FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Android klient pro rozvrh hodin na ČVUT v Praze
Student:	Petr Budík
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Pro studenty a vyučující ČVUT navrhněte a implementujte aplikaci pro operační systém Android, která bude pracovat s rozvrhem hodin na ČVUT, ke kterému se připojí pomocí KOSapi a Sirius API. Dále bude využívat Google Calendar API.

Klíčové funkce budou:
- vhodné zobrazení rozvrhů na Android telefonech a tabletech,
- offline přístup k vlastnímu rozvrhu,
- vyhledávání rozvrhů učitelů, učeben a vyučovaných předmětů,
- možnost sdílet vlastní studentský rozvrh s ostatními studenty,
- možnost upravit rozvrh (úpravy nebudou vkládány do databáze ČVUT),
- vyhledávání volného rozvrhového okénka. Uživatel označí více rozvrhů a aplikace zobrazí, kdy mají všechny tyto rozvrhy volno.

V práci

1. analyzujte alespoň tři konkurenční řešení,
2. navrhněte UI aplikace, návrh otestujte,
3. implementujte a otestujte aplikaci,
4. nasaďte aplikaci na Google Play alespoň v uzavřeném testování,
5. zhodnoťte výsledek a navrhněte možná rozšíření.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 10. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Android klient pro rozvrh hodin na ČVUT v Praze

Petr Budík

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

16. května 2019

Poděkování

Velice rád bych poděkoval vedoucímu práce Ing. Miroslav Balík, Ph.D za jeho čas a cenné rady.

Dále děkuji své rodině a přátelům, kteří mě při studiu podporují.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2019 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2019 Petr Budík. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Budík, Petr. *Android klient pro rozvrh hodin na ČVUT v Praze*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Účelem této práce je vytvoření mobilní aplikace pro zařízení s OS Android, která slouží pro práci s rozvrhy hodin na ČVUT v Praze. Vytvořená aplikace je určena pro studenty a zaměstnance na ČVUT v Praze.

Práce se zabývá analýzou existujících aplikací pro práci s rozvrhy, na základě této analýzy jsou vytvořeny konkrétní požadavky na vytvářenou aplikaci. Dále je v práci navrženo uživatelské rozhraní a vytvářená aplikace je implementována, testována a vydána na Google Play.

Vytvořená aplikace umožňuje automaticky získat a zobrazit rozvrhy z ČVUT v Praze, a také umožňuje uživatelům navzájem sdílet své osobní rozvrhy.

Klíčová slova návrh a implementace aplikace, práce s rozvrhem hodin, ČVUT v Praze, Android, Kotlin, MVVM, Sirius API, Google Calendar API

Abstract

Goal of the thesis is an implementation of a mobile OS Android application used as a timetable manager for timetables from CTU in Prague. The created application is intended to be used by students and employees at CTU in Prague.

The thesis describes analysis of existing timetable management applications. Said analysis is used as a basis for specific requirements of the created application. The application's user interface is then designed and the application is implemented, tested and released on Google Play.

The created application enables its users to automatically get and display their timetable from CTU in Prague. It also enables users to share their personal timetables with one another.

Keywords design and implementation of an application, timetable management, Android, Kotlin, MVVM, Sirius API, Google Calendar API

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době většina lidí vlastní mobilní zařízení. Na Českém vysokém učení technickém v Praze (dále jen *ČVUT v Praze*) má každý student a zaměstnanec vlastní rozvrh hodin. Pro zobrazení tohoto rozvrhu existuje na ČVUT v Praze webová aplikace Fittable, která však na mobilních zařízeních rozvrh příliš dobře nezobrazuje.

Cílem práce je vyvinout aplikaci pro operační systém Android zvané CTU-Timetable (dále jen *aplikace*), která bude sloužit pro práci s rozvrhy. Aplikace je určena pro studenty a zaměstnance ČVUT v Praze, kteří vlastní telefon nebo tablet s operačním systémem Android.

Téma jsem si zvolil, protože mnoho studentů pro zobrazování rozvrhu používá existující webovou aplikaci, která se na zařízeních s malou obrazovkou nezobrazuje dobře. Vytvořením mobilní aplikace tento problém vyřeším a navíc mi to umožní implementovat další funkcionality, které jiné aplikace pro práci s rozvrhy již mají.

V práci procházím celým procesem vytváření mobilní aplikace. Zabývám se analýzou existujících řešení, sběrem požadavků, identifikací užitečných softwarových knihoven, návrhem aplikace, její implementací, testováním a publikací na platformní obchod Google Play.

Kapitola 1

Cíl práce

Cílem práce je vytvořit funkční aplikaci pro OS Android, která bude sloužit pro práci s rozvrhem hodin z ČVUT v Praze. Je tedy určena primárně pro studenty a učitele na ČVUT v Praze.

Práce se bude zabývat porovnáním již existujících aplikací pro práci s rozvrhy, specifikací požadavků na aplikaci a nalezením vhodných softwarových knihoven, které implementaci aplikace usnadní.

Na základě předchozích kroků bude navrženo uživatelského rozhraní a aplikace bude implementována a otestována. Následně bude aplikace nahrána na platformní obchod Google Play.

Kapitola 2

Analýza

Tato kapitola shrnuje používaná API, analyzuje a porovnává již existující podobné aplikace pro práci s rozvrhy. Na základě této analýzy jsou vypracovány funkční, nefunkční požadavky a případy užití.

2.1 OS Android

Android je *open-source* operační systém vyvíjený firmou Google. Je navržen primárně pro zařízení s dotykovou obrazovkou – telefony a tablety. Dnes navíc existuje verze Android pro televize (*Android TV*), auta (*Android Auto*) a hodinky (*Wear OS*). Existují i varianty Android použité v počítačích, videokamerách a další elektronice. Dle [1] je OS Android součástí cca 74 % všech mobilních zařízení na světě.

2.1.1 Verze Android API

Dle [2] je nejnovější vydanou verzí OS Android je Android 9.0 Pie (API 28). Aplikace používá Android 6.0 Marshmallow (API 23) jako minimální verzi API, protože ji vyžaduje používaná knihovna Google Calendar API popsaná v sekci 4.3.1. Aplikace tak bude dle [3] použitelná na cca 71 % zařízeních s OS Android v roce 2018.

2.1.2 Užívaný programovací jazyk

V práci primárně používám programovací jazyk Kotlin, který je od roku 2017 oficiálně podporován [4] firmou Google. Je navržen tak, aby plně spolupracoval s jazykem Java. Mohu tak volně používat i knihovny napsané v jazyce Java.

V porovnání [5] s Javou jsou největší výhody Kotlinu:

Větší přehlednost kódu Kotlin obsahuje konstrukce, které výrazně snižují počet napsaných řádek. Dochází tak k úspoře kódu a obecně ke zvýšení přehlednosti programu.

Null safety Kontrola *null* hodnot probíhá při kompilaci. Pokud programátor potřebuje, aby proměnná mohla obsahovat *null* hodnotu, musí tuto proměnnou explicitně nastavit. Kotlin tak předchází *NullPointerException* výjimkám.

2.2 KOS a KOSapi

Komponenta Studium (KOS) je jedna z komponent informačního systému, který je používán na ČVUT v Praze. KOSapi [6] poskytuje aplikační rozhraní (API) v podobě RESTful webových služeb, které zprostředkovává přístup k vybrané části dat v databázi KOS. KOSapi poskytuje read-only přístup k vybrané podmnožině dat týkajících se bílé knihy, rozvrhů, studentů apod.

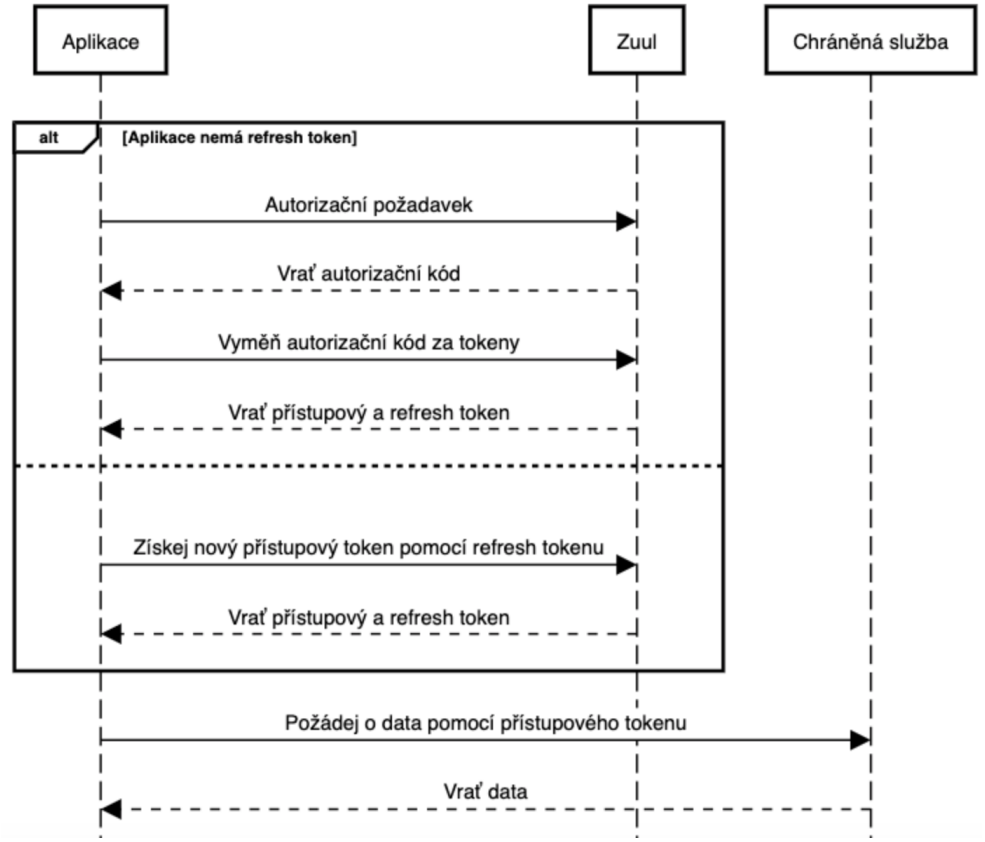
2.2.1 OAuth 2.0

OAuth 2.0 [7] je standardizovaný protokol užívaný pro autorizaci. Umožňuje aplikacím třetích stran získat přístup ke chráněným HTTP službám. K přístupu ke chráněným zdrojům tyto aplikace používají *přístupový token*, což je textový řetězec, který je získán z autorizačního serveru. Aplikace třetích stran tak nemají přístup k citlivým údajům, jako jsou uživatelská jména a hesla.

