FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Aplikace pro verifikaci zpráv
Student:	Aydar Mannanov
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout webovou aplikaci pro ověřování pravosti zpráv, které se objeví na sociální sítích např. Twitter. Aplikace musí být schopna stáhnout potřebné zprávy ze sociální sítě a umožnit uživatelům ohodnotit jejich pravdivost. Zaměřte se především na samotný návrh řešení a potřebných procesů pro dosažení věrohodného ověření pravdivosti zpráv. Zaměřte se i na různé možnosti ověření zprávy a důvěryhodnosti hodnotitele (lokace, počet ověření, externí data atd.). Při řešení se stačí omezit na konkrétní oblast např. ČR.

Postupujte dle následujících kroků:

1) Analyzujte existující řešení
2) Navrhněte a vymodelujte potřebné procesy
3) Vyberte vhodné technologie a navrhněte aplikaci
4) Implementujte prototyp aplikace
5) Proveďte uživatelské testování prototypu, včetně vyhodnocení úspěšnosti ověřování
6) Zhodnoťte předchozí body a navrhněte další možnosti rozvoje výsledného řešení

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 19. prosince 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Aplikace pro verifikaci zpráv

Aydar Mannanov

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Glazar

11. května 2022

Poděkování

Tímto bych chtěl poděkovat panu Ing. Filipu Glazarovi za vedení mé bakalářské práce, nápomocnost, ochotu, užitečné informace a cenné rady. Rád bych také poděkoval svým přátelům, kteří pro mě byli velkou oporou na cestě za bakalářským titulem. V neposlední řadě děkuji své rodině, a obzvláště svému otci, který ve mě vždy věřil.

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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2022 . . . . . .

Odkaz na tuto práci

Mannanov, Aydar. *Aplikace pro verifikaci zpráv*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022. Dostupný také z WWW: ⟨`https://projects.fit.cvut.cz/theses/4188`⟩.

Abstrakt

Tato práce se zaměřuje na návrh a implementaci webové aplikace pro ověřování zpráv (tzv. tweetů) ze sociální sítě Twitter. Hlavním účelem této aplikace je zabránit tomu, aby se ve webovém prostoru objevovaly falešné zprávy. Navržená aplikace dokáže ověřit tweety ze zdrojových zpráv tak, že se na ně dotazuje omezené skupiny uživatelů prostřednictvím dotazníku. Následně se na základě jejich odpovědí rozhodne, zda je zpráva důvěryhodná, či nikoli. Konečné výsledky průzkumu jsou ve formátu JSON. Aplikace načítá zprávy z Twitteru pomocí Twitter API. Backend je napsán v jazyce Java pomocí frameworku Spring. Frontend je napsán v jazyce TypeScript pomocí frameworku ReactJS a knihovny komponentu Bootstrap. Aplikace je implementována pomocí nástroje Docker.

Klíčová slova Twitter, Java, Spring, TypeScript, ReactJS, webová aplikace, verifikace zpráv, backend, frontend, Docker, Nginx

Abstract

This work focuses on the design and implementation of a web application for verifying messages (called tweets) from the social network Twitter. The main purpose of this application is to prevent fake messages from appearing in the web space. The created application can verify tweets from news sources by asking a limited group of users through a questionnaire and deciding whether the message can be trusted or not based on their answers. The resulting survey results are returned in JSON format.

The application retrieves news from Twitter using the Twitter API. The backend is written in Java using the Spring framework. The frontend is written in TypeScript using the ReactJS framework and the Bootstrap component library. The application is deployed using Docker.

Keywords Twitter, Java, Spring, TypeScript, ReactJS, web application, news verification, backend, frontend, Docker, Nginx

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Webový prostor se pro nás stal běžnou součástí našeho života. Dnes jsou pro nás jedním ze způsobů získávání nejaktuálnější informací sociální sítě. Bohužel jsou pro nás také zdrojem nepravdivých zpráv (tzv. „fake news“), a to zejména z toho důvodu, že většina lidí věří informacím, které si přečtou na internetu. V případě, že je odhalen podíl neověřených informací na celkovém počtu příspěvků, může dojít k tomu, že čtenáři budou zklamaní, a příspěvky tak ztratí na důvěryhodnosti.

Tato bakalářská práce je určena pro běžné uživatele webu, kteří mají dostatečné znalosti na to, aby byli schopni pomoci s případným ověřováním důvěryhodnosti zpráv, které se na webu vyskytují. Předem stanovené dotazy budou zasílány jen těm respondentům, kteří vyhověli nastaveným kritériím. Výsledky dotazníkového šetření budou přesměrovány na správce daného webu. Po důkladném prověření několika správci bude ohodnocena pravdivost takové zprávy, samozřejmě s možností jejího případného doplnění.

Při volbě tématu jsem vycházel ze své osobní touhy podílet se na zvýšení důvěryhodnosti informací, které se nachází na internetu. Problém množících se dezinformací je způsoben mimo jiné tím, že i běžný uživatel má dnes možnost se podílet na tvorbě webového obsahu, bez ohledu na to, zda má dostatečnou kvalifikaci v daném oboru. Pevně doufám a věřím, že po úspěšném uvedení této aplikace do provozu se podaří aspoň částečné vyřešit problém obrovského množství „fake news“ ve webovém prostoru.

Má bakalářská práce je rozdělena do čtyř částí: analýza, návrh, jeho implementace a následné testování prototypu. Jako zdroj dat pro vývoj aplikace News Verification App jsem si zvolil celosvětově známou sociální síť Twitter. V analytické části bude proveden výzkum, během kterého by mělo být zjištěno, zda podobná aplikace má v současném světě potenciál. Po důkladné analýze bude připraven návrh aplikace, který bude uveden do provozu v rámci testování. Stěžejním bodem výzkumu bude uživatelské testování vytvořeného prototypu aplikace. Získaná data budou sloužit jako podklad pro následné

ověřování pravdivosti zpráv.

1.1 Cíl práce

Hlavním cílem práce je navrhnout a implementovat prototyp webové aplikace pro ověřování zpráv, které se objevují na sociálních sítích. Žijeme dnes ve století informačního chaosu, proto je v současné době velmi důležité mít možnost si ověřit pravdivost získaných informací. Každý den se na webu objevuje nespočet falešných zpráv. Z tohoto důvodu by bylo užitečné vytvořit nástroj, který by sloužil k ověřování zpráv z různých zdrojů. Jedním ze způsobů ověření zprávy je anketování uživatelů daného webu. Na základě jejich odpovědí bude moci aplikace vyhodnocovat, zda jsou zprávy pravdivé či nikoli. Tato aplikace by měla být schopna, pomocí dobře vytvořených anketních otázek, se dotazovat omezené skupinu uživatelů a na základě jejich odpovědí vytvořit zpětnou vazbu o pravdivosti konkrétní zprávy.

Dalším cílem práce je analýza, zda podobná aplikace může reálně existovat a zda je možné ji případně použít pro ověření zpráv z různých zdrojů. V rámci tohoto cíle provedu testování podložené skutečnými zprávami a se skupinou skutečných uživatelů.

Kapitola 2

Úvod do problematiky

S fenoménem „fake news“ se potýkají výzkumníci po celém světě. Neurobiologové a matematici se taktéž zabývají podstatou falešných zpráv. Obzvláště dnes je důležité umět používat kritické myšlení a rozpoznávat nepravdivé informace. V této kapitole se vás pokusím seznámit s problematikou falešných zpráv a s tím, jak je rozpoznat. Na internetu se vyskytují dva druhy nepravdivých informací: dezinformace a misinformace.

2.1 Misinformace

Misinformace je nesprávná nebo zavádějící informace, která je neúmyslně nebo záměrně prezentována jako fakt. Dezinformace (viz. níže) jsou podmnožinou misinformací, jsou to ty nepravdivé informace, které jsou klamavé záměrně. Tento typ nepravdivých zpráv je šířen prostřednictvím fám. Lidé mohou například věřit misinformacím, protože spojují své vlastní emoce s tím, co slyší nebo čtou ve zprávách. Misinformace se může změnit v dezinformaci, pokud je sdílena skupinami či jednotlivci, kteří si jsou vědomi její nepravdivostí, a přesto ji záměrně šíří dále, aby vyvolali pochybnosti nebo podnítili rozkol.[3, 4]

2.2 Dezinformace

Nepravdivé informace, které se záměrně a často skrytě šíří (např. podsouváním fám) s cílem ovlivnit veřejné mínění a zastřít pravdu.[5]

2.3 Jak se můžu chránit před falešnými zprávami?

Jak poznat falešné zprávy? Existuje speciální postup ověřování faktů, který se nazývá „fact-checking“. Stavový diagram tohoto algoritmu je znázorněn na obrázku 2.1. Lze jej popsat následujícím algoritmem.[6]

- **Najděte zdroj zprávy**

  Častokrát, ale ne pokaždé, je dostačující věnovat pozornost zdroji zprávy, abychom zjistili její pravdivost. Přesto je ale důležité si uvědomit, že i zprávy z dobrého zdroje nemusí vždy znamenat, že jsou pravdivé. Proto je třeba si zprávy ověřit podle dalších zásad. Nejlepší z nich je vyhledat primární zdroj zprávy.

- **Číst dál než jen titulek**

  Snažte se nenechat ovlivnit šokujícím titulek zprávy. Novináři často volí chytlavé titulní názvy pro zvýšení popularity, lepší marketing nebo i šíření falešných informací. V ideálním případě vždy hledejte celý článek včetně dalších souvislostí.

- **Vyhledejte původní zdroj**

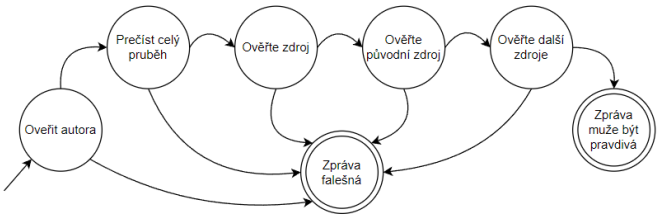
  Existují ve zprávě odkazy na zdroje informací? Navštivte tyto odkazy. Kdo je původcem dané zprávy? Oficiální orgány, renomované organizace nebo pochybní nejmenovaní svědci? Potvrzují informace i jiné zdroje?

- **Ověřte autora**

  Je ve článku uveden jeho autor? Jaké jsou jeho další články? Jsou o něm dohledatelné informace na internetu? Jakým tématům se věnoval dříve?

- **Získávejte zprávy z různých zdrojů**

  Dá se říct, že jsou v textu článku citelné emoce? Přílišná emocionalita ve zpravodajství je první známkou falšování zpráv nebo propagandy. Na jakou skupinu může článek cílit? Existuje někdo, kdo by mohl zájem o jeho šíření? A samozřejmě se zamyslete nad tím, jak moc vás osobně zajímá pravdivost či nepravdivost této zprávy. Nikdy nezapomínejte na svou vlastní předpojatost.



Obrázek 2.1: Stavový diagram pro ověření zpráv.

Kapitola 3

Analýza a návrh

Tato část mé bakalářské práce se bude týkat již existujících aplikací, které se věnují podobné problematice, a jejich následné analýze.

3.1 Existující řešení

Dnes existuje mnoho nástrojů a webových stránek, jejichž cílem je ověřit různá fakta a zprávy šířené na internetu.

3.1.1 BotSentinel

Bot Sentinel sleduje všechny účty na Twitteru. Platforma využívá strojového učení a umělou inteligenci k hodnocení účtů na Twitteru. Poté přidává a řadí účty do veřejně dostupné databáze, kterou může kdokoli procházet. Tento nastroj klasifikuje účty na Twitteru za pomocí tisíců účtů a miliony tweetů s využitím modelu strojového učení. Systém dokáže správně klasifikovat účty s přesností až 95 % a analyzovat stovky tweetů, aby přesně klasifikoval každý Twitter účet a poskytl snadně srozumitelnou zprávu.

Bot Sentinel je navržen tak, aby se dal snadno používat a poskytl co nejvíce dat. Veřejně poskytuje velmi podrobné informace o účtech na Twitteru, které jsou platformou sledovány. Návštěvníci tohoto webu mohou poté lépe a snadněji pochopit, jak nekalé účty šíří dezinformace a na co se zaměřují. Platforma se pokouší být co nejtransparentnější a poskytnout návštěvníkům co nejvíce datových bodů.[7]

3.1.2 BotSlayer

BotSlayer je aplikace, která napomáhá sledovat a odhalovat potenciální manipulaci s informacemi šířícími se na Twitteru. Nástroj byl vyvinut Observatoří sociálních médií na univerzitě v Indianě v USA.[8]

Tento nastroj využívá algoritmus detekce anomálií k identifikaci hashtagů, odkazů, účtů a médií. Webový ovládací panel umožňuje uživatelům prozkoumávat zprávy (tzv. tweety) a účty spojené s podezřelými kampaněmi na Twitteru a vyhledávat související obrázky a obsah na Googlu.

BotSlayer mohou používat například novináři, korporace či političtí kandidáti, aby v reálném čase mohli objevovat nové koordinované kampaně v oblastech svého zájmu, aniž by o těchto kampaních byli předem informováni. Systém je možné nainstalovat a nakonfigurovat v cloudu tak, aby monitoroval aktivitu botů v rámci stálého dotazu definovaného uživatelem.

BotSlayer je navržen jako crowdsourcingová platforma. Výměnou za bezplatnou službu jsou systému poskytována anonymní data, která jsou odesílaná zpět do výzkumného zařízení. Tento způsob je v souladu s podmínkami a pokyny společnosti Twitter. Získaná data mohou pomoci při výzkumu, a tak včasně odhalit fenomény manipulace s informacemi na sociálních sítích.

3.1.3 AltNews

AltNews je webová aplikace, která slouží k ověřování faktů, jimiž se snaží vyvracet dezinformace a špatné informace, se kterými se denně setkáváme na sociálních sítích i v populárních médiích.[9]

Kontrola faktů pomocí AltNews zahrnuje:

- ověřování výroků politických stran, politických představitelů a dalších osob na vedoucích pozicích napříč politickým spektrem.
- vyvracení dezinformací, které se šíří na sociálních sítích a na internetu.
- zkoumání nepřesných nebo zavádějících informací šířených populárními médii, ať již neúmyslně, nebo s cílem utvářet a měnit veřejné mínění ve prospěch politických subjektů.

Podle informací na hlavní stránce je AltNews finančně nezávislá aplikace, což je nespornou výhodou. Aplikace také podporuje ověřování faktů z různých zdrojů, což je také jistě výhoda. Nevýhodou aplikace je, že se zaměřuje na indický region, nicméně již nyní je možné ověřovat fakta i z jiných regionů.

3.1.4 Tabulka porovnávající nástroje pro detekci falešných zpráv

	BotSlayer	BotSentiel	AltNews
Sledování podezřelých účtů	+	+	-
Podpora různých zdrojů informací	-	-	+
Dotazování uživatelů na zprávy	-	-	-
Zaměření na globální trh	+	+	-
Finanční nezávislost	+	+	+
Používání algoritmů umělé inteligenci	+	+	-
Open source	+	-	-

Tabulka 3.1: Porovnání nástroje pro detekci falešných zpráv.

3.1.5 Shrnutí analýzy existujících řešení

Po provedené analýze mužů říct, že skutečně podobné aplikaci pro ověření zpráv již existují, ale mají svá úskalí. Například nemají možnost anketovat uživatele ohledně určité zprávu, ne všechny tyto nástroje jsou open source, a ne všechny z nich podporují různost zdrojů zpráv. Při návrhu a implementaci prototypu aplikace budu i těmto bodům věnovat pozornost.

3.2 Analýza a výběr zdrojů dat

V dnešní době na internetu existuje obrovské množství zdrojů zpráv (Facebook, Instagram, Twitter, Telegram atd.). V této fázi je mým úkolem analyzovat zdroje informací, které jsou dnes populární, a vybrat si ten, který nejlépe vyhovuje mým potřebám. V budoucnu plánuji přidal podporu pro všechny populární zdroje zpráv.

3.2.1 Facebook

Facebook je největší sociální sítí[10] společnosti Meta. Byl založen 4. února 2004 Markem Zuckerbergem a jeho spolubydlícími během studia na Harvardově univerzitě. V současné době využívá Facebook obrovské množství uživatelů jako hlavní zdroj aktuálních zpráv.[11]

Pro mé účely není tento zdroj zpráv vhodný, protože má komplikovaný API a nedisponuje možností pokročilého filtrování dotazů.

3.2.2 Instagram

Instagram je sociální sítí pro sdílení fotografií a videí. V dubnu 2012 společnost Meta koupila tuto síť za přibližně 1 miliardu dolarů v hotovosti a v akciích.[12]

Aplikace umožňuje uživatelům nahrávat mediální soubory, které lze upravovat filtry a organizovat pomocí hashtagů. Příspěvky lze sdílet veřejně nebo s předem schválenými sledujícími.

Stejně jako u API Facebooku je API Instagramu zbytečně složité a pro můj výzkum by vyžadovalo velmi hluboké znalosti. Nevýhodou je také omezený počet požadavků na API a krátké bezplatné zkušební období.

3.2.3 Twitter

Sociální síť pro sdílení veřejných zpráv (tzv. tweetu). Zprávy je možné publikovat a posílat pomocí webového rozhraní, SMS nebo klientského softwaru třetí strany, který je dostupný pro uživatele internetu všech věkových kategorií. Publikování krátkých textů ve formátu blogu se nazývá mikroblogování.[13] Jelikož byl Twitter vytvořen jako nástroj sloužící pro blogování, stal se klíčovým komunikačním prostředkem pro veřejně známé osobnosti a také vhodným nástrojem pro šíření zpráv médii. Je to však také sociální síť, kde se často vyskytují podezřelí uživatelé. Z tohoto důvodu se zde často objevují falešné zprávy.[14]

Twitter má velmi dobře zdokumentované API. Má také pokročilé vyhledávání aktuálních tweetů. Pro mé výzkumné účely Twitter poskytuje speciální přístup ke svému rozhraní API. Proto budu pro účely této bakalářské práce používat právě tento zdroj zpráv.

Kapitola 4

Návrh aplikace

Návrh aplikace je další fází mé práce. V této části se zaměřím na design aplikace a její základní procesy. Pro tento účel použiji jazyk pro modelovaní systému Unified Modeling Language (dále UML)[15]. Poté také definuji hlavní scénáře použití a funkční požadavky. Dále bude následovat samotný návrh prototypu včetně výběru vhodných technologií. Na závěr naznačím další potencionální možnosti vývoje aplikace.

4.1 Funkční požadavky

Funkční požadavky se používají k definování funkcí, kterými musí aplikace disponovat, aby ji bylo možné použít. Jedná se o seznam stručně popsaných základních funkcí aplikace.

Níže je uveden seznam **funkčních požadavků**:

- **F1 - Správa uživatelského účtu**

  Aplikace by měla dát svým uživatelům možnost zobrazit svůj profil. Po přihlášení by měla být dostupná většina funkcí aplikace. Noví uživatelé by měli mít možnost se zaregistrovat.

- **F2 - Získávání aktuálních zpráv od zdroje**

  Aplikace bude umožňovat správcům získávat aktuální zprávy ze zdroje, aby je bylo možné dále ověřit.

- **F3 - Správa dotazníků**

  Další součástí aplikace je možnost vytvoření nového dotazníku týkajícího se nějaké zprávy, jeho upravování a uzavírání.

- **F4 - Přidání otázek do dotazníku**

  Po vytvoření nového dotazníku aplikace umožní přidání dalších otázek a přiřazení skupiny uživatelů za účelem získání co nejpřesnějších výsledků.

- **F5 - Sběr odpovědí od uživatelů**

  Aplikace by měla být schopna přijímat odpovědi na otázky od respondentů.

- **F6 - Získání výsledků dotazování**

  Po uzavření dotazníku by aplikace měla poskytnout výsledky ve formátu vhodném pro další výzkum.

4.2 Nefunkční požadavky

Nefunkční požadavky jsou dodatečné technické požadavky zapracované do aplikace.

Níže je uveden seznam **nefunkčních požadavků**:

- **NF1 - Webová aplikace**

  Dostupnost aplikace bude zajištěna přes web.

- **NF2 - Podpora populárních prohlížečů**

  Aplikace by měla být spustitelná ve většině prohlížečů, které jsou dnes populární.

- **NF3 - Dostupnost**

  Aplikace by pro své uživatele měla být přístupná kdykoli.

- **NF4 – Rychlá spustitelnost**

  Aplikace by měla běžet v rámci Docker kontejneru, aby bylo možné aplikaci snadno a rychle spustit a nainstalovat na různých platformách.

4.3 Případy užití a účastníci

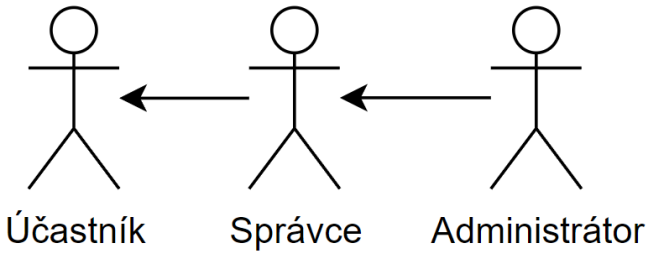
Případy užití (dále UC – z anglického „use case“) pomáhají detailně definovat a popsat funkčnost prototypu aplikace. Při implementaci aplikace je lze použít jako plán kontroly realizovaných funkcí. Nejprve je potřeba určit účastníky, kteří budou s aplikací pracovat.

4.3.1 Účastníci

Na obrázku 4.1 jsou zobrazeni všichni účastníci používající prototyp aplikace:

- **Účastník**, dostává dotazy ke konkrétním zprávám.
- **Správce** vytvoří dotazníky a rozešle je uživatelům, po obdržení odpovědí je schopen analyzovat výsledky dotazování.

- **Administrátor** sleduje aplikaci jako celek a kontroluje uživatele, aby bylo možné zabránit šíření falešných zpráv. V jeho kompetencích je také snížení ratingu uživatele.



Obrázek 4.1: Účastníci

4.3.2 Případy užití

Níže je uveden seznam **případů použití**. Každý UC je identifikován jednoznačným identifikátorem, názvem a krátkým vysvětlujícím popiskem.

1. **Správa uživatelských účtů**
   - **UC1** - Registrace
   - **UC2** - Přihlášení
   - **UC3** - Odhlášení
   - **UC4** - Editace účtů
   - **UC5** - Změna ratingu uživatele
2. **Správa dotazníků**
   - **UC6** - Získávání aktuálních zpráv od zdroje
   - **UC7** - Vytvoření dotazníku pro ověření zprávy
   - **UC8** - Přidání otázek do dotazníku
   - **UC9** - Odpovídání na otázky
   - **UC10** - Přijímání otázek
   - **UC11** - Zasílání otázek respondentům
   - **UC12** - Získávání odpovědí na otázky od respondentů
   - **UC13** - Ukončení dotazníku
   - **UC14** - Získání výsledků dotazování

Na diagramu v příloze B.1 jsou znázorněni účastníci a případy použití.

4.4 Diagram aktivit

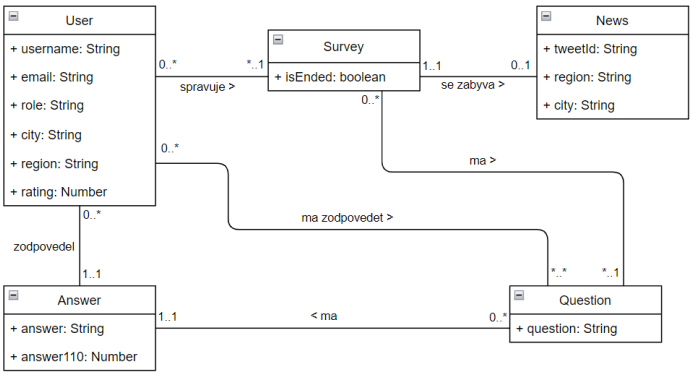
Funkčnost aplikace lze popsat jako posloupnost procesů, které v rámci ní probíhají. Diagram aktivit slouží k popisu a definování složitých událostí, které nelze jednoduše popsat slovy. K popisu těchto sekvencí použiji standard UML.

Na diagramu aktivit v příloze B.2 je popsán proces ověření pravdivosti zpráv počínaje od okamžiku spouštění aplikace až po okamžik získání potřebných dat. Diagram je rozdělen do tří sloupců: Účastník, Aplikace, Správce. V každém sloupci je zóna odpovědnosti jednotlivých subjektů. Řetězec aktivit začíná u správce, stejně jako u něj končí. Celý diagram je určen primárně pro použití ze strany správce webu. Cílem tohoto diagramu je popsat sekvenci procesů, které jsou nezbytné pro ověření pravdivosti zprávy.

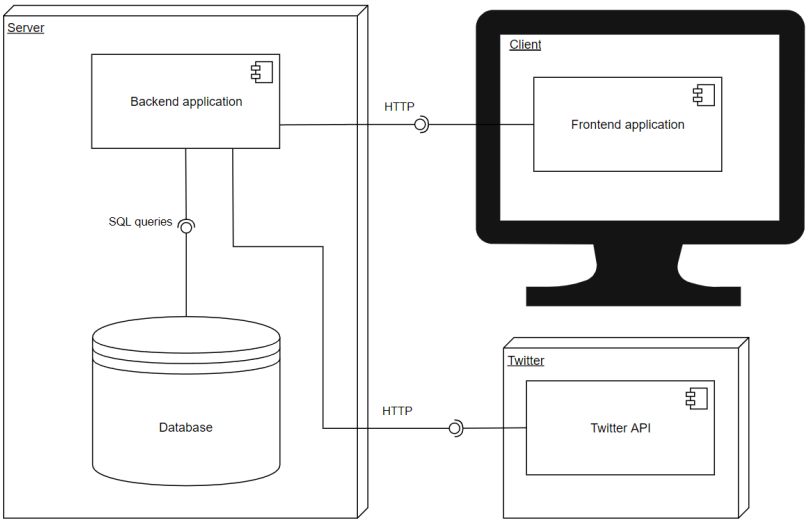
4.5 Doménový model

Doménový model slouží k definování struktury aplikace, tj. jednotlivých entit, jejich atributů a vztahů mezi nimi. Nezávisí na konkrétní platformě, ani na konkrétním programovacím jazyce či technologiích a ani na architektuře aplikace. K definici modelu se dnes běžně používá standard UML.[15]

Z doménového modelu lze pak snadno vytvořit databázový model, který se používá pro vkládaní dat a jejich následné uplatnění. S daty v databázovém modelu je možné provádět CRUD operace. Data budou uložena do relační databáze, se kterou se bude pracovat později. Na základě diagramu aktivit B.2 a diagramu případů užití B.1 jsem na obrázku 4.2 definoval doménový model aplikace News Verification App. Tento doménový model současně v sobě obsahuje pět entit: News, User, Survey, Question, Answer. V rámci dalšího vývoje aplikace navrhuji v budoucnu rozšířit doménový model o další entity.



