Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Virtuálna FIIT

Breakpoint
Dokumentácia k dielu

Vedúca tímu: Mgr. Alena Kovárová, PhD.
Členovia tímu: Bc. Filip Mazán**,** Bc. Veronika Olešová, Bc. Filip Šoltés, Bc. Michal Kučera, Bc. Jozef Karas, Bc. Daniel Pribul

Školský rok: 2014/2015

Obsah

1 Úvod

Určite sa už každý z vás aspoň raz ocitol v situácii, kedy sa dostal do nového prostredia a cítil sa úplne stratený. Nepomohli vám ani navigačné tabule či ľudia naokolo. Sme presvedčení, že práve my vám dokážeme ponúknuť riešenie na tento problém – vyvíjame mobilnú aplikáciu s mapami, navigáciou a mnohými ďalšími informáciami o budove a dianí v nej. Je celkom logické, že ako prvá budova, pre ktorú to robíme je budova našej školy, preto sa aj náš projekt volá Virtuálna FIIT. Takže momentálne je aplikácia určená predovšetkým študentom prvého ročníka, ktorí sa takisto cítia často na začiatku semestra stratení. Samozrejme, vývoj riadime tak, aby sa celé riešenie dalo jednoducho preniesť na akúkoľvek inú budovu, či už školu, úrady, nemocnice, nákupné centrá, biznis centrá alebo veľké firemné budovy. Do tohto projektu bolo investovaného už veľa úsilia našimi predchodcami. Takže už dnes aplikácia poskytuje študentom mnoho funkcionalít, medzi ktoré patrí napríklad prezeranie si máp fakulty, vlastného rozvrhu, obedov v okolí či odchod najbližších autobusov. My máme v pláne ich udržiavať, vylepšovať a rozširiť o nové.

V tomto dokumente je okrem globálnych cieľov stanovených na zimný a letný semester aj podrobný popis vyvíjanej aplikácie, jej celková architektúra a všetky jej moduly. V architektúre je najprv popísaný stav, v akom sme aplikáciu dostali, a potom aj stav v akom sa aplikácia aktuálne nachádza. Každý modul potom obsahuje zoznam používateľských príbehov a podrobný popis úloh, ktoré sa v rámci nich riešili. V prílohách sa nachádza používateľská a všetky potrebné inštalačné a integračné príručky.

2 Slovník pojmov

Výraz	Vysvetlenie
parsovanie	proces, pri ktorom sa vstupný text transformuje na určité dátové štruktúry
parser	nástroj, ktorý vykonáva parsovanie
cache	vyrovnávacia pamäť
CORS	skratka pre *Cross-origin resource sharing*;
BLE	skratka pre *Bluetooth low energy*
API	skratka pre *Application programming interface*
PhoneGap	voľne dostupný rámec na tvorbu mobilných aplikácii pomocou webových technológii
Cordova	súbor API, umožňujúci pristupovať vývojárom k natívnym funkciám zariadenia
front-end	časť aplikácie viditeľná bežným návštevníkom
AIS	skratka pre *Akademický informačný systém*

3 Globálne ciele

Najväčší potenciál vidíme v lokalizácii a následnej navigácii používateľa v budove. Toto sme si na začiatku semestra stanovili ako svoj hlavný cieľ v tomto projekte, pretože bez tejto funkcionality sa žiadny informačný systém viazaný na komplexné budovy neobíde. Takúto navigáciu rozhodne ocenia používatelia, ktorí neradi strácajú čas študovaním mapy. Na otvorených priestranstvách sa na tento účel používa GPS signál, ten však vo vnútorných priestoroch zlyháva, a teda je potrebné nájsť iné riešenie lokalizácie. Spomedzi existujúcich technológii nám vyšlo najpraktickejšie použitie nízkoenergetických vysielačov – *Bluetooth LE Beacon-ov*. Existujúce implementácie takéhoto riešenia nie sú dostatočne presné, čo sme prijali ako výzvu pre implementáciu vlastnej navigácie. Spôsobov, akým je možné problém uchopiť je viacero. Vzniká možnosť poňať projekt výskumne a zistiť, ktorá metóda by bola v danom prostredí najvhodnejšia.

Medzi naše ciele taktiež patrí získanie schopnosti práce v tíme. Na to, aby sme boli v tomto projekte úspešní, je potrebná dobrá tímová komunikácia, spolupráca a pochopenie jeden druhého.

3.1 Zimný semester

Do konca zimného semestra plánujeme vytvoriť prototyp, v ktorom bude fungovať lokalizácia v rámci miestnosti. Menej náročným, ale snáď rovnako dôležitým cieľom je aj prepojenie našej aplikácie so systémom *Askalot* (portál pre študentov na kladenie otázok a vyhľadávanie/získanie odpovedí). Tiež plánujeme vyriešiť problém so zabezpečením spojenia (to nebolo možné zabezpečiť kvôli obmedzeniam produkčného servera) tým, že si nainštalujeme svoj vlastný produkčný server a zoptimalizujeme tak aj celú komunikáciu medzi klientom a serverom. V prípade záujmu iných fakúlt STU radi rozšírime našu aplikáciu aj pre nich a to s minimálnou námahou.

3.2 Letný semester

Do konca školského roka sme si dali za cieľ vyvinúť plne fungujúci modul lokalizácie, ktorý bude spolu s celou aplikáciou opäť nasadený na *Google Play* obchod. V priebehu zimného

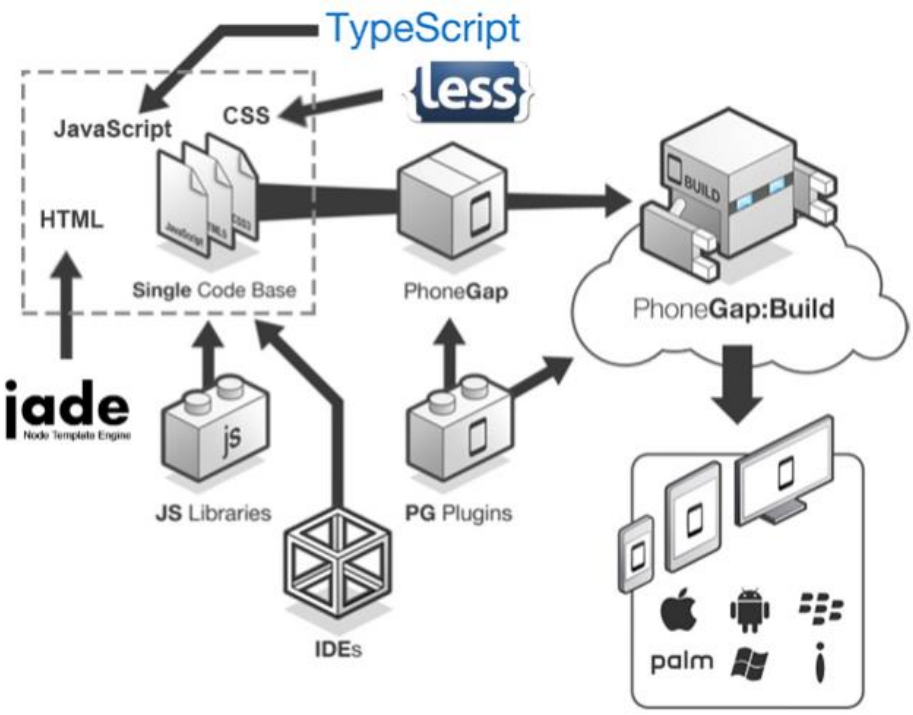
semestra sme sa zhodli na prerobení celej aplikácie odznova. Túto úlohu chceme splniť počas letného semestra spolu s vytvorením nového dizajnu aplikácie. Funkcionalitu budeme dôkladne testovať aj pomocou automatických testov či už na serverovej alebo klientskej strane. Chceme taktiež spolupracovať s našimi používateľmi najviac ako to bude možné, preto uvedieme do prevádzky beta testovanie.

4 Celkový pohľad

4.1 Pôvodná verzia

Zdedili sme aplikáciu, na ktorej pred nami pracovalo už niekoľko tímov. Verzia, ktorú sme dostali bola postavená na technológii PhoneGap, ktorá umožňuje vytvárať aplikácie pre rôzne mobilné platformy pomocou bežných webových technológií ako sú HTML, JavaScript alebo CSS. Aplikácia VirtualFiit, žiaľ podporuje iba operačný systém Android od verzie 2.3.3. Táto aplikácia je taktiež dostupná vo forme webovej stránky.

Na zjednodušenie práce s kaskádovými štýlmi bola využitá knižnica Less, na doplnenie JavaScript-u o OOP prvky knižnica TypeScript a šablóny boli vytvárané pomocou šablónového agregátu xjade. Na zjednodušenie vývoja aplikácie bol použitý Grunt.js.

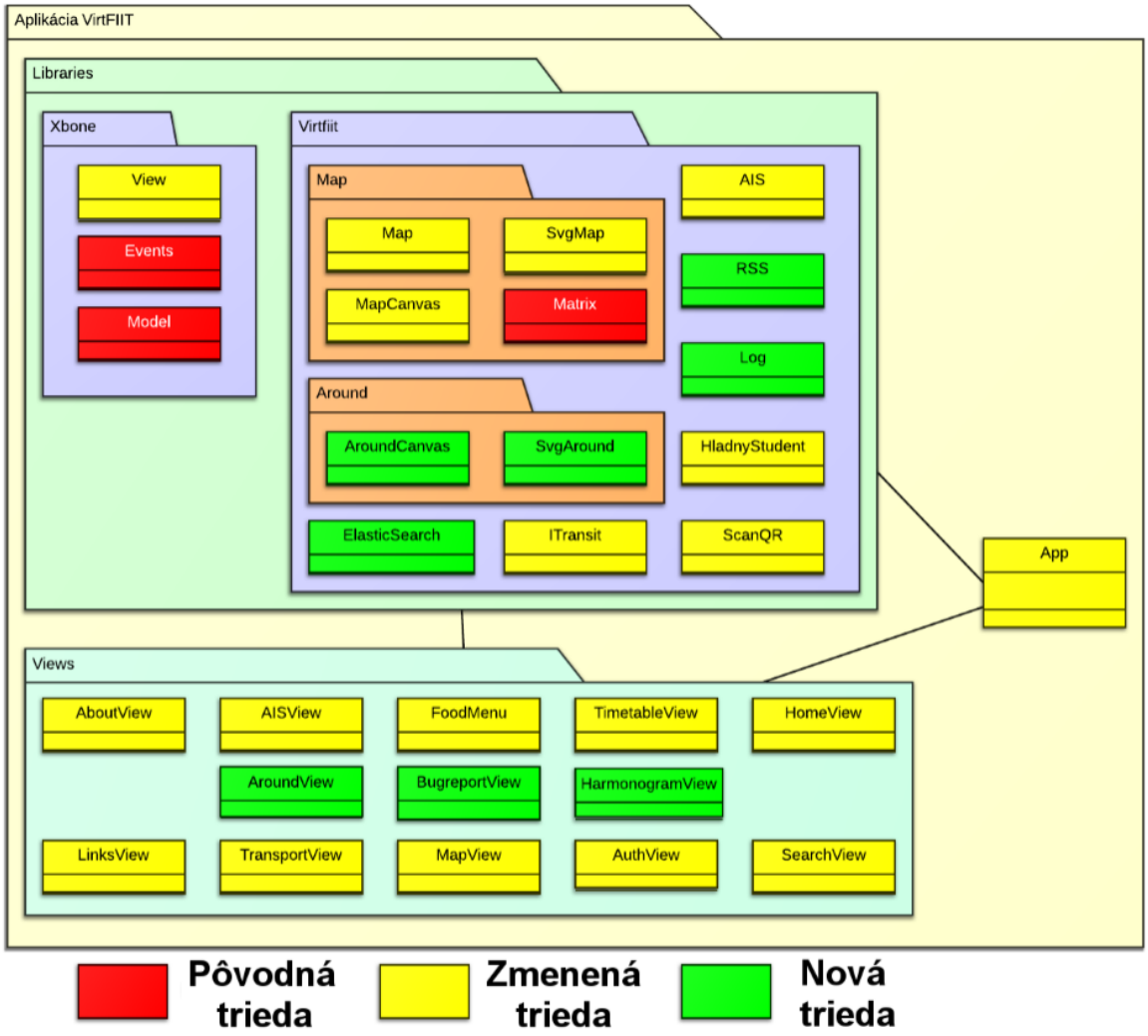


Obr. 1 Diagram použitých technológií a ich prepojenia (zdroj: minuloročný tím)

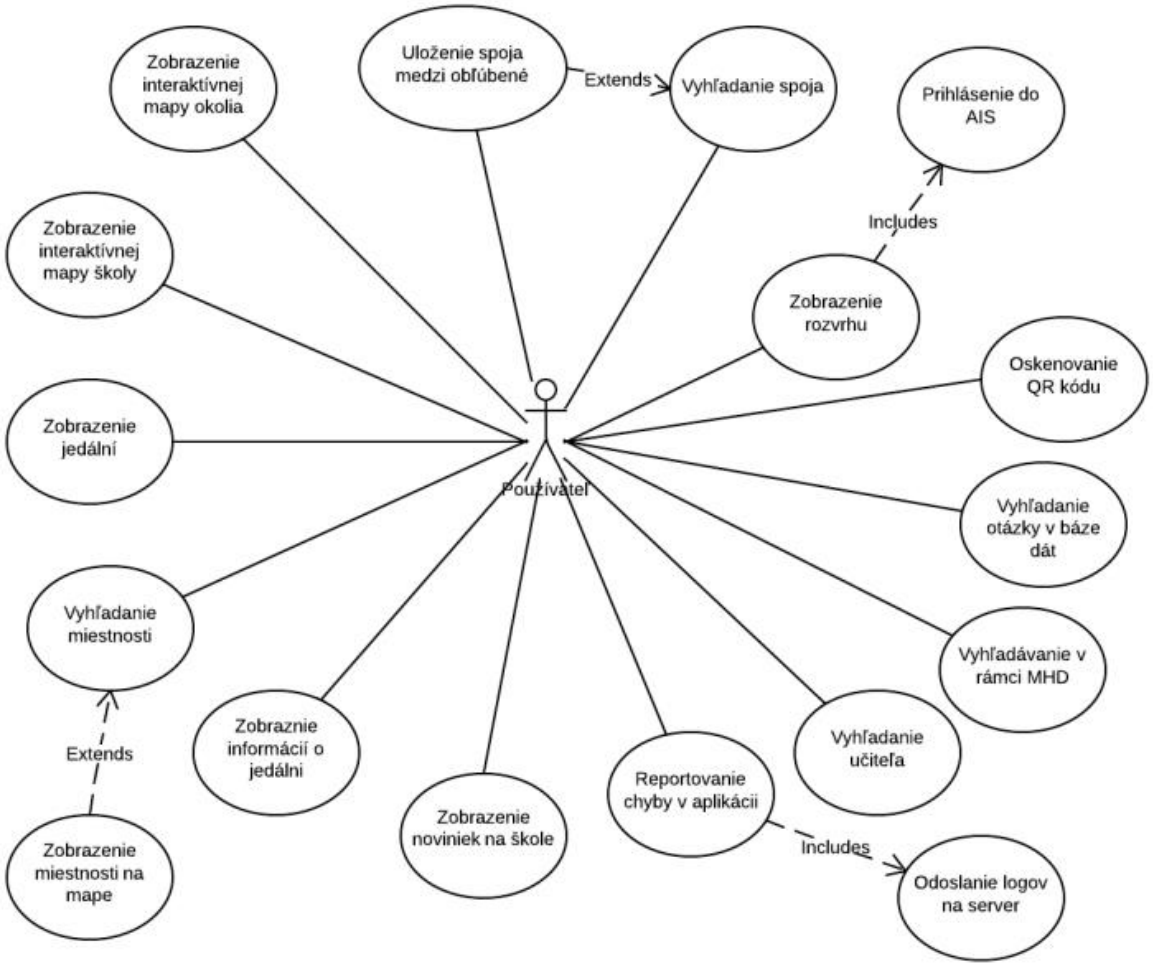
V aplikácii sa využívajú princípy MVC, kde dátový model, riadiaca logika a používateľské rozhranie boli od seba oddelené. Organizácia dostupných tried je zobrazená na Obr. 2. Hlavnou triedou aplikácie je trieda *App*.

Aplikácia pozostávala z týchto modulov: *AIS, Jedálne, Rozvrh, MHD, Vyhľadávanie, Mapy, QR kódy, BLE lokalizácia* a ďalších informačných obrazoviek. Bližší popis pôvodného stavu a našich zmien v týchto moduloch sa nachádza v kapitole *Moduly Virtuálna FIIT*.

Aplikácia pracuje aj s informáciami tretích strán. Od *is.stuba.sk* a *fiit.stuba.sk* získava parsovaním údaje o škole. Z *itransit.sk* a *hladnystudent.sk* získava prostredníctvom API cestovné poriadky a jedálne lístky.



Obr. 2 Diagram balíkov s vyznačenými zmenami predošlého tímu (zdroj: minuloročný tím)



Obr. 3 Diagram prípadov použitia identifikovaný predošlým tímom (zdroj: minuloročný tím)

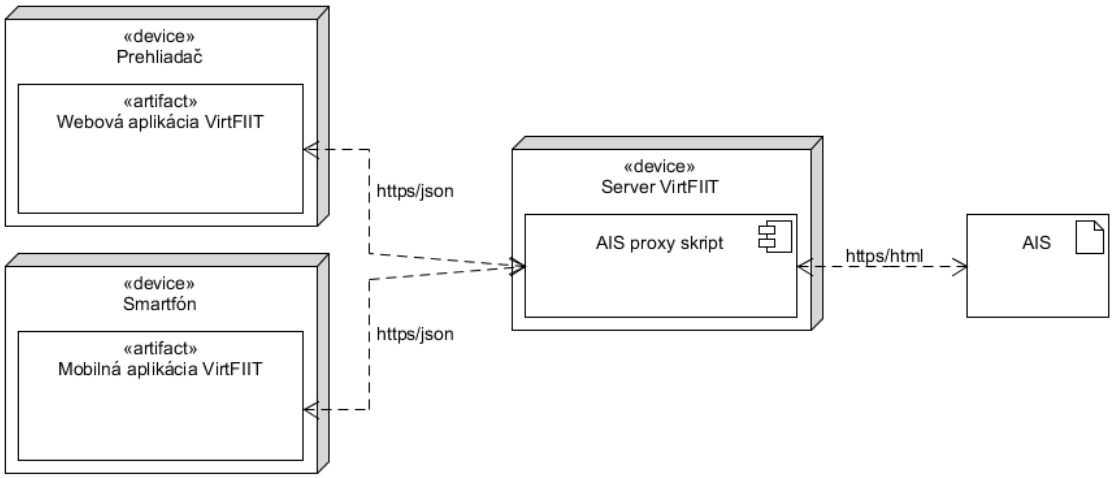
4.2 Naše zmeny

V priebehu semestra sme sa rozhodli ukončiť vývoj starej verzie aplikácie a začať vyvíjať odznovu. Naše zmeny, ktoré sme v rámci tohto projektu vykonali, sa preto dajú rozdeliť na dve časti. Na práce vykonané v starej verzii aplikácie a na vývoj novej verzie.

4.2.1 Stará aplikácia

Pomerne veľké množstvo času sme strávili opravou chýb, ktoré sme objavili v pôvodnej verzii projektu. Išlo o nefungujúce prihlasovanie pre študentov s vlastnými šablónami pre AIS, chybné zobrazovanie a načítavanie jedálnych lístkov a problém pri mazaní rozvrhu z *cache*.

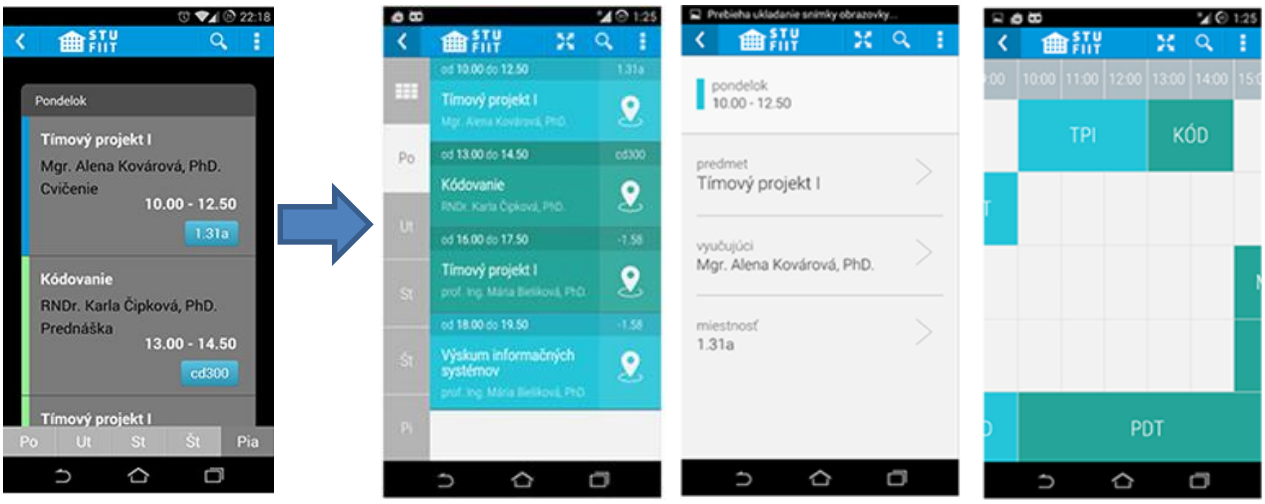
Prihlasovanie bolo presmerované cez náš server, kde sa využíva menej striktný parser. Prihlasovacie údaje sa budú po novom posielať cez protokol HTTPS. Zmeny sú zobrazené na *Obr.* 4.



Obr. 4 Diagram znázorňujúci nový spôsob prihlasovania

Niektoré obrazovky v aplikácii sme považovali za neprehľadné a rozhodli sme sa ich zmeniť. V starej aplikácii sme upravili harmonogram a rozvrh, pričom pri rozvrhu sa okrem vzhľadu pridávali aj ďalšie obrazovky pre detail predmetu/cvičenia a pre pohľad na celý týždeň. Tieto zmeny sú zachytené na Obr. 5.

a1) a2)



b1) b2) b3) b4)

Obr. 5 Upravené obrazovky, a – harmonogram, b – rozvrh (ľavá strana staré, pravá strana nové)

Keďže *PhoneGap* neobsahuje žiadne vhodné zásuvné moduly na prácu s BLE technológiou, vytvorili sme nový *Cordova* zásuvný modul s názvom *BeaconPlugin*. Ten zatiaľ dokáže skenovať BLE zariadenia v okolí.

