

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MODUL REGISTRÁCIE A PRIHLASOVANIA
WEBOVEJ APLIKÁCIE
BAKALÁRSKA PRÁCA

2021
FREDERIK KOHÁR

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MODUL REGISTRÁCIE A PRIHLASOVANIA
WEBOVEJ APLIKÁCIE

BAKALÁRSKA PRÁCA

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: RNDr. Marek Nagy, PhD.

Bratislava, 2021
Frederik Kohár



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Frederik Kohár
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Modul registrácie a prihlasovania webovej aplikácie
Registration and login module of web application

Anotácia: Webové aplikácie štandardne využívajú autentifikáciu užívateľov, na základe ktorej sprístupňujú ďalší obsah a funkcionality. Bežný prístup je formou mena a hesla. Pre deti, ktoré majú ešte problém s čítaním, je táto forma náročná. Preto treba uprednostniť komunikáciu prostredníctvom obrázkov. Zaujímavým rozšírením overenia je aj hlasová biometria. Webové aplikácie s veľkým množstvom užívateľov, ako sú napríklad školské portály, vyžadujú ich efektívny manažment. Kľúčovým prvkom je hlavne registrácia, ktorá by mala využiť aspoň jeden spoľahlivý prvok ako napríklad overenie cez email. Prípadne sa spoľahnúť na externé služby prihlasovania, ktorým sa dôveruje.

Cieľ: Vytvoriť modul v základnom Javascript kóde s prepojením na node.js a socket.io knižnicu. Modul bude obhospodarovať lokálnu databázu všetkých užívateľov portálu. Bude realizovať overovanie klasickou formou (meno+heslo), grafickou formou pre deti a overovanie cez externé služby (univerzitné prihlasovanie, google, facebook,...). Doplnkové overenie bude experimentálne cez hlasovú biometriu. Modul bude poskytovať aj lokálnu registráciu potvrdzovanú emailom.

Vedúci: RNDr. Marek Nagy, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.
Dátum zadania: 27.09.2020

Dátum schválenia: 30.09.2020
doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne vyhlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

V Bratislave, dňa 31.5.2021

.....

Frederik Kohár

Pod'akovanie: Chcem sa poďakovať svojmu školiteľovi RNDr. Marekovi Nagyovi PhD. za pomoc, cenné rady a čas, ktorý mi venoval počas tvorby mojej bakalárskej práce.

Abstrakt

Cieľom tejto bakalárskej práce bolo vytvoriť všestranný modul registrácie a prihlásenia webovej aplikácie. Používatelia sa registrujú do webovej aplikácie pomocou viacerých metód. Medzi tieto metódy patrí lokálna registrácia pomocou mena, hesla a emailu, registrácia cez Facebook, Google a Univerzitné konto. Zaregistrovaní používatelia sa ukladajú do lokálnej databázy. Po registrácii sa môžu používatelia prihlásiť do webovej aplikácie použitím zaregistrovanej metódy. Zaregistrovaní a prihlásení používatelia budú mať možnosť pridať ďalšie dve metódy, cez ktoré sa môžu prihlasovať. Tieto metódy sú prihlásenie pomocou hlasu a prihlásenie pomocou QR kódu. Pomocou pridaných metód sme umožnili prihlasovanie do aplikácie deťom, ktoré ešte majú problémy s písaním a čítaním. Výsledkom je modul registrácie a prihlasovania, ktorý je možné nasadiť na webovú aplikáciu. Obsahuje všetky vyššie spomenuté metódy a po registrácii ukladá používateľov do databázy.

Kľúčové slová: autentifikácia, Facebook, Google, QR kód, spektrogram

Abstract

The aim of this bachelor thesis was to create a comprehensive module for registration and login of a web application. Users register for the web application using several methods. These methods include local registration with username, password and email, registration via Facebook, Google and a University account. Registered users are stored in a local database. After registration, users can log in to the web application using the registered method. Registered and logged in users will be able to add two more methods through which they can log in. These methods are logging in by voice and logging in using a QR code. Using the added methods, we made it possible for children who still have problems writing and reading to log in to the application. The result is a registration and login module that can be deployed on a web application. It contains all the above-mentioned methods and stores users in the database after registration.

Keywords: authentication, Facebook, Google, QR code, spectrogram

Obsah

Úvod	1
1 Východiská	3
1.1 Meno a heslo	3
1.2 Aplikácie tretích strán	4
1.2.1 Saml 2.0	4
1.2.2 OAuth 2.0	4
1.2.3 Facebook	4
1.2.4 Zaregistrovanie aplikácie vo Facebooku	6
1.2.5 Google účet	6
1.2.6 Zaregistrovanie aplikácie v Google	8
1.3 QR kód	8
1.3.1 Zloženie QR kódu	8
1.3.2 Informácie v QR kóde	9
1.3.3 Základný prehľad procesu skenovania	10
1.4 Biometrické overovanie	10
1.4.1 Základné typy biometrického overovania	11
1.5 Problém pri registrácii detí	12
1.6 Registrácia pomocou mena, hesla a emailu	12
1.7 Registrácia používateľa vždy osobitnou metódou	12
1.8 Súčasný stav	13
2 Návrh	14
2.1 Roly používateľov	14
2.1.1 Prihlasovanie staršieho študenta	14
2.1.2 Prihlasovanie mladšieho študenta	15
2.1.3 Problém pri registrácii iba cez jednu metódu	16
2.2 Pridávanie a odstraňovanie metód	16
2.3 Použité nástroje a knižnice	16
2.3.1 Javascript	17
2.3.2 Node.js	17

2.3.3	Socket.io	17
2.3.4	bcrypt.js	17
2.3.5	Sequelize	17
2.3.6	Passport.js	17
2.3.7	Nodemailer	18
2.4	Databáza	18
2.4.1	Accounts	18
2.4.2	FacebookAccounts, GoogleAccounts, FMFIAccounts	19
2.4.3	VoiceAccounts	19
2.4.4	QrAccounts	19
2.4.5	LocalAccounts a EmailConfirms	19
3	Implementácia	20
3.1	Passport.js	20
3.2	Spektrogram	20
3.3	Registrácia metódou hlasu	21
3.3.1	Vypočítanie energie užívateľa	23
3.3.2	Prihlásenie metódou hlasu	24
3.3.3	Vzdialenosť polí	24
3.3.4	Vykreslenie spektrogramu	24
3.3.5	Testovanie registrácie a prihlásenia hlasom	25
3.4	Registrácia metódou QR kód	26
3.4.1	Implementácia QR kódu	27
3.4.2	Prihlásenie metódou QR kódu	27
3.5	Registrácia verzus pridávanie metódy	28
3.6	Registrácia meno, heslo a email	28
3.7	Facebook a Google	29
3.8	Implementácia Univerzitného prihlasovania FMFI	31
3.9	Používateľská príručka	32
	Záver	34
	Príloha A	38

Zoznam obrázkov

1.1	Facebook prihlásenie - používateľ ešte nie je prihlásený v prehliadači	5
1.2	Facebook prihlásenie - používateľ je prihlásený v prehliadači	5
1.3	Facebook Login Diagram [6]	6
1.4	Facebook App	6
1.5	Google prihlásenie - používateľ ešte nie je prihlásený v prehliadači	7
1.6	Google API [7]	7
1.7	Google App	8
1.8	QR kod	9
1.9	Modul prihlásenia a registrácie na stackoverflow.com [13]	13
2.1	Modul registrácie	14
2.2	Modul výberu možnosti prihlasovania	15
2.3	Modul prihlásenia	16
2.4	Relačný model databázy	18
3.1	Spektrogram [21]	21
3.2	Vzorkovacia frekvencia	23
3.3	Redukcia poľa	23
3.4	Spektrogram v aplikácií	25
3.5	QR kód	26
3.6	Poškodený QR kód	27

Úvod

V dnešnom modernom svete sa život presúva čoraz viac do online priestorov, čo má za následok vytvorenie množstva internetových stránok, ktoré používajú milióny ľudí po celom svete. Niektoré z týchto internetových stránok slúžia iba na prehliadanie, preto nie je pre používateľov potrebné ukladať obsah. Pre zložitejšie webové aplikácie sa musí používateľ autentifikovať, aby vedela webová aplikácia určiť, čo môže používateľ v danej aplikácii robiť. Základné pravidlo, ktoré platí pri práci na internete je, že každý používateľ by si mal zvoliť pre každú aplikáciu vždy iné heslo. Pri používaní viacerých takýchto zložitejších aplikácií sa môže stať, že pri tom množstve použitých hesiel ich používateľ môže ľahko zabudnúť. Preto sa v poslednej dobe začala dostávať do popredia autentifikácia pomocou tretej strany. Pri používaní internetu môžeme často vidieť, ako pri prihlásení máme okrem mena a hesla na výber aj prihlásiť sa cez Facebook[1] alebo Google[2]. To sú práve aplikácie tretích strán, kde sa zakladá na dôvere v tieto aplikácie.

S nárastom počtu webových aplikácií sa vytvorili aj aplikácie, ktoré sú určené deťom. Pre mladšie deti, ktoré majú ešte problém s čítaním a písaním je často náročné zapamätať si heslo a napísať ho na klávesnici. Rovnako majú zakázanú registráciu v aplikáciách tretích strán. Preto je potrebné pomôcť aj takýmto užívateľom webových aplikácií a nájsť spôsob, ako sa jednoducho, ale bezpečne autentifikovať do webovej aplikácie.

Dnešné moderné zariadenia nám neprinášajú pomoc pri autentifikácii iba pomocou softvérových riešení, ale taktiež aj vylepšenie hardvérových vlastností zariadení nám môže pomôcť pri riešení autentifikácie používateľov. Dnes máme web kamery, ktoré dokážu detailne rozpoznať človeka podľa tváre, mikrofón, ktorý dokáže snímať vlastnosti ľudskej reči alebo skener odtlačku prstov, ktorý dokáže v momente identifikovať človeka.

Cieľom tejto bakalárskej práce je vytvoriť všestranný modul prihlasovania webovej aplikácie s experimentálnym využitím biometrie človeka. V aplikácii sa bude taktiež možné autentifikovať pomocou aplikácie tretej strany. Používateľ si bude môcť vybrať aj viac možností, ktorými sa bude prihlasovať. Pre používateľov, ktorí nechcú používať pridané metódy, bude stále dostupná klasická autentifikácia menom a heslom.

Hlavná časť práce je rozdelená na tri kapitoly. V prvej kapitole opisujeme základné rozdelenie biometrických možností prihlasovania, fungovanie aplikácií tretích strán a

opíšeme si aj klasickú formu mena a hesla. Druhá kapitola opisuje architektúru a popisuje databázový model, kde budú uložení používatelia. V tretej kapitole si popíšeme, ako sme aplikáciu implementovali a na aké problémy sme narazili. Tretia kapitola tak tiež popisuje používateľskú príručku.

Kapitola 1

Východiská

V dnešnej dobe si každý človek chce uchovať svoje súkromie čo najviac v tajnosti. Z toho dôvodu je zrejmé, že všetci používatelia internetových služieb potrebujú mať nejaké údaje, prostredníctvom ktorých sa v službách na internete autentifikujú a potvrdia, že sú to naozaj oni. Používateľ sa môže overiť troma základnými spôsobmi, a to sú:

- Používateľ pozná kľúč - heslo
- Používateľ ma jedinečnú vlastnosť - hlas, oko, odtlačok prsta
- Používateľ vlastní kľúč - QR kód na kartičke

Následne po autentifikácii nastáva autorizácia. Autorizácia určuje, čo môže autentifikovaný používateľ vidieť a čo všetko môže robiť vo webovej aplikácii.

1.1 Meno a heslo

Autentifikácia pomocou mena a hesla sa zdá byť najjednoduchší spôsob prístupu do aplikácie. Avšak ľahko sa však môže stať, že používateľ zabudne svoje heslo a tým pádom navždy stratí prístup do aplikácie. V minulosti si užívateľ internetovej služby mohol zvoliť heslo bez požiadaviek na prístup do aplikácie. Kto si nebol vedomý hrozieb uhádnutia hesla iným človekom, ten si zvolil základné jednoduché heslo tvorené malými písmenami, alebo zvolil len čísla. To malo za následok nedostatočné zabezpečenie účtu a účet tak bol ľahko napadnuteľný. Postupom času vývojári internetových aplikácií začali pridávať rôzne požiadavky na heslo, ako napríklad povinnosť zvoliť aspoň jedno veľké písmeno z abecedy, povinnosť zmiešať písmená a čísla, alebo pridať k heslu nejaký iný znak ako je v abecede.