Obrázek 4.2: Doménový model



Obrázek 4.3: Architektura klient-server

4.6 Struktura aplikace

Aplikace se skládá z klientské a serverové části. Většina funkcí se zpracovává na serveru. Jelikož má aplikace využívá architekturu klient-server, je velmi flexibilní pro budoucí rozšiřování. Serverová část aplikace musí být schopna pracovat s databází a zároveň musí být dostatečně bezpečná pro ukládání citlivých dat. Klientská část také musí splňovat moderní standardy, tj. používat aktuální technologie. Mělo by být možné připojit klientskou část k serveru. Architektura klient-server též ponechává možnost v budoucnu rozšířit podporu pro další platformy včetně mobilní aplikace.

Klientská aplikace by měla běžet ve všech populárních prohlížečích. Klientská a serverová část spolu komunikují pomocí protokolu HTTP, v budoucnu bych při úpravách prototypu přidal podporu protokolu HTTPS.

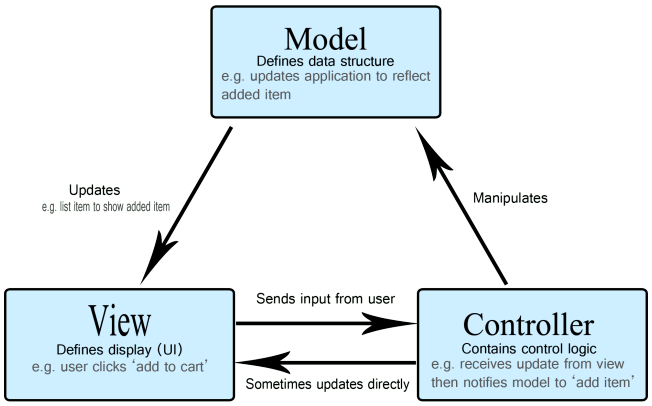
Na straně serveru bude klientům k dispozici rozhraní RESTful API, které obsluhuje základní funkce aplikace. API má všechny potřebné endpointy pro práci s entitami v databázi.

Na obrázku 4.3 je architektura mé klient-server aplikace.

Zaběhlou praxí při navrhování je rozdělení struktury aplikace do různých vrstev, tj. použití principu „Separation of concerns“ (dále SoC). Pro modelování architektury používám vzor Model-View-Controller (dále MVC) obrázek 4.4. Vrstvy Controller a Model jsou implementovány v serverové části. Vrstva View se nachází v klientské části.

4.6.1 Serverová aplikace

Serverová aplikace vyřizuje požadavky klientů, ukládá je a zpracovává data, proto je potřeba, aby byla dostatečně zabezpečená.



Obrázek 4.4: Model-View-Controller [1]

.

4.6.1.1 Model

Model poskytuje data a metody pro jejich zpracovávání: dotazy v databázi, kontrola správnosti. Model by měl být zcela nezávislý na jiných částech aplikace. Modelová vrstva by neměla znát návrhy elementů a ani způsob, jakým se budou zobrazovat. Výsledkem je možnost změnit reprezentaci dat a způsob jejich zobrazení, aniž by bylo nutné zasahovat do samotného Modelu.

4.6.1.2 Controller

Úkolem kontroléra je pracovat s byznysovou částí aplikace. Kontrolér bude zpracovávat dotazy od klientů, tj. zajišťovat spojení mezi datovou a prezentační vrstvou.

4.6.2 Klientská aplikace

Klientská aplikace slouží jako rozhraní mezi uživatelem a serverem. Jejím úkolem je zobrazovat přijatá data a zpracovávat požadavky uživatelů.

4.6.2.1 View

Prezentační vrstva je zodpovědná za načtení požadovaných dat z modelu a jejich odeslání uživateli.

4.6.3 Twitter API

V rámci bakalářské práce potřebuji mít možnost získávat zprávy z Twitteru. K tomu mi poslouží Twitter API. Výběr zdroje dat jsem provedl v části 3.2.

Pro přístup k tomuto API jsem si předem vyžádal přístupové klíče. Tyto klíče mi umožňují přijímat až 500 000 tweetů měsíčně. To je pro práci na prototypu dostačující, ale v případě rozšíření bude pravděpodobně potřeba přijímat více tweetů.

Twitter má velmi přívětivé a důmyslné API, které umožňuje načítat a filtrovat tweety podle tagů, geotagů, obsahu atd. Pro filtrování používá vlastní dotazovací jazyk, který pomáhá provádět složité dotazy. API také umožňuje přidávat a upravovat obsah na Twitteru. Další zajímavou funkcí je sledování trendů podle tagů a obsahu tweetů, v budoucnu by tato funkce mohla být využita ke sledování falešných zpráv.

V mé práci bude postačující vyžádání tweetů s aktualitami z konkrétních zdrojů. V budoucnu bych přidal možnost, aby správci mohli zadávat vlastní požadavky na API pomocí uživatelského rozhraní v závislosti na konkrétním cíli.

4.7 Technologie

Dalším velmi důležitým krokem při navrhování aplikace je výběr technologií, programovacích jazyků a vhodných frameworků. Při výběru jsem se zaměřil především na dostupnost, rychlost vývoje a na jednoduchost. Níže popisuji většinu technologií, které jsem ve své práci použil.

4.7.1 Pomocné nástroje

- **Git**

  Pro rychlý a bezpečný vývoj aplikace budu používat systém správy verzí Git se vzdáleným připojením ke službě Github, abych mohl ukládat kopie projektu, které nejsou na lokálním disku. Pomůže mi to také rychle přejít na starší verze kódu a rozšířit možnosti budoucího ladění.[16]

- **Docker**

  Docker je platforma, která umožňuje zabalení aplikací do kontejnerů a jejich spouštění na serverech. Docker umožňuje vkládat kód a závislosti do kontejnerů. Systémy založené na kontejnerech se tak snadno škálují, protože kontejnery lze přenášet a replikovat.[17]

Frontend, backend, server Nginx a Postgres databáze běží v kontejnerech.

- **Docker-compose**

  Docker-compose slouží ke správě více kontejnerů současně. Tento nástroj nabízí stejné funkce jako Docker, ale umožňuje pracovat se složitějšími aplikacemi.[18]

  Pomocí nástroje Docker-compose je možné spustit a nakonfigurovat celou aplikaci jedním příkazem.

- **Nginx**

  Nginx je HTTP server, reverzní proxy server, e-mailový proxy server a univerzální proxy server TCP/UDP. V rámci projektu ho budu používat jako proxy server pro přesměrování požadavků na backend.[19]

4.7.2 Backend

- **Java**

  Java je striktně typovaný, objektově orientovaný a univerzální programovací jazyk vyvinutý společností Sun Microsystems. Tento jazyk je z rodiny podobných jazyků C, což je při vývoji mé aplikace značnou výhodou, jelikož jsem se již během svého studia a práce s podobnými jazyky mnohokrát setkal.[20]

- **Spring framework**

  Tento framework poskytuje komplexní konfigurační a programovací model pro moderní podnikové aplikace v jazyce Java na jakékoliv platformě.[21]

  Klíčové vlastnosti frameworku:

  - Podpora dependency injection
  - Validace dat
  - Testovaní: mock objekty, TestContext framework, Spring MVC Test
  - Správa dat: transakce, JDBC, ORM
  - Spring MVC

- **Spring boot**

  Spring boot pomáhá vytvářet kompletní aplikace Spring v jazyce Java. Má vestavěný HTTP server Tomcat. Disponuje také možností automatické konfigurace aplikace ve všech fázích vývoje.[22]

- **Gradle**

  K sestavení backendu používám Gradle, což pomůže přidat do projektu všechny potřebné balíčky.[23] A to vše pomocí jediného konfiguračního souboru.

- **SQL**

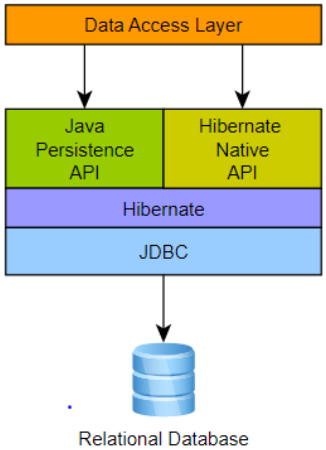
  SQL je strukturovaný dotazovací jazyk (Structured Query Language), který umožňuje ukládat, manipulovat a načítat data z relačních databází (RDBMS, DB).[24]

- **PostgreSQL**

  PostgreSQL je dialektem jazyka SQL, který pracuje s relačníma databáze. V dnešní době je poměrně populárním dotazovacím jazykem, a to především díky své flexibilitě a rychlosti. Velkou výhodou je to, že Postgres je open-source a má za sebou velkou komunitu.[25]

- **JPA**

  JPA (Java Persistence API) je specifikace Java EE a Java SE, která popisuje systém pro pohodlnou správu perzistence Java objektů v relačních databázových tabulkách. Samotná Java neobsahuje implementaci JPA, ale existuje mnoho implementací této specifikace od různých společností. Nejedná se o jediný způsob ukládání javovských objektů do databází (systémy ORM), ale v rámci Javy patří k nejoblíbenějším.[26]



Obrázek 4.5: JPA API [2]

- **Hibernate**

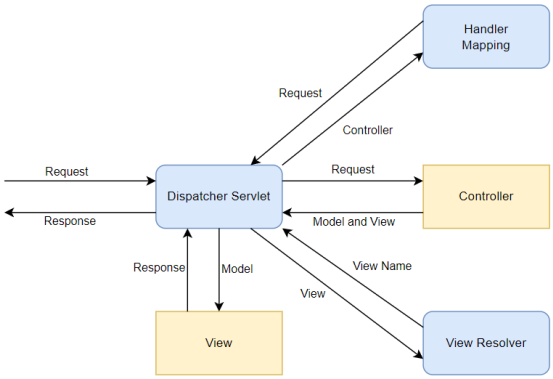
  Hibernate je jednou z nejpopulárnějších open source implementací nejnovější verze specifikace (JPA 3.0). To znamená, že JPA pouze popisuje pravidla a rozhraní API a Hibernate tyto popisy implementuje (viz. 4.5).[27]

- **Spring security**

  Jde o framework jazyka Java, který poskytuje mechanismy pro vytváření systémů autentizace a autorizace a dalších bezpečnostních funkcí pro průmyslové aplikace vytvořené pomocí frameworku Spring.[28]

- **Spring MVC**

  Jedná se o framework pro Spring, který pomáhá psát webové aplikace pomocí vzoru MVC. Princip Spring MVC je postaven na DispatcherServlet, který přijímá a zpracovává všechny požadavky HTTP a odpovědi na ně. Proces zpracování požadavků DispatcherServlet je znázorněn v následujícím diagramu 4.6.[29]



Obrázek 4.6: DispatcherServlet. Spring MVC.

- **JWT**

  JSON Web Token (dále JWT) je objekt JSON definovaný v otevřeném standardu RFC 7519 [30]. Je považován za jeden z bezpečných způsobů přenosu informací mezi dvěma stranami. Pro jeho vytvoření je třeba definovat hlavičku(header) s obecnými informacemi o tokenu a užitečné soubory(payload) jako je ID uživatele, role, podpis(signature) atd.

  Uživatel po autorizaci obdrží token, který může později použít pro své požadavky na server, tj. pro autentizaci.

4.7.3 Frontend

- **NPM**

  Jedná se o správce balíčků, který je součástí Node.js.[31]

- **JavaScript**

  JavaScript je multiparadigmatický programovací jazyk, který není striktně typovaný, bez znalosti této technologie nelze vytvářet moderní interaktivní webové stránky.[32]

- **Axios**

  Axios je jeden z nejpopulárnějších klientů HTTP pro prohlížeče a node.js založený na promise. Axios podporuje požadavky, přijímá odpovědi ze serveru, transformuje je a automaticky převádí do formátu JSON.

