Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Mobilní aplikace IS/STAG na platformě iOS

Plzeň 2022 Čarnogurský Jan

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jan ČARNOGURSKÝ**
Osobní číslo:	**A19N0025P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Mobilní aplikace IS/STAG na platformě iOS**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Nastudujte možnosti vývoje mobilních aplikací na platformě iOS.
2. Seznamte se s vybranými existujícími mobilními aplikacemi nad systémem IS/STAG a se způsoby jejich komunikace s tímto systémem.
3. Vyberte nástroje, knihovny a aplikační frameworky vhodné pro vývoj mobilní aplikace napojené na IS/STAG.
4. Navrhněte aplikaci pro mobilní přístup k IS/STAG pro různé uživatelské role.
5. Navrženou aplikaci realizujte, nasaďte do testovacího provozu alespoň na ZČU a otestujte důkladně její funkčnost.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Ing. Tomáš Kotouč**
Centrum informatizace a výpočetní techniky

Konzultant diplomové práce: **Ing. Ladislav Pešička**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **10. září 2021**
Termín odevzdání diplomové práce: **19. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2021

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 18. května 2022

Čarnogurský Jan

Poděkování

Rád bych touto cestou poděkoval Ing. Tomášovi Kotoučovi a Ing. Ladislavu Pešičkovi za odborné vedení mé práce, podnětné připomínky a korekci textu.

Abstract

This thesis deals with the implementation of a mobile application for the iOS platform connected to IS/STAG web services. The theoretical part is dedicated to the introduction of mobile operating systems. This is followed by a detailed focus on the issue of mobile application development for the iOS operating system. In the next section, existing mobile applications connected to IS/STAG are explored. This is followed by the design of the mobile application concept, which includes detailed design, implementation and distribution. Attention is also paid to testing the application.

Abstrakt

Tato diplomová práce se zabývá realizací mobilní aplikace pro platformu iOS napojené na webové služby IS/STAG. Teoretická část se věnuje seznámení se s mobilními operačními systémy. Následuje detailní zaměření se na problematiku vývoje mobilních aplikaci pro operační systém iOS. V další části jsou prozkoumány již existující mobilní aplikace napojené na IS/STAG. Následně navazuje návrh koncepce mobilní aplikace, který obsahuje detailní návrh, jeho implementaci a distribuci. Pozornost je věnována i testování aplikace.

Obsah

1 Úvod

Studenti v dnešní době využívají ke svému studiu mobilní telefony i počítače. Jelikož digitalizace se rozrůstá, telefony a počítače nám usnadňují život hned v několika směrech. Tím nejdůležitějším pro mladé lidi je samozřejmě studium. V současné době existují mobilní aplikace pro IS/STAG, které umožňují zobrazit přehled studia, avšak ne vždy jsou plně funkční, přehledné nebo podporují pouze roli studenta, nikoli roli učitele.

Na základě statistických dat (tabulka 3.1), poskytnutých CIV, vyplývá, že webový portál ZČU využívá v průměru za rok 2021 o 1,1 % více zařízení s operačním systémem *iOS* než *Android*. Aplikace pro platformu *Android* byla již vyvinuta. Komunikuje s webovými službami IS/STAG a má umožnit studentům a zaměstnancům vysokých škol, provozujících IS/STAG, dostupněji využívat vybrané funkce IS/STAG, jako například rozvrh, přehled zapsaných termínů zkoušek nebo jídelníček v menze. Jak jsem již zmínil, tato aplikace funguje pouze pro zařízení s operačním systémem *Android*, nikoli pro zařízení s operačním systémem *iOS*. Z tohoto důvodu jsem se rozhodl takovou aplikaci v rámci diplomové práci navrhnout a vytvořit.

Cílem této práce bylo nastudovat možnosti vývoje mobilních aplikací na platformě *iOS* a seznámit se s již existujícími aplikacemi pracujícími se systémem IS/STAG, a také se způsoby jejich komunikace s tímto systémem. Na základě výsledků tohoto zkoumání bylo záměrem vybrat vhodné nástroje, knihovny a aplikační frameworky pro vývoj mobilní aplikace napojené na IS/STAG. Nakonec byla navržena aplikace pro platformu *iOS*, která bude mít přístup k IS/STAG pro různé uživatelské role. Aplikace prochází důkladným testováním s využitím *TestFlight* a následně bude po domluvě s CIV nasazena na ZČU.

2 IS/STAG

IS/STAG (Informační systém studijní agendy) je informační systém určený pro administraci studia na vysoké škole, který pokrývá veškeré funkce s ním spojené, od přijímacího řízení až po vydání diplomu. Informační systém umožňuje vést seznamy studentů jak prezenční, tak i kombinované formy studia, seznamy studentů celoživotního vzdělávání i účastníků univerzity třetího věku.

Vůbec první nasazení IS/STAG bylo provedeno v roce 1993 na Západočeské univerzitě v Plzni, kde také tento systém vznikl. Aktuálně IS/STAG používá 13 vysokých škol v České republice, z toho je 11 veřejnoprávních vysokých škol a 2 soukromé vysoké školy.
Jak je uvedeno v popisu IS/STAG [20], mezi základní objekty a moduly systému patří:

- *Studijní programy, obory, plány a předměty* — evidence těchto objektů pro různé typy a formy studia. Je možné vést více verzí studijních plánů, jejich segmentů a bloků předmětů. Lze vést i kombinace studijních oborů (zejména pro pedagogická studia). Systém umí studijní plány přehledně vizualizovat.

- *Student* — kompletní evidence studenta. Osobní údaje, zápis na studijní program, roční zápisy na studijní plány, přerušení, krátkodobé výjezdy, matriční data. Možnost hromadného zápisu studentů do studia, hromadných zápisů na předměty či rozvrhové akce, atd..

- *Přijímací řízení* — lze vypsat přijímací obory a pro ně definovat sledované předměty z předchozího studia a testy. Pomocí zadávaných vzorců a parametrů umí nastavit různá pravidla pro přijetí a vyhodnotit, který z uchazečů je splnil. Uchazeč může zasílat papírovou přihlášku nebo ji podat prostřednictvím webové aplikace elektronické přihlášky. Generování zvacích dopisů a rozhodnutí včetně dopisních štítků.

- *Rozvrhy* — pro jednotlivé předměty lze vytvářet rozvrhové akce, které mohou být přiřazeny k určitému časoprostoru (místnost, čas) nebo mohou být aposteriorní (bez určení místnosti). U rozvrhové akce je možné nastavit, pro jaké studenty je určena (fakulta, typ či forma studia, studijní program, místo studia, atd.). Systém umožňuje vytvářet předzápisové kroužky a k nim přiřazovat rozvrhové akce. V rámci

rozšíření lze využít grafické prostředí pro tvorbu rozvrhu. Je možné přenášet data do a z jiných systémů pro tvorbu rozvrhu.

- *Předzápis* — zápis studenta na další akademický rok, včetně kontroly plnění podmínek studia. Student si může vybrat kroužek a tím se mu zapíší všechny rozvrhové akce kroužku nebo si může předměty vybírat sám. Systém studentovi nabízí předměty jeho studijního plánu, včetně kontrol jeho plnění. Student si může vybrat i předměty mimo svůj studijní plán.

- *Zkoušky* — je možné vypisovat zkouškové termíny, na které se pak studenti hlásí prostřednictvím aplikace ve webovém rozhraní. Hodnocení studentů předmětu lze ukládat v závislosti na termínu, rozvrhové akci či přímo. Všechny tyto operace může provádět dle nastavení učitel i sekretariát katedry.

- *Semestrální práce* — evidence a vypisování témat semestrálních prací, jejich výběr studenty, odevzdávání prací elektronicky, hodnocení prací učiteli, přenos prací do Odevzdej.cz a zjišťování plagiátorství.

- *Mobility studentů* — podpora krátkodobých výjezdů a příjezdů studentů. Zobrazování údajů o studijních plánech a předmětech v rámci ECTS, podpora procesu mobilit a tisk potřebných mezinárodních dokumentů.

- *Evaluace* — hodnocení kvality studia studenty formou vypisování a vyplňování online dotazníků. Následně sběr, statistické zpracování dat z dotazníků a proces zveřejnění a komentování výsledků.

- *Předpisy plateb* — možnost příprav předpisů plateb různých druhů (stipendia, poplatky za studium, poplatky za přijímací řízení, atd.) tedy závazky i pohledávky a přiřazování studentů na ně — ručně nebo pomocí uživatelsky definovatelného vzorce, případně plně automaticky (u přijímacího řízení po zadání přihlášky). Systém poskytuje informace pro výplaty bankou či pokladnou. Systém dovede načíst data o realizovaných úhradách z ekonomického informačního systému a provést automatické spárování.

- *Absolvent* — evidence absolventů, kvalifikačních prací, závěrečných zkoušek, tisk příslušných dokumentů (zadání vysokoškolské kvalifikační práce, zápis o závěrečné zkoušce, diplom, diploma supplement, certifikát, osvědčení atd.).

2.1 Webové služby

K IS/STAG je možné přistupovat přes nativního klienta, webové rozhraní či portál. Další alternativou jsou webové služby.

Webová služba je softwarový modul, který je dostupný přes internet a pro komunikaci a výměnu informací používá standardizované webové protokoly HTTP a HTTPS. Vyměňovaná data jsou nejčastěji ve formátu XML anebo JSON. Pokud jsou splněná ještě další kritéria, můžeme přístup nazvat SOAP pro výměnu dat pomocí XML a REST při výměně dat pomocí JSON.

IS/STAG poskytuje okolo 307 webových služeb, které tvoří tenkou vrstvu nad samou databází systému. Seznam dostupných webových služeb je k dispozici na adrese https://stag-ws.zcu.cz/ws/web.

Mezi nejzákladnější webové služby je možné zařadit: získání rozvrhu, přehled vypsaných zkoušek, informace o uživateli, informace o rozvrhových akcích, a tak dále. Informace jsou spojeny s konkrétními uživateli, a proto je nutné webovou službu zavolat s určitými parametry. Pokud jsou volány webové služby spojené se studenty, požadavek ve většině případů musí obsahovat parametr *osCislo*, který představuje osobní číslo studenta. V případě webových služeb pro učitele je nutné zadat parametr *ucitIdno*, který reprezentuje identifikační číslo učitele. Obecné webové služby požadují jako parametr *stagUser*. Ten znázorňuje přihlašovací jméno uživatele.

2.1.1 Ověření uživatele

Webové služby IS/STAG mohou být přístupné pouze pro přihlášené uživatele, popřípadě mohou být omezené pouze na určité role v systému. Uživatele je možné autentizovat dvěma způsoby. První možnost je přes autorizační hlavičku (*Authorization*) v požadavku. Hlavička bude obsahovat zakódované přihlašovací jméno a heslo uživatele. Druhou možnost představuje autentizační *cookie* nazvaná *WSCOOKIE*.

Autentizační *cookie WSCOOKIE* je možné nabýt dvěma způsoby. První způsob získání této *cookie* je po úspěšném požadavku s autentizační hlavičkou, po kterém vrátí server zmiňovanou *cookie* v odpovědi, tato *cookie*, má expiraci několik hodin. Druhý způsob je přes externí autentizační systém, který mají, až na výjimky, všechny vysoké školy provozující IS/STAG. URL pro externí přihlašovací systém je na adrese */ws/login*, jako parametr se musí zadat *originalURL*, který představuje URL, na kterou je uživatel přesměrován po úspěšném přihlášení a parametr *longTicket* s hodnotou *true*, díky kterému se prodlouží expirace *cookie* na 90 dní. Pokud se uživatel úspěšně přihlásil přes externí systém, je přesměrován na URL adresu, zadanou v pa-

rametru *originalURL*, a zároveň jsou do URL adresy doplněny parametry od externího systému:

- *stagUserTicket* — hodnota pro autorizační *cookie*.
- *stagUserName* — přihlašovací jméno uživatele.
- *stagUserRole* — role uživatele.
- *stagUserInfo* — informace o uživateli zakódovaná v Base64.

3 Mobilní operační systémy

Tato kapitola popisuje dva nejpoužívanější mobilní operační systémy na světě. U každého systému je krátce popsána historie, klíčová charakteristika a aktuální stav. Následuje porovnání globálního rozdělení těchto značek na trhu a zařízení s těmito operačními systémy přistupujícími na webový portál ZČU.

V další části jsou objasněny obecně možné přístupy vývoje mobilních aplikací pro tyto operační systémy. U každého druhu vývoje je popsán jeho princip, jeho klady a zápory. V poslední části této kapitoly jsou tyto přístupy porovnány.

3.1 Mobilní operační systémy

Mobilní operační systém je software umožňující mobilním telefonům, tabletům a ostatním chytrým zařízením, jako například chytré hodinky, běh aplikací a ostatních programů. Většina operačních systémů funguje pouze na specifickém hardware, jako například *iPhone* pracuje na *iOS*, *Google Pixel* funguje na systému *Android*.

3.1.1 Platforma Android

Android je *Unix-like* operační systém pro mobilní zařízení, vyvinutý společnosti *Google*, který je dostupný jako otevřený software. Operační systém byl poprvé vydán v roce 2008[19] a v současné době patří mezi nejrozšířenější operační systém na světě[29]. Android můžeme najít na většině dotykových zařízeních, mobilních telefonech, tabletech, ale nalezneme ho i v televizi nebo v autě.

Aktuálně nejnovější oficiální verzí je Android 12 (API 31)[30] s názvem *Snow Cone*, představený v poslední čtvrtině roku 2021[30]. Aktuálně je nainstalován na zhruba 12 %[28] zařízení podporující operační systém Android.

Výhodou operačního systému Android je jeho volnost. Uživatelé nejsou nuceni instalovat aplikace pouze z obchodu *Google Play Store*, ale mohou si aplikace instalovat přímo z internetu. Tento přístup ale otevírá možnosti pro různé viry nebo *malware*.

3.1.2 Platforma iOS

iOS, původně *iPhone OS¨*, je *Unix-like* operační systém pro telefony *iPhone*, vyvinutý společností *Apple*, který byl představen v roce 2007[4] spolu s první generací iPhone. Jedná se o druhý[29] nejrozšířenější operační systém po systému Android.

iOS je znám díky své podpoře starších zařízení. Aktuálně nejnovější *iOS 15*, který byl vydán v září roku 2021[31] je možné nainstalovat na *iPhone 6s*, který byl vydán v druhé polovině roku 2015[24]. Podpora starších zařízení má za následek, že 63 % ze všech *iPhone* zařízení má nainstalováno nejnovější *iOS 15*. Pokud by se brala v úvahu pouze ta zařízení, která byla představena za poslední čtyři roky, číslo se zvýší na 72 %[10].

Hlavní předností systému *iOS* je jeho jednoduchost uživatelského rozhraní. Ta spočívá v tom, že užívání zařízení s tímto operačním systémem je velmi intuitivní.

Apple je znám tím, že oproti své konkurenci drží uzavřený ekosystém svých zařízení. To znamená, že například umožňuje instalaci aplikací pouze oficiální cestou, přes *App Store*. Je to z toho důvodu, že *Apple* klade velký důraz na uživatelské rozhraní, bezpečnost a ochranu svých uživatelů. Pokud by uživatel chtěl obejít toto zabezpečení, musel by použít takzvaný *Jailbreak*, který umožní větší customizaci zařízení. Oficiální *Jailbreak* aktuálně pro nejnovější verzi *iOS* neexistuje, poslední verze je pro *iOS 14*[3].

3.1.3 Rozdělení trhu

Z celosvětového hlediska *Android* zaujímá okolo 71,7 % trhu, zatímco systém *iOS* pouze 27,5 %[29] . Z tohoto pohledu je patrné, že aplikace pro *Android* mají větší dosah na uživatele, než aplikace pro systém *iOS*. Ze statistických dat, poskytnutých centrem informatizace a výpočetní techniky (CIV) v tabulce 3.1, plyne ovšem úplně jiná skutečnost. Ze všech zařízení za rok 2021, které přistupují přes webový prohlížeč na portál ZČU, zaujímá systém *iOS* 6,0 %, zatímco systém *Android* 4,9 %. Z těchto dat se dá usoudit, že aplikace spojené se studiem na univerzitě jsou stejně důležité pro obě platformy, protože mají podobná zastoupení.

	Rok 2021												
	Leden	Únor	Březen	Duben	Květen	Červen	Červenec	Srpen	Září	Říjen	Listopad	Prosinec	Průměr
iOS	5,1%	3,5%	2,9%	4,0%	5,5%	6,4%	7,3%	6,6%	8,6%	7,5%	7,1%	7,9%	6,0%
Android	4,3%	2,9%	2,2%	3,0%	4,0%	4,9%	5,8%	5,6%	7,9%	6,4%	5,9%	6,1%	4,9%
Ostatní	90,6%	93,6%	94,9%	93,0%	90,5%	88,7%	86,9%	87,8%	83,5%	86,1%	87,0%	86,0%	89,1%

Tabulka 3.1: Přístup na stránky portálu ZČU za rok 2021 v %. *Tabulka zpracována autorem.*

Dle mého názoru převažuje používání platformy *iOS* nad platformou *Android* proto, že uživatelé studentského věku preferují platformu *iOS* nejen pro mobilní telefon, ale i pro jiná zařízení, a to například tablet, sluchátka, chytré hodinky, a jiné. Tento trend, kdy mladí lidé preferují značku *Apple* před značkami s platformou *Android*, mohu pozorovat i ve svém okolí.

3.2 Možnosti vývoje

Existují dva přístupy pro vývoj mobilních aplikací. Prvním přístupem je nativní vývoj a druhým je multiplatformní vývoj.

Výhodou nativního přístupu je, že vývojáři mají přístup ke všem dostupným funkcím mobilní platformy a mohou se zaměřit více na optimalizaci kódu pro daný typ zařízení. Při multiplatformním vývoji stačí pouze jeden kód pro obě platformy. V následujících kapitolách budou detailněji rozepsány oba přístupy, jejich výhody i nevýhody.

3.2.1 Nativní vývoj

Termín *nativní vývoj mobilních aplikací* označuje vývoj aplikace pro jednu specifickou platformu. Aplikace je vytvořena programovacími jazyky a nástroji pro tuto platformu. Pro platformu *Android* to jsou například jazyky *Java* nebo *Kotlin*, pro platformu *iOS* to můžou být programovací jazyky *Objective-C* nebo *Swift*.

Mezi výhody toho přístupu patří přístup ke všem API (*Application Programming Interface*) a nástrojům dané platformy. Dále bychom mohli zmínit možnost škálovatelnosti a výkonu aplikace, a to především pomocí přímého přístupu k funkcím platformy bez další mezivrstvy.

Nevýhodami nativního vývoje jsou hlavně cena a čas. Pokud se aplikace vyvíjí pro obě platformy, je nutné aplikaci naprogramovat dvakrát. To znamená, že je potřeba nasadit dva týmy, které budou na obou platformách pracovat odděleně.

Představiteli nativních aplikací jsou například *Waze*, *Spotify*, *Pokemon Go* a jiné.[23]

3.2.2 Multiplatformní vývoj

Pojem multiplatformní vývoj představuje proces vytváření aplikace, který funguje na několika platformách současně. Pro tento typ vývoje se využívají mimo jiné následující nástroje: *React Native*, *Xamarin* nebo *Flutter*. Při

multiplatformním vývoji se ušetří náklady, avšak za možný úkor kvality procesu. Je obtížné přizpůsobit aplikaci tak, aby optimálně běžela na různých platformách. Aplikace bude potřebovat další abstraktní vrstvu, což může vést ke snížení jejího výkonu.

Mezi nejzásadnější výhody patří nízká cena a rychlost vývoje. Díky tomu, že je potřeba pouze jeden kód pro obě platformy, je nezbytný jenom jeden vývojový tým.

Při multiplatformním vývoji nemusí být poskytnut přístup ke všem nativním knihovnám pro dané platformy, což považuji za velkou nevýhodu. Odepření přístupu může vést k tomu, že ne všechny funkce operačního systému budou dostupné. Přidání abstraktní vrstvy může také zapříčinit zpomalení aplikace. Dochází také ke zvýšení náročnosti údržby aplikace. Většina aplikací musí být aktualizována společně s každou aktualizací operačního systému. S více platformami, o které je nutné se starat, se zvyšuje frekvence aktualizací a údržby kódu.

Představiteli multiplatformních aplikacích jsou například *Facebook* nebo *Skype*.[21]

3.2.3 Zhodnocení

Pokud bychom porovnávali aplikace podle výkonu, je nativní přístup moudřejším rozhodnutím. Při nativním vývoji je možné lépe optimalizovat aplikace, což může vyústit ve zlepšení celkového výkonu aplikace.

Pokud by se jednalo o nejrychlejší uvedení na trh pro obě platformy, vyhrál by multiplatformní vývoj.

Častým omezujícím faktorem při tvorbě nových aplikací je rozpočet. V tomto případě, pokud si firma nemůže dovolit nativní aplikaci, multiplatformní vývoj je ideální volba. Umožňuje psát jediný zdrojový kód pro obě platformy.

Z pohledu bezpečnosti, pokud se jedná například o bankovní aplikaci, nativní aplikace obsahují vestavěné bezpečnostní funkce, jako jsou přístup ke šifrovanému uložišti nebo biometrická autentizace. Tyto principy ulehčí práci vývojářům k implementaci šifrování dat a různým bezpečnostním principům pomocí základních knihoven v operačním systému.

Posledním bodem je *UI/UX*. Nativní aplikace mají lepší možnosti uživatelského rozhraní, než aplikace multiplatformní, a to z toho důvodu, že nativní vývoj má přístup ke všem vestavěným knihovnám a komponentám pro tvorbu uživatelského rozhraní a dovolují více customizace.

Oba, nativní i multiplatformní přístup, mají spoustu různých výhod a nevýhod. Finální volba závisí na požadavcích projektu a schopnostech vývo-

jářů.

3.2.4 Závěr

Tato diplomová práce navazuje na diplomovou práci Marka Zimmermana *Mobilní aplikace IS/STAG na platformě Android*. Cílem jeho práce byla realizace mobilní aplikace komunikující s webovými službami IS/STAG pro platformu *Android*. Protože cílem mé práce je tvorba aplikace pouze pro operační systém *iOS*, není nutné realizovat mobilní aplikaci pro obě platformy současně a bude zvolen nativní vývoj aplikace. Tak bude možné aplikaci lépe optimalizovat pro operační systém, a zároveň bude možné přistupovat přímo na API operačního systému bez další mezivrstvy, která by mohla byla potřeba při multiplatformním vývoji.

4 Vývoj mobilní aplikace pro platformu iOS

V této kapitole jsou popsány možnosti vývoje mobilní aplikace pro platformu *iOS*. Jelikož zadání diplomové práce určuje vývoj mobilní aplikace pro *iOS*, je následující text zaměřen na nativní vývoj pro platformu *iOS*.

4.1 Xcode

Xcode je integrované vývojové prostředí (IDE) od firmy Apple, které slouží pro vytváření aplikací pro všechny *Apple* produkty. Byl vydán v roce 2003. *Xcode* poskytuje kompletní sadu nástrojů pro řízení celého vývojového cyklu od vytvoření, testování, profilování, až po vydání aplikace do *App Store*.

Xcode poskytuje běžné funkce vývojových prostředí: našeptávání, doplňování textu, zvýrazňování chyb, integraci s GIT a spoustu dalších. Příjemným rozšířením je také poskytování automatických oprav, nebo tipů pro opravu upozornění či chyb. *Xcode* obsahuje dále simulátory *Apple* zařízení pro otestování funkčnosti a správnosti zobrazení aplikace. Dále nabízí možnost zobrazení náhledu při návrhu designu, bez nutnosti spuštění aplikace. Na základě těchto funkcí je přirozené, že *Xcode* je vhodnou volbou pro *iOS* vývojáře.

Pro možnost instalace *Xcode* je nutné mít zařízení s operačním systémem *macOS*. Vývojové prostředí je možné získat v *App Store* zdarma. Instalace na *Linux* nebo Windows není možná.

Apple nepatrně vyvíjí nátlak na vývojáře, aby používali *Xcode* pro vývoj aplikací na *Apple* zařízeních, což značí i skutečnost, že pokud vývojář chce vydat svojí aplikaci do *App Store*, musí být aplikace sestavena přes *Xcode*.

4.2 Swift

Swift je *open-source* programovací jazyk pro programování aplikací na zařízení *Apple*, vydaný v roce 2014 společností *Apple*. Je nástupcem *Objective-C*. Cílem tohoto jazyku je vytvořit takový programovací jazyk, který umožní nejlepší přístup od systémového programování, až k vytváření mobilních a desktopových aplikací. Jazyk je navržen tak, aby byl snadno čitelný, bezpečný a rychlý.

Jazyk *Objective-C*, vyvinutý na počátku roku 1980, byl dlouhou dobou hlavním programovacím jazykem všech produktů *Apple*. Jedná se o objektově orientovaný programovací jazyk, který je odvozený z jazyka *C*. Programovací jazyk *Swift* podporuje základní koncepty, které dělaly *Objective-C* flexibilní, zejména dynamické zasílání zpráv, rozšiřitelné třídy, dynamické vazby a další. Tyto funkce mají své kompromisy ve výkonu a bezpečnosti, které *Swift* řeší.

Swift je staticky typovaný jazyk. To znamená, že datový typ každé proměnné musí být znám v čase kompilace, čímž může ověřit, že jsou typy v kódu použity správně. Výhodou staticky typového jazyka (oproti dynamickému) je udržitelnost, protože je vidět, s jakými typy se pracuje v kódu. Další výhodou je bezpečnost, a to díky tomu, že kompilátor může ověřit správné použití typů a může minimalizovat možný pád aplikace při běhu. Staticky typovaný jazyk je značně výkonnější než dynamický, protože zjišťování, které metody je třeba zavolat, není prováděno za běhu. V neposlední řadě můžeme zmínit podporu v IDE, kde moderní vývojové prostředí dokáží lépe napovídat při psaní kódu, popřípadě nabídnout lepší refaktoring.[22]

Od verze 5.3, je *Swift* dostupný také pro operační systém *Windows*. Používání *Swiftu* na *Windows* není pouze o překladači, ale o celém ekosystému. Ekosystém obsahuje překladač, standardní knihovny a základní knihovny, jako například *Foundation*. *Swift* je také dostupný pro *Linux* od verze *Swift* 2.2.[1]

4.3 Možnosti ukládání dat

V této kapitole jsou popsané různé možnosti pro práci s daty na zařízení iOS. Každá možnost je vhodná pro jiný princip použití. Například pro uložení citlivých údajů typu heslo je vhodné použít zašifrované uložiště, naopak pro uložení nastavení použijeme snadno přístupné uložiště.

4.3.1 Property list

Property list zkráceně *plist* je textový soubor, který obsahuje data ve formátu klíč-hodnota. *Plist* umožňuje jako hodnotu ukládat následující typy: *NSData*, *NSString*, *NSNumber*, *NSDate*, *NSArray*, *NSDictionary* nebo třídy implementující rozhraní *Codable*.

Pro čtení dat z *plistu* je nutné načíst celý soubor. Následně se k datům přistupuje jako ve slovníku. Pro zapsání dat je nutné nejdříve načíst celý soubor, aktualizovat potřebné hodnoty a všechna data uložit na disk.

4.3.2 UserDefaults

Jedná se o výchozí uživatelskou databázi ve formátu klíč-hodnota. Databáze dobře pracuje pro ukládání malých dat, protože se načítá při startu aplikace. V databázi je vhodné ukládat data, která nemusejí být zabezpečena, jako například různá nastavení aplikace typu lokalizace, nebo tmavý/světlý režim. Data jsou ukládána na disk jako *property list*. Přístup k databázi je *Thread-safe*.

Přístup k databázi je možný přes třídu jedináčka *UserDefaults*. Následný přístup k objektům je umožněn přes třídní atribut *standard*. Metoda pro čtení hodnot ve *Swiftu* se nazývá *object(forKey:)*. Metoda vrací typ *Any?*. Následně se třída *UserDefault* pokusí najít předaný klíč v databázi, avšak pokud třída zadaný klíč nenajde, vrátí se *nil*. Zápis dat do databáze se provádí metodou *set(_:forKey:* instance *UserDefaults*.

Rozdíl mezi *UserDefaults* a klasickým *plistem* je ten, že *plist*, používaný *UserDefaults*, není viditelný. Druhým rozdílem je, že *Apple* nedoporučuje ukládat velká data do *UserDefaults*, v tom případě je vhodné použít klasický *plist*, nebo úplně jiný přístup.[18]

4.3.3 SQLite

SQLite je knihovna, která implementuje odlehčený databázový engine, který je výkonný a hodící se do embedded zařízení, jako jsou mobilní telefony. *SQLite* patří mezi nejrozšířenější databáze na světě, mezi její uživatelé patří například *Bosh*, *Apple* nebo *Google*.

Na rozdíl od SQL databází, *SQLite* nemá oddělený server, ale čte a zapisuje data přímo z disku zařízení. Kompletní databáze se všemi tabulkami, triggery, indexy a náhledy je obsažena v jednom souboru na disku. Databáze je multiplatformní. Je možné jí překopírovat mezi 32-bitovým systémem a 64-bitovým nebo mezi systémy používající *big-endian* nebo *little-endian*.[27]

Pro integraci *SQLite* ve *Swiftu* existují hotové knihovny. Například *FMDB* nebo *SQLite.swift*.

4.3.4 Keychain

Aplikace může obsahovat citlivá data, která je nutné mít uložena bezpečně. V *iOS* existuje takzvaný *Keychain*, který představuje zašifrovanou databázi. Použití *Keychainu* je nejlepší možností pro uložení malých dat, která jsou kritická pro aplikaci typu uživatelské jméno, heslo nebo různé certifikáty.

Keychain je SQL databáze, která šifruje své položky pomocí algoritmu *AES-256-GCM*. Tabulky a jednotlivé řádky jsou šifrovány zvlášť. Databáze

existuje pouze jedna pro všechny. Přístup řídí démon, který určuje, ke kterým položkám *Keychainu* může proces nebo aplikace přistupovat. Položky lze sdílet mezi aplikacemi, pouze pokud jsou od jednoho vývojáře.[11]

Pro správu dat v *Keychainu* nabízí *Apple* framework *Security*, který poskytuje API (*Keychain Services API*) pro komunikaci. API podporuje všechny CRUD operace. Každá položka v *Keyichainu* má několik označení. Prvním označením je třída. Tento atribut určuje, jak se s položkou bude zacházet. Například *kSecClassGenericPassword* pro označení hesla nebo *kSecClassCertificate* pro označení certifikátu. Druhým označením je list atributů klíč-hodnota, který umožňuje pozdější vyhledání. Zde se například nastavuje účet, skupina, která má přístup k položce, datum vytvoření položky, komentář a spousta dalších.[12]

4.3.5 Core Data

Core Data je framework od společnosti *Apple*, který slouží pro správu datové vrstvy. Framework umožňuje ukládat, sledovat, upravovat a filtrovat data. *Core Data* není databáze, sice používá *SQLite* pro ukládání dat, ale framework není sám o sobě databáze. *Core Data* je *ORM*, ale umožňuje toho mnohem víc, jako například migrace, správu objektového grafu, validaci vstupů, sledování změn v datech a jiné. Jedná se o nadstavbu nad datovým úložištěm.[7]

Definice objektového grafu je umístěna v souboru *xcdatamodeld*. Při vývoji v *Xcode* se po kliknutí na tento soubor *Xcode* zobrazí náhled objektového grafu. Tento náhled může být užitečný při složitějších návrzích pro zobrazení a editaci atributů, vytváření relací, a tak dále.

4.3.6 Soubory

Čtení a zápis souborů a složek je možný přes *FileManager*. Ten umožňuje přistupovat do souborového systému. Je zde ale bezpečnostní omezení, a to, že aplikace může přistupovat pouze do souborového systému vyhrazeného pro aplikaci. Existují základní tři složky v kontejneru aplikace. První složkou jsou *Dokumenty*. Do této složky je vhodné ukládat obsah vytvořený uživatelem. Druhou složkou je *Knihovna*, zde je vhodné uchovávat dočasné soubory, které nevadí pokud se odstraní při nedostatku místa. Poslední složkou je *tmp*. Tato složka by měla sloužit pro ukládání dočasných souborů.[8]

4.3.7 Závěr

Jak z popsaných způsobů uložení dat pro *iOS* plyne, neexistuje žádné obecné řešení, které by vyhovovalo všem případům použití. Každý způsob má své klady a zápory.

4.4 Cocoa Touch

Cocoa je sada knihoven, frameworků a API používaných pro vytváření aplikací pro *Mac OS*. Obsahuje mechanismy pro vykreslování komponent na obrazovku, práci s textem, ukládání a otevírání souborů, komunikaci s operačním systémem, nebo také komunikaci přes síť.

Framework *Cocoa* se zaměřuje na dvě oblasti, třídy reprezentující objekty uživatelského rozhraní a třídy, které zjednodušují problémy, jako jsou například správa paměti, komunikace přes internetovou síť nebo operace se souborovým systémem.

Vývoj aplikací pro *iPhone* a *iPod* se podobá v mnoha ohledech vývoji aplikací pro *Mac OS*, ale vývoj mobilních aplikací vyžaduje doplňkovou sadu knihoven a nástrojů, která se nazývá *iPhone SDK*. *Cocoa Touch* je upravená verze *Cocoa* s knihovnami specifickými pro zařízení *iPhone*, *iPod*, *iWatch* a *Apple TV*.

První oblastí *Cocoa Touch* je *UIKit*. Framework obsahuje kolekci grafických komponent pro tvorbu UI aplikace pro zmíněná zařízení a dokáže zpracovávat dotykové události a vstupy. Zároveň dokáže řídit interakce mezi uživatelem, systémem a aplikací.

Druhým frameworkem, který *Cocoa Touch* obsahuje, je *Foundation*. Framework poskytuje objektově orientovanou abstrakci pro klíčové prvky operačního systému, kam patří například: nástroje pro správu dat, objektové obaly primitivních datových typů, správu paměti, odesílání událostí, a další. Všechny aplikace *Cocoa Touch* musí využívat tento framework, protože obsahuje třídy, které zajišťují fungování celé aplikace.[2][6]

4.5 Možnosti tvorby UI

Vývoj uživatelského rozhraní *iOS* aplikací je možné vytvářet několika způsoby. Níže jsou popsány možná řešení s jejich výhodami a nevýhodami.

4.5.1 Programový přístup

Programový přístup využívá framework *UIKit*. Pro popis co je na obrazovce se používá imperativní přístup. To znamená, že vývojář musí definovat, kdy se obrazovka aktualizuje, a jak se bude přecházet mezi stavy *UI*. Pokud by se například načetl nový zdroj dat do tabulky, ale nepřekreslila se obrazovka, uživateli by se nezobrazila nová data.

Ačkoli čistě programový návrh uživatelského rozhraní umožňuje velkou škálovatelnost, je poměrně složitý, protože je nutné, aby si vývojář dal pozor při vývoji na všechny stavy aplikace a jejich přechody. Návrh *UI* musí vypadat dobře na všech podporovaných zařízeních. Proto mnoho začátečníků volí zpočátku návrh uživatelského rozhraní pomocí *Storyboard* nebo *SwiftUI*. Při složitějších grafických náhledech je ale téměř vždy nutné použít minimálně kombinaci jiného přístupu s programovým přístupem.

4.5.2 Storyboard

Jedná se o interaktivní nástroj, využívající *UIKit* pro tvorbu uživatelského rozhraní, který byl vydán společně s *iOS 5*, v roce 2011[26]. Pomocí nástroje je možné navrhnout *UI*, bez nutnosti napsání jediného řádku kódu. Nástroj funguje na principu *drag and drop*, kde z je palety komponent možné vybrat jednu a umístit ji kamkoliv na plátno (*Canvas*), které představuje obrazovku zařízení. Je možné vybírat z různých vstupních prvků, tlačítek, obrázků, a dalších.

Výhodou tohoto nástroje je jednoduchost, dokonce i začátečník je schopný pomocí tohoto nástroje vytvořit komplexní *UI*. Dále je možné s nástrojem vytvořit vcelku rychle prototyp aplikace, bez nutnosti psaní několika set řádků kódu.

Nevýhodou tohoto nástroje je řešení konfliktů u správy verzí (*Source Control*), protože *Storyboard* není nic jiného, než XML dokument. Je tedy značně obtížné při sjednocování dvou verzí řešit konflikty, protože je komplikované rozumět XML obsahu. Jako další nevýhodu bychom mohli zmínit, že spolu se zvyšujícím se počtem obrazovek se nástroj může stát velice chaotickým. V poslední řadě je také důležité zdůraznit problémy s automatickým rozvržením obrazovky, který tento nástroj nabízí. Výsledek by se měl kontrolovat na podporovaných zařízeních, protože není vždy spolehlivý.

4.5.3 SwiftUI

SwiftUI je *framework* od *Apple*, který byl představen s *iOS 13*. Dle slov *Apple*: „SwiftUI pomáhá vytvářet dobře vypadající aplikace napříč všemi

platformami *Apple* pomocí jazyka Swift za použití co nejmenšího množství kódu...“ Jedná se o nástupce frameworku *UIKit*. *SwiftUI* přináší úplně nový způsob vytváření *UI*, a to pomocí deklarativního programování. Vývojář tedy pouze řekne, jak chce aby *UI* vypadalo, a jak má fungovat. Oproti imperativnímu programování, které je například u *UIKit*, kde bylo nutné spravovat všechny komponenty, které se mají zobrazit či skrýt ručně, ve *SwiftUI* se pouze určí pravidla, která určí, kdy se má jaká komponenta zobrazit a *framework* obstará všechno ostatní. *SwiftUI* je možné kombinovat s *frameworkem UIKit*.[16]

SwiftUI dokáže vytvářet náhled napříč všemi *Apple* produkty, což znamená, že pomocí jednoho jazyka je možné vytvářet vzhled na *iOS*, *macOS*, *tvOS*, nebo také na *watchOS*. Vývoj zjednodušuje automatický živý náhled, který ukazuje aktuální náhled, bez nutnosti spouštění aplikace v simulátoru. Kód je sám o sobě jednoduchý a čitelný.

Nevýhodou frameworku je podpora. *SwiftUI* je možné používat v projektech s minimální verzí podpory *iOS 13* a od verze *Xcode* 11. *SwiftUI* zatím nepodporuje všechny komponenty, které jsou dostupné v *UIKit*, takže v některých případech je nutné použít kombinaci *SwiftUI* s programovým přístupem. Neexistuje žádná přímá migrace z jiných nástrojů pro tvorbu náhledů na *iOS*.

Podpora reaktivního programování

Reaktivní programování popisuje návrhové paradigma, které spoléhá na asynchronní programovací logiku pro aktualizaci jinak statického obsahu v reálném čase. Události, které vyvolávají aktualizaci, mohou být cokoliv, například stisknutí tlačítka, uživatelský vstup, asynchronní načtení dat, a tak dále. Záleží, na jakém objektu bude nastaven posluchač událostí, a jak bude nastaven.

SwiftUI poskytuje jednoduchý způsob pro aktualizaci dat jak z externích událostí, tak pro vstupy od uživatele. *SwiftUI* automaticky aktualizuje změněná data v *UI*. *Framework* poskytuje nástroje, jako jsou například stavové proměnné, vazby pro propojení dat v aplikaci a *UI*. Tyto nástroje obstarávají správu *single source of truth* pro všechny data v aplikaci. Nástroje se používají přes obaly proměnných. Mezi obaly proměnných pro tok dat ve *SwiftUI* patří například[5]:

- *State* — Umožňuje pohledu ukládat lokální proměnlivý stav. Například *Bool* proměnná, která slouží jako přepínač.
- *Binding* — Umožňuje pohledu měnit data, která vlastní jiný pohled.

Typicky se jedná o vnořený pohled, který obsahuje referenci na data rodičovského pohledu.

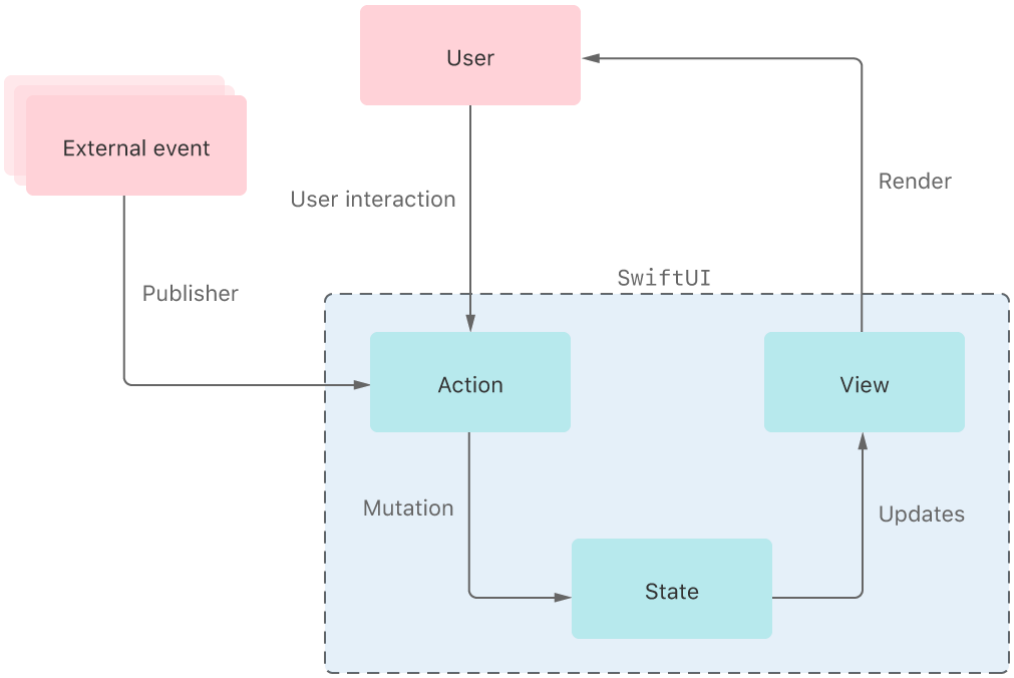
- *Environment* — Umožňuje číst data ze systému, jako například barevné schéma, možnosti přístupu, ale umožňuje si i definovat vlastní klíče s daty.
- *Published* — Tento obal je spojený s proměnnými uvnitř *ObservableObject*, a říká *SwiftUI*, že pokud se tato proměnná změní, pohled by se měl aktualizovat s novými daty.
- *ObservedObject* — Obal pro sledování externího objektu, který upozorní *SwiftUI*, pokud dojde ke změně. Podobný jako *StateObject*, s tím, že *StateObject* je pouze pro referenční typy, zatímco *ObservedObject* je pro externí objekty. Pokud by došlo k záměně, mohlo by dojít ke ztrátě dat.
- *StateObject* — Používaný pro uložení referenčního typu uvnitř pohledu, který zajistí, že hodnota nebude ztracena při přechodu na jiný pohled.
- *AppStorage* — Poskytuje přístup pro čtení a zápis pro hodnoty v *UserDefaults* úložišti. *SwiftUI* dokáže tedy zareagovat i na změny v tomto úložišti, například pokud se změní v jiném pohledu.

4.5.4 Závěr

Jelikož nově vyvíjená mobilní aplikace bude podporovat minimální verzi *iOS 15*, bude použit framework *SwiftUI*, který *Apple* prezentuje[15] jako moderní způsob pro vyvíjení uživatelského rozhraní. Pokud se při implementaci objeví nějaká překážka, kterou nebude možné vyřešit se *SwiftUI*, bude implementace doplněna pomocí *UIKit* s programovým přístupem.

4.6 Architektura mobilních aplikací

Architektura hraje klíčovou roli pro vývoj aplikace. Výběr architektury má vliv na velikost, strukturu, rozšiřitelnost a údržbu projektu. Neexistuje žádné obecné pravidlo pro výběr architektury, vždy záleží na konkrétním projektu. Při volbě architektury je brán v úvahu například typ projektu, technologie, které se budou používat, plány na rozšíření, služby třetích stran, schopnosti týmu, a tak dále.



Obrázek 4.1: Řízení stavů a toku dat ve *SwiftUI* [15]

4.6.1 MVC

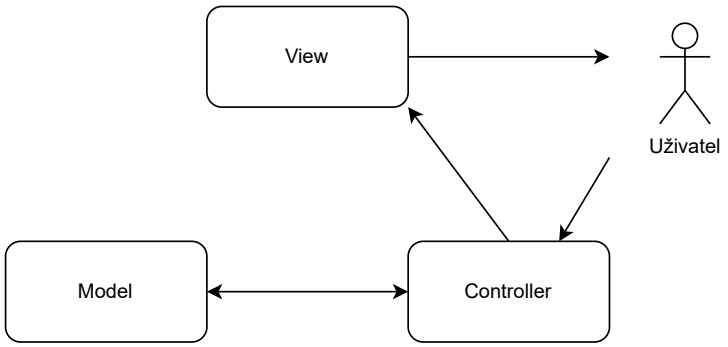
MVC patří mezi nejznámější multiplatformní architektury pro vývoj aplikací. Dělí aplikaci na tří nezávislé části: *Model*, *View* a *Controller*. Schéma architektury je znázorněna na obrázku 4.2. Architekturu je vhodné použít při vývoji *iOS* aplikace, pokud je zvolen programový přístup pro *UI*, kvůli použití *UIKit*.[17]

Model se stará o práci s daty. Nachází se v něm třídy, které komunikují s databází, parsery, výpočty, různé manažery nebo síťová komunikace. *View* představuje prezentační vrstvu aplikace. Její třídy jsou často opakovaně použitelné, protože neobsahují žádnou doménově specifickou logiku. *View* komunikuje s *Controllerem*, aby získal potřebná data, která zobrazí uživateli. *Controller* je část aplikace, která obsluhuje interakci s uživatelem. Odchytává vstup od uživatele a informuje *Model* a *View* aby se aktualizovaly dle požadavků.

Mezi hlavní znaky architektury patří *Controller* jako vstupní bod aplikace, mezi *Controllerem* a *View* je vazba 1:N, *View* nemá referenci na *Controller*, komponenty mohou být testovány samostatně.

Výhodou této architektury je, že jednotlivé části mohou být vyvíjeny paralelně, protože jsou rozdělené do nezávislých vrstev. Toto rozdělení po-

skytuje oddělení odpovědnosti (*Separation of contents*). Architektura nabízí vývoj řízený testy. Nevýhodou je kombinace *businnes* logiky s *UI*, jelikož *Controller* může obsahovat spoustu řádků kódu a stát se tak nepřehledným.



Obrázek 4.2: Diagram architektury MVC

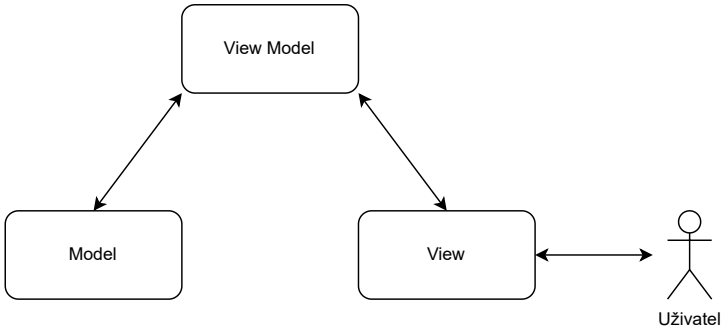
4.6.2 MVVM

MVVM se skládá z *Model*, *View* a *View Model*. Architektura nabízí obousměrnou vazbu dat mezi *View* a *View Model*, a díky tomu umožňuje automatizovat šíření změn. *View Model* využívá vzor pozorovatele pro provádění změn. Při vývoji *iOS* aplikace je vhodné použít tuto architekturu, pokud je zvoleno *SwiftUI* pro *UI* a to především kvůli tomu, jak *SwiftUI* usnadňuje implementaci propagaci změn mezi *View* a *View Modelem* pomocí vazeb.

Jako u *MVC*, *Model* se stará o manipulaci a ukládání dat. *View* je zodpovědné za zobrazování dat, která jsou získána z *View Modelu* a umožňuje interakci s uživatelem. *View Model* propojuje *View* a *Model*. Zároveň drží aktuální stav *View*.

Hlavními vlastnostmi je, že *View* je vstupním bodem aplikace, mezi *View* a *View Modelem* je vazba 1:N a *View* obsahuje referenci na *View Model*, viz obrázek 4.3.

Mezi klady *MVVM* patří oddělení *business* logiky z *UI*. Také zlepšení testovatelnosti díky tomu, že *View Model* nemá referenci na objekt, který ho vlastní, čímž dojde k ulehčení unit testování. Implementace vazeb při imperativním programování může někdy dělat problémy. *MVVM* se nehodí pro jednoduché *UI*.



Obrázek 4.3: Diagram architektury MVVM

4.6.3 Závěr

Při implementaci aplikace bude použita architektura *MVVM* z toho důvodu, že pro vývoj *UI* bude použito *SwiftUI*, které společně s touto architekturou velmi dobře funguje, a to především kvůli vazbám ve *SwiftUI*, které poskytuje a kvůli automatickým publikováním změn.

4.7 Publikace aplikace

V této kapitole jsou popsány požadavky a možnosti pro vydání aplikace na zařízení *iPhone*. Pro vydání aplikace je nutné mít zařízení *Apple* s nainstalovaným vývojovým prostředím *Xcode*, protože vydání aplikace je s ním silně spjato a neexistuje jiná možnost, jak distribuovat aplikaci bez něj. Určité distribuce navíc vyžadují, aby vývojář měl zaplacené předplatné, které mu umožní vydávat aplikace.

4.7.1 Apple Developer Program

Prvním krokem pro publikování aplikace do *App Store* je zřízení členství *Apple Developer Programu*. Tento program přináší vývojáři přístup k beta operačním systémům, k rozšíření vývojových nástrojů, možnosti vydání aplikace do beta testování (*TestFlight*) a ke spoustě dalších užitečných funkcí pro vývojáře. Především ale tento program umožňuje vydání aplikací do *App Store* za splnění určitých kritérií, viz dále. Cena tohoto programu je 99 USD ročně bez DPH[13]. v České republice je nutné k této částce připočíst ještě 21 % DPH, konečná částka za tento program je asi 2 800 Kč ročně.

4.7.2 Možnosti vydání aplikace

Existuje několik možností pro vydání aplikace, každá možnost má svoje klady, zápory. Níže budou popsány možné přístupy distribuce aplikace na zařízení *iOS*.

Personal Team Distribution

Pokud se zaregistruje vývojář jako *Apple developer* (bez zaplacení programu), automaticky dostane možnost publikovat aplikace na několika zařízeních přes vývojové prostředí *Xcode*. Toto sestavení vyprší do několika dnů. Tento způsob slouží především pro začátečníky a pro otestování základní funkčnosti aplikace na mobilním telefonu, před zaplacením *Apple Developer Programu*.

Zařízení, na kterém se má provést sestavení aplikace, musí být připojeno k počítači přes kabel. V *Xcode* se následně mezi možnostmi cíle sestavení aplikace (*target*) zobrazí připojené zařízení. Tuto volbu je nutné zvolit a následně provést spuštění aplikace v *Xcode*.

Ad hoc

Při zaplacení *Apple Developer Programu* získáte možnost *Ad hoc*, která umožňuje publikovat aplikace malé skupině lidí. Je zde omezení pro vydání aplikace na 100 zařízení ročně a sama aplikace má expiraci 1 rok. Vývojáři často tuto možnost využívají pro interní *alpha* testování v rámci svého *CI/CD*. Pro instalaci aplikace je nutné, aby aplikace měla zaregistrované *UUID* zařízení, na kterém se má provést instalace.[25]

Pro tento způsob distribuce je nutné sestavit aplikaci jako archív ve vývojovém prostředí *Xcode*, následně vybrat formu distribuce *Ad hoc*, která vytvoří archív, který je možné distribuovat dvěma způsoby. Prvním způsobem je připojení zařízení k počítači a manuální instalaci, druhým způsobem je OTA, přes webový prohlížeč.

Enterprise

Tato možnost slouží pro firemní účely. Umožní aplikaci distribuovat interně ve firmě mimo *App Store*. Jedná se *Ad hoc* distribuci bez jakéhokoliv omezení. Aby firma mohla zažádat o *Enterprise* distribuci, je potřeba aby měla přes 100 zaměstnanců, byla právní osobou, vytvářela aplikaci pouze pro interní účely a měla zabezpečený server, na který se dostanou pouze zaměstnanci firmy. V neposlední řadě musí absolvovat pohovor s *Apple*. Nad rámec těchto kritérií si ještě firma musí pořídit *Apple Developer Enterprise Program*, který stojí 299 USD ročně bez DPH.[14]

Pro distribuci je nutné sestavit aplikaci jako archív v *Xcode*, následně vybrat jako distribuci *Enterprise*, který vytvoří archív. Poté je nutné tento archív nahrát na zabezpečený firemní server, ze kterého ho půjde následně stáhnout.

Apple Store Connect

Jako člen *Apple Developer Programu*, získáte přístup k *Apple Store Connect*. Jedná se o webovou aplikaci sloužící pro správu vydaných aplikací. Web umožňuje spravovat produktové stránky jednotlivých aplikací, sledovat statistiky, spravovat jednotlivé verze aplikace, vydávat aplikace do testovacího prostředí *TestFlight*, vydávat aplikace do schvalovacího procesu pro vydání do *App Store*, a další. *Apple Store Connect* představuje mezivrstvu mezi sestavením aplikace a jejím vydáním.

Pro distribuci aplikace je nutné sestavit archiv v *Xcode* a následně vybrat druh distribuce *Apple Store Connect*, který automaticky nahraje vytvořený archív do *Apple Store Connect*.

TestFlight

TestFlight je oficiální metoda pro distribuci *beta* verze aplikace, která slouží pro získání zpětné vazby od externích uživatelů. O *TestFlight* by se dalo říci, že je takový *App Store* pro *beta* aplikace. Pomocí *TestFlight* je možné aplikace distribuovat až pro 100 interních testerů a až pro 10 000 externích testerů. Tato možnost distribuce je tedy vhodná jako nahrazení distribuce *Ad hoc*, pokud distribuce narazí na limit 100 zařízení, jen je nutné mít na paměti, že sestavení aplikace pro *TestFlight* má expiraci 90 dní.

Pro vydání aplikace do *TestFlight* je nutné nejdříve aplikaci nahrát do *Apple Store Connect*. Následně je možné k nově nahranému archívu aplikace přiřadit testery. Pokud jsou k aplikaci přiřazeni interní testeři, aplikace je ihned dostupná. Pokud se k aplikaci přiřadí externí testeři, je nejdříve nutné, aby aplikace prošla schvalovacím procesem. Po jeho schválení je aplikace připravená pro *beta* testování externími uživateli.

App Store

Je to aplikace pro distribuci oficiálních aplikací na zařízení s operačním systémem *iOS*, *iPadOS*, *macOS*, *tvOS* a *watchOS*. Pro vydání aplikace do *App Store* je nutné nejdříve aplikaci nahrát do *Apple Store Connect*. Následně je možné dát aplikaci ke schválení. Pokud *Apple* usoudí, že aplikace splňuje všechny podmínky pro nahrání do *App Store*, je aplikace vydána. Detailnější

popis schvalovacího popisu a dalších podmínek pro vydání aplikace je popsán v následující podkapitole.

4.7.3 Proces schvalování aplikace

Než je možné aplikaci vydat do *TestFlightu* nebo *App Store*, je nejprve nutné, aby aplikace prošla schvalovacím procesem. Úkolem schvalovacího procesu je odhalit aplikace, které nesplňují podmínky pro uvedení aplikací na zmíněných platformách. Hlavním smyslem schvalovacího procesu je otestovat aplikaci (její funkčnost), ochránit uživatele před krádeží dat, zabránit nevhodnému obsahu a především zajistit, aby aplikace dodržovala *UI/UX* standardy, předepsané v *Human Interface Guidelines*. Pokud schvalovací proces vyhodnotí neshodu s předpisy, aplikaci zamítne.

Před odesláním aplikace ke schválení je nutné vyplnit metadata o aplikaci. Ty v případě publikování do *Testflightu* zahrnuji poskytnutí základního popisu aplikace, testovacího účtu (pokud je potřeba), kontaktní informace a poznámky pro otestování aplikace. Pro publikování do *App Store* je kromě zmíněných metadat potřeba vyplnit kartu produktu, která obsahuje náhledy aplikace na různých typech zařízení, propagační text, popis, klíčová slova a datum publikování do *App Store*. To že aplikace projde schvalovacím procesem pro *TestFlight* neznamená, že automaticky projde schvalovacím procesem do *App Store*. V případě *TestFlightu* není schvalovací proces tak důsledný, protože se jedná o beta testování.

4.7.4 Human Interface Guidelines

Human Interface Guidelines (*HIG*) je příručka o tom, jak by mělo vypadat uživatelské rozhraní pro všechny platformy *Apple*. Příručka obsahuje informace pro designéry, které jim pomohou vytvářet přesvědčivější, intuitivnější a lepší návrhy aplikací. *HIG* nabízí ucelený pohled na klíčové prvky uživatelského rozhraní a osvědčené postupy, které pomohou implementovat funkce do aplikace. V příručce najdeme například to, jak se má zarovnávat obsah, jak zobrazovat vstupy pro uživatele, text, kontrast, a mnohem více.[9]

5 Analýza potřeb uživatelů

V této kapitole je popsána analýza, která má za úkol prozkoumat obecné potřeby uživatelů, spojených s používáním služeb IS/STAG na mobilních zařízeních, bez ohledu na druh operačního systému zařízení.

Analýza vznikla proto, že v současné době existuje mnoho aplikací obsahujících různé funkce, které mohou být užitečné, ale velmi často jsou spíše zbytečné a danou aplikaci komplikují. Ta se poté může zdát příliš složitá, a to může odradit uživatele od používání aplikace.

Spousta současných mobilních aplikací pro komunikaci s WS IS/STAG je zaměřena pouze na potřeby studentů, protože studenti tvoří převážnou většinu všech uživatelů. Je zde ale řada dalších potencionálních uživatelů, například učitelů, kteří by aplikaci pro komunikaci s webovými službami IS/STAG uvítali.

Cílem této analýzy je tedy prozkoumat různé role uživatelů a determinovat jejich potřeby tak, aby bylo možné navrhnout mobilní aplikaci, která by byla přínosná pro širší škálu uživatelských rolí.

5.1 Uživatelské role

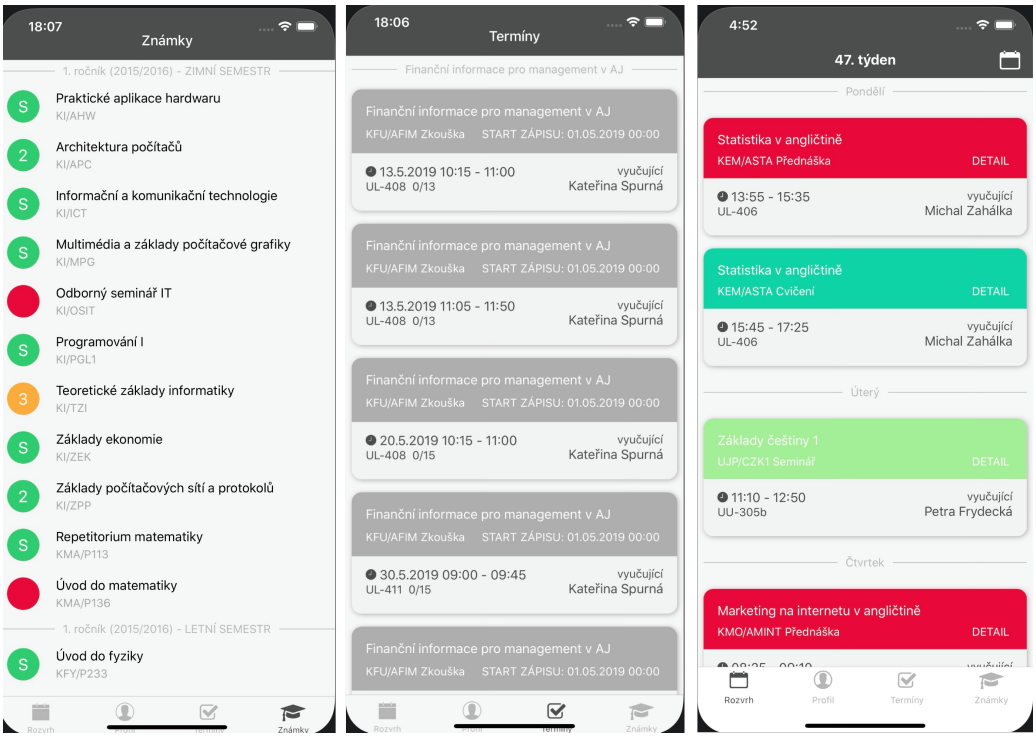
IS/STAG je rozsáhlý systém, který obsahuje spoustu uživatelských rolí. Jak z povahy systému plyne, mezi ty nejvíce zastoupené patří role student a vyučující. Nesmíme ovšem zapomínat i na další potencionální uživatele, jako jsou: studijní referent/referentka, prorektor, tajemník fakulty, a jiné.

5.2 Existující aplikace

Níže jsou uvedeny nejznámější aplikace, které komunikují s webovými službami IS/STAG. U každé je krátké seznámení s aplikací. Všechny obsahují běžné funkce, kterými jsou: zobrazení rozvrhu, zapsání a odepsání zkoušek, přehled zapsaných předmětů nebo souhrn studia. Kompletní porovnání funkcí je znázorněno v tabulce 5.1.

5.2.1 IS Stag

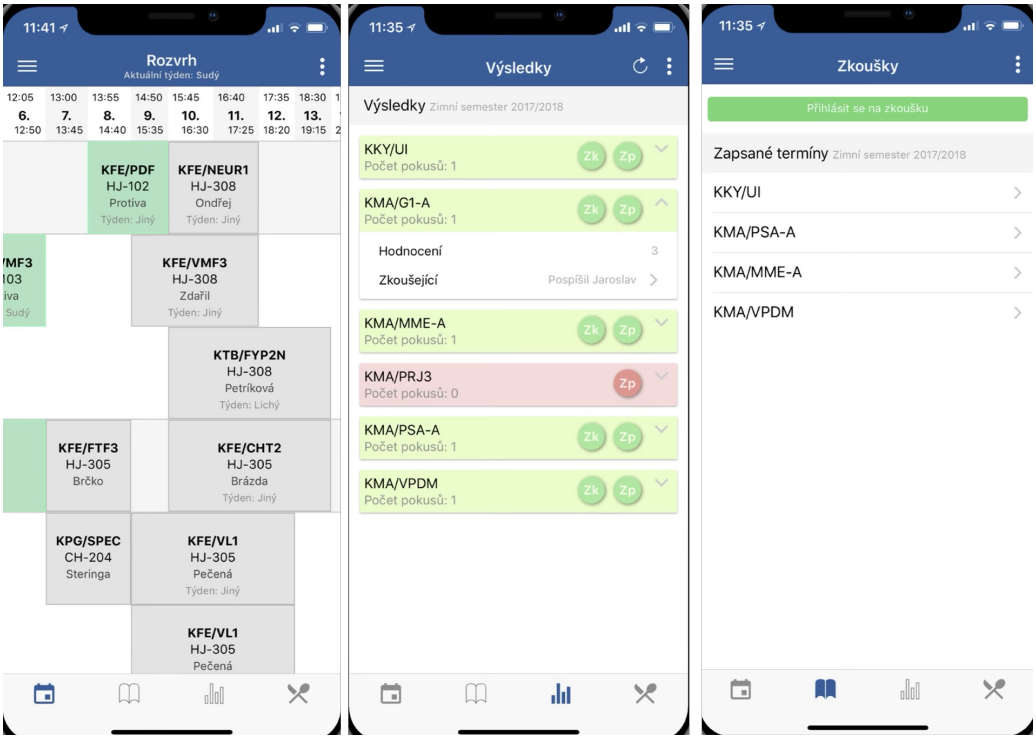
Autorem je Viktor Sinelnikov. Aplikace podporuje 13 vysokých škol a platformy *Android* a *iOS*. Mezi poskytovanými funkcemi je rozvrh, přihlášení a odhlášení ze zkoušek, informace o předmětech a průběh studia. Aplikace nebyla již nějaký čas aktualizovaná. Poslední aktualizace pro *iOS* proběhla 14.11.2020 (verze 1.1.6). Náhled aplikace je zobrazen na obrázku 5.1.



Obrázek 5.1: Ukázka aplikace IS Stag

5.2.2 Student ZČU

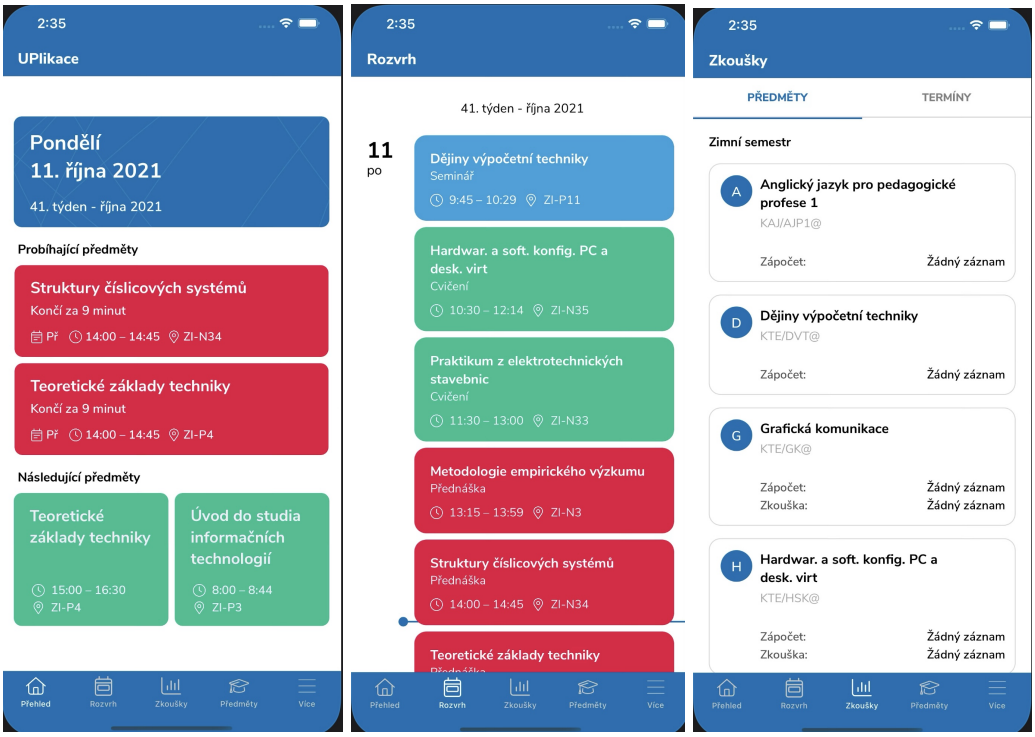
Autorem je Tomáš Balíček. Aplikace podporuje pouze Západočeskou univerzitu v Plzni, funguje pro platformy *Android* a *iOS* a nabízí běžné funkcionality. Student ZČU je navržená specificky pro ZČU, z čehož plyne výhoda obsažení většího množství informací, jako například jídelníčku v menze, webmailu, akcí na vysoké škole, checklistu s úkoly, kterým je dobré věnovat pozornost (jedná se například o žádosti o stipendium), rektorského volna, exkurzí a mnoho dalších. Aplikace v současné době nemá funkční rozvrh pro *iOS* zařízení. V minulosti měla aplikace problém i s přihlášením do aplikace, a proto byla pár měsíců nepoužitelná. Poslední aktualizace pro *iOS* je z 7.9.2020 (verze 2.0.5). Ukázku aplikace je možné vidět na obrázku 5.2.



Obrázek 5.2: Ukázka aplikace Student ZČU

5.2.3 UPlikace

Tato aplikace je určena pro Univerzitu Palackého v Olomouci, kde také vznikla. Aplikace podporuje platformu *iOS* i *Android*. Nad rámec běžné funkčnosti obsahuje interaktivní mapu Olomouce s vyznačením vysokoškolských budov, dále zobrazuje upozornění spojená se zapsáním známky, zkouškového termínu a blížícího se konce možnosti přihlášení/odhlášení ze zkouškového termínu. Aplikace je udržována, poslední aktualizace pro *iOS* proběhla 1.4.2022 (verze 3.5.0). Aplikace je zobrazena na obrázku 5.3.

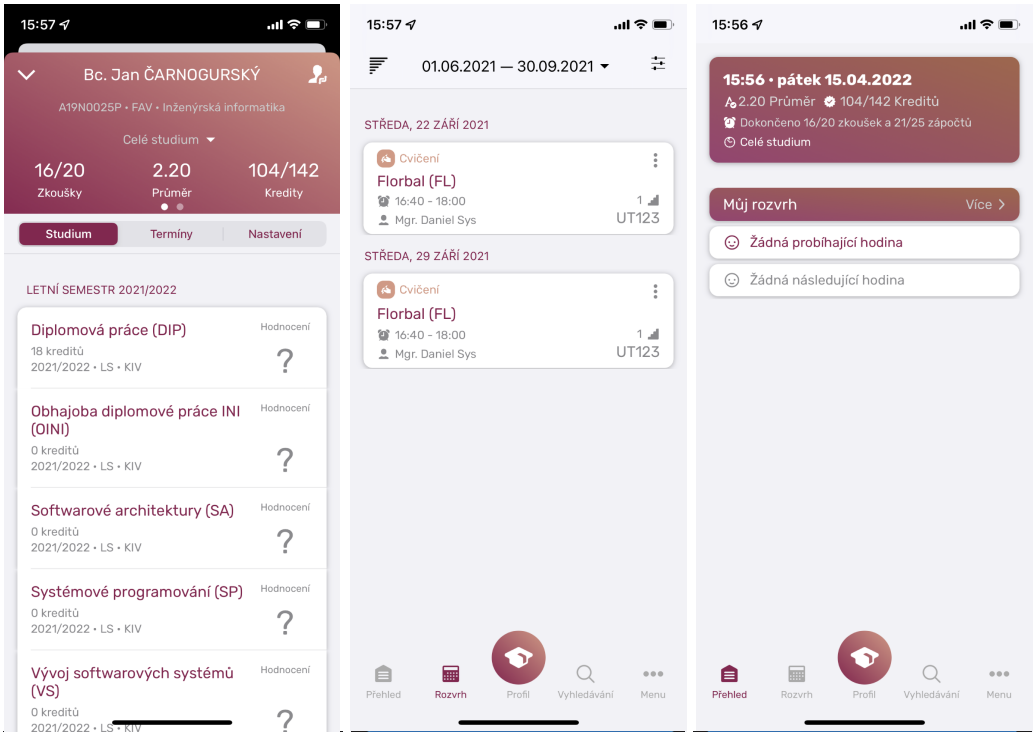


Obrázek 5.3: Ukázka aplikace UPlikace

5.2.4 Univerzita - IS STAG

Aplikace byla vytvořena Michalem Černým, podporuje všech 13 vysokých škol a funguje pro platformy *iOS* a *Android*. Kromě běžných funkcí nabízí aplikace vyhledávání zaměstnanců, studentů nebo místností. Další zajímavostí je zobrazování pracovních nabídek přímo v aplikaci. Ačkoli se nejedná o aplikaci přímo inicializovanou ZČU, nabízí aplikace jídelníček. Poslední aktualizace pro *iOS* je 19.3.2022 (verze 1.4.1). Náhled aplikace je zobrazen na obrázku 5.4.

