VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ANALÝZA UŽIVATELSKÝCH NASTAVENÍ SOCIÁLNÍCH SÍTÍ

ANALYSIS OF USER SETTINGS ON ONLINE SOCIAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN MLÝNEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. FILIP JANUŠ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Mlýnek Martin**
Program: Informační technologie
Název: **Analýza uživatelských nastavení sociálních sítí**
Analysis of User Settings on Online Social Networks
Kategorie: Bezpečnost

Zadání:

1. Prostudujte základy tvorby webových uživatelských rozhraní.
2. Seznamte se s experimentálním nástrojem Privchecker pro analýzu uživatelských účtů.
3. Navrhněte webové uživatelské rozhraní nástroji Privchecker.
4. Implementujte navržené rozhraní.
5. Proveďte testování použitelnosti tohoto rozhraní a proveďte testování uživatelských nastavení za pomocí implementovaných nástrojů alespoň na 20 uživatelích.

Literatura:

- JANUŠ, Filip. Pokročilá evaluace úrovně privátnosti v sociálních sítích. Brno, 2020. Master's Thesis. Brno University of Technology, Faculty of Information Technology. 2020-07-15. Supervised by Malinka Kamil. Available from: https://www.fit.vut.cz/study/thesis/18824/
- Saha, Debasmita & Mandal, Ardhendu & Pal, S. (2015). User Interface Design Issues for Easy and Efficient Human Computer Interaction: An Explanatory Approach. International Journal of Computer Sciences and Engineering. 3. 127-135.
- Privchecker: https://github.com/fila43/Privchecker

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Januš Filip, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Bakalářská práce se zabývá tvorbou webového uživatelského rozhraní. Cílem bylo navrhnout a implementovat rozhraní pro serverovou část bezpečnostního nástroje Privchecker, který se zabývá bezpečností uživatelů na sociálních sítí. Zvolený problém byl vyřešen vytvořením klientské webové aplikace založené na JavaScriptové knihovně React. Dále se zde řeší problematika předávání údajů, testování implementovaného rozhraní a analýza uživatelských nastavení.

Abstract

The bachelor thesis deals with the development of a web user interface. The goal was to design and implement an interface for the server part of the Privchecker security tool, which deals with the security of users on social networks. The problem was solved by creating a client web application based on the React JavaScript library. It also addresses the issue of transfering user's credentials, testing of the implemented interface and analysis of user settings.

Klíčová slova

uživatelské rozhraní, webová aplikace, TypeScript, testování, Python, uživatelská zkušenost, React, web scraping, Selenium, privátní skóre, sociální síť

Keywords

user interface, web application, TypeScript, testing, Python, user experience, React, web scraping, Selenium, privacy score

Citace

MLÝNEK, Martin. *Analýza uživatelských nastavení sociálních sítí*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Filip Januš

Analýza uživatelských nastavení sociálních sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Filipa Januše. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Mlýnek
9. května 2022

Poděkování

Chtěl bych poděkovat mému vedoucímu práce Ing. Filipu Janušovi za vedení této práce a za cenné rady, které mi poskytl. Dále bych chtěl poděkovat rodině a mým přátelům za podporu a v neposlední řadě lidem, kteří se podíleli na testování aplikace.

Obsah

Kapitola 1

Úvod

Každý den se člověk setkává se zařízeními jako je mobilní telefon, auto nebo kávovar. Při použití zařízení musí uživatel zadat vstupy do uživatelské rozhraní pomocí tlačítek, dotykového displeje případně hlasu. Zařízení následně vyvolá reakci na danou akci a sdělí své výsledky na výstupu. V historií byla uživatelská rozhraní nejprve na mechanických strojích a poté příchodem 20. století a vzniku počítače se začaly postupně objevovat uživatelská rozhraní jako součást počítačových programů. Na kvalitě uživatelského rozhraní může stát úspěch celé aplikace. S následným příchodem sociálních sítí se začínají shromažďovat data o uživatelích a následné úniky v posledních letech, nenechávají majitele účtu chladnými, a proto hledají aplikace, které slouží pro výpočet bezpečnosti na sociálních sítí, jednou z nich je i aplikace **Privchecker**.

Cílem práce je zvýšit povědomí uživatelů o úniku informací na sociálních sítích, k tomu je potřeba vytvořit grafické uživatelské rozhraní pro nástroj Privchecker, který slouží pro výpočet *privátní skóre* z interního nastavení účtu, pomocí matematických modelů. Privátní skóre numericky odráží riziko ztráty soukromí. Výsledek je potřeba uživateli sdělit jednoduchou vizuální cestou např. pomocí barev, abecedního ratingu, zda-li je jeho soukromí v pořádku, případně jak je jeho soukromí ohroženo. Tato práce chce také nabídnout uživateli interaktivní vizuální nástroj, který mu sdělí pomocí váhy, které atributy v dané sociální jsou nejrizikovější pro ztrátu informace a jaké kroky má uživatel provést pro zlepšení ochrany soukromí. Uživatelé, kteří používají bezpečnostní aplikace kladou velké požadavky na důvěryhodnost dané aplikace, jenž se dá zvýšit intuitivností a přívětivým vzhledem.

Druhá kapitola se zabývá rozborem již vytvořeného nástroje Privchecker. Na jakém principu funguje, jeho aktuální stav a architektura. Dále je zde řečeno jaké používá matematické modely a jak jsou vypočítány jednotlivé parametry, ze kterých se počítá privátní skóre.

V kapitole Uživatelské rozhraní se práce zaměřuje na teorií uživatelského rozhraní, druhy uživatelského rozhraní a koncept *uživatelské zkušenosti*, jenž se snaží analyzovat chování uživatele v aplikaci.

V kapitole Webová aplikace se rozebírají architektonické vzory a používané technologie pro webové aplikace a na závěr porovnávají aplikační rámce, které programátorovi můžou ulehčit práci.

V následující kapitole Implementace se bude čerpat ze získaných vědomostí z předchozí kapitoly a analýzou požadavků aplikace Privchecker, aby byl návrh webového uživatelského rozhraní, co nejpřesnější. Po návrhu uživatelského rozhraní, vytvořením wireframů (drátěné modely) se práce zabývá problematikou předávání uživatelských přístupových údajů, nutných změn na serverové části a implementací grafického uživatelského rozhraní.

Po implementaci aplikace je potřeba ji také otestovat, tím se zabývá kapitola Testování, kde bylo vybráno 20 testerů, aby otestovali aplikaci Privchecker, na základě informací od respondentů byly provedeny určité změny, které jsou zde prezentovány.

1.1 Související práce

Na téma bezpečnost na sociálních sítích bylo napsáno nespočet prací. Jednou z nich je práce [1] od Aghasiana a kol. Ti se snažili počítat privátní skóre z atributů uživatele a brali do úvahy rozdílnou úroveň sdílení stejných údajů na různých sociálních sítích. Vytvořili tedy rámec pro výpočet privátního skóre, který bere v potaz *citlivost* jednotlivých atributů a jejich *viditelnost*. Pojem viditelnost se Aghasian a kolegové oproti starším pracím rozhodli rozložit na tři části *přístupnost*, *spolehlivost* a *náročnost extrakce jednotlivých vlastností*. Na závěr jejich práce svůj model pro výpočet privátního skóre validovali na třech odlišných uživatelích.

V další práci [27] Li a kolegové navázali na práci Aghasiana a kolegů a rozhodli se využít jimi navrhovaný model, který mírně upravili v oblasti citlivosti atributů a nad tímto modelem prováděli experimenty, kde vzájemně porovnávali privátní skóre úplně odlišných uživatelů a rozdíly mezi jimi. Dalším experimentem bylo porovnat skóre v rámci modelů, tedy jimi nově vytvořeným modelem a s modely ostatních lidí včetně modelu Aghasiana a kol. Provedli také experiment, při které sledovali nárůst privátního skóre v případě změny atributu. Na svou práci poté Li a kolegové navázali [26], ale tentokrát využili data z jedné sociální sítě a sledovali vztahy mezi uživateli. Li a kolegové brali do úvahy informace, které o uživateli sdílí jeho přátelé, dále v práci definovali vzorec pro výpočet podobnost dvou uživatelů na základě chování.

Předchozí práce se výpočtem privátního skóre zabývaly převážně ve formě analyzování dat obsahující velké množství uživatelů a jejich vlastnosti. Většina projektů se spíše zaměřuje na vnější zkoumání privátnosti na sociálních sítích, jednou z nich je i služba PrivacySpy[1], která se zabývá zkoumání privátních podmínek, se kterými uživatel souhlasí při vytváření účtu na různých známých webových stránkách. PrivacySpy se snaží přehledně vyjádřit pomocí stupnice, tabulek bezpečnost na daném webu a jak daný web nakládá s uživatelovými soukromými informacemi. Jedním z nástrojů, který se snaží vypočítat privátní skóre na základě získaných atributů uživatele ze sociálních sítí respektive Facebooku je i nástroj usemp-pscore[2], na kterém pracovali Petkos a kol [36].

Příkladem použití webové uživatelského rozhraní pro ovládání bezpečnostní aplikace je produkt Trezor Suite[3], který slouží pro manipulaci s kryptoměnami uložené na hardwarové peněžence uživatele.

[1] https://privacyspy.org
[2] https://github.com/MKLab-ITI/usemp-pscore
[3] https://suite.trezor.io/web/

Kapitola 2

Bezpečnost na sociálních sítích

V této kapitole se definuje pojem sociální sítě, dále se proberou základní znalosti týkající se výpočtu rizika úniku soukromí na sociálních sítí jako je privátní skóre a faktory, které jej ovlivňují. Na závěr kapitoly budou řečeny informace o nástroji Privchecker, jeho princip a použité modely, pomocí kterých nástroj vypočítává privátní skóre.

2.1 Problematika

V dnešní době už není pochyb, že sociální sítě jsou velkým fenoménem a jsou hojně používány masami lidí po celém světě. Z undergroundového média, jenž používali převážně nadšenci, se sociální sítě staly nástrojem pro vyměňování informací, navazování vztahů, prodávání věcí, případně vyjádření pocitů v diskuzi. Každá sociální sít je jinak specializovaná. Linkedin se specializuje na profesní komunikaci (sdílení záznamů o zaměstnání, nabírání nových zaměstnanců). Zatímco Facebook je spíše sociální síť zaměřená na komunikaci mezi přáteli.

S používáním sociálních sítí se pojí i riziko ztráty soukromí a únik dat. Uživatel sdílí citlivé i necitlivé informace, aby mohl využít veškerou potřebnou funkcionalitu aplikace. Informace, které jsou sdíleny zahrnují osobní zájmy, názory, lokaci, kde se nachází, zaměstnání apod. Sdílení těchto informací může být zneužito pro krádež identity, cílený marketing s klamavou reklamou, pronásledování a kyberšikanu. [1]

V tomto ohledu sociální síť nabízí uživatelům nastavení soukromí, které lze přizpůsobit jejich potřebám. Tato nastavení jsou většinou velmi složitá a uživatel toto nastavení přeskočí a také si nemusí uvědomit, závažnost rizika úniku dané informace. Uživatel by měl mít právo chránit své soukromé informace před svou rodinou, přáteli, známými.

Existují určité modely pro výpočet a kvantifikaci rizik úniku soukromých informací. Použitím modelů, jejímž výstupem je tzv. privátní skóre se zvyšuje určitá míra povědomí o soukromí, díky které mohou jednotlivci mít lepší pohled na své soukromí a v případě nespokojenosti uplatňovat bezpečnostní opatření ke zvýšení úrovně svého soukromí. Jedním z nástrojů, které implementují tyto modely je i nástroj Privchecker.

2.2 Sociální síť

Sociální síť lze definovat jako graf $G = (V, E, P, B)$, kde množina uzlů V představuje uživatele sociální sítě. Množina E je množina hran, která představuje vztahy mezi jednotlivými uživateli. V závislosti na daném druhu sociální sítě lze říci jestli mají hrany směr nebo

ne. Pokud je v sociální síti navazování vztahů vzájemné, jedná se o *neorientovaný graf*. Příkladem je přátelství na Facebooku. V případě, kdy na sociální síti je aplikován model „sledování“, jedná se o *orientovaný graf*. Příkladem sociální sítě, která je orientovaným grafem je sociální sít Instagram. Množina $P = p1, p2, p3, ...pn$ představuje osobní informace vyplněné uživatelem, $p = a1, a2, a3, ...an$, představuje atributy daného uživatele a každé a představuje hodnotu daného atributu. Množina B je soubor akcií a chování, které může uživatel na sociální síti vykonat. Může sem patřit přidáni příspěvku, komentování, lajkování. [26]

2.3 Měření soukromí

Při měření soukromí na sociálních sítích se hlavními vstupy berou *citlivost* a *viditelnost* informací. Funkce označuje citlivost, každého atributu ze zdroje. Určení váhy citlivost je složitý úkol, jelikož každý uživatel má na soukromí svůj vlastní subjektivní názor, který může být ovlivněn i jeho původem a kulturou. Kromě výpočtu citlivosti je potřeba poskytnout vzorce pro výpočet viditelnosti. Viditelnost lze rozdělit do určitých faktorů. Faktory jsou dostupnost informací, obtížnost získávání informací a spolehlivost dat. Výsledný výstup výpočtu privátního skóre je kombinací viditelnosti a citlivosti jednotlivých atributů. [27]

2.4 Data o uživatelích

Jednotlivá data, ze kterých vznikají jednotlivé atributy lze rozdělit na strukturovaná a nestrukturovaná. Atributy tvořené z nestrukturovaných dat se musí složitě extrahovat z uživatelských příspěvků, obrázků za pomocí použití strojového učení nebo manuálně. Zatímco u textových, lze použít určitou automatizovanou analýzu, u obrázků a fotek je to mnohem složitější. V případě strukturovaných dat jsou data jednoduše extrahována z textových bloků jako je jméno, příjmení, věk. [1]

V případě nástroje Privchecker se nepočítá s nestrukturovanými daty a výhradně se pracuje se strukturovanými daty, přičemž jsou extrahována nastavení soukromí v rámci sociální sítě.

2.5 Privátní skóre

Privátní skóre slouží jako míra rizika úniku informací. Určit privátní skóre je velmi problematické, protože každý uživatel vnímá soukromí odlišně. V novějších studii od Li a kol. se snaží privátní skóre, pomocí tohoto vzorce. [27]

$$p = \frac{\sum_{i=1}^{m} v_i * s_i}{m}$$

Kde m je počet sociálních sítí, ze kterých se čerpají data. v_i je viditelnost atributu a s_i je citlivost atributu.

Nástroj Privchecker, nepočítá s vzájemným ovlivňováním atributů z různých sociálních sítí a používá pro výpočet privátního skóre model `C-PIDX`, jenž je detailněji probrán v sekci 2.6.1.

2.6 Nástroj Privchecker

Nástroj Privchecker slouží pro výpočet privátního skóre z vnitřního pohledu přihlášeného uživatele. Podporovanými sítěmi jsou Facebook, Instagram, Tumblr, Linkedin, Google, Pinterest a Twitter.

Privchecker je architektonicky rozdělen na 3 části. První částí je `extraktor`, který přistupuje ke sociálním sítím pomocí techniky zvané *web scraping*, jejímž smyslem je automatické procházení a stahování informací z webových aplikací. [23]

Extraktor stažené položky nastavení a jejich hodnoty zpracuje a uloží do předem daného formátu. Zpracované atributy nastavení ze sociální sítě evaluátor vloží na vstup modelu společně s váhami jednotlivých atributů. Vypočtené privátní skóre se předá vyhodnocovacímu modulu, který v rámci jedné sítě vypíše interval hodnot, ve kterém se pohybuje výsledné privátní skóre uživatele, včetně vypočtené hodnoty. Součástí vyhodnocovacího modulu je i nápověda, jejíž účelem je pomoct uživateli zlepšit soukromí na sociálních sítích. [23]

2.6.1 Použitý model

Použitým modelem pro výpočet privátního skóre v nástroji Privchecker je model `C-PIDX`, který kombinuje modely `M-PIDX` a `W-PIDX`. V rámci předchozí práce byl model `C-PIDX` zvolen, protože dobře reflektuje inkrementální změny atributů a také reflektuje změny aktuálního atributu. Další výhodou použití modelu `C-PIDX` je, že reaguje na nastavení, která jdou protichůdným směrem než ostatní nastavení. [23]

Příkladem může být nastavení ze sítě Facebook *Chcete kontrolovat příspěvky, ve kterých vás někdo označil, než se budou moct zobrazit na vaší timeline?*. Tím uživatel omezí zobrazování příspěvků na jeho profilu. V rámci modelu jsou tato protichůdná nastavení považována za nastavení, která pozitivně ovlivňují skóre. [23]

2.6.2 Nedostatky nástroje

Nástroj Privchecker sice poskytuje výpočet privátního skóre na více platformách, ale pokud uživatel zadá více sociálních sítí, výpočet celkového privátního skóre není nikterak ovlivněn. Jeden zdroj riziko ztráty soukromí nemusí odhalit, ale kombinací více sociálních sítí může odhalit ztrátu dat.

Ve studii [1] bylo prokázáno, že s nárůstem používání více sociálních sítí se zvyšuje riziko úniku informací. Příkladem úniku dat může být sdílení data narození na jedné platformě a na druhé ne. Paradoxem může také být, že uživatel by určité informace na jedné sociální síti nesdílel, ale na jiné sociální síti mu to přijde naprosto v pořádku. [1]

2.7 Citlivost atributu

Citlivost je riziko spojené s atributy uživatele, pokud se zvýší citlivost, zvýší se riziko představující odhalení soukromé informace. [1] Na zjištění citlivosti jednotlivých atributů lze použít kvocientový model pro výpočet citlivost nebo dotazník.

V případě nástroje Privchecker, byl použit zdroj vah citlivosti atributů studie, která citlivost získala pomocí dotazníku [23]. Množství odpovědí a škála výběru, při jednotlivých atributech se zde neuvádí. Naopak ve studii [27] počítali citlivost atributů také pomocí dotazníku, kde měli u každého atributu škálu od L1 (v pořádku) až po L5(znepokojivé). Hodnoty L1-L5 udávají procentuální počet odpovědí u daného atributu. Na výsledky z do-

tazníku poté aplikovali vzorec pro výpočet citlivosti atributu.

$$S = \frac{0,5 * L3 + L4 + 1,5 * L5}{1,5}$$

2.8 Viditelnost atributu

Viditelnost udává jak široce jsou přístupné jednotlivé atributy daného uživatele v rámci sociální sítě. Viditelnost atributů se různí. V případě telefonního čísla může být viditelnost v rámci přátel nebo úplně privátní. Naopak jméno a příjmení v určitých sítích je veřejná informace. [27]

Nástroj Privchecker bere viditelnost atributu, podle jeho nastavení. Rozděluje viditelnost do tří hodnot (Soukromé = 0, Přátelé = 1, Přátelé přátel = 2, Veřejný = 3). [23]

Novější studie se snaží rozdělit pojem viditelnost do tří faktorů. *Faktor přístupnosti*, který je významově ekvivalentní pojmu viditelnosti v rámci nástroje Privchecker. *Faktor obtížnosti extrakce*, který se snaží definovat náročnost získání atributu ze strukturovaných či nestrukturovaných dat. Hodnota obtížnosti je zadávána manuálně na základě subjektivního pocitu ($3 \rightarrow nizka, 2 \rightarrow stredni, 1 \rightarrow vysoka$) a výpočet obtížnosti probíhá následovně. [1] [27]

$$F_{dif} = \frac{\sum_{i=1}^{n} (d_i f_i)}{n}$$

Celková obtížnost extrakce atributu F_{dif} je tedy spočtena jako průměr obtížnosti extrakce atributu d_i ze všech platforem f_i, kde se atribut nachází [1].

Kritérium spolehlivosti udává s jakou spolehlivostí, je konkrétní atribut zveřejněn v jednom nebo více zdrojích. Spolehlivost informací atributu se zvyšuje s počtem zdrojů, které atribut obsahují. Pro výpočet spolehlivosti se používá funkce F_{rel}. [1]

$$F_{rel} = \frac{2}{1 + e^{-s}} - 1$$

Kde s je počet zdrojů, kde se atribut nachází. Výsledek funkce je v rozpětí (0, 1), kde se zvyšujícím počtem zdrojů zvyšuje spolehlivost pozorovaného atributu.

Kapitola 3

Uživatelské rozhraní

Tahle kapitola se zabývá teoretickou stránkou uživatelského rozhraní. Nejprve je potřeba si definovat, co vlastně uživatelské rozhraní představuje. Je zde uvedeno, co je potřeba pro správný návrh aplikace, protože pokud aplikace nezpřístupní uživateli informace ve formě, která je mu blízká, software je pak pro něj nepoužitelný. Probírají se zde jednotlivé druhy rozhraní, jejich výhody a nevýhody. Řeší se zde pojem user centered design a závěrem jsou rozebrány faktory uživatelské zkušenosti.

3.1 Co je to uživatelské rozhraní?

„Uživatelské rozhraní je soubor metod nebo prostředků, kterými koncový uživatel komunikuje s produktem softwarovým nebo hardwarovým zařízením“ [5]. Je navrženo tak, aby uživatelům poskytovalo nejefektivnější a nejpříjemnější přístup. Mnozí experti se dnes shodují na designu zaměřeném na uživatele a jeho potřeby, dříve byl i design zaměřený stroj. [5]

Rozhraní hraje velkou roli při použitelnosti a výběru aplikace. V posledních letech se začíná ukazovat, že čím dál více uživatelů používá větší počet různorodých zařízení. Některé typy aplikace se lépe ovládají na stolním počítači, jiné na mobilním zařízení, uživatel si vybere, kterou platformu zvolí pro vykonání daného úkolu. S rostoucím počtem platforem, je pro návrháře čím dál větší výzva vytvořit přívětivá rozhraní. [5]

Uživatelské rozhraní je tedy definováno jako prostředek, kterým uživatelé interagují s obsahem, aby dosáhli nějakého cíle. Úkolem návrhu uživatelského rozhraní je vytvořit prostředí, které uživateli a počítači umožňuje vyměňovat si zprávy potřebné k provedení konkrétního úkolu. Uživatelská rozhraní se liší systém od systému a uživatel od uživatele. [38]

3.2 Historie uživatelského rozhraní

V dobách, když byl počítač ještě v rané fázi se komunikovalo pomocí různých mechanických tlačítek, děrných štítků. Jednotlivé úkony se prováděly v dávkách a trvalo hodiny, dny, než se dostaly výsledky na výstup. Složitá uživatelská rozhraní byla považována za náročná, maximální výkon šel do softwaru, který prováděl výpočet. Všechno se změnilo příchodem rozhraní příkazového řádku. Snížila se celková odezva systému. Interakce se sestává z požadavku a následné odpovědi v rámci okamžiku. [39]

S postupným technologickým pokrokem bylo jasné, že přijde první grafické rozhraní. To se objevilo v 70. letech 20. století ve výzkumném centru Xerox Palo Research Center

a stalo se začátkem mnoha inovací. S příchodem operačního systému Mac OS System 1 se začíná objevovat vzor v podobě oken a ikon. Velkým skokem dopředu byl operační systém Windows 95 a jeho dodnes přetrvávající nabídka `START`, která se zachovala do dnešních dnů. [39]

Když se v 90. letech 20. století rozšířil World Wide Web a technologie HTML, objevuje se grafické rozhraní v rámci webových stránek. Přechod do 21. století lidem přinesl mobilní telefony, a když si lidé uvědomili, že mobilní telefon nemusí sloužit jen jako nástroj pro telefonování, rozmohl se vývoj aplikací pro mobilní zařízení. V současné době se můžeme setkat s rozhraními ovládané hlasem, případně ovládané pohybem těla, které zachytávají speciální kamery. [39]

3.3 Typy uživatelského rozhraní

Na základě vstupů do aplikace zadané uživatelem se prezentují výstupy. V jaké formě jsou vstupy a výstupy udává typ uživatelského rozhraní.

3.3.1 Rozhraní příkazového řádku (Command line interface)

Rozhraní příkazového řádku (CLI), je nízkoúrovňové rozhraní, kde vstupem je posloupnost znaků zadaná klávesnicí. Vhodné pro automatizaci, je jednodušší vykonat příkaz než posloupnost událostí v grafickém rozhraní. [2]

Nejznámějším příkladem CLI je terminál v UNIXových systémech nebo v systému Windows. Výhodou je rychlejší vykonávání akcí a stejnost příkazů v průběhu let. CLI má menší nároky na procesor. Pro vykonávání příkazů v CLI může vyžadovat zkušenosti nebo znalost příkazů, které jsou dostupné v manuálových stránkách týkající se CLI. Cílovými uživateli CLI jsou zkušenější uživatelé, většinou programátoři. [2]

3.3.2 Grafické uživatelské rozhraní (Grafical user interface)

Grafické uživatelské rozhraní (GUI) je výsledkem kombinací příkazů, menu, ikon. GUI se zdá pohodlnější pro méně technické uživatele, protože používá obrázky, grafiku, která reprezentuje funkce systému. GUI v rámci počítače využívá vstupy od klávesnice a myši. [2]

Složitost jednotlivých akcí je uživatelům skryta. Uživatel má okamžitou vizuální zpětnou vazbu. Nevýhodou GUI je větší náročnost na zdroje: paměť, procesorový čas. Dalším problémem může být, že některé funkce programu se při špatném návrhu uživatelům skryta. Když se používá grafické rozhraní, neznamená to nutně odstranění příkazového řádku. [2]

3.3.3 Hlasové uživatelské rozhraní (Voice user interface)

Hlasové uživatelské rozhraní je rozhraní (VUI), kde uživatel zadává příkazy pomocí hlasu. Největší výhodou je tedy, že může pracovat s aplikací aniž by by se dotýkal zařízení. Implementačně je VUI výzva, vzhledem k tomu, že tento typ rozhraní je v rané fázi vývoje. [2]

Ve VUI neexistuje žádná vizualizace, takže při pohledu na VUI uživatelé neví jistě co aplikace umí. V případě hlučného prostředí je větší šance při zadání příkazu chybná interpretace a provedení chybné události. Příkladem VUI jsou hlasoví asistenti jako je Alexa, Google asistent, Siri, Cortana. [2]

Existují i další typy rozhraní, v dnešní době již nepoužívané textové rozhraní, které bylo určitým mezistupněm mezi rozhraním příkazového řádku a grafickým rozhraním. Jelikož GUI

je nejpoužívanějším uživatelským rozhraním dnešní doby a zároveň je využíváno webovými aplikacemi, GUI bude součástí návrhu a implementace uživatelského rozhraní pro nástroj Privchecker.

3.4 Aspekty návrhu rozhraní

Před samotným návrhem uživatelského rozhraní je potřeba uvědomit si pro jakou sortu uživatelů je rozhraní zamýšleno. Bere se v potaz, že uživatelé nejsou stejní, každý myslí jiným způsobem a určité věci si vyloží po svém.

Pokud budou aplikaci vykonávat tělesně postižení lidé, je na místě se tím zabývat při návrhu. Když program bude vykonávat nevidomý člověk, je příhodné poskytnout mu zvukovou odezvu s pokyny a s následným výstupem [18]. Důchodce zřejmě nebude rozumět slovům mladé generace, pokud je to možné volí se slova srozumitelná pro všechny druhy uživatelů.

3.4.1 Výkon

Náročnost systému je velmi důležitá. Uživatel požaduje, aby aplikace bylo co nejméně náročná na hardware. S tím také souvisí náročnost grafického rozhraní, které je součásti většiny systémů. Významným měřítkem je doba načítání. Dobou načítání se myslí doba od spuštění aplikace k vykreslení první obrazovky uživatelského rozhraní. Pokud je doba načítání příliš dlouhá působí to špatným dojmem. Doba načítání závisí na počtu a typu prvků uživatelského rozhraní, které je potřeba vykreslit. [38]

Dobu načítání můžeme eliminovat snížením počtu ovládacích prvků (tlačítka, nabídky, dialogová okna). Pokud uživatelské rozhraní načítá obrázky, můžeme použít obrázky menšího rozlišení, které zároveň nepůsobí rozostřeným dojmem. [38]

3.4.2 Konvence

Při vytváření rozhraní by měl být respektovány uživatelsky známé konvence. Někdy je vhodné řídit se konvencemi než experimentovat. Uživatelé na rozhraní nepřicházejí na rozhraní s nulovými vědomostmi, ale mají zkušenosti z používání jiných systémů. Konvence se porušují zřídkakdy, tehdy kdy porušení konvence poskytne výhodu nebo se tím programátor vyhne problému. [5]

Důležitá je také konzistence v uživatelském rozhraní, stisknutí tlačítka neprovádí pokaždé jinou činnost [5]. U prvků podobných typů např. u tlačítek, které vykonávají důležitou činnost se nastavuje stejná barva. Pokud například tlačítka změní tvar při najetí myší, měla by reagovat podobným způsobem pro podobnou akci [5]. V kvalitní aplikaci se také většinou zachovává použití stejného písma.

3.5 User centered Design

User centered design je design zaměřený na uživatele. Jedná se filozofii, kde uživatel s jeho potřeby, požadavky a omezeními je středem pozornosti návrhu aplikace a jejího životního cyklu. Častou chybou při použití filozofie User centered designu je maximální soustředěnost na uživatele a zanedbání obchodních cílů. [40]

Jak již bylo dříve zmíněno je potřeba najít rovnováhu mezi oběma požadavky. User centered design je proces rozdělený do čtyřech úrovní a každá úroveň se liší řešením odlišných problému a má stanovené rozdílné výstupní cíle. Úrovně jsou následující

- **Požadavky** – V této části se specifikují obchodní cíle a požadavky, které je potřeba splnit pro úspěch aplikace. Je potřebné si také definovat velikost projektu, rizika a plán samotného vývoje. Výstupem je specifikace požadavků. [40]
- **Analýza uživatelů** – V této části mají návrháři za úkol získat hlubší poznatky o cílových uživatelích a pochopit jejich potřeby pomocí dotazníků, vytváření person, hloubkových rozhovorů. Výstupem je popis uživatelů. [40]
- **Tvorba** – Ve téhle fázi se návrhový tým zabývá samotnou tvorbou konkrétního uživatelské aplikace na základě získaných poznatků z předchozí části. Při tvorbě uživatelského rozhraní se využívá skicování, vytváření wireframů (drátěných modelů) a prototypů. Výstupem je aplikace. [40]
- **Vyhodnocení** – Fáze vyhodnocení se zabývá ověřením zdali výsledná aplikace podle daných představ. Jestli výsledek odpovídá představě lze zjistit pomocí uživatelského testování. Výstupem je pak zpráva z testování. [40]

  U uživatelského testovaní je potřeba definovat jaké informace se budou získávat od uživatele. Na základě získaných informací z testů se provádí změna v řešení aplikace. Je důležité začít s testováním co nejdříve, aby se co nejrychleji eliminoval problém, který může způsobit v budoucnu větší a nákladnou opravu. [40]

3.6 User experience

Termín *user experience* (uživatelská zkušenost) je podle Jakoba Nielsena a Dona Normana shrnut ve článku *The Definition of User Experience* (UX) jako „Uživatelská zkušenost zahrnuje všechny aspekty interakce koncového uživatele se společností, jejími službami a jejími produkty.“ [33]

User experience zahrnuje široký přístup různých znalostí a perspektiv, jako je informační architektura, obsah, vizuální design, interakce člověka s počítačem, zvukový design, lidské faktory, průmyslový design a architektura aplikace. [41]

Čím lepší uživatelská zkušenost dané aplikace je, tím větší je motivace zákazníka aplikaci využívat případně doporučit svým známým. Tím se vytváří lepší pozice a výhoda aplikace na trhu. Úspěch dobré uživatelské zkušenosti je správná práce se zpětnou vazbou koncových uživatelů, usnadnění funkcí, které jsou klíčové pro cíl použití. [41]

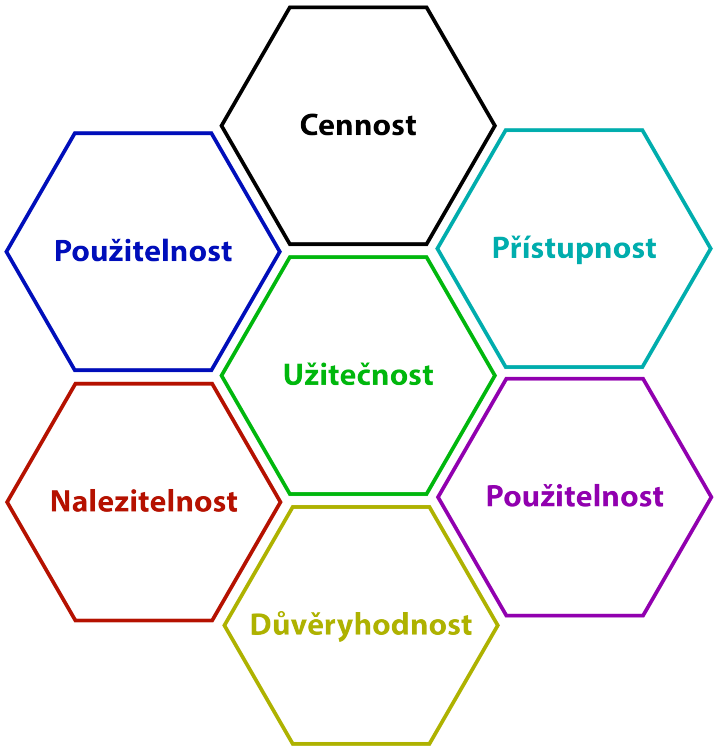
Je urychlen proces nutných částí např. registrace a jsou vytvořeny nástroje pro urychlování rozhodování zákazníka. Je potřeba vytvořit takové uživatelské rozhraní a uživatelský zážitek, který splňuje firmou definované obchodní cíle a zároveň je součást aplikace. Klíčem je tedy vysoce interaktivní a rychle reagující uživatelské rozhraní. [41]

Alfou a omegou všech aplikací je také bezproblémová zákaznická zkušenost, protože uživatel si z psychologického hlediska vsugeruje do paměti negativní zážitek než ten pozitivní. Od návrhu user experience se odvíjí všechny ostatní funkce a jakým způsobem jsou technicky implementovány.

3.6.1 Faktory ovlivňující User Experience

Peter Morville v roce 2004 definoval 7 základních faktorů uživatelské zkušenosti pomocí modelu včelí plástve, který lze vidět na obrázku 3.1. Model plástve pomáhá stanovit si priority na které části uživatelské zkušenosti se v aplikaci zaměřit. Lze tak upřednostnit jeden faktor před druhým a je tak lepší dělat kompromisy mezi jednotlivými faktory vědomě než nevědomě. [31]

Model dále podporuje modularitu a je možné rozdělit práci do více týmů a každý tým se bude specializovat na jeden faktor uživatelské zkušenosti. Model plástve může ve vývoji sloužit i jako kontrola toho, co bylo zatím ve vývoji u grafického uživatelského rozhraní splněno. [31]



Obrázek 3.1: Včelí plástev User Experience. Převzato z [31]

Užitečnost

Udává jak moc je obsah aplikace, který je poskytován užitečný, originální a jedinečný pro koncového uživatele. [31] [41]

Použitelnost

Aplikace je jednoduše naučitelná a rychle se její ovládání uživateli vsugeruje do paměti. Systém by měl uživateli poskytnout po seznámení větší pracovní efektivitu. Aplikace má minimum chyb a uživatel při kritických akcích má možnost zpětného kroku. Pokud nějaké chyby nastanou tak veškeré chyby jsou uživatelovi jasně interpretovány a uživatel ví co se systémem děje a má nad ním kontrolu. [31] [41]

Použitelnost není vlastnost, které lze dosáhnout předem daným počtem kroku daného postupu. Je propojená se skupinou koncových uživatelů a ověření použitelnosti se provádí

pomocí testování s reálnými uživateli např. sledováním kolik chyb udělá návštěvník při vykonávání registrace, případně rychlostí vykonání tohoto úkolu. [31] [41]

Žádanost

Faktor týkající se image, značky, pověsti firmy, zabývání se s emoční stránkou aplikace, jakého druhu (kladného či záporného) jsou vyvolávané emoce v uživateli.[31] [41]

Nalezitelnost

Uživatelé mají možnost nalézt a dostat se na webovou aplikaci. Řeší se zde optimalizace vyhledávačů, sdílení přes sociální sítě, mailové kampaně. [31] [41]

Přístupnost

Webová aplikace by měla být přístupná z heterogenně odlišných zařízeních jako mobilní telefon, počítač, televize za předpokladu, že je to pro daný druh zařízení vhodné. V dnešní době se používání různých úkonů v rámci aplikací přesouvá na mobilní telefony a lidé využívající jenom mobilní telefony tvoří nemalou část trhu. [31] [41]

Přístupnost se také týká neopomenutí zrakově či sluchově postižených lidí a jejich požadavků. Také bere do úvahy uživatelovo vzdělání. Zabývání se přístupností je velkým přínosem z hlediska marketingu, protože se díky tomu zvyšuje množství uživatelů, kteří aplikaci budou používat. [31]

Důvěryhodnost

Uživatelé jsou v používání internetových aplikací opatrní a proto je potřeba vyvolat dojem, že musí aplikaci věřit a pokládat obsah, který je jim podsouván za důvěryhodný. [31] [41]

U důvěryhodné aplikace je poté uživatel ochoten sdílet své osobní údaje, nakoupit produkt a provést další úkony, které můžou pro majitele webu vést k splnění obchodního cíle. Důvěryhodnost aplikace se dá dosáhnout pomocí reference mezi známými, použití šifrované komunikace pomocí protokolu `https` a využití certifikátů, případné ověření od nezávislé třetí strany jako jsou MCAfee[1]. [41]

Cennost

Produkt musí přinášet hodnotu. Hodnota produktu musí být výhodná jak pro vývojáře tak i pro zákazníka. Pokud aplikace dlouhodobě šetří čas nebo peníze uživatelů pak lze říct, že je vysoká pravděpodobnost úspěchu a zvýšení faktoru cennosti aplikace. [31]

[1] http://mcafee.com

Kapitola 4

Webová aplikace

Tato část poskytne teoretické znalosti v rámci vývoje webových aplikací, používané programovací jazyky pro vývoj webových aplikací a srovnání jednotlivých frameworků (aplikační rámců).

„*Webová aplikace* je aplikační program, který může být uložen na vzdáleném serveru a je pohodlně dostupný přes internet prostřednictvím rozhraní prohlížeče“ [8]. Informace z této kapitoly budou použity v následující implementaci uživatelského rozhraní pro nástroj Privchecker.

Webové aplikace mají oproti desktopovým, mobilním zařízením řadu výhod. Mezi přední výhodu webové aplikace patří univerzálnost. aplikace je spustitelná na různých druzích platforem jako je počítač, mobilní telefon případně televize, stačí k tomu jen již zmíněný prohlížeč. S tím se také souvisí nezávislost na operačním systému, webové aplikace nejsou vázány na verze případně druhy operačních systémů. Webové aplikace mají nízké nároky na hardware, lze je spustit na většině zařízení, co dokážou spustit prohlížeč. Webové aplikace bývají, ale častěji napadány hackery pomocí různých útoků [22]. [8]

4.1 Architektura klient server

Moderní webové aplikace jsou založeny na architektuře *klient-server*, kterou lze vidět na obrázku 4.1. V případě webové aplikace je klientem prohlížeč který zobrazuje uživatelské rozhraní aplikace a provádí logiku aplikace. Pro přenos dat mezi klientem a serverem se používá protokol `http` případně jeho šifrovaná verze `https`. [12]

Klient komunikuje se serverovou částí pomocí aplikační rozhraní serveru. Aplikační rozhraní je bývá většinou založeno na architektuře Representational state transfer (REST). Všechny zdroje mají unikátní URI identifikátory a REST poskytují čtyři základní metody pro přístup k těmto zdrojům (GET, POST, PUT, DELETE). [12]

Kód na straně serveru může být napsané v libovolném jazyce, ale za podmínky, že serverová část bude schopna komunikovat s klientskou částí přes předem definované aplikační rozhraní. Klientská část, co se týče v rámci použitých jazyků a jejich verzí je značně omezena prohlížečem. Většina prohlížečů je schopna minimálně interpretovat jazyky HTML (HyperText Markup Language), CSS (Cascade Style Sheets) a JavaScript. [37]

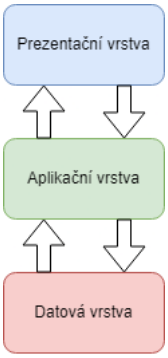
Obrázek 4.1: Architektura klient server. Vytvořeno na základě [12]

4.2 Třívrstvá architektura

Skládá se ze tří vrstev, které jsou hiearchisticky uspořádány, jak lze vidět na obrázku 4.2:

- **Prezentační vrstva** - Stará se o vykreslení uživatelského rozhraní.
- **Aplikační vrstva** - Jejím smyslem je starat se o byznys logiku aplikace.
- **Datová vrstva** - Řeší uchování dat aplikace.

Prezentační vrstva, je vrstva, která se zobrazuje uživatelovi. Uživatel do ní zadává vstupy. Získané vstupy jsou předány aplikační vrstvě, která je prostředníkem mezi vizualizací a ukládání dat. Aplikační vrstva v závislosti na kontextu data zpracuje a předá je datové vrstvě, která informace uloží určitou formou (databáze. soubor). [19]



Obrázek 4.2: Třívrstvá architektura. Vytvořeno na základě [19]

4.3 Druhy webových aplikací

Co se týče vývoje webových aplikací na straně klienta můžeme rozdělit do tří skupin.

4.3.1 Statický přístup

Historický první a nejzastaralejší přístup je statický. V tomhle případě se spíš jedná o webové stránky než aplikace. Webové stránky mají pevný neměnný vzhled a žádný nebo minimálně personalizovaný obsah. Pro většinu interakcí s webem, která běží na straně klienta, uživatel změní URI, kliknutím na odkaz, případně ji zadat manuálně do vyhledávacího řádku. [21]

Komunikace je jednostranná. Zobrazují se informace, které chtěl vývojář předat čtenářům. Ke tvorbě tohoto druhu stránky není zapotřebí velká znalost kódování. Udržování velkého množství statických stránek je nepraktické, neproduktivní a opakující. [21]

4.3.2 Dynamický přístup

U dynamického přístupu se využívá logiky na straně serveru, která zpracovává požadavky a na základě požadavků generuje dynamicky stránky HTML. Následně stránky s dynamickým obsahem zasílá prohlížeči uživatele. Web se znovu načte a prohlížeč zobrazí přijatou HTML stránku. Obsah HTML stránky se tedy mění v závislosti na čase, uživateli, uživatelské interakci. [21]

Tento přístup má taky své nedostatky. U vysoce interaktivních stránek, kde se krátkém časové sledu mění informace je potřeba pořád zasílat požadavky s vygenerováním nového HTML souboru na server. [21]

4.3.3 Jednostránkový přístup

Jednostránkové aplikace, anglicky *Single Page Aplication*, spoléhá na použití JavaScriptu na straně klienta, její veškerý obsah je tvořený jedinou stránkou. Důsledkem je rychlejší a příjemnější manipulace se stránkou. Většina dat jsou stažena při načtení aplikace a kontext stránky při interakci je bez čekání dynamicky měněn.[21]

Je-li potřeba komunikovat asynchronně se serverem, načítají, případně odesílají data prostřednictvím technik, jako je AJAX, které se slouží k interakci se serverem bez nutnosti znovu načítat stránku. Nevýhodou je, horší SEO (Search Engine Optimization, česky Optimalizace pro vyhledávače), což je soubor technik či praktik, dopomáhajících ke zlepšení pozice webových stránek nebo aplikací ve výsledcích vyhledávání [24], ale i to se dá řešit pomocí serverového vykreslení aplikace, čímž se aplikace připravuje o část rychlosti. [21]

Tenhle přístup se hodí pro stránky s vysokou mírou interakce jako jsou informační systémy, on-line burzy, sociální sítě [21]. Jelikož je vývoj takových aplikací pouze v základní verzi JavaScriptu výrazně složitý, tak se pro zjednodušení vývoje jednostránkových aplikací používají JavaScriptové aplikační rámce a knihovny, mezi nejpoužívanější aplikační rámce patří Angular, Vue a knihovna React.

4.4 Použité technologie

V této části se práce zabývá technologiemi pro vývoj webových aplikací, které budou následně využity pro vývoj uživatelského rozhraní pro nástroj Privchecker. Při jejím výběru byl brán ohled na stálost, rozšiřitelnost i modernost. Nejprve jsou představeny HTML a CSS pro vytvoření statického webu a pak jsou probrány dynamické technologie, jejímž zástupce je JavaScript a TypeScript.

Je zmíněna hlavní serverová technologie pomocí, které se extrahují data o soukromí z sociálních sítí. Závěrem jsou probrány podpůrné nástroje pro Javascript a nejpoužívanější aplikační rámce využívající jazyk JavaScript, jejich srovnání a výběr pro nástroj Privchecker.

4.4.1 HyperText Markup Language

HyperText Markup Language (HTML) je značkovací jazyk, který je základním stavebním kamenem pro vytváření webových aplikací a definuje význam a strukturu webového obsahu. Pro rozlišení jednotlivých druhů prvků v HTML dokumentu se používají *značky* (angl. tagy). Značky mají určitý význam, některé slouží pro vstupní pole, jiné zas jenom zobrazují text. Značky se dělí na párové a nepárové. [14]

Slovo HyperText ve zkratce značí odkazy, které jsou základním aspektem webu a vzájemně propojují webové aplikace, to buď v rámci jednoho webu, nebo mezi weby. Poslední verzí HTML je verze 5.3. [14]

4.4.2 Kaskádové styly

Kaskádové styly (CSS) jsou mechanizmus pro popis vzhledu jednotlivých prvků dokumentu napsaného v jazyce HTML nebo XML. Pomocí kaskádových stylů se může nastavit barva, velikost, písma, odsazení a další. Působnost stylů se dá omezit na jednotlivé značky případně třídy nebo identifikátory. Další funkcionalitou kaskádových stylů je vytváření animací. [13]

V dnešní době existuje velká řada aplikačních rámců, které ulehčují práci se stylováním. Lze je rozdělit na 2 druhy: Jedny jsou založeny na UI-komponentách (tzv. Component-based frameworks) [32] a ty, které jsou založené na psaní stylů pomocí určitých tříd (tzv. Utility-first frameworks) [34].

Při použití aplikačních rámců založené na komponentách se používají před připravené zabalené kusy kódu (komponenty), kterým je nastaven určitý styl. V případě aplikačních rámců založené na třídách je obdržen set před připravených tříd, které si sám programátor aplikuje na jim vytvořené kusy kódu.

Třídní aplikační rámce vývojář volí tehdy, kdy potřebuji při tvoření vzhledu aplikace větší volnost [34]. Jedním z nejpopulárnějších zástupců komponentního stylu je aplikační rámec Ant[1], u třídního rámců jeden z nejpoužívanějších je aplikační rámec TailwindCSS[2].

V dnešní době lze na trhu najít i CSS preprocesory (SASS, LESS), které pomocí vlastní syntaxe, definované preprocesorem vygeneruje CSS. Řeší praktické nedostatky CSS. Jedním z těchto nedostatků je tvoření mixinů, funkcí, smyček, proměnných. [30]

4.4.3 JavaScript

Javascript je skriptovací, interpretovaný dynamický, objektově orientovaný jazyk bez typové kontroly. Vznikl v roce 1995 a od roku 1997 je normován organizací European Computer Manufacturer Association (ECMA). Aktuální verze ECMAScritpu (JavaScript je obchodní název) je ES12 vydaná v červnu 2021. [10]

JavaScript převážně slouží vytváření dynamicky aktualizovaného obsahu, ovládání multimédií, animování obrázků v HTML dokumentu, tedy klientské části webových aplikací, ale v průběhu času a vývoje se JavaScript v podobě *Node.js* začal objevovat i na serverové části webových aplikací. Kód v JavaScriptu lze zneužít, protože se nachází v prohlížeči v otevřené formě. Další nevýhodou JavaScriptu je, že každý prohlížeč používá jiný JavaScriptový engine (motor). Google Chrome používá V8, Safari JavaScript Core, Mozilla SpiderMoneky. S použití rozdílných motorů se může stát, že prohlížeč nebude podporovat novější funkce či syntaxe. [10] [43]

4.4.4 TypeScript

Typescript vychází z JavaScriptu částečně řeší nedostatky JavaScriptu a přidává třídy, rozhraní, modifikátory přístupu a statické typování, díky kterému se programátoři můžou vyhnout nepříjemných chybám, se kterými se setkávají při psaní aplikace v JavaScriptu. TypeScript používá tzv. *transpiler*, překladač, který přeloží program napsaný v TypeScriptu

[1] https://ant.design/
[2] https://tailwindcss.com/

do ekvivalentního kódu v JavaScriptu. Díky TypeScriptu lze psát přehlednější kód, který lze snáze ladit. [25]

4.4.5 Selenium WebDriver

Selenium WebDriver populární aplikační rámec podporující velké množství jazyků (Python, JavaScript, Java). Selenium WebDriver se používá pro automatizované testovací skripty vykonávající funkční testování webových aplikací. Selenium WebDrvier má funkce pro přístup k elementům na webové stránce, postavené nad rozhraním webdriver. Webdriver poskytuje sadu procedur pro zjišťování a manipulaci s prvky DOM stromu v dokumentech a také pro řízení chování uživatelského agenta. [7] [16]

Selenium WebDriver vytváří jednotné rozhraní (ovladač) pomocí kterého přistupuje k prohlížeči. Mezi podporované prohlížeče patří Google Chrome, Firefox, Opera a Internet Explorer. Díky této vlastnosti tvůrce skriptů může modelovat jak uživatel interaguje s prohlížečem. [7]

Selenium WebDriver může spustit prohlížeč s viditelným oknem nebo může použít tzv. *headless* mód, který nepotřebuje ke své práci viditelné okno. Výhodou headless módu je, že uživatel nemůže narušit přímo provádění stahování informací z webových stránek. Nevýhodou je, že většina masově využívaných stránek má detekci na headless mód. [7]

Základními funkcemi, které se nejčastěji používají jsou:

- **get** – Přesměruje ovladač na zadanou webovou stránku.
- **until** – Funkce čeká dokud nenastane nějaká událost prvku udaného selektorem.
- **find_element/find_elements** – Funkce nalezne prvek / prvky obsažené na stránce pomocí zadáného selektoru. Selektor může být třída, identifikátor, XPath, CSS selektor.

4.5 Aplikační rámce

Zvyšujícím počtem požadavků na funkčnost webových aplikacích se dnes vývojáři setkávají s nespočetnou řadou aplikačních rámců a knihoven, které zvyšují funkcionalitu aplikace a také usnadňují a zrychlují aplikační vývoj. Mezi zástupce nejpoužívanější frameworků (aplikační rámců), případně knihoven lze zařadit frameworky Angular[3], Vue[4] a knihovna React[5].

4.5.1 Aplikační rámec versus knihovna

Knihovnu lze chápat jako soubor funkci, které může programátor ve svém kódu využít. Požadavky pro knihovny jsou aplikační nezávislost a schopná koexistence s ostatními knihovnami. Knihovna umožňuje volný přístup k funkcím, používají se jen nezbytně nutné. Aplikační rámec se dá chápat jako plnohodnotná sada nástrojů, která umožňuje vytvářet webovou aplikaci. Od knihovny se liší tím, že udává šablonu, která říká jakým způsobem se má aplikace vyvíjet. Výhodou aplikačního rámce je zlepšení organizace a struktury aplikace. [28]

[3] https://angular.io/
[4] https://vuejs.org/
[5] https://reactjs.org/

4.5.2 Angular

Angular je JavaScriptový aplikační rámec vyvíjený firmou Google v roce 2010 nejprve pod názvem AngularJS a následně zmodernizován jako Angular. I přesto, že není využíván firmou Google pro jejich projekty může se Angular pyšnit velkým zastoupením ve světě webových [17]. Angular je plnohodnotný aplikační rámec, což znamená, že obsahuje veškeré technologie potřebné pro sestavení plně funkční aplikace. Projekty v Angularu jsou strukturovány do tří hlavních prvků (Moduly, Komponenty a Služby) [35].

Základní prvky, komponenty definují a spravují pohledy, obsahují tedy šablonu v jazyce HTML s statickými či dynamickými prvky Angularu plnící prezentaci komponenty, součástí komponenty je také třída, která je oddělena od šablony a slouží jako její aplikační logika [35] [4]. Komponenty využívají jednotlivé služby (nepřímo souvisí s pohledy), poskytující konkrétní funkce, jejímž primárním cílem je zpracování dat [35] [4]. Komponenty se pak skládají do jednotlivých modulů [3]. Moduly shromažďují související kód do celků plnících určitou funkci [3]. Seskupením těchto modulů je pak definovaná celá aplikace [3]. Každá aplikace obsahuje kořenovou komponentu a modul [35].

Aplikační rámec Angular podporuje obousměrnou vazbu dat, při které komunikuje aplikační logika se šablonou a naopak [6]. Angular je komplexnější než React a Vue, odráží se to i na jeho velikosti. U Angularu lze také pozorovat pomalejší načítání oproti ostatním, příčinou je již zmíněná velikost a odlišný přístup s DOM. Angular se používá převážně u velkých projektů a také díky své větší komplexnosti má delší učící křivku oproti Reactu a Vue. Dvakrát ročně je vydána nová verze, ve kterých se může objevit označení, že některé metody jsou „zastaralé“ a budou v budoucnu zrušeny, důsledkem toho musí vývojáři předělávat značné části svých projektů. [17] [35]

4.5.3 React

React je JavasScriptová knihovna vzniklá v roce 2013 a vyvíjená firmou Meta, jenž ji využívá pro své aplikace Whatsapp, Instagram a Facebook. React se zaměřuje na prezentační vrstvu aplikace [17]. Základním prvkem jsou komponenty, což je třída nebo funkce. Komponenty jsou součástí prezentační vrstvy aplikace. Součástí komponent je syntaxe JSX (JavaScript XML), která zjednodušuje zápis HTML prvků v JavaScriptu a její propojení s aplikační logikou. Knihovna React využívá tzv. háky, které usnadňují práci se stavem komponenty a ostatními věcmi bez toho aniž by byla potřeba vytvořena další speciální třída [29]. Změní-li se stav komponenty, vykreslená znovu je pouze změněná komponenta. [35] [6] [17]

React má oproti Angularu pouze jednosměrnou vazbu dat mezi nadřazenou komponentou a podřazenou komponentou ve stromu DOM [6]. React obsahuje velké množství podpůrných knihoven umožňujících snazší vývoj aplikace [35] [6] [17]. V základní verzi React neobsahuje funkcionalitu např. pro navigování mezi stránkami nebo pro získávaní dat ze vzdáleného serveru. React současně s Angularem podporuje virtualizace a je zde možnost serverového renderingu [35].

Meta se snaží svoji knihovnu udržovat maximálně zpětně kompatibilní. Ve výjimečných případech, kdy už je potřeba starou věc zrušit existují nástroje pro migraci na novější verzi, přičemž funkcionalita projektu je zachována. Velikostně a rychlostně se React nachází mezi Angularem a Vue. Učící křivka Reactu je jednodušší než v případě Angularu. [17]

4.5.4 Vue

Vue je aplikační rámec řízený komunitou, který byl vytvořen Evanem You [17]. Základní část Vue se zaměřuje na zobrazování pohledů a strukturu aplikace tvořenou z komponent vlastnící instanci Vue. Komponenty obsahují šablonu z HTML, které jsou propojeny se JavaScriptovou části komponenty. Skriptová část komponenty obsahuje data, metody, pracují s jednotlivými elementy šablony. Dojde-li, ke změně obsahu komponenty, je stejně jako v případě Reactu vykreslena pouze změněná komponenta. Data komponenty jsou uloženy ve specializované funkci `data`, která vrací data jako anonymní funkce a jednotlivé komponenty jsou uspořádány ve stromové struktuře a předávají svým potomkům data pomocí objektu `props` [42]. [35] [6]

Veškeré šablony, aplikační logika a stylování komponenty je doporučené ukládat v rámci jednoho souboru tzv. SFCs (Single File Components) [35]. Vue stejně jako React, neobsahuje funkcionalitu pro navigaci, případně serverové vykreslení a je potřeba tuhle funkcionalitu doplnit v podobě knihoven třetí strany. [35]

Vue se může chlubit nejmenší velikostí a rychlostí podobnou knihovně React. Učící křivka Vue je ze všech nejjednodušší, což může být také dvojsečná zbraň a při složitějších projektech může docházet k nepřehlednému kódu [17]. Jelikož se jedná o poměrně nový projekt s rapidně rostoucími uživateli, lze očekávat v budoucnu částečné změny ohledně vlastností tohoto aplikačního rámce. Další nevýhodou je chybějící podpora virtualizace, která má za účel zlepšit výkonost aplikace [35]. [6] [17]

4.6 Responzivní design

Jelikož existuje mnoho různých velikostí obrazovek a poměrů stran je potřeba se zamyslet nad problematikou vzhledu webových aplikací. Vznikl koncept responzivního webového designu (RWD), soubor postupů, technik HTML a CSS které umožňují webovým aplikacím měnit jejich rozvržení a vzhled tak, aby vyhovovaly různým šířkám obrazovky, rozlišení. Je to myšlenka, která změnila způsob, jakým navrhujeme web pro více zařízení. [11]

Historicky se používaly 2 druhy designu stránek. První druh je *Design s pevnou velikostí a pozicí prvků*. Prvkům se nastavuje přesná výška a šířka a odsazení v pixelech. v případě, že měl koncový uživatel menší obrazovku prvky stránky se mohly začít překrývat, hrozilo riziko vzniknu horizontálních posuvníků. V případě většího monitoru vzniká zbytečně volné místo. [11]

Druhým historickým zástupcem je *Výplňový design*. Jednotlivým prvkům se nastavuje procentuální velikost v rámci nadřazeného elementu a v případě malých obrazovek prvky s výplňovým designem můžou vypadat zmáčknutě, na velkém monitoru jsou prvky zase roztažené. Oba přístupy jsou velmi problematické, protože vzhled stránky vypadal pěkně pouze vývojáře. [11]

Se vznikem mobilních zařízení, které uměly přistupovat na internet a webové stránky, vzniká také nátlak na společnosti. Společnosti, které chtěly podporovat mobilní telefony musely vytvořit zvláštní speciální mobilní verzi svých stránek s jinou adresou URI (často něco jako m.example.com). To znamenalo, že bylo nutné vyvinout a udržovat dvě samostatné verze webu. Mobilní verze často nabízely omezenou funkcionalitu a menší uživatelskou zkušenost. [11]

V roce 2010 vydává americký web designér Ethan Marcotte ve webovém magazínu A List Apart článek „Responsive web design“, ve kterém zavádí nový termín responzivního designu. Své myšlenky o responzivním designu rozdělil do tří částí: První technika se tý-

kala flexibilních mřížek pomocí CSS atributu `float` (plavat). Další trik, který zmiňuje se týká obrázků a nastavení vlastnosti `max-width` na 100 %. Poslední zmíněnou techniku je aplikování *media query* (dotaz média). [11]

4.6.1 Media query

Media query (dotaz media) umožňuje podmínit pravidla vlastnostmi zařízení [11]. V případě splnění podmínek se kaskádové styly obsažené v těle dotazu média aktivují. Na příkladu 4.1, lze vidět, že zobrazované médium je obrazovka a šířka výřezu obrazovky je menší nebo rovna 576 pixelům a nastavuje se velikost písma.

Běžný postup při používání media queries je vytvoření jednoho sloupce pro mobilní zařízení, poté vytváření více sloupců nad určitou velikostí obrazovky za podmínky, že se sloupce v pořádku vlezou. [11]

```
@media screen and (max-width: 576px) {
    p {
        font-size: 20px;
    }
}
```

Výpis 4.1: Použití media query

4.6.2 Flexibilní mřížka

Responzivní weby nepoužívají jenom media queries, které v určitých bodech změní velikost jednotlivých prvků. Aby vývojáři nemuseli cílit na veškeré velikosti obrazovek zařízení, které existují a vytvářet pro ně dokonalé rozvržení prvků, použijí flexibilní mřížky. [11]

Media queries a flexibilní mřížka jde spolu kombinovat, v případě, že flexibilní mřížka začíná vypadat nepěkně, lze ji od určitého bodu definovat pomocí media queries, změnit velikost. V počátcích responzivního designu se používalo pro rozvrženi v rámci flexibilní mřížky nastavení `float` pro jednotlivé prvky a flexibilita dosažena pomocí nastavení procentuální velikosti prvku v rámci, mřížky. Součet těchto prvků nesměl přesáhnout 100 %. [11]

Při výpočtu velikosti prvků $S_{\%}$ v rámci flexiblní mřížky, lze použít vzorec: [11]:

$$S_{\%} = \frac{velikost\ prvku}{velikost\ rodicovskeho\ prvku} * 100$$

.

4.6.3 Meta značka zobrazované oblasti

Značka meta na obrázku 4.2, říká, že šířka zobrazované oblasti mobilního webu se má nastavit na šířku obrazovky mobilního telefonu a změnit velikost dokumentu na 100 %, čímž se dokument zobrazí vývojářem zamýšlené velikosti optimalizované pro telefony. Důvod k použití meta značky je ten, že mobilní telefony mají tendenci lhát o své skutečné zobrazovací šířce. [11]

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

Výpis 4.2: Meta značka pro mobilní telefony

4.6.4 Flexbox

Flexbox je metoda jedno dimenzionálního rozložení prvků do řádku nebo do sloupce [9]. Prvky lze tak díky, flexboxu jednodušeji zarovnávat. Prvkům ve flexboxu, lze nastavit určitá míra pružnosti, nastavením vlastností `flex-grow` (zvětšení) a `flex-shrink` (smrštění). Hodnoty zmíněných vlastností udávají relativní velikost prvku vzhledem ostatním objektům ve flexboxu. V následujícím příkladu 4.3 lze vidět použití flexboxu, ve kterém jsou všechny prvky zarovnány doprostřed. [11]

```
main {
    display: flex;
    justify-content: center;
}
```

Výpis 4.3: Použití flexboxu, současně se zarovnáním předmětů na střed flexboxu

4.6.5 Grid

Grid (mřížka) dovoluje rozdělit prostor na určitý počet sloupců a řádků. Může k tomu použít jednotku `fr` (zlomek), jenž počet jednotek udává relativní velikost buňky vůči řádku, případně sloupci. Na příkladu lze vidět nastavení vlastnosti `grid-template-columm`, způsobující rozdělení na 2 řádky o velikosti v poměru 1:2. [11]

```
div {
    display: grid;
    grid-template-rows: 1fr 2fr;
}
```

Výpis 4.4: Použití gridu

4.6.6 Responzivita obrázků

V případě obrázků se nastavuje maximální šířka obrázku v procentech a v případě zmenšování nadřazeného kontejneru se zmenšuje i samotný obrázek. I tento přístup má určité nevýhody, jednou z nich je zobrazovaní příliš objemného a velkého obrázku, kterému je mnohonásobně zmenšena velikost. [11]

Nabízí se také možnost, že na mobilu aplikace bude chtít použít obrázek jiných poměrů stran nebo případně úplně jiný obrázek, kvůli šetření velikosti obrázku. Implementaci možnosti s více obrázky se pomocí značky `picture` v něj vnořené `source` značky společně s `img` značkou. [11]

4.6.7 Responzivita textu

Responzivní písmo se vytváří buď pomocí media queries v určitých kritických bodech šířky obrazovky nebo lze použít speciální jednotka `vh` týkající se viewportu (výřez okna v prohlížeči, kde se nachází webová aplikace). Kde x vw se rovná x % šířky viewportu. [11]

Kapitola 5

Implementace aplikace

V této kapitole se práce zabývá tvorbou uživatelského rozhraní pro nástroj Privchecker. Rozebrány jsou požadavky nástroje, poté je okomentován samotný návrh rozhraní. Dále se řešilo přihlašování, odhlašování ze sociální sítě. Je zde rozebrána problematika sdílení uživatelských údajů. Na závěr kapitoly je probrána klientská část aplikace Privchecker včetně její responzivnosti.

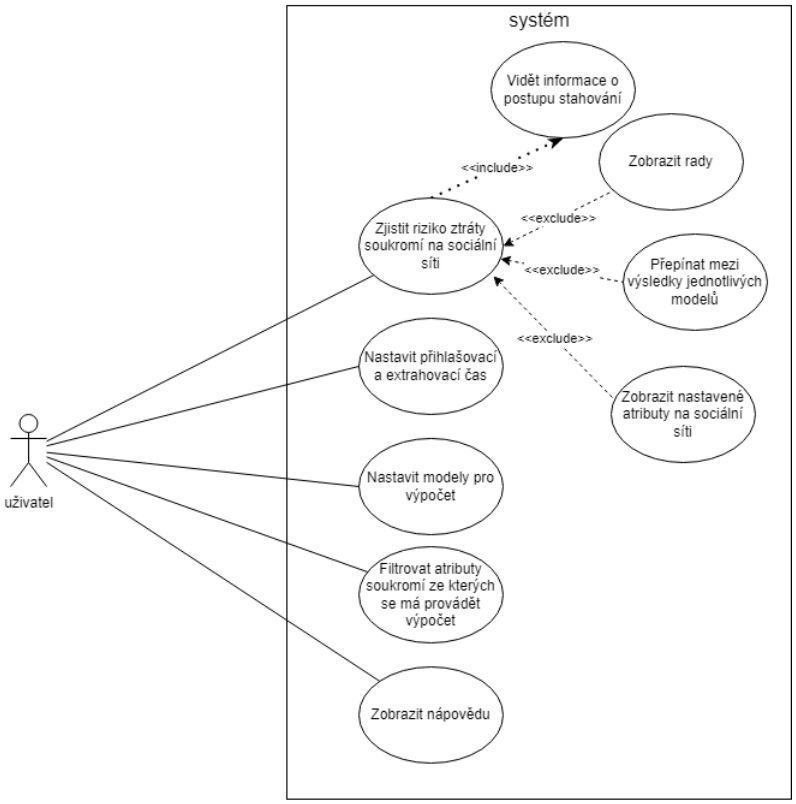
5.1 Požadavky aplikace

- Výběr sociální sítích, ze kterých se vypočítá privátní skóre.
- Zadání uživatelského jména a hesla bez pochybnosti o zneužití.
- Spuštění skriptu se zadanými údaji.
- Vizualizace výsledků a porovnání s ostatními modely.
- Zobrazení nápovědy a podnětů k zlepšení privátního skóre.

5.2 Diagram případu užití

Jak lze vidět na diagramu případů užití 5.1, systém bude obsahovat jeden druh uživatele. Nejdůležitějším případem užití je případ „Zjstit riziko ztráty soukromí na sociální síti“, díky němu uživatel zjistí, zdali se má obávat o své soukromí.

Případ užití „Nastavit modely pro výpočet“ se týká nastavení jednotlivých výpočetních modelů nad, kterými se má vypočítat privátní skóre. Důležitým případem užití je také případ užití „Nastavit přihlašovací a extrahovací čas“, který dává uživateli možnost jak rychle se mají stahovat data ze sociálních sítí a jak dlouho má serverová část čekat, než se uživatel přihlásí na danou sociální sít. Přihlašovací a extrahovací čas je probrán podrobněji v sekci 5.9.



Obrázek 5.1: Diagram případu užití

5.3 Návrh architektury

Před samotnou implementací bylo potřeba promyslet jakým způsobem bude propojeno grafické rozhraní aplikace s byznys logikou aplikace, která stahuje data ze sociálních sítí a vypočítává *privátní skóre*. Při návrhu architektury byl řešen rozpor mezi použitelností aplikace a bezpečností uživatele. Při návrhu byly brány v potaz tyto možnosti:

- **Možnost 1:** Umístění původního skriptu programu Privchecker a webového grafického rozhraní na vzdálené servery.

  Problematika: Uživatel by musel sdílet své přístupové údaje k sociálním sítím nebo by musel sdílet své uložené *cookies*[1]. V případě sdílení přístupových údajů je to bráno jako velké bezpečnostní riziko na úkor použitelnosti.

  Uživatel zadá jenom své přístupové údaje do webové aplikace a získá výsledek. Nemusí instalovat různé programovací jazyky, knihovny. V případě cookies by musel uživatel nalézt ve svém počítači soubor, ve kterém má jeho prohlížeč uložené cookies, zabalit pomocí archivačního softwaru a nahrát na server. Na severu rozbalit a načíst je. V obou případech by mohl útočník v síti zachytit jak cookies nebo přístupové údaje a zneužít je pro svůj prospěch.

  Dalším problémem je nutnost použití *headless* módu speciálního prohlížeče ovládaného instancí ovladače z knihovny Selenium, protože většina serverů nemá, kam vy-

[1]Cookies (Sušenky) - malý objem dat, uložený na straně klienta [15]

kreslit okno prohlížeče. Jak bylo také naznačeno v sekci 4.4.5, většina webových aplikací, včetně sociálních sítí detekují *headless* mód. Na základě detekce by mohlo dojít pozastavení účtu uživatele.

- **Možnost 2:** Přepsání původního skriptu Privchecker, který je napsaný v jazyce Python do JavaScriptu a ten zakomponovat jako součást klientské části obsahující grafické rozhraní a samotnou aplikaci spouštět na uživatelském počítači v prohlížeči, který dokáže spustit JavaScriptový kód.

  Problematika: V této možnosti by se zvýšila bezpečnost, protože data už nejsou odesílána přes síť a veškeré předávání přístupových údajů by se odehrávalo na počítači uživatele. Uživateli by stačilo jenom stažení aplikace a spuštění aplikace pomocí otevření souboru `index.html`. Problémem tohoto přístupu je, že ovladač prohlížeče z knihovny Selenium bez explicitního nastavení otevře novou instanci prohlížeče bez žádné historie, případně cookies.

  Při explicitním nastavení lze nastavit profil prohlížeče, ve kterém je uložena historie a cookies, které drží přihlášený stav na sociální síti. Zde je problém takový, že Google Chrome nedovoluje otevření stejného profilu ve dvou instancích prohlížeče. Největší problém, ale je v tom, že aplikační rámec Selenium, který nabízí možnost implementace v JavaScriptu využívá funkce, které jdou pouze spustit v Node.Js, která je serverovou implementací JavaScriptu a tudíž v prohlížeči jsou tyhle funkce nedostupné.

- **Možnost 3:** Vytvoření Docker Image (obraz Docker disku), ve kterém bude zabalena klientská část obsahující grafické rozhraní a také serverová část, která bude provádět stahování dat o soukromí a výpočet nad nimi.

  Problematika: Pro spuštění Docker Image je potřeba mít nainstalovanou aplikaci Docker. Instalace softwaru Docker, stažení a vložení disku do Dockeru není pro běžného uživatele jednoduchá. I přesto, že při vložení Docker Image není potřeba instalovat dodatečné programovací jazyky a knihovny, je tím stejně značně snížena použitelnost, ale na druhou stranu je zvýšena bezpečnost, protože by se přihlašovací údaje nebo cookies předávaly pouze na uživatelském stroji. Nevýhodou je opět je nutnost spouštění ovladače prohlížeče v headless, módu, protože v rámci Dockeru nelze napojit viditelné okno prohlížeče.

- **Možnost 4:** Vytvoření serveru na počítači uživatele a napojení grafického uživatelského rozhrání.

  Problematika: V tomhle případě by musel uživatel nainstalovat veškeré knihovny a programovací jazyk Python. Tahle možnost by se dala ulehčit vytvořením skriptu, který bude po spuštění instalovat potřebné knihovny, ale i přesto by musel uživatel nainstalovat jazyk Python. Dále je možnost využití viditelného okna prohlížeče, které je otevřeno ovladačem prohlížeče z aplikačního rámce Selenium.

  S tím přichází idea, že by šlo využít okna prohlížeče pro zadání přístupových údajů a uživatel by nemusel zadávat cestu k profilu prohlížeče nebo zadávat do grafického rozhraní přístupové údaje k sociálním sítím. Další nevýhodou je, že uživatel může narušit průběh stahování vypnutím okna prohlížeče. Instalací knihoven a možnost vypnutí okna je snížena použitelnost, ale veškerá funkcionalita aplikace je prováděna na počítači uživatele.

Jako nejlepší řešení se jeví Možnost 4, která byla následně implementována, V této možnosti je vhodný poměř bezpečnosti a použitelnosti. Samotné přihlašování na sociální sítě v této možnosti je rozebrána v následující sekci.

5.4 Uživatelské údaje

Ze začátku bylo potřeba vyřešit přihlašování do jednotlivých sociálních sítí. Původní skript po spuštění vyzval uživatele pro vložení sociálních sítí, na kterých se má výpočet uskutečnit a po zadání sociálních sítí, začne původní skript vyžadovat přihlašovací jméno a heslo. Uživatel, který je obeznámen určitými pravidly bezpečného chování internetu, ví že sdílení přístupových údajů jeho účtu aplikaci třetí strany je velmi podezřelá záležitost.

Některé webové aplikace poskytují přihlašování pomocí služby třetí strany OAuth. OAuth poskytují i některé sociální sítě. Aplikace, která chce využívat zpřístupnění uživatelových údajů je u služby registrována [20]. Příchozí uživatel po stisknutí tlačítka je přesměrován na stránku služby a autorizuje aplikaci a je zpátky vrácen na aplikaci, které služba předala autorizační klíč [20]. Autorizační klíč je poté poslán aplikační rozhraní služby a získá přístupový žeton [20]. Aplikace, kterou chce uživatel používat tak získala informace bez toho, aby uživatel předal aplikaci uživatelské jméno a heslo. Využití přístupu OAuth je pro aplikaci Privchecker nedostačující.

Aplikace využívá stahování interního nastavení soukromí na sociální síti a informace o nastaveném soukromí bohužel žádné aplikační rozhraní sociální sítě neposkytuje. Můžou se nabízet 2 možnosti proč tomu tak není. Zřejmě nikdo nepočítal s tím, že s informacemi bude chtít někdo pracovat. Z logického úsudku lze usoudit, že je to, protože sociální sítě chtějí zachovat integritu soukromí uživatele a nechtějí sdílet jeho nastavení. Téhle možnosti nahrávají také nedávné skandály a úniky.

Je potřeba dostat se k účtu uživatele bez toho aniž by musel uživatel zadat přihlašovací údaje do grafického uživatelského rozhraní aplikace třetí strany. Nabízí se řešení, jak již bylo naznačeno v sekci 5.3, že ovladač prohlížeče z aplikačního rámce Seleninum bude používat uživatelský profil v prohlížeči Google Chrome, do kterého se ukládají veškeré cookies a bude si pamatovat při každém přihlášení veškerou historií a držené přihlášené stavy na veškerých sociálních sítích.

Uživatelský profil je potřeba explicitně nastavit pomocí funkce `add_argument(user-data-dir=cesta)`. Funkce slouží pro určení cesty k rodičovské složce, ve které je uložena složka uživatelského profilu. Tenhle argument pro Google Chrome počítá s tím, že profilová složka se jmenuje Default. Aplikační rozhraní Google Chromu nabízí i funkci u které se dá nastavit jiné jméno složky.

Tento přístup sebou nese určitá úskalí. Uživatel musí mít nainstalovaný prohlížeč Google Chrome a uložení samotného profilu nabírá na objemu zabraného místa aplikace. Vlastnost profilu v prohlížeči Google Chrome lze využít pro účely aplikace Privchecker. Aplikace předtím než začne stahovat informace o nastaveném soukromí na sociální síti ověří, zdali je uživatel přihlášen. Pokud uživatel není přihlášen, aplikace a okno prohlížeče, které je otevřeno ovladačem z aplikačního rámce Selenium vyzve uživatele k přihlášení přímo na webovou stránku sociální sítě v prohlížeči.

Tím se obešlo zadávání uživatelských přihlašovacích údajů přímo do grafického uživatelského rozhraní aplikace Privchecker. Uživatelovi bude aplikace důvěryhodnější, protože se zvýšil faktor důvěryhodnosti uživatelské zkušenosti, který byl zmíněn v sekci 3.6.1.

5.5 Přihlašování

Protože pro správné vykonávání stahování dat nastavení soukromí ze sociální sítě je potřeba přihlášeného uživatele, bylo zapotřebí řešit jak poznat, že je uživatel přihlášen. Aplikační rámec Selenium nabízí funkci `until` instance třídy `WebDriverWait`, pomocí, které ovladač prohlížeče Google Chrome čeká až nastane určitá událost u prvku, který může být zadán různými možnostmi (XPath, třída, identifikátor, CSS selektor).

V rámci přihlašování u funkce `until` se čeká až nastane událost zobrazení prvku zadanou cestou. Pro každou sociální sít byl zkoumám HTML dokument na stránce s přihlašováním a na stránce, na kterou je uživatel přesměrován. Byl hledán **rozdílový** prvek, který se nachází na stránce, kterou je přesměrováno po správném přihlášení. Aplikace, v případě, že uživatel není přihlášen čeká určitý čas (ve výchozím stavu je v aplikaci nastaveno 120 sekund).

Aby uživatel věděl proč aplikace neextrahuje data ze sociální sítě je upozorněn v grafickém uživatelském rozhraní aplikace, že je potřeba se přihlásit do prohlížeče ovládaného aplikací. Vyprší-li čekací lhůta určená pro přihlašování, aplikace nevykonává žádné funkce pro extrahováni dat o soukromí a vyhodnocuje neúspěch. V případě úspěšného ověřeného přihlášení je přerušeno čekání a začíná se vykonávat stahování dat.

Tenhle generický přistup přihlašování bylo potřeba vylepšit u některých sociální sítích. V rámci sociální sítě Instagram bylo potřeba vylepšení v podobě automatického stisknutí tlačítka, jehož funkcí je zjistit, jestli uživatel chce uložit přihlašovací údaje. Bez stisknutí tohoto tlačítka není uživatelovi na Instagramu dokončen přihlašovací proces. Řešení této záležitosti je najití prvku pomocí ovladače prohlížeče a na objekt třídy `WebDriverElement`, který zapouzdří prvek tlačítka aplikovat metodu `click`, která vykoná stisknutí tlačítka.

U Googlu je problém, že Google má software, jenž dokáže detekovat, zdali prohlížeč je využíván automatizovaným softwarem. Dříve se využívalo obejití Googlu pomocí toho, že se uživatel přihlásil pomocí Google účtu do aplikace třetí strany a tím pádem se musel přihlásit i na Googlu. Jenže při přihlašování do aplikace třetí strany nebyla provedena žádná detekce automatizovaného softwaru. Automatizovaný software mohl odeslat přihlašovací údaje a následně po úspěšném procesu přihlášení přesměrovat na nastavení Google účtu, kde už byl přihlášen.

Tenhle trik využíval také i původní skript Privchecker. Bohužel si této mezery povšimli vývojáři z Googlu a chybu opravili. Proto bylo potřeba ovladač prohlížeče upravit pomocí rozšířené knihovny pro Selenium s názvem selenium-stealth.

Rozšíření pomocí funkce stealth, které je předán ovladač prohlížeče a nastavení uživatelského agenta, nastaví prohlížeč tak, aby se tvářil jako prohlížeč, který používá skutečný uživatel.

Bohužel tohle rozšíření funguje jenom pokud se uživatel přihlašuje na přímo do Google účtu. V případě, že uživatel chce využít přihlášení do sociální sítě např. Pinterest pomocí účtu Google a je přesměrován na stránku, kde má potvrdit poskytnutí informací, setká se s hláškou o nemožnosti přihlášení, protože byl detekován automatizovaný software.

5.6 Odhlášení

Jestliže uživatel již dále nechce být přihlášený na sociální síti ve speciálním prohlížeči z důvodů, že má více účtů na sociální síti nebo jenom nechce držet přihlášený stav ve speciálním prohlížeči, aplikace poskytuje možnost se odhlásit na sociální síti. Většina sociálních sítí má speciální adresu končící /logout, která uživatele odhlásí. Jediné, co je potřeba udělat pro

odhlášení, je přesměrování ovladače prohlížeče na speciální adresu a počkat, než se stránka načte.

U sociální sítě Facebook a Instagram je problém, že tyhle sociální sítě neposkytují tuhle adresu a je nutné simulovat uživatelské kroky. Nejprve je na hlavní stránce nalezen prvek rozbalovacího menu pomocí ovladače prohlížeče. Poté je aplikována funkce `click`, která nabídku rozbalí. Po rozbalení obsahu nabídky je potřeba ovladačem prohlížeče nalézt prvek v rozbalené nabídce, který opět po stisknutí funkcí `click` uživatele odhlásí ze sociální sítě.

5.7 Pomalé vypnutí prohlížeče

V případě úspěšného stažení veškerých informací o soukromí ze všech uživatelem zadaných sociální sítích se pomocí ovladače prohlížeče a jeho metody `close` vypne instance prohlížeče. Problém nastává tehdy, kdy veškeré výpočty týkající se modelů, jenž počítají privátní skóre sociálních sítích jsou dokončeny a musí se čekat na to, než se dokončí vypnutí instance prohlížeče a až po úspěšném dokončení se výsledek vrátí klientovi v klientské části aplikace.

K vyřešení tohoto problému se používá trik, ve kterém se pro vypnutí prohlížeče využije nově vytvořené vlákno, které asynchronně obslouží vypnutí prohlížeče a výsledek výpočtu privátního skóre se vrátí uživatelovi markantně rychleji.

5.8 Riziko omezení účtu uživatele na sociální síti

Web scraping není ze strany vývojářů sociálních sítí vítaná technika. Sociální sítě mají určité detekční techniky na odhalení automatizovaných skriptů. Při testování původního skriptu se prokázalo, že uživateli na sociální síti byla omezena na určitou dobu funkcionalita sítě a byl nucen pozměnit heslo.

Aby bylo zabráněno omezení účtu uživatele. Byla přidána do serverové části aplikace Privchecker knihovna undetected-chromedriver, která obchází detekční ochrany automatizovaného softwaru.

Další ochranou proti omezení účtu je použití knihovny selenium-stealth, která upravuje uživatelského agenta, tak aby simuloval reálného uživatele, který přistupuje na sociální sít. Další technika, která se používá v aplikaci Privchecker je použití funkce `sleep`, která uspí stahování dat na určitý čas zadaný parametrem. Doba čekání ve funkci `sleep` se dá změnit pomocí změny hodnoty proměnné, která uchovává *extrahovací čas*.

5.9 Extrahovací a přihlašovací čas

Jelikož se musí počítat s tím, že uživatelé, kteří budou používat aplikaci Privchecker můžou mít různě rychlý internet. Případně se speciální prohlížeč může neočekávaně zaseknout. Musí se brát do úvahy, jaká bude doba čekání na zobrazení prvku v HTML dokumentu, který reprezentuje hodnotu nastaveného atributu soukromí. Jak dlouho se bude čekat udává hodnota *extrahovacího* času. Zatímco *přihlašovací čas* udává maximální dobu čekání na přihlášení uživatele na sociální síť ve speciálním prohlížeči.

U přihlašovacího času bylo bráno v potaz, že uživatelé mají různou rychlost psaní a také možnost chybně zadaných přihlašovacích údajů, a proto byl přihlašovací čas nastaven ve výchozím stavu na 2 minuty. Extrahovací čas byl nastaven na 3 sekundy. Uživatel aplikace Privchecker může v klientské části aplikace v případě nevyhovujících hodnot, přenastavit

na hodnoty pro něj přijatelné. Podnětem pro změnu hodnot časů může být chybný výsledek výpočtu privátního skóre ze sociální sítě.

5.10 Vypnutí speciálního prohlížeče ze strany uživatele

Problémem při spuštění funkcionality na serverové části aplikace je, že se otevře okno speciálního prohlížeče. Bohužel aplikační rámec Selenium a jeho ovladač prohlížeče nedokáže zakázat manipulaci, ze strany uživatele. Takže uživatel může okno prohlížeče zavřít nebo může načíst jinou webovou stránku než, na které se má provádět *web scraping*. Aby se zabránilo chybnému vykonávání funkcionality, jsou veškeré funkce zabaleny do `try-except` bloků a zachytávají se výjimky `ElementNotFoundExcepiton`, `NoSuchWindowException` a `WebDriverException`.

`ElementNotFoundException` je výjimka, která se vyvolá při nenalezení prvku pomocí funkce `find_element` a `find_elements`. Výjimka `NoSuchWindowException` se vyvolá, pokud ovladač speciálního prohlížeče nemá okno prohlížeče a `WebDriverException` je obecná výjimka pro ovladač pro případ, že by nastala jiná chyba. Jestliže je vyvolána některá z těchto výjimek, je přerušena extrakce dat o soukromí ze sociální sítě.

5.11 Vytvoření aplikačního rozhraní

Jelikož je potřeba komunikace grafického uživatelského rozhraní se skriptem zastřešující stahování dat ze sociálních sítí a následný výpočet privátního skóre, bylo vytvořeno aplikační rozhraní.

K vytvoření REST (Representational state transfer) API byl použit aplikační rámec Flask a byly vytvořeny následující endpointy (koncové body), na které se dotazuje klientská část aplikace Privchecker. Veškerý použitý typ média, který se posílá na aplikační rozhraní je JSON.

5.11.1 Endpointy

- **/** – Zde se posílá POST požadavek, ve kterém se nachází pole `networks` s názvy všech sociální sítích, na kterých se mají extrahovat data o soukromí a provést výpočet privátního skóre. Dalším atributem je pole `models`, které udává, které modely se mají pro výpočet privátního skóre použít. Posledním atributem je pole `filter`, jehož obsahem jsou identifikátory atributů, jenž nebudou zahrnovány do výpočtu privátního skóre. Výsledkem dotazu je odpověď obsahující výsledek uživatelova privátního skóre, maximální a minimální skóre na zadaných sociálních sítích včetně rad pro zlepšení skóre.
- **/logout/<název sítě>** – Endpoint, který slouží k odhlášení účtu na sociální síti ve speciálním prohlížeči po zadání názvu sítě. Před odhlášením je ještě ověřeno, zdali je uživatel vůbec přihlášen, v případě, že uživatel přihlášený není, nebude se provádět proces odhlašování.
- **/login/<název sítě>** – Endpoint, který provede přihlašovací proces na zadané sociální síti. Nejprve ověří, jestli je uživatel přihlášen. Je-li uživatel přihlášen nepokračuje v procesu přihlašování. V opačném případě přesměruje uživatele na přihlašovací webovou stránku. Odešle uživateli na klientskou část zprávu o tom, že se má přihlásit a čeká, dokud se uživatel nepřihlásí. Výsledek přihlašování vrací jako odpověď.

- **/time** – Pokud se na „/time“ endpoint odesílá POST požadavek s obsahem `type`, který může nabývat hodnoty `parse` (extrahovací čas) a `login` (přihlašovací čas) a `value` obsahující dobu v sekundách. Pomocí tohoto endpointu se nastavuje čas čekání u přihlašování na sociální sít nebo u stahování dat.

  V případě GET požadavku jsou uživatelovi vráceny nastavené hodnoty extrahovacího a přihlašovacího času. Více o použití extrahovacího a přihlašovacího času pojednává sekce 5.9.

- **/attributes/<název sítě>** – Pokud se na endpoint „/attributes/<název sítě>“ odešle GET požadavek s názvem sítě, jako výsledek jsou vráceny veškeré atributy týkající se soukromí, jenž se stahují z požadavkem dané sociální sítě.

- **/edit/<název sítě>** – Endpoint „/edit/<název sítě>“ slouží pro otevření speciálního prohlížeče s načtením stránky na sociální síti, jenž se zabývá nastavování soukromí. Jméno této sítě je zadané GET požadavkem a odpovědí požadavku je výsledek otevření speciálního prohlížeče.

5.12 Stahování dat o soukromí ze sociálních sítí

Pro správnou funkcionalitu klientské části, která obsahuje grafické uživatelské rozhraní, bylo potřeba zprovoznit získávání uživatelova nastavení soukromí. Důvod, proč bylo nutné zprovoznit původní skript Privchecker je ten, že nebyl udržována web scraping mechanika pomocí, které se přistupovalo k jednotlivým hodnotám nastavení.

Vývojáři sociálních sítích totiž od roku 2018 pozměnili strukturu HTML dokumentu jednotlivých koncových bodů, na kterých se nachází nastavení o soukromí. Původní skript Privchecker obsahoval stahování dat o soukromí ze 7 sociálních sítí (Facebook, Twitter, Instagram, Google, LinkedIn, Pinterest a Tumblr). Z toho byly z části funkční sociální sítě Instagram a plně funkční síť LinkedIn.

U Facebooku byla v původním skriptu použita knihovna BeatifulSoup, která obsahuje třídu a při vytvoření její instance se do ní vložila stažená webová stránka. Problémem je knihovnou nedostatečně zvolená velikost struktury, která po vytvoření instance BeatifulSoup obsahuje webovou stránku. Jelikož webová stránka s nastavením soukromí na Facebooku má rozsáhlý HTML dokument, tak po uložení se obsah stránky ve struktuře přepisoval a bylo zde uloženo jenom spodní čtvrtina obsahu webové stránky, což mělo za následek to, že se nedalo ve struktuře vyhledat potřebné prvky. Řešením bylo nahrazení procházení webové stránky v BeatifulSoup aplikačním rámcem Seleniem. Další překážkou byla nutnost přepnutí do „okna“ prvku `iframe` (Značka `iframe` v HTML umožňuje vložit do vymezeného prostoru stránky obsah jiné stránky), ve kterém se nacházely hodnoty nastavení soukromí, protože vnořené značky v prvku `iframe`, aplikační rámec Selenium ignoroval.

Vývojáři Facebooku se pro použití prvku `iframe` uchýlili z toho důvodu, aby mohli použít starý vzhled a strukturu nastavení. Na stránce s koncovým bodem `settings?tab=privacy` nebylo potřeba kromě přepnutí do prvku `iframe` dále nic řešit a šlo využít původní volání funkce `find_elements` s nastavenými cestami k prvkům. Pro ostatní koncové body bylo potřeba vytvořit nové stahování dat s novými cestami k prvkům obsahující hodnoty nastavení. U nastavení sociální sítě Facebook šlo vidět, že každý koncový bod má jinou strukturu HTML dokumentu a jiný vzhled a s tím také souviselo nízké využití generičnosti při získávání jednotlivých prvků nastavení.

Twitter od poslední aktualizace původního skriptu Privchecker změnil vzhled a strukturu dokumentu HTML. Na rozdíl od sociální sítě Facebook vypadají koncové body Twitteru týkající se nastavení soukromí stejně, pouze s rozdílnými hodnotami popisů u nastavení jednotlivých hodnot soukromí. Díky tomu šla využít generičnost cest k jednotlivým prvkům nastavení soukromí na všech koncových bodech.

Pro sociální sítě Google, Tumblr a Pinterest bylo potřeba vložit nové cesty k jednotlivým hodnotám soukromí na webové stránce. V případě sociální sítě Instagram bylo nezbytné opravit stahování hodnot z přepínacích (radio) políček. Způsob ověření zdali je přepínací políčko zaškrtnuté se klasicky ověřuje pomocí existence atributu `checked` v prvku `input`, jenž reprezentuje přepínací políčko. Vývojáři z Instagramu se rozhodli jít jinou cestou a jediný způsob jak ověřit vybrané přepínací tlačítko je, získání kontejneru v podobě prvku `div`, jenž obsahuje přepínací tlačítko a zjistit hodnotu barvy. V případě zaškrtnutí políčka je barva kontejneru hodnoty `rgba(0, 149, 246, 1)` (modrá), v opačném případě nezaškrtnutého přepínače `rgb(219, 219, 219)` (šedá).

5.13 Problematika jednotlivých cest k získání hodnot

Problémem web scrapingu je, že vývojáři můžou pozměnit strukturu HTML dokumentu přidáním, odebráním značek, změnou použitých tříd a identifikátoru. Při výběru cesty k jednotlivým prvkům byl brán zřetel srozumitelné (v anglickém jazyce) názvy tříd a identifikátoru.

V případě vhodných názvů třídy nebo identifikátoru byly třída nebo identifikátor použity ve funkci `find_element` pro nalezení daného prvku. V případě nesrozumitelných názvů tříd nebo identifikátorů se zjišťovalo jestli jsou tyhle názvy při načtení stránky nebo při změně účtu generované. Pokud byly názvy generované, přešlo se ke krajnímu řešení, a to použití XPath daného prvku. U XPath se musí počítat s nevýhodou, že u XPath prvku je pravděpodobnost změny ze stran vývojářů mnohonásobně vyšší než u třídy a identifikátoru.

5.14 Použití socket.io na serverové části aplikace

Jelikož při GET/POST požadavku na aplikační rozhraní serverové části Privchecker nelze v průběhu posílat klientské části aktualizace o průběhu, uživatel tak není nijak informován, že je potřeba se přihlásit do speciálního prohlížeče a funkce `WebDriverWait` bude čekat celý přihlašovací čas, aby bylo pak uživateli oznámeno, že nebyla data o soukromí úspěšně stažena.

Aby se tenhle problém minimalizoval byla do aplikace implementována schránka z knihovny socket.io. Ze serveru pomocí socket.io jsou odesílány zprávy na klientskou část v průběhu operace stahování dat o soukromí. Jedná se o zprávy: „Přihlaste se na sociální síť <název sítě>“, „Začalo stahování ze sociální sítě <název sítě>“, „Stahovaní ze sociální sítě <název sítě> bylo dokončeno.“

Problém, který bylo potřeba řešit u socket.io, že v případě použití funkce `sleep` z systémové knihovny time a funkce `until` z knihovny Selenium serverová část odmítala reagovat na příchozí dotazy, zdali serverová část socket.io je živá. V případě funkce `sleep` ze systémové knihovny time, se funkce zaměnila za stejnojmennou funkci z knihovny socket.io. V situaci funkce `until` třídy WebDriverWait je to složitejší. Funkce `until` instance WebDriverWait využívá funkci `sleep` ze systémové knihovny time.

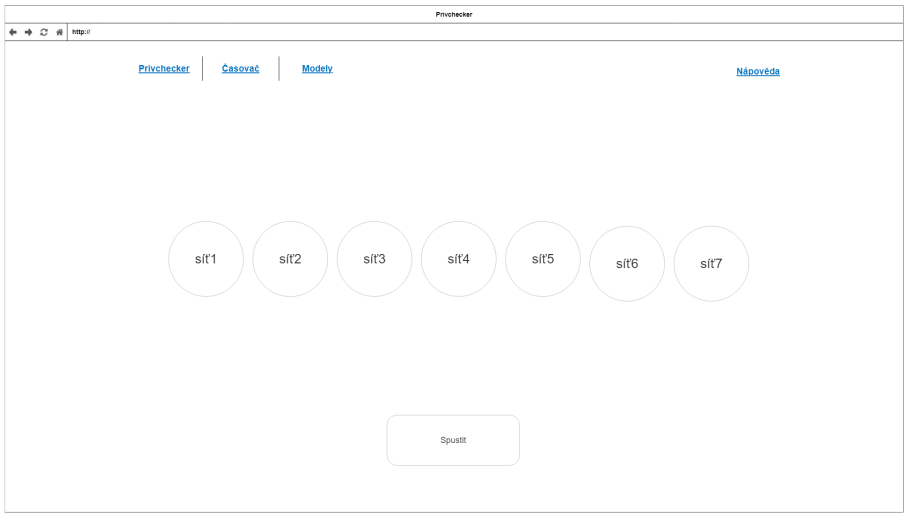
Je potřeba zvýšit interval, po který se bude klient socket.io pořád dotazovat na serverovou část socket.io, jestli je živá. Tím se docílí že v průběhu funkce `until` se klient socket.io neodpojí a nepokusí se vyvolání funkce `reconnect`. V průběhu funkce `reconnect` by mohla přijít zpráva od serverové části, která dokončila funkci `until`, ale klient socket.io by tuhle zprávu nezachytil, protože se zrovna pokouší obnovit spojení.

5.15 Klientská část Privcheckeru

Pro vytvoření klientské části aplikace Privchecker byl zvolena knihovna React se šablonou pro TypeScript. Knihovna React byla zvolena z důvodu stability a zpětné kompatibility a silné vývojářské základny. Zároveň React je vhodný pro středně velké projekty s jednodušší rozšiřitelností než aplikační rámec Vue.js a přijatelnější učící křivkou než Angular.

S Reactem byla použita také knihovna Axios, která zastřešuje a zjednodušuje asynchronní komunikace se serverovou částí aplikace, zasíláním http požadavků. Další podpůrnou knihovnou, která se v aplikaci nachází je knihovna socket-io-client, která slouží ke komunikaci na socketu, více o téhle komunikaci v kapitole 5.14. Důležitou knihovnou v aplikaci je knihovna Redux-React, která slouží pro ukládání stavů a dat a následný přístup z různých uzlů ve stromu React komponent. V projektu jsou obsaženy i méně významné knihovny FontAwesome a React-animated-css. FontAwesome je knihovna obsahující balíček ikon, které lze v rámci designu aplikace použít a knihovna React-animated-css zjednodušuje animování jednotlivých React komponent.

V této sekci jsou postupně probrány všechny stránky, které webová aplikace Privchecker obsahuje. Jsou zde řečeny významy jednotlivých komponent a problémy, jež nastaly při implementaci a jejich řešení. Na závěr je probrána implementace responzivnosti a tvorba hamburger menu.



Obrázek 5.2: Návrh - Hlavní stránka

5.15.1 Hlavní stránka

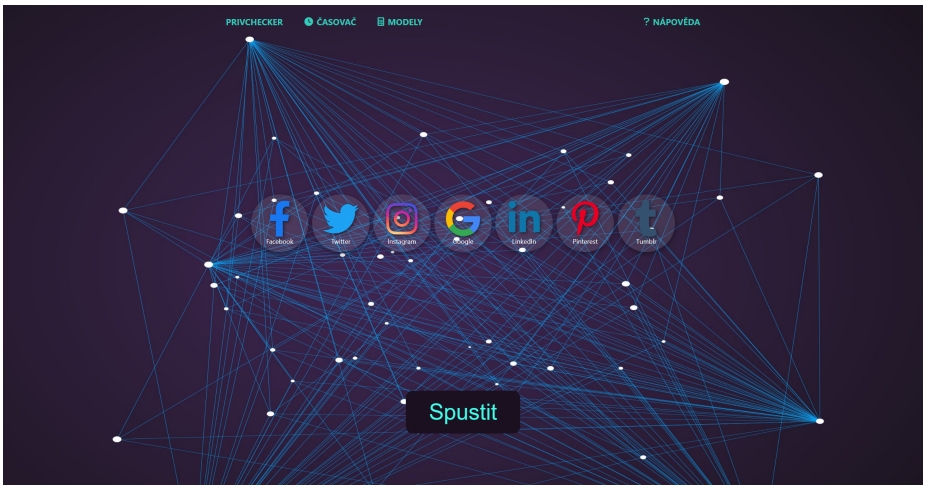
Hlavní stránka, kterou lze vidět na obrázcích 5.2 a 5.3, obsahuje v horní části navigaci, Navigace hlavní stránky je rozebrána v následující podsekci, dále uprostřed je list se so-

ciálními sítěmi. Každá síť je reprezentovaná ikonou a textem obsahující její název. Ikona i název sociální sítě mají fialovo-průhledný podklad pro lepší čitelnost.

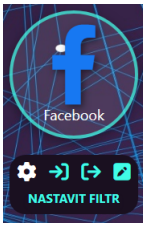
Stisknutím myši nad kruhem obsahující sociální síti se sociální sít přidá do požadavku, jenž se bude posílat na serverovou část aplikace a bude nad touto sociální síti provedeno stahování dat a výpočet *privátního skóre*. Po najetí myší na kruh zaštiťující sociální síť se zobrazí pod sociální sítí nabídka. Nabídka obsahuje: odkaz na *Filtr stránku* sociální sítě, zavolání koncových bodů /login/<název sítě> a /logout/<název sítě> a /edit/<název>.

Každý předmět v nabídce obsahuje ikonu, identifikující akci a textový popis akce. Využití ikon v kruhu sociální sítě a v samotné nabídce zvyšuje použitelnost v rámci uživatelské zkušenosti. Ve spodní části se nachází tlačítko pro spuštění výpočtu privátního skóre na zaškrtnutých sociálních sítích (odešle se požadavek na koncový bod /). Tlačítko „Spustit“ přidává v rámci uživatelské zkušenosti užitečnost.

Problém, který se musel řešit bylo vycentrování listu se sociálními sítěmi. Jelikož každá ikona zabírají různou šířku a výšku, musela být nastavena pevná šířka a výška kruhu a celý box, který obsahuje list se sociálním sítěmi musel být oproti navigaci roztažen po celé šířce *viewportu* (výřezu), z důvodu velikosti listu.



Obrázek 5.3: Implementace - Hlavní stránka



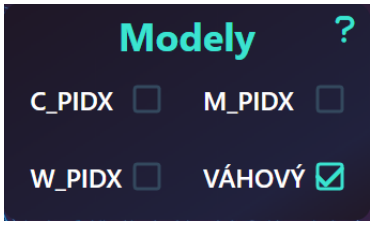
Obrázek 5.4: Implementace - sociální sít

Navigace na hlavní stránce

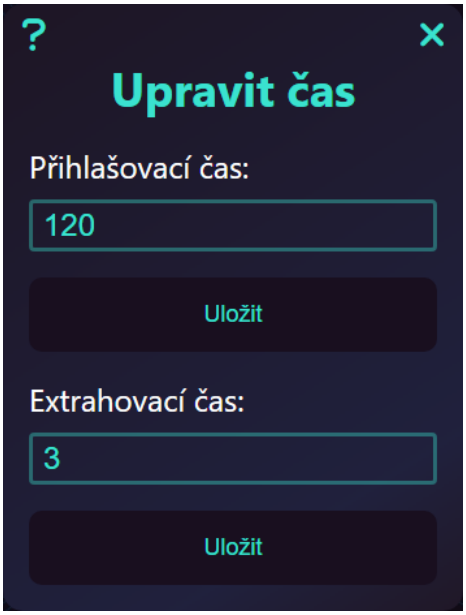
Veškerá tlačítka obsahují název a značku symbolizující význam akce. Navigace na hlavní stránce, jenž je vidět na obrázku 5.3, obsahuje název aplikace, tlačítko s názvem „Časovač“

pro otevření modálního okna. Modální okno, které je na obrázku 5.6 obsahuje formulář nastavování extrahovacího a přihlašovacího času. V případě stisknutí tlačítka „Uložit“ je odeslán požadavek s nastavením času na serverovou část aplikace. Výsledek je poté uživateli zobrazen pod textovým oknem, do kterého se zadává čas v sekundách.

Tlačítko názvem „Modely“ pro otevření modálního okna 5.5 s obsahem všech výpočetních modelů, které aplikace nabízí. Zaškrtnutím modelu je model přidán do požadavku, který je pak odeslán společně s filtrovanými atributy a s názvy sítí na serverovou část aplikace a bude se nad tímto modelem provádět výpočet privátního skóre. V levé části se nachází tlačítko „Nápověda“, které přesměrovává na stránku týkající se nápověd a rad jak používat aplikaci.



Obrázek 5.5: Implementace - Modální okno pro nastavení výpočetních modelů



Obrázek 5.6: Implementace - Modální okno pro nastavení časů

5.15.2 Čekací stránka

Po stisknutí tlačítka „Spustit“ je uživatel přesměrován na *Čekací stránku*, na které se nachází nadpis oznamující, že se provádí práce na serverové části aplikace. Dále je zde varovný text, jenž upozorňuje uživatele, co se stane, když zavře okno speciálního prohlížeče. Velkou

část stránky zabírá pruh znázorňující postup vykonávání implementace na serverové části aplikace.

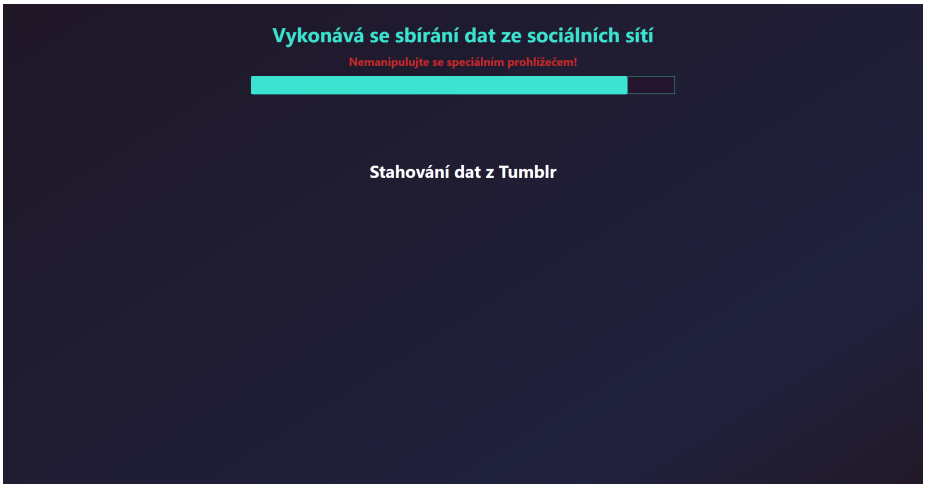
Problémem u pruhu znázorňující postup je ten, že je velmi složité určit přesně v jakém stadiu se nachází vykonávání extrakce dat o soukromí a výpočet privátního skóre. Je to způsobeno čekáním na načtení na stránky. Doba načítání stránky při stejně nastaveném *extrahovacím času* může kolísat a funkce `until` může být v jednom běhu vykonána rychleji a v druhém pomaleji.

Byla provedena série měření, ve kterém bylo zjištěno průměrný čas strávený u každé sociální sítě. Každou sekundu je přičítána Δt a v případě, že by hodnota postupu v pruhu překročila 100, je přidávání Δt zastaveno a čeká se až dorazí odpověď na požadavek ze serverové části a následně je upravena hodnota postupu aktualizována na 100.

Pod pruhem se nachází textové informace o postupu. Jelikož se může stát, že uživatel není přihlášen, je v tomhle textovém poli řečeno, jestli se má uživatel přihlásit na sociální síť. Jelikož požadavek využívá `http protokol` a musí čekat na dokončení stahování a výpočtu privátního skóre a nelze posílat v rámci jednoho požadavku průběžné informace, že je potřeba přihlášení na sociální sít.

Uživatel by mohl čekat celý přihlašovací čas, aby na konci zjistil, že si nevšiml, že je nutné přihlásit se na sociální sít. Kvůli zmíněnému problému bylo potřeba implementace v podobě zachytávání zpráv od serverové části aplikace na schránce socket.io a následné aktualizace do grafického rozhraní.

Obrázek 5.7: Návrh - Čekací stránka



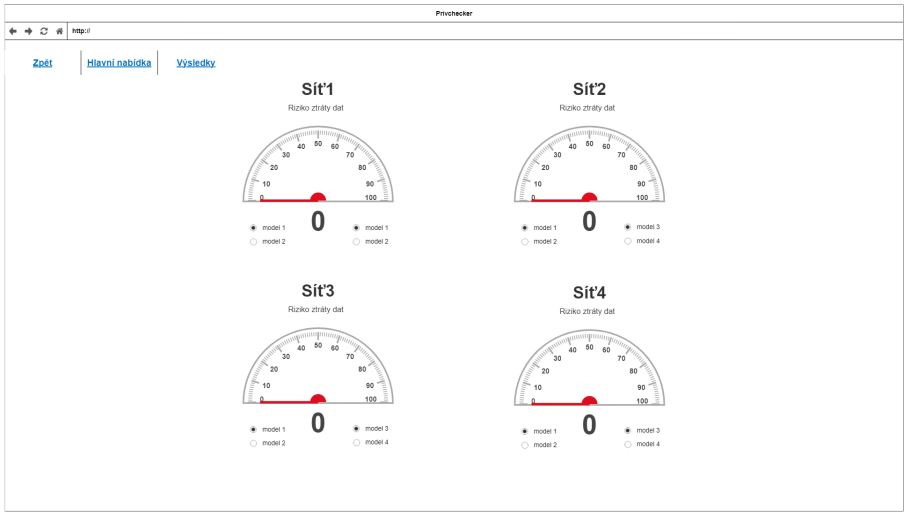
Obrázek 5.8: Implementace - Čekací stránka

5.15.3 Stránka s výsledky výpočtu

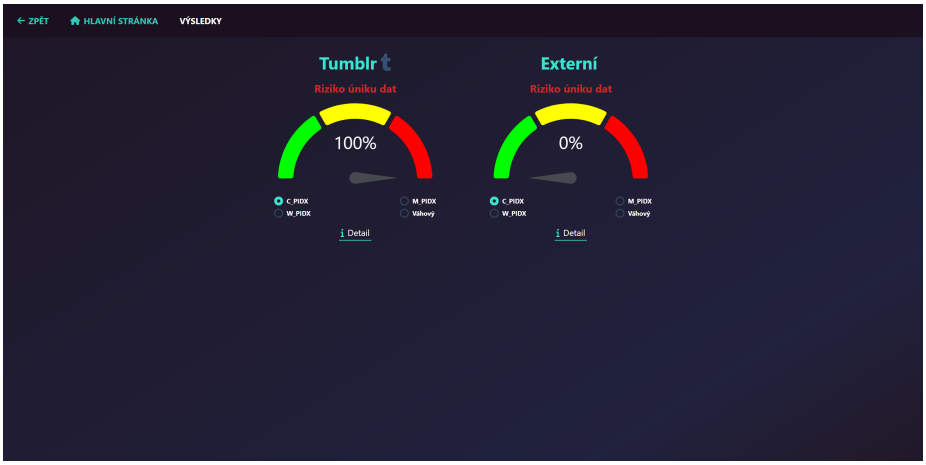
Stránka obsahuje mřížku, ve které se nachází výsledek požadavku, který byl poslán na koncový bod / z *Hlavní stránky*. Každý výsledek sociální sítě obsahuje název sítě s její značkou, dále graf s normalizovaným privátním skórem udaný v procentech jak lze vidět na obrázku 5.9.

Graf v implementaci 5.10 využívá barvy na znázornění rizika úniku dat. Červená znázorňuje problém s nastavením soukromí na dané sociální síti. Žlutá udává, že nastavení není ideální, ale zároveň není kritické. Zelená říká, že uživatel má správně nastavenou sociální síť.

V případě, že uživatel zahrnul v modálním okně „Modely“, jenž je vidět na obrázku 5.5, více modelů nad, kterými se mají provést výpočty privátního skóre, zobrazí se pod grafem pole přepínacích tlačítek a každý přepínač znázorňuje jednotlivý model. Po zaškrtnutí přepínače modelu se hodnota výsledku privátního skóre, jenž bylo vypočteno pomocí zvoleného modelu, zobrazí v grafu. Ve spodní části boxu obsahující výsledek sociální sítě se nachází odkaz „detail“ na *Detail stránku*.



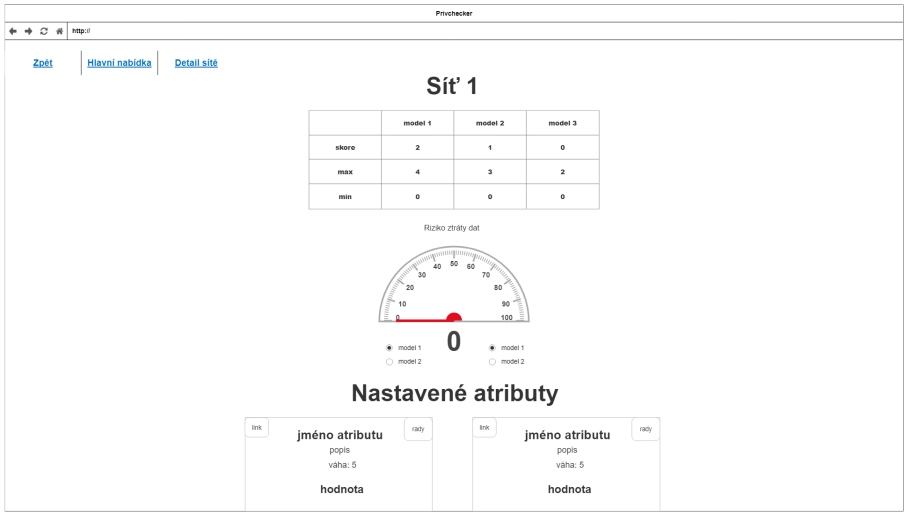
Obrázek 5.9: Návrh - Stránka s výsledky výpočtu



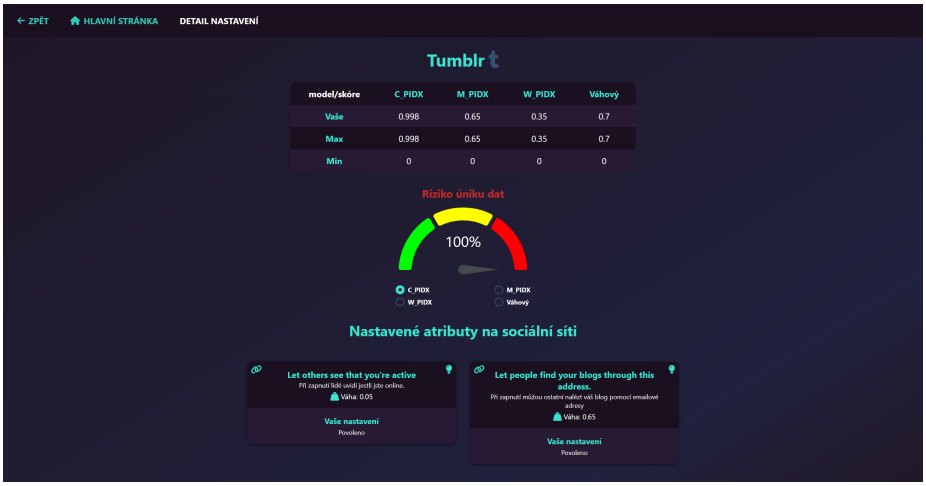
Obrázek 5.10: Implementace - Stránka s výsledky výpočtu

5.15.4 Stránka s detailem sociální sítě

Návrh Stránky s detailem sociální sítě lze vidět na obrázku 5.11 a finální implementace na obrázku 5.12. Jak lze vidět, z již zmíněných obrázků, stránka obsahuje detailní popis o výpočtu privátního skóre na zvolené sociální síti. V horní části je nadpis s názvem sociální sítě a jeho značkou. Pod nadpisem se nachází tabulka obsahující maximální, minimální a uživatelem dosažené privátní skóre na sociální síti s daným modelem a uživatelovo privátní skóre s použitými modely. Dále je zde totožný graf s polem přepínačů jako na stránce s výsledky výpočtu. Pod grafem se nachází mřížka s komponenty atributů, o kterých je následující sekce.



Obrázek 5.11: Návrh - Stránka s detailem výpočtu



Obrázek 5.12: Implementace - Stránka s detailem výpočtu

Atribut

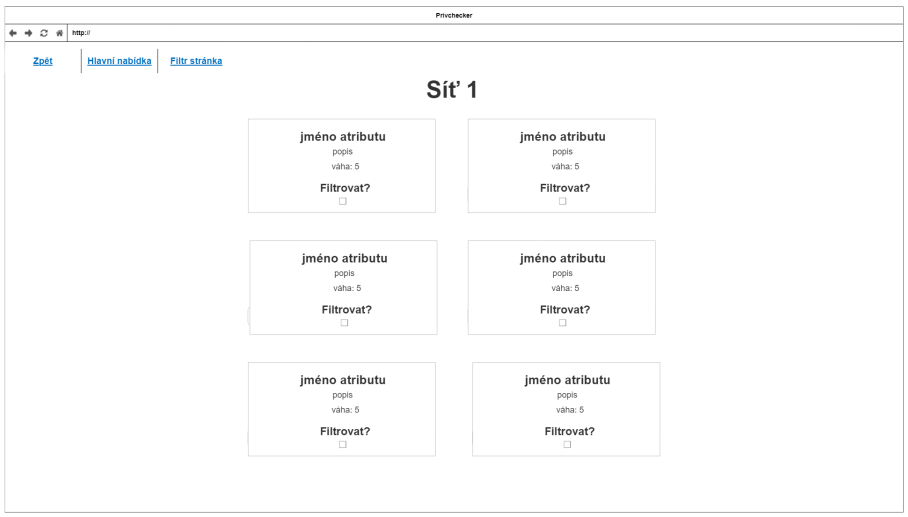
Komponenta *Atribut*, která je na obrázku 5.13, je karta znázorňující nastavovaný atribut soukromí na sociální síti. Karta obsahuje název atributu, popis atributu, váhu atributu, uživatelem nastavenou hodnotu. V levém rohu se nachází ikona, která přesměruje uživatele na stránku, na které si může daný atribut přenastavit. Po stisknutí ikony v pravém rohu se zobrazí modální okno, jejíž obsah je tabulka s radami, jak zlepšit nastavení daného atributu a případně výsledné skóre, pokud by tuhle hodnotu uživatel nastavil.

Obrázek 5.13: Implementace - Atribut

5.15.5 Filtr stránka

Po stisknutí tlačítka ozubeného kolečka, jenž je na vidět na obrázku 5.4, je uživatel přesměrován na stránku, na které může nastavit, které atributy nechce, aby se zahrnovaly do výpočtu privátního skóre na serverové části aplikace. Jak lze vidět na obrázcích 5.15 a 5.14, v horní části se nachází nadpis s názvem sítě s její značkou, pod nadpisem se nachází mřížka s modifikovanou komponentou *Atribut* s tím rozdílem, že neobsahuje značky v rozích a místo uživatelem nastavené hodnoty obsahuje zaškrtávací políčko, které při odškrtnutí je přidáno do filtru, který je součást požadavku, který je odesílán společně s názvy sítí a modely, při stisknutí tlačítka „Spustit“ na *Hlavní stránce*.

Při manipulaci se zaškrtávacím tlačítkem se zobrazí oznamovací lišta s výsledkem operace. U této lišty bylo nutné řešit její viditelnost při rychlejší manipulaci se zaškrtávacími políčky. Je tím myšleno to, že pokud uživatel rychle po sobě manipuloval s políčky, tak bylo nezbytné restartovat časovač, jenž po vypršení volá funkci, která schová oznamovací lištu.

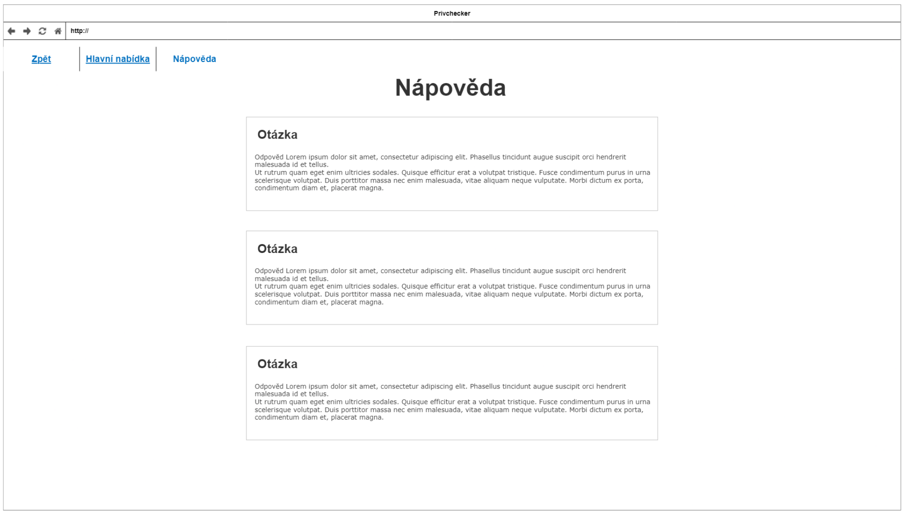


Obrázek 5.14: Návrh - Filtr stránka

Obrázek 5.15: Implementace - Filtr stránka

5.15.6 Stránka s nápovědou

Stránka s nápovědou, kterou lze vidět na obrázcích 5.16 a 5.17 obsahuje karty, které mají pomoci uživateli, pokud si neví rady, jak aplikaci Privchecker použít. Každá karta obsahuje otázku a po stisknutí tlačítka trojúhelníku i odpověď na danou otázku.



Obrázek 5.16: Návrh - Stránka s nápovědou

Obrázek 5.17: Implementace - Stránka s nápovědou

5.15.7 Responzivita aplikace

V závislosti, aby byla aplikace použitelná i na zařízeních s rozdílným rozlišení obrazovky, bylo potřeba implementovat do webového grafického rozhraní responzivnost. Responzivnost byla řešena do výšky a šířky 300x300 pixelů, protože aplikace Privchecker je určena převážně pro stolní počítače. Stolní počítače mají oproti mobilním zařízení v dnešní době velké obrazovky a nepředpokládá se, že uživatel bude mít obrazovku menší než zmiňovaných 300x300 pixelů.

Základní velikost písma pro všechny HTML značky byla ve stylovacím souboru `index.css` zvolena hodnota 62.5 %. Bylo to hlavně z důvodu použití jednotky rem, který při 100% velikosti písma je 16 pixelů. Při 62.5 % je hodnota jednotky rem 10 pixelů. S hodnotou 10 pixelů se lépe počítá.

Poté se ve všech CSS souborech pro určování vycpávky, odsazení, velikosti používá jednotka rem. V aplikace je využit trik, při kterém se při aktivování dotazu media sníží procentuální hodnota velikosti písma na 50 %. To má za následek zmenšení jednotky rem na 8 pixelů a tím pádem zmenšení veškerých písmen, vycpávek, odsazení, bez toho, aniž by bylo nutné psát pro každý prvek v HTML dokumentu dotaz média, ve kterém se sníží daná vlastnost v CSS selektoru.

I přestože byl aplikován trik s rem jednotkou, bylo potřeba v některých případech další úpravy, aby webová aplikace měla přívětivý vzhled. Na *Hlavní stránce* se nachází použití mřížky, která je ve výchozím stavu rozdělena na 7 sloupců, každý pro jeden kruh obsahující sociální síť.

Při zkracování šířky se počet sloupců zmenšoval nejprve na 3 sloupce a poté na 2 sloupce. Počet sloupců se zmenšoval pokaždé, když pravá hrana viewportu „narazila“ na list se sociálními sítěmi. Srážka se stane na 1200 pixelech a poté na 430 pixelech a na těchto hodnotách je nastaven média dotaz, který má v podmínce použitou vlastnost `max-width`, udávající maximální šířku, při které se má dotaz média aktivovat.

Důležité je seřazení dotazů média v souboru sestupně podle hodnoty `max-width`. Podobný postup byl aplikován na *Filtr stránce* a na *Stránce s detailním popisem sítě*, které

obsahují mřížku s atributy, které jsou rozděleny do dvou sloupců. Opět srážka na 1000 pixelech spouští dotaz média, který nastavuje mřížku s atributy na jeden sloupec.

Responzivita na stránce s výsledky výpočtů

Na ostatních stránkách je vytvořena třída `container`, která je použita u prvku, reprezentující kontejner obsahující veškerý obsah stránky. U ostatních stránkách je u této třída nastavena vlastnost `max-width` a na základě, této vlastnosti se obsah, který nemá udanou pevnou výšku nebo šířku zmačkává, jak bylo řečeno v sekci 4.6.4.

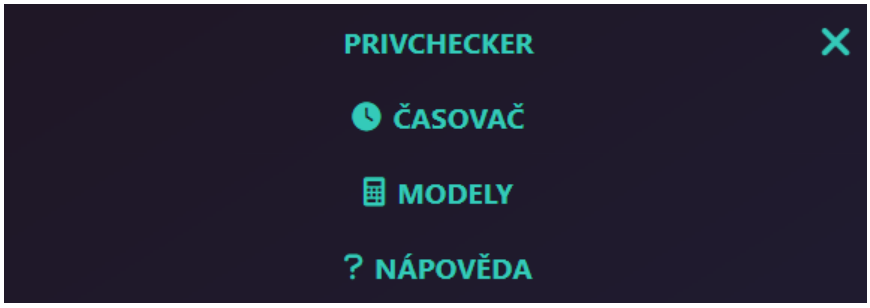
Bohužel grafy, které jsou obsaženy na stránce s výsledky výpočtů jsou ve formátu `svg` a hodnota šířky a výšky se na základě vlastnosti `max-width` ve třídě `container` nepropíše. Na téhle stránce se musí u třídy `container` použít pevná šířky udaná vlastností `width`.

Aby bylo možné udělat na téhle stránce responzivnost a zařídit, aby se jednotlivé grafy zmenšovaly společně se zmenšením šířky viewportu, bylo potřeba implementovat dotazy média, při kterých se sníží pevná šířka v třídě `container` a tím pádem se docílí propsání šířky do grafu, který je reprezentován v `svg` formátu.

Hamburgerové menu

Jelikož navigace obsahuje předměty a zmenšující šířkou viewportu se zmenšuje místo na předměty v navigaci, je potřeba použití techniky hamburgerového menu. Při určité šířce viewportu se schovají všechny předměty v nabídce a zobrazí se pouze ikona hamburgeru. Po stisknutí tlačítka hamburger menu se zobrazí rozvinuté menu, kde každý řádek obsahuje daný předmět v navigaci. Pro zavření rozvinutého menu se nachází v pravém rohu ikona s křížkem, která po stisknutí tlačítka menu zavře.

Kontejner, ve kterém jsou obsaženy značky hamburger menu pro otevření a pro zavření, je ve výchozím stavu neobsaženo v HTML dokumentu. Pokud je maximální šířka viewportu 580 pixelů je pomocí dotazu média zobrazena ikona hamburger menu. Pomocí háku `useState` se nastavuje viditelnost buď ikony pro otevření anebo ikony pro zavření. Vzhled navigace s předměty menu po stisknutí značky pro otevření navigaci, lze vidět na obrázku 5.18.



Obrázek 5.18: Implementace - Rozvinuté hamburger menu

Kapitola 6

Testování

V této kapitole je možné se dočíst o testování po dokončené implementaci aplikace Privchecker. Testování se zabírá kontrolou funkcionality a zjišťování jednotlivých faktorů uživatelské zkušenosti od testovacích uživatelů. Dále jsou zde stanoveny cíle, na které soustředilo v průběhu testování. Také je uvedena testovací množina uživatelů a jejich zastoupení na sociálních sítích. Bylo také provedeno testování, které se týkalo možnosti omezení účtu na sociální sítích. Závěrem je vyhodnoceno testování a uvedeny další možnosti rozšíření aplikace Privchecker.

6.1 Cíle testování

Cílem testování je zjištění funkcionality a celkových dojmů z výsledné aplikace Privchecker. Výstupem by měl být souhrn kritických nedostatků v rámci uživatelské zkušenosti a jejich faktorů, převážně použitelnost, přístupnost, důvěryhodnost a užitečnost, a její souvislost s designem grafického uživatelského rozhraní. Na základě analýzy výsledné aplikace byly stanoveny tyhle cílové body, které bylo potřeba v testování ověřit.

- **Aplikace donutí uživatele změnit nastavení na sociální síti** – Tenhle testovací cíl se zabývá užitečností aplikace pro uživatele. Výsledek vyhodnoceného privátního skóre a následné získání informací, ve kterých atributech nastavení soukromí má daný uživatel mezery, donutí uživatele přenastavit nastavení tak, aby měl privátní skóre v zelených hodnotách.
- **Uživatel je ochoten vložit přihlašovací údaje do prohlížeče** – V rámci tohoto testu je úkolem zjistit, jaké uživatel má pocity s vkládáním jeho přihlašovacích údajů ve speciálním prohlížeči na originální webovou stránku sociální sítě a vzájemné porovnání s původním skriptem.
- **Orientace v grafickém uživatelském rozhraní** – V rámci tohoto bodu je za úkol zjistit, jak se uživatel orientuje v grafickém uživatelském rozhraní. Jak často se dotazuje člověka, který vede testování (mě), jak udělat danou akci, či případné využívání nápověd a následné porozumění nápovědy.
- **Ověření responzivity aplikace** – Účelem bodu Ověření responzivnosti aplikace je zjištění vzhledu grafického uživatelského rozhraní na různých rozlišeních monitoru.

- **Rozdílnost privátního skóre na sociálních sítích mezi skupinami** – Cílem tohoto bodu je odhalit, jak moc se liší privátní skóre mezi jednotlivými uživateli z rozdílných skupin.
- **Zjištění možnosti omezení účtu na sociální síti** – Účelem je zjištění, zdali uživatel může získat omezení nebo zákaz používání sociální sítě, protože použil aplikaci Privchecker pro výpočet privátního skóre.

6.2 Testovací množina uživatelů

Vybraní reprezentanti na testovaní by měli být rozdělení v rámci rozmanitosti do dvou skupin. Obě skupiny mají rozdílné kritérium technické znalosti a povědomí o soukromí na sociálních sítích.

1. skupina – uživatelé technicky znalí, zajímají se, co mají nastaveno na sociálních sítích.
2. skupina – uživatelé technicky méně zdatní, jejich hlavní prioritou je využívání hlavních funkcionalit sociální sítě jako je povídání s přáteli nebo sdílení příspěvků.

Jelikož většina sociálních síti má nastavený minimální věk pro její používání na 13 let a zároveň většina mladých lidí vnímá internet jako běžnou součást, jenž je provází celým nebo většinu života, proto lidé, kteří zapadají do věkového rozmezí 13-40 let byli přiřazeni do první skupiny, technicky nadanějších lidí. Druhá skupina méně technicky nadaných zabírá věkové rozmezí od 40 let a více. Věk 40 let byl zvolen z důvodu, že v tomhle věku lze najít srovnatelné množství technicky nadaných i technicky neznalých lidí. Celkový počet testerů, kteří se podíleli na testování aplikace Privchecker, bylo 20.

6.3 Průběh testování

Na začátku testování byli uživatelé ujištění, že nejsou testování oni, ale výsledná aplikace Privchecker. Poté byli seznámeni s cíli testování. Následně obdrželi testovací protokol, který lze vidět na obrázku 6.1, v kterém měli zadané úkoly ke splnění. Celý průběh testování byl prováděn na počítači uživatele za přítomnosti vedoucího testování (mě). Po zahájení vykonávání testovacího protokolu byl uživatel pozorován, jak reaguje, jaké jsou jeho průběžné dojmy, o které byl požádán, aby je sděloval nahlas. Pokud si uživatel nevěděl rady, bylo mu všechno náležitě vysvětleno. Po splnění všech úkolů byla provedena finální diskuze a dotazník o celkových dojmech z aplikace Privchecker a jestli byla splněna uživatelova očekávání od téhle aplikace.

6.4 Zpětná vazba od uživatelů

Po skončení testování byla provedena závěrečná diskuze a shrnutí nad aplikací. Uživatelé také vyplnili dotazník, jehož výsledky lze vidět v příloze C.1.

Uživatelé ocenili intuitivnost hlavní funkcionality aplikace, což je výpočet privátního skóre a následné zobrazení výsledků. Uživatelé, také potvrdili přijatelný vzhled na rozdílných rozlišení obrazovek. Možnost vkládání přihlašovacích údajů do speciálního prohlížeče oproti původní implementaci skriptu byla pozitivně vítána ze stran všech uživatelů. Dalším pozitivem testování bylo, že nadpoloviční většina testujících uživatelů si uvědomila riziko

ztrát soukromí na sociálních sítích a rozhodla se na základě výpočtů privátního skóre přehodnotit své nastavení na sociální síti. Také uživatelé neobdrželi žádná omezení na sociálních sítích po použití aplikace Privchecker.

6.4.1 Připomínky

Testující uživatelé upozornili na výraznost zelené barvy (`hex. kód 41FCE7`). Byly proto uskutečněny kroky pro ztmavení téhle barvy na tmavší zelenou (`hex. kód 3BE3D0`). Čtyři lidé z druhé skupiny uživatelů se ptali na funkcionality a význam předmětů časovač a modely v navigaci. Ostatní uživatelé buď chápali význam nebo použili stránku s nápovědou. Pro lepší pochopení byly do modálních oken *Modely* a *Časovač* přidáno tlačítko pro otevření nápovědy. Uživatelé také upozornili na opakované vyvolávání animace střelky grafu, která se aktivovala pokaždé, když uživatel otevřel modální okno s radami pro zlepšení atributu soukromí. Animace střelky grafu byla na podnět vypnuta.

Další nedostatek, na který testovací uživatelé upozornili byla pomalá odezva serverové části aplikace po zavření speciálního prohlížeče. Funkcím `find_element`, `get`, jenž jsou součástí instance ovladače trvá dlouho uvědomění, že okno prohlížeče je zavřené. Aplikační rámec Selenium je primárně určený pro automatizované testování webových aplikacích a vývojáři, zřejmě nepočítali s použitím jako je u aplikace Privchecker. Tenhle nedostatek se nepodařilo odstranit. Jeden uživatel si stěžoval na nejasný obsah „Přidáno do filtru“, „Odebráno z filtru“ notifikace, která se vyvolá při zaškrtnutí zaškrtávacího políčka na *Filtr stránce.* u *Atributu.* Na jeho podnět byl obsah notifikace pozměněn na výstižnější.

Úkol
Spustit výpočet na zvolených sociálních sítích
Zjistit jestli je nastavení na sociální síti v pořádku
Zjistit hodnotu privátního skóre, max, min
Najít všechny atributy na sociální síti
Přenastavit atribut na sociální síti na základě rady
Vyzkoušet jiné výpočetní modely pro výpočet
Porovnat hodnoty privátního skóre mezi výpočetními modely
Odhlásit se ze sociální sítě
Přihlásit se na sociální sít pomocí nabídky na hlavní stránce
Zkusit se odhlásit, když už jsem odhlášen
Zkusit se přihlásit, když už jsem přihlášen
Vypnout speciální prohlížeč během vykonávání extrakce dat
Zjistit co je extrahovací a přihlašovací čas
Přenastavit extrahovací a přihlašovací čas
Být odhlášený na sociální síti a spustit výpočet skóre a nepřihlásit se
Vyzkoušet filtrování atributů soukromí

Tabulka 6.1: Úkoly pro testovací uživatele

6.4.2 Sociální sítě u uživatelů

V rámci testování bylo také zaznamenáno jaké sociální sítě používají jednotliví uživatelé. Největší zastoupení měla sociální síť Facebook. Další místa obsadily sociální sítě Google a Instagram. Nejméně používanou sociálními sítěmi jsou Pinterest a Tumblr.

6.4.3 Privátní skóre u uživatelů

V tabulce 6.2 lze vidět naměřené privátní skóre testujících uživatelů na všech sociálních sítích. Lze si povšimnout velké směrodatné odchylky mezi jednotlivými privátními skóre u sociální sítě Facebook, LinkedIn a Pinterest. Velikost směrodatné odchylky u sociální sítě Pinterest je způsobena tím, že privátní skóre se zde počítá jenom z jednoho atributu soukromí. U sociální sítě LinkedIn směrodatná odchylka a privátní skóre byli nejvíce ovlivněny atributy „Nalezení profilu pomocí telefonního čísla“ a „Nalezení profilu pomocí e-mailové adresy“, které uživatele nejčastěji na této sociální síti nastavovali. Na základě toho byly detekovány velké skoky mezi hodnotami „Všichni“, „Spojení druhého stupně“ a „Nikdo“ zmíněných atributů.

U sociální sítě Facebook je velký rozptyl mezi hodnotami způsobený velkým množstvím atributů nastavení soukromí. Také lze vidět v tabulce 6.2 opakující se hodnotu privátního skóre $0,8616$, která je shodná s privátním skóre nově vytvořeného účtu na sociální síti Facebook. Lze z toho vidět, že sociální sít Facebook nehledí na soukromí nově přidaných uživatelů. Uživatelé pak musí z vlastní iniciativy nastavení pozměnit, aby si zachovali určitou míru soukromí.

Oproti tomu sociální síť Twitter při nově vzniklém účtu má ve výchozím stavu zákaz přidávání polohy k příspěvkům a také nemožnost vyhledání daného účtu pomocí telefonního čísla nebo mailové adresy. Uživatelé sociální sítě Twitter nejčastěji kombinovali výchozí nastavení s atributem „Chránit Tweety“, který zaručuje, že příspěvky uživatele uvidí pouze jeho sledující. Sociální síť Twitter se může chlubit nejmenším průměrným privátním skórem.

U sociální sítě Google si lze povšimnout opakující se hodnot $0,4717$ a $0,5611$. Někteří testující uživatelé netušili, že je možnost v rámci sociální sítě Google nastavit atributy soukromí a rozdíl nastavení hodnot $0,4717$ a $0,5611$ spočívá pouze v nastavení fotografie. Uživatelé, kteří byli obeznámeni nastavením soukromí na sociální síti Google nejčastěji volili vypnutí atributů „Kontaktní údaje uložené z komunikace“ „Kontakty z vašich zařízení“, jenž se týká ukládání kontaktů a komunikace na servery Google. U sociální sítě Instagram uživatelé nejčastěji volili zapnutí nastavení „Soukromý účet“ jako v případě sociální sítě Twitter a celkově uživatelé dosahovali přijatelných hodnot privátního skóre s průměrem $0,5246$.

sítě [1]/uživatelé	FB	TW	IG	GG	LI	PR	TB
1	0,8634		0,6313	0,4717	0,929		
2	0,3998		0,5757		0,3429		
3	0,4159	0,1356	0,4607	0,2766			1
4	0,3905	0,4915	0,419	0,4067	0,3729		
5	0,382	0,2881	0,4677	0,5611			
6	0,4926	0,1356	0,6313	0,5611			
7	0,8616	0,1356				1	
8	0,4184	0,580	0,5163	0,2766			
9	0,3607	0,1356	0,4607	0,4717	0,3498		
10	0,3112			0,4473			
11	0,2939	0,580		0,3661			
12	0,2135			0,4473		1	
13	0,1315	0,580		0,3661	0,8534	0	
14			0,4607				
15	0,8616						
16	0,8616		0,6313	0,4717		1	
17	0,6835		0,6243			1	
18	0,8616				0,9015		
19			0,5163			1	
20	0,3949			0,5611			
průměr	0,5110	0,1662	0,5246	0,4373	0,6249	0,8333	1
směrodatná odchylka	0,2500	0,1412	0,1694	0,968	0,2966	0,3726	

Tabulka 6.2: Privátní skóre uživatelů na sociálních sítích. Privátní skóre bylo normalizováno do intervalu $\langle 0, 1 \rangle$ v rámci dané sociální sítě.

6.5 Možná rozšíření aplikace Privchecker

Nejlogičtějším rozšířením, které se nabízí je přidání dalších sociálních sítích, na kterých může uživatel provést výpočet privátního skóre. Další možností rozšíření by bylo přidání stránky, na které by si uživatel nastavil atributy soukromí na dané sociální síti. Uživatelovo nastavení by se odeslalo na serverovou část aplikace. Serverová část aplikace by pomocí knihovny Selenium a ovladače speciálního prohlížeče programově nastavila na sociální síti uživatelovo zadané nastavení atributů soukromí. Další možností vylepšení aplikace může být přidání grafu, který by uživateli sděloval, jak si stojí se svým privátním skóre oproti ostatním uživatelům, kteří aplikaci Privchecker používají.

[1] FB = Facebook, TW = Twitter, IG = Instagram, GG = Google, LI = LinkedIn, PR = Pinterest, TB = Tumblr

Kapitola 7

Závěr

Cílem bylo vytvoření webového grafického uživatelského rozhraní pro nástroj Privchecker. Na začátku byl nejprve probrán původní nástroj Privchecker, následně se prozkoumal výpočet privátního skóre a tvorba webových uživatelských rozhraní.

Podařilo se se vyřešit problém se zadáváním uživatelových přihlašovacích údajů. Oproti původnímu zadávání se údaje zadávají přímo do speciálního prohlížeče. Také bylo pro uživatele implementováno uložení přihlášeného stavu na sociálních sítích, aby nemuseli znovu zadávat přihlašovací údaje. Další překážkou bylo vznik omezení účtu při použití nástroje Privchecker, jenž se vyřešila vložením knihoven selenium-stealth a undetected-chrome.

Jelikož původní nástroj Privchecker nefungoval bylo potřeba funkce starající se o extrakci dat soukromí opravit. Byly vytvořeny nové cesty k jednotlivým značkám obsahující hodnoty o soukromí a dále byly přidány nové atributy soukromí. Bylo to potřeba z důvodu ověření funkcionality grafického uživatelského rozhraní a následné propojení se serverovou částí aplikace.

Uživatelské rozhraní nástroje Privchecker obsahuje 6 stránek. Na hlavní stránce uživatel zadává sociální sítě u kterých chce vypočítat privátní skóre. Poté čeká na vykonání výpočtu na čekací stránce. Výsledky si poté zobrazí na stránce s výsledky, kde se nachází grafy, jenž znázorňují bezpečnost uživatelova účtu. Pro získání detailního popisu stažených dat má uživatel možnost nahlédnout na detailní stránce sociální sítě. V případě, že si uživatel neví rady, aplikace obsahuje stránku s nápovědou. Pokud se uživatel chce zaměřit na určité atributy soukromí, je zde možnost zaškrtnutí atributů na stránce s filtrem. Součástí uživatelského rozhraní je možnost nastavení extrahovacího a přihlašovacího času, který ovlivňuje dobu čekání na stahování dat z jednotlivých značek nebo čekání serverové části na přihlášení.

Po implementaci aplikace bylo provedeno testování s uživateli. Od uživatelů byla získána zpětná vazba s podměty, které byly zapracovány do aplikace. Poté se analyzovalo jaké sociální sítě uživatelé používají, jaké mají privátní skóre a jejich nejčastější nastavené atributy soukromí. Také byl pro uživatele vytvořen instalační soubor. Součást instalace pomocí zmíněného souboru je automatické doinstalování potřebných požadavků jako je Python 3.10.0, Google Chrome a doplňující knihovny.

Literatura

[1] Aghasian, E., Garg, S., Gao, L., Yu, S. a Montgomery, J. Scoring user's privacy disclosure across multiple online social networks. *IEEE access*. Červen 2017, PP, s. 1–1, [cit. 2021-12-15]. DOI: 10.1109/ACCESS.2017.2720187. Dostupné z: https://www.researchgate.net/publication/318330101_Scoring_Users'_Privacy_Disclosure_Across_Multiple_Online_Social_Networks.

[2] AlanAI. *Types of User Interfaces*. 2022 [cit. 2022-02-10]. Dostupné z: https://alan.app/blog/types-of-user-interface/.

[3] Angular a Contributors. *Feature modules* [online]. 2022. 2022-02-28 [cit. 2022-03-05]. Dostupné z: https://angular.io/guide/feature-modules.

[4] Angular a Contributors. *Getting started with Angular* [online]. 2022. 2022-02-28 [cit. 2022-03-05]. Dostupné z: https://angular.io/start.

[5] Blair Early, A. a Zender, M. User interface design principles for interaction design. *Design Issues*. MIT Press. Červenec 2008, sv. 24, č. 3, s. 85–107. DOI: 10.1162/desi.2008.24.3.85.

[6] Borrelli, P. *Angular vs. React vs. Vue.js: Comparing performance* [online]. 2021. 2021-10-26 [cit. 2022-01-17]. Dostupné z: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/.

[7] community, S. *WebDriver | Selenium* [online]. Software freedom conservancy, 2022 [cit. 2022-04-20]. Dostupné z: https://www.selenium.dev/documentation/webdriver/.

[8] Contributor, T. *Web application (Web app)* [online]. © 2006 - 2022 [cit. 2022-01-11]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app.

[9] contributors, M. *Basic concepts of flexbox* [online]. 2021. 2021-11-28 [cit. 2021-12-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox.

[10] contributors, M. *JavaScript First Steps* [online]. 2021. 2021-12-30 [cit. 2022-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps.

[11] contributors, M. *Responsive design*. 2021. [cit. 2021-12-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design#modern_layout_technologies.

[12] CONTRIBUTORS, M. *Client-Server Overview* [online]. 2022. 2022-01-05 [cit. 2022-01-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview.

[13] CONTRIBUTORS, M. *CSS: Cascading Style Sheets* [online]. 2022. 2022-01-01 [cit. 2022-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS.

[14] CONTRIBUTORS, M. *HTML: HyperText Markup Language* [online]. 2022. 2022-01-01 [cit. 2022-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML.

[15] CONTRIBUTORS, M. *Using HTTP cookies*. 2022. [cit. 2022-04-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

[16] CONTRIBUTORS, M. *WebDriver | MDN* [online]. MDN, únor 2022 [cit. 2022-04-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/WebDriver.

[17] DAITYARI, S. *Angular vs React vs Vue: Which Framework to Choose* [online]. 2021. 2021-12-30 [cit. 2022-01-17]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

[18] DAREJEH, A. a SINGH, D. A review on user interface design principles to increase software usability for users with less computer literacy. *Journal of computer science*. Science Publications. Listopad 2013, sv. 9, č. 11, s. 1443–1450. DOI: 10.3844/jcssp.2013.1443.1450. Dostupné z: https://www.researchgate.net/publication/277589616_A_review_on_user_interface_design_principles_to_increase_software_usability_for_users_with_less_computer_literacy.

[19] EDUCATION, I. C. *Three-Tier Architecture* [online]. 2020. 2020-10-28 [cit. 2022-01-09]. Dostupné z: https://www.ibm.com/cz-en/cloud/learn/three-tier-architecture.

[20] GOOGLE. *Using OAuth 2.0 to Access Google APIs* [online]. 2021. 2021-12-09 [cit. 2022-04-22]. Dostupné z: https://developers.google.com/identity/protocols/oauth2.

[21] HERATH, C. *Web Development in 2021: Dynamic vs. Static vs. Single-Page Architecture* [online]. 2021. 2021-04-20 [cit. 2022-01-14]. Dostupné z: https://betterprogramming.pub/web-development-in-2021-dynamic-vs-static-vs-single-page-architecture-399d0c3defe6.

[22] INDEED EDITORIAL, T. *What Is a Web Application? How It Works, Benefits and Examples*. 2021. [cit. 2022-01-05]. Dostupné z: https://www.indeed.com/career-advice/career-development/what-is-web-application.

[23] JANUŠ, F. *Pokročilá evaluace úrovně privátnosti v sociálních sítích*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/18824/.

[24] KOĎOUSKOVÁ, B. *JAK NA SEO (NEJEN) PRO SINGLE-PAGE WEBOVÉ APLIKACE?* [online]. 2021. 2021-10-02 [cit. 2022-01-10]. Dostupné z: https://www.rascasone.com/cs/blog/seo-spa-jednostrankove-web-aplikace.

[25] LEASE, D. *TypeScript: What is it & when is it useful?* [online]. 2018. 2018-30-01 [cit. 2022-01-09]. Dostupné z: https://medium.com/front-end-weekly/typescript-what-is-it-when-is-it-useful-c4c41b5c4ae7.

[26] Li, X., Xin, Y., Zhao, C., Yang, Y., Luo, S. et al. Using User Behavior to Measure Privacy on Online Social Networks. *IEEE Access*. IEEE. 2020, sv. 8, s. 108387–108401. DOI: 10.1109/ACCESS.2020.3000780. Dostupné z: https://ieeexplore.ieee.org/document/9110892.

[27] Li, X., Yang, Y., Chen, Y. a Niu, X. A Privacy Measurement Framework for Multiple Online Social Networks against Social Identity Linkage. *Applied Sciences*. Multidisciplinary Digital Publishing Institute. Říjen 2018, sv. 8, s. 1790, [cit. 2021-12-15]. DOI: 10.3390/app8101790. Dostupné z: https://www.researchgate.net/publication/328036640_A_Privacy_Measurement_Framework_for_Multiple_Online_Social_Networks_against_Social_Identity_Linkage.

[28] Majda, D. *Knihovny vs. frameworky* [online]. 2009. 2009-10-14 [cit. 2021-11-24]. Dostupné z: https://majda.cz/blog/265/.

[29] Meta a Contributors. *Introducing Hooks - React* [online]. 2021. 2021-26-07 [cit. 2022-01-17]. Dostupné z: https://reactjs.org/docs/hooks-intro.html.

[30] Michálek, M. *Průvodce CSS preprocesory: který vybrat?* [online]. 2014. 2014-04-01 [cit. 2022-01-14]. Dostupné z: https://www.vzhurudolu.cz/blog/15-css-preprocesory-4.

[31] Morville, P. *User Experience Design* [online]. Semantic Studios, červen 2004. 2004-21-06 [cit. 2022-02-20]. Dostupné z: http://semanticstudios.com/user_experience_design/.

[32] Nesher, G. *CSS Architecture for Component-Based Applications* [online]. 2019. 2019-31-12 [cit. 2022-01-05]. Dostupné z: https://www.infoq.com/news/2019/12/components-css-architecture/.

[33] Norman, D. a Nielsen, J. *The Definition of User Experience (UX)* [online]. Norman Nielsen Group, © 1998-2022 [cit. 2022-02-20]. Dostupné z: https://www.nngroup.com/articles/definition-user-experience/.

[34] Oberlehner, M. *Thoughts about Utility-First CSS Frameworks*. 2019 [cit. 2022-01-12]. Dostupné z: https://markus.oberlehner.net/blog/thoughts-about-utility-first-css-frameworks/.

[35] Pattakos, A. *Angular vs React vs Vue 2022* [online]. 2022. 2021-04-01 [cit. 2022-01-17]. Dostupné z: https://athemes.com/guides/angular-vs-react-vs-vue/.

[36] Petkos, G., Papadopoulos, S. a Kompatsiaris, I. PScore: A Framework for Enhancing Privacy Awareness in Online Social Networks. In: Availability, R. 10th International Conference on a Security, ed. Srpen 2015, s. 592–600 [cit. 2022-02-15]. DOI: 10.1109/ARES.2015.80. ISBN 978-1-4673-6590-1.

[37] Ranjan, A., Kumar, R. a Dhar, J. A Comparative Study between Dynamic Web Scripting Languages. In: Kannan Rajkumar, A. F., ed. *Data Engineering and Management*. Springer Berlin Heidelberg, Leden 2012, s. 288–295. DOI: 10.1007/978-3-642-27872-3_43. ISBN 978-3-642-27871-6.

[38] Saha, D., Mandal, A. a Pal, S. User interface design issues for easy and efficient human computer interaction: an explanatory approach. *International Journal of Computer Sciences and Engineering*. 2015, sv. 3, č. 1, s. 127–135, [cit. 2021-11-10]. Dostupné z: https://www.researchgate.net/publication/294428623_User_Interface_Design_Issues_for_Easy_and_Efficient_Human_Computer_Interaction_An_Explanatory_Approach.

[39] Usabilla. *A Short History of Computer User Interface Design* [online]. 2017 [cit. 2021-11-14]. Dostupné z: https://medium.theuxblog.com/a-short-history-of-computer-user-interface-design-29a916e5c2f5.

[40] USA.gov. *User Centered Design Basics* [online]. 2020 [cit. 2022-02-20]. Dostupné z: https://www.usability.gov/what-and-why/user-centered-design.html.

[41] USA.gov. *User Experience Basics* [online]. 2020 [cit. 2022-02-20]. Dostupné z: https://www.usability.gov/what-and-why/user-experience.html.

[42] Vue, C. *Introduction | Vue.js* [online]. 2022. 2022 [cit. 2022-01-17]. Dostupné z: https://vuejs.org/guide/introduction.html.

[43] Čápka, D. *Lekce 1 - Úvod do JavaScriptu*. 2013 [cit. 2022-01-12]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk.

Příloha A

Obsah CD

Přiložené CD obsahuje:

- `src-fe` - zdrojový kód klientské části aplikace Privchecker (React)
- `src-be` - zdrojový kód serverové části aplikace Privchecker (Python)
- `src-combined` - zdrojový kód serverové části s vygenerovaným HTML z klientské části
- `latex` - zdrojový kód textu bakalářské práce
- `setupPrivchecker` - instalační program aplikace
- `bakalarska-prace` - PDF soubor s bakalářskou prací
- `demo` - demonstrační video

Příloha B

Postup instalace

B.1 Požadavky

- Windows 10
- připojení k internetu
- **Volitelně:**

  Google Chrome verze 99 a výše

  Python 3.10.0

B.2 Postupy

B.2.1 Postup 1

1. Spusťte soubor `setupPrivchecker.exe`.
2. Řiďte se pokyny instalace.
3. V instalaci pro Python zaškrtněte pole s názvem `Add Python 3.10.0 to PATH`. Pokud jste měli, předtím Python 3.10.0 nainstalovaný a nejste si jistí, že jej máte přidané v proměnné PATH, stiskněte tlačítko `modify`, `next` a poté zaškrtněte pole `Add Python to environment variables`.
4. Spusťte zástupce `Privchecker.bat` na ploše nebo v cílovém adresáři.

B.2.2 Postup 2

1. Stáhněte a nainstalujte `Python verze 3.10.0` a `Node verze 16.15.0`.
2. V instalaci pro Python zaškrtněte pole s názvem `Add Python 3.10.0 to PATH`. Pokud jste měli, předtím Python 3.10.0 nainstalovaný a nejste si jistí, že jej máte přidané v proměnné PATH, stiskněte tlačítko `modify`, `next` a poté zaškrtněte pole `Add Python to environment variables`.
3. Zkopírujte složky `src-fe` a `src-be`.
4. Ve složce `src-fe` spusťte v příkazovém řádku příkaz `npm install`.

5. Ve složce `src-be` spusťte skript `install.bat`.
6. Ve složce `src-be` spustíte serverovou část v příkazovém řádku pomocí příkazu `.\venv\Scripts\activate` a následně spusťte příkaz `python .\app.py`.
7. Ve složce `src-fe` spusťte pomocí příkazu `npm start` v příkazovém řádku klientskou část aplikace.

Příloha C

Dotazník

odpověď/ otázka	Nesouhlasím	Spíše nesouhlasím	Nejsem si jistý/(á)	Spíše souhlasím	Souhlasím
Ovládaní aplikace bylo zbytečně náročné	15	4	1	0	0
Často jsem nevěděl(a), co mám dělat	16	3	1	0	0
Graf skóre mi řekl, jak na tom jsem	0	0	0	2	18
Dokázal(a) jsem rozlišit důležitost atributů	0	0	2	3	15
V aplikaci se děly nepochopitelné věci	13	5	2	0	0
Aplikaci bych vyzkoušel(a) znovu	0	0	2	9	9
Aplikace se mi vzhledově líbila	0	0	1	5	14

Tabulka C.1: Dotazník k testování