  Axios používám pro pokročilejší backend dotazy, mohl bych to udělat s obvyklým fetch, ale pro další rozšíření jsem přidal tento balíček do projektu.[33]

- **TypeScript**

  TypeScript je striktně typovaný a kompilovaný jazyk. Výstupem kompilátoru je však stále stejný JavaScript, který je následně spuštěn prohlížečem. Striktní typování však snižuje počet potenciálních chyb, které mohou při vývoji v jazyce JavaScript vzniknout.[34]

  Výhody TypeScriptu:

  - Implementuje mnoho konceptů společných pro objektově orientované jazyky jako je dědičnost, polymorfismus, zapouzdření, modifikátory přístupu atd.
  - Vyvinuto jako open-source projekt
  - Jakýkoli program v jazyce JavaScript je program v jazyce TypeScript
  - Snadnější údržba, vývoj, škálování a testování než standardní JavaScript

  Tento jazyk používám společně s frameworkem React k vývoji frontendové části.

- **ReactJS**

  React je JavaScriptová knihovna, která slouží k vytváření uživatelského rozhraní. React se považuje za ideální nástroj pro tvorbu webových aplikací, a to především v situacích, kdy se jedná o aplikaci SPA (single-page application).[35]

- **Bootstrap**

  Bootstrap je bezplatná sada nástrojů pro stylování webových stránek a webových aplikací. Obsahuje designové šablony HTML a CSS pro typografii, webové formuláře, tlačítka, štítky, navigační boxy a další komponenty webového rozhraní včetně rozšíření
  JavaScriptu.[36]

  Jelikož používám Bootstrap uvnitř Reactu, tak přidám balíček react-bootstrap.

- **Twitter-Embed**

  Jedná se o balíček, který pomáhá vykreslovat obsahy z Twitteru přímo na stránkách React aplikaci.[37]

Kapitola 5

Implementace

Tato kapitola bakalářské práce se bude podrobně zabývat implementací. Popis implementace bude rozdělen na tři části: backend, frontend a nasazení aplikace. Tyto části budou doplněny kódem a obrázky s podrobným popisem.

V této kapitole samozřejmě nemohu popsat celý kód, proto popíšu pouze důležité části implementace. Celý zdrojový kód s potřebnými komentáři je k dispozici na přiložené SD kartě.

5.1 Backend

Hlavním úkolem backendu je přijímání a zpracovávání požadavků REST API, tj. spravování aplikace. Tyto požadavky lze odesílat nejen z frontendu, ale také přímo hostiteli prostřednictvím adresy *http://host/api*. Toto rozhraní pokrývá všechny CRUD operace nad entitami v rámci aplikace. Je také poměrně bezpečné, pro získání přístupu je třeba získat klíč JWT(autorizace) a odeslat jej v každém požadavku(autentizace), který jej vyžaduje. Na stránce Postmanu je celá dokumentace tohoto API [38].

Jádrem mého backendu je framework Spring, v němž je napsáno celé API. Použil jsem moduly frameworku Spring, jako je Spring MVC, Spring security, Hibernate, ale také některé podpůrné balíčky z repozitáře MVN. Gradle je zodpovědný za sestavení celého projektu. PostgresDB je zodpovědný za ukládání informací.

5.1.1 Struktura backendu

Obrázek 5.1.1 ukazuje adresářovou strukturu mého backendu. V souboru *build.gradle* jsou uloženy všechny závislosti pro sestavení backendu. Soubor *Dockerfile* obsahuje podrobné informace o sestavení kontejneru. Všechny klíče aplikací jsou uloženy ve složce *src/main/resources/keys* a nejsou sledovány systémem git, aby se zabránilo úniku informací. Sestavené soubory *.jar* jsou uloženy ve složce *build*, tyto soubory budeme ještě později potřebovat pro nasazení.

Obrázek 5.1: Adresářová struktura backendu.

```
.gradle ..... adresář s logy Gradle
build ..... adresář s binárními soubory
gradle ..... adresář Gradle
src ..... zdrojové kódy
    main
        java
            cz/cvut/fit/mannaayd/bachelor/backend
                component ..... komponenty aplikace
                    helper ..... pomocníci aplikací
                    mapper ..... mapovače aplikací
                config ..... konfigurační beany
                controller ..... kontroléry endpointů
                exception ..... výjimky
                model ..... modely aplikací
                    authorities ..... modifikátory přístupu
                    dto
                    ...
                repository ..... databázový repositář
                service ..... aplikační služby
                BackendApplication.java ..... entrypoint aplikace
        resources
            keys ..... bezpečnostní klíče
                keystore.jks ..... klíče pro podepisování JWT tokenů
            application.properties ..... konfigurace Spring Boot
    test ..... testy aplikace
.gitignore ..... seznam souborů, které má Git ignorovat
build.gradle ..... seznam závislostí pro Gradle
Dockerfile ..... konfigurace kontejneru Docker
gradlew ..... spuštění Gradle pro Linux
gradlew.bat ..... spuštění Gradle pro Windows
HELP.md ..... pokyny
settings.gradle ..... konfigurace Gradle
```

Složka *src/.../bachelor/backend* obsahuje všechny zdrojové soubory backendu aplikace. V této složce je uložen tzv. entrypoint (odkud se spouští cely backend) *BackendApplication.java*. Také jsou zde složky s modely, konfiguracemi, výjimkami, komponenty, kontroléry, repositáři a službami.

5.1.2 Model

Složka model obsahuje všechny modely, které backend potřebuje pro správu dat, tyto modely se ukládají do databáze. V této složce se nacházejí také mo-

dely dto, které jsou potřebné pro komunikaci s uživateli, a také modely, které jsou nezbytné pro zajištění přístupu k zabezpečeným endpointům aplikace.

Níže je uveden model uživatele aplikace. Anotace *@GeneratedValue* pomáhá automaticky vygenerovat id při vytváření a zároveň zabraňuje kolizi identifikátoru. Anotace *@Entity* nám také říká, že tato třída je entitou v databázi.

```
@Entity
@Table(name="appuser")
public class User{
    @Id
    @GeneratedValue
    private Long Id = 0L; // id of user, primary key
    @NonNull
    @Column(name="username", nullable = false, unique = true)
    private String username;
    @Column(name="password", nullable = false)
    private String password;
    @Column(name="email", nullable = false, unique = true)
    private String email;
    @Column(name="rank", nullable = false)
    private int rank;
    @Column(name="region", nullable = false)
    private String region;
    @Column(name="city", nullable = false)
    private String city;
    @Column(name="user_role", nullable = false)
    private String role;
    @ManyToMany(mappedBy = "respondents",
                cascade = CascadeType.PERSIST)
    private List<Question> questions;
    @OneToMany(mappedBy = "author")
    private List<Survey> surveys;
    @OneToMany(mappedBy = "user")
    private List<Answer> answers;
}
```

5.1.3 Repository

Rozhraní Repository umí provádět všechny CRUD operace v rámci tabulek v databázi. Toho je dosaženo pomocí modulu Hibernate, který poskytuje metody pro práci s relačními databázemi.

Spring Data JPA poskytuje různé možnosti definování a provádění dotazů:

- odkaz na pojmenovaný nativní dotaz nebo dotaz JPQL
- provádění dotazu odvozeného od názvu metody v Repository
- deklarovaní dotazu pomocí anotace *@Query*

Níže je uveden příklad *UserRepository*, který pracuje s tabulkou uživatelů. Toto rozhraní je zděděno z rozhraní *JpaRepository<I, E>*, kde I je typ identifikátoru entity a E je třída entity. Všechny metody, které provádějí operace s entitami, jsou odvozeny od názvů.

```
1  public interface UserRepository extends JpaRepository<User, Long> {
2      @Override
3      Optional<User> findById(Long id);
4
5      @Override
6      List<User> findAll();
7
8      Optional<User> findByUsername(String username);
9
10     List<User> findUsersByRegionAndCity(String region, String city);
11
12     List<User> findUsersByRegion(String region);
13 }
```

Konfigurace databáze se provádí v souboru *application.properties*, kde se nastavují všechny potřebné údaje pro připojení a také dialekt databáze, v mém případě je to Postgres. Níže jsou uvedeny řádky v konfiguračním souboru.

```
1  spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.↩
       PostgreSQLDialect
2  spring.jpa.hibernate.show-sql=true
3  spring.datasource.url=jdbc:postgresql:/host:port/postgres
4  spring.datasource.username=postgres
5  spring.datasource.password=goodpass
```

5.1.4 Service

Hlavní aplikační logika je zpracována v tzv. službách. Úkoly těchto služeb jsou velmi různorodé. Ve většině případů je úkolem zpracovávat požadavky z kontrolérů a zároveň pracovat s daty z databáze. V mém případě existuje také samostatná služba s názvem *TwitterService*, která odesílá požadavky HTTP na rozhraní TwitterAPI.

Služby jsou přidávány do kontrolérů pomocí tzv. dependency injection, což později vysvětlím v části o kontrolérech. Aby toto bylo možné, je třeba do třídy přidat anotaci *@Service*.

V softwarovém inženýrství je dependency injection návrhový vzor, ve kterém objekt přijímá další objekty, na kterých je závislý. Injektáž závislostí je formou inverze řízení (tzv. IoC), jejímž cílem je oddělení zodpovědností (tzv. SoC) spojených s konstrukcí objektů a jejich následném používáním.[39]

Níže je uvedená služba využívaná pro práci s uživateli aplikace, která pracuje s Repository uživatelů. Dále také používá mapovače pro převod entit na objekty DTO a naopak.

```
1  @Service
2  public class UserService implements IUserService {
3      public UserMapper userMapper;
4      public UserRepository userRepository;
5
6      ...
7
8      @Override
9      public List<User> getAll() {
10         return userRepository.findAll();
11     }
12
13     ...
14
15     @Override
16     public User save(UserDTO userDTO) throws NotExistsException {
17         return userRepository.save(userMapper.toEntity(userDTO));
18     }
19
20     ...
21
22 }
```

5.1.4.1 TwitterService

Pro práci s rozhraním Twitter API jsem nejprve chtěl použít již hotové řešení pro Javu nacházející se na stránce Developer Tools [40]. Po vyzkoušení všech dostupných řešení mi v době psaní backendu (05.03.2022) žádné z nich nevyhovovalo, a to z různých důvodů. Například jsem narazil na jednu zajímavou chybu v jedné z knihoven. Ta nebyla stabilní a při jednom jednoduchém požadavku na Twitter API vrátila 10000 tweetů, po několika minutách ladění jsem na požadavky vyčerpal polovinu svého limitu. Jiné knihovny mi nevyhovovaly, protože nepokrývaly mé případy použití aplikace.

Níže je uveden seznam nástrojů, které jsem testoval na svém případu použití:

- **JTW** Twitter V2 API klient knihovna pro Javu
- **twitter4j-v2** jednoduchý wrapper pro Twitter API v2, který je určen pro použití s Twitter4J.
- **twittered** Twitter API klient pro Javu

Tyto knihovny jsou v současné době aktivně vyvíjeny a myslím, že v budoucnu budou chyby, na které jsem narazil, opraveny. Po rešerši jsem se rozhodl vytvořit vlastní službu a použít Twitter API přímo a bez použití speciálních knihoven.

Níže je uveden zdrojový kód metody z *TwitterService* pro odesílání dotazů na API Twitteru, v tuto chvíli funguje pouze s posledními tweety ze zdrojů, v budoucnu bych chtěl přidat možnost pro správce webu vytvářet vlastní dotazy

na tento API. Účelem tohoto kódu je odeslat HTTP požadavek s konkrétním dotazem na data (tzv. „query“) do API.

```
1  public TweetListDTO getLastTweets(int count) throws Exception {
2
3      ...
4
5      headers.setBearerAuth(twitterCredentials.bearerToken); // adding ↩
           jwt token
6      headers.setAccept(Collections.singletonList(MediaType.↩
           APPLICATION_JSON));
7      HttpEntity<String> request = new HttpEntity<>(headers);
8      List<TweetDTO> tweetDTOs = new ArrayList<TweetDTO>();
9      for(String source : sources) {
10         String query = "?query=from:" + source +"&max_results=" + ↩
               Integer.toString(count/sources.size()); // link with query
11         String url = apiUrl + lastTweetsUrl + query; // query request ↩
               on Twitter API
12         ResponseEntity<TweetListDTO> response = this.restTemplate.↩
               exchange(url, HttpMethod.GET, request, TweetListDTO.class,↩
                1) // mapping recieved body into DTOs
13         tweetDTOs.addAll(response.getBody().getData());
14     }
15     TweetListDTO res = new TweetListDTO();
16     res.setData(tweetDTOs);
17     return res;
18 }
```

Pro přístup k API Twitteru jsou potřebné klíče, které ukládám do souboru *twitter_keys.json* na lokální disk. Níže je uveden kus kódu, který načte klíče ze souboru a vytvoří z nich objekt *TwitterCredentials*.

```
1  try {
2      ObjectMapper objectMapper = new ObjectMapper();
3
4      this.twitterCredentials = objectMapper.readValue(new File("↩
           twitter_keys.json"), TwitterCredentials.class); // mapping ↩
           twitter keys
5
6  } catch (IOException e) {
7      e.printStackTrace();
8  }
```

5.1.5 Controller

Úkolem kontrolérů je přijímat HTTP požadavky od některého klienta. Tyto požadavky jsou modelovány ve stylu architektury REST API a zahrnují všechny CRUD operace.