Způsob, jakým aplikace získá přístupový token, se obecně nazývá *authorization grant*. Existuje více typů grantů.

2.2.2 Zuul

Zuul [8] je OAuth 2.0 autorizační server, který zabezpečuje fakultní služby na celém ČVUT, včetně KOSapi. Aplikace používá typ *authorization grantu* zvaný *authorization code*. Způsob získání přístupového a refresh tokenu je znázorněn na obrázku 2.1.



Obrázek 2.1: *Authorization code* - získání a použití tokenů

2.3 Sirius API

Sirius [9] je API pro správu rozvrhů užívané na ČVUT v Praze. Umožňuje práci s rozvrhy studentů, učitelů, učeben a předmětů. Každá položka v rozvrhu je chápána jako samostatná událost (event). Všechny zdroje tohoto API jsou přístupné přes REST rozhraní, jehož výstupními formáty jsou JSON a iCalendar.

Sirius API získává data z databáze KOS pomocí KOSapi, je proto také chráněn pomocí autorizačního serveru Zuul popsaném v sekci 2.2.2. Rozvrhy jsou každý den aktualizovány a berou v úvahu i dny volna, zkoušky a mimofakultní kurzy.

Sirius API také umožňuje vyhledávání rozvrhů, které je navrženo pro automatické našeptávání (*autosuggestion*). Student může vyhledat rozvrhy ostatních studentů, ale nemůže je prohlížet.

2.3.1 Struktura rozvrhu

Každý rozvrh je chápán jako kolekce událostí. Dle dokumentace Sirius API, existující typy událostí jsou:

- přednáška,
- cvičení,
- laboratoř,
- zkouška,
- zápočtový test,
- jednorázová akce,
- omezení učitele,
- ostatní.

Aplikace nyní podporuje všechny tyto typy událostí a sama další nevytváří.

2.4 Srovnání existujících řešení

Pro aplikaci jsou klíčové tyto funkcionality:

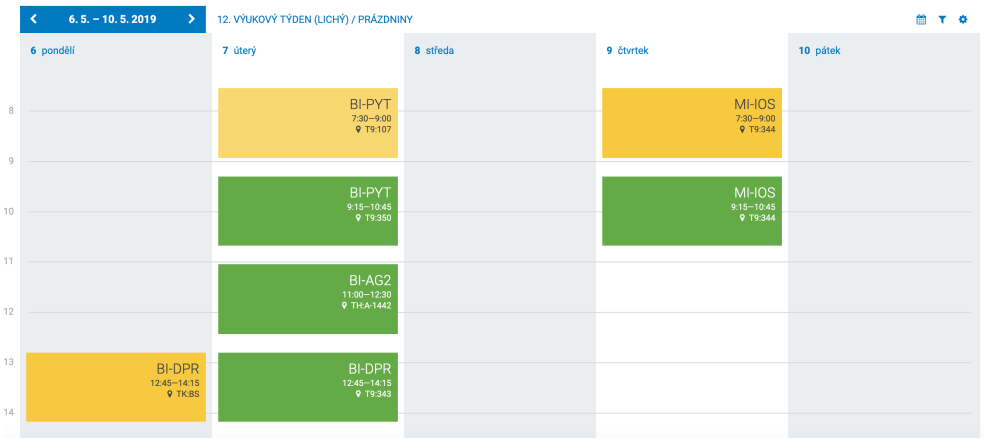
- optimalizované UI pro telefony a tablety s OS Android,
- možnost sdílet rozvrhy mezi vlastními zařízeními a mezi více lidmi,
- offline dostupnost dat,
- úprava rozvrhů,
- možnost vyhledání volného rozvrhového okénka,
- automatické získávání aktualizovaných rozvrhů z databáze KOS.

ČVUT v Praze má webovou aplikaci Fittable pro zobrazování rozvrhů popsanou v sekci 2.4.1. Na Google Play existuje mnoho Android aplikací pro práci s rozvrhy. Žádné z těchto řešení však nemá všechny funkcionality, které jsou požadovány.

2.4.1 Fittable

Autor: Jakub Jirůtka
Počet instalací: -
Počet recenzí: -
Hodnocení: -
Ukázka: viz. obrázek č. 2.2

Fittable [10] je webová aplikace napsaná v jazyce JavaScript. Byla vytvořena na FIT ČVUT v Praze a slouží jako hlavní front-end pro Sirius API. Uživatel může zobrazit svůj vlastní rozvrh, nebo vyhledat a zobrazit ostatní rozvrhy, které mu Sirius API nabídne.



Obrázek 2.2: Aplikace Fittable

Fittable poskytuje denní a týdenní zobrazení rozvrhů, jehož vzhled lze do jisté míry upravit. Také umožňuje vypnout zobrazování některých typů událostí. Uživatelské rozhraní není optimalizováno pro telefony a tablety. To je nejvíce viditelné na mobilních telefonech s malou obrazovkou. Pokud uživatel vybere událost, okénko s jejími informacemi se objeví mimo obrazovku a uživatel toto okénko musí hledat.

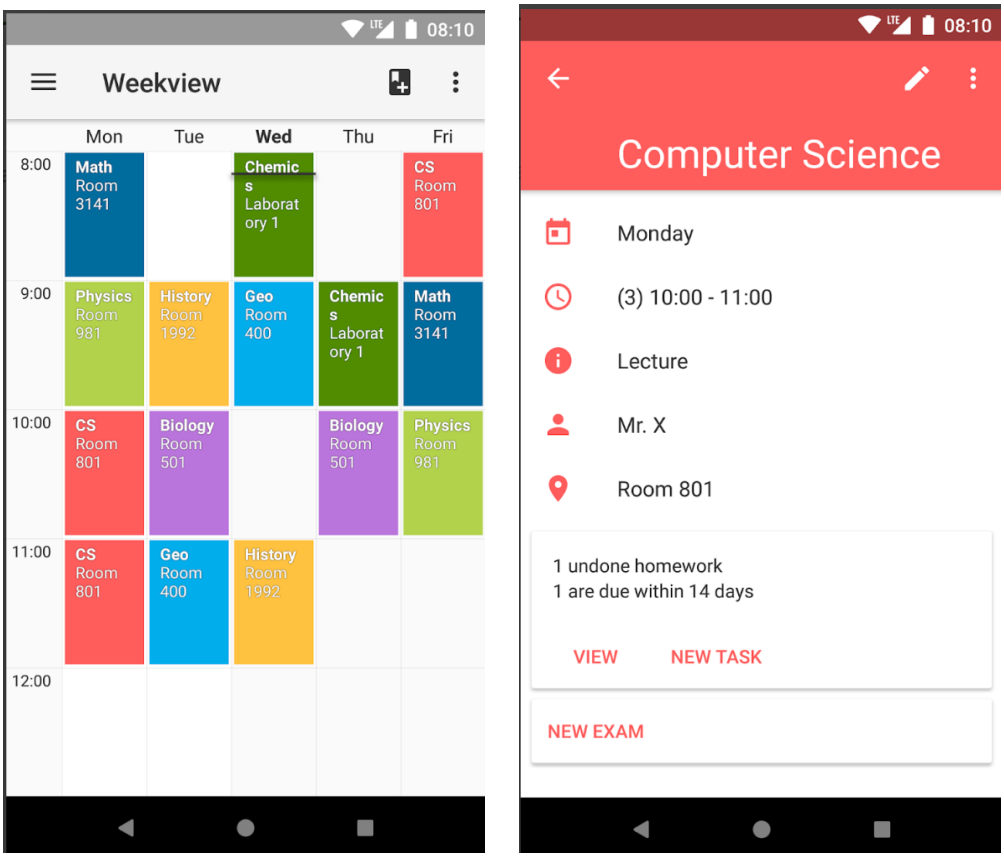
Fittable umožňuje stažení iCalendar souboru, kterou následně může uživatel vložit do vlastního kalendáře, ale ten pak není automaticky aktualizován.

Tabulka 2.1: Výhody a nevýhody Fittable

Výhody	**Nevýhody**
+ Má hezké a jednoduše použitelné UI. + Získává data ze Sirius API.	− UI není optimalizováno pro telefony a tablety. − Studenti nemohou prohlížet rozvrhy ostatních studentů.

2.4.2 Timetable

Autor: Gabriel Ittner Apps UG (haftungsbeschränkt)
Počet instalací: 1,000,000+
Počet recenzí: 40,600+
Hodnocení: 4,2/ 5 hvězd
Ukázka: viz. obrázek č. 2.3



Obrázek 2.3: Aplikace Timetable

Timetable [11] je Android aplikace pro práci s rozvrhy. Timetable poskytuje týdenní a denní zobrazení rozvrhů, jednotlivé události lze barevně rozlišit. Celkově je ovládání velice intuitivní. Lze zobrazit různý rozvrh hodin v lichý/sudý týden. Jediný problém nastává, pokud se některé události konají ve stejný čas, protože se pak v UI překrývají a občas lze označit pouze jednu z nich.

Timetable také má některé další užitečné funkce, jako je umožnění automatického vypnutí zvuku zařízení během vyučování. Bohužel neumožňuje jakékoliv sdílení nebo export dat.

Tabulka 2.2: Výhody a nevýhody Timetable

Výhody	Nevýhody
+ Má velmi dobré UI, jednoduše ovladatelné. + Vytváření úkolů a zkoušek pro předměty. + UI lze upravit. + Automatické vypnutí zvuku zařízení.	− Mezi zařízeními nelze sdílet data.

2.4.3 School Planner

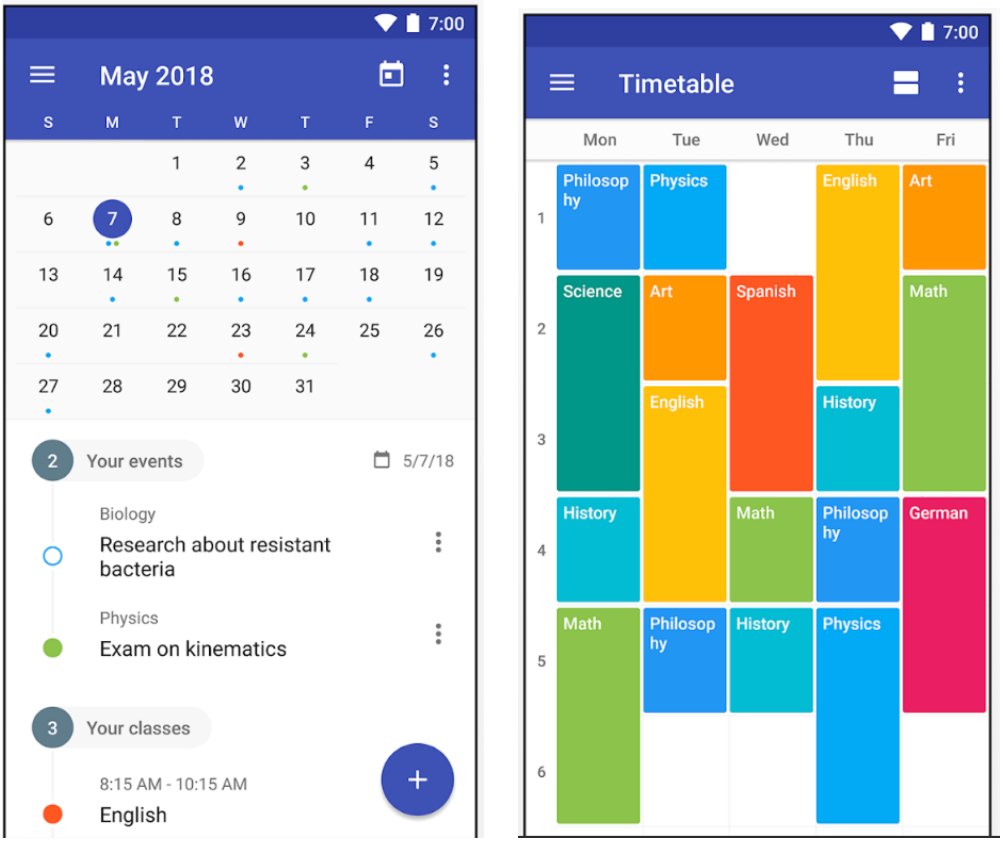
Autor: Andrea Dal Cin
Počet instalací: 1,000,000+
Počet recenzí: 75,300+
Hodnocení: 4,6/ 5 hvězd
Ukázka: viz. obrázek č. 2.4

School Planner [12] je Android aplikace pro práci s rozvrhy. UI má denní, týdenní a měsíční zobrazení. UI některých důležitých částí jako nastavení a vytváření nových předmětů je občas neintuitivní, až matoucí. Jednotlivé hodiny v rozvrhu nelze rozlišit podle jejich typu a nelze zobrazit rozdílný rozvrh v lichý/sudý týden.

School Planner umožňuje zálohovat a obnovit rozvrh, avšak tato funkce v době psaní této práce nebyla funkční.

Tabulka 2.3: Výhody a nevýhody School Planner

Výhody	Nevýhody
+ Má dobré UI. + Měsíční zobrazení předmětů, úkolů a zkoušek.	− UI je někdy matoucí. − Nelze rozlišit jednotlivé typy hodin.

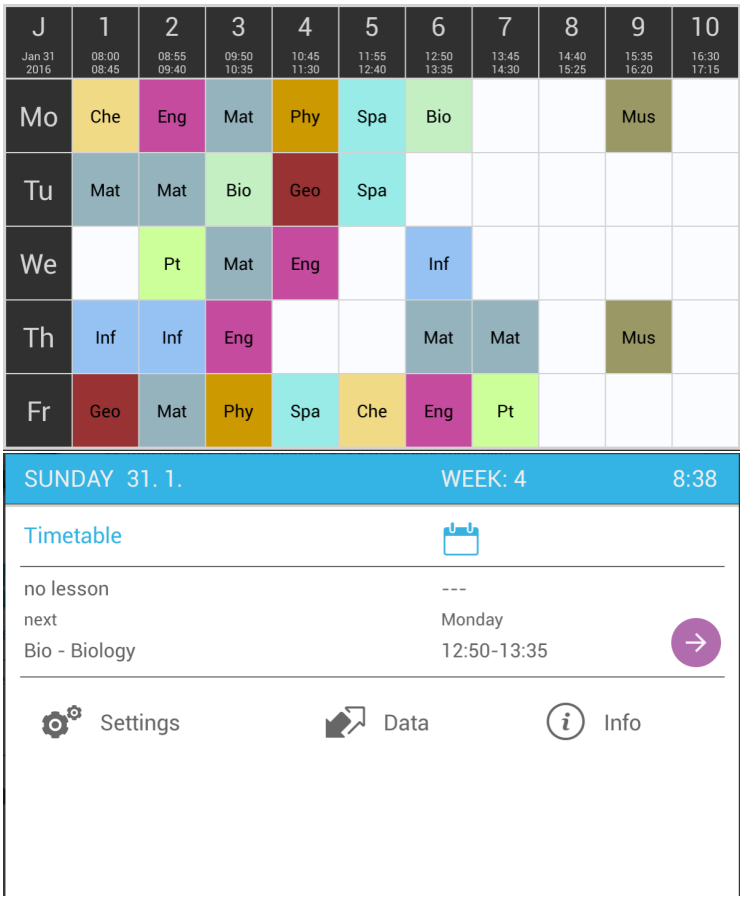


Obrázek 2.4: Aplikace School Planner

2.4.4 Quick Schedule (v češtině Rozvrh hodin)

Autor: IDAL MEDIA
Počet instalací: 1,000,000+
Počet recenzí: 22,000+
Hodnocení: 4,2/ 5 hvězd
Ukázka: viz. obrázek č. 2.5

Quick Schedule [13] je Android aplikace pro práci s rozvrhy. Její UI je celkově velmi špatně navržené. Poskytuje pouze velmi jednoduché týdenní zobrazení a z tohoto zobrazení se nedá dostat k nastavení a dalším funkcionalitám. Nastavení je přístupné pouze po startu Quick Schedule a pokud se k němu uživatel potřebuje dostat později, musí aplikaci úplně vypnout.



Obrázek 2.5: Aplikace Quick Schedule

Tabulka 2.4: Výhody a nevýhody Quick Schedule

Výhody	**Nevýhody**
+ Může zálohovat data na SD kartu.	– UI je velmi špatné. – K některým částem UI se lze dostat pouze po úplném restartu aplikace. – Sdílení rozvrhu nefunguje. – Mnoho chybějících funkcí.

2.4.5 Porovnání klíčových parametrů

Z tabulky č. 2.5 zjistíme, jak jednotlivá existující řešení vyhovují klíčovým parametrům zvoleným v sekci 2.4.

Tabulka 2.5: Porovnání míry splnění klíčových parametrů u existujících řešení

	Fittable	Timetable	School Planner	Quick Schedule	*Aplikace*
UI optimalizováno pro telefony a tablety s OS Android	✗	✓	✓	✗	✓
Rozvrhy lze sdílet	✓	✗	✗	✓	✓
Data dostupná offline	✗	✓	✓	✓	✓
Získávání rozvrhů z databáze KOS	✓	✗	✗	✗	✓
Úprava rozvrhů	✗	✓	✓	✓	✓
Vyhledání volného rozvrhového okna	✗	✗	✗	✗	✓

2.5 Specifikace požadavků na aplikaci

2.5.1 Funkční požadavky

F1 – Zobrazení rozvrhu: Aplikace má týdenní, třídenní a denní zobrazení, po startu je vždy zobrazen rozvrh přihlášeného uživatele. U každé události je zobrazeno její jméno a číslo učebny. Pokud uživatel na událost klikne, zobrazí tak její detailní informace.

F2 – Zobrazení informací o události: Aplikace má obrazovku pro zobrazení všech dostupných informací o události. Uživatel může tyto informace upravit, nebo celou událost vymazat. Také může kliknutím přejít na rozvrh učitelů, učebny nebo daného předmětu.

F3 – Aktualizace rozvrhů: Aktuální data z databáze KOS jsou stažena pomocí Sirius API a ukládána do jednotlivých kalendářů pomocí Google Calendar API. Stahování aktualizací probíhá automaticky, uživatel jej může spustit manuálně. Průběh aktualizace je popsán diagramem 3.1.

F4 – Vyhledávání rozvrhů: Uživatel může vyhledat jiné rozvrhy podle jejich jména. Aplikace rozvrhy nejdříve hledá mezi uloženými rozvrhy v Kalendáři Google, pak je hledá pomocí Sirius API. Uživatel může vyhledaný rozvrh označit, tento rozvrh je následně uložen do nového kalendáře ve službě Kalendář Google a stane se tak dostupný offline.

F5 – Sdílení rozvrhů: Studenti mohou vyhledávat rozvrhy ostatních studentů, ale nemohou je přímo zobrazovat pomocí Sirius API. Aplikace

umožní uživatelům sdílet své osobní kalendáře pomocí Google Calendar API. Sdílené kalendáře aplikace rozezná a začne je automaticky používat.

F6 – Vyhledání volného času: Uživatel vybere více rozvrhů a aplikace zobrazí všechny časové úseky, kde v těchto rozvrzích není žádná událost. Příkladem využití je setkání více lidí v čas, kdy všichni mají volno.

F7 – Manipulace s událostmi: Uživatel může vytvářet, upravovat a mazat události. Tyto změny jsou ukládány v kalendářích v Kalendáři Google, ale nejsou nijak ukládány pomocí Sirius API. Všechny způsoby manipulace s událostmi jsou popsány v diagramu 3.1.

2.5.2 Nefunkční požadavky

N1 – Aplikace pro OS Android: Aplikace je implementována pouze pro telefony a tablety se systémem OS Android. Postup výběru verze Android API je popsán v sekci 2.1.1.

N2 – Lokální přístupnost rozvrhů: Nejnovější stažená verze rozvrhů musí být přístupná i v případě, že uživatel nemá přístup k Internetu. To platí pro jeho osobní rozvrh, rozvrhy s uživatelem sdílené a další uživatelem označené rozvrhy.

2.6 Případy užití

Diagram případů užití zobrazuje chování systému z pohledu koncového uživatele. Skládá se z případů užití, systémů, externích uživatelů a vztahů mezi nimi. S pomocí tohoto diagramu si programátor dokáže aplikaci lépe představit, usnadňuje i komunikaci se zadavatelem práce.

Diagram případů užití aplikace je znázorněn na obrázku č. 2.6.

2.6.1 Popis případů užití

UC1 – Zobrazit rozvrh: Aplikace zobrazí rozvrh podle funkčního požadavku *F1* popsaném v sekci 2.5.1.

UC2 – Zobrazit vlastní rozvrh: Uživatel zobrazí svůj vlastní rozvrh. Aplikace tento rozvrh při spuštění automaticky zobrazí.

UC3 – Zobrazit detail události: Uživatel získá všechny informace o události, kterou vybere z rozvrhu.

UC4 – Vytvořit událost: Uživatel může ve vlastním rozvrhu vytvořit novou událost podle funkčního požadavku *F8* popsaném v sekci 2.5.1.

UC5 – Vymazat událost: Uživatel může vymazat jakoukoliv událost z vlastním rozvrhu podle funkčního požadavku *F8* popsaném v sekci 2.5.1.

UC6 – Upravit událost: Uživatel může upravit informace o kterékoliv události podle funkčního požadavku *F8* popsaném v sekci 2.5.1.

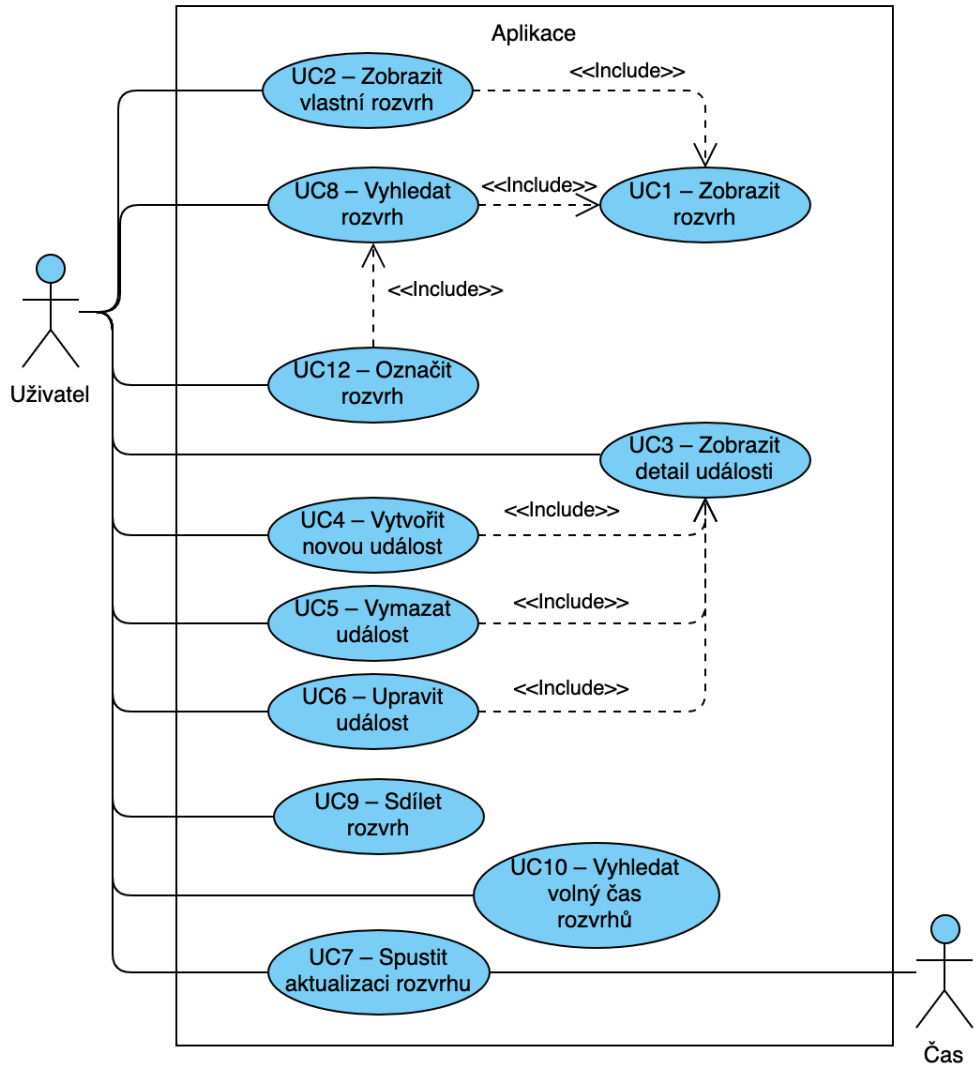
UC7 – Spustit aktualizaci rozvrhů: Uživatel může manuálně spustit aktualizaci rozvrhů.

UC8 – Vyhledat rozvrh: Uživatel podle jména vyhledá jiný rozvrh, který bude následně zobrazen místo rozvrhu uživatele.

UC9 – Sdílet rozvrh: Uživatel sdílí vlastní rozvrh s jiným uživatelem, nebo s dalším svým zařízením.

UC10 – Vyhledat volný čas rozvrhů: Uživatel označí více různých rozvrhů. Výsledkem je zobrazení rozvrhu s vyznačenými intervaly, ve kterých žádný z rozvrhů nemá události.

UC11 – Označit rozvrh: Uživatel označí vyhledaný rozvrh. Ten se tak stane dostupný offline.



Obrázek 2.6: Diagram případů užití.

Kapitola 3

Návrh

Tato kapitola popisuje, jakým způsobem jsou získávána data rozvrhů hodin z databáze KOS a jak jsou ukládána pomocí služby Google Calendar. Dále popisuje softwarovou architekturu používanou v aplikaci a popisuje důležité obrazovky uživatelského rozhraní.

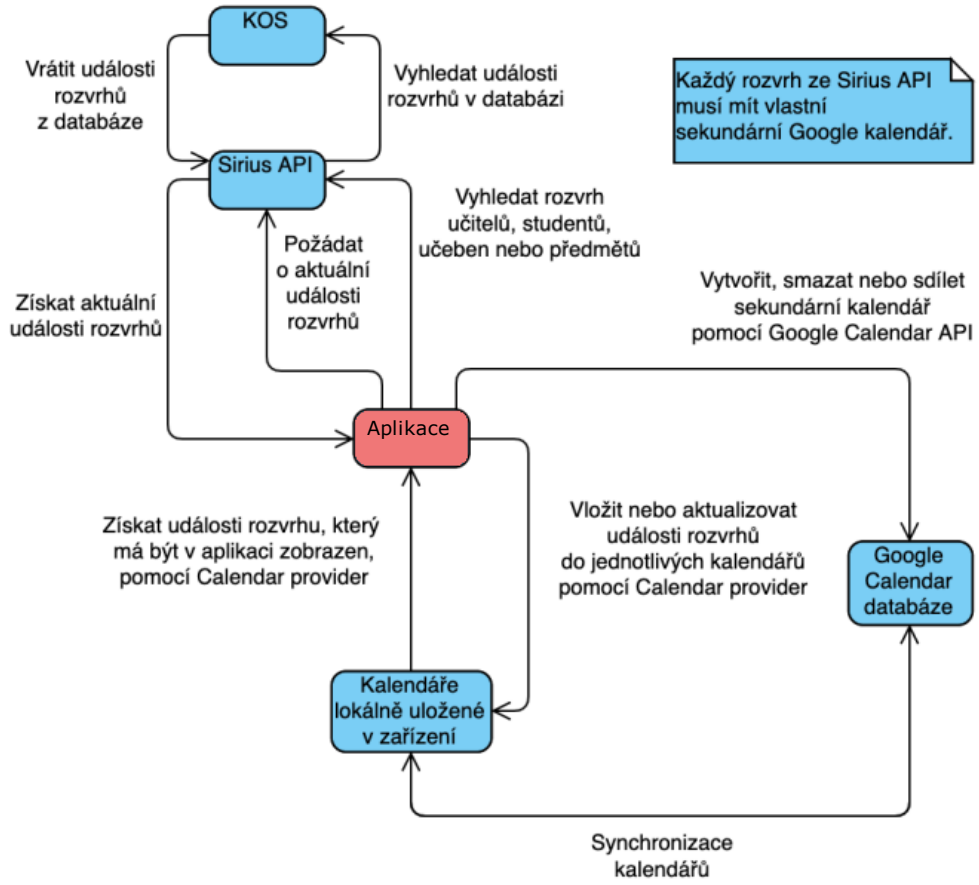
3.1 Získávání a ukládání dat rozvrhů

V diagramu 3.1 je znázorněno, jakým způsobem aplikace používá Sirius API, Calendar provider a Google Calendar API.

3.1.1 Získávání dat rozvrhů z databáze ČVUT v Praze

Hlavním zdrojem aktuálních dat rozvrhů ČVUT v Praze je databáze KOS, se kterou aplikace komunikuje pomocí Sirius API popsané v sekci 2.3.

Rozvrhy hodin jsou přístupné pouze pro zaměstnance a studenty ČVUT v Praze, proto Sirius API patří mezi fakultní služby, které jsou zabezpečené pomocí OAuth 2.0 autorizačního serveru Zuul, který je popsán v sekci 2.2.2. Uživatelé aplikace se musí přihlásit vlastním školním účtem, aplikace pro tento účel používá knihovnu AppAuth for Android popsanou v sekci 4.3.3.



Obrázek 3.1: Diagram znázorňující, jak aplikace používá Sirius API a Kalendáře Google.

3.1.2 Ukládání rozvrhů

Pro ukládání rozvrhů byla zvolena internetová služba Kalendář Google, která má tyto výhody:

- uložená data jsou dostupná z více zařízení;
- je multiplatformní, zařízení nemusí mít OS Android a ani se nemusí jednat o mobilní zařízení;
- obsahuje logiku sdílení kalendářů, které lze použít pro sdílení rozvrhů;
- obsahuje logiku synchronizace kalendářů mezi zařízeními a serverem.

Každý rozvrh z databáze ČVUT v Praze, který je uložený pomocí služby Kalendář Google, potřebuje mít vlastní kalendář. Je tak uložen rozvrh uživatele a následně další rozvrhy, které uživatel zvolí.

Jedním z požadavků na aplikaci je offline přístup k uloženým rozvrhům. Proto je pro vytváření, aktualizaci, čtení a mazání událostí v kalendářích používán Calendar provider popsaný v sekci 4.3.2. Ten pracuje s daty kalendářů, které již byly synchronizovány ze serverů a jsou lokálně k dispozici.

Aplikace potřebuje umět vytvářet, mazat a sdílet kalendáře, proto je použito Google Calendar API popsané v sekci 4.3.1, které komunikuje přímo se službou Kalendář Google. Calendar provider tuto funkcionalitu nemá.

3.2 Zvolená softwarová architektura

Pro aplikaci byla zvolena softwarová architektura Model-View-ViewModel (dále jen *MVVM*) s *Repository* patternem znázorněná na obrázku 3.2. MVVM je třívrstvá architektura:

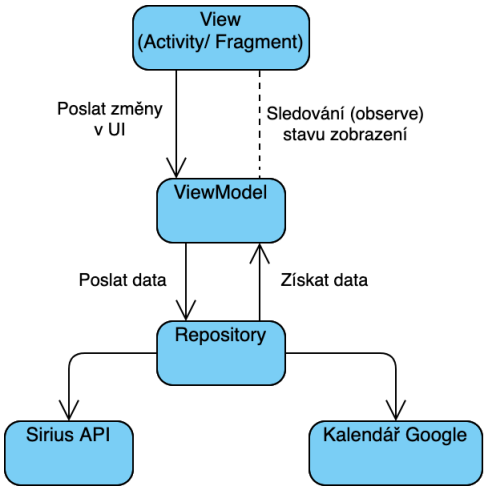
Repository je součástí datové vrstvy. S *Repository* komunikují třídy v jiných vrstvách, aby získali přístup k datům z různých externích zdrojů. *Repository* je tak jediná třída, která musí umět komunikovat s externími zdroji.

ViewModel obsahuje hlavní část logiky a drží stav zobrazení. Nedrží referenci na *View* a není závislý na jeho životním cyklu. Obsahuje proměnné typu LiveData popsané v sekci 4.3.8, ve kterých je uložen stav zobrazení.

View je prezenční vrstva, která se stará pouze o způsob zobrazování. Má vždy právě jeden *ViewModel*, ve kterém je uložen stav zobrazení v proměnných typu LiveData, které sleduje jako *Observer*.

Použití této architektury v Android aplikaci poskytuje následující výhody:

- kód je čitelnější a lépe testovatelný;
- data lze jednoduše sdílet mezi různými *Aktivitami/Fragmenty*;
- snižuje dopady změn konfigurace (změny v nastavení, otočení zařízení na výšku/ na šířku);
- vícevláknové programování je bezpečnější;
- byla vytvořena oficiální kolekce knihoven *Android Architecture Components* [14], které použití architektury MVVM velmi zjednodušují.



Obrázek 3.2: Diagram znázorňující MVVM architekturu.

3.3 Návrh UI

3.3.1 Zásady tvorby UI aplikace

Uživatelé očekávají, že vzhled a chování všech Android aplikací je konzistentní. Firma Google proto vytvořila Material Design guidelines a App Quality guidelines [15]. Jsou to zásady, které Google používá ve svých Android aplikacích a které by měli dodržovat všichni tvůrci Android aplikací.

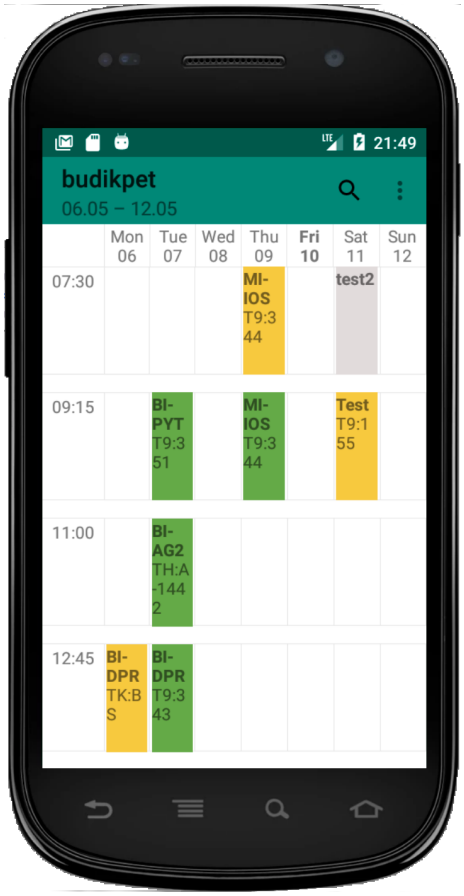
3.3.2 Popis obrazovek

V této sekci jsou představeny jednotlivé důležité obrazovky aplikace a jejich popis.

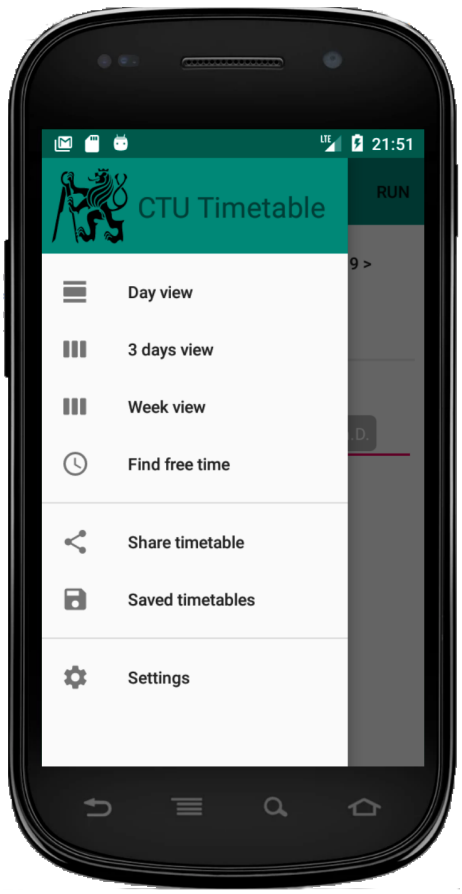
Postranní navigační lišta: Hlavní menu, které slouží pro navigaci mezi obrazovkami. Je přístupné z většiny obrazovek.

Týdenní zobrazení, třídenní a denní zobrazení: Zobrazují rozvrh hodin právě jednoho uživatele. Umožňují vyhledat jiného uživatele a zobrazit jeho rozvrh. Kliknutím na událost se zobrazí její detail. Kliknutím na prázdné pole může uživatel přidat událost, pokud si právě prohlíží svůj vlastní rozvrh hodin. Jde o stejnou obrazovku, která má pouze nastaven rozdílný počet zobrazených dní. Obrazovku pro zobrazení rozvrhu hodin bylo nutné vytvořit ručně.

Zobrazení detailu události: Zobrazuje všechny informace o dané události. Modře zbarvené texty reprezentují klikatelné odkazy na další rozvrhy hodin. Pokud si uživatel právě prohlíží svůj vlastní rozvrh hodin, může



Obrázek 3.3: Týdenní zobrazení



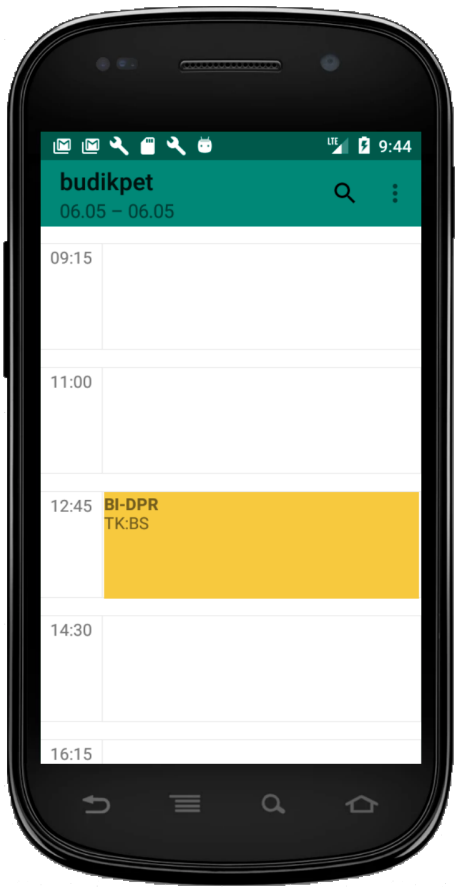
Obrázek 3.4: Postranní navigační lišta

vybranou událost upravit nebo vymazat. Z této obrazovky nelze otevřít postranní navigační lištu.

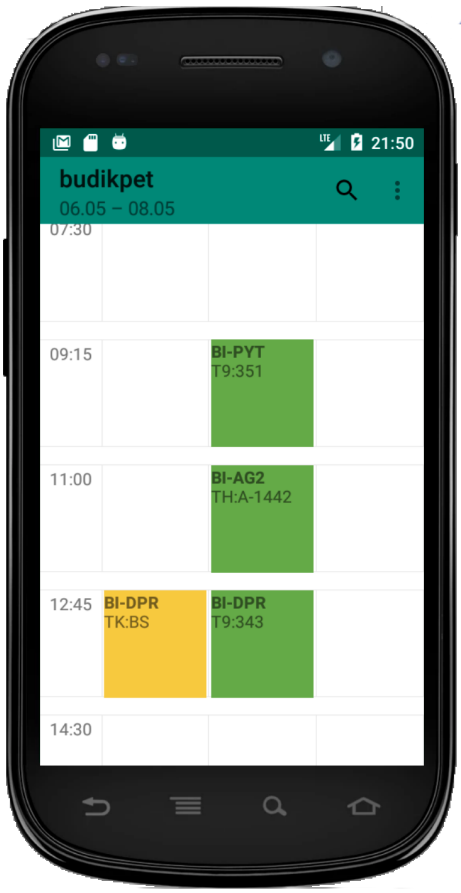
Editor událostí: Umožňuje úpravu nebo vytvoření událostí. Z této obrazovky nelze otevřít postranní navigační lištu.

Zobrazení volného časového okénka: Zobrazuje volné časové úseky vybraných kalendářů během jednoho týdne. Tato obrazovka je stejná jako týdenní, třídenní a denní zobrazení.

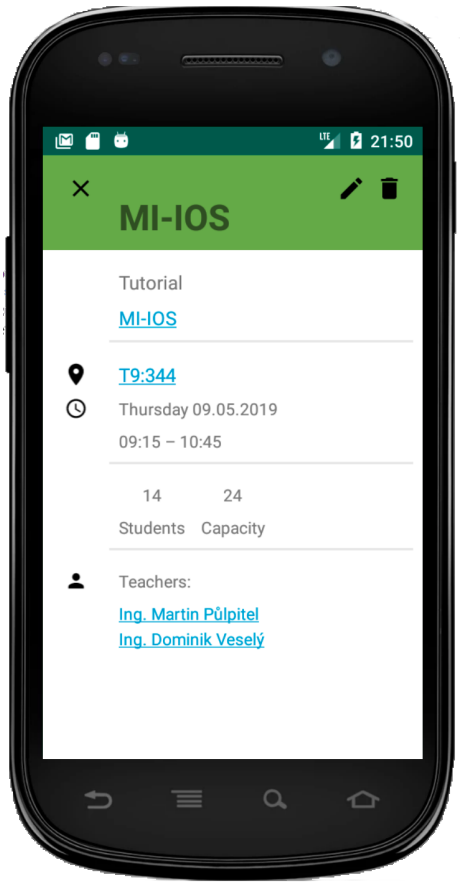
Nastavení zobrazení volného časového okénka: Umožňuje nastavit týden, časové rozmezí jednotlivých dní a vybrat rozvrhy hodin pro zobrazení volného časového okénka.



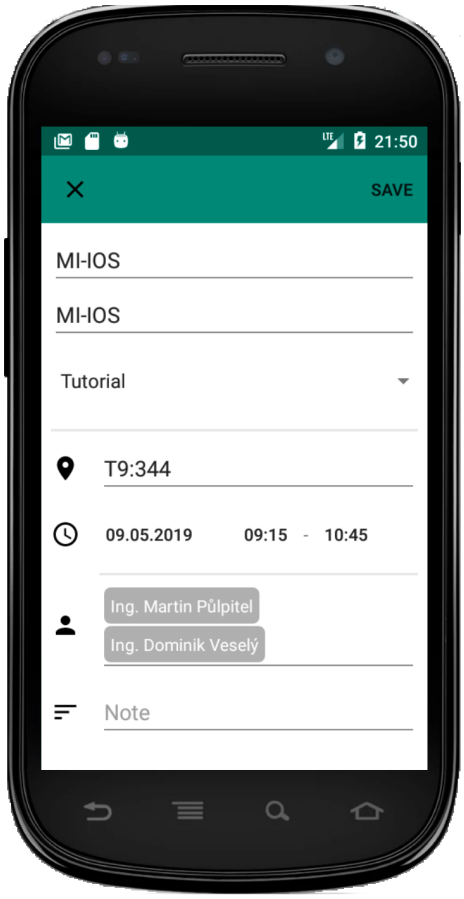
Obrázek 3.5: Denní zobrazení



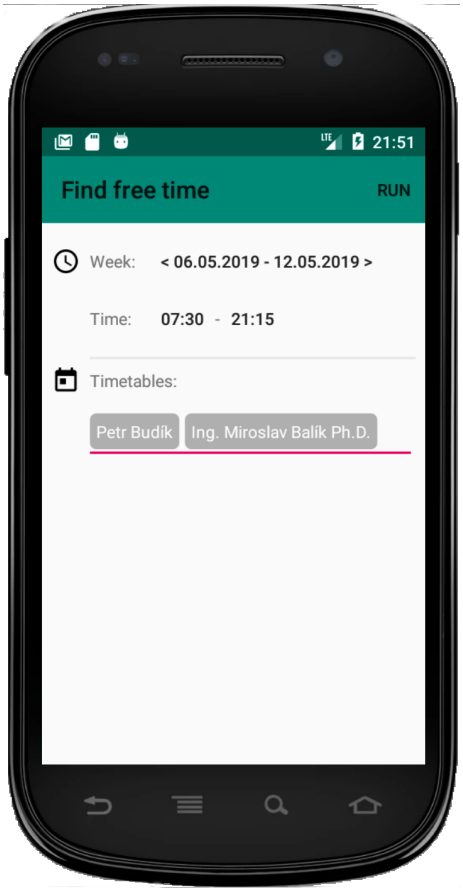
Obrázek 3.6: Třídenní zobrazení



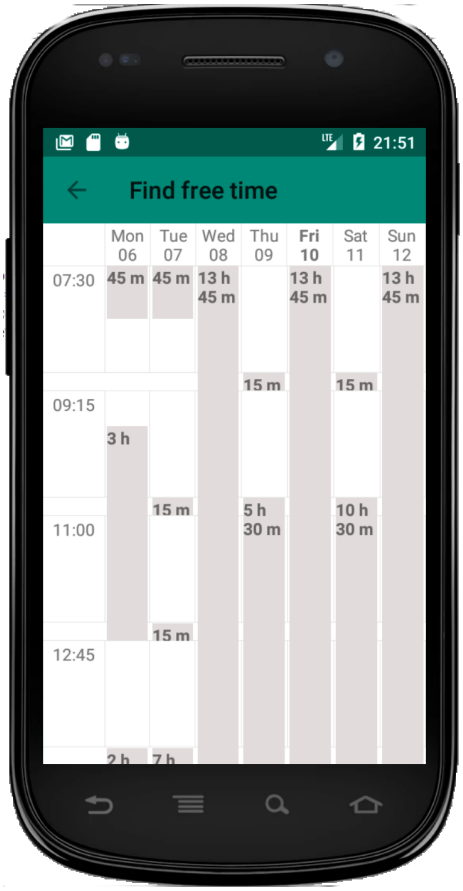
Obrázek 3.7: Zobrazení detailu události



Obrázek 3.8: Editor událostí



Obrázek 3.9: Nastavení zobrazení volného časového okénka



Obrázek 3.10: Zobrazení volného časového okénka

Kapitola 4

Implementace

V této kapitole popisuji použité vývojové nástroje, technologie a knihovny.

4.1 Vývojové prostředí

Pro vytváření aplikace bylo použito oficiální vývojové prostředí Android Studio 3.4. [16]. Pro vývoj nativních Android aplikací jde v současnosti o jediné oficiálně podporované IDE. Obsahuje pro programátora různé důležité nástroje – emulátor, editor uživatelského rozhraní, debugger, refaktorizace specifické pro Android atd.

4.2 Verzování kódu

Pro verzování aplikace byl použit nástroj Git, jako repositář bylo použito webové úložiště GitHub. Verzovací nástroj umožňuje snadné sledování historie vývoje aplikace, zjednodušuje sdílení kódu mezi více zařízeními a významně snižuje pravděpodobnost úplné ztráty práce.

4.3 Důležité použité knihovny

4.3.1 Google Calendar API

Google Calendar API [17] je REST API vytvořené firmou Google, které umožňuje používat většinu funkcí dostupných ve webovém rozhraní [18] internetové služby Kalendář Google. Umožňuje úplnou správu kalendářů a jejich událostí. API lze používat přímo pomocí HTTP požadavků, nebo pomocí klientských knihoven od firmy Google. Vždy vyžaduje internetové připojení. Tato knihovna vyžaduje Android 6.0 Marshmallow (API 23) nebo vyšší.

V aplikaci používám Android klientskou knihovnu [19] pro vytváření, mazání a sdílení kalendářů. Více o použití Google Calendar API v sekci 3.1.1.

4.3.2 Calendar provider

Calendar provider [20] je API, které je součástí OS Android. Pro jeho použití není vyžadováno internetové připojení, protože pracuje s lokálními daty kalendářů, které byly synchronizovány ze serverů.

V aplikaci je toto API použito pro vytváření, aktualizaci, čtení a mazání událostí v kalendářích. Více o použití Calendar provider v sekci 3.1.1.

4.3.3 AppAuth for Android

AppAuth for Android [21] je knihovna, která slouží jako klient pro komunikaci s OAuth 2.0 servery. Tato knihovna vyžaduje Android Jellybean (API 16) nebo vyšší.

V aplikaci je toto API použito pro autorizaci v Zuul serveru a získání přístupového tokenu k Sirius API. Přesný postup je popsán v sekci 2.2.2.

4.3.4 Dagger 2

V aplikaci jsem se rozhodl používat tzv. *dependency injection*. Tato technika umožňuje centralizovat vytváření instancí různých tříd. Tyto instance jsou následně předávány třídám, které o ně požádají.

Dagger 2 [22] je jedním z nejpoužívanějších *dependency injection* frameworků používaných při vývoji Android aplikací. Proto jsem se rozhodl pro jeho použití v aplikaci.

4.3.5 Guava

Guava [23] je soubor knihoven, které zahrnují nové typy kolekcí, knihovnu pro vytváření grafů, knihovnu pro zpracování textových řetězců apod.

V aplikaci používám pouze knihovnu *Ranges* pro implementaci vyhledávání volného časového okénka.

4.3.6 Retrofit 2.0

Retrofit [24] je knihovna, která usnadňuje komunikaci s REST API rozhraními. V aplikaci je tato knihovna použita pro komunikaci se Sirius API.

4.3.7 RxJava 2.0

RxJava [25] je knihovna, která významně usnadňuje psaní vícevláknového kódu. Je založena na *Observer* patternu a funkcionálním programování.

V aplikaci je tato knihovna nejčastěji používána pro zpracování výsledků z asynchronních volání na Sirius API a Google Calendar API. Umožňuje mi vytvářet složité sekvence asynchronních volání.

4.3.8 Android Architecture Components

Android Architecture Components [14] je oficiální kolekce knihoven, která významně usnadňuje implementaci architektury MVVM popsané v sekci 3.2.

LiveData [26] je součástí této kolekce knihoven a umožňuje používat *observable* proměnné. Na tyto proměnné se napojuje tzv. *observer*, který je upozorněn, pokud je stav proměnné změněn. LiveData respektují životní cyklus ostatních komponent aplikace (*Aktivita, Fragment, Služba*). Je tak zajištěno, že LiveData upozorňují pouze aktivní komponenty. To je při vytváření Android aplikací velmi důležitá vlastnost, protože stav komponent se mění často.

V aplikaci je LiveData použito pro zprostředkování komunikace mezi UI a ostatní logikou. Více o použití LiveData v sekci 3.2.

4.4 Android oprávnění

OS Android vyžaduje tzv. oprávnění u všech aplikací, která používají nějaké potencionálně zneužitelné funkce. Pro verze Android API 22 a nižší platilo, že uživatel musel odsouhlasit všechna oprávnění při instalaci aplikace.

Od verze Android 6.0 Marshmallow (API 23) jsou oprávnění rozdělená na bezpečná a nebezpečná. Bezpečná jsou odsouhlasena při instalaci aplikace, nebezpečná je nutné odsouhlasit za běhu aplikace.

Aplikace pro své fungování vyžaduje tato nebezpečná oprávnění:

READ_SYNC_STATS pro sledování průběhu synchronizace kalendářů se službou Kalendář Google.

GET_ACCOUNTS pro čtení a zápis dat do kalendářů.

Protože jsou tato oprávnění nezbytná pro fungování celé aplikace, je uživatel požádán o jejich udělení okamžitě po spuštění aplikace. V případě odmítnutí je aplikace uzavřena.

Kapitola 5

Testování

Aplikaci jsem během implementace průběžně testoval na emulátorech uvedených v tabulce č. 5.1, méně často jsem k testování používal fyzická zařízení popsaná v tabulce č. 5.2. Po dokončení implementace byla aplikace testována z pohledu uživatelské použitelnosti.

Emulátory byly vybrány tak, aby byla aplikace otestována na různých velikostech displeje.

Tabulka 5.1: Přehled používaných emulátorů.

Název	Verze OS	Rozlišení displeje	Pozn.
Nexus S	API 23	480 x 800	Google APIs[1]
Nexus 4	API 25	768 x 1268	Google APIs
Nexus 5	API 28	1080 x 1920	Google Play
Nexus S	API 28	480 x 800	Google APIs

Tabulka 5.2: Přehled používaných fyzických zařízení.

Název	Verze OS	Rozlišení displeje	Pozn.
Xiaomi Redmi Note 7	API 29	2340 x 1080	Telefon
Samsung Galaxy S7	API 23	2560 x 1440	Telefon
Xiaomi Mi A2 lite	API 29	2280 x 1080	Telefon

[1]Emulátor má přístup k podmnožině Google APIs, nedokáže stahovat aplikace z Google Play.

5.1 Testování programátorem

Během implementace jsem průběžně testoval všechny části aplikace, ale primárně jsem se zaměřil na součinnost získávání dat ze Sirius API a ukládání/získávání dat z Google Calendar API.

5.1.1 Unit testy

Během unit testování [27] je testována funkčnost jednotlivých metod jediné třídy nezávisle na ostatních třídách. Jde o nejmenší jednotku testování. Jejich účelem je ověřit, že jednotlivé funkce fungují tak, jak byly navrženy.

V rámci unit testů jsem otestoval načítání a ukládání rozvrhových událostí ve třídě *MainViewModel*.

5.1.2 Integrační testy

Během integračního testování [28] jsou testovány interakce mezi různými třídami. Jejich účelem je nalezení chyb, které se objeví při používání více tříd.

V rámci integračních testů jsem otestoval získávání dat ze Sirius API a jejich správné ukládání do kalendářů ve službě Kalendář Google.

5.1.3 Nalezené chyby

Během testování jsem na několik chyb, které jsou uvedeny dále.

Chyby při získávání dat ze Sirius API Jedním z testů, které jsem provedl, byl průchod a postupné načtení dat ze Sirius API z časového rozmezí 1. 1. 2016 – 6. 5. 2019. Během tohoto testu jsem narazil na několik problémů s dokumentací Sirius API. Malá část událostí neměla uvedenu místnost nebo předmět. Toto způsobilo pád aplikace, který se projevil pouze při pokusu načíst tyto události.

Problém byl jednoduše vyřešen ošetřením výstupu hodnot ze Sirius API. Protože jsem prošel celé časové rozmezí, pro které existují data, neměla by se tato chyba opakovat.

Problémy s načítáním sdílených rozvrhů Jednou z funkcí aplikace je možnost sdílet vlastní rozvrh hodin s jakoukoliv další osobou pomocí emailové adresy. Samotné sdílení je realizováno pomocí služby Kalendář Google.

Chyba nastala při průchodu sdíleného rozvrhu hodin jiného studenta. Protože průchod sdíleným rozvrhem je implementován stejně jako průchod všech ostatních rozvrhů hodin, tak se aplikace pokoušela o získání aktuálních dat ze Sirius API. Protože studenti nemají v Sirius API

přístup k rozvrhům hodin ostatních studentů, tento požadavek vyvolal chybu. K pádu aplikace nedocházelo, ale studenti nemohli bezpečně prohlížet rozvrh hodin ostatních studentů.

Tento problém byl vyřešen dalším ošetřením výstupu hodnot ze Sirius API.

5.2 Uživatelské testování použitelnosti

Po dokončení implementace a vlastního testování jsem nechal aplikaci otestovat dalšími pěti studenty. Dle [29] jde o ideální počet lidí pro testování UX. Cílem bylo zjistit, zda je UI srozumitelné a snadno použitelné pro uživatele, kteří s aplikací nepřišli do styku. Použitá fyzická zařízení jsou popsaná v tabulce č. 5.2.

Studentům, kteří se testování zúčastnili, byl krátce vysvětlen účel aplikace. Následně aplikaci vyzkoušeli dle připravených scénářů průchodu aplikací.

5.2.1 Scénáře

Z případů užití popsaných v sekci 2.6 byly vytvořeny scénáře průchodu aplikací tak, aby byly vyzkoušeny všechny její funkce.

1. **Spuštění a přihlášení:** Spusťte aplikaci a zobrazte si vlastní rozvrh hodin.

2. **Průchod rozvrhem:** Zobrazte vlastní rozvrh hodin v jiném než aktuálním týdnu. Vyhledejte vlastní rozvrh hodin z minulého roku.

3. **Manipulace s událostmi:** Zobrazte si a upravte událost. Vymažte událost. Přidejte událost.

4. **Sdílení rozvrhu:** Sdílejte vlastní rozvrh hodin s jiným studentem (testerem).

5. **Vyhledávání rozvrhů:** Vyhledejte si rozvrh místnosti, učitele nebo vyučovaného předmětu a lokálně si ho uložte. Vyhledejte si rozvrh hodin studenta, který s Vámi sdílí rozvrh. Pokuste se vyhledat rozvrh studenta, který s Vámi rozvrh hodin nesdílí.

6. **Vyhledání volného časového okénka:** Zjistěte, kdy se Vy a další učitel nebo učitelé můžete setkat tak, aby setkání s Vašimi rozvrhy nekolidovalo.

7. **Použijte nastavení:** Upravte původní nastavení tak, aby se rozvrh hodin zobrazoval jinak.

5.2.2 Poznatky z testování

První spuštění a přihlášení: Všichni uživatelé se do aplikace dokázali přihlásit bez problémů. Vyzkoušeli i různé chybové varianty průchodu jako nedání souhlasu s oprávněními nebo odhlášení se z účtu Google během běhu aplikace.

Průchod rozvrhem: Uživatelé sami zjistili, že rozvrhem hodin lze procházet pomocí přejetí prstem.

Manipulace s událostmi: Uživatelé si zobrazili detail události. Dále si vyzkoušeli úpravu, vymazání a vytvoření nové události.

Sdílení rozvrhu hodin: Se samotným sdílením rozvrhu hodin problém nebyl, všichni uživatelé ho dokázali bezpečně najít a použít.

Vyhledání a uložení rozvrhu hodin: Uživatelé vyhledali jiný rozvrh hodin a lokálně si ho uložili. Uživatelé vyhledat rozvrh bez přístupu k Internetu, ale aplikace v tu chvíli neukazovala výsledky ani hlášku, že uživatel k Internetu nemá přístup. Tato chyba byla opravena.

Zobrazení sdíleného rozvrhu hodin: Část uživatelů měla problém s nalezením rozvrhu hodin, který byl s nimi sdílen. Sdílený rozvrh se objeví sám mezi ostatními uloženými rozvrhy a žádné upozornění aplikace nedává. Tento problém zatím nebyl vyřešen, protože není možné jednoduše odlišit sdílené rozvrhy hodin od těch, které uživatel uložil sám.

Vyhledání volného časového okénka: Uživatelé tuto funkcionalitu ocenili.

Nastavení: Uživatelé ocenili možnost nastavení počtu zobrazených hodin, jejich začátku a délky.

Navigace: Všichni uživatelé sami našli postranní lištu.

Vzhled UI: Na vzhled obecně si nikdo nestěžoval.

Závěr

Cílem práce bylo vytvořit funkční aplikaci pro OS Android, která by sloužila pro práci s rozvrhem hodin na ČVUT v Praze. Pro ukládání rozvrhů byla zvolena internetová služba Kalendář Google, která má tyto výhody:

- uložená data jsou dostupná z více zařízení;
- je multiplatformní, zařízení nemusí mít OS Android a ani se nemusí jednat o mobilní zařízení;
- obsahuje logiku sdílení kalendářů, které lze použít pro sdílení rozvrhů;
- obsahuje logiku synchronizace kalendářů mezi zařízeními a serverem.

Vytvořená aplikace splňuje všechny klíčové požadavky:

1. UI aplikace je vytvořeno v souladu se zásadami Material Design, aplikace se tedy dobře zobrazuje na mobilech a tabletech s OS Android.
2. Synchronizace rozvrhů je implementována s použitím služby Kalendář Google. Každé zařízení má svoji lokální kopii, kterou může používat offline.
3. Vyhledávání rozvrhů učitelů, studentů, učeben a vyučovaných předmětů je realizováno pomocí Sirius API.
4. Sdílení rozvrhů je implementováno s použitím služby Kalendář Google.
5. Uživatel může přidávat, upravovat a mazat různé události.
6. Vyhledávání volného okénka bylo implementováno.
7. Dle požadavků byla aplikace nasazena na platformní obchod Google Play v otevřené beta verzi na adrese: *https://play.google.com/apps/testing/-cz.budikpet.bachelorwork.*

V práci jsem používal GitHub jako online úložiště. Pro aplikaci byla zvolena softwarová architektura MVVM s Repository patternem, která byla podpořena knihovnami Android Architecture Components, Dagger 2 a RxJava 2.0. Pro získání a uložení dat jsem používal knihovny Retrofit 2.0, AppAuth for Android, Google Calendar API a Calendar provider.

Většina existujících aplikací pro práci s rozvrhy umožňuje uživatelům vytvářet upomínky na úkoly, některé nabízí možnost automaticky vypínat zvuk během vyučování. V budoucnu by bylo možné aplikaci o tyto funkcionality rozšířit.

Literatura

[1] *Zastoupení OS Android vzhledem k ostatním operačním systémům pro mobilní zařízení.* [online], [cit. 29.12.2018]. Dostupné z: `http://gs.statcounter.com/os-market-share/mobile/worldwide`

[2] *OS Android 9 Pie.* [online], [cit. 29.12.2018]. Dostupné z: `https://developer.android.com/about/versions/pie/android-9.0`

[3] *Distribuce zařízení s OS Android podle verze AndroidAPI.* [online], [cit. 29.12.2018]. Dostupné z: `https://developer.android.com/about/dashboards`

[4] *Kotlin oficiálně podporován firmou Google.* [online], [cit. 29.12.2018]. Dostupné z: `https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/`

[5] *Porovnání jazyků Java a Kotlin.* [online], [cit. 2.1.2019]. Dostupné z: `https://kotlinlang.org/docs/reference/comparison-to-java.html`

[6] *Informace o KOSapi.* [online], [cit. 11.12.2018]. Dostupné z: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`

[7] *O Protokolu OAuth 2.0.* [online], [cit. 20.2.2019]. Dostupné z: `https://tools.ietf.org/html/rfc6749`

[8] *Dokumentace Zuul OAAS, autorizačního serveru fakulty.* [online], [cit. 20.1.2019]. Dostupné z: `https://rozvoj.fit.cvut.cz/Main/oauth2`

[9] *Kód aplikace Sirius na Github.com.* [online], [cit. 11.12.2018]. Dostupné z: `https://github.com/cvut/sirius`

[10] *Kód aplikace Fittable na Github.com.* [online], [cit. 7.12.2018]. Dostupné z: `https://github.com/cvut/fittable`

[11] *Timetable in Google Play.* [online], [cit. 7.12.2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.gabrielittner.timetable&hl=en

[12] *School Planer in Google Play.* [online], [cit. 8.12.2018]. Dostupné z: https://play.google.com/store/apps/details?id=daldev.android.gradehelper&hl=en

[13] *Quick Schedule in Google Play.* [online], [cit. 8.12.2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.idalmedia.android.timetable&hl=en

[14] *Android Architecture Components.* [online], [cit. 20.4.2019]. Dostupné z: https://developer.android.com/topic/libraries/architecture

[15] *Material Design a App Quality guidelines.* [online], [cit. 20.2.2019]. Dostupné z: https://developer.android.com/design

[16] *Android Studio.* [online], [cit. 8.5.2019]. Dostupné z: https://developer.android.com/studio

[17] *O Google Calendar API.* [online], [cit. 13.1.2019]. Dostupné z: https://developers.google.com/calendar/concepts/

[18] *Webové rozhraní Google Calendar.* [online], [cit. 13.1.2019]. Dostupné z: https://calendar.google.com/calendar/r

[19] *Klientská knihovna Google Calendar API pro Android.* [online], [cit. 13.1.2019]. Dostupné z: https://developers.google.com/gsuite/guides/android

[20] *Dokumentace Android Calendar provider.* [online], [cit. 20.1.2019]. Dostupné z: https://developer.android.com/guide/topics/providers/calendar-provider

[21] *Dokumentace AppAuth for Android.* [online], [cit. 20.1.2019]. Dostupné z: https://github.com/openid/AppAuth-Android

[22] *Dagger 2.* [online], [cit. 8.5.2019]. Dostupné z: https://google.github.io/dagger/

[23] *Guava.* [online], [cit. 9.5.2019]. Dostupné z: https://github.com/google/guava

[24] *Retrofit 2.0.* [online], [cit. 6.5.2019]. Dostupné z: https://square.github.io/retrofit/

[25] *RxJava 2.0.* [online], [cit. 6.5.2019]. Dostupné z: https://github.com/ReactiveX/RxJava

[26] *LiveData*. [online], [cit. 6.5.2019]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[27] *Unit testování*. [online], [cit. 13.5.2019]. Dostupné z: http://softwaretestingfundamentals.com/unit-testing/

[28] *Integrační testování*. [online], [cit. 13.5.2019]. Dostupné z: http://softwaretestingfundamentals.com/integration-testing/

[29] Nielsen, J.: *UX testování*. [online], [cit. 13.5.2019]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/

Příloha **A**

Seznam použitých zkratek

UI User Interface – Uživatelské rozhraní

GUI Graphical User Interface – Grafické uživatelské rozhraní

XML Extensible markup language

API Application Programming Interface – Rozhraní pro programování aplikace

OS Operation System – Operační systém

REST Representational State Transfer

JSON JavaScript Object Notation

iCalendar Internet Calendaring and Scheduling Core Object Specification

IDE Integrated Development Environment – Vývojové prostředí

Příloha B

Instalační příručka

B.1 Instalace z Google Play

1. Stáhněte si aplikaci z platformního obchodu Google Play na adrese: *https://play.google.com/apps/testing/cz.budikpet.bachelorwork*
2. Budete-li vyzváni k zapojení se do testování, potvrďte.
3. Aplikace byla nainstalována.

B.2 Instalace z přiloženého souboru

1. Překopírujte instalační soubor *ctuTimetable.apk* do telefonu.
2. V telefonu povolte instalaci aplikací z neznámých zdrojů (Menu -> Nastavení –> Zabezpečení).
3. V telefonu nalezněte instalační balíček a spusťte ho.
4. Aplikace byla nainstalována.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu USB Flash disku
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `ctuTimetable` ..... zdrojové kódy vytvořené aplikace pro OS Android
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `paper` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF