Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Aplikace pro správu studijní agendy

Bakalářská práce

2025 Jan Hrubeš

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Hrubeš**
Osobní číslo:	**I22100**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace pro správu studijní agendy**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce bude vytvořit aplikaci, která umožní studentovi zobrazit vybrané prvky ze studijního systému používaného Univerzitou Pardubice. Aplikace taktéž umožní zobrazit anketu o výuce pro vybrané předměty, které má student zapsané. Aplikace bude využívat API webové služby IS/STAG.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

LACKO, Ľuboslav. *Mistrovství – Android*. Computer Press, 2017. ISBN 978-80-251-4875-4.
JEMEROV, Dmitry a ISAKOVA, Svetlana. *Kotlin in action*. Shelter Island: Manning, [2017]. ISBN 978-1--61729-329-0.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Aplikace pro správu studijní agendy jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 12. 05. 2025 Jan Hrubeš v. r.

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu své bakalářské práce, Ing. Janu Panušovi, Ph.D., za odborné vedení, cenné rady, trpělivost a vstřícnost během celého procesu psaní bakalářské práce a vývoje aplikace.

V neposlední řadě děkuji své rodině a blízkým za trvalou podporu, motivaci a pochopení, které mi umožnily úspěšně tuto práci dokončit.

ANOTACE

Tato bakalářská práce se zaměřuje na návrh a vývoj mobilní aplikace pro operační systém Android, která slouží jako rozhraní k univerzitnímu informačnímu systému IS/STAG. Cílem je vytvořit moderní a přehlednou aplikaci, jež umožní zejména studentům snadný přístup k důležitým studijním informacím včetně možnosti hodnocení kvality výuky. Teoretická část se věnuje principům vývoje aplikací pro Android, jazyku Kotlin, systému IS/STAG a použitým technologiím. V praktické části je popsán proces návrhu, implementace a testování aplikace. Výsledkem je funkční prototyp postavený na moderních technologiích jako Jetpack Compose, Retrofit a Koin, který může sloužit jako základ pro budoucí rozvoj a případné nasazení v prostředí vysoké školy.

KLÍČOVÁ SLOVA

mobilní aplikace, Android, Kotlin, hodnocení, kvalita výuky, IS/STAG, API

TITLE

Study agenda application

ANNOTATION

This bachelor's thesis focuses on the design and development of a mobile application for the Android operating system, serving as an interface to the university information system IS/STAG. The aim is to create a modern and user-friendly application that provides students with easy access to essential academic information, including the ability to evaluate the quality of teaching. The theoretical part deals with the principles of Android application development, the Kotlin programming language, the IS/STAG system, and the technologies used. The practical part describes the process of designing, implementing, and testing the application. The result is a functional prototype built using modern technologies such as Jetpack Compose, Retrofit, and Koin, which may serve as a foundation for further development and potential deployment within a university environment.

KEYWORDS

mobile application, Android, Kotlin, evaluation, quality of teaching, IS/STAG, API

OBSAH

SEZNAM ILUSTRACÍ A TABULEK 9

SEZNAM ZKRATEK A ZNAČEK 10

ÚVOD 11

1 TEORETICKÁ ČÁST 12

1.1 Specifika Android aplikace 12

1.1.1 Architektura 12

1.1.2 Komponenty a životní cyklus 12

1.1.3 Oprávnění 13

1.1.4 Manifest 14

1.1.5 Distribuce 14

1.1.6 Testování 15

1.2 Jazyk Kotlin 15

1.3 Systém IS/STAG 16

1.3.1 Webové služby 17

1.4 Použité nástroje a knihovny 18

1.4.1 Android Studio 18

1.4.2 Jetpack Compose 19

1.4.3 Koin 19

1.4.4 Retrofit 20

1.4.5 JSON to Kotlin 21

1.4.6 Room 22

1.4.7 Work 23

2 PRAKTICKÁ ČÁST 24

2.1 Existující aplikace nad IS/STAG 24

2.1.1 Unizone 25

2.1.2 Univerzita.app 27
2.1.3 IS/STAG Mobile 28
2.2 Návrh aplikace 29
2.2.1 Sestavení funkčních a nefunkčních požadavků 29
2.2.2 Návrh grafického rozhraní 30
2.2.3 Struktura aplikace 31
2.3 Implementace aplikace 34
2.3.1 Příprava a konfigurace projektu 34
2.3.2 Android Resources 35
2.3.3 API rozhraní systému IS/STAG 35
2.3.4 API rozhraní hodnocení kvality výuky 37
2.3.5 Zpracovávání dat 38
2.3.6 Notifikace a běh na pozadí 39
2.3.7 Přihlašování 41
2.3.8 Uživatelské rozhraní 41
2.3.9 Testování aplikace 45
2.4 Možná další rozšíření aplikace 47
ZÁVĚR 49
POUŽITÁ LITERATURA 50
SEZNAM PŘÍLOH 54

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1: Prostředí mobilní aplikace TULapp (zdroj Google Play) [34].....26
Obrázek 2: Prostředí mobilní aplikace Univerzita.app (zdroj Google Play) [35] 27
Obrázek 3: Prostředí mobilní aplikace IS/STAG Mobile (zdroj Google Play) [32].....28
Obrázek 4: Návrh uživatelského rozhraní vyvíjené mobilní aplikace (zdroj vlastní) 31
Obrázek 5: Schéma balíčků v projektu (zdroj vlastní) 32
Obrázek 6: UML diagram tříd na doménové vrstvě (zdroj vlastní) 33
Obrázek 7: Schéma zpracovávání dat (zdroj vlastní) 38
Obrázek 8: Zobrazení notifikací (zdroj vlastní).....40
Obrázek 9: Schéma přechodů mezi obrazovkami (zdroj vlastní).....41
Obrázek 10: Prostředí vyvíjené aplikace (anglická verze, zdroj vlastní) 42
Obrázek 11: Prostředí vyvíjené aplikace (zdroj vlastní).....43
Obrázek 12: Prostředí vyvíjené aplikace (zdroj vlastní).....44
Obrázek 13: Prostředí vyvíjené aplikace (zdroj vlastní).....45

Tabulka 1: Přehled vysokých škol a jejich oficiálních mobilních aplikací 24
Tabulka 2: Seznam funkčních a nefunkčních požadavků.....29
Tabulka 3: Seznam použitých REST endpointů ve vyvíjené aplikaci.....36
Tabulka 4: Metody API hodnocení kvality výuky.....38

SEZNAM ZKRATEK A ZNAČEK

RA – rozvrhová akce

IS/STAG – informační systém studijní agendy

TUL – Technická univerzita v Liberci

UHK – Univerzita Hradec Králové

ZČU – Západočeská univerzita

UI – user interface

XML – extensible markup language

JSON – JavaScript object notation

API – application interface

REST – representational state transfer

SOAP – simple object access protocol

PDF – portable document format

URL – uniform resource locator

IDE – integrated development environment

JVM – Java virtual machine

ÚVOD

V současné době hrají mobilní technologie stále významnější roli v našem každodenním životě. Mobilní zařízení nejsou jen prostředkem osobní komunikace či zábavy, ale také důležitým nástrojem pro přístup k informacím, správě úkolů a organizaci času. Tento trend se výrazně projevuje i v oblasti vzdělávání, kde studenti i pedagogové očekávají možnost rychlého a pohodlného přístupu ke studijním materiálům, informacím o rozvrhu, výsledcích zkoušek či komunikaci s vyučujícími. Mobilní aplikace se v tomto kontextu stávají klíčovým prostředkem, jak tyto požadavky efektivně naplnit.

K vývoji mobilních aplikací mě přivedl předmět Mobilní technologie a aplikace vedený odborníkem z praxe. Programování v Kotlinu s moderními knihovnami mě nadchl natolik, že mě dovedl k nápadu vytvořit mobilní aplikaci pro univerzitu s využitím API univerzitního informačního systému.

Cílem této práce bude vytvořit funkční prototyp mobilní aplikace, který by mohl sloužit jako základ pro další rozvoj a případné nasazení v prostředí vysoké školy. Tato bakalářská práce se tedy zaměřuje na návrh a vývoj univerzitní mobilní aplikace pro operační systém Android, která bude napojena na informační systém IS/STAG. Aplikace nabídne uživatelům – především studentům, ale i zaměstnancům univerzity – přehledné, intuitivní a funkčně bohaté rozhraní pro každodenní práci se studijní agendou včetně možnosti hodnocení kvality výuky.

V teoretické části práce bude nejprve popsána specifika vývoje mobilních aplikací na platformě Android. Dále zde bude uveden popis používaného programovacího jazyka Kotlin. Samostatná kapitola bude věnována systému IS/STAG a možnostem integrace pomocí jeho webových služeb. V závěru teoretické části budou popsány nástroje a knihovny, které byly využity při vývoji aplikace.

Praktická část práce se bude věnovat především samotnému návrhu a implementaci aplikace. Nejprve bude provedena analýza stávajících řešení dostupných nad IS/STAG a bude následovat návrh vlastní aplikace, který bude vycházet z identifikace funkčních a nefunkčních požadavků. Součástí návrhu bude také vytvoření grafického rozhraní, definice struktury projektu a jednotlivých komponent. Implementační část pak bude popisovat konkrétní kroky při vývoji této mobilní aplikace.

1 TEORETICKÁ ČÁST

1.1 Specifika Android aplikace

1.1.1 Architektura

Architektura Android aplikací představuje klíčový aspekt při návrhu moderních softwarových řešení. Její správné zvolení ovlivňuje udržovatelnost, testovatelnost, rozšiřitelnost i samotnou čitelnost kódu. V průběhu vývoje Android platformy se ustálilo několik přístupů, z nichž nejvýznamnějšími jsou MVC (Model-View-Controller), MVVM (Model-View-ViewModel) a Clean Architecture. [1] [2]

Model-View-Controller je jedním z nejstarších architektonických vzorů. V tomto modelu je logika aplikace rozdělena mezi tři vrstvy – Model, který reprezentuje data a jejich logiku, View zajišťující uživatelské rozhraní a Controller, jenž zprostředkovává interakci mezi modelem a pohledem. Přestože je tento přístup koncepčně jednoduchý, v prostředí Androidu často dochází k nežádoucímu propojování vrstev a k tvorbě složitého nepřehledného kódu. [1] [2]

Na tento problém reaguje architektonický vzor MVVM, který do procesu zavádí ViewModel jako zprostředkující vrstvu mezi pohledem a modelem. ViewModel slouží jako držitel stavových dat a logiky prezentace, čímž snižuje závislost pohledu na konkrétní implementaci logiky a podporuje obousměrnou datovou vazbu. Tento přístup je v prostředí Androidu preferovaný zejména díky své kompatibilitě s knihovnami jako je LiveData a Flow. [1] [3]

Nejvyšší míru oddělení zodpovědností pak představuje tzv. Clean Architecture, navržená Robertem C. Martinem. Její principy spočívají v důsledném rozdělení aplikace do vrstev podle jejich zodpovědnosti a směru závislostí. Základem je vnitřní vrstva doménové logiky, na kterou navazují use-casy a teprve na vnějších vrstvách se nachází infrastruktura, jako například databáze nebo frameworkové komponenty. Tato architektura podporuje vysokou míru testovatelnosti a nezávislost na konkrétním prostředí nebo technologii, což ji činí ideální pro rozsáhlé a dlouhodobě udržované projekty. [1] [4]

1.1.2 Komponenty a životní cyklus

Android aplikace je tvořena sadou základních komponent, které společně definují její strukturu, chování a interakci s uživatelem i systémem. Tyto komponenty zahrnují především Aktivita, Fragment, Service, Broadcast Receiver a Content Provider. Každá z nich má přesně vymezenou odpovědnost a životní cyklus řízený operačním systémem Android. [2] [5]

Aktivita představuje vstupní bod uživatelského rozhraní. Každá aktivita odpovídá jedné obrazovce a její životní cyklus určuje, jak a kdy se jednotlivé fáze rozhraní vytvářejí, zobrazují a ničí. Klíčové metody cyklu zahrnují onCreate, onStart, onResume, onPause, onStop a onDestroy. Správné řízení těchto fází je zásadní pro efektivní správu paměti a zachování stavu aplikace při změnách, jako je rotace obrazovky. Při psaní kódu je třeba přepsat metodu onCreate, která nastaví view nebo kořenovou komponentu Jetpack Compose. Fragment je komponenta umožňující modulární a opakovatelný návrh uživatelského rozhraní. Fragmenty mohou být dynamicky přidávány nebo odebírány za běhu aplikace a často slouží jako části aktivit. Mají vlastní životní cyklus, který je synchronizován s hostující aktivitou, ale zahrnuje i specifické fáze jako onAttach, onCreateView nebo onDetach. [2] [5]

Service je komponenta určená k provádění dlouhodobých operací na pozadí bez nutnosti interakce s uživatelem. Může běžet i při minimalizaci aplikace. Rozlišujeme mezi klasickými službami (Service) a foreground services, které zobrazují notifikaci a mají vyšší prioritu. Její životní cyklus zahrnuje metody jako onCreate, onStartCommand a onDestroy. Broadcast Receiver slouží k zachytávání a reakci na systémové nebo aplikační události, jako je změna stavu sítě či příchozí zpráva. Jeho životní cyklus je velmi krátký – aktivuje se při přijetí vysílání a ihned po jeho zpracování končí. Z důvodu výkonnostních omezení by neměl vykonávat náročné operace. Content Provider je komponenta určená pro sdílení dat mezi aplikacemi. Umožňuje standardizovaný přístup k datům pomocí URI, ať už jde o databázi, soubory nebo jiné zdroje. Jeho životní cyklus je rovněž řízen systémem a zahrnuje metody jako onCreate a různé metody pro CRUD operace. [2] [3]

1.1.3 Oprávnění

V systému Android hrají oprávnění (permissions) klíčovou roli v ochraně uživatelských dat a zabezpečení zařízení. Umožňují aplikacím přistupovat k citlivým informacím nebo systémovým funkcím, jako je poloha, kamera, kontakty či připojení k internetu. Android využívá deklarativní přístup, kdy aplikace musí požadovaná oprávnění uvést v souboru AndroidManifest.xml, a v některých případech, zejména v novějších verzích, také získat výslovný souhlas uživatele za běhu aplikace. [2] [6]

Od verze Android 6.0 (Marshmallow) musí aplikace o tato oprávnění požádat explicitně za běhu, což zvyšuje transparentnost a dává uživateli větší kontrolu. Správa oprávnění má přímý dopad na funkčnost i uživatelský zážitek. Aplikace musí být navržena tak, aby korektně reagovala na situaci, kdy oprávnění není uděleno – například omezením funkcionality nebo

zobrazením vysvětlení, proč je dané oprávnění nezbytné. Z hlediska zabezpečení je důležité minimalizovat počet požadovaných oprávnění a používat je pouze tehdy, je-li to nezbytné. Tento princip tzv. „nejmenšího oprávnění“ (principle of least privilege) snižuje riziko zneužití aplikací i dopad případných zranitelností. [2] [6]

1.1.4 Manifest

Verze systému Android, soubor AndroidManifest.xml a proces distribuce aplikací tvoří základní pilíře vývoje a publikace aplikací na této platformě. Každý z těchto aspektů má své specifické požadavky, pravidla a dopad na výslednou kompatibilitu, chování a dostupnost aplikace pro koncové uživatele. Android je systém s rozsáhlou historií verzí, která přináší nové funkce, API a bezpečnostní vylepšení. Vývojáři musí při návrhu aplikace zohlednit nejen aktuální verzi systému, ale i zpětnou kompatibilitu. V souboru AndroidManifest.xml se definuje tzv. minSdkVersion (minimální podporovaná verze), targetSdkVersion (verze, vůči které je aplikace optimalizována) a compileSdkVersion (verze, proti které se kód překládá). Tyto hodnoty ovlivňují přístup k jednotlivým API a určují, na kterých zařízeních bude aplikace schopna běžet. [3] [7] [8]

AndroidManifest.xml je základní konfigurační soubor každé aplikace. Obsahuje informace o komponentách (aktivity, služby, přijímače, providery), požadovaných oprávněních, hardwarových nárocích i podporovaných verzích systému (minimální a cílové verze). Manifest je čten systémem při instalaci a slouží jako deklarace schopností aplikace. Správné nastavení tohoto souboru je klíčové pro funkčnost a bezpečnost aplikace, ale i pro její prezentaci v obchodu Google Play. [7] [9]

1.1.5 Distribuce

Distribuce aplikací probíhá primárně prostřednictvím obchodu Google Play, který poskytuje vývojářům nástroje pro správu verzí, analytiku a monetizaci. Alternativně lze aplikace distribuovat i pomocí přímé instalace souboru APK nebo přes jiné instalační kanály. Před publikací musí aplikace projít procesem digitálního podepisování a často i bezpečnostním nebo kvalitativním přezkumem ze strany platformy. Moderní přístup k distribuci podporuje i tzv. App Bundle, formát, který umožňuje optimalizaci instalace podle konkrétního zařízení (např. jazyk, rozlišení obrazovky). Tento způsob distribuce přináší výhodu menších instalačních balíčků a lepšího výkonu, zároveň však klade důraz na správnou strukturu projektu a testování jednotlivých variant. [9]

1.1.6 Testování

Testování Android aplikací je nedílnou součástí vývojového procesu, která zajišťuje správnou funkčnost, stabilitu a uživatelskou spokojenost. Vzhledem k široké škále zařízení, verzí systému a uživatelských scénářů je kvalitní testování klíčové pro úspěch aplikace na trhu. Testování lze rozdělit do několika úrovní podle rozsahu a cíle. Jednotkové testy (unit tests) ověřují jednotlivé části kódu, typicky metody nebo třídy, bez závislosti na prostředí Androidu. Jsou rychlé, spouštějí se na běžném JVM a slouží k ověření logiky aplikace. Pro jejich psaní se běžně využívají nástroje jako JUnit nebo Mockito. [4] [10]

Instrumentační testy (instrumented tests) běží přímo na zařízení nebo emulátoru a mají přístup k Android API. Umožňují testování interakce mezi komponentami, databází, rozhraním či síťovými vrstvami. Zajišťují realistické ověření chování aplikace v prostředí, ve kterém bude skutečně používána. Zvláštní kategorii představují UI testy, které simulují uživatelskou interakci s aplikací. Pomocí nástrojů jako Espresso nebo UI Automator lze ověřit funkčnost ovládacích prvků, tok obrazovek či správné zobrazování dat. Tyto testy pomáhají zajistit konzistenci a použitelnost aplikace z pohledu koncového uživatele. [10]

Důležitou součástí testování je také kontinuální integrace (CI), která automatizuje spouštění testů při každé změně v kódu. Tím se minimalizuje riziko zanesení chyb a urychluje vývojový cyklus. K dispozici jsou různé nástroje a služby, jako je GitHub Actions, Bitrise nebo Firebase Test Lab, které umožňují testovat na reálných zařízeních. V neposlední řadě je důležité testovat i mimo funkční rovinu – například výkon, spotřebu baterie, chování při ztrátě připojení či při změně orientace obrazovky. Tyto aspekty ovlivňují uživatelskou zkušenost stejně zásadně jako samotná funkčnost. [10]

1.2 Jazyk Kotlin

Kotlin je moderní programovací jazyk, který si od svého uvedení v roce 2011 získal značnou popularitu, zejména ve vývoji aplikací pro operační systém Android. Byl vyvinut společností JetBrains s cílem nabídnout výkonnější a bezpečnější alternativu k jazyku Java, který byl po dlouhá léta dominantním jazykem pro vývoj na této platformě. V roce 2017 byl Kotlin oficiálně uznán společností Google jako preferovaný jazyk pro vývoj Android aplikací, což výrazně přispělo k jeho rozšíření mezi vývojáři. [11] [12]

Jedním z hlavních důvodů, proč si Kotlin získal popularitu, je jeho syntaktická jednoduchost a expresivita. Ve srovnání s Javou umožňuje psát kratší a čitelnější kód, čímž snižuje množství

boilerplate kódu a zvyšuje produktivitu vývojářů. Například podpora null-safety eliminuje běžné chyby spojené s dereferencováním nulových hodnot, což zvyšuje spolehlivost aplikací a snižuje pravděpodobnost výskytu běhových chyb. Další významnou vlastností Kotlinu je interoperabilita s Javou. Vývojáři mohou v rámci jednoho projektu kombinovat kód napsaný v Kotlinu a Javě bez nutnosti složitých úprav. Tato vlastnost je klíčová zejména pro firmy s rozsáhlými kódovými základnami v Javě, protože umožňuje postupný přechod na modernější jazyk bez nutnosti kompletního přepisu aplikací. [11] [13]

Kotlin také přináší vylepšenou podporu pro asynchronní programování prostřednictvím tzv. korutin. Ty umožňují efektivnější správu souběžných úloh a zjednodušují práci s asynchronním kódem, což je v prostředí mobilního vývoje zásadní pro dosažení plynulé uživatelské zkušenosti. Díky této funkcionalitě je možné snadněji implementovat síťové požadavky, databázové operace nebo jiná výpočetně náročná zpracování bez blokování hlavního vlákna aplikace. Ekosystém Kotlinu je podpořen širokou škálou nástrojů a knihoven, které usnadňují vývoj aplikací. [11] [13] [14]

Dalším aspektem Kotlinu je také jeho podpora multiplatformního vývoje. Kotlin Multiplatform umožňuje sdílení kódu mezi různými platformami, včetně Androidu, iOS, backendových serverů a webových aplikací. Tento přístup může významně snížit náklady na vývoj a údržbu aplikací tím, že umožňuje opětovné využití značné části kódu napříč různými prostředími. Vzhledem k těmto vlastnostem je Kotlin stále častěji volen jako hlavní jazyk pro vývoj Android aplikací. Jeho moderní přístup k programování, vylepšená bezpečnost a vysoká míra produktivity z něj činí atraktivní volbu jak pro začínající vývojáře, tak pro zkušené profesionály. S neustálým vývojem a podporou ze strany Google a JetBrains lze očekávat, že jeho význam v oblasti mobilního vývoje bude i nadále růst. [11] [15]

1.3 Systém IS/STAG

Informační Systém Studijní Agendy, známý pod zkratkou IS/STAG, představuje komplexní softwarové řešení pro správu studijní agendy na vysokých školách v České republice. Pokrývá funkce od přijímacího řízení až po vydání diplomu. Systém byl vytvořen a je stále vyvíjen Centrem informatizace a výpočetní techniky Západočeské univerzity v Plzni (ZČU). Jeho hlavním cílem je usnadnění administrativních procesů spojených s evidencí studentů, studijních programů, předmětů a dalších souvisejících údajů. Tento systém využívá celkem devět vysokých škol. Všechny jsou veřejné vysoké školy. Celkem je v ČR 27 veřejných vysokých

škol. Systém IS/STAG je tak patří mezi nejpoužívanější systémy studijní agendy na veřejných vysokých školách. [16] [17]

IS/STAG nabízí široké spektrum funkcionalit, které pokrývají všechny klíčové aspekty studijního procesu. Studenti mohou prostřednictvím tohoto systému provádět registraci a zápis předmětů, sledovat svůj studijní plán, přistupovat k harmonogramu výuky a získávat důležité informace týkající se zkouškových termínů a hodnocení. Pro vyučující a administrativní pracovníky poskytuje IS/STAG nástroje pro evidenci docházky, vkládání výsledků zkoušek, správu sylabů a další aspekty akademické agendy. [16] [17]

Architektura IS/STAG je navržena jako modulární, což umožňuje přizpůsobení systému specifickým požadavkům jednotlivých univerzit. Díky webovému rozhraní a integraci s dalšími univerzitními informačními systémy (např. LMS Moodle, Microsoft Teams, personální a ekonomické systémy…) se jedná o flexibilní a efektivní nástroj pro správu akademického prostředí. Přístup k informacím je řízen na základě uživatelských rolí, což zajišťuje, že jednotlivé subjekty mají k dispozici pouze relevantní údaje. [16] [17]

1.3.1 Webové služby

IS/STAG rovněž poskytuje rozsáhlé API rozhraní, které umožňuje integraci s externími systémy a automatizaci procesů. Toto rozhraní je označováno jako Webové služby IS/STAG. Ačkoliv se toto rozhraní označuje jako webové služby nelze jej zaměňovat za webový portál IS/STAG, který slouží pro práci se systémem pro většinu uživatelů. Webové služby IS/STAG jsou navrženy na základě standardních protokolů, jako jsou REST a SOAP, což zajišťuje jejich širokou kompatibilitu. Prostřednictvím API mohou vývojáři přistupovat k datům o studentech, předmětech, rozvrzích a výsledcích zkoušek, což umožňuje vytváření vlastních aplikací nebo napojení na univerzitní portály. Důležitým aspektem API je bezpečnost, která je zajištěna autentizací a autorizací přístupu ke konkrétním datům, čímž se minimalizuje riziko neoprávněné manipulace s informacemi. [18] [19]

Webové služby poskytují data v různých datových formátech pro protokol SOAP se jedná o XML dokumenty a pro protokol REST se jedná se sérii různých datových formátů, mezi nimiž jsou například známé formáty JSON, XML, prostý text (plaintext), Excel, PDF. Ne všechny služby jsou však poskytovány ve všech formátech. Pro protokol SOAP je k dispozici strojově zpracovatelný popis rozhraní ve formátu WDL, který lze importovat do nástrojů pro generování kódu. Pro protokol REST je tento popis dat ve formátu WADL dostupný jen experimentálně.

Přehled a popis endpointů a možných úkonů je popsán v Katalogu webových služeb IS/STAG, který je online veřejně dostupný s každou instalací webových služeb. [18] [19] [20]

Pro autentizaci a autorizaci se používá přihlášení přes webový portál, která po úspěšném přihlášení předá autentizační token. Dokumentace zmiňuje i jiný způsob pro použití v interních aplikacích, ale ten není veřejně přístupný. Aplikace se serveru pak autentizuje pomocí tokenu předaného pomocí http hlavičky autorizace typu basic. Uživatelé systému mají přidělené role, kterých mohou jich mít i vícero najednou a tyto role je opravňuji přes API provádět dané úkony. [18]

1.4 Použité nástroje a knihovny

1.4.1 Android Studio

Android Studio představuje primární integrované vývojové prostředí (IDE) určené pro tvorbu aplikací v operačním systému Android. Tento software, vyvinutý společností Google, se stal standardem v oblasti mobilního vývoje a poskytuje komplexní nástroje a funkce, které usnadňují proces programování, testování a ladění aplikací. Android Studio je založeno na platformě IntelliJ IDEA od společnosti JetBrains, což mu zajišťuje moderní a flexibilní rozhraní spolu s pokročilými funkcemi pro editaci kódu. Mezi klíčové vlastnosti patří pokročilé nástroje pro refaktoring, podpora verzovacích systémů a efektivní správa projektu prostřednictvím systému Gradle. IDE rovněž integruje vizuální nástroje pro návrh uživatelského rozhraní, které usnadňují tvorbu aplikací odpovídajících různým typům zařízení. [21] [22]

Primárním programovacím jazykem pro vývoj v Android Studiu je Kotlin, který Google v roce 2017 oficiálně označil jako preferovaný jazyk pro Android. Kromě toho IDE podporuje jazyk Java, který byl původně dominantním jazykem v Android ekosystému, a také C++ pro náročnější aplikace využívající nízko úrovňovou optimalizaci. V neposlední řadě podporuje moderní frameworky a knihovny, jako je Jetpack Compose. [12] [21]

Jednou z klíčových součástí Android Studia je vestavěný emulátor, který umožňuje testování aplikací na různých verzích Androidu bez nutnosti fyzických zařízení. Tento emulátor poskytuje podporu pro testování v různých hardwarových konfiguracích, včetně různých velikostí obrazovek, rozlišení a verzí API. IDE dále obsahuje nástroje pro unit testing, instrumented testing a performance profiling, které usnadňují identifikaci a odstranění chyb v aplikaci. Prostřední Android Studia je integrováno s ekosystémem Google, což znamená snadnou synchronizaci s Google Play Console, Firebase a dalšími službami. [21] [22]

1.4.2 Jetpack Compose

Jetpack Compose je moderní framework pro tvorbu UI v ekosystému Android, který představuje revoluční změnu oproti tradičnímu přístupu k vývoji uživatelského rozhraní pomocí XML. Tento deklarativní framework, vyvinutý společností Google, vychází z paradigmat funkcionality a reaktivity, což jej přibližuje moderním technologiím, jako je React nebo SwiftUI. [23]

Hlavní odlišnost Jetpack Compose oproti původnímu Android View System spočívá v deklarativním přístupu. Místo explicitní definice hierarchie prvků UI v XML a jejich propojení s logikou aplikace se v Compose vytváří uživatelské rozhraní přímo v jazyce Kotlin pomocí funkcí označených anotací @Composable. Tyto funkce popisují vzhled a chování jednotlivých prvků UI a při změně stavové informace se automaticky rekonstruuje pouze relevantní část rozhraní, což přispívá k optimalizaci výkonu. [4] [23] [24]

Další klíčovou vlastností Jetpack Compose je jeho flexibilita a modularita. Komponenty jsou snadno znovupoužitelné a lze je skládat do složitějších celků. Tento přístup podporuje principy návrhových vzorů, jako je například kompozice místo dědičnosti, což usnadňuje údržbu a rozšiřování kódu. Díky těsné integraci s Kotlinem lze navíc využít výhod moderního jazyka, jako jsou lambda výrazy, funkce vyššího řádu a typová bezpečnost. Jetpack Compose také přináší jednotný přístup k vývoji pro různá zařízení. Podporuje nejen mobilní aplikace pro Android, ale také další platformy, jako jsou chytré hodinky s Wear OS, tablety nebo dokonce desktopové aplikace. Tím se výrazně snižuje nutnost udržovat separátní kódové báze pro různé platformy a zvyšuje se efektivita vývoje. [4] [23]

Z hlediska vývoje a produktivity nabízí Jetpack Compose několik zásadních výhod. Díky nástroji Android Studio a jeho interaktivnímu náhledu lze okamžitě vidět změny v uživatelském rozhraní bez nutnosti znovu sestavovat celou aplikaci. Tato funkce značně urychluje iterativní vývoj a umožňuje vývojářům rychle testovat různé varianty UI. Ačkoli Jetpack Compose přináší mnoho výhod, jeho přijetí mezi vývojáři stále podléhá určitým výzvám. Přechod ze staršího View Systemu může být náročný, zejména pro rozsáhlé projekty s hlubokou integrací existujících komponent. [23] [25]

1.4.3 Koin

V současném vývoji Android aplikací se klade velký důraz na správu závislostí, která umožňuje efektivní a udržitelný kód. Jedním z nástrojů, který v této oblasti získává na popularitě, je Koin. Jedná se o lehký a snadno použitelný framework pro Dependency Injection (DI) v Kotlinu.