4.2.2 Nová aplikácia

Po 3. šprinte sme sa rozhodli vytvoriť aplikáciu Virtuálna FIIT nanovo. Dôvodov bolo viacero. V prvom rade sa nám nepáčil technický stav aplikácie, spôsob, akým bola naprogramovaná a použité technológie, ktoré spôsobovali množstvo problémov a zbrzďovali vývoj. V druhom rade to bol aj zastaralý vzhľad aplikácie a jej slabá použiteľnosť. Rozhodli sme sa preto začať odznovu, použiť iné technológie, zmeniť dizajn a prehodnotiť niektoré funkcie.

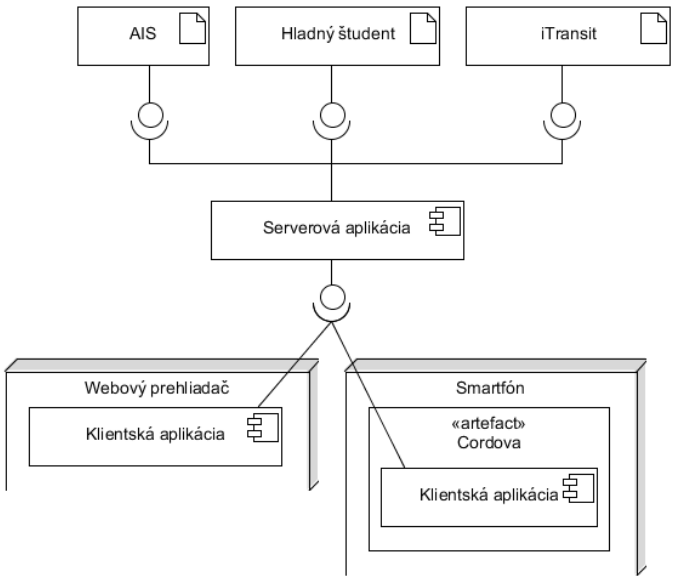
Pri výbere vhodnej technológie sme kládli dôraz na multiplatformovosť, pohodlnejší vývoj a plynulejší chod aplikácie (odstránenie sekania, plynulejšie prechody medzi obrazovkami, minimalizovať načítavacie časy, ...). Rozhodli sme sa pre rámec *Ionic*[1], ktorý tieto naše podmienky spĺňal. Ten má v sebe už zabudovaný JavaScript rámec *AngularJS*[2] a CSS rozšírenie *Sass*[3]. *Ionic* navyše obsahuje množstvo predpripravených CSS tried, ktoré výrazne urýchlujú prototypovanie obrazoviek. Pri práci so šablónami sme sa rozhodli vynechať nástroj *XJade* a využívať budeme iba HTML doplnené o direktívy z *AngluarJS*. *Ionic* využíva takisto

[1] http://ionicframework.com/
[2] https://angularjs.org/
[3] http://sass-lang.com/

technológiu PhoneGap, čiže tá zostala nezmenená. Serverovú časť, ktorá bola pôvodne naprogramovaná v jazyku PHP, sme sa rozhodli kvôli zjednoteniu s klientskou časťou prerobiť pomocou technológie Node.js. Napriek tomu, že aj v minulosti bola cieľom podpora viacerých platforiem, aplikácia bola vždy dostupná iba pre Android a web a v danom stave ju nebolo možné sprístupniť pre ostatné platformy. Použitím rámca *Ionic* sme vyriešili aj tento problém, pričom aplikácia je pripravená aj pre nasadenie na *iOS* a neskôr bude možné ju jednoducho prispôsobiť aj pre *Windows Phone.*

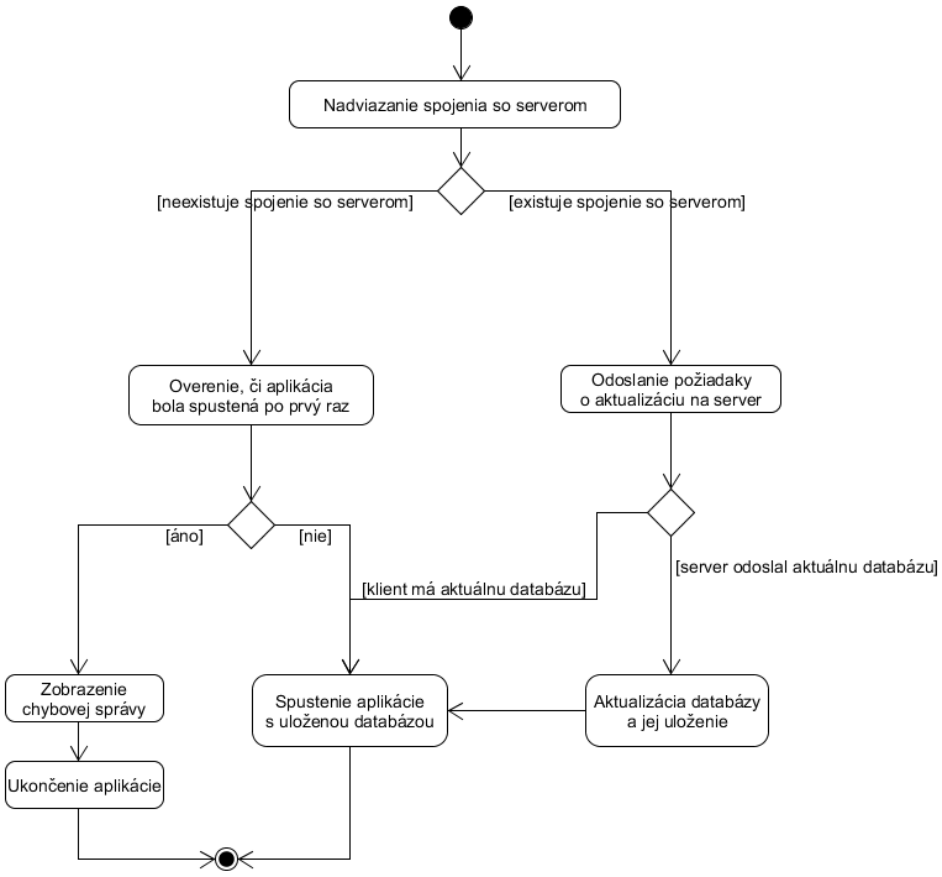
Po architektonickej stránke sme sa rozhodli prerobiť aplikáciu z "tučného" na "tenkého" klienta. Aplikácia po novom už iba zobrazuje informácie, ktoré sú kompletne spracovávané na strane servera. Týmto sme vyriešili napríklad doterajší problém, kedy pri každej drobnej zmene (napríklad v rozvrhu) bolo nutné pripraviť novú verziu aplikácie. Ďalšou novinkou je aj to, že aplikácia získava informácie o jedálňach a MHD už iba z nášho servera a nie priamo zo serverov tretích strán. Podmienkou vždy bolo aj fungovanie v prípade, že nie je dostupné internetové pripojenie. Z toho dôvodu sa všetky poskytované informácie ukladajú ako na strane klienta, tak aj na strane servera. Oproti starej verzii bola vďaka prechodu na protokol HTTPS zvýšená aj bezpečnosť aplikácie.

Na Obr. 6 je zobrazený celkový pohľad na architektúru na najvyššej úrovni abstrakcie.



Obr. 6 Celkový pohľad na architektúru

Na Obr. 7 nižšie sa nachádza diagram aktivít popisujúci proces aktualizácie databázy pri štarte aplikácie pri rôznych podmienkach.

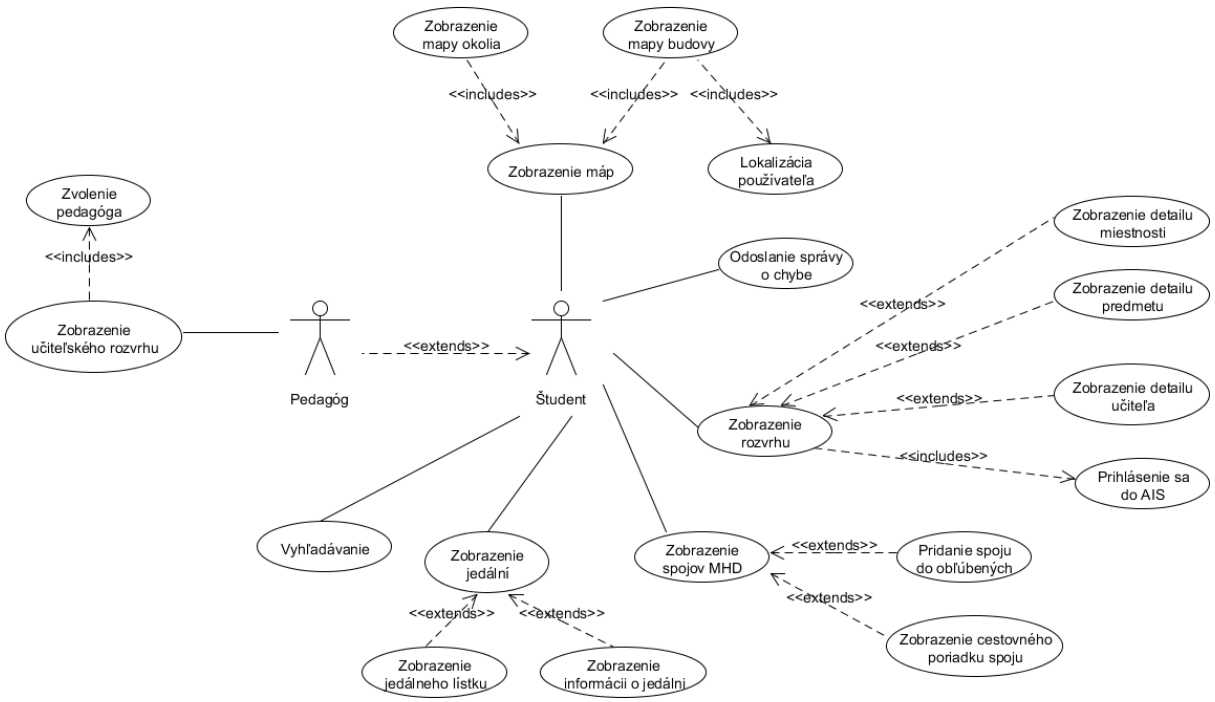


Obr. 7 Diagram aktivít aktualizácie databázy

Klientská časť

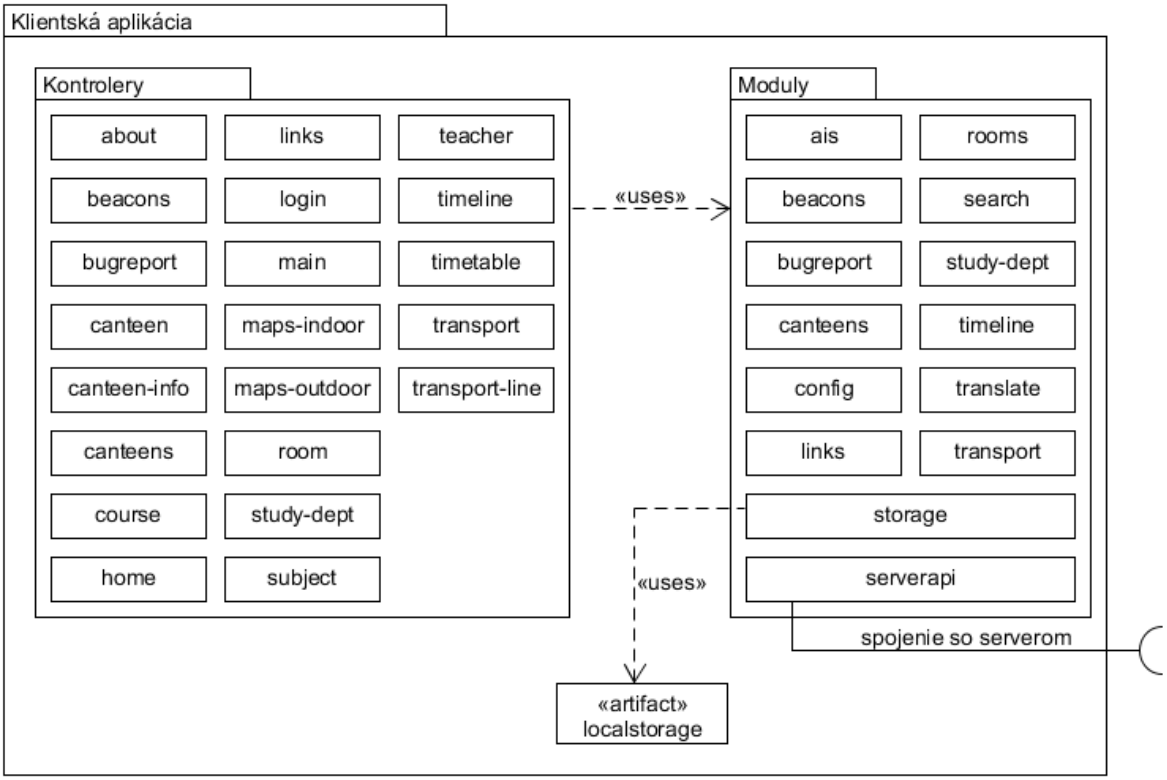
Všetky existujúce funkcie zostali v novej verzii zachované, pričom boli pridané ďalšie. Novinkami sú napríklad najbližšia prednáška v úvodnej obrazovke alebo úradné hodiny knižnice. Používatelia si navyše môžu meniť niektoré aspekty aplikácie aj v novo pridaných nastaveniach.

Diagram prípadov použitia, ktoré sme identifikovali v novej aplikácii, sa nachádza na Obr. 8.



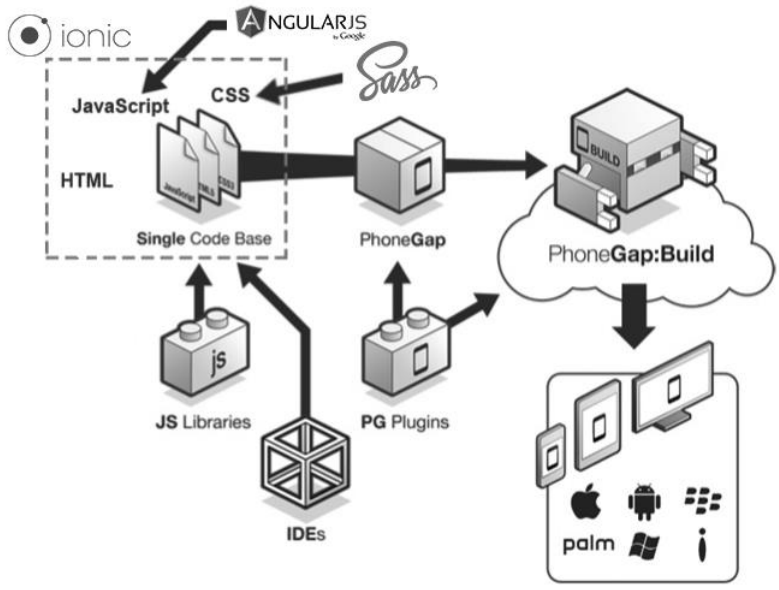
Obr. 8 Diagram prípadov použitia nového systému

Architektúra nového systému je vyobrazená na *Obr. 9.* V novej aplikácii nevyužívame žiadne triedy ani inú funkcionalitu zo starej.



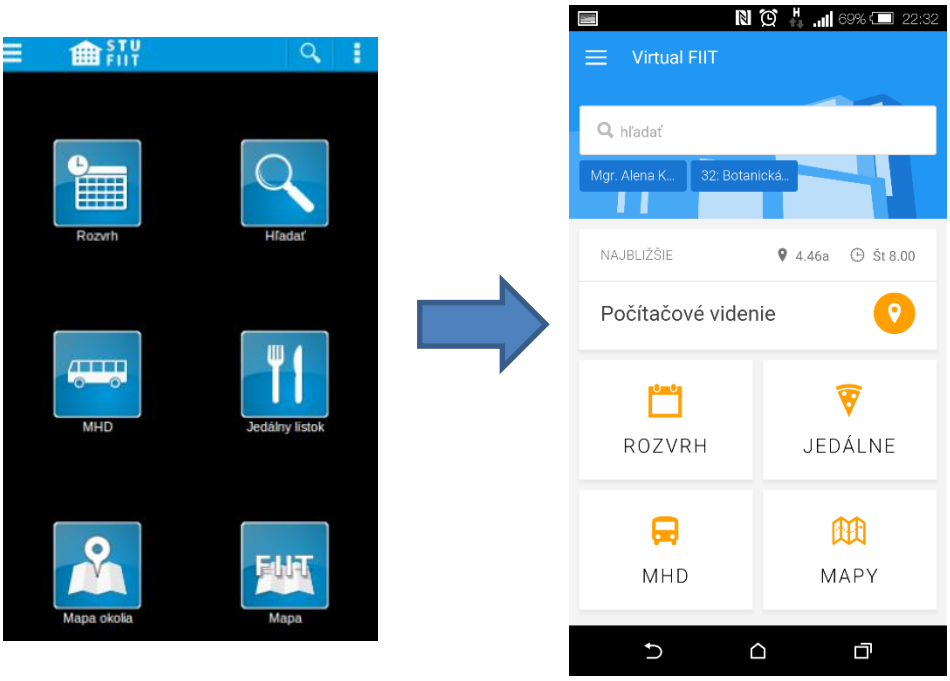
Obr. 9 Diagram balíkov nového systému

Použité klientské technológie a ich prepojenie je znázornené na *Obr. 10.*

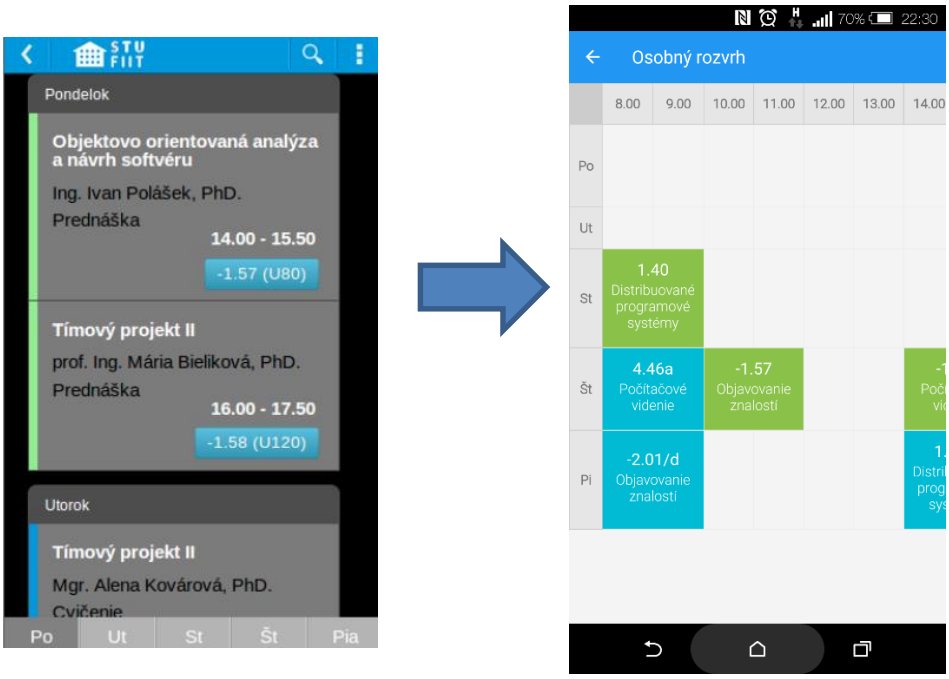


Obr. 10 Upravený diagram použitých klientských technológií a ich prepojenie

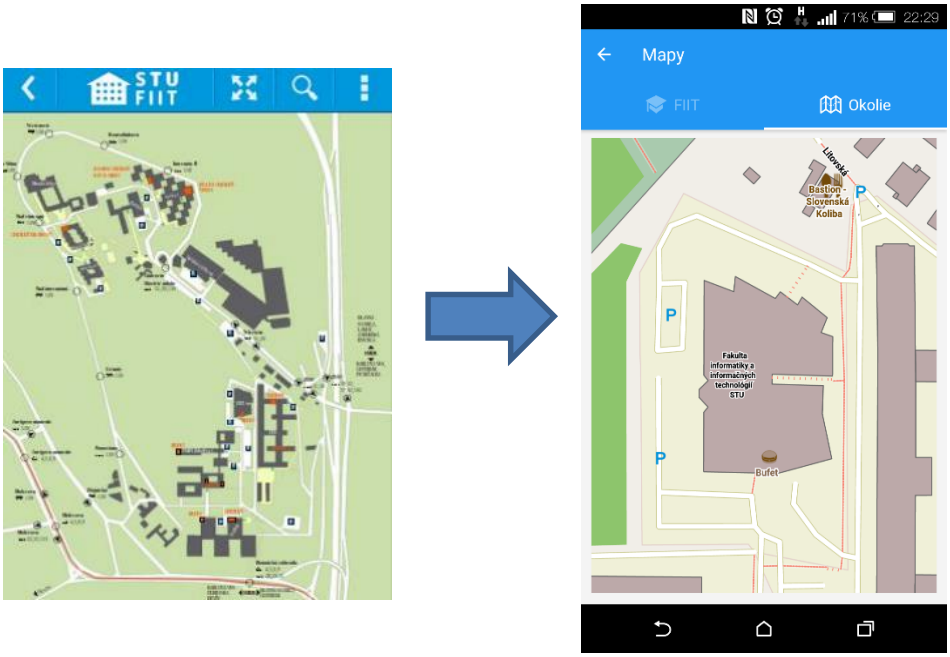
Po vizuálnej stránke sme sa rozhodli dať prednosť modernejšiemu a prehľadnejšiemu dizajnu. Nový dizajn bol navrhnutý v štýle tzv. plochého dizajnu, ktorý bol čiastočne inšpirovaný Material dizajnom od Google. Dôraz bol kladený na obsah, prehľadnosť a konzistentnosť. Použité boli svetlejšie farby, jednotné ikonky a organizácia pomocou kariet. Porovnanie starých obrazoviek s novými je možné vidieť na Obr. 11. Ostatné obrazovky sú kvôli veľkému množstvu priložené v Príloha A.



a1) a2)



b1) b2)



c1) c2)

Obr. 11 Upravené obrazovky, a – domov, b – rozvrh, c – mapa okolia (ľavá strana staré, pravá strana nové)

Serverová časť

V tejto sekcii je podrobne opísaná architektúra serverovej časti aplikácie, pričom nevyhnutné kroky na jej integráciu sú uvedené v Príloha C.

Nástroje

Pre serverovú stranu sme vybrali rámec `expressjs` a to z dôvodu relatívne veľkej komunity a tiež malého rozsahu rámca, keďže rozsiahlejšia funkcionalita pre naše využitie nie je potrebná.

Architektúra

Na najvyššej úrovni je aplikácia postavená na základe návrhového vzoru fasáda, keďže vyžíva významnú vlastnosť rámca a to smerovanie REST dopytov jednotlivým funkciám pomocou tzv. *routes*.

Každá z *routes* je fasáda pre svoj vlastný modul. Tieto súbory sa nachádzajú v adresári `server/routes/` a nenachádza sa v nich žiadna biznis logika, len jednoduché spustenie požadovanie funkčnosti v module a vrátenie dát pomocou *callback* funkcie.

Pred tým, než je dopyt poslaný do jednej z *routes* je vložený do modulu nazvanom *middleware*. V tomto module je vykonané predspracovanie každého dopytu.

Ako fasádu môžeme chápať aj organizáciu jednotlivých modulov. V každom sa nachádza súbor `index.js`, ktorý obsahuje jednu exportovanú funkciu, pre každú funkciu publikovanú pomocou API. Úlohou `index.js` je spájať funkcionalitu ostatných súborov, pričom štruktúra je nasledovná:

1. validácia vstupu,
2. business logika,
3. vytvorenie jednotne štruktúrovaného výstupného objektu.

Jednotlivé časti logiky sú podľa zamerania rozdelené do súborov. Súbory v module (okrem `index.js`) majú v názve názov modulu ako prefix.

Pri vytváraní business logiky je dodržiavaná zásada, že jednotlivé moduly nie sú vzájomne previazané. Pripustiteľné je jedine naviazanie na modul `common`, ktorý združuje spoločnú funkcionalitu.

Smerovanie dopytu:

API --> middleware --> routes --> fasáda konkrétneho modulu --> validácia --> [odoslanie dopytov na externé API, agregácia, parsovanie, cache -->] vytvorenie výstupného objektu

Middleware

V module *middleware* je vykonávané predspracovanie konkrétnych dopytov pričom pre každý úkon existuje jednotlivý súbor. Súbory *middleware* sa nachádzajú v adresári `server/middleware`.

V module je buď dopyt spracovaný a preposlaný tak, ako bol prijatý (napr. server side logger), alebo upravený pred odoslaním (napr. injektovanie hlavičiek v CORS).

Routes

Routes fungujú ako fasáda pre jednotlivé komponenty aplikácie. Ako už bolo spomenuté, v týchto súboroch sa nenachádza žiadna business logika, ale len jednoduché preposlanie dopytu do príslušného modulu.

V *routes* sú rozdeľované dopyty z jednotlivých URL na konkrétne funkcie modulu. Tiež je tu spravidla oddelená *meta* časť dopytu. Ak však funkčnosť modulu túto časť vyžaduje, dopyt je preposlaný celý (napr. nahlasovanie chyby).

Validácia

Pre validáciu existuje v moduloch, kde je to potrebné, súbor `<nazov_modulu>Validator.js`. Tento modul poskytuje funkcie pre validáciu konkrétnych prípadov. Validačné funkcie vracajú príznak, či bola validácia úspešná a taktiež pridávajú chybovú hlášku do chybového poľa.

Každá z konkrétnych validačných funkcií využíva jednu alebo viac všeobecných validačných funkcií z modulu `common/validation`, pričom ich využíva s konkrétnymi parametrami.

Všeobecné validačné funkcie sú všetky *callback* funkcie `genericValidator`, ktorá vyhodnotí výsledok *callbacku* a pridá chybovú hlášku, ak je zadaná.

Dopyty na externé servery

Pre dopyty na externé servery existuje v moduloch, kde je to potrebné, súbor `<nazov_modulu>Request.js`. Tento modul poskytuje funkcie pre vyskladanie konkrétnych dopytov na externé dopyty.

Pre samotné vykonávanie dopytov používame plugin *request*. Keďže však značná časť logiky bola rovnaká, rozhodli sme sa vyrobiť wrapper tohto pluginu, a to hlavne z dôvodu jednotného ošetrovania chýb.

Všetky moduly preto nepoužívajú priamo plugin *request*, ale wrapper umiestnený v module `common.request`.

Jednotlivé funkcie wrappera sú namapované na funkcie samotného pluginu, sú však upravené pre naše potreby.

Cache

Pri dopytoch na externé API je veľmi dôležité cachovanie, keďže ním znižujeme celkový čas dopytu. Pre cache využívame databázu `redis`.

Ak modul využíva cachovanie, sú všetky potrebné metódy združené v súbore `<nazov_modulu>Cache.js`, pričom tento súbor využíva funkcie z generického cache modulu v `common/cache`. V tomto module je vytvorené zdieľané pripojenie do databázy a exportovaný klient, ktorý je využívaný konkrétnymi modulmi.

Najdôležitejšou funkciou generického modulu je funkcia `loadOrFetchJson`, ktorá buď načíta dopytované dáta z cache, alebo ak sa v cache nenachádzajú, načíta ich pomocou funkcie zadanej ako parameter.

Parsovanie

Pre parsovanie HTML je využitá existujúca knižnica `cheerio`. Pre modul, kde je potrebné parsovanie, existuje samostatný súbor s názvom `<nazov_modulu>Parser.js`. V tomto súbore je vytvorená trieda združujúca metódy potrebné na parsovanie jednotlivých údajov.

Pre každý údaj, ktorý je potrebný v module (napr. rozvrh, počet emailov, ...) je vytvorená jedna *public* metóda, pričom pomocné metódy sú zapuzdrené označením pomocou prefixu `_`.

HTML je do triedy vložené pomocou konštruktora, pričom je celý reťazec uložený ako premenná objektu. Z toho vyplýva potreba vytvorenia samostatných objektov pre jednotlivé dopyty.

Vytvorenie výstupného objektu

Pre každú službu ponúkanú pomocou API aplikácie je vytvorený výstupný objekt s presne definovanou štruktúrou. O túto funkcionalitu sa stará funkcia module `common/formatter`.

Objekt má formát

```
{
        success:true/false,
        data: {<data>},
        errors: [<chybove_kody>]
}
```

5 Moduly Virtuálna FIIT

5.1 AIS a Rozvrh

Pôvodná verzia: Tento modul sprístupňuje používateľom informácie o miestnostiach, vyučujúcich, predmetoch, a rozvrhoch a zabezpečuje aj prihlasovanie do aplikácie. Nakoľko AIS neposkytuje žiadne API, tieto informácie sa získavajú parsovaním webovej verzie AIS. To sa vykonáva v pravidelných intervaloch alebo pri prihlasovaní. Používateľovi sa po prihlásení vyparsuje jeho osobný rozvrh a následne zobrazí na mobilnom zariadení. Rozvrh sa mu uloží do cache, aby sa dal neskôr prezerať aj bez internetu. V pôvodnej verzii bol rozvrh reprezentovaný formou dlhého zoznamu.

Naše zmeny v pôvodnej verzii: Zistilo sa, že prihlasovanie do aplikácie zlyhá, ak má používateľ nastavenú v AISe inú ako základnú šablónu. Prihlasovanie sme teda presmerovali cez náš server, kde sa využíva menej striktný parser. Prihlasovacie údaje sa budú po novom posielať cez protokol https. Opravili sme taktiež chybu, ktorá spôsobovala, že bolo možné vidieť cudzí rozvrh. Navrhli a implementovali sme novú verziu rozvrhu, kde pribudol pohľad na celý týždeň a upravili sme aj vzhľad aktuálnej verzie.

Nová verzia: V novej verzii sme ponechali všetku funkcionalitu z pôvodnej. Rozvrh sa už však neparsuje na klientovi, ale na serveri. Riešili sme takisto aktualizáciu rozvrhu. Výsledkom je priebežné dopytovanie sa na server, či je potrebné rozvrh používateľa aktualizovať.

***Používateľský príbeh**: Používateľ sa chce v aplikácii prihlásiť, aby si mohol pozrieť svoj rozvrh.*

5.1.1 Pozrieť sa na šablóny AIS, prečo parser zlyháva

VFIIT-14, Veronika Olešová, Filip Šoltés

Analýza - Používatelia s istými šablónami majú problém prihlásiť sa do AIS cez mobilnú aplikáciu. Po zadaní prihlasovacích údajov a následnej snahe o prihlásenie sa, je obsah formulára vymazaný a používateľ sa nie je schopný dostať ku svojmu rozvrhu. Používateľ nedokáže zistiť z akého dôvodu sa toto deje. Dôvodom zlyhávania šablón je nesprávny formát ich rozšíreného HTML. Príkladom je nesprávne ukončenie HTML elementu alebo chýbajúce úvodzovky pri

odkazovaní sa na obrázok. Domnievame sa, že *DOMParser*, ktorý je v programe použitý, potom nedokáže spracovať takéto HTML a používateľ sa nemôže prihlásiť.

Riešenie - Dočasne sme tento problém vyriešili tak, že pri každom prihlásení sa so zlou šablónou vypisujeme chybovú hlášku. V ďalšom šprinte si dávame za cieľ nahradiť *DOMParser* menej striktným parserom.

Testovanie - Otestované boli 3 šablóny s nesprávnym formátom, pri ktorých bola úspešne vypísaná chybová hláška. So správnou šablónou sa používateľ prihlási stále bez problémov. Test prebiehal na mobile HTC Desire 500 s Androidom verzie 4.1.2.

5.1.2 Kompletne prerobiť interakciu s AIS

VFIIT – 36, Veronika Olešová, Filip Mazán, Filip Šoltés

Analýza - Táto úloha nadväzuje na úlohu "VFIIT-14 - Pozrieť sa na šablóny AIS, prečo parser zlyháva?", kde sme sa rozhodli pre nahradenie *DOMParsera* menej striktným parserom a zároveň na chybu "VFIIT-25 Oprava webovej verzie aplikácie kvôli prihlasovaniu do AIS". Príčinou tejto chyby bola reštriktívna politika CORS (*Cross-origin resource sharing*), ktorá nám nedovoľovala z moderných prehliadačov pristupovať priamo k AIS. Bolo potrebné nájsť spôsob, ako obísť CORS ochranu a zároveň sa vysporiadať so šablónami, ktoré tvoria nevalidné HTML prvky - tieto prvky odmietal doterajší *DOMParser* spracovať.

Návrh - Navrhnuté bolo riešenie s použitím nášho servera ako proxy medzi aplikáciou a AIS.

Riešenie - Vyvinuli sme teda serverový skript, ktorému pošleme šifrovane cez protokol https prihlasovacie údaje. Tento skript cez knižnicu cURL komunikuje s AIS a sťahuje potrebné stránky. Tieto sú potom spracovávané štandardným parserom zabudovaným v jazyku PHP, ktorý dovoľuje aj nestriktné dokumenty. Na výstupe zo skriptu sú už spracované údaje z AIS, konkrétne ID študenta a jeho rozvrh. Klient stiahne tieto dáta vo formáte JSON a používa ich ako doteraz.

Testovanie - Aplikáciu sme počas implementácie ladili pomocou webového rozhrania, kde sme ju potom aj testovali. Bolo testované správne zobrazovanie rozvrhov či už počas prebiehajúceho alebo skúškového obdobia. Naše riešenie bolo testované aj na mobilných telefónoch v ladiacom

móde, kedy fungovalo všetko podľa očakávaní. Riešenie však nemožno nasadiť skôr, ako budeme mať na našom serveri validný a overený certifikát.

***Používateľský príbeh**: Študent si požičia kamarátov mobil, aby si mohol pozrieť svoj rozvrh.*

5.1.3 Opravenie prihlásenia do AIS - treba mazať pamäť cache

VFIIT-19, Michal Kučera

Analýza - Používateľ po prihlásení mohol vidieť rozvrh predchádzajúceho používateľa. Rozvrh sa po prihlásení ukladá do pamäte *localStorage*, odkiaľ sa pravdepodobne po odhlásení/prihlásení nevymazával.

Návrh - Vymazať rozvrh z pamäte *localStorage* vždy pri odhlásení používateľa.

Riešenie - Do metódy *logout()* v triede *ais,* bolo pridané mazanie rozvrhu z pamäte *localStorage* a nastavenie aktuálneho rozvrhu na *null.*

Testovanie – Mazanie starého rozvrhu z localStorage bolo testované na zariadení Xperia Z s Androidom 4.4.4 pomocou aplikácie *WeinRe*. Nasimulovanie reálnej situácie si vyžadovalo použitie dvoch AIS účtov a preto testovanie prebehlo na viacerých zariadeniach ako aj na viacerých účtoch a všetko fungovalo podľa očakávania.

***Používateľský príbeh**: Študentovi chýba v rozvrhu prehľad celého týždňa v rámci jednej obrazovky, aby sa vedel lepšie zorientovať.*

5.1.4 Prerobenie rozvrhu

VFIIT-24, VFIIT-51, Daniel Pribul, Michal Kučera

Analýza – Pôvodný rozvrh sa študentom javil ako neprehľadný. Všetky položky rozvrhu boli zobrazené pod sebou, pričom na obrazovku sa nezmestili viac ako 2 naraz. Chýbal pohľad na celý týždeň.

Návrh – Navrhli sme pridať ďalší pohľad na rozvrh, ktorý by zobrazoval celý týždeň a bolo by možné si ho ľubovoľne približovať (Obr. 5 – *b3*). Pri takomto pohľade mali byť v rozvrhu použité iba skratky predmetov. Po kliknutí na skratku by sa mal zobraziť detail danej položky

rozvrhu (Obr. 5 – *b4*), ktorý by už obsahoval všetky potrebné údaje a odkiaľ by sa dalo prekliknúť na miestnosť, vyučujúceho alebo predmet, tak ako v pôvodnej verzii. Pôvodná obrazovka mala zostať zachovaná, pričom by na nej boli zobrazené iba položky rozvrhu aktuálne zvoleného dňa (Obr. 5 – *b2*). Menu s dňami zostalo zachované, ale zmenilo sa na vertikálne. Navrchu pribudlo tlačidlo na zmenu pohľadu.

Riešenie – Bol vypracovaný grafický návrh, ktorý bol prezentovaný zvyšným členom tímu. Implementácia potom pokračovala podľa návrhu, ale problém nastal s ľubovoľným približovaním, ktoré sa nám ani po dlhej dobe nepodarilo vyriešiť. Nakoniec sme to vyriešili iba 1-stupňovým približovaním cez tlačidlo v hornej lište. Bola upravovaná trieda *timetableView.ts* a pridané 2 nové šablóny *timetable-extended.xjade* a *detail.xjade*. Zmeny boli robené tak, aby sa prejavili aj v rozvrhoch profilov vyučujúcich, miestností a predmetov. Kvôli zlému časovému odhadu a neočakávaným nástrahám zo strany použitých technológií sa nám nepodarilo dokončiť úlohu za jeden šprint a musela byť dokončená v tom nasledujúcom.

Testovanie – Testovanie úspešne prebehlo na rôznych zariadeniach s operačným systémom Android od verzie 2.3.3 po 4.4.4.

***Používateľský príbeh**: Používateľ si chce aj v novej aplikácii prezerať svoj osobný rozvrh.*

5.1.5 Implementovanie prihlasovania a rozvrhov

VFIIT-98, Veronika Olešová

Analýza – Zobrazovanie rozvrhu používateľom je dôležitá súčasť aplikácie. Preto nesmie chýbať ani v novej implementácii.

Návrh – Rozvrh bude v prehľadnej tabuľke ako sme to spravili aj v starej aplikácii. Po kliknutí na cvičenie/prednášku sa zobrazí detailná obrazovka o tomto kurze, kde sa používateľ bude môcť ďalej prekliknúť na celkový predmet alebo na učiteľa učiaceho tento kurz. Obrazovka s informáciami o celkovom predmete bude obsahovať rozvrh všetkých cvičení a prednášok. Obrazovka učiteľa bude zase obsahovať jeho kontaktné údaje a osobný rozvrh usporiadaný podľa dní. Zobrazovanie ďalšieho zoznamu predmetov, ktoré učí, sa nám zdalo zbytočné. Používateľ sa bude môcť odhlásiť pomocou tlačidla v ľavom menu.

Riešenie – Všetky potrebné informácie z úložiska získavame prostredníctvom fabriky *ais.js*. Takisto tu pomocou funkcie *getTimetable* asynchrónne získavame informácie o rozvrhu aktuálneho používateľa vo formáte JSON, ktorý sa získa POST požiadavkou na server. Zaslaním mena a hesla používateľa nám server vráti JSON s jeho rozvrhom.

Fakt, či je používateľ "prihlásený" do systému, sa zisťuje na základe toho, či sa v úložisku nachádza kľúč *ais_login* a *ais_timetable*. Odhlásenie používateľa je teda veľmi jednoduché - stačí vymazať tieto kľúče.

Používateľov tabuľkový rozvrh je v controlleri timetable.js reprezentovaný ako dvojrozmerné pole, kde stĺpce predstavujú časy po jednej hodine a riadky zasa dni. Vo výhľade *timetable.html* je potom možné vykresľovať tabuľku pomocou direktívy *ng-repeat* nasledovným spôsobom:

```
<tbody>
<tr ng-repeat="day in ctrl.timetable">
<th>{{ctrl.borders.days[$index]}}</th>
                <td ng-repeat="course in day | filterDay"
colspan={{course.duration}}>
<div>{{ctrl.getRoomNumber(course.room_id)}}</div>
</tbody>
```

, kde *ctrl.timetable* je spomínané pole, ktoré sa vo vonkajšom cykle prechádza po riadkoch (dni) a vo vnútornom po stĺpcoch (hodiny).

Približovanie a posúvanie tabuľkového rozvrhu je umožnené vďaka direktíve *ion-scroll*. Táto direktíva potrebuje k správnemu fungovaniu poznať výšku a šírku jej obsahu. Kvôli nedefinovanej výške sme mali problémy s vertikálnym posúvaním tabuľky, ktorý sme dočasne vyriešili dynamickým priradením rozmerov na základe veľkosti obrazovky zariadenia.

Na obrazovke zobrazujúcej informácie o učiteľovi sa nachádza aj jeho osobný rozvrh, v ktorom sme museli zoskupiť predmety podľa dní pomocou vlastného filtra. Funkcia na získavanie rozvrhu aktuálneho učiteľa je implementovaná vo fabrike *ais.js*. Podobne je vyriešený rozvrh predmetu (cvičenie a prednášky doň spadajúce), kde tiež zoskupujeme kurzy podľa dní.

Kvôli ľahšiemu presmerovaniu z tabuľkového rozvrhu alebo učiteľského rozvrhu na URL *courses/id* sme sa rozhodli generovať si vlastné ID pre každé cvičenie a prednášku. Tieto ID sú v tvare deň_hodina_id-kurzu_id-učiteľa. Tento krok je nevyhnutný kvôli tomu, že cvičenia ani prednášky nemajú vlastné id v AISe.

5.1.6 Navrhnutie serverovej časti modulu AIS

VFIIT-95, Filip Šoltés

Návrh – Pre serverovú časť AIS modulu sme sa rozhodli zachovať rovnakú logiku aj formát dát ako v staršej verzii. Zmena v návrhu sa týka len programovacieho jazyka, dodržania architektonických konvencií a dôkladnejšieho ošetrenia vstupov.

Okrem existujúcej funkcionality sme sa rozhodli nachystať serverovú časť pre novú vlastnosť v aplikácii - zobrazovanie počtu neprečítaných mailov.

Riešenie – Pri implementácii sme využili testom riadený vývoj, pričom pokrytie modulu testami je > 90%.

Po HTTP dopyte na definovanú URL vráti serverová časť dáta vo formáte JSON. Parametre sú predávané pomocou metódy POST. Pri AIS module nevyužívame cache, ale rozvrh je vždy vyparsovaný priamo z AIS. Pred dopytom do AIS je vždy uskutočnené, pričom *cookie* je uložená pomocou vlastnosti použitej knižnice. S uloženou *cookie* potom vykonávame ďalšie dopyty.

Pre parsovanie sme vybrali knižnicu, ktorá dokáže spracovať aj nevalidné HTML, keďže nevalidnosť niektorých AIS tém bol problém v staršej verzii aplikácie.

Testovanie – Testovanie sme vykonali pomocou aplikácie *Postman* a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.1.7 Prepisovanie cache rozvrhu

VFIIT-152, Filip Mazán, Filip Šoltés

Analýza – Hlavne na začiatku semestra sa stáva situácia, kedy sa osobný rozvrh používateľa zmení, a tým pádom sa stáva v aplikácii neaktuálny.

Návrh – Navrhli sme nové serverové volanie *isTimetableRecent*, pomocou ktorého zistíme, či má aplikácia požiadať používateľa o znovu prihlásenie. Týmto sa používateľovi aktualizuje jeho osobný rozvrh.

Riešenie – Funkcionalita bola pridaná na strane servera i klienta.

Testovanie – Navrhnutá funkcionalita bola implementovaná a úspešne otestovaná.

5.1.8 Pridanie učiteľov do kancelárii

VFIIT-195, Veronika Olešová

Analýza – V pôvodnej verzii aplikácie boli pre každú miestnosť, ktorá je kanceláriou, zobrazení prislúchajúci vyučujúci. V novej verzii sme na to zabudli, preto je potrebné túto funkcionalitu doimplementovať.

Riešenie – Vo fabrike *rooms.js* si z miestnej pamäte (*localStorage*) nájdeme všetkých vyučujúcich pre danú miestnosť, ktorých mená potom vypíšeme pomocou direktívy *ng-repeat* v *room.html*.

Testovanie – Zobrazovanie vyučujúcich k danej kancelárii bolo otestované na všetkých našich zariadeniach a fungovalo bez problémov.

5.2 Jedálne

Pôvodná verzia: Tento modul sprístupňuje študentom aktuálnu ponuku jedální v okolí školy. Jedálne lístky sa získavajú pomocou API zo serveru *hladnystudent.sk*. Získané dáta sa ukladajú do *localStorage*, aby boli dostupné aj bez internetu.

Naše zmeny v pôvodnej verzii: Opravili sme viacero chýb, ktoré znemožňovali používateľom korektne zobrazovať jedálne lístky alebo zobrazovali neaktuálne údaje. Jedálny lístok sa v niektoré dni na zariadeniach s menším displejom nedal posúvať. Počas stretnutia sa prišlo aj na chybu, kedy sa na rôznych zariadeniach zobrazoval rozdielny jedálny lístok.

Zobrazenie jedálnych lístkov je jedna z najobľúbenejších funkcií aplikácie a denne sa na ňu spolieha množstvo študentov. Preto by mala fungovať korektne.

Nová verzia: Aj v tomto module ostala funkcionalita nezmenená a všetku logiku vrátane API volaní sme presunuli na server. Okrem cachovania na klientovi sme implementovali aj cachovanie na serverovej strane.

***Používateľský príbeh**: Študent je hladný a chce si pozrieť jedálny lístok.*

5.2.1 Oprava chyby - Zobrazovanie obedov

VFIIT-20, Filip Mazán, Jozef Karas

Analýza - Na niektorých mobiloch (napríklad Huawei Y300) sa vyskytla chyba v zobrazovaní obedov. Chyba bola v tom, že miesto toho, aby si používateľ mohol zrolovať (zhora nadol) vygenerovanú stránku a pozrieť všetky obedy dostupné na daný deň v danej jedálni, mohol len rolovať sprava doľava a tým videl len pár obedov.

Riešenie - Našli sme chybu, ktorá bola vo vykresľovaní obedov. Bolo ubratých cca 20px vo vykresľovaní. Kvôli chybnému rozkladaniu vznikali dlhé slová. Preto za každú čiarku a bodku treba doplniť medzeru.

Testovanie - Zmeny boli nasadené bez problémov pričom testovanie úspešne prebehlo na zariadení Huavei Y300 s verziou Androidu 4.1.1.

5.2.2 Obedy - oprava pamäte cache

VFIIT – 49, Jozef Karas

Analýza - Zistili sme, že niekedy obedy v aplikácií Virtuálna FIIT nesedia s obedmi na www.hladnystudent.sk, odkiaľ ťaháme informácie. Chyba bude najskôr v zlom mazaní *cache.*

Riešenie - Upravili sme zdrojový kód tak, že v súbore hladnystudent.js bol vo funkcií vymazania starých dát zmenený jeden riadok tak, aby pred načítaním dát - obedov z internetu vymazal všetky predchádzajúce informácie.

Testovanie - Oprava bola úspešne odskúšaná na všetkých našich Android zariadeniach v tíme.

***Používateľský príbeh:** Používateľ si chce aj v novej aplikácii prezerať jedálne lístky najbližších jedální.*

5.2.3 Spraviť stránku s obedmi, zatiaľ po dizajnovej stránke netreba

VFIIT-73, Filip Šoltés

Analýza – V rámci prerábania celej aplikácie je potrebné prerobiť aj obrazovku, na ktorej je možné zobraziť jedálny lístok bufetov a jedální v blízkosti školy. Je tu možné vybrať si konkrétny deň a konkrétnu jedáleň, pričom na obrazovke za zobrazí zoznam s názvami jedál, ich cenami, alergénmi a pod.

Návrh – Keďže naším cieľom je čo najviac odľahčiť klientskú stranu aplikácie, rozhodli sme sa API volania presunúť na serverovú časť. V aplikácii bude vytvorený ovládač *CantineController*, ktorý bude vykonávať POST volania na serverovú časť a tak získavať dáta. Dáta budú ukladané do lokálneho úložiska, aby boli prístupné aj bez internetového pripojenia.

Riešenie – Obrazovka Jedálne bola pridaná podľa návrhu

Testovanie – Obrazovka bola otestovaná na všetkých zariadeniach, ktoré vlastníme. Bolo testované postupné prepínanie dní a jednotlivých jedální.

5.2.4 Vytvorenie funkcie zobrazovania obedov podľa nového návrhu

VFIIT-92, Filip Mazán

Analýza – Jednou z funkcionalít aplikácie je zobrazovanie jedální a obedov v nich. Je teda nutné naimplementovať tieto funkcie do novej aplikácie.

Návrh – Funkcionalita zahŕňa obrazovky:

- zobrazenie jedální
- zobrazenie obedov
- zobrazenie podrobností jedálne

Naimplementované by malo byť aj lokálne ukladanie do pamäti. Dáta sú sťahované cez serverové rozhranie.

Riešenie – Myšlienky z návrhu boli implementované a úspešne otestované.

5.2.5 Navrhnutie serverovej časti pre obedy

VFIIT-94, Filip Šoltés

Analýza – Keďže sme sa v novej verzie aplikácie rozhodli zmeniť architektúru, bolo potrebné vytvoriť komplexnejšie riešenie serverovej časti. Kvôli jednotnosti zdrojového kódu sme sa rozhodli pre implementáciu v *node.js*, pričom sme vybrali rámec *express.js*. V novej verzii aplikácie sme sa rozhodli funkciu vyhľadávania samozrejme ponechať.

Návrh – Tak ako v starej verzii aplikácie voláme API poskytnuté od portálu *hladnystudent.zones.sk*. Funkcionalita by mala ostať nezmenená, kvôli vyššej robustnosti aplikácie je však potrebné dôkladné ošetrenie vstupov a taktiež implementácia cache na serverovej strane.

Riešenie – Keďže serverová časť pre jedálne bola implementovaná ako prvá v serverovej časti aplikácie, pri implementácii sme urobili najviac návrhových a architektonických riešení. Celá aplikácia pracuje viacvrstvovo a využívame hlavne návrhový vzor fasáda.

Po HTTP dopyte na definovanú URL vráti serverová časť dáva vo formáte JSON. Parametre sú predávané pomocou metódy POST.

Pri implementácii sme využili testom riadený vývoj, pričom pokrytie modulu testami je 100%.

5.2.6 Cachovanie obedov na serverovej strane

VFIIT-134, Veronika Olešová

Analýza – Doposiaľ bolo implementované cachovanie obedov len na klientskej strane. Aby sme predišli opakovanému dopytovaniu na portál *hladnystudent.zones.sk* so žiadosťou o rovnaké dáta počas istého intervalu, chceme umožniť cachovanie aj na serveri.

Riešenie – Keďže na cachovanie na serveri používame *key-value* databázu *Redis*, aj obedy budú uložené v tejto databáze. Hlavnú funkcionalitu predstavuje metóda *loadOrFetchJson* v súbore *common/cache.js*, ktorá sa na základe existencie prijatého kľúča rozhodne či vráti dáta prislúchajúce danému kľúču alebo zavolá funkciu na stiahnutie nových dát z hladného študenta v prípade, že takýto kľúč neexistuje. Samotný kľúč obedov je určený pomocou id jedálne a dňa,

pre ktorý je jedálny lístok určený: `'canteens.menu.' + params.canteenId + params.day.replace(/-/g, '')`.

Testovanie – Správnosť cachovania bola overená pomocou grafického rozhrania databázy *Redis*.

5.3 MHD

Pôvodná verzia: Tento modul sprístupňuje študentom odchody spojov MHD zo zastávok z okolia školy. Cestovné poriadky sa získavajú pomocou API zo serveru *itransit.sk*. Získané dáta sa ukladajú do *localStorage*, aby boli dostupné aj bez internetu.

Naše zmeny v pôvodnej verzii: Na tomto module sme žiadne zmeny nevykonali.

Nová verzia: Funkcionalita modulu MHD ostala nezmenená a API volania na *itransit* sa premiestnili na server.

Používateľský príbeh: *Študent si chce aj v novej aplikácii prezrieť presné časy odchodov MHD, aby sa vedel pohodlne dostať na miesto, kde potrebuje.*

5.3.1 Navrhnutie serverovej časti pre MHD

VFIIT-96, Filip Šoltés

Návrh – Kvôli zmene architektúry celej aplikácie sme museli markantne prerobiť návrh MHD modulu. Chceli sme, aby klient nemusel vykonávať žiadnu agregáciu, ale dostal len dáta, ktoré priamo zobrazí.

Server vykoná potrebné dopyty na API *itransit* a vytvorí objekt vo formáte, ktorý netreba ďalej upravovať.

Rozhodli sme sa zachovať načítavanie týždňa dopredu.

Riešenie – Pre funkcionality sme implementovali samostatný modul, ktorý načítava potrebné dáta z API a agreguje ich do samostatného objektu. Tento objekt obsahuje pole zastávok, pre ktoré zobrazujeme spoje, pričom každá zastávka obsahuje zoznam spojov s odchodmi a konečnými stanicami. Každý spoj je unikátne identifikovaný.

V globálnom konfiguračnom súbore pre MHD modul sú definované skupiny tak, ako majú byť zobrazené, pričom pre zaradenie spojov do skupín sú využité unikátne identifikátory poslané zo servera pri každom spoji.

V MHD môže nastať stav, že jeden spoj má počas dňa premenlivú poslednú zastávku. Tento stav sme vyriešili tak, že ak má spoj v konkrétnom čase inú poslednú zastávku ako zvyčajne, obsahuje objekt pre konkrétny čas atribút s poslednou zastávkou.

Agregované dáta ukladáme do cache, pričom pri každom dopyte od klienta server odosiela cestovný poriadok na týždeň dopredu.

Testovanie – Testovanie sme vykonali pomocou aplikácie *Postman* a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.3.2 Implementácia klientskej časti pre MHD

VFIIT-105, Daniel Pribul

Analýza – Ďalšou funkcionalitou, ktorú sme sa rozhodli reimplementovať aj do novej verzie aplikácie je zobrazovanie odchodov autobusov.

Návrh – Funkcionalita pozostáva z dvoch obrazoviek:

- Zobrazenie najbližších odchodov autobusov, rozdelených podľa zastávky, smeru cesty autobusu a čísla autobusu. Obrazovka zahŕňa kategóriu obľúbených spojov.
- Zobrazenie rozvrhu odchodu vybranej linky. Na tejto obrazovke je možné pridať linku medzi obľúbené.

Riešenie – Obrazovky boli implementované podľa návrhu.

5.3.3 Prerobiť linky v MHD

VFIIT-203, Daniel Pribul

Analýza – V MHD obrazovke po návrate z detailu spoju chceme, aby záložky zastávok ostali rozkliknuté tak, ako keď sme klikli na detail.

Návrh – Umožniť zapamätanie stavu obrazovky, resp. pomocou parametra v URL rozkliknúť príslušnú zastávku.

Riešenie – Rozkliknutie príslušnej zástavky je zabezpečené pomocou parametra stavu. Tento parameter obsahuje JSON zoznamu zastávok s hodnotami true a false, ktoré značia, či je príslušná zastávka rozkliknutá.

Na zabezpečenie pamätania si rozkliknutých zastávok po kliknutí na detail spoju sme aktualizovali Ionic na verziu rc5, ktorá má implementované cachovanie predchádzajúcich stránok.

Testovanie – Riešenie bolo otestované na všetkých zariadeniach.

5.4 Vyhľadávanie

Pôvodná verzia: Vyhľadávanie miestností, učiteľov a predmetov.

Naše zmeny v pôvodnej verzii: Na tomto module sme žiadne zmeny nevykonali.

Nová verzia: Funkcionalita tohto modulu ostala nezmenená, zmenil sa len vzhľad vyhľadávacieho poľa.

Používateľský príbeh: *Študent je zvyknutý na rýchle vyhľadávanie z hlavnej obrazovky, čo by uvítal aj v novej aplikácii.*

5.4.1 Implementovanie vyhľadávania z hlavnej obrazovky

VFIIT-101, Michal Kučera

Analýza – V novej verzii aplikácie sme sa rozhodli funkciu vyhľadávania samozrejme ponechať. Samotné vyhľadávanie v starej aplikácii fungovalo spoľahlivo a rýchlo. Zmeniť sme sa rozhodli iba jeho vzhľad a umiestnenie tak aby pasovalo k novému dizajnu aplikácie. V starej aplikácii bolo dostupné cez menšie tlačidlo v hornej lište a cez väčšie, ktoré bolo umiestnené na úvodnej obrazovke. Väčšie tlačidlo sme sa rozhodli nahradiť rovno samotným vyhľadávacím polom. Týmto by sa malo vyhľadávanie stať ešte dostupnejším a malo byt pomôcť používateľom rýchlejšie sa dostať k potrebným informáciám.

Návrh – Pri vyhľadávaní v starej aplikácii sa pojmy hľadali v súbore *data/search.json*. Ten obsahoval pole objektov s nasledujúcimi atribútmi: *key, name, type* a *id*. Toto pole vznikalo na serveri, kde sa spojili dáta o miestnostiach, ľuďoch, zástavkách a predmetoch a predspracovali sa do potrebnej podoby. Toto sme sa rozhodli v novej aplikácii ponechať, s tým rozdielom, že pole s dátami bude uložené v *localStorage* a bude sa aktualizovať podľa potreby spolu s ostatnými dátami a nie iba pri novej verzii aplikácie.

Pri spustení vyhľadávania sa otvorí *modal box*, ktorý bude obsahovať vyhľadávacie pole, zoznam výsledkov a pozadie bude tmavé a čiastočne priehľadné. Zadaný text sa bude dať zmazať a výsledky budú označené ikonkou podľa typu výsledku (osoba, predmet, ...). *Modal box* sa bude skrývať po zavretí klávesnice. Po novom budú navyše pod vyhľadávacím polom na úvodnej obrazovke umiestnené skratky na posledné hľadané výrazy. Po stlačení presmerujú používateľa priamo na hľadanú stránku.

Riešenie – Bol vytvorený *modal box* zo šablóny *search.html*. Logika vyhľadávania bola umiestnená vo *factory/search.js*. Problém pri riešení nastal s vyhľadávacím polom na úvodnej obrazovke. Samotné pole je totiž vhodné umiestniť čo najvyššie, aby pod ním zostal priestor na výsledky aj pri vysunutej klávesnici. Pole z úvodnej obrazovky sa ale nedalo rozumne posunúť vyššie nad lištu bez toho, aby sa presunulo v DOM z úvodnej obrazovky do *modal boxu* (a potom naspať). *Modal box* preto vždy obsahuje vlastné vyhľadávacie pole a to z úvodnej obrazovky sa jednoducho počas vysúvania *modal boxu* (zospodu) posunie smerom hore "do stratena" a *focus* sa prepne na pole v *modal boxe*. Zvyšok funkcionality bol implementovaný bez väčších problémov podľa navrhnutého riešenia a grafického návrhu.

Testovanie – Fungovalo na všetkých zariadeniach podľa očakávania.

5.5 Mapy

Pôvodná verzia: Aplikácia obsahuje mapu školy a mapu okolia. Mapy sú vo formáte SVG, čo umožňuje interakciu používateľa s objektami na mape. Na viaceré objekty (miestnosti, jedálne, zastávky) je možné kliknúť alebo ich podľa potreby farebne vyznačiť (miestnosti). Mapy je možné približovať.

Naše zmeny v pôvodnej verzii: V pôvodnej verzii sme žiadne zmeny nevykonali.

Nová verzia: Modul bol vytvorený nanovo ale pôvodná funkcionalita bola ponechaná. Mapy budovy ostali nezmenené, vymenili sme len mapu okolia za aktuálnejšiu a detailnejšiu. Táto mapa predstavuje stále SVG obrázok. Obe mapy sú interaktívne tak, ako to bolo v pôvodnej aplikácii.

Používateľský príbeh: *Študent chce mať prehľad o miestnostiach v budove aj prostredníctvom novej aplikácie.*

5.5.1 Spraviť stránku s mapou budovy, zatiaľ SVG formát + tlačidlá poschodí

VFIIT-74, Veronika Olešová

Analýza – Pôvodnú funkcionalitu máp chceme postupne sprístupniť aj v novej aplikácii. Zo začiatku bude stačiť, ak sa po kliknutí na tlačidlo poschodia zobrazí prislúchajúca mapa v SVG formáte. Nie je potrebné implementovať približovanie mapy ani kliknutie na miestnosť.

Riešenie – Tlačidlá sú zatiaľ prevzaté z CSS komponentov rámca *Ionic*:

```
<div class="button-bar-inline">
```

Ich dizajn bude treba navrhnúť tak, aby boli všetky prehľadne rozmiestnené.

Na načítanie SVG obrázka sme nakoniec použili *ng-include*, ktorý vloží všetky informácie z tohto súboru do html:

```
<ng-include src="ctrl.getFloorSource()"></ng-include>
```

Skúšali sme aj iné načítavanie obrázkov akým je napríklad použitie elementu <*object*>, pri ktorom sa vyskytli problémy na telefónoch (na webovej stránke to nevykazovalo žiadne problémy). Problém bol úspešne odstránený avšak použitie *ng-include* sa nám zdá ako najspoľahlivejšie riešenie.

Testovanie – Zobrazovanie máp sa správa na všetkých našich mobiloch podľa očakávaní.

5.5.2 Urezané zobrazovanie pri približovaní mapy

VFIIT-175, Michal Kučera

Analýza – Na niektorých zariadeniach sa vyskytol problém s mapami, kedy mapa budovy bola zobrazená iba v hornej polovici obrazovky. Rozšírením aplikácie o modul *Crosswalk* sa tento problém rozšíril na všetky zariadenia. Mapa je zabalená v elemente *ion-scroll*, ktorý sa stará o približovanie a posúvanie.

Návrh – Upraviť výšku mapy pomocou CSS alebo v prípade potreby ju prepočítavať cez JavaScript po vzore mapy okolia.

Riešenie – Po zväčšení výšky *ion-scroll* na 100% sa kontajner s mapu roztiahol na celú obrazovku, avšak v prípade, že samotné SVG bolo na výšku menšie ako kontajner, SVG zostalo na vrchu kontajnera. Vycentrovať na výšku sa nám podarilo už iba úpravou všetkých SVG, ktorým bolo treba nastaviť atribúty *width* a *height* taktiež na 100%.

Testovanie – Mapy boli úspešne otestované na viacerých zariadeniach s rôznymi veľkosťami displejov (4” a viac) a rôznou verziou operačného systému (Android 4.0 a viac).

Používateľský príbeh: *Študent by v novej aplikácii uvítal detailnejšiu a aktuálnejšiu mapu okolia.*

5.5.3 Implementovanie mapy okolia

VFIIT-146, Veronika Olešová

Analýza – Aj v novej aplikácii chceme mať možnosť prezerať si mapu okolia budovy FIIT. V pôvodnej verzii predstavoval mapu okolia len jednoduchý obrázok SVG. Naším cieľom je vynoviť toto SVG alebo použiť *OpenStreetMap* API.

Návrh – Medzi naše požiadavky na mapu okolia patrí aj možnosť používania máp offline. Kvôli tomu by bolo potrebné pri *OpenStreetMap* API implementovať cachovanie, čo by bolo implementačne a pamäťovo náročné. Jednoduchším riešením by teda bolo použitie vygenerovaného obrázka SVG z *OpenStreetMap*.

Ďalšou požiadavkou je interaktivita s touto mapou - po kliknutí na jedálne Eat and Meat, Venza a jedáleň s bufetom FEI a FIIT bude používateľ presmerovaný na aktuálny jedálny lístok danej jedálne. Taktiež sa môže používateľ dostať na rozpis odchodov autobusov a električiek po kliknutí na jednu zo zastávok Zoo alebo Botanická záhrada.

Riešenie – Najprv sme rozdelili obrazovku máp na dve časti - mapa budovy a mapa okolia. Potom sme sa pokúsili priamo vygenerovať SVG mapu z *OpenStreetMap*, ktorá bola však príliš veľká - dosahovala až 5mb. Príčinou bolo príliš detailné zmapovanie zvoleného územia a tým pádom obsahovalo aj príliš veľa vektorov.

Alternatívne riešenie predstavoval export *OpenStreetMap* dát a ich následná úprava. Program na úpravu takýchto dát sa volá JOSM, pomocou ktorého sme odstránili prebytočné informácie v mape a pridali niektoré chýbajúce. Tento program pracuje výlučne s OSM dátami a umožňuje jednoduchú úpravu dát v grafickom rozhraní:

Obr. 12 Grafické rozhranie programu JOSM

Z OSM dát je potom potrebné vygenerovať SVG obrázok, na čo sme použili program *Maperitive*. Takto vygenerované SVG má veľkosť už len 250kb oproti 5mb.

Do mapy sme nad jedálne a MHD pridali neviditeľné obdĺžniky, vďaka čomu sa používateľ pohodlne preklikne na zodpovedajúcu obrazovku. Text klikatelných jedální a MHD sme odlíšili farebne aj veľkosťou od neklikatelných objektov.

SVG mapa je v html zahrnutá nasledujúcim spôsobom:

```
<ng-include ng-click="handleClick($event)" src="getMapSource()"></ng-include>
```

Po kliknutí na túto mapu sa zavolá funkcia *handleClick* s parametrom *$event*, ktorý predstavuje práve zakliknutý element. Obdĺžnik jedální obsahuje atribút *canteen* s id prislúchajúcej jedálne:

```
<rect x="988.4368286132812" y="305.4902648925781" width="144.1228485107422"
height="140.17428588867188" canteen="1"/>
```

MHD obsahuje naopak atribút *bus_stop*. Na základe týchto atribútov vieme zistiť, či používateľ klikol na jedálne (odkaz na konkrétnu jedáleň), MHD (odkaz na základnú obrazovku/transport) alebo niekde úplne inde.

Testovanie – Posúvanie a približovanie mapy bolo na slabších zariadeniach pomalšie. Odkazy na jedálne a MHD fungovali na všetkých zariadeniach.

5.5.4 Mapa okolia – zameranie pohľadu na FIIT

VFIIT-177, Veronika Olešová

Analýza – Pri prvom otvorení obrazovky mapy okolia je mapa zameraná na jej ľavý horný okraj. Viacerým sa toto zameranie nepáčilo a uvítali by, keby bol miesto toho pohľad na našu fakultu.

Riešenie – Presunutie pohľadu je možné pomocou služby *$ionicScrollDelegate*. Do direktívy *ion-scroll* v súbore *maps-outdoor.html* sme najprv pridali atribút *delegate-handle="focusFIIT"*. Službu *$ionicScrollDelegate* sme zavolali zo súboru *maps-outdoor.js* po úspešnom vložení SVG mapy okolia do HTML nasledovne:

```
$ionicScrollDelegate.$getByHandle('focusFIIT'). scrollTo(1750 -
$scope.getWidth() / 2, 2050 - $scope.getHeight() / 2);
```

Testovanie – Presunutie pohľadu na fakultu FIIT bolo otestované na všetkých našich zariadeniach a správalo sa podľa očakávaní.

5.5.5 Zobrazenie načítavacieho okna pri mapách okolia

VFIIT-199, Veronika Olešová

Analýza – Pri vstupovaní na obrazovku s mapou okolia aplikácia na chvíľu prestane reagovať. Je to kvôli tomu, že mapa okolia je príliš veľká a načítanie celého SVG je tým pádom pomalé.

Návrh – Bolo by vhodné používateľa informovať o procese načítavania tejto mapy prostredníctvom načítavacieho okna (*loading screen*).

Riešenie – Najprv sme načítavanie SVG obrázkov riešili jednoducho pomocou *ng-include*:

```
<ng-include ng-click="handleClick($event)" src="getMapSource()"></ng-include>
```

Neskôr sme na to našli knižnicu *SVGInjector* (https://github.com/iconic/SVGInjector), ktorej použitie je možné vidieť v súbore *maps-outdoor.js*.

Načítavacie okno zobrazujeme pred každým vstupom na obrazovku máp okolia pomocou sledovania udalosti *$ionicView.beforeEnter*. Blok, ktorý sa zavolá pri tejto udalosti, obsahuje tiež vloženie obrázku SVG do HTML pomocou spomínaného *SVGInjectora*. Načítavacie okno sa skryje po vložení obrázku.

Testovanie – Zistili sme, že načítavacie okno sa zobrazí len pri prvom otvorení mapy okolia. Tento problém sme neboli schopní vyriešiť a myslíme si, že Ionic nedokáže správne riešiť podobné situácie s veľkými SVG obrázkami.

5.6 QR Kódy

Pôvodná verzia: Aplikácia obsahuje čítačku QR kódov, ktorá po nasnímaní kódu zobrazí profil vyučujúceho. QR kódy obsahujú vizitku vo formáte *vCard* a mali by byť rozmiestnené po celej škole.

Naše zmeny v pôvodnej verzii: Na tomto module sme žiadne zmeny nevykonali.

Nová verzia: Do novej verzie sme QR kódy zatiaľ nezahrnuli.

5.7 BLE lokalizácia

Nový modul: Tento modul sa venuje navigácii používateľa v rámci budovy školy. Pozícia sa bude určovať na základe sily *bluetooth* signálu vysielaného z BLE zariadení rozmiestnených po škole. Aktuálny stav umožňuje určiť polohu na testovacej ploche – chodba na 3. poschodí školy.

***Používateľský príbeh**: Študent chce v aplikácii na mape budovy vidieť, kde sa práve nachádza, aby sa vedel rýchlejšie zorientovať.*
Tento používateľský príbeh sme sa rozhodli riešiť kvôli tomu, že patrí medzi naše hlavné ciele a chceli by sme našim spolužiakom pomôcť lepšie sa orientovať v budove našej školy.

5.7.1 Vytvorenie zásuvného modulu na detekciu a zobrazenie BLE zariadení

VFIIT-4, Filip Mazán, Veronika Olešová

Analýza – Keďže navigácia vnútri budovy pomocou GPS nefunguje, zvolili sme technológiu BLE Beacon. T.j. potrebujeme zistiť, ako cez *PhoneGap* dokážeme pracovať s BLE vysielačmi. *PhoneGap* neobsahuje žiadne použiteľné zásuvné moduly, ktoré by túto funkciu podporovali, treba spraviť vlastný.

Návrh – Vytvoriť natívny *Cordova* zásuvný modul na platformu Android, ktorý zabezpečuje skenovanie BLE zariadení. Zásuvný modul treba prepojiť s front-endom aplikácie Virtuálna FIIT, a teda zobrazovať údaje získané o zásuvného modulu.

Riešenie – Pridaný bol *BeaconPlugin*, ktorý vykonáva navrhovanú funkcionalitu. Do hlavného menu bola pridaná položka „Beacony“, ktorá na obrazovke zobrazí skenovaciu obrazovku, na ktorej sa zobrazujú nájdené BLE zariadenia.

Testovanie - Zmeny boli nasadené a otestované na dvoch zariadeniach, ktoré podporujú technológiu BLE. Testovanie nevykázalo žiadne chyby.

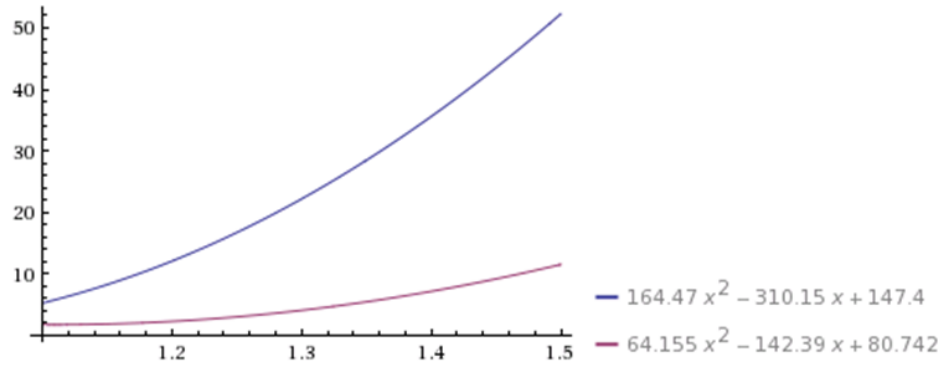
5.7.2 Vybudovať aplikáciu na zber, zozbierať dáta

VFIIT-72, Filip Mazán, Veronika Olešová

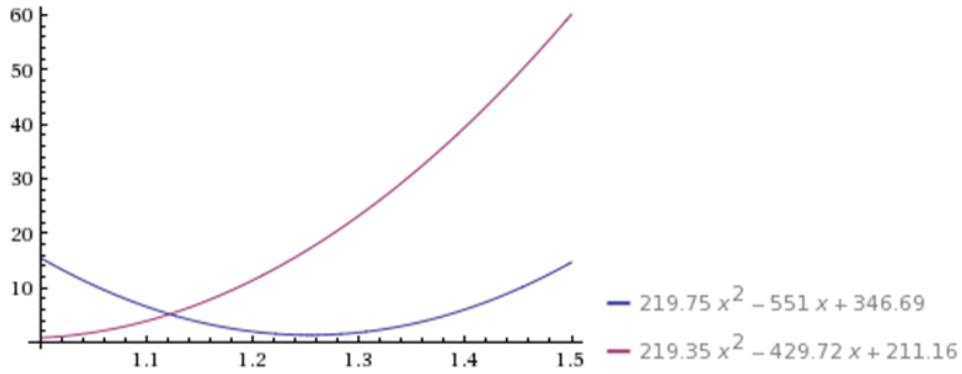
Analýza – Aby sme zistili, ako sa signál Bluetooth šíri v priestore, je nutné vykonať testovanie. Zo zistených dát budeme vedieť, ako signál so vzdialenosťou od vysielačov klesá a budeme schopní usúdiť, či má zmysel pokračovať v ceste trilaterácie alebo sa začať zaoberať metódami postavenými na *fingerprintingu* - digitálnych odtlačkoch priestoru, ako napr. neurónové siete, Bayesove filtre či časticové filtre.

Návrh – Ku zberu dát potrebujeme navrhnúť aplikáciu na OS Android, ktorá je schopná zachytávať silu signálu Bluetooth vysielaného z viacero vysielačov. Aplikácia musí vedieť zozbierané dáta poskytnúť v podobe určenej na ďalšie spracovanie (export do csv súboru). Aplikácia bude potom využitá na zbery v dvoch priestoroch - chodba a otvorený priestor s použitím dvoch typov vysielačov - na batérie a USB vysielač. Dáta budú vyhodnotené a bude nájdená čo najlepšia funkcia opisujúca pokles sily signálu so vzdialenosťou.

Riešenie – Vytvorili sme aplikáciu na zber dát, dáta sme zozbierali podľa návrhu a vyhodnotili.



Obr. 13 Meranie na chodbe, modrá - USB, červená - batéria. Os X - podiel sily signálu a sily vysielača, os Y - vzdialenosť



Obr. 14 Meranie v otvorenom priestore, modrá - USB, červená - batéria. Os X - podiel sily signálu a sily vysielača, os Y - vzdialenosť

Z grafov vidno, že spôsoby lokalizácie založené na odhade vzdialenosti od vysielača na základe sily signálu je veľmi nepresné a závisí nielen od typu vysielača, ale aj od priestoru, v ktorom je umiestnený. Preto má význam sa zaoberať metódami postavenými na inom princípe.

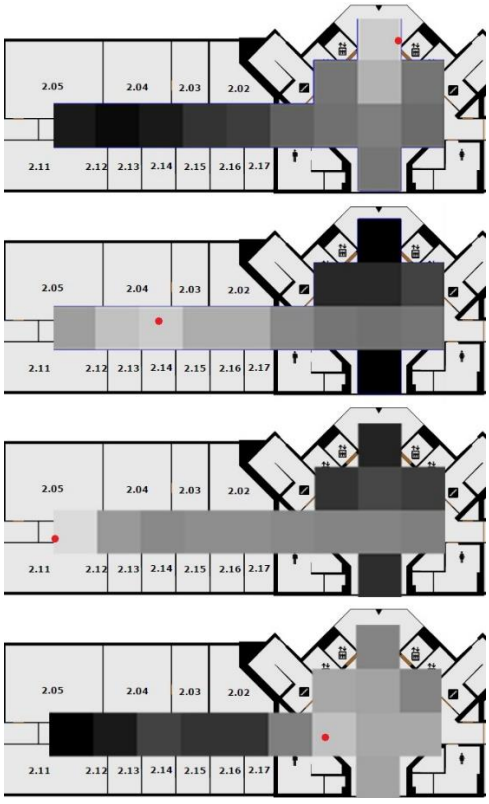
5.7.3 Spracovanie nazbieraných dát, vytvorenie vizualizácie

VFIIT-90, Filip Mazán

Analýza – Na základe predošlej analýzy beaconov sme sa rozhodli vydať cestou neurónových sietí. Na toto však potrebujeme vizualizovať podobu šírenia signálu z nameraných hodnôt. Cieľom je navrhnúť a implementovať skripty na spracovanie dát a ich vizualizáciu.

Návrh – Z nazbieraných surových dát je nutné vytvoriť dáta vhodné pre vizualizáciu a pre učiaci proces. Navrhli sme preto niekoľko skriptov v jazyku *Python*, ktoré túto úlohu plnia.

Riešenie – Vytvorili sme skripty spracovávajúce dáta a spustili ho na vzorke zozbieraných dát. Dáta zachytávajú 100 meraní zo 14 referenčných bodov v polkrídle jedného poschodia našej fakulty. Na chodbe boli umiestnené 4 beacony označené bodkou na obrázku. Intenzita signálu je vyjadrená jasnosťou farby - biela predstavuje najsilnejší signál, čierna najslabší.



Obr. 15 Intenzita signálu v polkrídle jedného poschodia našej fakulty

5.7.4 Vykresliť bodku na mape pre lokalizáciu

VFIIT-212, Filip Mazán

Analýza – Vychádzajúc z dlhodobého plánu bolo dôležité prezentovať stav vývoja lokalizácie vo vnútri budov. Doposiaľ bol síce vytvorený *BeaconPlugin* - natívny modul aplikácie ako knižnica, ale nebolo implementované samotné intuitívne zobrazenie lokácie používateľa.

Návrh – Navrhovaným riešením je prepojiť spomínanú natívnu knižnicu so samotnou mapou budovy.

Riešenie – Implementovali sme zobrazovanie ukazovateľa lokácie používateľa ako červený bod na mape budovy. Na podporovaných poschodiach fakulty sa v ladiacom režime zobrazí aj tlačidlo spúšťajúce lokalizačný proces. Následne sa bod vykreslí na mapu poschodia.

Testovanie – Riešenie bolo otestované na telefóne podporujúcom BLE technológiu dosahujúc priemernú presnosť určenia polohy 2 až 3 metre.

5.8 Knižnica

Nový modul: Táto obrazovka sa v aplikácii pred tým nenachádzala. Obsahuje úradné hodiny knižnice a odkaz na mapu.

Používateľský príbeh*: Študent chce ísť do knižnice, ale nechce tam ísť naprázdno. Hodila by sa mu informácia o úradných hodinách knižnice.*

5.8.1 Úradné hodiny knižnice

VFIIT-10, Michal Kučera, Jozef Karas

Analýza – V aplikácií sa v pôvodnom stave nedali nájsť úradné hodiny knižnice. Úradné hodiny študijného oddelenia sa už ale v aplikácii nachádzali. Odkaz v menu smeroval na miestnosť, pričom úradné hodiny boli vypísane v jej popise.

Návrh – Pridať informácie o knižnici po vzore študijného oddelenia.

Riešenie – Do bočného menu bola pridaná položka knižnica, ktorá po vzore študijného oddelenia odkazovala na stránku miestnosti (v tomto prípade miestnosti -1.43). Do súboru *rooms.json* sme k príslušnej miestnosti do atribútu *desc* pridali úradné hodiny knižnice. Aby sa pri stlačení tlačidla "zobraziť miestnosť" knižnica korektne vyznačila na mape, bolo potrebné do súboru *map/-1.svg* pridať atribút *path* pre túto miestnosť. Nakoniec sa vytvorila nová ikona pre študijné oddelenie, nakoľko stará bola vhodnejšia práve pre knižnicu.

Testovanie – Zmeny boli nasadené bez problémov pričom testovanie úspešne prebehlo na zariadeniach Xperia Z s verziou Androidu 4.4.4 a Huawei Y300 s verziou Androidu 4.1.1.

5.9 Študijné oddelenie

Pôvodná verzia: Táto jednoduchá obrazovka poskytovala informácie o úradných hodinách študijného oddelenia, možnosť zobrazenia miestnosti na mape a zoznam pracovníčok.

Naše zmeny v pôvodnej verzii: V pôvodnej verzii sme nevykonali žiadne zmeny.

Nová verzia: Tento modul bol vytvorený nanovo s pôvodnou funkcionalitou.

***Používateľský príbeh**: Študent si potrebuje vedieť rýchlo vyhľadať otváracie hodiny študijného oddelenia, jeho pracovníčky, prípadne si chce zobraziť túto miestnosť na mape.*

5.9.1 Pridať študijné oddelenie

VFIIT-166, Daniel Pribul

Analýza – Tak ako v predchádzajúcej verzii aplikácie, aj v novej chceme používateľom poskytnúť informácie o študijnom oddelení.

Návrh – Poskytované informácie budú rovnaké ako v predchádzajúcej verzii, t.j. číslo miestnosti, odkaz na zobrazenie na mape, otváracie hodiny, zoznam študijných poradcov s odkazom na detail poradcu.

Riešenie – Zobrazenie študijného oddelenia bolo pridané do bočného menu. Implementácia zodpovedá návrhu.

Testovanie – Navrhnutá funkcionalita bola úspešne otestovaná.

5.10 Harmonogram

Pôvodná verzia: Táto obrazovka obsahuje zoznam dôležitých dátumov na fakulte. Odkaz na túto obrazovku sa nachádza v bočnom menu.

Naše zmeny v pôvodnej verzii: Pridanie časovej osi a zmena vzhľadu.

Nová verzia: Tento modul bol vytvorený nanovo s pôvodnou funkcionalitou.

***Používateľský príbeh**: Študent chce prehľadnejší harmonogram, ktorý sa mu nebude zlievať, aby v ňom vedel rýchlejšie nájsť to, čo potrebuje.*

5.10.1 Zmeniť neprehľadný harmonogram

VFIIT-31, Jozef Karas, Michal Kučera

Analýza – Po spýtaní sa okoloidúcich spolužiakov sme zistili, že aktuálne zobrazenie harmonogramu je neprehľadné, pretože dátumy sa opticky zlievajú s udalosťami, čo výrazne

zhoršuje čitateľnosť. Inšpirovali sme sa už existujúcimi riešeniami na internete a rozhodli sa dátumy oddeliť od udalostí časovou osou.

Návrh – Vytvoriť prvotný grafický návrh, v rámci tímu ho prekonzultovať, aplikovať zmeny a týmto štýlom dopracovať ku konečnej podobe. Tá sa potom implementuje do aplikácie.

Riešenie – Do nového zobrazenia harmonogramu sme pridali časovú os modrej farby s bielymi bodkami s dátumami a akciami (Obr. 5 – *A2*). Pokiaľ bol dátum časové rozhranie (od - do), časová čiara medzi dvoma bodmi bola tiež biela.

Testovanie – Nová verzia harmonogramu bola úspešne otestovaná na všetkých našich Android zariadeniach v tíme.

***Používateľský príbeh**: Študent chce mať prehľad o dôležitých dátumoch na fakulte aj v novej aplikácii.*

5.10.2 Pridanie harmonogramu

VFIIT-127, Daniel Pribul

Analýza – Tak ako v predchádzajúcej verzii aplikácie, aj v novej chceme používateľom poskytnúť prehľad udalostí semestra.

Návrh – V predchádzajúcej verzii sa zobrazoval harmonogram pre celý rok, rozdelený na letný a zimný semester. V novej verzii navrhujeme zobrazovať všetky nadchádzajúce a tri uplynulé udalosti.

Riešenie – Zobrazenie harmonogramu bolo pridané do bočného menu. Implementácia zodpovedá návrhu.

Testovanie – Navrhnutá funkcionalita bola úspešne otestovaná

5.11 O aplikácii

Pôvodná verzia: Táto obrazovka obsahuje informácie o aplikácii, tímoch ktoré na nej pracovali a aktuálnej verzii.

Naše zmeny v pôvodnej verzii: Okrem aktualizácie údajov sme na tejto obrazovke nič nemenili.

Nová verzia: V novej verzii aplikácie bola táto obrazovka vytvorená nanovo.

Používateľský príbeh: *Študent chce prehľadnejšiu a použiteľnejšiu aplikáciu, aby sa mu s ňou lepšie pracovalo.*

5.11.1 Spraviť stránku O aplikácii s funkčnými linkami

VFIIT-70, Filip Mazán

Analýza – V rámci prerábania celej aplikácie je nutné vytvoriť obrazovky, ktoré sa nachádzajú v pôvodnej aplikácii. Jednou z nich je obrazovka O aplikácii. Táto obsahuje textové informácie, odkazy na predošlé tímy a linku na formulár pre odoslanie chyby.

Návrh – Vytvorený bude ovládač *AboutController*, v ktorom budú prehľadne zapísané predošlé tímy spolu s linkami na ich domovskú webovú stránku. Súbor zobrazenia vypíše statické textové informácie, predošlé tímy a linku na formulár. Linka na obrazovku O aplikácii bude pridaná do hlavného menu aplikácie.

Riešenie – Obrazovka *O aplikácii* bola pridaná podľa návrhu.

Testovanie – Obrazovka bola otestovaná na všetkých zariadeniach, ktoré vlastníme.

5.12 Nahlasovanie chýb

Pôvodná verzia: Táto obrazovka slúži na odosielanie informácii o chybe zo strany používateľov.

Naše zmeny v pôvodnej verzii: V starej verzii aplikácie sme na tejto obrazovke nič nemenili.

Nová verzia: Obrazovka bola spravená nanovo. Riešili sme takisto prípad spadnutia aplikácie a odosielanie metadát pri nahlasovaní chýb.

Používateľský príbeh: *V prípade, že by študent narazil na chyby v novej aplikácii, chcel by mať možnosť oznámiť túto chybu vývojárom.*

5.12.1 Spraviť nahlasovanie chýb s funkčným formulárom a hláškami používateľovi

VFIIT-69, Daniel Pribul

Analýza – Potreba získavať dáta od používateľov o chybách v aplikácii zostáva aktuálna aj v novej verzii. Stránka obsahuje pole na vyplnenie textu a tlačidlo na odoslanie správy o chybe so spätnou väzbou používateľovi.

Návrh – Správa o chybe sa bude odosielať pomocou metódy POST http protokolu. Po odoslaní sa zobrazí informácia o úspešnom, resp. neúspešnom odoslaní správy o chybe. V prípade úspechu sa správa z formulára vymaže.

Riešenie – Nahlasovanie chýb bolo implementované podľa návrhu.

Testovanie – Nahlasovanie chýb bolo otestované na všetkých zariadeniach, ktoré vlastníme.

5.12.2 Prerobiť nahlasovanie chýb – klientská časť

VFIIT-112, Filip Mazán

Analýza – V každej mobilnej aplikácii by sa mala nachádzať možnosť, ako používateľ môže nahlásiť problém v aplikácii. Je teda potrebné pridať podobný formulár aj do našej.

Návrh – Vytvorený by mal byť formulár zatiaľ s dvoma poliami - povinný popis chyby a nepovinný e-mail používateľa. Po odoslaní správy na server by sa mal odoslať e-mail s chybou na tímovú adresu.

Riešenie – Myšlienky z návrhu boli implementované a úspešne otestované.

5.12.3 Prerobiť nahlasovanie chýb – serverová časť

VFIIT-113, Filip Šoltés

Návrh – V novej aplikácii sme sa rozhodli implementovať funkcionalitu odosielania chyby podobne ako v starej verzii aplikácie, pričom sme pridali voliteľný parameter e-mail.

Riešenie – Funkcionalitu sme implementovali pomocou samostatného modulu v serverovej časti. Po spracovaní dopytu sú dáta zvalidované a to konkrétne validita e-mailu (ak je poslaný) a kontrola popisu na dĺžku a prítomnosť nepovolených špeciálnych znakov.

Po zvalidovaní vstupov sú tieto odoslané emailom využitím rozšírenia *nodemailer*.

Testovanie – Testovanie sme vykonali pomocou aplikácie *Postman* a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.12.4 Pozrieť ako poslať notifikáciu o spadnutí aplikácie

VFIIT-145, Filip Mazán

Analýza – Aplikácie postavené na rámci *Angular* môžu zlyhať na chybe, ktorá celú aplikáciu nezatvorí, ale len spôsobí zlyhanie interpreta JavaScript-u. Takéto chyby je možné odchytávať a zasielať o nich správy na server.

Riešenie – Bol nájdený spôsob a miesto v kóde, kde je vhodné tieto chyby odchytávať a odosielať o nich správy. Ďalší postup bude vytvorenie serverového rozhrania.

5.12.5 Správa o chybe by mala odosielať aj metadáta

VFIIT-160, Filip Mazán

Analýza – V súčasnosti sa so správou o chybe neodosielajú žiadne dodatočné dáta, na základe ktorých by bolo možné spôsobenú chybu lokalizovať, reprodukovať a opraviť.

Návrh – Spolu so správou o chybe by sa mali odosielať aj metadáta o zariadení, verziách aplikácie, rámcov, databázy a pod.

Riešenie – Do správy o chybe boli pridané nasledovné metadáta:

- verzia aplikácie
- ladiaci režim
- informácie o lokálnej pamäti
- platforma a jej verzia
- verzia *Cordova*

- verzia *Angular*
- verzia databázy
- verzia *BeaconPlugin*

Testovanie – Navrhnutá funkcionalita bola implementovaná a úspešne otestovaná.

5.12.6 Znížiť úroveň zabezpečenia nahlasovania chýb

VFIIT-142, Filip Šoltés

Návrh – Validácia poľa *description* pri odosielaní chyby je nastavená príliš reštriktívne a preto nie je možné vlastnosť plnohodnotne používať.

Riešenie – Na serverovej časti sme upravili validáciu poľa *description* pre nahlasovanie chýb. Odstránili sme validáciu na špeciálne znaky a zachovali sme len validáciu na povolenú dĺžku poľa.

Testovanie – Testovanie sme vykonali pomocou aplikácie *Postman* a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.13 Domovská obrazovka

Pôvodná verzia: Obrazovka obsahuje 6 tlačidiel na *Rozvrh, Hľadanie, Jedálne, Mapu FIIT, Mapu okolia* a *MHD*.

Naše zmeny v pôvodnej verzii: V pôvodnej verzii aplikácie sme túto obrazovku nemenili.

Nová verzia: V novej verzii prešla aplikácia niekoľkými zmenami. Počet tlačidiel sa zúžil na 4 a pribudla informácia o najbližšej vyučovacej hodine. Odkaz na vyhľadávanie bol nahradený samotným vyhľadávacím poľom.

Používateľský príbeh: *Študent chce v novej aplikácii prehľadnejšiu domovskú obrazovku, ktorá bude obsahovať všetky dôležité informácie.*

5.13.1 Spraviť úvodnú stránku (homescreen) s ikonkami

VFIIT-71, Michal Kučera, Daniel Pribul

Analýza – Úvodná obrazovka v pôvodnej verzii aplikácie podľa nás nevyužívala naplno svoj potenciál. Pôvodných 6 tlačidiel sme sa rozhodli zúžiť na 4. Rozhodli sme sa vynechať tlačidlo vyhľadávania a nahradiť ho samotným vyhľadávaním. Zvyšok vzniknutého miesta sme sa rozhodli zaplniť informáciou o najbližšej hodine a odkazom na miestnosť na mape, kde táto hodina bude prebiehať. Dostupný bude aj odkaz na podrobnejší detail tejto hodiny.

Návrh – Vypracovanie grafického návrhu, zapracovanie pripomienok od členov tímu a implementovanie schváleného návrhu do aplikácie. Zatiaľ iba staticky.

Riešenie – Bol vypracovaný prvotný grafický návrh, ktorý bol predvedený zvyšku tímu. Následne boli zapracované pripomienky a postupne sa týmto spôsobom dopracovalo k výslednej podobe úvodnej stránky (Obr. 11 – *a2*). Tá sa potom implementovala. Pri testovaní sa vyskytol problém pri vysunutí klávesnice, kedy sa celá obrazovka zúžila na cca polovicu svojej pôvodnej veľkosti. Tento stav sa podarilo vyriešiť nastavením výšky obrazovky v súbore *home.js* vždy pri načítaní obrazovky.

Testovanie – Testovanie úspešne prebehlo na všetkých zariadeniach v našom tíme s verziou operačného systému Android v rozsahu 2.3.5 až 4.4.4.

5.13.2 Najbližší predmet na hlavnej obrazovke

VFIIT-103, Michal Kučera

Analýza – Táto funkcia je novinka novej verzie aplikácie. Bude pridaná na úvodnú obrazovku na úkor tlačidiel, ktorých obrazovka po novom obsahuje menej.

Návrh – Funkcia bude dostupná iba po prihlásení a sprístupní názov a čas najbližšej hodiny a miestnosť kde sa bude odohrávať. Ponúkne používateľovi aj zobrazenie na mape, pričom v budúcnosti to môže byť aj priamo navigácia do miestnosti. Najbližšia hodina bude zobrazená aj 20 minút po jej začatí aby sme pomohli ľuďom, ktorí meškajú a nevedia rýchlo nájsť správnu miestnosť. Metóda na určenie najbližšej hodiny sa nachádza vo *factory/ais.js*. V prípade, že

používateľ nie je prihlásený, zobrazí sa namiesto najbližšej prednášky výzva na prihlásenie. Po jej stlačení bude používateľ presmerovaný na prihlásenie.

Riešenie – Riešenie bolo vypracované podľa návrhu.

5.13.3 Vyriešiť problém pri otáčaní hlavnej obrazovky

VFIIT-205, Michal Kučera

Analýza – Ionic štandardne umožňuje otáčanie medzi *landscape* a *portrait* zobrazením na všetkých obrazovkách. *Landscape* zobrazenie ale spôsobovalo problémy pri niektorých obrazovkách a dohodli sme sa, že najschodnejšie riešenie bude preto *landscape* v takýchto prípadoch zakázať.

Návrh – Zakázať *landscape* orientáciu pre všetky obrazovky okrem rozvrhu. V prípade, že ju nebude možné zakázať len na niektorých obrazovkách, zakáže sa na všetkých.

Riešenia – Na tento účel slúži v moderných prehliadačoch JS objekt *screen.orientation.lock(),* ktorý ale zatiaľ nemá dobrú podporu a navyše v našom prípade vyžaduje použitie zobrazenia na celú obrazovku (*full screen*), ktoré ale nie je pre našu aplikáciu vhodné. Použili sme teda alternatívne riešenie, pri ktorom sa nastavila v *preferencies* v súbore *config.xml* orientácia na hodnotu *portrait*. Toto riešenie ale nastavuje orientáciu pre konkrétnu aktivitu (pre Android) a tá je v hybridných aplikáciách zvyčajne len jedna pre všetky obrazovky. V praxi to znamená, že *landscape* orientácia bude zakázaná pre všetky obrazovky.

Testovanie – Riešenie cez úpravu *config.xml,* bolo úspešne otestované na všetkých zariadeniach v tíme.

5.14 Nastavenia

Pôvodná verzia: Neexistovala podobná obrazovka obsahujúca nastavenia aplikácie.

Nová verzia: Pre umožnenie zmeny základných nastavení aplikácie sme sa rozhodli implementovať obrazovku, v ktorej bude možné tieto nastavenia upravovať.

Používateľský príbeh: *Študent chce mať možnosť prispôsobiť si aplikáciu svojim potrebám.*

5.14.1 Vytvoriť nastavenia

VFIIT-198, Daniel Pribul

Návrh – V nastaveniach je možné nadstaviť oneskorenie zobrazovania najbližšej hodiny, pustiť ladiaci režim, vymazať cache a v prípade, že aplikáciu používa zamestnanec fakulty, tak je tu pole na nastavenie mena zamestnanca. Týmto spôsobom sa chceme vyhnúť nutnosti zadania prihlasovacích údajov pedagógov.

Riešenie – Všetky nastavenia sú uložené v *local storage* a ich zmena je okamžite uložená. Zoznam zamestnancov fakulty je prebraný z jsonu *people.json* a zoradený podľa mena abecedne vzostupne. Na tento účel bolo nutné odstrániť tituly zamestnancov, zoradiť ich, a následne zobraziť pomocou *selectbox*u. Zobrazenie nastavení môže kvôli spracovávaniu tohoto zoznamu trvať mierne dlhšie.

Testovanie – Riešenie bolo otestované na všetkých zariadeniach.

5.15 Ostatné

Úlohy, ktoré nie je možné jednoznačne priradiť k žiadnemu modulu.

Používateľský príbeh: *Študent chce prehľadnejšiu a použiteľnejšiu aplikáciu, aby sa mu s ňou lepšie pracovalo.*

5.15.1 Vytvoriť localStorage factory

VFIIT-75, Filip Mazán

Analýza – Tak ako v pôvodnej aplikácii, je potrebné perzistentne uchovávať dáta v zariadení. Na toto slúži tzv. *localStorage*, ktorú poskytujú ako prehliadače, tak aj aplikácie v mobilných zariadeniach. Preto je nutné túto úložnú kapacitu sprístupniť do našej aplikácie.

Návrh – Úložisko sprístupníme pomocou štruktúry *Factory* v AngularJS umiestnenú v súbore *storage.js* a bude obsahovať aspoň metódy na uloženie, načítanie a vymazanie hodnoty.

Riešenie – Úložisko bolo implementované podľa návrhu. Sprístupňuje metódy:

Tab. 1 Implementované metódy

metóda	popis
`$storage.get($key)`	načíta a vráti hodnotu s kľúčom $key
`$storage.set($key, $value)`	uloží hodnotu $value s kľúčom $key
`$storage.contains($key)`	vráti hodnotu pravda/nepravda podľa toho, či sa v úložisku nachádza hodnota s kľúčom $key
`$storage.delete($key)`	vymaže hodnotu s klúčom $key
`$storage.isAvailable()`	vráti hodnotu pravda/nepravda podľa toho, či je úložisko na zariadení podporované
`$storage.getLength()`	vráti počet hodnôt uložených v úložisku

Testovanie - Úložisko bolo otestované na všetkých zariadeniach, ktoré vlastníme.

5.15.2 Prerobenie pohľadov, ladiaceho módu a veci okolo

VFIIT-91, Filip Mazán

Analýza – Vo fáze začínajúceho vývoju je potrebné správne navrhnúť architektúru stavov aplikácie a pridať podporu pre vývojárov v podobe ladiaceho režimu.

Návrh – Do aplikácie boli zanesené konvencie vytvárania nových stavov v súbore app.js. Ladiaci mód je možné zapnúť z menu O aplikácii tlačidlom naspodku obrazovky. V ladiacom režime sa zobrazuje viacero informácii ako v produkčnom móde.

Riešenie – Myšlienky z návrhu boli implementované a úspešne otestované.

5.15.3 Prekladací modul

VFIIT-104, Filip Mazán

Analýza – Je zlou praktikou vpisovať akékoľvek reťazce priamo do kódu aplikácie z dôvodu ľahšej portability, podpory viacjazyčnosti a modularity systému ako celku. Je preto vhodné mať všetky lingvistické prvky na jednom mieste.

Návrh – Vytvorený bol súbor *lang.sk.js* obsahujúci všetky jazykové výrazy v podobe slovenskej fabriky. Vytvorená bola všeobecná fabrika *translate* poskytujúca prekladové funkcie. V rámci

uľahčenia prekladov aj vo výhľadoch bol vytvorený aj filter *translate*. Jeho použitie vo výhľade je veľmi jednoduché, napr.:

```
{{'app.canteens.today_is_closed' | translate}}
```

Podporované by mali byť samozrejme aj výrazy dynamicky doplňované informáciami. Parametre sú značené číselne v poradí od 1 predchádzané znakom percenta. Takéto výrazy majú tvar napr.:

```
"app.canteens.today_is_open": "Dnes je otvorené od %1 do %2"
```

Ich použitie je tiež jednoduché:

```
$translate.getTranslation(['app.canteens.today_is_open', today_ot.from,
today_ot.to])
```

Riešenie – Myšlienky z návrhu boli implementované a úspešne otestované.

5.15.4 Globálna konfigurácia

VFIIT-106, Filip Mazán

Analýza – Aby sme sa vyhli problému z minulosti, kedy pri každej zmene nastavenia aplikácie či zmien v rozvrhoch, predmetoch a pod. bolo nutné znova vybudovať aplikáciu a nasadiť ju na *Google Play*, sme pristúpili k možnosti konfigurácie, ktorú si aplikácia sama stiahne zo serveru.

Návrh – Vždy pri štarte aplikácie sa overí verzia konfigurácie. V prípade že existuje novšia verzia, server ju odošle aplikácii. Verziovanie databázy je na základe dátumu zmeny v konfigurácii.

Riešenie – Myšlienky z návrhu boli implementované a úspešne otestované.

5.15.5 Globálna konfigurácia – serverová časť

VFIIT-107, Filip Šoltés

Návrh – Keďže v starej verzii aplikácie bolo nutné publikovať novú verziu aplikácie pri každom obnovení databázy, rozhodli sme sa tento systém prerobiť a navrhnúť dynamické obnovovanie

databázy zo servera. Cieľom je, aby klient serveru poslal kód verzie databázy, ktorú aktuálne používa a server mu odoslal len tie dáta, ktoré sa zmenili.

Riešenie – Funkcionalitu sme implementovali pomocou samostatného modulu v serverovej časti, ktorá spracováva dopyty od klientov a modulu, ktorý obsahuje samotné dáta a konfiguráciu. V konfigurácii je záznam pre cestu ku každej časti konfigurácie spolu s kódom každej konkrétnej časti. Po spracovaní dopytu sa vytvorí objekt, ktorý obsahuje len časti s kódom väčším, ako bol prijatý od klienta. Server tiež pre klienta vygeneruje nový kód databázy, ktorý je maximom kódov všetkých častí na serveri.

Testovanie – Testovanie sme vykonali pomocou aplikácie Postman a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.15.6 Normalizácia chybových kódov na serverovej strane

VFIIT-133, Veronika Olešová

Analýza – Bolo by vhodné, ak by boli všetky chybové kódy na serveri uložené v jednom súbore.

Riešenie – Vytvorili sme súbor *ErrorCodes.js*, v ktorom sú chybové kódy zoskupené podľa modulov. Chybový kód pre nedefinované prihlasovacie meno je v tomto súbore uložený nasledujúcim spôsobom:

```
module.exports = {
ais: {
loginNotDefined: 'ais.validation.login.notDefined',
```

Príklad použitia chybovej hlášky v súbore aisValidator.js:

```
var loginIsDefined = validation.isDefined(params.login, this.errors,
errorCodes.loginNotDefined);
```

Testovanie – Zámerne boli vyvolané všetky chybové hlášky a tak overené ich správne zobrazovanie.

5.15.7 Validácia aktualizácie databázy na serveri

VFIIT-144, Veronika Olešová

Analýza – Je potrebné validovať klientskú verziu databázy, ktorá by sa mala spravidla skladať z 10 čísel. V prípade prijatia validnej verzie od klienta môže následne prebehnúť aktualizácia tejto databázy.

Návrh – Každý znak z reťazca danej verzie treba overiť, či predstavuje číslo.

Riešenie – Zhodli sme sa na tom, že overovanie každého znaku by predstavovalo príliš striktnú validáciu. Validovanie reťazca verzie ako celok na číslo je postačujúce riešenie.

Testovanie – Testovanie prebiehalo pomocou aplikácie Postman správnymi a nesprávnymi dopytmi na server. Odpovede zo servera zodpovedali očakávaniam.

5.15.8 Prerobiť dizajn aplikácie

VFIIT-125, Michal Kučera

Analýza – Spolu s prechodom na Ionic sa zmenil aj vzhľad aplikácie. Už samotný Ionic prostredníctvom predpripravených CSS tried poskytuje obstojný vzhľad ale my sme sa ho rozhodli posunúť ešte ďalej a prispôsobiť ho naším potrebám. V niektorých veciach (výber farieb, horná lišta) sme sa inšpirovali Material dizajnom od Google.

Postupne boli vytvárané grafické návrhy pre kľúčové obrazovky a tie sa priebežne podľa týchto návrhov implementovali. Keďže aj návrhy sa postupne vyvíjali, postupne začali vznikať problémy s nekonzistentnosťou niektorých návrhov a vyvstála potreba zjednotiť vzhľad celej aplikácie.

Návrh – Bol navrhnutý nový a jednotný vzhľad obrazoviek. Taktiež boli vytvorené aj chýbajúce návrhy pre tie obrazovky, ktoré boli implementované bez nich. Toto boli napríklad obrazovky pre detaily hodiny, vyučujúceho, miestnosti a predmetu (ďalej len obrazovky pre AIS).

Hlavná myšlienka jednotného vzhľadu spočívala v tom, že všetky informácie (pokiaľ to bude možné) budú zobrazené v kartách. Pre tieto potreby boli navrhnuté nové CSS triedy box, box-

header, box-subheader a box-content. Ďalej bolo treba vytvoriť Sass premenné pre farby písma, typu hodiny a pozadia.

Spôsob, akým by sa mali aplikovať CSS triedy a pravidlá, ktoré by sa mali pri navrhovaní a implementovaní dizajnových zmien dodržovať budú spísané vo zvlášť dokumente, nakoľko v tejto dokumentácii na to nie je dostatočný priestor.

Riešenie – Všetky zmeny sa týkali prevažne Sass a HTML. Boli vytvorené globálne CSS triedy (box a box-*) a premenné ($primary-text, $secondary-text, $excercise atď.) a pomocou nich boli zmeny implementované. Úpravou prešlo aj zvyšné CSS, pričom boli odstránené zbytočné alebo nepoužívané pravidlá a komplikované riešenia nahradené jednoduchšími. Výsledok týchto zmien je znázornený na obrázkoch v Príloha A.

Testovanie – Zmeny boli otestované na viacerých rôznych zariadeniach a všetko fungovalo podľa očakávania.

5.15.9 Implementovať monitorovaciu službu na server

VFIIT-143, Filip Šoltés

Návrh – Keďže serverová časť nie je odolná voči všetkým chybám je potrebné vyrobiť riešenie, ktoré bude monitorovať jej neočakávané pády.

Riešenie – Monitorovanie bolo implementované v jazyku BASH. Serverovú aplikáciu spúšťame v cykle, pričom pri páde presmerovávame chybový výstup do súboru. Po každom páde tiež odošleme email na definovanú adresu, pričom aplikácia je reštartovaná.

Ak serverová časť spadne viac ako definovaný počet krát za definovanú dobu, aplikáciu ďalej neštartujeme, ale je odoslaný email o prekročení definovaného počtu pádov za definovanú dobu.

Testovanie – Testovanie sme vykonali pomocou kontrolovaných pádov aplikácie a nezaznamenali sme problém pri žiadnej z implementovaných vlastností.

5.15.10 Lepší používateľský zážitok

VFIIT-168, Filip Šoltés, Filip Mazán

Analýza – Vlastník aplikácie i vývojári obecne požadujú rôzne štatistiky využitia aplikácie. My sme sa rozhodli v prvej fáze zaznamenávať činnosť používateľom vzhľadom na serverové volania. Z týchto dát by bolo potom možné určiť štatistiky používania jednotlivých funkcionalít aplikácie a korelovať ich s atribútmi používateľov.

Návrh – Na to, aby sme mohli efektívne zbierať dáta zo serverových volaní, je nutné pridať niekoľko vlastností ku každej požiadavke na server. Tieto vlastnosti by mali obsahovať najmä anonymný identifikátor používateľa a metadáta o aplikácii a zariadení.

Riešenie – Každá požiadavka na server teraz obsahuje okrem dátovej časti aj metadáta, ktoré zahŕňajú:

- anonymný identifikátor používateľa generovaný vždy po prvom štarte aplikácie - jedná sa konkrétne o UUID v1
- verzia aplikácie
- platforma
- verzia platformy
- príznak, či aplikácia beží pod rámcom *Cordova*
- verzia *Angular* rámca

Testovanie – Implementácia úpravy požiadaviek bola upravená ako na klientskej, tak aj na serverovej strane a úspešne otestovaná medzi tímom a neskôr aj v *Google Play Beta*.

5.15.11 Problémy s výkonom na androidoch < 4.4

VFIIT-172, Veronika Olešová, Filip Mazán

Analýza – Z nášho dôkladného prieskumu správ o natívnych chybách sme predpokladali, že chyby sú spôsobené zastaralým operačným systémom Android. Po hľadaní tohoto typu problému na internete sme zistili, že verzie OS Android menšie ako 4.4 zobrazujú obsah v štandardnom prehliadači, ktorý nepodporuje všetku potrebnú funkcionalitu, alebo ju vykonáva neefektívne

alebo chybne. Podarilo sa nám ale nájsť *CrossWalk Project*[4], ktorý implementuje vlastný vstavaný prehliadač postavený na rámci *Chromium*.

Návrh – Navrhujeme zahrnúť *CrossWalk Project* do našej aplikácie za účelom zvýšenia výkonu a stability.

Riešenie – Projekt *CrossWalk* bol úspešne zahrnutý do aplikácie, čo malo za následok značné zvýšenie výkonu a odstránenie sekania.

Testovanie – Aplikácia bola spolu s *CrossWalk* Projektom dôkladne otestovaná a nasadená do *Google Play Beta*.

5.15.12 Nesprávne zobrazovanie nedostupnej databázy

VFIIT-186, Daniel Pribul

Analýza – Konkrétnejšie informovanie používateľov o chybe považujeme za dôležité a preto sme sa rozhodli v prípade, že telefón nie je pripojený na internet a potrebuje sťahovať dáta zo servera, oznámiť túto informáciu používateľovi.

Návrh – V samotnom prehliadači nie je možné zistiť, ci je používateľ pripojený na internet. Preto je nutné použiť *cordova-plugin-network-information*. Následne je nutné pridať vyvolanie *pop-upu* s oznámením o chýbajúcom internetovom pripojení do každého volania serveru.

Riešenia – Funkcionalita bola implementovaná podľa návrhu.

Testovanie – Riešenie bolo otestované na všetkých tímových telefónoch.

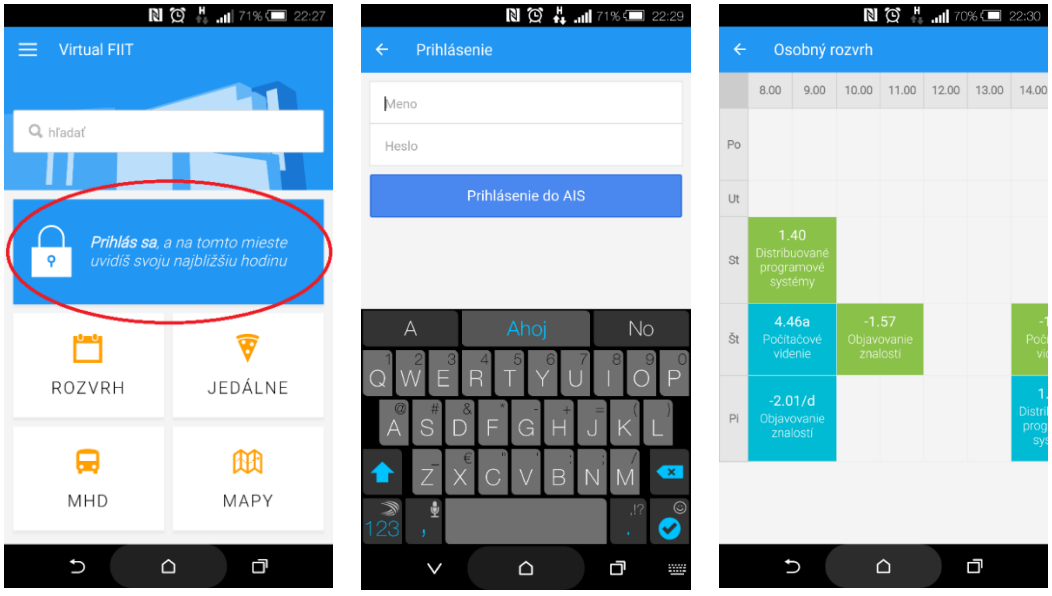
[4] https://crosswalk-project.org/

Príloha A Používateľská príručka

Táto príloha obsahuje návod na použitie existujúcej funkcionality v tejto aplikácii. Pomocou obrázkov obrazoviek popisuje používateľské príbehy akými sú prezeranie rozvrhu, vyhľadávanie výrazov, prezeranie odchodu autobusov a električiek, jedálnych lístkov, máp budovy a okolia, zisťovanie ďalších informácií o budove a nahlasovanie chýb aplikácie.

A.1 Prezeranie rozvrhu

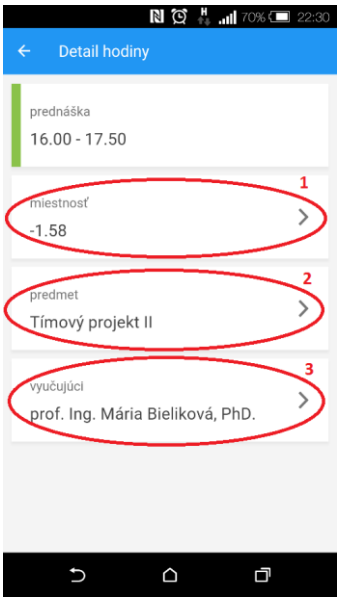
Po zapnutí aplikácie sa používateľovi naskytne pohľad na hlavnú obrazovku (Obr. 16a), ktorá mu ponúka možnosť prihlásenia sa do AISu. Kliknutím na vyznačený priestor sa zobrazí obrazovka na prihlásenie (Obr. 16b) s formulárom, do ktorého používateľ vpíše svoje prihlasovacie údaje do AISu. Pokiaľ prihlasovanie prebehlo úspešne, používateľ si môže prezerať svoj osobný rozvrh v prehľadnej tabuľke, čo je možné vidieť na Obr. 16c. Položky v tejto tabuľke, ktoré sú vyznačené zelenou farbou, predstavujú prednášky, pričom cvičenia sú zvýraznené modrou farbou. Každá z položiek obsahuje číslo miestnosti, v ktorej sa daná prednáška/cvičenie koná a samotný názov predmetu. Na každú z nich je možné kliknúť, čím je používateľ presmerovaný na detailnú obrazovku daného predmetu - Obr. 17.



a) b) c)

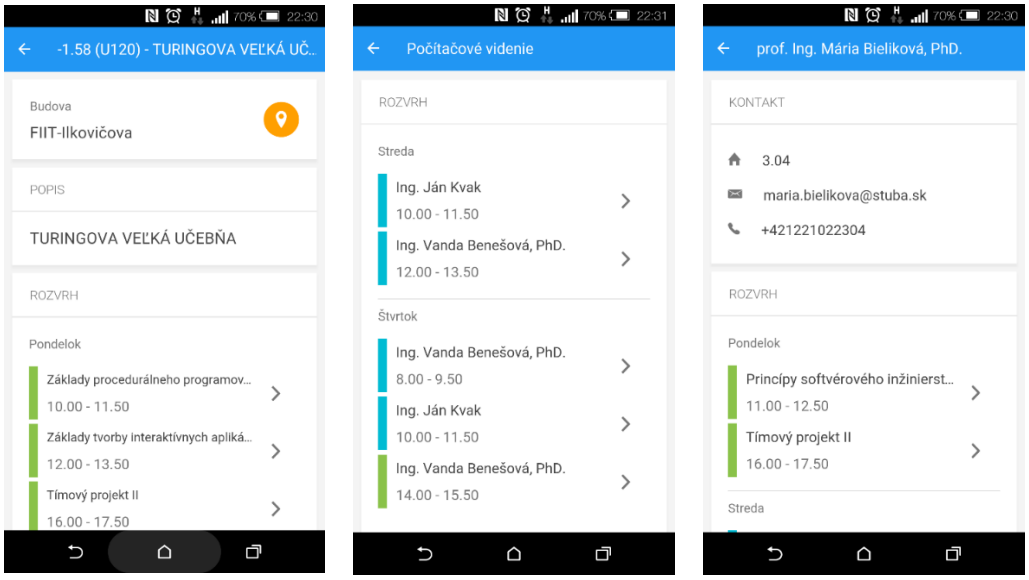
Obr. 16 a) hlavná obrazovka, b) obrazovka na prihlásenie sa, c) tabuľkový rozvrh používateľa

Na tejto obrazovke (Obr. 17) je možné vidieť, či ide o prednášku alebo cvičenie, presný čas trvania daného kurzu, miestnosť, v ktorej sa predmet vyučuje, názov predmetu a vyučujúceho. Odkazy, na ktoré je v tejto obrazovke možné kliknúť, sú vyznačené červenou elipsou spolu s poradovým číslom. Poradie odkazov na tejto obrazovke zodpovedá poradiu obrazoviek na Obr. 18.



Obr. 17 Obrazovka detailu predmetu

Obrazovka detailu miestnosti (Obr. 18a) obsahuje názov budovy, v ktorej sa daná miestnosť nachádza, názov tejto miestnosti a rozvrh prednášok/cvičení, ktoré sa v tejto miestnosti počas týždňa vyučujú. Táto obrazovka môže takisto obsahovať meno správcu miestnosti alebo meno vyučujúceho v prípade, že ide o kanceláriu. Ak sa používateľ dostane na obrazovku k predmetu (Obr. 18b), má možnosť vidieť zodpovedajúci rozvrh prednášok a cvičení spolu s údajom, či ide o letný alebo zimný semester. Detail vyučujúceho obsahuje jeho meno, umiestnenie kancelárie, kontaktnú emailovú adresu (voliteľné telefónne číslo) a jeho osobný rozvrh.

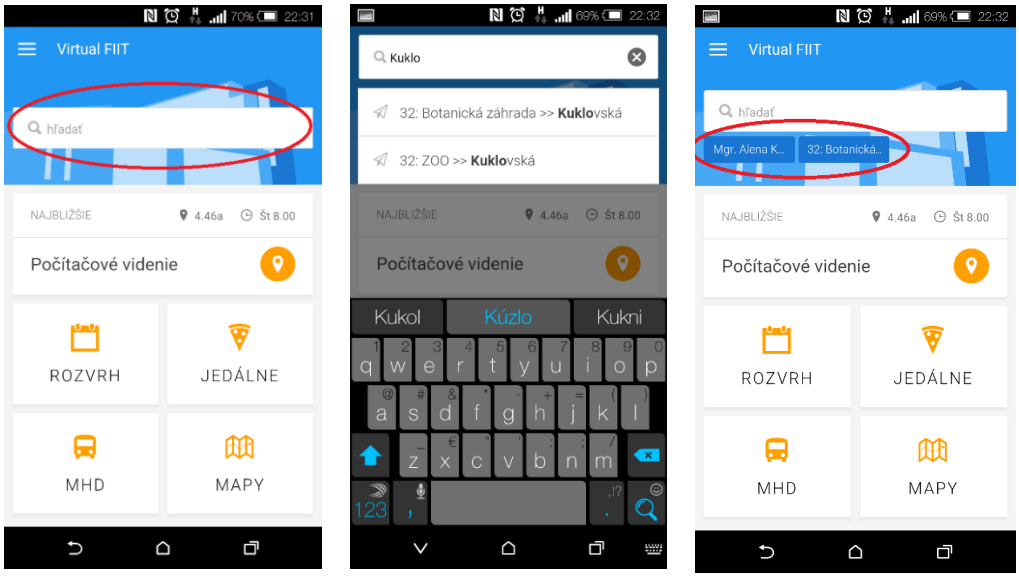


a) b) c)

Obr. 18 a) obrazovka pre miestnosť, b) predmet, c) vyučujúceho

A.2 Vyhľadávanie

V našej aplikácii je možné vyhľadávať nad množinou vyučujúcich, predmetov, miestností a MHD zastávok. Na hlavnej obrazovke (Obr. 19a) sa nachádza vyhľadávací panel, do ktorého môže používateľ začať písať hľadaný objekt z uvedenej množiny, čo je znázornené na Obr. 19b. Tento vyhľadávací panel používateľovi počas písania dynamicky ponúka najvhodnejšie výrazy. Úspešné vyhľadanie výrazu sa ukladá do histórie hľadaní používateľa, čo sa zobrazuje na hlavnej obrazovke (Obr. 19c).

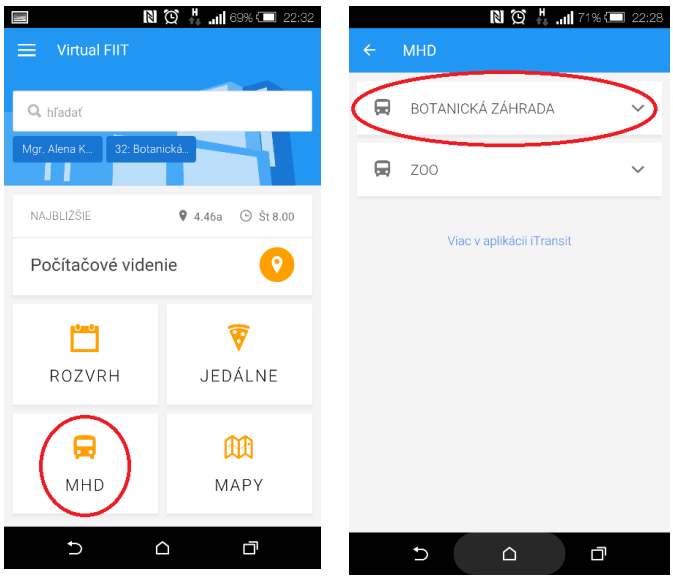


a) b) c)

Obr. 19 a) hlavná obrazovka, b) obrazovka pre vyhľadávanie, c)hlavná obrazovka s históriou vyhľadávania

A.3 Prezeranie odchodov spojov

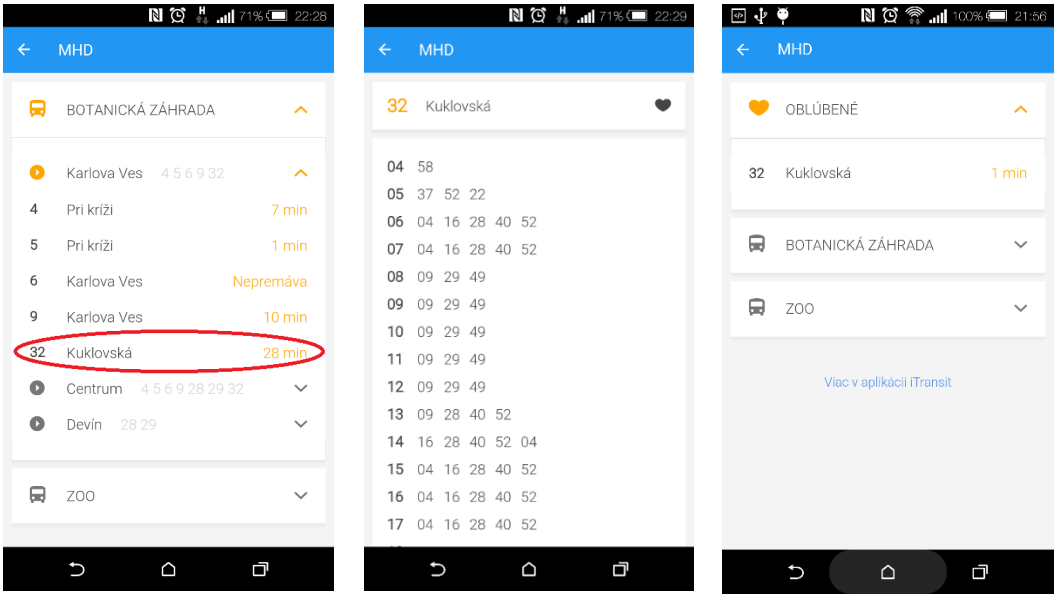
Aplikácia poskytuje aj prehľadný zoznam odchodov autobusov a električiek. Používateľ sa k tejto funkcionalite dostane kliknutím na položku *MHD* v hlavnej obrazovke (Obr. 20a). Hlavná obrazovka zastávok MHD (Obr. 20b) obsahuje 2 hlavné zastávky: *Botanická záhrada* a *ZOO*. Po kliknutí na jednu z nich, v našom prípade na zastávku *Botanická záhrada*, sa otvorí zoznam električiek a autobusov (Obr. 21a) spolu s ich konečnými zástavkami, na ktoré sa vieme z tejto zastávky dostať. Pri každej električke alebo autobuse je uvedený zostávajúci čas ich príchodu.



a) *b)*

Obr. 20 a) hlavná obrazovka, b) hlavná obrazovka zastávok MHD

V prípade, že používateľ klikne na jednu z položiek v tomto zozname, napríklad autobus číslo 32, otvorí sa mu nová obrazovka (Obr. 21b) so všetkými časmi odchodov tohto autobusu. Pri každom autobuse/električke sa nachádza ikona ♥, pomocou ktorej si používateľ môže pridať daný spoj medzi svoje obľúbené spoje, čoho výsledok sa zobrazuje na hlavnej obrazovke zastávok MHD (Obr. 21c).

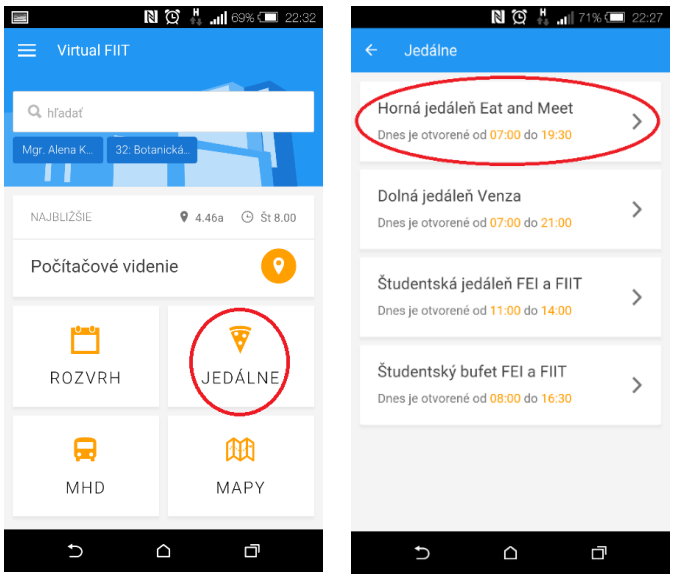


a) *b)* *c)*

Obr. 21 a) rozkliknutá obrazovka zastávok MHD, b) obrazovka zobrazujúca časy odchodov pre konkrétny autobus, c) obrazovka s pridaným obľúbeným spojom

A.4 Prezeranie jedálnych lístkov

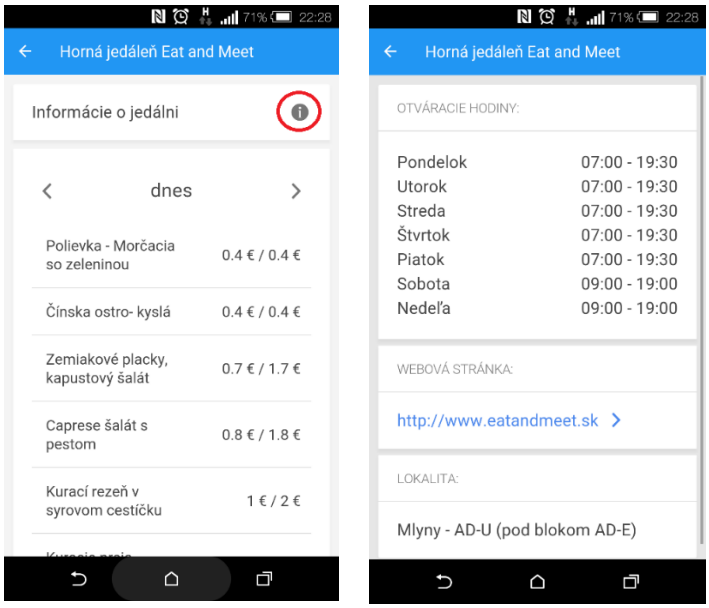
Naša aplikácia momentálne poskytuje jedálne lístky štyroch jedální, ktorých zoznam si používateľ môže prezrieť na obrazovke (Obr. 22b), ktorá sa zobrazí po kliknutí na položku *Jedálne* na hlavnej obrazovke (Obr. 22a). Pri každej jedálni je zobrazená otváracia doba pre aktuálny deň. Po kliknutí na jednu z jedální je používateľ presmerovaný priamo na aktuálny jedálny lístok, pričom na prepínanie medzi dňami (nasledujúci alebo predchádzajúci deň) sú k dispozícii šípky.



a) *b)*

Obr. 22 a) hlavná obrazovka, b) hlavná obrazovka jedální

Cena jedla, ktorá je uvedená ako prvá, platí pre študentov a druhá predstavuje bežnú cenu. Kliknutím na ikonu sa otvorí obrazovka (Obr. 23b) obsahujúca informácie o aktuálnej jedálni, akými sú otváracie hodiny, lokalita a jej webová stránka.



a) *b)*

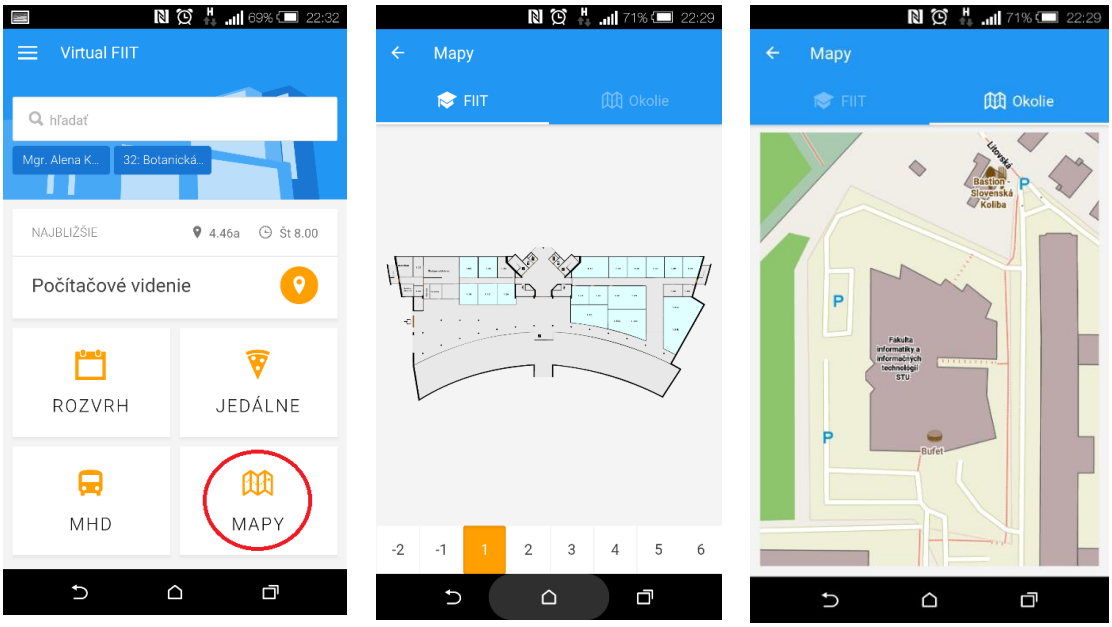
Obr. 23 a) obrazovka jedálneho lístka, b) informácie o konkrétnej jedálni

A.5 Prezeranie máp budovy a okolia

Na ľahšie zorientovanie sa v budove a mimo nej poskytuje naša aplikácia mapy budovy a okolia. K týmto mapám je možné dostať sa cez hlavnú obrazovku (Obr. 24a) kliknutím na položku *Mapy*. Obrazovka máp sa skladá z dvoch častí: obrazovka mapy budovy (Obr. 24b) a mapy okolia (Obr. 24c). V mape budovy je možné prepínať sa medzi jednotlivými poschodiami budovy pomocou dolných tlačidiel. Obe mapy sú interaktívne – je možné ich posúvať, vzďaľovať a približovať, ale hlavne klikať na isté časti mapy. V mapách budovy sú to miestnosti, na ktoré kliknutie spôsobí presmerovanie používateľa na detail miestnosti (Obr. 18a). Naopak, v mape okolia dokáže používateľ kliknúť na vyznačené jedálne - 

alebo zastávky - 

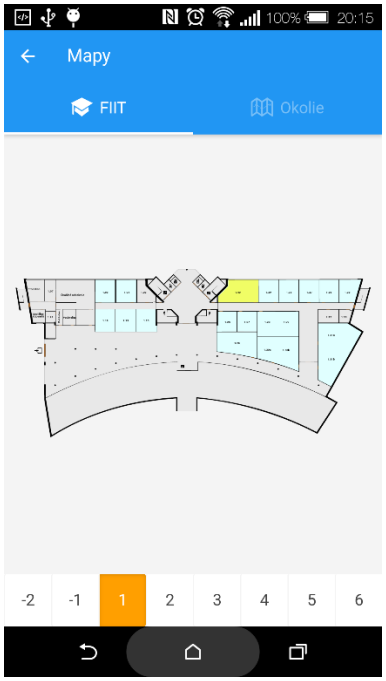
. Kliknutím na jedáleň je odkázaný na obrazovku s jedálnym lístkom (Obr. 23a) a kliknutím na zastávku zase na obrazovku k tejto zastávke (Obr. 21a).



a) *b)* *c)*

Obr. 24 a) hlavná obrazovka, b) obrazovka poschodia budovy, c) obrazovka mapy okolia

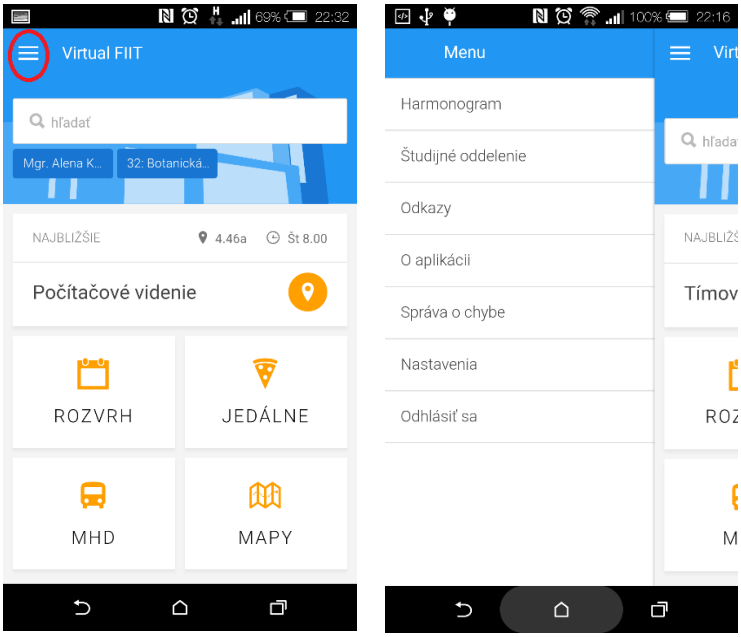
Na mapu budovy sa môže používateľ dostať z každej obrazovky, na ktorej sa nachádza interaktívna ikona  . Táto ikona sa spája s konkrétnou miestnosťou, preto je po zobrazení mapy budovy daná miestnosť vysvietená, čo sa dá vidieť na Obr. 25.



Obr. 25 Vysvietená miestnosť na mape budovy

A.6 Zistenie ďalších informácií o budove

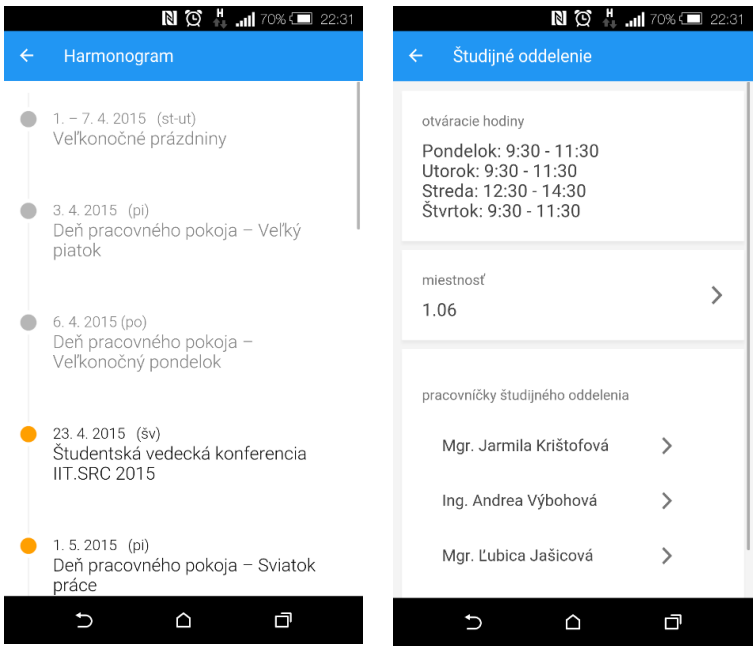
Študenti používajúci túto aplikáciu sa zaujímajú aj o ďalšie informácie o budove, medzi ktoré patrí fakultný harmonogram, otváracie hodiny študijného oddelenia, užitočné odkazy či informácie o samotnej aplikácii. Všetky tieto údaje sa dajú nájsť práve v bočnom menu (Obr. 26b), ktoré sa otvorí buď kliknutím na ikonu na hlavnej obrazovke (Obr. 26a) alebo vytiahnutím tohto menu z ľavého okraju monitora, čo je možné z ktorejkoľvek obrazovky.



a) b)

Obr. 26 a) hlavná obrazovka, b) bočné menu aplikácie

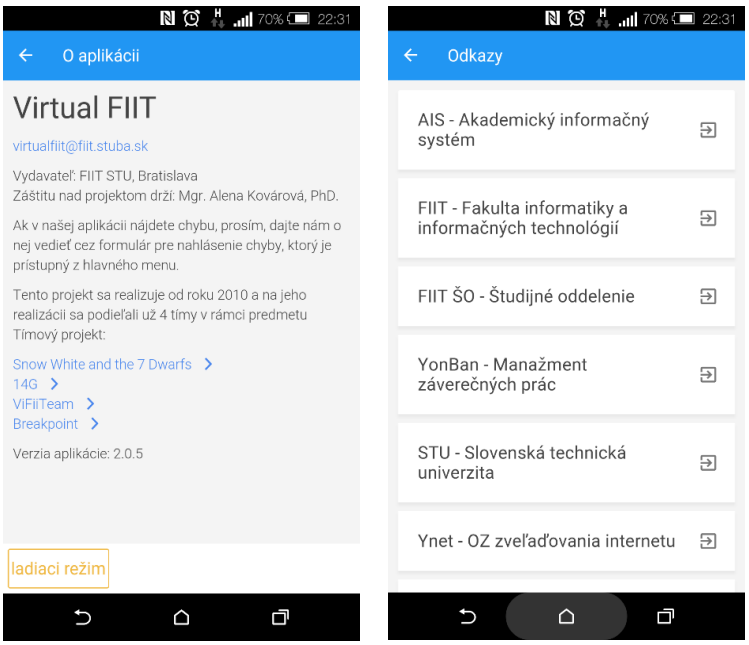
Po kliknutí na položku *Harmonogram* v bočnom menu sa zobrazí prehľadný fakultný harmonogram (Obr. 27a). Kvôli jednoduchému zorientovaniu sa v dátumoch udalostí sú položky, ktoré sa už udiali, vyznačené šedou farbou. K študijnému oddeleniu (Obr. 27b) sú poskytnuté informácie o otváracích hodinách, čísle miestnosti, kde sa toto oddelenie nachádza a menách pracovníčok, ktoré tu pracujú.



a) b)

Obr. 27 a) obrazovka harmonogramu, b) obrazovka študijného oddelenia

Na obrazovke *O aplikácii* (Obr. 28a) sú uvedené odkazy na predošlé tímy, ktoré na tomto projekte pracovali, verzia aplikácie a ďalšie informácie. Obrazovka s odkazmi (Obr. 28b) poskytuje študentom užitočné externé odkazy týkajúce sa fakulty a štúdia na nej.



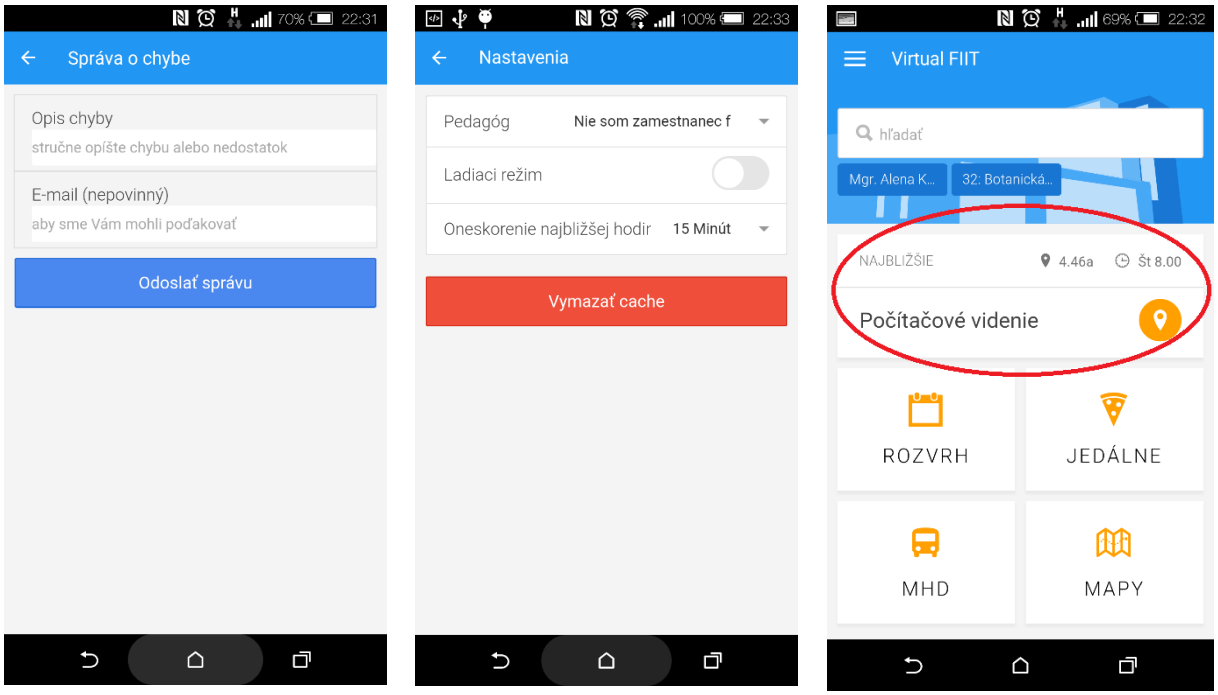
a) b)

Obr. 28 a) obrazovka O aplikácii, b) odkazy na dôležité stránky

A.7 Nahlásenie chyby a nastavenie aplikácie

V bočnom menu (Obr. 26b) sa taktiež nachádza obrazovka (Obr. 29a), pomocou ktorej nám používateľ môže nahlásiť chybu aplikácie. Povinnou položkou v tomto formuláre je *Opis chyby* a nepovinnou *E-mail* pre prípad, že by používateľ chcel dostať spätnú väzbu na nahlásenú chybu.

V aplikácii sa nachádzajú tri rôzne nastavenia (Obr. 29b), ktoré si môže používateľ prispôsobiť. Prvým z nich je voľba, či je aktuálny používateľ zamestnancom fakulty (inak by sme takého zamestnanca nedokázali prihlásiť do AISu). Ak sa používateľ označí ako jeden zo zamestnancov, musí uviesť správne prihlasovacie údaje do AISu na obrazovke Obr. 16b. Zapnutie ladiaceho režimu len zviditeľní ďalšie informácie o aplikácii nachádzajúce sa na obrazovke O aplikácii (Obr. 28a). Posledným nastavením je počet minút, ako dlho sa má zobrazovať najbližšia hodina na hlavnej obrazovke (Obr. 29c) po jej začatí. Maximálna možnosť oneskorenia je 1 hodina.



a) b) c)

Obr. 29 a) obrazovka na nahlasovanie chýb, b) nastavenia aplikácie, c) hlavná obrazovka

Príloha B Automatické testovanie

Táto príloha obsahuje návod na spojazdnenie a spustenie automatických testov vytvorených pre aplikáciu Virtual FIIT. Je doporučené ich spúšťať po každej zmene kódu.

B.1 Spojazdnenie automatických testov na počítači

Tu je jednoduchý návod, ako spojazdniť tieto testy na vašom PC:

Krok 1: Stiahnuť a nainštalovať Javu do vášho operačného systému

Ako prvé si potrebujete nainštalovať JDK (Java development kit) do vášho operačného systému. Java sa dá stiahnuť z tejto stránky.

Krok 2: Stiahnuť a nainštalovať Eclipse

V druhom kroku je treba si nainštalovať vývojové prostredie. Tento návod bude postavený na programe Eclipse, ktorý si môžete stiahnuť z tejto stránky. Pokiaľ chcete rozchodiť automatické testy na inom vývojovom prostredí, je potreba použiť google.

Krok 3: Stiahnuť WebDriver Jar súbory

Je potreba si ešte stiahnuť WebDriver Jar súbory. Na tejto stránke nájdite odsek Selenium Client & WebDriver Language Bindings a vyberte na stiahnutie Java verziu pre automatické testy (Obr. 30).

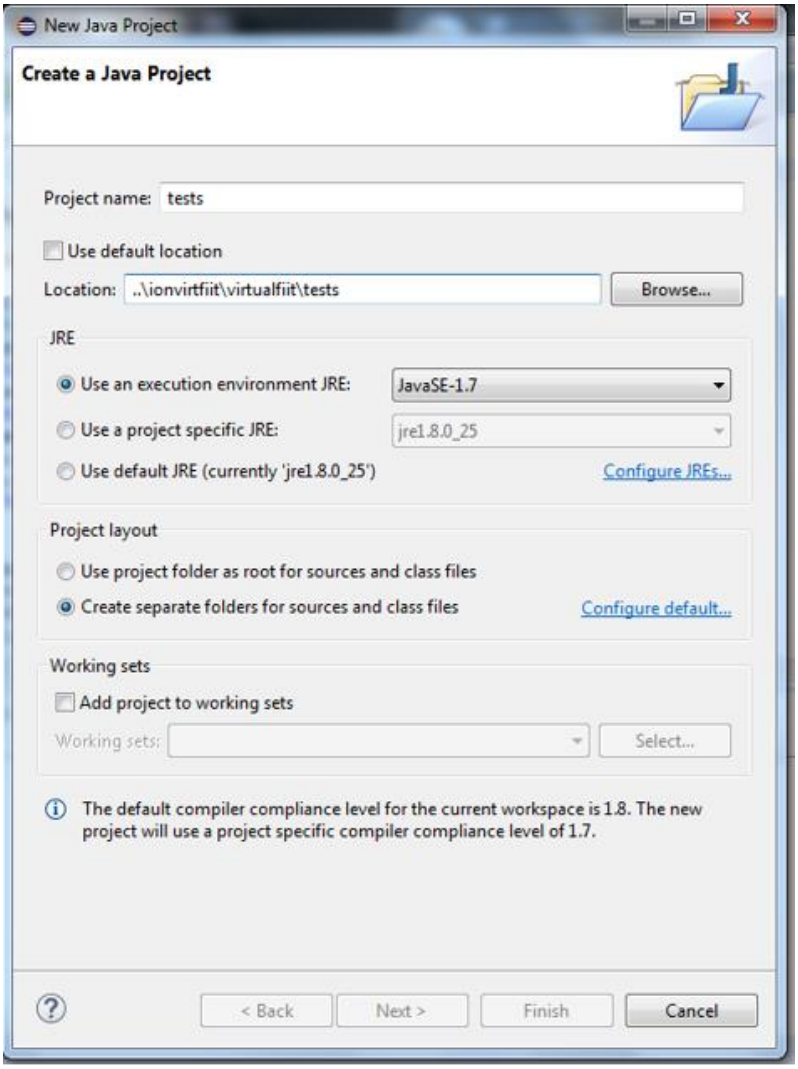
Language	Client Version	Release Date			
Java	2.33.0	2013-05-22	Download	Change log	Javadoc
C#	2.33.0	2013-05-22	Download	Change log	API docs
Ruby	2.32.0	2013-04-09	Download	Change log	API docs
Python	2.33.0	2013-05-22	Download	Change log	API docs

Obr. 30 Stiahnutie knižníc pre spustenie seleniových testov

Tieto súbory sú už ale obsiahnuté vo vývojovej verzii aplikácie Virtual FIIT, preto ak ju máte v počítači, netreba už tieto súbory samostatne ťahať. Nachádzajú sa v zložke ..\ionvirtfiit\virtualfiit\tests\Selenium lib.

Krok 4: Spustite Eclipse a otvorte v ňom priečinok s testami

Je potreba vložiť testy v programe Eclipse na workspace. To sa robí File > New > Java Project. Otvorilo sa nové okno. Odškrtnite Use default location a do textového poľa je potrebné zadať presnú lokáciu testov (zložka ..\ionvirtfiit\virtualfiit\tests). Následne stačí kliknúť na tlačidlo Finish (Obr. 31).



Obr. 31 Pridanie projektu Testy na workspace programu Eclipse

Krok 5: Pridanie externých Jar súborov

Keďže pracujeme každý na iných počítačoch, je pravdaže jasné, že cesta k automatickým testom bude rozdielna. Preto je potrebné ich pridať pred spustením ručne.

Postupnosť krokov:

- Pravý klik na projekt "tests" > Vybrať "Properties" > Vybrať "Java build path" > Prekliknúť na záložku "Libraries"
- Klik na tlačidlo "add external JARs" > vybrať všetky *.jar zo zložky "\ionvirtfiit\virtualfiit\tests\SelLib"
- Klik na tlačidlo "add external JARs" > vybrať všetky *.jar zo zložky "\ionvirtfiit\virtualfiit\tests\SelLib\libs"

Testy selenium pre aplikáciu Virtual FIIT sú pripravené na spustenie.

B.2 Spustenie automatických testov aplikácie Virtual FIIT

Sú 4 testy pre aplikáciu Virtual FIIT:

- Test prihlásenia
- Test jedální
- Test MHD
- Test Mapy

Všetky tieto testy obsahujú asserty, čo je kontrolný príkaz. Na správne fungovanie assertov spúšťajte testy nasledovne:

roletové menu "run" > "Run Configurations..." > Záložka "Arguments" > do textového poľa "VM arguments" je treba napísať "-ea".

Teraz vám už nič nebráni, aby ste spustili automatické testy.

Príloha C Inštalačné a integračné príručky

V tejto príručke sa nachádzajú návody na správne nastavenie vývojárskeho prostredia (klientská a serverová časť) a nasadenie aplikácie na Google Play a server stavba.fiit.stuba.sk.

C.1 Nasadzovanie na Google Play

Pred nasadením je potrebné skontrolovať server, na ktorý sa aplikácia dopytuje v `app/www/js/factories/serverapi.js`. URL musí byť nastavená na produkčný server `stavba.fiit.stuba.sk`.

Pre fungovanie uvedených príkazov je tiež potrebné mať nastavené systémové premenné `ANDROID_HOME` a `VFIIT_HOME`.

Pri nasadzovaní aplikácie na google play treba vykonať nasledovné:

1. Inkrementovať verziu v `app/package.json`
2. Spustiť `gulp` v adresári `app`
3. Spustiť build aplikácie pomocou skriptu
   `ionic build --release`
4. Podpísať vytvorené APK pomocou príkazu
   ```
   jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore
   $VFIIT_HOME/app/keys/virtualfiit.keystore android-release-
   unsigned.apk virtualfiit
   ```
5. Vykonať _zipalign_ podpísaného APK pomocou príkazu
   ```
   `$ANDROID_HOME/build-tools/<verzia>/zipalign -v 4 android-release-
   unsigned.apk  VirtualFIIT.apk`
   ```
6. Vloženie APK do Google Play pomocou developerskej konzoly

Pri nasadení je potrebné skontrolovať verziu, ktorú má vkladané APK. Je treba brať do úvahy, že nie je možné do Google Play vložiť APK s menším kódom verzie. Súčasný stav je 2000X. Pri builde pomocou `Crosswalk` sme zaznamenali problém s automatickým vkladaním dvoch číslic nakoniec. Ak by sme to neriešili, museli by sme každú ďalšiu verziu označiť 2000XYY a viac.

C.2 Nasadzovanie na server stavba.fiit.stuba.sk

Pri nasadzovaní aplikácie na server stačí využiť skripty v `scripts/deploy`. Skript `deploy_client.sh` slúži na vykonanie klientskej časti nasadzovania. Skript `deploy_server.sh` slúži na nasadenie aplikácie na serverovej strane. Pre korektné nasadenie musia zbehnúť obe časti.

Pre zbehnutie skriptov je potrebné nastavenie systémovej premennej `VFIIT_HOME` na klientskej strane.

Pred nasadením je dôležité skontrolovať server, na ktorý sa aplikácia dopytuje v `app/www/js/factories/serverapi.js`. Server musí byť nastavený na URL produkčného servera `stavba.fiit.stuba.sk`.

Skripty vytvoria .zip archívy zdrojových kódov servera a klienta, pričom sú pridané len potrebné časti. Na serveri je klientská časť aplikácie jednoducho nahradená za novú verziu.

Pri serverovej časti sú najprv zálohované aplikačné logy, následne sú nahradené zdrojové kódy a nakoniec je zabitý aktuálny proces serverovej časti a naštartovaný nový.

C.3 Integrácia serverovej časti

C.3.1 Inštalácia vývojárskeho prostredia

1. nainštalujte `node.js`, presný príkaz závisí od distribúcie

```
sudo pacman -S modejs
        -ALEBO-
sudo apt-get install
nodejs
        -ALEBO-
sudo yum install
nodejs
        ...
```

2. nainštalujte požadované závislosti pomocou príkazu

```
npm install
```

Príkaz je nutné spustiť v adresári `server/`

1. nainštalujte spúšťač testov (je potrebné aby bol nainštalovaný globálne) pomocou príkazu

```
sudo npm -g install
mocha
```

C.3.2 Spustenie servera

Server pre vývoj spustite pomocou skriptu `server/bin/www` pomocou príkazu

```
./bin/www -env dev
```

Pre vývoj odporúčame použitie nástroja `nodemon` vďaka ktorému nie je potrebné ručne reštartovať server po každej zmene. `nodemon` nainštalujte pomocou príkazu

```
sudo -g install nodemon
```

C.3.3 Spúšťanie testov

Testy vytvárajte v adresári `server/test`. Adresárová štruktúra musí kopírovať adresárovú štruktúru zdrojových súborov pričom súbory testov musia byť nazvané podľa zdrojového súboru, na ktorý sa vzťahujú použitím vzoru `*.test.js`. Adresáre, v ktorých sa testy nachádzajú musia v názve obsahovať príponu `-test`. Správne nazvanie súborov a adresárov je nevyhnutné pre spustenie testov.

Všetky testy spustite pomocou príkazu

```
mocha $(find test -name '*.js')
```

C.4 Inicializácia klientského vývojového prostredia

Nasledovný postup slúži ako pomôcka pri prvotnej inicializácii vývojového prostredia pre klientskú aplikáciu.

1. Uistite sa, že všetok potrebný softvér je nainštalovaný. Vyžaduje sa najmä:
    a. Java SDK
    b. Android SDK
    c. Python 2, Python 3
    d. Git
    e. Node.js, npm
2. Inicializujte si Git repozitár projektu:
    a. Vytvorte priečinok, kde si želáte projekt inicializovať.
    b. V príkazovom riadku sa presuňte do želaného adresára.
    c. Spustite príkaz

```
git clone git@bitbucket.org:virtualfiit/virtualfiit.git
```

3. V príkazovom riadku prejdite do priečinka *app* a spustite nasledovné príkazy. V prípade, že nastala pri spúšťaní príkazov chyba, skúste ju riadne prečítať a opraviť. Najskôr konzultuje problém na internete (v drvivej väčšine prípadov je možné riešenie nájsť), a pokiaľ problém nevyriešite, potom kontaktujte predošlý tím.

```
npm install -g cordova ionic bower
npm install
ionic platform add android
```

4. Nasledovným príkazom aplikáciu vybudujete a spustíte na emulátore alebo pripojenom Android zariadení.

```
ionic build
ionic run
```