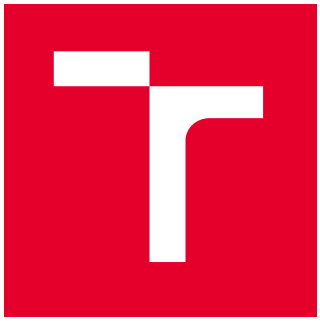


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE ATRIBUTOVÉHO AUTENTIZAČNÍHO SYSTÉMU

WEB APPLICATION OF ATTRIBUTE-BASED AUTHENTICATION SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Roman Klampár

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Petr Dzurenda, Ph.D.

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Roman Klampár ***ID:*** 221551

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Webová aplikace atributového autentizačního systému

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s vývojem moderních GUI webových aplikací, např. Progressive Web Apps, AWS, Node.js atd. Nastudujte problematiku atributové autentizace, pomocí tzv. Attribute-based Credentials. Dále analyzujte možnosti integrace knihoven a API nižších programovacích jazyků a propojením s chytrými telefony s OS Android (NFC, BLE). Výstupem práce bude návrh a implementace GUI webové aplikace autentizačního systému, která rovněž umožní využívat kryptografické knihovny nižších programovacích jazyků a bude schopna komunikovat s uživatelskou Android aplikací na chytrém telefonu. Webová aplikace bude nabízet grafické rozhraní pro entity autentizačního systému: ověřovatel, vydavatel a revokační autorita.

DOPORUČENÁ LITERATURA:

[1] MENEZES, Alfred, Paul C VAN OORSCHOT a Scott A VANSTONE. Handbook of applied cryptography. Boca Raton: CRC Press, c1997. Discrete mathematics and its applications. ISBN 0-8493-8523-7.

[2] ATER, Tal. Building progressive web apps: bringing the power of native to the browser. O'Reilly Media, Inc., 2017.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Táto bakalárska práca sa zaoberá vytvorením webových aplikácií pre entity vydavateľa a overovateľa atribútového autentizačného systému Adospio. Webové aplikácie boli implementované pomocou mikroframeworku Flask, frameworku Vue.js. a databáze PostgreSQL. Aplikácia vydavateľa umožňuje vytváranie nových digitálnych certifikátov obsahujúce podpísané atribúty. Druhou aplikáciou je entita overovateľ, pre ktorú je vytvorené grafické užívateľské rozhranie určené k správe overovacieho skriptu.

KĽÚČOVÉ SLOVÁ

webová aplikácia, Python, Flask, Vue.js, PostgreSQL, Google Auth, REST, autorizácia, Docker, atribútová autentizácia, ctypes, digitálne certifikáty

ABSTRACT

This bachelor thesis deals with the creation of web applications for the issuer and verifier entities of an Attribute-Based Credentials system Adopsio. The web applications were implemented using the microframework Flask, the Vue.js framework, and the PostgreSQL database. The issuer application allows the creation of new digital certificates containing signed attributes. The second application is the verifier entity, for which a graphical user interface is created to manage the authentication script.

KEYWORDS

web application, Python, Flask, Vue.js, PostgreSQL, Google Auth, REST, authorization, Docker, Attribute-Based Credentials, digital certification

Vysadené pomocou balíčku `thesis` verzie 4.07; `http://latex.feec.vutbr.cz`

KLAMPÁR, Roman. *Webová aplikace atributového autentizačního systému*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 60 s. Bakalárska práca. Vedúci práce: Ing. Petr Dzurenda, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Roman Klampár
VUT ID autora:	221551
Typ práce:	Bakalárska práca
Akademický rok:	2021/22
Téma závěrečnej práce:	Webová aplikace atributového autentizačního systému

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Ďakujem vedúcemu bakalárskej práce Ing. Petrovi Dzurendovi, Ph.D. za jeho cenné rady, poznatky a konzultácie, ktoré mi boli nápomocné pri tvorbe tejto práce. Ďalej by som sa rád poďakova Willimu Lazarovovi za trefné poznatky. Veľká vďaka smeruje k mojej rodine, ktorá ma vždy podporuje a dodáva silu v ďalších krokoch. Nakoniec by som sa rád poďakoval kamarátom za psychickú podporu.

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

S príchodom celosvetovej pandémie koronavírusu (skrátene *COVID-19*) bolo potrebné zaistiť spoľahlivý dôkaz, ktorý by umožnil preukázanie o prekonaní ochorenia, vakcinácií, alebo negatívneho výsledku testu na ochorenie COVID-19. Za týmto účelom vznikli digitálne certifikáty, ktoré zastrešujú jednotný a spoľahlivý dôkaz. V Českej republike vznikli dve mobilné aplikácie *Tečka* a *čTečka*, ktoré zaisťujú uchovanie a overenie platnosti digitálnych certifikátov za pomoci QR kódu.

Aktuálne riešenie digitálnych certifikátov má množstvo bezpečnostných a technických nedostatkov. Pri procese overovania sa overuje výhradne certifikát, ale nie poskytujúca osoba. To môže viesť k zdieľaniu jedného platného certifikátu medzi viacerými ľudmi. Ďalší nedostatok vzniká taktiež pri overovaní, kde sú po naskenovaní QR kódu zobrazené osobné údaje overovanej osoby. V niektorých prípadoch sa počas overovania fotografujú certifikáty čo znamená, že okrem overenia platnosti sa archivujú osobné údaje - zhromažďovanie osobných údajov.

Prvá kapitola 1 bakalárskej práce je venovaná webovým aplikáciam Bude v nej predstavená architektúra REST a technológia *Progressive Web Application*. Následne sú predstavené technológie využívané k vývoji frontendu a backendu webových aplikácií. V druhej kapitole 2 sú predstavené možnosti implementácie knižníc nižších programovacích jazykov do webových aplikácií a technológie určené k bezdrôtovej komunikácií. V tretej kapitole 3 bude predstavený pojem *atribútová autentizácia*. V závere teoretickej kapitole 4 sa nachádza popis aktuálneho riešenia digitálnych certifikátov.

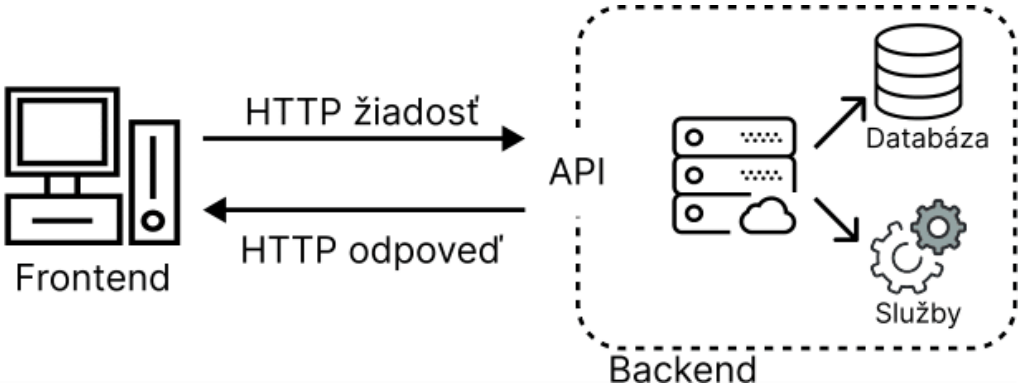
Cieľom bakalárskej práce je zúžitkovať získané informácie a navrhnúť 5 webové aplikácie pre entity vydavateľa a overovateľa atribútového autentizačného systému. V poslednej kapitole 6 je predstavený postup implementácie serverovej a klientskej časti webových aplikácií. Výsledkom práce sú dve webové aplikácie, kde vydavateľ sprostredkuje správu digitálnych certifikátov pre prihlásených užívateľov. Overovateľ umožňuje spustenie aplikácie na overovanie digitálnych certifikátov pomocou grafického rozhrania.

1 Webové aplikácie

Webová aplikácia je druh aplikačného programu, ktorý sa nachádza na vzdialenom serveri a užívateľovi je poskytovaný prostredníctvom internetového pripojenia v kombinácií s ľubovoľným webovým prehliadačom nainštalovaným na počítačovom zariadení.

Väčšina moderných aplikácií využíva architektúru *Representational State Transfer* (skrátene *REST*). REST slúži k architektonickému návrhu aplikácií postavených na modele *klient - server*. Najbežnejšie použitie REST je v kombinácii s protokolom *Hypertext Transfer Protocol* (skrátene *HTTP*), ktorý umožňuje vytvorenie *Application Programming Interface* (skrátene *API*). Táto kombinácia sa taktiež nazýva *REST API*. Komunikácia medzi klientom a serverom pri použití REST API využíva základné HTTP metódy - GET, POST, PUT, DELETE a PATH. Dáta, ktoré sa medzi klientom a serverom prenášajú sú zvyčajne vo formáte *JavaScript Object Notation* (skrátene *JSON*), alebo *Extensible Markup Language* (skrátene *XML*). [1]

Na obrázku 1.1 je znázornená schéma webovej aplikácie pri využití REST API a základných častí, *frontend* 1.1 a *backend* 1.2. Na strane backendu sa môže nachádzať databázový systém poprípade dodatočné služby alebo skripty.



Obr. 1.1: Schematické znázornenie webovej aplikácie.

Progresívne webové aplikácie

Progresívne webové aplikácie (skrátene *PWA*) je druh aplikácií, ktoré pomohli vyriešiť problém s kompatibilitou natívnych aplikácií pri použití na rôznych platformách. Táto technológia umožňuje za relatívne krátky čas vyvinúť plnohodnotnú aplikáciu použiteľnú pre rôzne platformy. V dokumente [2] bola myšlienka PWA popísaná ako zjednotenie zážitku z internetu na mobilných ale aj iných zariadeniach, medzi ktoré

môžu patriť notebooky, tablety, alebo zariadenia s rôznym rozsahom pixelov. Užívateľ môže použiť PWA pomocou webového prehliadača, alebo si ju nainštalovať do zariadenia kde sa bude tváriť ako natívna aplikácia.

Podľa dokumentu [3] by mala každá PWA implementovať štyri komponenty, *Web App manifest*, *Service Worker*, *Application shell architecture* a *Transport Layer Security*.

Web App manifest je súbor vo formáte JSON, ktorý udáva vzhľad natívneho rozhrania. Umožňuje vývojárom nastavovať aplikáciu ako napr. zvoliť spôsob zobrazenia užívateľovi, alebo spôsob, akým ju možno spustiť. Ďalej v sebe zahŕňa názov webovej aplikácie, autorov, ikony a umiestnenie. [3]

Service Worker je skript, ktorý reaguje na interakciu od užívateľa a riadi sieťové požiadavky. Tieto aktivity si ukladá do tzv. *medzipamäti* aplikácie čo nám umožňuje prácu v offline režime. Okrem offline režimu zabezpečuje tzv. *push notifications*, ktoré aj pri zatvorenej aplikácie budú upozorňovať užívateľov na určité akcie. Ďalšou dôležitou funkciou, ktorú zabezpečuje Service Worker je *background synchronization*, ktorá zdržuje akcie pokiaľ sa neobnoví stabilné pripojenie. [3]

Application shell architecture sa skladá zo základných konštrukčných prvkov, ktoré nám napomáhajú k tomu, aby aplikácia bežala offline. Medzi základné kritéria *shellu* podľa dokumentu [2] patrí rýchly čas načítania, ukladanie do vyrovnávacej pamäti a zobrazovanie dynamického obsahu.

Transport Layer Security (skrátene *TLS*) je štandard, ktorý sa používa na zabezpečenie komunikácie medzi dvoma zariadeniami. Vyžaduje sa komunikovať cez protokol HTTPS a inštalácia *Secure Sockets Layer* (skrátene *SSL*) certifikátu na servery. [3]

1.1 Frontend webovej aplikácie

Vzhľad webovej aplikácie zaisťuje frontend. Hlavnou úlohou pri vývoji frontendu je vytvoriť prostredie, ktoré bude pre užívateľov intuitívne a elegantné. K splneniu tejto úlohy sa využívajú jazyky *HTML*, *CSS* a *JavaScript*. V dnešnej dobe existuje množstvo knižníc, ktoré vznikli za účelom zjednodušenia vývoja užívateľského rozhrania (skrátene *UI*). Primárne sa jedná o knižnice postavené na programovacom jazyku JavaScript, ktorý patrí medzi najpopulárnejší jazyk pre vývoj frontendu.

1.1.1 Framework Angular

Za vznikom stojí spoločnosť Google, ktorá ho naďalej udržuje a vyvíja. Prvá verzia niesla názov *AngularJS* a bola vyvinutá jazykom JavaScript. Postupom času bola

vydaná nová verzia, ktorá zmenila celý koncept frameworku. Medzi najdôležitejšie zmeny, ktoré priniesla bola zmena názvu na *Angular2* a vývojového jazyka na *TypeScript* (skrátene *TS*). [4]

Štruktúra

Angular2 (synonymum *Angular*) je založený na komponentoch, ktorý sa vytvára pomocou troch súborov. Prvým je súbor obsahujúci šablónu HTML, ktorá stanovuje štruktúru vykresleného komponentu. Druhým je súbor obsahujúci TS, ktorý definuje celú logiku komponentu. Posledný súbor obsahuje kaskádové štýly, ktoré sa starajú o vizuálnu stránku.

S príchodom TS sa v Angulari objavil nový koncept nazvaný *dekorátor*. Dekorátor je funkcia vyvolaná symbolom `@`, ktorá poskytuje informácie o spracovaní tried, metód, alebo vlastností počas behu aplikácie. Dekorátor `@Component` identifikuje triedu, nad ktorou sa nachádza ako triedu komponentu. Tejto triede sú následné nastavené všetky potrebné súbory k vykresleniu. [5]

Proces vytvárania projektu môže byť časovo náročný z dôvodu robustnej sady závislostí, ktoré Angular obsahuje. V tomto prípade je možné využiť nástroj *Command Line Interface* (skrátene *CLI*), ktorý uľahčuje proces vytvorenia a spracovania nového projektu.

1.1.2 Framework Vue.js

Vue.js je predstavovaný ako progresívny JavaScriptový framework. Toto označenie dostal za jeho ľahkosť, všestrannosť a udržateľnosť pri rozsiahlom projekte. Zaujímavosťou tohto frameworku je, že vznikol ako *open-source* projekt finančne podporovaný komunitou vytvorenou počas propagácie.

Štruktúra

Pri vytváraní projektu je odporúčané použiť nástroj *Vue CLI*, ktorý vytvára základnú štruktúru projektu. Taktiež umožňuje vývojárovi nastaviť najpoužívanejšie závislosti, ktoré sa stiahnu a nainštalujú do projektu v priebehu generovania. Architektúra Vue.js je postavená na komponentoch, ktoré získavajú vlastnosť znovu použitia a tým sa znižuje duplicitnosť kódu. Komponenty je možné štruktúrovať využitím konceptu *Single-File Components* (skrátene *SFC*). Tento koncept rozdeľuje komponent v jednom súbore na tri menšie komponenty. Prvým je `<template>` obsahujúci šablónu HTML. Druhým je `<script>` kde sa nachádza celá logika pre danú komponentu a posledným je `<style>`, ktorý obsahuje kaskádové štýly CSS a tým zabezpečuje vizuálnu stránku. [6]

Dôležitou vlastnosťou Vue.js je použitie direktív. Tie sa používajú pre úpravu atribútov šablóny HTML a začínajú sa písmenom `v-`. Príkladom môže byť `v-if`, ktorá umožňuje vykreslenie prvku HTML na základe splnenej podmienky. Taktiež je možné rozšíriť podmienku o ďalšie pomocou direktív `v-else` a `v-else-if`.

1.1.3 Zhrnutie frontendových frameworkov

V tejto časti budú spomenuté pozitívne a negatívne vlastnosti oboch frameworkov. Vue.js ponúka podrobnú dokumentáciu, ktorá napomáha hlbšiemu poznaniu frameworku a práce s jednotlivými vlastnosťami. Z pohľadu vytvárania komponentov má veľkú výhodu hlavne vďaka konceptu SFC. Pre Vue.js existuje vývojoví nástroj, ktorý je možné pridať do webového prehliadača a v prípade potreby umožňuje ručné upravovať komponenty. Aj keď je dodávaný s jazykom JavaScript je možné pridať rozšírenie TS. Veľkou nevýhodou Vue.js je komunitný vývoj, ktorý môže viesť k vzniku anomálií.