Koin umožňuje vývojářům efektivně spravovat závislosti mezi jednotlivými komponentami aplikace a tím usnadňuje jejich testovatelnost a modularitu. [25] [26]

Na rozdíl od robustnějších řešení, jako je Dagger, se Koin vyznačuje jednoduchou konfigurací a přehlednou syntaxí, která využívá Kotlin DSL (Domain Specific Language). Tento přístup eliminuje potřebu generování kódu a anotací, což zjednodušuje jeho integraci do projektu. Vývojáři mohou snadno definovat moduly, ve kterých deklarují závislosti a určují jejich životní cyklus. Koin podporuje různé typy správy závislostí, jako je single (singleton), factory (vytváření nové instance při každém volání) nebo scoped (omezený na konkrétní kontext, například ViewModel). [26]

Další klíčovou vlastností Koinu je jeho kompatibilita s architekturou doporučovanou společností Google. Lze jej snadno integrovat s komponentami Jetpack, jako jsou ViewModel, LiveData nebo Room, což usnadňuje vývojářům dodržování osvědčených postupů. Díky své jednoduchosti se Koin často používá v menších a středně velkých projektech, kde není potřeba složitějšího řešení, které by například vyžadovalo rozsáhlou konfiguraci. Testovatelnost je dalším důležitým aspektem, kde Koin vyniká. Poskytuje mechanismy pro snadnou výměnu závislostí v testovacím prostředí, což umožňuje efektivní psaní jednotkových testů a zvyšuje kvalitu výsledného softwaru. [26]

Přestože Koin nabízí mnoho výhod, je důležité zvážit i jeho možné nevýhody. Kvůli tomu, že závislosti jsou definovány přímo v runtime prostředí, může být složitější odhalit případné chyby v konfiguraci už v době kompilace. To je zásadní rozdíl oproti Daggeru, který provádí validaci během kompilace a poskytuje tak větší jistotu správnosti kódu. Nicméně díky své jednoduchosti a intuitivní syntaxi je Koin stále jedním z nejpopulárnějších řešení pro dependency injection v Kotlinu a Android aplikacích. [26]

1.4.4 Retrofit

V digitálním věku, kdy většina aplikací využívá webové služby, je důležité mít efektivní a spolehlivý nástroj pro komunikaci s REST API. Jednou z nejpopulárnějších knihoven pro tuto úlohu v prostředí jazyka Java a Kotlin je Retrofit. Retrofit, vyvinutý společností Square, poskytuje jednoduchý a flexibilní způsob, jak implementovat HTTP komunikaci a práci se vzdálenými zdroji dat v mobilních i serverových aplikacích. [27] [4]

Retrofit je založen na principu mapování HTTP volání na Java nebo Kotlin metody pomocí anotací. Namísto ruční tvorby HTTP požadavků umožňuje deklarativní přístup, při kterém

programátor definuje rozhraní obsahující abstraktní metody, které odpovídají konkrétním HTTP operacím. Knihovna se stará o jejich implementaci a serializaci dat. Retrofit podporuje různé formáty dat, včetně JSON a XML, a k jejich konverzi využívá flexibilní mechanismus, který umožňuje integraci s různými knihovnami pro serializaci, jako jsou Gson, Moshi nebo Jackson. Tento modulární přístup usnadňuje přizpůsobení se požadavkům jednotlivých projektů a zajišťuje efektivní zpracování datových formátů. [27]

Jednou z hlavních předností Retrofitu je jednoduchost jeho použití. Deklarativní syntaktická struktura umožňuje snadné pochopení kódu a jeho udržovatelnost. Knihovna také podporuje synchronní i asynchronní volání, což umožňuje flexibilitu při implementaci různých typů aplikací. Retrofit také nabízí rozsáhlou podporu pro interceptory, což umožňuje snadnou manipulaci s HTTP hlavičkami, ověřováním a logováním požadavků. Díky integraci s knihovnou OkHttp poskytuje Retrofit robustní nástroje pro správu připojení a cachování dat. [27]

Další klíčovou výhodou Retrofit je silná typovost. Použití generických typů při definici API metod zajistí, že odpovědné objekty jsou přímo mapovány na očekávané datové struktury, čímž se minimalizuje potřeba manuálního zpracování http odpovědí. Retrofit je často preferovanou volbou při vývoji Android aplikací, kde se standardně používá ve spojitosti s MVVM architekturou a knihovnami, jako je LiveData nebo Flow. Integrace s technologiemi, jako jsou RxJava nebo moderními Kotlin Coroutines, navíc umožňuje pokročilé způsoby zpracování dat a efektivní řízení vláken. [27] [4]

1.4.5 JSON to Kotlin

JSON to Kotlin je jednoduchý bezplatný plugin určený pro vývojové prostředí Android Studia. Jak už název napovídá je určen pro generování datových tříd v jazyce Kotlin na základě vstupu ve formátu JSON. Plugin umí generovat třídy v různých formátech podle používané JSON knihovny. Podporuje formáty několika populárních knihoven jako jsou GSON od Googlu, kotlinx.serialization od JetBrains a další. Eliminuje tak nutnost ručního psaní datových tříd. [28]

Vygenerované soubory jsou přidány do konkrétního balíčku a podle toho umístěny v adresářové struktuře, takže na rozdíl od podobných webových generátorů není třeba nic stahovat, ručně umisťovat a přepisovat direktivy. Navíc generování probíhá kompletně off-line na zařízení, takže není třeba se u projektů pracujících s citlivými daty bát o jejich zneužití provozovatelem

těchto služeb. Dále nabízí různé možnosti nastavení formátování a generování anotací, tak aby kód co nejlépe odpovídal zbytku projektu. [28]

1.4.6 Room

Databázová knihovna Room představuje moderní řešení pro perzistentní ukládání dat v rámci mobilních aplikací na platformě Android. Tato knihovna usnadňuje práci s lokálními databázemi a představuje doporučený standard pro manipulaci s relačními daty. Vychází z technologie SQLite, avšak oproti původnímu API přináší značné zjednodušení správy dat a eliminaci běžných chyb při práci s databázemi. [2] [29]

Základní architektura Room je založena na třech hlavních komponentech: entitách, datových přístupových objektech (DAO) a databázové třídě. Entity reprezentují tabulky v databázi a jsou definovány pomocí anotací, což zajišťuje přehledný a strukturovaný způsob definice modelu dat. DAO slouží k abstrakci přístupu k databázi, což znamená, že aplikace nepracuje s přímými SQL příkazy, ale s metodami definovanými v rozhraních. Tento přístup minimalizuje možnost vzniku chyb, zvyšuje čitelnost kódu a usnadňuje údržbu. Poslední klíčovou součástí je samotná databázová třída, která spojuje entity a DAO a zajišťuje správu databáze. [2] [29]

V porovnání se standardním SQLite API poskytuje Room několik významných výhod. Mezi největší přínosy patří statická kontrola SQL dotazů přímo při kompilaci, což eliminuje možné syntaktické chyby, které by se jinak objevily až za běhu aplikace. Další důležitou vlastností je podpora migrací databází, která umožňuje plynulé přizpůsobení struktury databáze v průběhu času, aniž by došlo ke ztrátě dat. Další klíčovou vlastností je kompatibilita s architekturou MVVM (Model-View-ViewModel), která je v moderních Android aplikacích preferovaná. Room podporuje reaktivní programování pomocí LiveData a Flow, což znamená, že změny v databázi mohou být automaticky reflektovány v UI, aniž by bylo nutné explicitně aktualizovat uživatelské rozhraní. Tato funkce přispívá k větší efektivitě a reaktivnímu řízení stavů v aplikaci. [29]

Z hlediska výkonnosti je Room optimalizována tak, aby minimalizovala režijní náklady spojené s prací nad databází. Interně využívá SQLite a zachovává jeho efektivitu, avšak zároveň poskytuje abstrakci, která redukuje složitost správy databázových operací. Při vhodném použití asynchronních operací a cachovacích mechanismů lze zajistit vysokou responzivitu aplikace i při manipulaci s větším objemem dat. [29]

1.4.7 Work

Knihovna Work je klíčovým nástrojem pro správu asynchronních úloh na platformě Android. Tato knihovna umožňuje efektivní plánování a vykonávání úloh na pozadí s ohledem na systémové podmínky, životní cyklus aplikace a uživatelský kontext. Díky své flexibilitě a schopnosti spolehlivě vykonávat práci i v případě ukončení aplikace nebo restartu zařízení se stala preferovaným řešením pro řadu vývojářů. [30] [25]

WorkManager, hlavní komponenta knihovny Work, poskytuje jednotné API pro práci s úlohami na pozadí, čímž nahrazuje starší mechanismy, jako jsou AlarmManager, JobScheduler nebo Firebase JobDispatcher. Jeho hlavní předností je schopnost automaticky volit nejvhodnější mechanismus pro vykonávání úloh podle verze systému a dostupných zdrojů. Tím WorkManager usnadňuje vývoj aplikací a eliminuje nutnost explicitního rozhodování o způsobu plánování úloh. Jedním z klíčových rysů WorkManageru je podpora různých typů úloh, včetně jednorázových a opakujících se úloh. Ty mohou být naplánovány s podmínkami, jako je připojení k síti, stav nabití baterie nebo volná paměť zařízení. [30] [25]

Další významnou vlastností knihovny Work je její schopnost zachovávat perzistenci úloh. To znamená, že pokud je aplikace ukončena nebo zařízení restartováno, naplánované úlohy budou automaticky obnoveny a vykonány, jakmile budou splněny zadané podmínky. Tato funkcionalita zajišťuje spolehlivost a konzistenci provádění úloh na pozadí, což je klíčové zejména pro synchronizaci dat, zálohování nebo dlouhodobé zpracování úloh. WorkManager je také navržen s ohledem na efektivní správu systémových prostředků. Díky optimalizovanému plánování minimalizuje dopad na výkon a spotřebu baterie, čímž přispívá k lepší uživatelské zkušenosti. Zároveň podporuje moderní architektonické principy ve vývoji Android aplikací. [30]

2 PRAKTICKÁ ČÁST

2.1 Existující aplikace nad IS/STAG

Abych mohl zahájit návrh vlastního řešení mobilní aplikace, je nezbytné nejprve provést analýzu stávajících řešení, která jsou již dostupná na trhu. Tento krok je klíčový pro pochopení současných trendů, identifikaci silných a slabých stránek konkurenčních aplikací a nalezení příležitostí pro inovaci.

Z tohoto důvodu jsem se rozhodl vytvořit přehled vysokých škol, které využívají informační systém IS/STAG, a to na základě podrobného průzkumu mobilních aplikací dostupných v oficiálních obchodech (Google Play pro Android a App Store pro iOS). Tento přehled mi poslouží jako výchozí bod pro další analýzu funkčnosti, uživatelského rozhraní a celkové kvality těchto řešení. Budu vycházet ze seznamu škol dostupných na Wikipedii. [16]

Tabulka 1: Přehled vysokých škol a jejich oficiálních mobilních aplikací

Vysoká škola	**Název aplikace**	**Pro Android**	**Pro iOS**
Univerzita Tomáše Bati ve Zlíně – Moje UTB	Moje UTB	Ano	Ano
Univerzita Palackého v Olomouci	UPlikace	Ano	Ano
Jihočeská univerzita v Českých Budějovicích	StuduJU	Ano	Ano
Univerzita Pardubice	-	Ne	Ne
Univerzita Jana Evangelisty Purkyně v Ústí nad Labem	UJEPapp	Ano	Ano
Univerzita Hradec Králové	-	Ne	Ne
Technická univerzita v Liberci	TULapp	Ano	Ano
Veterinární univerzita Brno	-	Ne	Ne
Západočeská univerzita v Plzni	IS/STAG Mobile	Ano	Ano

(zdroj vlastní)

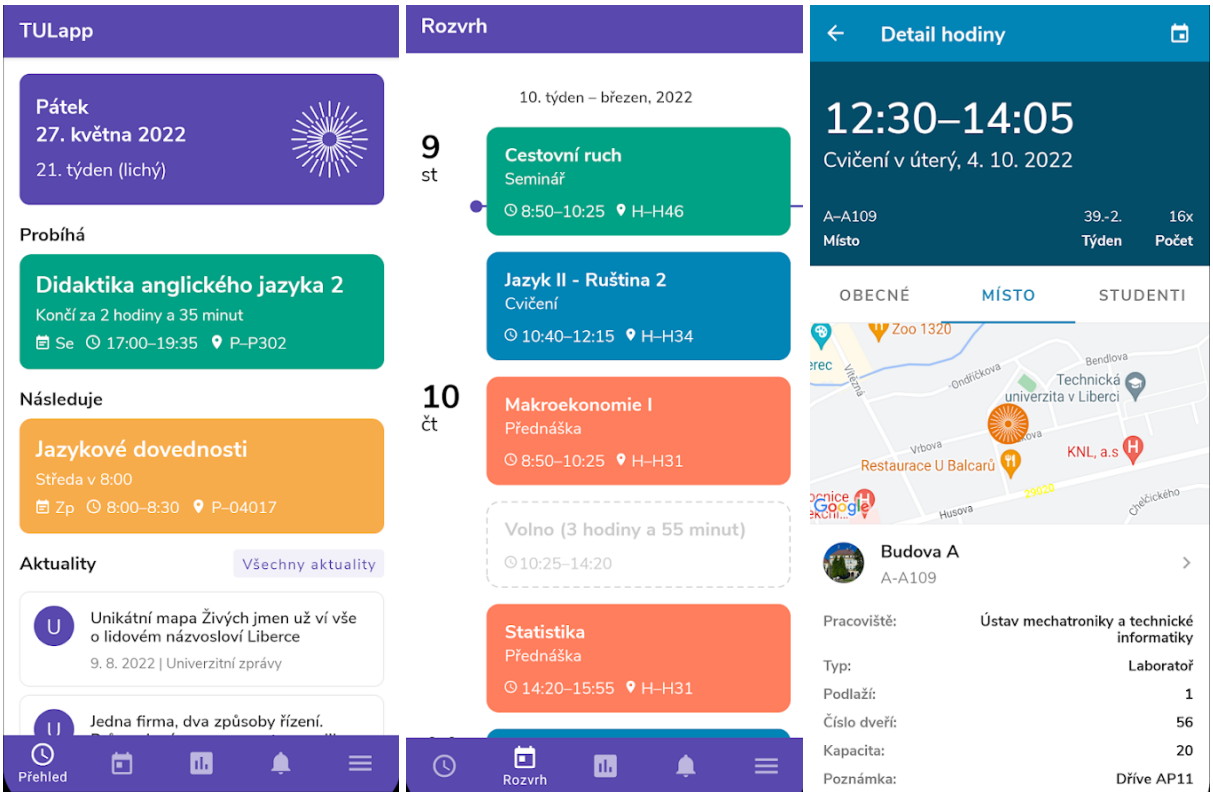
Z provedené analýzy dostupných řešení vyplývá, že z devíti vysokých škol využívajících informační systém IS/STAG má šest z nich vlastní mobilní aplikaci. V pěti případech (tedy všech kromě aplikace ZČU) se jedná o řešení unizone, vyvinuté společností beecode, s.r.o. Z toho lze vyvodit, že vysoké školy obvykle neinvestují do vývoje vlastních aplikací, ale dávají přednost využití již existujících a ověřených platforem. Toto rozhodnutí je pravděpodobně motivováno zejména finančními a časovými náklady spojenými s vývojem aplikace na míru. [31]

Kromě těchto univerzitních aplikací existují také řešení, která mohou využívat studenti napříč různými vysokými školami. Jedním z nich je IS/STAG Mobile, oficiální aplikace vyvinutá samotnými autory systému IS/STAG. Je určena pro všechny vysoké školy, které tento systém používají. Dalším příkladem je Univerzita.app, vytvořená studenty Univerzity Hradec Králové (UHK), která je rovněž dostupná všem školám s tímto systémem a nejedná se o oficiální produkt žádné konkrétní univerzity. Pro účely analýzy a porovnání funkcionalit tedy budu pracovat se třemi vybranými aplikacemi: unizone, IS/STAG Mobile a Univerzita.app. K vyzkoušení a používání některých aplikacím bohužel nemám přístup, a tak budu vycházet ze zveřejněných popisů aplikací. [32] [33]

2.1.1 Unizone

Jak již bylo zmíněno unizone je produkt určený pro řešení na míru a není dostupný jako samostatná aplikace. Funkce produktu unizone se mohou mírně lišit v závislosti na tom, co si daná instituce nechala naimplementovat navíc oproti základnímu produktu. Mezi takové funkcionality může například patřit implementace informační systémů mimo IS/STAG (intranet, novinky z webu, systémy kolejí a menzy atd.). [31]

Níže nalezneme ukázku prostřední mobilní aplikace (v tomto případě se jedná o aplikaci TULapp ve verzi pro systém Android):



Obrázek 1: Prostředí mobilní aplikace TULapp (zdroj Google Play) [34]

Podle snímků obrazovek aplikace vidíme, že aplikace nabízí moderní líbivé barevné grafické prostředí zapadající do grafického designu dané univerzity. Aplikace mimo jiné podporuje i tmavý režim.