1.2 Aplikácie tretích strán

V tejto časti si opíšeme, ako sa využívajú aplikácie ako Facebook, Google, Twitter[3] vo vlastnej internetovej službe. Využitie aplikácií tretích strán dáva užívateľom veľkú výhodu v tom, že si nemusia pamätať žiadne nové prihlasovacie mená a heslá.

1.2.1 Saml 2.0

SAML(Security Assertion Markup Language) je štandard na výmenu autentifikačných a autorizačných údajov medzi bezpečnostnými doménami. SAML 2.0 je vylepšenie verzie SAML 1.0. SAML 2.0 je protokol, ktorý je založený na XML(Extensible Markup Language), ktorý používa bezpečnostné tokeny, ktoré obsahujú potvrdenia na prenos informácií o používateľovi medzi poskytovateľom identity(Identity Provider) a poskytovateľom služieb (Service Provider)[5]

1.2.2 OAuth 2.0

OAuth 2.0 (RFC 6749) je aplikačný rámec, ktorý umožňuje aplikáciám tretej strany získať obmedzený prístup k službe HTTP a to buď v mene vlastníka prostriedku, alebo povolí aplikácií tretej strany získať prístup vo svojom mene [7]

1.2.3 Facebook

Facebook je sociálna sieť, cez ktorú môžu ľudia komunikovať prostredníctvom chatu, hlasovým hovorom, alebo video hovorom. Používajú ju milióny ľudí na svete, preto je túto metódu vhodné pridať do aplikácií s potrebou autentifikovať sa. Užívateľ, ktorý sa chce prihlásiť do aplikácie, musí mať vytvorený účet na Facebooku. Pri kliknutí na tlačidlo prihlás cez Facebook sa používateľ dostane na prihlasovací modul Facebooku, kde ak je prihlásený tak užívateľa to vráti naspäť do našej aplikácie vid' obr.1.2. Ak tam ešte prihlásený nie je, ľahko vyplní potrebné údaje, a následne ho prihlási do našej aplikácie vid' obr.1.1.

Diagram na obrázku 1.3 znázorňuje, ako aplikácie využívajú autentifikáciu pomocou Facebooku.

1. Pri vstupe na link, alebo uvítaciu webstránku sa zobrazí tlačidlo na prihlásenie službou Facebook. Používateľ klikne na toto tlačítko aby sa prihlásil do webovej aplikácie. Po kliknutí bude presmerovaný na Facebook.

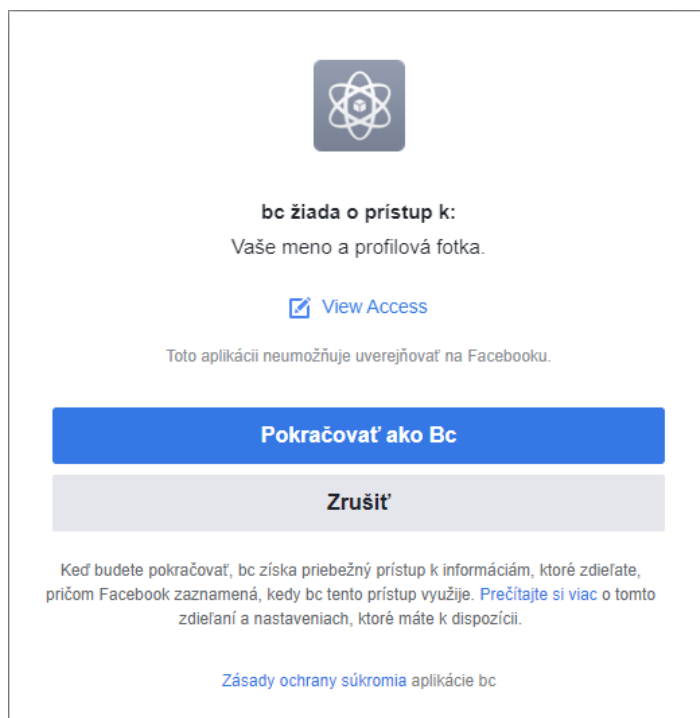
2. Facebook overí ID žiadosti a potom užívateľa presmeruje na prihlasovaciu stránku.

1. Používateľ vloží Facebookové prihlasovacie údaje a potvrdí formulár.

2. Facebook overí prihlasovacie údaje a vytvorí požiadavku na presmerovanie na `redirect_url`. `Redirect_url` je link do našej aplikácie, ktorá sa už postará o zvyšok



Obr. 1.1: Facebook prihlásenie - používateľ ešte nie je prihlásený v prehliadači



Obr. 1.2: Facebook prihlásenie - používateľ je prihlásený v prehliadači

spracovania.

3. Internetový prehliadač otvorí `redirect_url`.

4. Webstránka na adrese `redirect_url` opäť zavolá Facebookovému serveru s požiadavkou na `access_token`.

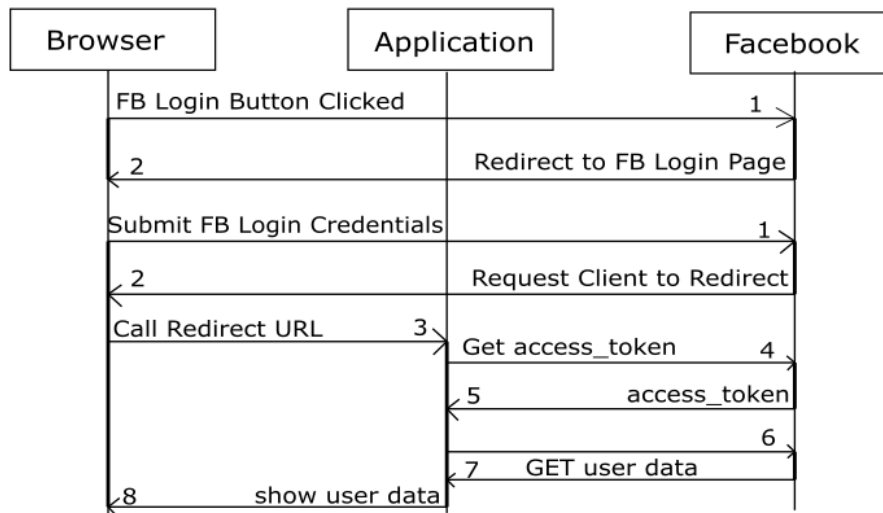
5. V prípade, že overenie užívateľa na Facebooku je úspešné, server odošle späť `access_token`.

6. Po obdržaní `access_tokenu` aplikácia opäť zavolá Facebook s požiadavkou na informácie o užívateľovi, do požiadavky priloží `access_token`.

7. Facebook po overení `access_tokenu` pošle späť informácie o užívateľovi.

8. Naša aplikácia presmeruje užívateľa na stránku, ktorá užívateľovi zobrazí informácie, ktoré o ňom má.

[6] [4]



Obr. 1.3: Facebook Login Diagram [6]

1.2.4 Zaregistrovanie aplikácie vo Facebooku

Aby sa mohla používať autentifikácia cez Facebook vo vlastnej internetovej službe, je potrebné Facebooku povedať, ako sa bude aplikácia volať a na čo sa bude používať. Na webovej stránke <https://developers.facebook.com/> sa vytvorí nová aplikácia, pomocou ktorej sa získava App ID a App Secret obr.1.4, cez ktoré sa bude internetová služba identifikovať Facebooku.

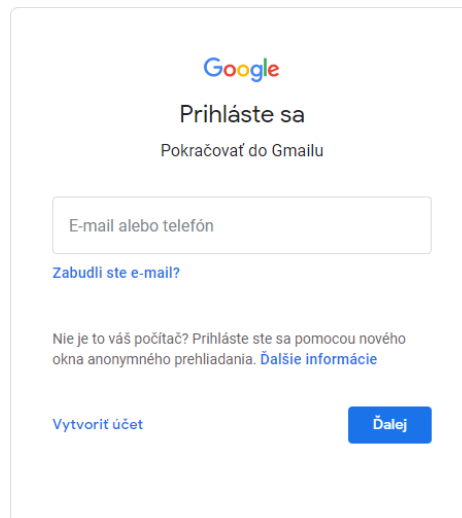
App ID 415277056169601	App Secret 405a739d32e1b805343869ed1240c522 Reset
Display Name bc	Namespace
App Domains localhost	Contact Email kohar.f@gmail.com

Obr. 1.4: Facebook App

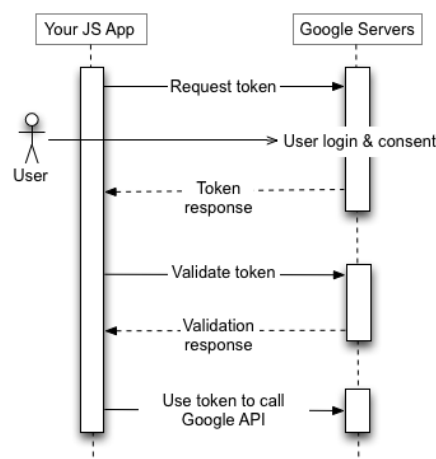
1.2.5 Google účet

Google účet používajú podobne ako Facebook milióny ľudí na svete. Pri registrácii na Google účet dostávame niekoľko užitočných aplikácií, ktoré môžeme využívať. Máme k dispozícii email, kde vieme komunikovať. Podstatná časť tohoto účtu je aj Google Disk, kde si vieme uložiť rôzne súbory, vieme ich zdieľať s inými používateľmi. Disk nám zabezpečí, že aj pri poruche počítača naše súbory ostanú nedotknuté a chránené. Pri vyhľadávaní rôznych informácií na internete sa používa najpopulárnejší Google vyhľadávač, kde na vyhľadávanie informácií nemusíme byť ani zaregistrovaný cez Google účet. Preto je vhodné aj túto metódu prihlasovania pridať do svojej internetovej služby.

Google pracuje podobne ako Facebook pri prístupe do našej aplikácie. Pri stlačení tlačidla Prihlás cez Google sa používateľ presmeruje na prihlasovací modul Google, kde ak nie je prihlásený, vyplní svoj email a heslo. Google skontroluje, či je všetko správne, a presmeruje ho to do našej aplikácie ako prihláseného používateľa. Ak už je v prehliadači prihlásený, nemusí vyplňovať meno a heslo, ale bude automaticky prihlásený v našej internetovej službe.



Obr. 1.5: Google prihlásenie - používateľ ešte nie je prihlásený v prehliadači

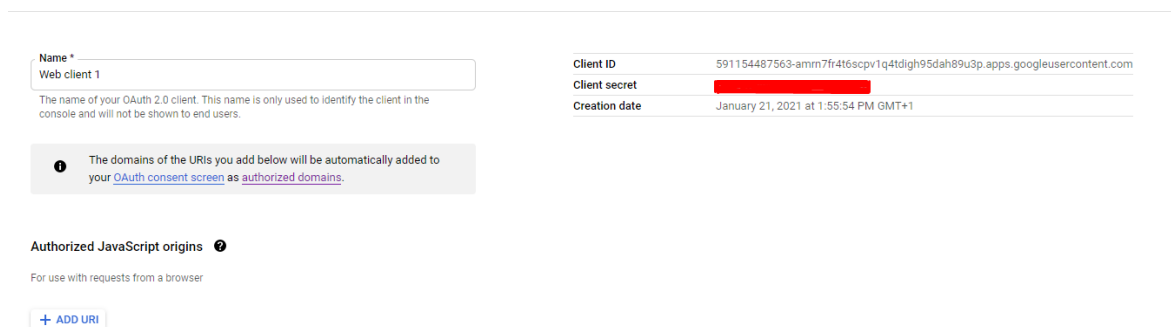


Obr. 1.6: Google API [7]

Na obrázku 1.6 si môžeme lepšie pozrieť, ako funguje celá autentifikácia pomocou Google. Autorizácia začína tým, že aplikácia presmeruje prehliadač na Google, kde sa posielajú parametre, ktoré označujú typ požadovaného prístupu. Na strane Google sa spracováva autentifikácia používateľa, výber relácie a súhlas používateľa. Ako výsledok dostávame prístupový token, ktorý by mal klient pred zahrnutím do žiadosti overiť. Ak nastane vypršanie platnosti tokenu, aplikácia proces zopakuje.[7]

1.2.6 Zaregistrovanie aplikácie v Google

Podobne, ako vo Facebooku, aj v Google je potrebné internetovú službu najskôr zaregistrovať, tentokrát na webovej stránke <https://console.developers.google.com>, kde tiež aplikácia získava `Client ID` a `Client Secret` obr 1.7. Tie budú potrebné na identifikovanie sa aplikácie pre službu Google.