Při odesílání požadavku HTTP na rozhraní REST API je třeba zadat typ požadavku, cestu, hlavičky a tělo. Pro autorizované požadavky je třeba zadat hlavičku Authorization s hodnotou tokenu JWT. Podrobnější informace o tomto procesu budou uvedeny v části Bezpečnost.

Typy požadavků mohou být různé, v mém API používám pouze základní typy – GET, POST, PUT a DELETE. Tyto typy pokrývají všechny CRUD operace a pro prototyp budou dostačující.

Při implementaci kontrolérů jsem se rozhodl použít vlastní rozhraní *IController<DTO>*, jež implementuje některé třídy kontrolérů. Toto rozhraní deklaruje základní endpointy, které umožňují provádět základní CRUD operace nad entitami v aplikaci, které jsou pak implementovány ve třídách. Níže je uveden kód.

```
1 public interface IController<DTO>{
2     ResponseEntity<List<DTO>>   all();
3     ResponseEntity<DTO>         byID(Long id);
4     ResponseEntity<DTO>         updateById(Long id, DTO dto);
5     ResponseEntity<DTO>         create(DTO dto);
6     void                        delete(Long id);
7 }
```

Níže je uvedena třída, která implementuje rozhraní *IController*. Pro demonstrační účely jsem implementoval dva endpointy – požadavek GET jsem použil pro získání jednoho dotazníku, požadavek POST pro přidaní dotazníku. Anotace *@RestController* je nutná k tomu, aby Spring framework poznal, že se jedná o kontrolér a že může přijímat požadavky, tím pádem tyto požadavky je třeba namapovat. Anotace *@CrossOrigin* vypíná bezpečnostní politiku CORS, u reálného produktu by se toto provádět nemělo. Požadavek je autorizován díky anotaci *@PreAuthorize*. Anotace *@GetMapping* a *@PostMapping* definují typ požadavku a jako parametr přebírají cestu požadavku. Tuto cestu lze také nastavit, aby přijímala parametry, v mém případě přijímá id dotazníku při požadavku GET.

```
1  @RestController
2  @CrossOrigin
3  public class SurveyController implements IController<SurveyDTO> {
4
5      ...
6
7      @Override
8      @PreAuthorize("hasAnyAuthority('ADMIN', 'MANAGER', 'USER')")
9      @GetMapping("/surveys/{id}")
10     public ResponseEntity<SurveyDTO> byID(@PathVariable Long id) {
11         try {
12             Optional<Survey> survey = surveyService.getSurveyById(id);
13             if(survey.isEmpty()) throw new NotExistsException("Survey"↩
                   );
14             return new ResponseEntity<>(surveyMapper.toDTO(survey.get↩
                   ()), HttpStatus.OK);
15         } catch (Exception e) {
16             throw new ResponseStatusException(HttpStatus.NOT_FOUND);
17         }
18     }
19 }
```

Odpověď na požadavek *GET /users/id* je uvedena níže.

```
{
    "Id": 1,
    "username": "mannaayd",
    "email": "mannaayd@cvut.cz",
    "password": null,
    "rank": 1,
    "region": "Prague",
    "city": "CZ",
    "role": "ADMIN",
    "questionIds": [
        12
    ],
    "surveyIds": [
        11
    ],
    "answerIds": []
}
```

5.1.6 Bezpečnost

Používání takové aplikace by samozřejmě mělo být pro uživatele bezpečné, jelikož krádeže osobních údajů nejsou v dnešní době ničím neobvyklým. Z tohoto důvodu je na backendu implementováno hashování hesel, které udržuje uživatelské heslo v bezpečí.

Samotná aplikace také odděluje uživatelské role pomocí Spring Security. Má aplikace disponuje třemi rolemi: ADMIN, MANGER a USER. Administrátor má přístup ke všemu, správce může vytvářet dotazníky, přidávat do nich otázky, uzavírat je a získávat z nich výstupy. Uživatel může pouze na tyto dotazníky odpovídat.

Níže je uvedena část kódu, která heslo zahashuje a následně přidá nového uživatele do databáze.

```
@Override
@PostMapping("/register")
public ResponseEntity<UserDTO> create(@RequestBody UserDTO userDTO) {
    try {
        String password = userDTO.getPassword();
        userDTO.setPassword(passwordEncoder.encode(password)); // ↩
            hashing password
        userDTO.setRank(1);
        userService.save(userDTO);
        userDTO.setPassword("");
        return new ResponseEntity<>(userDTO, HttpStatus.OK);
    } catch (Exception e) {
        throw new ResponseStatusException(HttpStatus.NOT_FOUND);
    }
}
```

Uživatel se musí před použitím aplikace autentizovat. Za tímto účelem musí zadat své uživatelské jméno a heslo a odeslat požadavek *POST /login*, pokud

nebudou údaje správné, server zahlásí chybu 401 Unauthorized, v případě správného zadání se objeví kód 200 OK, dále zde bude obsažen token JWT, který je potřebný k autorizaci dalších požadavků v aplikaci.

Níže je uvedena odpověď serveru na požadavek na přihlášení.

```
{
    "jwt": "eyJ0e ... 00hCA",
    "id": 1,
    "username": "mannaayd",
    "email": "mannaayd@cvut.cz",
    "role": "ADMIN"
}
```

Následující metoda z *LoginControlleru* získá přihlašovací jméno a heslo a porovná je s údaji v databázi. Pokud toto bude úspěšné, uživatelská data budou přidána do hashmapy. Tato mapa je zakódována do JWT podepsáním klíčem RSA.

```
@PostMapping("/login")
    public ResponseEntity<JwtDTO> login(@RequestBody LoginFormDTO loginForm) {

        UserDetails userDetails;
        try {
            userDetails = userDetailsService.loadUserByUsername(loginForm.getUsername());
        } catch (UsernameNotFoundException e) {
            throw new ResponseStatusException(HttpStatus.UNAUTHORIZED,"User not found");
        }
        if (passwordEncoder.matches(loginForm.getPassword(), userDetails.getPassword())) {
            // define fields map for claims
            Map<String, String> claims = new HashMap<>();
            // setting roles
            String authorities = userDetails.getAuthorities().stream()
                    .map(GrantedAuthority::getAuthority)
                    .collect(Collectors.joining(" "));
            // adding fields into encode json
            claims.put("username", loginForm.getUsername());
            claims.put(WebSecurityConfig.AUTHORITIES_CLAIM_NAME, authorities);
            claims.put("userId", String.valueOf(1));
            Optional<User> user = userService.getUserByUsername(userDetails.getUsername());
            // create jwt using JwtHelper component
            String jwt = jwtHelper.createJwtForClaims(loginForm.getUsername(), claims);
            return new ResponseEntity<>(new JwtDTO(jwt, user.get().getId(), user.get().getUsername(), user.get().getEmail(), user.get().getRole()), HttpStatus.OK);
        }
        throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "User not authenticated");
    }
```

Klíč JWT je vytvořen v níže uvedené komponentě *JwtHelper.java*.

```
1  @Component
2  public class JwtHelper {
3      private final RSAPrivateKey privateKey;
4      private final RSAPublicKey publicKey;
5
6      ...
7
8      public String createJwtForClaims(String subject, Map<String, ↩
           String> claims) {
9          Calendar calendar = Calendar.getInstance();
10         calendar.setTimeInMillis(Instant.now().toEpochMilli());  // ↩
               get today date in epoch format
11         calendar.add(Calendar.DATE, 1); // setting expire time for one↩
                day
12         // create jwt builder
13         JWTCreator.Builder jwtBuilder = JWT.create().withSubject(↩
               subject);
14         claims.forEach(jwtBuilder::withClaim);
15         // sign and return jwt token
16         return jwtBuilder
17                 .withNotBefore(new Date())
18                 .withExpiresAt(calendar.getTime())
19                 .sign(Algorithm.RSA256(publicKey, privateKey));
20     }
21 }
```

Všechny klíče jsou uloženy ve složce *src/main/resources/keys* a git u nich nesleduje změny. Cesty k těmto klíčům jsou označeny v souboru *application.properties*.

5.2 Frontend

Další částí implementace je frontend. Ten slouží jako interaktivní vizuální prezentace dat pro uživatele. Hlavním úkolem frontendu je vykreslovat stránky HTML. K interaktivitě stránek se používá JavaScript. Frontend zároveň také musí komunikovat s backendem.

Ve své práci jsem se rozhodl používat nejmodernější technologie pro vývoj frontendů. Základem frontendu je framework ReactJS, který se používá k vytváření interaktivních aplikací SPA. Rozhodl jsem se napsat frontend v jazyce TypeScript, abych se v budoucnu vyhnul problémům se škálovatelností kvůli dynamickému typování. Pro pokročilou komunikaci HTTP jsem použil balíček axios. V budoucnu bude možné pomocí tohoto balíčku přidat podporu pro stahování a nahrávání souborů do backendu. Pro přidání vizuálních prvků do prototypu používám balíček s komponentami stylu Bootstrap.

5.2.1 Struktura frontendu

Struktura souborů frontendu je poměrně jednoduchá, konfigurační soubory frontendu jsou v kořenovém adresáři. Soubor *package.json* obsahuje všechny závislosti balíčků pro NPM. Konfigurace jazyka TypeScript probíhá v souboru *tsconfig.json*. Všechny příkazy pro sestavení kontejneru Docker jsou uvedeny

Obrázek 5.2: Adresářová struktura frontendu.

- `node-modules` ..... stažené potřebné balíčky
- `public` ..... veřejné soubory
- `src` ..... zdrojové kódy
  - `components` ..... komponenty
  - `services` ..... služby
  - `types` ..... typy
  - `App.css` ..... stylování stránek
  - `App.tsx`. ..... hlavní komponent
  - `index.css` ..... stylování stránek
  - `index.tsx` ..... entrypoint aplikace
  - `react-app-env.d.ts` ..... proměnné prostředí
- `scripts` ..... skripty pro sestavení projektu
- `.gitignore` ..... seznam souborů, které má Git ignorovat
- `Dockerfile` ..... konfigurace kontejneru Docker
- `package.json` ..... seznam závislostí pro NPM
- `package-lock.json` ..... úplný seznam závislostí pro NPM
- `README.md` ..... pokyny
- `tsconfig.json` ..... konfigurace TypeScript

v souboru Dockerfile. Složka *src* obsahuje všechny zdrojové soubory ReactJS jako jsou komponenty, typy, služby atd. Následující obrázek 5.2.1 znázorňuje strukturu souborů frontendu.

5.2.2 Component

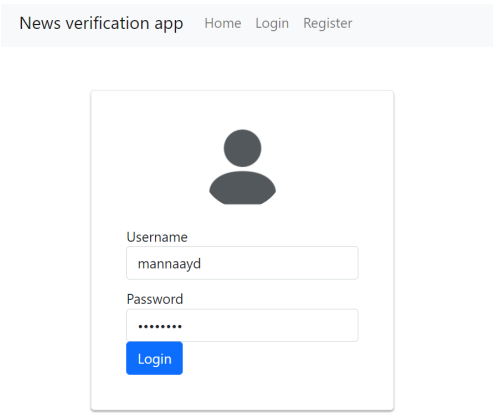
Komponenta React je funkcí vracející prvek React nebo třídu JavaScriptu implementující metodou *render()*, která navrací prvek React. Komponenty deklarované pomocí funkce jazyka JavaScript nebo TypeScript se proto nazývají „funkční komponenty“, komponenty deklarované pomocí tříd jazyka JavaScript se nazývají „komponenty z tříd“. Pro vytvoření komponenty z třídy je třeba zdědit třídu z React třídy *Component<Props, State>*, kde Props je typ atributu komponenty a State je typ stavu komponenty.

Ve své aplikaci používám pouze komponenty z tříd. Komponenty mají v aplikaci velmi důležitou funkci, pomáhají oddělit vizuální komponenty stránky a pracovat s každou z nich zvlášť.

V rámci tohoto projektu jsem rozdělil komponenty do tří typů: stránky, desky a karty. Ty budou popsány v následujících kapitolách.