Veľkou výhodou frameworku Angular je, že za vznikom stojí spoločnosť Google, ktorá taktiež zabezpečuje zaškoľovanie nových vývojárov. Vo verzii 2 je programátor obmedzený len na použitie TS. Angular sa primárne odporúča používať pri väčších projektoch a v tímovej spolupráci. Porovnaniu frameworkov Vue.js a Angular je venovaná štúdia [7], v ktorej sa autor zameriava na hlavné rozdiely a rýchlostné testy.

1.2 Backend webovej aplikácie

Backend webovej aplikácie je to čo užívateľ nevidí. Je stavebným blokom zabezpečujúcim celú logiku webovej aplikácie. Vo väčšine prípadoch býva v kombinácií s databázou, v ktorej sa uchovávajú informácie od užívateľa. Taktiež je na tejto strane vytvorené API umožňujúce získať uchované informácie a zaslať ich na stranu frontendu.

1.2.1 Framework Flask

Flask je mikroframework postavený na programovacom jazyku Python, ktorý umožňuje vytváranie webových aplikácií. Podľa oficiálnej dokumentácie [8] slovo *mikro* znamená, že Flask si kladie veľký dôraz na udržanie čo najjednoduchšieho jadra so závislosťami nevyhnutnými pre vývoj webových aplikácií. Pre Flask je vytvorených veľké množstvo externých závislostí, ktoré sú spravované správcom balíčkov `pip`. Po pridaní vytvárajú abstrakciu nad určitou operáciou, ktorá sa v základe nenachádzala, alebo ich chceme nahradiť.

Vytvorenie aplikácie

Syntax Flasku je veľmi jednoduchá, a preto je možné vytvoriť celú webovú aplikáciu v jednom súbore. Tento spôsob sa moc často nepoužíva a namiesto toho sa rozdeľuje kód do viacerých súborov, ktoré sú logicky zlúčené do adresárov. Flask vyžaduje aby boli v projekte vytvorené dve zložky s názvom *templates* a *statics*. Templates obsahuje všetky používané šablóny - súbory s koncovkou `.html`. Narozdiel od toho zložka statics zahŕňa všetky položky používané v šablónach. Konkrétne sa jedná o CSS, JavaSript a obrázky.

Flask je v základe dodávaný so šablónovacím jazykom *Jinja2* používaným k vytvoreniu HTML. To umožňuje získať dynamické dáta a vykresliť ich na stranu klienta. Jinja2 používa špeciálne oddeľovače, do ktorých sa vkladajú premenné poprípade určitá logika využívajúca napríklad nekonečný cyklus. Oddeľovač `{{....}}` sa používa k vykresleniu na šablónu.

Vo výpise 1.1 je demonštrovaná ukážka vytvorenia minimálnej aplikácie, ktorá vo webovom prehliadači vypíše správu *Hello there!*. Dôležitou súčasťou je dekorátor `@app.route('/')`, ktorý sa viaže na konkrétnu funkciu a vytvára pre ňu *Uniform Resource Locator* (skrátene *URL*). Po zadaní URL do webového prehliadača je vyvolaná logika funkcie.

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route("/")
6  def hello_there():
7      return "<h1>Hello there!</h1>"
```

Výpis 1.1: Ukážka webovej aplikácie Flask.

1.2.2 Framework Laravel

Laravel je webový framework postavený na skriptovacom jazyku PHP. Je založený na architektonickom návrhu *Model-View-Controll* (skrátene *MVC*). *Model* je komponent zaisťujúci komunikáciu s databázou. *View* zaisťuje vykreslenie UI aplikácie obsahujúcej HTML, CSS a JavaScript. Posledný z komponent je *Controller*, ktorý zabezpečuje komunikáciu medzi komponentami Model a View. Existuje široká škála externých závislostí, ktoré môžu byť pridané do vytvoreného projektu pomocou správcu balíčkov `composer`.

Vytvorenie aplikácie

Oficiálna dokumentácia Laravelu [9] poskytuje detailný popis vytvorenia projektu pomocou inštalácie Laravelu, alebo pomocou nástroja `composer`. Po vytvorení projektu je automaticky vygenerovaná štruktúra obsahujúca predvytvorené adresáre. Najdôležitejšie adresáre sú pomenované ako *Models* a *Controllers*. Models obsahuje celú logiku aplikácie ako napríklad práca s databázou. Vytváranie nových modelov je možné za pomoci príkazu `make:model`. Narozdiel od toho Controllers obsahuje požadované ovládače - spojenie medzi modelom a pohľadom. Vytvorenie nových ovládačov je možné pomocou príkazu `make:controller`. Pre doplnenie informácií o všetkých adresároch viz. [10].

1.2.3 Zhrnutie backendových frameworkov

Táto časť je určená k porovnaniu spomenutých frameworkov. Medzi dôležité aspekty porovnania bude patriť možnosť práce s databázou, možnosť štruktúrovania väčšieho projektu a dostupnosť nástrojov k integrácií knižníc napísaných v jazyku C.

Flask je mikroframework a z toho dôvodu v základe neobsahuje potrebnú abstrakciu k práci s databázou, ale použitím nástroja `pip` je možné jednoducho pridať potrebnú závislosť. Vhodný je pre prácu s menšími projektami, pretože neobsahuje striktné pravidla pri štruktúrovaní čo môže viesť k mnohým duplicitám a neprehľadnosti. Flask umožňuje integrovanie knižníc jazyka C vďaka štandardnej knižnici `ctypes`.

Laravel sa preslávil hlavne vďaka kvalitnému nástroju určenému k práci s databázou, ktorý sa nachádza v základe frameworku bez nutnosti inštalácie. Vhodný je pre väčšie projekty a to vďaka predvytvoreným adresárom, ktoré sú logicky štruktúrované. Veľkou nevýhodou je, že neposkytuje možnosť integrácie knižníc jazyka C.

2 Analýza technológií

Táto kapitola predstavuje možné spôsoby integrácie knižníc jazyka C do webových aplikácií a najpopulárnejšie bezdrôtové komunikačné technológie, ktoré sú dostupné v mobilných zariadeniach.

2.1 Integrácia knižníc jazyka C

V rámci tejto práce bude nutné integrovať knižnicu napísanú v jazyku C do webovej aplikácie. Táto podkapitola predstaví dva možné spôsoby integrovania.

2.1.1 Ctypes

Ctypes je štandardná knižnica jazyka Python poskytujúca pracovné nástroje ku komunikácii s knižnicou napísanou v jazyku C. Poskytuje tzv. *marshalling*[1], ktorý zaisťuje kompatibilitu dátových typov pri komunikácií pretože Python a C uchovávajú dáta rozličným spôsobom. Dobrým príkladom je použitie celých čísiel kde Python umožňuje ukladanie veľkých čísiel naproti tomu C musí presne špecifikovať veľkosť. [12]

Vo výpise 2.1 sa nachádza ukážková funkcia napísaná v jazyku C, ktorá na vstupe získa dve celé čísla a vykoná na nich matematickú operáciu sčítania. Následne je vrátený výsledok v celom čísle.

```
1  int sum(int x, int y) {
2      return x + y;
3  }
```

Výpis 2.1: Ukážka funkcie napísanej v jazyku C (názov súboru sum.c).

Po vytvorení požadovaného kódu musí nasledovať kompilácia, ktorá vytvorí objekt požadovaný k založeniu zdieľanej knižnice. To je možné docieliť zadaním príkazu `gcc -shared -o lib.so -fPIC sum.c`[13] do terminálového okna. V rámci projektu musí byť knižnica pridaná pomocou príkazu `import ctypes`. Vo výpise 2.2 sa nachádza ukážka volania funkcie zo zdieľaného súboru, ktoré je možno rozdeliť do troch krokov [14]:

- **Načítanie knižne** vytvorením inštancie triedy `ctypes.CDLL()` kde formálnym parametrom je cesta ku zdieľanej knižnice.
- **Zabalenie** vstupných parametrov na požadovaný dátový typ, ktoré sa nastavujú pomocou metódy `argstypes`.

[1] Marshalling je proces prevodu reprezentácie objektu v pamäti do iného formátu vhodného na uloženie alebo prenos do iných softvérových aplikácií[11].

- **Oznámiť** ctypes návratový dátový typ funkcie pomocou metódy `restype`.

```
1    import ctypes
2
3    x, y = 2, 5
4    lib = ctypes.CDLL('lib.so')
5    lib.argtypes = [ctypes.c_int, ctypes.c_int]
6    lib.restype = c_int
7    result = lib.sum(x, y)
8    }
```

Výpis 2.2: Ukážka použitia knižnice ctypes.

2.1.2 WebAssembly

WebAssembly (skrátene *Wasm*) je nový typ programovacieho jazyk, ktorý umožňuje spustenie výpočtovo náročných aplikácií vo webovom prehliadači. Aj keď sa jedná o programovací jazyk, tak jeho primárne využitie nieje pi vytváraní nového kódu, ale aby umožňoval skompilovať kód programovacieho jazyku C, C++, Rust apod. Wasm vznikol ako rozšírenie k jazyku JavaScript, ktorému poskytuje skompilovaný kód iného jazyku vo forme knižnice. Toto rozšírenie vypĺňa medzeru JavaScriptu, ktorou je výkon. [15]

Emscripten je *open-source* nástroj umožňujúci kompilovanie kódu napísaného v C a C++, ktorý je možno spustiť za pomoci JavaScriptu vo webovom prehliadači. Neexistujú obmedzenia na vstupný obsah a preto je možné skompilovať do Wasm výkonné hry, zvukové stopy, rozsiahle programy pracujúce s vykreslovaním grafiky apod. Po nainštalovaní je nástroj dostupný z terminálového okna pod príkazom `emcc`, alebo `em++`. [16]

Kompilácia môže byť realizovaná viacerými spôsobmi. Najjednoduchším je, že okrem súboru `wasm` vytvorí šablónu HTML, ktorá sa použije v rámci testovania vzniknutého súboru obsahujúceho JavaScript. Tento spôsob je možné realizovať zadaním príkazu `emcc sum.c -o sum.html`, kde názov `sum.c` odkazuje na kód nachádzajúci sa vo výpise 2.1. Následne stačí zobraziť súbor `sum.html` prostredníctvom protokolu HTTP vo webovom prehliadači. [17]

2.2 Bezdrôtové komunikačné technológie

V tejto podkapitole budú predstavené najznámejšie bezdrôtové technológie zabezpečujúce prenos dát medzi zariadeniami. Použitie bezdrôtových technológií má za cieľ zvýšiť produktivitu, zjednodušiť a zrýchliť prenos dát.

2.2.1 Quick Response kód

Quick Response (skrátene *QR*) kód, je typ maticového kódu skladajúci sa z dvoch farebne odlišných blokov - najčastejšie používané farby *bielej* a *čiernej*. Na rozdiel od štandardného čiarového kódu umožňuje dvojsmerné čítanie čo má za následok zápis objemnejšieho množstva dát. QR kódy sa rozdeľujú do verzií, ktoré ovplyvňujú veľkosť výsledného obrazcu a maximálny počet uložených dát. [18]

QR kódy sa môžu vytlačiť na papier, poprípade zobraziť v mobilnom zariadení a k získaniu obsahu je potrebná čítačka, ktorá sa v dnešnej dobe nachádza skoro v každom zariadení s fotoaparátom. V prípade chýbajúcej podpory fotoaparátu v mobilnom zariadení je možné stiahnuť a použiť aplikáciu tretích strán.

Ľudské oko nedokáže rozpoznať dáta nachádzajúce sa v QR kóde. Preto je dôležité skenovať dáta od overených zdrojov [19]. Vo vytvorenom obrazci sa môžu nachádzať podvrhnuté URL, ktoré po naskenovaní presmerujú užívateľa na tzv. *phishingovú* webovú stránku. Existujú webové stránky, ktoré po navštívení automaticky stiahnu obsah.

2.2.2 Bluetooth Low Energy

Bluetooth je bezdrôtová technológia, ktorá sprostredkuje prenos dát medzi rôznymi zariadeniami. Od verzie 4.0 je dostupná v dvoch základných variantách, *Bluetooth Classic* a *Bluetooth Low Energy* (skrátene *BLE*) [20]. Tieto technológie majú odlišnú architektúru a nie sú navzájom kompatibilné - vystupujú ako dve nezávislé rádia. Dnešné zariadenia s podporou *Bluetooth* podporujú len jednu z variant, alebo oboje súčasne.

Ako názov napovedá BLE je nízko energitický variant Bluetooth. Z dôvodu šetrenia napájacieho zdroju prenáša malé množstvo dát. S BLE sú spojené dva dôležité pojmy, *GAP* a *GATT*. GAP zaisťuje viditeľnosť vysielaného zariadenia s obsahujúcimi informáciami použitými ku komunikácií medzi dvoma zariadeniami. Definuje viacero rolí, z ktorých dve role zabezpečujú spojenie. Konkrétne sa jedná o rolu [21]:

- **Periférnu** (*Slave*) - Zariadenia inzerujúce svoju prítomnosť a prijímajú spojenie od zariadenia s označením master, ktorý poskytuje pravidlá v rámci komunikácie. Jedná sa o malé zariadenie s nízkou spotrebou, ktoré majú obmedzené zdroje - napríklad hodinky moniturujúce srdcový tep.
- **Centrálnu** (*Master*) - Zariadenia zisťujúce prítomnosť dostupných slave zariadení, ktorým je ponúknuté zahájenie spojenia. Obvykle sa jedná o zariadenia s väčším výkonom oproti zariadeniam slave - napríklad mobilné zariadenie, tablety apod.

Slave zariadenie umožňuje zasielanie viacerým zariadeniam informáciu o svojej prítomnosti. Po naviazaní spojenia s jedným master zariadením sa informácie o

prítomnosti prestanú zasielať po dobu zaniknutia spojenia. *GATT* stanovuje spôsob výmeny informácií a dát prenášanými medzi dvoma spojenými zariadeniami. Skladá sa z dvoch rolí [20]:

- **Server** po naviazaní spojenia informuje klienta o svojich poskytovaných službách a vlastnostiach. Sprístupňuje odpovede na klientove požiadavky.
- **Klient** zasiela požiadavky k získaniu, alebo zápisu informácií na nachádzajúcich sa na serveri.

2.2.3 Near Field Communication

Near Field Communication (skrátene *NFC*) [22] je bezdrôtová komunikačná technológia, ktorá dosahuje krátku vzdialenosť - jednotky centimetrov. Z dôvodu malej prenosovej rýchlosti je vhodná pre prenos menšieho množstvo dát.

NFC umožňujú prácu v troch režimoch [22], *Card Emulation Mode*, *Reader/Writer Mode* a *Peer-to-peer Mode*. **Card Emulation Mode** dovoľuje zariadeniu s NFC fungovať ako čipová karta - mobilný telefón môže byť použitý ako náhrada kreditnej karty. **Reader/Writer Mode** umožňuje zariadeniu s NFC fungovať ako čítačka NFT značiek. **Peer-to-peer Mode** umožňuje vzájomnú komunikáciu medzi dvoma zariadeniami s podporou NFC - prenos dát a súborov.

Aplikácie používajúce komunikáciu pomocou technológie NFC možno rozdeliť do štyroch skupín [22]:

- **Touch and go** zahajuje prenos dát až po tom čo sa dve zariadenia dotknú.
- **Touch and confirm** zaisťuje, že komunikácia bude zahájená až po užívateľskom potvrdení použitím hesla.
- **Touch and connect** vytvára spojenie peer-to-peer medzi dvoma zariadeniami NFC, ktoré umožňuje prenos dát - napríklad fotografie, hudobnú stopu, telefónne číslo apod.
- **Touch and explore** umožňuje preskúmanie kapacity aplikácie, objavovať funkcie a navrhované služby.

3 Atribútová autentizácia

V dnešnom digitálnom svete existuje obrovské množstvo aplikácií, ale aj služieb vyžadujúcich osobné údaje o danom používateľovi. Z tohto dôvodu užívatelia ale aj nariadenia ako napríklad *GDPR* a *eIDAS* kladú dôraz na väčšie zabezpečenie ochrany súkromia a osobných dát.

Jedným z možných riešení ako sa vyhnúť prezradeniu nežiadúcich osobných údajov je použitie tzv. *atribútovej autentizácie*. Pri atribútovej autentizácií overovateľ neoveruje identitu užívateľa predkladaných dát, ale držanie a existenciu tzv. *atribútov* - vlastností. Atribúty môžu byť napríklad vek narodenia, splnená vakcinácia apod. Počas fáze overovania je možné nahradiť dátum narodenie užívateľa za atribút *"starší než 18 rokov"*. Vďaka tomuto mechanizmu sa overovateľ nedozvie dátum narodenia užívateľa, ale získa odpoveď splnil, alebo nesplnil. [23]

3.1 Vlastnosti atribútovej autentizácie

Medzi základne kryptografické vlastnosti atribútového autentizačného systému, ktoré zaručujú určitú úroveň ochrany patria [23]:

- **Anonymita** - Užívateľ odhaľuje len vybrané atribúty, ktoré neumožňujú identifikovanie užívateľa.
- **Nesledovatelnosť** - Zo strany vydavateľa nemôže vzniknúť profilovanie či sledovanie užívateľov.
- **Nespojiteľnosť** - Overovateľ nie je schopný priradiť overené atribúty užívateľovi - užívateľ sa v rámci jednej inštitúcie overí viackrát, ale overovateľ to nezistí.
- **Selektívne odhalenie atribútov** - Užívateľ musí byť schopný zvoliť atribúty, ktoré bude odhaľovať overovateľovi.
- **Neprenositeľnosť** - Digitálne poverenie je vytvorené pre konkrétnu osobu.
- **Nepadateľnosť** - Užívateľ nie je schopný vytvoriť platné digitálne poverenie.
- **Revokácia** - Pri strate atribútov existuje možnosť zneplatnenia užívateľa.

3.2 Entity atribútovej autentizácie

V atribútovom autentizačnom systéme sa obvykle vyskytujú štyri entity [23], ktoré zahrávajú rozličné úlohy.

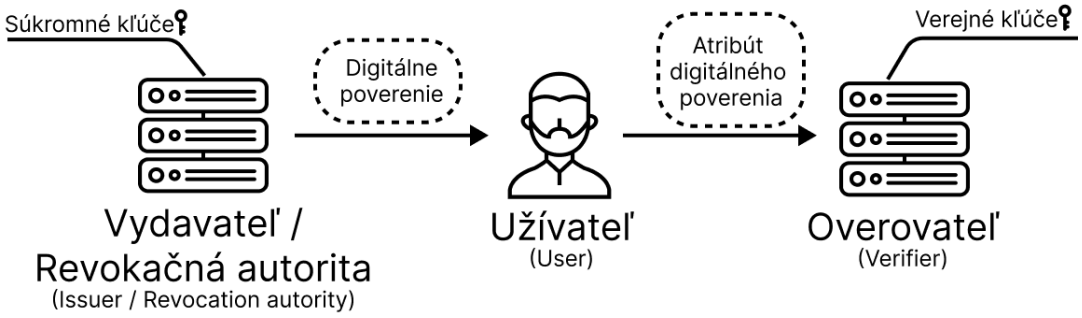
- **Vydavateľ** (angl. *Issuer*) vytvára digitálne poverenia užívateľom obsahujúce atribúty, ktoré sú jednotlivo podpísané súkromným kľúčom.
- **Revokačná autorita** (angl. *Revocation authority*) nie je povinnou entitou systému, ale vo väčšine systémov je žiadúcou. V prípade existencie má za úlohu

revokovať podozrivých užívateľov zo systému a v praxi býva často zlúčený s entitou vydavateľa.

- **Užívateľ** (angl. *User*) uchováva digitálne poverenie obsahujúce atribúty, ktorými sa anonymne preukazuje entite overovateľa.
- **Overovateľ** (angl. *Verifier*) overuje vlastníctvo poskytovaných atribútov od entity užívateľa pomocou jednotlivých verejných kľúčov atribútov získaných od entity vydavateľa.

3.3 Adopsio

Adopsio [23] je implementovaná schéma atribútového autentizačného systému vychádzajúca zo schémy *FKVAC* [24]. Na obrázku 3.1 je zobrazené schematické znázornenie použitej schémy. Entita vydavateľa je zlúčená s entitou revokačnej autority, ktorá vytvára digitálne poverenie obsahujúce podpísané atribúty, ktoré sú zaslané entite užívateľa. Následne užívateľ predkladá požadované atribúty entite overovateľa, ktorý ich overí.



Obr. 3.1: Schematické znázornenie atribútového autentizačného systému.

4 Digitálne certifikáty

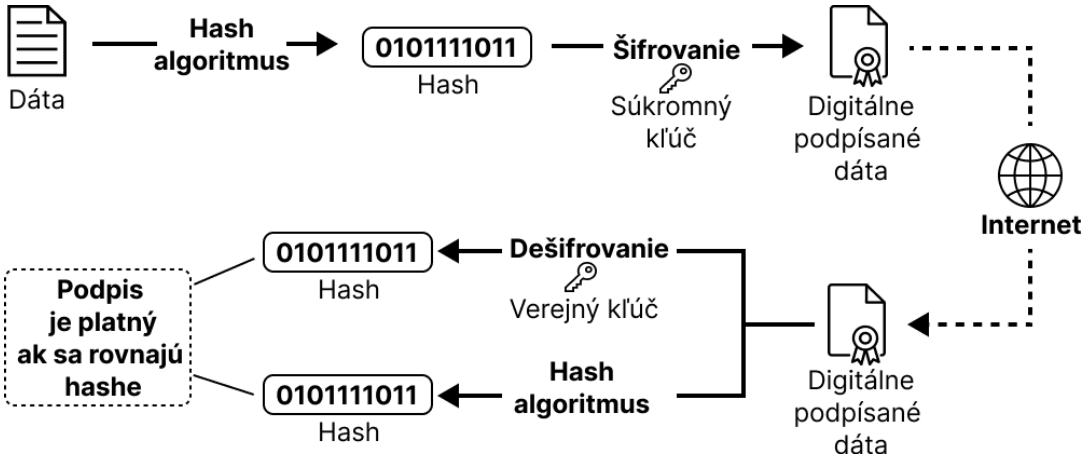
Covid certifikát vznikol za účelom preukázania držiteľa o jeho imunite, alebo o jeho bezinfekčnosti. Existuje v dvoch rozličných variantoch, *papierová* a *digitálna*. Preferovanejšia varianta je digitálna a to vďaka mobilných zariadení, ktoré sú súčasťou života každého človeka. Covid certifikáty podľa covid portálu Českej republiky[25] je možné rozdeliť do štyroch skupín, certifikát o:

- **absolvovaní očkovania**, ktorý je možné získať po očkovaní prvou, alebo druhou dávkou. Platný sa považuje po 14-tich dňoch od podania poslednej dávky.
- **negatívnom výsledku testov**, ktorý preukazuje, že osoba podstúpila test, ktorého výsledok bol negatívny. Na území Českej republiky je možné podstúpiť PCR, alebo AG test.
- **prekonaní ochorenia**, ktoré potvrdzujú, že osoba bola v minulosti infikovaná a vytvorila si dostatok protilátok. Certifikát nadobúda platnosť po 10-tich dňoch od pozitívneho PCR testu. Preukázanie o prekonaní ochorenia týmto certifikátom je možné po dobu 180-tich dní od prvého pozitívneho PCR testu.
- **kontraindikáciách** môže byť vygenerovaný osobám, ktoré zo zdravotných dôvodov nemôžu podstúpiť zaočkovaniu vakcínou. Tento typ certifikátu musí byť v kombinácií s certifikátom obsahujúci negatívny výsledok vykonaného PCR testu.

4.1 Digitálne podpisy

Pri používaní covid certifikátoch musí existovať bezpečnostný mechanizmus, ktorý by dokázal potvrdiť jeho pravosť poprípade odhaliť falzifikát. *Digitálny podpis* je typ *elektronického podpisu*, ktorý slúži k overeniu pravosti a integrity digitálnych dokumentov. Vytvára digitálny otlačok, ktorý je pre každého užívateľa jedinečný a vďaka tomu umožňuje identifikovať zasielateľa. Digitálny podpis zaisťuje digitálnemu dokumentu *nepopierateľnosť*, *integritu*, *dôvernosť* a *autenticitu*. Pri práci s digitálnymi podpismi je využitá *asymetrická kryptografia*, ktorá sa vyznačuje používaním páru kľúčov - *verejný* a *súkromný* kľúč. Zvyčajne sa používa v kombinácií s *hašovacími funkciami*, ktorý zaisťuje jedinečný reťazcoví identifikátor o pevnej dĺžke[26].

Na obrázku 4.1 je schematický znázornený proces vytvorenia a overenia digitálneho podpisu. Užívateľ zoberie vstupné dáta, z ktorých pomocou hašovacieho algoritmu vytvorí textový reťazec zložený z čísel a písmen. Digitálny podpis vznikne pomocou súkromného kľúču, ktorým sa podpíše hašový reťazec. Následne sú zaslané dáta s podpísaným hašom strane overovateľa, ktorý zo získaných dát vypočíta haš a porovná ho s podpísaným. Ak sú zhodné oba haše tak digitálny podpis je platný.



Obr. 4.1: Schematické znázornenie digitálneho podpisu.

4.2 Digitálne certifikáty preukazujúce bezinfekčnosť

Veľa organizácií po celom svete sa rozhodlo vyvíjať systémy digitálnych certifikátov. Aj keď väčšina organizácií používa k overovaniu QR kódy, z hľadiska spôsobu overovania legitímnosti a platnosti certifikátov boli použité rozdielne prístupy. Taktiež boli použité odlišné technológie na vybudovanie infraštruktúry zdieľania informácií napomáhajúce overenie certifikátoch v iných štátoch [27].

Overovanie platnosti certifikátu používa technológiu QR kódu, kde overovateľovi stačí naskenovať kód, po ktorom uvidí informácie o type certifikátu a držiteľovi. Infraštruktúra je postavená na technológií *blockchain*.

4.2.1 EU Digital COVID Certification

Skrátene EUDCC, je digitálny certifikát vytvorený *Európskou Úniou*, ktorý má za cieľ umožniť volný pohyb v rámci celej EU počas pandémie. Certifikát sa týka všetkých spomenutých typov certifikátov. [27]

Vydaný je v digitálnej, alebo papierovej verzii, ktorý obsahuje základné informácie o držiteľovi spolu s QR kódom, ktorý je podpísaný digitálnym podpisom. K vzájomnej komunikácií bola vytvorená brána, ktorá umožňuje overenie certifikátov na základe PKI v celej EU.

Tečka/Čtečka

Česká republika bola medzi prvými krajinami v Európe, ktorá povolila spustenie digitálnych certifikátov vydané EU. Za týmto účelom vznikli dve mobilné aplikácie *Tečka* a *čTečka*. Tečka by sa dala prirovnať k digitálnej peňaženke, do ktorej je možné vkladať všetky druhy certifikátov povolené Českou republikou. Aplikácia sama o

sebe neobsahuje dáta, ale umožňuje zobraziť QR kód, v ktorom sa dáta nachádzajú. Dáta sú uložené vo formáte JSON, v ktorých sa nachádzajú informácie ako *meno a priezvisko*, *dátum narodenia*, *informácie o očkovaní* a *podpis certifikátu*. [23]

ČTečka je aplikácia určená k overeniu platnosti digitálnych certifikátov. K funkčnosti vyžaduje fotoaparát, ktorý umožní naskenovanie QR kódu. Po prečítaní informácií získaných z QR kódu overí platnosť digitálneho podpisu a výsledok spolu s ostatnými informáciami zobrazí na displeji.

Tento spôsob overenia prináša mnoho technických problémov. Pri overovaní sa overuje platnosť vystaveného digitálneho certifikátu, ale nie identitu držiteľa. To môže viesť k zdieľaniu jedného platného certifikátu medzi viacerými ľudmi. Z dôvodu, že digitálne podpisy sú postavené na asymetrickej kryptografii, tak pri odcudzení súkromného kľúča je možné vytvárať si vlastné platné certifikáty.

5 Návrh webových aplikácií

Táto kapitola sa zameriava na návrh webových aplikácií. V prvej podkapitole budú popísané požiadavky na jednotlivé aplikácie 5.1, ktoré vychádzajú z vlastnosti entít. Na základe požiadavok bude popísaný návrh architektúr 5.2 a použitých technológií. Spôsob autentizácie oboch aplikácií bude predstavený v podkapitole 5.3, za ktorou nasleduje stručný popis návrh dátového modelu 5.4. Na samom konci tejto kapitoly bude predstavený návrh UI 5.5.

5.1 Požiadavky na webové aplikácie

Táto podkapitola sa bude venovať požiadavkám potrebným pre návrh a implementáciu webových aplikácií pre entity atribútového systému. Konkrétne sa jedná o entity *vydavateľa* a *overovateľa*. Požiadavky vychádzajú zo základných funkcií popísaných v kapitole 3.

Vydavateľ

Medzi hlavné požiadavky patrí možnosť vytvárať, zobrazovať a odstraňovať digitálne certifikáty a taktiež vakcíny. Musí umožniť registráciu a správu nových užívateľských účtov, ktoré budú môcť vstupovať do systému na základe platnej autentizácie a autorizácie. Užívateľom bude umožnené zobraziť, alebo stiahnuť pre nich vytvorené digitálne certifikáty. Všetky potrebné informácie o certifikátoch a existujúcich užívateľoch budú uchované v databázovom systéme. Pre atribútový autentizačný systém bude nutné pridať možnosť generovania nových systémových parametrov a taktiež kryptografických kľúčov.

Overovateľ

Musí graficky umožňovať výber atribútov určených k overeniu digitálneho certifikátu. Jednotlivé overenia budú uchované v logovacom súbore, ktorý bude zobrazený na GUI vo forme tabuľky. Z dôvodu bezpečnosti bude pridaná možnosť zapnutia a vypnutie skriptu bežiaci na pozadí, ktorý zabezpečuje celú logiku overenia. Rovnako ako *vydavateľ* bude do aplikácie pridaný autentizačný systém, ktorý zamedzí prístup neovereným užívateľom.

5.2 Architektúra webových aplikácií

Podkapitola je venovaná logickému prepojeniu hlavných komponent webových aplikácií. Pre aplikáciu vydavateľa 5.2.1 a overovateľa 5.2.2 budú predstavené návrhy

architektúr. K obom návrhom bol zvolený podobný prístup, ktorý sa rozchádza pri použití dodatočných služieb a spôsobu nasadenia do produkcie. Zo začiatku budú predstavené komponenty, ktoré sú identické pre obe webové aplikácie.

Základným stavebným kameňom oboch webových aplikácií je mikroframework Flask, ktorý zohráva úlohu na pozadí serveru ako napr. prepojenie aplikácie s databázovým serverom, obsluhu služieb potrebných pre funkčnosť aplikácií. Stabilný beh webovej aplikácie zaisťuje aplikačný server *Green Unicorn* (skrátene *Gunicorn*). Gunicorn je *rozhranie brány webového serveru* (skrátene *WSGI*), ktorý vznikol za účelom spúšťania webových aplikácií napísaných v jazyku Python. Medzi jeho hlavné úlohy patrí správa pracovných procesov, ktoré ovplyvňujú súbežný počet požiadaviek a odpovedí zaslaných na server. V oboch aplikáciách sú nastavení štyria synchronní pracovníci - spracovávajú požiadavky jeden po druhom.

UI webových aplikácií je postavené na technológií Vue.js, ktorému sú venované samostatné podkapitoly. V podkapitole 5.5 bude predstavený návrh UI webových aplikácií. Implementácie bude popísaná v podkapitole 6.3.

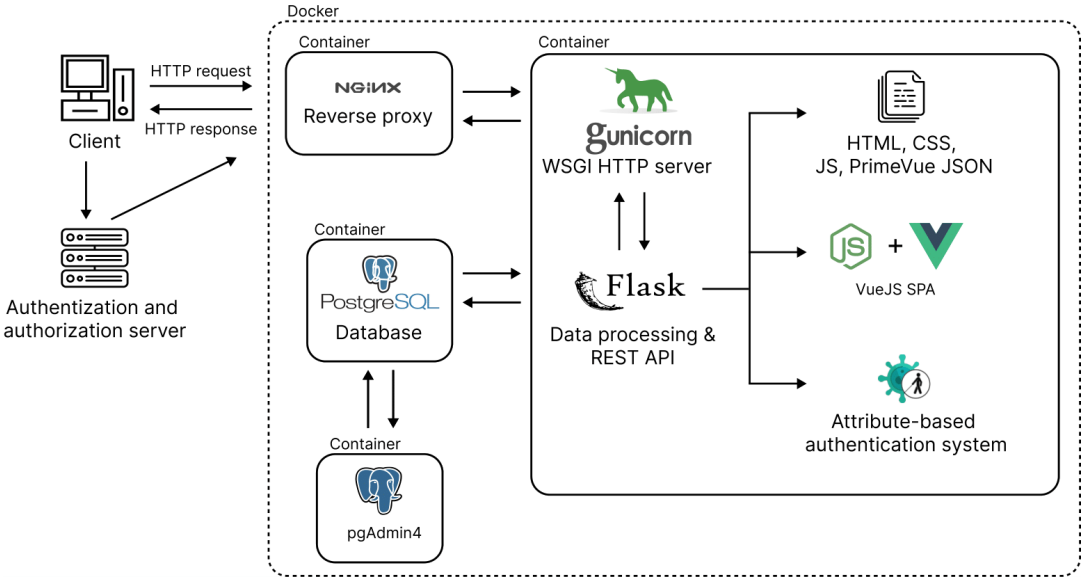
Všetká komunikácia medzi klientom a serverom je realizovaná pomocou REST API, ktorá využíva HTTP metódy. Detailnejší popis REST API sa nachádza v kapitole 1.

Databázový server oboch webových aplikácií je realizovaný pomocou *PostgreSQL*. PostgreSQL je systém pre správu relačných databáz (angl. *Object-Relational Database Management System*), ktoré sa používajú na ukladanie dát do tabuliek. Vo webovej aplikácií sa nachádza na strane serveru a priamo komunikuje s webovou aplikáciou napísanou Flaskom. Implementácia a komunikácia s frameworkom bude popísaná v sekcii 6.3. Návrh tabuliek pre jednotlivé webové aplikácie sa nachádza v samostatnej podkapitole 5.4.

Autentizácia užívateľov je poskytovaná službou tretej strany, vďaka ktorej nebude potrebné uchovávať prihlasovacie údaje v databáze. V oboch aplikáciach je autentizácia poskytovaná službou od spoločnosti *Google*, ktorej sa bude detailnejšie venovať v nasledujúcej podkapitole 5.3.

5.2.1 Architektúra vydavateľa

Webová aplikácie vydavateľa sa skladá z viacerých komponentov, ktoré sú obalené do tzv. *kontajnerov* (angl. *containers*). Hlavným predstaviteľom kontajnerov je softvér *Docker*. Docker je open-source softvér, ktorý sprostredkuje spustenie a údržbu aplikácií v izolovanom prostredí. Na obrázku 5.1 sa nachádza schematické prepojenie kontajnerov s obsahujúcimi komponentami. Pri práci s viacerými kontajnermi je použitý nástroj *docker-compose*.



Obr. 5.1: Schematické znázornenie webovej aplikácie vydavateľa.

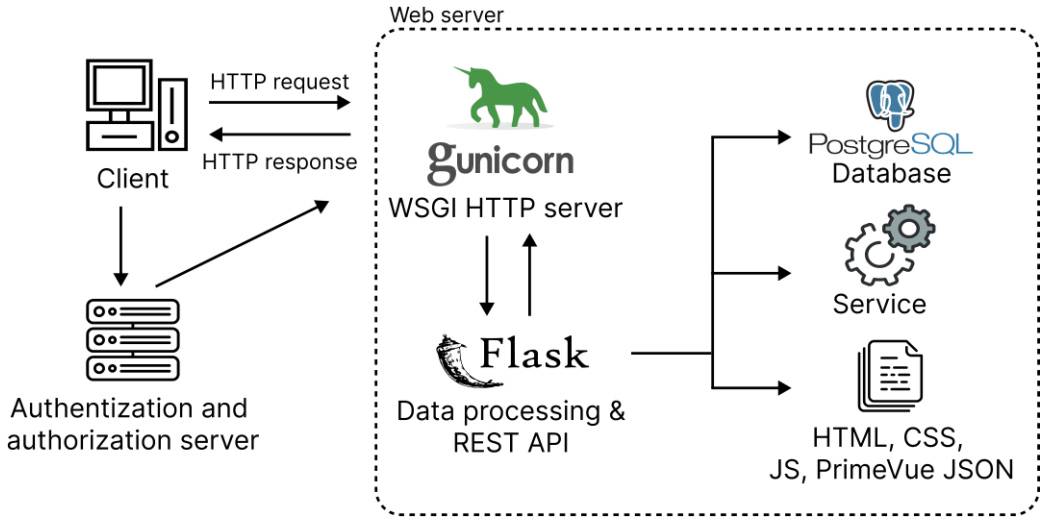
Požiadavky od klienta sú spracované webovým serverom a reverzným proxy *Nginx*, ktoré sú preposielané smerom k aplikačnému serveru. Na Nginx servery bolo nastavené doménové meno, pod ktorým sa skrýva celá webová aplikácia.

Do systému bol pridaný kontajner obsahujúci softvér *pgAdmin4*, ktorý umožňuje prácu s databázovým serverom v grafickom užívateľskom rozhraní (skrátene *GUI*).

Atribútový autentizačný systém sa nachádza na strane serveru. Z dôvodu používania kontajnerov a ich izolovaných vlastností, bolo potrebné z atribútového systému vytvoriť tzv. *docker obraz* (angl. *docker image*). Obraz dockeru je šablóna, podľa ktorej je možné vytvoriť kontajner. Po vytvorení základného obrazu boli do kontajnera pridané potrebné závislosti, balíčky a samotný kód webovej aplikácie.

5.2.2 Architektúra overovateľa

Na obrázku 5.2 sa nachádza schematické prepojenie hlavných komponentov webovej aplikácie overovateľa. Na rozdiel od aplikácie vydavateľa nevyužíva technológie kontajnerov, reverzného proxy a ani aplikáciu zaisťujúcu atribútový systém. Overovateľ umožňuje správu služieb, konkrétne sa jedná o aplikáciu overovateľa, ktorá je napísaná v jazyku *Java* a v systéme je nastavená ako služba.



Obr. 5.2: Schematické znázornenie webovej aplikácie overovateľa.

5.3 Autentizácia a autorizácia

V dnešnom svete väčšina moderných aplikácií vyžaduje, aby užívatelia overili svoju identitu. S týmto sú spojené dva základné pojmy. *Autentizácia* má za úlohu overiť identitu jednotlivca a *autorizácia* riadi prístup na základe pridelených rolí a oprávnení. Najbežnejší spôsob auntentizácie je overenie užívateľa na základe prístupového mena, poprípade e-mailu a hesla. Tento spôsob vyžaduje implementovanie vlastného autentizačného mechanizmu a správu prihlasovacích údajov čo môže viesť k rôznym kybernetickým útokom cielených na ich odcudzenie.

Autentizáciu oboch webových aplikácií zaisťuje služba poskytovaná spoločnosťou *Google* a autorizácia je realizovaná na základe pridelených rolí. V prípade vydavateľa je pridaná možnosť autorizácie na základe jednorázového API kľúča, konkrétne pre jeden koncový bod rozhrania REST API.

Autentizácia a autorizácia poskytovateľa služby

Autentizácia pomocou poskytovateľa služby je v dnešnom svete veľmi rozšírená, pretože sa zbavuje zodpovednosti za správu užívateľských prístupových údajov. Ďalšou možnou výhodou, je že si užívateľ nemusí vytvárať nový prihlasovací účet, ale môže použiť už existujúci od rozličných poskytovateľov - *Google*, *Github*, *Facebook* a iné.

Obe webové aplikácie prenechávajú celú časť autentizácie na strane poskytovateľa služby od spoločnosti *Google*. V dokumentácií od spoločnosti Google [28] sa nachádza detailný postup presmerovania autentizácie na stranu poskytovateľa a následné zaslanie autorizačného tokenu. Autorizačný token umožňuje webovej aplikácií komunikovať s poskytovateľom v mene užívateľa.

Overenie správnej autorizácie je na základe emailovej adresy užívateľa. Po overení je vytvorený cookies súbor, ktorý je uložený na strane klienta a pri každom dotazovaní na server je použitý k autorizácií.

Autorizácia na základe rolí

Po úspešnej autentizácii je dôležité určiť rozsah oprávnení pre overeného užívateľa. V tomto momente nastupuje autorizácia na základe rolí. Samotná autorizácia je realizovaná len pre webovú aplikáciu vydavateľa, v ktorej existujú tri role. Webová aplikácia umožňuje získanie len jedného typu role.

Prvý užívateľ, ktorý sa úspešne prihlási získa automaticky rolu *admin*. Rola *admin* má oprávnenie spravovať všetky užívateľské účty, ktoré sa nachádzajú v systéme. Operácie, ktoré môže vykonávať nad ostatnými užívateľmi je mazanie účtov, pri ktorej sa zmažú aj všetky existujúce digitálne certifikáty danej osoby a zmena role registrovaného užívateľa. Ak sa v systéme nachádza posledný užívateľ, automaticky mu bude zmenená rola na *admin*.

Ďalší prihlásený užívatelia obdržia rolu *user*. Užívateľ s rolou *user* má obmedzenia, ktoré mu umožňujú zobrazovať jeho vytvorené digitálne certifikáty a vygenerovať jednorázový token, na základe ktorého môže tieto certifikáty stiahnuť vo formáte JSON.

Ako posledná je rola *issuer*, ktorá je určená pre vydavateľov digitálnych certifikátov. Oprávnenia pre túto rolu sú najrozsiahlejšie, pretože umožňujú vytváranie a mazanie digitálnych certifikátov, spolu s vakcínami. Užívateľ s rolou *issuer* má taktiež oprávnenie generovať nové systémové parametre a kryptografické kľúče využívajúce k správnemu fungovaniu atribútového autentizačného systému.

Autorizácia pomocou API kľúču

Z požiadaviek uvedených v podkapitole 5.1 vyplýva, že webová aplikácia vydavateľa musí byť schopná umožniť užívateľom stiahnuť pre nich určené digitálne certifikáty. Dôležitou časťou tejto požiadavky je zabezpečiť aby overený užívateľ mohol pristupovať k svojim certifikátom - potrebné zaistiť autorizáciu.

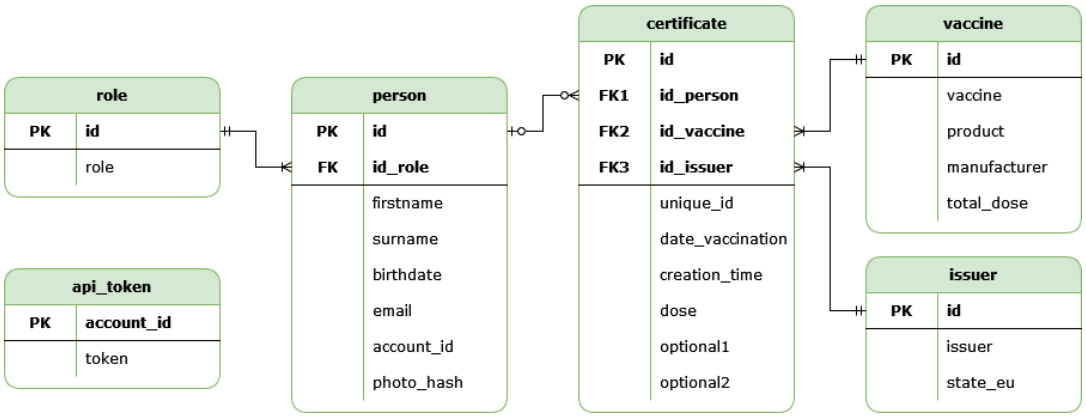
Riešením tejto požiadavky je použitie *API kľúča*. Prihlásenému užívateľovi s rolou user je pridaná možnosť generovania jednorázového API kľúča, ktorý je možné použiť pre získanie digitálnych certifikátov z koncového bodu API. Jednorázový API kľúč znamená, že po prvom použití sa zničí a k ďalšej práci je potrebné vygenerovať nový. Pri práci s API kľúčom je ho potrebné zabaliť do hlavičky požiadavku GET - parameter hlavičky `X-API-KEY`.

5.4 Dátový model

Dátový model popisuje vzťahy dátových prvkov a napomáha k porozumeniu práce s dátami. Možno ho rozdeliť do viacerých skupín, ktoré definujú logickú štruktúru - ako sa dáta logicky ukladajú. *Entity Relationship Diagram* (skrátene ERD) je typ vývojového diagramu, ktorý graficky zobrazuje entity a ich vzťahy v databázovom systéme. Aj keď nezahŕňa štandardy návrhu, je definovaná sada symbolov, ktoré sa pri návrhu využívajú ako sú obdlžníky, kosoštvorce a spojovacie čiary. Pri návrhu ERD bol využitý onlinový softvér *draw.io*, ktorý umožňuje vytvárať vývojové diagramy, diagramy UML a pod.

5.4.1 ER diagram vydavateľa

Vo webovej aplikácií vydávateľa sa nachádzajú viaceré entity: *person*, *certificate*, *vaccine*, *issuer*, *role* a *api_token*. Entity a vzťahy medzi nimi sú zobrazené na obrázku 5.3.



Obr. 5.3: ERD vydavateľa.

Role

Typová tabuľka obsahujúca všetky dostupne role, ktoré sú použité pri autorizácií užívateľov. Skladá sa z primárneho kľúča *id* a so stĺpca *role*, ktorého dátový typ je textový reťazec a obsahuje názvy rolí.

Person

Do tabuľky *person* sú vkladané základné údaje o užívateľovi, na základe ktorých sú vytvárané digitálne certifikáty. Záznam v tabuľke vznikne registráciou užívateľa.

Stĺpce *account_id* a *email* sú získané pomocou autorizačného tokenu od poskytovateľa. Je vo vzťahu N:1 s tabuľkou *role* - jednému užívateľovi je priradená konkrétne jedna rola.

Vaccine

Tabuľka *vaccine* obsahuje informácie o existujúcich vakcínach.

Issuer

Tabuľka *issuer* obsahuje základné informácie o vydavateľovi digitálnych certifikátov.

Certificate

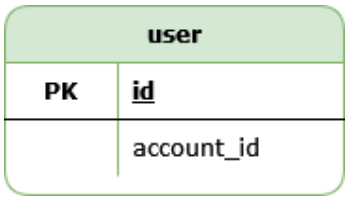
Tabuľka *certificate* obsahuje obsahuje informácie o vytvorenom digitálnom certifikáte. Vo vzťahu je s tabuľkami *vaccine*, *issuer* a *person*. Vzťah medzi jednotlivými tabuľkami je N:1, ktoré sú povinné až na vzťahu medzi tabuľkou person. Tento vzťah je nie je povinný a preto môže, ale aj nemusí existovať.

Api_token

Tabuľka *api_token* nie je vo vzťahu so žiadnou tabuľkou. Obsahuje vygenerované jednorázové kľúče, ktoré sú po použití automaticky odstránené. Primárnym kľúčom je jedinečný identifikátor užívateľského účtu získaného od poskytovateľa autorizácie.

5.4.2 ER diagram overovateľa

Na rozdiel od *vydavateľa* webová aplikácia overovateľa požaduje len uchovanie údajov o užívateľskom účte, ktorý sa prihlasuje do aplikácie. Pre tento dôvod sa v systéme nachádza jedna entita *user*, viz. obrázok 5.4.



Obr. 5.4: ERD overovateľa.

User

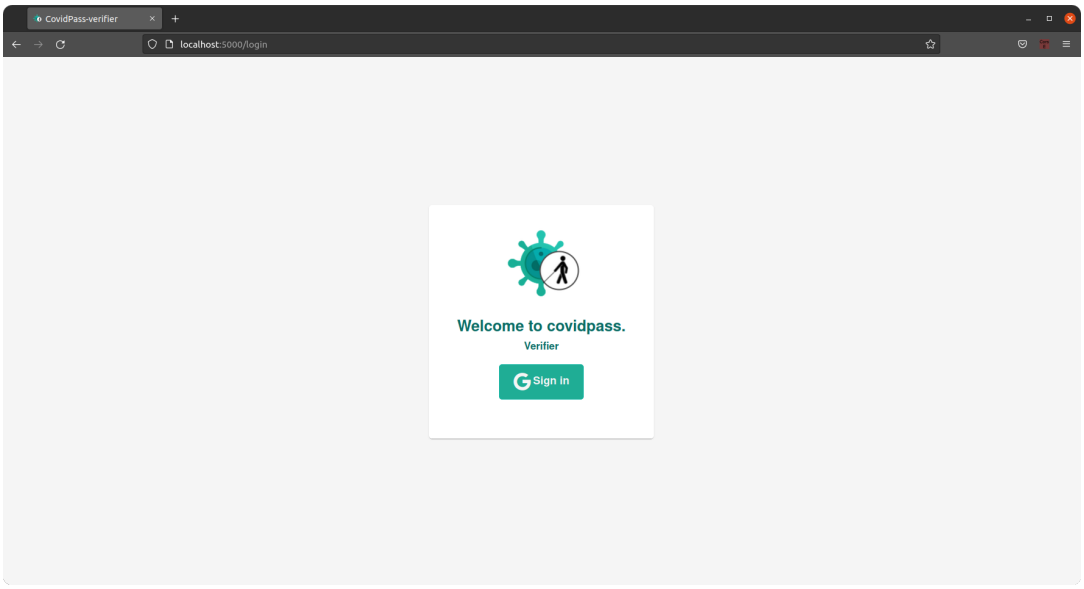
Tabuľka *user*, je obdoba tabuľky *person* vo webovej aplikácií vydavateľa. Obsahuje jedinečný identifikátor, ktorý je automaticky získaný z emailového účtu užívateľa po prvom prihlásení do webovej aplikácie.

5.5 Užívateľské rozhranie

V dnešnom digitálnom svete sa život stáva stále viac závislým na internete, webových aplikáciach a mobilných zariadeniach. V dôsledku toho sa vo väčšine prípadoch pri vývoji webových aplikácií zameriava hlavne na vytvorení atraktívneho a efektívneho UI, ktoré zaujme a priláka širokú škálu užívateľov. Proces návrhu a vývoja atraktívneho UI je časovo veľmi náročný. Z tohto dôvodu existujú dodatočné knižnice, ktoré obsahujú návrhárske šablóny napísané v jazykoch HTML, CSS a JavaScriptu. V prípade oboch webových aplikácií je použitá knižnica *PrimeVue*, ktorej sa detailnejšie bude venovať v podkapitole 6.3.

Užívateľské rozhranie overovateľa

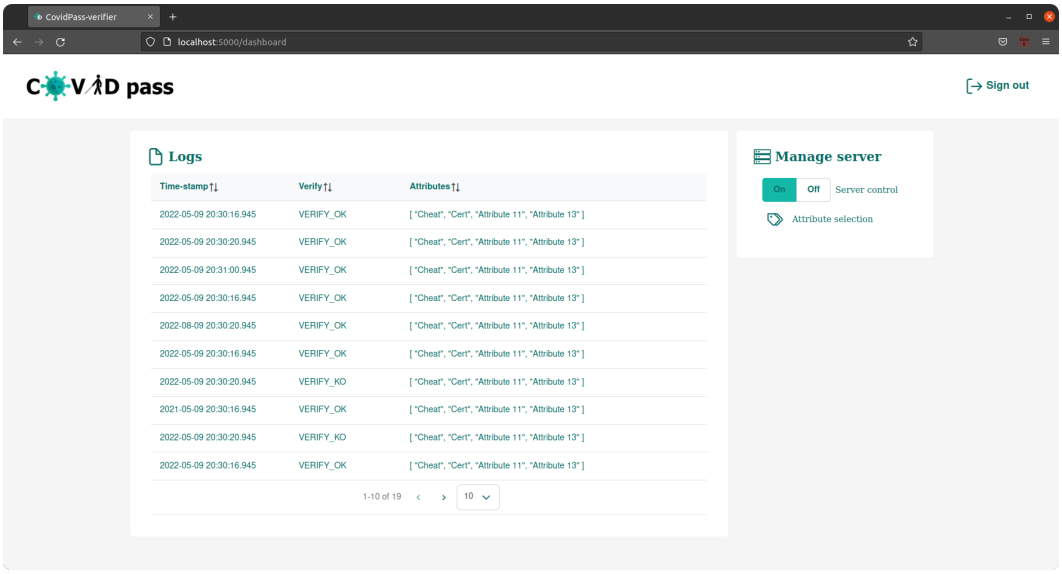
Keď príde užívateľ na webovú aplikáciu overovateľa je vyzvaný k prihláseniu do aplikácie, viz. 5.5 - zobrazí sa úvodná stránka s tlačidlom *Sign up*, ktoré po stlačení presmeruje užívateľa na prihlásenie sa pomocou google účtu.



Obr. 5.5: Prihlasovacia stránka overovateľa.

Po úspešnom prihlásení je zobrazená hlavná stránka celej aplikácie, viz. 5.6. Na tejto stránke sa nachádza tabuľka obsahujúca záznamové logy z činnosti overenia.

Práva časť stránky sa skladá z dvoch tlačidiel - prvé je typu *options*, ktoré umožňuje zapnutie a vypnutie aplikácie overovateľa a druhé tlačidlo je v podobe *ikony*, ktoré zobrazí *sidebar* s možnosťami výberu atribútov, ktoré budú požadované pri procese overenia.



Obr. 5.6: Hlavná stránka overovateľa.

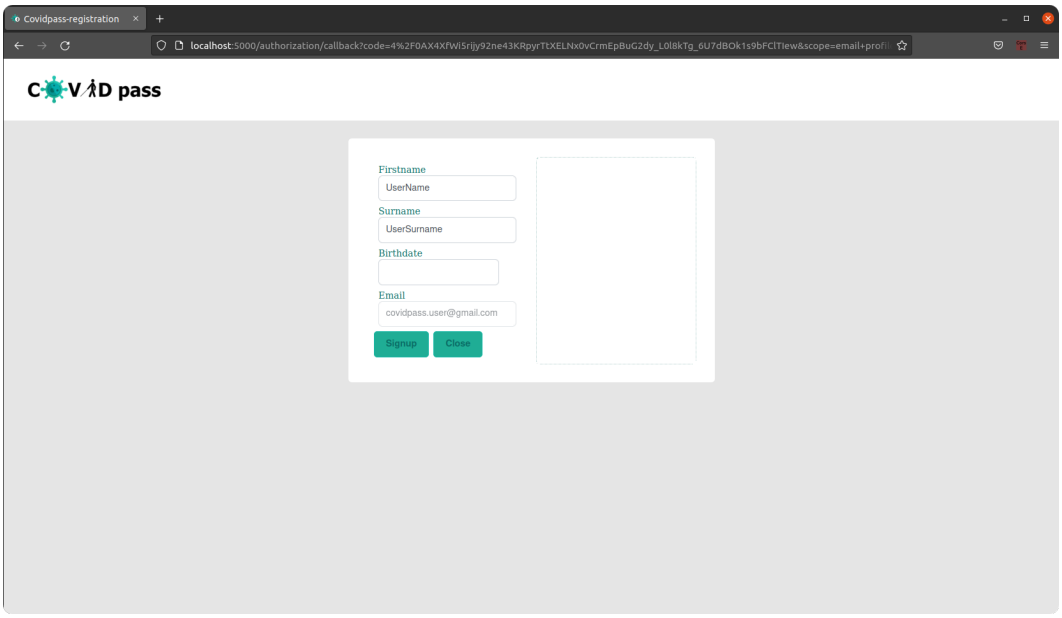
Užívateľské rozhranie vydavateľa

Keď príde užívateľ na webovú aplikáciu vydavateľa je rovnakým spôsobom ako na strane overovateľa vyzvaný k prihláseniu. Prihlasovacia stránka je identická ako na strane overovateľa 5.5 s malým rozdielom a to zmena podnadpisu *Issuer*.

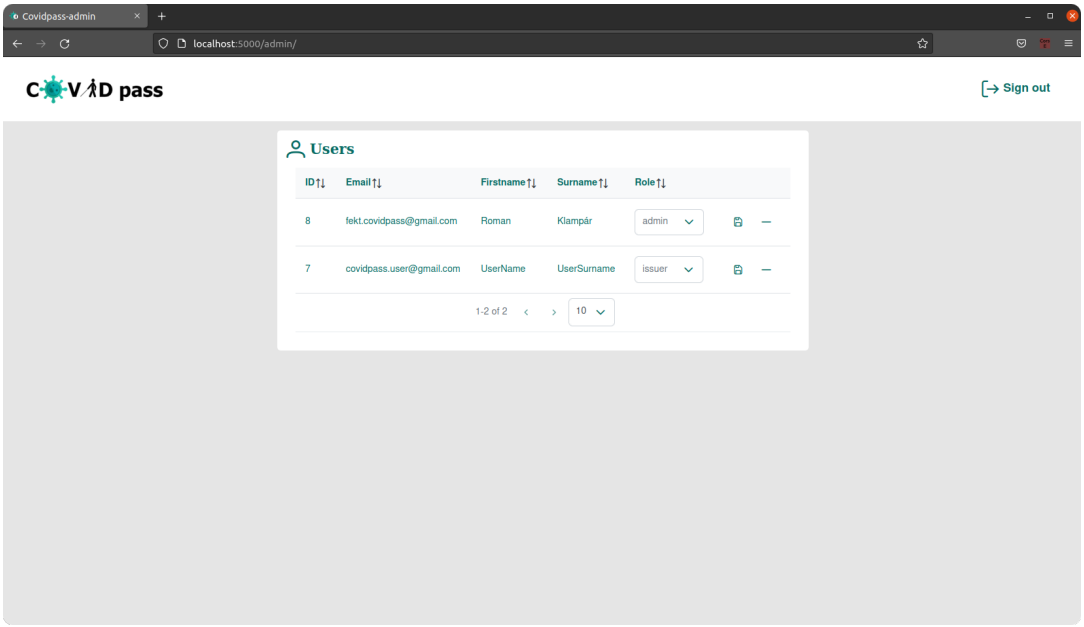
Po úspešnom autentizovaní sa aplikácia v databáze overí či existuje záznam o užívateľovi. V prípad chýbajúceho záznamu bude vyzvaný k registrácií, viz. 5.7. Niektoré údaje sú získané z účtu Google - meno, priezvisko a emailová adresa.

Prvý zaregistrovaní užívateľ získa rolu admin a následne mu je zobrazená stránka s tabuľkou, na ktorej sa nachádza zoznam všetkých aktívnych užívateľov v systéme. Tabuľka umožňuje odstrániť užívateľov, alebo im zmeniť role. Viz. 5.8.

Pri prihlásení do systému s rolou user je užívateľovi zobrazená stránka znázornená na obrázku 5.9. Stránka je rozdelená na dve hlavné časti. Na ľavej strane sa nachádza tabuľka, ktorá obsahuje všetky existujúce digitálne certifikáty prihláseného užívateľa. Po kliknutí na záznam z tabuľky je zobrazený sidebar, ktorý obsahuje všetky informácie o konkrétnom certifikáte. Na pravej strane sa nachádza box s dvoma tlačidlami - *QR code* a *Hex key*. Po stlačení oboch tlačidiel je zobrazený jednorázový token, ktorý sa zobrazuje v rôznej podobe na základe stlačeného



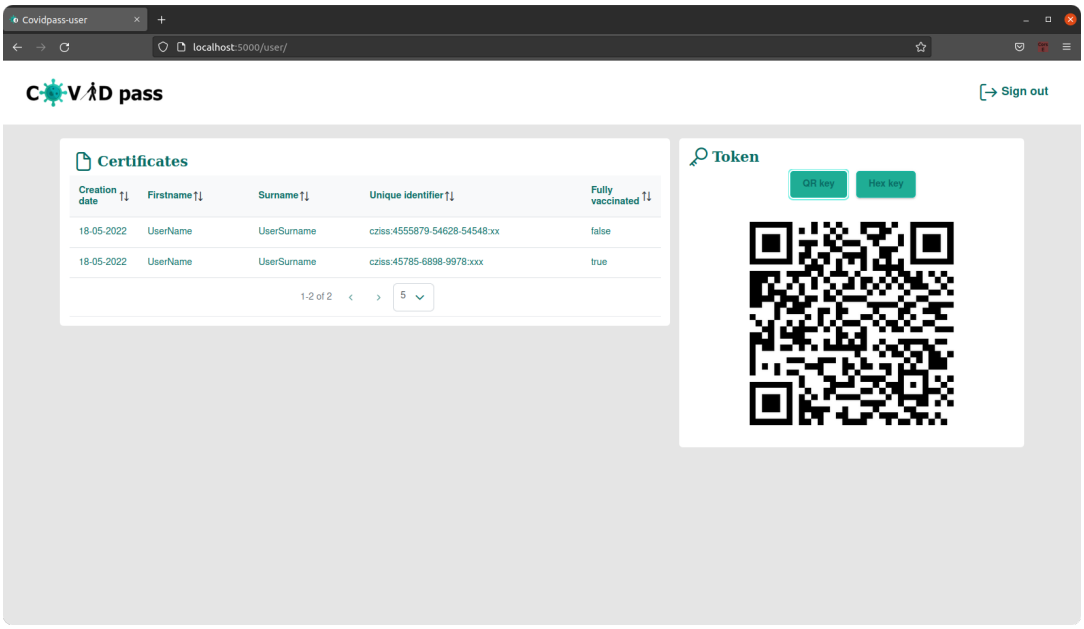
Obr. 5.7: Registrácia užívateľa vo webovej aplikácií vydavateľa.



Obr. 5.8: Admin sekcia vo webovej aplikácií vydavateľa.

tlačidla - QR code zobrazí QR kód obsahujúci token a Hex key, predstaví token v šestnástkovej sústave.

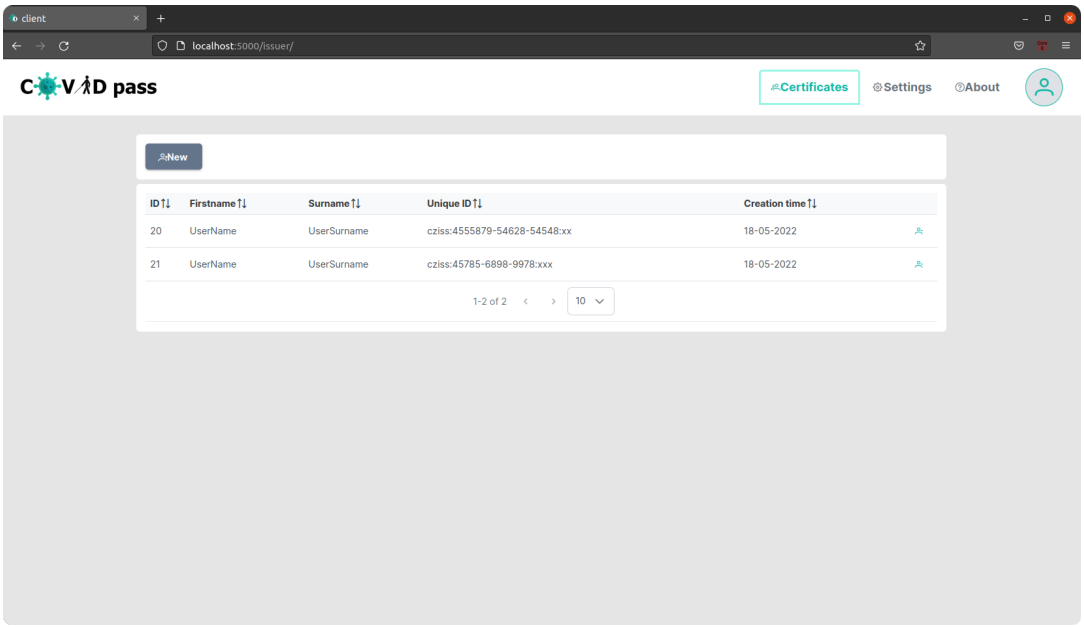
Po prihlásení užívateľa s rolou issuer je užívateľovi zobrazená tzv. *Single Page Application* (skrátene *SPA*). Je možné prepínať medzi dvoma stránkami. Úvodná stránka sa skladá z dvoch častí - hlavička s navigátorom a tabuľka, viz. 5.10. Tabuľka obsahuje záznam všetkých vytvorených certifikátov. Vo vrchnej časti tabuľky



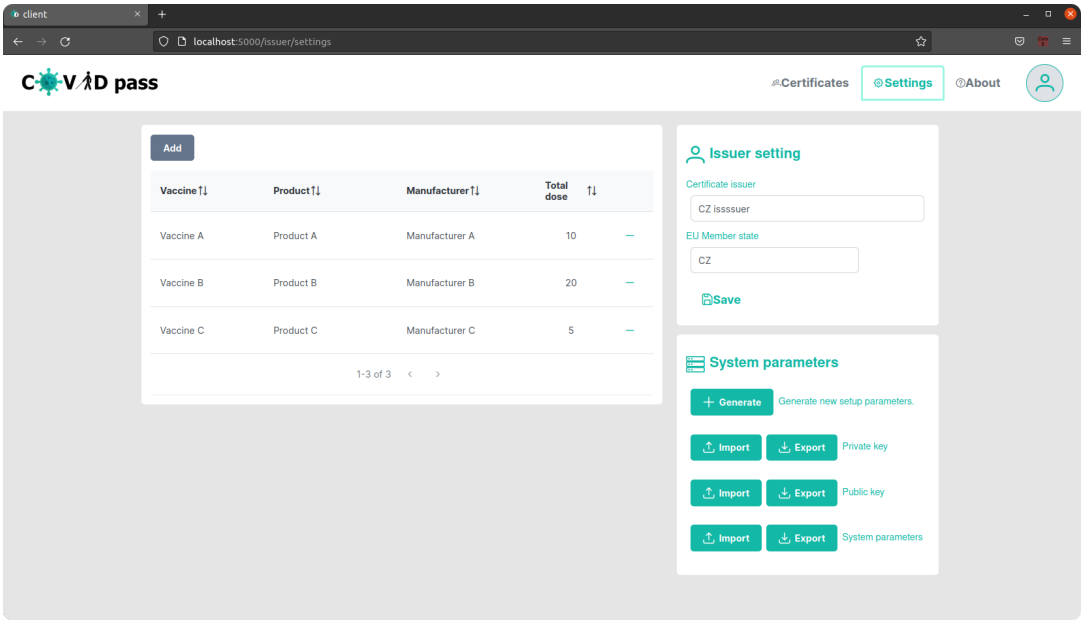
Obr. 5.9: Užívateľka sekcia webovej aplikácie vydavateľa.

sa nachádza tlačidlo *New*, ktoré po stlačení zobrazí dialógové okno vo formáte formuláru, ktoré slúži k vytváraniu nových certifikátov. Po kliknutí na tlačidlo *Settings*, ktoré sa nachádza v hlavičke na pravej strane bude užívateľovi zmenená stránka, viz. 5.11. Táto stránka sa skladá z troch častí. Na ľavej strane sa nachádza box s tlačidlom a tabuľkou, ktorá obsahuje záznam všetkých existujúcich vakcín v systéme, ktoré môžu byť použité pri vytváraní digitálnych certifikátov. Po stlačení tlačidla *Add* je zobrazené dialógové okno obdobne ako v prípade pridávania certifikátov.

Na pravej vrchnej časti sa nachádza box základných informácií o vydavateľovi digitálnych certifikátov. Tieto údaje sa budú nachádzať v každom certifikáte. V dolnej časti je vyobrazený box, ktorý umožňuje správu systémových parametrov a kryptografických kľúčov pre autentizačný atribútový systém. Po stlačení tlačidla *Generate* je zobrazené dialógové okno, ktoré po zakliknutí spustí proces generovania nových parametrov. Tlačidlá *Export* umožňujú stiahnutie parametrov vo formáte JSON.



Obr. 5.10: Vydavateľská sekcia webovej aplikácie vydavateľa.



Obr. 5.11: Nastavenia vydavateľskej sekcie.

6 Implementácia webových aplikácií

Táto kapitola je venovaná implementácii webovej aplikácie vydavateľa a overovateľa. Zo začiatku bude popísané pracovné prostredie, v ktorom vznikli. V kapitole sa ďalej bude nachádzať popis implementácie serverovej a klientskej časti kde budú predstavené nástroje a požadované závislosti.

6.1 Pracovné prostredie

V rámci vývoja bol vytvorený virtualizovaný operačný systém (skrátene *OS*) v bezplatnom programe *VMware Workstation 16 Player*. Výhodou použitia virtualizovaného OS je v odizolovaní hostiteľského OS od vývojového - pri vzniku fatálnej chyby bude ovplyvnený len virtualizovaný OS. Pre prácu bola zvolená distribúciu *Ubuntu 20.04*, ktorá je postavená na *linuxovom* jadre. *Ubuntu* je komunitou považovaná za užívateľsky prívetivú z pohľadu nastavenia a samotnej práce.

Novo nainštalovaný OS v základe neobsahuje niektoré potrebné závislosti a nástroje, ktoré umožňujú vývoj webových aplikácií. Vývoj aplikácií bude realizovaný v textovom editore od spoločnosti *Microsoft* s názvom *VS Code*, ktorý vďaka dodatočným balíčkom umožňuje prácu s programovacím jazykom *Python* a *JavaScript*. VS Code nie je štandardným nástrojom Ubuntu, preto je potrebné vykonať inštaláciu. Existujú rôzne spôsoby inštalácie avšak zvolený spôsob bol pomocou nástroja *snap*, ktorý spravuje binárne závislosti k spusteniu programov. 6.1.