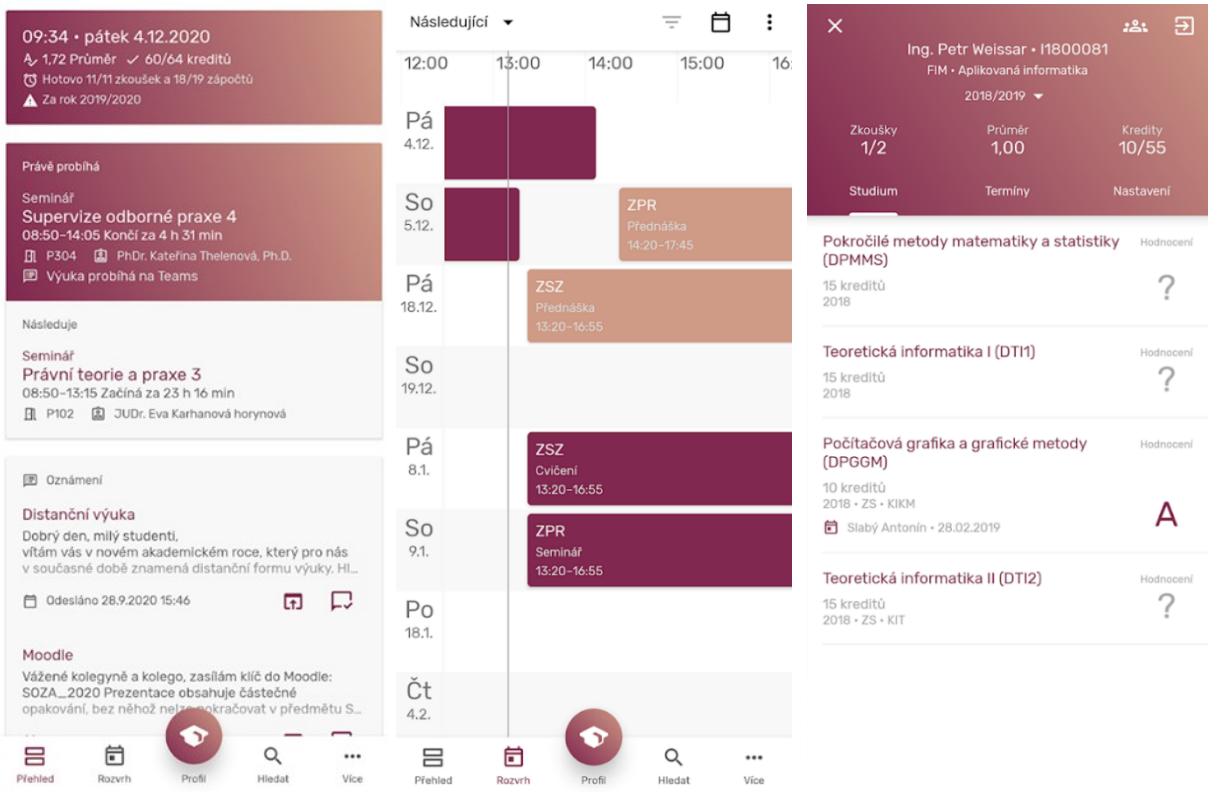
Aplikace pro studenty nabízí přehlednou obrazovku s aktuálními i nadcházejícími rozvrhovými akcemi, detailní rozvrh zahrnující jak rozvrhové akce, tak i zkouškové termíny, seznam zapsaných předmětů s podrobnými informacemi (sylaby, anotace, vyučující atd.), souhrn udělených zápočtů a známek a přehled zkouškových termínů. Studenti si mohou zapisovat či odhlašovat zkoušky, dostávat upozornění na nové zkouškové termíny, známky nebo termíny registrace a odhlášení. Aplikace také umožňuje přístup ke kvalifikačním pracím a notifikace o posudcích. Pro vyučující je k dispozici přehledná hlavní obrazovka a rozvrh výuky. [34]

Mezi funkcionality specifické pro danou instituci patří interaktivní mapa kampusu s vyznačením univerzitních budov, odkazy na menzovní aplikaci, univerzitní e-mail a další systémy školy, stejně jako aktuální zprávy z webu instituce. Aplikace navíc podporuje widgety zobrazující nadcházející rozvrhové akce nebo přehled dnešního rozvrhu. Univerzita také aktivně svou aplikaci propaguje v rámci informačních materiálů pro nové studenty. [34]

2.1.2 Univerzita.app

Aplikace univerzita.app je bezplatná mobilní aplikace určená pro studenty různých škol. Vzhledem k tomu, že aplikace není financovaná žádnou institucí, tak k pokrytí nákladů na vývoj používá komerčních sdělení. Tato sdělení se většinou dle mých zkušeností týkají volných pracovních pozic a volnočasových akcí v okolí univerzity. Nové verze tohoto této aplikace také podporují i jiné systémy než IS/STAG, jedná se například o systémy ČVUT. [35]

Níže nalezneme ukázku z prostředí mobilní aplikace (ve verzi pro systém Android):



Obrázek 2: Prostředí mobilní aplikace Univerzita.app (zdroj Google Play) [35]

Podle snímků obrazovek aplikace vidíme, že tato aplikace také nabídne moderní grafické prostřední se střídmějšími barevnou paletou než aplikace přechozí. Aplikace také podporuje světlý i tmavý režim.

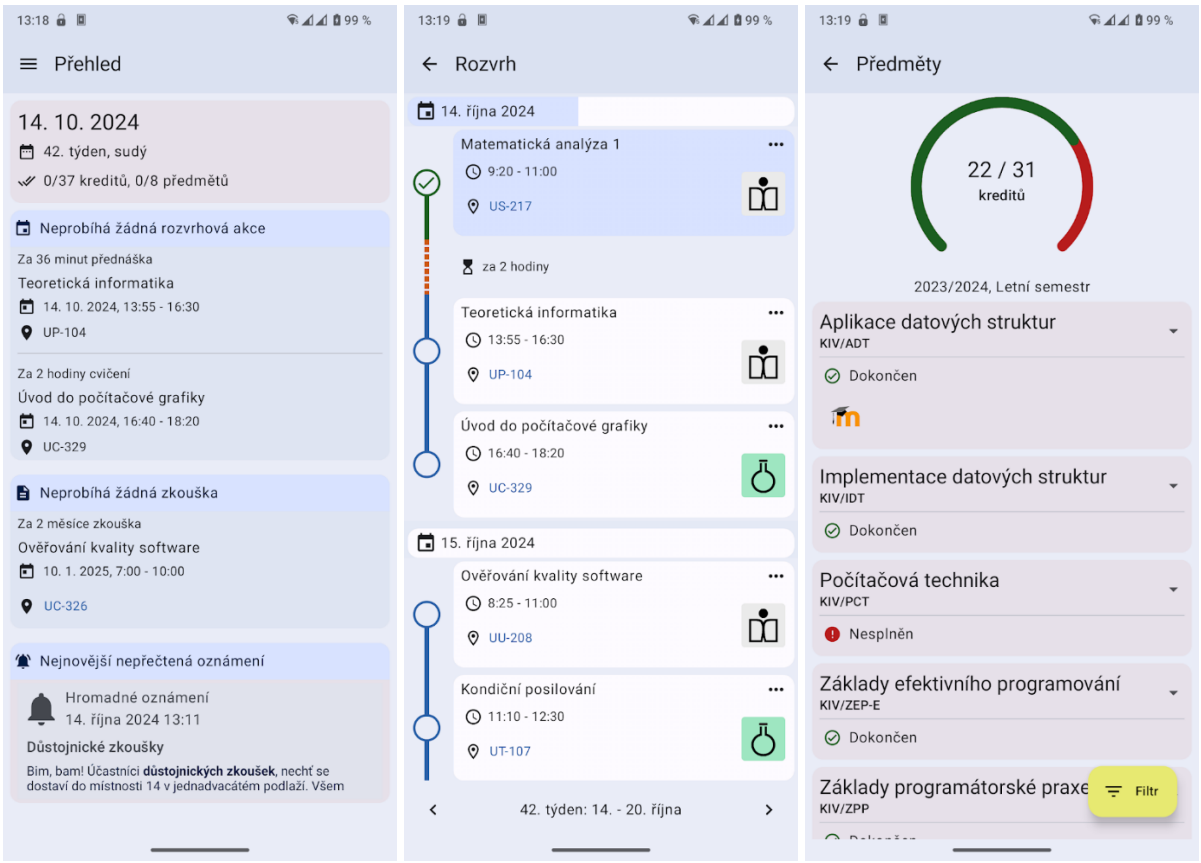
Mezi hlavní funkcionality patří přehledná úvodní obrazovka zobrazující studijní výsledky a aktuální rozvrh. Rozvrh lze zobrazit v různých formátech – jako seznam, tabulku nebo jako časovou agendu pro jednotlivé dny. Aplikace umožňuje vyhledávání studentů, vyučujících, předmětů a zkouškových termínů v rámci celé univerzity (jak jsou uživatelé zvyklí z webového IS/STAG). [35]

Součástí je také profil s detailním přehledem předmětů a známek včetně rozšířených informací o studijních výsledcích. Aplikace podporuje notifikace o probíhajících rozvrhových akcích, nově vypsaných termínech a udělených známkách. K dispozici je také harmonogram akademického roku. Uživatelé si mohou přizpůsobit nastavení aplikace, například intervaly obnovy dat, jazyk, vzhled aplikace nebo synchronizaci na pozadí. Pro vyučující nabízí navíc přehled vedených kvalifikačních prací a možnost zobrazení dříve přijatých oznámení generovaných z IS/STAG. [35]

2.1.3 IS/STAG Mobile

IS/STAG Mobile je oficiální mobilní aplikací pro systém IS/STAG vyvíjená ZČU a je dostupná pro všechny instituce, které systém využívají a je oficiální aplikací univerzity. Dostupnost funkcionalit v této aplikaci se může lišit v závislosti na instituci, ZČU si zřejmě pro své účely do aplikace přidala funkcionality navíc. [36]

Níže nalezneme ukázku prostředí mobilní aplikace (ve verzi pro systém Android):



Obrázek 3: Prostředí mobilní aplikace IS/STAG Mobile (zdroj Google Play) [32]

Podle snímků obrazovek můžeme vidět, že grafické prostřední aplikace není tak vizuálně atraktivní jako u předchozích aplikací a je zde určitý prostor pro zlepšení. Grafické prostředí aplikace pro iOS je dle mého názoru řešeno lépe.

Aplikace nabízí studentům přehledné zobrazení rozvrhových akcí ve formě rozvrhu, seznam všech studovaných předmětů s informacemi o jejich aktuálním stavu a přehled zkouškových předmětů s možností zápisu a odhlášení. Studenti ZČU mají navíc k dispozici funkci zobrazování jídelníčku. Součástí aplikace je také zobrazení oznámení z IS/STAG, která studenty informují o důležitých událostech a změnách. [32] [36]

Pro vyučující aplikace poskytuje přehledné zobrazení rozvrhových akcí ve formě rozvrhu, stejně jako přístup k jídelníčku a notifikacím. Kromě toho umožňuje vyučujícím zasílat hromadná oznámení studentům, a dokonce i zapisovat známky přímo do systému, čímž usnadňuje administrativní správu výuky. [32]

2.2 Návrh aplikace

2.2.1 Sestavení funkčních a nefunkčních požadavků

Na základě funkcionalit existujících aplikací a požadavků univerzity vyplývajících ze zadání této práce jsem sestavil následující seznam funkčních a nefunkčních požadavků na vyvíjenou mobilní aplikaci. Seznam je zpracován ve formě tabulky níže:

Tabulka 2: Seznam funkčních a nefunkčních požadavků

Funkční požadavky	**Nefunkční požadavky**
Přehledová obrazovka	Moderní grafický design
Zobrazení týdenního rozvrhu výuky	Použitý grafického schématu univerzity
Zobrazení termínů zkoušek	Dostupnost off-line
Přihlašování na termíny zkoušek	Známé prostřední z jiných aplikací
Zobrazení sylabu předmětů	Jednoduché nastavení
Zobrazení známek z předmětů	Intuitivnost prostředí
Odeslání hodnocení kvality výuky	Relativně malá instalační velikost
Seznámení studentů prvních ročníků s VŠ	
Odkazy na jiné systémy univerzity	
Podpora pro studenty i vyučující	

(zdroj vlastní)

Mezi hlavní funkční požadavky jsem zařadil přehledovou obrazovku, zobrazení týdenního rozvrhu, termínů zkoušek, sylabu předmětů a dosažených známek. Tyto funkcionality se objevují ve všech analyzovaných aplikacích a lze je tedy považovat za určitý standard, který by neměl chybět v žádné univerzitní mobilní aplikaci. Rozhodl jsem se však rozšířit funkcionalitu aplikace i o prvky, které v dosavadních řešeních často chybí, a které mohou výrazně zvýšit její přínos pro uživatele. Patří mezi ně možnost hodnocení kvality výuky, funkce pro snadnější orientaci nově příchozích studentů ve vysokoškolském prostředí a integrované odkazy na webové stránky a další informační systémy univerzity. Tyto doplňkové funkce nejen rozšiřují spektrum využití aplikace, ale zároveň jí propůjčují jistou míru originality a přidané hodnoty oproti stávajícím řešením.

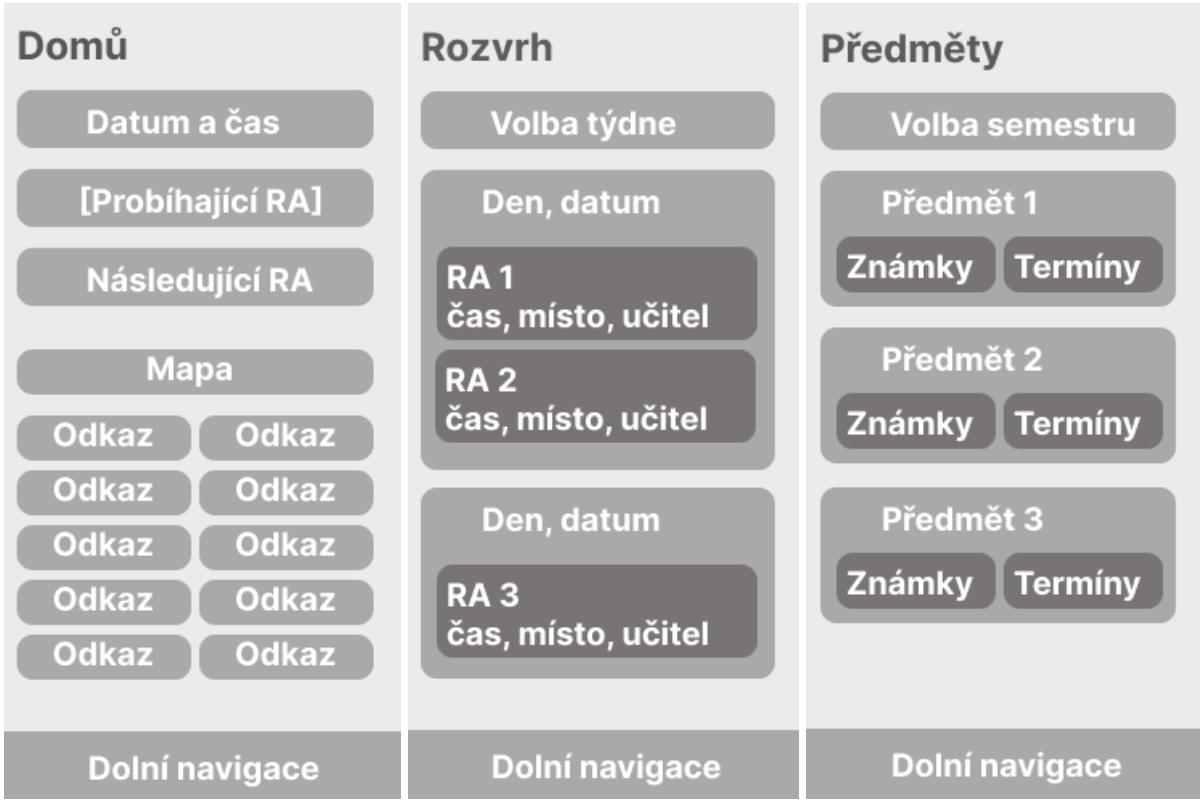
V oblasti nefunkčních požadavků kladu důraz především na moderní grafický design, který bude ladit s vizuálním stylem dané univerzity a tím posílí estetický dojem z aplikace, na který je v dnešní době kladen důraz. Dalším důležitým požadavkem je možnost využívat většinu funkcí i v off-line režimu, což zlepšuje použitelnost aplikace bez ohledu na aktuální připojení k internetu. Uživatelské prostředí by mělo být maximálně intuitivní, s jednoduchým nastavením, aby aplikace byla snadno ovladatelná i pro méně technicky zdatné uživatele. V neposlední řadě je vhodné usilovat o co nejmenší instalační velikost aplikace, která usnadní její rychlé stažení a instalaci i na zařízeních s omezenou kapacitou úložiště.

Výhled do budoucna na funkcionality, které by mohly být v budoucnu přidány jsem popsal v kapitole „Možná další rozšíření aplikace“.

2.2.2 Návrh grafického rozhraní

Na základě funkčních a nefunkčních požadavků jsem sestavil návrh grafického rozhraní mobilní aplikace. Aplikaci jsem rozdělil do pěti hlavních sekcí: domů, rozvrh, předměty, hodnocení (kvality výuky) a profil, které jsou uživatelům přístupné přes dolní navigaci aplikace. Tato navigace zajišťuje konzistentní a rychlý přístup ke klíčovým částem aplikace. Sekce domů bude obsahovat informace aktuálních a nadcházejících událostí, mapu kampusu a odkazy na další systémy. Sekce rozvrh bude obsahovat rozpis výuky podle dnů s informacemi o času, místě a vyučujícím. Sekce předměty bude obsahovat seznam zapsaných předmětů podle semestru s možností zobrazit známky a termíny zkoušek. Sekce hodnocení bude obsahovat seznam rozvrhových akcí pro které je třeba hodnocení vyplnit. Sekce profil bude obsahovat informace o přihlášeném uživateli, popř. nastavení aplikace.