Níže je uveden příklad kódu komponenty ve formě třídy.

```
1  import { Component } from "react";
2  import UserService from "../services/UserService";
3  type State = { // state of component
4    content: string
5    text: string
6  }
7
8  type Props = {} // params to pass by
9
10 export default class Home extends Component<Props, State> {
11   constructor(props: Props) {
12     super(props);
13     this.state = {
14       content: "News verification app",
15       text: "This is a prototype news verification app. This app pulls↩
             tweets from twitter and making surveys."
16     };
17   }
18   componentDidMount() {} // calling when component is attached to ↩
       another component
19   render() { // render element
20     return (
21       <div className="container">
22         <header className="jumbotron">
23           <h3>{this.state.content}</h3>
24         </header>
25         {this.state.text}
26       </div>
27     );
28   }
29 }
```

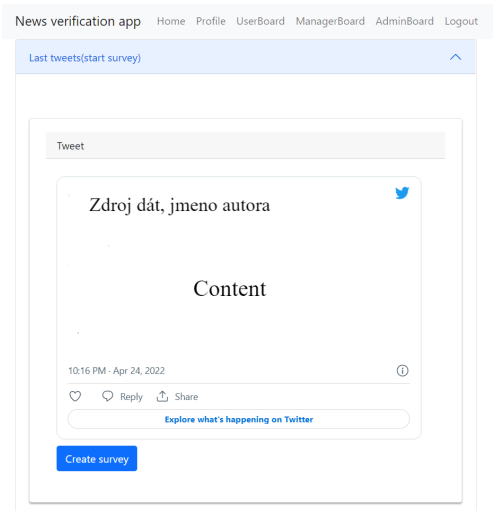


Obrázek 5.3: LoginComponent

5.2.3 Board

Komponenty desky jsou potřebné k zobrazení prvků uživatelského rozhraní, které jsou přístupné pouze uživatelům s příslušnou rolí. V mé aplikaci se

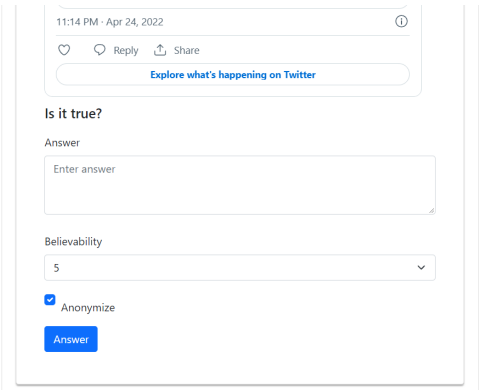
nachází tři typy desky: pro uživatele, pro správce a pro administrátora. Na desce uživatele je možnost odpovědět na otázky, pokud nějaké jsou. Deska správce umožňuje vytvářet dotazníky, přidávat do nich další otázky a získávat z nich výstupy. V desce administrátora lze zobrazit všechny uživatele. Na obrázku 5.4 je zobrazena část desky pro správce.



Obrázek 5.4: TweetCard uvnitř ManangerBoard komponenty

5.2.4 Card

Karta je v podstatě komponentním prvkem, který má na stránce HTML nějakou funkci. V mém případě existují čtyři typy karet: pro otázku, pro dotazování, pro tweet a pro uživatele. Níže v 5.5 je uveden příklad karty otázky pro uživatele.



Obrázek 5.5: QuestionCard komponenta

5.2.5 Service

Služby jsou třídy, které jsou potřebné ke zpracování různých příkazů. Účely těchto služeb jsou velmi různorodé.

Například pro zpracování přihlášení, odhlášení a registrace používám službu *AuthService* s metodami *login*, *logout* a *register*. Dále také používám službu *TweetService*, která disponuje metodou pro získání posledních tweetů.

Níže je uveden kód služby *QuestionService*, která slouží k získání informací o otázce v dotazníku a také k přidání nové otázky.

```
1  class QuestionService {
2      getQuestionTweet(id?: number) {
3          // authHeader is a method that gets JWT token from local ↩
               storage and puts it in header
4          return axios.get(API_URL + 'questions/' + id + '/tweet',  ↩
               authHeader())
5      }
6      // add new Question
7      postQuestion(question: Question) {
8          // returns Promise of http response
9          return axios.post(API_URL + 'questions', question, authHeader↩
               ())
10     }
11 } export default new QuestionService;
```

5.3 Nasazení aplikace

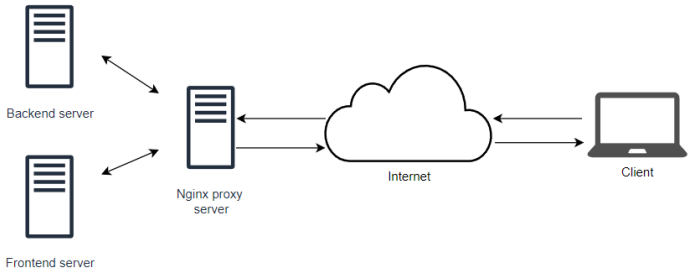
Další částí implementace je nasazení, v této části jde o nastavení aplikace a prostředí, ve kterých budou jednotlivé časti aplikace běžet, a především o nástrojích, které tyto funkce poskytují.

5.3.1 Nginx

Možností, jak použít systém Nginx je opravdu hodně. Já ji používám jako proxy server, což znamená, že jeho úkolem bude přesměrovávat požadavky a odpovědi klientů na server. Níže je uvedena konfigurace *default.conf* Nginxu.

```
1  server{
2      listen 80;
3
4      location /api/ {
5          proxy_pass http://backend:8080/ ;
6      }
7
8      location / {
9          proxy_pass http://frontend:3000 ;
10     }
11 }
```

Princip tohoto serveru je znázorněn na obrázku 5.6.



Obrázek 5.6: Nginx proxy server

5.3.2 Docker-compose

Docker-compose slouží ke spuštění a konfiguraci všech Docker kontejnerů najednou. Pomocí jediného souboru *docker-compose.yml* lze nasadit Docker kontejnery frontend, backend, proxy a databázi.

Pro sestavení kontejnerů je třeba spustit příkaz *docker-compose build*. Pro spuštění všech kontejnerů je nutné spustit příkaz *docker-compose up*. Pro ukončení je třeba spustit příkaz *docker-compose down*.

Níže je uvedená část souboru *docker-compose.yml*.

```
1
2       ...
3
4   frontend:
5     container_name: frontend
6     build:
7       context: ./frontend
8       dockerfile: Dockerfile
9     ports:
10      - "3000:3000"
11    restart: unless-stopped
12    environment:
13      CHOKIDAR_USEPOLLING: "true"
14    volumes:
15      - ./frontend/node_modules:/usr/app/node_modules
16      - ./frontend:/usr/app
17
18      ...
```

Jsou vytvořeny Docker kontejnery pro provoz frontendu, backendu, proxy serveru a databáze. Pro komunikaci se zbytkem aplikace má každý kontejner svůj vlastní port. Příkazy pro sestavení frontendů a backendů se nacházejí v souborech Dockerfile. Tyto soubory používají hotové obrazy z DockerHubu, například pro backend používám obraz *gradle:jdk17-alpine*, který pomáhá při spouštění aplikace Gradle.

Dockerfile pro backend.

```
FROM gradle:jdk17-alpine
WORKDIR /opt/app
COPY . .
RUN gradle build -x test
ENTRYPOINT ["java","-jar","build/libs/backend-0.0.1-SNAPSHOT.jar"]
```

Dockerfile pro frontend.

```
FROM node:alpine
WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH
COPY package.json ./
COPY package-lock.json ./
COPY tsconfig.json ./
RUN npm i
COPY . ./
# Build for production.
RUN npm run build --production
# Install static server
RUN npm install -g serve
# Run application production
CMD serve -s build
```

Kapitola 6

Uživatelské testování

Další kapitola se zaměřuje na uživatelské testování. V této kapitole budou prezentovány výsledky provedeného testování aplikace. Jako vstupní zdroje pro testování jsem použil dva skutečné zpravodajské příspěvky zveřejněné na sociální síti Twitter, které byly vybrány náhodně z příspěvků maximálně dva měsíce starých od okamžiku provedení náhodného výběru.

Po sběru potřebných dat se uskutečnilo dotazníkové šetření, kterého se účastnili předem vybraní respondenti. Jejich volba byla založena na stanovených kritériích.

Na základě získaných dat správce webu rozhodl o pravdivosti zkoumaného příspěvku. Následně jeho verdikt bude prezentován v rámci výsledků.

6.1 Výběr zpráv

Pro testování byly vybrány dvě zprávy z Twitteru. U jedné ze zpráv předpokládám, že se její pravdivost potvrdí, u druhé ne. Nicméně je také potřeba zmínit, že Twitter již disponuje vlastními mechanismy pro vstupní filtraci zpráv za účelem potvrzení jejich pravdivosti. Vysoká funkčnost těchto mechanismů výrazným způsobem komplikovala hledání zprav, u kterých by se jejich pravdivost mohla zpochybnit. A to z toho důvodu, že se nepravdivé zprávy se nedostávaly ke konečnému uživateli.

Jako potenciálně falešná zpráva byla zvolena zpráva o „vylučování“ vysokoškolských studentů z evropských univerzit, jež pocházejí z Ruska a Běloruska. Volba této konkrétní zprávy byla rovněž ovlivněna osobním zájmem o prozkoumání možných důsledků dané zprávy vůči této skupině, ke které jako ruský student ČVUT patřím. Vzhledem k tomu, že k okamžiku psaní bakalářské práce nebyla v tomto kontextu zaznamenána žádná oficiální opatření, vedlo to ke vzniků oprávněných pohybností ohledně pravdivosti této zprávy.

Pro ověření pravdivosti této zprávy jsem se obrátil na oficiální zdroj ale i na cílenou skupinu, jíž se zpráva týkala. Jednak jsem vycházel z veřejných vyjádření představitelů předních vysokých škol nacházejících se v České re-

publice a jednak jsem provedl dotazníkové šetření mezi studenty českých VŠ ruského a běloruského původu, kteří se mohli podělit o své vlastní zkušenosti. V rámci dotazování byl použit prototyp aplikace.

Výsledky provedeného testování jsou součástí kapitoly 7.

6.2 Zpráva: „vylučování ruských a běloruských studentů z evropských VŠ“.

Obrázek 6.1: Potenciálně falešná zpráva

6.2.1 Veřejná vyjádření oficiálních představitelů

Vznik této nepravdivé zprávy je nepochybně spojen s dřívějšími případy šikany vůči studentům ruské národnosti. V této návaznosti se představitelé vlády ČR postavili proti potenciální diskriminaci na základě národnosti. Dne 25. února 2022 vyzval ministr školství Petr Gazdík žáky, studenty a učitele, aby se vyvarovali šikaně nebo slovních útoků proti osobám původem z Ruska. ”V naší zemi lidská práva dodržujeme, respektujeme individualitu a jen pouhá příslušnost k národnosti nesmí být záminkou ke konfliktům,”řekl v této souvislosti ministr Gazdík.[41]

Výše popsaný příklad tedy nepotvrzuje změnu pozice České republiky vůči ruským studentům. Proto dle mého názoru tato konkrétní zpráva o „vylučování“ vyžadovala prověření.

Dne 22. dubna 2022 se k dané problematice v rámci rozhlasového pořadu týdeníku Reflex o aktuálním dění vyjádřil emeritní rektor Univerzity Karlovy prof. MUDr. Tomáš Zima, DrSc., MBA, dr. h. c. mult.[42]. „Je to typický „fake news“, lživá zpráva“, řekl v pořadu emeritní rektor Univerzity Karlovy Tomáš Zima. „Na univerzitě studují lidé z více než 100 zemí světa. Univerzita Karlova, a i jiné české vysoké školy jsou otevřené studentům všech národností.“ Na základě daného prohlášení, a také vzhledem k vysokému postavení prof. MUDr. Tomáše Zimy, DrSc. v rámci univerzity, by se proto případná „vyloučení“ neměla očekávat.

6.2.2 Dotazníkové šetření

V rámci výzkumu jsem zvolil metodu dotazníkového šetření. Odpovědi jednotlivých respondentů jsou zaneseny v tabulce níže. Položené otázky jsou uzavřeného typu s použitím modifikované Likertovy škály. Jedná se o techniku používanou pro měření postojů v dotaznících. Tato metoda byla vytvořena americkým psychologem Rensisem Likertem. Běžný počet odpovědí v Likertově škále je mezi 5 až 7. V rámci mého šetření si respondenti mohli vybírat z 5 možností, kde volba 1 znamenala zásadní nesouhlas, a volba 5 – největší míru souhlasu – tabulka 6.1. Na začátku šetření byla respondentu položena otázka, zda vůbec o zprávě slyšel. Tímto způsobem bylo určováno, zda je další dotazování vhodné.[43]

1	2	3	4	5
Zásadně nesouhlasím	Spíše nesouhlasím	Nevím	Spíše souhlasím	Naprosto souhlasím

Tabulka 6.1: Použitá Likertova škála

Účastník	Země původu	Země studia	Vysoká škola	Datum dotazování	Věk
Respondent 1	Bělorusko	Česko	ČVUT	22.03.2022	24
Respondent 2	Rusko	Česko	ČVUT	02.04.2022	22
Respondent 3	Rusko	Česko	ČVUT	04.04.2022	23

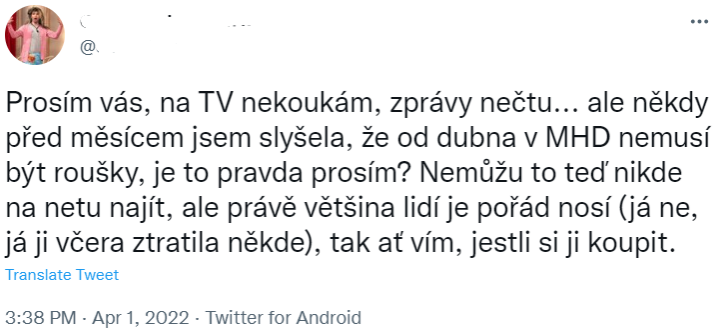
Tabulka 6.2: Souhrnné informace o respondentech 1, 2, 3

První dotazování bylo provedeno tři týdny po 1. březnu 2022, to znamená tři týdny po datu zveřejnění dané zprávy. To považuji za dostatečně dlouhou dobu pro zaznamenání případných změn. Další dotazníky byly vyplňovány 2. a 4. dubna 2022.

