VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ A MOBILNÍ APLIKACE PRO SEZNAMOVÁK

WEB AND MOBILE APP FOR SEZNAMOVÁK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAN MIKULÍK**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/21212/2017/xmikul53

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Mikulík Jan, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Webová a mobilní aplikace pro Seznamovák**
Web and Mobile App for Seznamovák
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s vývojem online aplikací s webovým i mobilním frontendem.
2. Popište aktivitu Seznamovák a analyzujte potřeby komunikace s pořadateli a účastníky.
3. Navrhněte webovou a mobilní aplikaci usnadňující komunikaci a zvyšující zapojení účastníků.
4. Prototypujte způsob interakce s aplikací a jednotlivé prvky uživatelského rozhraní a testujte je na uživatelích.
5. Implementujte celou aplikaci (webové i mobilní rozhraní).
6. Testujte vytvořené uživatelské rozhraní na uživatelích a iterativně je vylepšujte.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Android Developers: https://developer.android.com/index.html
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Jan Řezáč: Web ostrý jako břitva, Baroque Partners, 2014

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4, značné rozpracování bodu 5.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce bylo navrhnout a implementovat mobilní a webovou aplikaci pro seznamovací pobyt Seznamovák. Webová aplikace slouží pro administraci dat, která jsou následně stahována ze serveru mobilní aplikací. Mobilní aplikace slouží pro zobrazení stažených dat, jako jsou notifikace, program na Seznamováku a mapa areálu. Čtenář bude seznámen nejdříve s analýzou požadavků, která obsahuje také podobné již existující aplikace. Poté následuje rozbor použitých technologií. Nakonec návrh výsledného řešení a jeho implementace s testováním.

Abstract

Main task of this thesis is to design and implement mobile and web application for acquainted stay called Seznamovák. Web application serves for data administration, which are afterwards downloaded by mobile application. Mobile application is used to display downloaded data, such as notifications, program of Seznamovák and area map. The reader will be introduced to requirement analysis with already existing applications. It will also describe analysis of used technologies. Then final solution design and it's implemention with testing.

Klíčová slova

mobilní aplikace, webová aplikace, android, seznamovák, nette, rest, slim

Keywords

mobile application, web application, android, seznamovák, nette, rest, slim

Citace

MIKULÍK, Jan. *Webová a mobilní aplikace pro Seznamovák.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Webová a mobilní aplikace pro Seznamovák

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Mikulík
23. května 2018

Poděkování

Poděkování patří především vedoucímu práce panu prof. Ing. Adamu Heroutovi Ph.D. za odbornou pomoc a cenné rady při tvorbě této práce. Dále bych chtěl poděkovat týmu Seznamováku za inspiraci pro tuto práci a následné testování výsledku této práce.

Obsah

Kapitola 1

Úvod

Cílem této práce bylo navrhnout mobilní aplikaci běžící na platformě Android pro usnadnění informovanosti účastníků seznamovacího pobytu Seznamovák. Ten se koná zejména pro první ročníky vysokých škol v Brně. Dalším cílem byla také webová aplikace pro administraci mobilní aplikace.

Na Seznamovák jezdí každoročně přes 500 studentů, a to je poměrně náročné na organizaci. V tomto směru by měla pomoci mobilní aplikace, která by zobrazovala program celého Seznamováku, notifikace pro upozornění na změny v programu, nebo nějaké jednorázové akce a mapu areálu, aby se účastník mohl sám zorientovat bez další pomoci.

Tato práce nejdříve analyzuje problematiku a zkoumá podobné již existující aplikace, dále rozebírá použité technologie důležité k vypracování této práce. Následně rozebírá návrh řešení, a nakonec popisuje implementaci výsledného řešení.

Výstupem této práce je mobilní aplikace běžící na platformě Android obsahující již výše uvedené, webová aplikace běžící na Nette frameworku jako substituce pro uživatele zařízení jiných platforem, která navíc slouží i k administraci a jednoduché webové API vytvořené pomocí Slim frameworku na bázi REST požadavků. Mobilní aplikace je mimo jiné umístěna v obchodě Google Play[1].

[1] https://play.google.com/store/apps/details?id=org.seznamovak.seznamovak

Kapitola 2

Analýza problému

V této kapitole se rozebírá samotný Seznamovák, dále se podrobněji rozeberou dvě mobilní aplikace podobného stylu a na závěr se rozeberou požadavky jak by měla aplikace fungovat.

2.1 Popis akce Seznamovák

Seznamovák je seznamovací pětidenní pobyt, který probíhá každý rok před začátkem zimního semestru pro studenty prvních ročníků Brněnských univerzit. Každý rok na Seznamovák jezdí okolo 500 studentů. Program je bohatý a dobrovolný, takže každý účastník si vybírá, na kterou aktivitu by chtěl přijít. Zároveň ale občas nastávají náhlé změny v programu, kdy organizátoři musí obcházet všechny účastníky a informovat je. Seznamovák se koná v kempu ATC Merkur v Pasohlávkách, co je poměrně velký areál, takže na začátku je pro účastníka náročnější se zorientovat.

2.2 Podobné již existující aplikace

Colours Of Ostrava

Aplikace pro hudební festival Colours of Ostrava[1]. Design aplikace je na první pohled velmi čistý, jednoduchý a přehledný, co je velmi důležité pro uživatele, aby aplikaci následně používal. Po spuštění se zobrazuje seznam aktualit ohledně dění na festivalu, co je pro uživatele přínosné aplikaci nadále používat i když ji nespustil za daným účelem. Aplikace jinak obsahuje informace o areálu, účinkujících a možnost notifikace v případě, že za chvíli bude vystupovat oblíbený interpret. Aplikace taky obsahuje mapu, která je ve formátu velkého obrázku, dle mého názoru by bylo lepší použití přímo Google mapy a informace doplnit jako nadstavbu pro lepší interakci s uživatelem. Hodnocení aplikace je přes 4 hvězdy, hodně uživatelů si přímo v komentářích na Google play stěžovalo na chybu, na kterou ale bylo urgentně zareagováno záplatou. V současnosti má 10 až 50 tisíc stažení.

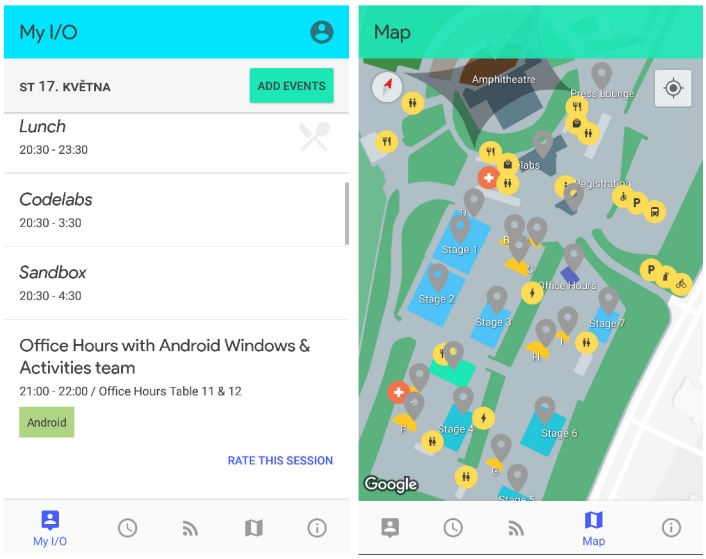
[1] https://play.google.com/store/apps/details?id=cz.merca.colours

Obrázek 2.1: Seznam aktualit a mapa v aplikaci Colours of Ostrava

Google I/O

Aplikace pro konferenci Google I/O[2], je vyvíjená společností Google, takže je zde vidět značné použití směrnic Material Designu, který je vysvětlen na straně 13. Aplikace je tedy na použití poměrně jednoduchá, navigace v aplikaci je umístěná dole do několika hlavních kategorií. Navigace obsahuje rozvrh uživatele, přehled všech událostí, seznam zpráv ohledně konference, mapu a užitečné informace pro uživatele spolu s nastavením samotné aplikace. Ačkoli jsou možnosti aplikace dost komplexní, tak jsem se po krátkém projití v aplikaci snadno zorientoval. Nejvíce mne zaujala mapa konference, u které se dost inspiruji. Hodnocení aplikace má stejně jako v minulém případě přes 4 hvězdy, v komentářích si uživatelé pouze stěžují, že není dostupný zdrojový kód aplikace, což již v tomto čase neplatí. Počet stažení je v rozmezí 500 000 až 1 000 000, což je na takovou konferenci dosti velké číslo. Aplikace je dostupná od verze Android OS 5.0, která jako první využívá již zmiňovaný Material Design.

[2] https://play.google.com/store/apps/details?id=com.google.samples.apps.iosched



Obrázek 2.2: Zobrazení osobního programu a mapy v aplikaci Google I/O

2.3 Cíl práce

Cílem této práce by měla být mobilní a webová aplikace. Mobilní aplikace by měla obsahovat:

- Notifikace – Uživatel by měl být upozorněn na důležité dění, např. změny v programu, reklamy na aktivity a různě.
- Program – Aplikace by měla mít kompletní program včetně detailních informacích o každé aktivitě, jako je čas aktivity a místo. Také by měla nabízet možnost ohodnocení aktivity jako zpětnou vazbu od účastníků.
- Mapu areálu společně se zobrazenými významnými lokacemi, také by účastníka měla navigovat na zvolené místo.

Aplikace by tedy měla být na bázi klient-serveru kdy si data bude stahovat ze serveru s možností uložení dat do databáze z důvodu ušetření dat a navíc také Wi-Fi signálu pouze v určitých částech areálu. Vstup do aplikace by byl přes sociální přihlášení (Facebook, Google) nebo pomocí údajů přes které se účastník přihlašuje do systému Seznamováku. Uživatelské rozhraní by mělo splňovat směrnice Material Designu. Aplikace by měla být dostupná v obchodu Google Play.

Webová aplikace by měla sloužit na správu dat, které si mobilní aplikace bude stahovat. Webová aplikace bude obsahovat:

- Správu notifikací – Vytvoření a úpravu ještě neproběhlých notifikací s časem upozornění na notifikaci.
- Správu programu – Zobrazení kalendáře, možnosti libovolné manipulace s aktivitami programu.
- Správu lokací v mapě – Zobrazení Google mapy, možnost přidání a úpravy lokací.

Use-case účastníka

Aplikace bude pro účastníky dostupná před začátkem Seznamováku. Když se účastník bude chtít podívat jaký program ho čeká, zapne aplikaci a podívá se. Aplikace bude využívat naplánované notifikace, takže např. notifikace ohledně hlavní schůze je upozorní, že se mají dostavit, anebo náhlá změna nadcházející aktivity v průběhu dne. Pokud účastník bude chtít vědět, kde se určitá aktivita koná, může si to na mapě zobrazit a pomocí sledování polohy se tam může odnavigovat.

Kapitola 3

Použité technologie

Jakožto cíl této práce bylo vytvoření mobilní a serverové aplikace, je nutné uvést několik základních technologií, které byly použity pro zhotovení výsledného řešení. Pro mobilní aplikaci byla zvolena platforma Android, která je aktuálně na trhu mobilních zařízení nejpočetnější v rámci počtu mobilních zařízení.

Nicméně nesmí se na ostatní platformy úplně zapomenout, neboť i tak je počet zařízení s operačním systémem iOS poměrně vysoký, proto je později popsáno řešení tohoto problému. Nabízí se také možnost využít multiplatformního řešení jakožto Reactu, nebo Xamarinu, ale z důvodu mých dřívějších zkušeností s programováním na platformě Android a zájmu prohloubit mé znalosti v tomto oboru jsem se rozhodl právě pouze pro Android.

Pro webovou aplikaci, která by měla sloužit pro administraci dat pro mobilní aplikaci jsem zvolil Nette framework 3.2.1, jelikož s ním mám již zkušenosti a na tuto práci vyhovuje mým potřebám. K Nette jsem doplnil ještě Slim framework 3.2.3 pro REST protokol, na kterém je tento framework postaven.

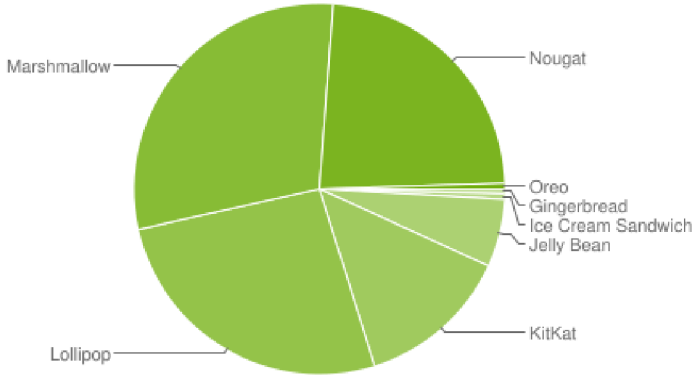
3.1 Android

Mobilní operační systém [4] je založený na Linuxovém jádře. Je dostupný jako open-source (otevřený software). Je orientovaný na chytrá zařízení, zejména chytré telefony. Aplikace můžou být psány v jazycích Java, C++ a od roku 2016 také v jazyce Kotlin[1].

Jednotlivé verze operačního systému jsou číslovány a od verze 1.5 je každá verze také označována názvy různých sladkostí, například Oreo či Nougat. Každá verze má také svoji API úroveň.

V případě vývoje aplikace je nutné stanovit minimální úroveň API aplikace, které značí od které verze systému bude aplikace podporována.

[1] Kotlin je staticky typovaný jazyk, který kombinuje objektově-orientované a funkcionální programování [5].

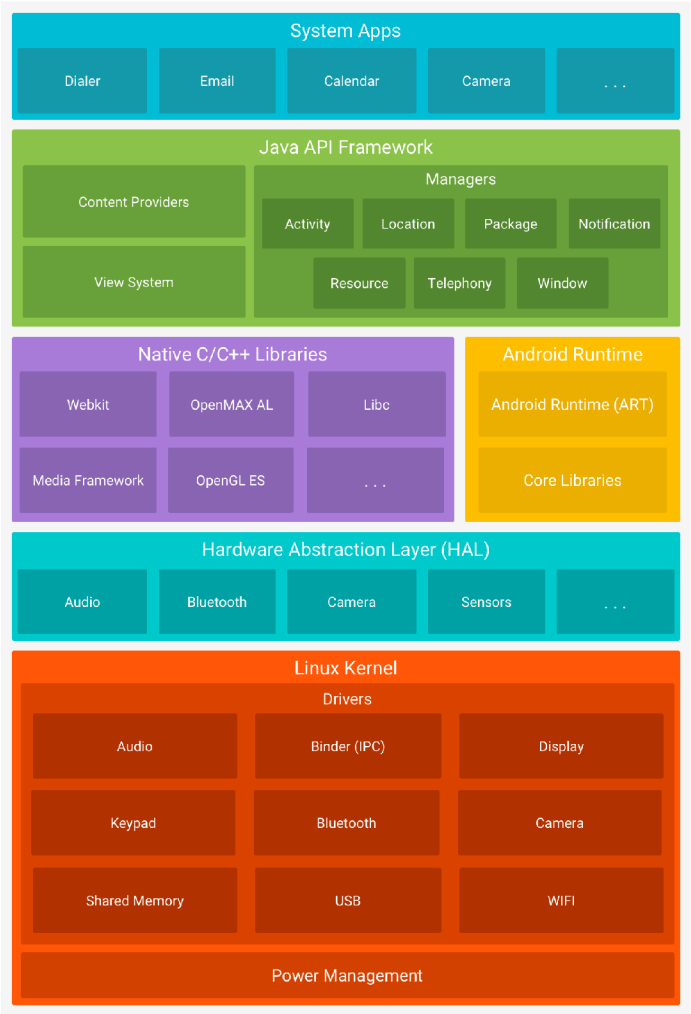


Obrázek 3.1: Zastoupení verzí operačního systému na aktuálním trhu ke dni 23.12.2017.

3.1.1 Architektura OS

Architektura operačního systému je rozdělena do 6 částí [4]:

- Jádro (Linux Kernel) – Jádro tvoří základ operačního systému, Android platforma je postavena na Linuxovém jádře ve verzi 2.6 s určitými změnami. Jádro poskytuje podporu pro režii paměti, procesů, napájení apod. Také obsahuje drivery pro jednotlivé komponenty.
- Abstraktní hardwarová vrstva (HAL) – HAL poskytuje standardní rozhraní, které vystavuje schopnosti zařízení na vysoko-úrovňové Java API Framework. Sestává z několika knihovních modulů, kde každý implementuje rozhraní pro specifický typ hardwarové komponenty, jako např. kamera nebo bluetooth. Když API framework žádá o přístup ke komponentě, systém načte knihovní modul pro tu komponentu.
- Android Runtime (ART) – Slouží primárně pro běh aplikací. Pro zařízení běžící na verzi 5.0 a vyšší, každá aplikace běží ve svém vlastním vlákně spolu s instancí ART. Ve starších zařízení slouží pro běh aplikací virtuální stroj Dalvik Virtual Machine.
- Nativní knihovny – Napsané v C++, implementující některé základní funkce systému. Android platforma poskytuje API Java frameworku vystavit funkcionalitu některých nativních knihoven, např. grafické API OpenGL ES skrze Java OpenGL API přidává podporu pro vykreslování a manipulaci 2D a 3D grafiky v aplikaci.
- Java API Framework – Celý soubor funkcí operačního systému je dostupný skrze API psané v jazyce Java. Tyto API formují bloky, které jsou potřebné pro tvorbu aplikací.
- Systémové aplikace – Soubor aplikací pro základní operace např. pro emaily, SMS zprávy, procházení internetu atd. Systémové aplikace slouží pro uživatele a také provádí klíčové operace pro vývojáře, které usnadňují práci při vývoji aplikace.



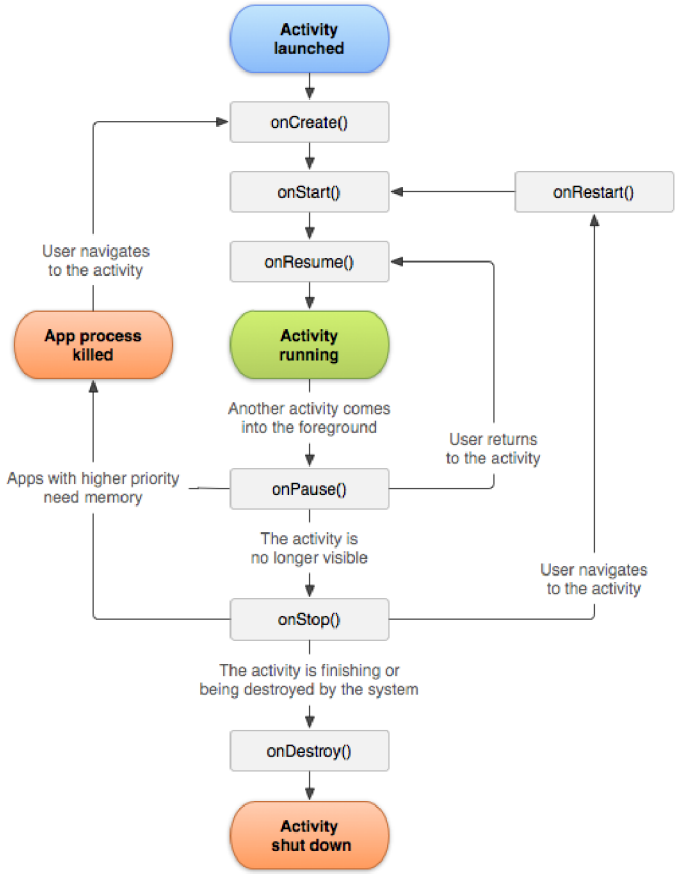
Obrázek 3.2: Architektura operačního systému Android. Architekturu lze rozdělit na 6 sekcí – Jádro, Abstraktní hardwarová vrstva, Android Runtime, Nativní knihovny, Java API Framework a Systémové aplikace. Každá tato část je tvořena jednotlivými komponentami, které definují funkce OS.

3.1.2 Komponenty aplikace

Aplikační framework Android platformy umožňuje vytvářet bohaté a inovativní aplikace pomocí sady znovupoužitelných komponent [4][6].

Aktivity (Activities)

Aktivity jsou jedním ze základních stavebních bloků aplikací na Android platformě. Slouží jako vstupní bod interakce uživatele s aplikací a jsou také klíčové pro to, jak se uživatel pohybuje v aplikaci. Každá aktivita má svůj životní cyklus, ve kterém se přepíná v průběhu běhu aplikace, který je znázorněný na obrázku 3.3. Na obrázku jsou také znázorněny metody, které se spouští při změně stavu aktivity.



Obrázek 3.3: Životní cyklus aktivity. Z obrázku je jasné, že aktivita se může nacházet v několika stavech, všechny tyto stavy lze zachytit pomocí již předdefinovaných callbacků.

Fragmenty (Fragments)

Fragment představuje chování nebo část uživatelského rozhraní v aktivitě. Fragmenty lze kombinovat v jedné aktivitě pro vytvoření multi-panelového uživatelského rozhraní a opětovně použít fragment v jiné aktivitě. Fragment si lze představit jako modulární část aktivity, která má svůj životní cyklus, přijímá vlastní vstupní události a může být libovolně přidán nebo odebrán, zatímco aktivita stále běží.

Služby (Services)

Služba je aplikační komponenta, která může provádět dlouhodobé operace na pozadí, nemá tedy žádné uživatelské rozhraní, takže po spuštění služby nezávisí na tom, zda je uživatel stále v aplikaci. Služby se dají také považovat jako jeden z typů vláken, které je možné provozovat na Android platformě.

Poskytovatelé obsahu (Content Providers)

Poskytovatelé obsahují můžou pomoci aplikaci spravovat přístup k datům, které si uložila, nebo jiné aplikace a provozuje možnost sdílet data ostatním aplikacím.

Intenty (Intents)

Intent je objekt zasílání zpráv, který může být použit k žádosti o akci z jiné komponenty aplikace. Ačkoli intenty usnadňují komunikaci mezi komponentami několika způsoby, ve většině případech se ale používají následující dva případy použití:

- Spuštění aktivity – Intent popisuje aktivitu, která se má spustit a obsahuje další potřebná data.
- Spuštění služby – Používá se pro spuštění jednorázových operací (např. stažení souboru), intent popisuje službu, kterou má spustit a obsahuje další potřebná data.

Zdroje (Resources)

Zdroje jsou oddělená část aplikace od zdrojových kódů, která obsahuje dodatečné soubory a statický obsah jako návrhy uživatelského rozhraní aktivit, obrázky, texty apod.

3.1.3 Práce se sítí

Aby aplikace měla přístup k síti, je nutné deklarovat v konfiguračním souboru povolení `android.permission.INTERNET`. Povolení slouží k obeznámení uživatele k jakým prostředkům má aplikace přístup. Bez deklarování povolení aplikace nemůže přistupovat k internetu.

Od API 11 je implicitně nastavený *StrictMode*, který hlídá, zda se veškerá práce se sítí děje mimo hlavní vlákno aplikace. Pokud tak není učiněno, aplikace se ukončí s výjimkou `NetworkOnMainThreadException`. To se děje hlavně z důvodu, že na hlavním vlákně probíhají operace s uživatelským rozhraním. Pokud by na hlavním vlákně bylo například čekání na zprávu, aplikace by nereagovala na interakci uživatele a aplikace by byla označena, že neodpovídá a uživatel by ji musel ukončit.

Android používá dvě implementace HTTP klienta. Java knihovny `java.net.HttpURLConnection` a `AndroidHttpClient`, která se již ale nepoužívá [6].

OkHttp

OkHttp[2] je knihovna pro jednoduší práci s HTTP. Umí zpracovávat jak synchronní, tak asynchronní požadavky. Definuje 3 základní typy: `Request`, `Response`, a `Call`. `Request` je synchronní požadavek, který je následován odezvou `Response`. `Call` slouží pro asynchronní volání.

Retrofit2

Retrofit2 je typově-bezpečný REST klient pro Android. Knihovna poskytuje mohutný framework pro autentifikaci a interakci s API společně se zasíláním HTTP požadavků prostřednictvím knihovny `OkHttp`.

Tato knihovna vychází z obou výše zmíněných implementací, ze kterých obsahuje to nejlepší.

3.1.4 Android Studio

Android Studio [6] je oficiální integrované vývojové prostředí pro vývoj Android aplikací, je založené na IntelliJ IDEA prostředí od JetBrains. Krom výkonného textového editoru a

[2] http://square.github.io/okhttp/

vývojových nástrojů nabízí několik dalších vymožeností pro zvýšení produktivity při vývoji aplikací, jako např.

- Flexibilní sestavovací systém Gradle
- Rychlý emulátor se spoustou možností
- Jednotné prostředí pro vývoj pro všechna zařízení
- Okamžité spuštění změn do běžící aplikace bez nutnosti sestavení nové APK
- Šablony kódu a integrace Githubu a importování vzorových ukázek kódu
- Rozsáhlé testovací nástroje a frameworky
- Podpora C++ a NDK
- Vestavěná podpora pro Google Cloud Platform

Gradle

Android Studio používá sestavovací systém Gradle pro pokročilé sestavování, automatizaci a správy procesu sestavení aplikace a zároveň přináší programátorovi možnost vlastní konfigurace pomocí vlastních skriptů.

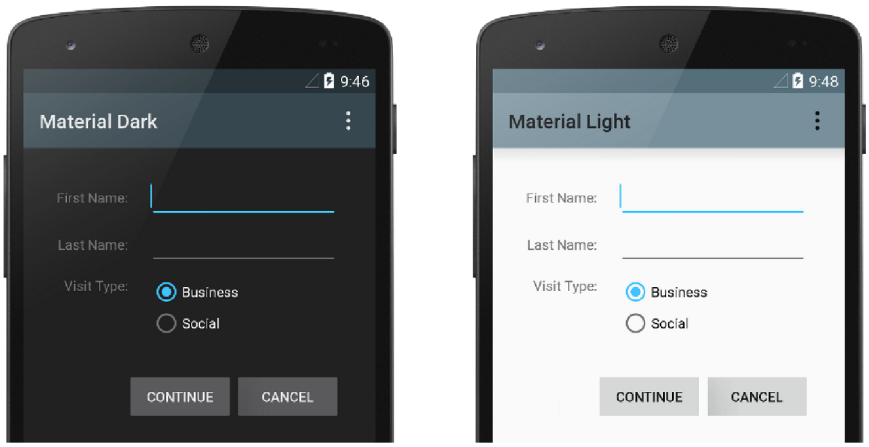
3.1.5 Material Design

Material design je obsáhlý průvodce pro vizuální, pohybový a interaktivní design napříč platformami a zařízeními. Material design je na platformě Android dostupný od verze 5.0, tento styl se následně rozšířil i do iOS a webových aplikací. Na stránkách Material Designu[3] je např. dostupných přes 900 ikon, kde každá je určená pro unikátní účel, aby následně uživatel věděl bez dalšího hledání co daná ikona znamená [6][3].

Cílem bylo vytvořit vizuální jazyk, který syntetizuje klasické principy dobrého designu. Byly vytvořeny tyto principy:

- Materiál je metafora – Sjednocující teorie prostoru a systému pohybu. Povrchy a okraje materiálu poskytují vizuální značky, které jsou zakotveny v realitě. Základy světla, povrchu a pohybu jsou klíčem k tomu, jak se pohybují předměty, interagují a existují v prostoru a ve vztahu k sobě navzájem.
- Tučně, graficky, záměrně – Záměrná volba barev, písma, hran objektů a dále.
- Pohyb poskytuje význam – Pohyb je smysluplný a vhodný, sloužící k udržení pozornosti. Zpětná vazba je jemná, ale jasná. Přechody jsou efektivní, avšak soudržné.

[3] https://material.io/icons/



Obrázek 3.4: Při použití Material Designu [6] na Androidu je nutné vybrat témata stylu, které mají již přednastavené styly pro celou aplikaci, V základu je na výběr mezi tmavým stylem a světlým stylem.

3.1.6 Použité knihovny

Crashlytics

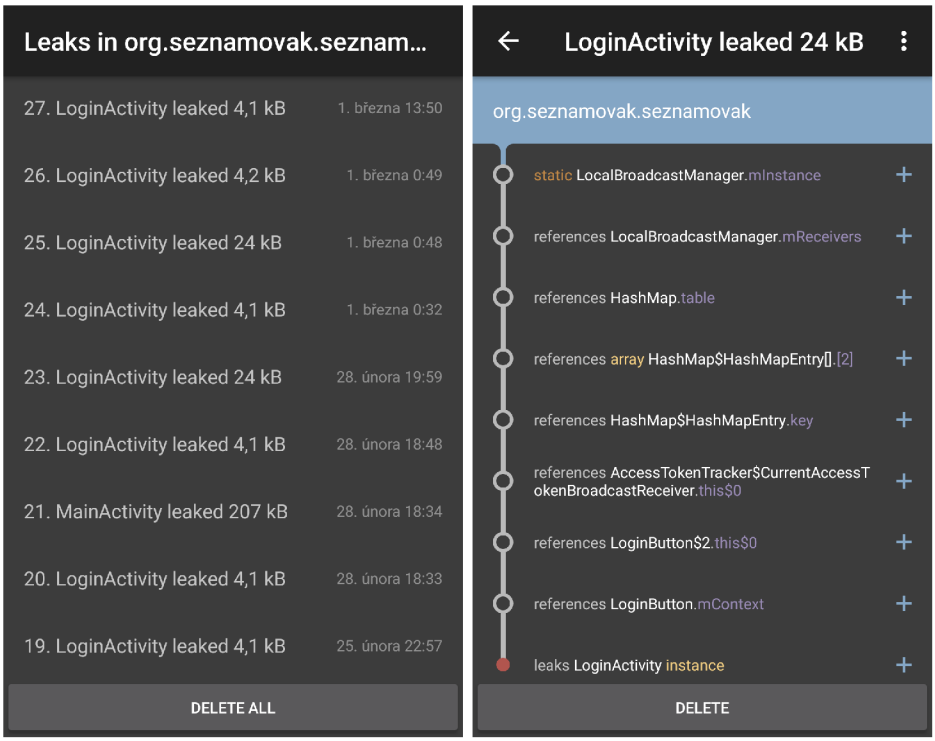
Crashlytics [8] slouží pro zaznamenávání chyb aplikace, když je aplikace již v produkci mezi uživateli. Obsahuje intuitivní webové rozhraní pro přehled všech pádů aplikace, počtu aktivních uživatelů a počtu nových uživatelů. Umožňuje také zaznamenávat akce, které programátor může implementovat v aplikaci.

Firebase JobDispatcher

Firebase JobDispatcher [10] je knihovna pro plánování úloh na pozadí aplikace. Z důsledku náročnosti na operační pamět a výdrž baterie zařízení byla uvedena třída JobDispatcher, která slouží pro jednoduché plánování úloh, nicméně tato třída je dostupná až pro zařízení s Api 21, takže proto vznikl Firebase JobDispatcher, který poskytuje kompatibilitu použití třídy JobDispatcher od verze Api 9.

LeakCanary

LeakCanary [9] je open-source knihovna pro detekci paměťových úniků pro Android a Javu. Automaticky monitoruje aplikaci a v momentě, kdy nastane únik, tak uživatele upozorní následným výpisem stromu objektů. To je zejména užitečné, neboť se programátor naučí jak psát aplikace čistě. Jak aplikace vypadá při používání je znázorněno na obrázku 3.5.



Obrázek 3.5: Na obrázku vlevo lze vidět seznam úniků, kde u každého úniku je název aktivity, která způsobila únik a velikost úniku. Na obrázku vpravo je vidět samotný strom, kde jsou reference na objekty, které zůstaly neuvolněny. Důvod tohoto úniku je neuvolnění Facebook přihlašovacího tlačítka před ukončením aktivity.

3.2 Web

3.2.1 Nette Framework

Nette [1] je obsáhlý Framework pro PHP, který výrazně zjednodušuje tvorbu webových aplikací a je open-source. Jeho autorem je český vývojář David Grudl. Je napsaný v PHP5 s plným využitím objektů (OOP). Jeho hlavními přednostmi jsou vlastní šablonový systém Latte, ladící nástroj Tracy, výrazně zjednodušuje práci s databází a je bezpečný.

Nette je klasický MVC framework, Model-View-Controller (MVC) je softwarová architektura, která vznikla z potřeby oddělit u aplikací s grafickým rozhraním kód obsluhy (controller) od kódu aplikační logiky (model) a od kódu zobrazujícího data (view). Tím jednak aplikaci zpřehledňuje, usnadňuje budoucí vývoj a umožňuje testování jednotlivých částí zvlášť.

- Model je datový a funkční základ aplikace, definuje logiku aplikace, jako např. komunikaci s databází.
- View neboli pohled, je ta vrstva aplikace, která určuje co se má koncovému uživateli zobrazit v prohlížeči. Obvykle obsahuje Latte šablonu s HTML kódem, Latte je šablonovací jazyk, který do HTML šablon umožňuje vkládat data z PHP pomocí speciálních značek.

- Controller je v Nette presenter, je to komponenta, která zpracovává požadavky uživatele a na jejich základě vykonává různé činnosti, které pomocí View předá uživateli, presenter tedy funguje jako takový prostředník.

Google Map API

V této práci je také použit doplněk Google Map API[4] od Petra Olišara pro ulehčení práce s mapou. Umožňuje bez použití Javascriptu inicializovat mapu a vložit na místa značky, co je hlavní účel této sekce.

3.2.2 REST Api

REST (Representational State Transfer) – je architektura rozhraní, navržená pro distribuované prostředí. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům, zdrojem mohou být data, stejně jako stavy aplikace. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu [7]:

- `GET (Retrieve)` – Používá se pro získání zdrojů, vyskytuje se na každém webu, jedná se o požadavek na stránku, který je ve tvaru

  ```
  GET /api/user/pepa
  Host: www.example.com
  ```

  To odpovídá adrese www.example.com/api/user/pepa

- `POST (Create)` – Slouží pro vytvoření dat

- `DELETE` – Používá se pro smazání dat

  ```
  DELETE /api/user/pepa
  Host: www.example.com
  ```

- `PUT (Update)` – Operace změny je podobná POSTu s tím rozdílem, že voláme konkrétní URI konkrétního zdroje, který chceme změnit, a v těle předáme novou hodnotu.

3.2.3 Slim Framework

Slim [2] je mikro framework, který pomáhá rychle vytvářet jednoduché, ale výkonné webové aplikace a API. V podstatě je Slim dispečer, který přijímá HTTP požadavek, vyvolá vhodné volání a vrací HTTP odpověď, to je výhodné zejména při použití REST. Slim na rozdíl od jiných frameworků je velmi rychlý a velmi malý, takže práce s ním je velmi jednoduchá.

[4] https://github.com/Olicek/GoogleMapAPI

Kapitola 4

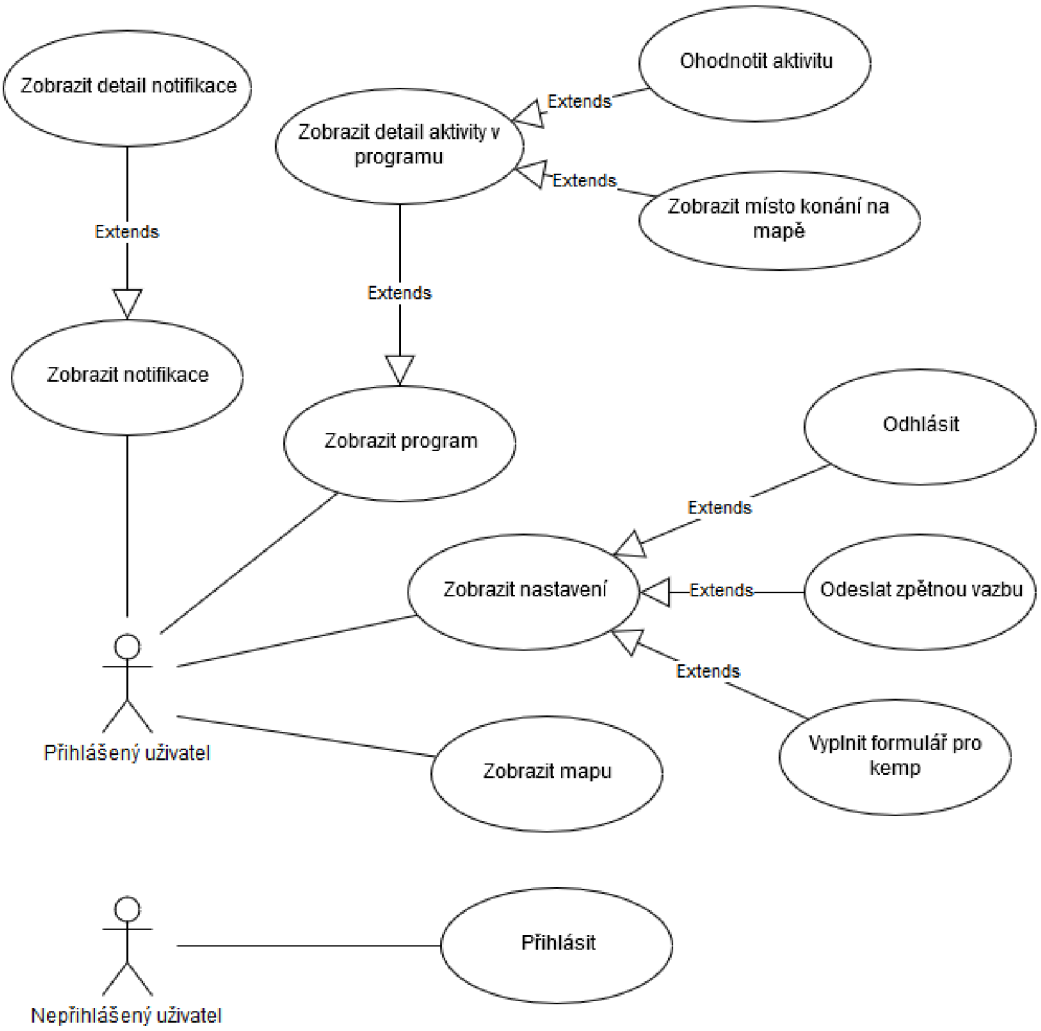
Návrh

Tato kapitola se zabývá návrhem výsledného řešení na základě analýzy provedené v kapitole 2. Nejdříve je popsán návrh mobilní aplikace, kde se popisuje návrh uživatelského rozhraní doplněný diagramem případu užití, následně je popsáno testování uživatelského rozhraní, a nakonec je návrh funkčnosti celé aplikace. Návrh webové aplikace je popsán obdobně. Dále je také popsán návrh komunikace mezi serverem a mobilní aplikací, co je podstatnou částí této práce a na závěr je popsán návrh databáze, ve které si aplikace bude ukládat data.

Návrhu bylo věnováno podstatné množství času kvůli nutnosti udělat aplikace dostatečně intuitivní a jednoduché. Špatně navržená aplikace by se odrazila na použitelnosti aplikace.

4.1 Mobilní aplikace

Jak již bylo zmíněno, mobilní aplikace by měla zobrazovat notifikace, celý program a mapu lokací. Výslednou funkcionalitu jde nejlépe popsat diagramem případů užití na obrázku 4.1. Ve srovnání s již popsanou analýzou problému se pouze přidalo nastavení, kde je možnost odeslat zpětnou vazbu a odeslání formuláře pro kemp. Je tam také případ užití přihlášení, ovšem přihlášení je po prvotním spuštěním aplikace povinné a dále při odhlášení, uživatel tedy nemůže dál v aplikaci pokračovat, aniž by byl přihlášený.



Obrázek 4.1: Diagram případů užití mobilní aplikace.

4.1.1 Návrh uživatelského rozhraní

Uživatelské rozhraní následuje směrnice Material Designu popsané na straně 13, tedy je kladen důraz na univerzální ovládání a vzhled, který uživatel najde ve většině moderních aplikací.

Navigace

Základem aplikace je navigace, na kterou je kladen velký důraz a měla by obsahovat ty nejzákladnější sekce v aplikaci, v našem případě notifikace, program a mapa.

Navigace může být tvořena pomocí následujících základních vzorů:

- Embedded navigation – Pro aplikace, které mají jedno hlavní okno, např. kalkulačka.
- Tabs (Panely) – Pro aplikace, u kterých je vhodné často přepínat mezi panely, lze jen pro pár oken.
- Bottom navigation bar (Panel s dolní navigací) – Stejné využití jako v případě Tabs, ale je výhodnější pro mobilní zařízení, protože je v dolní části také navigace zařízení.

- Navigation drawer – Pro aplikace, které mají několik oken a není vhodné je zobrazovat přes jiné druhy navigace.

Jedná se pouze o výběr těch nejběžnějších, jelikož druhů navigací je mnoho. Pro tuto práci je nejvíce vhodná navigace s dolním panelem.

Toolbar

Toolbar je hlavní panel aplikace, většinou v horní části aplikace, obsahuje název aplikace nebo i obrázek, nebo v případě Navigation drawer obsahuje ikonku hamburgeru. Označuje se taky jako Action Bar.

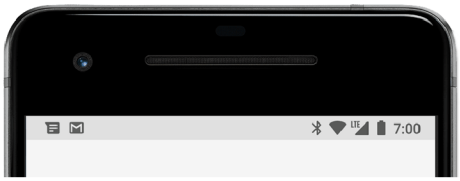
Tělo aplikace

Navigace a Toolbar jsou v aplikaci pevné části, které se v aplikaci zobrazují pořád. Takzvané tělo aplikace již ale tvoří daná sekce, kterou si uživatel může zvolit.

Notifikace a Status bar

Notifikace je zpráva, která se zobrazuje mimo uživatelské rozhraní aplikace ze které je poslána. Používá se pro upozornění nebo připomínku nebo jinou časově naplánovanou informaci z aplikace. Při kliknutí na notifikaci je uživatel přenesen do aplikace [6].

Status bar je horní proužek, který obsahuje základní informace o zařízení jako čas či stav baterie. V levé části se pak zobrazují ikony notifikací. Uživatelé můžou přejet dolů po Status baru a to otevře **Notification drawer**, který obsahuje detail všech zobrazených notifikací. Každá notifikace může mít seznam akcí, například odpověď na přijatý email.



Obrázek 4.2: Uživateli se notifikace nejdříve ukáže ve Status baru jako ikona.

4.1.2 Návrh funkce aplikace

Při prvotním spuštění aplikace se po uživateli žádá, aby se přihlásil – Přihlásit se je možné přes Facebook, Google anebo přímo přes účet Seznamováku. Poté se aplikace dotazuje na server, zda je uživatel zaregistrovaný, jinak podá zpětnou vazbu o neúspěchu. Po úspěšném přihlášení se uživateli zobrazí seznam notifikací.

Notifikace

Seznam notifikací je realizován kontejnerem, který je možné posunovat a každá notifikace je vizuálně oddělena oddělovačem. U každé notifikace je možné nastavit barvu (červenou a modrou) pro rozdělení důležitosti. Ilustrace jak by měl seznam notifikací v aplikaci vypadat je zobrazena na obrázku 4.3.

Obrázek 4.3: Každá notifikace obsahuje nadpis, který notifikaci vystihuje. Text notifikace obsahuje popis notifikace, tento text může být libovolně dlouhý a na více řádků. Každá notifikace obsahuje i čas, na který notifikace upozorňuje.

Každá notifikace obsahuje i čas upozornění, kdy aplikace odešle notifikaci, která se objeví ve Status baru. V Notification draweru se pak nachází nadpis notifikace a text notifikace. Po kliknutí na notifikaci se otevře samotná aplikace.

Program

V horní části by se měl nacházet panel s vybraným dnem. Samotný program je pak vykreslený pomocí časových bloků, kde každý má přiřazenou barvu a název aktivity, jako je znázorněno na obrázku 4.4. V levé části programu se nachází časová osa. Začátek programu by měl začínat v 7 hodin ráno a končit v noci.

Program je možné posunovat do čtyř směrů, vše ale záleží podle velikosti displeje zařízení. Pokud výška displeje přesahuje výšku samotného programu, pak je program protažen na výšku, jinak je nutné posunovat zobrazení nahoru a dolů. Naopak pokud se v programu nachází příliš vysoký počet aktivit, kdy už není možné aktivity čitelně vykreslit vedle sebe, tak se zobrazení posunuje do stran. Každá aktivita v programu má přiřazený čas začátku a konce, podle toho se odvodí výška časového bloku, šířka se vypočítá podle počtu aktivit, které současně běží s danou aktivitou. Pokud je v programu vybraný aktuální den, tak je zobrazena i čára indikující aktuální čas, společně se zašednutím již uplynulé části programu.

Obrázek 4.4: Grafický návrh programu v mobilní aplikaci. Dole lze vidět Bottom navigation bar popsaný na straně 18, nalevo je časová osa, kde hned napravo od ní se zobrazují bloky programu. V horní části aplikace je Toolbar popsaný na straně 19 s logem Seznamováku. Pod Toolbarem se nachází přepínání zobrazeného dne v programu.

Po kliknutí na aktivitu se zobrazí nové okno s detailem samotné aktivity. Nachází se zde popis aktivity, místo konání společně s tlačítkem pro zobrazení na mapě a také hodnocení v podobě pěti hvězd.

Mapa

Mapa pokrývá celý prostor těla aplikace, kde je zobrazen areál kempu společně s omezením možností posunu po mapě pro větší přehlednost. Mapa by měla zobrazovat pouze terén, nikoli interní popisky různých míst, které se nachází v kempu. Zároveň by měla mít vlastní barevné schéma, které by jasně odlišovalo objekty na mapě. Cesty by měly být šedé, přírodní krajina zelená, stavby oranžové a voda modrá. Vše ostatní by se nemělo zobrazovat. Mapa by měla obsahovat také několik bodů, které značí některá významná místa pro účastníky. Po kliknutí na značku se zobrazí název místa. Příklad těchto značek je znázorněn na obrázku 4.5.

Obrázek 4.5: Příklad značek, které se budou zobrazovat na mapě. Značka nalevo značí budovu s bungalovy, značka uprostřed znázorňuje kde se konají společné občerstvování a značka napravo značí vybranou lokaci, kterou si uživatel vyhledal v detailu aktivity.

Nastavení

Nastavení aplikace by mělo přejímat již zavedený styl, který lze najít ve většině aplikacích. Tento styl se vyznačuje seznamem položek, kde každá položka znázorňuje nějaké nastavení. Každá položka by měla mít na levé straně text akce a pod tímto textem případně i popis akce. Na pravé straně by se měla nacházet akce, pokud tomu odpovídá kontext, například u povolování notifikací by byl přepínač. Položky by měly být vizuálně odděleny nadpisy podle jejich obsahu. V nastavení aplikace by měly být položky pro vypnutí ohlášení notifikací nebo vypnutí vibrací. Dále možnost výběru intervalu aktualizace dat a také možnost pro okamžitou aktualizaci. Potom formuláře a odhlášení.

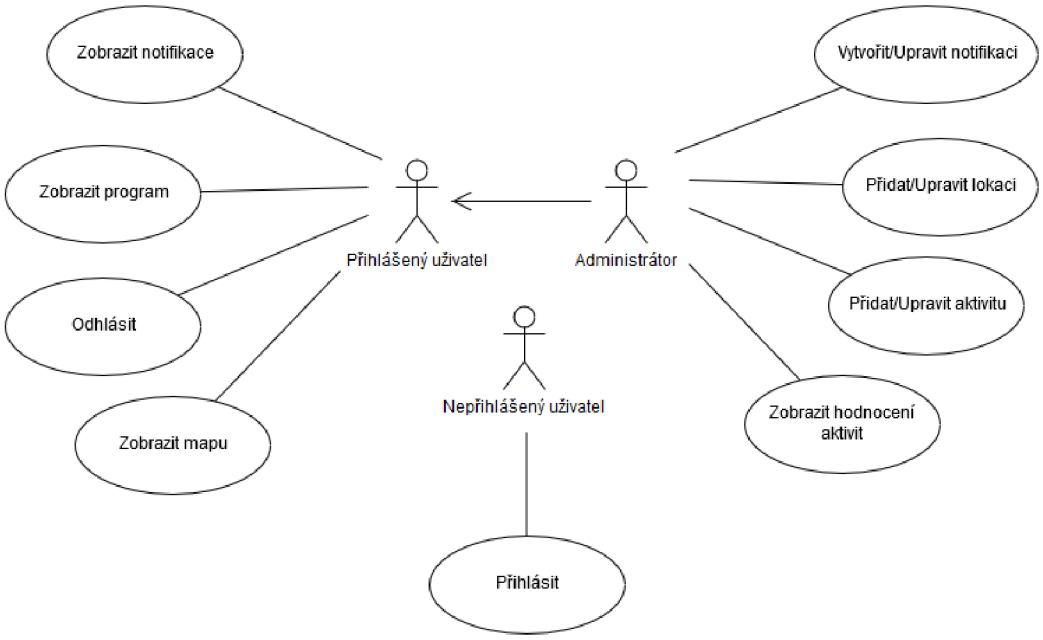
4.1.3 Testování uživatelského rozhraní

Pro ověření, zda je uživatelské rozhraní opravdu intuitivní a jednoduché bylo několik budoucích uživatelů požádáno o vyzkoušení aplikace a zhodnocení vzhledu aplikace. Uživatelé se v aplikaci zorientovali téměř ihned, některým se nelíbilo zvýraznění notifikací nebo se neshodli na volbě barev v mapě, ale ve výsledku testování dopadlo velmi kladně.

4.2 Webová aplikace

Z analýzy vyplývá, že webová aplikace by měla sloužit pro administraci dat pro mobilní aplikaci. Nicméně vzhledem k absenci iOS verze a ostatních systémů se nabídlo vytvoření kopie mobilní aplikace pro web jako náhrada. Touto změnou se aplikace rozděluje na dvě části: Administrační a informační. Administrační slouží pro editaci dat, které pak vidí uživatelé na webu či na mobilním zařízení. Informační slouží pro zobrazení dat, zde by se obsah neměl lišit od obsahu v mobilní aplikaci.

Funkcionalitu administrační části lze opět nejlépe popsat diagramem případů užití na obrázku 4.6.



Obrázek 4.6: Diagram případů užití administrace webové aplikace. Vyskytují se zde tři aktéři, přihlášený uživatel má stejné možnosti jako v mobilní aplikaci, administrátor má přístup do všech sekcí a může je upravovat.

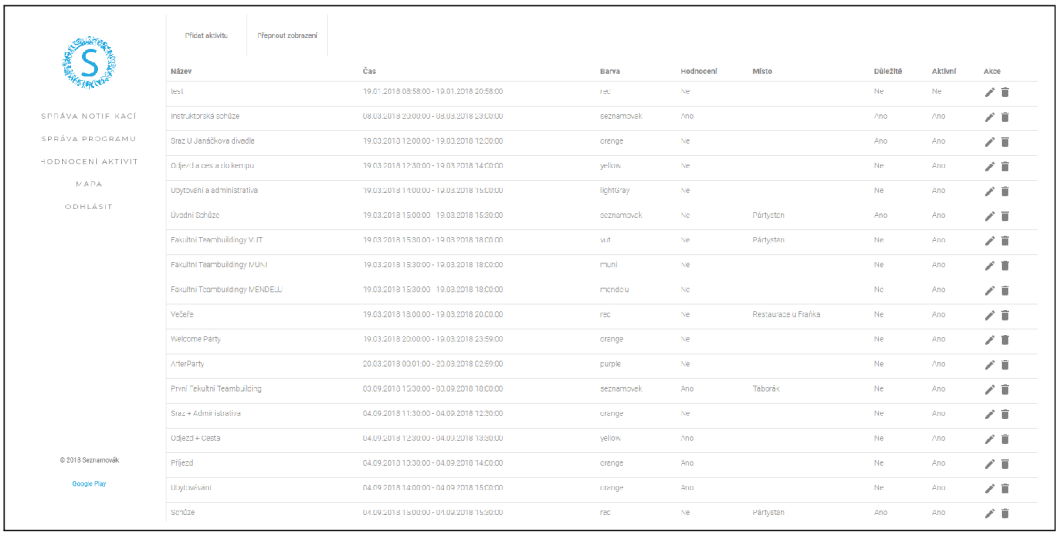
4.2.1 Návrh uživatelského rozhraní

Vzhled administrace následuje vzhled v mobilní aplikaci, je kladen důraz na jednotnost vzhledu a zobrazení i v prohlížeči na mobilních zařízeních [11]. Na levé straně stránky se nachází panel hlavní navigace v aplikaci, zde je hlavní rozdíl oproti mobilní aplikaci, neboť položky jsou zobrazeny jako seznam. Zbytek stránky pak tvoří tělo aplikace.

Tělo aplikace

V administrační sekci je tělo aplikace převážně tvořeno tabulkou obsahující relevantní informace dle zvolené sekce v navigaci. Každý řádek tabulky má na pravé straně tlačítka pro editaci nebo smazání záznamu, jak je znázorněno na obrázku 4.7. Nad tabulkou se nachází panel operací, který obsahuje možnosti jako vytvoření nového záznamu nebo přepnutí zobrazení mezi informační a administrační verzí, to se převážně hodí v situaci kdy administrátor může zkontrolovat, zda se informace uživatelům korektně zobrazují. Tento styl se nachází ve všech sekcích. V případě mapy je navíc nad tabulkou zobrazena mapa.

Tělo aplikace informační sekce se liší od zvolené sekce v navigaci, nicméně se vzhledem neliší od mobilní aplikace.



Obrázek 4.7: Návrh administrační sekce aplikace. Nalevo se nachází navigace pro administraci hlavních sekcí. Tělo aplikace obsahuje tabulku, která zobrazuje výpis všech aktivit v programu, u každé z nich jsou na pravé straně operace editace a odstranění záznamu. Lze zde také vidět horní panel obsahující akce již popsané výše.

4.2.2 Návrh funkcionality systému

Při prvním načtení systému se uživateli zobrazí přihlašovací stránka, zde se musí přihlásit svými údaji (ostatní uživatelé mimo Seznamovák se tedy do systému nedostanou), které má uložené v systému pro Seznamovák, který obsahuje všechny informace o účastnících. Tento administrační systém je se systémem Seznamováku propojen. Po úspěšném přihlášení se uživateli zobrazí již samotná data. Pokud se jedná o administrátora, zobrazí se odpovídající informace dle zvolené položky v navigace. V menu je pro administrátora navíc dostupná sekce o hodnocení aktivit, kterou tvoří tabulka všech ohodnocených aktivit, které jsou sjednocené do seznamu počtů hvězdiček a průměrného ohodnocení.

Pokud se do systému přihlásí účastník, zobrazí se mu stejně interpretovaná data, které jsou popsány v sekci o mobilní části.

4.3 Návrh komunikace mezi serverem a mobilní aplikací

Jelikož klíčová vlastnost aplikace je stahování informací ze serveru, je nutné uvést jak spočívá návrh komunikace mezi aplikací a serverem. Server bude implementovat REST rozhraní, takže se převážně bude jednat o `GET` a `POST` požadavky. Komunikace by měla být řízena pomocí stavových kódů, kdy `200` je pro úspěch, `404` nenalezeno a `422` značí, že se stala chyba. Odpovědi s chybovými kódy by měly být doplněny slovním popisem co se stalo špatně. Data by aplikace měla stahovat v určitém intervalu, který si uživatel bude moci změnit v nastavení. V případě odesílání dat z aplikace by to mělo být provedeno ihned, jelikož se nejedná o pravidelné aktualizace.

Intervaly aktualizace jsem vybral *15 minut, 30 minut, hodina, dvě hodiny a 8 hodin.* Je možné, že se do Seznamováku tyto informace upraví, jelikož je neurčité jak se to v ostrém provozu projeví.

Mimo jiné si uživatel může informace stáhnout sám pomocí tlačítka v nastavení, které ho upozorní, že data byla stažena.

Ze serveru si aplikace bude načítat následující:

- Seznam notifikací
- Program
- Seznam lokací na mapě

Naopak z mobilní aplikace se budou na server odesílat následující údaje:

- Údaje pro přihlášení – Při přihlášení, ať už přes sociální sítě nebo Seznamovák je nutné nejdříve odeslat údaje na server a čekat na odpověď pro ověření, jestli bylo přihlášení úspěšné.
- Formuláře – Po vyplnění formuláře se informace odesílají na server
- Hodnocení aktivit – Po ohodnocení aktivity v programu se hodnocení odešle na server

V případě odesílání formulářů a hodnocení je nutné si dát pozor na situaci, kdy zařízení nemá internet. Jednou z variant může být fronta požadavků na odeslání na server, které se odešlou v momentě, kdy je zařízení již připojené k internetu.

4.4 Návrh databáze

Pro ukládání informací je nutné zavést databázi, která bude informace ukládat v podobě tabulek. Na serveru se bude nacházet centrální databáze. Nicméně je důležité, aby mobilní aplikace fungovala i bez internetového připojení, proto musí mít i ta svoji databázi, která bude mírně ořezána. Mimo tyto databáze bude ještě využívána databáze systému Seznamováku, ze které si aplikace čte uživatelské účty pro ověření přihlášení.

Centrální databáze bude obsahovat pět tabulek. Tabulky budou pro notifikace, aktivity programu, lokace, hodnocení a přihlášení přes sociální sítě. Tabulky pro notifikace a aktivity programu jsou popsány v tabulkách 4.1 a 4.2.
Tabulka lokací obsahuje zeměpisné údaje a název ikony, která se na mapě zobrazí jako značka, dále tabulka hodnocení obsahuje identifikátor aktivity, ohodnocení a identifikátor uživatele a tabulka sociálních údajů obsahuje identifikátor účtu, email podle kterého se páruje s již existujícím účtem a typ účtu.

Databáze mobilní aplikace bude obsahovat tabulky pro notifikace, aktivity programu, lokace a hodnocení. Zde je rozdíl oproti centrální databázi pouze u hodnocení, kde jsou záznamy jenom o hodnoceních uživatele a ne všech uživatelů.

Název sloupce	Datový typ	Popis
id	INTEGER, PRIMARY KEY	unikátní klíč notifikace
title	TEXT	Nadpis notifikace
body	TEXT	Text notifikace
type	INTEGER	Typ notifikace (Běžná, sociální nebo důležitá)
remindAtTime	TIMESTAMP	Čas upozornění uživatele na notifikaci
time	TIMESTAMP	Čas události na kterou je notifikace

Tabulka 4.1: Schéma a popis sloupců tabulky pro notifikace

Název sloupce	Datový typ	Popis
id	INTEGER, PRIMARY KEY	unikátní klíč aktivity
name	TEXT	Název aktivity
description	TEXT	Popis aktivity
color	TEXT	Textová reprezentace barvy
startTime	TIMESTAMP	Čas začátku aktivity
endTime	TIMESTAMP	Čas konce aktivity
canBeRated	INTEGER	Příznak, zda se aktivita může hodnotit
location	INTEGER	Unikátní klíč lokace
isImportant	INTEGER	Příznak, zda se jedná o důležitou aktivitu. V programu je pak speciálně vyznačena.
isActive	INTEGER	Příznak, zda má být aktivita aktivní

Tabulka 4.2: Schéma a popis sloupců tabulky pro aktivity programu

Kapitola 5

Implementace

Tato kapitola se zabývá implementací návrhu popsaného v minulé kapitole. Nejdříve se rozebírá mobilní aplikace a jak byly implementovány jednotlivé sekce aplikace společně s výsledky jak aplikace vypadá. Následně se stejným způsobem rozebírá i webová aplikace. Dále je rozebráno jak bylo implementováno API pro REST a na závěr je popsáno testování a spotřeba mobilní aplikace.

5.1 Mobilní aplikace

5.1.1 Implementační prostředí

Aplikace je implementována v programovacím jazyce Java, přesněji Android Java ve vývojovém prostředí Android Studio 3 popsaném na straně 12. Aplikace je sestavena systémem Gradle nakonfigurovaném souborem *build.gradle*, každý Android projekt obsahuje dva konfigurační soubory, jeden v kořenovém adresáři projektu, který slouží pro celý projekt (to se hodí při více modulech v projektu) a druhý ve složce app, který je pouze pro daný modul. V rámci této práce byl upravován pouze ten ve složce app.

Listing 5.1: Ukázka konfiguračního souboru pro sestavení

```
android {
    compileSdkVersion 27
    buildToolsVersion '27.0.3'
    defaultConfig {
        applicationId "org.seznamovak.seznamovak"
        minSdkVersion 17
        targetSdkVersion 27
        versionCode 9
        versionName '1.0.2'
    }
    buildTypes {
        debug {
            ext.enableCrashlytics = false
            debuggable true
        }
        release {
            minifyEnabled false
```

```
            proguardFiles getDefaultProguardFile('proguard-
                android.txt'), 'proguard-rules.pro'
            debuggable false
        }
    }
}
```

Nachází se zde základní informace o aplikaci, se kterými pak pracuje obchod Google Play při publikování aplikace. Jsou zde informace jako `compileSdkVersion`, která značí jaké Api aplikace využívá, `buildToolsVersion` značí verzi SDK balíčku, `defaultConfig` zapouzdřuje nastavení pro všechny varianty sestavení aplikace, `applicationId` značí unikátní identifikaci, pod kterou pak bude aplikace k nalezení na Google Play, `minSdkVersion` značí minimální verzi Api, pod kterou si lze aplikaci nainstalovat. `targetSdkVersion` určuje úroveň Api používanou k otestování aplikace. Poslední dva parametry slouží pro identifikaci verze aplikace, `versionCode` je unikátní číslo, které musí být při aktualizaci vyšší než předchozí vydání a `versionName` zastupuje verzi aplikace, která je viditelná pro uživatele na Google Play. `buildTypes` slouží pro konfiguraci variant sestavení aplikace, výchozí volbou je debug a release. Debug slouží pro účely vývoje a release pro vydání aplikace. V ukázce kódu je v debug variantě vypnutí CrashLytics popsané v použitých knihovnách v kapitole 3.1.6 a nastavení parametru `debuggable`, což umožňuje s aplikací pracovat v Android Studiu. V release sekci se nachází parametr `minifyEnabled` pro zmenšení velikosti zdrojových kódů a `proguardFiles` slouží k obfuskaci tříd při sestavování aplikace pro znesnadnění čtení zdrojového kódu [4].

Další klíčovou části v konfiguračním souboru je blok `dependencies`, kde jsou nalinkovány všechny použité knihovny v projektu. Nachází se mezi nimi již popsané knihovny Crashlytics, LeakCanary, Retrofit a Firebase JobDispatcher. Dále také základní AppCompat, Design a Support knihovny, knihovny pro přihlášení přes Facebook a Google, knihovny pro mapu a knihovnu ThreeTen Android Backport[1] pro práci s časem.

Dalším důležitým souborem v projektu je *AndroidManifest.xml*, který obsahuje reference na všechny aktivity, služby a meta-data. Jsou zde uvedeny také různá oprávnění, které aplikace potřebuje a které pak uživatel při instalaci aplikace musí schválit. V našem případě jsou práva pro internet, polohu a zjištění připojení. `Application` element tvoří základ aplikace, nastavují se zde atributy pro ikony a grafické téma. Grafické téma bylo zvoleno světlé *Theme.AppCompat.Light.NoActionBar*. Samotné téma z velké části vyhovuje návrhu, takže nebylo třeba něco měnit. Byla použita varianta bez Action Baru, protože na toto místo byl vložen prvek `Toolbar`, který více vyhovuje popsanému návrhu.

Dále se nacházejí meta-data, kde jsou vypsány důležité klíče a hesla pro externí knihovny – v ukázce je klíč pro Google Maps API. Následuje seznam všech aktivit, u každé je fixně nastavena orientace, jelikož se nenašlo využití zobrazení obsahu na šířku. A nakonec jsou služby, které aplikace může spouštět, bez uvedení služby v tomto souboru se služba nespustí.

[1] https://github.com/JakeWharton/ThreeTenABP

Listing 5.2: Ukázka manifest souboru

```
<manifest xmlns:android="http://schemas.android.com/apk/res/
   android"
    package="org.seznamovak.seznamovak">

    <uses-permission android:name="android.permission
    .INTERNET" />
    <uses-permission android:name="android.permission
    .ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission
    .ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission
    .READ_PROFILE" />

    <application
        android:name=".Seznamovak"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher_round"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
                ...
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" />
        ...
        <service
            android:name=".Notification.NotificationJobService
               "
            android:exported="false">
            <intent-filter>
                <action android:name="com.firebase.
                   jobdispatcher
                .ACTION_EXECUTE" />
            </intent-filter>
        </service>
```

5.1.2 Souborová hierarchie

V kořenovém adresáři projektu se nachází adresář `app`, který obsahuje hlavní modul aplikace. Dále adresáře `build` a `gradle`, které jsou vygenerovány pomocí Android Studia, vyskytuje se zde také několik souborů, např. `gradle.build` pro celý projekt.

V adresáři `app` jsou adresáře `build`, který obsahuje výstupy sestavení vygenerovaná pomocí Android Studia, `libs` pro privátní knihovny (v rámci této práce nevyužito) a `src`, kde jsou umístěny všechny zdrojové soubory projektu. Krom adresářů je zde několik souborů, avšak je třeba zmínit pouze jediný relevantní a to další `gradle.build`, tentokrát pro daný modul, který byl dopodrobna popsán výše. V adresáři `src/main` se nachází hlavní rozdělení zdrojových souborů a také *AndroidManifest.xml*. Adresář `java` obsahuje všechny zdrojové kódy a `res` obsahuje všechny ostatní soubory potřebné v aplikaci.

Adresář `res` obsahuje několik podadresářů – drawable, layout, menu, mipmap, raw, values a xml.

- drawable – bitmapové soubory nebo XML soubory, které obsahují grafické prvky, např. gradient nebo ohraničení objektu.
- drawable-xxx – Z důvodu existence zařízení různých velikostí a hustoty pixelů se zavedlo rozdělení na 5 základních rozlišení – hdpi, mdpi, xhdpi, xxhdpi a xxxhdpi. To je na Android platformě realizováno pomocí názvu složek, kde rozlišení je suffixem názvu složky odděleným pomlčkou, tedy například drawable-xxhdpi.
- layout – XML soubory, které definují vzhled uživatelského rozhraní.
- layout-v21 – To stejné jako layout, jenom pro zařízení s API stejné nebo vyšší než verze 21. Tímto způsobem lze udržovat unifikovaný vzhled na všech API. V případě této práce to bylo z důvodu určitých prvků Material Designu, který nastoupil až právě v API 21.
- menu – XML soubory definující hlavní menu v aplikaci
- mipmap-xxx – drawable soubory spouštěcích ikon aplikace.
- raw – Libovolné soubory uložené v originální podobě.
- values – XML soubory, které obsahují hodnoty jako řetězce, barvy a styly.
- xml – XML soubory, které se jinam nevešly.

5.1.3 Databáze

Třída, která se stará o vytvoření a práci s databází se nazývá `DatabaseHandler` a dědí z `SQLiteOpenHelper`. V konstruktoru třídy se nachází volání na konstruktor dědící třídy, kde jsou parametry pro název databáze a verzi databáze, pokud se od zavedení databáze mění návrh, tak je po každé změně nutné zvýšit verzi databáze, aby proběhla aktualizace ve formě provedení metody `onUpgrade()`. V této metodě se smažou všechny tabulky a v metodě `onCreate()`, která se zavolá hned poté co se registrují jednotlivé tabulky popsané v návrhu aplikace na straně 25. Další metody v této třídě slouží pro manipulaci s obsahem jednotlivých tabulek.

5.1.4 Komunikace se serverem

Požadavky

Samotná komunikace se serverem je realizována pomocí knihovny *Retrofit2* popsané v sekci práce se sítí 3.1.3. Třída pro správu této knihovny se nazývá `Api`. Zde se vytváří

instance Retrofit knihovny podle návrhového vzoru Singleton v metodě `getClient()`. Metoda `getAPIService()` slouží pro získání volání z rozhraní `ApiService`, kde příklad vypadá následovně

```
@GET("events")
Call<List<CalendarResponse>> getEvents();
```

kde `@GET` značí typ požadavku a `"events"` určuje kam se má požadavek poslat.

Na další ukázce lze vidět funkci pro získání všech aktivit programu, kde se nejdřív získá instance `ApiService` a dále se zařadí požadavek do fronty a asynchronně se po přijetí odpovědi zavolá funkce `onResponse()`, kde se přes Callback posílá výsledek. Jelikož není praktické mít tyto funkce přímo v aktivitách, kde se s danými daty pracuje, tak je nutné vytvořit Callback, přes který se výsledek asynchronního volání přenese do aktivity. Funkce `onFailure()` se zavolá pouze když nedorazí žádná odpověď.

Listing 5.3: Ukázka Api volání

```
public static void getEvents(final CalendarCallBack
   calendarCallBack) {
    ApiService client = getAPIService();
    client.getEvents().enqueue(new Callback<List<
       CalendarResponse>>() {
        @Override
        public void onResponse(Call<List<CalendarResponse>>
           call,
        Response<List<CalendarResponse>> response) {
            calendarCallBack.successful(response.body());
        }

        @Override
        public void onFailure(Call<List<CalendarResponse>>
           call,
        Throwable t) {
            calendarCallBack.fail(t);
        }
    });
}
```

Pravidelné aktualizace

Jelikož je nutné pravidelně se dotazovat serveru na aktualizace dat, tak je potřeba zavést nějakou službu, která bude běžet na pozadí a vždy v daném intervalu zavolá funkce pro stažení dat, lze použít např. `AlarmManager`. Nicméně zde se nabízí použití knihovny *Firebase JobDispatcher* popsané v sekci 3.1.6, která běží na pozadí a je optimální v rámci spotřeby baterie.

Hlavní třídou, která se stará o spuštění služby je `ApiDownloaderJobDispatcher`, která obsahuje metodu `CreateJob()`, která nejdříve z nastavení aplikace získá interval aktualizace a poté následuje konfigurace úkolu. Nejdříve se zvolí, která služba se má spouštět, v tomto případě se jedná o `ApiDownloaderJobService`, která stáhne všechna data pomocí asynchronního vlákna `AsyncTask` a uloží je do databáze. Následně se nastaví, že se má úkol opakovat, jeho životnost na napořád, dále časové okénko provedení, aby se úkol provedl zrovna kdy se to systému nejvíce hodí. Nakonec se nastaví přepsání stejného naplánovaného úkolu, pokud již úkol byl naplánovaný a omezení, že zařízení musí být připojené k internetu, to je obzvlášť užitečné, neboť v případě kdy uživatel zrovna v časovém oknu provedení není připojen k internetu, tak se úkol provede v momentě, kdy se připojí k internetu a nedojde tedy k vynechání aktualizace. Po konfiguraci se úkol naplánuje.

5.1.5 Sekce aplikace

Jádro aplikace

`Application` je hlavní třídou, která se spustí při spuštění aplikace před jakoukoli aktivitou. Zde lze umístit část kódu, která se má provádět pouze jednou. Je tam umístěna inicializace *Crashlytics, LeakCanary, inicializace knihovny pro práci s časem a nastavení ApiDownloaderJobDispatcher* pro stahování dat ze serveru.

Další důležitou aktivitou je `MainActivity`, která se stará o toolbar a spodní navigaci. V případě toolbaru přidává přepínání do nastavení, jelikož je ikonka ozubeného kolečka vložena jako obrázek. Navigace se stará o přepínání mezi třemi fragmenty, které pak zobrazují příslušný obsah. Důvod zvolení fragmentů je z hlediska konzistence zobrazení, jelikož celá obrazovka při přepnutí položky v navigaci se měnit nepotřebuje, ke změně dochází pouze v těle aplikace. V rámci Material Design návrhu při stisknutí systémového tlačítka pro vrácení zpět je nutné kontrolovat zvolenou položku. Pokud se jedná o notifikace, která je hlavní položkou v navigaci, tak se aplikace přepne do pozadí, jinak se navigace přepne do notifikací a po opětovném stisknutí tlačítka dojde k přepnutí na pozadí. Nachází se zde také vnitřní třída `NetworkStatusReceiver`, která dědí z abstraktní třídy `BroadcastReceiver`. Tato třída kontroluje, zda je zařízení připojené k internetu a pokud ne, tak zobrazí panel, který uživatele upozorňuje, že není připojený k internetu, to je vidět na obrázku 5.1.

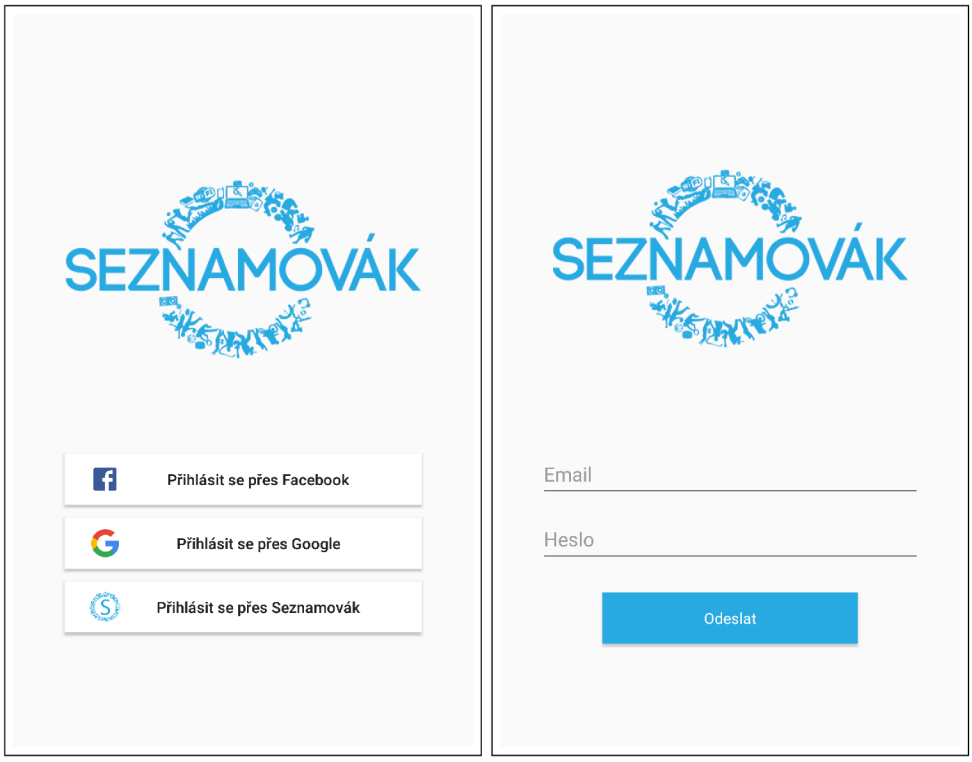
Obrázek 5.1: Na obrázku lze vidět aktivitu `MainActivity` s aktivním fragmentem `NotificationFragment`, která se stará o seznam notifikací. Je viditelný také panel, který značí, že zařízení není připojené k internetu.

Přihlášení

Přihlášení v aplikaci je prvotní akcí při spuštění aplikace, stará se o to aktivita `LoginActivity`. Nejprve je kontrola, zda je uživatel již přihlášený, u Facebooku se to provádí přes přístupový token, u ostatních je to potřeba zaznamenávat manuálně v nastavení aplikace.

V případě přihlášení přes Facebook a Google se sledují informace jako ID uživatele, podle kterého se bude nadále identifikovat v aplikaci a email, přes který se účet spojí se zaregistrovaným účtem na Seznamováku, odešle se tedy přes Api požadavek na server a čeká se na odpověď, pokud je úspěšná, tak to uživatele přenese do hlavní části aplikace, v jiném případě to uživateli zobrazí příslušnou hlášku pomocí prvku `Toast`.

Pokud se uživatel bude chtít přihlásit přes účet Seznamovák, tak aplikace uživatele požádá o email a heslo, tyto údaje jsou poslány na server a znovu se čeká na odpověď, kde je reakce stejná jako u ostatních metod. Přihlášení lze vidět na obrázku 5.2.



Obrázek 5.2: Přihlašovací stránka aplikace, kde si lze vybrat mezi třemi způsoby přihlášení. Při zvolení přihlášení přes Seznamovák se objeví formulář pro zadání údajů jak je znázorněno na pravém obrázku.

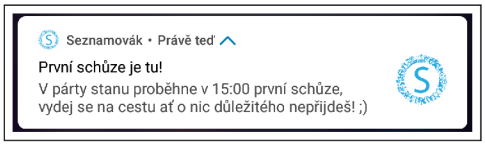
Notifikace

V momentě přepnutí navigace na notifikace se vytvoří fragment **NotificationFragment**. V rámci inicializace se získává seznam notifikací z databáze, pokud je výsledek prázdný, tak se aplikace pokusí notifikace stáhnout. To může nastat v případě prvního spuštění aplikace, kdy ještě neproběhla automatická aktualizace dat. Následně se odstraní notifikace, které ještě nebyly oznámeny, seřadí se a nastaví se adaptér `NotificationAdapter`, který je připojený na `RecyclerView`. V adaptéru se ještě nastavuje barva pozadí a také detekce kliknutí na položku. Po kliknutí na položku se vytvoří nový Intent popsaný v sekci 3.1.2 a vloží se do něj objekt notifikace pod klíčem "Notification", poté se nastartuje nová aktivita **NotificationDetailActivity**, která zobrazuje kompletní informace o notifikaci. Z důvodu jednoduchosti této aktivity nebude podrobněji rozebrána. Seznam notifikací lze vidět na obrázku 5.1.

Při stahování notifikací dochází také ke spuštění služby `Firebase JobDispatcher` 3.1.6, kterou nastavuje třída `NotificationJobDispatcher`. Zde se nastaví čas spuštění, který se počítá jako rozdíl aktuálního času od času upozornění v sekundách, dále že se má úloha naplánovat i po restartu zařízení, neopakuje se a případně přepsat již existující naplánovanou úlohu (v případě dřívějšího naplánování), ID naplánované úlohy neboli *Tag* odpovídá řetězci "seznamovak-notification-"a ID notifikace.

Při spuštění úlohy se spustí asynchronní vlákno `AsyncTask`, které na pozadí zavolá metodu `notifyUser()` pro vytvoření notifikace. Zde se vytvoří instance `NotificationManager`, která následně vyvolá notifikaci. Nejdříve se ale vytvoří notifikační kanál, pokud se jedná o zařízení s Android Oreo, poté se vytvoří instance `NotificationCompat.Builder`, která defi-

nuje samotný obsah notifikace jako kanál, který je nazván jako `"seznamovak-app-channel"`. Potom text, ikony, zvuk a vibrace, a nakonec se nastaví priorita notifikace na nejvyšší. Nakonec se zavolá metoda `notify()`, která vytvoří notifikaci, která je vidět na obrázku 5.3.



Obrázek 5.3: Notifikace, která se zobrazí v Notification drawer při roztažení Status baru.

Program

Při zvolení programu v navigaci se vytvoří fragment **CalendarFragment**. Stejně jako u notifikací probíhá načtení z databáze a pokud je výsledek prázdný, tak se aplikace pokusí data stáhnout. Dále se nastavuje detektor gest, který snímá swipování do stran a podle toho přepíná dny.

Následně se seznam aktivit seřadí podle data a vyfiltrují se pouze aktivity, které jsou v přítomný den nebo později.

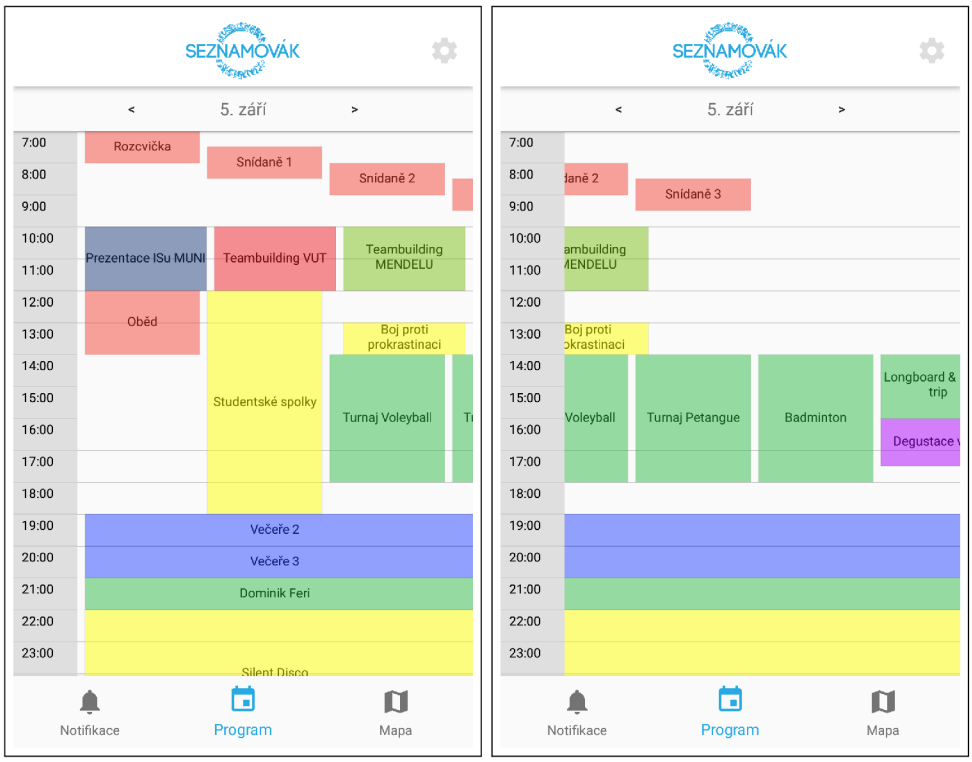
Jelikož je nutné, aby se v programu posunovalo do všech stran, bylo potřeba inteligentně zvolit strukturu rozvržení UI prvků. Konečné řešení spočívá ve vertikálním `ScrollView`, které zajistí posun nahoru a dolů. Uvnitř se nachází horizontální `LinearLayout`, které všechny své prvky umístí vedle sebe. Nachází se v něm `FrameLayout` a `HorizontalScrollView`, kde `FrameLayout` obsahuje časovou osu a `HorizontalScrollView` samotný program, který lze posunovat doprava a doleva. Díky tomu, že jsou tyto dva prvky vedle sebe, tak při posunování do stran je časová osa stále umístěna v levé části obrazovky.

Pro vykreslení časové osy byla vytvořena třída `HourView`. V metodě `onMeasure()`, která se vykonává ještě před samotným vykreslováním probíhá kontrola velikosti výšky displeje, jelikož při velkých rozlišeních by vznikalo volné místo, takže bylo potřeba vypočítat výšku jedné hodiny. Ve vykreslovací metodě `onDraw()` se nejdříve vykreslí pozadí a následně text hodin s oddělovači podle předdefinovaných rozměrů.

Pro vykreslení jednotlivých aktivit slouží třída `DayView`. V metodě `onMeasure()` probíhá kontrola, zda se v aktuálním dni nachází více aktivit ve stejný čas, že je potřeba program posunovat do stran. Podle toho se vypočítá šířka plátna a v případě vyššího rozlišení na výšku i výška jedné hodiny. Hlavní funkcí ve vykreslovací metodě `onDraw()` je `handleEvents()`, která postupně prochází každou aktivitu, kterou vykresluje. Pro každou aktivitu se vypočítá horní a spodní souřadnice, následně se vypočítá s kolika aktivitami se aktivita kříží – rekurzivně prochází aktivity konající se před danou aktivitou a po aktivitě. Podle toho se vypočítá levá a pravá souřadnice. Následně probíhá kontrola, zda text aktivity je delší než délka mezi levou a pravou souřadnicí, v tom případě se text vykresluje pomocí metody `drawText()`, která v rámci českého jazyka správně rozloží text na více řádků. Jinak se text vykreslí běžným způsobem. Po vykreslení se obdélníček znázorňující aktivitu uloží do kolekce `eventRects`. Pokud je zobrazený zrovna aktuální den, tak se vykreslí čára znázorňující čas a zešedne již uběhnutá část programu.

Každá aktivita lze rozkliknout pro dodatečné informace. Pro odchytávání kliknutí slouží metoda `onTouchEvent()`, která odposlouchává pohybové události, které lze vyčíst z objektu `MotionEvent`. Lze získat například polohu dotyku nebo typ akce (dotyk prstu, zvednutí nebo pohyb). V této metodě se sleduje dotyk, kdy v moment dotyku prstu se zaznamená čas dotyku a při zvednutí prstu se čas porovná s dobou stisku a pokud je menší než 200

ms, tak se jedná o kliknutí. Následně se prochází kolekce `eventRects`, kde se u každého obdélníčku kontroluje, zda souřadnice dotyku odpovídají souřadnicím obdélníčku.
V tom případě se vytvoří nový Intent a vloží se do něj objekt aktivity pod klíčem "CalendarEvent", poté se nastartuje nová aktivita **EventDetailActivity**.

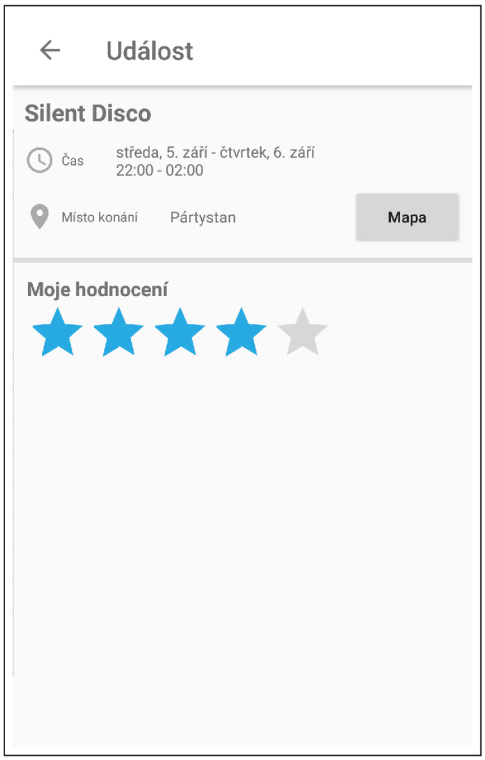


Obrázek 5.4: Znázorněný program, kde se od 14:00 kryje několik aktivit, které již není možné vykreslit bez rozšíření plátna do boku. Napravo je znázorněné, jak to vypadá když je zobrazení posunuto.

Detail Aktivity

Nejdříve se načte objekt aktivity pomocí klíče "CalendarEvent", dále se nastaví View a načte se hodnocení z databáze. Nastaví se detekce změny hodnocení, která se ihned převrhne do databáze. Hodnocení je realizováno pomocí prvku `AppCompatRatingBar`, který zajišťuje funkčnost i na starších zařízeních. Pokud aktivita má definované místo na mapě, tak je zobrazeno tlačítko pro ukázání na mapě. Při stisku tlačítka se vytvoří *Intent*, do kterého se přidá ještě integer pod klíčem "fragmentNumber", který značí mapu a řetězec "locationId". To všechno se nastavuje z toho důvodu, že není možné se přímo přepnout z aktivity na fragment jiné aktivity, takže je nutné nejdříve spustit **MainActivity**, která při definování zmíněných parametrů rovnou zobrazí mapu s červeně vyznačeným místem, kde se aktivita koná.

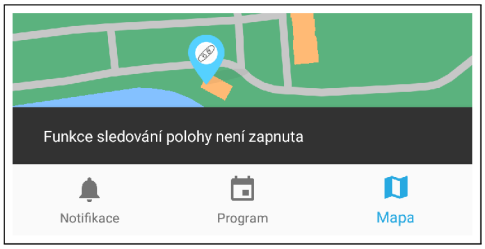
Ještě předtím než se aktivita ukončí je nutné odeslat hodnocení na server. Tedy v metodě `onDestroy()` je kontrola, zda bylo změněno hodnocení a pokud ano, tak se odesílá POST požadavek obsahující ID aktivity, hodnoty a ID uživatele na server. Detail aktivity je zobrazen na obrázku 5.5.



Obrázek 5.5: Detail aktivity v programu. Toolbar obsahuje tlačítko pro vrácení zpět. Pod názvem aktivity je zobrazený lokalizovaný čas. Vedle místa je tlačítko pro zobrazení místa na mapě. Hodnocení se mění tahem či dotykem na hvězdu.

Mapa

Při zvolení mapy v navigaci se vytvoří fragment **MapFragment**, který implementuje rozhraní `OnMapReadyCallback`. Po spuštění se zavolá metoda `getMapAsync()`, která asynchronně stáhne mapu. Následně ještě proběhne kontrola zapnuté GPS lokace, kdy se v případě vypnuté GPS v dolní části displeje objeví zpráva ve formě *SnackBaru*, který je zobrazen na obrázku 5.6.



Obrázek 5.6: Upozornění uživatele na vypnuté sledování polohy.

Dále se zkontroluje, zda má uživatel povoleno sledování polohy a případně se zobrazí žádost o povolení. V momentě, kdy je mapa stažena se vyvolá metoda `onMapReady()`, která má v parametru mapu, se kterou lze libovolně pracovat. Nejdříve se nastaví styl mapy, který byl vytvořen pomocí webové aplikace Google Maps APIs Styling Wizard[2], kde výstupem

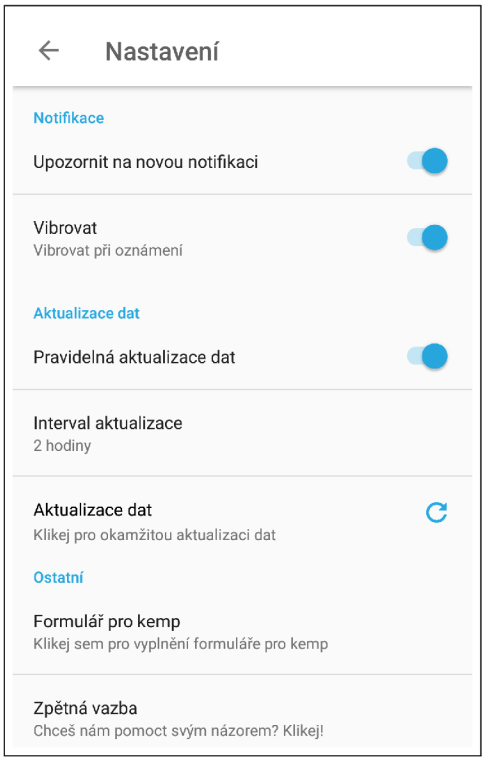
[2] https://mapstyle.withgoogle.com/

je JSON soubor. Ve stylu bylo nastaveno skrytí textů a omezení vykreslování pouze cest, terénu, budov a vody. Dále se nastavuje omezení rozsahu mapy pouze na kemp a přiblížení mapy. Nakonec se kamera mapy vycentruje na definovanou pozici a přidají se značky získané z databáze.

Obrázek 5.7: Mapa pokrývá celý prostor těla, na mapě jsou umístěny značky míst, nad kterými se při kliknutí zobrazí název.

Nastavení

O nastavení se stará aktivita **SettingsPrefActivity**, která vytváří fragment **MainPreferenceFragment** a který vychází z `PreferenceFragment`. V tomto fragmentu se načítá XML soubor umístěný v */res/xml/pref_main.xml*, který definuje vzhled prvků v nastavení. Dále se již nastavují pouze detektory kliknutí.



Obrázek 5.8: Nastavení aplikace.

5.1.6 Nasazení aplikace

Aplikace byla na Google Play[3] vydána již na začátku roku v alfa verzi, kdy byla podrobně testována jinými uživateli a byla kontrolována kompatibilita mezi různými zařízeními. O měsíc později byla aplikace posunuta do úplného vydání pro veřejnost ve verzi 1.0. Po drobných opravách je aplikace ve verzi 1.0.2 v době psaní této práce.

5.2 Webová aplikace

Webová aplikace byla vytvořena pomocí Nette Frameworku popsaném v sekci 3.2.1. Byl použit programovací jazyk PHP společně se značkovacím jazykem HTML a kaskádovými styly CSS. Aplikace je navržena pomocí návrhové architektury MVC (Model-View-Controller), která udržuje jednotlivé třídy čisté a přehledné. Byla vytvořena také MySQL databáze dle návrhu popsaného v kapitole 4.4.

Aplikace běží na serveru Seznamováku s vlastní subdoménou na adrese `https://app.seznamovak.org`. Spolupracuje také přímo s databází systému Seznamováku.

5.2.1 Souborová hierarchie

V kořenovém adresáři aplikace se nachází adresář `home`, který obsahuje celou aplikaci společně s Nette frameworkem, v ostatních adresářích se nachází obrázky, javascriptové soubory či kaskádové styly. Nachází se zde však také soubor `index.php`, který spouští samotnou aplikaci.

[3] https://play.google.com/store/apps/details?id=org.seznamovak.seznamovak

Uvnitř adresáře `home` se nachází 5 složek:

- `app` – Obsahuje všechny zdrojové kódy aplikace.
- `log` – Zde se shromažďují všechny logovací soubory jako *error.log*, *exception.log* a *info.log*.
- `temp` – Zde se ukládají dočasné soubory a cache stránek pro rychlejší načtení.
- `tests` – Testovací prostředí (v této práci to nebylo použito).
- `vendor` – Adresář obsahující všechny knihovny včetně samotného Nette frameworku.

Uvnitř adresáře `app` se nachází adresář `config`, která obsahuje dva konfigurační soubory s koncovkou `neon`[4]. Jedná se o `config.local.neon`, který obsahuje informace o databázi a `config.neon`, který značí kde jsou umístěny Presentery, rozšíření, služby apod.

Dalším adresářem je `model`, který jak už z názvu vypovídá obsahuje modely aplikace. Adresář obsahuje následující modely:

- `Authenticator` – Třída ověřující přihlášení uživatelů.
- `BaseRepository` – Třída pro stanovení základních funkcí, které pak využívají všechny dědící třídy.
- `EventsRepository` – Třída pro aktivity v programu.
- `NotificationsRepository` – Třída pro notifikace.
- `LocationsRepository` – Třída pro místa na mapě.
- `RatingsRepository` – Třída pro hodnocení aktivit.
- `Ratings` – Třída pro zaznamenání všech hodnocení pro danou aktivitu.

V tomto adresáři se nachází ještě adresář `DatePicker`, který je v této práci použit jako doplněk pro zobrazení výběru data při kliknutí na pole.

Dalším adresářem je `presenters`, který obsahuje následující soubory:

- `BasePresenter` – Stejně jako u modelu je také přítomna třída, která obsahuje obecné metody využitelné ve všech ostatních třídách. Lze zde najít metody pro získání formátu času nebo kontroly zda je uživatel administrátor.
- `EventPresenter` – Presenter pro práci s programem.
- `NotificationPresenter` – Presenter pro práci s notifikacemi.
- `MapPresenter` – Presenter, který načítá Google Maps API pro zobrazení mapy se značkami.
- `RatingPresenter` – Tento presenter se stará o hodnocení aktivit.

Poslední dva adresáře jsou `router`, který obsahuje soubor s routama a `templates`, který obsahuje všechny layout soubory *latte*.

[4]Neon je snadno čitelný formát pro serializaci dat.

Model

Pro přístup k informacím v databázi je třeba vytvořit model, který operuje jenom s informacemi z databáze a předává je dál. Seznam modelů již byl zmíněn výše, nicméně samotné modely se od sebe neliší, aspoň ty co pracují s databází. Převážně se v těchto třídách využívá `Database Exploreru` z Nette, který umožňuje práci s databází bez nutnosti použití SQL jazyka. Jako příklad je uvedena metoda pro získání všech aktivních aktivit programu v databázi.

Listing 5.4: Ukázka získání dat z databáze

```
public function getEvents() {
    return $this->getTable()->where("isActive = 1")->order('
       startTime ASC, endTime ASC');
}
```

View

Každá HTML stránka je vygenerována pomocí šablon `Latte`, které jsou integrovány přímo v Nette frameworku. Jde o jednoduchou syntaxi, která se od HTML moc neliší a určité prvky rozhodně zjednodušuje, například použití cyklů foreach. Obsah těchto šablon je pouze informační a primitivní, takže není důvod pro důkladnější popis.

Controller

Controllery jsou v Nette Presentery, které zpracovávají veškeré požadavky prohlížeče. Zde se také nachází veškerá logika této aplikace. Každý presenter má speciálně nazvané metody. Stránky, které zobrazují obsah musí mít funkci `renderStranka()`, aby systém věděl, že se na dané URL nachází opravdová stránka, jinak by byl uživatel přesměrovaný na chybovou stránku. Výchozí je `renderDefault()`. Všechny tyto stránky musí mít také stejně pojmenované *latte* soubory.

Dalším typem metod jsou akce, které jsou pojmenovány ve tvaru `actionStranka()`. Tyto metody se zavolají ještě před render metodou, která není povinná, ale v tom případě musí být v metodě přítomno přesměrování na jinou stránku.

Posledním typem jsou komponenty, které mají název ve tvaru `createComponentJmenoKomponenty()`. Nejčastěji se jedná o formulář, který nejčastěji bývá napsán ve tvaru `createComponentAddForm()`. Tato metoda vrací vytvořenou komponentu, jako je formulář. Ještě se uvnitř metody přidává callback `onSuccess`, který značí která metoda se má zavolat po odeslání formuláře.

Jako zpětnou vazbu uživateli se využívá metody `flashMessage()`, která může být zobrazena po přesměrování.

5.2.2 Sekce aplikace

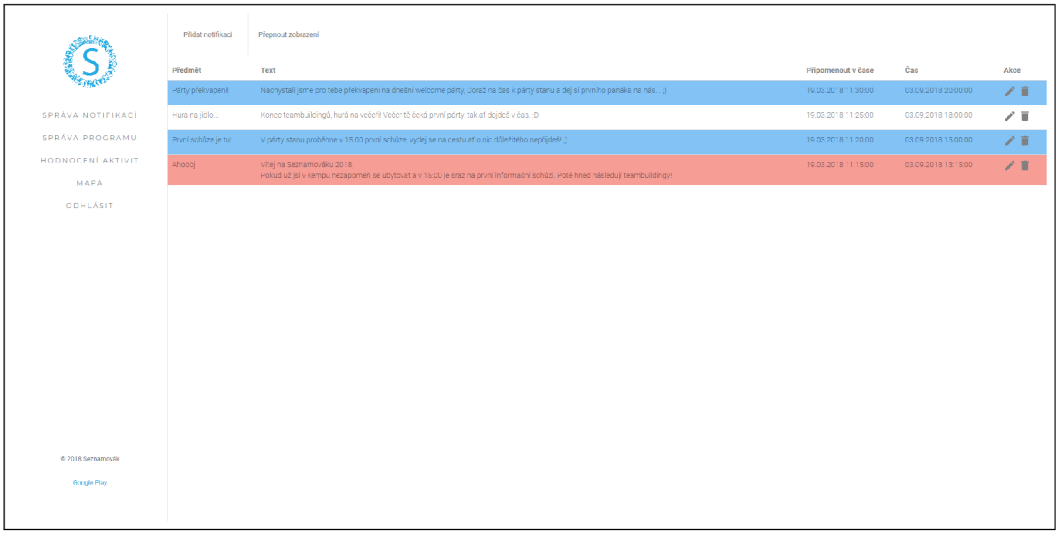
U notifikací, programu a mapy se mění zobrazení dle zvoleného módu – informační nebo administrační. Toto chování probíhá v metodě `renderDefault()`, kdy se nejdříve zkontroluje, zda je uživatel administrátor. Pokud ne, tak se nastaví mód na informační. Administrátor si módy může libovolně měnit pomocí tlačítka „Přepnutí zobrazení“, které obsahuje odkaz s parametrem označení módu. Vedle tohoto tlačítka se vyskytuje možnost přidání nové položky, která přesměruje uživatele na stránku s formulářem. V tabulce jsou poté v každém řádku přítomny tlačítka pro editaci či smazání záznamu.

Přihlášení

O přihlášení se stará `SignPresenter`, který po odeslání přihlašovacího formuláře zavolá metodu `login()`, který vyhledá dostupný Autentifikátor, v tomto případě třídu `Authenticator` a ta ověří přes databázi, zda jsou údaje správné. Pokud ano, tak korektně přiřadí roli uživateli přesměruje jej na hlavní stránku, jinak dá uživateli vědět, kde nastala chyba.

Notifikace

Podle zvoleného módu se zavolají metody modelu `NotificationRepository` pro získání notifikací a výsledek se pošle do view skrz proměnnou `$this->template->notifications` společně se zvoleným módem. V šabloně latte se pak podle módu vykreslí obsah v podobě tabulky. V informačním módu tabulka obsahuje jeden sloupec a pro každou notifikaci tři řádky – Stejně jako v mobilní aplikaci je na prvním řádku nadpis, na druhém text a na třetím čas. V administračním módu jsou informace vyčteny jako v podobě výpisu databáze pro větší přehlednost. To lze vidět na obrázku 5.9. V informačním módu se zobrazují notifikace, které mají čas připomenutí dříve než je aktuální čas (stejně jako v mobilní aplikaci).



Obrázek 5.9: Seznam notifikací v administračním módu.

Program

Vykreslení opět probíhá v metodě `renderDefault()`, která má kromě parametru módu ještě další dva parametry `day` a `op`. Nejdříve se zkontroluje mód zobrazení. V administrační části se uživateli zobrazí seznam všech aktivit v tabulce s možnostmi úpravy a smazání. V informační části se vykresluje grafický program.

Program je vykreslený pomocí HTML a CSS. Nejdříve se vygenerují styly, které určují výšku odpočítaného času pro každou půlhodinu. Všechen vygenerovaný kód se vypíše ve View pomocí proměnné bez escapování znaků, to znamená, že to prohlížeč rozezná jako HTML kód a ne jako pouhý text.
Pro čas 11:00 styl vypadá následovně:

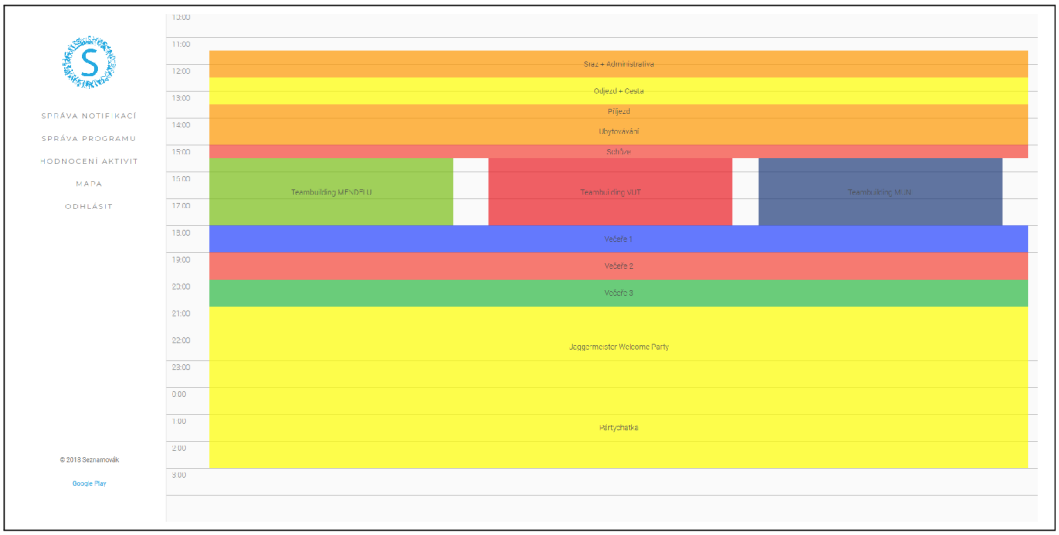
```
.time_1100      {top: 200px;}
```

Poté se vykreslí jednotlivé hodiny, kde atribut *class* obsahuje zmíněný styl.

Pokud je parametr `day` prázdný, tak se načte aktuální čas, který se sníží o jeden den z důvodu hledání programu starší než je ten den. Pro tuto operaci se používá metoda `getDayAfterEvents()`. Ta vrátí celý den programu, který následuje po zadaném dni. Následně se program vykreslí pomocí metody `drawEvents()`, která prochází každou položku v seznamu, poté vypočítá výšku časového bloku, stejně jako v mobilní aplikaci vyhledá aktivity, které se kříží pomocí funkcí `findCrossingBeforeEvents()` a `findCrossingAfterEvents()`. Pomocí toho se zjistí pozice na kterou se má časový blok vykreslit. Výsledný objekt vypadá v HTML kódu následovně:

```
<a href="detail/1"><div style="height: 50px; width: 94%; left:
    80px; top: 225px; opacity: 0.7; filter: alpha(opacity=70);
    line-height: 50px;" class="calendarItem orange">Sraz +
   Administrativa</div>
```

Přepínání dnů v programu je uděláno pomocí parametrů v odkazech, kdy `op` značí, jestli se požaduje další nebo předchozí den a `day` obsahuje původní den.



Obrázek 5.10: Vykreslený program v informačním módu.

Mapa

Mapa je vykreslena pomocí addonu Google Maps API v metodě `createComponentMap()`, která je vypsána v následujícím kódu. Nejdříve se inicializuje mapa, poté se nastaví velikost mapy. Zde bylo nutné výšku nastavit jako velikost viditelné části. Následuje nastavení pozice, na kterou se má mapa zaměřit společně s typem zobrazení mapy. Poté probíhá načtení lokací z databáze a vytvoření značek.

Listing 5.5: Konfigurace Google Maps API

```
$map = $this->map->create();
$map->setProportions('100%', '100vh');

$map->setCoordinates(array(48.899765, 16.569098))
    ->setZoom(17)
    ->setType(MapAPI::TERRAIN);
```

```
$markers = $this->markers->create();
$markers->fitBounds(true);

$markersFromDb = $this->locationsRepository->getAll();
if ($markersFromDb != null) {
    foreach ($markersFromDb as $marker) {
        $markers->addMarker(array($marker['latitude'], $marker
            ['longitude']), \Oli\GoogleAPI\Markers::DROP)
                ->setMessage('<h1 class="map_description_title
                    ">' . $marker['name'] . '</h1>')
                ->setIcon("/images/markers/marker_" . $marker[
                    'iconName'] . ".png");
    }
}

$map->addMarkers($markers);
```

Hodnocení aktivit

V metodě `renderDefault()` se prochází všechna hodnocení z databáze. Každá aktivita má svůj objekt `Rating`, který shromažďuje všechna hodnocení do počtu hvězd a nakonec hodnocení zprůměruje. Postupně se tedy prochází a zaznamenávají počty hvězd. Nakonec se výsledek uloží do proměnné `$this->template->ratings`, která se ve View vypíše do tabulky. Hodnocení aktivit lze vidět na obrázku 5.11

Obrázek 5.11: Hodnocení aktivit v administraci.

5.3 API

API v této práci slouží jako prostředník mezi webem a mobilní aplikací. API je umístěno přímo na serveru Seznamováku s vlastní subdoménou na adrese https://api.seznamovak.org. Spolupracuje také přímo s databází systému Seznamováku. Samotné API reaguje na příchozí

HTTP požadavky GET a POST a odesílá výsledky ve formátu JSON zpět. Je postaveno na Slim Frameworku popsaném v kapitole 3.2.3, který je založen na tzv. cestách (routes), které se spustí v momentě, kdy přijde požadavek přesně na tu cestu. Příkladem lze použít tento začátek funkce:

```
$app->get('/v1/events', function ($request, $response) { ... }
```

Kde `get` znamená typ požadavku, `"/v1/events"` znamená cestu (v tomto případě to ve výsledku vypadá jako https://api.seznamovak.org/v1/events). Dále následuje již definice funkce, která poté vrátí výsledek.

5.3.1 Souborová hierarchie

V kořenovém adresáři se nachází 3 adresáře: `logs` pro uchování zaznamenaných logů, `src` pro zdrojové kódy a `vendor` obsahující zdrojové kódy Slim frameworku. Součástí kořenového adresáře je také soubor *index.php*, který slouží k načtení Slim frameworku. Vyskytují se tam také funkce pro práci s hesly, které jsou převzaty z Nette, aby hesla od mobilní aplikace byla možná ověřit. Uvnitř `src` adresáře jsou tři PHP soubory. *settings.php* nastavuje parametry pro logování a údaje pro databáze, *dependencies.php* tyto parametry sestavuje do objektů a *routes.php* obsahuje všechny cesty.

5.3.2 Funkcionalita API

Jak již bylo uvedeno, API odpovídá na požadavky, které na něj přichází. V tomto případě to může být mobilní aplikace, která zasílá GET požadavky, že chce notifikace, program či lokace. API obratem zašle dotaz na databázi a výsledek pošle ve tvaru JSON se stavovým kódem 200. Příklad jak takový JSON vypadá lze vidět v následujícím kódu.

Listing 5.6: Ukázka JSON souboru který je odpovědí na GET požadavek.

```
[
  {
    "id":"1",
    "name":"Sraz + Administrativa",
    "description":null,
    "startTime":"2018-09-04 11:30:00",
    "endTime":"2018-09-04 12:30:00",
    "color":"orange",
    "canBeRated":"0",
    "location":null,
    "locationName":null,
    "isImportant":"0"
  }
  ...
]
```

Pokud nastala chyba, tak se odesílá JSON s vyplněným atributem `"status"` nebo `"error"` se stavovým kódem 422.

Seznam GET cest:

- `"/v1/events"` – Vrací celou kolekci aktivit programu

- `"/v1/notifications"` – Vrací celou kolekci notifikací
- `"/v1/locations"` – Vrací celou kolekci lokací

Seznam POST cest:

- `"/v1/rating"` – Vloží nebo aktualizuje hodnocení v databázi a odesílá odpověď se stavovým kódem 200, pokud bylo vše v pořádku, jinak 422.
- `"/v1/userdetail"` – Kontroluje, zda uživatel existuje v databázi Seznamováku podle emailu získaného z Facebook nebo Google účtu a vloží ID do databáze. Vrací stavový kód 200 pokud uživatel existuje v databázi, pokud nastala chyba, tak 422. Pokud uživatel nebyl nalezen, tak vrací 404.
- `"/v1/login"` – Kontroluje, zda uživatel existuje v databázi Seznamováku, pokud ano, tak pomocí funkce `calculateHash()` vygeneruje hash zadaného hesla a pokud jsou stejné, tak vloží ID uživatele do databáze a vrátí odpověď se stavovým kódem 200, jinak vrací 422. Pokud uživatel neexistuje, tak vrátí 404.

5.4 Testování

Testování proběhlo v rámci interního týmu Seznamováku, 6 lidí se podílelo na testování při samotném vývoji – především odzkoušením mobilní aplikace na svém telefonu. Po dokončení vývoje byla aplikace představena zbytku týmu Seznamováku, kde proběhlo další otestování. Byli přítomni i uživatelé zařízení s iOS, takže si vyzkoušeli webovou verzi aplikace.

Předmětem testování bylo zjistit, zda je aplikace dostatečně intuitivní a jednoduchá. Uživatelům byl představen scénář, kdy se ocitli první den na Seznamováku a jejich úkolem bylo se v aplikaci zorientovat a vyhledat aktivitu v programu, kterou bylo nutné najít v pravé části programu, která není zpočátku viditelná, ale je nutné se na ni posunout. Dále následovalo ohodnocení aktivity v programu, mimo jiné se také testovala stabilita aplikace, jelikož je u operačního systému Android známé, že na některých zařízeních se chyby vůbec nemusí projevit. Hodinu po ukončení testování na uživatele ještě čekala notifikace, kterou následně museli oznámit, že dorazila.

Výsledek testování byl víceméně úspěšný, uživatelé se v aplikaci zorientovali do minuty a notifikace dorazily téměř všem. Testování se ovšem neobešlo bez pádů aplikace, které byly zaznamenány a dále i opraveny.

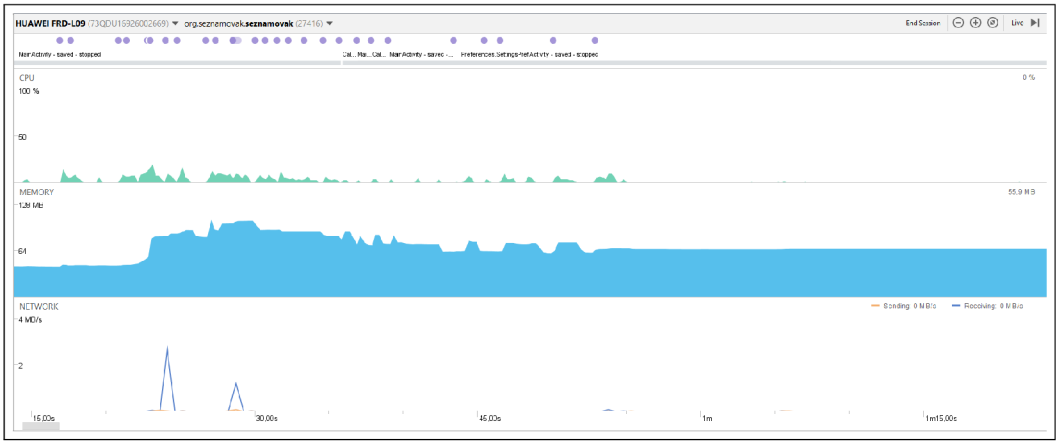
5.4.1 Testování spotřeby aplikace

U mobilních aplikací je důležité, jak se aplikace chová při různých operacích a jak spotřebovává internetové data. Pokud aplikace příliš vytěžuje procesor, systém ji vyhodnotí jako špatně optimalizovanou a aplikaci vypne. V jiném případě dá vědět uživateli, že aplikace spotřebovává moc baterie a uživatel ji vypne, nebo rovnou odinstaluje.

Z toho důvodu byla spotřeba aplikace otestována v prostředí Android Studia 3 nástrojem *Android Profiler*. Testování probíhalo na zařízení Honor 8 s verzí systému Android 7.0.

Jak vypadá průběh analýzy lze vidět na obrázku 5.12. Průměrné využití procesoru během používání je okolo 4%, při zobrazení mapy je vytížení zhruba na 12%. Z hlediska operační paměti aplikace zabírá v průměru 75 MB a ustálí se to na 65 MB. Při použití mapy to narostlo až o 30 MB. Datové vytížení při přihlášení bylo nejvíce u Facebooku okolo 10 kB, u Google a přihlášení přes Seznamovák bylo 4 kB. Při aktualizaci dat, tedy

stažení všech notifikací, aktivit a lokací je spotřeba 66 kB, což se může projevit v delším časovém měřítku, pokud má uživatel zapnutou automatickou aktualizaci každých 15 minut. Zobrazení mapy je spotřebou dat nejvíce náročné, úvodní stažení mapy vezme okolo 1,5 MB dat, což je poměrně hodně. Mapa se poté načítá z operační paměti. Spotřeba dat při hodnocení aktivity je velmi malá, tj. okolo 4 kB.



Obrázek 5.12: Spotřeba mobilní aplikace Seznamovák. V horní části je zobrazeno využití procesoru. Uprostřed je využití operační paměti a v dolní části je spotřeba dat.

Výsledky testování jsou poměrně uspokojující, vytknout by se dala spotřeba dat při zobrazení mapy. Řešením by byla implementace off-line řešení.

Kapitola 6

Závěr

Hlavním úspěchem této diplomové práce bylo vytvoření mobilní a webové aplikace pro seznamovací pobyt Seznamovák. Mobilní aplikace byla implementována na platformě Android, která komunikuje se serverem pomocí REST knihovny retrofit2. Pracuje s plánovací službou Firebase JobDispatcher, která plánuje spouštění notifikací pro upozornění uživatele na událost a také pravidelnou aktualizaci dat ze serveru. Její další významnou funkcí je grafické zobrazení programu Seznamováku pomocí 2D grafiky. Aplikace také pracuje s Google Maps, ve které značí důležité lokality v areálu kempu. Aplikace byla vydána na Google Play[1].

K aplikaci byla vytvořena i téměř identická webová verze běžící na Nette Frameworku pro uživatele ostatních platforem a pro administraci dat. Bylo také vytvořeno jednoduché REST API, které běží na Slim Frameworku, které slouží krom získání dat také pro přihlášení uživatele nebo zaznamenání hodnocení aktivity.

Aplikace byla podrobena testování týmem Seznamováku. Ostré spuštění aplikace bude v září tohoto roku, kdy započne další ročník Seznamováku.

6.1 Další vývoj

Během vývoje aplikace přibylo několik nápadů na vylepšení, které se nicméně nevlezly do základní verze této práce. Každopádně je velmi pravděpodobné, že se do začátku září realizují.

Jedná se o tyto vylepšení:

- Off-line mapa – Zavedení off-line mapy by ušetřilo spoustu dat. Při manipulaci s mapou si aplikace stáhne okolo 3 MB dat. Mapa je sice nějakou dobu v paměti, ale poté je nutné ji stahovat znovu.
- Převod do MVVM architektury – Převodem do této architektury by byla aplikace čistější a ubylo by paměťových úniků.
- Jednoduchá navigace v mapě – Areál je poměrně velký, takže se navigace hodí pro uživatele, kteří jsou v areálu poprvé a neznají okolí. Navigací by byla šipka, která by ukazovala směr cesty od uživatele k zadanému cíli.

[1] https://play.google.com/store/apps/details?id=org.seznamovak.seznamovak

- OAuth2 autentifikace – Momentálně komunikace se serverem není nijak zabezpečena. Přidáním autentifikace by API přijímalo požadavky pouze od autorizovaného klienta, kterým je mobilní aplikace.
- Přihlašování na placené aktivity přímo v aktivitě – Na Seznamováku se konají některé placené aktivity, na které je nutné se přihlásit. Přidáním registrace přímo do aplikace by organizátorům ulehčilo nápor účastníků, kteří se chtějí zapsat.
- iOS verze – Prozatím si uživatelé můžou aplikaci zobrazit ve webové verzi, nicméně by bylo vhodné implementovat aplikaci i na druhý nejpoužívanější mobilní operační systém.

Literatura

[1] *Nette Framework*. [Online; navštíveno 26.12.2017].
URL https://doc.nette.org/cs/2.4/

[2] *Slim Framework*. [Online; navštíveno 26.12.2017].
URL https://www.slimframework.com/

[3] Dalvi, K.: *Material Design – Getting Started Guide for UX Designers. Medium*, 2016.

[4] Google: *Android Developers*. [Online; navštíveno 06.05.2018].
URL https://developer.android.com/

[5] Heller, M.: *What is Kotlin? The Java alternative explained. InfoWorld*, 2017.

[6] Kristin Marsicano, Bill Phillips., Chris Stewart: *Android Programming: The Big Nerd Ranch Guide, Third Edition*. Big Nerd Ranch Guides, 2017, ISBN 9780134706061.

[7] Malý, M.: *REST: architektura pro webové API. Zdroják.cz*, 2009.

[8] Montemayor, M.: *Integrating Crashlytics to monitor your Android app's health. Medium*, 2017.

[9] Ricau, P.-Y.: *LeakCanary: Detect all memory leaks! Medium*, 2015.

[10] Talesra, R.: *Firebase JobDispatcher @AndroidMonk. Medium*, 2017.

[11] Řezáč, J.: *Web ostrý jako břitva*. Baroque partners, 2014.

Příloha A

Obsah CD

Na CD se nachází adresáře:

- `/src` – Adresář se zdrojovými kódy
  - `/android` – Zdrojové kódy aplikace na Androidu
  - `/web` – Zdrojové kódy webové aplikace
  - `/api` – Zdrojové kódy API
- `/doc` – Adresář s technickou zprávou
  - `/latex` – Zdrojové kódy technické zprávy
  - `/pdf` – Technická zpráva ve formátu PDF
- `/media` – Propagační materiály
  - `/video` – Propagační video
  - `/plakat` – Plakát
  - `/market` – Snímky aplikace z Google Play