```
$ sudo apt - get install snapd -y
$ sudo snap install code -y
```

Výpis 6.1: Prehľad inštalácie textového editoru VS Code.

Ďalším dôležitým krokom pri príprave pracovného prostredia je zabezpečenie podpory pri práci s programovacími jazykmi, na ktorých sú postavené zvolené frameworky. OS postavený na linuxovom jadre obsahuje základné závislosti medzi, ktoré patrí *interpreter* jazyka Python. Dodatočné balíčky budú pridané pomocou nástroja *pip3*, ktorý spravuje všetky závislosti jazyka Python. Prehľad inštalácie sa nachádza vo výpise 6.2.

```
$ sudo apt-get -y install python3 -pip
```

Výpis 6.2: Prehľad inštalácie nástroja pip3.

K správe závislosti bol nainštalovaný balíček *virtuálneho prostredia*, ktorý spravuje všetky závislosti pre konkrétny projekt v izolácií od globálnych. Inštalácia a vytvorenie virtuálneho prostredia sa nachádza vo výpise 6.3.

```
$ pip3 install virtualenv
$ virtualenv venv
```

Výpis 6.3: Nastavenie virtuálneho pracovného prostredia.

Pre prácu s VueJS je nutné zaistiť *runtime enviroment*, ktorý umožňuje spúšťať JavaScriptový skript. Túto úlohu bude zastrešovať *Node.js*, ktorý sa v operačnom systéme nenachádza. Inštalácia Node.js bude vykonaná pomocou nástroja *nvm*, ktorý spravuje rôzne verzie Node.js, medzi ktorými je možné prepínať. Okrem samotného Node.js je automaticky nainštalovaný nástroj npm, ktorý uľahčuje pridávanie a aktualizovanie dodatočných balíčkov. Prehľad požadovanej inštalácie viz. 6.4.

```
$ nvm install v16.14.0
$ nvm alias default v16.14.0
```

Výpis 6.4: Prehľad inštalácie Node.js.

6.2 Serverová časť

Serverová časť oboch webových aplikácií je postavená na mikroframeworku Flask a dodatočných závislostí, ktoré nie sú štandardnými knižnicami jazyka Python. Preto je potrebné nastaviť virtuálne prostredie, do ktorého budú nainštalované balíčky. Viz. 6.5. Všetky použité závislosti sa nachádzajú v prílohe A. Po nainštalovaní všetkých závislostí je potrebné ich pridať do projektu pomocou príkazu `import <názov_balíčku>`.

```
$ . venv/bin/activate
(venv)$ pip3 install <balíček>
```

Výpis 6.5: Prehľad inštalácie požadovaných závislosti.

6.2.1 Štruktúra projektu

Dôležitou časťou pri písaní každej aplikácie je dobré rozvrhnutie štruktúry kódu - rozmiestnenie jednotlivých súborov, ktoré by mali byť zlúčené po logických blokoch. Pri rozvrhnutí štruktúry webových aplikácii je použitá technika *Blueprint*. Flask Blueprint rozdeľuje štruktúru aplikácie do tzv. *modulov*, ktoré môžu obsahovať statické súbory, šablóny a pohľady obdobne ako v aplikácií Flask. Po vytvorení modulu je nutné ho zaregistrovať do inštancie Flasku pomocou funkcie `register_blueprint()` a s názvom plánu ako formálnym parametrom.

Webová aplikácie vydavateľa sa skladá zo štyroch Blueprint modulov a troch globálnych modulov, ktoré vznikli za účelom prehľadnosti. V stromovom výpise sa

nachádza rozloženie vytvorených modulov vydavateľa, ktoré sa nachádzajú v adresári *src* - zdrojový kód. Súbory s koncovým slovom *blueprint* obsahujú koncové body, ktoré odpovedajú klientovi na požiadavky HTTP vo formáte JSON. Súbory so *services* obsahujú pomocné funkcie ako napríklad interakcia s databázovým serverom poprípade usporiadanie väčšieho množstva dát do formátu JSON. Priečinok *adopsiolib* obsahuje funkcie, ktoré umožňujú prácu s externou knižnicou napísanou v jazyku C. Konkrétne sa jedná o knižnicu atribútového autentizačného systému.

```
src
├── admin ..... adresár obsahuje funkcie admina
│   ├── templates
│   │   └── admin.html
│   └── admin_blueprint.py
├── adopsiolib ..... adresár obsahuje zdrojový kód atribútového systému
│   ├── mcl ..... adresár obsahujúce dátové štruktúry mcl
│   ├── setup ..... adresár obsahujúci systémové parametre a kľúče
│   ├── adopsio.py
│   └── constants.py
├── auth ..... adresár obsahuje zdrojový kód autentizácie a autorizácie
│   ├── templates
│   │   ├── login.html
│   │   └── signup.html
│   ├── auth_blueprint.py
│   └── auth_services.py
├── issuer ..... adresár obsahuje koncové body pre SPA
│   ├── issuer_blueprint.py
│   ├── issuer_services.py
│   └── serialized_out.py
├── models ..... adresár obsahujúci modely k vytvoreniu databáze
├── static ..... adresár obsahujúci obrázka a kaskádový štýl CSS
└── user ..... adresár obsahuje funkcie užívateľa
    ├── templates
    │   └── user.html
    ├── user_blueprint.py
    └── user_services.py
```

Naproti tomu webová aplikácia overovateľa obsahuje tri moduly, z ktorých sú dva typu Blueprint. V stromovej štruktúre sa nachádza ukážka rozloženia modulov.

```
src
├── auth ..... adresár obsahuje zdrojový kód autentizácie a autorizácie
│   ├── templates
│   │   └── login.html
│   └── auth.py
├── static
└── verifier ..... adresár obsahuje zdrojový kód GUI
    └── setup
        └── attrs.json ..... súbor obsahujúci predvolené atribúty
```

```
templates
  dashboard.html
verifier.py
```

6.2.2 Prepojenie databázového serveru

Pri práci s databázou PostgreSQL je použitá knižnica *Flask-SQLAlchemy*, ktorá je rozšírením knižnice *SQLAlchemy* pre Flask. SQLAlchemy využíva techniku *Object Relation Mapper* (skrátene *ORM*), ktorá mapuje parametre objektu (tried) do štruktúr tabuľky relačnej databáze.

Vo výpise 6.6 sa nachádza trieda modelu, z ktorej je vytvorená tabuľka pomocou príkazu zadaného z terminálu `create_all()`. Zahŕňa informácie o priradenom *primárnom kľúči*, jednotlivé dátové typy riadkov a ďalšie možnosti.

```
1  class Vaccine(db.Model):
2
3      __tablename__ = TableField.VACCINE
4
5      id = db.Column(db.Integer, primary_key=True)
6      vaccine = db.Column(db.String(32), unique=True,
7          nullable=False)
8      product = db.Column(db.String(32), nullable=False)
9      manufacturer = db.Column(db.String(32),
10         nullable=False)
11     total_dose = db.Column(db.Integer, nullable=False)
```

Výpis 6.6: Model tabuľky vaccine.

Práca s tabuľkou je rovnaká ako so štandardnou triedou používanou v jazyku Python. Vďaka dedeniu triedy `Model` je možné využiť metódy triedy ako napríklad filtrovanie na základe podmienky. V príklade 6.7 sa nachádza metóda, ktorá získa záznam vakcíny ako objekt na základe názvu. V prípade, že sa v databáze nenachádza žiaden záznam s hľadaným názvom vracia sa prázdna hodnota *None*.

```
1      def vaccine_by_name(name):
2          return Vaccine.query.filter(
3              (Vaccine.vaccine == name)
4          ).first()
```

Výpis 6.7: Dotazovanie na vakcínu na základe názvu.

6.2.3 Prepojenie knižnice Adopsio

Adobsio je kryptografická knižnica, ktorá je napísaná v jazyku C. Interakciu knižnice Adopsio s webovou aplikáciou vydavateľa zastrešuje štandardná knižnica jazyka Python s názvom *ctypes*, ktorý je predstavený v kapitole 2.1.

V knižnici Adopsio sú využité vlastné dátové štruktúry, ktoré sa v jazyku Python nenachádzajú. Vytvorenie identických štruktúr je možné za pomoci dedičnej triedy `Structure`. Na obrázku 6.1 je uvedený príklad kde na ľavej strane je umiestnená štruktúra z knižnice Adopsio a na pravej je znázornená identická štruktúra za pomoci knižnice ctypes.

```
typedef struct
{
 mclBnFr sk;
} issuer_private_key_t
```

```
# Structure mcl Fr
class _FR(Structure):
 _fields_ = [('v',
   c_ulonglong *
   SystemSetup.MCLBN_FR_UNIT_SIZE)]

# Structure adopsio issuer_private_key_t
class _ISSUER_PRIVATE_KEY(Structure):
  _fields_ = [('sk', _FR)]
```

Obr. 6.1: Prehľad štruktúry využitím knižnice ctypes.

Pre prácu s API funkciami knižnice Adopsio je vytvorená zabaľovacia funkcia, ktorá na pozadí vytvorí dotaz a vracia stavový kód, viz. 6.8. Úspešné spracovanie je oznámené hodnotou 0. Zo získaného stavového kódu pri použití funkcie `adopsio_error_to_string()` je možné získaný ľudsky čitateľný výstup.

```
1 def wrap_function(lib, funcname, restype, argtypes):
2     func = lib.__getattr__(funcname)
3     func.restype = restype
4     func.argtypes = argtypes
5     return func
```

Výpis 6.8: Zabaľovacia funkcia pre prácu s Adopsio.

6.2.4 Autentizácia a autorizácia

Z podkapitoly 5.3 vyplýva, že autentizácia je prenechaná na poskytovateľovi služieb. Dôležitou funkciou, ktorú je potrebne implementovať sa nazýva tzv. *callback* - spätné volanie. Adresa URL na koncový bod funkcie je potrebné zaregistrovať na strane poskytovateľa a slúži k presmerovaniu overeného klienta na webovú aplikáciu. V prípade nastavenia callbacku je povolené používať adresu *localhost* vrámci lokálneho

testovania, alebo *doménové meno* vrámci produkcie. Vo funkcii callback sa overuje pravosť autorizačného tokenu s emailom užívateľa.

Po úspešnom overení sa vo webovej aplikácií vytvorí súbor cookies pomocou knižnice *Flask-Login*. Súbor cookies je malé množstvo dát, ktoré sa ukladá na strane klienta a je ich možné použiť vrámci opätovného prístupu. Autorizácia vo webových aplikáciach sa zaisťuje na základe získaných rolí. Popis jednotlivých rolí použitých v aplikáciach sa nachádza v podkapitole 5.3. Túto operáciu zaisťuje knižnica *Flask-Principal*. Vo výpise 6.9 sa nachádza koncový bod, ktorý je prístupný pre rolu admin a po zavolaní zobrazí textový reťazec. V prípade, že užívateľ nevlastní rolu admin bude vyvolaná výnimka s číslom 403 - neautorizovaný užívateľ. Taktiež sa tu nachádza dekorátor funkcie `@login_required`, ktorý obmedzuje dotazovanie na prihlásených užívateľov. V prípade neovereného užívateľa bude koncový bod ignorovať požiadavku.

```
1     admin_permisson = Permisson(NeedRole('admin'))
2
3     @admin.route('/pozdrav', methods=['GET'])
4     @login_required
5     @admin_permisson.require(http_exception=403)
6     def index_person():
7         return jsonify('Ahoj, admin!')
```

Výpis 6.9: Ukážka prístupný na základe role admin.

6.3 Klientska časť

Strana klienta, alebo inak nazvané frontend je postavený na progresívnom webovom frameworku *Vue.js*. Detailnejší popis frameworku a možná alternatívna technológia použiteľná k vývoji sa nachádza v podkapitole 1.1.

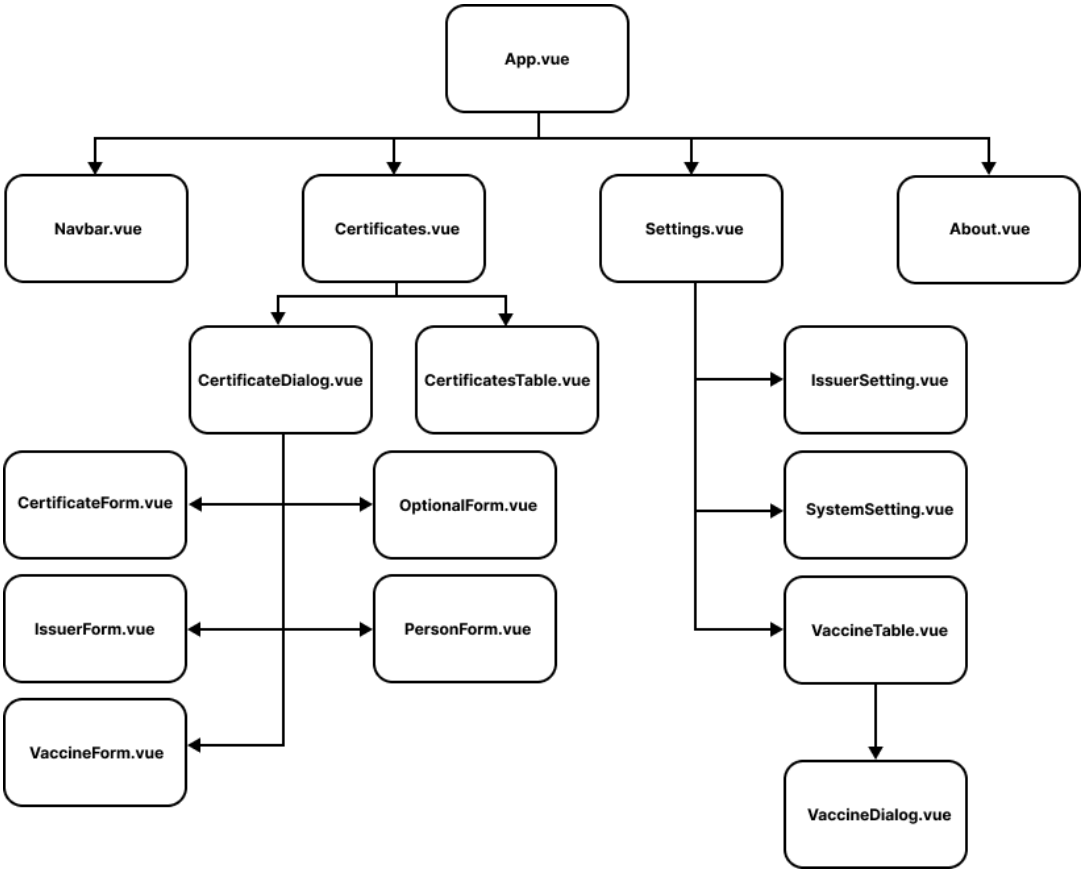
Prvotné nastavenie projektu je možné vykonať dvoma spôsobmi, *s nástrojom*, alebo *bez nástroja* na zostavenie. Prvý spomenutý spôsob je realizovateľný s použitím nástroja *Vue CLI*, ktorý vytvorí predkonfigurovaný projekt. Druhý spôsob je možné realizovať pridaním existujúceho skriptu nachádzajúceho na internete do šablóny HTML. V tejto práci sa vyskytujú oba spôsoby, kde projekt zostavený nástrojom Vue CLI sa nachádza vo webovej aplikácií vydavateľa prístupnej pre užívateľov s rolou issuer. Ostatné sekcie vydavateľa a overovateľa sú zostavený bez nástroja.

6.3.1 Zostavenie s nástrojom

Pred samotným používaním je potrebné nástroj *Vue CLI* nainštalovať. To je možné vykonať pomocou `npm` - správca balíčkov jazyka JavaScript, zadaním príkazu `npm install -g @vue/cli`. Projekt sa následne vytvorí zadaním príkazu `vue create <názov>`. Počas vytvárania je možné nastaviť základné závislosti, ktoré sú automaticky stiahnuté a nastavené hneď po vytvorení.

Rozdelenie komponent

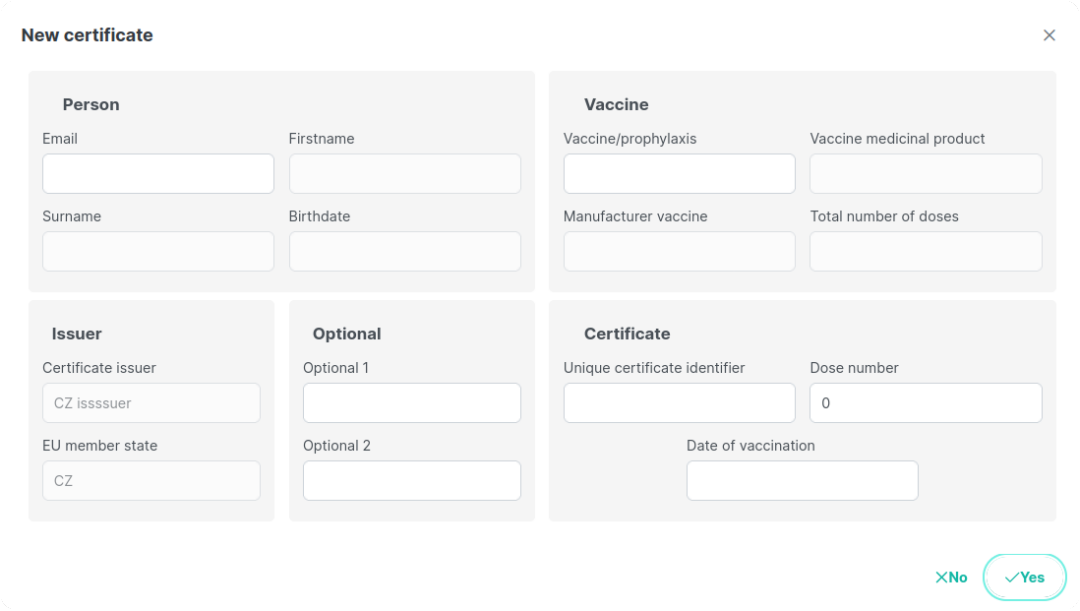
Najzákladnejšou vlastnosťou Vue.js je možnosť rozdelenia častí blokov kódu do komponent. Táto vlastnosť umožňuje zníženie duplicitného písania kódu a zároveň ho sprehľadniť. Na obrázku 6.2 sa nachádza ukážka prepojenia použitých komponent. Koreňovým komponentom je súbor *App.vue*, ktorý obsahuje navigačnú hlavičku s jednotlivými *pohľadmi* (angl. *router*). Pod týmito komponentami sa nachádzajú ďalšie podradené komponenty. Nadradený komponent sa nazýva rodič a podradený dieťa.



Obr. 6.2: Rozloženie komponent webovej aplikácie vydavateľa s rolou issuer.

Táto časť aplikácie obsahuje šestnásť komponent. Medzi tie najdôležitejšie komponenty patrí `Navbar.vue` a `Certificates.vue`. Prvý spomenutý komponent je označený ako hlavička každej stránky obsahujúca navigačné menu. Obsahuje záznam všetkých možných stránok, medzi ktorými umožňuje prepínať. Taktiež slúži ako pútač, ktorý obsahuje logo projektu a tlačidlo k odhláseniu sa z webovej aplikácie. Odhlásenie je realizované volaním koncového bodu nachádzajúceho na serveri pomocou balíčku `axios`. Balíčku axios sa bude detailnejšie venovať s samostatnej časti.

Druhým spomenutým komponentom je stránka certifikátov. Obsahuje dva podradené komponenty - dialógové okno a tabuľku. Tabuľka obsahuje všetky získané záznamy vytvorených digitálnych poverení zo databázového systému. Poverenia sa môžu mazať, ale aj pridávať. V prípade vytvorenia nového certifikátu je zobrazené dialógové okno 6.3, ktoré sa skladá z piatich podradených komponentov. Každý podradený komponent je formulár s vlastnou správou jednotlivých textových polí a obsahujúci metódu `setForm()`, ktorá zabezpečuje zaslanie dát. Na strane rodiča sú všetky podradené dokumenty zaregistrované a importované do časti templates. Pri importovaní je do každej časti pridaná hodnota pomocou `v-model`, ktorá zviaže dáta na strane dieťaťa s rodičom. Ďalej je pridaná referencia, ktorá odkazuje na metódu nachádzajúcu v podradenom komponente. Po stlačení tlačidla *Yes* je vďaka vytvorenej referencii vyvolaná metóda, ktorá upravuje hodnotu získanú z `v-model` a zaslaná rodičovi pomocou `emit`.



Obr. 6.3: Dialóg vytvorenia nového digitálneho poverenia.

Knižnica UI

PrimeVue je UI knižnica, ktorá bola navrhnutá pre Vue.js. Zahŕňa širokú škálu komponentov a šablón, ktoré je možné jednoducho implementovať do existujúceho projektu. Je stavebným kameňom UI oboch webových aplikáciach. V tejto podkapitole bude predstavený spôsob implementácie pri vytvorenom projekte nástrojom *Vue CLI*. *PrimeVue* nie je štandardnou knižnicou VueJS a preto je potrebné stiahnuť všetky potrebné závislosti do vytvoreného projektu, viz. 6.10. Konkrétne sa jedná o balíček *PrimeVue*, *PrimeIcons* a *PrimeFlex*, ktorý obsahuje nástroje pri práci s CSS.

```
$ npm install primevue@^3 --save
$ npm install primeicons --save
$ npm install primeflex --save
```

Výpis 6.10: Stiahnutie závislostí PrimeVue.

Po úspešnom stiahnutí je potrebné zaregistrovať konfiguráciu knižnice a použité komponentu. Komponenty budú využívané v celom projekte a preto je potrebné vykonať registráciu globálne - nastaviť v súbore *main.js*. Viz. 6.11.

```
1  import {createApp} from 'vue';
2  import App from './App.vue';
3
4  // import PrimeVue knižnice
5  import PrimeVue from 'primevue/config';
6  import 'primevue/resources/primevue.min.css';
7  // import nástroja PrimeFlex
8  import 'primeflex/primeflex.css';
9  // import PrimeIcons
10 import 'primeicons/primeicons.css';
11 // import komponentu tlačidla
12 import Button from 'primevue/button';
13
14 const app = createApp(App);
15 // sprístupnenie komponentu
16 app.component('Button', Button);
17
18 // globálna registrácia balíčku PrimeVue
19 app.use(PrimeVue);
```

Výpis 6.11: Registrácia PrimeVue.

Od tohoto momentu je možné použiť registrovaný komponent v šablóne, ktorý bude umiestnený vo vnútri `<div>` a v sekcii `templates`. Viz. 6.12. Na obrázku 6.4 sa nachádza grafické zobrazenie komponenty tlačidla.

```
1    <Button type="button" class="button-1␣p-button-lgs">
2      <i
3        class="pi␣pi-google␣button-1-color"
4        style="font-size:␣2rem"
5      ></i>
6      <span
7        class="ml-1␣font-bold␣button-1-color"
8      >Sign in</span>
9    </Button>
```

Výpis 6.12: Použitie komponentu tlačidla.

Obr. 6.4: Grafické zobrazenie komponenty tlačidla.

Komunikácia so serverom

Ku komunikácií so serverom je použitý HTTP klient `axios` umožňujúci vytvárať HTTP požiadavky - `GET`, `POST` apod. Vo výpise 6.13 je demonštrovaná metóda, ktorá vykoná HTTP GET požiadavok na stranu serveru. Parametrom je URI odkazujúce na koncový bod REST API, na ktorom sa nachádzajú požadované dáta. Po zaslaní požiadavku sú získané dáta spracované v časti `.then()`, ktorá v ukážke vypíše dáta do webovej konzoly a priradí obsah dát do premennej `this.vaccines`. V prípade vzniku chybového stavu je zachytený v časti `.catch()` a vypísaný do konzole.

Axios taktiež umožňuje využitie asynchrónneho požiadavku za pomoci kľúčových slov `async` a `await`. To umožňuje funkčnosť webovej aplikácie, ktorá na pozadí čaká odpoveď na zaslanie požiadaviek. Kľúčové slovo `async` je umiestené pred funkciou, ktorá je označená za asynchrónnu. V asynchrónnej funkcii sa musí nachádzať slovo `await`, ktoré pozastaví prácu vykonávajúcu vo funkcii a znovu ju spustí po obdržaní odpovede - vykoná sa kód v časti `.then()`.

```
1     async fetchAllVaccines() {
2         await axios.get(BASE_URL + '/vaccines')
3         .then((resp) => ) {
4             console.log(resp.data);
5             this.vaccines = resp.data;
6         }
7         .catch((err) => {
8             console.log(err.resp.data);
9         })
10    };
```

Výpis 6.13: Demonštrácia GET požiadavku využitím axios.

6.3.2 Zostavenie bez nástroja

Zostavenie projektu bez nutnosti nástroja je možné za pomoci CDN skript. Jedná sa o skripty, ktoré obsahujú záznamy verzií jednotlivých balíčkov na vzdialenom serveri. Možnou výhodou je, že pri použití identickej verzie balíčku vo viacerých weboch je obsah uložený do tzv. *medzipamäte* (angl. *cache*). To má za následok, že obsah sa nebude musieť znovu sťahovať.

Pri práci s projektom vytvoreným pomocou nástroja Vue CLI sa pracuje so súborom končiacim `.vue`, v ktorom vzniká kód vo formáte SFC. Naproti tomu pri použití CDN skriptu je vytvorený súbor `.html`, do ktorého sa pomocou tagu `<script>` umiestní kód napísaný v JavaScripte - v prípade tohto projektu sa jedná o inštanciu Vue.js. Týmto spôsobom sú vytvorené ostatné stránky webových aplikácií. V prípade vydavateľa sa jedná o štyri súbory - `login.html`, `signup.html`, `user.html` a `admin.html`. Overovateľ zahŕňa dva súbory - `login.html` a `dashboard.html`. V týchto stránkach sa taktiež nachádza knižnica axios a UI knižnica PrimeVue, s ktorými sa pracuje obdobne ako pri použití v projekte vytvorenom za pomoci Vue CLI. Vo výpise 6.14 sa kód použitý k vytvoreniu identického tlačidla nachádzajúceho sa na obrázku 6.4.