Slyšel/la jste o tom, že se hrozí „vylučování“ ruských a běloruských studentů?	
Respondent 1	Ano
Respondent 2	Ano
Respondent 3	Ano
Setkáváte se osobně s diskriminací vzhledem ke svému původu?	
Respondent 1	1
Respondent 2	1
Respondent 3	2
Ruští/Běloruští studenti jsou diskriminování na základě svého původu.	
Respondent 1	1
Respondent 2	1
Respondent 3	2
Po vypuknutí válečného konfliktu se bojím, že bude vyloučen z VŠ.	
Respondent 1	2
Respondent 2	1
Respondent 3	2
Moje studijní podmínky jsou stejné jako před změnou politické situace.	
Respondent 1	5
Respondent 2	5
Respondent 3	4

Tabulka 6.3: Odpovědi respondentů

6.3 Zpráva: „Zrušení povinnosti nosit roušky v prostředích hromadné dopravy od 14.04.2022“.



Obrázek 6.2: Potenciálně pravdivá zpráva

V první polovině dubna 2022 se v médiích psalo o brzkém zrušení povinnosti mít zakrytá ústa v prostředcích hromadné dopravy. Covidová opatření se

ale zavádí na základě vývoje epidemické situace, která není z obecně známých důvodů předvídatelná. Proto jakákoli vyjádření ze strany subjektů, které nepatří mezi oficiální představitele příslušných resortů, nevyvolávají nezpochybnitelnou důvěru. Oficiální stanovisko na webu Ministerstva zdravotnictví bylo umístěno až necelé dva týdny po objevení se těchto zpráv v médiích. Vzhledem k tomu, že nebylo možné získat kompletní informace, protože se týkaly stále nezveřejněných vládních rozhodnutí, tak jsem provedl výzkum postojů ohledně pravdivosti dané zprávy ihned po jejím zveřejnění.

6.3.1 Veřejná vyjádření oficiálních představitelů.

Jak již bylo zmíněno v předchozím odstavci, nebylo možné ověřit pravdivost zprávy z oficiálních zdrojů, jelikož nebyla ještě zveřejněna. Nicméně potvrzení této zprávy na sebe nenechalo dlouho čekat, dne 14. dubna 2022 bylo na COVID PORTÁLU, který je spravován Ministerstvem zdravotnictví, publikováno upravující opatření o povinnosti mít zakrytá ústa a noc. Prostory hromadné dopravy byly vyjmuty ze seznamu míst, kde tato povinnost platí.[44]

6.3.2 Dotazníkové šetření

Účastník	Země původu	Země pobytu	Používáte MHD	Datum dotazování	Věk
Respondent A	Česko	Česko	Ano	03.04.2022	28
Respondent B	Česko	Česko	Ano	05.04.2022	24
Respondent C	Rusko	Česko	Ano	07.04.2022	25

Tabulka 6.4: Souhrnné informace o respondentech A, B, C

Pro zapojení do šetření museli respondenti splňovat určitá kritéria. Zaprvé, museli mít trvalý pobyt na území ČR. Za druhé, museli používat prostředky hromadné dopravy, jelikož se právě jich zpráva bezprostředně týkala.

Slyšel/la jste o tom, že se chystá zrušení povinnosti nosit roušky v prostředcích hromadné dopravy?	
Respondent A	Ano
Respondent B	Ne
Respondent C	Ano
Myslím si, že rozvolnění proběhne v brzké době.	
Respondent A	2
Respondent B	2
Respondent C	1
Jsem si jistý, že zprávy podobného typu podléhají předchozímu prověření..	
Respondent A	2
Respondent B	1
Respondent C	2
Mám předchozí zkušenosti s dezinformacemi kolem epidemická situace v ČR.	
Respondent A	4
Respondent B	5
Respondent C	4
Spoléhám se na pravdivost zpráv, které zasahují celou společnost.	
Respondent A	1
Respondent B	2
Respondent C	1

Tabulka 6.5: Odpovědi respondentů

Kapitola 7

Vyhodnocení výsledků

V předchozích kapitolách byla navržena a implementována aplikace na základě uskutečněné analýzy již existujících alternativních řešení pro ověřování pravdivosti zpráv. Jedním z hlavních vypozorovaných rozdílů byl odlišný přístup ke sběru dat. Detailněji je to popsáno v podkapitole 7.2.

7.1 Zhodnocení návrhu aplikace

Návrh aplikace News Verification App je odvozen od analýzy již existujících řešení totožné problematiky popsané v kapitole 5. V této části byly navrženy případy použití aplikace pro dosažení cíle bakalářské práce.

7.2 Zhodnocení implementace

Technologie, které byly zvoleny pro implementaci, byly úspěšně aplikovány na prototyp aplikace. Avšak doménový model zvolený v návrhu aplikace neumožňoval realizovat všechny plány. Celá implementace prototypu aplikace je k dispozici na přiložené SD-kartě, jež je součástí této bakalářské práce.

Během přípravy aplikace byl použit princip SoC, tím pádem jsem rozdělil aplikaci na backend a frontend část. V části backend se mi podařilo vytvořit API, celá dokumentace se nachází na webovém odkazu[38]. Backend mimo jiné bezprostředně komunikuje s webovým portálem Twitter, a to skrze Twitter API. Část frontend odpovídá za celé uživatelské rozhraní pro komunikaci s backendem. Mezi další úkoly frontendu také patří vyhledávání nových zpráv, které vyžadují prověření. Dále frontend odpovídá za zasílání dotazníků respondentům vybraných na základě předem zvolených kritérií. Získané odpovědi se vracejí v datovém formátu JSON. Následně jsou použity pro posuzování pravdivosti zpráv.

7.3 Návrhy k vylepšení

Při testování aplikace byly zaznamenány okamžiky, které by vyžadovaly přidání dalších funkcí. Pro lepší přehlednost jsem vyjmenoval další možnosti rozvoje aplikace. Mezi mnou navrhovaná vylepšení patří:

1. Pokročilé hodnocení míry důvěryhodnosti uživatelů
2. Vytváření a následná správa skupin uživatelů
3. Zapojení dalších zdrojů zpráv
4. Prověření splnění jednotlivých kritérií uživatelem
5. Podpora dalších formátů vkládaných ze strany respondentů dat
6. Podpora různých typů otázek
7. Ověření podezřelých aktivit na uživatelských účtech pomoci algoritmu umělé inteligence
8. Přidání možnosti získat symbolickou odměnu za poskytnutou pomoc při ověřování pravdivosti zpráv

7.4 Zhodnocení provedeného testování

V kapitole 6 jsou k dispozici výsledky provedeného testování pravdivosti. Pravdivost první zprávy o „Vylučování ruských a běloruských studentů z evropských VŠ“ byla vyvrácena jak ze strany oficiálního představitele tak i ze strany případných osob, kterých se daná zpráva týkala. Nikdo z dotázaných nevypozoroval diskriminaci vůči své osobě na základě svého původu. Taktéž žádný z respondentů nezaznamenal zhoršení studijních podmínek po vypuknutí válečného konfliktu. Vzhledem k absenci reálných následků a oficiálnímu vyjádření emeritního rektora Univerzity Karlovy si dovoluji tvrdit, že danou zprávou lze považovat za „fake news“.

Druhá zpráva o „Zrušení povinnosti nosit roušky v prostředích prostředcích hromadné dopravy, dubna 2022 “ vyžadovala informace od představitelů příslušného ministerstva, která ale byla zveřejněna na COVID PORTÁLU až dva týdny po objevení se zprávy v médiích. Mezitím bylo provedeno dotazníkové šetření, na jehož základě byl zaznamenán postoj respondentů vůči pravděpodobnosti takového rozvolnění opatření. Získaná data od dotazovaných nenaznačovala jejich nedůvěru ve zprávu. Jako potvrzení věrohodnosti, které nevyžadovalo další prověření, sloužil příspěvek Ministerstva zdravotnictví ze dne 14. dubna 2022, který skutečně upravoval místa s povinností mít zakrytá ústa a nos.

Závěr

Hlavním cílem bakalářské práce bylo navrhnout a implementovat webovou aplikaci pro verifikaci zpráv. Výslednou aplikaci hodnotím jako úspěšnou až na menší výhrady, které jsem popsal v kapitole 7.

Vytvořená aplikace umožňuje uživatelům ověřovat zprávy ze sociální sítě Twitter. Při vývoji této aplikace byly využity dnes populární programovací jazyky. Backend byl napsán v jazyce Java s využitím frameworku Spring. Frontend byl napsán v jazyce JavaScript s frameworkem ReactJS a stylováním stránek pomocí Bootstrap, což je knihovna komponentů pro stylování klientských aplikací. Komunikace s aplikací probíhá pres proxy server Nginx. Pro ukládaní všech potřebných dat byla použita moderní relační databázi Postgres. Celá aplikace je nasazena na kontejnerizační nastroj Docker.

Během implementace backendu jsem narazil na několik problémů při komunikaci s API Twitteru. Zkoušel jsem použít všechny knihovny z MVN repositářů pro práci s tímto API. Nakonec jsem ale rozhodl udělat vlastní službu pro přímou komunikaci s API Twitteru. Celá dokumentace tohoto API je veřejně dostupná na stránkách portálu Postman.[38]

Při vytváření frontend časti jsem používal nejmodernější technologie pro vývoj. Mezi ně patří TypeScript, což je dnes standardně používaný nástroj při vytváření podobných projektů, které dříve byly obvykle psány v jazyce JavaScript. S použitím frameworku ReactJS se podařilo vytvořit flexibilní a rychlý frontend s možností budoucího rozšíření funkcí.

Po důkladné implementace následovalo uživatelské testování prototypu aplikace. Během tohoto testování byla potvrzena funkčnost dané aplikace pro účely ověřování pravdivosti zpráv. Považuji za nezbytné rozšířit prototyp o další typy otázek, což by umožnilo sběr relevantnějších dat.

Cíl stanovený na začátku této bakalářské práce považuji za splněný. Daná práce může být použita pro další vývoj ve směru zvyšování důvěryhodnosti zpravodajských příspěvků ve webovém prostoru. Tím pádem, dokáže se podílet na vyřešení nejen problému, popsanému v této práci, ale i na dalších problémech, spojených s šířením dezinformací.

Literatura

[1] MVC - MDN Web Docs Glossary: Definitions of Web-related terms — MDN [online]. [cit. 2022-04-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[2] Diffrence between JPA API and hibernate native API [online]. [cit. 2022-04-18]. Dostupné z: https://stackoverflow.com/questions/49475385/diffrence-between-jpa-api-and-hibernate-native-api

[3] Merriam-Webster. (n.d.). Misinformation. In Merriam-Webster.com dictionary [online]. [cit. 2022-04-12]. Dostupné z: https://www.merriam-webster.com/dictionary/misinformation

[4] Gebel, M.: Misinformation vs. disinformation: What to know about each form of false information, and how to spot them [online]. [cit. 2022-04-12]. Dostupné z: https://www.businessinsider.com/misinformation-vs-disinformation

[5] Merriam-Webster. (n.d.). Disinformation. In Merriam-Webster.com dictionary [online]. [cit. 2022-04-13]. Dostupné z: https://www.merriam-webster.com/dictionary/disinformation

[6] How to Spot Real and Fake News [online]. [cit. 2022-04-12]. Dostupné z: https://www.mindtools.com/pages/article/fake-news.htm

[7] Bot Sentiel About page [online]. [cit. 2022-04-13]. Dostupné z: https://botsentinel.com/info/about

[8] Bot Slayer [online]. [cit. 2022-04-13]. Dostupné z: https://osome.iu.edu/tools/botslayer

[9] Alt News About page [online]. [cit. 2022-04-13]. Dostupné z: https://www.altnews.in/about/

[10] Most popular social networks worldwide as of January 2022, ranked by number of monthly active users [online]. [cit. 2022-04-14]. Dostupné z: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[11] Carlson, N.: The full story of how Facebook was founded [online]. [cit. 2022-04-14]. Dostupné z: https://www.businessinsider.com/how-facebook-was-founded-2010-3

[12] Upbin, B.: Facebook Buys Instagram For $1 Billion. Smart Arbitrage [online]. [cit. 2022-04-14]. Dostupné z: https://www.forbes.com/sites/bruceupbin/2012/04/09/facebook-buys-instagram-for-1-billion-wheres-the-revenue/?sh=76525c8b4b8a

[13] What is a microblog? [online]. [cit. 2022-04-14]. Dostupné z: https://sproutsocial.com/glossary/microblog/

[14] Dizikes, P.: Study: On Twitter, false news travels faster than true stories [online]. [cit. 2022-04-14]. Dostupné z: https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308

[15] About The Unified Modeling Language Specification Version 2.5.1 [online]. [cit. 2022-04-15]. Dostupné z: https://www.omg.org/spec/UML/

[16] Git SCM page [online]. [cit. 2022-04-30]. Dostupné z: https://git-scm.com/

[17] Docker Desktop overview [online]. [cit. 2022-04-30]. Dostupné z: https://docs.docker.com/desktop/

