Stredná priemyselná škola  
Slovenských partizánov 1132/52, 017 01 Považská Bystrica

# Školský informačný systém

Stredoškolská odborná činnosť

č. odboru: 11 – Informatika

Riešitelia

Považská Bystrica Tomáš Tomovčík

2019

Ročník štúdia: **štvrtý**

# ABSTRAKT

TOMOVČÍK, Tomáš: Školský informačný systém. Stredná priemyselná škola Považská Bystrica, Slovenských partizánov, 1132/52, 017 01 Považská Bystrica. Vedúci práce: Ing. Ľubica Kováčová. Mesto obhajoby: SPŠ Považská Bystrica, 2019. 24 strán.

Práca je rozdelená do troch častí. Prvá časť obsahuje dva návrhy dizajnu rozhrania. Druhá časť práce je zameraná na programovanie produkčnej verzie. Obsahuje popis zvoleného frameworku, knižníc a funkcií. Tretia časť je zameraná na testovanie dvoma spôsobmi – manuálne a pomocou testovacej knižnice.

**Kľúčové slová:**

škola, informačný systém, PWA, internet

# Čestné vyhlásenie

„Vyhlasujem, že prácu stredoškolskej odbornej činnosti na tému Školský informačný systém, som vypracoval samostatne, s použitím uvedených literárnych zdrojov. Prácu som neprihlásil a ani neprezentoval v žiadnej inej súťaži, ktorá je pod gestorstvom MŠMVVaŠ SR.“

V Považskej Bystrici ..............................................

.............................................

# Poďakovanie

Ďakujem môjmu konzultantovi Ing. Ľubici Kováčovej za rady a pripomienky pri tvorbe tejto práce. Taktiež by som chcel poďakovať všetkým ľuďom zo skupiny Vývojári a Andremu Kotthausovi za pomoc pri implementácií nových funkcií.

# OBSAH

0 [ÚVOD](#_0_ÚVOD) 7

1 [PROBLEMATIKA A PREHĽAD LITERATÚRY](#_1_PROBLEMATIKA_A) 8

1.1 Programovací jazyk Javascript 8

1.2 VCS 9

2 [CIELE PRÁCE](#_2_CIELE_PRÁCE) 10

2.1 Jednoduchosť inštalácie a používania 10

2.2 Modulárne rozhranie 10

2.3 Samostatný chod aplikácie 11

2.4 Podpora starších zariadení a operačných systémov 12

3 NÁVRH APLIKÁCIE 13

3.1 Prototyp 13

3.2 Prototyp: Dizajn aplikácie 14

3.3 Prototyp: Programovanie vo frameworku Electron 15

3.4 Prototyp: Testovanie 18

4 PROGRAMOVANIE PRODUKČNEJ VERZIE 19

4.1 Výber a testovanie Javascript frameworkov 19

4.2 Dizajn aplikácie 20

4.3 Implementácia funkcií 21

4.4 Testovanie 22

5 ZÁVERY PRÁCE 23

6 ZHRNUTIE 23

7 ZOZNAM POUŽITEJ LITERATÚRY 24

PRÍLOHY

# Zoznam ilustrácií

**Číslo obrázka Názov obrázka Strana**

Obr. 1 Javascript logo 8

Obr. 2 Nodejs logo 8

Obr. 3 Hodnotenie z Google Lighthouse 10

Obr. 4 Štruktúra aplikácie 11

Obr. 5 Podiel verzií Windows-u na 12

Obr. 6 Prvé UI 14

Obr. 7 Hlavná obrazovka nového UI 15

Obr. 8 Štruktúra Electron aplikácie 15

Obr. 9 Funkcie aktuálneho času a dátumu 16

Obr. 10 Funkcia na zobrazovanie rozvrhu hodín 17

Obr. 11 Nastavenia aplikácie 18

Obr. 12 Syntax Cypress-u 19

Obr. 13 Názov a logo aplikácie 20

# 0 ÚVOD

V tejto práci sme sa rozhodli riešiť problematiku zobrazovania školských informácií na nástenkách v školách. Konkrétne sme sa zamerali na zobrazovanie informácií zo systému EduPage a dodatočných informácií, ako napr. aktuálna vyučovacia hodina a pod. Hlavným cieľom bolo zjednodušenie informovania žiakov o aktuálnych udalostiach týkajúcich sa školy a školských akcií. Cieľom bolo vytvoriť modulárny systém, ktorý bude využiteľný na rôznych typoch zariadení (PC, mobily, IoT), bude sa dať spravovať cez webovú stránku so zabezpečeným prístupom a možnosťou synchronizácie nastavení na všetkých pripojených zariadeniach.

# 1 PROBLEMATIKA A PREHĽAD LITERATÚRY

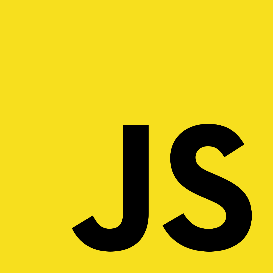
# 1.1 Programovací jazyk Javascript

Javascript (ďalej len JS) (Obr. 1) je objektovo orientovaný interpretovaný skriptovací jazyk, ktorý nie je potrebné kompilovať a využíva zabudované komponenty a objekty prehliadača.

JS je klientsky skript. To znamená že server odošle požadované súbory spolu s JS skriptami do prehliadača a tam sú následne spustené. Javascript napriek podobnému názvu nie je Java. Java je samostatný programovací jazyk, ktorý sa podobá JS iba jeho syntaxou. Obmedzeniami Javascriptu sú, že koncový používateľ môže JS zakázať v prehliadači, čím môže (a s veľkou pravdepodobnosťou aj ovplyvní) fungovanie aplikácií a stránok.

JS sa pôvodne používal iba na webe, kde dopĺňal funkcionalitu stránky. Ide napr. o ukladanie dát pomocou cookies (alebo v prípade HTML5 do localStorage). Neskôr sa začal používať server-side pomocou frameworku ako napr. NodeJS (Obr. 2).

ECMAScript (ďalej len ES) je štandard, ktorý určuje podmienky pre vývoj nových skriptovacích jazykov. Ide o štandard ECMA-262, ktorý reprezentuje tieto špecifikácie. ES poskytuje základné pravidlá, detaily a požiadavky, ktoré by mal skriptovací jazyk používať.

Obr. 1 – Javascript logo Obr. 2 – nodejs logo

# 1.2 VCS

Version Control System (VCS) je systém, ktorý umožňuje udržiavať pokope rôzne verzie súborov (najčastejšie zdrojových súborov programov), zisťovať rozdiely medzi nimi, zlučovať zmeny vykonané v rovnakom súbore, atď. VCS je veľmi dôležitý v každom len trochu väčšom projekte, najmä softwarovom. Jednotliví členovia tímu ukladajú svoju prácu na jedno miesto. Môžu odtiaľ získavať rôzne verzie súborov, vrátane tzv. Commitov (rezov), t. j. verzie súborov k určitému časovému okamihu alebo logickému bodu.

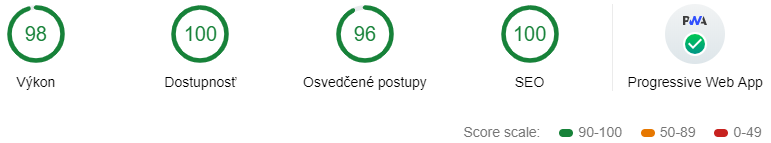
# 2 CIELE PRÁCE

Pri tvorbe tohto projektu som si stanovil niekoľko cieľov

1. Jednoduchosť inštalácie a používania
2. Modulárne rozhranie
3. Samostatný chod aplikácie
4. Zachovaná podpora starších operačných systémov a zariadení

# 2.1 Jednoduchosť inštalácie a používania

Jedným z primárnych zameraní pri vývoji bola jednoduchosť nielen inštalácie ale aj samotného používania. Kládol som dôraz aj na optimalizáciu aplikácie za pomoci Google Lighthouse (Obr. 3), kde sa mi poradilo dodržať všetky podmienky pre zjednodušenie používania aj ľuďom so zrakovým zdravotným znevýhodnením.



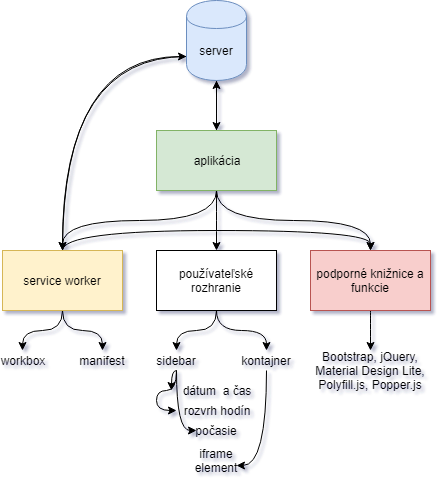
Obr. 3 - Hodnotenie aplikácie z Google Lighthouse

# 2.2 Modulárne rozhranie

Aj keď hlavným cieľom bolo vytvoriť aplikáciu využiteľnú v školskom prostredí (hlavne stredné školy), snažil som sa ju naprogramovať tak, aby bola zachovaná modularita. Týmto sa zabezpečí možnosť prechodu z jedného prostredia do iného v pomerne krátkom čase. Samotná aplikácia bola už pri návrhu rozdelená do niekoľkých častí – a to hlavného kontajnera a bočného panela.

# 2.3 Samostatný chod aplikácie

Ďalším z cieľov bola automatizácia a zníženie nárokov na údržbu. Tento cieľ sa mi podarilo dosiahnuť s pomocou Google Firebase, kde som využil možnosti hosťovania a aktualizácie aplikácie. Do Progressive Web App (PWA) bol pridaný service worker Workbox, ktorý si kontroluje aktualizácie. V prípade, že nejaká aktualizácia bude k dispozícii, automaticky ju stiahne, nové súbory aktualizuje a pridá do vyrovnávacej pamäte. Aktualizácia sa aplikuje až po obnovení aplikácie, t. j. opätovnom načítaní alebo spustení. Na obrázku (Obr. 4) je zobrazený systém fungovania aplikácie.

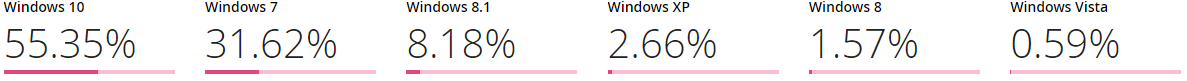


Obr. 4 - Štruktúra aplikácie

V prípade, že sa vyskytnú nejaké chyby, aplikácia sa prepne do offline režimu a bude pokračovať na archivovaných súboroch až do okamihu, pokiaľ sa problém nevyrieši. Okrem Firebase je v aplikácií aj Google Analytics, kde je k dispozícií prehľad stavu a používania. Na základe týchto informácií máme možnosť optimalizovať tie časti aplikácie, ktoré sú najčastejšie používané. V budúcnosti plánujem pridať AI, ktoré si bude sledovať prevádzku a časy používania a aplikácia si bude kompletne sama riadiť chod a správu.

# 2.4 Podpora starších operačných systémov a zariadení

Aplikácia je navrhnutá tak, aby bola spustiteľná aj na starších zariadeniach a operačných systémoch.



Obr. 5 - Podiel verzií Windows-u na trhu na Slovensku, Q1 2019

Z údajov uvedených v tabuľke (Obr. 5) je vidieť, že Windows XP a 7 sa stále pozívajú, napriek tomu že podpora pre Windows XP skončila v apríli 2014 a pre Windows 7 skončí v roku 2020. Vo finálnej verzií sa zachovala plná podpora pre operačné systémy GNU+Linux (Jadro >=2.x), macOS 10.6.x, Windows >=5.1 (XP) a prehliadače Mozilla Firefox 58, Google Chrome 67, Samsung Internet (N/A) a Opera >=20.x. Microsoft Edge a Safari nebolo testované, nakoľko ho takmer nikto nepoužíva a má nedostatočnú podporu.

# 3 NÁVRH APLIKÁCIE

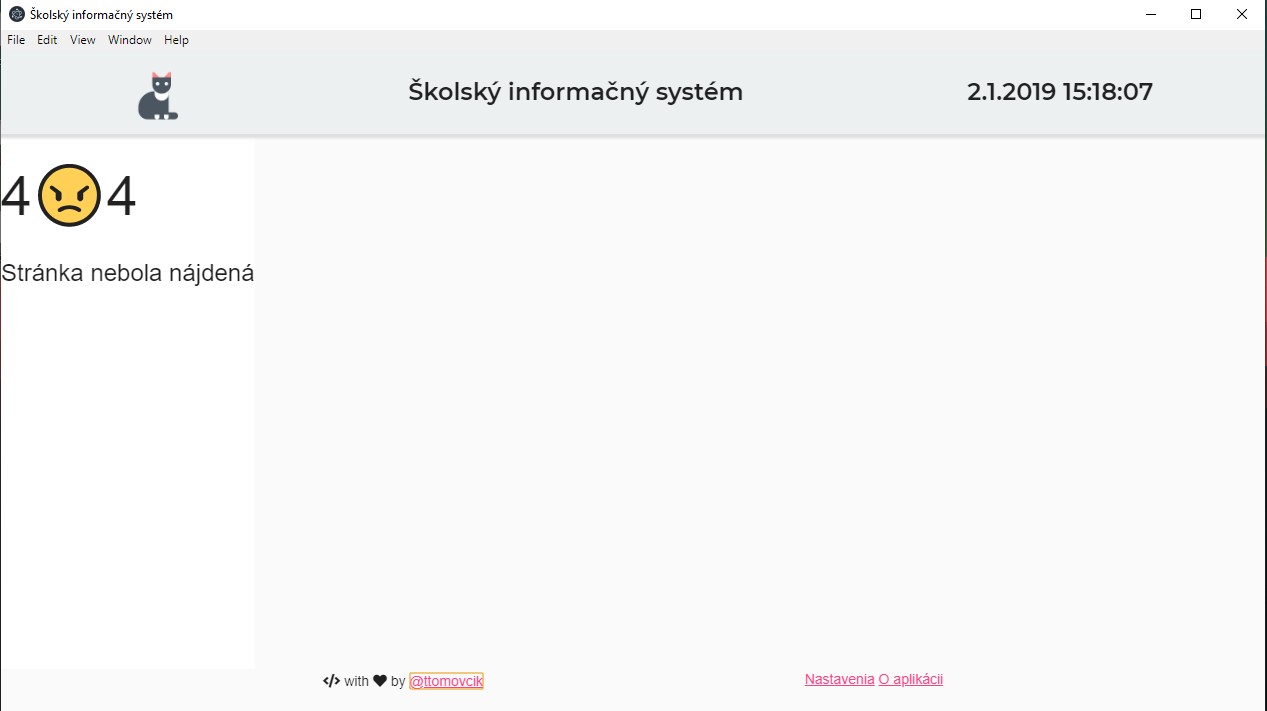
V procese navrhovania sa brali do úvahy všetky požiadavky a stanovené ciele a spravil som približný výpočet času potrebného na vývoj x2. T. j., ak napr. vývoj zaberie mesiac, výsledný čas potrebný by bol dva mesiace. Pôvodne som tento projekt robil pre Android. Nakoniec sme sa rozhodli prepísať túto aplikáciu pre viaceré zariadenia, pretože EduPage už má aplikáciu pre Android a nechcel som opakovať a vytvárať niečo, čo už na trhu je. Preto tvorba informačného systému, ktorý bude fungovať ako „nástenka“ a aplikácia súčasne sa javil ako lepší nápad. Práce na Android verzii aplikácie boli ukončené a archivované a začalo sa s návrhom, tvorbou prototypu a samotnej finálnej verzií po testovaní prototypu.

# 3.1 Prototyp

Tvorba prototypu sa začala vytvorením Git repozitára na stránke Github. Následne boli pridané základné súbory potrebné na identifikáciu repozitára a projektu samotného – konkrétne licencia, súbor README.MD a package.json. Ďalej som si zvolil framework, ktorý budem používať a programovací jazyk, keďže Android aplikácia bude nahradená neskôr PWA verziou. Zvolil som Electron (Obr. 8), ktorého repozitár „electron-quick-start“ som si naklonoval, prispôsobil a začal s tvorbou projektu na Github. Po dokončení testovacieho dizajnu som začal pridávať funkcie a testovať prototyp aplikácie. Pred tým, než som začal pracovať na tomto projekte som používal programovací jazyk Java, a Javascript bol pre mňa nový, takže som experimentoval ako funguje, na čo si treba dávať pozor a pomaly sa učil na chybách. Tvorba prototypu nebola plynulá ako som si predstavoval. Počas celej doby programovania som používal textový editor Visual Studio Code, ktorý vďaka rozšíreniam uľahčil prácu a spravil ju prehľadnejšou vďaka zvýrazňovaniu syntaxe kódu.

# 3.2 Prototyp: Dizajn aplikácie

Pred tvorbou prototypu som spravil najskôr dizajn v programe Gravit Designer a Adobe Xd. Tieto návrhy som ukázal niekoľkým ľuďom (Obr. 6).



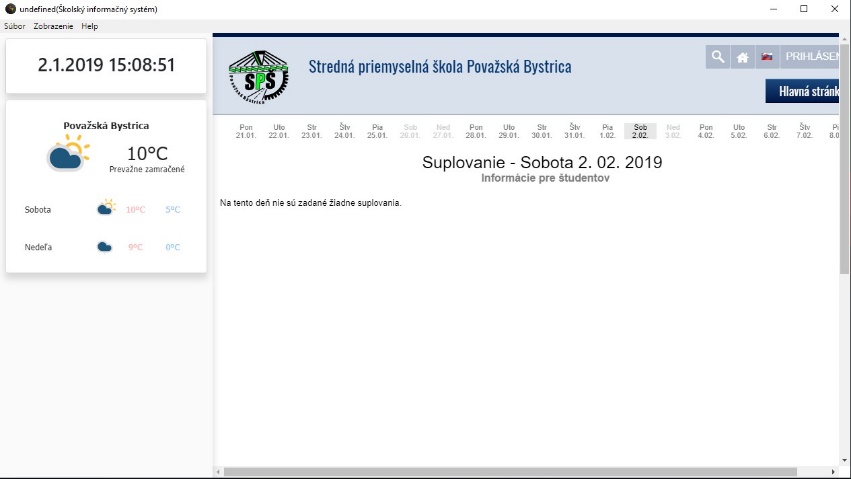
Obr.6 - Prvé UI

Na základe ich pripomienok a návrhov som sa rozhodol pôvodný dizajn upraviť.