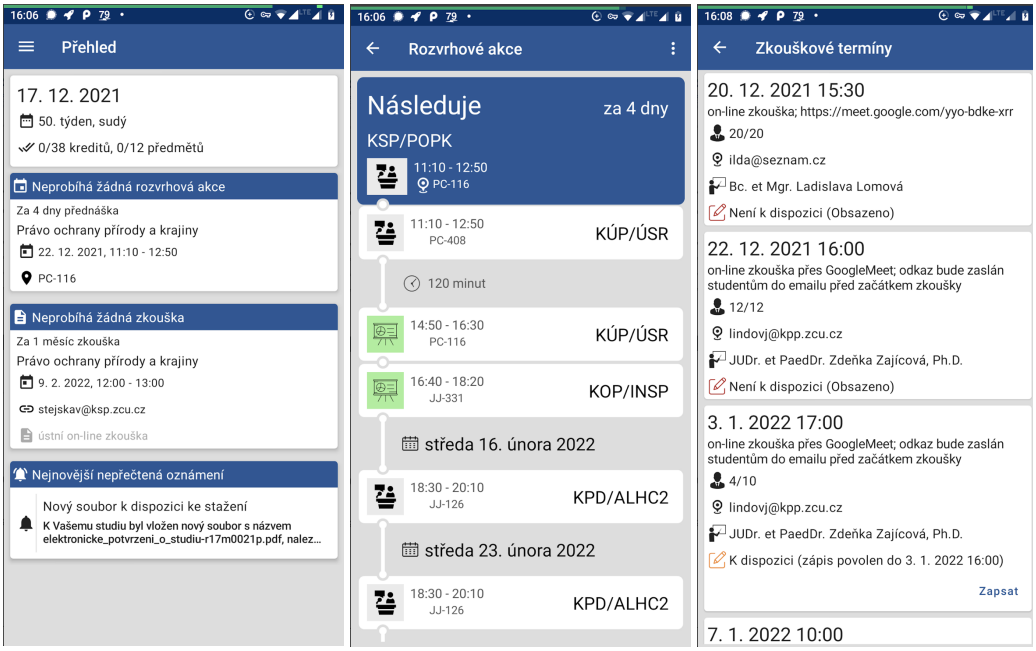
Aplikace, mimo to, že komunikuje s webovými službami IS/STAG, komunikuje také s vlastním serverem sloužícím právě pro tuto aplikaci. V podmínkách použití se uvádí, že přihlašovací údaje jsou poslány na tento server, ale slouží pouze pro ověření uživatele a v žádném případě se na serveru neukládají.



Obrázek 5.4: Ukázka aplikace Univerzita - IS STAG

5.2.5 IS/STAG Mobile

Aplikace, vytvořená Markem Zimmermannem, vznikla v rámci diplomové práce na ZČU a podporuje 13 vysokých škol. Jako jediná bere v úvahu i další uživatelskou roli, a to roli učitele. Nevýhodou aplikace je funkčnost omezená pouze pro platformu *Android*. Nabízí studentovi běžné funkcionality, stejně jako výše popsané aplikace. Dále v IS/STAG Mobile najdeme notifikace se zapsanými známkami a novými zkouškovými termíny. Pro roli učitele je možné zobrazit rozvrh, zápis známek nebo lze vytvořit hromadný email ve výchozí aplikaci mobilního telefonu. Aplikace je udržována a stále vyvíjena. Poslední aktualizace pro *Android* je ze dne 25.4.2022 (verze 1.9.0). Ukázku aplikace je možné vidět na obrázku 5.5.



Obrázek 5.5: Ukázka aplikace IS/STAG Mobile

5.2.6 Bezpečnostní problém aplikací

Ačkoli by se na první pohled mohlo zdát, že všechny výše zmíněné aplikace jsou funkční, ve všech uvedených aplikacích, kromě IS/STAG Mobile, je drobný bezpečnostní problém. Aplikace s každým požadavkem na server odesílají uživatelské jméno a heslo pro ověření uživatele. To znamená, že tyto údaje jsou uloženy v lokální databázi na zařízení, ale není jasné v jaké podobě. Jediná aplikace IS/STAG Mobile používá autorizační *cookie*, popsanou v sekci 2.1.

5.2.7 Celkové srovnání aplikací

V následující tabulce 5.1 je zobrazen výčet funkcí aplikací. Znakem „x“ jsou označeny funkce, které daná mobilní aplikace podporuje. Z tabulky je patrné, že neexistuje žádná univerzální mobilní aplikace, která by obsahovala úplně všechny funkce.

	IS Stag	Student ZČU	Uplikace	Univerzita	IS/STAG Mobile
Podpora iOS	x	x	x	x	
Podpora Android	x	x	x	x	x
Podpora více vysokých škol používající IS/STAG	x			x	x
Zobrazení rozvrhu	x	x	x	x	x
Přehled/Zápis zkoušek	x	x	x	x	x
Přehled zapsaných předmětů	x	x	x	x	x
Souhrn studia (absolvované předměty s výsledky)	x	x	x	x	x
Zobrazení jídelníčku menzy		x		x	x
Vyznačení budov na mapě		x	x		
Zobrazení lichého/sudého týdne		x	x	x	x
Zobrazení emailů		x			
Zobrazení studentských akcí		x			
Checklist (akce, kterým věnovat pozornost)		x			
Celkový stav kreditů				x	x
Notifikace			x		x
Průběh aktuálního dne			x	x	x
Možnost zvolit rozsah pro zobrazení rozvrhu				x	
Vyhledávání				x	
Pracovní nabídky				x	
Podpora pro učitele (víc než jen rozvrh)					x

Tabulka 5.1: Celkové srovnání funkcí aplikací

5.3 Průzkum potřeb uživatelů

V této kapitole jsou popsány dotazníky, které byly použity pro získání informací o potřebách uživatelů. Veškerá data byla následně vyhodnocena.

5.3.1 Použité dotazníky

Pro rešerši potřeb uživatelů byly použity dva typy dotazníků. Obecný dotazník, který byl respondentům poskytnut přes sociální sítě, neposkytl dostatečné množství odpovědí pro uživatelskou roli učitele, a proto byl vytvořen druhý dotazník — zaměřený dotazník, který byl zaslán e-mailem náhodně vybraným vyučujícím a dalším zaměstnancům fakult.

Obecný dotazník Dotazník byl vytvořen přes portál *Survio*. Pro jeho sdílení byly využity sociální sítě. Dotazník byl zaměřen na všechny typy uživatelských rolí.

Cílem tohoto dotazníku bylo získat povědomí o tom, jak lidé vnímají spojení studia/práce s mobilní aplikací. Dále bylo zjišťováno, jaké funkce jsou v mobilní aplikaci klíčové, a které jsou naopak postradatelné. Pokud uživatel nějakou mobilní aplikaci pro své studium již používá, byl v dotazníku vytvořen prostor pro zpětnou vazbu k aplikaci. Pokud uživatel žádnou mobilní aplikaci pro své studium nepoužívá, má v dotazníku možnost sdělit, proč tomu tak je. Poslední otázka byla zaměřena na případné nápady a návrhy k aplikacím pro komunikaci s WS IS/STAG.

Dotazník byl složen z následujících otázek:

1. Pohlaví (Muž, Žena).
2. Fakulta (Seznam fakult).
3. Pozice (Student - navazující, Student - bakalář, Učitel, Doktorand, Jiná).
4. Jakou aplikaci používáte při svém studiu (IS Stag, Student ZČU, Univerzita, Žádná, Jiná).
5. (Nepovinná) Pokud nějakou aplikaci používáte, chybí vám v ní nějaká funkce, kterou používáte na portále? (Otevřená otázka)
6. (Nepovinná) Pokud žádnou aplikaci nepoužíváte, vysvětlete proč? (Otevřená otázka)
7. Seřaďte funkce od nejdůležitějších:
    - Rozvrh
    - Zápis/přehled zkoušek
    - Vyhledání předmětů s jejich sylabem
    - Informace o zapsaném předmětu
    - Přehled zapsaných předmětů
    - Aktuální stav kreditů
    - Widget s rozvrhem
    - Příjem a odesílání školních emailů
    - Vyhledání učitelů

- Vyhledání učeben s rozvrhem
- Možnost nastavit si deadline

8. (Nepovinná) Nějaká další myšlenka, o kterou by ses chtěl podělit? Nějaká další funkce, kterou portál obsahuje, ale není v možnostech?

Zaměřený dotazník Tento dotazník byl směřován na konkrétní uživatele, kterými byli převážně učitelé, ale také další pracovníci vysoké školy, jako je sekretářka nebo studijní referentka. Jak bylo zmíněno výše, důvodem, proč tento dotazník vznikl, je, že původní obecný dotazník neměl takový dopad na jiné typy uživatelských rolí než je student, jelikož se autor tohoto dotazníku pohybuje v odlišných sociálních skupinách. Tento dotazník cílil na získání informací o tom, jak o mobilních aplikacích smýšlejí ostatní typy uživatelů kromě studentů, a jestli vůbec o mobilní aplikace jeví zájem.

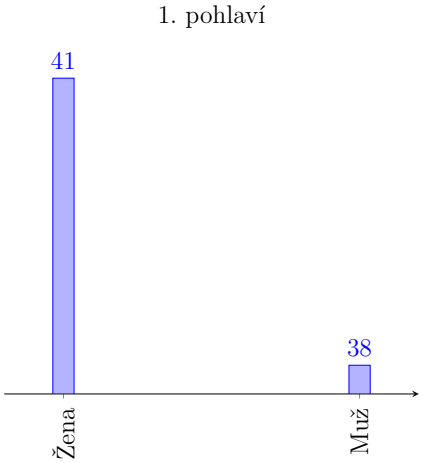
Protože se jednalo o cílený dotazník, byl poněkud stručnější. První otázkou je zjištěno, jestli daný uživatel jeví o mobilní aplikace zájem. Druhá otázka slouží pro zjištění nejpoužívanějších funkcí uživatele ve webovém prohlížeči IS/STAG.

Dotazník byl složen z následujících otázek:

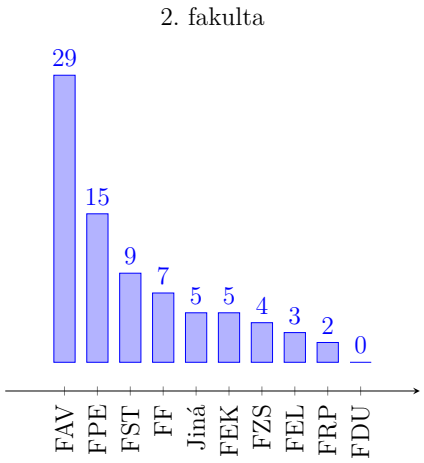
- Je pro Vás mobilní aplikace, která by uměla komunikovat s IS/STAG zajímavá?
- Pokud ano, jaké funkce IS/STAG využíváte nejvíce ve webovém prohlížeči, a byly by pro Vás zajímavé v mobilní aplikaci?

5.3.2 Obecný dotazník — odpovědi

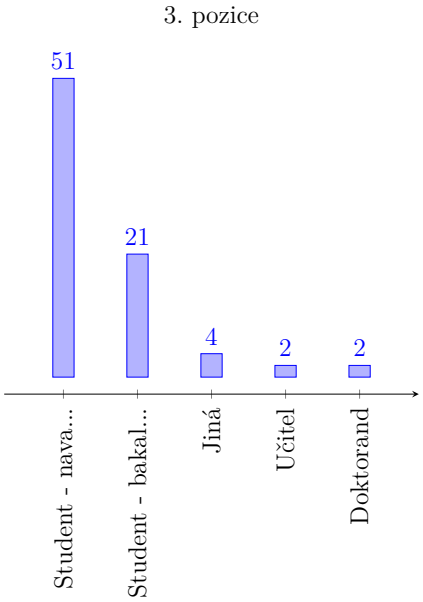
Obecného dotazníku se zúčastnilo celkem 79 respondentů. Odpovědi z dotazníků jsou mimo jiné zobrazeny na obrázcích 5.6, 5.7, 5.8, 5.9, 5.10. Graf obsahuje titulek, který odpovídá položené otázce.



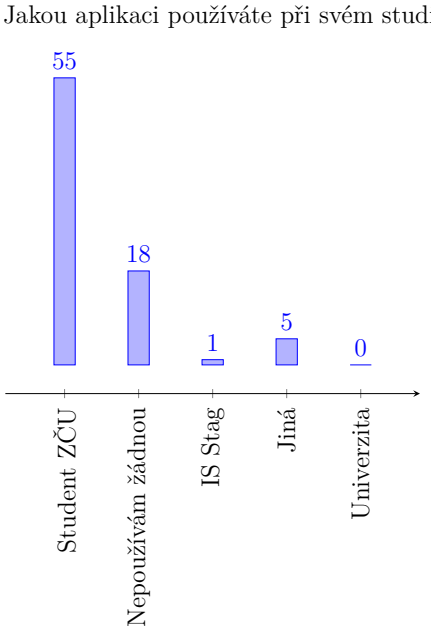
Obrázek 5.6: Výsledky 1. otázky z obecného dotazníku



Obrázek 5.7: Výsledky 2. otázky z obecného dotazníku



Obrázek 5.8: Výsledky 3. otázky z obecného dotazníku



Obrázek 5.9: Výsledky 4. otázky z obecného dotazníku

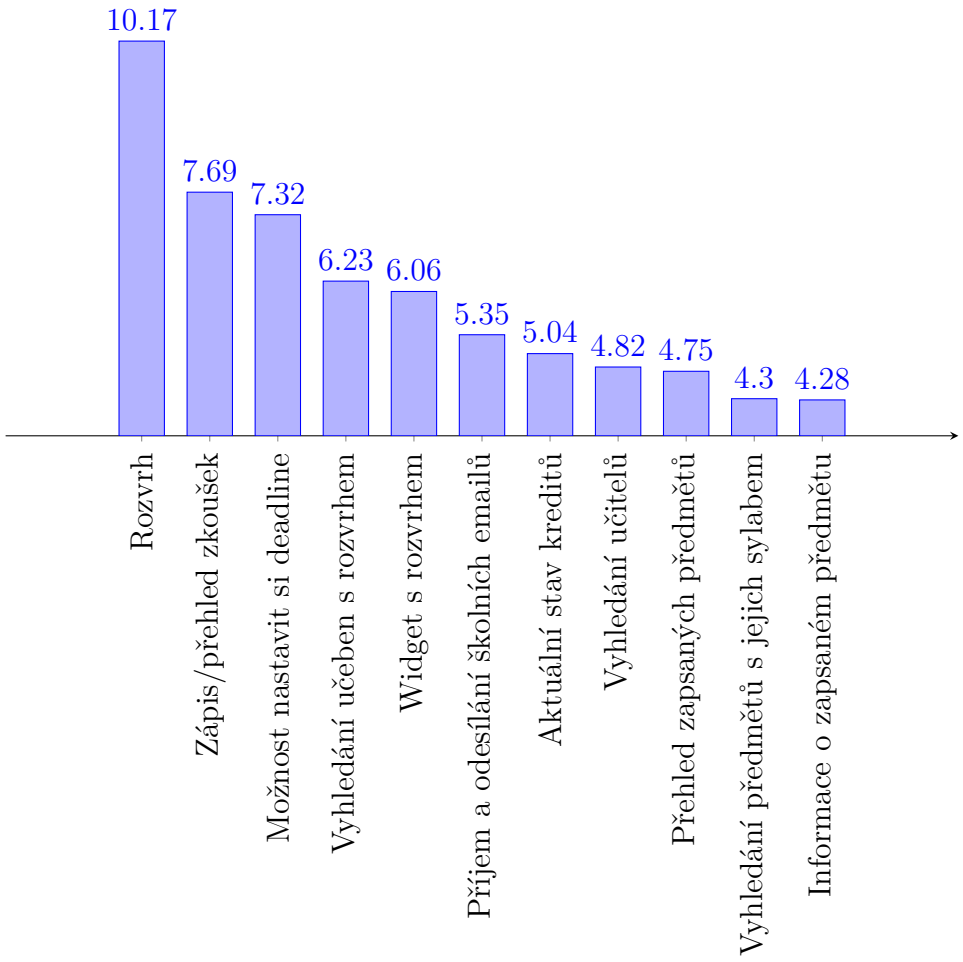
5. Pokud nějakou aplikaci používáte, chybí vám v ní nějaká funkce, kterou používáte na portále? Z celkových 79 respondentů odpovědělo na tuto otázku 30 respondentů. Mezi nejčastější odpovědi patří (seřazeno od nejčastějších):

- email s možností odpovědi,
- sledování stavu kreditů,

- odkaz na Courseware z aplikace,
- přepnutí rozvrhu na konkrétní rozsah,
- notifikace spojené se zkouškami (nové termíny, zapsání známek, upozornění na nadcházející zkoušky),
- mapa s budovami fakult.

6. Pokud žádnou aplikaci nepoužíváte, proč? Z celkových 78 respondentů, odpovědělo na tuto otázku 17 respondentů. Mezi nejčastější odpovědi patří (seřazeno od nejčastějších):

- respondent si vystačí s webovým prohlížečem,
- aplikace byla nějakou dobu nefunkční, což vedlo k tomu, že jí respondent přestal používat,
- nekompatibilní telefon,
- nevyhovující aplikace.



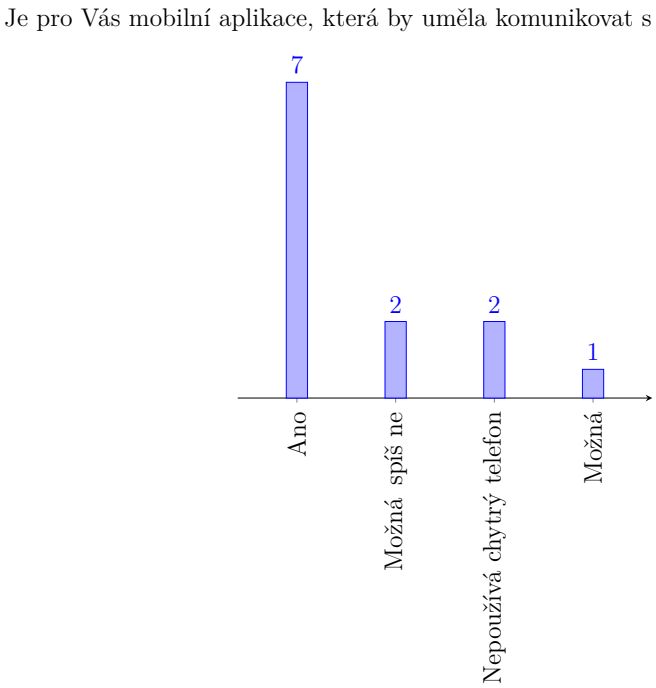
Obrázek 5.10: Výsledky 7. otázky z obecného dotazníku

8. Nějaká další myšlenka, o kterou by ses chtěl podělit? Nějaká další funkce, kterou portál obsahuje, ale není v možnostech? Z celkových 78 respondentů, odpovědělo na tuto otázku 17 respondentů. Mezi nejčastější odpovědi patří (seřazeno od nejčastějších):

- žádost o ubytovací stipendium a přehled plateb,
- obsazenost menzy,
- možnost změnit předmětu třídu/budovu,
- aktuality,
- upozornění na jednotlivé předměty před začátkem výuky.

5.3.3 Cílený dotazník — odpovědi

Cíleného dotazníku se zúčastnilo 12 respondentů z celkových 13 dotázaných. Odpovědi z dotazníků jsou mimo jiné zobrazeny na obrázku 5.11. Graf obsahuje titulek, který odpovídá položené otázce.



Obrázek 5.11: Výsledky 1. otázky ze zaměřeného dotazníku

Pokud ano, jaké funkce IS/STAG využíváte nejvíce ve webovém prohlížeči, a byly by pro Vás zajímavé v mobilní aplikaci? Mezi nejčastější odpovědi patří (seřazeno od nejčastějších):

- zobrazení rozvrhu,
- seznam studentů na rozvrhových akcích,
- hromadný email,
- přehled termínů a zkoušek,
- kvalifikační práce (základní informace o pracích, u kterých je učitel veden),
- přehled (následující hodina, číslo týdne),
- vyhledání studentů a zobrazení informací o nich,

- Courseware,
- vyhledání místností s rozvrhem,
- kvalita výuky (pro studenty),
- zadávání známek,
- export rozvrhu do kalendáře.

5.4 Vyhodnocení výsledků

V rámci analýzy byl proveden průzkum, který se zaměřoval na mobilní aplikaci, která by umožnila komunikovat s webovými službami IS/STAG. Cílem bylo tedy zjistit od uživatelů klíčové vlastnosti, které by v mobilní aplikaci ocenili. Při analýze nebyl brán ohled na mobilní platformu. Naopak, byl brán ohled na různé typy uživatelů, kteří mohou se systémem pracovat, a také byly zohledněny jejich různé kombinace (doktorand/učitel).

Na základě výsledků obecného dotazníku můžeme prohlásit, že mezi nejdůležitější vlastnosti pro studenty patří: zobrazení rozvrhu, správa zkoušek, přehled zapsaných předmětů, školní email a aktuální stav kreditů.

Ze zaměřeného dotazníku, který byl směřován na ostatní role (učitel, doktorand, sekretářka), vyplynulo, že nejdůležitější funkce jsou rozvrh, seznam studentů na rozvrhových akcích a hromadný email.

Tato analýza poskytuje především obecný pohled na to, jak lidé přemýšlejí nad spojením studia/práce s mobilní aplikací pro co největší možný přehled a usnadnění práce. Na základě získaných výsledků bude navrhnuta mobilní aplikace, který bude vycházet z dat získaných od respondentů.

6 Návrh aplikace

V této kapitole budou navrženy funkce aplikace, které budou implementovány v první fázi vývoje, tzn. v rámci této diplomové práce. Následující směr vývoje je popsán v kapitole 8.

6.1 Struktura budoucí aplikace

Cílem diplomové práce je vytvořit mobilní aplikaci fungující na webových službách IS/STAG pro více uživatelských rolí. Role student a učitel jsou dvě nejvyužívanější role v systému, a proto byly vybrány pro implementaci v první fázi vývoje.

Návrh funkcí aplikace bude vycházet z analýzy popsané v kapitole 5. Z analýzy vyplynuly podklady, na základě kterých budou vybrány funkce, které budou implementovány v první fázi vývoje. Protože se jednalo o rozsáhlou analýzu funkcí aplikace, bude vybrána podmnožina funkcí, z důvodu velké časové náročnosti a určitého rozsahu diplomové práce. To, že aplikace nebude obsahovat ihned všechny funkce, nebude mít na její funkčnost žádný větší vliv, protože budou vybrány ty nejdůležitější funkce, které by měly pokrýt běžné potřeby uživatelů.

Zmíněné atributy, navržené pro zobrazení v komponentách, jsou vybrány na základě zkušeností autora této diplomové práce nebo jsou inspirovány jinými existujícími aplikacemi, popsanými v kapitole 5.2. Navržená data k zobrazení v jednotlivých pohledech by měla obsáhnout všechny požadavky, které by mohl uživatel požadovat k zobrazení.

Uvedené webové služby, jež budou sloužit jako zdroj dat pro pohled nebo funkci, vycházejí z dokumentace IS/STAG, popsané na URL https://stag-ws.zcu.cz/ws/web. Každé webové službě odpovídá název operace. V textu bude odkazování pouze na tyto operace, pro lepší čitelnost. Příklad: místo adresy */ws/services/rest2/orion/getKalendarOsoby* bude uvedeno *getKalendarOsoby*.

Aplikace bude implementována tak, aby umožňovala přihlášení pouze jednoho uživatele. Přihlášení více uživatelů nebude možné. Z tohoto faktu plyne návrh databáze, která bude mít spíše funkci *cache* pro konkrétního uživatele.

6.1.1 Komunikace s webovými službami

Veškerá komunikace s webovými službami bude probíhat asynchronně z toho důvodu, aby nedocházelo k blokování uživatele. Pokud se budou nějaká data načítat, bude obsah nahrazen informací o tom, že jsou data načítána. Po načtení dat bude automaticky obsah překreslen s novými daty. Pokud při komunikaci nastane nějaká chyba, nebudou načtena žádná data.

6.1.2 Obecná část

Tato kapitola popisuje obecné funkce, které budou obsahovat všechny role přihlášené do aplikace.

Přihlášení

Na základě popisu autentizace webových služeb IS/STAG v kapitole 2.1.1 by bylo vhodné autentizovat uživatele v aplikaci. Díky autentizaci bude aplikace moci získat všechna potřebná data pro fungování celé aplikace. Aby uživatel mohl používat aplikaci, bude vyžadováno jeho přihlášení. Pro autentizaci uživatele bude použita metoda externího ověření, protože se jedná o doporučovaný způsob autentizace uživatele používajícího webové služby IS/STAG.

Výběr univerzity

Aktuálně nad IS/STAG funguje 13 vysokých škol. Aplikace bude umožňovat komunikaci s webovými službami každé univerzity. Výběr univerzity bude spojený s přihlášením tak, aby bylo možné odkázat pro přihlášení na správnou univerzitu. Na přihlašovací obrazovce si uživatel zvolí svou univerzitu. Výpis univerzit bude obsahovat jméno univerzity a její logo.

Vysoká škola má doménu, na které provozuje webové služby. Seznam těchto domén je možné získat operací *getSeznamProvozovanychWS*. Z toho důvodu, že se neočekává velká fluktuace vysokých škol používajících služby IS/STAG, budou tyto domény staticky zaneseny do zdrojového kódu.

Denní přehled

Denní přehled, neboli výchozí obrazovka aplikace po přihlášení, bude obsahovat dvě komponenty. První komponentou je přehled zbývajících povinností pro konkrétní den a druhou částí jsou poznámky.

Přehled zbývajících povinností bude zobrazovat aktuálně probíhající nebo budoucí rozvrhovou akci. Z komponenty bude možné přejít na rozvrh. Rozvrhová akce bude zobrazovat čas konání, název rozvrhové akce a učebnu. Typ rozvrhové akce bude kromě textu představovat také barva pozadí. Přednášky budou šedé, cvičení světle zelené, semináře tmavě zelené a zkoušky a zápočty žluté. Pokud uživatel klikne na rozvrhovou akci, zobrazí se její detail (viz 6.1.2). Jestliže uživatel nebude mít v aktuální den žádné rozvrhové akce, zobrazí se text s touto informací. Data do komponenty se budou načítat asynchronně.

Druhou komponentou budou poznámky. Uživatel si bude moci napsat vlastní poznámky do aplikace. K poznámce bude možné nastavit datum, ze kterého bude komponenta počítat počet dní do vypršení. Uživatel si bude moci nastavit například mezní termín úkolu, termín zkoušek nebo jiné poznámky. Kliknutím na poznámku se otevře její editace. Komponenta bude obsahovat tlačítka pro zobrazení všech poznámek i pro přidání poznámky. Pokud uživatel nebude mít vytvořené žádné poznámky, zobrazí se text s touto informací. Podrobnější popis poznámek je popsán v kapitole 6.1.2.

Při načítání dat o rozvrhových akcích bude nutné určit, pro jakou roli se data načítají. V případě studenta bude použita operace *getRozvrhByStudent*. Pokud se bude jednat o učitele, bude nutné použít jinou operaci, a to *getRozvrhByUcitel*. Když uživatel nebude mít roli student nebo učitel, nezobrazí se žádná data. Poznámky budou načítány z *SQLite* databáze, pomocí frameworku *Core Data*.

Zobrazení rozvrhu

V rozvrhu bude možné vybírat konkrétní datum pro zobrazení rozvrhových akcí uživatele. Bude poskytnuta možnost zrychlené volby na aktuální den. V první fázi vývoje půjdou zobrazit rozvrhové akce pouze pro jeden konkrétní den, kde budou všechny vypsány pod sebou.

Jednotlivé rozvrhové akce budou seřazeny podle času, vzestupně. Pokud bude víc rozvrhových akcí začínat ve stejný čas, budou seřazeny podle názvu. Rozvrhová akce bude obsahovat čas konání, dobu trvání, typ, název, zkratku katedry a názvu rozvrhové akce, jméno učitele a učebnu. Typ rozvrhové akce bude kromě textu také představovat barva pozadí, stejně jako je tomu v denním přehledu. Přednášky budou šedé, cvičení světle zelené, semináře tmavě zelené a zkoušky a zápočty žluté. Pokud uživatel klikne na rozvrhovou akci, zobrazí se její detail. Jestliže uživatel nebude mít v aktuální den žádné rozvrhové akce, zobrazí se text s touto informací.

Při načítání dat o rozvrhových akcích bude nutné určit, pro jakou roli se

data načítají. V případě studenta bude použita operace *getRozvrhByStudent*. Pokud se bude jednat o učitele, bude nutné použít jinou operaci a to *getRozvrhByUcitel*. Když uživatel nebude mít roli student nebo učitel, nezobrazí se žádná data.

Detail rozvrhové akce

Detail rozvrhové akce bude obsahovat detailní informace o konkrétní rozvrhové akci, která byla zvolena v rozvrhu nebo v denním přehledu. Detail rozvrhové akce bude zobrazovat celý název, místnost, čas konání, typ, zkratku katedry, zkratku názvu, počet kreditů, týdenní rozsah hodin, formu a způsob zakončení předmětu, garanta předmětu, vyučující předmětu, podmíněné předměty, cíl předmětu, požadavky na studenta, obsah předmětu a doporučenou literaturu. Data by měla být zobrazena v čitelném formátu. Pokud budou data z webové služby obsahovat odkaz na *Courseware*, bude v rohu obrazovky ikona pro otevření tohoto linku v internetovém prohlížeči.

Pokud to bude možné, bylo by vhodné zobrazit studenty na dané rozvrhové akci. U těchto studentů bude zobrazeno jméno, příjmení, ročník, fakulta a studovaný obor.

Data se budou načítat asynchronně v momentě otevření detailu. Pro načtení informací o rozvrhové akci bude použita operace *getPredmetInfo*. Informace o studentech budou načteny přes operaci *getStudentiByRoakce*.

Poznámky

V aplikaci bude možné psát si vlastní poznámky. Poznámka bude obsahovat nadpis, popis a možnost zvolit datum pro nastavení mezního termínu. Poznámky budou zobrazeny na denním přehledu a ve výpisu všech poznámek. Nad poznámkami bude možné vykonávat operace zobrazení, editace, vytváření a mazání. Ve výpisu poznámek bude poznámka obsahovat nadpis a popis poznámky na maximálně tři řádky, zbytek bude oříznut.

Poznámky budou uloženy v *SQLite* databázi v paměti telefonu. Pro přístup do databáze bude použit *framework Core Data*. Poznámky budou napojeny na uživatele přes jeho přihlašovací jméno.

Lokalizace aplikace

Jelikož na vysokých školách je poměrně dost cizinců, aplikace bude podporovat dva jazyky, a to češtinu a angličtinu. Jako výchozí jazyk aplikace se bude načítat jazyk nastavený dle systému, kde jako výchozí jazyk, pokud systémový jazyk nebude čeština ani angličtina, bude zvolena angličtina.

Volbu jazyka aplikace bude možné nastavit v aplikaci, která přetíží systémový jazyk. Volba jazyka aplikace nebude ztracena ani po úplném vypnutí aplikace.

Volba jazyka bude uložena jako *String*, který bude obsahovat kód zvoleného jazyka. Protože volba jazyka bude uložena jako primitivní datový typ, bude tato hodnota uložena v *UserDefaults*.

Jídelníček

Protože se jedná o aplikaci vyvíjenou na ZČU, bylo by vhodné, aby aplikace obsahovala informace specifické pro tuto vysokou školu. Oficiální stránky ZČU zahrnují webovou službu pro získání jídelníčku v menze na Borech a v ulici Kollárova. V aplikaci bude možné zobrazit tato data. Menu bude rozděleno podle menzy, a dále na hlavní jídla a polévky. U každého jídla budou zobrazena čísla alergenů a ceny pro různé strávníky (student, zaměstnanec, externí). Bude možné zvolit konkrétní den pro zobrazení jídelníčku.

Pro načtení dat jídelníčku konkrétní menzy bude použita webová služba *www.zcu.cz/rest/canteen/{id}*. Hodnota *id* dovoluje nastavit, pro kterou menzu má vrátit výsledky. V případě menzy na Borech se doplní „B“, v případě menzy v Kollárově ulici „K“. Webová služba dovoluje přes parametr *daysShift* posunout jídelníček. Hodnotou je tedy počet dní pro posunutí.

6.1.3 Studentská část

V této kapitole budou navrženy funkce aplikace pro roli student. Diagram užití je zobrazen na obrázku 6.1.

Přehled zkoušek

Aplikace bude zobrazovat aktuálně vypsané zkoušky pro přihlášeného studenta. Zkoušky budou rozděleny podle předmětů. U každé zkoušky bude uveden název předmětu, informace, zda se jedná o zkoušku nebo zápočet, datum a čas konání, jméno zkoušejícího, místo konání, obsazenost termínu, poznámka od učitele a informace, v jakém časovém rozmezí je možné se na termín zapsat či odepsat. Pozadí komponenty představující termín bude mít modrou barvu, pokud je možné se zapsat, červená barva bude představovat plně obsazený termín a žlutá barva bude nastavena, pokud je student na termínu zapsán. Kromě barevného označení bude komponenta obsahovat tlačítko pro provedení akce zápisu, odzápisu nebo informaci o tom, proč se nedá na daný termín zapsat. Pokud student nemá žádné zkoušky, zobrazí se mu tato informace.

Zkoušky budou načítány operací *getTerminyProStudenta*.

Zápis a odzápis zkoušek

Z přehledu zkoušek bude možné provést zapsání či odepsání termínu. Po stisku tlačítka pro provedení akce by měl být uživatel informován o výsledku operace. Pokud operace proběhne v pořádku, kromě informace o úspěšném provedení akce se také změní pozadí komponenty termínu dle zmíněných pravidel a jeho obsazení. V případě neúspěšné operace bude uživatel pouze informován.

Pro zápis zkoušek existuje operace *zapisStudentaNaTermin*, pro odzápis zkoušky se používá *odhlasStudentaZTerminu*.

Profil studenta

Na profilu studenta budou zobrazeny tyto informace: jméno a příjmení, osobní číslo, email, zkratka katedry, studovaný obor a ročník. K těmto informacím budou zobrazeny ještě jeho průběžné statistiky. Tyto průběžné statistiky studenta budou zahrnovat aktuální stav kreditů a celkový vážený průměr známek za celý průběh studia. Dále by měl profil studenta zobrazovat jeho výsledky studia. Průběžné statistiky budou počítány právě z těchto výsledků.

Aktuální stav kreditů bude počítán z úspěšně absolvovaných předmětů. Předpokládaný celkový počet kreditů, který by měl student získat, bude 60 kreditů za každý předpokládaný rok studia.

Pro získání těchto všech dat bude nutné zavolat několik webových služeb. Informace o studentovi je možné získat operací *getStudentInfo*. Pro výpis všech požadovaných informací bude nutné složit dvě operace. Operaci *getPredmetyByStudent* a *getZnamkyByStudent*. Z toho důvodu, že neexistuje univerzální způsob, jak získat všechny tyto informace pomocí jedné operace. Jelikož se tato data během roku často nemění, budou tyto informace uloženy do *SQLite* databáze pro ušetření síťové komunikace. Data budou načtena vždy během přihlášení uživatele. Pro jejich správu bude použit framework *Core Data*. Tato uložená data budou odstraněna z databáze v momentě odhlášení uživatele.

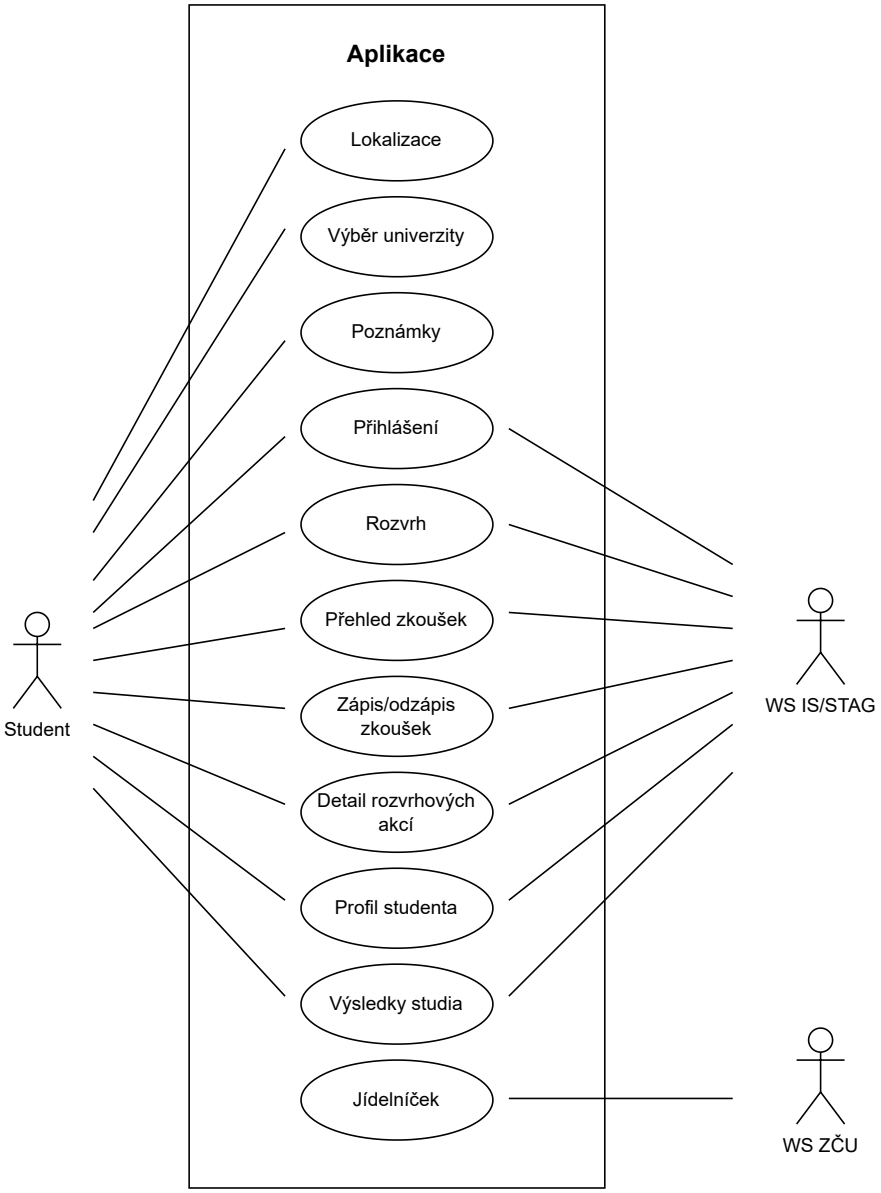
Zobrazení výsledků studia

Výsledky studenta budou zobrazeny na profilu studenta a budou seskupeny podle školních roků, mezi kterými bude možné přepínat. U každého seskupení bude uvedena informace o celkovém počtu zapsaných kreditů, o počtu

získaných kreditů, vážený průměr známek za školní rok, celkový počet zapsaných předmětů a počet úspěšně absolvovaných předmětů.

Jednotlivé předměty v seskupení budou dále rozděleny na letní a zimní semestr. Každá komponenta, představující studovaný předmět, bude obsahovat název předmětu, zkratku katedry, zkratku předmětu, počet kreditů, datum se jménem a příjmením učitele, který udělil zápočet, datum se jménem a příjmením učitele, který udělil zkoušku a hodnocení předmětu. Pokud není ještě udělena známka z předmětu, nebudou vyplněny hodnoty spojené se zápočtem, zkouškou a hodnocením.

Proces pro získání dat je totožný se získáním dat pro profil studenta, který je popsán v předchozím bodě 6.1.3.



Obrázek 6.1: Diagram užití pro roli student

6.1.4 Učitelská část

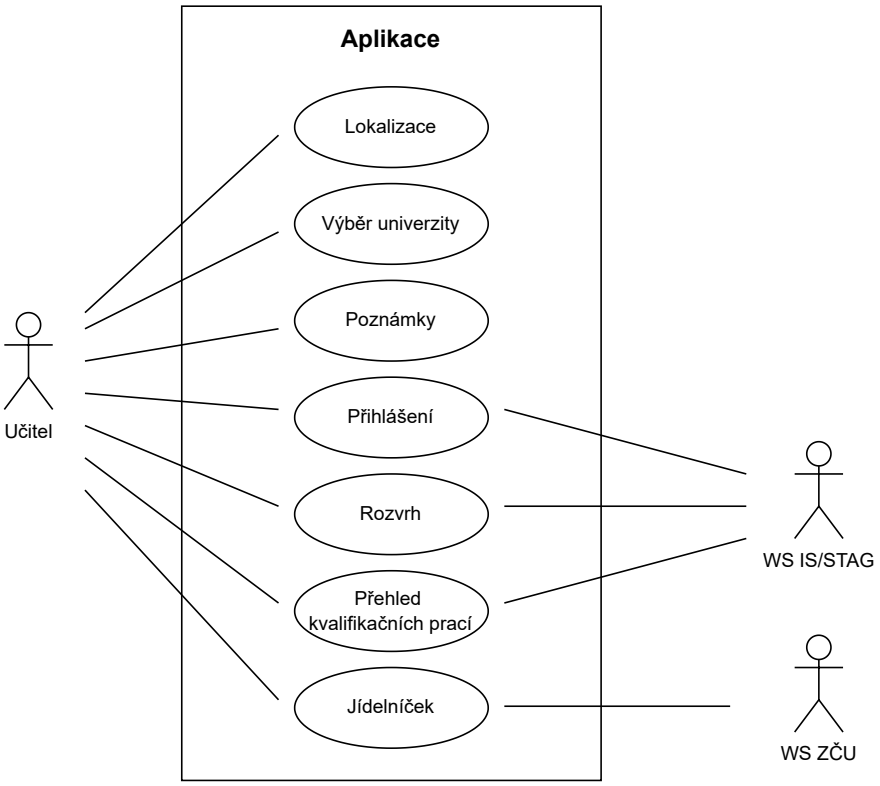
V této kapitole budou popsány funkce, které jsou nad rámec obecných funkcí spojených s rolí učitel. Diagram užití je zobrazen na obrázku 6.2.

Přehled kvalifikačních prací

Učitel si bude moci zobrazit kvalifikační práce, u kterých je pro aktuální akademický rok přiřazen. U každé kvalifikační práce učitel uvidí tyto infor-

mace: typ, název, fakultu, jméno a příjmení studenta, přihlašovací jméno studenta, které slouží pro odvození e-mailu na studenta, datum odevzdání a jeho vztah k práci, čímž je myšleno, jestli je učitel u práce veden jako vedoucí, konzultant, a tak dále.

Získání dat s kvalifikačními pracemi pro učitele je zajištěn operací *getKvalifikacniPrace*.



Obrázek 6.2: Diagram užití pro roli učitel

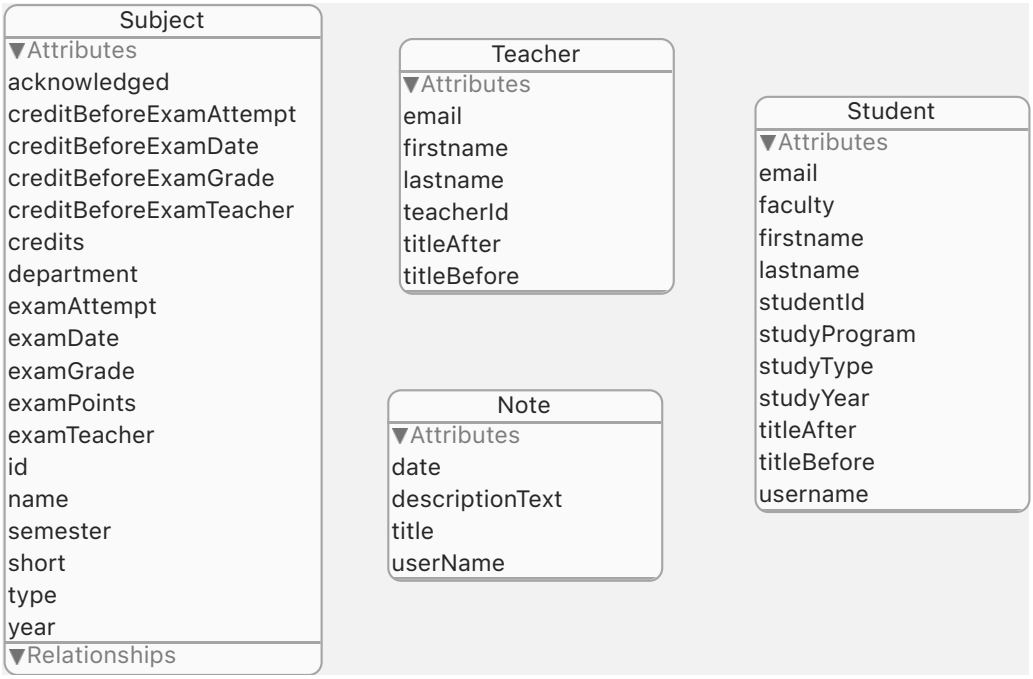
6.2 Databázová struktura

Jako databáze byla zvolena *SQLite* databáze v kombinaci s *Core Data* frameworkem z toho důvodu, že framework poskytuje *ORM* a zároveň nabízí další užitečné funkce. Dalším faktorem je to, že je framework poskytován přímo od *Apple* a pro jeho použití stačí použít interní knihovnu. Není tedy potřeba, aby aplikace byla závislá na třetí straně.

Návrh byl vytvořen pomocí editoru v *Xcode* pro tvorbu databázového návrhu *Core Data*. Náhled modelu z *Xcode* je možné vidět na obrázku 6.3. Primární klíče nejsou v modelu zobrazeny z toho důvodu, že si je framework spravuje sám.

Databáze bude mít roli *cache*, která uschová data i po úplném vypnutí aplikace. Kdyby byla data uložena pouze v *cache*, po vypnutí aplikace by došlo k jejich ztracení. Protože bude možné, aby byl v aplikaci přihlášený vždy pouze jeden uživatel, jsou zbytečné relace mezi tabulkami, protože se budou načítat vždy všechna data z tabulky a není nutné je napojovat přes cizí klíče. Veškerá data v tabulkách budou odstraněna při odhlášení uživatele (kromě dat v tabulce *note*). Zde jako jednoznačný identifikátor pro přiřazení poznámky k uživateli bude sloužit atribut *userName*, který bude obsahovat přihlašovací jméno uživatele. Data do tabulek budou načítána po úspěšném přihlášení uživatele. Tabulky v systému budou mít následující využití:

- *Student* — pokud je přihlášený uživatel s rolí student, budou do této tabulky uloženy informace o něm, získané z externí přihlašovací služby.
- *Teacher* — pokud je přihlášený uživatel s rolí učitel, budou do této tabulky uloženy informace o něm, získané z externí přihlašovací služby.
- *Subject* — pokud je přihlášený uživatel s rolí student, budou v této tabulce uloženy jeho studijní výsledky, data do této tabulky jsou složena ze dvou operací *getPredmetyByStudent* a *getZnamkyByStudent*.
- *Note* — do této tabulky se budou ukládat poznámky, které si uživatel v aplikaci vytvoří.



Obrázek 6.3: Návrh databázového modelu v *Xcode*

7 Popis implementace

Tato kapitola popisuje detailní implementaci návrhu aplikace a její jednotlivé části. Před samotnou implementací je nutné zvolit minimální verzi *SDK*, pro kterou bude možné aplikace nainstalovat. Na základě popisu platformy *iOS* v kapitole 3.1.2, byl zvolen *iOS 15* z důvodu zachování co nejdelší podpory a přístupu k nejnovějším verzím knihoven a frameworků. Aktuálně ke květnu 2022 to znamená podporu 72 %[10] všech zařízení se systémem *iOS*.

7.1 Struktura aplikace

Adresářová struktura aplikace je členěna do několika částí a dodržuje architekturu *MVVM*. Níže je rozepsána adresářová struktura aplikace s popisem, co je v dané složce uloženo.

Facades

Obsahuje třídy s návrhovým vzorem fasáda. Jedná se o třídy, které vytvoří jednoduché rozhraní pro jinou třídu tím, že poskytují ucelené rozhraní k jiným třídám. Namísto obsažení referencí nespočtu tříd, stačí pouze jedna reference na fasádu, která obsahuje veškeré reference za příslušnou třídu.

Calculators

Zde jsou uloženy třídy sloužící pro výpočty v aplikaci. Obsahuje například výpočty statistik pro průběh studentovo studia.

Repositories

Zdrojové kódy pro práci s databází.

CoreDataModel

Obsahuje konfigurační soubor pro konfiguraci databázového modelu pomocí *Core Data*.

Base

Obsahuje sdílené třídy pro práci s daty napříč celou aplikací. Tyto třídy jsou dále rozděleny do dalších složek podle jejich působnosti.

- *Helpers* — pomocné třídy pro práci s daty.
- *Extensions* — obsahuje rozšíření existujících tříd.
- *Modifiers* — obsahuje modifikátory, které se používají v šablonách.
- *Styles* — definice společných *UI* stylů komponent.
- *Constants* — konstanty používané v aplikaci.
- *Enums* — obsahuje výčtové typy aplikace.
- *Views* — obsahuje společné *UI* komponenty napříč moduly.
- *Mappers* — třídy pro mapování dat na objekty.
- *Entities* — obsahuje veškeré třídy entit používané v aplikaci.

Modules

Každý pohled má svůj modul. Modul je dále členěn na *Screen*, *Views* a *ViewModel*.

- *Screen* — základní třída pohledu, vždy jedna třída.
- *Views* — komponenty pohledu. Obsahuje třídy, které představují komponenty v základním pohledu.
- *ViewModel* — obsahuje třídu *ViewModel* pro konkrétní pohled.

Services

Obsahuje služby používané v aplikaci. Zahrnuje třídy pro komunikaci s různými webovými službami nebo pro práci s lokalizací aplikace.

Managers

Složka obsahuje třídy pro práci s daty.

Preview Content

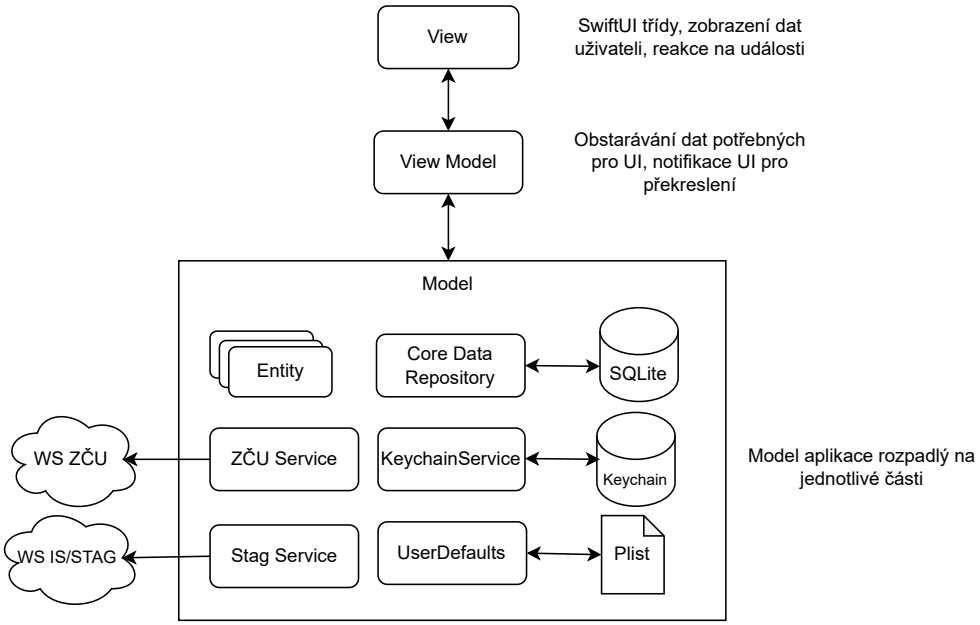
Definice obrázků používané v mobilní aplikaci.

Nezařazené soubory

- *StagApp.swift* — třída, představující vstupní bod aplikace
- *ContentView.swift* — třída, která obsahuje obsah aplikace. Třída je volána ze vstupního bodu.
- *Info.plist* — konfigurační soubor aplikace
- *Localizable.strings* — lokalizační soubor, obsahující překlady textů v aplikaci.

7.2 Architektura

Jako architektura aplikace byla zvolena architektura *MVVM*, z důvodu popsaných u popisu architektur. Na obrázku 7.1 je znázorněno schéma vytvořené aplikace.



Obrázek 7.1: Architektura vytvořené aplikace

Pohled (*View*) aplikace představují *SwiftUI* třídy, které obsluhují řízení celé aplikace. Třídy reagují na události, na základě kterých jsou těmto třídám poskytnuta data z *View Modelu*. Pro přehlednost má každý pohled právě jeden *View Model*.

View Model představuje prostředníka mezi *View* a *Modelem*. *View Model* je vytvářen ve *SwiftUI* třídě a přes konstruktor mu jsou poskytnuty všechny

jeho potřebné závislosti. Tyto závislosti jsou ve většině případů třídy, komunikující s vrstvou *Modelu*. Každý *View Model* implementuje rozhraní *ObservableObject*, které umožní *SwiftUI* detekovat změny proměnných ve *View Modelu*.

Vrstva *Model* je složena z několika částí, jednotlivé části je možné vidět na obrázku 7.1. Vrstva poskytuje zdroj dat pro *View*. V aplikaci se používá hned několik zdrojů: dva externí, mezi něž patří webové služby IS/STAG a ZČU. Další zdroje jsou lokální a skládají se z *SQLite* databáze, se kterou se komunikuje přes *Core Data Repository*, *KeychainService*, která obstarává komunikaci s *Keichain* databází. Posledním je *plist*, který se využívá v *UserDefaults*.

7.3 UI

Pro uživatelské rozhraní je zvolen framework *SwiftUI* z toho důvodu, že je prezentován od *Apple* jako moderní způsob pro vytváření mobilních aplikací. Dalším důvodem je fakt, že *SwiftUI* funguje perfektně s architekturou *MVVM*, která byla zvolena. Zároveň framework umožňuje vytvářet multiplatformní uživatelské rozhraní. V budoucnu se tím umožní snadnější implementace *UI* například pro *iPad* nebo *Apple Watch*.

Uživatelské rozhraní je navrženo tak, aby odpovídalo současným směrům a trendům. Cílem bylo navrhnout minimalistické uživatelské rozhraní. Jako vzory byly použity již existující aplikace nebo známá platforma *dribble.com*, která obsahuje spoustu zajímavých grafických návrhů všech typů aplikací. V současné době aplikace nepodporuje tmavý režim.

7.4 Ukládání dat

Aplikace funguje s několika různými způsoby uložení dat. Níže jsou popsány všechny typy úložišť, která jsou v aplikaci využita. U každého typu je uveden důvod výběru, jak je s úložištěm manipulováno, a jaká data obsahuje.

7.4.1 UserDefaults

V tomto úložišti jsou uloženy primitivní datové typy, které jsou použity globálně v aplikaci, a je tedy nutné mít k nim rychlý a snadný přístup. *SwiftUI* poskytuje obal (*wrapper*) na toto úložiště, nazvaný *AppStorage*, přes který je možné přistupovat k uloženým proměnným jak pro čtení, tak pro zápis

a pokud je hodnota použita v kódu a změní se, *SwiftUI* zareaguje a překreslí obrazovku. Tímto způsobem je například vyřešeno přihlášení. Pokud dojde odkudkoliv z kódu ke změně hodnoty *isLogged*, zobrazí se uživateli obrazovka s přihlášením.

Aplikace k tomuto úložišti přistupuje dvěma způsoby. Pokud je nutné reagovat na změny v tomto úložišti přímo v šabloně, například pro kontrolu přihlášení, viz popis v předchozím odstavci, tak se k těmto datům přistupuje v šabloně přes obal *AppStorage*. V ostatních případech se k hodnotám v úložišti přistupuje přes instanci *singletone UserDefaults*.

Data uložená v *UserDefaults*:

- *isLogged* — *bool* hodnota, která obsahuje informaci o tom, zda je uživatel přihlášen.
- *selectedUniversity* — *int* hodnota, jež obsahuje interní ID zvolené univerzity. Na základě této proměnné se vybírá doména pro webové služby nebo obrázek univerzity při přihlášení.
- *language* — *string* proměnná, obsahuje kód jazyka dle mezinárodního standardu *ISO 639-1*.
- *isStudent* — *bool* proměnná, která určuje, zda je přihlášený uživatel student. Na základě této hodnoty jsou uživateli zobrazeny/skryty funkce.
- *hasTeacherId* — *bool* proměnná, jež určuje, zda přihlášený uživatel má číslo učitele. Na základě této hodnoty jsou uživateli zobrazeny/skryty funkce.

Tato data nejsou nikdy vymazána, slouží jako základní nastavení aplikace. Hodnoty *isStudent* a *hasTeacherId* jsou vyplněny při přihlášení uživatele do systému.

7.4.2 Keychain

Aplikace toto úložiště používá pro ukládání citlivých dat, pro které není doporučeno uložení ve volně přístupném úložišti. Je nutné zachovat jejich původní podobu.

Pro přístup k tomuto úložišti je vytvořena třída *KeychainManager*, která obaluje veškerý přístup k tomuto úložišti.

Uložená data v *Keychain*:

- *cookie* — *string* hodnota obsahující autorizační *cookie* získanou z WS IS/STAG.
- *username* — *string* hodnota, která obsahuje přihlašovací jméno uživatele, nebo-li *stagLogin* nutný u webových služeb.

Všechna data v *Keychain* jsou citlivá, a proto jsou odstraněna z úložiště v momentě odhlášení uživatele.

7.4.3 Core Data

Aplikace si ukládá data, která potřebuje zachovat i po úplném ukončení aplikace. Pro uchování těchto dat byl vybrán framework *Core Data*, který ukládá data do databáze. Framework poskytuje *ORM*, který značně usnadní práci s objekty a zrychlí vývoj. Přehled tabulek s atributy je zobrazen na obrázku 6.3.

Definice datového modelu je uložena v souboru *CoreDataModel*. Pro přístup ke každé tabulce existuje samostatná třída s návrhovým vzorem *Repository*. Tyto třídy dále dědí *CoreDataRepository*, která obstarává základní komunikaci s frameworkem.

Veškerá data z databáze jsou odstraněna po odhlášení uživatele, kromě tabulky *Note*, ta zůstává. Z tohoto popisu chování vzniká problém s propojením uživatele s jeho poznámkami. Poznámky jako jedinečný identifikátor uživatele používají přihlašovací jméno uživatele.

7.5 Asynchronní komunikace

Asynchronní komunikace je řešena pomocí integrovaných funkcí ve *Swiftu*. Metody, které se volají asynchronně, musejí být označeny klíčovým slovem *async*. Při volání těchto asynchronních metod je nutné před volání doplnit klíčové slovo *await*. Kombinace těchto dvou klíčových slov má za výsledek to, že vykonávání kódu čeká na návrat z asynchronní metody předtím, než pokračuje dál.

7.6 Přihlášení uživatele

Tato kapitola popisuje proces přihlášení uživatele. Je zde popsán detailní popis implementace externího přihlášení a získání dat o uživateli. Proces přihlášení se skládá z těchto částí: autentizace uživatele, získání role uživatele, proces po přihlášení. Dále se zabývám i problémem s přihlášením uživatele.

7.6.1 Autentizace uživatele

Autentizace je řešena externí službou. Pro implementaci tohoto ověření je nutné otevřít *WebView*, z kterého se získají po úspěšném přihlášení data z externí služby. Pro tuto implementaci není možné využít *SwiftUI*, protože zatím neobsahuje podporu této komponenty. Byl použit framework *UIKit* a pomocí něj vytvořena komponenta *ExternalLoginWebView*, která se otevře po stisknutí tlačítka „Přihlásit se“ na přihlašovací obrazovce. *WebView* se otevře s přihlašovací obrazovkou dané univerzity s požadovanými parametry. *WebView* naslouchá na URL adrese, definované v předaném parametru *originURL*, na kterou je uživatel přesměrován po úspěšném přihlášení. Při přístupu na tuto URL *WebView* získá z URL parametry předané externí službou, vytvoří z nich objekt *ExternalLoginResult* a předá ho *ViewModelu*, který s ním dále pracuje. Nakonec se *WebView* zavře.

7.6.2 Získání role uživatele

V aplikaci je o roli uživatele rozhodnuto na základě informací o uživateli vrácených z externí přihlašovací služby *ExternalLoginResult*. Hodnoty jsou v atributu *stagUserInfo* zakódované v *Base64*. Pokud v datech existuje atribut *osCislo*, je rozhodnuto, že uživatel má roli student. Pokud v datech existuje atribut *ucitIdno*, je rozhodnuto, že uživatel má roli učitel. Je možné, aby měl uživatel obě role zároveň.

7.6.3 Proces po přihlášení

Po úspěšném přihlášení je zavolána metoda *processExternalLogin()* ve třídě *LoginViewModel*. Metoda jako parametr přijímá hodnotu *ExternalLoginResult*.

Pokud se jedná o uživatele s rolí student, je zavolána metoda *fetchStudentInfo()* z třídy *StagService* pro získání dat o studentovi. Data jsou uložena do databáze pomocí *StudentRepository*. Dalším krokem je získání výsledků zapsaných předmětů. Data jsou získána ze dvou zdrojů. První metoda *StagService::fetchSubjectResults()* vrací výsledky studia. Druhá metoda *StagService::fetchSubjects()* vrací dodatečné informace o předmětu, jako je například celé jméno předmětu, počet kreditů a další data, která neobsahují první dotaz a jsou důležitá pro zobrazení dat v pohledech. Data jsou uložena do databáze třídou *SubjectRepository*.

Když má uživatel roli učitel, z WS IS/STAG se získají dodatečné informace o učiteli pomocí metody *StagService::fetchTeacherInfo()*. Informace jsou následně uloženy do databáze za použiti *TeacherRepository*.

Přihlášení spravuje externí služba, takže pokud dojde k tomu, že se nepodaří ve *WebView* přihlásit uživatele buď z důvodu chyby, nebo neplatných přihlašovacích údajů, samotná služba informuje uživatele. V případě, že uživatel zavře sám *WebView*, nic se nestane a uživatel si může znovu otevřít *WebView* s přihlášením přes tlačítko „Přejít na přihlášení“.

7.6.4 Řešení problému s přihlášením

Při implementaci vyvstal problém, kdy z nespecifického důvodu nebylo možné otevřít přihlašovací obrazovku pro vysokou školu DEMO ZČU ve *WebView*. Vždy se otevřela pouze bílá stránka a v konzoli se zobrazila chyba *Error Domain=NSURLErrorDomain:-1017*. Po usilovném zkoumání problému se nepodařilo problém vyřešit. Problém spočívá pravděpodobně v nastavení serveru, protože na ostatních vysokých školách proběhlo přihlášení bez problému. Protože se jedná o testovací prostředí, bylo rozhodnuto, že se implementuje odlišný přihlašovací proces. Přihlašování je obstaráno na straně mobilní aplikace. Přihlášení je provedeno pomocí metody *StagService::fetchUserLogin()*, která obsahuje autorizační hlavičku se zadaným přihlašovacím jménem a heslem. Při neplatném přihlášení je zobrazena uživateli chyba. Pokud je přihlášení korektní, je z odpovědi serveru uložená autorizační *cookie*. Nevýhodou tohoto procesu je, že je autorizační *cookie* platná pouze několik hodin. V budoucnu bude tato nedokonalost opravena.

7.7 Komunikace s webovými službami

Pro komunikaci s webovými službami bylo nutné vyřešit, jak se komponenta zachová, pokud se pro ni momentálně načítají data. Bylo nezbytné zavést mezistav, který indikuje uživateli, že komponenta pracuje, aby nedocházelo k přechodu mezi výchozí hodnota → zobrazení načtených dat, protože rychlost načtení dat závisí na velikosti dat a rychlosti internetového připojení. Uživatel by se mohl mylně domnívat, že v daný den nemá rozvrh, přestože se pouze nestačila načíst data. V následujícím textu je popsáno řešení, jak dochází v pohledech k přechodu mezi stavy: výchozí hodnota → načítání dat → zobrazení načtených dat. Následuje popis implementace tříd a jejich metod pro komunikaci s různými API.

7.7.1 Uživatelské rozhraní

Veškerá komunikace s webovými službami probíhá asynchronně, aby uživatel nebyl blokován. Pokud se načítají data do komponenty, je její obsah nahra-

zen komponentou *LoadingView*, která obsahuje točící se kolečko s textem „Načítání“. Po načtení dat je komponenta *LoadingView* nahrazena načteným obsahem.

Tato logika je obsluhována ve *View Modelu* spadající pod pohled pomocí proměnné *state*, která je výčtový typ *AsyncState*. Před zavoláním asynchronní metody pro načtení obsahu se nastaví proměnná na hodnotu *fethingData*, po načtení obsahu se nastaví zpět na *idle*. V šabloně existuje přepínač (*switch*), který na základě této hodnoty překresluje komponenty. Metoda pro načtení dat je spojena se zobrazením komponenty.

7.7.2 Volání webových služeb ZČU

Webové služby ZČU, které se využívají pro načtení jídelníčku menzy obstarává třída *CanteenService*. Metoda *fetchMenu()* na základě parametrů vrací požadovaná data pro jídelníček. V parametrech je možné nastavit, pro kterou menzu se budou zobrazovat data, o kolik dnů bude posunut jídelníček, a jazyk, ve kterém bude jídelníček.

7.7.3 Volání webových služeb IS/STAG

Komunikaci s webovými službami IS/STAG zajišťuje třída *StagService*. Tato třída obsahuje metody pro komunikaci s WS IS/STAG. Při volání webové služby (kromě operace), je nutné zjistit, pro kterou univerzitu se bude operace vykonávat. Zároveň je možné definovat, v jakém formátu nebo jazyku bude odpověď ze serveru přes parametry *outputFormat* a *language*. K požadavku je také nutné přiložit autorizační *cookie* pro autentizaci uživatele.

Pro zjištění této konfigurace se používá metoda *getConfiguration()*, která z *UserDefaults* získá ID zvolené univerzity, na základě kterého se získá URL univerzity. Z *Keychain* obstará autorizační *cookie* a v poslední řadě získá nastavení jazyka přes službu *LanguageService*. Tato konfigurace je použita v metodě *getBaseRequest()*, která vytváří základní objekt požadavku na webovou službu. Parametr *outputFotmat* je staticky nastaven na hodnotu „json“, protože tato volba je preferována před výchozím formátem XML.

7.8 Lokalizace

Aplikace podporuje češtinu a angličtinu, jako výchozí jazyk je zvolen systémový jazyk. Jazyk je možné zvolit v nastavení aplikace. Pokud systémový jazyk není čeština ani angličtina, je jako výchozí jazyk zvolena angličtina.

O správnou lokalizací textů se stará třída *LanguageService*, která na základě nastavené hodnoty *language* v *UserDefaults* vrací kód jazyka. Třída je implementována podle návrhového vzoru *singletone*, takže je možné k ní přistoupit z kódu odkudkoliv.

Lokalizace pro aplikaci je nastavena ve *SwiftUI* pohledu *StagApp*. Pohled představuje počáteční bod aplikace, který obsahuje další pohledy. Přes metodu *environment* je nastaven klíč *locale*, který změní jazyk ve všech ostatních šablonách.

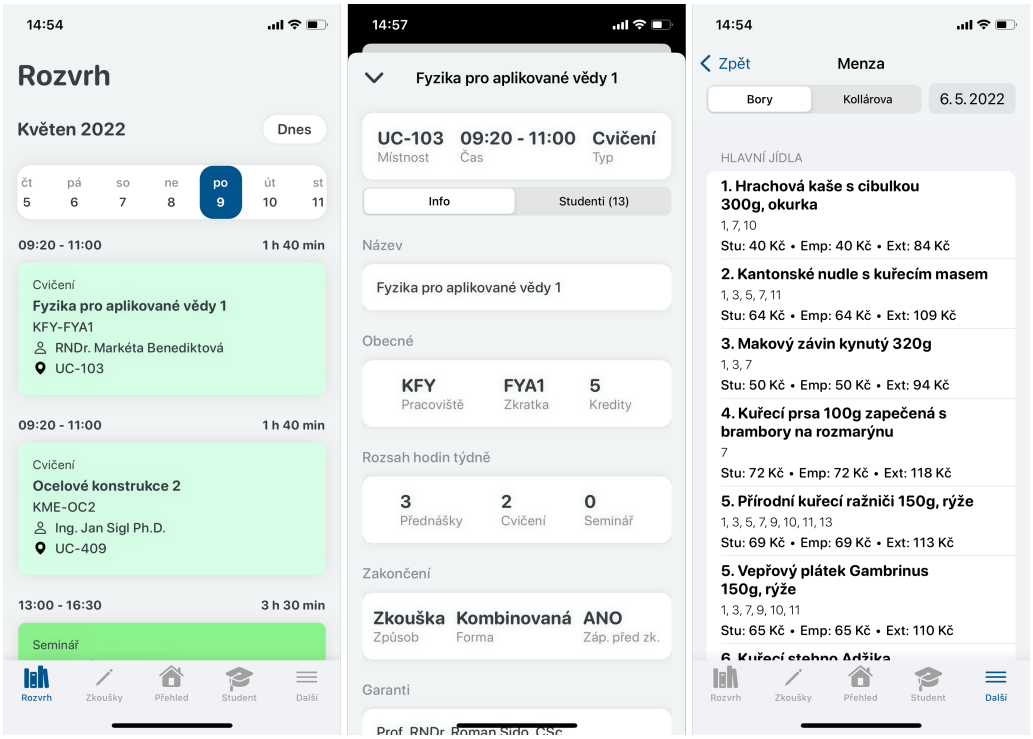
7.9 Navigace aplikace

Základní navigace aplikace (spodní menu) je implementována v souboru *ContentView*. Pro navigaci byla použita komponenta *TabView*, která je součástí *SwiftUI* a dovoluje na základě stavu přepínat obrazovku mezi pohledy potomků. Dle uživatelských rolí uživatele se zobrazují v menu položky. Pro roli učitel jsou viditelné položky: rozvrh, přehled a další. Pro roli student jsou zobrazeny stejné položky a navíc jsou doplněny o zkoušky a profil studenta. Záložka „další“ zobrazuje pohled, který slouží jako další menu, aby zůstalo základní menu přehledné.

Pro zobrazení detailu nějaké informace, jako je například detail rozvrhové akce, je použita komponenta *sheet*. Jedná se o pohled, který je zobrazen přes existující pohled, ale je to modální okno. Dovoluje uživatelům zavřít toto modální okno posunem dolů. *Sheet* je implementován jako metoda (*.sheet()*), která se nastaví na pohled, na kterém se bude zobrazovat. V metodě jsou uvedeny atributy *isPresented* (určuje, jestli je *sheet* viditelný) a *content*, ve kterém je definován obsah, který se nachází v *sheet*.

Další možnou navigace, která se v aplikaci používá, je *NavigationLink*. Tento druh navigace vloží nové *view* na navigační zásobník, čímž dojde k tomu, že nový pohled překryje ten starý. V novém pohledu je horní lišta, která obsahuje tlačítko „zpět“, které zavře nově otevřený pohled. Lišta může také obsahovat další tlačítka nebo název. Tento druh navigace je použit v sekci „další“, například pro zobrazení jídelníčku.

Jednotlivé druhy navigací jsou zobrazeny na obrázku 7.2. Na prvním obrázku zleva se nachází obrazovka se spodním menu. Prostřední náhled zobrazuje použití komponenty *sheet* pro zobrazení detailu rozvrhové akce. Poslední obrazovka ukazuje použití navigace přes *NavigationLink*.



Obrázek 7.2: Ukázka druhů navigací, 1 — základní navigace, 2 — navigace přes *sheet*, 3 — navigace přes *NavigationLink*

8 Další možná rozšíření

V současné době je aplikace v testování a nabízí funkce, které pokryjí většinu nejdůležitějších požadavků pro roli student a učitel. Protože ale IS/STAG poskytuje širokou škálu webových služeb, nabízí se spoustu možností, jak aplikaci dále rozšiřovat. V této kapitole jsou popsány budoucí plány s vývojem aplikace.

8.1 Nové funkce

Z analýzy potřeb uživatelů, popsané v kapitole 5, zbývají k implementaci funkce, na které kvůli časové náročnosti a určitému rozsahu diplomové práce nebyl prostor. Tyto funkce budou analyzovány a v případě podpory webových služeb budou implementovány do mobilní aplikace. Konkrétně se jedná o funkce: vyhledání učeben s rozvrhem, doručení a odeslání školních emailů, vyhledávání učitelů, widget s rozvrhem, vyhledání předmětů s jejich sylabem, hromadný email, přehled termínů a zkoušek pro učitele, vyhledání studentů a zobrazení informací o nich, formuláře kvality výuky, zadávání známek a export rozvrhu do kalendáře.

Při vývoji mobilní aplikace byla objevena spousta nových možností budoucího vývoje. Nejzajímavější z nich se zdá být možnost načtení si uživatele přes NFC a JIS kartu. Tato funkce by byla pro roli učitel a administrativní pracovníky vysoké školy. Po načtení uživatele by se zobrazil seznam akcí, jako například zobrazení dat o studentovi, zapsání známky a další. Aktuálně WS nabízejí operaci *getStudiaByCisloKarty* pro získání informací o studentovi, na základě jeho JIS karty, ze kterých je následně možné získat jeho osobní číslo, které zpřístupňuje další informace o studentovi. Navrhovaný princip užití: po ústním zkoušení by měl vyučující možnost načíst studentovy údaje a zapsat mu tímto způsobem známku.

Dalším plánovaným rozšířením jsou notifikace. Například IS/STAG při vypsání zkouškového období nebo obdržení známky zasílá uživateli upozornění. Notifikace je možné vidět na portále v seznamu notifikací, který je rozbalen po kliknutí na ikonu zvonku v záložce *Moje studium*.

Aplikace byla navržena pro snadná budoucí rozšíření.

8.2 Rozšíření stávajících funkcí

Ačkoli jsou funkce v aplikaci bez problémů, nabízí se možnost k jejich vylepšení. První úprava by se zabývala výpisem kvalifikačních prací pro učitele. Aktuálně se zobrazují všechny kvalifikační práce, u kterých je učitel veden pro současný akademický rok. Bylo by vhodné přidat možnost filtrování, která by usnadnila vyhledávání kvalifikační práce. Operace *getKvalifikacniPrace*, která je použita pro získání kvalifikačních prací, nabízí spoustu parametrů, přes které lze filtrovat výsledky.

Ze zpětné vazby testování bylo zjištěno, že uživatelům chybí klasické zobrazení rozvrhu, jako je na portále. Do obrazovky rozvrhu by bylo příhodné přidat funkci, která by umožňovala přepínat mezi výpisem současným a klasickým.

9 Testování aplikace

Tato část práce je zaměřena na způsoby popisu testování mobilní aplikace. V prvním oddílu kapitoly je popsán test pro ověření korektního zobrazení pro různé rozměry obrazovek. Následuje charakteristika několika uživatelských testů pro ověření funkčnosti mobilní aplikace.

9.1 Ověření funkčnosti a zobrazení na iOS zařízení s různými uhlopříčkami

Pro ověření funkčnosti byly vybrány mobilní telefony s různými typy úhlopříček. Byly vybrány takové modely, které pokrývají nejmenší i největší možnou úhlopříčku.

- *iPhone 6s* — 4,7”
- *iPhone Xs* — 5,7”
- *iPhone 13* — 6,1”
- *iPhone 13 mini* — 5,4”
- *iPhone 13 Pro Max* - 6,7”

Test probíhal pomocí simulátoru zařízení přes *Xcode* s *iOS 15.4*. Testovalo se validní rozložení komponent na obrazovce s vyplněnými daty. Následovalo otestování funkčnosti. V tomto případě se kontrolovalo správné chování komponent. Testovaly se animace přechodů, výběry dat a rolování pohledů. Testování objevilo drobné nedostatky, například v profilu studenta byl příliš malý prostor v rolování komponenty pro zobrazení výsledků, což působilo nepřehledně na malých zařízeních typu *iPhone 6s*. Dalším problémem byly dlouhé texty, které přesahovaly hranice komponent. Tento problém byl vyřešen pomocí metody *truncationMode()*, který dlouhé texty odřízl a konec nahradil třemi tečkami. Veškeré zjištěné nedostatky byly opraveny.

9.2 Testovací scénáře

Předchozí test sloužil především pro otestování správného zobrazení. Tyto testovací scénáře mají za úkol důkladně otestovat funkčnosti mobilní aplikace.

9.2.1 Výběr univerzity

Jako nepřihlášený uživatel klikněte na přihlašovací obrazovce na výběr univerzity. Zobrazí se seznam vysokých škol. Vyberte vysokou školu.

Očekávané chování: Po výběru vysoké školy se změní logo školy na přihlašovací obrazovce. Pokud se jedná o vysokou školu DEMO ZČU, zobrazí se pole pro přihlašovací jméno a heslo, v opačném případě je zobrazeno tlačítko přejít na přihlášení, viz popis v kapitole 7.6.4.

9.2.2 Nesprávné přihlášení

Spusťte aplikaci jako nepřihlášený uživatel. Vyberte si univerzitu a přejděte na externí přihlášení. Zadejte neplatné přihlašovací údaje a odešlete formulář. Zavřete externí přihlášení.

Očekávané chování: Po odeslání neplatných přihlašovacích údajů externí služba zobrazí chybu. Po zavření přihlášení se zobrazí uživateli původní obrazovka s přihlášením.

9.2.3 Korektní přihlášení

Spusťte aplikaci jako nepřihlášený uživatel. Vyberte si univerzitu a přejděte na externí přihlášení. Zadejte platné přihlašovací údaje.

Očekávané chování: Po odeslání platných přihlašovacích údajů Vás externí služba přesměruje na stránku s informací, že dojde k přesměrování zpět do aplikace. Přihlášení se zavře a uživatel je přesměrován na obrazovku „přehled“.

9.2.4 Výpadek sítě

Spusťte aplikaci, přihlaste se a vypněte internetové přihlášení. Proklikejte aplikaci a otestujte, že aplikace neskončí chybou.

Očekávané chování: Ihned po vypnutí internetového připojení se uprostřed obrazovky zobrazí text s informací, že internetové připojení není dostupné. Na obrazovkách se načtou prázdná data, kromě obrazovky „student“ a komponenty „poznámky“, které se načítají z lokálního uložiště.

9.2.5 Ověření správnosti zobrazených dat

Spusťte aplikaci a přihlaste se. Na jiném zařízení se přihlaste pod stejným účtem na webový portál. Projděte jednotlivé obrazovky v mobilní aplikaci a porovnejte data.

Očekávané chování: Data by měla být totožná, protože webový portál a WS IS/STAG vycházejí z jedné databáze. Jediným rozdílem může být zobrazení pouze vybraných parametrů pro mobilní aplikaci.

9.2.6 Test lokalizace

Spusťte aplikaci a přihlaste se. Projděte aplikaci a ověřte, že máte všechny texty ve Vašem jazyce. Přejděte do nastavení a změňte jazyk. Projděte znovu aplikaci a ověřte, že texty jsou zobrazeny v nově zvoleném jazyku.

9.2.7 Přihlášení na zkoušku

Spusťte aplikaci a přihlaste se jako student. Přejděte na obrazovku „zkoušky“. Vyberte si předmět, u kterého je možný zápis. Klikněte na tlačítko „zapsat“. Ověřte na webovém portálu, že jste zapsáni ve zvoleném zkouškovém termínu.

Dodatečné informace: Termín je možné najít na DEMO ZČU, pokud se přihlásíte jako učitel. Následně vyhledejte v prohlížení libovolný zkouškový termín. Vyberte si termín a přejděte na seznam zapsaných studentů. Přes osobní číslo se přihlaste za studenta. Před testováním se odhlaste ze zkouškového termínu.

Očekávané chování: Po stisknutí tlačítka se zobrazí modální okno s informací, zda se povedlo zapsat na vybraný termín. Zkouškový termín se zbarví do žluté a zvýší se počet studentů zapsaných pro vybraný zkouškový termín. Na webovém portálu bude zkouškový termín zapsán.

9.2.8 Odepsání ze zkoušky

Spusťte aplikaci a přihlaste se jako student. Přejděte na obrazovku „zkoušky“. Vyberte si předmět, u kterého je možné se odepsat. Klikněte na tlačítko „odepsat“. Ověřte na webovém portálu, že nejste zapsáni ve zvoleném zkouškovém termínu.

Dodatečné informace: Termín je možné najít na DEMO ZČU, pokud se přihlásíte jako učitel. Následně vyhledejte v prohlížení zkouškový termín. Vyberte si termín a přejděte na seznam zapsaných studentů. Přes osobní číslo se přihlaste za studenta.

Očekávané chování: Po stisknutí tlačítka se zobrazí modální okno s informací, jestli se povedlo odepsat termín. Zkouškový termín zmodrá a sníží se počet studentů na zkouškovém termínu. Na webovém portálu nebude zkouškový termín zapsán a bude možné se na něj přihlásit.

9.2.9 Zobrazení jídelníčku

Spusťte aplikaci a přihlaste se. Přejděte do zobrazení *další* a přejděte do menzy. Zvolte pracovní den, kdy je otevřená menza, zvolte Bory a nechte si zobrazit data. Porovnejte výsledky s daty na adrese www.zcu.cz/cs/Canteen, pokud jste zvolili jiný než aktuální den, bude nutné upravit výběr dne i na webové adrese.

Očekávané chování: Mobilní aplikace bude obsahovat stejná data, která jsem uvedena na webové stránce.

9.3 Výsledky testů

Testování podle scénářů odhalilo drobné *UI* nedostatky nebo chyby v překladu aplikace. Všechny byly opraveny. Co se týče samotného chování aplikace, nebyl nalezen žádný problém.

Testování probíhalo v kombinaci s účtem autora na produkčním serveru IS/STAG ZČU a s DEMO účtem na DEMO ZČU pro otestování role učitel a dalších účtů s rolí student.

9.3.1 Profilování aplikace

Spolu s testováním bylo provedeno profilování aplikace. Pro profilování byl použit integrovaný profiler v *Xcode*. Pro testování byl použit jeden z demo účtů studenta na STAG demo.

Dle naměřených výsledků aplikace zabírá okolo 20 MB RAM zařízení, CPU bylo v jednotkách procent zátěže. Profiler při průchodu aplikací, která zahrnovala: přihlášení a zobrazení rozvrhu, zkoušek, statistik studenta, jídelníčku a nastavení uvádí čtení z disku 6,1 MB, zápis na disk 4,3 MB a datový

přenos (internet) 50 KB. Vyšší hodnoty u čtení a zápisu na disk jsou způsobeny interními procesy aplikace, kdy si aplikace sama ukládá a načítá různou *cache*.

10 Závěr

V rámci diplomové práce byla realizována nativní mobilní aplikace pro operační systém *iOS*, která umožňuje komunikaci s webovými službami IS/STAG pro role student a učitel. Nejdříve byly prozkoumány existující mobilní aplikace, které aktuálně fungují na webových službách IS/STAG. Výsledkem zkoumání bylo nalezení společných vlastností, které by měla navrhovaná mobilní aplikace obsahovat. Na základě těchto poznatků byl vytvořen dotazník pro získání širšího povědomí o tom, jak uživatelé přemýšlejí o tomto typu mobilní aplikace. Respondenty byly poskytnuty doplňující informace o aplikaci — co má aplikace obsahovat, případná další vylepšení, která dotazovaným ve stávajících aplikacích chyběla. Po získání dostatečného počtu dat proběhla analýza, na jejímž základě byl zvolen návrh mobilní aplikace.

Řešení bylo implementováno pomocí jazyka *Swift* a frameworku *SwiftUI* pro uživatelské rozhraní. Persistenci dat zajišťovaly úložiště *UserDefaults*, *Keychain* a framework *Core Data*.

Pozornost byla věnována též testování celkové mobilní aplikace. Z povahy aplikace probíhaly testy podle scénářů. Bylo otestováno jednak správné zobrazení na různých rozměrech zařízení, jednak správné chování, bez nebo s měnícími se podmínkami internetového připojení. Funkčnost aplikace byla důkladně otestována uživatelskými scénáři.

Navržená aplikace se od těch stávajících liší tím, že umožňuje komunikaci s webovými službami IS/STAG jak v roli student, tak v roli učitel a zároveň je určena pro zařízení s operačním systémem *iOS*. Prostřednictvím vcelku detailních dotazníků bylo úspěšně získáno dostatek dat, které přispěly k pokrytí všech základních požadavků potencionálních uživatelů.

Protože je IS/STAG velmi velkým systémem, který umožňuje mnoho způsobů, jakými je možné vést budoucí vývoj mobilní aplikace, jsou v samostatné kapitole popsány další plány s nadcházejícím směřováním vývoje mobilní aplikace.

Jak již bylo zmíněno, aplikace pro komunikaci s webovými službami IS/STAG pro operační systém *iOS* byla navržena a nasazena do *beta* testování. K testování byli pozvání studenti ZČU prostřednictvím sociálních sítí, zadání i cíl diplomové práce tedy považuji za splněné.

Seznam použitých zkratek

API Application Programming Interface

CIV Centrum informatizace a výpočetní techniky

HIG Human Interface Guidelines

IS/STAG Informační systém studijní agendy

JSON JavaScript Object Notation

MVC Model-view-controller

MVVM Model–view–viewmodel

NFC Near Field Communication

UI User Intefrace

UX User Experience

WS Webová služba/webové služby

XML Extensible Markup Language

ZČU Západočeská univerzita v Plzni

Literatura

[1] Abdulrasool, S. *Introducing Swift on Windows* [online]. 2020. [cit. 2022/04/20]. Dostupné z: https://www.swift.org/blog/swift-on-windows/.

[2] Boudreaux, T. *Programming the iPhone User Experience*. O'Reilly Media, Inc, 2009. ISBN 9780596155469.

[3] checkra1n. *Jailbreak for iPhone 5s through iPhone X, iOS 12.0 and up* [online]. 2022. [cit. 2022/05/02]. Latest Release. Dostupné z: https://checkra.in.

[4] Holland, P. *The iPhone at 15: Steve Jobs revealed his greatest product in 2007* [online]. 2022. [cit. 2022/04/10]. Dostupné z: https://www.cnet.com/tech/mobile/the-iphone-at-15-steve-jobs-revealed-his-great-product-15-years-ago/.

[5] Hudson, P. *All SwiftUI property wrappers explained and compared* [online]. 2021. [cit. 2022/04/25]. Dostupné z: https://www.hackingwithswift.com/quick-start/swiftui/all-swiftui-property-wrappers-explained-and-compared.

[6] Inc., A. *What Is Cocoa?* [online]. 2013. [cit. 2022/04/25]. Dostupné z: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html.

[7] Inc., A. *Core Data* [online]. 2022. [cit. 2022/04/21]. Dostupné z: https://developer.apple.com/documentation/coredata.

[8] Inc., A. *FileManager* [online]. 2022. [cit. 2022/04/21]. Dostupné z: https://developer.apple.com/documentation/foundation/filemanager/.

[9] Inc., A. *iOS Design Themes* [online]. 2022. [cit. 2022/05/08]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/.

[10] Inc., A. *App Store* [online]. 2022. [cit. 2022/04/11]. iOS and iPadOS usage. Dostupné z: https://developer.apple.com/support/app-store/.

[11] Inc., A. *Keychain data protection* [online]. 2022. [cit. 2022/04/21]. Dostupné z: https://support.apple.com/cs-cz/guide/security/secb0694df1a/web.

[12] Inc., A. *Using the Keychain to Manage User Secrets* [online]. 2022. [cit. 2022/04/21]. Dostupné z: https://developer.apple.com/documentation/security/keychain_services/keychain_items/using_the_keychain_to_manage_user_secrets.

[13] Inc., A. *Choosing a Membership* [online]. 2022. [cit. 2022/05/08]. Dostupné z: https://developer.apple.com/support/compare-memberships/.

[14] Inc., A. *Apple Developer Enterprise Program* [online]. 2022. [cit. 2022/05/08]. Dostupné z: https://developer.apple.com/programs/enterprise/.

[15] Inc., A. *State and Data Flow* [online]. 2022. [cit. 2022/04/25]. Dostupné z: https://developer.apple.com/documentation/swiftui/state-and-data-flow.

[16] Inc., A. *SwiftUI* [online]. 2022. [cit. 2022/04/25]. Dostupné z: https://developer.apple.com/xcode/swiftui/.

[17] Inc., A. *About App Development with UIKit* [online]. 2022. [cit. 2022/05/08]. Dostupné z: https://developer.apple.com/documentation/uikit/about_app_development_with_uikit.

[18] Inc., A. *UserDefaults* [online]. 2022. [cit. 2022/04/20]. Dostupné z: https://developer.apple.com/documentation/foundation/userdefaults.

[19] JavaTpoint. *Android Versions* [online]. 2021. [cit. 2022/03/05]. Dostupné z: https://www.javatpoint.com/android-versions.

[20] Jiroušek, I. P. *Informační systém studijní agendy - IS/STAG* [online]. 2022. [cit. 2022/04/05]. Dostupné z: https://is-stag.zcu.cz/zakaznici/propagacni-materialy/stagMat-deskyPDF-allInOne.pdf.

[21] Meta platforms, I. *Who's using React Native?* [online]. 2022. [cit. 2022/05/02]. Dostupné z: https://reactnative.dev/showcase.

[22] Neuburg, M. *iOS 15 Programming Fundamentals with Swift*. O'Reilly Media, Inc, 2021. ISBN 1098118502.

[23] Nguyen, K. A. *Top 7 best native app example in 2022 that merchants can learn from* [online]. 2021. [cit. 2022/05/02]. Dostupné z: https://magenest.com/en/native-app-example/.

[24] PhonesData. *Apple iPhone 6s specs, review, options, comparasions* [online]. 2022. [cit. 2022/04/10]. Dostupné z: https://phonesdata.com/en/smartphones/apple/iphone-6s-3254/.

[25] Pietro Rea, K. R. *5. Internal Distribution* [online]. 2022. [cit. 2022/05/08]. Dostupné z: https://www.raywenderlich.com/books/ios-app-distribution-best-practices/v1.0/chapters/5-internal-distribution.

[26] Schultz, M. *iOS 5 To Be Released on October 12* [online]. 2011. [cit. 2022/04/25]. Dostupné z: https://www.macrumors.com/2011/10/04/ios-5-to-be-released-on-october-12/.

[27] SQLite. *About SQLite* [online]. 2022. [cit. 2022/04/21]. Dostupné z: https://sqlite.org/about.html.

[28] StatCounter. *Mobile & Tablet Android Version Market Share Worldwide* [online]. 2022. [cit. 2022/03/05]. Dostupné z: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide.

[29] StatCounter. *Mobile Operating System Market Share Worldwide* [online]. 2022. [cit. 2022/4/10]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[30] TechCrunch. *Google's brand new Android 12 operating system launches today* [online]. 2021. [cit. 2022/03/05]. Dostupné z: https://techcrunch.com/2021/10/19/android-12-launched/.

[31] TechRadat. *iOS 15 latest features, and what will change on your iPhone* [online]. 2022. [cit. 2022/04/11]. Dostupné z: https://www.techradar.com/news/ios-15-update.

Přílohy

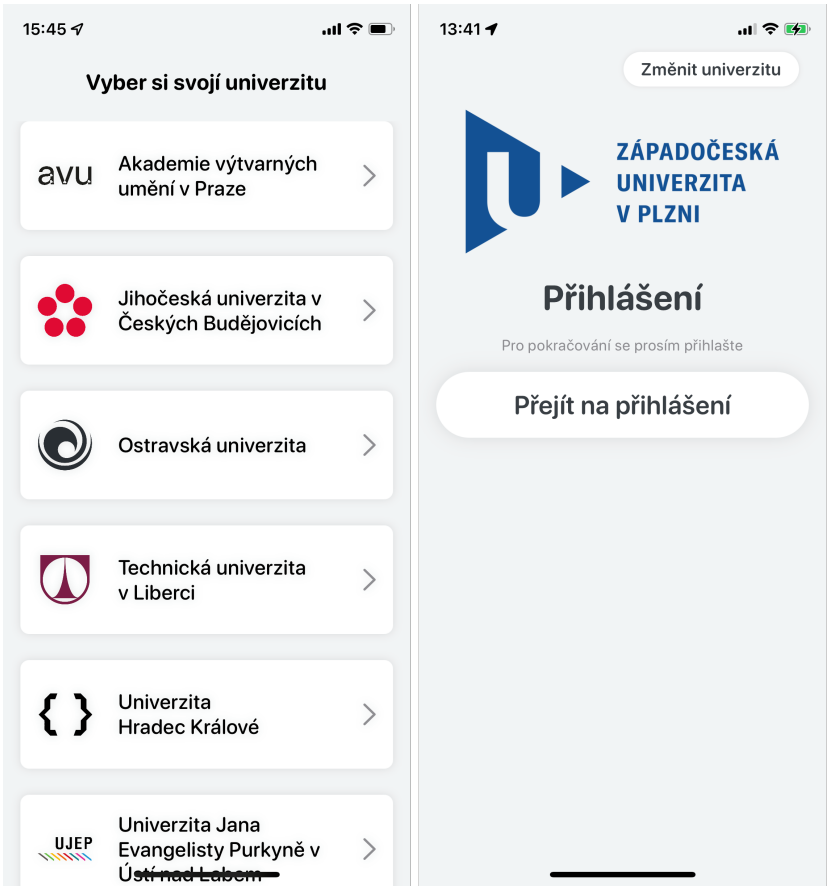
A Instalace

Aplikace se nyní nachází v *beta* testování. Z toho důvodu ji není možné nainstalovat z *App Store*, ale je distribuována přes *TestFlight*. *TestFlight* je oficiální metoda pro distribuci *beta* aplikací, která slouží pro získání zpětné vazby od veřejnosti před jejím oficiálním vydáním do *App Store*.

Pro instalaci aplikace je nutné mít v telefonu operační systém minimálně *iOS 15*. Dále je nutné stáhnout z *App Store* aplikaci *TestFlight*. Pro přihlášení do testování aplikace je nutné na mobilním telefonu otevřít adresu https://testflight.apple.com/join/8swnLNzg, která vás zařadí do testování aplikace. Po zařazení do testování se aplikace objeví v *TestFlight*. Následně je možné aplikaci nainstalovat. V blízké době bude aplikace vydána v *App Store*.

B Uživatelská příručka

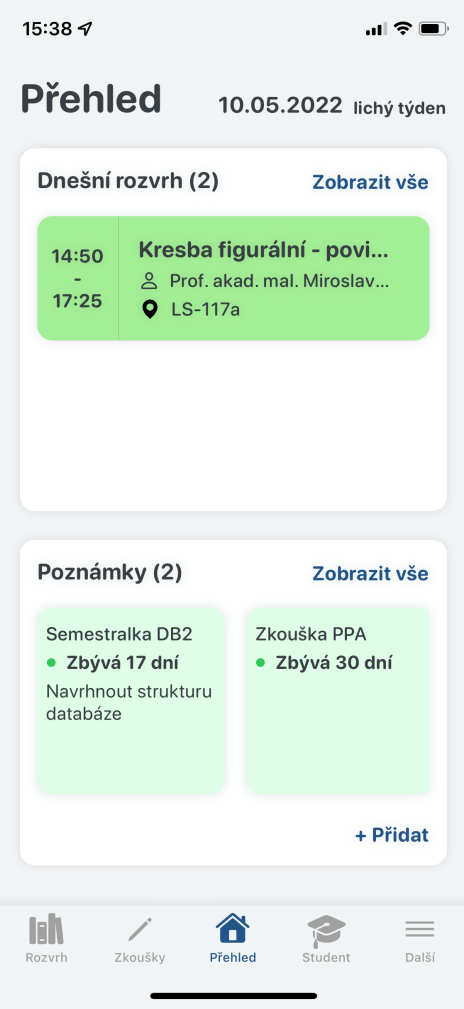
Po prvotním spuštění aplikace je zobrazen uživateli výběr vysoké školy, na kterou je možné se přihlásit. Po zvolení univerzity je zobrazena uživateli přihlašovací obrazovka, viz obrázek B.1. Logo na přihlašovací obrazovce se může lišit na základě zvolené vysoké školy.



Obrázek B.1: Náhled přihlášení a výběru vysoké školy

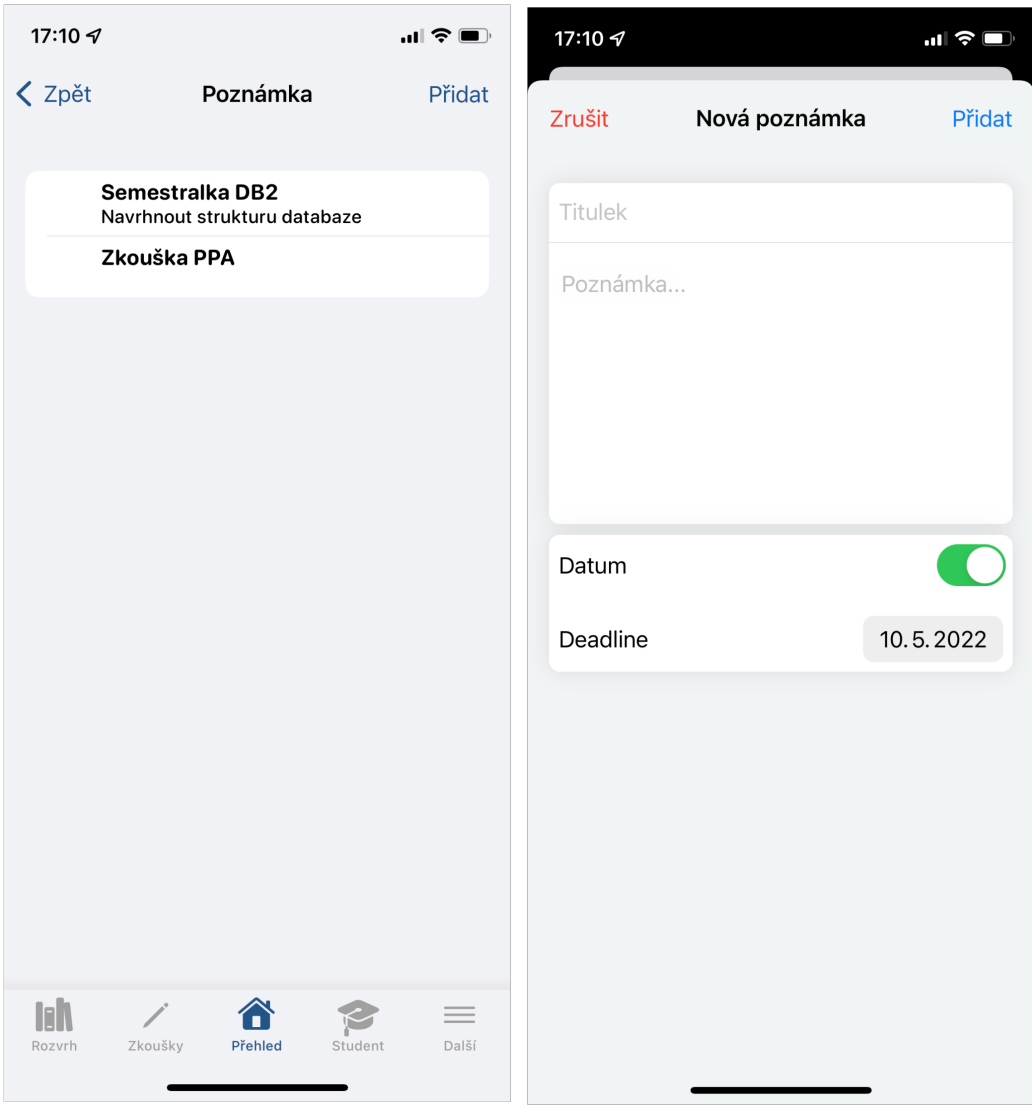
Po stisknutí tlačítka „Přejít na přihlášení“ je otevřen prohlížeč pro přihlášení uživatele. Externí přihlášení se liší na základě zvolené univerzity. Po úspěšném přihlášení je zobrazena výchozí obrazovka aplikace, a to „Přehled“. Ukázka obrazovky je zobrazena na obrázku B.2.

Přehled zobrazuje dnešní datum s označením lichého nebo sudého týdne. Dále máme na obrazovce komponentu „dnešní rozvrh“, která ukazuje aktuálně probíhající nebo budoucí vyučovací hodinu v rozvrhu. Komponenta „poznámky“ umožňuje uživateli zanést vlastní poznámky do aplikace. U jednotlivých poznámek je možné nastavit mezní termín, který aplikace automaticky přepočítává do zbývajících dnů.



Obrázek B.2: Náhled přehledu

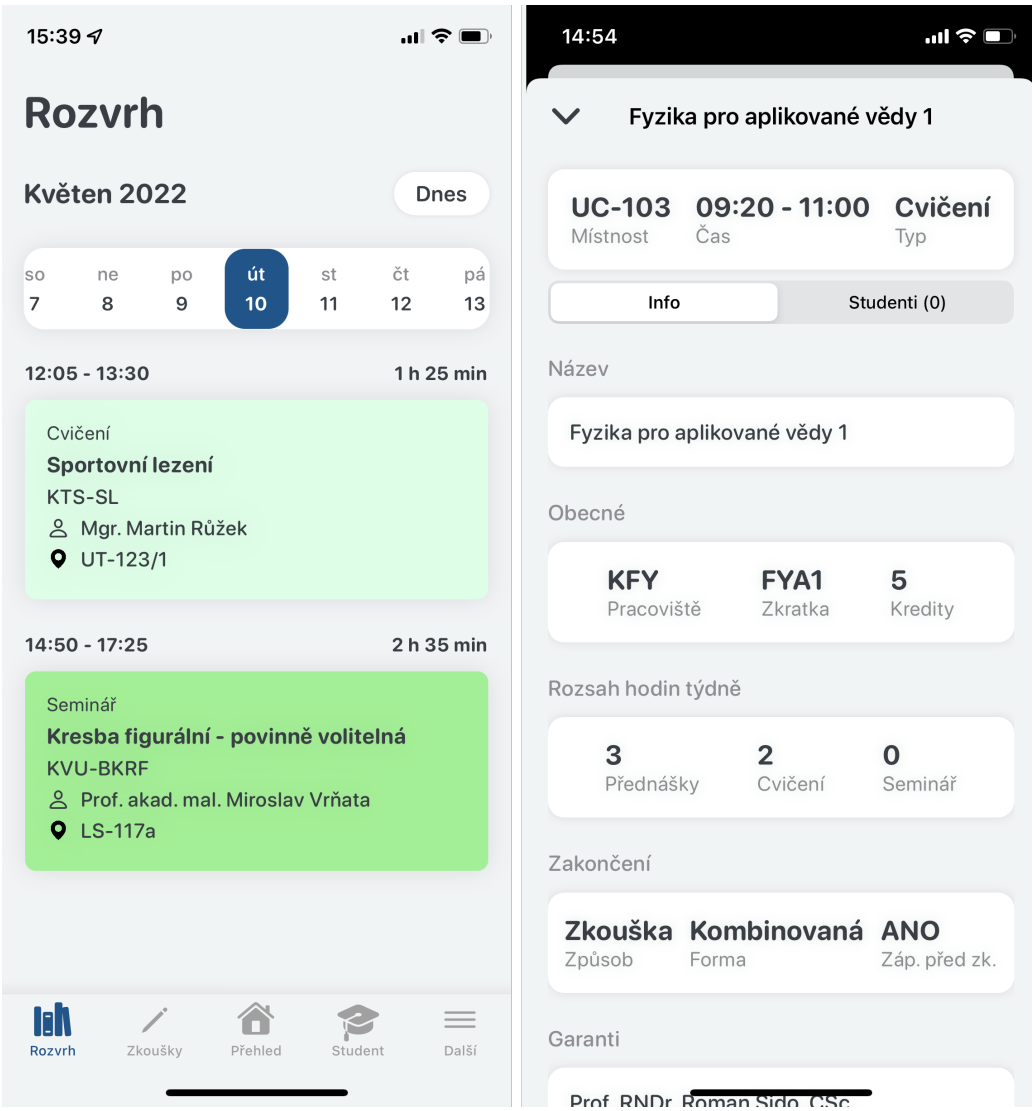
Výpis všech poznámek je možné zobrazit přes tlačítko „Zobrazit vše“ v komponentě poznámek. Jedná se o jednoduchý výpis se všemi poznámkami. Poznámku lze smazat přetažením řádku doleva. Nové poznámky přidáme buď z komponenty poznámky nebo výpisu poznámek přes tlačítko „Přidat“. Ukázka obrazovky s výpisem poznámek a přidáním nové poznámky je zobrazena na obrázku B.3.



Obrázek B.3: Náhled výpisu a přidání poznámek

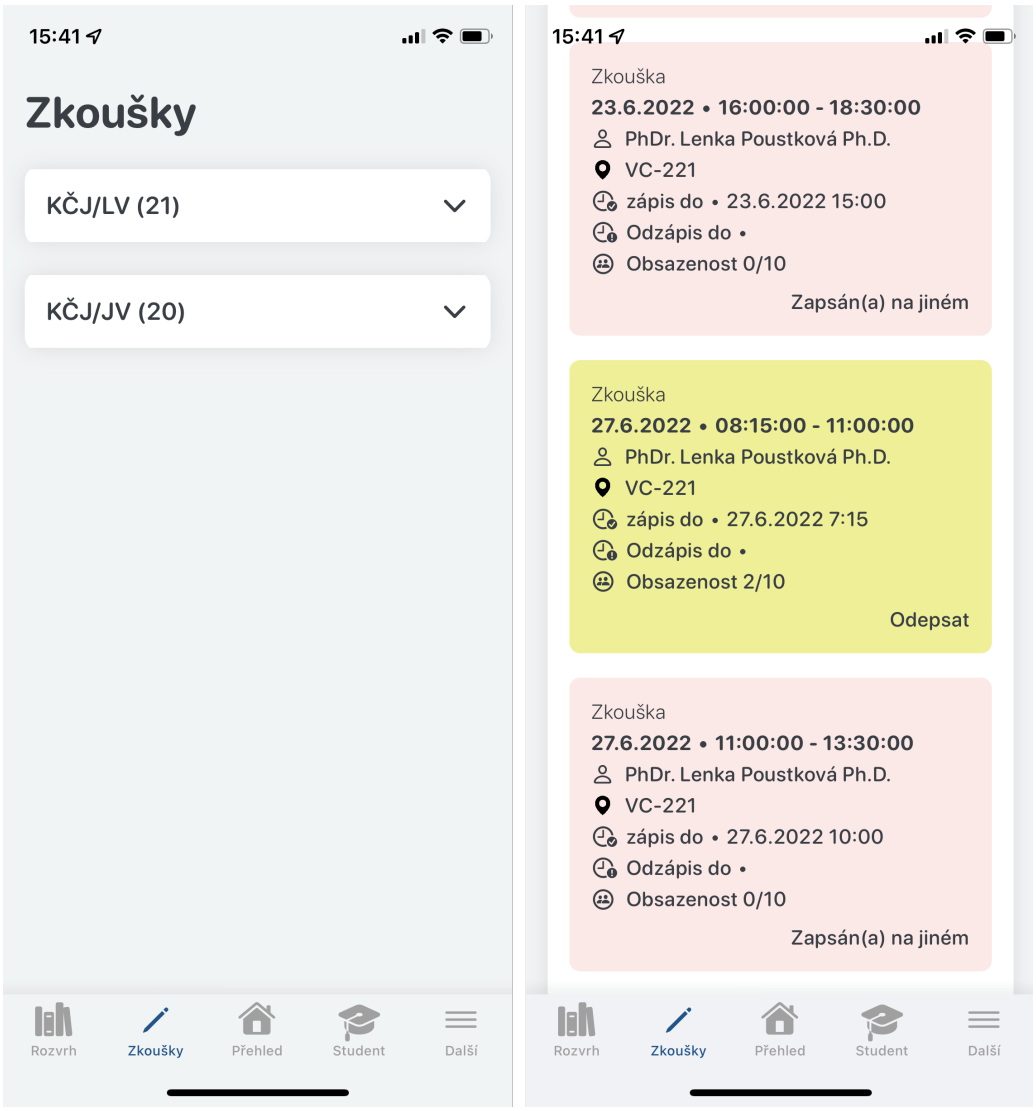
Ve spodní části obrazovky je menu, ve kterém je zvýrazněná aktuálně otevřená obrazovka. Mezi obrazovkami je možné přepínat kliknutím na ikonu.

Pokud je zvolena možnost „Rozvrh“, je uživateli zobrazen jeho rozvrh s rozvrhovými akcemi, který je možný vidět na obrázku B.4. Po kliknutí na rozvrhovou akci je zobrazen její detail s dodatečnými informacemi. Detail je možný zavřít šipkou v horním menu nebo přejetím prstem dolů.



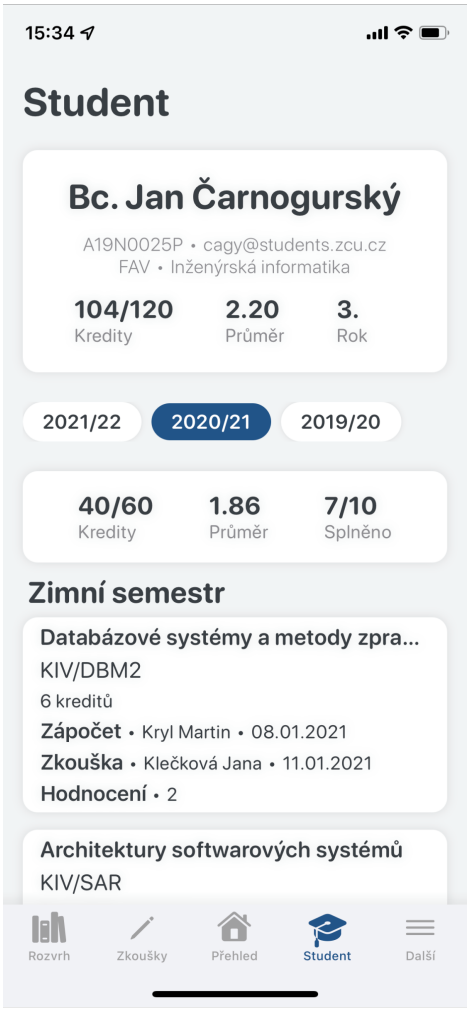
Obrázek B.4: Náhled rozvrhu s detailem rozvrhové akce

Při zvolení obrazovky „Zkouška“ jsou studentovi zobrazeny jeho zkouškové termíny. Termíny jsou seskupeny podle předmětu. U každého předmětu je číslo představující počet vypsaných termínů. Termíny, na které je možné se zapsat, jsou zbarveny modře, předměty, na které se nelze přihlásit, jsou červené. Zapsané termíny jsou zobrazeny žlutě. Pokud není možné se zapsat na termín, je u termínu uveden důvod. Náhled je zobrazen na obrázku B.5.



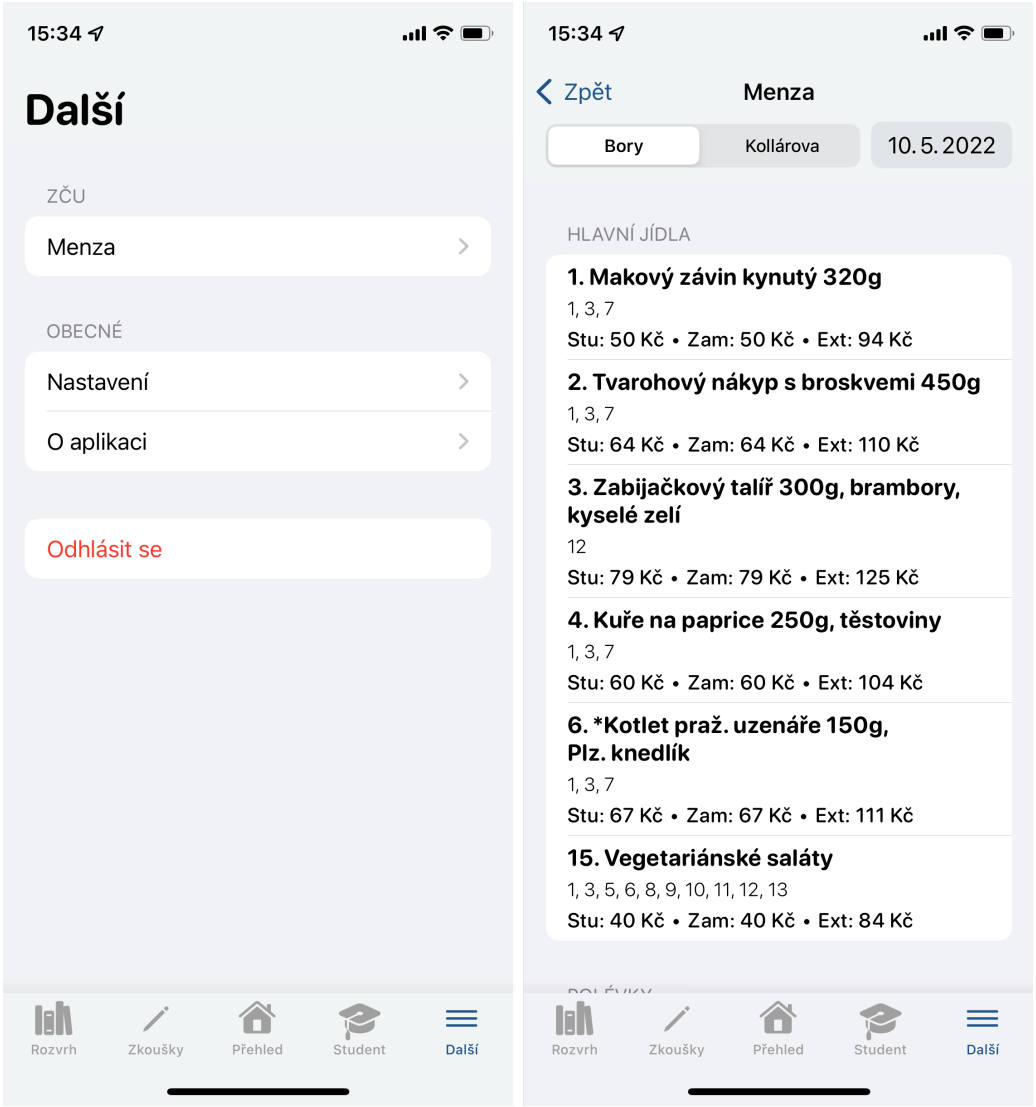
Obrázek B.5: Náhled zápisu zkoušek

Pokud je přihlášen student, může si zobrazit svůj průběh studia na obrazovce „Student“. V horní části obrazovky jsou zobrazeny údaje o studentovi. Na obrazovce se počítají nejen celkové statistiky studia, ale i údaje jednotlivých studijních let. Předměty jsou seskupeny podle akademického roku. Ukázku je možné vidět na obrázku B.6.



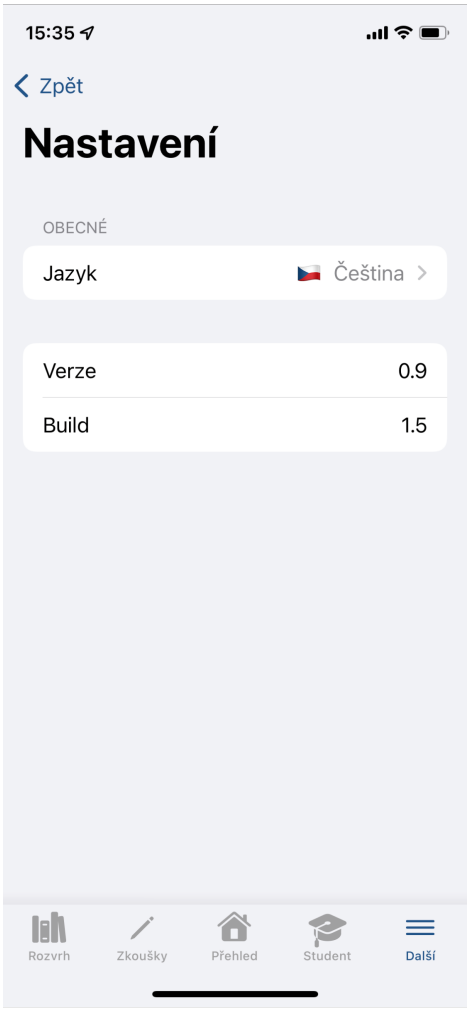
Obrázek B.6: Náhled statistik studenta

Sekce „Další“ představuje formu navigace. Jako první je v menu položka „Menza“. Tato volba je viditelná, pouze pokud je uživatel ze ZČU a jedná se o jídelníček v menze. V jídelníčku je možné volit menzu na Borech a v ulici Kollárova. Dále je možné zvolit datum, pro který se má jídelníček zobrazit. Ukázka je na obrázku B.7.



Obrázek B.7: Náhled pohledu *Další* a jídelníčku

Další položkou je nastavení aplikace. V nastavení je možné vybrat jazyk aplikace. Zároveň je zobrazeno číslo verze a *buildu*, viz obrázek B.8.



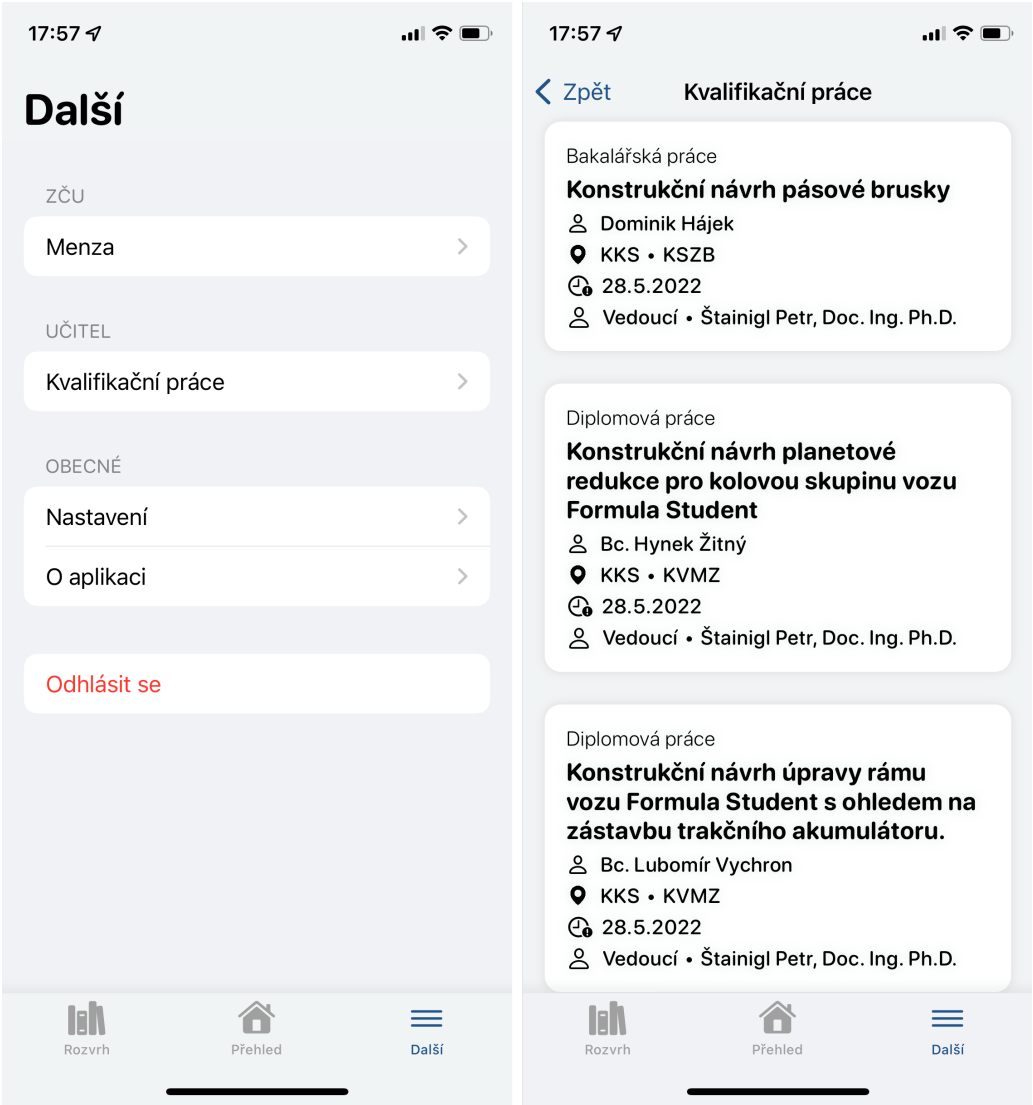
Obrázek B.8: Náhled nastavení

Položka „O aplikaci“ obsahuje popis aplikace. Konkrétně vysvětluje, proč a jak tato mobilní aplikace vznikla. Dále obsahuje kontakt na autora mobilní aplikace. Ukázka je zobrazena na obrázku B.9.

Poslední položkou v menu je funkce „Odhlásit se“, která odhlásí přihlášeného uživatele. Po odhlášení je zobrazena uživateli přihlašovací obrazovka.

Obrázek B.9: Náhled informací o aplikaci

Na obrázku B.10, je vidět prostředí, pokud je přihlášen uživatel pouze s rolí učitel. Ze spodního navigačního menu zmizely položky „Zkoušky“ a „Student“. V sekci „Další“ naopak přibyla sekce učitel, která obsahuje výpis kvalifikačních prací, u kterých je učitel veden.



Obrázek B.10: Náhled obrazovky pro učitele a výpisu kvalifikačních prací

C Obsah ZIP

Obsah souboru ZIP je popsán v následujícím seznamu:

- `Text_prace` — složka obsahující text diplomové práce, obrázky a zdrojové kódy textu v LaTeXu.
- `Poster` — adresář obsahující výsledný poster ve formátu `pdf` a `pub.`
- `Aplikace_a_knihovny` — složka obsahující zdrojové kódy, dokumentaci aplikace, výslednou sestavenou aplikaci a návod na sestavení aplikace.
- `Readme.txt` — textový soubor popisující obsah DVD souboru.