F3 **Fakulta elektrotechnická**
Katedra počítačů

Diplomová práca

Diplomová práca

Frontend platformy pro sdílené ověřování faktů

Bc. Roman Bútora
Otevřená informatika

Květen 2023
`https://github.com/aic-factcheck/fact_checking_fe`
Vedúci práce: Ing. Jan Drchal, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bútora** Jméno: **Roman** Osobní číslo: **508455**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Frontend platformy pro sdílené ověřování faktů

Název diplomové práce anglicky:

Crowd-sourcing Platform Frontend for Fact-checking

Pokyny pro vypracování:

Cílem je navrhnout a vyvinout frontend systému pro sdílené (crowd-sourced) ověřování faktů (fact-checking). Cílem systému bude zejména sběr tvrzení (a jejich zdrojů), které mají být v další fázi, potenciálně jinými uživateli, ověřeny. Celá platforma bude navržena s cílem snadného strojového zpracování sbíraných dat.
1) Prozkoumejte existující crowdsourcing platformy. Zaměřte se na ověřování faktů a strojové zpracování nasbíraných dat.
2) Zvolte vhodné technologie, navrhněte a naimplementujte frontend systému.
3) Zpracujte zejména následující dva případy užití: 1) sběr textových tvrzení, včetně textů zdroje, 2) ověření tvrzení doplněné množinou důkazních textů.
4) Prozkoumejte a naimplementujte základní prvky gamifikace, které budou motivovat uživatele k rozšiřování databáze ověřených tvrzení.
5) Otestujte a vyhodnoťte kvality systému s využitím testerů. Přístup k testerům poskytne vedoucí práce - bude se jednat buď o studenty žurnalistiky FSV, nebo o členy NLP skupiny v rámci Centra umělé inteligence FEL.

Seznam doporučené literatury:

[1] Allen, Jennifer, et al. "Scaling up fact-checking using the wisdom of crowds." Science advances 7.36 (2021): eabf4393
[2] Pinto, Marcos Rodrigues, et al. "Towards fact-checking through crowdsourcing." 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2019.
[3] Sotirakou, Catherine, Theodoros Paraskevas, and Constantinos Mourlas. "Toward the Design of a Gamification Framework for Enhancing Motivation Among Journalists, Experts, and the Public to Combat Disinformation: The Case of CALYPSO Platform." International Conference on Human-Computer Interaction. Springer, Cham, 2022.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Drchal, Ph.D. katedra teoretické informatiky FIT

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

Datum zadání diplomové práce: **30.01.2023** Termín odevzdání diplomové práce: **26.05.2023**

Platnost zadání diplomové práce: **22.09.2024**

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

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Poďakovanie / Prehlásenie

Chcel by som poďakovať všetkým profesorom za skvelé dva roky na tejto škole, vďaka ktorým sa moje vedomosti v oblasti softvérového inžinierstva výrazne rozvinuli a dobre ma pripravili na moju pracovnú kariéru. Samostatná vďaka patrí vedúcemu práce Ph.D. Ing. Janovi Drchalovi za spoluprácu na zaujímavom projekte FactCheck a môjmu spolužiakovi Bc. Rastislavovi Kopálovi, s ktorým som na ňom spolupracoval a mal na starosti backend. Rád by som poďakoval aj firmám Avast a Gen Digital, ktoré nám poskytli prostriedky na tvorbu projektu. Vďaka patrí aj študentom Fakulty sociálních věd Univerzity Karlovy za otestovanie a vyjadrenie názoru na našu aplikáciu. V neposlednom rade patrí poďakovanie aj mojim rodičom, ktorí ma počas ťažkých chvíľ vedeli vždy podporiť.

Vyhlasujem, že som predloženú diplomovú prácu vypracoval/a samostatne a že som uviedol všetky použité informačné zdroje v súlade s s Metodickým pokynom o dodržiavaní etických princípov pri vypracovaní vysokoškolskej práce.

V Prahe, 26. 5. 2023

.....

Abstrakt / Abstract

Cieľom tejto diplomovej práce je vytvoriť frontend pre crowdsourcingovú platformu určenú na overovanie faktov. Na úvod rozoberieme problematiku crowdsourcingu, overovania faktov a porovnáme súčasné najznámejšie platformy dostupné na webe. V ďalšom kroku preskúmame dostupné technológie pre frontend a vyberieme vhodné riešenie. Analyzujeme funkčné a nefunkčné požiadavky nášho systému a pomocou zvolených technológií implementujeme riešenie. Opíšeme architektúru aplikácie, ktorú sme navrhli v spolupráci s vedúcim práce Ing. Janom Drchalom a spolužiakom Bc. Rastislavom Kopálom, zodpovedným za backend aplikácie. Naše riešenie otestujeme pomocou automatických nástrojov na testovanie a dobrovoľných testerov. V závere vyhodnotíme výsledky a navrhneme zlepšenia pre budúci rozvoj aplikácie.

Kľúčové slová: Informačný systém, crowdsourcing, overovanie faktov, frontend, používateľské rozhranie, React, TypeScript, Cypress

The aim of this thesis is to create a frontend for a crowdsourcing platform designed for fact-checking. First, we explain the theme of crowdsourcing, fact-checking and we compare the current most well-known platforms available online. Next, we explore the available technologies for the frontend and choose a suitable solution. We analyze the functional and non-functional requirements of our system and implement the solution using the selected technologies. We describe the architecture of the application that we have designed in collaboration with the thesis supervisor Ing. Jan Drchal and classmate Bc. Rastislav Kopál, responsible for the backend of the application. We test our solution using automated testing tools and volunteer testers. Finally, we evaluate the results and suggest improvements for future development of the application.

Keywords: Information system, crowdsourcing, fact-checking, frontend, user interface, React, TypeScript, Cypress

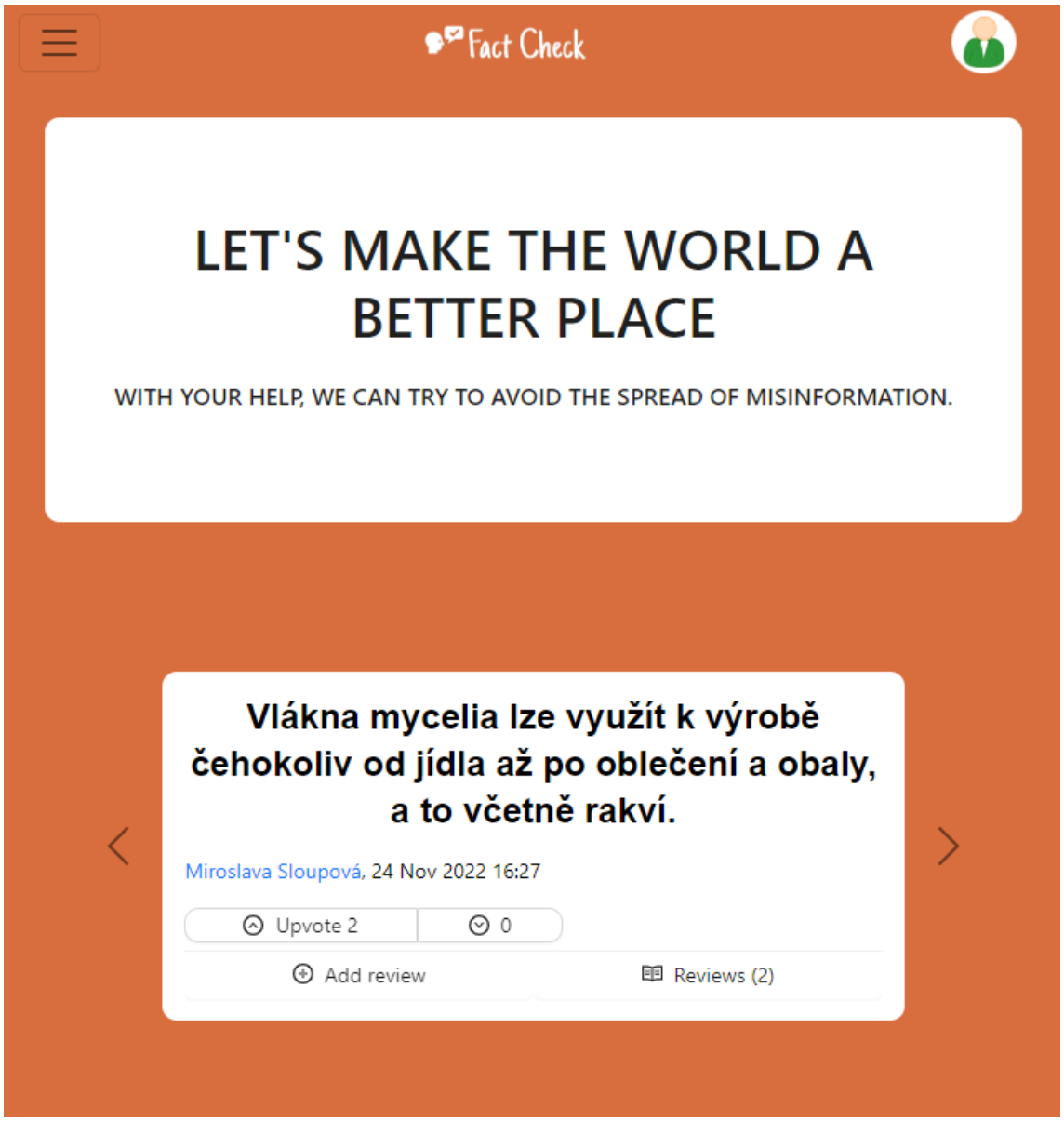
Title translation: Crowd-sourcing Platform Frontend for Fact-checking

Obsah /

/ Obrázky

Kapitola 1
Úvod

Aplikácia FactCheck je webová aplikácia, ktorej účelom je zbieranie dát od používateľov na internete. Pre použitie aplikácie je nutné sa zaregistrovať a byť prihlásený. Po prihlásení je užívateľ presmerovaný na úvodnú stránku, na ktorej sa oboznámi s jej cieľmi a spôsobom použitia. Následne môže využívať plnú funkcionalitu aplikácie.



Obrázok 1.1. Screenshot úvodnej stránky z našej aplikácie FactCheck

Hlavnou funkciou portálu je vkladanie článkov z internetu alebo iných médií. Po vložení článku môže užívateľ vložiť úryvok textu, ktorý obsahuje potencionálne dôležité

tvrdenie. Tieto tvrdenia môžu následne iní používatelia hodnotiť, overovať a komentovať. Pre seriózność článkov a hodnotení je potrebné vkladať aj ich zdroj.

Dáta užívateľov sa ukladajú do databázy za účelom ich následného použitia pri tvorbe verejne dostupnej databázy FactChecku. Zároveň chceme, aby bola databáza vhodne navrhnutá pre strojové spracovanie umelej inteligencie. Cieľom webovej aplikácie FactCheck je pomôcť ľuďom overovať fakty, o ktorých nemajú dostatočné znalosti a informácie. Zároveň chceme docieliť, aby toto overovanie faktov bolo zábavné, poučné a jednoducho použiteľné, pre každého užívateľa internetu.

Pre motiváciu používateľov k zbieraniu dát je v aplikácii využitá gamifikácia, ktorá zabezpečuje priraďovanie bodov užívateľom za rôzne aktivity. Aktivitou je buď pridanie nového článku, tvrdenia alebo hodnotenia. Body bude tiež možné získať za sociálne interakcie, umiestnenie sa v rebríčku najaktívnejších užívateľov a pridávanie reakcií na príspevky. Užívateľ je motivovaný bodmi budovať svoje skóre (kredibilitu). Toto skóre je uvedené v rebríčku najlepších (resp. najaktívnejších) užívateľov a slúži ako váha jeho hlasu pri overovaní faktov.

V rámci tejto diplomovej práce rozoberieme existujúce riešenia fact-checkingových platforiem, zhodnotíme ich vlastnosti a navrhneme zlepšenia, ktoré by mohli viesť k boju proti dezinformáciám. Na základe našich vedomostí z návrhu softwarových riešení navrhneme štruktúru celej aplikácie a vyberieme technológie, ktoré použijeme na implementáciu. Naštudujeme si teóriu, ktorá nám s výberom pomôže a v teoretickej časti ju opíšeme. Spomenieme tiež patterny potrebné pre správny, bezpečný a plynulý chod aplikácie. Pre zlepšenie používateľského rozhrania tiež rozoberieme získané znalosti z oblasti UX.

Následne tieto poznatky aplikujeme a vytvoríme systém, ktorý otestujeme automatickými testami a vyhodnotíme dotazník, ktorý vyplnia reálni používatelia našej aplikácie. Na základe dotazníka na záver navrhneme dodatočné zlepšenia, ktoré po skončení diplomovej práce budeme implementovať a aplikáciu naďalej rozvíjať.

Kapitola 2
Úvod do problematiky

Existencia internetu má nespochybniteľný dopad na to, akým spôsobom ľudia prijímajú informácie. Dnešné technológie umožňujú každému propagovať svoje názory na celom svete v reálnom čase. Sociálne siete sa tak stávajú silným nástrojom šírenia informácií, ktoré môže hocikto tvoriť a zdieľať. Negatívnym dopadom tohto trendu je šírenie veľkého množstva falošných správ, ktoré zdanlivo pôsobia ako pravdivé.

Šírenie falošných správ je výzvou pre dnešnú spoločnosť, pretože proces boja proti dezinformáciám je komplikovaný a nákladný. Na vyhľadávanie správ uverejnených na internete a detekciu ich pravdivosti sa zameriavajú rôzne organizácie na kontrolu faktov po celom svete. Túto prácu zvyčajne vykonávajú novinárske organizácie, ktoré zisťujú fakty hľadaním spoľahlivých zdrojov na podporu svojich záverov. Problém spočíva v tom, že proces overovania faktov organizáciami je pomalý v porovnaní s rýchlosťou, akou sa šíria dezinformácie [1].

Podľa Pinta existujú štyri hlavné prístupy k overovaniu faktov: overovanie faktov jednotlivých tvrdení, odhaľovanie falošných článkov, lov na trollov a meranie spoľahlivosti spravodajských zdrojov [1].

V tejto diplomovej práci sme navrhli proces overovania faktov založený na overovaní tvrdení z konkrétnych článkov na internete, ktorý by mohol vykonávať ktokoľvek. Táto decentralizovaná sieť by mala zodpovednosť za vykonanie všetkých krokov týkajúcich sa posúdenia pravdivosti správ predložených na kontrolu. Vďaka tomu dáme možnosť ľuďom, aby sa samy vyjadrili k tomu, čo ich zaujíma a mohli reagovať na podnety iných ľudí.

2.1 Crowdsourcing

Crowdsourcing možno definovať ako využitie informačných technológií (IT) na outsourcing akejkoľvek organizačnej funkcie strategicky definovanej skupine aktérov - ľudských alebo neľudských - formou otvorenej komunikácie [2] .

Tento pojem súvisí s myšlienkou, že internet môže pomôcť jednotlivcom zhromažďovať alebo vyberať zmysluplné informácie z potenciálne obrovského množstva ľudí [3]. Wikipédia je známym príkladom crowdsourcingu, pretože obsahuje obrovské množstvo materiálov, ktoré udržiava výlučne decentralizovaná a neznáma komunita.

Podobný prístup sa využíva v rôznych internetových platformách, ktoré umožňujú používateľom spolupracovať a zároveň využívať informácie poskytované touto komunitou .

Jeden z hlavných problémov tradičnej metódy overovania faktov - práce profesionálnych overovateľov je jej malá škálovateľnosť. Riešenie preto vidíme v crowdsourcingu. Potenciálne riešenie popisované Pintom skúma overovanie dezinformácií v rôznorodých skupinách laikov.

Komplexita Pintovej metódy spočíva hlavne vo výbere podnetov. Výskum ukázal, že laici dokážu detekovať pravdivosť informácií, keď sú požiadaní, aby zhodnotili súbor

tvrdení starostlivo vybraných výskumníkmi. Schopnosť ľudí rozpoznať falošné tvrdenia však do značnej miery závisí od povahy samotného tvrdenia.

Preto je pri posudzovaní schopnosti davov zvýšiť škálovateľnosť programov na overovanie faktov nevyhnutné použiť škálu výrokov, ktorá presne odráža výzvy, ktorým čelia spoločnosti sociálnych médií pri odhaľovaní dezinformácií [1].

V štúdii Allena a kol. spoločnosť Facebook poskytla výber článkov na hodnotenie [4]. Tieto články boli vybrané z obsahu zverejneného na Facebooku s dôrazom na občianske alebo zdravotné témy, potenciálne dezinformácie alebo široko zdieľaný obsah. Pomocou tohto výberu bolo možné priamo vyhodnotiť schopnosť davu overovať fakty na súbore potenciálne problematického obsahu, ktorý platformy sociálnych médií zvyčajne smerujú k profesionálnym overovačom faktov.

V tejto štúdii preskúmali 207 spravodajských článkov. Následne porovnali hodnotenia presnosti troch profesionálnych overovateľov faktov, ktorí vykonali overenie každého článku s hodnoteniami 1 128 Amerických obyvateľov. Tí hodnotili každý článok na základe jeho titulku a hlavného odseku.

Zistenia poukazujú na skutočnosť, že priemerné hodnotenia malých, politicky rôznorodých davov bežných ľudí vykazujú koreláciu s priemernými hodnoteniami profesionálnych overovateľov faktov, porovnateľnú s koreláciou medzi hodnoteniami rôznych overovateľov faktov, a presne predpovedajú, či väčšina overovateľov faktov klasifikovala titulok ako pravdivý [4].

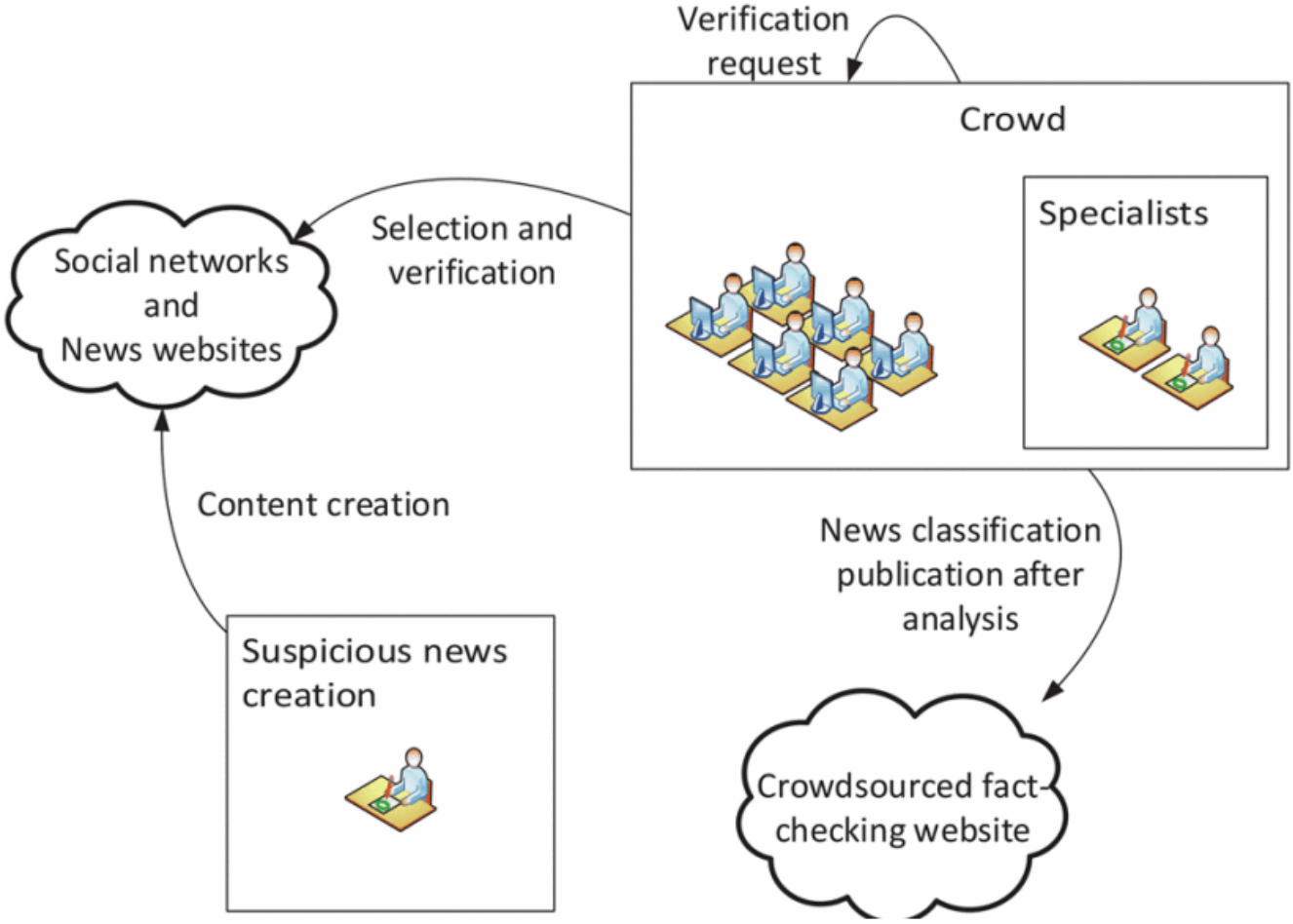
Ďalšie pozorovania Allena a kol. naznačujú, že rozpoznanie vydavateľa každého titulku prináša mierny nárast súhlasu s hodnoteniami overovateľov faktov. Z toho dôvodu sme sa rozhodli do aplikácie pridať mená autorov jednotlivých príspevkov. Ľudia sa tak môžu prekliknúť na jeho profil a vidia, aký obsah pridáva, koľko ľuďom prídu jeho články zaujímavé a koľko súhlasov/nesúhlasov s tvrdeniami a hodnoteniami dostal. Ľudia si tak môžu spraviť predstavu o jeho názoroch a na základe jeho predošlých úsudkov určiť, či ide z ich pohľadu o dôveryhodnú osobu.

2.2 Falošné správy

Falošné správy a možnosť, že môžu ovplyvniť politickú, hospodársku a sociálnu situáciu, vyvolávajú celosvetové obavy. Na pochopenie toho, ako sa šíria falošné správy, Vosoughi a kol. použili súbor údajov s fámami na Twitteri v rokoch 2006 až 2017. Približne 126 000 fám šírilo asi 3 milióny ľudí. Falošné správy sa dostali k väčšiemu počtu ľudí ako pravda; 1% falošných správ sa rozšírilo k 1000 až 100 000 ľuďom, zatiaľ čo pravda sa zriedkavo rozšírila k viac ako 1000 ľuďom [5] . Z toho usudzujeme, že falošné správy sa šíria rýchlejšie a preto je dôležité vytvárať prostredie a nástroje, ktoré môžu ich šírenie obmedziť.

Odborníci sa snažia vykonávať overovanie faktov v politických diskusiách a odborných článkoch. Nemajú však potenciál davu, ktorý decentralizovaným spôsobom môže vytvoriť veľkú sieť overovania, čím sa proces overovania urýchli. Práca davu dobrovoľným spôsobom môže vyriešiť problém s nákladosťou overovania faktov, ktorý vykonávajú profesionálne organizácie, a tak by sa tento model mohol stať ešte atraktívnejším [1].

Zistilo sa napríklad, že úsudok rôznorodej, nezávislej skupiny laikov prekonáva úsudok jediného experta v rôznych oblastiach vrátane hádania úloh, lekárskych diagnóz a predpovedí podnikových ziskov [4] . Z toho usudzujeme, že by decentralizovaná platforma na FactChecking mohla viesť k šíreniu pravdivých informácií pri dostatočne veľkej zainteresovanosti rôznorodých skupín.



Obrázok 2.1. Diagram overovacej štúdie Pinta [1]

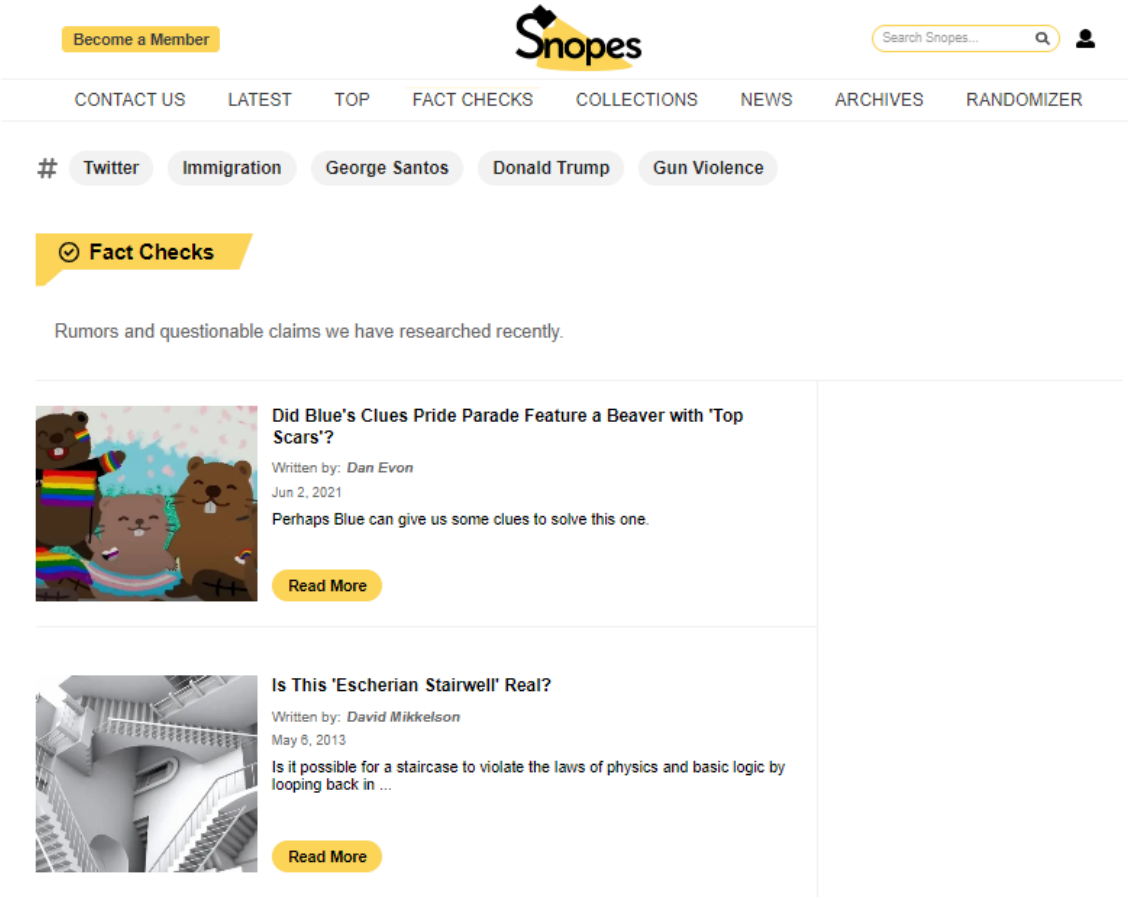
2.3 Porovnanie existujúcich riešení Fact - Checkingu

Platformy na overovanie faktov zohrávajú čoraz dôležitejšiu úlohu, pretože poskytujú užívateľom spôsob ako rozoznať pravdivé informácie od nepravdivých či zavádzajúcich informácií. Viaceré webové stránky a organizácie sa venujú overovaniu informácií a vyvracaniu hoaxov. Nasledujúca kapitola poskytne komplexné porovnanie niektorých populárnych platforiem na overovanie faktov, ako sú FactCheck.org, Snopes, PolitiFact a FactCheck, rozoberie ich výhody a nevýhody a navrhne nápady na zlepšenie.

2.3.1 Snopes

Snopes má intuitívne rozhranie a ľahko sa v ňom orientuje. Používatelia môžu rýchlo nájsť informácie o konkrétnych témach prostredníctvom vyhľadávacieho panela alebo prechádzať rôzne kategórie.

Stránka poskytuje aj prehľadnývlastný systém hodnotenia, ktorý obsahuje hodnotenia : Prebieha výskum, Pravda, Z väčšej časti pravda, Mix, Z väčšej časti nepravda, Nepravda, Nenájdené dôkazy, Nedokázané, Zastaralý (zmenil sa napr. zákon), Zlý titulok, Správny citované, Nesprávne citované, Legenda, Scam, Overené, Satira, Bolo to myslené ako satira, Stiahnutie z trhu a Troll. Všetky tieto hodnotenia sú v tabuľke vysvetlené, z pohľadu bežného užívateľa však usudzujeme, že pre prvého návštevníka môže byť mätúce množstvo kategórií, hlavne tie, ktoré znejú podobne (pravda a overené).



Obrázok 2.2. Screenshot z webu Snopes

Metodika overovania faktov nie je jednotná, ale líši sa v závislosti od témy. Tím redakcie vykonáva predbežný prieskum, pokúša sa kontaktovať zdroj tvrdenia, aby získal podporné informácie, a vyhľadáva tlačené informácie a štatistické zdroje.

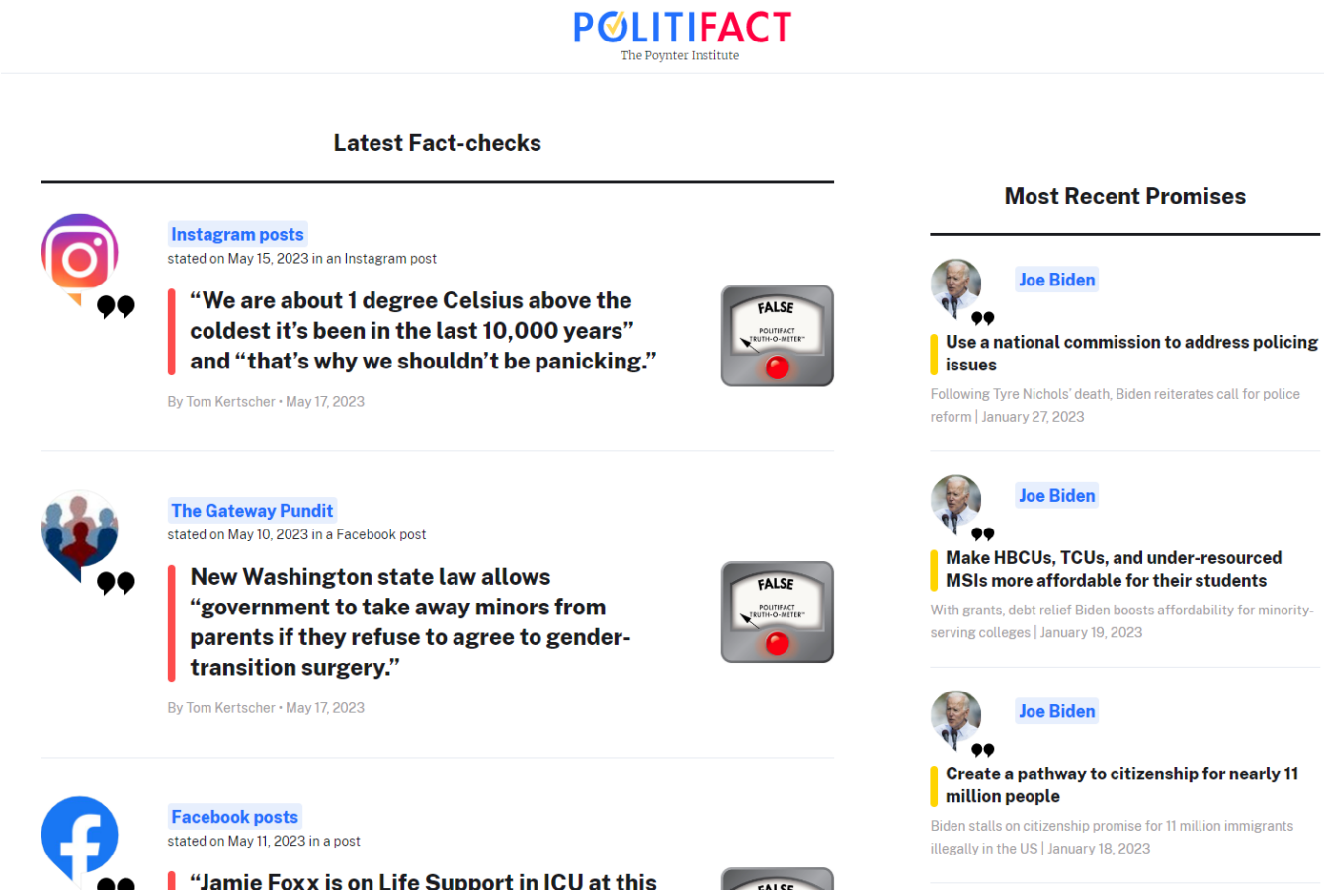
Témy sa vyberajú na základe záujmu čitateľov, trendoch na Google a sociálnych platformách. Témy môžu zahŕňať text šírený online, príspevky na sociálnych sieťach, obrázkové makrá/memá, videá, tlačené materiály a články z iných webových stránok a publikácií. Snopes.com sa snaží používať nestranné zdroje informácií a údajov a všetky zverejnené zdroje a odkazy uvádza v časti Zdroje.

Zásadou organizácie Snopes je okamžite opraviť faktické chyby a objasniť všetky potenciálne mätúce alebo nejednoznačné tvrdenia vo svojich článkoch.

2.3.2 PolitiFact

PolitiFact má moderný dizajn, ktorý používateľom ukazuje tvrdenia z rôznych sociálnych sietí ako Facebook, Instagram, či Twitter. Webová stránka obsahuje hodnotiaci systém `Truth-O-Meter`, ktorý hodnotí tvrdenia ako Pravdivé, Väčšinou pravdivé, Polopravdivé, Väčšinou nepravdivé, Nepravdivé a takzvané Horiace nohavice (v prípade že zdroj uviedol absurdné tvrdenie) . Používatelia môžu vyhľadávať konkrétne výroky alebo prechádzať rôzne kategórie.

PolitiFact sa zaväzuje prezentovať pravdivé fakty bez predsudkov a zaujatosti. Novinári tejto webovej stránky sa vyhýbajú vyjadrovaniu politických názorov alebo účasti na politickom procese, vrátane neposkytovania politických príspevkov alebo práce na kampaniach.



Obrázok 2.3. Screenshot z webu Politifact

PolitiFact je platforma na overovanie faktov ocenená Pulitzerovou cenou, ktorá sa pri hodnotení tvrdení opiera o profesionálnych novinárov. Používajú prísnu metodiku, ktorá zahŕňa hĺbkový výskum, konzultácie s odborníkmi a dodržiavanie prísneho etického kódexu.

Novinári portálu PolitiFact samy hľadajú výroky, ktoré by mohli overiť. Sledujú najmä prepisy rozhovorov, prejavy, správy, tlačové správy, brožúry z kampaní, televíziu a sociálne médiá. Čitatelia môžu posielať svoje podnety emailom. Pri rozhodovaní o tom, ktoré tvrdenia preveria, berú do úvahy 4 otázky :

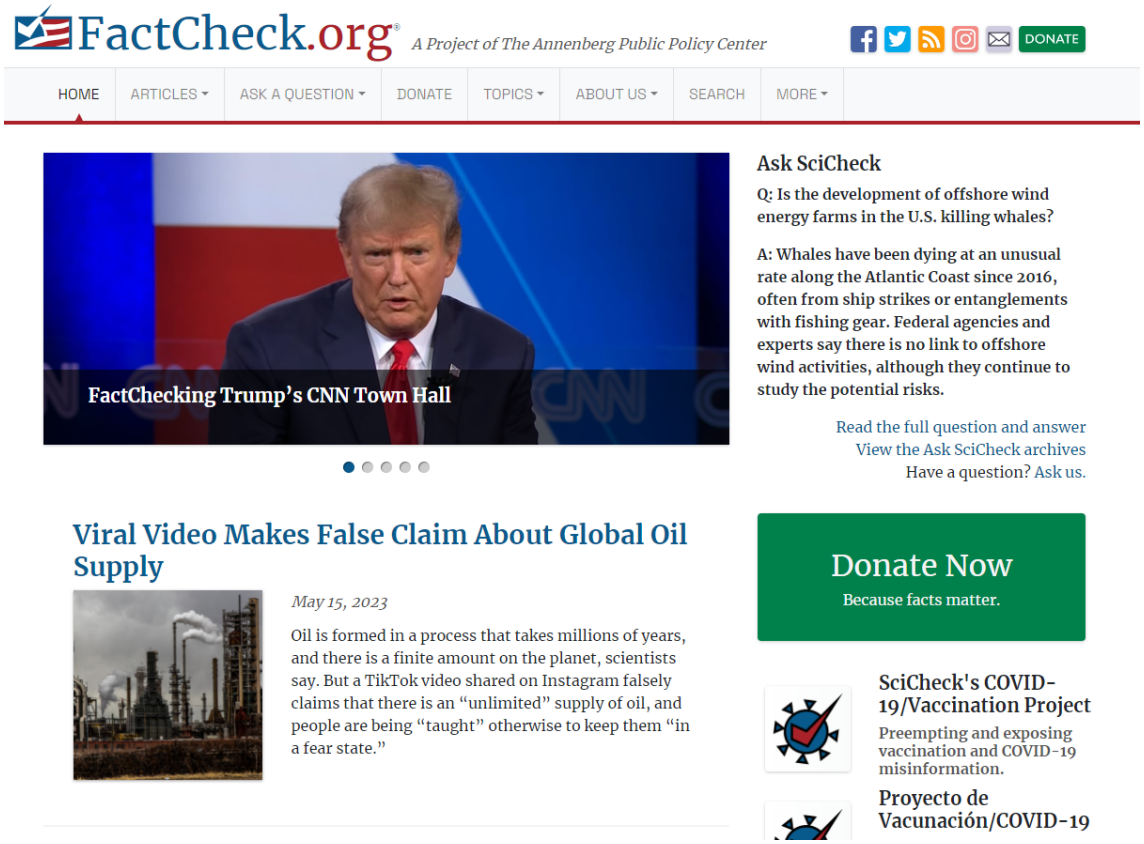
- Znie tvrdenie zavádzajúco alebo nesprávne?
- Je výrok významný? Vyhýbajú sa drobným chybám v tvrdeniach, ktoré sú zjavne prešľapom.
- Je pravdepodobné, že výrok budú šíriť a opakovať iní?
- Mohol by typický človek, ktorý by výrok počul alebo si ho prečítal, uvažovať: Je to pravda?

Ďalej analyzujú tzv. Flip-O-Meter, ktorý ukazuje, či daný človek (a ako zásadne) zmenil svoju pozíciu. Taktiež sledujú predvolebné sľuby a to, do akej miery sa skutočne naplnili.

2.3.3 FactCheck.org

FactCheck.org je prvá stránka, ktorá sa nám na google zobrazila po zadaní kľúčového slova fact-check. Má a jednoduché rozhranie, ktoré používateľom umožňuje prehľadávanie informácií. Stránka obsahuje články na politcké témy a používatelia môžu

vyhľadávať konkrétne tvrdenia alebo prechádzať kategórie. Na stránke tiež nájdeme odkaz na populárne tvdenia, pričom ich popularita sa určuje na základe toho, ktoré otázky sa užívatelia pýtajú najčastejšie tvorcov aplikácie. Platforma nemá špecifický systém hodnotenia, ale namiesto toho poskytuje podrobné vysvetlenia spolu s grafmi a obrázkami na vyvrátenie alebo potvrdenie tvrdení. Obsahuje tiež sekciu Ask a question, kde sa môžu užívatelia opýtať na konkrétne fakty.



Obrázok 2.4. Screenshot z webu FactCheck.org

FactCheck.org je nestranický, neziskový projekt, ktorý prevádzkuje Annenberg Public Policy Center. Platforma využíva tím skúsených novinárov, ktorí vykonávajú dôkladný výskum a konzultujú s odborníkmi, aby overili fakty tvrdení politikov a verejných činiteľov.

FactCheck.org spolupracuje s Facebookom, ktorý mu poskytuje odkazy na obsah označený ako potencionálne nepravdivý. Zoznam preverujú a vyberajú samy, o čom budú písať. Informácie, ktoré označia ako nepravdivé následne posielajú Facebooku, ktorý ho následne používa na obmedzovanie šírenia nepravdivých strán (nie je špecifikované, akým spôsobom).

2.3.4 Obmedzenia platforiem:

Všetky tri platformy majú jedinečné spôsoby metódy vyberania obsahu a preverovania faktov, ale chýba im možnosť pridávania vlastných článkov a tvrdení od bežných používateľov. Používatelia síce môžu kontaktovať tvorcov webu, avšak nie je zaručené, že ich príspevok bude pridaný. Taktiež proces overovania faktov môže byť zdĺhavý a pri veľkom množstve dotazov nie je možné na všetky odpovedať. Ďalším obmedzením je, že tieto platformy nie sú vhodné pre strojové spracovanie dát, pretože im v

niektorých prípadoch chýbajú odkazy na pôvodné zdroje. Tieto obmedzenia bránia platformám využiť kolektívne znalosti a skúsenosti ich používateľskej základne, čo by mohlo potenciálne zlepšiť proces overovania faktov.

2.3.5 Nápady na zlepšenie

Zapojenie používateľov: podporovanie používateľov, aby predkladali tvrdenia alebo články na overenie faktov, čím sa posilní pocit komunity a umožní sa používateľom prispieť k procesu hľadania pravdy. Týmto spôsobom by užívatelia boli motivovaný tráviť viac času na platforme FactCheck a mohli sa dozvedieť viac aktuálnych zaujímavých faktov.

Gamifikácia: zavedenie prvkov gamifikácie do týchto platforiem by mohlo potenciálne zvýšiť motiváciu používateľov zdieľať údaje a prispievať k procesu overovania faktov. Funkcie, ako sú odmeny, odznaky, rebríčky a výzvy, môžu motivovať používateľov k aktívnejšiemu zapojeniu, a tým zlepšiť schopnosť platforiem identifikovať a vyvracať dezinformácie.

Transparentnosť: poskytnutie komplexnej otvorenej diskusie o metodike a zdrojoch použitých na overovanie faktov, aby používatelia pochopili, ako sa dospelo k záverom, a podporte dôveru v platformu. Užívatelia by mohli lepšie pochopiť súvislosti, ak by na rovnakú tému mohli reagovať ľudia využívajúci rôzne zdroje.

Kontrola faktov v reálnom čase: vytvorenie nástroja, ktoré umožní overovanie faktov v reálnom čase, keď sa tvrdenia a dezinformácie šíria online. Za týmto účelom vytvára druhý tím model umelej inteligencie, ktorý bude v budúcnosti prepojený s našou aplikáciou a poskytoval by nezávislý názor na danú problematiku na základe jeho databázy znalostí a faktov.

2.3.6 Zhodnotenie, prečo implementujeme vlastný systém

Neustálym vývojom a prispôsobovaním sa meniacemu sa informačnému prostrediu môžu platformy na overovanie faktov zohrávať kľúčovú úlohu pri presadzovaní pravdy a boji proti dezinformáciám v digitálnom veku. Platformy na overovanie faktov, ako sú FactCheck.org, Snopes, PolitiFact zohrávajú dôležitú úlohu v boji proti dezinformáciám. Na zvýšenie svojej účinnosti sme sa rozhodli implementovať vlatný systém a prijať rôzne stratégie, ako napríklad zvýšenie zapojenia používateľov, zavedenie gamifikácie pokrytia a podpora transparentnosti.

2.4 Charakteristika respondentov, ktorý testujú aplikáciu

Pri návrhu nášho riešenia sa snažíme systém vytvoriť tak, aby ho mohli použiť ľudia aj bez technických znalostí a znalostí v obore FactCheckingu. Preto sa snažíme, aby celý systém bol jednoduchý a umožnil každému ho využiť.

V našom projekte sme mali v zimnom semestri k dispozícii skupinu študentov publicistiky, ktorý testovali náš systém. Sú to študenti bakalárskeho štúdia Fakulty sociálních věd na Karlovej univerzite v Prahe, ktorý dostali za úlohu náš systém použiť a vyplniť náš dotazník. Na základe získaných poznatkov sme sa pokúsili náš systém vylepšiť. Ako bolo v Pintovom prieskume zmienené, v prieskume je dôležité osloviť ľudí z rôznorodých nezávislých skupín, preto sme sa rozhodli v letnom semestri osloviť s úlohou aj ďalších ľudí v rôznych vekových kategóriách a pracovných oboroch. V kapitole o testovaní získané výsledky budeme analyzovať, zhodnotíme ich odpovede a navrhneme vylepšenia.

Kapitola 3
Použité technológie

3.1 Typy prístupu webových aplikácií

Multi-page aplikácie (MPA) a Single-page aplikácie (SPA) reprezentujú dva rozdielne prístupy vo vývoji webových stránok, pričom každý má svoje výhody a nevýhody. Hlavný rozdiel je v spôsobe, akým aplikácia načítava dáta a mení štruktúru elementov. V minulosti bola väčšina webových stránok implementovaná ako MPA. To znamená, že pri každom requeste sa znova načítala celá stránka. Napriek ich popularite mal tento prístup nedostatky, medzi ktoré patrí najmä slabá responzivita a interakcia s prostredím [6]. Vznikom frontendových frameworkov sa zjednodušil vývoj SPA.

3.1.1 Multi-page application

Multiple-page Application je klasický prístup, akým boli implementované prvé web stránky. Hlavnou výhodou MPA je, že umožňuje aplikáciu rozdeliť na samostatné celky, ktorých zdroje sú oddelené. Ďalšou výhodou bolo, že prehliadače ako Google mali v minulosti problém s indexovaním stránok, ktoré automaticky generujú svoj obsah. Podľa ich tvrdenia však už tento problém odstránili a dokážu rendrovať aj stránky s JavaScriptom [7]. Optimalizácia SEO však môže byť v prípade SPA väčšou výzvou najmä pre menej skúsených vývojárov.

3.1.2 Single-page application

Single-page application je typ webovej aplikácie, ktorá namiesto prechádzania na ďalšie stránky SPA dynamicky aktualizuje obsah na tej istej stránke za pomoci JavaScript frameworkov ako React, Angular alebo Vue. Pri počiatočnom načítaní stránky si stiahne celú aplikáciu a následne komunikuje so serverom iba v prípade, že potrebuje načítať nové dáta. Tento prístup poskytuje plynulejšie používateľské prostredie, pretože používateľ nemusí čakať na načítanie novej stránky. SPA sú tiež zvyčajne rýchlejšie ako viacstránkové aplikácie, pretože dokážu znížiť množstvo údajov prenášaných medzi klientom a serverom a môžu ukladať údaje do vyrovnávacej pamäte na strane klienta. Používanie SPA má však aj niektoré potenciálne nevýhody, ako napríklad pomalšie počiatočné načítanie (je potrebné stiahnuť celú stránku) a ťažšia optimalizácia pre vyhľadávače (SEO). Napriek tomu sa SPA široko používajú pre moderné webové aplikácie, ktoré uprednostňujú rýchle a responzívne používateľské rozhrania.

3.1.3 Zvolenie prístupu

Na základe dostupných informácií sme vyhodnotili, že použitie SPA bude vhodnejšou voľbou pre našu aplikáciu. Rozhodli sme sa tak najmä preto, že na našej stránke je množstvo akcií, ktoré môže užívateľ vykonať. V prípade použitia MPA by bol chod aplikácie pomalý a nepôsobil by plynulo. Inšpirovali sme sa tiež populárnymi sociálnymi médiami ako Facebook, ktoré tiež využívajú princíp SPA.

3.2 HTML

HTML je značkovací jazyk používaný na vytváranie a štruktúrovanie obsahu na webe. Je to značkovací jazyk určený na tvorbu webových stránok a používa sa v kombinácii s CSS a JavaScript. Spolu so spomínanými technológiami CSS a JavaScript je štandardizovaný spoločnosťou W3C, ktorá zabezpečuje kompatibilitu s rôznymi prehliadačmi a poskytuje návody pre tvorbu web stránok. Jazyk HTML používa elementy a atribúty na definovanie štruktúry a obsahu dokumentu DOM (Document Object Model) . Umožňuje vytvárať nadpisy, odseky, odkazy, obrázky, videá a iné typy multimediálneho obsahu. Jazyk HTML je platformovo nezávislý, čo znamená, že ho možno zobraziť na akomkoľvek zariadení s webovým prehliadačom bez ohľadu na operačný systém. Aktuálnou verziou je HTML5, ktorá obsahuje nové prvky, ako napríklad zvuk, video a canvas grafiku.

3.3 CSS

Kaskádové štýly (CSS) sú ďalšou z troch základných technológií pre návrh webových stránok. Používa sa na definovanie vizuálnej reprezentácie dokumentov HTML. CSS poskytuje možnosť oddeliť definíciu vizuálnej strany webovej stránky od jej štruktúry.

Umožňuje taktiež používať konzistentné štýly na rôznych lokalitách. Pomocou CSS môžeme definovať štýly nadpisov, odsekov, odkazov a iných prvkov. Pomocou CSS môžu vývojári vytvárať zložité rozloženia, upravovať rozstupy medzi prvkami a pridávať vizuálne efekty, ako sú gradienty, tiene a animácie.

Kód CSS môže byť zložitý a náročný na správu, najmä v prípade väčších webových stránok. Okrem toho môžu rôzne prehliadače interpretovať štýly CSS odlišne, čo môže spôsobiť problémy s kompatibilitou na rôznych zariadeniach a platformách.

Kaskádový štýl umožňuje vytvoriť viaceré vrstvy štýlov, pričom platí, že sa aplikuje posledné pravidlo. Ak máme teda dve rôzne pravidlá pre vnorený element, aplikuje sa to posledné.

Poslednou verziou je CSS3, ktorého hlavnou výhodou je umožnenie media queries. Vďaka nim môžeme pre jeden element napísať rôzne pravidlá, ktoré sa aplikujú na základe toho, aké zariadenie používame. Týmto spôsobom je možné vytvoriť lepšiu responzivitu na rôznych typoch zariadení s odlišným rozlíšením a pomerom strán.

3.4 Porovnanie JavaScriptu vs. TypeScriptu a výber technológie

JavaScript a TypeScript sú dva základné programovacie jazyky pre vývoj webových stránok. V nasledujúcej kapitole zhrnieme základné rozdiely medzi nimi a zhodnotíme ich kladné a negatívne vlastnosti.

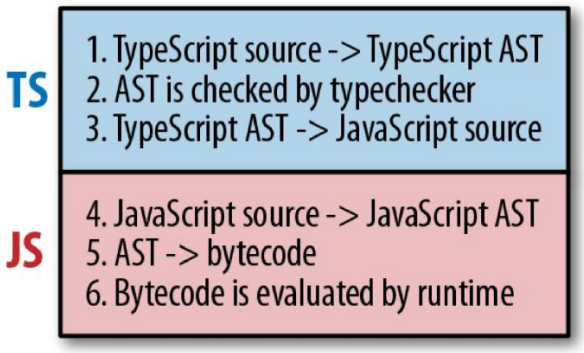
JavaScript je vysokoúrovňový interpretovaný programovací jazyk, ktorý umožňuje vývoj dynamických a interaktívnych webových aplikácií. JavaScript rozširuje možnosti statických web stránok o dynamické zobrazovanie dát z backendu, multimédií, tvorbu 2D/3D grafiky, videí a ďalších funkcionalít.

TypeScript je staticky typová nadmnožina jazyka JavaScript, vyvinutá spoločnosťou Microsoft s cieľom rozšíriť JavaScript o voliteľné statické typovanie prostredníctvom dátových typov.

Kompilácia - JavaScript je interpretovaný jazyk, čo znamená, že ho JavaScriptový engine spúšťa priamo bez potreby samostatnej kompilácie. JavaScriptový engine prehliadača číta a interpretuje kód riadok po riadku, prevádza ho na strojový kód a vykonáva ho.

TypeScript je kompilovaný jazyk, ktorý sa prekladá do JavaScriptu. Prvou fázou je parsovanie. Kód sa analyzuje kompilátorom jazyka TypeScript, ktorý z neho vytvorí model abstraktného syntaktického stromu (AST) [8]. Ďalej kompilátor skúma kód s cieľom určiť typy premenných, funkcií a výrazov. Okrem toho hľadá typové chyby, ako napríklad priradenie textu do premennej obsahujúcej číslo. Následne kompilátor vytvorí kód v JavaScripte, typové anotácie sú odstránené.

Kompilácia TypeScriptu do JavaScriptu zabezpečuje kompatibilitu s akýmkoľvek prostredím a knižnicami, ktoré podporujú JavaScript.



Obrázok 3.1. Schéma kompilácie TypeScriptu [8]

Typovanie - Dynamické typovanie jazyku JavaScript môže viesť k chybám počas behu a obmedzovať výkon, najmä v rozsiahlych aplikáciách. Na rozdiel od JavaScriptu TypeScript využíva statické typovanie na zisťovanie chýb počas procesu kompilácie [8]. Z našich skúseností získaných počas vývoja usudzujeme, že pevná štruktúra TypeScriptu pomáha rozdeliť kód do logických celkov, vďaka čomu je prehľadnejší a ľahšie sa vyvíja nová funkcionalita. Zároveň eliminuje počet chýb, ktoré môžu nastať počas runtime.

Udržateľnosť - JavaScript je flexibilný programovací jazyk, ktorý umožňuje vývojárom rýchlo experimentovať a testovať nápady. Táto flexibilita môže byť výhodou pri vytváraní inovatívnych riešení. Správa a údržba rozsiahleho kódu však môže byť náročná. Absencia striktného typovania môže spôsobiť nekonzistentnosť a chyby, ktoré sa ťažko identifikujú.

Prísne typovanie jazyka TypeScript a doplnkové funkcie, ako sú rozhrania a menné priestory, zjednodušujú správu a údržbu veľkých projektov. Typový systém umožňuje vývojárom identifikovať chyby už na začiatku životného cyklu vývoja, čím sa predchádza nákladným chybám.

Bezpečnosť - JavaScript je náchylný na rôzne bezpečnostné zraniteľnosti vrátane útokov typu XSS (cross-site scripting) a zámeny typov. Absencia striktného typovania môže tieto zraniteľnosti ešte zhoršiť a útočníci ich môžu ľahšie zneužiť.

Statické typovanie jazyka TypeScript môže zmierniť niektoré bezpečnostné riziká tým, že presadzuje typové obmedzenia a uľahčuje komplexnejšiu analýzu kódu. Jazyk TypeScript však nie je odolný voči všetkým bezpečnostným zraniteľnostiam, pretože sa v konečnom dôsledku kompiluje do jazyka JavaScript.

Zhrnutie - JavaScript aj TypeScript majú svoje výhody a nevýhody. JavaScript je poskytuje lepšiu prispôsobivosť a rozsiahly ekosystém, ale v rozsiahlych aplikáciách

sa môže stať neprehľadným. TypeScript ponúka lepšiu udržateľnosť a bezpečnosť typov, hoci na úkor dodatočných nástrojov a nastavení. Výber medzi JavaScriptom a TypeScriptom si vyžaduje dôkladné posúdenie požiadaviek projektu, odborných znalostí a osobných preferencií. V zimnom semestri bola aplikácia FactCheck implementovaná v JavaScripte, avšak časom sa kód stal menej prehľadným. V letnom semestri sme sa preto rozhodli celý kód prepísať do TypeScriptu, vďaka čomu bola aj implementácia novej funkcionality jednoduchšia a počas vývoja vznikalo menej bugov.

3.5 Porovnanie frontend frameworkov a výber technológie

Frontendové frameworky zmenili spôsob vytvárania webových aplikácií. Pre rozsiahle weby obsahujúce množstvo funkcionality sa stáva kód písaný v čistom JavaScripte neprehľadným. Väčšina aplikácií taktiež obsahuje opakujúce sa komponenty, ktorých kód by sme museli kopírovať, ak by sme nepoužívali žiadny framework. Frontendové frameworky poskytujú vhodné možnosti pre vytvorenie logickej štruktúry kódu, vďaka ktorej je celý projekt prehľadnejší a vytvorenie novej funkcionality jednoduchšie. Z prieskumu stránky Stack Overflow, ktorého sa zúčastnilo viac ako 70 000 developerov, vyplýva, že dve najpoužívanejšie technológie používané na frontende sú React a Angular [9]. Tu treba podotknúť, že Angular je framework a React je knižnica. Hlavné rozdiely medzi nimi sú:

- Data binding – Data binding je väzba medzi premennou v kóde a jej využitím v HTML časti komponentu alebo v podriadenom komponente. Angular používa tzv. two-way data binding. Ak napríklad vložíme do formulára text, tento text je automaticky uložený v príslušnej premennej. Takisto ak zmeníme obsah premennej, okamžite sa aktualizuje zobrazenie. React používa one-way data binding. To znamená, že zmena stavu (state) ovplyvní, čo používateľ vidí na obrazovke (view). To ale neplatí opačne.
- Krivka učenia – React je jednoduchší na pochopenie pre začiatočníkov [10]
- Štruktúra kódu – Angular používa pevnú štruktúru Model View Controller. React umožňuje väčšiu flexibilitu
- DOM – DOM je objektový model používaný pre HTML. Angular používa tzv. real DOM, React používa virtual DOM. Výhoda virtual DOM je v tom, že pri úprave jedného komponentu nie je potrebné rendrovať obsah celého DOM, zmení sa len konkrétny element. Táto skutočnosť má pozitívny dopad na celkovú výkonnosť Reactu [10]
- Jazyk – React používa natívne JavaScript, ale umožňuje aj použitie TypeScriptu (ktorý sme si nakoniec vybrali). Angular používa TypeScript.

Obe technológie poskytujú dostatočnú funkcionalitu pre vytvorenie aplikácie FactCheck. Pre našu implementáciu sme si vybrali React, pretože je viac používaný (veríme že sa nám v budúcnosti zíde) a z dokumentácie je takisto ľahšie pochopiteľný. Navyše sme s ním už v minulosti mali nejaké skúsenosti.

3.6 React

React je knižnica na vývoj používateľského rozhrania založená na jazyku JavaScript. Hoci React nie je jazyk, ale knižnica, vo vývoji webových stránok sa používa vo

veľkej miere. Ponúka okrem samotného používateľského rozhrania aj rôzne rozšírenia na podporu architektúry celých aplikácií, ako napríklad Flux a React Native. V nasledúcej kapitole zhrnieme dôvody, prečo sme sa rozhodli použiť túto technológiu [11] .

3.6.1 Charakteristika kľúčových vlastností

React uľahčuje vytváranie dynamických webových aplikácií, pretože vyžaduje menej programovania v porovnaní s AngularJS alebo klasickým JavaScriptom [12] . React využíva virtuálny DOM, čím vytvára webové aplikácie rýchlejšie. Virtual DOM porovnáva predchádzajúce stavy komponentov a aktualizuje len tie položky v Real DOM, ktoré boli zmenené, namiesto toho, aby sa všetky komponenty aktualizovali znova, ako to robia bežné webové aplikácie [12].

Základným prvkom Reactu sú opakovane použiteľné komponenty. Komponenty sú stavebnými prvkami každej aplikácie React a jedna aplikácia sa zvyčajne skladá z viacerých komponentov. Tieto komponenty majú svoju logiku, ovládacie prvky a možno ich opakovane používať v každej časti aplikácie, čo výrazne skracuje čas vývoja aplikácie. Rozdelenie aplikácie do samostatných komponentov zvyšuje jej prehľadnosť. Ak jeden komponent používame na viacerých miestach, zmena v komponente sa prejaví na všetkých miestach. Použitím props môžeme tiež naše komponenty prispôsobovať na rôznych miestach aplikácie.

Knižnica sa riadi jednosmerným tokom údajov. To znamená, že pri navrhovaní aplikácie React vývojári často vnášajú podriadené komponenty do nadradených komponentov. Keďže dáta tečú jedným smerom, je jednoduchšie ladiť chyby a zistiť, kde sa v danom momente vyskytuje v aplikácii problém. React sa dá ľahko naučiť, pretože kombinuje základné koncepty HTML a JavaScriptu s niektorými uľahčeniami. Facebook vydal rozšírenie `React Developer Tools` pre Chrome, ktoré možno použiť na ladenie aplikácií React. Vďaka tomu je proces ladenia webových aplikácií React rýchlejší a jednoduchší.

3.7 Výber technológie pre používateľské rozhranie

Najznámejšie knižnice pre UI komponenty sú Bootstrap, Material UI a Antd. Po preštudovaní ich dokumentácie sme usúdili, že všetky tri knižnice ponúkajú dostatočné možnosti pre vytvorenie dizajnu našej aplikácie. Pre náš projekt sme si vybrali Antd, pretože nás zaujalo množstvo návodov na ich stránke, ako vytvoriť používateľsky prívetivý dizajn. Dokumentácia obsahuje aj sekciu copywriting, ktorá nám pomohla s návrhom čo najjednoduchšieho užívateľského rozhrania. Pri tvorbe navigačného panela sme použili aj prvky z Bootstrapu. Urobili sme to z dôvodu, že sme mali konkrétny návrh toho, ako chceme aby panel vyzeral a akým spôsobom bol responzívny pre mobilné telefóny. S knižnicou Antd sa nám to nepodarilo docieliť, avšak pripúšťame, že mohlo ísť o nedostatočnú znalosť knižnice, poprípade jej nedostatočnú dokumentáciu.

3.8 Ant Design

Ant Design je kolekciou komponentov pre návrh používateľského rozhrania a implementáciu React. Antd poskytuje množstvo funkcií pre vizuálny návrh aplikácie. Alternatívou k Antd je Material UI, navrhnutý spoločnosťou Google v roku 2014.

Narozdiel od Antd ponúka aj platenú verziu, ktorá obsahuje väčšiu škálu komponentov.

Obe technológie poskytujú kvalitné stavebné prvky na vytváranie jednoduchých a intuitívnych rozhraní aplikácií. Ant Design aj Material UI založené na React.js, a teda sú vhodné na vytváranie rozhraní špeciálne pre jednostránkové aplikácie. Aplikáciu CodeSandbox môžete použiť na náhľad komponentu spolu so zdrojovým kódom a je súčasťou opisu komponentu v dokumentácii.

Priestorové usporiadanie je východiskovým bodom systematického vizuálneho dizajnu. Rozdiel oproti tradičnému grafickému dizajnu spočíva v tom, že priestorové rozloženie používateľského rozhrania by malo byť založené na dynamickom a systematickom pohľade. Ant Design je vytvorený na základe princípu `krásy poriadku`, čo návrhárom frontendovej aplikácie umožňuje vytvoriť user-friendly priestorové rozvrhnutie.

Pri definovaní systému rozloženia vo vizuálnom systéme navrhujeme vychádzať z týchto 5 aspektov [13] :

- Jednotný rozmer plátna – S cieľom minimalizovať náklady na komunikáciu je potrebné zjednotiť rozmery (výšku a šírku) návrhovej dosky v rámci organizácie.
- Prispôsobenie – Existujú dva typické typy prispôsobenia: ľavo-pravé rozloženie a rozloženie zhora nadol. V ľavo-pravom je bežnou praxou zafixovať ľavý navigačný panel a dynamicky škálovať pravú pracovnú oblasť. V druhom type definujeme minimálnu hodnotu pre okrajové oblasti na oboch stranách. Po dosiahnutí hraničnej hodnoty okrajovej oblasti sa dynamicky zmenší medziplocha hlavného obsahu.
- Jednotka mriežky – Základná jednotka mriežky je 8, čo zodpovedá väčšine bežných zobrazovacích zariadení.
- Raster – Ant Design používa architektúru s 24 mriežkami, kde každá mriežka reprezentuje 1/24 obrazovky
- Spoločné mierky – S cieľom pomôcť dizajnérom rôznych úrovní dosiahnuť konzistenciu pri navrhovaní stránky navrhla spoločnosť Ant Design koncept spoločných mier UI

3.9 UX design

Používateľské rozhranie sa dá chápať ako forma dialógu. Presné a jasné slová sú ľahšie porozumiteľné a vhodný tón dokáže vzbudovať pocit dôvery. Preto by sa pri návrhu rozhrania mal brať copywriting vážne. Pri vytváraní obsahu je potrebné si uvedomiť niekoľko osvedčených pravidiel, medzi ktoré patria napríklad [13] :

- Uvažovať z pohľadu používateľa – pri vyjadrovaní obsahu by sa mal klásť dôraz na to, čo môžu s obsahom urobiť používatelia.
- Vyjadrovať sa dôsledne – Vo väčšine situácií nie je nutné, aby UI popisovalo všetky podrobnosti. Vo všeobecnosti je preferovaný krátky a výstižný obsah. Odborné výrazy by mali korešpondovať štandardom v danej oblasti
- Umiestniť dôležité informácie na viditeľné miesta – dôležité informácie navrchu, väčšia veľkosť písma
- Použiť vhodný jazyk – ten istý obsah môže byť pre rôznych používateľov vyjadrený rôznymi spôsobmi
- Umiestňovať dôležité informácie na viditeľné miesta

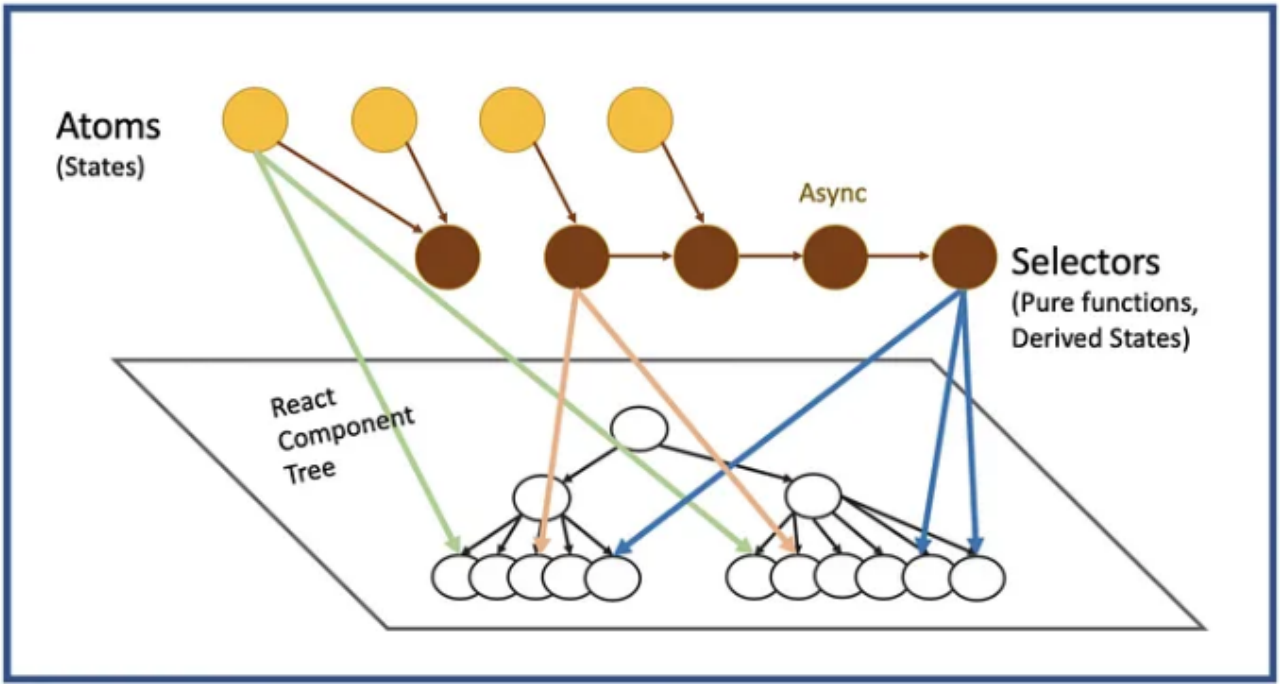
Knižnica Antd vo svojej dokumentácii uvádza viaceré návody pre copywriting, podľa ktorých odporúča postupovať pri návrhu frontendových riešení. Aplikovaním týchto zaužívaných patternov sa užívateľ jednoduchšie orientuje v systéme a v neposlednom rade na neho pôsobí používanie webu celkovo lepší dojem.

3.10 State management

Ako sme v kapitole o React-e spomínali, jednou z kľúčových vlastností je tvorba komponentov. Komponenty často krát používajú rovnaké dáta, ktoré je potrebné načítať z API. S rastúcou veľkosťou aplikácie môže nastať situácia, kde celá aplikácia beží pomaly preto, že každý komponent neustále odosiela požiadavky na backend a aktualizuje svoj stav. Moderné frontendové aplikácie preto využívajú state management, ktorý zabazpečuje jednotnosť dát, ich vzájomné zdieľanie a zároveň redukuje počet volaní na backend. Existujú rôzne implementácie pre state management, pričom najpoužívanejšia je Redux. My sme si vybrali Recoil, ktorý je experimentálnou knižnicou od Facebooku a v budúcnosti by mohol byť vhodným riešením pre mnohé projekty pre jeho rýchlosť a taktiež jednoduchosť. Z dokumentácie Recoilu sme usúdili, že pre vytvorenie rovnakej funkcionality je potrebné napísanie výrazne menšieho množstva kódu než v Reduxe. Takisto Recoil považujeme za jednoduchší nástroj na pochopenie ako Redux.

3.10.1 Recoil

Recoil umožňuje vytvoriť graf toku dát, ktorý prúdi od atómov (zdieľaný stav) cez selektory (čisté funkcie) až po komponenty v React-e. Atómy sú jednotky stavu, z ktorých môžu komponenty čítať a aktualizovať dáta. Selektory transformujú tento stav buď synchrónne, alebo asynchrónne. Podľa Elad Elrom-a, Recoil sa stane štandardom pre state management v Reacte a stojí za námahu sa ho naučiť než pokračovať v podpore reduxu a middlewaru [14] .



Obrázok 3.2. Zjednodušený graf reprezentujúci fungovanie state managementu [15] .

3.10.2 Atómy

Atómy sú jednotky stavu. Dajú sa aktualizovať a čítať: keď sa aktualizuje atóm, každá komponenta, ktorá číta hodnotu atómu sa znovu vykreslí s novou hodnotou. Môžu sa vytvárať aj za behu. Atómy možno použiť namiesto lokálneho stavu React

komponentov. Ak sa ten istý atóm použije z viacerých komponentov, všetky tieto komponenty zdieľajú ich stav.

Atómy potrebujú jedinečný kľúč, ktorý sa používa na ladenie, perzistenciu a pre niektoré pokročilé API, ktoré umožňujú zobraziť mapu všetkých atómov. Podobne ako stavy komponentov React, aj ony majú predvolenú hodnotu.

3.10.3 Selektory

Selektor je čistá funkcia, ktorá prijíma atómy alebo iné selektory ako vstup. Keď sa tieto vstupné atómy alebo selektory aktualizujú, funkcia selektora sa prepočíta. Komponenty môžu čítať hodnotu selektorov rovnako ako atómy a potom byť pri zmene selektorov znovu vykreslené.

Selektory sa používajú na výpočet odvodených údajov, ktoré sú založené na stave. To nám umožňuje vyhnúť sa nadbytočnému stavu, pretože v atómoch je uložená minimálna množina stavu, zatiaľ čo všetko ostatné sa efektívne vypočíta ako funkcia tohto minimálneho stavu. Keďže selektory sledujú, ktoré komponenty ich potrebujú a od akého stavu závisia, robia tento funkčný prístup veľmi efektívnym. Ak máme napríklad pole 3 objektov, ktoré sa načítali z API a chceme vymazať jeden z nich, normálne by sme museli odoslať DELETE request na vymazanie a následne odoslať GET request na vrátenie aktuálneho stavu poľa. S využitím selektora jednoducho odošleme DELETE request a zmeníme vnútorný stav aplikácie (vymažeme objekt z pôvodného listu). Týmto spôsobom redukujeme nadbytočnú zátaž na backend.

3.11 Komunikačné rozhranie

Nasledujúca podkapitola vysvetľuje spôsob komunikácie medzi frontendovou a backendovou časťou aplikácie FactCheck. Popisuje protokol HTTP, rozhranie REST a formát výmeny údajov JSON. Backendovú časť aplikácie mal na starosti Bc. Rastislav Kopál. Pri tvorbe rozhrania sme tiež zistili, akú kritickú úlohu zohráva použitie TypeScriptu. Vďaka nemu sme mohli vytvoriť komunikáciu, ktorá pracuje s pevne štruktuovanými dátami.

3.11.1 HTTP

Protokol HTTP (Hypertext Transfer Protocol) je všeobecný, bezstavový, objektovo orientovaný protokol aplikačnej vrstvy pre distribuované, hypermediálne informačné systémy [16].

Protokol HTTP ako protokol aplikačnej vrstvy je nad transportnou vrstvou a pod prezentačnou vrstvou. Bol navrhnutý začiatkom 90. rokov 20. storočia a je rozšíriteľným protokolom, ktorý sa časom vyvíjal.

Na internete sa komunikácia HTTP zvyčajne uskutočňuje prostredníctvom protokolu TCP/IP[16]. Štandardne sa používa port 80, ale môže byť použitý aj iný.

Protokol HTTP je založený na princípe požiadavky a odpovede. Klient nadviaže spojenie so serverom a odošle mu požiadavku vo forme metódy požiadavky, URI, verzie protokolu a dodatočnými požiadavkami (napríklad telo). Server odpovie klientovi správou so statusom , ktorý obsahuje verziu protokolu a HTTP kódom úspešnosti, po ktorom nasleduje správa podobná MIME obsahujúca informácie o serveri, metainformácie o entite a možný obsah tela [16].

3.11.2 REST

REST (REpresentational State Transfer) je architektúra založená na webových štandardoch, ktorá používa protokol HTTP. Rest je orientovaný dátovo, pričom každá zložka je zdrojom a zdroj je prístupný cez spoločné rozhranie pomocou štandardných metód HTTP. REST prvýkrát predstavil Roy Fielding v roku 2000.

V architektúre REST poskytuje server prístup k zdrojom a klient REST pristupuje k zdrojom a upravuje ich. Každý zdroj je identifikovaný pomocou URI a ID. V RESTe definujeme 4 základné metódy:

- GET – poskytuje prístup na čítanie zdrojov.
- POST – používa sa na vytvorenie nového zdroja.
- DELETE – slúži na odstránenie zdroja.
- PUT – slúži na aktualizáciu existujúceho alebo vytvorenie nového zdroja.

3.11.3 JSON

JavaScript Object Notation (JSON) je light-weight textový formát na výmenu údajov, ktorý umožňuje jednoduché zapisovanie a čítanie dát. Formát JSON sa skladá z dvojíc názov : hodnota. Hodnota môže byť buď objekt alebo pole. Hodnota môže byť reťazec (v dvojitých úvodzovkách), číslo, logická hodnota (true alebo false), null, objekt alebo pole. Objekt je neusporiadaná množina dvojíc názov/hodnota, uzavretá kučeravými zátvorkami . Pole je usporiadaná kolekcia hodnôt uzavretá hranatými zátvorkami []. Tieto dve štruktúry sa môžu vkladať do seba, čo umožňuje jednoduchú reprezentáciu zložitých dátových štruktúr [17].

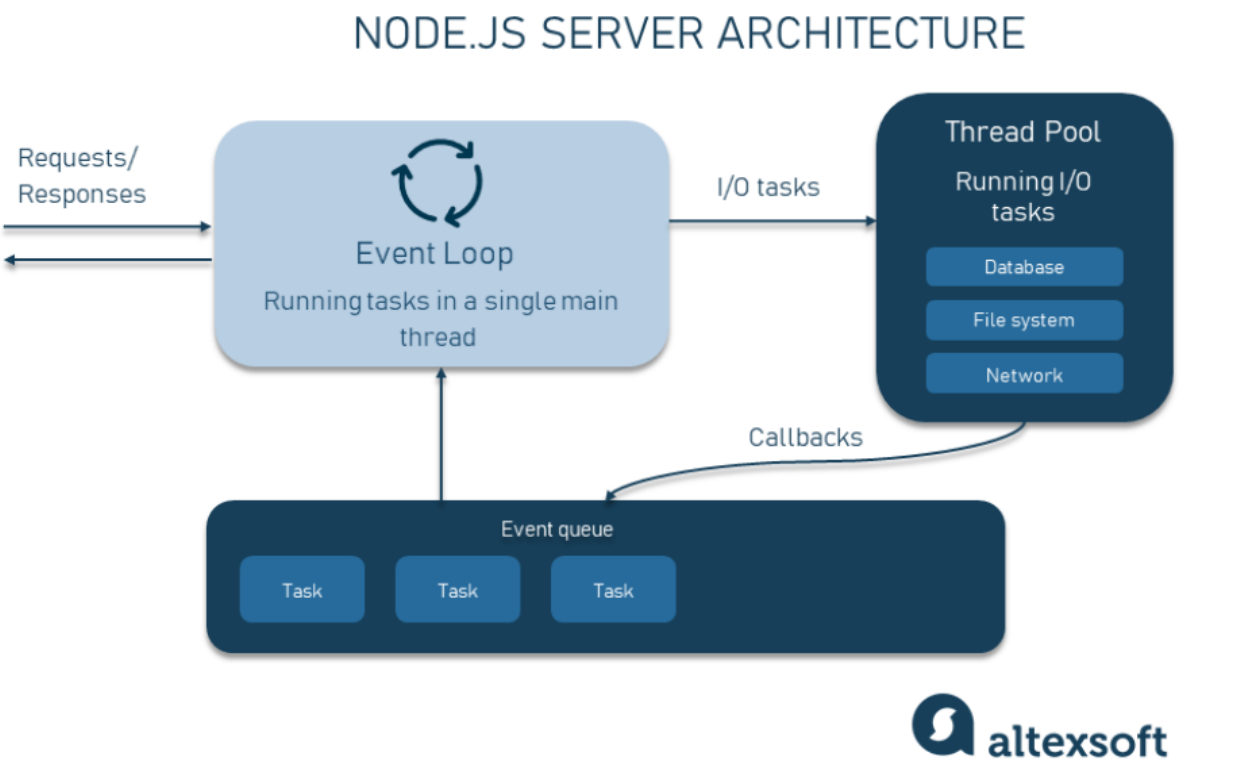
Vďaka tomu je ideálnym formátom na prenos údajov medzi serverom a webovou aplikáciou, pretože ho možno ľahko používať pomocou jazyka JavaScript na strane klienta. Používa sa v rozhraniach API RESTful na odosielanie a prijímanie údajov. Pred JSONom bol primárnym formátom na výmenu údajov XML, avšak JSON vďaka jednoduchému používaniu stáva štandardom v komunikačných rozhraniach. Podľa Nurseitova a kol. je JSON taktiež rýchlejším formátom výmeny ako XML [18].

3.12 NodeJS

Node.js je open-source runtime prostredie, založené na JavaScriptovom engine, napísané v jazykoch JavaScript a C/C++. Node.js spúšťa kód JavaScript na strane servera pomocou just-in-time kompilácie. Používa sa na vývoj backendu a poskytuje prostredie pre vývoj API v systémoch Linux, Windows a OS X.

3.12.1 Vlastnosti NodeJS

Medzi hlavné vlastnosti NodeJS patria neblokované I/O operácie a asynchrónne spracovanie požiadaviek. Pri tvorbe web aplikácií je bežnou požiadavkou, že funkcia potrebuje načítať údaje zo siete, spracovať ich a potom vrátiť výsledok. Pri synchrónnom spracovaní to znamená, že aplikácia by musela čakať, kým funkcia dostane údaje a vykoná svoju prácu, čo by v konečnom dôsledku zablokovalo iné operácie. V Node.js beží JavaScript na jednom vlákne a spracovanie je asynchrónne. Namiesto čakania na dokončenie funkcie po načítaní údajov teda Node.js spustí ďalšie požiadavky, pričom tie I/O (ako načítanie údajov) spracuje na pozadí [19]. Táto funkcia je umožnená vďaka spätným volaniam a promises.



Obrázok 3.3. Serverová architektúra NodeJS [20]

Spätné volania sú funkcie, ktoré sa volajú pre I/O operácie po ich dokončení. Môžu byť pridané do fronty udalostí a po jeho vyčistení obslúžené v hlavnom vlákne. Spätné volania môžu byť vnorené do iných spätných volaní, čo komplikuje vykonávanie kódu. Novším prístupom k spracovaniu asynchrónneho kódu je použitie promises. Ide o objekty vo funkciách, ktoré namiesto čakania na vrátenie hodnoty `sľubujú` jej vrátenie neskôr, zatiaľ čo sa vykonávajú iné operácie.

Na udržanie behu procesu má uzol Node slučku udalostí [19] . Slučka udalostí zhromažďuje nové spätné volania a pýta sa na nové prichádzajúce požiadavky z frontu udalostí, keď sú ostatné operácie dokončené, následne sa slučka opakuje znova.

3.13 Voľba typu databázy - SQL, NoSQL

Pre účely vývoja našej aplikácie bolo potrebné zvoliť vhodný typ databázy pre ukladanie údajov. Existujú dva typy databáz : SQL a NoSQL. Z našich získaných znalostí z predmetu Databázové systémy 2 sme usúdili, že využitie NoSQL databázy s JSON štruktúrou by mohlo byť vhodným riešením. Najprv však zhodnotíme výhody a nevýhody NoSQL databáz.

3.13.1 Výhody NoSQL

- Flexibilné dátové modely – NoSQL databázy majú zvyčajne veľmi flexibilné schémy. Flexibilná schéma umožňuje jednoducho vykonávať zmeny v databáze podľa toho, ako sa menia požiadavky. Objekty môžu obsahovať rôzne atribúty, to v prípade SQL nie je možné
- Rýchle dotazy – Údaje v databázach SQL sú zvyčajne normalizované, takže pri dotazoch na jeden objekt alebo entitu musíme spájať údaje z viacerých tabuliek. S rastúcou veľkosťou tabuliek sa spájanie môže stať náročnejšie. Údaje v databázach NoSQL sú však zvyčajne uložené spôsobom, ktorý je optimalizovaný pre dotazy. Pri používaní MongoDB platí pravidlo, že údaje, ku ktorým sa pristupuje spoločne, by mali byť uložené spoločne. Dotazy zvyčajne nevyžadujú spájanie, takže dotazy sú veľmi rýchle [21]
- Horizontálne škálovanie – Väčšina databáz SQL vyžaduje vertikálne škálovanie (migráciu na väčší server). Naopak, väčšina databáz NoSQL umožňuje horizontálne škálovanie, čo znamená, že môžeme kedykoľvek pridať viac „lacných“ serverov
- Jednoduché pre vývojárov – Niektoré databázy NoSQL, ako napríklad MongoDB, mapujú svoje dátové štruktúry na štruktúry populárnych programovacích jazykov. Toto mapovanie umožňuje vývojárom ukladať údaje rovnakým spôsobom, akým ich používajú v kóde svojej aplikácie. Hoci sa to môže zdať ako triviálna výhoda, toto mapovanie môže vývojárom umožniť napísať menej kódu, čo vedie k rýchlejšiemu vývoju a menšiemu počtu chýb

3.13.2 Nevýhody NoSQL

- nepodporujú transakcie ACID – atomicita, konzistencia, izolácia, trvanlivosť. Na riešenie týchto prípadov použitia pridala MongoDB vo verzii 4.0 podporu pre transakcie ACID s viacerými dokumentmi a vo verzii 4.2 ich rozšírila tak, aby zahŕňali zdieľané clustre [21].
- Zrelosť – SQL databázy existujú výrazne dlhšie a preto existuje viac nástrojov pre manipuláciu s nimi
- Konzistentnosť údajov – Databázy NoSQL používajú model eventuálnej konzistencie, čo znamená, že databázy nemusia okamžite zobrazovať aktualizované údaje. To môže viesť k problémom v aplikáciách, kde sa vyžaduje silná konzistencia.
- Na voľbu NoSQL databázy je potrebné mať dobrú znalosť konkrétnej technológie. Existujú napríklad grafové databázy, ktoré sú vysoko efektívne pre analýzu vzťahov, ale nemusia byť vhodnou voľbou pre štandardné dotazy

3.14 MongoDB

Node.js, React aj MongoDB natívne podporujú JSON (JavaScript Object Notation). Ako sme kapitole o JSONe spomínali, JSON sa ľahko číta a zapisuje, takže je ideálnou voľbou na komunikáciu medzi serverom a klientom [22]. MongoDB ukladá údaje vo formáte podobnom JSON - nazývanom BSON (Binary JSON). Tento formát umožňuje bezproblémovú integráciu s Node.js. Táto kompatibilita zjednodušuje prácu s údajmi a eliminuje potrebu zložitého mapovania údajov.

Podľa dokumentácie MongoDB je MongoDB vhodnou voľbou pre aplikácie napísané v Node.js a Reacte a spolu s Express.js tvorí tzv. MERN stack (naša aplikácie je teda tiež MERN) [21] . MongoDB ponúka pre naše potreby vhodný dopytovací

jazyk, ktorý podporuje komplexné vyhľadávanie a manipuláciu s údajmi vrátane filtrovania, agregácie a textového vyhľadávania. Z týchto dôvodov sme sa rozhodli pre použitie tohto typu databázy.

3.15 Autentifikácia

Pre potreby bezpečnej komunikácie klienta so serverom je potrebné vytvorenie systému, ktorý bude vedieť po prihlásení identifikovať používateľa pri každej požiadavke na server. Najľahším spôsobom je posielanie mena a hesla v hlavičke požiadavky, avšak tento spôsob nie je bezpečný.

JWT tokeny poskytujú bezpečnú a efektívnu metódu prenosu údajov medzi stranami. Pochopením ich štruktúry a toho, ako prístupové tokeny a tokeny obnovenia fungujú, môžeme vytvoriť bezpečné systémy overovania a autorizácie a preto sme sa rozhodli túto technológiu použiť v našom projekte.

3.15.1 JWT tokeny

JWT token je samostatný, digitálne podpísaný token, ktorý sa skladá z troch častí: hlavičky, užitočného zaťaženia a podpisu. Hlavička obsahuje informácie o type tokenu a algoritme použitom na podpisovanie. V užitočnom zaťažení sa nachádzajú skutočné údaje, ktoré sa majú bezpečne preniesť, a podpis sa generuje pomocou tajného kľúča na overenie pravosti tokenu.

Prístupový token poskytuje abstrakciu, ktorá nahrádza rôzne autorizačné konštrukcie (napr. používateľské meno a heslo, tvrdenie) pre jediný token, ktorému rozumie server zdrojov. Táto abstrakcia umožňuje vydávanie prístupových tokenov platných na krátke časové obdobie a odstraňuje potrebu servera zdrojov rozumieť širokému spektru autentifikačných schém.

3.15.2 Štruktúra tokenu JWT

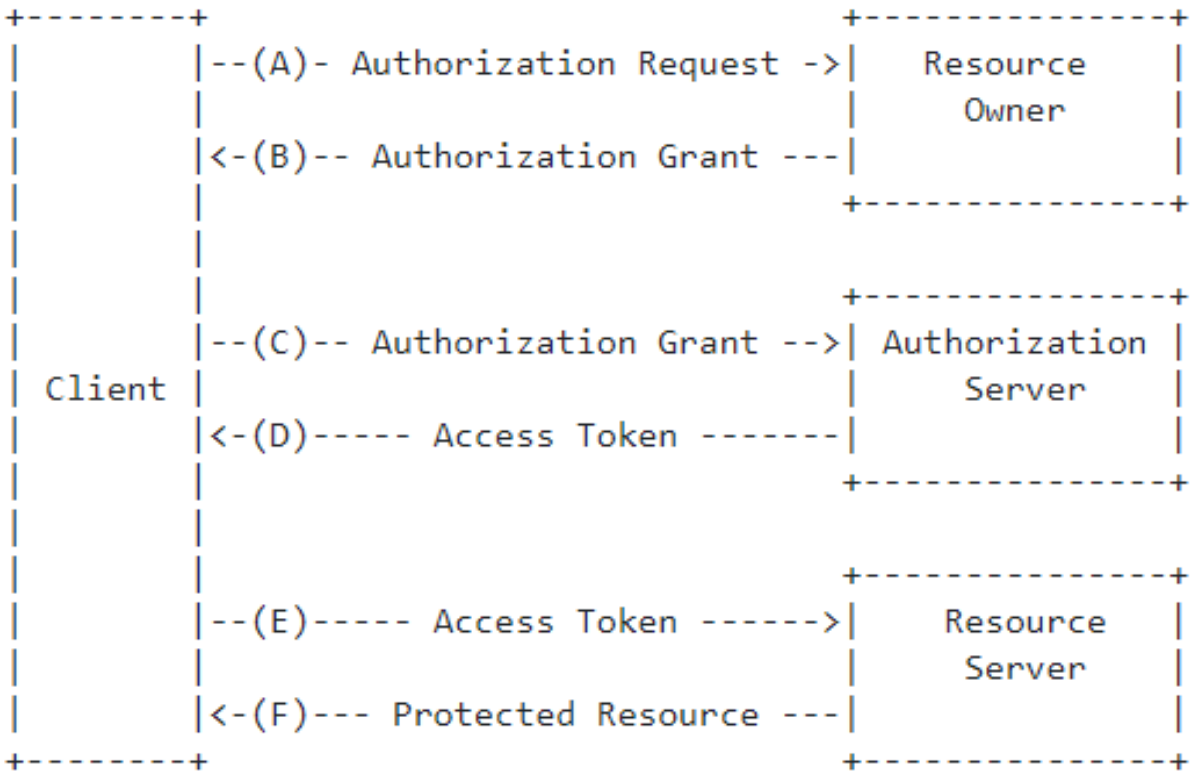
Tokeny JWT majú špecifický formát: hlavička.payload.podpis. Každá časť je zakódovaná v base64Url a potom spojená s oddeľovačom bodiek (`.`). Táto štruktúra umožňuje jednoduchý prenos tokenov JWT prostredníctvom adresy URL, parametrov POST alebo hlavičiek HTTP.

Payload - známe aj ako claim, obsahuje prenášané údaje. Obsahuje štandardné tvrdenia, ako napríklad `iss` (emitent), `sub` (subjekt) a `exp` (čas vypršania platnosti), ako aj vlastné tvrdenia špecifické pre aplikáciu. Podpis - vytvorí sa zakódovaním hlavičky a užitočného zaťaženia, ich spojením s bodkou (`.`) a následným podpísaním tohto reťazca pomocou tajného kľúča so špecifikovaným algoritmom. Prístupové tokeny a tokeny obnovenia

Header - hlavička sa zvyčajne skladá z dvoch častí: typu tokenu, ktorým je JWT, a použitého algoritmu, napríklad HMAC SHA256 alebo RSA SHA256. Je zakódovaný v Base64Url a tvorí prvú časť JWT.

Payload obsahuje trdenia. Máme k dispozícii teda súbor tvrdení, napríklad: iss (emitent), exp (čas platnosti), sub (subjekt) a aud (publikum). Tieto tvrdenia nie sú povinné, ale odporúčané, aby poskytovali systému informácie o konkrétnom užívateľovi. Payload môže obsahovať aj ďalšie atribúty, ktoré definujú vlastné tvrdenia, napríklad rolu užívateľa. Zvyčajne sa na vytvorenie subjektu používateľa OpenID Connect používa tvrdenie subject (subjekt). Payload je zakódovaný v Base64Url a tvorí druhú časť JWT.

Na vytvorenie podpisovej časti sa zakódovaná hlavička a zakódovaný payload podpíšu pomocou podpisového algoritmu z hlavičky. Podpis sa používa na overenie, že vydavateľ JWT je ten, za koho sa vydáva, a na zabezpečenie toho, že správa nebola po ceste zmenená.



Obrázok 3.4. Abstraktný tok protokolu [23]

V autentifikačných systémoch sa JWT často používajú ako prístupové tokeny a obnovovacie tokeny. Prístupové tokeny sú krátkodobé tokeny, ktoré umožňujú používateľom prístup k chráneným zdrojom, zatiaľ čo obnovovacie tokeny sú tokeny s dlhšou životnosťou, ktoré sa používajú na získanie nových prístupových tokenov po uplynutí ich platnosti.

Prístupové tokeny sa zvyčajne vydávajú pri prihlásení používateľa a poskytujú mu prístup k určitým zdrojom na obmedzený čas. Po uplynutí platnosti už používateľ nemá prístup k chráneným zdrojom. Hlavnou výhodou krátkodobých prístupových tokenov je, že minimalizujú riziko neoprávneného prístupu v dôsledku krádeže tokenu.

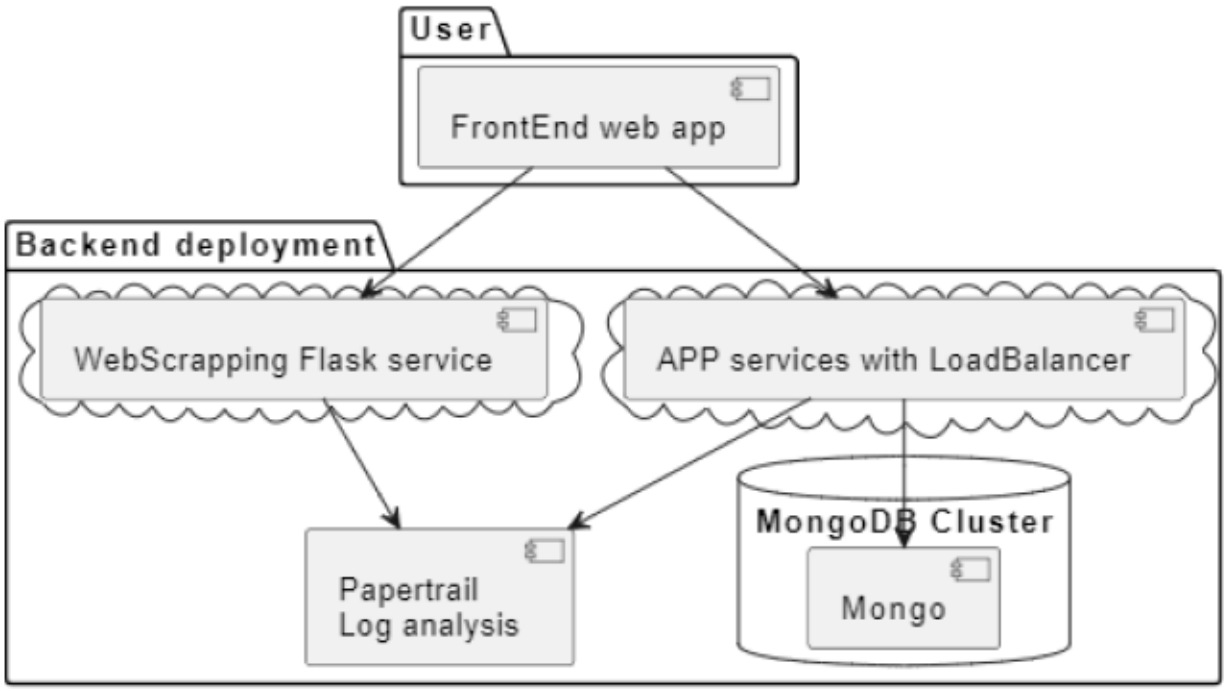
Obnovovacie tokeny sa používajú po vypršaní platnosti prístupového tokenu. Namiesto toho, aby sa používateľ musel znova prihlásiť, môže obnovovací token vymeniť za nový prístupový token. Tento proces sa uskutočňuje na pozadí, čím sa zabezpečí plynulý používateľský zážitok. Obnovovacie tokeny sú zvyčajne bezpečne uložené na serveri a vystavujú sa len v prípade potreby.

Kapitola 4
Architektúra aplikácie

Aplikácia sa skladá z dvoch hlavných modulov: user (client-side) a backend. Tieto dve časti spolu komunikujú pomocou HTTPS protokolu. Frontendová časť je implementová ako React aplikácia, backendová časť je implementovaná ako RESTful servis. Našou úlohou bolo vytvorenie frontendovej časti aplikácie.

Aplikácie je postavený na architektúre mikroservisov [24] , pričom obsahuje dva servisy. Prvý servis (App services with LoadBalancer) je naprogramovaný vo frameworku NodeJS a vykonáva logiku aplikácie FactCheck. Logika spočíva v spracovávaní požiadavkov na autentifikáciu používateľov a na manipuláciu so zdrojmi (články, tvrdenia, hodnotenia, ...). Druhý servis je python aplikácia trafilatura, ktorá má na starosti transformáciu URL odkazu článku na titulok a čistý text, ktorý následne vložíme do formulára na pridanie článku. Našou úlohou je vytvorenie frontendovej časti aplikácie, ktorá bude s týmito servisami správne komunikovať.

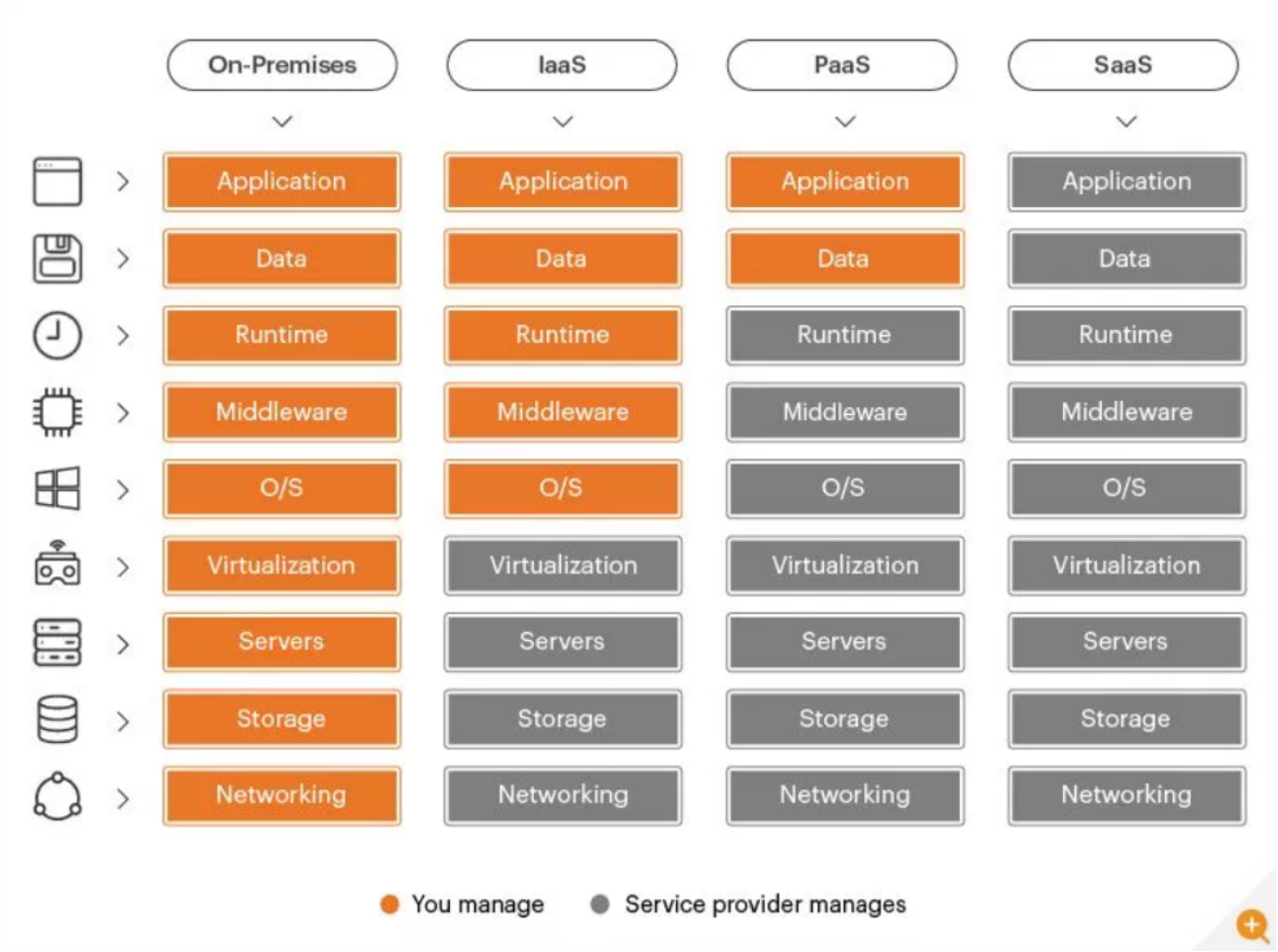
Ďalšou súčasťou backendu je databáza Mongo, ktorá obsahuje dáta z nášho portálu. . Poslednou súčasťou je logovací framework Papertrail, ktorý ukladá logy zo servisov za účelom sledovania ich správneho fungovania.



Obrázok 4.1. Schéma architektúry aplikácie

4.1 Nasadenie na server + GitHub

Platforma ako služba (PaaS), softvér ako služba (SaaS) a infraštruktúra ako služba (IaaS) sú tri základné modely služieb cloud computingu, z ktorých každý poskytuje používateľom rôzne úrovne abstrakcie a kontroly. V nasledujúcej kapitole rozoberieme tieto tri modely a zhodnotíme výber technológie pre náš projekt.



Obrázok 4.2. Schéma modelov služieb cloud computingu. [25]

- IaaS – je model poskytovania cloudových služieb, ktorý poskytuje používateľom úplnú kontrolu nad svojou infraštruktúrou bez nutnosti udržiavať fyzický hardvér. Poskytuje cloudové výpočtové zdroje ako sú siete, úložiská a iné hardwarové komponenty. Architekti si môžu zvoliť konkrétne komponenty, ktoré pre chod systému potrebujú. IaaS sa člení na tri hlavné časti: výpočtovú, sieťovú a úložnú. Nevýhodou tohoto typu architektúry je, že je potrebné presne nakonfigurovať každý zdroj a optimalizovať výber hardwaru pre požiadavky aplikácie.
- PaaS – je model, v ktorom poskytovateľ cloudu poskytuje všetku backendovú infraštruktúru vrátane sietí, middlewaru, serverov, úložísk a virtualizácie prostredia. To znamená, že používatelia môžu očakávať vopred nakonfigurované prostredia na spúšťanie a predvídateľné možnosti škálovania, ukladania a zabezpečenia. Používatelia majú prístup k rôznym nástrojom a konfiguračným nastaveniam, ale nemôžu meniť základný operačný systém alebo sieťové nastavenia. DigitalOcean App Platform je príkladom ponuky PaaS. Umožňuje používateľom rýchlo a jednoducho vytvárať, nasadzovať a automaticky škálovať aplikácie bez

prerušenia. PaaS je optimalizovaná skôr na jednoduché nasadenie než na správu komplexnej infraštruktúry.

- SaaS – je model, pri ktorom softvérovú aplikáciu poskytuje dodávateľ tretej strany prostredníctvom internetu. Aplikácie SaaS sa vytvárajú v cloudových infraštruktúrach a sú prístupné odkiaľkoľvek s pripojením na internet. Zo služieb IaaS, PaaS a SaaS poskytuje SaaS používateľovi najviac vrstiev abstrakcie. Poskytuje prístup k softvéru, ale nie jeho tvorbu, údržbu alebo úpravu. Používatelia sú vystavení len rozhraniu, s ktorým komunikujú. Medzi príklady produktov SaaS patria Google Docs, Slack, Adobe Creative Suite a Office 365. Pri použití SaaS na vytvorenie a údržbu aplikácie môže byť náročné zmeniť poskytovateľa. V závislosti od služby to môže znamenať rozsiahlu migráciu, stratu údajov alebo úplne nové nastavenia.

4.2 Digitalocean app

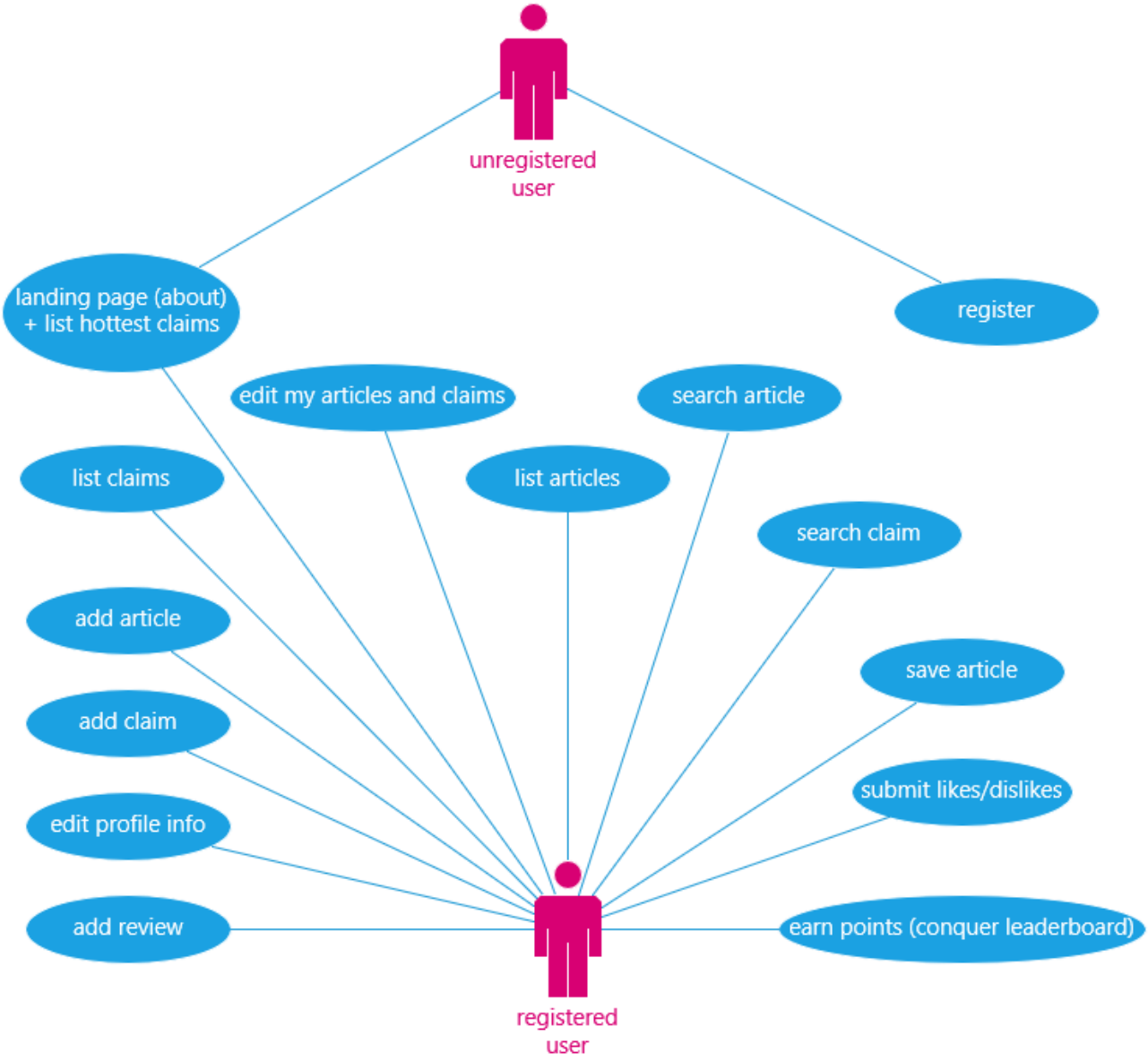
Pre naše potreby agilného vývoja sme si vybrali typ nasadenia PaaS, vďaka ktorému je možné ľahko prepojiť microservice architektúru našich služieb. Aplikácia je nasadená na serveri od spoločnosti DigitalOcean, ktorý ponúka cloudové riešenia pre rôzne typy služieb. DigitalOcean sme si vybrali, pretože ponúka platformu na správu rôznych typov aplikácií s prepojením na databázu a na stránke je k dispozícii veľa návodov, ktoré krok za krokom popisujú postup nasadenia a konfigurácie všetkých jej komponentov. Na webe DigitalOcean sú taktiež k dispozícii monitorovacie služby, ktoré generujú analýzy záťaže cpu, pamäte, počte reštartov a iných vecí. V prípade budúcej poteby je tiež možné nakonfigurovať premenné prostredia, DNS, load balancer a ďalšie funkcie.

4.3 Prepojenie s GitHubom a CI/CD

V aplikácii DigitalOcean je implementovaná CI/CD, ktorá zabezpečuje synchronizáciu medzi repozitármi na GitHube a po vložení zmien do hlavnej branche buildu je novú verziu komponentu. Zbuildovanie je implementované so zero-downtime, takže aj počas kompilovania novej verzie môžu užívatelia aplikácie naplno využívať funkcionalitu starej verzie.

Kapitola 5
Funkcionalita aplikácie

Na nasledujúcom obrázku môžete vidieť graf, ktorý popisuje jednotlivé funkcie systému. Na hornej strane je figúrka reprezentujúca registrovaného užívateľa, na dolnej strane neregistrovaného užívateľa. Obláčiky reprezentujú aktivity, ktoré užívateľ môže na webe vykonávať. Na tvorbu use-case diagramu sme použili software Visio od spoločnosti Microsoft.



Obrázok 5.1. Use-case diagram funkcionality na frontendovej časti aplikácie.

5.1 Systémové požiadavky

Systémové požiadavky definujú súbor atribútov, ktoré musí náš systém spĺňať. Systémové požiadavky sa rozdeľujú na funkčné a nefunkčné. V nasledujúcej kapitole rozoberieme oba typy a definujeme všetky požiadavky pre oba typy požiadaviek.

5.1.1 Funkčné požiadavky

Funkčné požiadavky sú schopnosti, ktoré systém musí spĺňať aby vyriešil problém [26]. Funkčné požiadavky zahrňujú teda všetky akcie, ktoré naša frontendová aplikácia musí spĺňať aby vyhovovala našim kritériám. Medzi naše funkčné požiadavky patria:

- Autentifikácia – formulár pre prihlásenie a registráciu, použitie šifrovania a technológií ako JWT v autorizačnej hlavičke pre bezpečnú komunikáciu so serverom
- Úvodná strana – v prvej verzii aplikácie Fact-Check užívatelia občas nechápali, ako použiť systém. Táto sekcia slúži užívateľom pre prvý kontakt s našou aplikáciou a umožní im jednoducho pochopiť fungovanie systému
- Pridávanie článkov – hlavnou funkciou portálu je crowd-sourcing. Na pridávanie článkov vytvoríme dotazník s možnosťou stiahnutia dát z externej URL, ktorú užívateľ zadal
- Pridávanie tvrdení – ku každému článku môže hociktorý používateľ pridávať tvrdenia, ktoré z neho vyplývajú a nie je si istý ich pravdivosťou
- Vytváranie hodnotení – užívateľ môže hodnotiť ľubovoľné tvrdenia a reagovať na ne. Aby bolo možné overiť pravdivosť jeho odpovede, je potrebné zbierať pôvod týchto tvrdení (URL adresy).
- Zviditeľnenie zaujímavého obsahu – vytvorenie spôsobu, vďaka ktorému sa bude zaujímavý obsah zobrazovať v populárnom obsahu
- Gamifikácia – získavanie bodov za aktivitu na portáli. Užívateľ získava body tým, že pridáva obsah (články, tvrdenia, hodnotenia) a reaguje na ne. Na protáli vytvoríme tzv. Leaderboard - rebríček najaktívnejších overovateľov faktov. Účelom gamifikácie je vytvoriť kompetitívne prostredie v našej aplikácii a docieliť tým zvýšenie interakcií s našou aplikáciou
- Úprava profilu – v prípade, že užívateľ potrebuje z nejakého dôvodu zmeniť svoje osobné údaje alebo upraviť obsah, ktorý pridal, je potrebné vytvorenie sekcie profilu

5.1.2 Nefunkčné požiadavky

Nefunkčné požiadavky sú opakom funkčných požiadaviek a slúžia na vymedzenie množiny riešení podľa vnútorných vlastností. Zjednodušene povedané - špecifikujú, ako má systém fungovať. Nefunkčné požiadavky sú napríklad znovupoužiteľnosť elementov, miera dostupnosti systému, požiadavky na výkon a podobne. Medzi naše nefunkčné požiadavky patrí:

- Grafické spracovanie – keďže je aplikácia určená pre bežných užívateľov, je dôležité, aby pôsobila graficky prívetivo
- Responzivita – chceme zabezpečiť, aby sa dala aplikácia použiť aj na mobilných zariadeniach a preto je potrebné zvoliť vhodné komponenty a správne nastaviť ich CSS
- Kompatibilita s prehliadačmi – používatelia môžu používať rôzne prehliadače, preto chceme zaistiť, aby na najpoužívanejších typoch prehliadačov fungovali správne

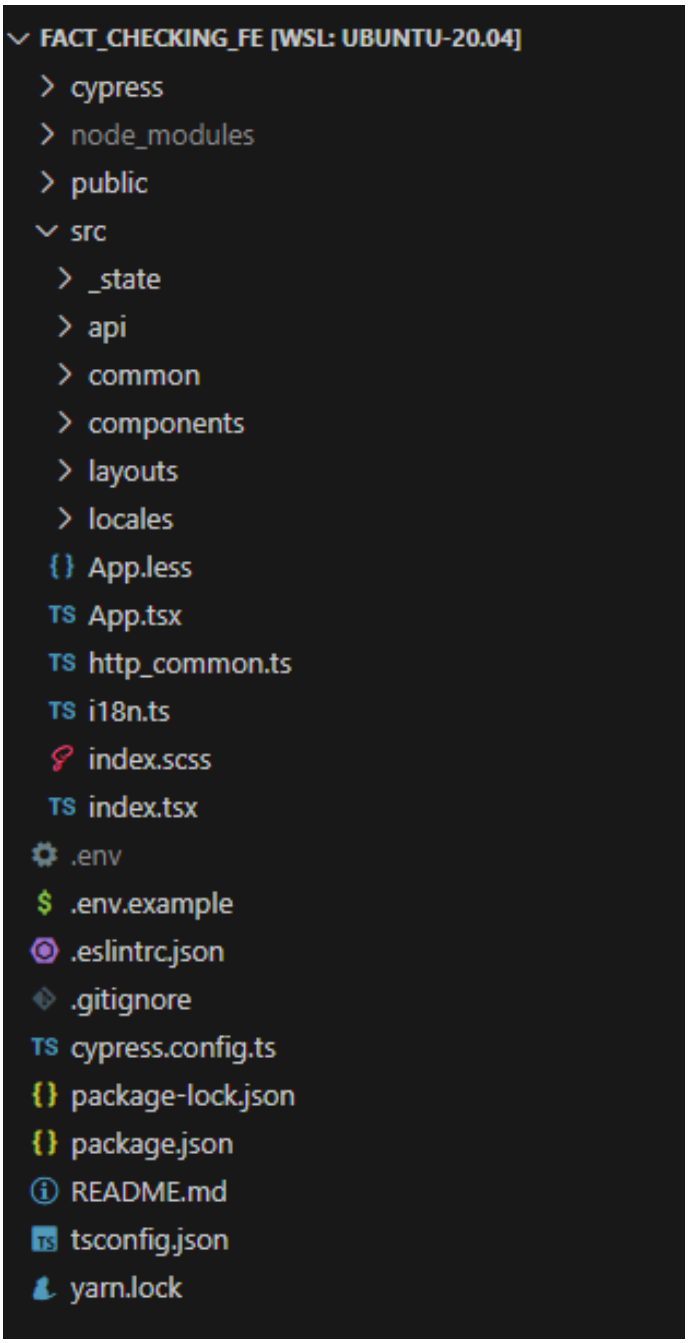
5.2 Základné dátové štruktúry systému

- Article (článok) – je základným zdrojom informácií, z ktorého užívatelia čerpajú fakty. Môže to byť článok z internetu, knihy, časopisu, alebo iného zdroja informácií. Článok obsahuje nadpis, text a jazyk. K článku je potrebné uviesť zdroj (resp. URL adresu). Po vyplnení URL adresy zdroja a kliknutí na `Load text from URL` je možné stiahnuť text a nadpis pomocou externej API webscraping service, ktorá je potrebná pre účely strojového spracovania. Článok, ktorý nás zaujal, môžeme označiť ikonou hviezdičky.
- Claim (tvrdenie) – úryvok z pridaného článku, ktorý poukazuje na ľubovolný fakt, ktorého pravdivosť chceme overiť. Má malý rozsah - zvyčajne sa jedná o jednu vetu. Mala by byť atomická - poukazovať na jeden konkrétny fakt. Užívateľ môže pridať ľubovolné množstvo claimov k ľubovoľnému článku (nemusí byť autorom článku). Obsahom tvrdenia je len samotný text. Claim môžeme zviditeľniť na portáli tým, že mu dáme Upvote. Downvote funguje opačne. Aktuálny stav Upvote môžeme vidieť vedľa ikony Upvote/Downvote.
- Review (hodnotenie) – hodnotenie, ktoré sa vzťahuje ku konkrétnemu claimu. Ku každému claimu môže užívateľ pridať jednu review. Na hodnotenie môžeme reagovať pomocou troch tlačidiel - Súhlasím/Nesúhlasím/Nedostatočné info. V prípade, že sa jedná o Nedostatočné info, nie je potrebné pridať zdroj.
- Stats (štatistika) – obsahuje informácie o aktivite užívateľa, z ktorých backend počíta skóre. Medzi tieto informácie patrí počet pridaných článkov, počet ľudí, ktorým sa článok páčil, počet pozitívnych a negatívnych reakcií na tvrdenia a počet reakcií na hodnotenia (pozitívne, negatívne, nedostatočné info).

Aby sme zabezpečili správnu komunikáciu s backendom a správne zobrazovanie dát na frontende, pre každý komponent systému je dôležité určiť presnú štruktúru. Súbor common.ts obsahuje definície všetkých objektov, pre ktoré v TypeScripte definujeme dátové typy. Príkladom definície z nášho súboru je napríklad Claim:

```
export interface IClaim {
  _id: string;
  article: {
    _id: string;
  };
  text: string;
  createdAt: string;
  nPositiveVotes: number,
  nNegativeVotes: number,
  nReviews: number,
  addedBy: {
    _id: string;
    firstName: string;
    lastName: string;
  };
}
```

5.3 Štruktúra kódu



Obrázok 5.2. Štruktúra aplikácie FactCheck

- cypress – priečinok obsahujúci testy so stromovou štruktúrou podľa knižnice Cypress. Obsahuje hlavne end-to-end testy očíslované podľa poradia v sériovom testovaní. Ďalej sa v ňom nachádza priečinok screenshots, kde môžeme ukladať screenshoty z rôznych stavov počas testovania za účelom hľadania chýb.
- nodemodules – Tento priečinok sa nachádza v každej React aplikácii a obsahuje všetky knižnice, ktoré sú potrebné pre našu aplikáciu. Je automaticky generovaný a o jeho správu sa stará nástroj yarn, ktorý na základe súboru package.json stiahne z npm registra dané knižnice. Yarn je alternatívou pre npm a poskytuje

funkcie pre updatovanie knižníc a príkazom yarn audit dokáže skontrolovať verzie knižníc. Na základe známych bezpečnostných hrozieb dokáže analyzovať, ktoré knižnice je vhodné odstrániť alebo aktualizovať. Vytvára tiež yarn.lock súbor, ktorý definuje graf závislostí.

- public – obsahuje obrázky a ikony. Ikony aplikácie a používateľa sme vložili priamo do neho, pre vkladanie ďalších obrázkov sme vytvorili vnorený priečinok pictures
- src – obsahuje hlavnú logiku aplikácie. Prvým priečinkom je state. State obsahuje súbory definujúce Recoil stavy, ktoré si ukladáme. Api obsahuje všetky volania na backend, ktoré v aplikácii používame. Pre prehľadnosť sme tieto volania rozdelili do servisov podľa toho, s akými zdrojmi manipulujú (napr users service). Common obsahuje všetky objekty a ich dátové typy, ktoré naša aplikácia používa, ako napríklad článok alebo tvrdenie. Components obsahuje znovupoužiteľné komponenty, ktoré sa nachádzajú v rôznych častiach aplikácie, ale zároveň netvoria celú plochu obrazovky. Layouts obsahuje plochy obrazovky, ktoré tvoria logické časti aplikácie, ako napríklad zoznam populárnych článkov. Plochy obrazovky sa skladajú z komponentov v priečinku components. Locales obsahuje preklady premenných do českého, slovenského a anglického jazyka. Tento priečinok používa knižnica i18n, ktorá sa stará o preklad do jazyka, ktorý si zvolil užívateľ kliknutím na vlajku jazyka. App.tsx definuje štruktúru obrazovky - v hornej časti hlavičku, v strede načítaný obsah a v dolnej časti pätu stránky. App.tsx používa knižnicu react-router-dom, ktorá rozdeľuje aplikáciu do logických celkov a po presmerovaní užívateľa na inú plochu obrazovky automaticky načíta nové komponenty. httpcommon.ts je súbor, ktorý definuje pomocou knižnice axios http klienta (url adresu, hlavičky, ..). index.scss definuje štýly, ktoré používajú naše komponenty. index.tsx obsahuje základ single-page-application a vytvára koreňový element App.
- .env – súbor obsahuje procesné premenné a jeho obsah nezverejňujeme na GitHube. Jeho príkladnú verziu môžeme vidieť v súbore .env.example
- .gitignore – obsahuje priečinky a súbory, ktoré používame v projekte, ale z bezpečnostných dôvodov ich nezverejňujeme (napr. .env)
- eslintrc.json – súbor obsahuje `good practices` pre vytváranie kódu s pevnou štruktúrou. Používa ho knižnica eslint, ktorá automaticky detekuje chyby, ktoré by mohli nastať v našom kóde. Spolupracuje s našim vývojovým prostredím Visual Studio Code a vizuálne vyznačuje tieto chyby, pri niektorých dokonca navrhuje možnú opravu. V súbore eslintrc.json môžeme definovať, pre aké typy súborov chceme knižnicu použiť a aké pravidlá chceme (nechceme) použiť.

5.4 Zoznam komponentov a rozloženia obrazovky

- AddReview – Komponent obsahujúci formulár na pridávanie hodnotení
- CreateArticle – Komponent obsahujúci formulár na pridanie článku
- EditArticle – Komponent obsahujúci formulár na upravenie článku
- Article – Komponent obsahujúci článok
- CreateClaim – Komponent obsahujúci formulár na pridanie tvrdenia
- EditClaim – Komponent obsahujúci formulár na upravenie tvrdenia
- Claim – Komponent obsahujúci tvrdenie
- EditProfile – Komponent obsahujúci formulár na upravenie profilu
- MyArticles – Komponent obsahujúci zoznam článkov používateľa

- MyClaims – Komponent obsahujúci zoznam tvrdení používateľa
- MyTitle – Komponent definujúci vlastný štýl nadpisu
- ProfileSideBar – Komponent obsahujúci bočný panel
- Review – Komponent obsahujúci hodnotenie
- Reviews – Komponent obsahujúci zoznam hodnotení
- Scoreboard – Komponent obsahujúci skóre užívateľa a tabuľku najaktívnejších používateľov
- SignIn – Komponent obsahujúci prihlásenie do aplikácie
- SignUp – Komponent obsahujúci registráciu do aplikácie
- Logout – Komponent obsahujúci odhlásenie z aplikácie
- NotificationContext – Komponent obsahujúci kontext pre zobrazovanie notifikácií

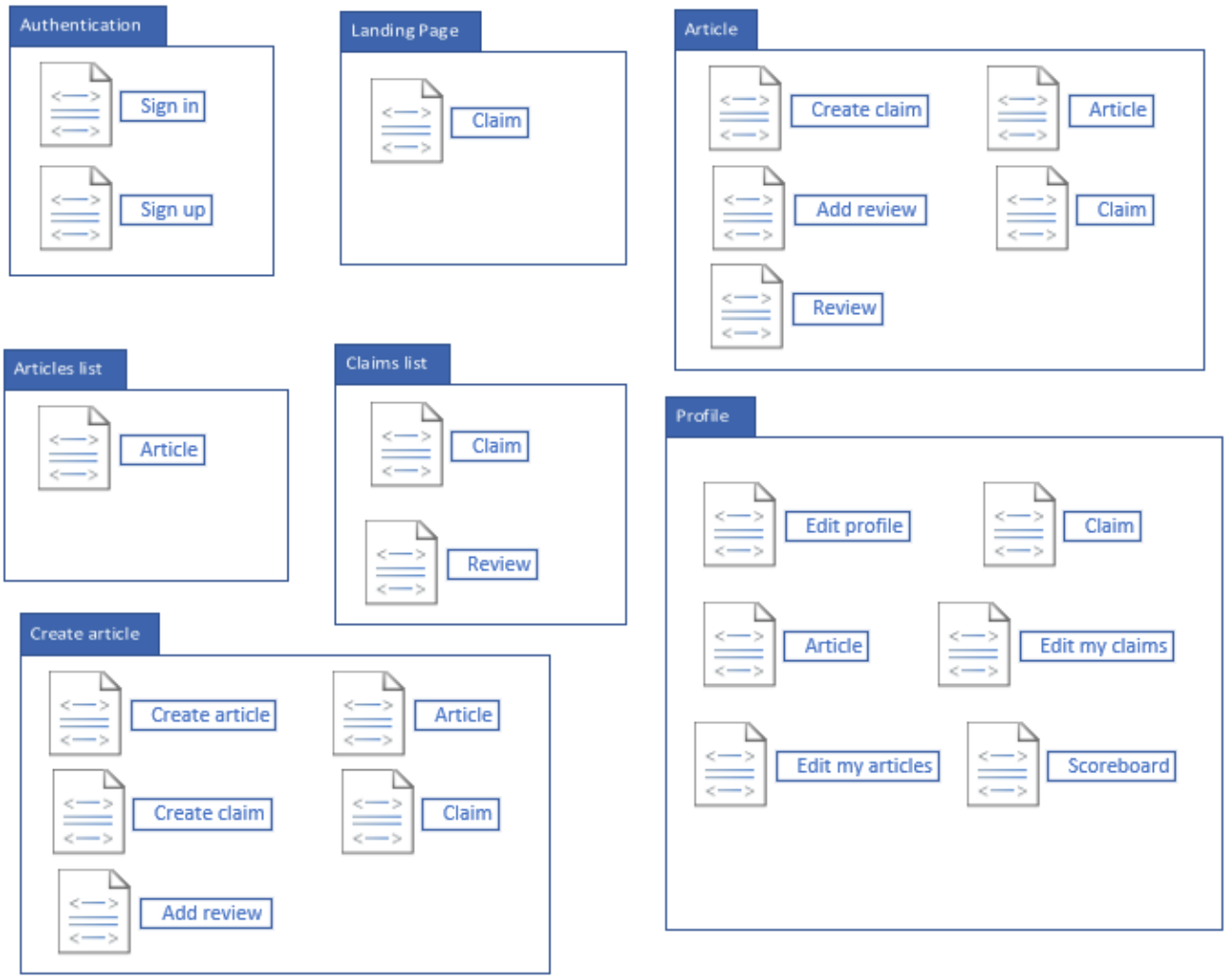
5.5 Service pattern

Na začiatku implementácie v zimnom semestri sme všetky volania na API mali v jednom súbore. Časom sme zistili, že pridávaním nových API volaní sa náš kód stáva neprehľadným. Preto sme sa rozhodli vytvoriť pattern pre API volania, ktorý bude API volania rozdeľovať do logických celkov podľa ich funkcionality. API volania sú implementované ako statické metódy. Všetky tieto funkcie zároveň dedia konfiguráciu axios serveru zo súboru http_common.ts. V tomto súbore definujeme dve služby : FactCheck backend a webscraping službu pre získanie textu článku. Zároveň v ňom definujeme interceptor pre odpovede z backendu. Interceptor pre odpovede slúži na to, že keď počas používania aplikácie uplynie platnosť accessTokenu a dostaneme HTTP odpoveď s kódom 401 (Unauthorized), automaticky zavoláme funkciu refreshToken(), obnovíme accessToken a pôvodný request sa zopakuje. Týmto spôsobom zabezpečíme, že chod aplikácie bude plynulý a užívateľ sa nemusí znova prihlasovať na tom istom zariadení.

```
// Reviews service

import { factCheckBe } from '../http_common';
import { IReview } from '../common/types';

export default class ReviewsService {
  static getReviews = (articleid: string, claimid: string) => {
    return factCheckBe
      .get⟨IReview[]⟩(
      `/articles/${articleid}/claims/${claimid}/reviews`,
    );
  };

  static voteReview = (idReview: string, rating: number) => {
    const token = localStorage.getItem('accessToken');
    const headers = { Authorization: `Bearer ${token}` };
    return factCheckBe
      .post⟨number⟩(
      `/vote?reviewId=${idReview}`,
      { rating },
      { headers },
```

```
    );
  };

  static addreview = (
    articleid: string,
    claimid: string,
    values: IReview,
  ) => {
    const token = localStorage.getItem('accessToken');
    const headers = { Authorization: `Bearer ${token}` };
    return factCheckBe
      .post⟨IReview⟩(
      `/articles/${articleid}/claims/${claimid}/reviews`,
      values,
      { headers },
    );
  };
}
```

5.6 Popis systému

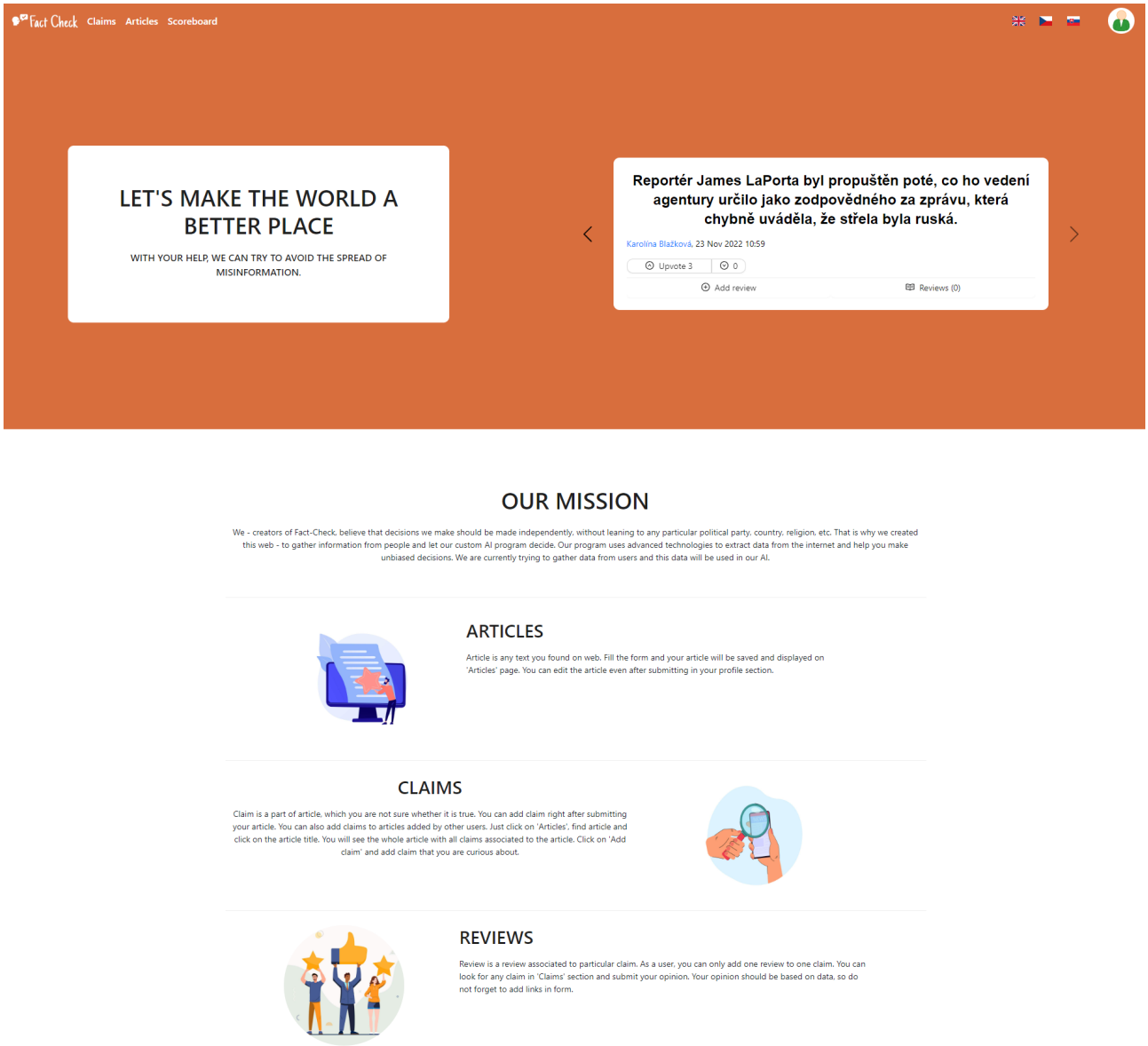


Obrázok 5.3. Rozloženie stránok do komponentov

Aplikácia FactCheck je implementovaná ako single-page application. To znamená, že jej komponenty sa dynamicky načítavajú miesto klasického načítavania celej stránky. Interakcia s aplikáciou teda pôsobí viac prirodzene a zlepšuje užívateľský zážitok. Na vrchu aplikácie je pevne umiestnený navigačný panel, ktorý po kliknutí na logo aplikácie presmeruje užívateľa na hlavnú stránku. Ďalej sa v paneli nachádza link na zoznam obľúbených článkov a obľúbených tvrdení. Na pravej strane je umiestnená ikona používateľa, ktorá slúži ako tzv. dropdown button - tlačidlo ktorým rozbalíme menu. Pomocou neho môže užívateľ navštíviť svoj profil alebo sa odhlásiť. V navigačnom paneli sa nachádzajú 3 vlajky pomocou ktorých si môže užívateľ nastaviť predvolený jazyk v ktorom bude celá aplikácia. Na preklad sme použili JS knižnicu i18n.

5.7 Rozloženia obrazovky

Landing Page je prvá stránka, ktorá sa zobrazí, keď užívateľ navštívi portál Fact-Check. Z užívateľskej ankety, ktorej cieľom bolo vyjadriť názor na prvú verziu Fact-Checku v zimnom semestri, sme zistili, že noví používatelia niekedy nerozumejú, ako portál používať. Preto sme sa rozhodli spraviť jednoduchú pútavú stránku, ktorá obsahuje informácie o tom, čo je cieľom aplikácie a taktiež opisuje jeho základné komponenty (článok, tvrdenie, hodnotenie). Úvodná stránka obsahuje list obľúbených tvrdení. Obľúbené tvrdenia sú tvrdenia s najväčším počtom Upvotes a na úvodnej strane sú zobrazené ako karuzel - slideshow 8 cyklicky opakujúcich tvrdení. Z dokumentácie Antd knižnice o copywritingu sme sa dozvedeli, že prvá stránka by mala pôsobiť na užívateľa jednoducho. Preto sme sa rozhodli vysvetliť fungovanie stránky podľa celého toku informácií - vytvorenie článku, tvrdenia a následne hodnotení. Stránka tiež obsahuje paragraf o tom, aký je jej účel a obsahuje aj nás - autorov aplikácie Fact Check spolu s našimi osobnými fotografiami.



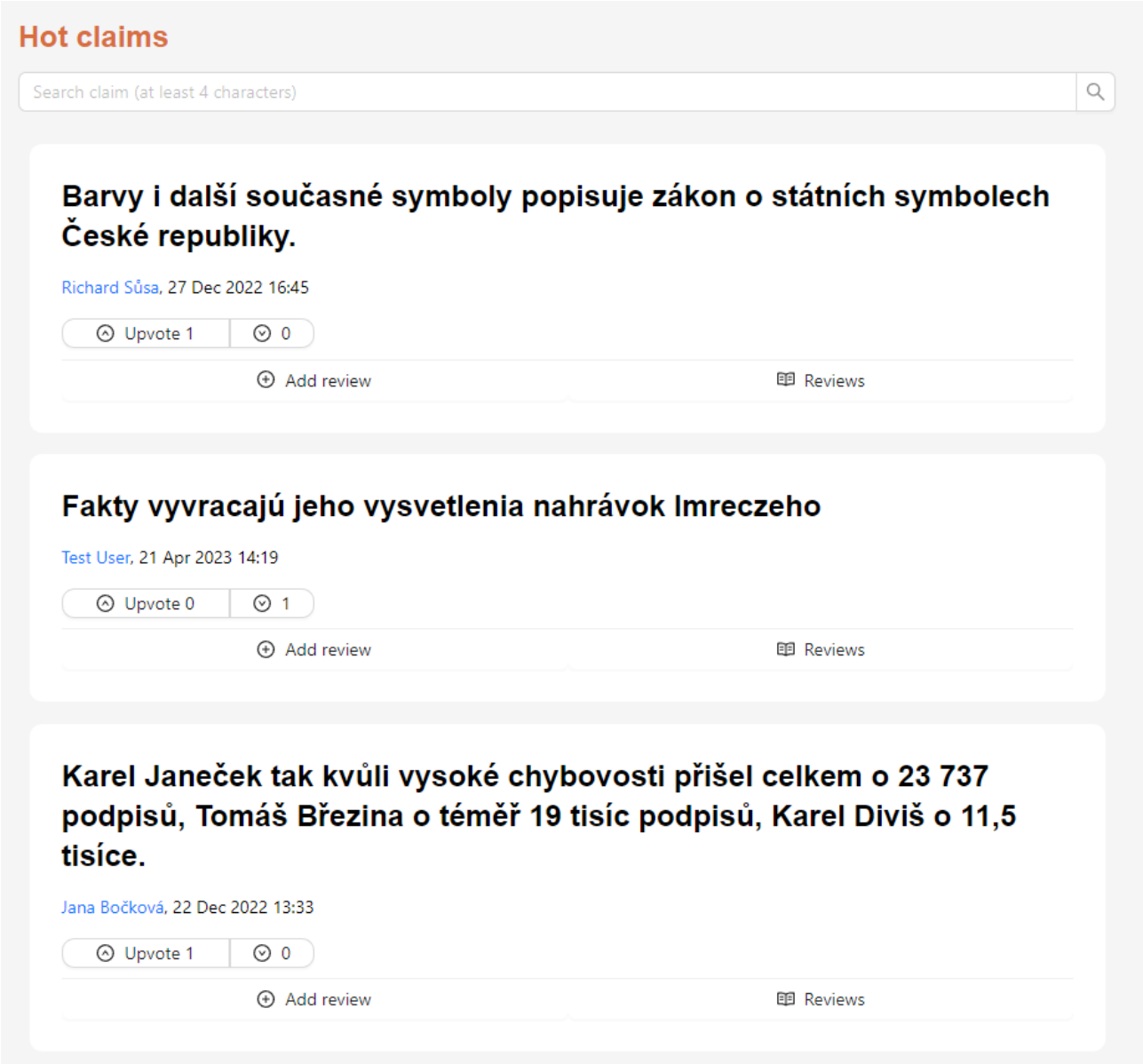
Obrázok 5.4. Úvodná stránka

Authentication je podstránka, ktorej účelom je overenie identity. Pokiaľ nie je užívateľ zaregistrovaný, má možnosť vytvoriť si profil vyplnením krátkeho formulára. Formulár obsahuje základné informácie, ktoré sú dôležité pre identifikáciu - meno, priezvisko, mail a heslo. Po úspešnom vytvorení profilu je užívateľ automaticky prihlásený a môže využiť plnú funkcionalitu portálu Fact-Check. Pri prihlásení sa z backendu vrátia informácie o tzv. session, ktoré ukladáme do localStorage. Session obsahuje accessToken, refreshToken a expireIn. Na základe týchto premenných vieme konštruovať autorizačné headery requestov.

Obrázok 5.5. Prihlásenie do aplikácie

Claims list je podstránka, ktorá obsahuje zoznam tvrdení, zoradených podľa počtu sociálnych interakcií. Každé tvrdenie obsahuje text, autora a dátum pridania. Pre vytvorenie platformy na zdieľanie informácií je potrebné implementovať nejaký spôsob, ktorým vieme rozhodnúť, ktorý obsah je pre užívateľov pútavý. Preto sme vymysleli spôsob, akým môžu užívatelia vyjadriť svoj názor. Na základe pozorovaní populárnych sociálnych médií sme navrhli komponent Claim, ktorý obsahuje štyri tlačidlá - Upvote, Downvote, Add review a Reviews list. Upvote slúži na vyjadrenie názoru, že daný claim je pravdivý a zaujímavý, preto chceme aby sa v priečkach zobrazoval vyššie. Downvote slúži opačne na vyjadrenie nesúhlasu. Tento pattern sme odpozorovali z populárneho portálu Quora, na ktorom užívatelia pridávajú otázky na rôzne témy a môžu ich hodnotiť. Add review je implementovaný ako modal okno, ktoré obsahuje formulár pre vytvorenie nového hodnotenia. Každé

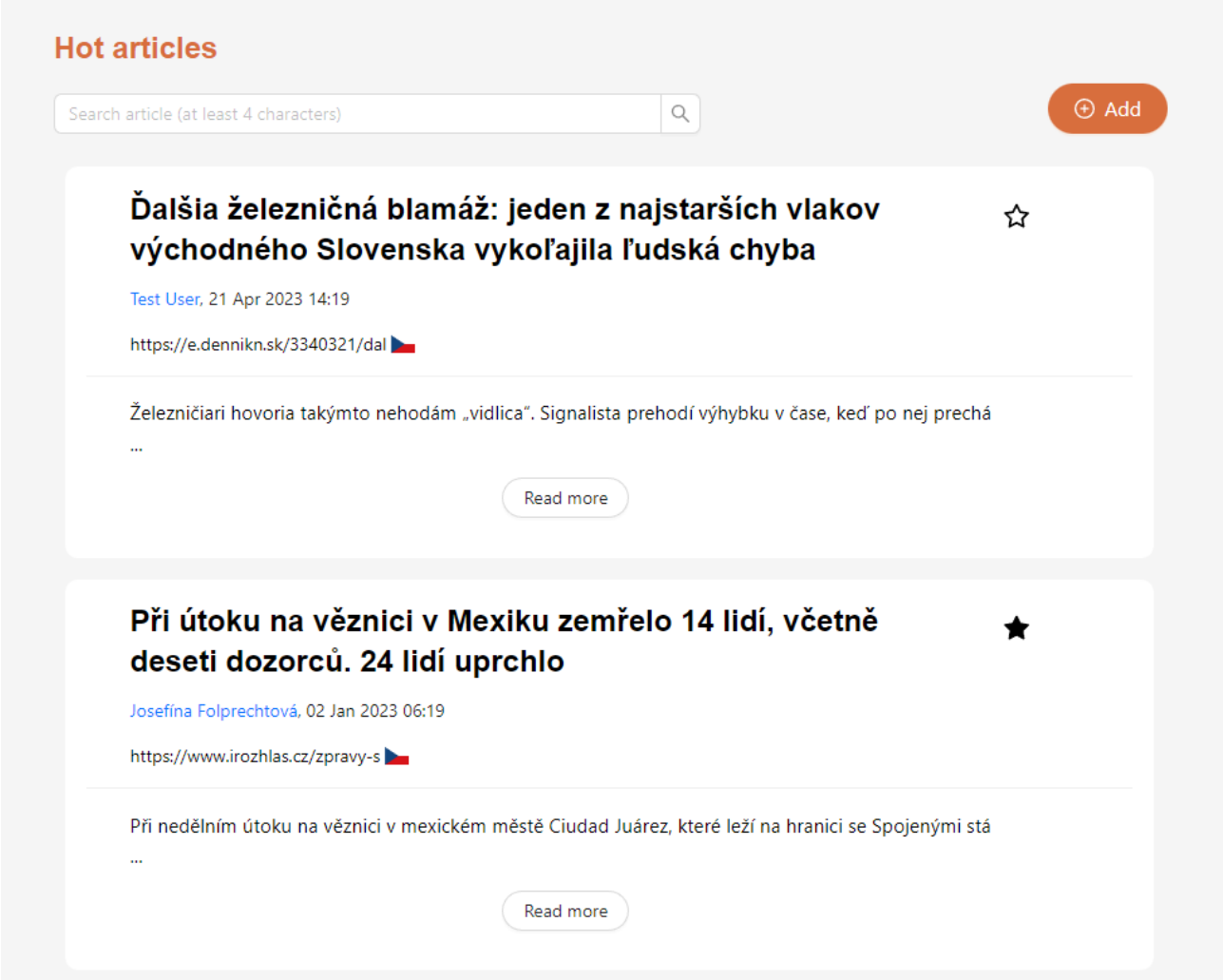
hodnotenie obsahuje text, celkový dojem (pozitívny, negatívny, chýbajú informácie) a URL adresu potvrdzujúcu samotnú pravdivosť hodnotenia. Po kliknutí na text Claimu je užívateľ presmerovaný na stránku, ktorá obsahuje zdroj informácií tvrdenia (článok) . Reviews list je tiež modal okno, v ktorom sa nachádza zoznam hodnotení k danému tvrdeniu. Užívateľ sa môže vďaka hodnoteniam iných dozvedieť, ktoré populárne fakty sú podľa užívateľov pravdivé a ktoré sú naopak klamlivé alebo zavádzajúce.



Obrázok 5.6. Zoznam tvrdení

Articles list je podstránka, ktorá obsahuje zoznam článkov. Článku môže užívateľ udeliť hviezdičku, ktorou vyjadrí, že je preňho zaujímavý. Po prejdení myšou sa zobrazí tooltip - vysvetlenie, čo hviezdička znamená. Každý článok obsahuje nadpis, text, autora, dátum pridania, jazyk (český, slovenský, anglický, iný) a pôvodný link, odkiaľ článok pochádza. Kliknutím na link je užívateľ presmerovaný na pôvodnú stránku, ktorá sa zobrazí v novom okne. Vedľa linku je svg obrá-

zok vlajky, ktorá reprezentuje jazyk, v ktorom je článok napísaný. Po kliknutí na nadpis článku je užívateľ presmerovaný na celé vlákno - článok s tvrdeniami a hodnoteniami užíateľov. Text článku sa nezobrazuje celý, aby vizuálne nezaplnil celú obrazovku neprehľadným textom. Ak si užívateľ chce prečítať celý text, po kliknutí na `Read more` sa zobrazí celý. Po kliknutí na autora článku sa užívateľ presmeruje na jeho profil.

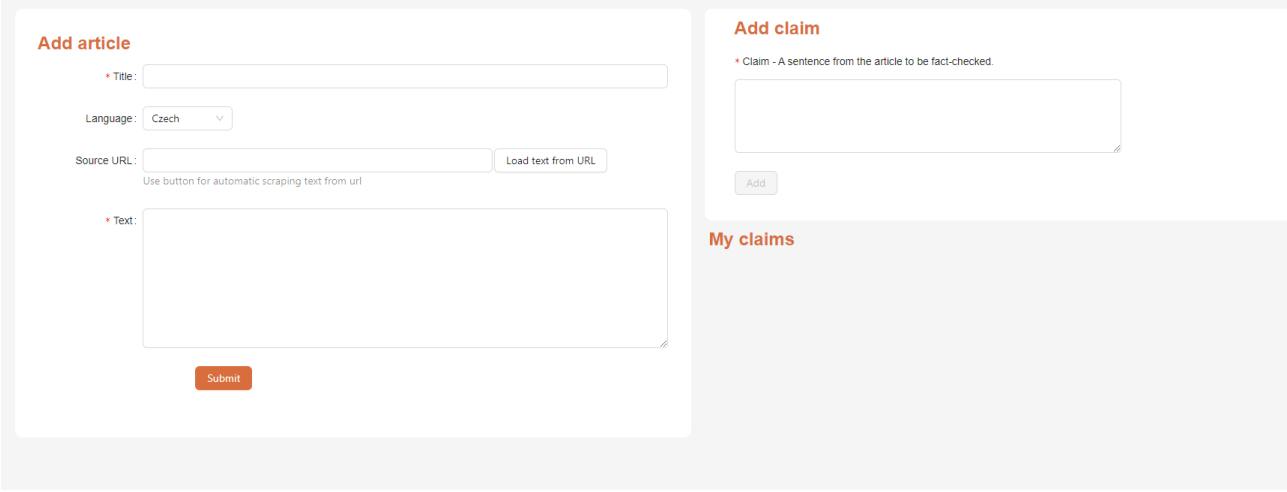


Obrázok 5.7. Zoznam tvrdení

Article je podstránka, ktorá zobrazuje celé vlákno informácie. Prvou časťou je samotný článok, ktorý obsahuje ľubovoľné množstvo tvrdení. Tie sú zobrazené pod ním a poukazujú na fakty, ktoré vyplývajú z článku. Používateľ môže pridať ďalšie tvrdenie, ktoré považuje za dôležité alebo zaujímavé. Používateľ môže pridávať tvrdenia aj k článkom, ktoré nepridal on. Na jeho tvrdenie môžu užívatelia pridať hodnotenie, ktoré slúži ako overenie faktu.

Obrázok 5.8. Tok informácie

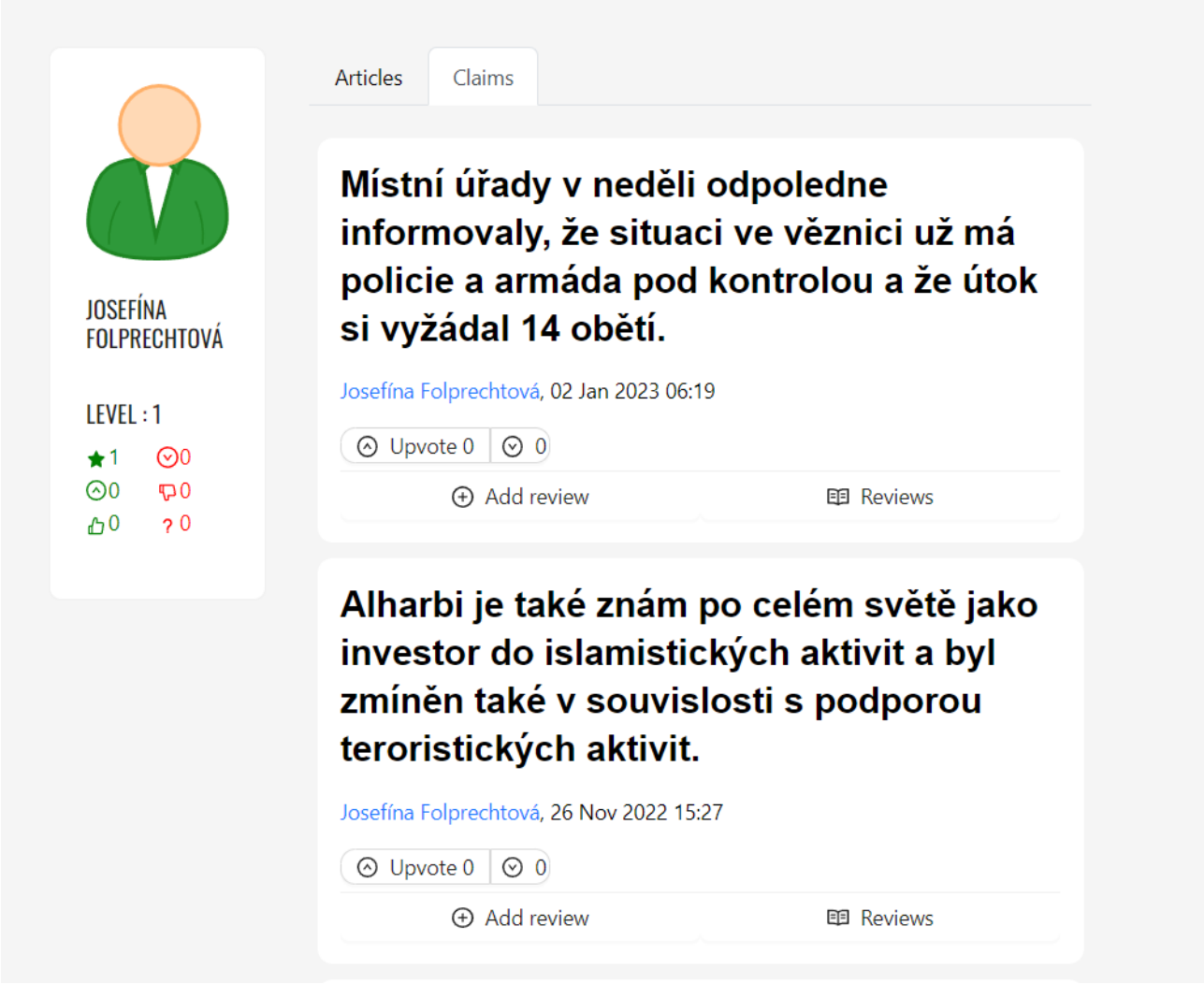
Create article je podstránka slúžiaca na pridávanie nových zdrojov informácií. Scraping API zjednodušuje spôsob pridávania článkov. V prvej verzii FactCheck musel užívateľ všetky info manuálne zadávať. Pridanie je zložené zo 4 inputov - nadpisu, url adresy, textu a jazyka. V prvom riadku užívateľ vyplní URL adresu článku a po kliknutí tlačidla `Load text from article` sa stiahne titulok a samotný text článku. Ak by stiahnutý text nevyhovoval užívateľovi alebo chcel pridať len jeho časť, po stiahnutí ho môže stále editovať. Po úspešnom pridaní článku je odblokované tlačidlo `Add claim`. Užívateľ tak môže pridávať tvrdenia ihneď po pridaní samotného článku.



Obrázok 5.9. Vytvorenie článku a následne claimu

Profile je podstránka, ktorá zobrazuje profil užívateľa. Na profile vidíme jeho meno, štatistiky úspešnosti, zoznam jeho článkov a tvrdení. Štatistiky zahrňujú:

- Level – celkové skóre dosiahnuté na základe interakcií
- Obľúbené články – počet `hviezdičiek`, ktoré dostali jeho články
- Upvotes – počet Upvotes získaných z tvrdení
- Likes – počet `Súhlasím` na hodnoteniach
- Downvotes – počet Downvotes získaných z tvrdení
- Dislikes – počet `Nesúhlasím` na hodnoteniach
- Miss info – počet `Chýbajúce info` na hodnoteniach

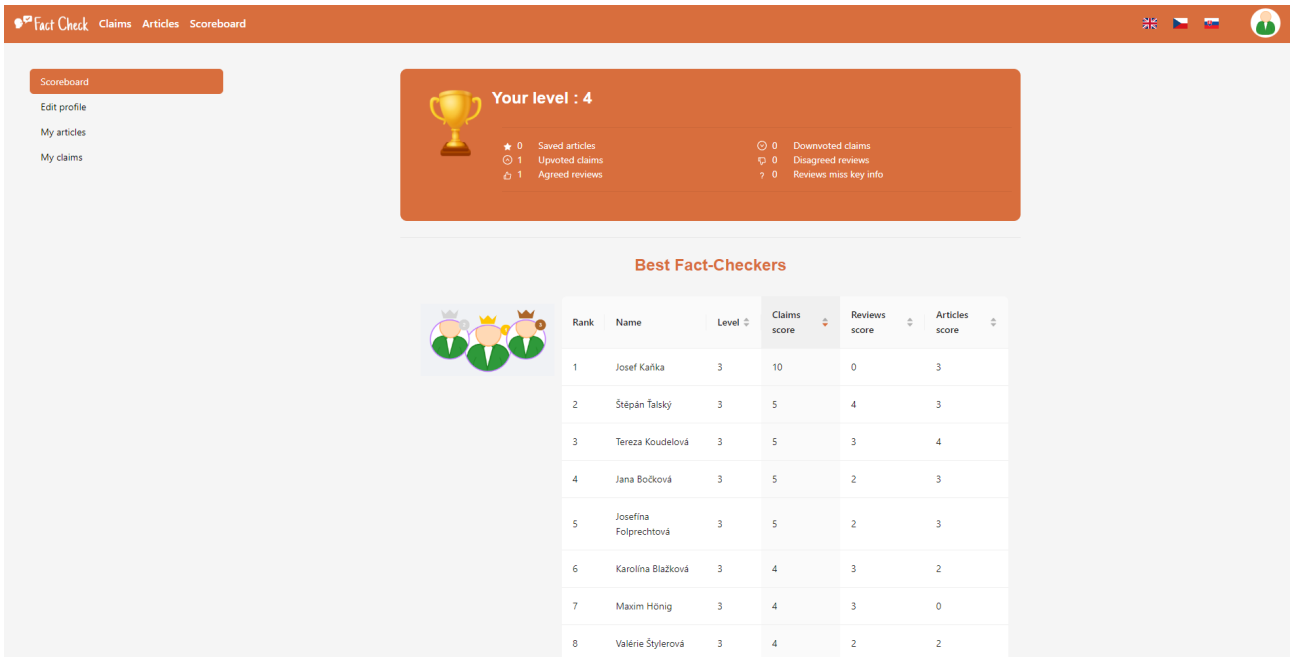


Obrázok 5.10. Profil užívateľa

Na profile iných užívateľov môžeme vidieť, aké fakty daný človek pridáva a hodnotiť ich. Podľa štúdie Pinta a kol. si takto môže užívateľ urobiť lepší obraz o autorovi článkov.

My profile je podstránka, ktorá umožňuje užívateľovi upravovať jeho dáta. V prvej sekcii sú zobrazené kompletné štatistiky nášho profilu (rovnaké ako na profile iných užívateľov, vyššie popísané) a Leaderboard - tabuľka najaktívnejších užívateľov. V druhej sekcii môže upraviť svoje osobné dáta - meno, priezvisko a mail.

Ďalšia sekcia obsahuje list `My Articles`, kde môžeme vidieť všetky články, ktoré sme pridali. V modálnom okne je možné každý z článov upravovať. Posledná časť `Profile` sekcie je list `My Claims`, ktorá sa skladá zo všetkých tvrdení užívateľa k všetkým článkom. V modálnom okne je možné editovať text každého tvrdenia.



Obrázok 5.11. Môj profil

5.8 Optimalizácia počtu volaní

V prvej verzii aplikácie FactCheck sme si na serveri všimli, že aj keď sa len jeden používateľ prihlási do aplikácie, na server odosielame zbytočne veľké množstvo requestov. Dôvodom je to, že pri každom pridaní nového príspevku (článku,tvrdenia) alebo jeho úprave robíme opakovaný request na zistenie aktuálneho stavu zoznamu. Napríklad keď používateľ číta článok, pridá tvrdenie a následne ide na svoj profil aby si pozrel svoje tvrdenia, znova sme prevolávali API na vrátenie tvrdení konkrétneho používateľa. Z toho dôvodu sme usúdili, že niektoré dáta je vhodné si ukladať. Použitím state managementu sme tak optimalizovali počty volaní na niektoré URL.

```
// optimized version of create claim
claimsService.createClaim(article._id, values).then((res: any) => {
  const mergedClaims = [...claims];
  res.key = res.data._id;
  mergedClaims.push(res.data);
  claimForm.resetFields(['text']);

  // set claims of article
  setClaims(mergedClaims);

  //set user's claims
  const mergedMyClaims = [...myClaimsList, newClaim];
  setMyClaimsList(mergedMyClaims);
```

```
    closeModal();
    notification.info({
      message: t('successfully_added_claim'),
      description: t('gained_20'),
    });
  }).catch((e) => message.error(e));
```

5.9 Gamifikácia a body za aktivitu

Pri tvorbe gamifikácie sme sa inšpirovali návrhom podobnej platformy CALYPSO, ktorej autor uvádza mnoho príkladov, že gamifikácia môže mať pozitívny vplyv na zapojenie užívateľov do systému [27]. Aby sme teda užívateľov motivovali, vytvorili sme návrh gamifikácie, vďaka ktorému za aktivitu na našom portáli môžu získavať body. Body môžu získať za vytvorenie článku, tvrdenia, hodnotenia, umiestnenie v tabuľke najaktívnejších užívateľov a hlasovanie. Pri vytvorení nového príspevku im zobrazíme notifikáciu, koľko bodov a za čo získali.

Používatelia zatiaľ môžu hlasovať a vidieť, koľko hlasov dostali jednotlivé príspevky. Získané body vidia na svojom profile a na základe počtu dosiahnutých bodov zobrazujeme ich level. Počítanie levelu zabezpečuje backend a požíva na to funkciu **level =** $\lfloor 0.25 * \sqrt{experience} \rfloor$.

Podobne ako v prípade CALYPSO platformy, užívateľov je potrebné hodnotiť nielen na základe množstva pridaného obsahu, ale aj ich dôveryhodnosti [27]. Našim cieľom je, aby používatelia získavali `kredibilitu` za to, že im niekto dal súhlas/nesúhlas s ich tvrdením a hodnotením. Kredibilita bude ovplyvňovať level daného používateľa. Táto funkcionalita bohužiaľ zatiaľ nie je implementovaná na backende, ktorý má na starosti kolega Bc. Rastislav Kopál a bude implementovaná v ďalšej verzii FactChecku.

5.9.1 Využitie gamifikácie v overovaní faktov

Naša platforma funguje ako decentralizovaný systém, v ktorom môže overovať fakty každý. V takomto systéme je však potrebné vedieť odlíšiť, ktorý obsah možno považovať za populárny a predovšetkým správny. Preto sme navrhli model zobrazovania užívateľského obsahu nasledovne:

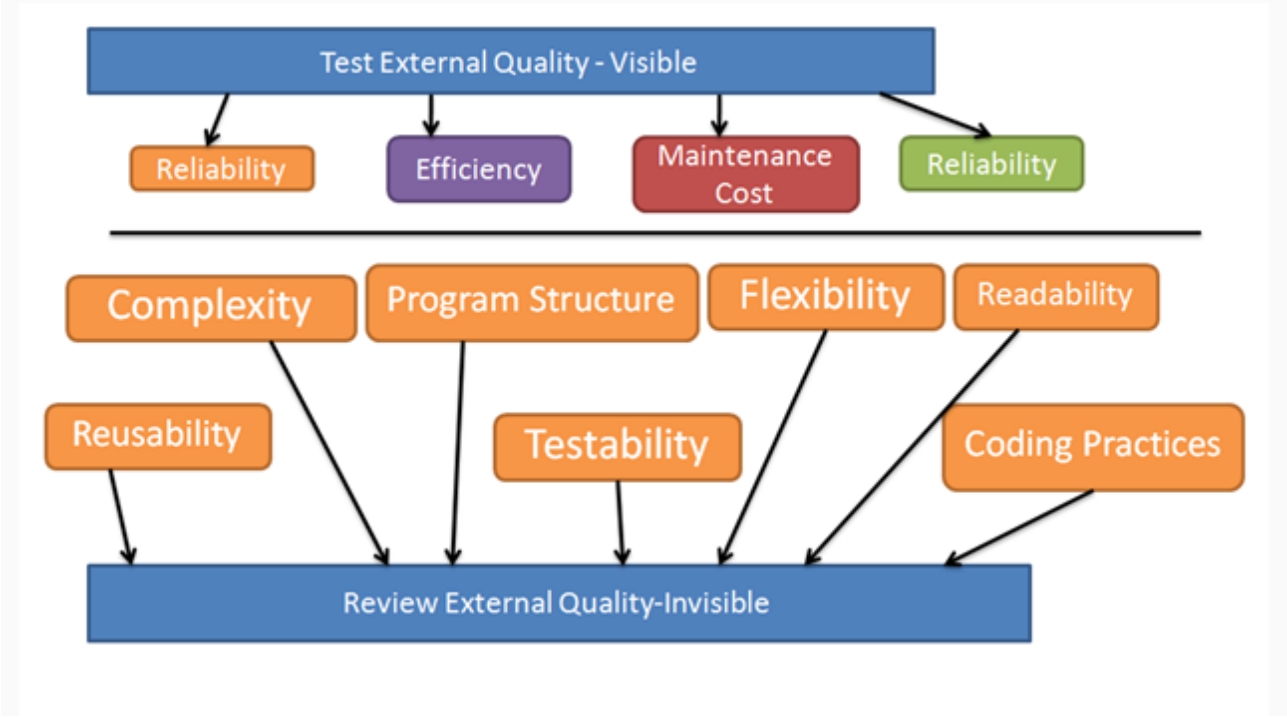
- Hot claims – tvrdeniam užívatelia udeľujú Upvotes/Downvotes na základe toho, ako považujú dané tvrdenie za zaujímavé a dôležité. Na základe počtu získaných Upvotes zoraďujeme tvrdenia v tejto sekcii - zviditeľňujeme tvrdenia, ktoré sú medzi užívateľmi populárne
- Reviews – overovanie faktov funguje na základe pridávania hodnotení, ktoré obsahujú zdroj URL adresu. Užívatelia môžu vyjadriť ku každému hodnoteniu názor: Súhlasím/Nesúhlasím/Chýba info. Na základe počtu reakcií zoraďujeme hodnotenia k jednotlivým tvrdeniam. Používame na to sortovaciu funkciu na frontende, ktorá udelí každému hodnoteniu skóre vo formáte **(1+0.1xlevel) x (súhlasí - nesúhlasí - 0.1x chýba info)** . Takto zobrazujeme do popredia overenia tvrdení, s ktorými najviac ľudí súhlasí. Podobný štýl využíva populárna platforma StackOverflow, na ktorej sa užívatelia môžu pýtať rôzne technické otázky. Odpovede ľudí, ktoré majú vysoký počet pozitívnych interakcií zobrazuje na popredných priečkach. Tento systém považujeme za veľmi efektívny, o čom svedčí

aj obľúbenosť stránky StackOverflow. Náš systém umožní každému, aby sa podieľal na overovaní faktov a ostatným používateľom dáme „moc“ ovplyvniť, na akých priečkach sa bude jeho obsah zobrazovať pri overovaní faktov. Zároveň dosiahneme to, že dôveryhodní užívatelia budú prioritne zobrazovaní pri overovaní faktov. Ich level bude hrať tiež rolu - užívateľom s vysokým levelom toto skóre ešte znásobíme.

```
//points for actions
const userActions = {
  createArticle: 8,
  createClaim: 20,
  createReview: 30,
  appearAtLeaderboard: 10,
  vote: 3,
};
```

Kapitola 6
Testovanie a zaistenie kvality

Keď hovoríme o kvalite softvéru, hovoríme vlastne o hodnotení softvéru na základe určitých atribútov. Kvalita softvéru sa definuje na základe skúmania vonkajších a vnútorných vlastností softvéru. Vonkajšia kvalita je definovaná na základe toho, ako softvér funguje v scenári v reálnom čase v prevádzkovom režime a ako je užitočný pre svojich používateľov. Vnútorná kvalita sa na druhej strane zameriava na vnútorné aspekty, ktoré závisia od kvality napísaného kódu. Používateľ sa viac zameriava na to, ako softvér funguje na vonkajšej úrovni.



Obrázok 6.1. Atribúty kvality softwaru. [28]

Pri tvorbe testov je vhodné použiť osvedčené patterny, vďaka ktorým sa predchádza vzniku chýb počas fáze vývoja softvéru [29].

- Písať testovateľný kód – písanie testovateľného kódu zahŕňa dodržiavanie zásad, ako je napríklad oddelenie závislostí komponentov v systéme
- Testovať včas a často – integrácia testovania do pracovného postupu vývoja a pravidelné spúšťanie testov pomáha rýchlo identifikovať problémy a zabezpečuje, že kód zostane stabilný a spoľahlivý počas celého procesu vývoja.
- Používať popisné názvy testov – jednotné konvencie v štruktúre a v názvoch testov uľahčujú pochopenie rozsahu a zámeru testu.
- Testovať okrajové prípady – okrem testovania očakávaného správania je nevyhnutné otestovať aj okrajové prípady a potenciálne zlyhania, ktoré môžu nastať v málo pravdepodobných situáciách

- Automatizovať testovanie – na automatizáciu procesu testovania a spúšťanie testov pri každej úprave kódu používame nástroje na kontinuálnu integráciu (CI), ako napríklad Travis CI alebo Jenkins

Zabezpečenie toho, aby softvérová aplikácia fungovala správne a neobsahovala chyby, si vyžaduje dôkladné testovanie. Testovanie softvéru je kľúčovým aspektom životného cyklu vývoja softvéru a zvyčajne sa rozdeľuje na tri hlavné typy: unit testovanie, integračné testovanie a end-to-end testovanie. Nasledujúca kapitola sa bude zaoberať účelom, výhodami a technikami týchto troch úrovní testovania.

6.0.1 Unit testovanie - základ kvality softvéru

Unit testovanie je proces izolovaného testovania jednotlivých komponentov alebo jednotiek kódu, ako sú funkcie alebo metódy. Hlavným cieľom testovania jednotiek je overiť, či sa jednotlivé komponenty správajú správne a či spĺňajú svoje požiadavky [30].

Unit testy zvyčajne píšu vývojári a sú automatizované, čo znamená, že sa môžu vykonávať rýchlo a často. Dobre navrhnutá sada unit testov pomáha identifikovať chyby už v ranom štádiu vývoja, čo uľahčuje ich opravu a znižuje náklady na ňu. Okrem toho jednotkové testy slúžia ako dokumentácia očakávaného správania kódu a môžu pomôcť ostatným vývojárom pochopiť jeho funkčnosť.

6.0.2 Integračné testovanie

Integračné testovanie sa zameriava na overovanie interakcií medzi viacerými komponentmi alebo jednotkami aplikácie. Cieľom tohto typu testovania je zabezpečiť, aby komponenty bez problémov spolupracovali a aby medzi nimi správne prúdili údaje.

Integračné testy sa často vykonávajú po unit testoch, pretože vyžadujú, aby jednotlivé komponenty už boli otestované izolovane. Integračné testovanie sa môže vykonávať na rôznych úrovniach granularity, od testovania interakcie medzi dvoma komponentmi až po testovanie komunikácie medzi rôznymi subsystémami alebo mikroslužbami.

Integračné testovanie pomáha odhaliť problémy súvisiace s rozhraniami komponentov, výmenou údajov a komunikačnými protokolmi. Identifikovaním týchto problémov pred nasadením systému môžu vývojári predísť potenciálnym zlyhaniam a zvýšiť celkovú spoľahlivosť aplikácie.

6.0.3 End-to-end testovanie - kontrola celého toku aplikácie

End-to-end testovanie je komplexný prístup k testovaniu, ktorý hodnotí celú aplikáciu z pohľadu používateľa. Cieľom tohto typu testovania je zabezpečiť, aby všetky komponenty, subsystémy a externé závislosti fungovali spoločne a poskytovali očakávaný používateľský zážitok [31].

End-to-end testy simulujú reálne používateľské scenáre a overujú, či sa aplikácia správa podľa očakávaní na rôznych platformách, v rôznych zariadeniach a v rôznych sieťových podmienkach. Tento typ testovania je kľúčový pre identifikáciu problémov, ktoré sa nemusia prejaviť počas testovania jednotky alebo integračného testovania, ako sú napríklad problémy so službami tretích strán, databázovými pripojeniami alebo prvkami používateľského rozhrania.

Vzhľadom na svoj rozsiahly rozsah môže byť end-to-end testovanie časovo a zdrojovo náročnejšie ako iné typy testovania. V dôsledku toho sme zvolili

moderný prístup testovania frontend aplikácií - testovanie priamo v browseri. Jej fungovanie zhrnieme v ďalšej kapitole.

6.0.4 Zhrnutie prístupu

Unit testovanie, integračné testovanie a testovanie end-to-end sú základnými zložkami spoľahlivej stratégie testovania softvéru. Pochopením ich rozdielnych úloh a prínosov môžu vývojári vytvárať spoľahlivé softvérové aplikácie, ktoré sa agilne testujú už v procese vývoja aplikácie. Uplatňovanie komplexného prístupu k testovaniu, ktorý zahŕňa tieto tri úrovne testovania, pomáha minimalizovať chyby, znížiť náklady na vývoj a zabezpečiť pozitívnu skúsenosť používateľov.

6.1 Testovanie frontendovej aplikácie

Pre JavaScript je k dispozícii množstvo testovacích frameworkov, pričom každý z nich má svoje jedinečné funkcie a výhody. Medzi najobľúbenejšie frameworky patria Cypress, Jest, Mocha a Jasmine. My sme si zvolili použiť na frontende nástroj Cypress, pretože sme ho počas štúdia používali na predmete ZKS a teda máme s ním skúsenosti.

6.1.1 Cypress

Cypress je open-source framework pre end-to-end testovanie napísaný v JavaScripte. Je navrhnutý tak, aby bolo testovanie webových aplikácií jednoduché a celý proces simuloval reálny scenár použitia. Cypress beží priamo v prehliadači, čo umožňuje autentiticky testovať webové aplikácie - rovnakým spôsobom ako používateľ u seba. Vďaka tomu môže Cypress vykonávať interaktívne testovanie v reálnom čase a vývojárom poskytuje prehľadnú analýzu výsledkov testov.

Cypress ponúka niekoľko funkcií, ktoré ho odlišujú od iných nástrojov na testovanie webových aplikácií. Cypressu dokáže vytvárať screenshoty v rôznych fázach procesu testovania, čo umožňuje vracať sa v priebehu vykonávania testu dopredu a späť, identifikovať problémy a tak pochopiť sled udalostí, ktoré k nim vedú.

6.1.2 Výhody používania aplikácie Cypress

Pri zmene kódu aplikácie alebo testov Cypress automaticky načíta a znovu spustí príslušné testy. Opakované načítanie v reálnom čase poskytuje okamžitú spätnú väzbu o vplyve zmien.

Inteligentné čakanie na sprístupnenie prvkov znižuje potrebu manuálneho čakania a časových limitov v testovacom kóde. Automatické čakanie tiež zvyšuje spoľahlivosť a udržiavateľnosť balíka testov.

Cypress podporuje viacero prehliadačov vrátane prehliadačov Chrome, Firefox a Microsoft Edge. Kompatibilita s viacerými prehliadačmi umožňuje testovať aplikácie na rôznych platformách a v rôznych prostrediach.

Je vhodným nástrojom na end-to-end testovanie, pretože sériou akcií v prehliadači vie overiť, či všetky komponenty a subsystémy spoločne fungujú podľa očakávaní [32] . Séria akcií predstavuje flow, ktorý si vieme predstaviť ako graf a tým pádom prejsť všetky jeho vetvy.

Kapitola 7
Tok systému a jeho testovanie

Naša aplikácia sa skladá z troch celkov : backend FactCheck, backend webscraping a frontend. Backendovú časť má na starosti Bc. Rastislav Kopál a jej fungovanie testujeme pomocou automatizovaných unit testov. Z fungovania aplikácie sme usúdili že najlepšou metódou pre testovanie frontendu bude napísanie end-to-end testov, ktoré zaistia nielen správnosť informácií, ale aj celý tok aplikácie. V nasledujúvej kapitole rozoberieme testovanie do hĺbky a navrhneme spôsob na testovanie.

Reprezentácia systému pomocou grafu toku nám pomôže lepšie pochopiť, ako sa používateľ môže pohybovať v našom systéme. Vďaka grafovej reprezentácii môžeme navrhnúť vhodnú štruktúru testov, ktoré prejdú od počiatočného stavu načítania aplikácie až po jeho koniec. Keďže by bolo časovo náročné (niekedy až nemožné) testovať každú kombináciu používateľských akcií, zvolíme toky, ktoré najčastejšie môžu nastať pri bežnom používaní aplikácie. V takomto toku sa pokúsime otestovať nielen bežné scenáre, ale aj hraničné hodnoty (stavy), ktoré môžu nastať.

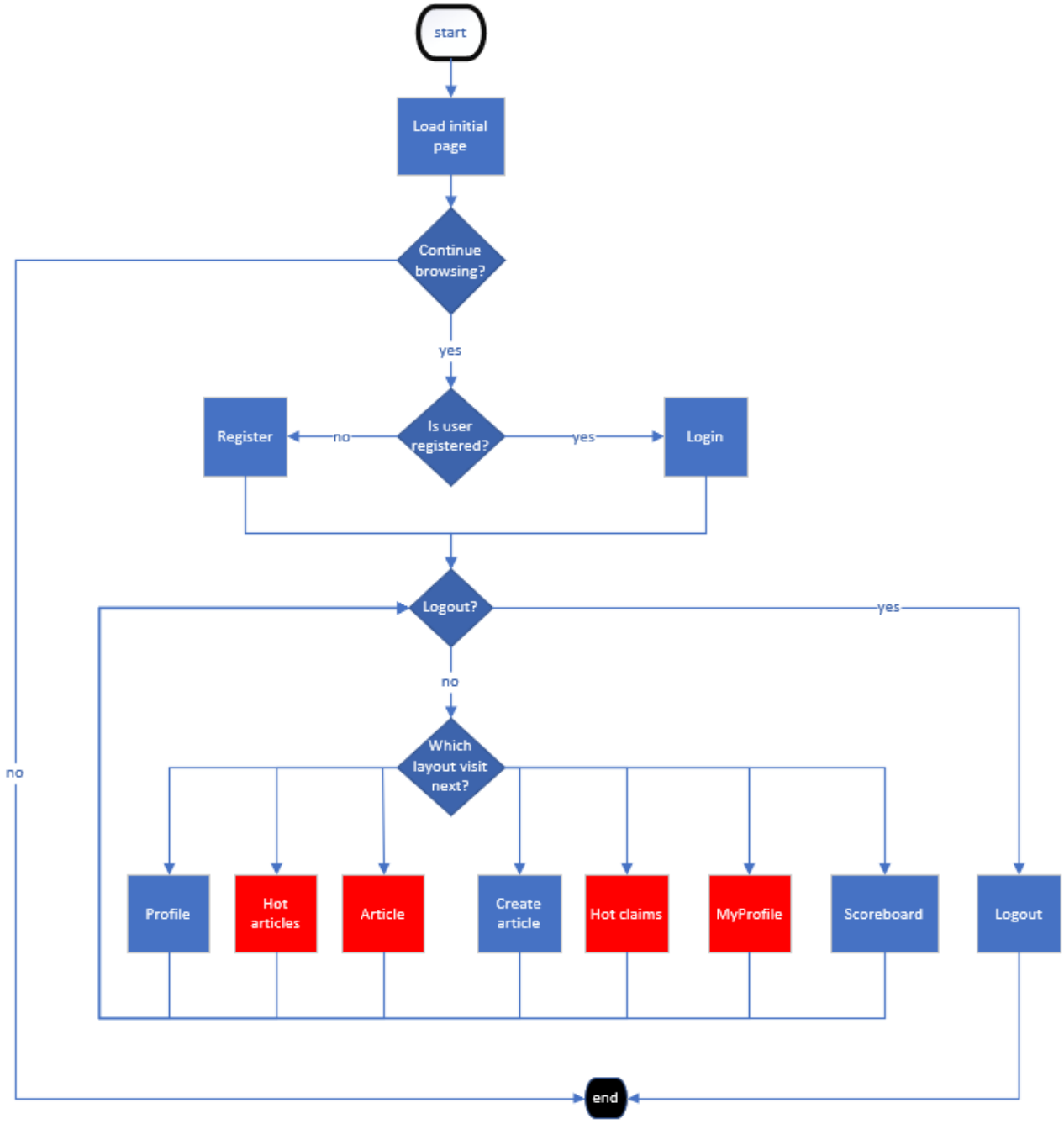
V nasledujúcom obrázku môžeme vidieť základnú grafovú reprezentáciu našej aplikácie. Štart reprezentuje biela gulička, koniec čierna. Štvorcom reprezentujú akcie (čo používateľ vidí alebo spraví) a kosoštvorce reprezentujú rozhodnutia, ktoré môže v danej situácii spraviť. Z dôvodu prehľadnosti nie je možné zobraziť všetky procesy v systéme, preto sme komplexné procesy obsahujúce rozhodnutia označili červenou farbou. Podprocesy obsahujú ďalšie rozhodnutia a akcie, ktoré môže užívateľ urobiť. Komplexné procesy sme ďalej rozbili na atomické procesy a ich grafy toku vložili do prílohy diplomovej práce.

Z grafu sme určili kritické cesty. Testy sme rozdelili do modulov podľa logických celkov. Logický celok je v našom prípade jeden layout, na ktorom používateľ interaguje so systémom. V písaní testov sme sa snažili uvažovať z pohľadu užívateľa a testovať aj okrajové situácie, ako napríklad kliknutie na Upvote viackrát po sebe. Vďaka tomu sme prišli na rôzne chyby a riešenia. Napríklad pri spomínanom Upvote/Downvote sme vytvorili funkciu, korá drží aktuálny stav kliknutia a po opakovanom kliknutí na rovnakú reakciu neodošle request na API. Tento pattern sme tiež použili pri hodnoteniach a uložení článkov.

Testovanie funguje tak, že najprv lokálne spustíme backend aj frontend aplikácie. V konzole otvoríme pomocou príkazu „npx cypress open –browser electron“ prehliadač a spustíme naše testy. End-to-end testy spúšťame iba lokálne, a sú znova spustené pri každej lokálnej zmene kódu.

V nasledujúcom zozname popisujeme naše end-to-end testy

- Register – prvý základný test, ktorý overuje registráciu užívateľa. V prvom kroku navštívi hlavnú stránku a následne klikne na tlačidlo registrácie. Vyplní formulár a odošle ho. Následne kontrolujeme, či bola požiadavka na registráciu vyplnená správnami hodnotami a či sa vrátila správna odpoveď z backendu



Obrázok 7.1. Tok systému

(vrátane prístupových tokenov) . Posledným krokom je overenie zaregistrovaného užívateľského rozhrania.

- Login – druhý test, ktorým overujeme prihlásenie používateľa. Najprv navštívime hlavnú stránku, klikneme na tlačidlo prihlásenia, vyplníme formulár a odošleme požiadavku na server. Overujeme vrátenie prístupových tokenov a načítanie prihláseného UI.
- Create article – podľa názvu testu, tento test má na starosti kontrolu pridávania článku. Formulár môžeme vyplniť dvoma spôsobmi : ručne alebo načítaním info z webscraping service. Testujeme oba spôsoby.

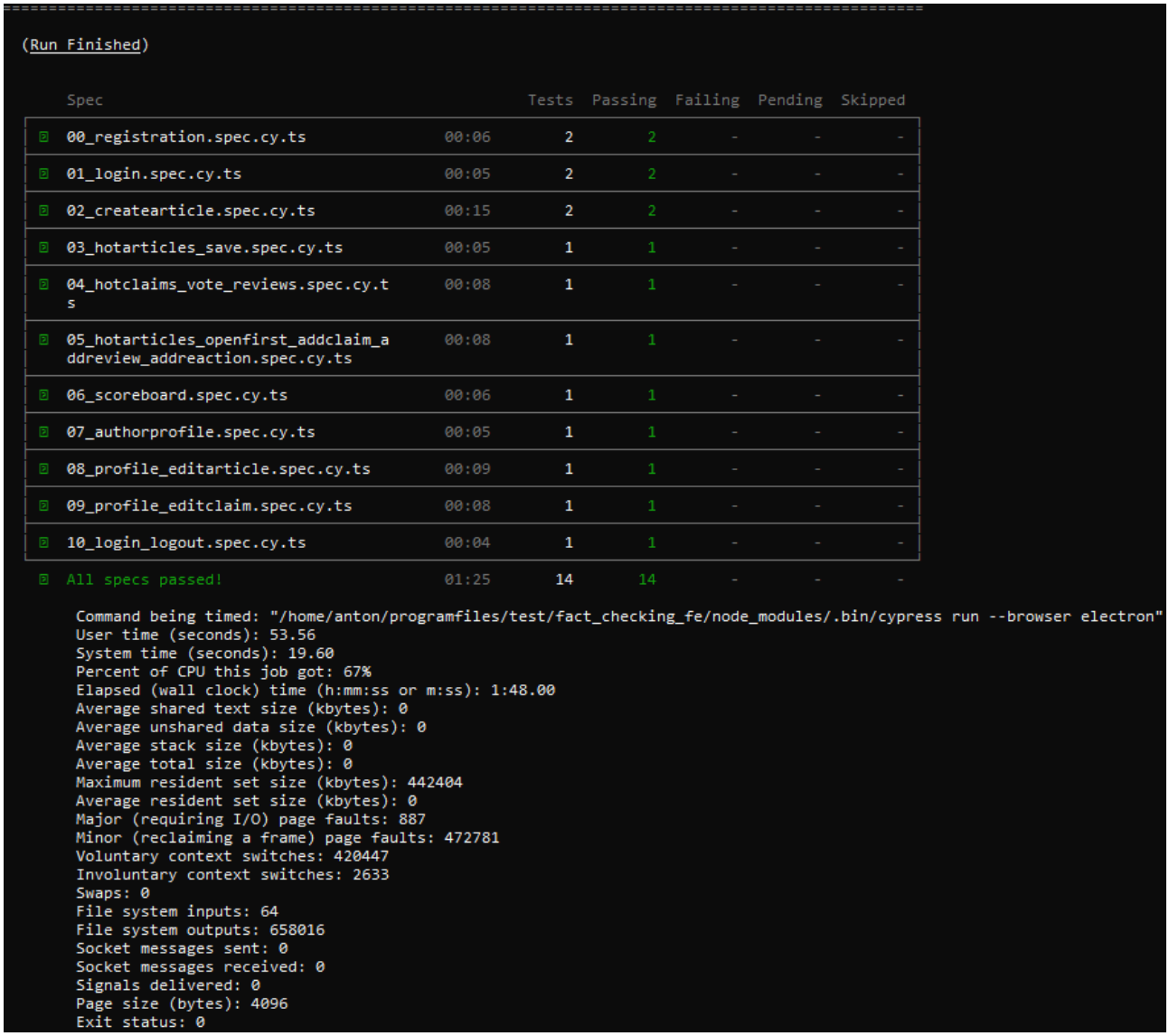
- Hot articles + save – najprv sa prihlásime do aplikácie a následne otvoríme podstránku `Hot articles`. Prvý z nich sa pokúsime uložiť. Ak je tento článok uložený, tak ho naopak vymažeme z uložených. Overujeme teda obe možnosti a správnu odpoveď z backendu.
- Hot claims + vote – prihlásime sa do systému, otvoríme podstránku `Hot claims` a pokúsime sa prvý claim dvakrát Upvotovať a následne dvakrát Downvotovať. Overujeme, že pri Upvote aj Downvote odišla len jedna požiadavka na server (pri opätovnom kliknutí na to isté tlačidlo nechceme dvakrát posielať request) a zároveň overujeme správnosť odpovede z backendu (na základe HTTP kódu)
- Hot articles, open first, add claim, add review - podľa názvu článku, testujeme načítanie článkov a otvorenie prvého z nich na samostatnej podstránke. Následne sa pokúsime pridať tvrdenie a hodnotenie.
- Scoreboard – tento test overuje správne načítanie tabuľky nášho skóre
- Profile – testujeme otvorenie profilu a načítanie našich článkov a tvrdení
- Edit article – po prihlásení otvoríme náš profil a pokúsime sa editovať názov a zdroj článku, overujeme správnosť odoslanej požiadavky a odpoveď z backendu
- Edit claim – po prihlásení otvoríme náš profil a pokúsime sa editovať text tvrdenia, overujeme správnosť odoslanej požiadavky a odpoveď z backendu
- Logout – overujeme prihlásenie a následné odhlásenie zo systému. Skontrolujeme, že accessToken bol z nášho localStorage vymazaný

7.0.1 Štatistiky automatického testovania

V nasledujúcom obrázku môžeme vidieť štatistiky automatického testovania, ktoré sme spúšťali vo WSL. Použili sme na to príkaz

```
/usr/bin/time -v \$(npm bin)/cypress run --browser electron
```

Je potrebné podotknúť, že backend aplikácie bol spustený na pozadí a takisto používame Windows WSL. Tieto skutočnosti môžu mať za následok ovplyvnenie niektorých parametrov, ako napríklad počet page faults. Celkovo testový čas od spustenia Cypressu až po skončenie posledného testu bol 1 minúta 48 sekúnd. Všetky end-to-end testy úspešne prešli. Najväčšie množstvo využitej pamäte bolo menej ako 0.5 GB, z toho usudzujeme že aj „najlacnejší“ ponúkaný server s veľkosťou pamäte RAM 1GB na cloude DigitalOcean by postačoval na vykonanie testov. My sme tieto testy spúšťali iba lokálne, pretože pre spustenie na cloude by sme potrebovali ďalšiu inštanciu backendu v testovacom prostredí.



Obrázok 7.2. Výsledky automatického testovania

7.1 Hodnotenia užívateľov

Našu aplikáciu testovali dve skupiny ľudí - študenti publicistiky a bežní používatelia. Celkovo náš dotazník vyplnilo 6 študentov publicistiky a 3 nezávislí ľudia. V nasledujúcej kapitole zhrnieme otázky a odpovede na náš formulár:

- Na jakém zařízení jste používali aplikaci? – možnosti boli notebook, stolný pc, smartphone a tablet. Túto otázku sme sa pýtali hlavne z dôvodu, aby sme zistili či a a koľko ľudí by ju použilo na smartfóne. Na smartfóne ju použil jeden z 9 ľudí (naša optimalizácia pre smartfóny nebola zbytočná). Najviac ľudí (7) ju použilo na notebooku a jeden človek na stolnom pc.
- Jak hodnotíte celkový dojem z webové aplikace? (1 nejlepší, 5 nejhorší) – 1 človek hodnotil známkou 1, najviac ľudí (6) známkou 2, známku 3 a 4 udelil jeden človek.
- Jak hodnotíte jednoduchost používání aplikace? (1 nejlepší, 5 nejhorší) – 1 človek hodnotil známkou 1, najviac ľudí (5) známkou 2, známku 3 udelil jeden a známku 4 dvaja používatelia.

- Dokázali byste použít aplikaci FactCheck bez toho, aby Vám někdo vysvětlil, jak se používá? Pokud ne, popište jaké jste měli problémy. – 5 ľudí odpovedalo áno, jeden „Áno ale chvíľu mi trvalo prísť na to že sa musím prihlásiť“, jeden „Dokázal by som. Možno by som však na náhľadovej vstupnej stránke zdôraznil potrebu registrácie.“, jeden „Asi ne. Nebyla jsem si jistá, jestli nutně musím přidat article, abych mohla vytvořit jakýkoli claims (ale třeba to byl jen můj problém).“ a jeden „Zvážil bych, jestli nedat odkaz na vkládání textů už na úvodní stranu. Jinak se jde ke všemu intuitivně proklikat“.
- Vidíte budoucnost v používání podobné aplikace na zdílení a ověřování faktů? – všetci respondeti odpovedali áno. Jeden z nich vyzdvihol jej potrebu na Slovensku, ktorá vraj v prieskume vyšla ako jedna z krajín, kde najvia ľudí verí hoaxerom. Jeden užívateľ napísal, vraj vo Francúzsku k niečomu podobnému pristúpila tlačová agentúra AFP, ktorá robí fact-check pre Facebook ako externá organizácia.
- Myslíte, že může být podobná webová aplikace prospěšná běžným uživatelům? Pokud ne, proč? – Všetci respondenti odpovedali áno. Jeden z nich vyjadril názor, že má význam ale ľudia veriaci hoaxom jej výsledkom pravdepodobne nebudú aj tak dôverovať. Ďalší zdôraznil, že je potrebné definovať lepšiu metodiku vyvracania faktov.
- Poslední otázka - jaký je Váš názor na aplikaci FactCheck? Co by jste na ní zlepšili? – medzi najdôležitejšie postrehy patrí : chýba editovanie hodnotení, sekcia QA pre najčastejšie otázky alebo videonávod, návody na overovanie dezinformácií na webe, loading stavy pre načítanie článkov (pravdepodobne programátor) a „V některých případech nešlo tvrzení či recenze odeslat. Bylo potřeba stránku zavřít a znovu načíst, pak už vše šlo bez problému.“. Posledný problém sme konzultovali s testerkou a narazili sme na problém, že niektoré polia nevyplnila. Pri nesprávnom vyplnení niektorých formulárov sme zabudli pridať notifikácie o neúspešnom pridaní so správou z backendu, čo je nesprávne vyplnené, resp. aká chyba nastala. Tento problém, ako aj loading stavy aplikácie sme už opravili.

V minulom semestri náš systém testovali iní študenti a pýtali sme sa ich rovnaké otázky. Pre porovnanie, 7 z 8 študentov odpovedalo, že bolo pre nich ťažké sa zorientovať na webe a pochopiť, čo je vôbec ich úlohou. Na otázku, či vidia budúcnosť v používaní podobnej aplikácie odpovedal jednoznačné áno iba 7 z 11 respondentov. Snažili sme sa taktiež implementovať nápady, ktoré respondeti mali - pridanie českého/slovenského jazyka, pridanie vysvetlenia systému (stále je však čo zlepšovať), grafický dizajn a vyhľadávanie.

Z ankety usudzujeme, že implementovanie tohoto typu aplikácie má jednoznačne zmysel. Hlavný nedostatok aplikácie vidíme v pochopení funkcionality. Aj keď je na úvodnej strane popis funkcií a účelu systému, je potrebné vytvorenie lepšieho návodu na použitie. Taktiež by bolo vhodné pridať sekciu o tipoch, ako overovať fakty na webe. Pre jej produkčné nasadenie bude tiež potrebné prediskutovať metodiku konečného vyhodnotenia faktu.

Kapitola 8
Budúci rozvoj aplikácie Fact Check

Vo vývoji našej aplikácie aktívne spolupracujeme s tímom v obore umelej inteligencie a profesormi z fakulty FEL na ČVUT. Spolupracujeme s firmami Avast a Gen Digital, ktoré nám poskytli prostriedky na jeho podporu. V jej fungovaní vidíme potenciál do budúcna a preto sa aj po skončení štúdia chystáme tento projekt naďalej rozvíjať a udržiavať. Preto sme na základe spoločných diskusií a užívateľských recenzií navrhli zopár nápadov, ktoré by mohli pomôcť spraviť túto aplikáciu viac pútavú pre používateľov a poskytnúť možnosť jej administrácie priamo v užívateľskom rozhraní. V tejto kapitole rozoberieme prvky, ktoré počas nasledujúcich mesiacov plánujeme vytvoriť.

- Vytvorenie lepšieho návodu na používanie – z užívateľského dotazníka sme zistili, že by bolo najvhodnejšie pridať videonávod a sekciu QA na najčastejšie otázky.
- Vytvorenie `professional` módu – pre komplexné vyhodnocovanie informácií je potrebné zavedenie role experta do systému. Expert je napríklad žurnalista, ktorý musí byť nejakým spôsobom overený (kompletný proces musíme premyslieť) a mohol by celé tvrdenie rozhodnúť. Expert môže využívať sekciu `Populárne články` a `Populárne tvrdenia`, aby videl obsah ktorý momentálne ľudí zaujíma. Expert by mal možnosť finálne rozhodnúť pravdivosť tvrdenia. Z ankety vyplýva, že žurnalisti nemajú úplnú dôveru vo verejné overovanie faktov.
- Vytvorenie administračného portálu – pre budúci rozvoj aplikácie je potrebné vytvorenie systému pre správu užívateľského obsahu. Uvedomujeme si, že podobne ako v mnohých iných sociálnych platformách, aj tu je potrebné mať podporu pre mazanie príspevkov, ktoré by mohli fyzicky, etnicky alebo iným spôsobom urážať či napádať určité skupiny ľudí. Z tohto dôvodu plánujeme vytvorenie administračného systému, ktorý bude slúžiť adminovi (adminom) systému vidieť prehľadné štatistiky všetkých typov príspevkov (články, tvrdenia, hodnotenia). Tieto príspevky by mohli byť topované podľa počtu negatívnych reakcií, poprípade by sme užívateľom mohli poskytnúť pri každom príspevku možnosť nahlásiť nevhodný príspevok. Admin by mal právo tento príspevok vymazať a zabanovať používateľa na určitý čas.
- Umožnenie zdieľania na sociálnych platformách – zdieľaním dosiahnutého skóre na sociálnych platformách by pomohli používatelia rozšíriť náš nápad medzi väčšie okruhy ľudí. Podobne ako v iných hrách, týmto skóre by sa užívatelia mohli prezentovať a upozorniť na svoj obsah. Cez odkaz by sa nový používatelia mohli zaregistrovať a stránka by ich sama oboznámila s jej fungovaním. Momentálne je ale potrebné vytvorenie všeobecných podmienok, ktoré by spĺňali požiadavky spoločnosti Meta zdieľať náš obsah.

Kapitola 9
Záver

Prvotným cieľom diplomovej práce bolo oboznámanie sa s problematikou v oblasti Fact-Checkingu. Analyzovali sme súčasné najpopulárnejšie riešenia a zhodnotili ich funkcie. Sumarizovali sme, prečo tieto riešenia nevyhovujú našim potrebám vytvorenia crowdsourcingovej aplikácie, kde môžu používatelia sami pridávať tvrdenia, o ktorých pravdivosti pochybujú a zároveň overovať tvrdenia ostatných užívateľov.

V spolupráci s vedúcim práce PH.D. Ing. Janom Drchalom sme navrhli riešenie, ktoré by vyhovovalo našim potrebám. Počas semestra sme toto riešenie agilne vyvíjali a konzultovali zlepšenia, ktoré by mohli zvýšiť obľúbenosť aplikácie medzi bežnými používateľmi. Ako hlavnú stratégiu sme navrhli vytvorenie gamifikácie - odmeňovania používateľov za aktivitu. Rozobrali sme teóriu frontend technológií a zvolili vhodné technológie na implementáciu riešenia.

Postupne sme kompletne implementovali frontendovú časť aplikácie Fact Check. Počas implementácie sme spolu s Bc. Rastislavom Kopálom konzultovali backendovú časť - funkcionalitu API, technológie na jej použitie, výber databázy, nasadenie celej aplikácie a automatizáciu buildovania novej verzie po zmene kódu na GitHube. Počas vývoja frontendu sme narazili na niektoré problémy, ktoré spôsobovali pomalý chod aplikácie. Na jej plynulejší chod a zníženie záťaže backendu sme implementovali state management pomocou JS knižnice recoil a navrhli ďalšie zlepšenia z pozorovaní iných sociálnych platforiem.

Na konci prvého aj druhého semestra sme vytvorili dotazník, ktorým sme získali spätnú väzbu na našu aplikáciu. Spätná väzba nám pomohla vyriešiť dva bugy a dala nápad, ako náš portál spraviť jednoduchším na pochopenie pre nových používateľov. Portál sme teda otestovali nielen automatizovanými ale aj používateľskými testami.

Na záver práce sme zhrnuli, aké zlepšenia by sa na našom portáli dali implementovať. Tieto zlepšenia sme rozobrali a počas nasledujúcich mesiacov ich budeme implementovať a pracovať na zlepšení technologickej aj funkčnej stránke aplikácie FactCheck.

Literatúra

[1] Marcos Rodrigues Pinto, Yuri Oliveira de Lima, Carlos Eduardo Barbosa a Jano Moreira de Souza. *Towards fact-checking through crowdsourcing.* In: *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 2019. 494–499.

[2] Jan H Kietzmann. Crowdsourcing: A revised definition and introduction to new research. *Business horizons*. 2017, 60 (2), 151–153.

[3] Joseph G Davis. From crowdsourcing to crowdservicing. *IEEE Internet Computing*. 2011, 15 (3), 92–94.

[4] Jennifer Allen, Antonio A Arechar, Gordon Pennycook a David G Rand. Scaling up fact-checking using the wisdom of crowds. *Science advances*. 2021, 7 (36), eabf4393.

[5] Soroush Vosoughi, Deb Roy a Sinan Aral. The spread of true and false news online. *science*. 2018, 359 (6380), 1146–1151.

[6] Ali Mesbah a Arie Van Deursen. *Migrating multi-page web applications to single-page Ajax interfaces.* In: *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*. 2007. 181–190.

[7] *Deprecating our AJAX crawling scheme.* `https://webmasters.googleblog.com/2015/10/deprecating-our-ajaxcrawling-scheme.html`.

[8] Boris Cherny. *Programming TypeScript: making your JavaScript applications scale*. O'Reilly Media, 2019.

[9] *Developer Survey 2022.* `https://survey.stackoverflow.co/2022/##most-popular-technologies-webframe`.

[10] *React vs Angular: Which JS Framework to Pick for Front-end Development?* `https://radixweb.com/blog/react-vs-angular`.

[11] *React documentation.* `https://reactjs.org/docs/getting-started.html`.

[12] Anurag Kumar a Ravi Kumar Singh. Comparative analysis of angularjs and reactjs. *International Journal of Latest Trends in Engineering and Technology*. 2016, 7 (4), 225–227.

[13] *Antd documentation.* `https://ant.design/docs/spec/copywriting/`.

[14] Elad Elrom. *Integrating State Management*. 2021.

[15] *Recoil: A New State Management Library Moving Beyond Redux and the Context API.* `https://betterprogramming.pub/recoil-a-new-state-management-library-moving-beyond-redux-and-the-context-api-63794c11b3a5`.

[16] Tim Berners-Lee, Roy Fielding a Henrik Frystyk. *Hypertext transfer protocol–HTTP/1.0.* .

[17] *Introducing JSON*.
https://www.json.org/json-en.html.

[18] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds a Clemente Izurieta. Comparison of JSON and XML data interchange formats: a case study.. *Caine*. 2009, 9 157–162.

[19] *Official Node.js guide*.
https://nodejs.org/en/docs/guides.

[20] *The Good and the Bad of Node.js Web App Programming*.
https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/.

[21] *Learn MongoDB and Advance Your Career*.
https://learn.mongodb.com/.

[22] Benymol Jose a Sajimon Abraham. *Exploring the merits of nosql: A study based on mongodb.* In: *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*. 2017. 266–271.

[23] *The OAuth 2.0 Authorization Framework: Bearer Token Usage*.
https://datatracker.ietf.org/doc/html/rfc6750.

[24] Jose A Valdivia, Xavier Limon a Karen Cortes-Verdin. *Quality attributes in patterns related to microservice architecture: a Systematic Literature Review.* In: *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2019. 181–190.

[25] *Key Difference between SaaS, PaaS and IaaS: Level of Control*.
https://www.eginnovations.com/blog/saas-vs-paas-vs-iaas-examples-differences-how-to-choose/.

[26] M Jose Escalona a Nora Koch. Requirements engineering for web applications–a comparative study. *Journal of web Engineering*. 2003, 193–212.

[27] Catherine Sotirakou, Theodoros Paraskevas a Constantinos Mourlas. *Toward the Design of a Gamification Framework for Enhancing Motivation Among Journalists, Experts, and the Public to Combat Disinformation: The Case of CALYPSO Platform.* In: *HCI in Games: 4th International Conference, HCI-Games 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26–July 1, 2022, Proceedings*. 2022. 542–554.

[28] *What is Software Quality Assurance?*
https://www.test-institute.org/What_is_Software_Quality_Assurance.php.

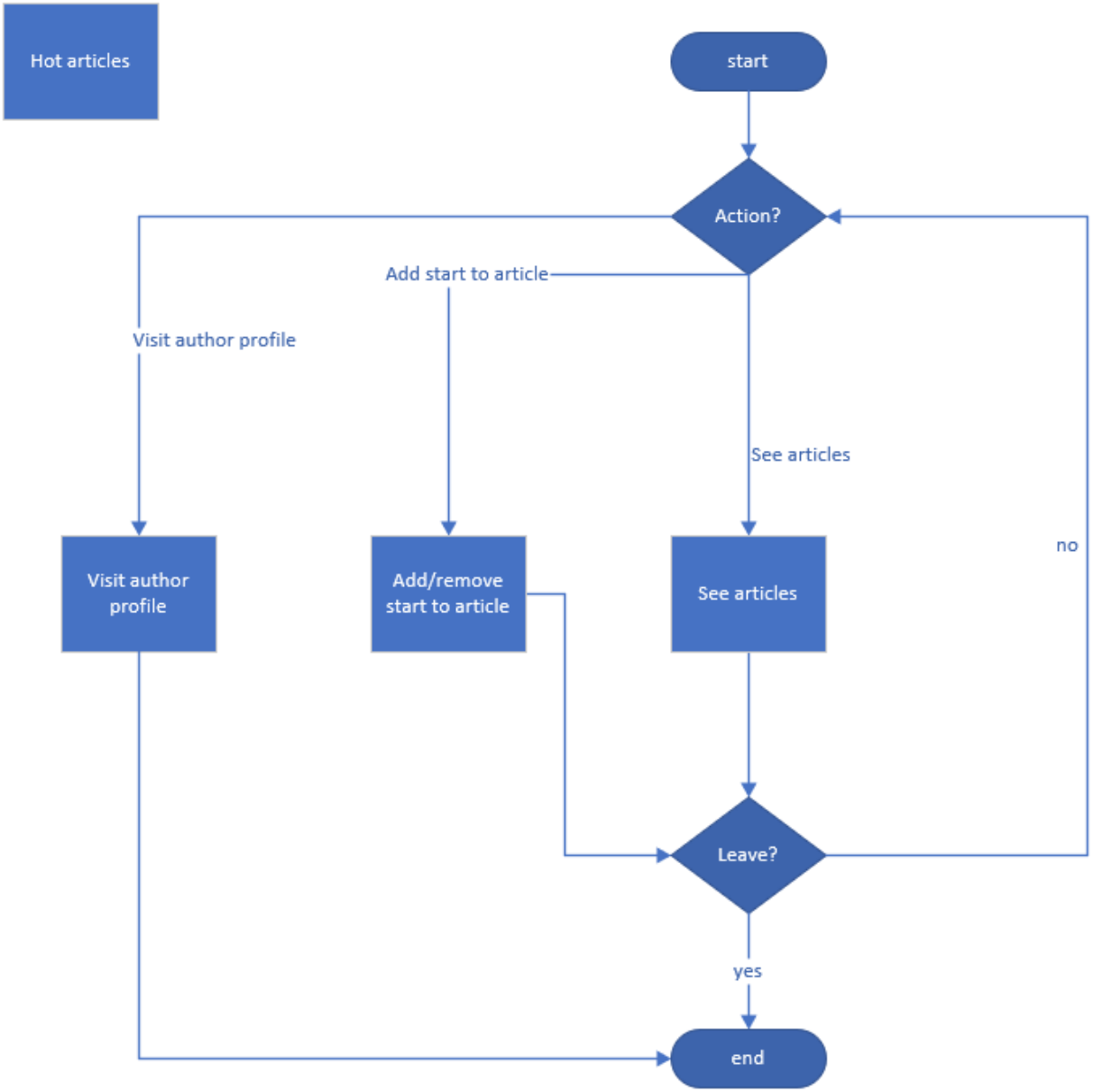
[29] Stephen Vance. *Quality code: software testing principles, practices, and patterns*. Addison-Wesley, 2013.

[30] Michael Olan. Unit testing: test early, test often. *Journal of Computing Sciences in Colleges*. 2003, 19 (2), 319–328.

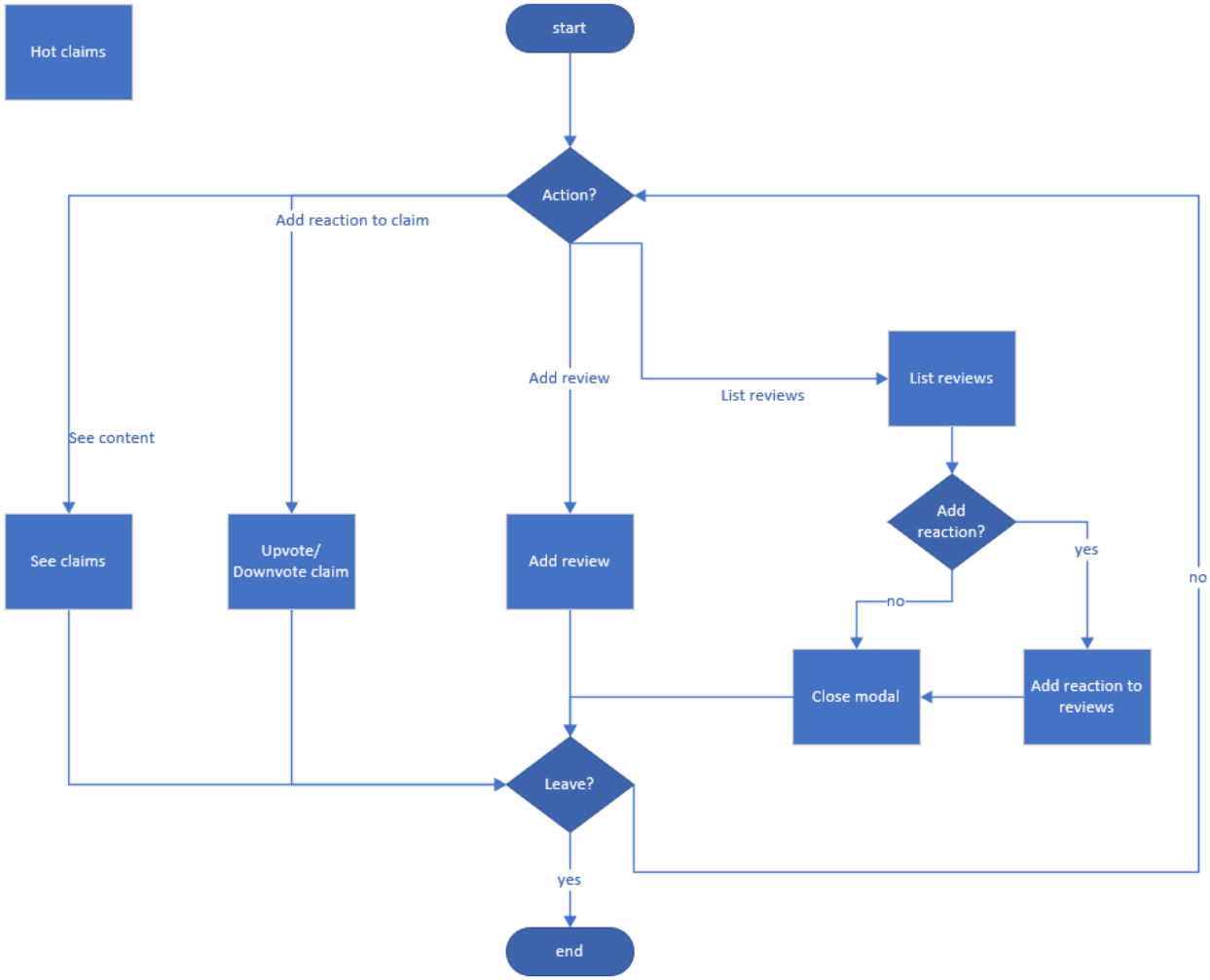
[31] Wei-Tek Tsai, Xiaoying Bai, Ray Paul, Weiguang Shao a Vishal Agarwal. *End-to-end integration testing design.* In: *25th Annual International Computer Software and Applications Conference. COMPSAC 2001*. 2001. 166–171.

[32] *Cypress official documentation.*
`https://docs.cypress.io/`.

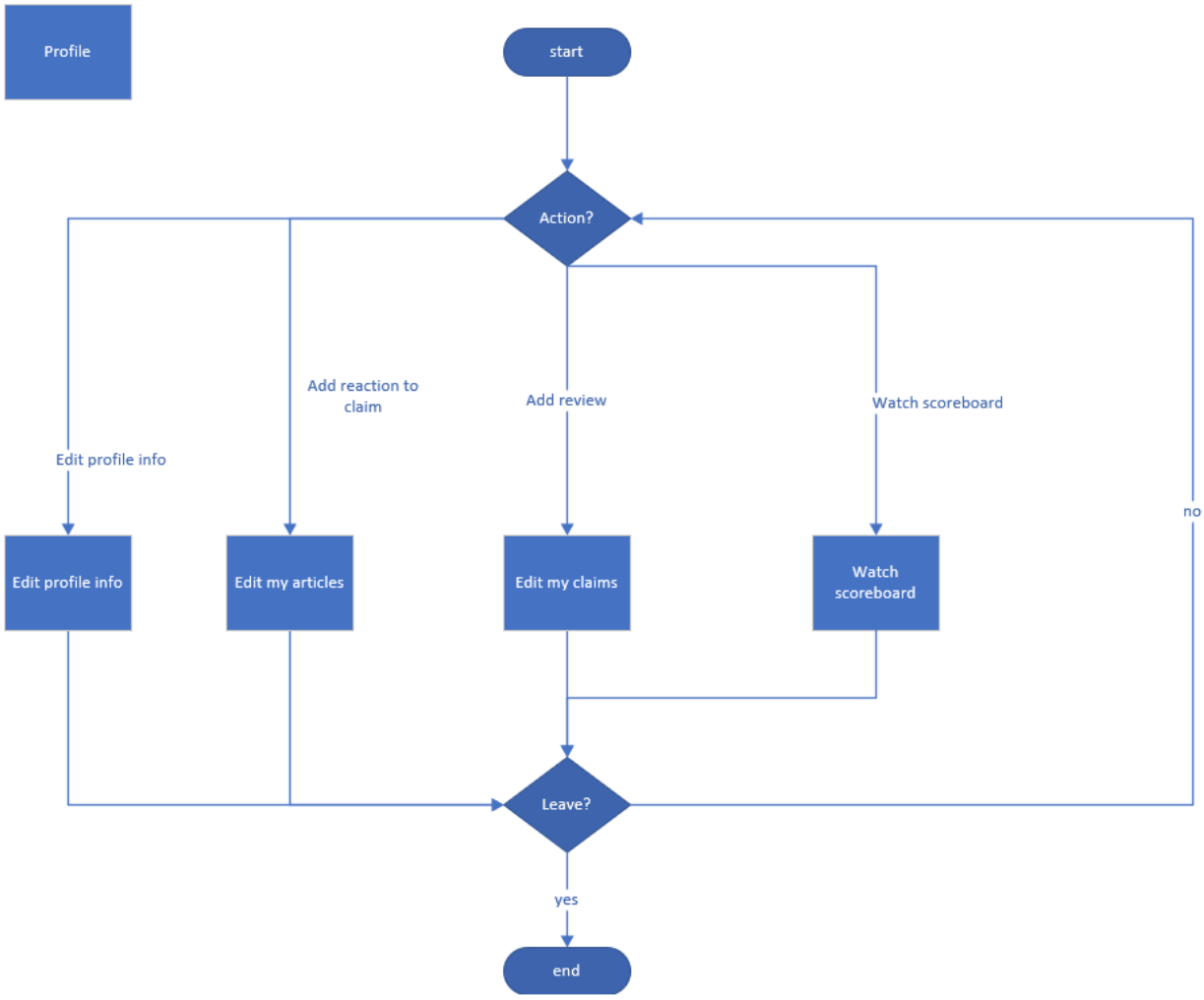
Príloha A
Grafy tokov komplexných podstránok s rozhodnutiami



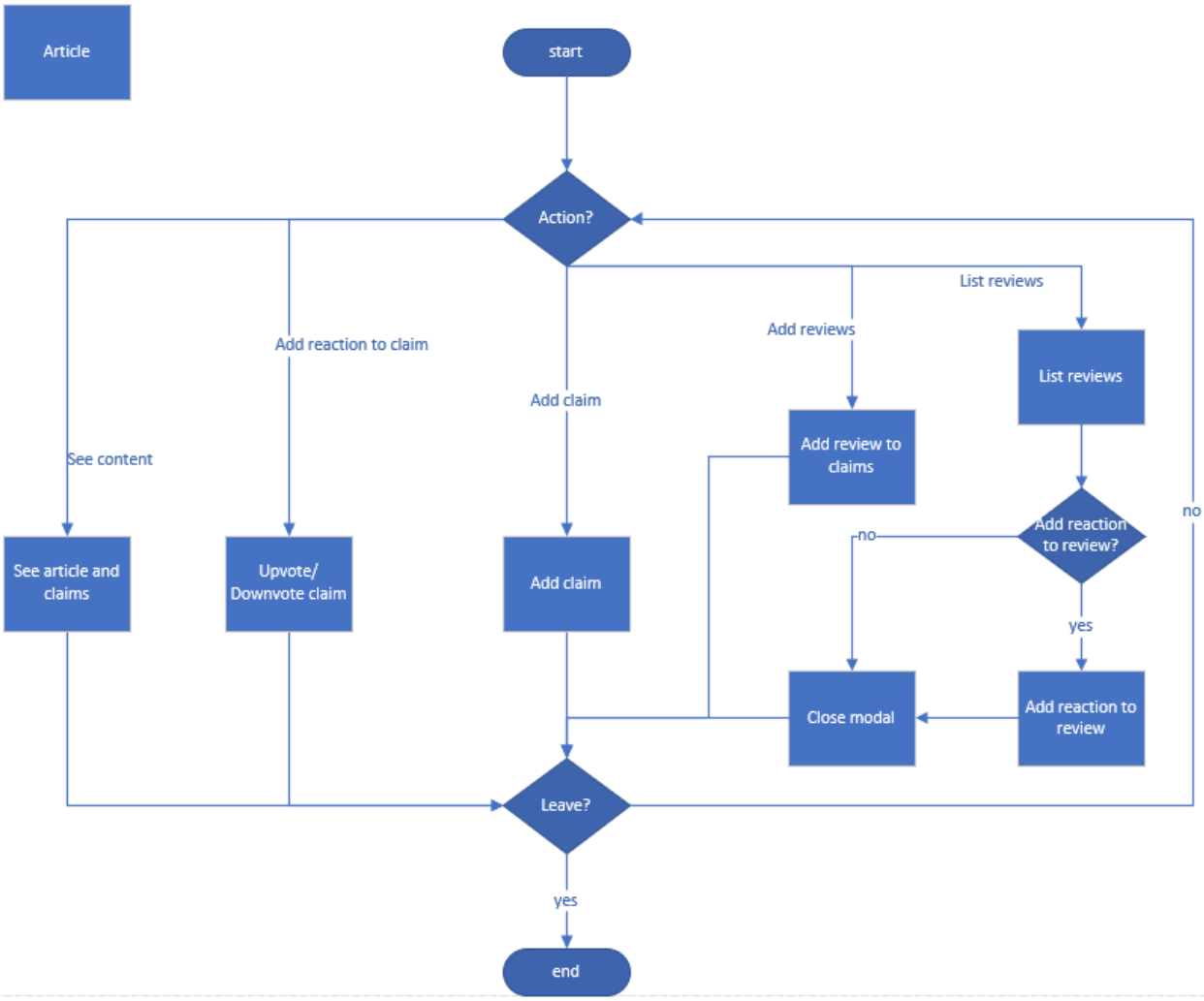
Obrázok A.1. Tok podstránky populárne články



Obrázok A.2. Tok podstránky populárne tvrdenia



Obrázok A.3. Tok podstránky profil



Obrázok A.4. Tok podstránky článok