```
<p-button
  @click="login"
  class="button-1 p-button-lg"
>
  <i
    class="pi pi-google button-1-color"
    style="font-size: 2rem"
  ></i>
  <span
    class="ml-1 font-bold button-1-color"
  >Sign in</span>
</p-button>
```

Výpis 6.14: Použitie komponentu tlačidla.

Záver

Cieľom bakalárskej práce bolo zoznámiť sa s modernými technológiami k vytvoreniu webovej aplikácií a možných spôsobov implementácie knižníc nižších programovacích jazykov. Ďalšou požiadavkou bolo zoznámiť sa s atribútovým autentizačným systémom, ktorý je základom webových aplikácií.

Získané informácie boli uplatnené pri vývoji dvoch webových aplikácií. Konkrétne sa jednalo o entity vydavateľa a overovateľa atribútového systému Adopsio - výstup tohtoročnej diplomovej práce.

Obe webové aplikácie prenechávajú autentizáciu na spoločnosti Google. V aplikáciach je následne riešenia autorizácia na základe pridelených rolí. K získaniu vygenerovaných digitálnych certifikátov je vytvorený koncový bod, ktorý autorizuje užívateľa na základe jednorázového autorizačného API kľúča.

Aplikácia vydávateľa sa skladá z troch sekcií, *užívateľ*, *admin* a *user*. Užívateľská časť sprístupňuje digitálne certifikáty obsahujúce atribúty s podpismi tzv. *sigmy*. Tieto certifikáty je možné zobraziť v bočnom panely, alebo stiahnuť do mobilného zariadenia, na ktorom sa nachádza vhodná aplikácia - výstup diplomovej práce. Ak užívateľ zažiada o stiahnutie certifikátov bude spustený atribútový systém Adopsio, ktorý zo získaného atribútu vytvorí podpis. Sekcia admin má za úlohu spravovať všetkých užívateľské účty. To zahrňuje možnosť mazania, alebo zmena role. Posledná sekcia je určená pre vydavateľa, ktorý má za úlohu vytvárať nové certifikáty, poprípade ich mazať. V tejto sekcii boli pridané vlastnosti ako predvytvorenie vydavateľského účtu a vakcíny, ktoré sa používajú pri generovaní digitálnych certifikátoch. zaisťuje možnosť vytvorenia digitálnych certifikátov. Z tejto sekcie je možné spravovať systémové parametre atribútového autentizačného systému.

Aplikácia overovateľa obsahuje grafické užívateľské rozhranie, pomocou ktorého je možné ovládať službu overovateľa - výstup tohtoročnej diplomovej práce. Grafické rozhranie umožňuje zobrazenie logovacích záznamov overenia a možnosť zvoliť overovacie atribúty.

Literatúra

[1] CHOLIA, S.; SKINNER, D.; BOVERHOF, J. *NEWT: A RESTful service for building High Performance Computing web applications, 2010* Gateway Computing Environments Workshop (GCE). IEEE, 2010. p. 1-11.

[2] SHARMA, V.; VERMA, R.; PATHAK, V.; PALIWAL, M.; A JAIN, P. *Progressive web app (PWA)-one stop solution for all application development across all platforms.* Delhi, India, 2019 [cit. 2021-12-04] Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol, 5(2), 1120-1122.

[3] ALTEXSOFT. *Progressive Web Apps: Core Features, Architecture, Pros and Cons* [online]. 2018-05-10 [cit. 2022-05-28]. <`https://www.altexsoft.com/blog/engineering/progressive-web-apps/`>

[4] WOHLGETHAN, Eric. *Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js* [online]. Hamburg, 2018 [cit. 2022-05-24]. Dostupné z: <`https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf`>. Bakalárska práca. Hamburg University of Applied Sciences.

[5] KNOWLEDGEHUT SOLUTIONS PRIVATE LIMITED. Exploring the Various Decorators in Angular. *Knowledgehut.com* [online]. 2022-04-20 [cit. 2022-05-24]. Dostupné z: <`https://www.knowledgehut.com/blog/web-development/decorators-angular`>

[6] Introduction. *Vue.js* [online]. [cit. 2022-05-25]. Dostupné z: <`https://vuejs.org/guide/introduction.html`>

[7] NOVAC, Ovidiu Constantin, et al. Comparative study of some applications made in the Angular and Vue. js frameworks. *In: 2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*. IEEE, 2021. p. 1-4.

[8] PALLETS. *Foreword* [online]. [cit. 2022-05-25]. Dostupné z: <`https://flask.palletsprojects.com/en/2.1.x/foreword/`>

[9] LARAVEL LLC. Installation. *Laravel.com* [online]. [cit. 2022-05-29]. Dostupné z: <`https://laravel.com/docs/9.x/installation`>

[10] LARAVEL LLC. Directory Structure. *Laravel.com* [online]. [cit. 2022-05-29]. Dostupné z: <`https://laravel.com/docs/9.x/structure`>

[11] TECHOPEDIA INC. *Marshalling* [online]. [cit. 2022-05-24]. Dostupné z: <https://www.techopedia.com/definition/16408/marshalling>

[12] PYTHON SOFTWARE FOUNDATION. *Ctypes — A foreign function library for Python* [online]. [cit. 2021-12-04]. Dostupné z: <https://docs.python.org/3.8/library/ctypes.html

[13] SHAW, Graham. *Build a shared library using GCC* [online]. 2010 [cit. 2022-05-24]. Dostupné z: <http://www.microhowto.info/howto/build_a_shared_library_using_gcc.html>

[14] ANDERSON, Jim. Python Bindings: Calling C or C++ From Python. *RealPython* [online]. 2020-03-02 [cit. 2022-05-25]. Dostupné z: <https://realpython.com/python-bindings-overview>

[15] MOZILLA FOUNDATION. *WebAssembly Concepts* [online]. 2022-04-27 [cit. 2022-05-25]. Dostupné z: <https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts>

[16] EMSCRIPTEN CONTRIBUTORS. About Emscripten. *Emscripten.org* [online]. [cit. 2022-05-29]. Dostupné z: <https://emscripten.org/docs/introducing_emscripten/about_emscripten.html>

[17] MOZILLA FOUNDATION. *Compiling a New C/C++ Module to WebAssembly* [online]. 2022-05-19 [cit. 2022-05-25]. Dostupné z: <https://developer.mozilla.org/en-US/docs/WebAssembly/C_to_wasm>

[18] TIWARI, Sumit. An introduction to QR code technology. In: *2016 international conference on information technology (ICIT)*. IEEE, 2016. p. 39-44. [cit. 2022-05-24]

[19] AO KASPERSKY LAB. QR Code Security: What are QR codes and are they safeto use?. *Kaspersky.com* [online]. [cit. 2021-12-04]. Dostupné z: <https://www.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan>

[20] TOSI, Jacopo, Fabrizio TAFFONI, Marco SANTACATTERINA, Roberto SANNINO a Domenico FORMICA, 2017. Performance Evaluation of Bluetooth Low Energy: A Systematic Review. *Sensors* [online]. 17(12), 2898. ISSN 1424-8220. Dostupné z: <https://doi.org/10.3390/s17122898>

[21] PUNCH THROUGH, 2013. *How GAP and GATT Work* [online]. [cit. 2022-05-17]. Dostupné z: <https://punchthrough.com/how-gap-and-gatt-work/>

[22] SINGH, Manmeet Mahinderjit; ADZMAN, K. A. A. K.; HASSAN, Rohail. Near Field Communication (NFC) *technology security vulnerabilities and countermeasures*. International Journal of Engineering & Technology, 2018, 7.4.31: 298-305. [cit. 2022-05-24]

[23] BENEŠ, Vlastimil a VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, 2021. *Automatizovaná detekce ochranných prostředků a stavu indikujícího onemocnení (ADOPSIO) VI04000069: Návrh bezpečnosti a funkční koncepce systému.*

[24] HAJNÝ, J.; DZURENDA, P.; CAMENISCH, J.; DRIJVERS, M. *Fast Keyed-Verification Anonymous Credentials on Standard Smart Cards*. In ICT Systems Security and Privacy Protection. Springer Nature Switzerland, 2019. s. 286-298 [cit. 2022-05-23]. ISBN: 978-3-030-22312-0.

[25] MINISTERSTVO ZDRAVOTNICTVÍ, 2021. Certifikát oprovedeném očkování (testování a prodělání nemoci). *COVID PORTÁL* [online]. 2022-05-05 [cit. 2022-05-30]. Dostupné z: <`https://covid.gov.cz/situace/registrace-na-ockovani/certifikat-o-provedenem-ockovani-testovani-prodelani-nemoci`>

[26] CISA. *Security Tip (ST04-018): Understanding Digital Signatures* [online]. 2021-02-01 [cit. 2022-05-29]. Dostupné z: <`https://www.cisa.gov/tips/st04-018`>

[27] KAROPOULOS, G., KAMBOURAKIS, G., KOULIARIDIS, V., HERNANDEZ-RAMOS, L. J., *A Survey on Digital Certificates Approaches for the COVID-19 Pandemic* [online]. IEEE Access, 2021 [cit. 2021-12-10] Dostupné z: <`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9558786`>, 9: 138003-138025.

[28] Using OAuth 2.0 to Access Google APIs, 2021. *Google Identity* [online]. [cit. 2022-05-22]. Dostupné z: <`https://developers.google.com/identity/protocols/oauth2`>.

Zoznam symbolov a skratiek

Adopsio	Automatizovaná detekce ochranných prostředků a stavu indikujícího onemocnění
API	Application Programming Interface
BLE	Bluetooth Low Energy
CDN	Content Delivery Network
CLI	Command Line Interface
CSS	Cascading Style Sheets
eIDAS	electronic IDentification, Authentication and trust Services
ERD	Entity Relationship Diagram
EUDCC	EU Digital Covid Certificate
FKVAC	Fast Keyed-Verification Anonymous Credentials
GAP	Generic Access Profile
GATT	General Agreement on Tariffs and Trade
GDPR	General Data Protection Regulation
GUI	Graphic User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
MVC	Model-View-Controler
NFC	Near Field Communication
ORM	Object-Relational Mapping
OS	Operating System
PWA	Progressive Web Apps

QR	Quick Response
REST	Representational State Transfer
SFC	Single-File Components
SPA	Single-Page Application
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TS	TypeScript
UI	User Interface
URI	Uniform Resource Identifier
UML	Unified Modeling Language
URL	Uniform Resource Locator
Wasm	WebAssembly
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language

Zoznam príloh

A Požadované balíčky vydavateľa

Na výpisu A.1 sa nachádza zoznam potrebných závislostí využitých pri vývoji a spustení webových aplikácií. Balíčky sú nainštalované vo virtuálnom prostredí pomocou nástroja *pip3*. K získanie nainštalovaných balíčkov je možné využiť príkaz *pip3 freeze*.

```
Flask==2.1.1
Flask-Cors==3.0.10
Flask-Login==0.6.0
Flask-Migrate==3.1.0
Flask-SQLAlchemy==2.5.1
psycopg2-binary==2.9.3
gunicorn==20.1.0
pathlib2==2.3.7.post1
psycopg2-binary==2.9.3
python-dotenv==0.19.2
oauthlib==3.2.0
requests==2.27.1
pyOpenSSL==22.0.0
numpy==1.22.3
```

Výpis A.1: Prehľad požadovaných balíčkov na strane serveru.

B Ukážka digitálneho certifikátu

Na obrázku B.1 sa nachádza ukážkové digitálne poverenie získané pre užívateľa na základe vygenerovaného kľúča API. K získaniu je možné použiť nástroj `curl`, ktorý sa v distribúcií Ubuntu 20.0.4 inštaluje príkazom `sudo apt-get install curl`. Po nainštalovaní je potrebné zadať príkaz `curl <názov_serveru>/user/certs -H 'X-API-KEY: <API_KEY>'`.

```
{
  "status": true,
  "creation_time": "01-05-2022",
  "email": "covidpass.user@gmail.com",
  "certificate": {
    "firstname": "UserName",
    "surname": "UserSurname",
    "birthdate_day": "5",
    "birthdate_month": "3",
    "birthdate_year": "2022",
    "photo_hash": "whnquipwmyxinuqccisswldarheqfzgp",
    "unique_id": "uuid:dasbczxc-dscnjsd-542368-dasczn",
    "vaccination_day": "10",
    "vaccination_month": "4",
    "vaccination_year": "2022",
    "dose": "5",
    "total_dose": "10",
    "completed_vaccination": "False",
    "vaccine": "Vaccine A",
    "product": "Product A",
    "manufacturer": "Manufacturer A",
    "issuer": "CZ issuer",
    "state_eu": "CZ",
    "optional1": "Brno",
    "optional2": "B007"
  },
  "sigma": {
    "sigma_0": "1 141664372877704045411731836130197586495757727632096124496069720078668789893296 872676961430033546724727380036858394081089230002756115874996736955687107433 6",
    "sigma_1": "1 701303837479194590100939997428109472459382849453670285081663918010955 3195113 6037421768854728589528201864023835974427947207544687736723333276565183411524",
    "sigma_2": "1 886721487105201558276550576472908320954287868081480497778600963116053738186 2 516547542381191671969356779333480385019373738277212711240277079251498199461 40",
    "sigma_3": "1 128277014443278072805519293319699268119250337726942688717522769743394359268 0 92287131074283088543003683474647480798951586750590568244542111203585342526 17",
    "sigma_4": "1 143563011020205793889842569411349068426226966599329068264963589593086337696891 8034790518327025316663159641333841910699747791930663702072140337240357599071",
    "sigma_5": "1 477489740518898093898340222100317670089373529995909103512448709520838560222 75 706986643346380802855596084177015910434254941776745888653115742256965810 1386",
    "sigma_6": "1 109579188378483036439650927029627817559648840743019065978876356990250272983 16 662230781943764863337851578598842745390740068630267955748560204231260747005 0",
    "sigma_7": "1 370678329961754419663389378085016148437661968768710372574787000713380446336 0 727210477995916793464722236484021530352683094103901736546574229536975488274 1",
    "sigma_8": "1 427195079695740906886720291569743312969893429223215965510208531239139969256 5 160137220257441283027182349612231531091827099543669848991996563159880802006 33",
    "sigma_9": "1 122827816549742659361610409464819528141929454398959337804120822138867076787874 1003614868111412348784411742426255220945358289274667284308646607709868764382 0",
    "sigma_10": "1 165057834754873440216527160747108959785421392407514796063797429933605452337 9 136060816810565829490046050632187498372185393672783994849206838179764892810 85",
    "sigma_11": "1 137067677422636332511993152079773402911244420052351500118278325831827464646 91 984312700390967587548209327613880183342406817199480622414359570907490461925",
    "sigma_12": "1 422279097335561073117705785294953244051713579878411010619311508662578593400 1 152268847803786408944519608583036320213975955298201509944762985323894006270 46",
    "sigma_13": "1 504459288494268748342303553587466926635346265684590422960077997547482450873 1 130020978480222929596257224451224169933422692454213961359257547144670412207344",
    "sigma_14": "1 802075187357053351121850418608754101023771106363781589443314862439093222154 7480578937525315157621393296259011026439501230271521715781160327552250220341",
    "sigma_15": "1 102304780858347070255926561360687959817618290000683128647637623415205295385 87 994229538016530463349033928516106189323120074379038936457817155412969418455 5",
    "sigma_16": "1 127517318752062916418655584134848871896285947186304381104912279010814868369 34 153082010085845636018413514313954623971001826750427229909594172238947264122 75",
    "sigma_17": "1 766247626384651113013345500872841820694237214309654260400082368302075101653 4 128489468901891410675932243545109116454474651201893736210421098890705840103 0",
    "sigma_18": "1 884906354853967323671491487431403236483553691110752308036178833745135488153 3 100923615062696509509140968431881793111935621563902995807929555041432672803 5",
    "sigma_19": "1 158601636936799044535592937991520866445036967267492944218669134108648085161075 9974242860492379876637514033412651527043293250827225693514084526022743784617",
    "sigma_20": "1 141891919938419125784828156870495935019408395218175254597751291301402318596 43 558089277647771105579052222858758334997401016751663106630484334835840746819 2"
  }
}
```


Obr. B.1: Ukážka digitálneho certifikátu.

C Obsah elektronickej prílohy

V prílohe sa nachádzajú zdrojové kódy webovej aplikácie overovateľa a vydavateľa. Súbor s názvom `manual.pdf` obsahuje užívateľskú príručku pri používaní oboch aplikácií.

Obsah

- `/` — koreňový adresár priloženého archívu
  - `web-adopsio` — koreňový adresár webovej aplikácie vydavateľa
    - `services` — adresár obsahuje komponenty webovej aplikácie vydavateľa
      - `client` — adresár obsahuje frontend webovej aplikácie vydavateľa
      - `db` — adresár obsahuje nastavenia databázového serveru
      - `nginx` — adresár obsahuje nastavenia reverzného proxy
      - `server` — adresár obsahuje backend webovej aplikácie vydavateľa
        - `migrations` — adresár s migrovanou databázou
        - `src` — adresár zdrojového kódu backendu
          - `admin`
          - `adopsiolib`
          - `auth`
          - `issuer`
          - `models`
          - `static`
          - `user`
          - `constant.py`
      - `Dockerfile` — súbor s nastavením kontajneru pre frontend a backend
    - `README.md` — súbor s obecným popisom projektu
    - `docker-compose.yml` — správa kontajner, spúšťací súbor
  - `web-verifier` — koreňový adresár webovej aplikácie overovateľa
    - `migrations` — adresár s migrovanou databázou
    - `src` — adresár zdrojového kódu backendu
      - `auth`
      - `static`
      - `verifier`
      - `config.py`
      - `model.py`
    - `config.yaml` — súbor obsahuje premenné prostredia
    - `manage.py` — súbor obsahuje spúšťacie skripty
    - `requirements.txt` — súbor obsahuje potrebné závislosti projektu
    - `wsgi.py` — súbor slúži k spusteniu webovej aplikácie overovateľa
  - `manual.pdf` — súbor pdf obsahujúci užívateľský manuál