Najčastejšou pripomienkou bolo, že aplikácie nepôsobí dostatočne „živo“. Niekoľko ďalších návrhov bolo:

* počasie,
* animácie,
* jednoduchšie zobrazenie, pestrejšie farby a väčší kontrast.

Na základe týchto pripomienok som preriešil možné zmeny, skúšal experimentovať s farbami, písmami a rozložením aplikácie. Taktiež som sa pýtal niekoľkých ľudí v skupine dizajnérov, čo si myslia o prvom návrhu (Obr. 6) a o novom návrhu, ktorý som vytvoril (Obr. 7). Viacerí sa zhodli, že je oveľa lepší, no aj tak tam chýbalo niekoľko vecí. Bolo ťažké rozoznať prvky a ich ohraničenia. Riešením bolo pridanie tieňov pod prvky, zvýšenie kontrastu a pridanie medzier medzi prvky. Výsledkom bol nový návrh (Obr .7)

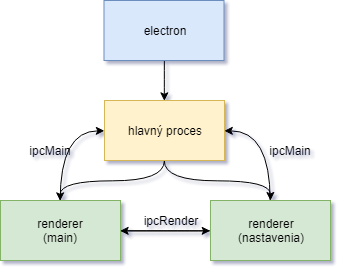


Obr. 7 - Hlavná obrazovka nového UI

Tento návrh počas testovania zaujal oveľa viac. Zaujal nielen spolužiakov, ale aj učiteľov.

# 3.3 Prototyp: Programovanie vo frameworku Electron

Po dokončení návrhu UI a funkcií, začal som s tvorbou samotnej aplikácie. Nainštaloval som si balíky nodejs, electron. bootstrap a material-design-lite. Vytvoril som niekoľko súborov, kde každý sa staral o inú časť aplikácie.



Obr. 8 - Štruktúra Electron aplikácie

Do hlavného súboru som pridal základné funkcie na načítanie electronu a zobrazenie render procesu, ktorý obsahoval jadro aplikácie. Následne som pridal nastavenia a spravil základný dizajn, aby som mohol testovať, či všetko funguje. Nainštaloval som balíky mocha, electron-store a console-helper. Balík mocha bol na testovanie aplikácie, electron-store na ukladanie nastavení a console-helper na logovanie výstupu do príkazového riadku a vývojárskych nástrojov Chromium. Na stránke StackOverflow som našiel takmer všetky odpovede a riešenia na problémy, a tak som začal s tvorbou elementu, ktorý bude zobrazovať aktuálny čas (Obr. 9). Moja funkcia síce fungovala na 50%, no prvok, ktorý mala aktualizovať neaktualizovala. Chyba bola nakoniec v zlom pomenovaní prvku „document.getElementbyId(‚dateTime‘);“.



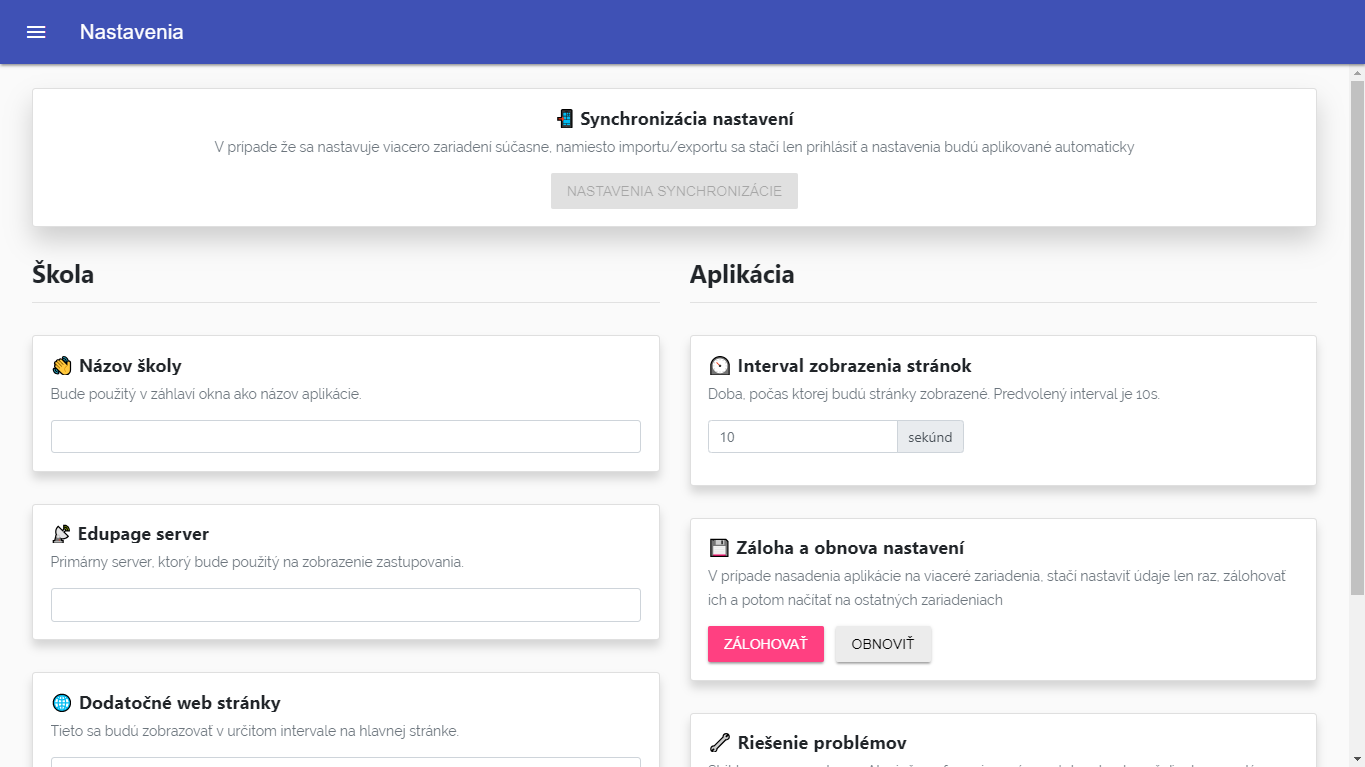
Obr. 9 - Funkcie na zobrazenie aktuálneho času a dátumu v hlavnom okne.

Po implementácií tejto funkcie som začal s pridávaním zobrazovania rozvrhu hodín, t .j. zvonenie na hodiny a prestávky (Obr. 10). Táto funkcia sa dá do budúcna vylepšiť a použije sa EduPage API, pomocou ktorého sa načíta skutočný rozvrh hodín školy a bude sa zobrazovať v určitých intervaloch.



Obr. 10 – Funkcia na zobrazovanie rozvrhu hodín

Po pridaní tejto funkcie som pridal miniaplikáciu pre počasie, ktorá bola riadená na diaľku iným serverom. Následne som sa pustil do prerábky nastavení, ich systému ukladania a vzhľadu. Použil som Material Design Lite a Bootsrap pre UI a vytvoril jedno záhlavie s názvom okna a 2x2 mriežku s nastaveniami (Obr. 11).



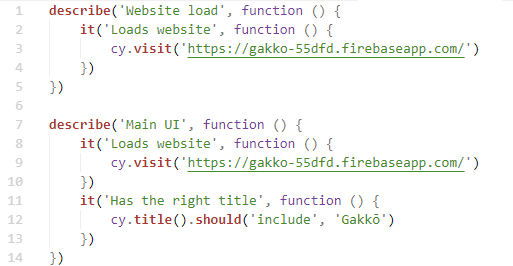
Obr. 11 – Nastavenia aplikácie

Taktiež som pridal automatické ukladanie nastavení, ktoré sa neskôr osvedčili ako užitočný detail. Je v ňom taktiež overovanie vstupu, čím sa predišlo chybám pri ukladaní. Ide o niekoľko jednoduchých „if“ funkcií.

Pri testovaní sa vyskytla chyba s načítaním stránok. Po zmene stránky sa pôvodná stránka neodstránila z pamäte. Toto sa dalo opraviť pomocou elementu „webview“, ktorý poskytoval electron a vďaka dokumentácii sa dal aj jednoducho prispôsobiť. Dokázal načítať stránky zo zoznamu.

# 3.4 Prototyp: Testovanie

Pri testovaní prototypu som používal knižnice mocha, a za pomoci jedného vývojára sa nám podarilo implementovať aj Cypress (Obr. 12). Mocha bol použitý na testovanie jednoduchých funkcií ako získanie aktuálneho času a dátumu. Cypress bol zasa použitý na uľahčenie práce pri testovaní, kde stačilo len raz napísať na čo sa má kliknúť, skontrolovať, napísať a následne to automaticky spravil v prostredí prehliadača Chrome, ktorý bol spustený v režime ladenia na diaľku.



Obr. 12 - Syntax Cypress-u

Cypress sa dal jednoducho nastaviť a stačilo ho len spustiť a všetky testy vykonal sám, čím ušetril niekoľko desiatok minút manuálneho klikania a písania.

# 4 PROGRAMOVANIE PRODUKČNEJ VERZIE

Po dokončení tvorby prototypu som sa pustil do prepisu aplikácie ako PWA s použitím knižníc a postupov overených Googlom. Archivoval som electron verziu a vytvoril si základnú web stránku na lokálnom serveri. Postupne som pridal potrebné nové veci a portoval staré knižnice z prototypu do produkčnej verzie.

# 4.1 Výber a testovanie Javascript frameworkov

Pred začatím programovania som sa poradil s niekoľkými programátormi ohľadom knižníc a náročnosti. Nakoniec som sa rozhodol pre Angular, React, Vue.js a klasický Javascript, s tým, že by som si potrebné veci napísal sám a získal tým väčší prehľad o tom ako aplikácia funguje, no zabralo by mi to viac času a zvyšuje sa tým riziko tvorby chýb. Ako prvý som začal skúšať AngularJS. Podľa hodnotení je jeden z najpoužívanejších a najjednoduchších. Základ aplikácie som mal vytvorený za niekoľko minút. Avšak nakoľko som zvolil Angular, Router a Typescript (superset vecí z javascriptu) bolo pre mňa zložité portovanie starých javascript funkcií do Typescriptu. Nie že by bol Typescript zložitý, no nestihol by som všetko načas a aplikácia by sa nedala testovať podľa plánu, čo som si nemohol dovoliť. Tak som prešiel na React, kde boli výsledky tie isté. Vue.js bol iný. Jeho dokumentácia bola omnoho prehľadnejšia, a dal sa rýchlejšie naučiť. Jedinou komplikáciou bola implementácia zobrazovania zvonenia. Všetko ostatné sa mi podarilo, dokonca aj ukladanie nastavení. Po pomerne dlhom hľadaní možných riešení na StackOverflow som to nakoniec vynechal. Nakoniec som zvolil iba základ, začal programovať vlastné veci bez použitia frameworkov pre samotnú aplikáciu. Jediný použitý framework bol jQuery, ktorý bol súčasťou balíka Bootstrap [1].

# 4.2 Dizajn aplikácie

Použil som pôvodný dizajn, kde som opravil chyby a pridal niekoľko funkcií. Rozhodol som sa pre Material Design Lite od Googlu, ktorý vznikol v roku 2015 ako zbierka postupov a pravidiel pre dizajn aplikácií pre Android. MDL dostal väčšiu aktualizáciu 2.0 v roku 2018, pre ktorú som sa rozhodol a použil som knižnicu MDL [8]. Ďalej som vybral šablónu zo stránky getMDL, konkrétne „fixed navbar with navigation drawer“. Týmto sa pridal responzívny dizajn aplikácie a taktiež sa použili vopred definované farby indigo a ružová. Indigo ako primárna farba a ružová pre dodatočné elementy ako tlačidlá a podtituly (Obr. 13).



Obr. 13 – Názov a logo aplikácie

# 4.3 Implementácia funkcií

Začal som s prepisom a úpravou funkcií a prvkov, ktoré budú mať na starosti nielen to čo uvidí koncový používateľ, ale aj služby, ktoré budú spustené na pozadí, ako napr. automatické aktualizácie a ukladanie súborov pre použitie offline (service worker Workbox od Googlu). Základné veci ako zobrazenie dátumu a počasia boli jednoduché. Hodiny sa skladali z niekoľkých funkcií. Prvá funkcia získala aktuálny čas a dátum v neupravenom formáte. Druhá funkcia so vstupným parametrom „i“ skontrolovala, či je vstupná hodnota menšia ako 10. Tieto funkcie budú spustené každých 500ms a vykonávať všetky potrebné úkony na zobrazenie času a dátumu v hlavnom UI. Pri zvonení a rozvrhu hodín bol problém so zosúladením aktuálneho času s časom zvonenia školy. Tento problém som vyriešil pomocou funkcie switch (Obr. 10) a spojenia dvoch hodnôt – hodín a minút do jednej premennej.

Ďalej som za pomoci Google dokumentácie a portálu DesignCourse pridal službu service worker (Workbox), ktorá zabezpečuje automatické aktualizácie a ukladanie súborov na offline použitie. Workbox si pri kompilácií a publikovaní zdrojového kódu vytvoril MD5 kontrolný súčet súborov a ich verziu vo formáte semver (Sémantický systém verzií). V prípade, že kontrolné súčty nesenia, Workbox skontroluje a prípadné chyby napraví. Ak sa zmení verzia súborov a prejdú kontrolou integrity, stiahne ich a pridá do vyrovnávacej pamäte. Tu sa vyskytol problém s ukladaním súborov a knižníc tretích strán. Tento problém som vyriešil pridaním licencií tretích strán a stiahnutím týchto súborov lokálne k zdrojovému kódu. Ako posledné som pridal nastavenia, kde bolo potrebné zaistiť overovanie vstupu používateľa, aby sa neukladali prázdne textové polia bez žiadnej zmeny alebo obsahu.

# 4.4 Testovanie

Po dokončení tvorby PWA verzie som začal s testovaním aplikácie. Použil som kombináciu manuálneho testovania a použitia testovacej knižnice Cypress (Obr. 12) . Pri manuálnom testovaní sa vyskytlo pretečenie pamäte. Tento problém bol spôsobený neodstránením starých stránok po načítaní novej. Oprava bola pomerne jednoduchá. Stačilo odstrániť obsah prvku iframe a opätovne ho načítať.

Automatické (integračné) testy v Cypress kontrolovali základnú funkcionalitu, ako načítavanie a ukladanie nastavení. Po dokončení Cypress uložil záznam obrazovky a protokol o testovaní, kde som videl aké problémy sa vyskytli pri testovaní. Celé testovanie trvalo približne 4 dni a v tomto časovom úseku sa vyskytlo približne 20 chýb, z ktorých najčastejšie boli chyby pri logických operáciách a chyby syntaxe.

# 5 ZÁVERY PRÁCE

Výsledkom tejto práce je funkčná progresívna webová aplikácia s modulárnym rozhraním. Síce táto aplikácia nebola jednoduchá na tvorbu, prepis z Android aplikácie na prototyp Electronu a následne ako PWA bol pomerne plynulý a rýchly. Necelých 600 hodín a približne 40 tisíc riadkov kódu. Po pár týždňoch sa aplikácia začala používať aj v Kanade a postupne sa testuje na školách. Aplikácia prešla iba jednou zmenou dizajnu, ktorá sa ujala. Pri PWA verzií bolo vytvorených 14 verzií, kde po 13. verziu sa pridávali nové funkcie a v 14. verzií sa začalo na oprave modularity a chýb. Počas týchto zmien sa zachováva existujúci dizajn a menia sa veci na pozadí. Posledná vec, ktorá bude implementovaná je časovač a aplikácia môže byť uverejnená ako Beta verzia. Beta sa bude testovať v reálnom prostredí, aby sa zistili nedostatky a po dokončení sa prejde na Release verziu, ktorá bude schopná fungovať bez problémov.

# 6 ZHRNUTIE

Všetky ciele sa mi podarilo splniť a výsledná aplikácia je funkčná v štádiu „alfa“ verzie. Dá sa používať, no neprešla dostatočným testovaním na to, aby sa dala každodenne používať.

Na aplikácií stále pracujem a postupne pridávam nové funkcie a opravujem chyby, ktoré sa objavia pri testovaní.

Aplikácia sa aktuálne testuje v Kanade, kde ju testuje firma GTmetrix, ktorá testuje jej rýchlosť a optimalizáciu. Výsledné hodnotenie bolo 92% a z testu YSlow 95%.Taktiež sa testovalo v Googli, kde výsledok bol pre mobily 78% a PC 99%.

Z knihy Hacking the hacker [7] som sa naučil ako zabezpečiť aplikáciu, aby sa predišlo útokom vo forme Buffer Overflow a iným. Napriek tomu, že kniha bola zameraná na C, C++ a Assembly, takmer všetko sa dalo aplikovať aj na Javascript.

# 7 ZOZNAM POUŽITEJ LITERATÚRY

[1] Bootstrap, <https://getbootstrap.com>, 01/2019

[2] Catapult systems, <https://blogs.catapultsystems.com>, 01/2019

[3] Firebase Console, <https://console.firebase.com>, 01/2019

[4] GTmetrix, <https://gtmetrix.com>, 01/2019

[5] GitHub, <https://github.com>, 02/2019

[6] Google Developers <https://developers.google.com>, 01/2019

[7] Hacking the hacker, Robert A. Grimes, ISBN: 978-1-119-39621-5, ePub vydanie, <https://wiley.com>

[8] Material Design, <https://grtmdl.io>, 12/2018

[9] Medium, <https://medium.com>, 02/2019

[10] Microsoft Developer Network <https://docs.microsft.com>, 02/2019

[11] Mozilla Developer Network <https://developer.mozilla.org>, 01/2019

[12] Sitepoint, <https://sitepoint.com>, 01/2019

[13] StackOverflow forums, <https://www.stackoverflow.com>, 11/2018

[14] freeCodeCamp, <https://freecodecamp.org>, 01/2019

[15] jQuery API, <https://api.jquery.com>, 01/2019

[16] w3schools, <https://w3schools.com>, 01/2019