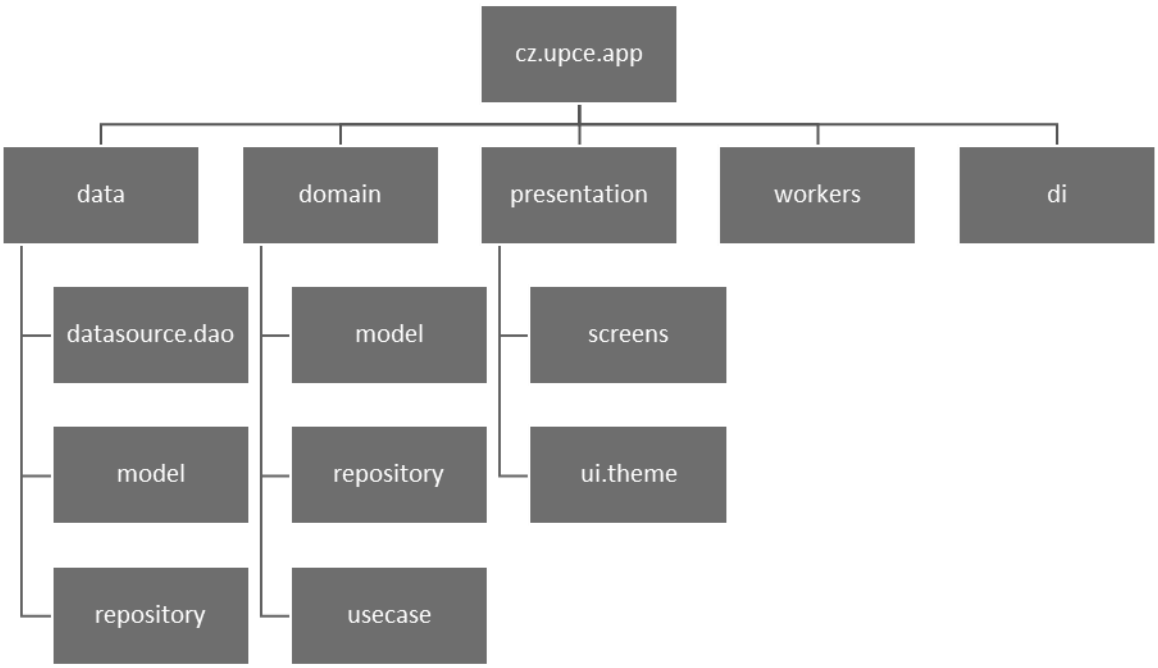
Rozložení grafického rozhraní jsem navrhoval v bezplatném webovém nástroji Figma. Jako inspiraci jsem použil grafické prostředí existujících aplikací, přechod díky známému designu z jiných aplikací může být pro uživatele jednodušší. Rozhraní je navrženo s důrazem na jednoduchost – každý prvek má jasnou funkci, informace jsou členěny do samostatných bloků, což zlepšuje přehlednost. Navržené komponenty odpovídají typickým činnostem studenta a jsou uspořádány tak, aby umožnily jejich snadné provedení. Jelikož se jedná o raný návrh, design ještě neobsahuje konkrétní barvy ani vizuální prvky univerzity. To však umožňuje soustředit se čistě na logiku rozložení.



Obrázek 4: Návrh uživatelského rozhraní vyvíjené mobilní aplikace (zdroj vlastní)

2.2.3 Struktura aplikace

Pro potřeby přehledné a udržovatelné architektury jsem navrhl strukturu balíčků vycházející z principů Clean Architecture, která je široce využívaná při vývoji moderních Android aplikací. Tento přístup rozděluje aplikaci do tří základních vrstev – datové, doménové a prezentační – přičemž každá vrstva má jasně definované odpovědnosti. Struktura kódu pak odráží toto členění prostřednictvím logicky organizovaných balíčků. Nad rámec této základní architektury byly přidány i specializované balíčky pro background workers a dependency injection, které zvyšují modularitu a testovatelnost aplikace. [1]

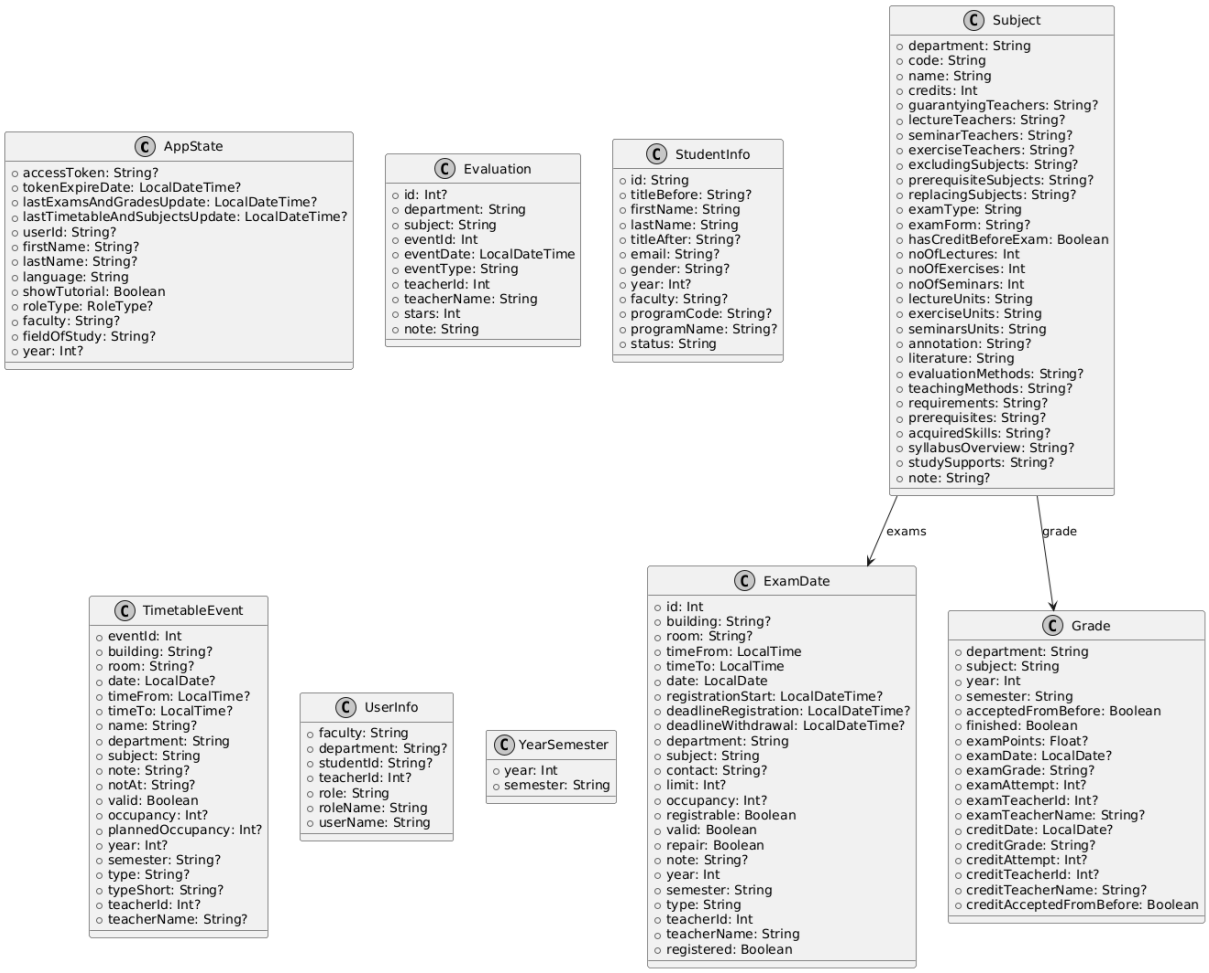


Obrázek 5: Schéma balíčků v projektu (zdroj vlastní)

Datová vrstva obsahuje implementaci přístupu k datům z externích zdrojů (např. síť, databáze). Balíček data.datasource zahrnuje rozhraní pro knihovny Retrofit (síťová komunikace) a Room (lokální databáze). V rámci data.datasource.dao jsou umístěna DAO rozhraní definující přístup k tabulkám v databázi. Balíček data.model obsahuje datové třídy reprezentující síťové a databázové entity, a také tzv. mappery, které slouží k převodu mezi různými datovými reprezentacemi. Implementace datových operací, jež odpovídají rozhraním definovaným v doménové vrstvě, se nachází v balíčku data.repository. [1] [29]

Doménová vrstva představuje srdce aplikace – je nezávislá na konkrétní implementaci datových zdrojů a obsahuje čistou obchodní logiku. Balíček domain.model obsahuje datové třídy, které jsou používány jak v doménové, tak i prezentační vrstvě. Balíček domain.repository definuje rozhraní pro přístup k datům, která jsou následně implementována ve vrstvě data. V balíčku domain.usecase se nacházejí tzv. use cases, které agregují, transformují či jinak připravují data k prezentaci, a zároveň zapouzdřují specifické části logiky aplikace. [1]

Níže nalezneme návrh datových tříd, které patří do doménové vrstvy ve formě UML diagramu:



Obrázek 6: UML diagram tříd na doménové vrstvě (zdroj vlastní)

Na prezentační vrstvě je obsažen veškerý kód zodpovědný za zobrazení dat uživateli. V balíčku presentation.screen se nachází jednotlivé obrazovky aplikace, implementované pomocí Jetpack Compose, členěné dále podle logiky nebo funkce. Balíček presentation.ui.theme obsahuje konfigurační soubory pro vizuální vzhled aplikace tedy barvy, typografii, komponenty, přizpůsobené vizuální identitě univerzity. Navigace mezi obrazovkami je centralizována v kořenové části prezentační vrstvy a umožňuje přehledné řízení pohybu napříč aplikací. [1]

Balíček worker zahrnuje implementace tříd dědících z třídy Worker pro asynchronní zpracování úloh na pozadí (např. synchronizace dat). Balíček di obsahuje definice závislostí a moduly pro knihovnu Koin, která slouží pro dependency injection a podporuje snadnou správu instancí napříč aplikací. V hlavním balíčku cz.upce.app jsou umístěny klíčové prvky aplikace – hlavní aktivita (MainActivity), třída aplikace (Application class) a globálně přístupné konstanty.

2.3 Implementace aplikace

2.3.1 Příprava a konfigurace projektu

V prostředí Android Studio jsem vytvořil projekt typu „Prázdná aktivita“ s minimální podporovanou verzí Android 9 a s použitím nástroje Gradle pro sestavení. Výchozí šablona projektu již obsahovala jednu aktivitu využívající grafické rozhraní založené na Jetpack Compose, takže nebylo nutné přidávat další knihovny pro Compose. Pro integraci potřebných nástrojů a knihoven jsem upravil konfigurační soubory build.gradle.kts a libs.versions.toml, kde se jednotlivé závislosti registrují. [4] [25]

Do projektu jsem přidal knihovnu Navigation Compose, která zajišťuje navigaci v rámci jedné aktivity. Pro komunikaci s REST API jsem použil knihovnu Retrofit, doplněnou o tzv. converter factory, konkrétně GSON od Googlu, který umožňuje automatickou konverzi objektů z/do formátu JSON. Dále jsem implementoval knihovnu Room, která slouží pro práci s lokální SQLite databází a využívá se zejména pro cachování dat z API. Koin jsem zvolil jako řešení pro dependency injection, přičemž jeho základní verze neobsahuje podporu pro Jetpack Compose, a proto bylo třeba přidat i Compose-specific modul. Pro podporu asynchronního zpracování jsem do projektu integroval Kotlin Coroutines včetně Android-specific rozšíření. Nechyběla ani knihovna Material Icons, která obsahuje rozsáhlou kolekci moderních ikon ve stylu Material Design. Nakonec jsem přidal knihovnu Work pro spouštění úloh na pozadí. [37]

Knihovna Room navíc, na rozdíl od ostatních, vyžaduje registraci speciálního build pluginu. Tento plugin se spouští při sestavení projektu, kontroluje správnost SQL syntaxe v anotacích, generuje databázové schéma na základě datových tříd a vytváří pomocné třídy i inicializační SQL skripty, které jsou potřebné pro běh samotné knihovny. [29]

Knihovna Koin také potřebuje specifickou běhovou konfiguraci. Při spuštění aplikace je zapotřebí inicializovat Koin v kódu třídy aplikace pomocí metody startKoin, ta přebírá odkaz na instanci této třídy, aby ji bylo možno později injektovat konstruktorů tříd, které ji potřebují při vytváření svých instancí. Také je třeba provést úpravu manifestu, aby správně fungoval start Koinu pro použití s knihovnou Work. [37]

V souboru AndroidManifest.xml jsem provedl několik úprav. Definoval jsem vlastní třídu aplikace a změnil způsob spouštění aktivit na režim singleTop, který zajišťuje, že daná aktivita nebude znovu vytvořena, pokud již běží, což je důležité pro správné fungování navigace v Jetpack Compose. Kvůli přihlašování přes systém STAG jsem definoval URL schéma

app://upce-app, které zajišťuje přesměrování do aplikace. Použití protokolu app má výhodu v tom, že se neplete s běžnými webovými stránkami a vždy směřuje do aplikace.

Pro správné fungování notifikací v novějších verzích Androidu (od verze S) je nutné vytvořit tzv. notifikační kanály. Vzhledem k tomu, že aplikace zobrazuje pouze jeden typ notifikace, stačí vytvořit jeden kanál s běžnou prioritou. Tento kanál je nejvhodnější registrovat při startu aplikace v rámci její třídy. [6]

2.3.2 Android Resources

Aplikace podporuje vícero jazykových mutací, k tomu používá s tzv. string resources. Tyto datové soubory ve formátu XML obsahují dvojice klíč–hodnota a jsou uloženy ve složce res/values. Výhodou tohoto řešení je, že přepínání jazyků zajišťuje systém sám, a v případě Jetpack Compose dochází k automatickému přizpůsobení uživatelského rozhraní aktuálnímu nastavení zařízení. Vytvořil jsem jazykové varianty pro češtinu a angličtinu, přičemž angličtina je nastavena jako výchozí jazyk. Pokud tedy nějaký překlad chybí, systém použije anglickou verzi. Díky tomu se angličtina zobrazuje všem uživatelům, kteří nemají jako výchozí jazyk v systému nastavenou češtinu. [38]

Pro podporu komplexnějších textů, které jsou použity v průvodci studiem pro nové studenty, kde je potřeba nadpis a text je použit resource typu string array. Soubor pro daný jazyk obsahuje dvě string array: tutrial-headlines a tutorial-texts. U těchto textů se změny načítají za běhu, protože zde není systémová implementace pro Jetpack Compose a musím tak činit ručně.

Grafické prvky, jako jsou obrázky, jsou uloženy ve složce res/drawable, která podporuje jak bitmapové formáty PNG, tak i vektorové XML soubory. V aplikaci jsou využívány pouze dva druhy obrázků – loga univerzity a mapa kampusu. Loga jsou dostupná ve dvou variantách: světlé (barevné) a tmavé (bílé nebo černé). Systém Android automaticky vybírá vhodný obrázek podle aktuálně nastaveného motivu (světlý nebo tmavý režim), přičemž tato volba se může dynamicky měnit i za běhu aplikace.

2.3.3 API rozhraní systému IS/STAG

V aplikaci využívám REST rozhraní webových služeb informačního systému IS/STAG. Základní URL adresa pro přístup k těmto REST endpointům je pro Univerzitu Pardubice https://stag-ws.upce.cz/ws/services/rest2/. Na tuto adresu se následně připojuje název konkrétního endpointu a metody, kterou chce aplikace zavolat. Pro komunikaci s tímto

rozhraním používám datový formát JSON, s výjimkou operací spojených s přihlašováním a odhlašováním z termínů, kde API podporuje pouze prostý text (plaintext). [18]

Rozhraní REST ve variantě IS/STAG má však několik specifických problémů, se kterými je nutné při implementaci počítat. Prvním zásadním nedostatkem je absence oficiální, platné dokumentace výstupních datových struktur. I když je ke každému endpointu dostupný popis ve formátu WADL (na adrese endpointu s příponou ?wadl), tento formát je označen jako experimentální a podle mých zkušeností není spolehlivý – často neodpovídá reálné struktuře odpovědi. Navíc je pravděpodobně generován na základě interního kódu STAGu, který je napsán v Javě. Tento jazyk nepodporuje tzv. nullovatelnost u primitivních typů stejným způsobem jako Kotlin, což výrazně komplikuje práci s datovými typy a zvyšuje riziko chyb při mapování dat.

Další problematickou oblastí je samotná struktura a konzistence datových typů. Ačkoli JSON formát běžně podporuje číselné hodnoty, boolean typy, texty, pole a objekty, IS/STAG API s těmito typy pracuje často nejednotně. Například některé číselné hodnoty jsou předávány jako textové řetězce, boolean hodnoty jsou místo true/false vyjádřeny pomocí písmen A/N, a datumy a časy nejsou ve standardizovaném UTC formátu ISO 8601, ale ve specifickém českém formátu, a navíc zabaleny do objektu, což je zbytečně komplikované. Místo jednotného způsobu reprezentace chybějících hodnot se někdy používá hodnota null, jindy prázdný řetězec. Kromě toho se v rozhraní vyskytuje i nekonzistentní a místy nevhodné pojmenovávání atributů, což dále snižuje přehlednost a srozumitelnost odpovědí.

Pro volání některých metod je navíc nutné být přihlášen a mít přidělenou odpovídající roli. V případě mnou vybraných metod se to týká všech kromě té, která slouží k získání informací o konkrétním předmětu. Na základě katalogu dostupných endpointů a metod jsem s ohledem na funkční požadavky aplikace vybral celkem devět metod rozdělených mezi pět různých endpointů. [18]

Z katalogu dostupných endpointů jsem pro použití v aplikaci vybral následující:

Tabulka 3: Seznam použitých REST endpointů ve vyvíjené aplikaci

Metoda	**Adresa**	**Popis**
GET	terminy/getTerminyProStudenta	Dostupné zkouškové a zápočtové termíny pro přihlášeného studenta

GET	znamky/getZnamkyByStudent	Známky ze zapsaných předmětů přihlášeného studenta
GET	student/getStudentInfo	Informace o přihlášeném studentovi
GET	predmety/getPredmetyByStudent	Seznam studovaných předmětů přihlášeného studenta
GET	rozvrhy/getRozvrhByStudent	Kompletní rozvrh přihlášeného studenta
GET	rozvrhy/getRozvrhByUcitel	Kompletní rozvrh přihlášeného vyučujícího
GET	predmety/getPredmetInfo	Kompletní informace o zadaném předmětu včetně syllabu
GET	terminy/zapisStudentaNaTermin	Zapsání přihlášeného studenta na zkouškový/zápočtový termín
GET	terminy/odhlasStudentaZTerminu	Odepsání přihlášeného studenta ze zkouškového/zápočtového termínu

(zdroj vlastní)

V aplikaci je API rozhraní implementováno pomocí interface data.datasource.StagApi.

2.3.4 API rozhraní hodnocení kvality výuky

V současné době univerzita neposkytuje žádné oficiální API rozhraní, které by umožňovalo zpracování hodnocení kvality výuky způsobem navrženým v této aplikaci. Z tohoto důvodu jsem vytvořil vlastní jednoduché rozhraní, které jsem testoval pomocí bezplatného Mock API serveru. Toto rozhraní v základní podobě nevyužívá žádnou formu autentizace, nicméně vhodným opatřením pro budoucí nasazení by bylo například zavedení statického tokenu. Ten by sice zamezil odesílání hodnocení mimo samotnou aplikaci, avšak nelze tím zaručit naprostou bezpečnost – token by mohl být potenciálně extrahován z aplikace pomocí specializovaných nástrojů.

Navržené API pracuje s objektem hodnocení, který zahrnuje několik základních atributů: identifikátor a typ rozvrhové akce, datum a čas jejího konání, jméno a ID vyučujícího, samotné hodnocení ve škále od 1 do 5 a nepovinné slovní vyjádření názoru. Důležitým prvkem návrhu

je skutečnost, že API neukládá žádné osobní údaje ani identifikátory studenta, čímž je zajištěna skutečná anonymita poskytnutého hodnocení.

Nevýhodou tohoto jednoduchého řešení je však absence kontroly proti duplicitnímu odeslání. Není totiž možné zamezit situaci, kdy uživatel odešle stejné hodnocení vícekrát například z různých zařízení, na kterých má aplikaci nainstalovanou. Tento problém by bylo možné teoreticky částečně omezit například logikou na straně serveru nebo zavedením časového limitu mezi opakovanými hodnoceními.

Toto jednoduché API poskytuje jeden endpoint s následujícími metodami uvedenými v tabulce:

Tabulka 4: Metody API hodnocení kvality výuky

Metoda	**Adresa**	**Popis**
GET	evaluations	Hodnocení výuky pro daný předmět
POST	evaluations	Odeslání nového hodnocení výuky
PUT	evaluations	Úprava existujícího hodnocení výuky

(zdroj vlastní)

V aplikaci je API rozhraní implementováno pomocí interface data.datasource.EvaluationApi.

2.3.5 Zpracovávání dat

Obrázek 7: Schéma zpracovávání dat (zdroj vlastní)

Data jsou pravidelně na pozadí stahována z rozhraní IS/STAG ve formátu JSON pomocí knihovny Retrofit. Ta zajišťuje HTTP komunikaci, parsování odpovědi a její převod do připravených datových tříd, které přesně odpovídají formátu dat poskytovaných API. Následně dochází k transformaci těchto dat pro potřeby uložení do databáze. V tomto kroku jsou odstraněny atributy, které nejsou pro funkčnost aplikace relevantní, čímž se zmenšuje celkový objem zpracovávaných a ukládaných dat.

Uložená data jsou zpracována pomocí knihovny Room, která ve výchozím stavu podporuje pouze základní primitivní typy jako jsou int, float, String a boolean. Při ukládání tedy dochází

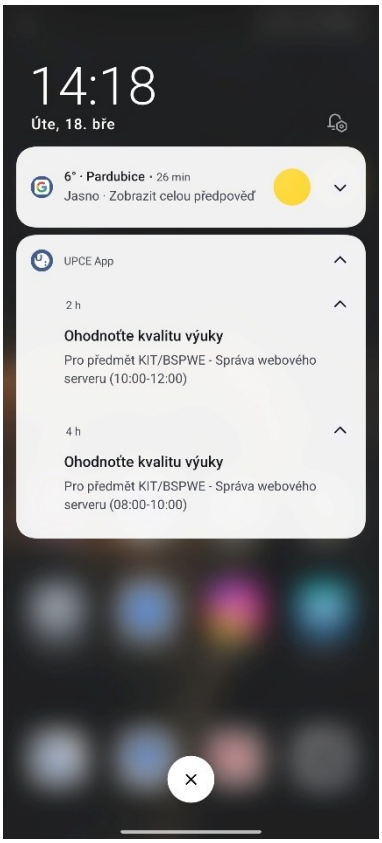
i k převodu složitějších datových struktur a k jejich normalizaci, jak je podrobněji popsáno v kapitole o API rozhraní IS/STAG. Při načítání dat z databáze dochází k jejich opětovné transformaci pro potřeby zobrazení v uživatelském rozhraní – typicky jde o spojování objektů podle referencí a převod řetězcových hodnot, jako je datum nebo čas, na odpovídající objektové reprezentace. Tato data jsou následně dále zpracována v tzv. use case třídách a poté zobrazena v uživatelském rozhraní. [29]

2.3.6 Notifikace a běh na pozadí

Aplikace využívá možnosti provádět úlohy na pozadí, k čemuž slouží knihovna androidx.work. Tato knihovna však ve výchozím stavu není kompatibilní s knihovnou Koin. Z tohoto důvodu bylo nutné provést specifické úpravy – do manifestu aplikace byl přidán provider zakazující automatickou inicializaci knihovny WorkManager a v třídě aplikace byla knihovna inicializována ručně. K tomu byl vytvořen vlastní KoinWorkerFactory, který na základě názvu požadované třídy vytvoří její instanci, přičemž požadované závislosti jsou zajištěny pomocí Koinu. [37]

Běh na pozadí je v aplikaci využíván především pro dvě funkce – aktualizaci dat z IS/STAG a zobrazení notifikací s výzvou k ohodnocení kvality výuky. Pro tyto účely byly vytvořeny tři samostatné třídy tzv. workerů: DataUpdaterWorker, NotificationSchedulerWorker a NotificationWorker.

Proces aktualizace dat je rozdělen na dvě části: první se týká aktualizace rozvrhu a seznamu předmětů (pouze u role studenta), druhá pak aktualizace známek a termínů zkoušek. Aktualizace rozvrhu a předmětů probíhá jednou denně, zatímco známky a zkouškové termíny jsou aktualizovány jednou za hodinu. Toto rozdělení vychází z rozdílných nároků na objem přenesených dat a výpočetní náročnost ovlivňující výdrž baterie. Spouštění úloh probíhá pouze tehdy, pokud je dostupné neměřené připojení k internetu (typicky Wi-Fi). V případě, že podmínky nejsou splněny (například kvůli spořiči baterie), je úloha automaticky odložena o jednu hodinu.



Obrázek 8: Zobrazení notifikací (zdroj vlastní)

Zobrazování notifikací používá dva workery: jeden pro naplánování notifikací a druhý pro jejich zobrazení. Worker pro plánování notifikací se spouští jednou denně. Najde si rozvrhové akce, které se konají další den a pro každou z nich naplánuje spuštění workeru zobrazení notifikace na čas konce rozvrhové akce mínus 10 minut a předá se id této rozvrhové akce. Čas zobrazování notifikace jsem zvolil, aby se zobrazovaly v čase pravděpodobně reálného konce, a ne konce dle rozvrhu, jelikož na některých fakultách univerzity se používají orientační časy a ne reálné.

Pokud se plánování nepodaří spustit, tak se pokus opakuje za 1 hodinu. Není však garantováno, že každý den skutečně proběhne a následující den tak nepřijdou žádné notifikace. Toto může nastat např. v případě, že poslední průběh byl před půlnocí a další se o hodinu odloží na další den, ale pak se notifikace plánují na pozítří a ne zítřek. Worker zobrazování notifikace si při spuštění načte informace o rozvrhové akci a odešle notifikaci. Při kliknutí na notifikaci se zruší a otevře se aplikace s obrazovkou na vyplnění hodnocení kvality výuky. [30]

2.3.7 Přihlašování

Aplikace při svém prvním spuštění vyžaduje přihlášení a bez něj ji nelze používat. Aplikace používá přihlašování přes IS/STAG. Přihlašovací obrazovka obsahuje pokyny. Při kliknutí na tlačítko přihlásit se se otevře webový formulář. Po úspěšném přihlášení je uživatel přesměrován na adresu app://upce-app za ní systém IS/STAG připojí query string obsahující informace nutné k autentizaci. Jedná se o:

- stagUserTicket – token k autentizaci přes API rozhraní,
- stagUserInfo – JSON s informacemi o uživateli a jeho rolích zakódovaný do base64,
- a parametry značené jako zastaralé, které by se již neměly používat. [18]

Tyto údaje jsou v aplikaci uchovávány v objektu AppState, který je spravován pomocnou třídou AppStateHolder a umožňuje reagovat na jejich případné změny. Trvalé uložení dat je řešeno pomocí SharedPreferences. Aplikace rovněž sleduje platnost autentizačního tokenu, který má standardní dobu životnosti 90 dní. Po jejím uplynutí je uživatel vyzván k novému přihlášení.

Zpracování přihlašovacího procesu i notifikací je řešeno pomocí tzv. intents. Přihlášení probíhá formou přesměrování z externího webového prohlížeče, a je tedy nutné zaregistrovat odpovídající intent filter v manifestu aplikace. Naopak zobrazování obrazovky pro hodnocení kvality výuky probíhá uvnitř samotné aplikace, a není tedy potřeba jej explicitně registrovat. Reakce na tyto intenty se zpracovávají ve třídě MainActivity prostřednictvím metod onCreate (v případě, že aktivita ještě neběžela) nebo onNewIntent (pokud aktivita již byla aktivní).

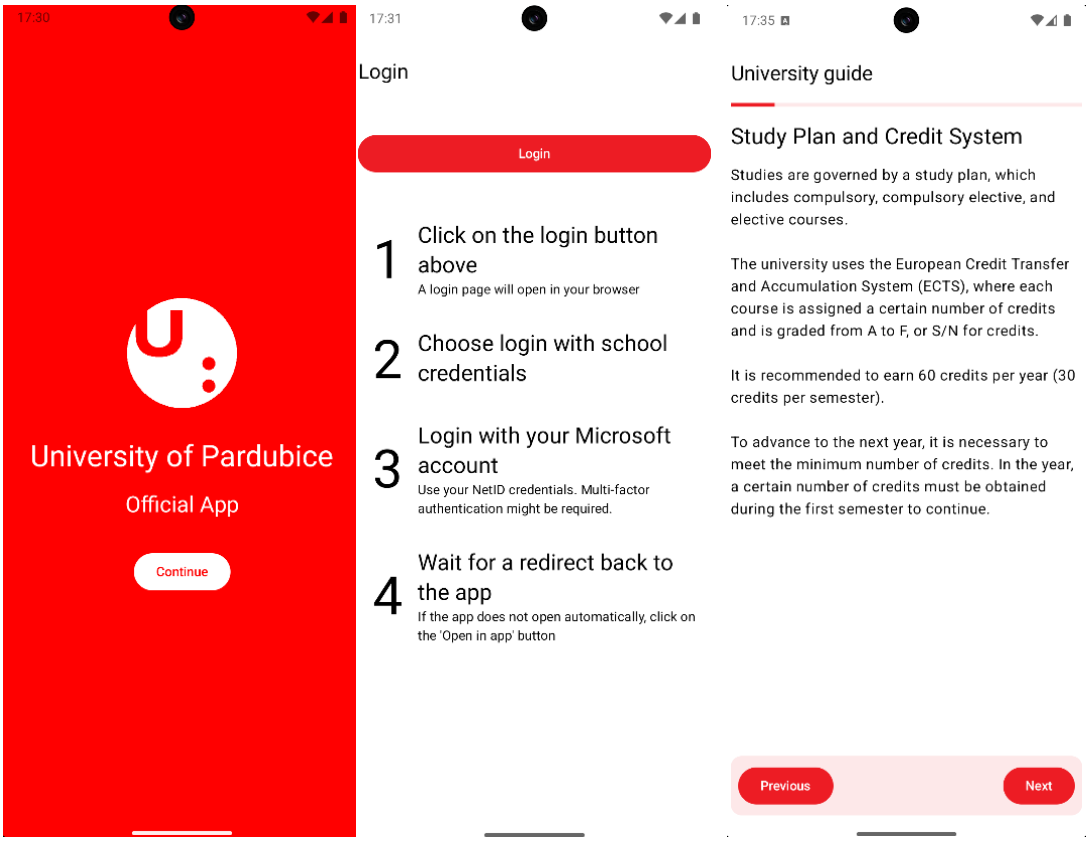
2.3.8 Uživatelské rozhraní

Uživatelské rozhraní jsem vytvořil pomocí knihovny Jetpack Compose. Navigace v aplikaci při prvním spuštění vypadá následovně (v hranatých závorkách jsou kroky, které se za určitých podmínek nemusí zobrazit):

Obrázek 9: Schéma přechodů mezi obrazovkami (zdroj vlastní)

Obrazovky zobrazované při první spuštění aplikace (či při smazání dat aplikace): úvodní obrazovka Vítejte přivítá uživatele v aplikaci, obsahuje logo univerzity a tlačítko pokračovat. Poté následuje obrazovka přihlášení, ta obsahuje tlačítko pro otevření přihlašovacího formuláře a pokyny ve formě číslovaných bodů pro přihlašování. Po kliknutí se otevře přihlašovací stránka

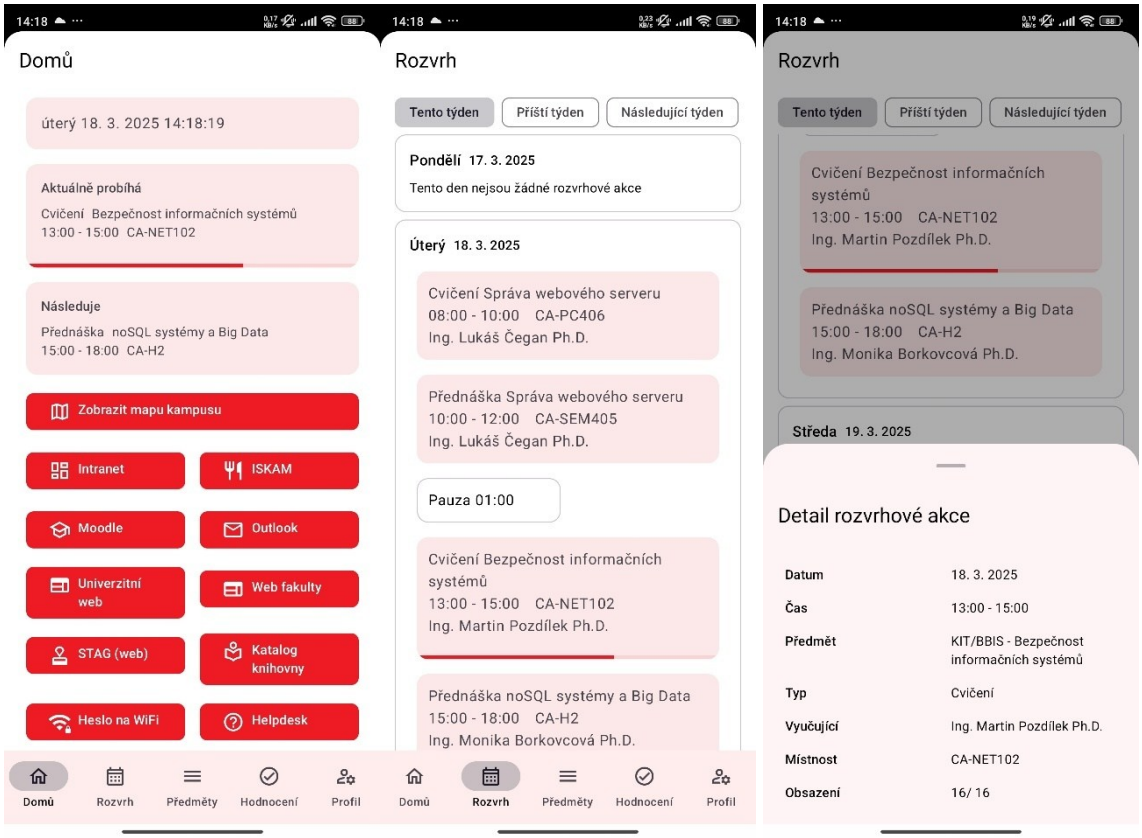
ve výchozím webovém prohlížeči. Při úspěšném přihlášení je uživatel přesměrován zpět do aplikace na obrazovku výběr role, ta se zobrazí uživateli, který má v IS/STAG přidělenou více než 1 roli, v opačném případě se vynechá a nastaví se jediná možná role. Je možné vybrat pouze role student a vyučující, jiné role (studijní referentka, sekretářka katedry, garant…) nejsou touto aplikací podporovány. Poté proběhne úvodní stažení všech potřebných dat pro off-line použití aplikace. Po úspěšném stažení všech dat se zobrazí obrazovka Průvodce studiem pro nové studenty se zobrazí pouze studentům prvních ročníků dva měsíce před a po začátku nového akademického roku (tzn. červenec až říjen).



Obrázek 10: Prostředí vyvíjené aplikace (anglická verze, zdroj vlastní)

Průvodce studiem pro nové studenty je nabízí novým studentům, kteří pravděpodobně nepřišli do styku s vysokoškolským vzděláváním a jsou na této univerzitě poprvé možnost se seznámit s odlišnostmi mezi vysokou školou a střední školou, ze které pravděpodobně přicházejí. Studenti se seznámí s průběhem akademického roku, průběhem výuky, kreditovým systémem (evropský systém ETCS), systémem STAG a tvorbou rozvrhu, s budovami a kampusem univerzity, s možnostmi stravování v univerzitní menze a ubytování na kolejích, s univerzitní knihovnou a možností výjezdu a studia v zahraničí v rámci evropského projektu Erasmus. Také

je seznámí s možnostmi městské, příměstské a vlakové dopravy v Pardubicích a možnostmi parkování u budov univerzity.

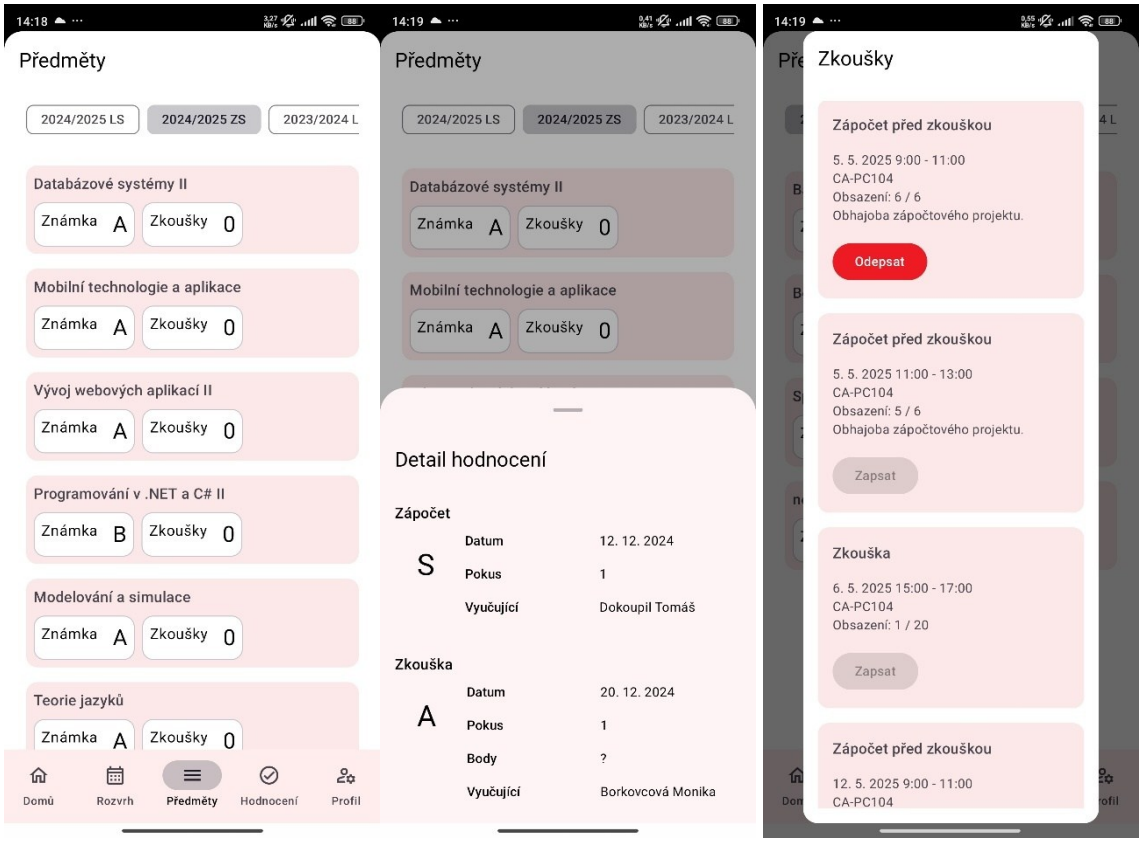


Obrázek 11: Prostředí vyvíjené aplikace (zdroj vlastní)

Hlavní obrazovka Domů, která se zobrazuje po úvodním nastavení aplikace a při jakémkoli dalším spuštění aplikace obsahuje aktuálně důležité informace. V horní části informuje o aktuálně probíhající rozvrhové akci (pokud nějaká taková existuje) a o následující rozvrhové akci. Tyto části lze rozkliknout a zobrazit detail ve vyskakovacím okně. V hlavní části zobrazuje možnost zobrazit mapu kampusu (přímo off-line dostupné v aplikaci) a tlačítka s odkazy a univerzitní informační systémy (otevřou se ve výchozím webovém prohlížeči). Jedná se o intranet (studentský nebo zaměstnanecký podle role uživatele), ISKAM, Moodle, Outlook, STAG, katalog knihovny, univerzitní a fakultní web (podle fakulty uživatele), hesla na síť Eduroam (WiFi, pouze studenti) a ServiceDesk. V dolní části je navigace aplikace přechází se mezi obrazovkami Domů, Rozvrh, Předměty (pouze pro roli student), Hodnocení (kvality výuky, pouze pro roli student), Profil (uživatele).

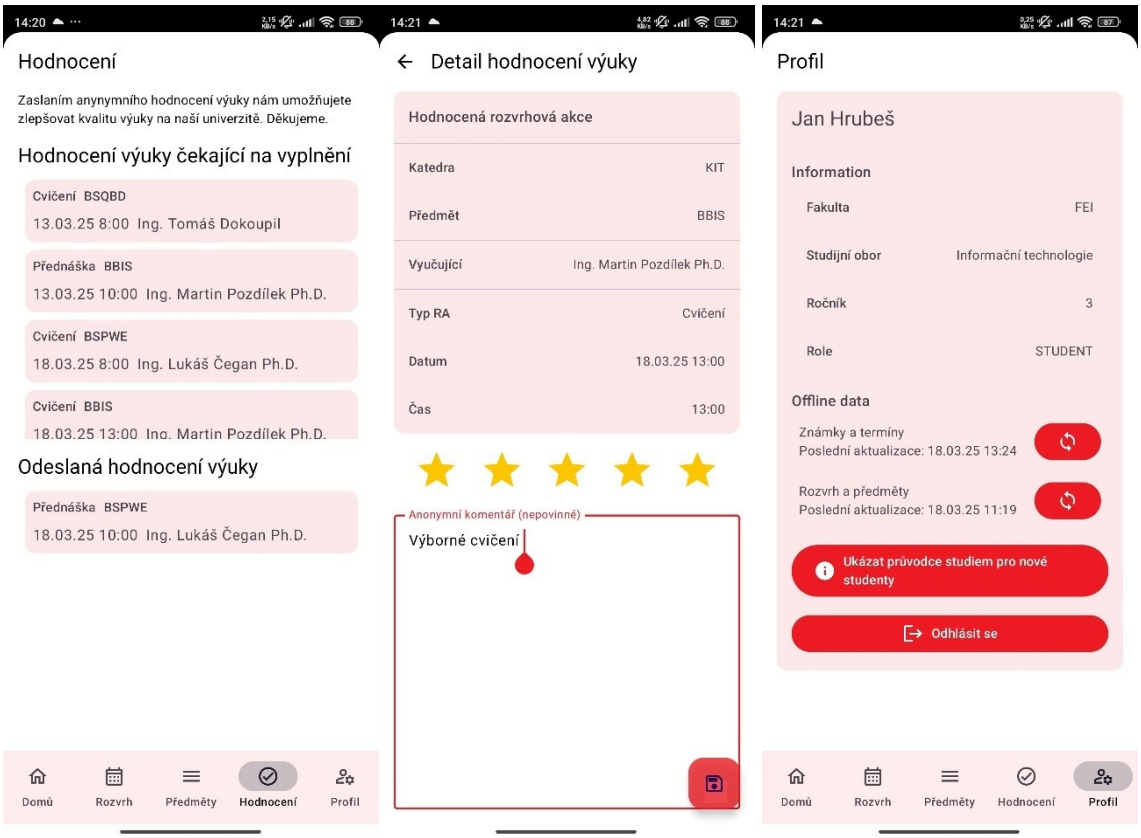
Obrazovka Rozvrh zobrazuje rozvrhové akce v daném týdnu rozdělené podle dnů v týdnu, u nich zobrazuje datum. Je možnost zvolit aktuální týden (výchozí volba), příští týden a následující týden (přespříští týden). Pokud je daný den na sebe rozvrhové akce nenavazují a je

mezi nimi mezera, tak je tato mezera a její délka zobrazeny. Kliknutím na rozvrhovou akci lze zobrazit vyskakovací okno s detaily. Rozvrhovými akcemi se ve smyslu tohoto rozvrhu rozumí kromě výuky také zápočty, zkoušky a jiné akce v rozvrhu.



Obrázek 12: Prostředí vyvíjené aplikace (zdroj vlastní)

Obrazovka Předměty zobrazuje studované předměty podle semestru a akademického roku. U předmětu zobrazí název známku a počet dostupných zkouškových termínů. Kliknutím na známku se zobrazí detail udělených hodnocení ve vyskakovacím okně, a to jak ze zápočtu (pokud ho předmět má), tak i ze zkoušky. Zobrazí datum udělení, jméno hodnotitele, pokus a získané body. Kliknutím na počet termínů se zobrazí vyskakovací okno se seznamem dostupných zkoušek. Zobrazí datum a čas, místnost, obsazení a popisek. Tlačítko zapsat/odepsat je aktivní u termínů, u kterých systém STAG vyhodnotil, že je možné se na něj zapsat/odepsat.



Obrázek 13: Prostředí vyvíjené aplikace (zdroj vlastní)

Obrazovka hodnocení kvality výuky zobrazuje rozvrhové akce, u kterých bylo hodnocení vyplněno a rozvrhové akce, u kterých je ještě možné hodnocení vyplnit. Hodnocení je možné vyplnit u akcí, které proběhli v posledních pěti dnech. Po rozkliknutí se zobrazí další obrazovka a zobrazí se detaily akce a formulář k vyplnění. Míru spokojenosti s kvalitou výuky lze dát najevo pomocí počtu hvězdiček (1 špatné, 5 výborné). Lze volitelně zanechat také slovní hodnocení (může mít více řádků). Kliknutím na tlačítko uložit se odešle a opět se zobrazí seznam hodnocení.

Obrazovka Profil uživatele obsahuje informace o aktuálně přihlášeném uživateli. Zobrazuje se zde stav aktualizace off-line dat a případně lze ručně vynutit aktualizaci. Studenti si mohou kdykoliv zpětně zobrazit průvodce studiem pro nové studenty. Na této obrazovce také probíhá odhlášení z aplikace.

2.3.9 Testování aplikace

Funkčnost aplikace jsem průběžně testoval pomocí ručního testování na zařízeních se systémem Android, tak i částečně pomocí automatizovaných jednotkových (unit) a instrumentálních testů. K testování na virtuálním zařízení jsem používal emulátor obsažený v Android Studiu. Vybral

jsem emulaci Google Pixel s nejnovějším Androidem 15 a univerzální zařízení se starším, ale stále v praxi dosti využívaným Androidem 10. Tyto verze jsem zvolil kvůli jejich odlišnému přístupu k právům na notifikace. Zatímco Android 10 má všechny notifikace povolené, Android 15 je standardně blokuje a pro jejich aktivaci je nutné uživatelské potvrzení prostřednictvím systémového dialogu. [25]

K testování na reálném zařízení jsem využil svůj mobilní telefon značky Xiaomi s operačním systémem Android 14 a nadstavbou HyperOS (dříve MIUI). Tento systém je známý svým specifickým přístupem k řízení oprávnění aplikací a agresivní správou běžících procesů, což z něj činí ideální prostředí pro simulaci reálného používání a identifikaci potenciálních problémů. Na základě mých zkušeností nelze u systémů Android obecně spoléhat na to, že funkčnost ověřená na jednom zařízení se bude shodně projevovat i na zařízení jiného výrobce – rozdíly v nadstavbách mohou být zásadní. [25]

Pro testování API rozhraní systému IS/STAG jsem využil testovací instanci provozovanou Západočeskou univerzitou (ZČU). Tento server je určen pro vývoj a školení uživatelů systému a umožňuje simulaci různých uživatelských rolí a scénářů, které by jinak na produkčním serveru nebylo možné testovat. Nicméně, testování na tomto serveru přináší i určitá omezení. Server se každou noc automaticky resetuje do výchozího stavu, což znamená, že veškeré změny provedené během testování jsou nenávratně ztraceny. Dále může jakýkoli uživatel upravit konfiguraci serveru, smazat data nebo způsobit jeho dočasnou nedostupnost. V pravidelných intervalech dochází také k regeneraci dat (pravděpodobně jednou týdně), což negativně ovlivňuje stabilitu automatizovaných testů. Navíc autentizační relace, která by standardně měla být platná 90 dní, bývá při nočním resetu předčasně ukončena – v praxi docházelo k odhlášení již po několika hodinách. [20]

Automatizované testy jsem využil pro ověření komunikace jak s API rozhraním IS/STAG, tak i s API pro hodnocení kvality výuky. Pro tyto účely jsem vytvořil dvě třídy jednotkových testů, které ověřují správnou autentizaci pomocí tokenu, úspěšné stažení dat a jejich převod na aplikační objekty. Dále jsem implementoval instrumentální testy, které probíhají přímo na zařízení s Androidem. Tyto testy slouží k ověření správného načtení, transformace a uložení dat do databáze, kterou v aplikaci reprezentuje knihovna Room. Testování na fyzickém zařízení je v tomto případě důležité, protože Room je optimalizována právě pro běh v prostředí skutečného zařízení a umožňuje zároveň testovat interakce s různými konfiguracemi systému. [29]

Pro ruční testování Android studio nabízí několik nástrojů, kterých jsem využil, a kterými lze (většinou v reálném čase) sledovat dění v systému. Nabízí například možnost sledovat síťovou aktivitu a analyzovat pomocí toho http požadavky na API rozhraní. Dále je například možnost v reálném čase sledovat změny a procházek a dotazovat databázi, nad kterou běží knihovna Room. Při běhu úloh na pozadí se hodí nástroj, který zobrazí všechny naplánované úlohy. Je zde i nástroj pomocí kterého lze procházet souborový systém na zařízení a přistoupit k soukromým datům aplikace. [21]

2.4 Možná další rozšíření aplikace

Vyvíjená aplikace je díky své architektuře navržena tak, aby umožňovala snadné budoucí rozšíření. Dodržování principu oddělení závislostí (separation of concerns) mezi jednotlivými komponentami přispívá k udržitelnosti a modulárnosti celého systému.

Jednou z potenciálních oblastí rozšíření je modul pro hodnocení kvality výuky. V aktuální verzi aplikace je možné hodnocení odesílat pouze anonymně a přímo v mobilní aplikaci. Z důvodu zachování anonymity není technicky zcela zamezeno opakovanému hodnocení jedním uživatelem. Do budoucna by bylo možné vytvořit systém, který by umožnil odesílání anonymního hodnocení a současně zajistil, aby každý uživatel mohl hodnotit pouze jednou. Dále by bylo vhodné rozšířit funkcionalitu i o přístup k výsledkům hodnocení – ať už pro samotné vyučující, tak pro garanty předmětů, vedoucí kateder nebo fakultní management. Tato funkcionalita by si však vyžádala vývoj komplexnější webové aplikace s navazujícím API, což přesahuje rámec této bakalářské práce.

Dalším možným rozšířením je integrace systémů pro zobrazování novinek, oznámení a událostí z webových stránek univerzity, fakult či intranetu (studentského i zaměstnaneckého). Tato upozornění by mohla být zobrazována prostřednictvím notifikací v aplikaci. V současnosti musejí uživatelé tyto portály aktivně navštěvovat, což je nepraktické a vede k tomu, že část důležitých informací může zůstat nepovšimnuta. Integrovaný informační kanál by výrazně zefektivnil komunikaci mezi univerzitou a jejími studenty či zaměstnanci.

Další užitečnou funkcí by mohla být integrace systému pro objednávání stravy v univerzitní menze. Uživatelé by se tak mohli vyhnout nutnosti pravidelného přihlašování na webovém portálu a vše by mohli pohodlně spravovat přímo v mobilní aplikaci. Cílem je, aby uživatel aplikaci prakticky vůbec nemuseli opouštět a veškeré důležité služby měli k dispozici na jednom místě.

Teoreticky se nabízí také možnost integrace API služeb společnosti Microsoft, které univerzita aktivně využívá. Vzhledem k vysoké implementační náročnosti a existenci oficiálních aplikací pro tyto účely však nemusí být plná integrace efektivní. Alternativní přístup by mohl spočívat v přesměrování na konkrétní aplikaci (např. Teams, Outlook), pokud je nainstalována v zařízení uživatele.

V neposlední řadě by bylo možné implementovat další části API systému IS/STAG určené zejména pro vyučující a garanty předmětů, jako je například zadávání známek, vypisování termínů nebo správa studentů. Některé zajímavé funkcionality pro studenty (např. zápis rozvrhu) však aktuálně nelze implementovat kvůli absenci veřejně dostupného API. V takových případech je třeba zvážit, zda je jejich implementace v mobilní aplikaci vůbec praktická, nebo zda by nebylo vhodnější využití responzivního webového rozhraní.

Obecně lze konstatovat, že současným trendem je centralizace služeb do jedné aplikace. Tento přístup si postupně osvojují i univerzity, a není důvod, proč by se vývoj v tomto směru neměl dále rozvíjet i na úrovni této aplikace.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a implementovat mobilní aplikaci pro operační systém Android, která by sloužila jako moderní nástroj pro přístup k univerzitnímu informačnímu systému IS/STAG. Aplikace je určena především pro studenty, ale i zaměstnance vysoké školy a jejím hlavním záměrem je nabídnout uživatelům jednoduchý a rychlý přístup k důležitým studijním informacím a funkcím, jako je rozvrh, přihlášky na zkoušky, hodnocení kvality výuky a další.

V teoretické části byly popsány klíčové aspekty vývoje aplikací pro platformu Android, včetně architektury, komponentového modelu a správy životního cyklu aplikace. Dále byl představen programovací jazyk Kotlin, knihovny a nástroje, které byly v rámci vývoje využity. Důležitým prvkem bylo také seznámení s informačním systémem IS/STAG a jeho rozhraním pro komunikaci s externími aplikacemi pomocí webových služeb.

V praktické části práce byla provedena analýza existujících aplikací nad IS/STAG a následně navržena vlastní aplikace. Součástí této fáze byl návrh funkčních a nefunkčních požadavků, struktury aplikace, uživatelského rozhraní a pak konkrétní implementace jednotlivých funkcionalit. Byly popsány i obtíže, které mě provázely při vývoji této aplikace zejména pak při integraci API rozhraní. Výsledná aplikace umožňuje uživatelům pohodlný přístup k informacím z IS/STAG, přihlašování pomocí univerzitního účtu, zobrazení rozvrhu, práce s notifikacemi či off-line přístup k datům.

Vývoj aplikace byl realizován s důrazem na přehlednost, udržitelnost a moderní vývojové postupy. Při implementaci byly využity aktuální technologie, jako je Jetpack Compose pro tvorbu UI, architektura MVVM, knihovny Koin pro správu závislostí či Retrofit pro komunikaci s REST API.

Aplikace představuje funkční základ, který může být dále rozvíjen. Mezi možnosti budoucího rozšíření patří například integrace dalších částí IS/STAG, API jiných informačních systémů, rozšířená podpora notifikací nebo přizpůsobení aplikace potřebám konkrétních fakult. Důležitým krokem by rovněž mohlo být zveřejnění aplikace a získání zpětné vazby od reálných uživatelů, která by umožnila další optimalizaci a vylepšení.

Výsledky dosažené v této práci ukazují, že vývoj mobilní aplikace napojené na univerzitní informační systém je nejen možný, ale i prakticky využitelný a přínosný. Vzniklá aplikace může usnadnit přístup ke studijním informacím a zefektivnit každodenní práci.

POUŽITÁ LITERATURA

[1] *Guide to app architecture*. Online. 2025. Dostupné z: https://developer.android.com/topic/architecture. [cit. 2025-04-20].

[2] LACKO, Ľuboslav. *Mistrovství - Android*. Computer Press, 2017. ISBN 978-80-251-4875-4.

[3] LACKO, Ľuboslav. *Vývoj aplikací pro Android*. 2015. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[4] MOSKALA, Marcin. *Android Development with Kotlin*. 2017. Packt Publishing, 2017. ISBN 978-1-78712-368-7.

[5] *Components of an Android Application*. Online. 2025. Dostupné z: https://www.geeksforgeeks.org/components-android-application/. [cit. 2025-04-20].

[6] *Permissions on Android*. Online. 2025. Dostupné z: https://developer.android.com/guide/topics/permissions/overview. [cit. 2025-04-20].

[7] *App manifest overview*. Online. 2025. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro. [cit. 2025-04-20].

[8] *Application fundamentals*. Online. 2025. Dostupné z: https://developer.android.com/guide/components/fundamentals. [cit. 2025-04-20].

[9] *Distribute Your Apps & Games on Google Play*. Online. 2025. Dostupné z: https://developer.android.com/distribute. [cit. 2025-04-20].

[10] *Test apps on Android*. Online. 2025. Dostupné z: https://developer.android.com/training/testing. [cit. 2025-04-20].

[11] *Kotlin Programming Language*. Online. 2025. Dostupné z: https://kotlinlang.org/. [cit. 2025-04-20].

[12] *Kotlin (programming language)*. Online. 2025. Dostupné z: https://en.wikipedia.org/wiki/Kotlin_(programming_language). [cit. 2025-04-20].

[13] JEMEROV, Dmitry a ISAKOVA, Svetlana. *Kotlin in action*. Shelter Island: Manning, 2017. ISBN 978-1-61729-329-0. Dostupné také z: https://kramerius.techlib.cz/kramerius-web-client/uuid/uuid:82cdff2a-c59d-41b3-8051-ee907602cace.

[14] *Kotlin Docs*. Online. 2025. Dostupné z: https://kotlinlang.org/docs/home.html. [cit. 2025-04-09].

[15] *Kotlin Multiplatform*. Online. 2025. Dostupné z: https://kotlinlang.org/docs/multiplatform.html. [cit. 2025-04-20].

[16] *IS/STAG*. Online. 2025. Dostupné z: https://cs.wikipedia.org/wiki/IS/STAG. [cit. 2025-04-20].

[17] *IS/STAG - Informační systém studijní agendy*. Online. 2025. Dostupné z: https://is-stag.zcu.cz/. [cit. 2025-04-20].

[18] *Webové služby nad IS/STAG*. Online. 2025. Dostupné z: https://is-stag.zcu.cz/napoveda/web-services/ws_ws.html. [cit. 2025-04-20].

[19] *IS/STAG - Webové služby*. Online. 2025. Dostupné z: https://is-stag.zcu.cz/zajemci/pristupy/webservices.html. [cit. 2025-04-20].

[20] *Portál DEMO ZCU*. Online. 2025. Dostupné z: https://stag-demo.zcu.cz/. [cit. 2025-04-20].

[21] *Android Studio*. Online. 2025. Dostupné z: https://developer.android.com/studio. [cit. 2025-04-20].

[22] *Android Studio*. Online. 2025. Dostupné z: https://en.wikipedia.org/wiki/Android_Studio. [cit. 2025-04-20].

[23] *Jetpack Compose UI App Development Toolkit*. Online. 2025. Dostupné z: https://developer.android.com/compose. [cit. 2025-04-20].

[24] *Layouts in views*. Online. 2025. Dostupné z: https://developer.android.com/develop/ui/views/layout/declaring-layout. [cit. 2025-04-20].

[25] WAMBUA, Madona. *Modern Android 13 Development Cookbook: Over 70 recipes to solve Android development issues and create better apps with Kotlin and Jetpack Compose*. 2023. Packt Publishing Ltd, 2023, 2023. ISBN 987-1-80323-557-8.

[26] *Koin - The pragmatic Kotlin Injection Framework*. Online. 2025. Dostupné z: https://insert-koin.io/. [cit. 2025-04-20].

[27] *Retrofit*. Online. 2025. Dostupné z: https://square.github.io/retrofit/. [cit. 2025-04-20].

[28] *JSON To Kotlin Class (JsonToKotlinClass)*. Online. 2025. Dostupné z: https://plugins.jetbrains.com/plugin/9960-json-to-kotlin-class-jsontokotlinclass-. [cit. 2025-04-20].

[29] *Save data in a local database using Room*. Online. 2025. Dostupné z: https://developer.android.com/training/data-storage/room. [cit. 2025-04-20].

[30] *WorkManager*. Online. 2025. Dostupné z: https://developer.android.com/jetpack/androidx/releases/work. [cit. 2025-04-20].

[31] *Unizone*. Online. 2025. Dostupné z: https://www.unizone.cz/. [cit. 2025-04-20].

[32] *IS/STAG Mobile*. Online. 2025. Dostupné z: https://play.google.com/store/apps/details?id=org.zimmma.isstag&hl=cs. [cit. 2025-04-20].

[33] *Univerzita App*. Online. 2025. Dostupné z: https://univerzita.app/. [cit. 2025-04-20].

[34] *TULapp*. Online. 2025. Dostupné z: https://play.google.com/store/apps/details?id=cz.unizone.tul&hl=cs. [cit. 2025-04-20].

[35] *Univerzita.app*. Online. 2025. Dostupné z: https://play.google.com/store/apps/details?id=cz.weissar.university&hl=cs. [cit. 2025-04-20].

[36] *IS/STAG - Mobilní aplikace*. Online. 2025. Dostupné z: https://is-stag.zcu.cz/zakaznici/mobilni_aplikace/. [cit. 2025-04-20].

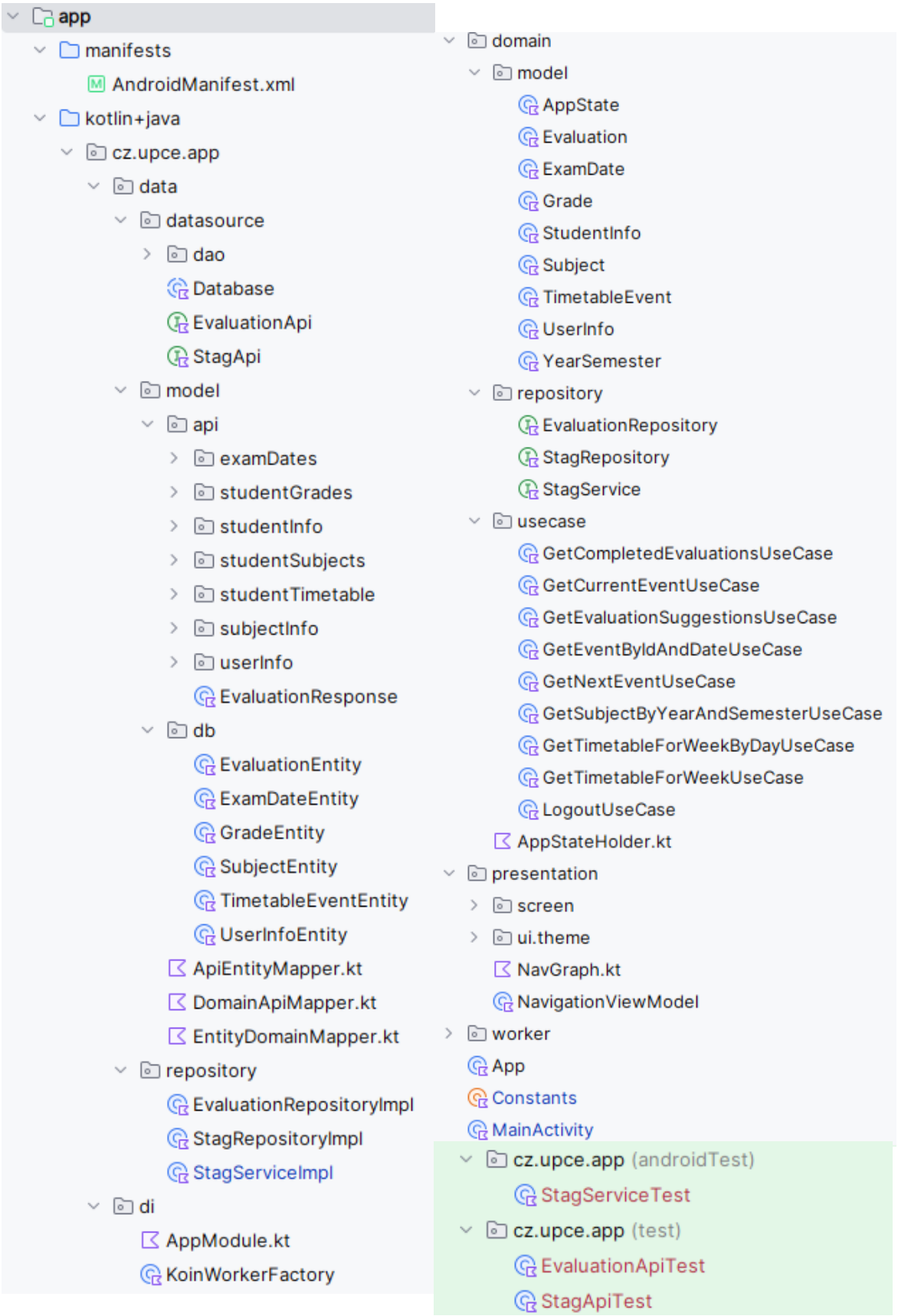
[37] *Start Koin on Android*. Online. 2025. Dostupné z: https://insert-koin.io/docs/reference/koin-android/start. [cit. 2025-04-20].

[38] *App resources overview*. Online. 2025. Dostupné z: https://developer.android.com/guide/topics/resources/providing-resources. [cit. 2025-04-20].

SEZNAM PŘÍLOH

Příloha A: Struktura projektu vyvíjené aplikace

Příloha B: Zdrojové kódy mobilní aplikace

Příloha C: Instalační balíček mobilní aplikace

PŘÍLOHA A: Struktura projektu vyvíjené aplikace