Obr. 1.7: Google App

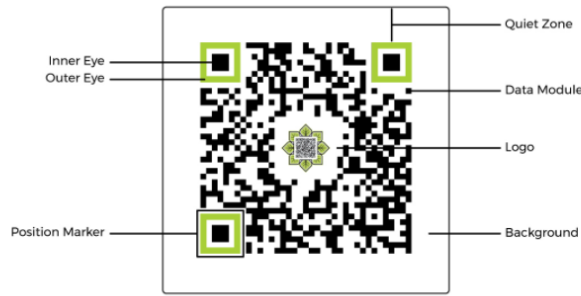
1.3 QR kód

QR kód (quick response code) je typ čiarového kódu, ktorý poznáme z produktov v supermarkete. QR kód sa stará o zakódovanie informácií do usporiadaných štvorcov. Tieto informácie zo štvorcov je následne možné odkódovať do informácie. Oproti klasickému čiarovému kódu má QR kód značné výhody, ako napríklad veľkosť dát uložených v kóde. Ďalšia veľká výhoda je, že nie je potrebné vlastniť žiadny špeciálny nástroj, ale stačí použiť akýkoľvek dostupný smartfón. Taktiež prečítanie a spracovanie QR kódu je rýchlejšie ako klasický čiarový kód. Najbežnejšie farby QR kódu sú čierne štvorce na bielom pozadí, avšak pri tvorbe vlastného QR kódu sa dajú určiť ľubovoľné farby.[8]

1.3.1 Zloženie QR kódu

V tejto časti si ukážeme, aké informácie vieme vyčítať z QR kódu pri pohľade naň.

- Data module (dátový modul) - je to štandardná jednotka QR kódu, kde sú uložené zakódované informácie. Usporiadanie týchto dátových modulov tvorí väčšinu QR kódu.
- Position marker (značka polohy) - Na každom QR kóde sa nachádzajú tri značky polohy ktoré sú zložené z `inner eye` (vnútorné oko) a `Outer eye` (vonkajšie oko).



Obr. 1.8: QR kod

Tieto značky polohy sú dôležité na to, aby skener vedel rýchlo určiť pozíciu QR kódu a lokalizovať smer skenovania.

- Quiet zone (čistá zóna) - Tichá oblasť je prázdna časť QR kódu okolo celého dátového modulu a značiek polôh. Používa sa na pomoc skenerom lepšie určiť miesto, kde začína a končí QR kód.

1.3.2 Informácie v QR kóde

V QR kóde máme tiež uložené informácie o QR kóde ako veľkosť, úroveň opravy chýb a dátový typ.[5]

1. Veľkosť - vzrastajúcou popularitou QR kódov sa ich produkcia postupne zvyšovala, preto bolo potrebné zväčšiť ich veľkosť v počte riadkov a stĺpcov. Najmenšie QR kódy aké poznáme majú 21 stĺpcov a 21 riadkov čo umožňuje mať 441 dátových modulov. Táto verzia s 21 riadkami a 21 stĺpcami je označovaná ako verzia 1. Druhá, väčšia verzia je zložená z 25 riadkov a 25 stĺpcov. To znamená, že do QR kódu verzie 2 sa vojde 625 dátových modulov. Postupným pokračovaním sa dostaneme až do verzie 40, ktorá sa skladá z 177 riadkov a 177 stĺpcov. Tu sa vojde až 31329 dátových modulov.
2. Úroveň opravy chýb - Pri používaní QR kódu vo vytlačenej forme sa môže stať, že časť kódu zašpiníme alebo inak poškodíme, čo by malo znefunkčnúť celý QR kód. Avšak QR kód obsahuje informáciu o úrovni opravy chýb. Preto bude pri vhodnom nastavení možné použiť aj poškodený kód, aj keď z neho nejakú časť úplne odstránime. Princíp je taký, že z hlavného QR kódu sa vyberú jedinečné body na jednoznačné určenie kódu. Tieto jedinečné body budú uložené do pôvodného QR kódu ako záloha. Poznáme 4 úrovne korekcie chýb v QR kóde, ktoré určujú aké množstvo záložných údajov sa uloží v QR kóde. To znamená, že čím väčšia úroveň korekcie, zvýši sa aj počet riadkov a stĺpcov potrebných na uloženie jedinečných bodov. Tieto úrovne sú nasledovné:

- (a) L - dovoľuje poškodenie do 7%
 - (b) M - dovoľuje poškodenie do 15%
 - (c) Q - dovoľuje poškodenie do 25%
 - (d) H - dovoľuje poškodenie do 30%
3. Dátový typ - QR kódy môžu obsahovať až 7 089 číselných znakov alebo 2 953 alfanumerických znakov. Môžu tiež ukladať bajty a kanji, ale tie sa používajú menej často. Tieto čísla predpokladajú najnižšiu úroveň korekcie chýb. To znamená, že použitie QR kódu zahŕňa čokoľvek, čo na komunikáciu používa čísla, písmená, interpunkciu a symboly.

1.3.3 Základný prehľad procesu skenovania

1. Skener QR kódov najprv rozpozná 3 značky polohy v QR kóde. Pri použití vhodnej čistej zóny sa skener rýchlo zorientuje a je informovaný o tom, kde sú okraje QR kódu.
2. Skener začína vpravo dole, kde sa nachádza indikátor režimu, ktorý označuje, akým dátovým typom je určený zvyšok kódovaných údajov. Indikátor režimu je zložený zo štyroch dátových modulov.
3. Po zistení potrebných údajov z indikátoru režimu sa skener posunie vyššie na indikátor počtu znakov, ktorý je zložený z ôsmich dátových modulov. Tieto dátové moduly označujú, počet znakov kódovaných údajov.
4. Následne po získaní identifikátorov počtu znakov a režimu sa skener posúva ďalej cez dátové moduly až kým nenarazí na koncový identifikátor.
5. Po prečítaní koncového identifikátoru skener pokračuje na dátové moduly, kde je uložená úroveň opravy chýb.

1.4 Biometrické overovanie

Biometrické overenie sa používa v oblasti zabezpečenia, kde si používateľ nemusí pamätať žiadne prihlasovacie meno, heslo alebo email. Namiesto toho sa používajú biometrické vlastnosti používateľa, pomocou ktorých sa overuje, či má osoba prístup ku konkrétnemu zariadeniu. Každý človek má v tele nejaké jedinečné fyzikálne a biologické vlastnosti, ktoré sa ľahko dajú porovnať s predtým nahratými vlastnosťami v databáze. Prihlasovanie pomocou biometrie sa nepoužíva iba v počítačoch, ale často slúži aj ako vstup cez dvere alebo brány. Aj pre moderné smartfóny platí, že výrobcovia sa snažia pridať do svojich zariadení biometrické overenia na vstup to telefónu. Najprv to bol

snímač odtlačku prstu na zadnej strane telefónu, postupom času sa to prepracovalo na snímač odtlačku prstu vložený priamo v displeji mobilného telefónu. Taktiež sa začalo v najnovších telefónoch používať overenie pomocou rozpoznávania tváre.

1.4.1 Základné typy biometrického overovania

V tejto časti si popíšeme základné typy biometrických overovaní, ktoré sa v súčasnosti používajú najviac.

Skenery odtlačkov prstov

Princíp skenovania odtlačkov prstov je založený na princípe, ktorý sa používal už dávno, keď sa pomocou atramentu a papiera porovnávali odtlačky prstov. Každý jednotlivec má iný odtlačok prsta, čo dáva výhodu jednoznačne určiť každého jednotlivca. Môže to mať však aj nevýhody. Jedným z príkladov nevýhody je, ak si používateľ poraní prst. Vtedy sa môže časť kože spáliť alebo úplne odstrániť. Novšie verzie skenerov odtlačkov prstov sa vedia dostať až pod kožu, kde snímajú elektrické signály pod kožou, čo pomáha pri bezpečnosti a spoľahlivosti tohoto systému. Je to lepšie preto, pretože pri obyčajnom porovnávaní odtlačku prsta môže niekto umelo vytlačiť odtlačok prsta, ktorý len priloží k snímaču. [9]

Rozpoznávanie tváre

Táto technológia funguje na princípe porovnávania s pôvodnou schválenou tvárou v databáze. Technológia vykonáva desiatky meraní z ktorých vytvára odtlačky tváre používateľa, ktorý sa snaží získať prístup. Ak sa dostatočný počet odtlačkov tváre zhoduje so schválenou tvárou, vtedy je udelený prístup. Biometrické overenie pomocou rozpoznávania tváre je však zložitý problém, pretože porovnávanie nemusí byť správne pri pohľade s iných uhlov, alebo môže mať problém s rozpoznávaním dvoch podobne vyzerajúcich ľudí.

Identifikácia hlasu

Táto technológia sa používa na overenie osoby pomocou rozpoznávania jej hlasových vzorov. Identifikácia hlasu funguje veľmi dobre hlavne preto, že každý človek sa svete má jedinečné fyzické, fonetické aj morfológické vlastnosti. Preto je bezpečnosť tejto technológie veľmi vysoká, keďže je veľmi odolná voči podvodom. Veľká výhoda tejto technológie je dostupnosť na zariadeniach. Mikrofóny sú prakticky v každom mobilnom telefóne, nemusia mať fotoaparát alebo iné snímače. Taktiež mikrofón obsahuje veľká časť notebookov. Ak sa náhodou stane, že používateľ má bežný stolný počítač

bez mikrofónu, mikrofón sa dá získať ľahko po napojení slúchadiel, kde aj tie cenovo najdostupnejšie mikrofón obsahujú.[11]

Očné skenery

Prvý typ očných skenerov je skener sietnice, ktorý funguje tak, že smerom do oka premieta jasné infračervené svetlo, pomocou ktorého vytvára viditeľné vzory krvných ciev, ktoré skener číta a porovnáva s informáciami v databáze. Druhý typ očného skenera je skener na rozpoznávanie dúhovky. Tento skener funguje podobne ako skener sietnice, avšak tu neporovnáva krvné cievy, ale hľadá jedinečné vzory v farebnom kruhu okolo očnej zrenice. Nevýhodou tejto technológie je, že pre človeka, ktorý musí mať okuliare alebo nosí kontaktné šošovky je to nepoužiteľné.[10]

1.5 Problém pri registrácii detí

Deti, ktoré ešte nevedia dobre čítať a písať, budú mať problém sa dostať do akejkoľvek internetovej služby. Môžu mať problém nezapamätať si heslo alebo prihlasovacie meno. Preto je vhodné myslieť aj na ne a umožniť im jednoduchý, ale bezpečný spôsob ako tieto služby využívať. Ako dobrá možnosť, ktorá pomôže deťom v prihlasovaní by mohla byť autentifikácia pomocou ich hlasu, alebo QR kódom uloženým na kartičke.

1.6 Registrácia pomocou mena, hesla a emailu

Ak je súčasťou metódy spomenutej vyššie(meno a heslo) aj povinnosť zadania emailu, tak zabudnuté heslo už nebude žiadny problém, pretože používateľ si môže ľahko poslať nové heslo na email a podľa inštrukcií si môže znova nastaviť prístupové údaje také, aké chce. Niekedy sa môže stať, že dôsledkom nepozornosti užívateľ vyplní zlé emailovú adresu, čo bude pre užívateľa ten istý problém ako keby žiadnu emailovú adresu nezadával. Preto je vhodné vymyslieť spôsob aby sa to nestalo, napríklad povinnosť potvrdiť verifikačný link v emaile. Ak náhodou mail nepríde, užívateľ bude mať informáciu o tom, že jeho prihlasovacie údaje nie sú v poriadku.

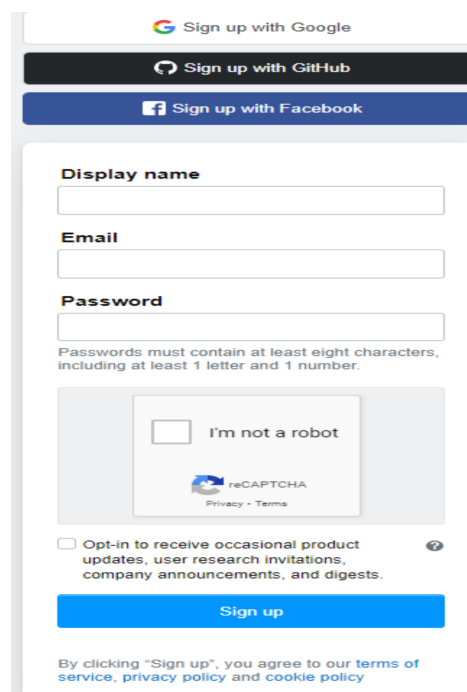
1.7 Registrácia používateľa vždy osobitnou metódou

Ak sa používateľ zaregistruje pomocou viacerých metód, bude to mať za následok viac účtov v našej internetovej službe. To niekedy nechceme, pretože to môže mať za následok, že na jednom účte si uloží svoju prácu, ale pri prihlásení cez iný účet neuvidí svoju prácu. Preto je dobré vymyslieť spôsob ako tomu zabrániť. Za vhodné riešenie by sa mohlo považovať zakázanie registrácie viacerými metódami. To však nie

je možné, pretože každá metóda má potrebné iné údaje. Kvôli tomu internetová služba nevie zistiť, ktorý zákazník sa chce registrovať.