[18] Overview of Docker Compose [online]. [cit. 2022-04-30]. Dostupné z: https://docs.docker.com/compose/

[19] Nginx Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://docs.nginx.com/

[20] Java Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://docs.oracle.com/en/java/

[21] Spring Framework 5.3.19 [online]. [cit. 2022-04-18]. Dostupné z: https://spring.io/projects/spring-framework

[22] Spring Boot [online]. [cit. 2022-04-30]. Dostupné z: https://spring.io/projects/spring-boot

[23] Gradle Build tool [online]. [cit. 2022-04-30]. Dostupné z: https://gradle.org/

[24] ISO/IEC 9075-1:2016 Information technology — Database languages — SQL [online]. [cit. 2022-04-30]. Dostupné z: https://www.iso.org/standard/63555.html#:~:text=ISO%2FIEC%209075%2D1%3A2016%20describes%20the%20conceptual%20framework,parts%20of%20ISO%2FIEC%209075

[25] PostgreSQL 14.2 Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://www.postgresql.org/docs/

[26] Introduction to the Java Persistence API [online]. [cit. 2022-04-30]. Dostupné z: https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html

[27] Hibernate ORM [online]. [cit. 2022-04-30]. Dostupné z: https://hibernate.org/orm/documentation/6.0/

[28] Spring Security [online]. [cit. 2022-04-30]. Dostupné z: https://docs.spring.io/spring-security/reference/index.html

[29] Introduction to Spring Web MVC framework [online]. [cit. 2022-04-30]. Dostupné z: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

[30] RFC 7519 - JSON Web Token (JWT) [online]. [cit. 2022-04-18]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7519

[31] NPM package manager [online]. [cit. 2022-04-30]. Dostupné z: https://www.npmjs.com/

[32] JavaScript Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/web/javascript

[33] Axios Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://axios-http.com/docs/intro

[34] TypeScript Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://www.typescriptlang.org/docs/

[35] React Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://en.reactjs.org/docs/getting-started.html

[36] React Bootstrap Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://react-bootstrap.github.io/

[37] React Twitter Embed Package Documentation [online]. [cit. 2022-04-30]. Dostupné z: https://www.npmjs.com/package/react-twitter-embed

[38] News verification app API v1 [online]. [cit. 2022-04-14]. Dostupné z: https://documenter.getpostman.com/view/18054820/UVz1MCBi

[39] Fowler, M.: Inversion of Control Containers and the Dependency Injection pattern [online]. [cit. 2022-04-21]. Dostupné z: https://www.martinfowler.com/articles/injection.html

[40] Twitter API v2 tools libraries [online]. [cit. 2022-04-21]. Dostupné z: https://developer.twitter.com/en/docs/twitter-api/tools-and-libraries/v2

[41] Drgáč, Z.: Učitel z VŠE útočil na ruské studenty. Ministr školství nenávist odsoudil [online]. [cit. 2022-04-29]. Dostupné z: https://tn.nova.cz/zpravodajstvi/clanek/455756-ucitel-z-vse-utocil-na-ruske-studenty-ministr-skolstvi-nenavist-ve-skolach-odsoudil?campaignsrc=tn_clipboard

[42] Zita Senková, T. Z.: Tomáš Zima: České školy jsou otevřené studentům všech národností. Zpráva o vyhozené ruské studentce je lež [online]. [cit. 2022-04-29]. Dostupné z: https://dvojka.rozhlas.cz/tomas-zima-ceske-skoly-jsou-otevrene-studentum-vsech-narodnosti-zprava-o-8732629

[43] McLeod, D. S.: Likert Scale Definition, Examples and Analysis [online]. [cit. 2022-04-29]. Dostupné z: https://www.simplypsychology.org/likert-scale.html

[44] Povinnost mít zakrytá ústa na vybraných místech [online]. [cit. 2022-04-30]. Dostupné z: https://covid.gov.cz/opatreni/rousky-respiratory/povinnost-mit-zakryta-usta-na-vybranych-mistech

Příloha A

Seznam použitých zkratek

UML Unified Modeling Language

SQL Structured Query Language

MVC Model View Controller

NPM Node Package Manager

REST Representational State Transfer

API Application Programming Interface

DTO Data transfer object

HTML HyperText Markup Language

ORM Objektově relační mapování

HTTP HyperText Transfer Protocol

JSON JavaScript Object Notation

URL Uniform Resource Locator

SoC Separation of concerns

JWT JSON Web Token

SPA Single-page application

JPA Java Persistence API

CRUD Create Read Update Delete

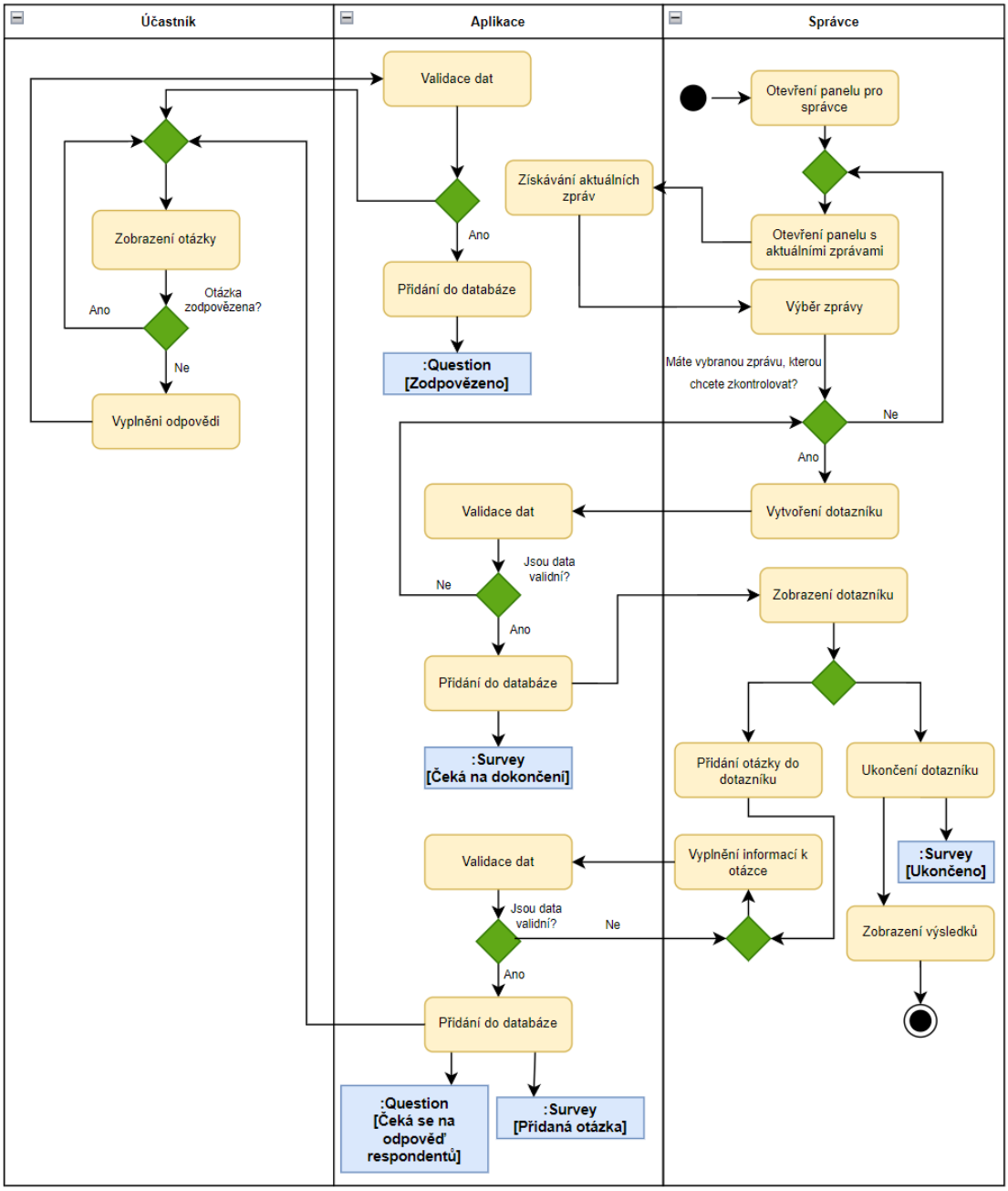
IoC Inversion of Control

CORS Cross-Origin Resource Sharing

Příloha B

Diagramy

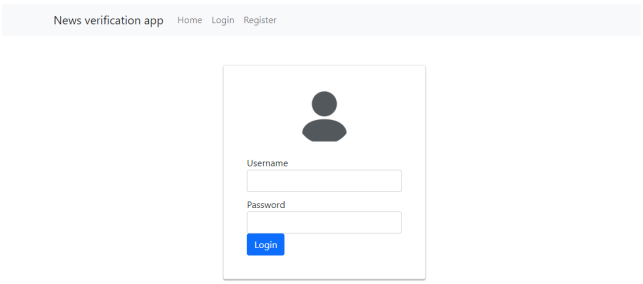
Obrázek B.1: Případy užití



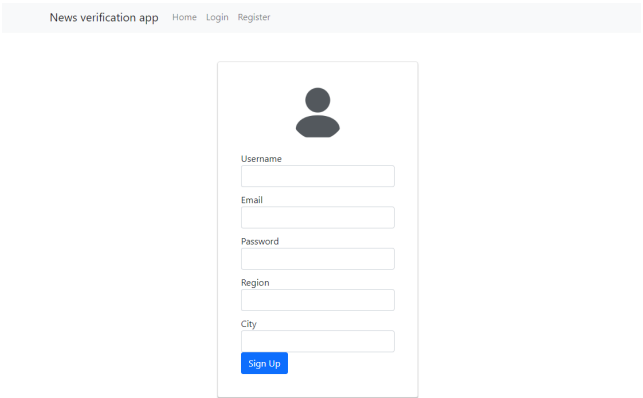
Obrázek B.2: Diagram aktivit

Příloha C

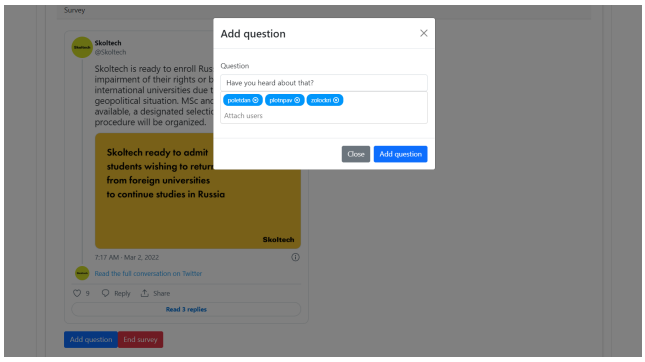
Výsledná aplikace



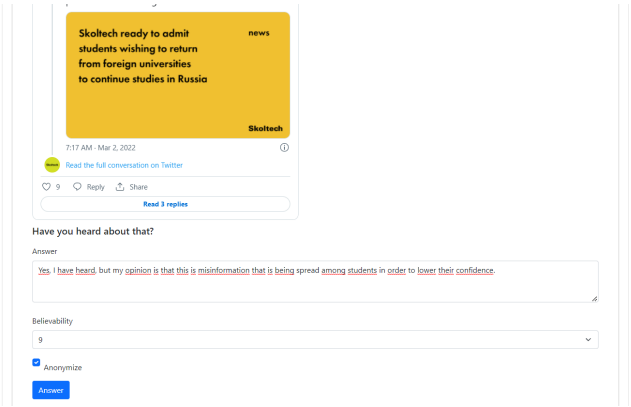
Obrázek C.1: Přihlášení



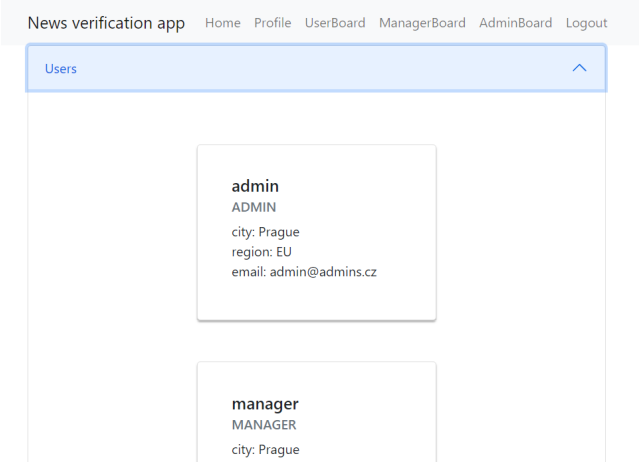
Obrázek C.2: Registrace



Obrázek C.3: Přidaní otázky



Obrázek C.4: Odpovídaní na otázku



Obrázek C.5: Seznam uživatelů

Příloha D

Obsah přiloženého SD

- `src`
  - `prototype` ..... zdrojové kódy implementace
    - `backend` ..... zdrojové kódy backendu
    - `frontend` ..... zdrojové kódy frontendu
    - `nginx` ..... konfigurace nginx
    - `docker-compose.yml` ..... konfigurace docker-compose
    - `README.txt` ..... návod na nasazení
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF