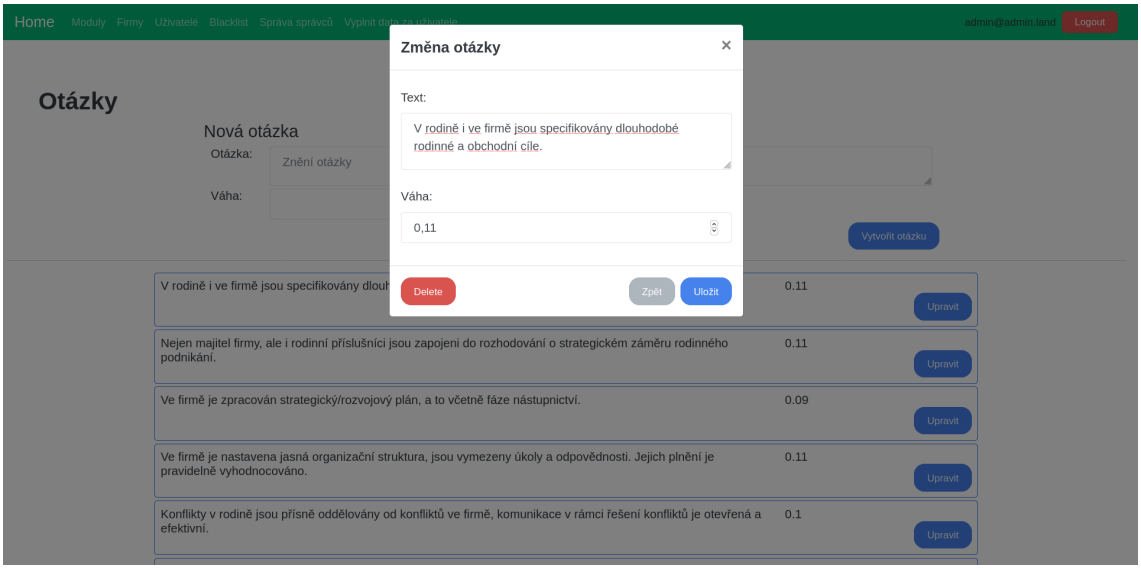


Obrázek 6.2: Responzivní zobrazení modulů na malém displeji.



Obrázek 6.3: Dotazníky.

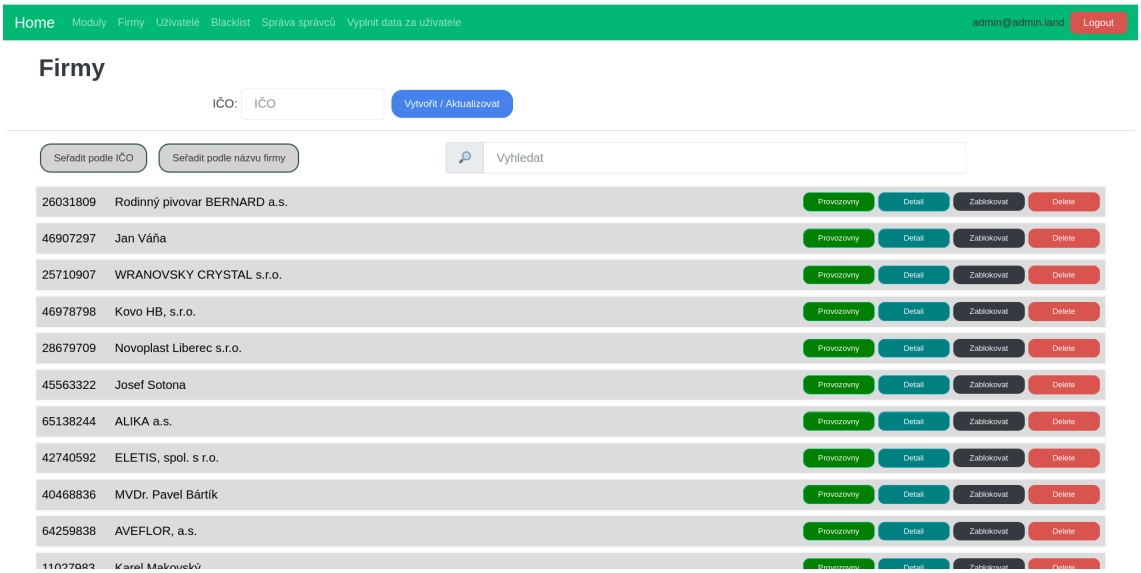
Editovatelným modulům lze upravovat jejich základní informace, přidávat stupně hodnocení a vytvářet jednotlivé dotazníky. Každý dotazník pak lze také upravovat a přidat stupně hodnocení. Pro dotazník se také vytváří otázky, které jsou tvořeny textem a vahou.



Obrázek 6.4: Otázky.

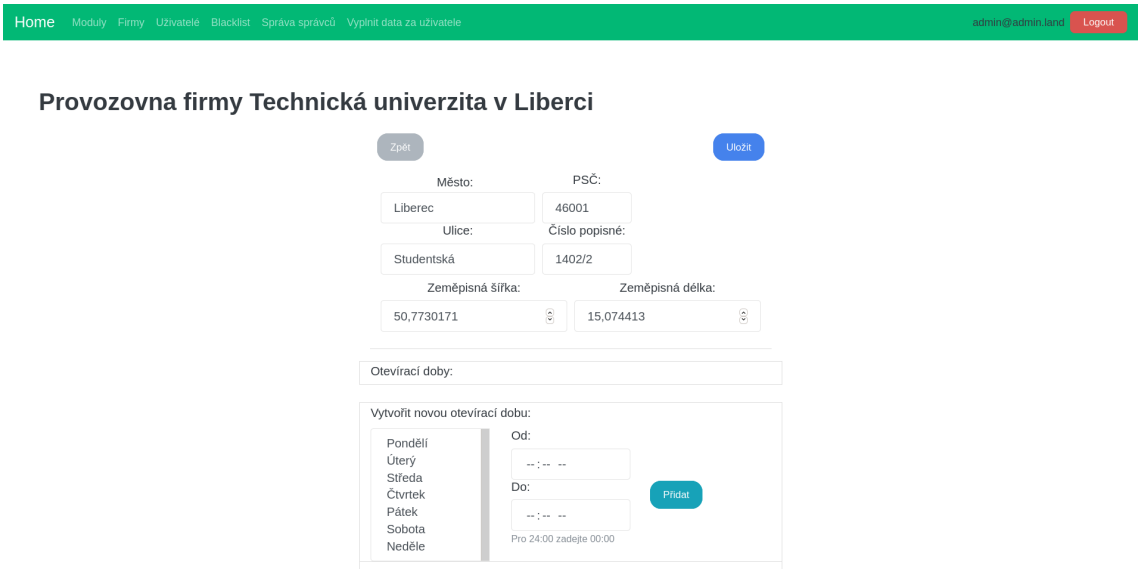
6.1.2 Firmy

Stránka /admin/firmy obsahuje seznam firem vedených v systému. Firmu lze manuálně vytvořit bez uživatele. Firmy lze seřadit podle IČO a názvu. Také lze vyhledat konkrétní hodnotu. K vyhledávání, seřazení a stránkování je využita knihovna List.js.



Obrázek 6.5: Firmy.

Firmu lze smazat kompletně se všemi jejími údaji v systému. Také ji lze zablokovat, aby po smazání již nešla znovu vytvořit. O firmě jsou vedeny mnohé informace. Strojově načtená data z ARES nelze měnit, upravit lze pouze ostaní informace. Firmě lze také vytvářet provozovny, které se zobrazují na mapě v mobilní aplikaci. Ze základu je vytvořena jedna provozovna podle adresy z ARES. Při vytvoření provozovny se využije Nominatim pro zjištění zeměpisné šířky a délky, která je využívána pro přesné umístění na mapě. Provozovně lze také nastavit otevírací dobu. Pro pohodlné nastavení otevírací doby je využit klientský JavaScript, kde se před odesláním na server zeleně zobrazí nově přidávané otevírací doby a červeně ty, které jsou určeny pro smazání. Tímto způsobem tak lze vidět jak bude výsledná otevírací doba vypadat ještě před tím, než je uložena. Vzhledem k tomu, že se otevírací doba často opakuje po několik dnů, lze vybrat všechny takové dny najednou a přidávat dobu všem s využitím tagu select s atributem multiple.



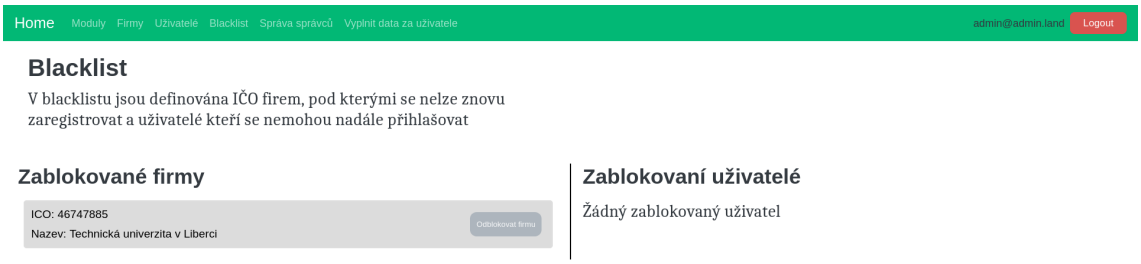
Obrázek 6.6: Provozovna.

6.1.3 Uživatelé

Na stránce /admin/uzivatele jsou zobrazeni všichni uživatelé vedení v systému. Uživatele lze seřadit podle emailu, a také podle IČO a názvu jejich firmy. Lze také přejít na detail firmy uživatele, zablokovat firmu nebo uživatele a smazat samotného uživatele.

6.1.4 Blacklist

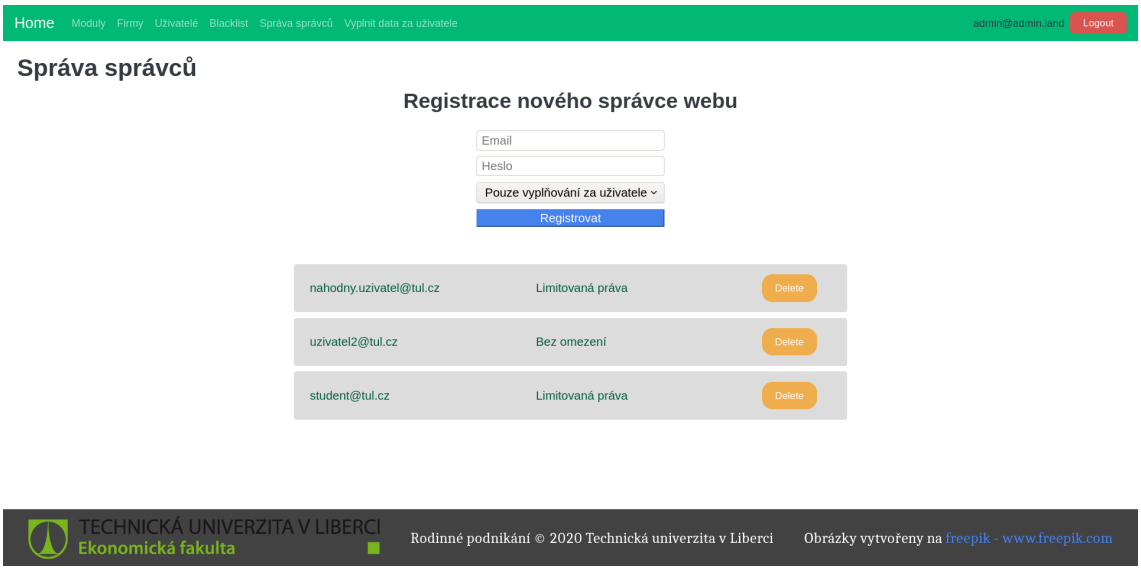
Stránka /admin/blacklist obsahuje seznamy všech zablokovaných uživatelů a firem v systému. Zablokovanou firmu nelze znovu vytvořit a zablokovaný uživatel se už ani znovu nepřihlásí. Na této stránce lze zablokované položky odblokovat.



Obrázek 6.7: Blacklist.

6.1.5 Správa správců

Na stránce /admin/admins jsou uživatelé administračních stránek. Lze zde vytvořit nového uživatele s neomezeným oprávněním, nebo omezeným pouze pro vyplňování dat za uživatele.



Obrázek 6.8: Správa administrátorů.

6.1.6 Vyplnit data za uživatele

Na této stránce /admin/login_as_uzivatel se lze přihlásit do uživatelské části pro firmy za konkrétního uživatele a vyplnit tak za něho modul. Tato možnost je zde přidána z toho důvodu, že firma může vyplnit modul v papírové podobě a do systému ho pak musí zavést někdo z administrace.

6.2 Uživatelská část

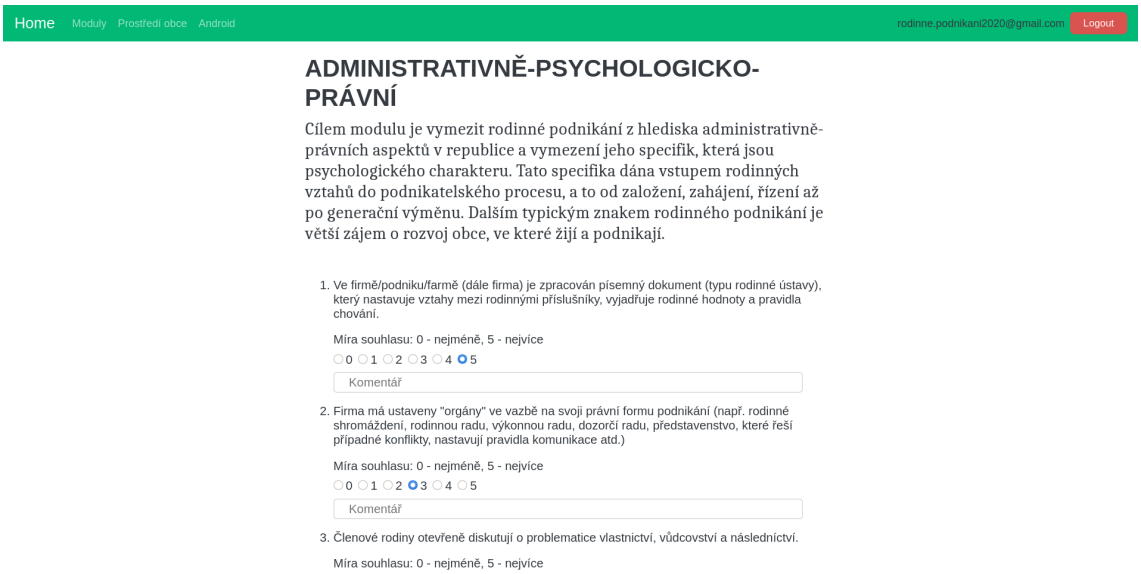
Do uživatelské části se přihlašuje pomocí Google SignIn. Na přihlašovací stránce je umožněno využít identitu od Google, kde si uživatel zvolí, který účet chce použít. Do tohoto účtu se přihlásí svými přihlašovacími údaji a pokud jsou správné, Google vygeneruje přístupový token uživatele pro aplikaci. Tento token se následně odešle na server jako náhrada jména a hesla. Na serveru je token zvalidován, aby se ověřila jeho platnost. Pokud je token platný, lze o uživateli zjistit základní informace, jako jsou email a profilový obrázek. Pokud se uživatele podařilo pomocí Google ověřit, je uživatel úspěšně přihlášen a vytvoří se pro něj na serveru Session.

Po přihlášení se uživatel dostane na dashboard, na kterém si může stáhnout jednu z případových studií firem, která byla vytvořena na Ekonomické fakultě, ve formátu pdf. Z této stránky lze také navigovat do zbylých částí aplikace.

6.2.1 Moduly

Na této stránce /moduly jsou vidět všechny moduly dostupné k vyplnění a moduly, které již uživatel vyplnil.

Dostupné moduly může uživatel vyplnit. Při spuštění modulu se uživateli nejprve zobrazí informace o modulu a z jakých dotazníků se modul skládá. Dále už vyplňuje dotazníky. Každý dotazník se vyplňuje zvlášť a na jednotlivé otázky lze odpovědět číselnou mírou souhlasu v hodnotách 0 (nejméně) až 5 (nejvíce). Každou odpověď také může volitelně okomentovat. Po vyplnění se dotazník vyhodnotí. K tomu je využito Fetch API, pomocí kterého se na server odpovědi odešlou a server po zpracování vypočte procentuální úspěšnost a odešle celkový výsledek za dotazník, který se uživateli následně zobrazí. Tento výsledek je právě daný stupeň hodnocení, pod který uživatelova procentuální úspěšnost nejblíže spadá. Tímto způsobem se postupně vyhodnotí každý dotazník modulu a nakonec se vypočte celkový výsledek za celý modul. U vyplněných modulů si uživatel může stáhnout jeho výsledek vygenerovaný ve formátu pdf pomocí modulu pdfmake.



Obrázek 6.9: Vyplňování dotazníku modulu.

6.2.2 Android

Na stránce /android je pro uživatele k dispozici ke stažení instalační balíček pro Android aplikaci. Tuto aplikaci může využívat každý uživatel zaregistrovaný v systému.

Obrázek 6.10: Stahování Android aplikace.

6.3 Implementace REST API

Jelikož už byl zprovozněn webový server pro realizaci webových stránek, RESTful API tak může fungovat na tomto webovém serveru. Aby bylo API odděleno od zbytku webové aplikace, všechny její URL začínají segmentem /api.

6.3.1 Zabezpečení s OAuth

Aby toto API mohli používat jen uživatelé tohoto systému, je nutné povolit k němu přístup jen pro ověřené uživatele. Jelikož je REST bezstavový, bude vhodné použít také k zabezpečení bezstavový protokol.

K tomu se využije bezstavového protokolu OAuth k ověřování identity za pomocí tokenů, konkrétně JWT (JSON Web Token).

6.3.2 Ověřování identity

Po přihlášení na klientovi přes Google se odešle získaný token na server. Token se zde dekóduje a ověří se, že daný uživatel je skutečně součástí tohoto systému. Pokud ověření uživatele proběhlo v pořádku, postupuje se podle protokolu OAuth. Pro uživatele se vygenerují tokeny pro přístup a obnovení. K tomu se používá JWT. Do těchto tokenů se zakóduje datum expirace. Token pro obnovení se navíc uloží do databáze a oba tokeny se odešlou zpět klientovi. Nyní může klient využívat přístupový token k přistupování do API dokud nevyprší jeho platnost, poté musí klient zažádat o nový přístupový token s pomocí obnovovacího tokenu. V případě, že by i obnovovacímu tokenu vypršela platnost, bude se muset uživatel na klientu znovu přihlásit přes Google.

6.3.3 Vytvoření endpointů pro potřebné URL

Přes API je nutné zpřístupnit detalní informace o firmách a jejich provozovnách. Při zobrazení mnoha firem na mapě však není nutné hned načítat detailní data všech firem a postačí o nich jen základní informace.

URL potřebné pro mobilní aplikaci vypadají takto:

- POST /api/login pro ověření tokenu od Google a vygenerování nové sady JWT tokenů
- POST /api/refresh pro ověření obnovovacího tokenu a vydání nového přístupového tokenu
- GET /api/firmy pro základní informace o firmách a jejich provozovnách k zobrazení na mapě
- GET /api/firmy/:id pro detailní informace o konkrétní firmě

Všechna data jsou posílány ve formátu JSON, jelikož se na serveru používá Javascript a nemusí se tak nijak dál měnit.

6.4 Hosting

Webový server běží na adrese rp.tul.cz. Server využívá operační systém Windows 10. K webovému serveru běžícímu v NodeJS se doporučuje používat Nginx jako reverse proxy. [18] Nginx má řadu funkcí vyspělého webového serveru a je mnohem snazší na něm docílit požadovaného nastavení. Je velmi snadné nastavit pro přenos dat šifrování s https nebo využívat http verzi 2. Nginx také kešuje obsah který se nemění a zvyšuje tak rychlost načítání určitých částí webových stránek.

Aby po případném restartu systému byl zajištěn co nejrychleji chod webserveru, je v plánovači úloh nastaveno automatické spuštění procesů po startu systému pro Nginx a NodeJS aplikaci.

Zároveň je na serveru spuštěna i databáze. Tím, že běží na stejném serveru není třeba šifrovat spojení mezi webserverem a databází. Přenos dat mezi databází a webserverem je tak velmi rychlý. Přístup po síti do databáze je pro bezpečnost zablokovaný přes firewall.

7 Mobilní aplikace

Vizualizace je tvořena formou mobilní aplikace pro operační systém Android. K zobrazení je použita mapa od Google za využití Maps SDK pro Android. Je umožněna filtrace dat na základě sekcí kódů NACE, klasifikace ekonomických činností, ve 3 úrovních. Dále je možné vybrat město, nastavit vzdálenost od pozice uživatele a vybrat pouze provozovny, které mají právě otevřeno. Také je možnost zobrazit detailní informace každé firmy a provozovny.

Pro připojování k API serveru je využito Google SignIn a OkHttp, http klienta pro Android. [19] Přihlašování uživatele funguje podobně jako na webových stránkách. Uživatel se přihlásí do svého účtu s Google identitou. Google výměnou za přihlášení vygeneruje nový token přihlášeného uživatele pro aplikaci. Tento token je pak přibalen do requestu vytvořeném s OkHttp na server, kde proběhne ověření uživatele. Token se nejprve zvaliduje, čímž se ověří identita uživatele. Po validaci tokenu lze zjistit uživatelův email, který je dále vyhledán v databázi systému. Pokud je uživatel s tímto emailem zároveň již evidován v systému, znamená to, že uživatel má přístup do mobilní aplikace a vytvoří se pro něho přístup vygenerováním JWT tokenů. Při dalších requestech na server tak bude využíván přístupový token, dokud nevyprší jeho platnost. Poté se využije obnovovací token, se kterým se z mobilní aplikace zažádá o vygenerování nového přístupového tokenu. Data jsou ze serveru posílány ve formátu JSON. Pro převádění JSON objektů do Java objektů je využita knihovna Gson.

7.1 Úvod do aplikace

Uživatelé jsou nejdříve seznámeni s aplikací. K tomu je využita knihovna AppIntro, která pomáhá vytvořit dobře vypadající úvodní část aplikace. [20] Poté se přihlásí s účtem od Google, kterým se přihlašovali na webových stránkách systému.

7.2 Hlavní menu

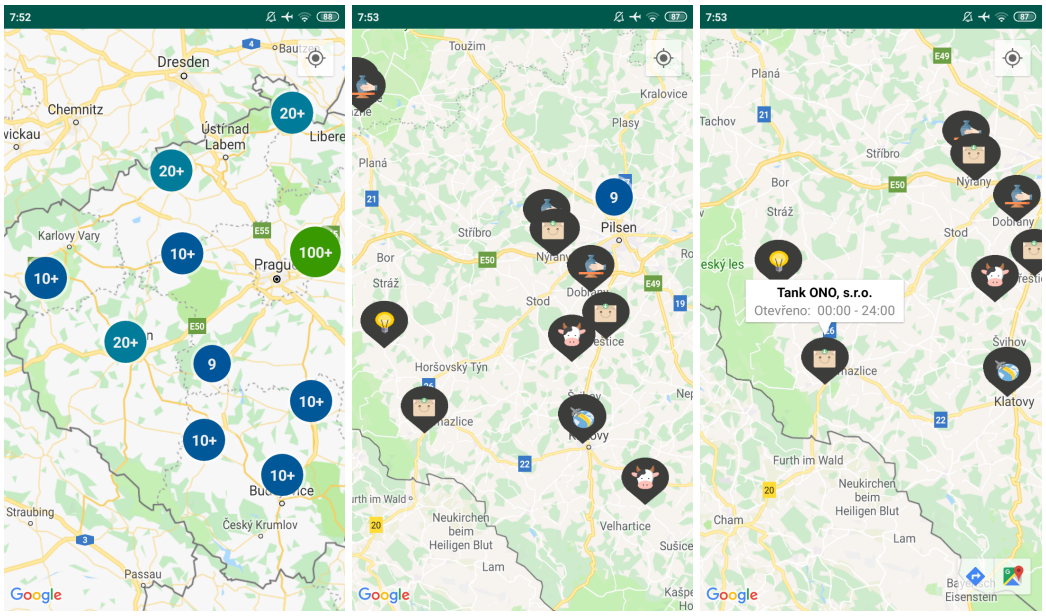
Pokud jsou uživatelé vedeni v systému, zpřístupní se jim hlavní menu. Z hlavního menu si mohou vybrat, jestli chtějí zobrazit všechny provozovny nebo je vyfiltrovat, také si mohou zobrazit profil své firmy.

7.3 Filtrace

Pokud uživatel zvolí zobrazení s využitím filtrace, spustí se aktivita pro filtraci. Zde může nastavit ekonomickou činnost firem pomocí výběru kódů NACE. Ekonomickou činnost lze vybírat ve třech úrovních, což umožňuje vyhledávat od nejvyšší kategorie ekonomické činnosti, dále jejími podkategoriemi a nakonec i konkrétním kódem pro nejnižší dělení ekonomické činnosti. Dále je možnost definovat, ve kterém městě se má provozovna nacházet, jak blízko musí provozovna být vůči lokaci uživatele a jestli má provozovna právě otevřeno.

7.4 Provozovny na mapě

Provozovny se na mapě zobrazují s rozdílnou ikonou podle hlavní ekonomické činnosti firmy. Pokud je příliš velké množství provozoven blízko sebe, je obtížné vybrat z nich tu požadovanou. Proto je využita třída ClusterManager, která provozovny v takovém případě zobrazí ve shlucích. Kliknutím na shluk se pak přiblíží pohled na provozovny uvnitř shluku.



(a) Oddálená mapa. (b) Přiblížená mapa. (c) Vybraná provozovna.

Obrázek 7.1: Zobrazení provozoven na mapě.

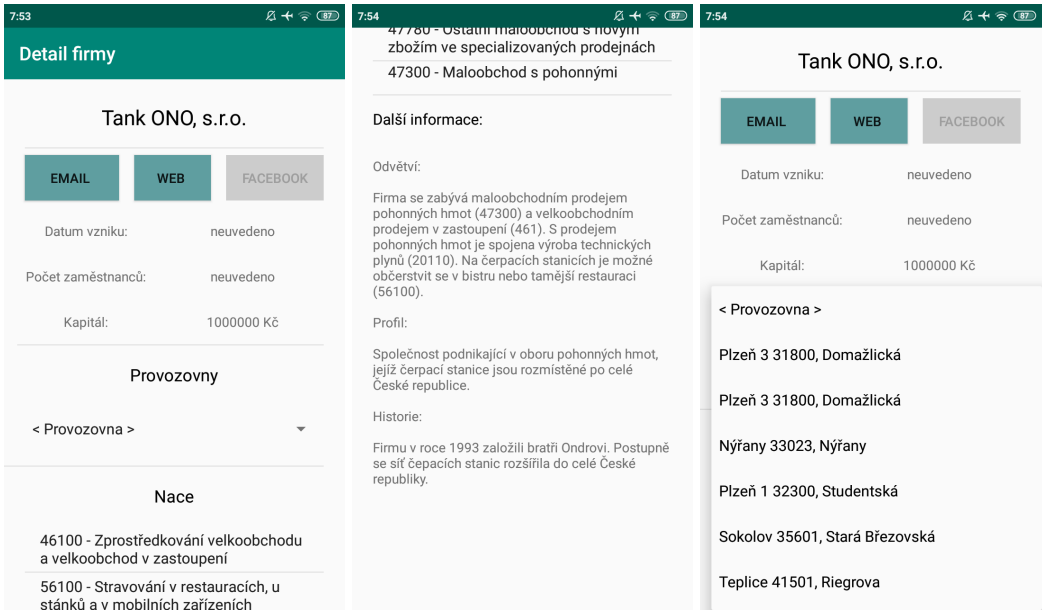
7.5 Detail provozovny

Na detailu provozovny se uživateli zobrazí její adresa a otevírací doba. Také je zde tlačítko pro zahájení navigace na lokaci provozovny pomocí aplikace Google Maps a tlačítko pro zobrazení detailních informací o firmě.

7.6 Detail firmy

V detailu firmy se zobrazí základní informace o firmě. Pokud má firma nastavený email, web nebo facebook, může uživatel použít tlačítka pro využití těchto informací. V případě, že tyto informace nejsou známy, tlačítka jsou deaktivována. Při kliknutí na tlačítko EMAIL se Android pokusí spustit emailový klient uživatele s vyplněným příjemcem. Pokud je definován web nebo facebook, aplikace se při kliknutí pokusí spustit webový prohlížeč uživatele s vyplněnou adresou.

Dále může uživatel vybrat jednu z provozoven firmy, tím se spustí detail pro danou provozovnu. Také jsou uvedeny všechny ekonomické činnosti firmy a v případě, že je uživatel pro firmu vyplnil, jsou k dispozici i doplňující informace o firmě, jako je popis vytvářeného produktu nebo vize firmy.



(a) Detail firmy. (b) Další informace. (c) Seznam provozoven.

Obrázek 7.2: Detail firmy.

8 Závěr

Cílem této práce bylo vytvořit systém, který umožňuje pověřeným osobám z Ekonomické fakulty vytvářet testovací moduly pro testování vitality rodinné firmy a správu jejich dat. Rodinným firmám má tento systém umožnit vyplnění testovacích modulů a vyhodnotit tak nejen vitalitu svojí firmy. Zároveň má být vytvořena i mobilní aplikace určená pro rodinné firmy, které se otestovaly v systému. Aplikace má umožnit seznámit se s ostatními otestovanými rodinnými firmami a usnadnit navázání kontaktu. V této práci byla vytvořena mobilní aplikace pro operační systém Android ve vývojovém prostředí Android Studio. Aplikaci mohou používat pouze rodinné firmy registrované v systému. V této aplikaci lze zobrazit ostatní rodinné firmy jak na mapě, tak i formou seznamu. Výsledky k zobrazení lze vyfiltrovat pomocí NACE, vzdálenosti od uživatele nebo zda je momentálně v době jejich otevírací doby. Na mapě lze spustit navigaci přes aplikaci Google Maps na vybranou provozovnu. Z detailu firmy se pak lze dostat na uvedené webové stránky přes webový prohlížeč nebo spustit emailový klient pro napsání zprávy na kontaktní email firmy. Aby měla Android aplikace zabezpečený přístup k datům z databáze, bylo vytvořeno RESTful API, které odkrývá jen potřebná data pro chod Android aplikace. Data jsou ukládána v relační databázi PostgreSQL. To se dále rozšířilo o nadstavbu databáze PostGIS. Za pomocí metody této nadstavby ST_Distance_Sphere byla zajištěna rychlá a přesná funkcionalita výpočtu vzdálenosti provozovny od lokace uživatele přímo v databázi. Byla vytvořena webová aplikace za použití technologií NodeJS a frameworku Express na serveru a frameworku Bootstrap na klientu. Webová aplikace slouží jak pro administraci, tak i pro běžného uživatele.

V administrační části lze vytvářet nové testovací moduly. Pro každý modul je možné vytvořit libovolný počet dotazníků a pro každý dotazník libovolný počet otázek. Testovací moduly je také možné publikovat a archivovat pro určení dostupnosti pro uživatele. Také je možnost upravovat data o firmách a jejích provozovnách. Dále je umožněno vytvářet nové uživatele administrační části. Ti mohou mít pouze limitované oprávnění, aby mohli pouze vyplňovat data do testovacích modulů za uživatele. V uživatelské části se lze zaregistrovat s pomocí platného IČO rodinné firmy, které je ověřováno v informačním systému ARES. Lze doplnit další informace, které mohou být zobrazeny v Android aplikaci ostatními uživateli. Dále umožňuje vyplnit publikované testovací moduly a otestovat si tak nejen svoji vitalitu. Stanovené cíle tedy vytvořený systém splňuje. Vytvořené RESTful API bylo pro potřebu navrženo pouze pro získávání dat z databáze. Do budoucna by se tedy mohl systém rozšířit o funkci přidávání dat o firmě jejím vlastníkem i přímo z mobilní aplikace.

Seznam odborné literatury

[1] ATHA, Hiral. Web Application Development [online]. [vid. 13. 12. 2020]. Dostupné z: https://www.moveoapps.com/blog/web-application-development-guide/

[2] Single-page application vs. multiple-page application [online]. Neoteric [vid. 15. 12. 2020]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[3] Server-side web frameworks [online]. Mozilla [vid. 16. 12. 2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

[4] Introduction to Node.js [online]. Nodejs.dev [vid. 18. 1. 2021]. Dostupné z: https://nodejs.dev/learn

[5] Express/Node introduction [online]. Mozilla [vid. 25. 1. 2021]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[6] MVC [online]. Mozilla [vid. 25. 1. 2021]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[7] HOYOS, Mario. What is an ORM and Why You Should Use it [online]. [vid. 5. 1. 2021]. Dostupné z: https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a?gi=578540c53f5

[8] Bootstrap Tutorial [online]. TutorialRepublic [vid. 26. 1. 2021]. Dostupné z: https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/

[9] SHERRY, Hsu. Session vs Token Based Authentication [online]. [vid. 28. 1. 2021]. Dostupné z: https://medium.com/@sherryhsu/session-vs-token-based-authentication-11a6c5ac45e4

[10] TEA, Matt. A Massive Guide to Building a RESTful API for Your Mobile App [online]. [vid. 29. 1. 2021]. Dostupné z: https://savvyapps.com/blog/how-to-build-restful-api-mobile-app

[11] Geocoding API [online]. Google [vid. 5. 2. 2021]. Dostupné z: https://developers.google.com/maps/documentation/geocoding/start

[12] Cloud Firestore [online]. Google [vid. 12. 2. 2021]. Dostupné z: https://firebase.google.com/docs/firestore

[13] CONNOLLY, Thomas M. a Carolyn E. BEG, 2015. Database systems: a practical approach to design, implementation, and management. Sixth edition. Boston: Pearson. ISBN 978-0132943260.

[14] HEROUT, Pavel. Učebnice jazyka Java. 3., rozš. vyd. České Budějovice: Kopp, 2007. ISBN 978-8072323234.

[15] GRIFFITHS, Dawn a David GRIFFITHS. Head First Android Development: A Brain-Friendly Guide. 2nd edition. Sebastopol, CA: O'Reilly Media, 2017. Head First series. ISBN 978-1491974049.

[16] ConstraintLayout [online]. Android for developers [vid. 18. 2. 2021]. Dostupné z: https://developer.android.com/training/constraint-layout

[17] Bcrypt [online]. npmjs [vid. 25. 2. 2021]. Dostupné z: https://www.npmjs.com/package/bcrypt

[18] SHANI, J.H. Using NGINX to Serve .NET Core, Nodejs, or Static Contents [online]. [vid. 14. 3. 2021]. Dostupné z: https://dzone.com/articles/using-nginx-to-serve-net-core-nodejs-or-static-con

[19] OkHttp [online]. GitHub [vid. 2. 3. 2021]. Dostupné z: https://github.com/square/okhttp

[20] AppIntro [online]. GitHub [vid. 8. 3. 2021]. Dostupné z: https://github.com/AppIntro/AppIntro

Přílohy

Obsah přiloženého CD

- Text bakalářské práce
  - bakalarska_prace_2021_Jan_Pilar.pdf
- Zdrojový kód programu
  - Webová aplikace (v programovacím jazyce Javascript)
  - Mobilní aplikace pro Android (v programovacím jazyce Java)