1.8 Súčasný stav

Ukážku integrácie modulov prihlasovania môžeme nájsť pri navštívení webovej stránky www.stackoverflow.com. Na stránke sa dá registrovať pomocou Facebooku, Google a Github účtu a pomocou mena a hesla. Pri zaregistrovaní cez meno, heslo a email je nutné potvrdiť aktiváciu. To znamená, že je potrebné sa prihlásiť na email, a kliknúť na aktivačný kód, ktorý následne používateľa presmeruje na stránku a až vtedy je možné ju využívať ako prihlásený člen. Pri opätovnom pokuse o registráciu rovnakého emailu používateľa systém upozorní, že už registrovaný bol a vypýta si emailovú adresu, kde zašle návod na obnovenie účtu. Ak užívateľ zadá iný email ako pri prvotnej registrácii, zmení sa mu aj v jeho účte na stackoverflow.com. Pri použití Facebook účtu používateľa presmeruje na Facebookovskú stránku, kde vyplní email a heslo. Následne to vráti už prihláseného užívateľa. Ak užívateľ teraz použije prihlasovanie Google účtom, stránka ho presmeruje na stránku Google kde sa prihási a to ho vráti ako prihláseného užívateľa. Ak systém zistí, že email na Facebooku a na Google sú zhodné, tak sa použije účet skôr registrovaného účtu. Zobrazený modul registrácie je na obrázku 1.9



Sign up with Google

Sign up with GitHub

Sign up with Facebook

Display name

Email

Password

Passwords must contain at least eight characters, including at least 1 letter and 1 number.

I'm not a robot

reCAPTCHA

[Privacy - Terms](#)

Opt-in to receive occasional product updates, user research invitations, company announcements, and digests.

Sign up

By clicking "Sign up", you agree to our [terms of service](#), [privacy policy](#) and [cookie policy](#)

Obr. 1.9: Modul prihlásenia a registrácie na stackoverflow.com [13]

Kapitola 2

Návrh

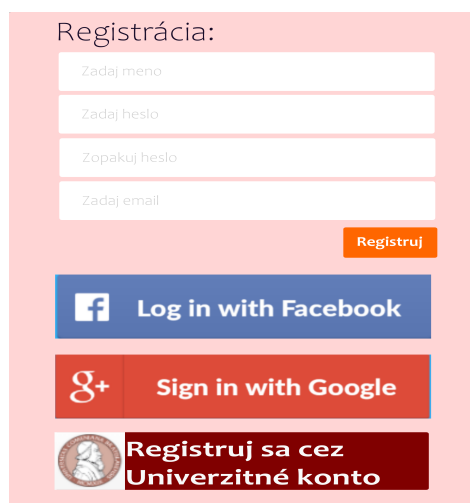
V časti návrh si opíšeme, ako bude prihlasovací modul vyzerat', a detailnejšie si opíšeme navrhnutý databázový model.

2.1 Roly používateľov

- starší študent** - študent nad 16 rokov
- mladší študent** - študent pod 16 rokov
- zodpovedná osoba** - rodič, alebo zástupca mladšieho študenta

2.1.1 Prihlasovanie staršieho študenta

Starší študent sa registruje pomocou jednej zo štyroch hlavných metód. Čiže cez Facebook, Google, Univerzitné konto a registrácia menom, heslom, emailom. Modul je zobrazený na obrázku 2.1.



Registrácia:

Zadaj meno

Zadaj heslo

Zopakuj heslo

Zadaj email

Registruj

Log in with Facebook

Sign in with Google

Registruj sa cez Univerzitné konto

Obr. 2.1: Modul registrácie

Používateľ si vyberie jednu z nich, ak to bude Facebook, Google, Univerzitné konto, tak ho aplikácia presmeruje na prihlásenie sa do konkrétnej metódy, odkiaľ dostaneme potrebné informácie na zaregistrovanie sa do portálu. Ak si zvolí registráciu emailom, musí zadať používateľské meno, heslo a email. Po registrácii bude musieť ísť na svoj email, kde potvrdí aktivačný link. Po vstupe do portálu sa používateľovi zobrazí modul na výber prihlasovacích možností, kde si môže zvoliť, cez ktoré metódy sa chce prihlasovať v budúcnosti. Modul je zobrazený na obrázku 2.2.



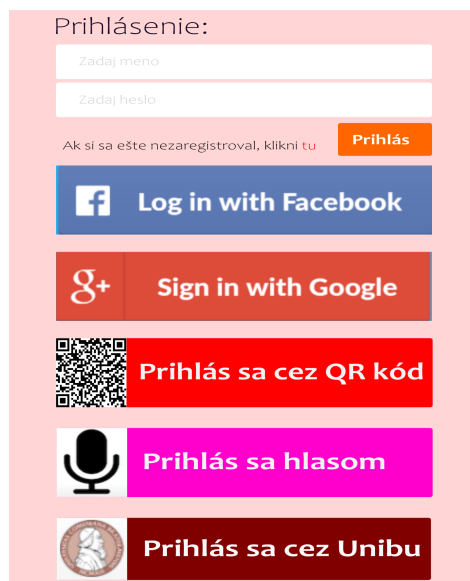
Tvoje možnosti prihlásenia:	
Facebook	<input checked="" type="checkbox"/>
Google	<input type="checkbox"/>
Uniba	<input type="checkbox"/>
QR kód	<input type="checkbox"/>
Hlas	<input type="checkbox"/>
Meno a heslo	<input type="checkbox"/>

Obr. 2.2: Modul výberu možnosti prihlasovania

Postup bude rovnaký ako pri registrácii, avšak tu už používateľ nebude môcť vytvoriť dva rôzne účty, ale všetky metódy prihlasovania sa uložia k jeho id v aplikácii. Vždy je potrebné, aby bola zvolená aspoň jedna možnosť prihlasovania. V týchto možnostiach pridáme aj prihlasovanie pomocou rozpoznanie hlasu a prihlasovanie pomocou QR kódu. Pri budúcom prístupe na stránku sa pre prihlásenie zobrazí nasledovný modul prihlásenia vid' obr.2.3

2.1.2 Prihlasovanie mladšieho študenta

Pri registrácii mladšieho študenta chceme, aby mala k jeho účtu prístup aj jeho zodpovedná osoba. To zabezpečíme tak, že rodič sa zaregistruje cez svoju obľúbenú metódu. Následne po vstupe do portálu a v module pre možnosti prihlásenia zvolí jednu z metód - prihlasovanie cez QR kód alebo prihlasovanie pomocou hlasu. Týmto zabezpečíme, že mladšiemu študentovi bude stačiť pri prihlásení ukázať jeho vygenerovaný QR kód na kameru, alebo povedať heslo do mikrofónu. Ak sa stratí QR kód, stačí, keď sa zodpovedná osoba prihlási do portálu, a tento kód si môže vygenerovať nanovo.



Obr. 2.3: Modul prihlásenia

2.1.3 Problém pri registrácií iba cez jednu metódu

V súčasnom svete chceme, aby sme mali všetky informácie hneď, rovnako požadujeme okamžitý prístup ku všetkým našim potrebám. Ak sa používateľ zaregistruje cez jednu metódu, očakáva, že prístup do našej internetovej služby bude vždy dostupný. Čo však môže nastať je to, že používateľ, ktorý sa registruje napríklad iba cez Facebook, nebude mať prístup k našej aplikácii, pretože v tej dobe môže mať Facebook náhly výpadok. Preto bude vždy lepšie si zvoliť viac metód prihlasovania.

2.2 Pridávanie a odstraňovanie metód

V module výberu možnosti prihlásenia obr. 2.2 bude možné pridané metódy aj vypnúť. Pri registrácií QR kódu ako svojej prihlasovacej možnosti je možné, že používateľ tento QR kód stratí, alebo niekto v jeho neprítomnosti urobí fotografiu QR kódu, musíme zabezpečiť, aby sa dal vygenerovať nový QR kód. Taktiež môže používateľ chcieť zmeniť svoj účet vo Facebook metóde zmeniť na iný. Preto sme sa rozhodli pri vypnutí pridanej metódy vymazať používateľove údaje z databázy a pri ďalšom kliknutí na pridanie metódy sa zapíše do databázy.

2.3 Použité nástroje a knižnice

Aby sme vytvorili funkčnú aplikáciu, musíme použiť vhodné technológie, ktoré budú tiež ľahko upraviteľné v budúcich úpravách aplikácie.

2.3.1 Javascript

Javascript je jazyk prehliadačov. S nárastom tvorby a využívania webových stránok sa z jazyka Javascript stal jeden z najpoužívanejších jazykov. Javascript má niekoľko veľkých výhod. Jednou z veľkých výhod je, že pri deklarácií premenných nemusíme určiť, či pôjde o integer, string alebo iné typy. [14].

2.3.2 Node.js

Node.js je Open-source prostredie Javascriptu, ktoré vykonáva kód mimo webového prehliadača. Slúži na spúšťanie skriptov na strane servera. [15] Taktiež je možné pomocou neho inštalovať rôzne balíčky a knižnice, ktoré nám pomôžu pri práci s vývojom. Inštaluje sa príkazom `npm install`.

2.3.3 Socket.io

Socket.IO je knižnica, ktorá umožňuje v reálnom čase obojsmernú komunikáciu založenú na udalostiach medzi prehliadačom a serverom. [16]

2.3.4 bcrypt.js

Bcrypt je knižnica, ktorá pomáha so zašifrovaním hesla pri vkladaní do databázy a taktiež tieto zašifrované heslá dokáže dešifrovať. [17]

2.3.5 Sequelize

Sequelize je ORM(Object-relational mapping) pre Node.js. Sequelize zjednodušuje prácu s databázou a umožňuje používateľovi nášho modulu zvoliť, akú databázu chce používať. Na výber má z databáz Postgres, MySQL, MariaDB, SQLite a Microsoft SQL Server. Pri práci s databázou nepožívame klasické príkazy pre výber, aktualizáciu alebo odstránenie z databázy, namiesto toho sa používajú príkazy ako `Tabuľka.create`, `Tabuľka.update`, alebo `Tabuľka.destroy`. [18]

2.3.6 Passport.js

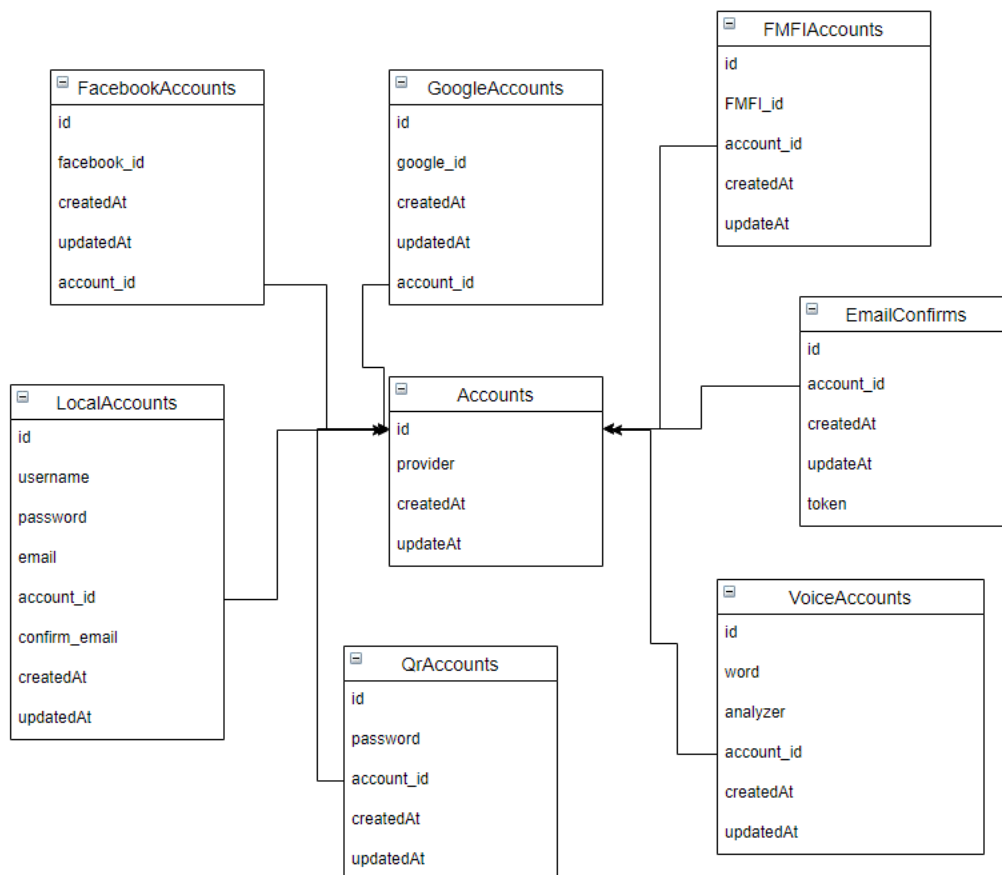
Passport.js je knižnica, ktorá nám pomáha s autentifikáciou používateľa. Má svoje vlastné balíčky ako `passport-facebook`, `passport-local`, `passport-oauth` a `passport-saml`, ktoré potrebujeme použiť v našom module registrácie a prihlasovania webovej aplikácie. [19]

2.3.7 Nodemailer

Keďže potrebujeme, aby používateľ potvrdil svoju registráciu do aplikácie musíme zabezpečiť, aby sme mu boli schopní poslať email na jeho adresu, ktorú zadá pri registrácii. Posielanie emailov vyrieši knižnica nodemailer.js.[12]

2.4 Databáza

Aby bolo možné vytvoriť prihlasovací modul, tak si musíme vytvoriť databázu a v nej uchovávať používateľov. Relačný model databázy je navrhnutý tak, aby uchovával iba tie najpotrebnejšie veci pre autentifikáciu. Relačný model databázy si môžeme pozrieť na obrázku 2.4.



Obr. 2.4: Relačný model databázy

2.4.1 Accounts

Tabuľka Accounts obsahuje iba primárny kľúč id typu integer a provider typu varchar. V stĺpci provider budú uložené údaje o tom, ktorou metódou sa používateľ zaregistroval do našej webovej aplikácie. Stĺpce createdAt a updatedAt sa vytvorili

použitím Sequelize.js a určujú, kedy používateľ vytvoril účet, alebo ho aktualizoval. Tieto stĺpce budú typu datetime.

2.4.2 FacebookAccounts, GoogleAccounts, FMFIAccounts

Tabuľky FacebookAccounts, GoogleAccounts, FMFIAccounts sú rovnako štruktúrované. Obsahujú primárny kľúč id, ktorý je typu integer, facebook_id, google_id, FMFI_id sú stĺpce vytvorené pre používateľove id z aplikácie tretej strany. Pre tento stĺpec sme nastavili typ varchar, pretože google_id obsahuje aj nečíselné znaky. Cudzí kľúč account_id sa odkazuje na primárny kľúč tabuľky Accounts.

2.4.3 VoiceAccounts

Keďže každý používateľ si môže nahráť do databázy viac slov, ale pre každé slovo si môže uložiť iba jeden hlasový odtlačok, musíme zabezpečiť unikátne spojenie stĺpcov account_id a word. Do stĺpca analyzer sa budú ukladať polia ako varchar.

2.4.4 QrAccounts

Pre užívateľov, ktorí si zvolia ako možnosť prihlasovania aj metódu QR kódu budú uložený do tabuľky QrAccounts. Na potrebné informácie o používateľovi si potrebujeme uložiť jeho account_id a password, pod ktorým rozumieme náhodne vygenerovaný reťazec.

2.4.5 LocalAccounts a EmailConfirms

Do tabuľky LocalAccounts budeme ukladať používateľov, ktorí sa registrujú do aplikácie pomocou mena, hesla a emailu. Stĺpce pre meno a email musí mať každý používateľ jedinečné, preto im zvolíme vlastnosť UNIQUE. Confirm_email bude slúžiť na rozdelenie používateľov, ktorí svoj email potvrdili a ktorí ešte nie. Tento stĺpec bude typu boolean. Pri registrácii každého nového používateľa nastavíme confirm_email na false a po potvrdení emailovej adresy ho zmeníme na true. Používateľ sa bude môcť prihlásiť iba s potvrdenou emailovou adresou. Na prácu s emailami a potvrdzovacími url linkami použijeme tabuľku EmailConfirms, ktorá bude pozostávať zo stĺpcov token, typu varchar a ako v každej z tabuliek, aj tu sa bude stĺpec account_id odkazovať na hlavnú tabuľku Accounts.

Kapitola 3

Implementácia

V tejto kapitole si popíšeme, ako sme implementovali našu aplikáciu, opíšeme si problémy, na ktoré sme narazili a zaujímavosti, ktoré sa nám podarili dosiahnuť. Na konci kapitoly sa nachádza používateľská príručka, ktorá popisuje, čo používateľ, ktorý chce našu aplikáciu používať potrebuje urobiť pre správne spustenie a funkčnosť modulu.

3.1 Passport.js

Knižnicu Passport.js používame na celé pozadie autentifikácie používateľa. Na to, aby sme passportu určili, že sme od používateľa dostali správne dáta a je možné ho prihlásiť, musíme najprv nastaviť v aplikácii serializáciu a deserializáciu používateľa. Nastavíme to dvoma funkciami ktoré si môžeme pozrieť nižšie. Následne stačí, keď po kontrole používateľových údajov spustíme príkaz `cb(null, user)`; kde `user` je používateľ vybraný z databázy a následne ho passport uloží do session ako prihláseného.[20]

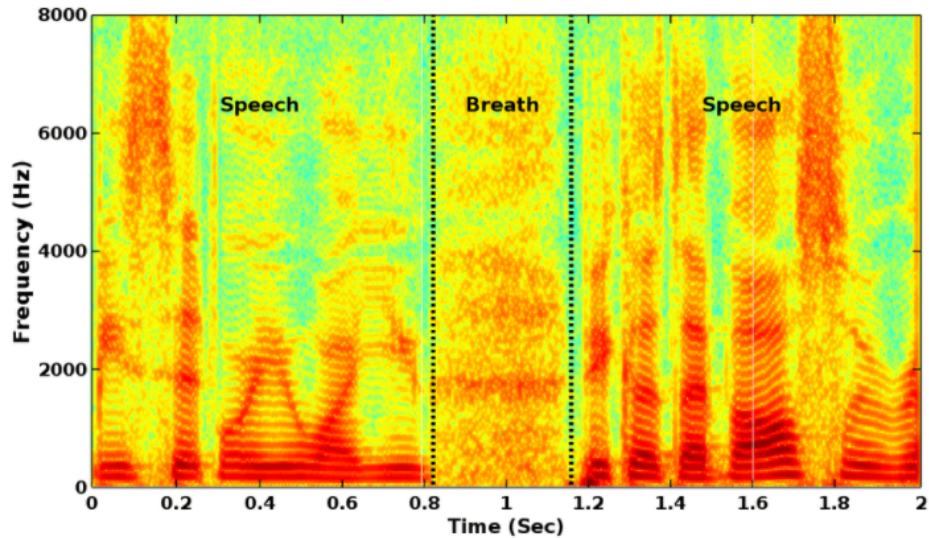
```
1 passport.serializeUser(function (user, cb) {
2     cb(null, user);
3 });
4
5 passport.deserializeUser(function (user, cb) {
6     cb(null, user);
7 });
```

Kód 3.1: Serializácia a deserializácia

3.2 Spektrogram

Od toho času, ako používateľ zapne svoj mikrofón v prehliadači, budeme používateľovi taktiež zobrazovať jeho hlas v grafickej forme. Na vykreslenie hlasu použijeme spektrogram, ktorý podľa výšky čísla jednotlivých frekvencií zafarbí daný pixel na

teplejšie, alebo chladnejšie farby. Obrázok spektrogramu si môžeme pozrieť na obrázku 3.1. Na tomto obrázku vidíme, že čas plynie horizontálne a jednotlivé hodnoty z frekvencií plynú vertikálne. Taktiež si môžeme všimnúť, že najviac zafarbená je časť od 0Hz po 5000Hz, teda predpokladáme, že táto časť je najdôležitejšia na rozpoznanie používateľa.

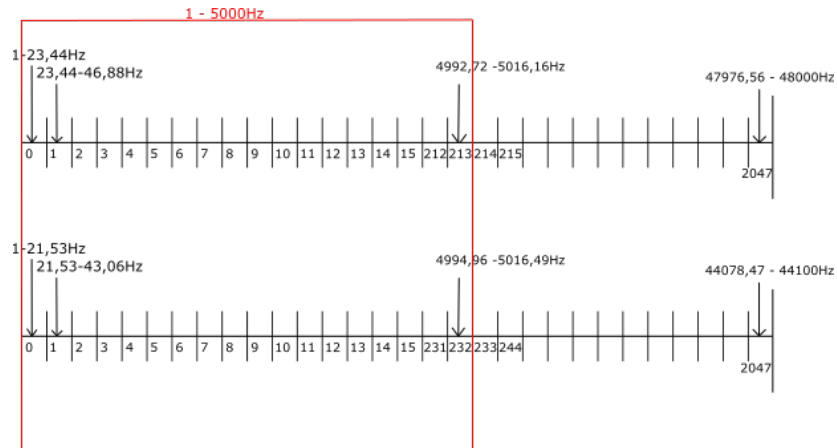


Obr. 3.1: Spektrogram [21]

3.3 Registrácia metódou hlasu

Na registráciu pomocou hlasovej biometrie musí byť používateľ prihlásený v našej webovej aplikácii, takže sa musí najprv registrovať niektorou z iných metód, ktoré aplikácia poskytuje. Týmto získame prístup k používateľovmu id, čo znamená, že používateľ nemusí zadávať žiadne používateľské meno. Od používateľa potrebujeme iba to, aby si vybral nejaké slovo, ktoré ide prečítať a ku ktorému sa priradia jeho jedinečné vlastnosti reči. Po uložení vybraného slova webová aplikácia spustí požiadavku na webový prehliadač s tým, že potrebuje prístup k mikrofónu. Následne sa používateľovi zobrazí výzva na povolenie využitia mikrofónu našou webovou aplikáciou, ak má používateľ pripojených viac mikrofónov, môže si vybrať, ktorý mikrofón sa má použiť. Toto povolenie je potrebné iba vtedy, ak užívateľ prichádza do webovej aplikácie prvý krát. Pri ďalšom navštívení stránky si už webový prehliadač pamätá, že táto aplikácia má prístup povolený. K prístupu k mikrofónu sme použili `getUserMedia`[22], kde sme nastavili, že chceme informácie iba z mikrofónu, pretože web kamera pre nás nie je teraz potrebná. Prvý problém, ktorý nám táto metóda prináša je, že rôzne typy zariadení môžu mať inú vzorkovaciu frekvenciu (sample rate). Vzorkovacia frekvencia vyjadruje počet vzoriek za sekundu, ktoré dokáže mikrofón zachytiť. Notebooky majú štandardnú

vzorkovaciu frekvenciu 44100Hz, pre tablety je to 48000Hz. Tieto štandardné vzorkovacie frekvencie sa využijú vo WebAudioApi [23]. V súčasnej dobe nie je možné tieto štandardné vzorkovacie frekvencie modifikovať. To znamená, že vždy keď zavoláme funkciu, ktorá nám vráti pole prvkov, tak dostaneme pole dĺžky 48000 s jednotlivými hodnotami pre každú frekvenciu. Na získanie dát z Analyzéra[24] sme použili funkciu `getBytesFrequencyData` [25], ktorá nám vráti pole frekvencií prepočítaných na decibely. Aby sme nemuseli pracovať s takými veľkými dátami, využijeme schopnosť analyzéra, a tou je nastavenie `fftSize`[26]. Pre `fftSize` môžeme nastaviť iba hodnoty, ktoré sú mocniny dvojky. Nastavíme `fftSize` na 2048. Urobíme výpočet, kde vzorkovaciu frekvenciu podelíme veľkosťou `fftSize`. Teda $48000 / 2048 = 23,44$. V tomto momente už nebudeme mať pole dĺžky 48000, ale pole bude mať podobu [1-23.44Hz, 23.44-46.88Hz,...]. Keďže ľudský hlas sa pohybuje vo frekvenciách od 1 - 5000Hz, musíme nájsť prvky, ktoré patria pod túto hranicu. Vydelíme teda $5000 / 23,44$, čo je približne 213 prvkov. Teda k tomu, aby sme vedeli rozpoznať hlasové vlastnosti používateľa, musíme vybrať práve prvých 213 prvkov. Pri notebookoch a vzorkovacej frekvencii 44100 máme prvých 5000Hz uložených v $5000 / (44100 / 2048)$, čo je približne 232 prvkov. Rôzna vzorkovacia frekvencia by nerobila problém v prípade, že by používateľ pracoval vždy na rovnakom zariadení. Avšak ak sa zaregistruje cez notebook a bude sa pokúšať prihlásiť cez tablet, nebude to fungovať. Preto dôležité zredukovať tieto polia na spoločný počet prvkov. Tento počet nastavíme na 200. Takže, ak sa používateľ zaregistruje cez notebook, tak vieme že pole, kde sú frekvencie od 1 - 5000hz je na prvých 213 prvkoch. Toto pole zredukujeme tak, že z poľa dĺžky 213 prvkov odpočítame veľkosť poľa na ktorú chceme dané pole zredukovať, teda 200. Zistili sme, že pole je väčšie o 13 prvkov, čo znamená, že z poľa veľkosti 213 potrebujeme odstrániť 13 prvkov. Aby sme zachovali čo najpresnejšie informácie v poli, nebudeme tieto prvky odstraňovať zo začiatku alebo z konca. To zabezpečíme tak, že nájdeme ktoré pole má väčší počet prvkov. Od tohto počtu prvkov odpočítame veľkosť menšieho poľa. Takto získaný rozdiel nám udáva, koľko prvkov musíme vymazať z väčšieho poľa. Na zistenie toho, ktoré prvky máme vymazať vypočítame index týchto prvkov. Na výpočet použijeme vzorec, kde dlhšie pole vydelíme rozdielom týchto dvoch polí. Index si označíme ako i a teda z výpočtu sme zistili, že musíme odstrániť každý i -ty prvok z väčšieho poľa. Pre ukážku riešenia daného problému si môžeme pozrieť obr. Vidíme, že máme pole značené od 0 po 29, čo znamená, že veľkosť nášho poľa je 30. Určíme si, že potrebujeme nové pole dĺžky 20 tak, aby sme ho poškodili čo najmenej. Vypočítame si teda, že musíme odstrániť 10 prvkov. To znamená, že keď z 30 prvkov potrebujeme odstrániť 10 prvkov, tak musíme odstrániť každý tretí prvok. To vypočítame ako $30 / 10$. Takto sme zabezpečili čo najmenšie poškodenie poľa.



Obr. 3.2: Vzorkovacia frekvencia



Obr. 3.3: Redukcia poľa

3.3.1 Vypočítanie energie užívateľa

Aby sme si do databázy zbytočne neukladali informácie, kedy používateľ nehovorí, vypočítame si energiu používateľa v danom momente. To vypočítame tak, že si spočítame všetky decibely v poli, ktoré aktuálne dostaneme a vydělíme počtom frekvencií. Tu sme testovaním zistili, že keď používateľ začne hovoriť, tak sa táto hodnota pohybuje od 15 decibelov a vyššie. Preto sme nastavili spodnú hranicu na 15 decibelov, ktorú keď prekročí, tak sa odštartuje ukladanie do poľa, ktoré uložíme do databázy. Tu si nastavíme semafor, že používateľ začal hovoriť. Na zistenie, kedy používateľ prestal hovoriť použijeme práve tento semafor, ktorý vypneme, keď používateľovi klesne priemerná hranica frekvencie pod 15 decibelov. To sa niekedy môže stať aj v polovici slova, napríklad pri krátkom nádychu alebo znížení hlasitosti. Tento problém sme zabezpečili tak, že pri klesnutí priemernej hranice pod 15 decibelov nastavíme počítadlo, ktoré začne počítat čas. Keď táto hranica prekročí jednu sekundu, môžeme si byť istý, že používateľ prestal hovoriť. Pre poslanie informácií na server sme použili metódu fetch, ktorá na pozadí používateľa presmeruje na `/auth/voice/registration`, kde metódou POST pošleme dáta z nášho vytvoreného poľa pre frekvencie a slovo, ktoré používateľ zadal.

Takto prijaté dáta na serveri spracujeme, skontrolujeme, či sa tam toto slovo ešte pre daného užívateľa nenachádza, a následne uložíme do databázy.

3.3.2 Prihlásenie metódou hlasu

Pri prihlásení metódou hlasu je postup takmer rovnaký ako pri registrácii metódou hlasu. Tu však používateľ musí zadať svoje id. Po zadaní id sa pošle požiadavka na server, ktorú aplikácia spracuje, vyhľadá v databáze používateľa so zadaným id, a vráti odpoveď v podobe náhodne vybraného slova, ktoré si používateľ zadal pri registrácii. Taktiež si z databázy získame hlasové vlastnosti užívateľa pre dané slovo. Následne sa spustí mikrofón, hlasové informácie sa zanalyzujú a do nového poľa sa uložia dané frekvencie. Teraz máme dve polia, jedno z databázy, druhé od používateľa. Na porovnanie podobnosti týchto dvoch polí použijeme knižnicu `euclidean-distance`[27], ktorá nám vypočíta ako ďaleko sú od seba tieto polia vzdialené. Následne určíme hranicu, pri akej vzdialenosti používateľa prihlási, alebo nie.

3.3.3 Vzdialenosť polí

Na to, aby sme mohli použiť knižnicu na výpočet vzdialenosti dvoch polí musíme zabezpečiť, že počet prvkov v oboch poliach bude rovnaký. To zabezpečíme tak, že nájdeme, ktoré pole má väčší počet prvkov. Od tohto počtu prvkov odpočítame veľkosť menšieho poľa. Takto získaný rozdiel nám udáva, koľko prvkov musíme vymazať z väčšieho poľa. Na zistenie toho, ktoré prvky máme vymazať vypočítame index týchto prvkov. Na výpočet použijeme vzorec, kde dlhšie pole vydělíme rozdielom týchto dvoch polí. Index si označíme ako `i` a teda z výpočtu sme zistili, že musíme odstrániť každý `i`-ty prvok z väčšieho poľa.

3.3.4 Vykreslenie spektrogramu

V našom registračnom a prihlasovacom module sme na vykreslenie použili cyklus cez jednotlivé frekvencie. Tento cyklus si môžeme pozrieť na nasledujúcom kóde, kde podľa veľkosti hodnoty na danej frekvencii zvyšujeme teplotu farieb. Na obrázku 3.4 si môžeme pozrieť, ako vyzerá spektrogram v našej aplikácii po povedaní vety 'Ahoj, ako sa máš ?'.

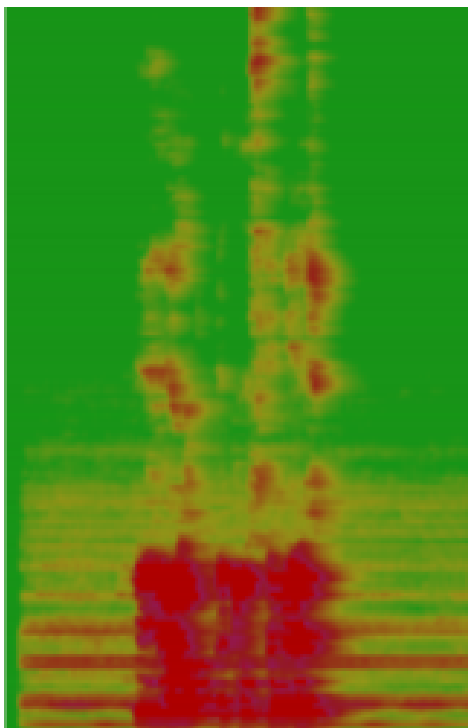
```
1 drawSpectrogram(freqs, instance, canvas){
2     const x = canvas.width - 1;
3     const h = canvas.height / freqs.length + 0.5;
4     for (let i = 0; i < freqs.length; i++) {
5         let r = freqs[i];
6         let g = "100%";
7         let b = "50%";
8         instance.ctx.beginPath();
9         instance.ctx.strokeStyle = `hsl(${r},${g}, ${b})`;
10        instance.ctx.moveTo(x, instance.canvas.height - i * h);
```

```

11         instance.ctx.lineTo(x, instance.canvas.height - (i * h + h
12             ));
13     instance.ctx.stroke();
14 }

```

Kód 3.2: Vykreslenie spektrogramu



Obr. 3.4: Spektrogram v aplikácií

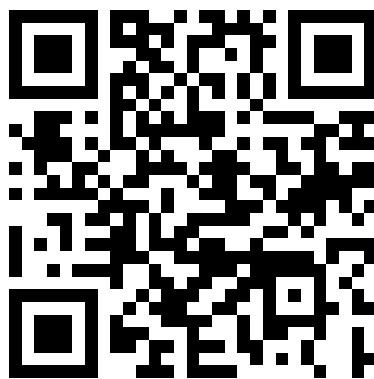
3.3.5 Testovanie registrácie a prihlásenia hlasom

Na testovanie sme využili dospelého muža, dospelú ženu a dve deti vo veku približne 10 rokov (chlapec a dievča). Testovanie prebiehalo tak, že najprv dospelý muž nahrál slovo 'Frederik'. Následne použil prihlásenie hlasom, kde prečítal uložené slovo. Po porovnaní aktuálnych údajov s údajmi z databázy sme dostali euklidovu vzdialenosť v hodnote približne 350. Nasledovala dospelá žena, ktorej bola euklidova vzdialenosť od mužského hlasu pri prečítaní toho istého slova približne 600. Dievča malo po porovnaní s mužom vzdialenosť približne 900. Vzdialenosť dospelého muža od desať ročného chlapca bola približne 750. Nakoniec sme otestovali, akú vzdialenosť získame, keď chlapec nahrá slovo do databázy a dievča sa skúsi prihlásiť prečítaním jeho slova. Táto vzdialenosť bola približne 550. Po prezretí týchto vzdialeností sme zistili, že hodnotu prihlásenia by sme mohli nastaviť približne na 350. Tu však nastal problém pri hluku spôsobenom okolím. Reprodukotor sme umiestnili 5 metrov od mikrofónu a pustili sme

hudbu pri miernej hlasitosti. Pri porovnaní vzdialeností toho istého používateľa sme získali euklidovu vzdialenosť 500. Výsledky nášho testovania nám nepriniesli očakávané presné rozdelenie používateľov. Náš modul dosahuje presnejšie výsledky iba pri rozdelení medzi mužom, ženou a dieťaťom. Pri porovnávaní viacerých detí môžeme očakávať, že vzdialenosti nebudú veľmi rozdielne. Preto pridanie tejto metódy do webovej aplikácie stojí na zvážení používateľa, avšak v momentálnom stave to neodporúčame.

3.4 Registrácia metódou QR kód

Na registráciu pomocou QR kódu sa musíme, podobne ako pri registrácii pomocou hlasu, prihlásiť a registrovať niektorou z inej metódy. Takto registrovaný a prihlásený užívateľ nám poskytne svoje id, ktoré je uložené v databáze pre našu webovú aplikáciu. Takto získané id spojíme pomocou dvojbodky s náhodne vygenerovaným textovým reťazcom. Tento náhodný textový reťazec ukladáme do databázy ako heslo pre príslušné ID. Na vykreslenie textového reťazca do QR kódu používame knižnicu `qrcode`[28] na serverovej strane. Na webový prehliadač pošleme už takto vygenerovaný obrázok. Tento vygenerovaný obrázok si môže používateľ stiahnuť a vytlačiť, alebo odfotiť mobilným telefónom. Používateľovi, ktorý si QR kód vytlačí odporúčame nastaviť veľkosť obrázku na minimálne 1,5 centimetra na výšku a 1,5 centimetra na šírku. Na ukážku sa používateľovi s id 93 vygeneruje náhodný reťazec v podobe c983e721d4feb, teda celý náš kód bude preložený v textovom formáte vyzeráť nasledovne 93:c983e721d4feb. QR kód vygenerovaný z tohoto reťazca je vyobrazený na obrázku 3.5. Pri strate alebo úniku údajov z QR kódu je potrebné zabezpečiť aby bolo možné zmeniť QR kód.



Obr. 3.5: QR kód

3.4.1 Implementácia QR kódu

Pre autentifikáciu pomocou QR kódu sme sa pokúsili docieľiť to, aby mal QR kód čo najmenej riadkov a stĺpcov. Hlavný dôvod je ten, že čím je menej riadkov a stĺpcov, tým môžeme QR kód vytlačiť v menších rozmeroch. Aby bol QR kód bezpečný, tak musíme zvoliť vhodnú dĺžku náhodného reťazca, ktorý bude použitý ako heslo. Tento náhodný reťazec je zložený z malých písmen anglickej abecedy a z čísel od nula až po deväť. Pri nastavení veľkosti hesla na 13 sa spolu s dvojbodkou a ID používateľa vygeneruje QR kód verzie 2, čo predstavuje kód zložený z 25 riadkov a 25 stĺpcov. Nastavením dĺžky hesla na 13 docielime, že možných kombinácií všetkých hesiel je 36^{13} , čo po prepočte vyjde na 170 581 728 179 578 208 256 všetkých možností. Keďže sa predpokladá, že niektorí používatelia si QR kód iba vytlačia a nezabezpečia proti poškodeniu, tak sme sa rozhodli QR kódu nastaviť najvyššiu úroveň opravy chýb, čo predstavuje písmeno H. Pri tomto nastavení sa veľkosť QR kódu zvýši na verziu 3, čo predstavuje 29 riadkov * 29 stĺpcov. Táto veľkosť za zvýši kvôli tomu, že QR kód musí uchovať viac dát ako zálohu ako je spomínané v sekcii Východiská. Postupným poškodzovaním QR kódu sme sa dostali k tomu, že naša webová aplikácia prečíta a rozanalyzuje aj takýto kód na obr. 3.6.



Obr. 3.6: Poškodený QR kód

3.4.2 Prihlásenie metódou QR kódu

Pre prihlásenie pomocou QR kódu znova použijeme rozhranie `getUserMedia()[22]`, kde tentokrát nastavíme, že požadujeme iba zapnutie web kamery. Rovnako ako pri práci s mikrofónom, aj tu sa sa prvej návšteve webovej aplikácie bude požadovať potvrdenie na prístup k web kamere. Takto spustené video vložíme do canvas elementu, ktorý slúži na vykreslenie. Týmto zabezpečíme, že používateľ vidí spätnú väzbu v reálnom čase a tým docielime to, že uvidí kde presne sa nachádza QR kód a prípadne ak

časť kódu je mimo záberu kamery tak vie, že ho musí presunúť. Na analýzu QR kódu použijeme knižnicu JSQR.js[29] na strane webového prehliadača. Po nastavení QR kódu pred kameru naša webová aplikácia analyzuje QR kód, po zanalyzovaní QR kódu sa video zatvorí a webová aplikácia kód preloží na reťazec a tento reťazec rozdelíme podľa dvojbodky na id a heslo. Toto id a heslo pošleme na server cez `"/auth/qrcode"`, kde server skontroluje, či sa toto id nachádza v databáze a či má k nemu správne heslo. Ak áno, používateľ môže pokračovať v práci ako prihlásený, ak nie, vypíše sa chybová hláška. Aby sme takýto QR kód dokázali prečítať a zanalyzovať, je potrebné aby kamera vedela zachytiť všetky potrebné detaily. Pri vytlačení QR kódu vo veľkosti 1,5 centimetra dĺžka 1,5 centimetra výšky je potrebné tento QR kód umiestniť pred kameru približne vo vzdialenosti 5 centimetrov. Čím bude náš vytlačený QR kód vytlačený vo väčších rozmeroch, tým ho môžeme vzdialiť od kamery ďalej. Pri rozmeroch QR kódu 5 centimetrov do dĺžky a 5 centimetrov do výšky našej aplikácií postačí, aby používateľ nastavil kód približne 50 centimetrov od obrazovky. Ak sa používateľ rozhodne používať mobilný telefón namiesto vytlačenej formy, snímač QR kódu nemusí vždy správne fungovať pretože pri nastavení slabého jasu na mobilnom telefóne je takmer nemožné rozdeliť QR kód na farby čiernu a bielu, čo znamená že webová aplikácia nevie určiť dáta z QR kódu. Pri nastavení príliš vysokého jasu na mobilnom telefóne nastáva problém, že jas je niekoľkonásobne vyšší ako okolité svetlo, čo znamená, že display mobilného telefónu vyžaruje príliš veľa svetla a my môžeme vidieť, že naša kamera sníma tento QR kód veľmi rozmazane. Aj pre tento problém bolo vhodné použiť vykreslenie výstupu z kamery do obrazovky, pretože používateľ vidí, ako reaguje kamera na jeho QR kód a podľa potreby si môže zvýšiť alebo znížiť jas.

3.5 Registrácia verzus pridávanie metódy

V ďalších bodoch sa budeme venovať ostatným metódam, ktoré sú dostupné v našej webovej službe. Kým pre hlas a QR kód platilo, že si ich mohol pridať iba prihlásený užívateľ, pre ostatné metódy platí, že sa môže registrovať buď nový používateľ, alebo si ich chce pridať už existujúci používateľ. Na to, aby si mohol pridať tieto metódy musí byť prihlásený. To, či je momentálne prihlásený, zistíme, ak `req.session.passport` alebo `req.session.passport.user` nie je `undefined`.

3.6 Registrácia meno, heslo a email

Pri registrácií metódou menom, heslom a emailom využijeme podobné vlastnosti ako má väčšina terajších aplikácií. Týmito vlastnosťami sú, že používateľ musí zadať užívateľské meno, email, heslo a potvrdenie hesla. Na to, aby užívateľ mohol vklá-

dať tieto údaje sme vytvorili formulár v HTML. Jednotlivé kolónky sú `<label>` elementy. Týmto elementom `<label>` vieme nastaviť, aký formát vstupu od používateľa do nich očakávame. Pre meno použijeme klasický formát `text`, pre email nastavíme formát na `email`, čo pomáha s kontrolou emailu. Ak teraz používateľ nezadá zavináč a odošle formulár, webový prehliadač mu automaticky vypíše, aký formát emailu očakáva. Pre heslo sme zvolili formát `"password"`, čo je potrebné kvôli bezpečnosti. Takto zvolený formát zabezpečí, že sa do kolónky nevypíše napísaný znak, ale je na miesto neho napísaná hviezdička. Takto nikto neuvidí, aké heslo je napísané a neuvidí to ani sám používateľ. Preto sa môže ľahko stať, že urobí preklep a následne sa nebude vedieť prihlásiť, preto sme pridali ďalšiu kolónku na potvrdenie hesla, kde ak sa heslá nerovnajú, vypíše sa chybová hláška. Minimálnu dĺžku mena a hesla sme nastavili na 6 znakov. V prípade, že používateľ vyplnil všetko správne sa na serveri toto heslo zašifruje pomocou knižnice `bcrypt`. Po zašifrovaní hesla musíme zistiť, či je používateľ prihlásený.

Ak áno, z momentálneho prihlásenia si vypýtame jeho hlavné id, ktoré používa v aplikácií. Následne uložíme do tabuľky `LocalAccounts` jeho meno, zašifrované heslo, získané hlavného id pre aplikáciu a hodnotu potvrdený email nastavíme na `false`. Po uložení do databázy sa užívateľovi pošle verifikačný link na potvrdenie, že sa chce registrovať. Po kliknutí na link sa užívateľovi hodnota potvrdeného emailu nastaví na `true`.

Ak používateľ nie je momentálne prihlásený, preto predpokladáme, že si vytvára nový účet. Postup je rovnaký ako v predchádzajúcej časti, avšak tu najskôr vytvoríme nový účet pre našu aplikáciu v tabuľke `Accounts`, a následne sa id z tejto tabuľky nastaví ako `account_id` v tabuľke `LocalAccounts`.

3.7 Facebook a Google

Keďže registrácia a prihlasovanie pomocou Facebooku a Google funguje na rovnakom princípe, zhrnieme to do tejto sekcie. Predpokladáme, že vývojár už má zaregistrovanú svoju aplikáciu vo Facebooku aj v Googli. Keďže webová aplikácia využíva knižnicu `socket.io`, potrebujeme zabezpečiť, aby sa webová stránka, kde má užívateľ otvorenú našu webovú aplikáciu nikdy neobnovila. Avšak pri pridávaní týchto metód nastal problém, že keď používateľ klikne na prihlásenie pomocou Facebooku alebo Google, aplikácia otvorí novú záložku, kde sa má používateľ prihlásiť. Preto sme využili schopnosť `socketu` a tou je pridávanie záložiek do miestností.

```
1 ioServer.on("connection", (socket) => {
2   for (let i = 0; i < sessions.length; i++) {
3     if (sessions[i].sessionId === socket.request.session.id) {
4       sessions[i].sockets.push(socket.id);
5     } else if (i === sessions.length) {
```

```

6     sessions.push(makeNewObj(socket));
7   }
8 }
9 if (sessions.length === 0) sessions.push(makeNewObj(socket));
10 socket.join(socket.request.session.id);
11 socket.on("disconnect", () => {
12   for (let i = 0; i < sessions.length; i++) {
13     if (sessions[i].sockets.includes(socket.id)) {
14       delete sessions[i].sockets[sessions[i].sockets.indexOf(socket.
15         id)];
16     }
17   }
18 });

```

Kód 3.3: Funkcia na vytvorenie socketu

Takto sme si zabezpečili, že každá nová záložka je v miestnosti spolu s našou webovou aplikáciou, takže teraz tieto záložky dokážu komunikovať o tom, či sa prihlásenie na Facebooku alebo Googli podarilo.

Aby sme mohli použiť knižnicu Passport.js, potrebujeme mu nainicializovať potrebné údaje.

```

1 this.passport.use("facebook" + (this.isRegistration ? "-reg" : ""),
2   new strategy({
3     clientID: "15614646465465165165165165",
4     clientSecret: "abc1651651651ca165165c1a1651ca16516ca16516",
5     callbackURL: "moja-stranka.sk/auth/facebook/callback",
6     passReqToCallback: true, })

```

Kód 3.4: Inicializácia metódy Facebooku

ClientID a clientSecret sú údaje, ktoré sme získali zaregistrovaním svojej aplikácie v službe. CallbackUrl je url link, na ktorý nás Facebook presmeruje po úspešnom alebo neúspešnom prihlásení. Na to, aby sme mohli používať údaje z Facebooku v do funkcie, v ktorej riešime prihlásenie a registráciu, potrebujeme passReqToCallback nastaviť na true. Do tejto funkcie posielame callbackovú funkciu cb, cez ktorú sa nastavením na cb(null, false); neprihlásime. Naopak, ak používateľa nájdeme v databáze a všetko prebehne v poriadku, používateľa prihlásime príkazom cb(null, res1);

V nasledujúcom kroku musíme nastaviť, čo sa stane, keď aplikácia dostane get požiadavku na /auth/facebook/callback. V prípade, že používateľ sa vo Facebooku úspešne prihlásil, presmeruje ho na /auth/success. Ak nie, presmerujeme ho na /auth/fail.

```

1 this.express.get("auth/facebook/callback",
2   this.passport.authenticate("facebook"),
3   { failureRedirect: "/auth/fail",
4     successRedirect: "/auth/success" } ) );

```

Kód 3.5: Presmerovanie po prihlásení na Facebooku

`/auth/success` a `/auth/fail` sú jednoduché stránky, ktoré slúžia práve pre spomínaný problém s presmerovaním stránky. Po načítaní stránky `/auth/fail` sa toto okno iba zatvorí.

```
1 window.addEventListener('load', (e) => {
2     window.close();
3 })
```

Kód 3.6: `/auth/fail`

Ak používateľa presmeruje na stránku `/auth/success`, znamená to, že sa mu podarilo prihlásiť vo Facebooku. Po tom, ako sa dostane na `/auth/success`, odošle sa cez socket správa na našu webovú aplikáciu príkazom `socket.emit('loggedIn')`; a toto okno sa automaticky zatvorí.

```
1 window.addEventListener('load', (e) => {
2     var socket = io();
3     socket.emit('loggedIn');
4     socket.on('closeWindowConfirm', () => {
5         window.close();
6     });
7 })
```

Kód 3.7: `/auth/success`

3.8 Implementácia Univerzitného prihlasovania FMFI

Rovnako ako pri implementácii Facebook a Google služieb do našej aplikácie, musíme aj teraz knižnici Passport.js nainicializovať potrebné údaje. Tieto budú odlišné ako pre Facebook a Google, pretože technológia SAML 2.0 je odlišná od technológie OAuth 2.0, ktoré používajú spomínané služby. Pri technológii SAML 2.0 musíme Passport.js poskytnúť údaje zobrazené na kóde nižšie.

```
1 this.passport.use("uniba", new strategy({
2     path: moja-stranka.sk/auth/uniba/callback,
3     protocol : "https://",
4     host: "moja-stranka.sk",
5     issuer: "passport-saml",
6     entryPoint:
7     "https://idp.uniba.sk/idp/profile/SAML2/POST/SSO",
8     cert: "",
9     privateKey: "",
10    passReqToCallback: true,
11    },
```

Kód 3.8: Funkcia na vytvorenie socketu

Na to, aby sme mohli použiť službu prihlasovania cez FMFI musíme vytvoriť certifikát, ktorý uložíme do cert spomenutom v kóde vyššie. Aby bol certifikát správny je potrebné, aby bol podpísaný nejakou autoritou, pretože samopodpísaný certifikát nie je vhodný. Na skúšobnom serveri sa nám však nepodarilo vytvoriť takýto certifikát podpísaný autoritou, preto nebolo možné túto metódu prihlásenia otestovať.

3.9 Používateľská príručka

Keďže jedna z hlavných požiadaviek od zadávateľa bola, aby nasadenie na server bolo čo najjednoduchšie. Preto sme sa snažili o čo najväčší komfort pre zadávateľa a vytvorili sme balíček, kde na spustenie treba urobiť nasledovné. V hlavnom adresári je vytvorený konfiguračný súbor s názvom .env, v ktorom je potrebné nastaviť nasledovné údaje.

```
1 DB_USER = root // Prihlasovacie meno do databazy
2 DB_HOST = localhost // host databazy
3 DB_DATABASE = bakalarka // nazov databazy
4 DB_PASSWORD = '' // heslo databazy
5 DB_PORT = 3306, // port databazy
6 DB_DIALECT = 'mysql' // Preferovana databaza, na vyber mysql, mariadb,
    sqlite, postgres, mssql
7
8 RURL = http://localhost:8080 // RootUrl - domena webovej aplikacie
9
10 FB_CLIENT_ID = 415277056169601 // Client_id po zaregistrovani
    aplikacie vo Facebooku
11 FB_CLIENT_SECRET = 405a739d32e1b805343869ed1240c522 // Client_secret
    po zaregistrovani aplikacie vo Facebooku
12 FB_CALLBACKURL = /auth/facebook/ //netreba menit
13 FB_CALLBACKURL_REGISTRATION = /auth/facebook/registration/ //netreba
    menit
14
15 UNIBA_CALLBACKURL = /auth/uniba/ //netreba menit
16 UNIBA_REGISTRATION = /auth/uniba/registration/ //netreba menit
17
18 LOCAL_CALLBACKURL = /auth/local/ //netreba menit
19 LOCAL_CALLBACKURL_REGISTRATION = /auth/local/registration/ //netreba
    menit
20
21 QR_CODE_CALLBACKURL = /auth/qrcode/ //netreba menit
22 QR_CODE_CALLBACKURL_REGISTRATION = /auth/qrcode/registration/ //netreba
    menit
23
24 VOICE_CALLBACKURL = /auth/voice/ //netreba menit
```

```

25 VOICE_CALLBACKURL_REGISTRATION = /auth/voice/registration/ //netreba
    menit
26
27 GOOGLE_CLIENT_ID = 591154487563-amrn7fr4t6scpv1q4tdigh95dah89u3p.apps.
    googleusercontent.com //Client_id po zaregistrovani aplikacie v
    Google.
28 GOOGLE_CLIENT_SECRET = Co7Ma019hYkbf8mYY0diXraM // Client_secret po
    zaregistrovani aplikacie v Google.
29 GOOGLE_CALLBACKURL = /auth/google/ //netreba menit
30 GOOGLE_CALLBACKURL_REGISTRATION = /auth/google/registration/ //netreba
    menit
31
32 COOKIE_SECRET = "" //zvolte lubovolny retazec
33 COOKIE_NAME = usr_cookie //zvolte lubovolny retazec
34
35 UNIBA_CERT = '' // Certifikat podpisyany autoritou.

```

Kód 3.9: .env

Po nastavení správnych údajov je potrebné v termináli vjsť do domovského adresáru aplikácie a napísať príkaz `npm install`. Po spustení príkazu sa začnú inštalovať všetky potrebné npm balíčky, ktorých názvy sú uložené v konfiguračnom súbore `package.json`. Rovnako je potrebné nastaviť údaje o databáze v databázovom priečinku v `/database/config/config.json`. Po inštalácii balíčkov bude potrebné vytvoriť tabuľky do databázy. Pomocou `Sequelize.js` sme vytvorili súbory, ktoré slúžia na migráciu. Spustením `npx sequelize db:migrate` sa vytvoria tabuľky v nami zvolenej databáze.

Po úspešnom nainštalovaní balíčkov a vytvorení tabuliek spustíme príkaz `node index.js start` a náš prihlasovací modul sa zobrazí na našej webovej stránke.

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť rozšírený prihlasovací a registračný modul webovej aplikácie. Bolo potrebné nájsť aplikácie tretích strán, ktorým dôverujeme. Taktiež sme museli nájsť vhodný obrázkový spôsob na pomoc pri prihlasovaní pre deti, ktoré ešte nevedia písať a čítať. Registrovaných používateľov bolo potrebné uložiť do lokálnej databázy. Pre používateľov, ktorí chcú náš prihlasovací modul vo svojej webovej aplikácii bolo potrebné, aby nasadenie na server netrvalo dlho a aby nespôsobilo veľa práce. Taktiež bolo potrebné pridať modul, kde si používateľ bude môcť pridať aj iné metódy prihlasovania ako tie, ktorými sa registroval. Vyskúšali sme si aj experimentovanie s ľudským hlasom, ktorý sme sa snažili rozpoznať a následne určiť, ktorému používateľovi tento hlas patrí.

Všetky požiadavky boli úspešne implementované, ale nepodarilo sa nám otestovať prihlasovanie cez Univerzitné konto, pretože sa nám nepodarilo získať skúšobný prístup pre náš lokálny server. Úspešnou implementáciou nášho prihlasovacieho modulu sme dosiahli celkový prínos do problémov autentifikácie používateľa. Používatelia si už nebudú musieť pamätať ďalšie nové prihlasovacie meno alebo heslo, budú schopní sa registrovať a prihlásiť jednoducho pomocou Facebooku alebo Google. Taktiež budú študenti schopní použiť svoje Univerzitné konto. Pre deti, ktoré majú problém s písaním a čítaním sme vymysleli spôsob prihlasovania cez QR kód, ktorý stačí ukázať na kameru a aplikácia ich po rozanalyzovaní QR kódu prihlási. Experimentálna časť prihlasovania a registrácie pomocou ľudského hlasu má určité medzery a preto stojí za zváženie, či ju používateľ nášho prihlasovacieho modulu bude využívať vo svojej webovej aplikácii.

Nakoľko vytvorenie takéhoto všestranného prihlasovacieho modulu bolo časovo náročné a nepodarilo sa nám dostať rozpoznávanie hlasu na perfektnú úroveň, chcel by som v budúcom vývoji aplikácie pokračovať práve v tejto oblasti. Tiež stojí za zváženie pridanie metódy prihlasovania cez Github alebo cez Apple.

Zdroje a použitá literatúra

- [1] Facebook, Citované dňa 29.máj 2021, Dostupné na: <https://www.facebook.com/>
- [2] Google, Citované dňa 29.máj 2021, Dostupné na: <https://www.google.com/>
- [3] Twitter, Citované dňa 29.máj 2021, Dostupné na: <https://www.twitted.com/>
- [4] SAML2 SSO Integration, Citované dňa: 14. máj 2021. Dostupné na: <https://www.ibm.com/docs/en/essm/10.1.3?topic=configuration-saml2-sso-integration>
- [5] D Hardt, Ed. The Oauth 2.0 Authorization Framework. Microsoft 2012. ISSN:2070-1721
- [6] Joseph, Kulandai, Java Facebook Login with OAuth Authentication, Javapapers, 16. október 2014. Citované dňa: 7.február 2021. Dostupné na: <https://javapapers.com/java/java-facebook-login-with-oauth-authentication/>
- [7] Using OAuth 2.0 to Access Google APIs Citované dňa: 18. máj 2021. Dostupné na: <https://developers.google.com/identity/protocols/oauth2>
- [8] How to Make a QR Code | Creating QR Codes, Scott, SproutQR, 3. september 2020: Citované dňa: 21.január 2021. Dostupné na: <https://www.sproutqr.com/blog/how-to-make-a-qr-code>
- [9] Biometric Authentication Overview, Advantages & Disadvantages, ANA DAS-CALESCU, Heimdal, 1. máj 2019: Citované dňa: 21.január 2021. Dostupné na: <https://www.sproutqr.com/blog/how-to-make-a-qr-code>
- [10] <https://www.idmerit.com/blog/fraud-prevention-solutions-iris-scanning-vs-retina-scanning/>
- [11] Voice biometrics: The voice print will become online banking's greatest ally, Bank Services, Banco Bilbao Vizcaya Argentaria, 21 august 2020. Citované dňa: 7. február 2021. Dostupné na: <https://www.bbva.com/en/voice-biometrics-the-voice-print-will-become-online-bankings-greatest-ally/>

- [12] nodemailer, Citované dňa: 28.máj 2021, Dostupné na: <https://nodemailer.com/about/>
- [13] stackoverflow, Citované dňa: 17.máj 2021, Dostupné na: <https://stackoverflow.com/>
- [14] CROCKFORD, Douglas. Javascript:The Good Parts. YAHOO! PRESS, 2008. ISBN: 978-0-596-51774-8.
- [15] About Node.js, Citované dňa: 17.máj 2021, Dostupné na: <https://nodejs.org/en/about/>
- [16] What Socket.IO is, Citované dňa: 17.máj 2021, Dostupné na: <https://socket.io/docs/v4>
- [17] bcrypt, Citované dňa: 17.máj 2021, Dostupné na: <https://www.npmjs.com/package/bcrypt>
- [18] Sequelize, Citované dňa: 17.máj 2021, Dostupné na: <https://sequelize.org/master/>
- [19] Passport, Citované dňa: 17.máj 2021, Dostupné na: <https://www.passportjs.org/>
- [20] Configure Passport, Citované dňa: 17.máj 2021, Dostupné na: <https://www.passportjs.org/docs/configure/>
- [21] Dostupné na: https://www.researchgate.net/figure/Spectrogram-of-a-speech-signal-with-breath-sound-marked-as-Breath-whose-bounds-are_fig1_319081627
- [22] MediaDevices.getUserMedia(), Citované dňa: 14.máj 2021. Dostupné na: <https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia>
- [23] Web Audio API ,14.máj 2021. Dostupné na: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
- [24] AnalyserNode, Citované dňa 22.máj 2021, Dostupné na: <https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode>
- [25] AnalyserNode.getByteFrequencyData(), Citované dňa 22.máj 2021, Dostupné na: <https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode/>
- [26] AnalyserNode.fftSize, Citované dňa 22.máj 2021, Dostupné na: <https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode/fftSize>
- [27] euclidean-distance, Citované dňa 22.máj 2021, Dostupné na: <https://www.npmjs.com/package/euclidean-distance>

- [28] qrcode, Citované dňa 26.máj 2021, Dostupné na:
<https://www.npmjs.com/package/qrcode>
- [29] jsQR, Citované dňa 26.máj 2021, Dostupné na:
<https://www.npmjs.com/package/jsqr>

Príloha A: obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu a súbory s výsledkami experimentov. Zdrojový kód je zverejnený aj na stránke <https://www.st.fmph.uniba.sk/~kohar2/bc/>.

V hlavnom priečinku bakalárskej práce sa nachádza env súbor `.env`. V tomto súbore si používateľ nastaví dôležité konfiguračné údaje. Podpriečink database je určený pre knižnicu Sequelize.js. Tu treba doplniť údaje v podpriečinku `config` a subore `config.json`. Ďalej sa tu nachádza súbor `index.js`, kde konfigurujeme a spúšťame server. V `module.js` inicializujeme potrebné metódy. V `package.json` sú uložené názvy balíčkov, ktoré sa nainštalujú pomocou `npm install`. V podpriečinku `oauth` sú všetky metódy registrácie a prihlasovania v osobitnom súbore. Tento podpriečink patrí serverovskej strane a nastavujeme tu, kedy používateľa vložíme do databázy, vymažeme ho z databázy alebo prihlásime ho. Podpriečink `www` je určený pre klienta. Sú v ňom uložené všetky potrebné obrázky. Sú tam `html` súbory a to `index.html`, `fail.html`, `success.html`. Taktiež tu nájdeme `base.css`, ktoré si používateľ môže upraviť podľa vlastných preferencií na vzhľad stránky. V podpriečinku `classes` sú registrácia, prihlásenie a pridávanie metódy rozdelené do samostatných súborov.