VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MOBILNÍ APLIKACE PRO ZAMĚSTNANCE VUT S PODPOROU BIOMETRIE

MOBILE APPLICATION FOR EMPLOYEES WITH BIOMETRICS SUPPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ALENA TESAŘOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Prof. Ing. TOMÁŠ HRUŠKA, CSc.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání diplomové práce

Studentka: **Tesařová Alena, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Bioinformatika a biocomputing
Název: **Mobilní aplikace pro zaměstnance VUT s podporou biometrie**
Mobile Application for Employees with Biometrics Support
Kategorie: Uživatelská rozhraní
Zadání:

1. Seznamte se s technologií React Native pro tvorbu mobilních aplikací.
2. Analyzujte nejpoužívanější části Informačního systému pro zaměstnance napříč fakultami VUT.
3. Navrhněte uživatelské rozhraní pro mobilní telefony pro vybrané nejpoužívanější části IS VUT.
4. Navrhněte možné použití biometrických technologií, například bezpečné přihlašování, podepisování a schvalování pomocí otisku prstu.
5. Implementujte vybrané moduly mobilní aplikace pro zaměstnance.
6. Proveďte uživatelské testování aplikace na omezené skupině zaměstnanců VUT a na základě výsledků doporučte další rozvoj aplikace.
7. Vytvořte plakát nebo video prezentující práci s aplikací.

Literatura:

- Bonnie Eisenman, Learning React Native: Building Native Mobile Apps with JavaScript, ISBN-13: 978-1491929001
- Tony Buzan, Mobile App Development with Ionic, revised edition (anglicky), ISBN: 9781491998120
- Dále dle pokynu vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hruška Tomáš, prof. Ing., CSc.**
Konzultant: Marušinec Jaromír, Ing., Ph.D., CVIS VUT
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 22. října 2020

Abstrakt

Tato práce se zabývá návrhem a realizací mobilní aplikace pro zaměstnance VUT v Brně. Výsledná aplikace je psaná pro platformy Android a iOS v jazyku React Native. Jejím cílem je ulehčení každodenních činností zaměstnance, které vede k zefektivněni práce a snížení administrativy. Teoretická část se věnuje výběru nejdůležitějších modulů z informačního systému na základě statistických dat a dotazníků. Realizační část obsahuje návrh uživatelského rozhraní vybraných modulů a popis implementace aplikace. Výsledná aplikace byla otestována skupinou zaměstnanců a získala velmi dobré slovní hodnocení.

Abstract

This thesis deals with the design and implementation of a mobile application for BUT employees. The final application is written for Android and iOS in React Native. The goal is to facilitate the daily activities of an employee, which leads to more efficient work and reduced administration. The theoretical part deals with the selection of the most important modules in the information system based on statistical data and questionnaires. The implementation part contains the design of the user interface of selected modules and a description of the application implementation. The final application was tested by a group of employees and received a very good verbal evaluation.

Klíčová slova

Android, iOS, React Native, mobilní aplikace, biometrie, TypeScript, rozvrhy, informační systém, uživatelské rozhraní

Keywords

Android, iOS, React Native, mobile application, biometry, TypeScript, schedules, information system, user experience

Citace

TESAŘOVÁ, Alena. *Mobilní aplikace pro zaměstnance VUT s podporou biometrie*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing. Tomáš Hruška, CSc.

Mobilní aplikace pro zaměstnance VUT s podporou biometrie

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením profesora Ing. Tomáše Hrušky, CSc. Další informace mi poskytli Ing. Rudolf Musil a Ing. Jaromír Marušinec Ph.D., MBA. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Alena Tesařová
17. května 2021

Poděkování

Děkuji vedoucímu diplomové práce Ing. Tomáši Hruškovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále bych chtěla poděkovat řediteli CVIS Ing. Jaromírovi Marušincovi za celkovou podporu, Ing. Rudolfovi Musilovi za pomoc při řešení technických problémů a dalším kolegům ze CVIS za konzultaci během řešení a poskytování zpětné vazby.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Život bez mobilních aplikací si již málokdo dovede představit. Zjednodušují denní rutiny a zpříjemňují uživatelům pracovní i osobní život. O mobilní aplikace se zajímají hlavně velké firmy, které si chtějí posunout byznys na vyšší úroveň. Aplikace jim umožňuje jednodušší a bližší komunikaci se zákazníkem. V případě vysokých škol může mimo jiné ulehčit komunikaci mezi studenty a vyučujícími, čímž může přispět ke zkvalitnění výuky nebo ulehčení všedních úkonů zaměstnance jako je například správa docházky a nepřítomnosti.

VUT používá více informačních systémů. Často zaměstnanci nemají řádné školení a nemusí si vždy vědět rady, jak se dostat ke konkrétním funkcím a modulům, které potřebují. Více informačních systému znamená i horší orientaci a roztříštěnost. Zaměstnanci často pracují v terénu, kde nemají u sebe stolní počítač a jsou pak plně závislí na mobilním zařízení. Přístup k jistým modulům navíc nemusí být na telefonu možný nebo nemají moduly responzivní uživatelské prostředí. Zaměstnanci pak nezbývá nic jiného než vzít tužku a papír a doplnit údaje do informačního systému (IS) později na počítači. Zadávání jim pak vezme čas, který mohli strávit efektivněji.

Zaměstnanci vykonávají různé funkce na VUT. Jedná se o učitele, externisty, akademické a technicko-hospodářské pracovníky. Každý používá IS jiným způsobem, pro každého jsou důležité jiné moduly.

Tato práce se zabývá vybráním nejdůležitějších modulů k následnému vytvoření návrhu mobilní aplikace pro zařízení Android a iOS ulehčující práci zaměstnance. Výběr byl proveden na základě kvantitativních dat z IS a na základě dotazníku určený zaměstnancům. Aplikace bude sloužit všem zaměstnancům se zaměřením především na učitele všech fakult VUT využívající centrální informační systém. Tento systém je spravovaný součástí VUT, Centrem Výpočetních a Informačních služeb (CVIS), pro kterou je tato práce vytvářena. Smyslem práce je tedy vytvořit aplikaci, která bude komunikovat se školským informačním systémem a poskytovat zaměstnanci aktuální informace.

V první kapitole jsou představeny různé technologie pro vývoj mobilních aplikací se zaměřením na framework React Native. Druhá kapitola zpracovává analýzu současného stavu, kde jsou popsány vybrané moduly IS. Kapitola 4 se zabývá návrhem uživatelského rozhraní mobilní aplikace, kde je upřesněna funkcionalita s ohledem na nejpoužívanější moduly. Popis implementace vybraných modulů společně s technickými detaily je obsažen v kapitole 5. V kapitole 6.2 je popsáno testování na vybrané skupině zaměstnanců a na základě výsledků je na konci kapitoly navržen budoucí vývoj aplikace.

Kapitola 2

Vývoj mobilních aplikací

Uživatelé mobilních zařízení často preferují používání aplikací instalovaných na jejich mobilních zařízeních. Používají je pro jejich denní rutiny a zlehčují jim práci všedního života. Může se jednat například o rezervování lístků, kupování lístků do kina nebo sledování videí. Hlavními zástupci na trhu s mobilními aplikacemi jsou Android, iOS a Windows Phone. Kvůli rozdílům v operačních systémech je třeba vývoj cílit na jednu platformu, což je drahé a často takový vývoj stojí mnoho úsilí. Aplikace pro Android je třeba psát v Javě, pro iOS se jedná o Objective-C a jazyk .NET pro Windows Phone. Takový vývoj může stát pak spoustu času, vyžaduje různé experty pro konkrétní programovací jazyk a vývoj se pak může velmi prodražit. Firma si proto před vývojem musí pořádně rozmyslet cíle, rozpočet a zhodnotit, jestli se jí skutečně takové řešení vyplatí. Takové rozhodnutí může být obtížné a nemělo by se uspěchat. Jedná o výbornou příležitost, jak zlepšit interakci se zákazníkem, prohloubit povědomost značky nebo dokonce získat příjem navíc. Na druhou stranu pokud jsou cíle aplikace nejisté, zákazníci budou nespokojení a firma může přijít o prestiž a investované peníze. [26, 25]

V následujících kapitolách budou postupně představeny základní typy přístupů k vývoji mobilních aplikací. Základní rozdělení je na aplikace:

- webový,
- nativní,
- hybridní.

Ke každému typu budou uvedeny výhody a nevýhody současně s alespoň jedním příkladem konkrétní technologie. Konec kapitoly se bude zabývat zabezpečením mobilních aplikací se zaměřením na biometrii.

2.1 Webový vývoj

Mobilní webové aplikace jsou webové stránky, které jsou zobrazené v mobilním prohlížeči. Právě kvůli velikosti obrazovky mobilního zařízení nemusí být některé webové stránky správně zobrazeny. Optimalizované stránky pro mobilní zařízení se nazývají responzivní. Platí pro ně, že se obsah stránky zobrazí podle tvaru a velikosti jeho kontejneru. Sloupce, bloky textů, obrázky a další komponenty se adaptují šířce displeje anebo dokonce mohou úplně zmizet. [10]

Pro vývoj se používají klasické webové technologie jako je HTML5, CSS3 a JavaScript, které dovolí vývojáři snadný, levný a rychlý vývoj. Tyto aplikace mohou být použity mezi

všemi platformami na rozdíl od nativních aplikací. Další výhody a nevýhody shrnuje následující seznam:

Výhody [3]

1. Vývoj je levný a rychlý díky známým technologiím.
2. Nasazení je možné na více platformách.
3. Aplikace jsou velmi dobře udržovatelné.
4. Možný přístup k nativním funkcím přes řešení třetích stran (např. *PhoneGap*).

Nevýhody

1. Výkon může být překážka v případě, kdy běží více webových stránek v mobilním prohlížeči.
2. Ladění a testování je obtížnější na telefonu než na webu.
3. Těžší generování zisku. Neexistuje žádný App Store, který by se staral o prodej.
4. Vyžaduje nutně internetové připojení.

Oblast testování a ladění se může zdát jako maličkost, ale ve skutečnosti představuje velký problém. Testování na počítači může rychle vést k opravě chyb, ale testování v mobilních emulátorech nenabízí tak přesné výsledky, co se týká výkonnosti na mobilním zařízení. Efekty jako jsou animace u navigací mohou zvýšit přehlednost a uživatelský zážitek působit, jenže vyžaduji dost paměti. Proto na mobilních zařízeních může dojít k vyčerpání paměti a stránka se nemusí správně načíst. [3]

2.1.1 React

React je open source, frontendová knihovna vytvořená firmou Facebook v roce 2013 a používá se k tvorbě interaktivní webových stránek [34]. React je založený na tom, že každá aplikace se skládá z částí nazývaných komponenty, které mohou být znovu použity. Komponenty Reactu jsou nejčastěji vykreslovány pomocí JSX (JavaScript eXtension), který umožňuje psát HTML kód v JavaScriptu. Samotná knihovna React avšak nestačí k vykreslení komponenty do prohlížeče. Tvůrci Reactu avšak vytvořili balíček React DOM přesně pro tento účel [13, 21]. Existují dva způsoby, jak vyrobit aplikaci složenou z komponent:

- zdola nahoru – vytvořit menší komponenty, ze kterých se budou skládat komplexnější komponenty,
- shora dolů – začít od hlavní (top) komponenty a postupně implementovat její následníky.

JSX

JSX je syntaktické rozšíření JavaScriptu, které se nejvíce používá v knihovně React pro popis uživatelského rozhraní. Jeho použití avšak není podmínkou (stále je možné psát aplikaci v čistém JavaScriptu). React je založen na tom, že logika aplikace je velmi úzce spjata s logikou reagování na vstupy uživatelů anebo změně stavu aplikace. Snaží se tímto způsobem

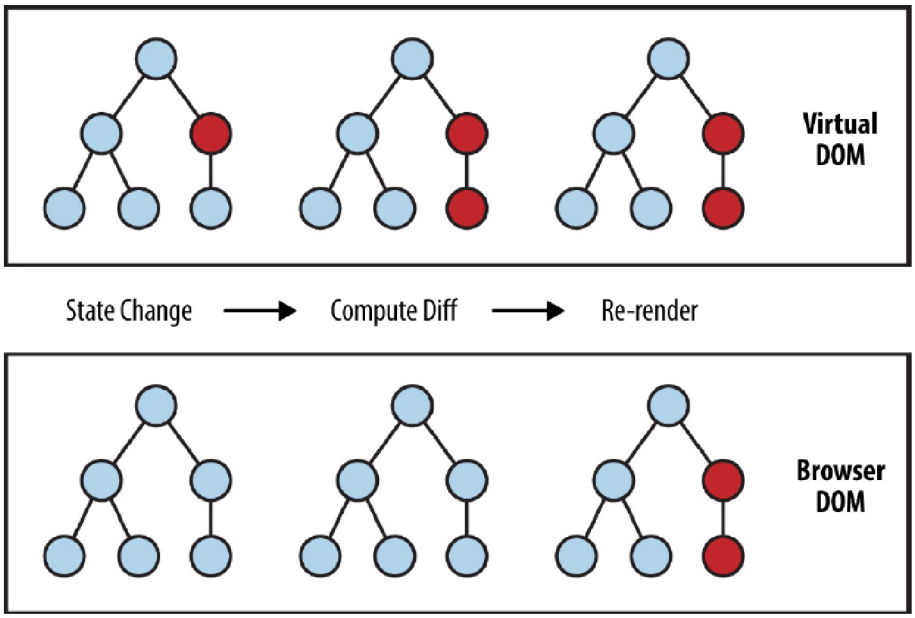
ulehčit kontrolu nad aplikací. Jednoduchý příklad komponenty jako funkce, která vrací kód v JSX. Příklad názorně zobrazuje velkou podobnost s jazykem HTML.

```
const App = () => {
    return <h1>Hello world React</h1>
};
```

React DOM

DOM (Document Object Model) je rozhraní k manipulaci HTML, které si specifikují jednotlivé webové prohlížeče. Díky tomu je možné například pomocí JavaScriptu ke všem elementům přistupovat a modifikovat je. Změny se odehrávají pouze na straně prohlížeče. Nevýhodou této reprezentace je náročnost nadměrných změn, které mají velký vliv na výkon aplikace, jelikož se s každou změnou musí znova vytvořit celý strom aplikace.

Z tohoto důvodu byl navržen virtuální DOM od tvůrců Reactu, který obsahuje kopii DOMu v paměti. React proto využívá tento virtuální model k vytváření a editaci elementů. Změny komponent neopotřebovávají zdroje, jelikož není zatím potřeba nic vykreslovat. Tento virtuální model je následně porovnán s aktuálním modelem stránky. Výsledkem je změna modifikovaných uzlů ve stromové struktuře DOMu. Díky různým heuristikám je porovnání velmi rychlé. Průběh je zobrazen na obrázku 2.1. Změna jednoho uzlu se uvažuje jako změna celého podstromu tohoto uzlu. [1]



Obrázek 2.1: Překreslení DOM na základě změny stavu ve virtuálním DOMu. Obrázek převzat z [17].

2.2 Nativní vývoj

Na rozdíl od webových aplikací, nativní mobilní aplikace neběží ve webovém prohlížeči. Musí je uživatel stáhnout z konkrétního obchodu pro danou platformu jako je například App Store nebo Google Play. Po instalaci se objeví ikona aplikace na ploše telefonu a je

možné k ní přistoupit. Nativní aplikace je postavena pomocí nástrojů a knihoven pro vývoj softwaru (SDK) pro konkrétní operační systém. Pro Android je aplikace postavena pomocí Java Development Kit na platformě Java, pro iOS se vyvíjí pomocí iOS SDK s využití jazyka Swift.

Obecně nativní aplikace reagují rychleji na uživatelské vstupy, mají nejlepší výkon a poskytují další speciální funkce (jako je například offline ukládání dat). Na druhé straně pak stojí vývoj a údržba takové aplikace. Oprava aplikační logiky nebo přidání nové funkcionality je potřeba přidat na více míst podle počtu podporovaných platforem. Tento přístup nemusí být ideální pro aplikace, které nepotřebují přístup k nativním funkcím telefonu nebo jim nezáleží na výkonu. Další výhody a nevýhody shrnuje následující soupis.

Výhody [3]

1. Dostupnost a vystavení na App Store, Google Play a dalších.
2. Přístup k nativním funkcím telefonu (fotoaparát, kamera, kontakty, stav sítě).
3. Jednodušší generování zisku. Uživatelé jsou srozumění s tím, že za aplikace se někdy musí zaplatit.

Nevýhody

1. Vysoká cena na vývoj a podporu (udržovatelnost).
2. Kód může být nasazen pouze na jednu platformu.
3. Závislost na podmínkách konkrétního obchodu, kam chceme aplikaci umístit pro zákazníky (např. Apple má striktní podmínky pro vystavení aplikace).

2.2.1 Android

Android je operační systém založený na Linuxu a Javě. Originálně byl vytvořen jako startup firmou stejného jména, Android. V roce 2005 Google koupil Android a vzal si na starost vývoj [23]. Jedná se o open source, díky čemuž ho mohou výrobci volně používat ve svých zařízeních. Statistiky z portálu Statista (z období září 2020) ukazují, že systém Android používá přes 70 % lidské populace. Jedná se proto o nejrozšířenější operační systém pro mobily celosvětově. Aplikace jsou dostupné přes obchody jako je Google Play Store, Samsung Galaxy Store nebo Huawei AppGallery.

Vývoj

Pro vývoj se používá programovací jazyk Java a Kotlin. Oba jazyky jsou kompilované do bajtkódu, který pro svůj běh potřebují JVM (Java Virtual machine). Historicky se pro kompilaci aplikací používal *Dalvik*, který kompiloval bajtkód Javy pro konkretní nativní zařízení při spouštění aplikace. Problémem bylo pomalé spouštění aplikací, a proto byl od verze Androidu 5.0 nahrazen ART (Android Runtime). *ART* přidalo lepší optimalizace jako je například *Ahead-of-time* kompilace tedy kompilace během instalace aplikace do zařízení a správu paměti (Garbage collector). Kompilace probíhá následujícím způsobem

1. Zkompiluje se kód v javě do DEX (ART/Dalvik Executable) souborů obsahující bajtkód.

2. APK Packager pak zabalí soubory do APK souboru. Tento soubor pak obsahuje soubor DEX a dále může obsahovat taky obrázky nebo již přednaplněnou databázi.

3. Volitelně může být tento soubor podepsaný (některé obchody s aplikacemi mohou podpis vyžadovat). Podepisování má na starosti APK Packager. Pro podpis je potřeba soubor nazývaný *keystore*, který slouží jako certifikát pro ověření aplikace.

Pro vývoj se nejčastěji používá prostředí Android Studio dostupné pro operační systémy Windows i Linux.

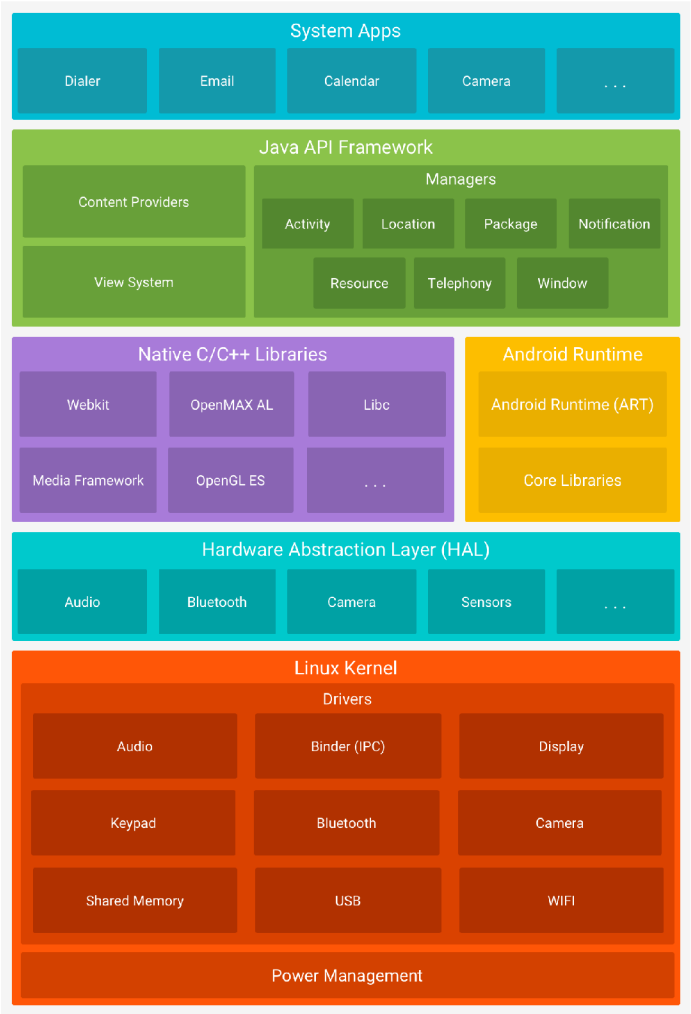
Systém

Systém je napsaný v Javě, jazyku C a C++. Schéma architektury je zobrazeno na obrázku 2.2. Linuxové jádro tvoří základní spodní vrstvu architektury a obsahuje nezbytné hardwarové ovladače, jako je například ovladač pro displej nebo fotoaparát. Nad ní běží hardwarová abstraktní vrstva (HAL) poskytující abstrakci mezi hardwarem a zbytkem softwaru. Aby bylo možné vykonat bajtkód, nesmí chybět vrstva s ART. Android dále poskytuje sadu knihoven napsaných v C/C++ (kvůli výkonu), které zpřístupňují funkcionalitu k nativním rozhraním telefonu. Patří sem knihovna *WebKit* pro spouštění webového prohlížeče, knihovna *SSL* pro zabezpečení přístupu k internetu, OpenGL ES (Embeded System) pro programování 2D a 3D grafiky a další užitečné knihovny. Funkcionalita, kterou Android nabízí, je přístupna přes JAVA API., které obsahuje [16, 18]:

- **Content Providers** umožňuje aplikacím sdílet data mezi ostatními aplikacemi.
- **View System** se používá se pro vytváření prvků uživatelského rozhraní.
- **Activity Manager** kontroluje a stará se o životní cyklus aplikací a zásobník aktivit.
- **Window Manager** je zodpovědný za zobrazování oken.
- **Package Manager** je systém díky kterému ví aplikace o jiných aplikacích nainstalovaných na daném zařízení.
- **Telephony Manager** poskytuje informace i telefonních služeb.
- **Resource Manager** poskytuje přístup k zdrojům, jako jsou obrázky, texty, nastavení barev atd.
- **Location Manager** poskytuje přístup k poloze telefonu.
- **Notification Manager** umožňuje aplikacím zobrazovat upozornění a upozorňovat uživatele.

2.2.2 iOS

iOS je mobilní operační systém vytvořený firmou Apple Inc. Pro jejich zařízení zahrnující iPhone nebo iPod Touch. Jedná se o druhý nejpoužívanější operační systém napříč mobilními zařízeními. Nejedná se o open source jako v případě Androidu, ale existují zde části, která spadají pod open source. Používání kódu spadá pod *Apple Public Source* licenci. [2]



Obrázek 2.2: Architektura operačního systému android. Obrázek převzat z [18].

Vývoj

Pro vývoj aplikací je potřeba pracovat na operačním systému OSX a používat uživatelské prostředí XCode, které obsahuje řasu pomocný nástrojů (SDK) pro vývoj. Jedná se o kompilátory, simulátory, emulátory a řadu dalších frameworků pro vytváření aplikací. Aplikace se píší standardně v jazyky **Swift**, který byl vyvinut jako náhrada jazyka Objective-C, který se používal původně. Základní kostra projektu obsahuje soubor *AppDelegate*, který má dvě hlavní funkce:

1. Definuje *App Delegate* třídu, která je zodpovědná za vytvoření okna aplikace, kde je vykreslen obsah aplikace a poskytuje místo pro reakce na změny stavu aplikace.

2. Vytváří vstupní bod a koordinuje vstupní a výstupní události aplikace.

Další důležitým souborem je *ViewController*, který definuje vlastní chování komponent, koordinuje tak informace mezi daty a zobrazením (View) a stará se o životní cyklus jejich obsahu. Pro tvorbu uživatelského rozhraní se používá *Interface Builder*. Nástroj podobně jako v Android studio umožňuje pomocí vybrání a přetáhnutí prvku vkládat tyto prvky na plátno a jednoduše měnit jejich atributy. Napojení se třídou View Controller zajišťuje

Identity Inspector. K definování interakcí v aplikaci a pro správné zajištění chování komunikace mezi View v Storyboadu a Controllerem je potřeba vytvořit akce, které budu reagovat na události a reference na objekty UI nazývané *outlets*. [2]

2.3 Hybridní vývoj

Hybridní přístup je kombinací nativního a webového přístupu. Jsou distribuovány do aplikačních obchodů a mají přístup k nativním funkcím telefonu stejně tak jako nativní aplikace. Největší jejich výhodou je možnost spuštění na více platformách bez nutnosti měnit výrazně kód. Pro vývoj se používají klasické webové technologie jako je HTML, CSS a JavaScript, které jsou zabaleny do nativního kontejneru a zkompilovány na jakoukoli platformu. Příkladem hybridní technologie je Cordova a Ionic.

Výhody

- Vývoj je **jednodušší** než vývoj nativních aplikací kvůli klasickým technologiím (HTML).
- S tím souvisí i **rychlost** a **cena** výsledného produktu.
- Není potřeba žádné speciální experty pro danou platformu. [26]

Nevýhody

- Stránky jsou vykresleny pomocí WebView, které zpomaluje výkon aplikace.
- Těžké hledání chyb pro vývojáře na specifické platformě.
- Závislé na nativních zásuvných modulech. S novou funkcí zařízení je potřeba počkat na vytvoření pluginu, jinak není možné novou funkci použít. [4]

2.3.1 Ionic

Ionic je open source nástroj pro tvorbu uživatelského rozhraní multiplatformních mobilních a webových aplikacích vytvořený v roce 2013. Kombinuje několik technologií a nástrojů. Je založený na technologiích Apache Cordova a AngularJS, kde Cordova poskytuje prostředí, kde aplikace běží a Angular se používá pro vývoj. Poslední verze navíc obsahuje sadu komponent dovolující uživateli si kromě Angularu vybrat mezi Reactem a Vue.js. Je dokonce možné využít ionic komponenty bez použití frameworku.

Pro vytvoření projektu je třeba použít ionic Command Line Interface (CLI), který obsahuje sadu nástrojů používaných v příkazovém řádku. Jelikož se jedná o webovou aplikaci, Ionic používá NPM (baličkovací systém pro Node.js) pro správu závislostí projektu.

2.4 React Native

React Native (RN) je JavaScriptový framework pro psaní nativní aplikací pro iOS a Android. Je založen na knihovně React popsané v kapitole 2.1.1 s rozdílem, že je cílen na mobilní platformy. Používá tedy všechnu funkcionalitu reactu jako je jazyk JSX, virtuální DOM a systém komponent. Navíc, React Native obsahuje *RN bridge* pro vykreslení aplikace do jazyku Objective-C pro iOS nebo do jazyku Java pro Android. Výhodou je vykreslení

aplikace pomocí nativních UI komponent. Nejedná se tedy o používání WebView (na rozdíl od hybridních aplikací) a proto působí aplikace jako nativní. [17]

Výhody

Největší výhodou je vykreslení aplikace pomocí standardních nativních prvků daného zařízení. Tímto vystupuje nad technologiemi Cordova nebo Ionic, jelikož ty používají k vykreslení WebView. Jejich přístup je sice funkční, ale některé nativní funkce nemůže WebView nahradit. Příkladem může být přístup ke kameře nebo různé další detaily jako jsou animace, které vezmou často spoustu úsilí k vytvoření, ale mohou se rychle stát neaktuálními.

Jelikož je framework založený na JavaScriptu, má rychlou křivku učení a vývoj není zdlouhavý, což souvisí s nižší cenou vývoje. Oproti nativním aplikacím je výhodou, že není potřeba psát kód zvlášť na konkretní platformu. V React Native je možné používat inteligentní nástroje pro ladění kódu, které nabízí vývojové nástroje nacházející se v prohlížecích (např. Safari nebo Chrome). Další důležitou výhodou je, že při vývoji není nutné neustále kompilovat kód, jelikož se změny aplikují automaticky. Tato výhoda může ušetřit několik minut denně jednomu vývojáři, což pro zaměstnavatele znamená několik hodin měsíčně, které by mu měl normálně zaplatit.

Princip

Stejně tak jako v Reactu, používá se zde virtuální DOM, kterým se zabývá sekce 2.1.1. Princip je podobný s jedním rozdílem. React Native namísto vykreslení aplikace do DOMu prohlížeče, vyvolá Objective-C API k vykreslení komponent pro iOS a Java API k vygenerování komponent pro Android. RN Bridge je takzvaný most napsaný v C++ a Java, který zajišťuje správné namapování komponent do nativních elementů, proto například komponenta *<View>* se převede na komponentu *<UIView>* pro IOS. React Native aktuálně podporuje iOS a Android, ale díky této abstraktní vrstvě virtuálního DOMu by mohl podporovat i další platformy, pokud by někdo sestrojil bridge.

Komponenty

Komponenty React Native, stejně tak jako v Reactu, jsou základním elementem této knihovny. Nejjednodušší způsob, jak definovat komponentu je napsat JavaScriptovou funkci:

```
function Welcome(props) {
    return <Text> Hello, {props.name} </Text>
}
```

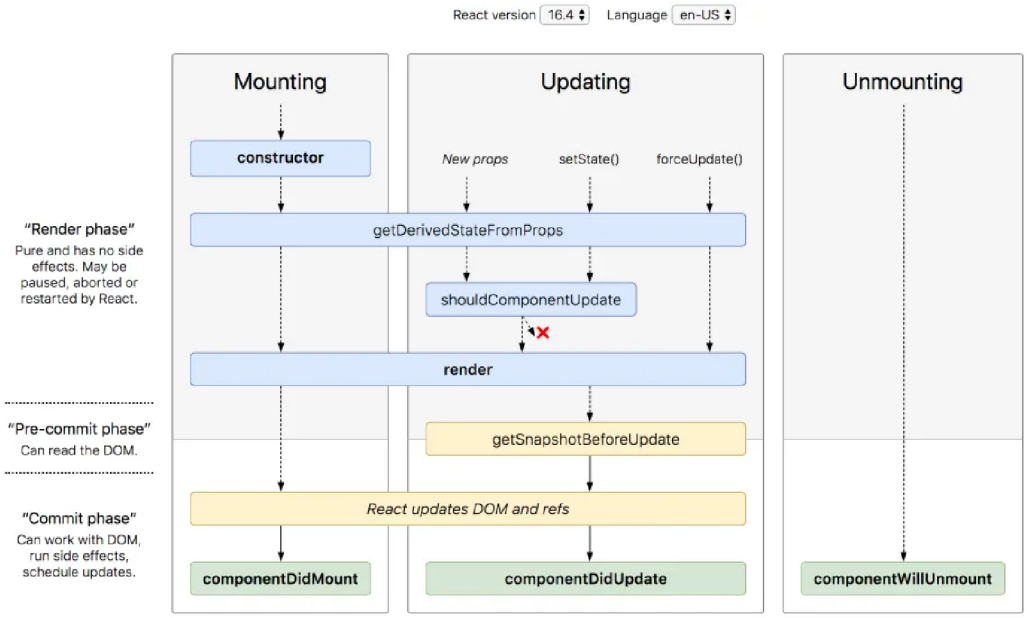
Tato funkce má jeden parametr a vrací React element. Nazývá se funkční komponenta. Druhou kategorií jsou třídní komponenty, které zjednodušují práci při vývoji, jelikož přidávají další možnosti jako je lokální stav komponenty nebo metody životního cyklu. Ať už se jedná o třídní nebo funkční komponentu, tak musí vždy platit, že komponenty nikdy nesmí modifikovat svůj stav. Změna komponenty se musí uskutečnit nahrazením pozměněné kopie této komponenty. Další důležitým pravidlem od tvůrců knihovny je, že všechny komponenty musí být **čisté** tzn., nesmí měnit svoje vstupy a pro stejné vstupy musí vždy vrátit stejné výstupy. Příklad nečisté komponenty:

```
function Welcome(account, amount) {
    account.total -= amount;
}
```

Tato komponenta modifikuje svůj vstupní parametr, proto není označena za čistou.

Na obrázku 2.3 lze vidět životní cyklus komponenty společně s dostupnými metodami a časovými okamžiky, kdy jsou tyto metody volány. Život komponenty má 3 fáze – navázání (mounting), aktualizace (updating) a odpojení (unmounting). Render je nejpoužívanější metodou, která způsobí vykreslení komponenty a volá se v průběhu navázání a aktualizace komponenty. Jedná se vždy o čistou funkci a nelze v ní měnit stav. [20]

- **ComponentDidMount** se zavolá v momentě, co byla komponenta vykreslena do DOMu.
- **ComponentWillUnmount** se volá vždy po odstranění komponenty. V této fázi je vhodné resetovat stav a uvolnit paměť.
- **ComponentDidUpdate** se zavolá při změně stavu nebo vlastností (props) komponenty. Uvnitř metody je možné měnit stav a proto je potřeba dávat pozor na zacyklení aplikace.
- **GetDerivedStateFromProps** se volá vždy před vykreslením komponenty (před metodou render). Reaguje na změnu vlastností komponenty změnou jejího stavu.
- **ShouldComponentUpdate** je metoda, která vrací logickou hodnotu na základě toho, jestli má být komponenty aktualizována nebo ne. Používá se k optimalizacím.
- **GetSnapshotBeforeUpdate** se volá po vykreslení zavolání funkce *render* a vrací hodnotu zpracovanou v metodě componentDidUpdate. Používá se například ke změně velikosti okna v případě asynchronního renderování.



Obrázek 2.3: Životní cyklus komponenty. Převzato z [24].

Typy komponent

React Native využívá speciální komponenty jako je například View nebo Text, které se kompilují do nativních komponent pro každou platformu. V následujícím seznamu se nachází seznam nejpoužívanějších komponent.

- **Text Input** – Jedná se o textový vstup, který umožňuje uživateli komunikovat s aplikací. Nejdůležitější vlastnosti jsou *onChangeText* a *onSubmitEditing*.
- **Seznamy** – K vytváření seznamů se typicky používají komponenty *FlastList* a *SectionList*. Komponenty je vhodné použít u často měnících se dat. Data jsou definována parametrem *data* a formát buňky parametrem *renderItem*.
- **View** – Pohledy slouží podobně jako tag *div* v HTML. Do *View* je obalována většina komponent k ulehčení stylování. Existuje zde systém flex podobný flexboxu z CSS, který je v React Native používán na umisťování, skládání a zarovnávání komponent.

Stylování komponent

Stylování probíhá velmi podobně jako v HTML s několika malými změnami v syntaxi. Každý prvek má k dispozici jiné parametry při stylování. Na rozdíl od HTML pracujeme se styly jako s objekty, které můžeme deklarovat pro vícenásobné použití. Takové řešení je doporučeno z důvodu přehlednosti a optimalizace, jelikož nedojde k jejich opětovnému vykreslení při volání funkce *render* [29]. Příklad klasického aplikování stylů na komponentu obsahující text:

```
const style = {
    backgroundColor: "white",
    fontSize: "16px"
}
const text = (
    <Text style={style}>
        Text s~aplikovaným stylem
    </Text>
)
```

React Native nabízí ale i možnost dynamického stylování s využitím vlastností komponenty. Tímto je umožněno ovlivňovat vzhled individuálních komponent, i když jejich základní vzhled je založený na společné šabloně.

Stav komponent

Komponenty využívající stavy pro data měnící se v průběhu času. Typické využití stavů je pro reagování na vstupy od uživatelů nebo změna obsahu obrazovky na základě časovače. Při práci se stavy je běžnou praktikou vytvoření několika bezstavových komponent, které se starají pouze o vykreslení dat a mají nad sebou komponentu, která využívá stavů a posílá tyto stavy svým potomkům pomocí `props` [32]. Vytvoření stavu se ve třídních komponentách dělá pomocí `setState` a v netřídních pomocí hooku *useState*. Zde se nachází příklad třídní komponenty a použití `setState`.

```
class Counter extends React.Component {
    state = { counter: 0 }
    render() {
        const { counter } = this.state
        return <Button onPress={() => {this.setState(prevCounter =>
          ({ counter: prevCounter + 1 }))}} title="Increment"/>
}}
```

Stav je uchováván jako objekt v konstruktoru a můžeme k němu jednoduše přistoupit pomocí `this.state`. Měnit jej lze pomocí `setState`, která přijímá za argument předchozí stav a vrací nový změněný stav. Stav není dovoleno měnit přímo, ale mělo by být změněno pouze pomocí `setState`. V netřídních komponentách ke stejnému účelu slouží hook `useState`, který přijímá jeden argument, kterým je inicializační hodnota stavu a vrací dvě hodnoty: stav a funkci sloužící ke změně stavu.

```
const Counter = () => {
    const [counter, setCounter] = React.useState(0)
    return <Button onPress={() => {setCounter(counter+1)}} title="Increment"/>
}
```

Jak lze vidět na příkladu, řešení pomocí hooku je přehlednější. Stejná funkcionalita zabrala 2x méně řádků. Po změně stavu dojde k automatickému vykreslení celé komponenty a jeho potomků, takže není nutné se o aktualizaci manuálně starat. Tato vykreslení nemusí proběhnout okamžitě, React Native může tyto požadavky chvíli pozdržet a provést více aktualizací současně.

2.5 Zabezpečení mobilních aplikací

Mobilní aplikace se stávají čím dál více součástí našeho všedního života. Lidé si berou telefon do práce, školy, do postele a často i na záchod. Najdou se i lidé, kteří ho prakticky nepouští z ruky. Aplikace používáme na placení, plánování, posílání zpráv, komunikaci nebo i k práci, obsahují často velmi citlivé a soukromé údaje a je potřeba je mít dostatečně zabezpečeny.

Nejnižší míru komfortu a zabezpečení nabízí něco, co víme (např. heslo nebo PIN). Hlavní idea přístupu je náhodná a lehce zapamatovatelná informace, což v sobě skrývá úskalí relativně lehkého získání této tajné skutečnosti nepovolanou osobou. Jelikož slovníkový útok na alfanumerické heslo zabere přibližně 5,5 hodin [9], na libovolné heslo 480 hodin [9], je potřeba vybírat složitější hesla. To ovšem může znamenat, že heslo zapomeneme. Vyšší míru zabezpečení a komfortu nabízí biometrie, tj. něco co jsme. Více o biometrii a biometrickém systému se bude nacházet v následujících kapitolách společně s vyhodnocením bezpečnosti.

Biometrie

Podle definice ze studijní opory Biometrických systémů, biometrie *„je automatické rozpoznávání lidí na základě jejich charakteristických anatomických rysů (např. obličej, otisk prstu, duhovka, sítnice) a charakteristického chování (např. podpis, chůze)“*. Skutečností je, že biometrickou vlastnost zřejmě doma nezapomeneme, ale přesto může dojít k nelegálnímu okopírování a následnému použití biometrické vlastnosti k přihlášení do aplikace. Naše otisky prstů zanecháváme téměř všude a vyrobit z otisku umělý prst není těžké. U fotografií se stejně tak může stát, že nás někdo vyfotí a dostane se pomocí naší fotky do aplikace.

K **výhodám** biometrie řadíme:

- odrazuje od podvodů,

- zvyšuje bezpečnost,
- nemůže být lehce ztracena či zapomenuta,
- zvyšuje pohodlí.

Biometrie má však i **nevýhody**:

- výstupem je skóre porovnání,
- nemůže být anulována v případě napadnutí,
- biometrický systém je napadnutelný a nezachovává soukromí.

Změřené výsledky se budou odlišovat při každém novém snímání, i u autentické osoby. Tato variabilita nastává změnou prostředí, stářím uživatele, tak i systémovými chybami. Základními požadavky na variabilitu jsou nízká **vnitrotřídní** variabilita a vysoká **mezitřídní** variabilita. Nízká vnitrotřídní variabilita znamená, že nebude docházet k výrazným změnám mezi vzorky stejného uživatele. Naopak mezitřídní variabilita má být vysoká, aby bylo možné rozpoznat jednotlivé uživatele od sebe navzájem.

Biometrický systém

Zjednodušené zobrazení biometrického systému je znázorněno na obrázku 2.4. Systém se skládá ze dvou modulů:

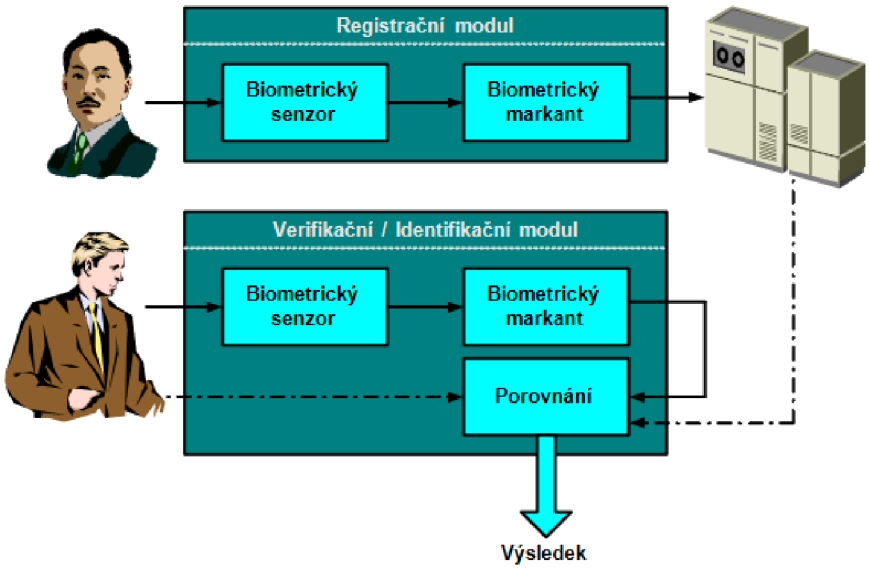
- **registrační** modul,
- **verifikační** modul.

V obou modulech se na začátku nachází biometrický senzor, který slouží k získání biometrické informace a k jejímu převedení do digitálního světa. Dále se nachází v obou modulech položka biometrický markant. Jedná se o extrahované rysy z biometrické informace, které v registračním modulu jsou uloženy do databáze. Naopak verifikační modul neprovádí ukládání, ale načítá markanty z databáze, aby mohl porovnat aktuální biometrický markant s údajem v databázi. Po provedení dostaneme výsledek, který závisí na nalezení shody nebo neshody a operačním módu (verifikace či identifikace).

Biometrické vlastnosti

Biometrické vlastnosti můžeme rozdělit do dvou kategorií. První kategorií jsou **anatomické** vlastnosti, do kterých řadíme otisk prstu, obličej, DNA, duhovku oka, sítnici oka, geometrii ruky, dlaň, termogram obličeje, termogram ruky, podpis a další. Tato anatomická vlastnost je vždy přítomna a není lehce ovlivnitelná změnou okolí. Druhou kategorií jsou **dynamické** vlastnosti (chování či jednání) jako je například hlas, gestikulace obličeje, dynamické vlastnosti podpisu, pohyby rtů nebo chůze. Tyto vlastnosti jsou často spojeny s nějakou akcí uživatele a často vedou k odlišné sadě vzorků 2.4. V mobilních zařízeních se můžeme zatím setkat s oběma kategoriemi. U statického je příkladem otisk prstu a u dynamického se setkáváme s nakreslením gesta, které může být uloženo i s informacemi o rychlosti psaní nebo tlaku vyvinutému na mobilní zařízení.

V souvislosti s těmito rysy rozlišujeme dva typy biometrických systémů.



Obrázek 2.4: Biometrický systém. Převzato od M. Drahanského z [14].

- **Unimodální biometrický systém** je systém používající pouze jednu biometrickou vlastnost. Nevýhodou je nižší spolehlivost, ale za to má nižší pořizovací náklady.
- **Multimodální biometrický systém** používá buď více příznaků jedné vlastnosti (např. statické a dynamické vlastnosti podpisu), a nebo používá více biometrických vlastností (např. rozpoznání obličeje a sítnice). Tyto systémy zvyšují spolehlivost, jsou robustnější k útokům, ale mají velkou pořizovací cenu.

Každá biometrická vlastnost má jisté atributy. Mezi základní z nich patří univerzalita (každá osoba by měla mít tuto biometrickou vlastnost), jedinečnost, získatelnost, výkonnost (biometrická vlastnost se nesmí změnit nebo zestárnout), přijatelnost, bezpečnost proti falšování a finanční náklady na pořízení. [5]

V praxi se nejčastěji setkáváme s otiskem prstu a rozpoznáním podle obličeje. Obě řešení mají vysokou univerzalitu, jsou lehce získatelné, dobře akceptovatelné, levné, ale jsou lehce zfalšovatelné. Vědci už dokonce vyvinuli systém *DeepMasterPrints* založený na principu neuronových sítí, který podle skutečných otisků prstů vygeneroval umělé vzorky. S jeho pomocí se dokázali dostat do zařízení s úspěšností až 77 %. [6]

Na konci roku 2018 zveřejnila společnost Forbes test, který měl ověřit úroveň zabezpečení chytrých telefonů. Mobilní telefony s operačním systémem Android odemkl trojrozměrný model obličeje vytisknutý na 3D tiskárně, zatímco chytré telefony iPhone X v testu obstály výborně. Důvodem jsou použité technologie. Společnost Apple investovala mnoho času na vývoj vlastního systému Face ID, který pro ověření používá hloubkovou mapu. Na druhou stranu mobilní telefony Android nabízí pouze levnější variantu skenování tváře. [7]

Závěrem je, že každá biometrická vlastnost je napadnutelná. Důležitějším faktem je, kolik úsilí bude muset vyvinout podvodník, aby se k citlivým údajům dostal, a jestli se mu krádež vyplatí. Z tohoto hlediska je lepší používat biometrické zabezpečení pro ochranu našich osobních údajů než použití hesla, které je jednodušeji prolomitelné.

Kapitola 3

Analýza současného stavu

Před návrhem aplikace je potřeba řádně prozkoumat existující moduly pro zaměstnance na VUT. Následující kapitola se bude zabývat obecně IS a procesy, které zde probíhají se zaměřením na zaměstnance. V první kapitole je upřesněno, kdo je zaměstnanec VUT a jaké jsou jeho práva a povinnosti. V sekci 3.2 jsou popsány existující informační systémy VUT, které zaměstnanci běžně využívají ke své práci. Poslední sekce 3.3 popisuje vybrané moduly IS rozdělené do tří kategorií na provozní, studijní a vědecké. Z těchto modulů jsou následně v sekci 3.4 vybrány statisticky nejdůležitější moduly.

3.1 Zaměstnanec VUT

Slovo *zaměstnanec* je velmi široký pojem. Může se jednat o akademické pracovníky nebo o další zaměstnance, kteří se podílejí na tvůrčí činnosti nebo zajišťují administrativní, ekonomické, organizační a technické činnosti, případně se podílejí na činnosti vzdělávací [33]. Každý z nich vykonává na škole jinou funkci a jiným způsobem používá informační systém. Někomu stačí informace na webu, kde si zobrazí výplatu a naopak někdo potřebuje komplexnější pohled s více daty a operacemi a musí použít jiný systém.

Všichni zaměstnanci mají několik věcí společných. Jsou povinni respektovat předpisy a ustanovení VUT, ve kterých je mimo jiné povinnost informační systémy VUT řádně využívat a zadávat do nich včas požadované nebo potřebné informace [33]. Jak již zaznělo v předchozí větě, která je citovaná ze Statutu VUT, VUT IS není jeden, ale je jich více. Součásti VUT se buď rozhodly používat svůj vlastní IS anebo používají centrální systém. Za chod a správnost informací zodpovídá zaměstnanec jmenovaný do funkce uvedené ve statutu fakult (např. tajemník).

Zaměstnanec je ovlivněn svou funkcí a vztahem k fakultě nebo její součásti. Každá součást má jiné procesy a řeší věci v jiné oblasti. Na FITu se bude více řešit hodnocení cvičení než na FaVU, kde je důležitější konzultace s vedoucím ohledně jeho maleb, a na FSI se spíše bude řešit docházka studentů na výuku. Ústavy mohou mít jinou metodiku zadávání známek. Někteří si píší jednotlivá hodnocení na papír, spočítají je na konci semestru a zadají finální body do systému. Někteří zadávají body hned. Dále mohou mít jiný způsob vykazování docházky do práce – elektronicky nebo na papír.

VUT se člení na fakulty, vysokoškolské ústavy a další součásti, takto: [33]

Fakulty

1. Fakulta stavební (dále pouze **FAST**),

2. Fakulta strojního inženýrství (dále pouze **FSI**),
3. Fakulta elektrotechniky a komunikačních technologií (dále pouze **FEKT**),
4. Fakulta architektury (dále pouze **FA**),
5. Fakulta chemická (dále pouze **FCH**),
6. Fakulta podnikatelská (dále pouze **FP**),
7. Fakulta výtvarných umění (dále pouze **FaVU**);
8. Fakulta informačních technologií (dále pouze **FIT**),

Vysokoškolské ústavy

1. Ústav soudního inženýrství (**ÚSI**),
2. Centrum sportovních aktivit (**CESA**),
3. Středoevropský technologický institut (**STI**),

Další součásti

1. Centrum výpočetních a informačních služeb (**CVIS**),
2. Institut celoživotního vzdělávání (**ICV**),
3. Koleje a menzy (**KAM**),
4. Nakladatelství VUTIUM,
5. Rektorát,
6. Ústřední knihovna.

V čele fakult je děkan, kterého jmenuje a odvolává rektor. Děkan pak jmenuje a odvolává tajemníka fakulty. Jeho období je čtyřleté a mohou ho zastupovat v určeném rozsahu proděkani, které jmenuje a odvolává. Všechny fakulty mají následující orgány: akademický senát, děkana, vědeckou radu, disciplinární komisi a tajemníka (s výjimkou FaVU, která místo vědecké rady má uměleckou radu).

Dále se dělí na jednotlivá pracoviště, které lze souhrnně vidět v tabulce 3.1. Všechny mají děkanát a všechny kromě FaVU obsahují ústavy a různá centra. Děkanát je výkonným útvarem pro zabezpečení provozu, hospodářské a administrativní činnosti, které blíže stanovuje Organizační řád jednotlivých fakult. Ústav je základním útvarem fakulty, který zajišťuje vzdělávání v akreditovaných programech a dále zajišťuje vědecko-výzkumnou, vývojovou a další tvůrčí činnost.

Za každou fakultu je vybrána funkce a k ní osoba, která zodpovídá za chod IS. Upřesnění těchto funkcí si stanovuje fakulta sama ve svém statutu. Pokud zde není osoba uvedena, předpokládá se, že za IS je zodpovědný integrátor dané fakulty.

U vysokoškolských ústavů není v čele děkan, ale ředitel, kterého jmenuje a odvolává rektor. Organizace se lehce liší od fakult. Mají pouze vědeckou radu, ředitele a ÚSI má navíc ještě laboratoře a knihovnu. Všechny používají IS VUT.

	Organizační struktura	Zodpovědnost za IS
FAST	ústavy, Děkanát, KIC a centra (centrum AdMaS[1])	Centrum informačních služeb - CIT
FSI	ústavy, Děkanát a specializovaná pracoviště (laboratoř přenosu tepla a proudění, NETME Centre)	předseda rady pro IS FSI
FEKT	ústavy, Děkanát a výzkumná centra (Centrum výzkumu a využití obnovitelných zdrojů energie, Centrum senzorických, informačních a komunikačních ústavů)	-
FA	ústavy, Děkanát a centra (knihovna, výpočetní centrum,modelové centrum, galerie mini)	předseda rady pro IS FA VUT
FCH	ústavy, Děkanát z výzkumná centra (Centrum materiálového výzkumu)	-
FP	ústavy, Děkanát a další součást (knihovna, technicko-provozní oddělení)	-
FaVU	ateliéry, kabinety, katedry, Děkanát, ekonomické oddělení a knihovna	předseda rady pro IS FaVU
FIT	ústavy a další součásti (Centrum výpočetní techniky, Výzkumné centrum informačních technologií)	vedoucí CVT FIT

Tabulka 3.1: Organizační struktura a zodpovědnost za IS podle fakult, informace vychází ze Statutu daných fakult.

3.2 Informační systémy VUT

Informačních systémů existuje na VUT více. Na FIT a FAST dokonce částečně používají svůj vlastní informační systém a pouze některá data se nahrávají do centrálního systému (za nahrání dat zodpovídá integrátor). Centrálním systémem se myslí systém spravovaný na CVISu. Vývoj začal v roce 2002. Systém se nazývá Apollo. Jedná se o desktopovou aplikaci vyvíjenou pro systém Windows. Je napsaný v jazyku Objective-Pascal a stále se vyvíjí. Kvůli zjednodušení přístupu do IS a s vývojem moderních technologií vznikl druhý IS na webu, který kopíruje základní funkce systému Apollo.

3.2.1 Apollo

Systém je postaven na třívrstvé architektuře od firmy Asta Technology Group. V této architektuře nepřistupuje klient přímo k datům databáze, ale používá se k přístupu aplikační server. Díky tomu je přesunuta aplikační a datová logika na aplikační server a tím jsou sníženy požadavky na klientskou aplikaci. Navíc je tím dosaženo větší bezpečnosti. Aplikačních serverů může být i několik v závislosti na tom, kolik uživatelů systém využívá. Pro potřeby VUT by jeden server nestačil, proto jich je v současné době deset.

Aplikační servery

Aplikační servery nazývané **Akira** jsou postaveny na technologii Asta, který přistupuje k centrálnímu datovému skladu se zohledněním pokročilých přístupových práv. Server se

po přihlášení uživatele stará o zpracování SQL dotazu a předávání výsledku klientovi s tím, že klient neposílá dotaz přímo, ale pouze řekne, který z uložených dotazů chce vykonat a pošle příslušné parametry. Díky tomu nelze data z databáze získat neoprávněným způsobem nebo je dotazy modifikovat. Data mezi Akirou a Apollem jsou komprimovaná, posílána v binárním kódu. Veškeré operace, které uživatel provede, se zaznamenávají do logu, takže je možné zpětně všechny přístupy dohledat.

Modularita

Celý informační systém je modulární. Uživatelé mají možnost si stáhnout do klienta jen části (moduly) systému, které používají. Klient pak zajistí automatickou aktualizaci modulu, připojení k aplikačnímu serveru, zamykání po delší době neaktivity, ukládání uživatelského nastavení a spuštění jednotlivých modulů.

3.2.2 Web

Kvůli jednoduššímu přístupu k IS přizpůsobené pro běžné uživatele, vzniklo další rozhraní IS na webu. Má méně funkcí. Soustředí se pouze na ty podstatné a je přehlednější. Jelikož se jedná o novější systém, nejsou zde zatím implementované všechny moduly. Pro některé zaměstnance se jedná o pohodlnější přístup, jelikož jsou spíše zvyklí na webové aplikace. Další výhodou je přítomnost popisků po najetí myší na většinu elementů na stránce, větší rychlost a celková přehlednost stránek pomocí vizuálních prvků.

Informační systém je psaný v jazyku PHP využívající centrální databázi. Dotazy SQL jsou přímo psány v kódu na rozdíl od systému Apollo. Používá se zde framework Zend (Techer, E-přihláška). Některé moduly používají vlastní framework (např. Studis). Je možné ho rozdělit do více kategorií.

- **Intraportál** je pro všechny uživatele. Obsahuje především provozní agendu, která nesouvisí se studiem. Jedná se například o novinky VUT, informace o výplatě nebo informace o čerpaných obědech. Pro vývoj se používá vlastní framework.
- **Studis** obsahuje veškeré moduly související se studiem od zápisů, registraci předmětů, vyučování, tvorba rozvrhu, správa stipendií, elektronický index, až po konec studia, závěrečné práce, potvrzení dodatku k diplomu. Pro vývoj se používá vlastní framework.
- **Teacher** slouží učitelům a obsahuje funkce, které potřebuje učitel k učení studentů. Nachází se zde modul zadávání známek, vypisování termínů, zadávání aktualit předmětů nebo je zde kompletně implementován modul závěrečných prací. Pro vývoj se používá framework Zend.
- **E-přihláška** slouží uchazečům k vyplnění přihlášky na VUT. Pro vývoj se používá framework Zend.
- **E-learning** je samostatný informační systém (Moodle) sloužící ke zveřejňování studijních materiálů, testů, vytváření anket či dotazníků pro studenty a je hlavním prostředkem komunikace učitele se studenty.

3.3 Používané moduly pro zaměstnance

V předchozí kapitole byla popsaná specifika součástí VUT pro lepší přiblížení pojmu zaměstnanec. Většina zaměstnanců používá centrální informační systém, který je spravován součástí CVIS, ale některé fakulty používají svůj vlastní systém (FIT a FAST). V tomto případě je velmi důležitá (alespoň částečná) integrace těchto systémů do centrálního systému. Za integraci je zodpovědný systémový integrátor. Následující text bude analyzovat pouze centrální IS, jelikož z něho se budou čerpat data pro tvorbu mobilní aplikace.

Informační systém lze rozdělit na tři části podle role uživatele:

1. Provozní část
2. Studijní část
3. Vědecká část

Provozní část

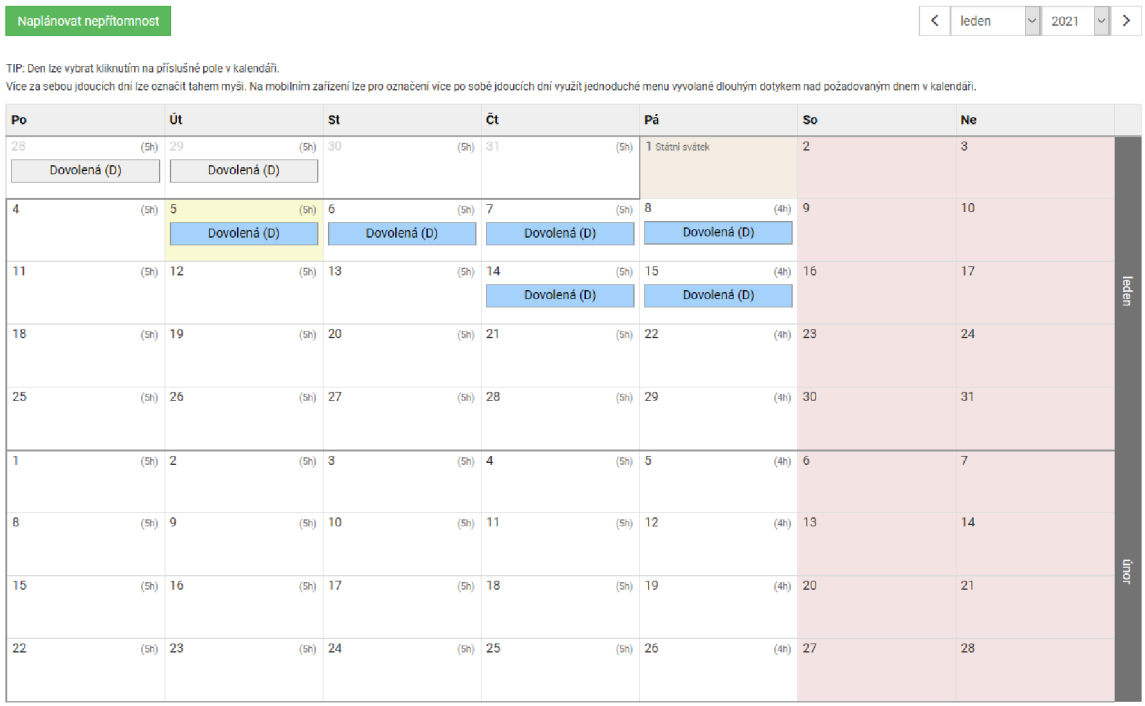
Provozní část obsahuje moduly určené pro zaměstnance. Na webu IS jsou dostupné pod záložkou Intraportál. Je možné ho rozdělit na dvě části – na část osobní a administrativní. Do osobní části patří například posílání VUT zpráv, zobrazení elektronické výplatní pásky za určitý měsíc nebo výpis volání v sekci VUT mobil. Administrativní moduly slouží ke zjednodušení evidence různých výkonů jako je například zaevidování nepřítomnosti, služební cesty nebo podepsání záznamu v podpisové knize.

Docházka a plánovaní nepřítomnosti

Tento modul poskytuje přehled nepřítomnosti zaměstnanců na pracovišti. Poskytuje informace o plánované, schválené nebo aktuálně čerpané nepřítomnosti v rámci pracoviště VUT a možnost zadávání, plánovaní a jejího schvalování. Právo na spuštění modulu by měli mít všichni zaměstnanci. Rozlišují se zde navíc tři oprávnění:

- Právo na **zobrazení** nepřítomnosti na vybrané součásti (lze použít k zobrazení nepřítomnosti součásti, kde nemá pracovní poměr a není zde ani vedoucí ani zástupce).
- Právo na **schvalování** nepřítomnosti na vybrané součásti (uživatel může schvalovat nepřítomnost, ale nechodí mu notifikace o naplánované nepřítomnosti).
- Právo na **zobrazení navíc záložky Schváleno** s přehledem schválené nepřítomnosti.

Základní výměra dovolené za kalendářní rok činí u akademických pracovníků 8 týdnů, u ostatních 6 týdnů [zdroj Kolektivní smlouva z roku 2019]. Nárok na čerpání dovolené vzniká po odpracování 60 dní. Schvalování provádí přímý nadřízený tj. vedoucí organizační jednotky, zástupce vedoucího nebo nadřízený schvalovatel organizační jednotky podle organizační struktury, s výjimkou těch vedoucích, kteří neschvalují nepřítomnost. Pokud daná jednotka nemá takovou osobu, schvaluje nadřízený schvalovatel nadřízené organizační jednotky, dokud se nějaký nenajde. Vedoucí pracovník kromě schvalování může i naplánovat nepřítomnost svým podřízeným.

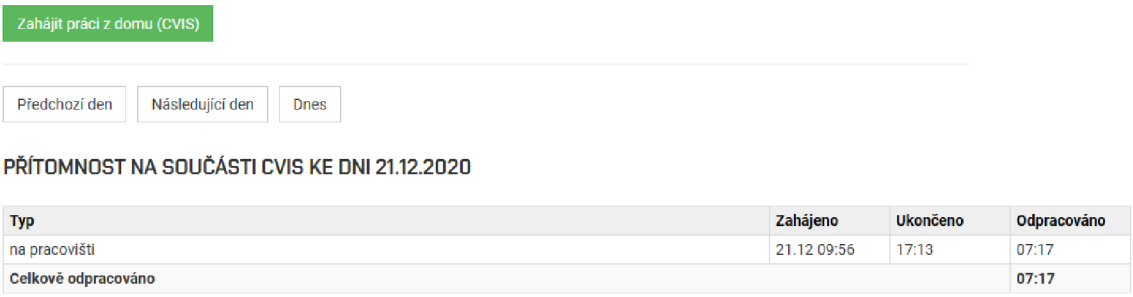


Obrázek 3.1: Prostředí zadávání nepřítomnosti na webu v sekci Intraportál

Grafické rozhraní je velmi intuitivní (obrázek 3.1) řešeno formou kalendáře s možností vybrání konkrétních dnů a zadání detailů nepřítomnosti. Ve formuláři je vždy potřeba vyplnit typ nepřítomnosti, datum od, datum do a volitelně místo pobytu a poznámku. Typ nepřítomnosti je například: dovolená, práce mimo pracoviště, lékař, náhradní volno, svatba, soud, pohřeb, neplacené volno, narození dítěte a další. U některých typů nepřítomnosti není vyžadováno schválení. V těchto případech bere pouze vedoucí pracovník plán nepřítomnosti na vědomí. Modul kontroluje nárok na dovolenou a upozorňuje uživatele na případné překročení nároku, nicméně plánování v případě překročení nebrání. Pokud jsou data vložena přímo do SAP (např. rodičovská dovolená), jsou do modulu přenášena do 24 hodin od zadání. Nepřítomnost se plánuje a schvaluje pro každý pracovní poměr zvlášť.

Podobný kalendář lze také zobrazit pro přehled nepřítomnosti oddělení. Přehled je důležitý, aby kolegové byli navzájem informováni, kdo a jaký den bude na pracovišti.

IS kromě evidence nepřítomnost může zobrazit i denní přehled práce. Tento modul není zapnutý na všech fakultách. Umožňuje zároveň zahájení práce z domu kliknutím na tlačítko Zahájit práci z domu na hlavní straně a podobným způsobem ukončení práce. Systém pak sčítá celkový odpracovaný čas, jak je vidět na obrázku 3.2.



Obrázek 3.2: Evidovaná docházka na pracovišti

Po rozhovorech s různými zástupci fakult jsem zjistila, že evidování práce na pracovišti se liší mezi jednotlivými ústavy a fakultami. Na ústavu CESA se docházka řeší ručně. Prakticky to dopadá tak, že jednou týdně nebo měsíčně si zapíšou docházku za celý měsíc, aby měli vykázaný čas. Na FITu se elektronicky zaznamená příchod a odchod při pípnutí zaměstnanecké karty u vstupu. Každý zaměstnanců pak vidí v tabulce v IS FIT souhrn odpracované práce a může si čas libovolně editovat, což se může hodit v případě, kdyby zaměstnanec zapomněl na zaevidování polední přestávky nebo příchodu do práce. Na FAVu vyplňují tabulku v Excelu a na konci měsíce posílají tabulku vedoucímu k podepsání. Na FEKTu mají knihu příchodů.

V souladu s ustanovením § 96 zákoníku práce je zaměstnavatel povinen vést u jednotlivých zaměstnanců evidenci pracovní doby s vyznačením začátku a konce směny. Tento zákoník přímo nedefinuje pojem evidence ani formu vedení evidence pracovní doby. Záleží tedy pouze na zaměstnavateli, který způsob zvolí, přičemž musí zohlednit zákonem stanovený minimální rozsah, správnost údajů i přehlednost a prokazatelnost. Patří navíc mezi nejčastěji kontrolované oblasti ze strany inspektorátu práce [15].

Modul evidence nepřítomnosti je dostupný v Apollu i na webu. Na webu je pohled trochu zredukovaný, chybí zde informace o zbývajících dnech dovolené kolegů ze stejného oddělení, jejich dlouhodobý plán nepřítomnosti a roční přehledy.

VUT mobil

VUT mobil je jeden z nejpoužívanějších modulů (ze statistik z webu za období 2019 – 2020), jelikož mnoho zaměstnanců projevilo zájem o zaměstnaneckou SIM kartu, která nabízí výhodnější volání, posílání SMS a využití dat. Získání tarifu probíhá jednoduše. Nejprve je elektronicky podepsána smlouva z domova nebo kanceláře a následně je potřeba zažádat o převod čísla od stávajícího operátora. Je možné si požádat o nové číslo a SIM na výdejně SIM.

Modul umožňuje zobrazení všech evidovaných telefonních čísel s informacemi o tarifu. U konkrétního čísla je možné si zobrazit vyúčtování pro daný měsíc. Modul neumožňuje si zobrazit počet vyčerpaných jednotek (dat, SMS, minut) za měsíc, který právě běží. Tato informace se do modulu nahraje až několik dnů po ukončení měsíce, kdy je spočítána celková částka tarifu. U zaměstnanců probíhá platba automaticky srážkou ze mzdy. I když započtení proběhne až na přelomu měsíce (tj. po splatnosti faktury), jedná se o platební metodu dohodnutou ve smlouvě a nebude stržena žádná sankce. Modul je dostupný v systému Apollo i na webu.

eVýplata

Elektronické výplatní pásky jsou dostupné nejdříve 6. pracovní den pro akademické pracovníky a technicko-hospodářské zaměstnance. Výplatní termín je 8. pracovní den. Univerzita používá k evidenci ekonomický informační systém SAP. Kromě VUT používá tento systém dalších 5 veřejných vysokých škol v České republice (mezi které patří například Mendelova univerzita v Brně nebo Univerzita Palackého v Olomouci). Tyto univerzity vytvořily uzavřenou smlouvu pro členy sdružení bez právní subjektivity s název Koordinační centrum SAP VVŠ. Účelem koordinačního centra je koordinace implementace zákaznických nastavení zavádění nových funkcionalit a sdílení infrastruktury pro provoz informačních systémů VVŠ.

Na webu je možné zobrazit výplatní pásku pouze přes rozhraní systému SAP, jak lze vidět na obrázku 3.3.

Obrázek 3.3: eVýplata v rozhraní SAP

Požadavky

Systém zadávání požadavků na informační podporu je platný od 1. 4. 2018. Slouží k zadávání požadavků týkající se informačních systémů, eSpisu a SAPu. Neměl by obsahovat žádné důvěrné informace a je nutné požadavky zadávat jednotlivě a konkrétně vždy ke konkrétní oblasti a s danou prioritou na zpracování.

Pokud má součást autora požadavku definovaného schvalovatele, je požadavek odeslán schvalovateli ke schválení nebo vyřešení. Schvalovatel je volitelný. Po schválení je požadavek předán moderátorovi dle zvolené součásti VUT a oblasti požadavku. Moderátor má pak za úkol přiřadit úkol řešiteli a volitelně může určit i záložního řešitele a to na základě kategorie požadavku nebo ručně. [11]

Pracovní cesty

Pracovní cesty jsou implementované na jiné subdoméně VUT s grafickým rozhraním, které neodpovídá jednotnému vizuálnímu stylu VUT[2]. Jedná se o systém navržený na FSI a v současnosti používaný pro evidenci a schvalování cestovních příkazů. Formulář pro přidání nové cesty je velmi dlouhý a mimo jiné obsahuje údaje o datu, času a místu cesty společně s informacemi o detailech přepravy (místo nástupu, dopravní prostředek, spolucestující) a nakonec informace o odhadovaných nákladech a zálohách.

Studijní část

Studijní sekce se týká modulů, které souvisí se studenty. Na webu je agenda dostupná pro učitele pod záložkou Teacher. Patří sem například vkládání hodnocení, zadávání termínu

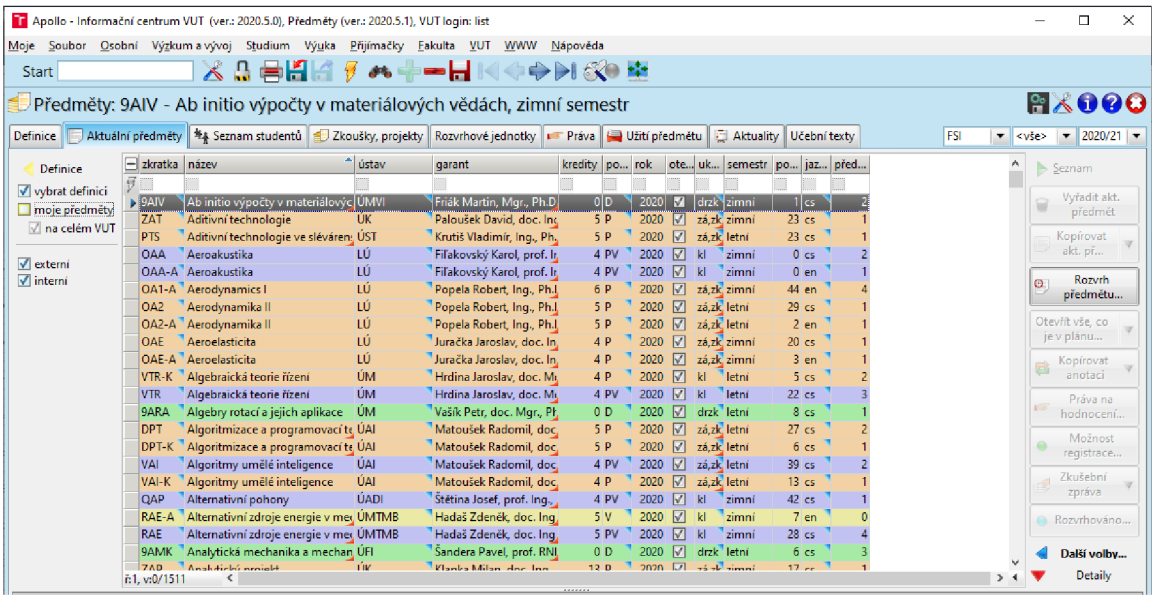
[2]Jednotný vizuální styl je možné si zobrazit a stáhnout z oficiálních stránek VUT na https://www.vutbr.cz/jvs.

na zkoušky, projekty nebo další dílčí hodnocení, nebo závěrečné práce. Do této sekce je možné zařadit i e-learning.

Předměty

Modul předměty je naimplementovaný jak v Apollu, tak na webu. Je určen k zobrazení, úpravě, vytváření předmětů, k prohlížení seznamu zaregistrovaných a zapsaných studentů, k zadávání hodnocení a nastavovaní práv vyučujících [27]. Na obrázku 3.4 je zobrazený tento modul v Apollu – je zde možné filtrovat podle fakulty, roku, mých předmětů a typu předmětu (externí, interní). U každého předmětu existuje 8 dalších záložek.

- **Definice** – zobrazen seznam všech definic předmětů vybrané fakulty.
- **Seznam studentů** – zobrazí všechny studenty předmětu společně s jejich hodnoceními, modul umožňuje filtrovat studenty podle času, skupin, uživatelských skupin, vyučujících,
- **Zkoušky, projekty** – slouží k zadávání nových termínů k typům hodnocení (zápočet, zkouška, laboratoře, cvičení), jejichž chování je parametrizováno fakultou.
- **Rozvrhové jednotky** - informace o všech jednotkách, které se vkládají do rozvrhu,
- **Práva** – zobrazení všech uživatelů mající daná práva k předmětu.
- **Užití předmětu** – seznam oborů, ve kterých je předmět ve studijním plánu.
- **Aktuality** – správa aktualit u předmětu.
- **Učební texty** – určené spíše pro vkládání textů, které se v čase nemění (např. učebnice, skripta), zobrazí se na hlavní straně předmětu na webu.



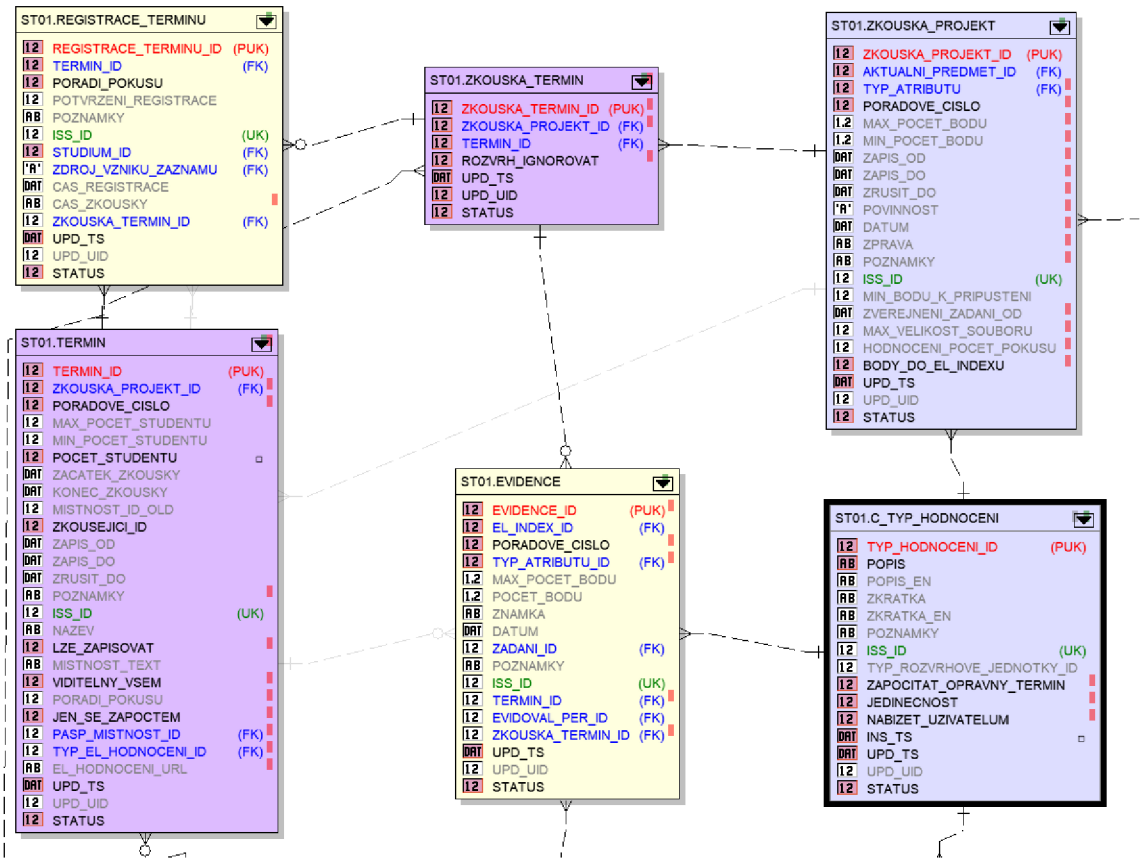
Obrázek 3.4: Seznam zobrazených předmětů

U seznamu studentů je dále možné vidět různé seznamy studentů. Studenty si lze zde filtrovat podle času, skupin nebo podle vyučujících. K možnosti editace výsledků studentů,

je potřeba mít zapnutý režim Povolit editaci (pokud je možnost zašednutá a nejde zaškrtnout, nemá přihlášený uživatel práva k zápisu nebo je uzavřena zkušební zpráva). Hodnotit lze přímo u zapsaných studentů. V tomto případě se studentovi uděluje hodnocení bez přihlášení na termín. Pokud ale existuje termín v seznamu vypsaných termínů, vždy je potřeba hodnocení zadávat tam. Pak lze hodnocení zobrazit v dílčích hodnoceních studenta v okně po označení konkrétního studenta.

Termíny

Termíny obsahují komplexní strukturu a jejich zadávání je odlišně naimplementované v Apollu než na webu. Horší pochopení kontextu z grafického rozhraní navíc komplikuje skutečnost existence více různých práv, které mají odlišný pohled na data. Proto k jejich pochopení je potřeba analyzovat strukturu databáze a organizaci dat.



Obrázek 3.5: Základní schéma k zadávání termínů

U termínů jsou zásadní tři tabulky – zkouska_projekt, zkouska_termin a termin. Jejich strukturu lze vidět na obrázku 3.5. Pro každý vytvořený termín musí existovat záznam v zkouska_projekt a může obsahovat 0 až N termínů propojené přes zkouska_termin, kde jsou ještě další informace o termínu. Termíny můžeme rozdělit podle několika kritérií.

Podle bodů

- **bodované** – studenti jsou hodnoceni číselnou hodnotou (body),

- **nebodované** – studenti jsou hodnocení známkou podle ETCS (číselník je generován každý rok)

Podle zadávání

- **automatický termín** – vygeneruje se až v momentě, co učitel zadává hodnocení (za předpokladu, že žádné takové již neexistuje), povoleno pouze u některých typů hodnocení,
- **vložený termín** – je potřeba ho vložit ručně a pak se k němu přiřazuje hodnocení.

Bodovanost termínů závisí na tom, jestli je vyplněno pole `max_pocet_bodu` a `min_pocet_bodu`. Další důležité atributy jsou:

- pořadové číslo – pokud uvedeno, termín se vztahuje k uvedenému pořadí termínu,
- povinnost – logická hodnota udávající, jestli je vyžadováno termín hodnotit,
- body do elektronického indexu – logická hodnota, která určuje, jestli se mají body propisovat do elektronického indexu.

Zápočet

Studijního a zkušebního řád VUT[3] říká, že zápočtem se potvrzuje aktivní účast studenta na práci během semestru a splnění všech požadavků, jímž bylo udělení zápočtu podmíněno, případně prokázání odborné způsobilosti rozpravou při kolokviu. Pokud student zápočet nebo klasifikovaný zápočet nezíská, neabsolvoval předmět. Udělení zápočtu je proto jedna z nejdůležitějších funkcí učitele. Z hlediska struktury databáze můžeme zápočet rozdělit do dvou skupin:

- typ hodnocení zápočet (vkládá se do termínů)
- zápočet v elektronickém indexu (vkládá se rovnou do elektronického indexu)

Pokud je vytvořen termín zápočet, vkládá se hodnocení k tomuto termínu a do elektronického indexu se pak zápočet propisuje pomocí možnosti Přepočítat hodnocení (v Apollu), na webu se přepočet dělá automaticky.

Web má propracovanější kontrolu vkládání hodnocení než Apollo. V případě, že má být vložen do IS nebodovaný zápočet, web automaticky vkládá zápočet do *evidence*. Není totiž potřeba vkládat informaci k termínu, jelikož by se informace stejně hned přepsala do indexu. V případě ale, že existuje nějaké bodované dílčí hodnocení nebo je zápočet bodovaný, je potřeba přidat hodnocení k termínu.

Rozvrhy

Rozvrhy jsou významným modulem, bez kterého by studium nebylo možné. Mění se každý semestr a studenti si v něm často zjišťují, kde mají být na přednášce. Stejně tak učitelé potřebují vědět, kdy učí a kde mají být. Základním pohledem na rozvrh v IS je osobní rozvrh, který obsahuje události přihlášeného vyučujícího. Existují ale i další pohledy na rozvrh

[3] Dostupný na oficiálních stránkách VUT: https://www.vutbr.cz/uredni-deska/vnitrni-predpisy-a-dokumenty/-d149085/studijni-a-zkusebni-rad-vut-p140885

pro místnost, předmět, konkrétní přednáškové skupiny, fakulty, ústav, studijní skupinu, ročník a další. Souhrn všech definovaných pohledů se nachází v systému Apollo v modulu rozvrhy.

Každé rozvrhové okno obsahuje základní identifikaci předmětu, poznámku, časový rozvrh výuky, místnost a její obsazenost a dále může obsahovat i odkaz na studenty patřící ke konkrétnímu oknu, povinnost výuky, odkaz na Systém Moodle a na video výuku (používá se platforma Teams). Tento odkaz je automaticky vygenerovaný a učitel ho nemusí použít.

Povinnost výuky je definována pro konkrétní rozvrhovou jednotku předmětu, společně s informacemi o hodinové dotaci za celý semestr a typem jednotky. Typy rozvrhových jednotek jsou například:

- přednáška,
- cvičení,
- laboratorní cvičení,
- speciální seminář,
- konzultace,
- ateliér,
- cvičení z TV.

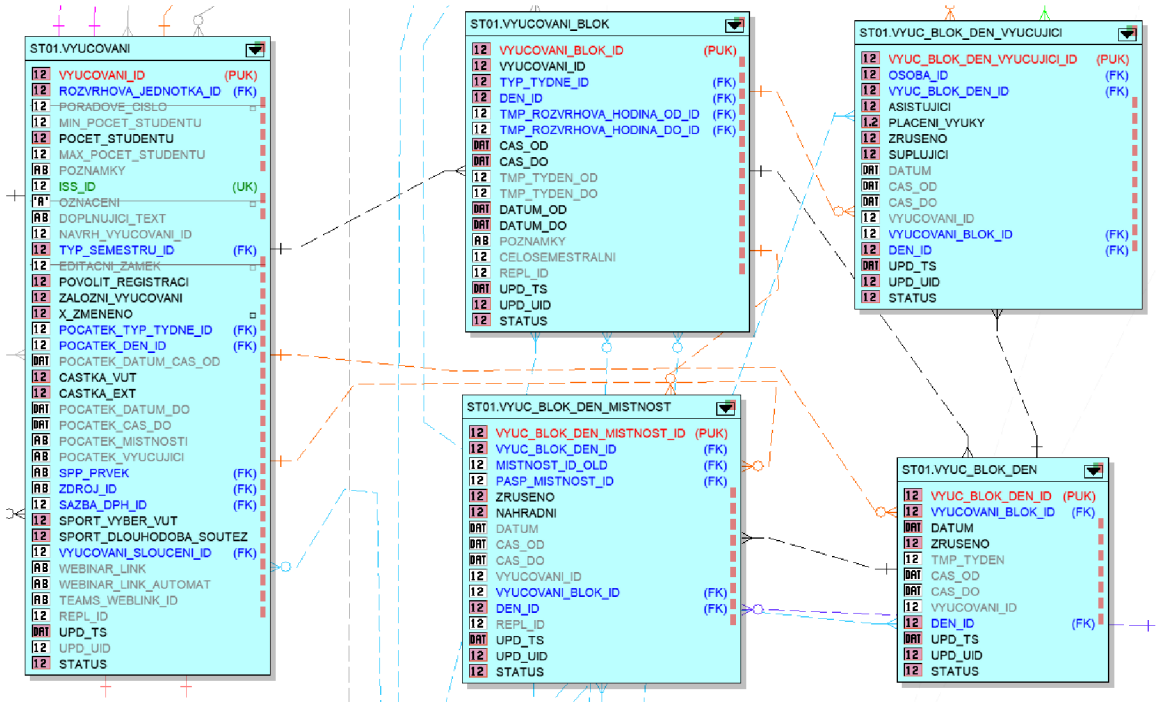
Systém zajišťující rozvrhy je velmi komplexní a není náplní práce ho důkladně studovat. Důležité je, že systémově jsou uložena všechna rozvrhová okna a je možné je editovat či přidávat. Neexistuje tedy jeden centrální rozvrh (jak je tomu například na FITu), který by určoval celému ročníku rozvrh na výuku, ale každý student si na začátku semestru svůj rozvrh volí. Velkou výhodou modulu je možnost zobrazit si všechny týdny rozvrh, kde hned učitel vidí dny, které má volné a které učí.

Minimalizované schéma jde vidět na obrázku 3.6. Základní tabulkou je *Vyucovani*, indikující Vyučování, na které jsou registrovaní studenti. Jedná se například o přednášku matematiky. Student smí navštěvovat pouze jedno vyučování vázající se na jednu rozvrhovou jednotku. Jedno vyučování pak může probíhat na půlky – jedno v podělí 11:00 – 12:50 a druhá půlka ve čtvrtek v 9:00 – 10:00 (*Vyucování_blok*). Student musí navštěvovat oba bloky. Blok probíhá v každém týdnu, do kterého patří. Pokud tedy probíhá blok sudé i liché týdny, bude se nacházet 13 záznamů (jelikož je 13 týdnů) v tabulce *Vyucovani_blok_den*. Vyučující a asistenti jsou vázaní na daný blok dne (tabulka *Vyuc_blok_den_vyucujici)* učící v jedné až N místností (*Vyuc_blok_den_mistnost*). Tímto způsobem je možné indikovat jakoukoli změnu vyučujícího nebo místnosti pro konkrétní den. Zrušení bloku je zaznamenáno pomocí sloupce Zrušeno. [zdroj Ing. Petr Kohut, CVIS – oddělení vývoje]

Nevýhodou zobrazení rozvrhu v systému Apollo je rychlost zobrazení. Pokud má rozvrh více položek, operace může trvat několik sekund. Na druhou stranu obsahuje větší množství pohledů na data než jsou na webu. V obou případech se jedná o zobrazení, kde na svislé ose se nachází den a na vodorovné jsou časové úseky.

Závěrečné práce

Závěrečné práce jsou jedním z modulů, který není kompletně naimplementovaný na webu. Proces začíná nejpozději na začátku semestru, kdy se vypisují a vybírají témata a končí



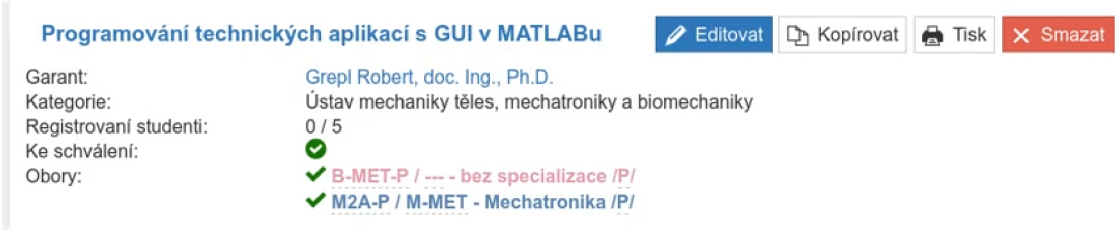
Obrázek 3.6: Základní schéma pro konkrétní vyučování

po obhájení závěrečné práce před státní komisí. Různé fakulty řeší závěrečné práce trochu jiným způsobem. Ke znázornění procesu jsem si vybrala Fakultu strojního inženýrství a v následujících odstavcích bude popsán zjednodušený průběh od podávání závěrečných prací až ke státní závěrečné zkoušce a vložení hodnocení.

FSI používá primárně IS web, ale vkládání hodnocení SZZ je možné pouze přes systém Apollo. Celý proces začíná u vytváření předběžného zadání závěrečné práce (dále jen ZP), kde je potřeba vyplnit obory, jejichž studentům se má předběžné zadání nabízet. Všechna předběžná zadání si je potřeba nechat schválit a až pak se na ně mohou registrovat studenti. Na obrázku 3.7 je zobrazeno, jak vypadá schválené předběžné zadání. U každého předběžného zadání (označované v Apollu jako Okruh) je nutné zadat maximální počet studentů, kterým může být předběžné zadání přiděleno. Po schválení se téma zobrazí studentům v modulu Registrace ZP, kde se mohou na práci registrovat. Po registraci je potřeba zajít za vedoucím a domluvit se na finální verzi ZP. Vedoucí může změnit položky zadání jako je charakteristika, název práce, prameny literatury. Dokonce může studenta zamítnout pro konkrétní téma práce. Editace ZP končí v momentě vytištění a předání ZP k podpisu děkanovi. Učitelé vidí všechna svá zadání v modulu Moje (předběžná) zadání, kde je možné vyhledávat ZP i z minulých let.

Po vložení práce do systému má oponent za úkol si práci přečíst a ohodnotit ji. Vkládá se celkové hodnocení a dále dílčí hodnocení. IS nabízí několik typů dílčích hodnocení jako je například splnění požadavků a cílů zadání, postup a rozsah řešení, adekvátnost použitých metod, vlastní přínos a originalita logické uspořádání práce a další. Každý typ je určený pro konkrétní funkce (oponent, vedoucí) a typ práce. Pomocí parametrizace se určí, jakým způsobem bude dílčí a celkové hodnocení hodnoceno. Možnosti jsou následující:

- hodnocení známou podle ECTS,
- slovní hodnocení,



Obrázek 3.7: Schválené předběžné téma závěrečné práce pro dva obory M-MET-P a M2A-P.

- hodnocení body.

Po odevzdání ZP a splnění dalších podmínek k připuštění k SZZ si může student podat přihlášku k SZZ a vybrat si termín z navrhovaných termínů. Ve stejném modulu se mu pak zobrazí číslo komise, datum, čas a členové komise. Seznam komisí, přiřazování studentů ke komisi a vkládání hodnocení se dělá přes Apollo. U každého studenta lze vidět jeho vážený průměr, přehled absolvovaných předmětů za celé studium, obsah závěrečné práce, odevzdané soubory a posudky. [zdroj Ing. Pavel Miček, PhD., CVIS – oddělení databází]

Moodle

Moodle je softwarový balíček pro tvorbu výukových systémů a elektronický kurzů na internetu. Je poskytován zdarma jako open source. VUT ho již používá přes 10 let stejně tak jako řada jiných univerzit jako je Masarykova univerzita, Jihočeská univerzita a Technická univerzita Ostrava [zdroj Mgr. Martin Kučera, CVIS – oddělení databází]. Jedná se o externí systém implementovaný na web VUT a data nejsou přímo propojená s centrální databází VUT. Všechny aktivní předměty musí být proto do systému moodle překopírovány (řeší se automaticky). Záleží pak na učiteli, jestli kurz využije nebo ne.

Ke každému kurzu je možné vkládat sekce pro logické rozdělení kurzu a zlepšení přehlednosti. U každé sekce je možné přidat činnost nebo studijní materiál. Jedná se například o diskuzní fóra, obsahy přednášek, místo k odevzdání úkolu a další.

- **Diskuzní fóra** umožňují účastníkům vést asynchronní diskuze, tj. diskuze, které probíhají po delší dobu.
- **Přednášky** umožňují vložit dokument s náplní přednášky.
- **Odevzdárny** slouží studentům k odevzdávání úkolů nebo projektů.
- **Testy** nabízí možnost vkládání testovacích otázek (možnost nadefinovat různé typy odpovědí).
- **Ankety** umožňují učiteli položit otázku a definovat výběr z více odpovědí.
- **Dotazníky** jsou podobné jako anketa, ale navíc umožňují širší možnosti větvení otázek.
- **Wikipedie** slouží ke společnému vytváření a editaci webové stránky.

Testy jsou vytvářené z banky úloh, kterou si musí učitel nadefinovat před vytvořením testu. Moodle umožňuje vytvářet testové kategorie, ke kterým vkládá testové otázky. Testové úlohy mohou být různých typů (např. výběr z možných odpovědí, pravda/nepravda,

dlouhé odpovědi). Při vytváření lze zadat kromě typu odpovědi, velikost testového pole (u textových odpovědí), povolení příloh, zamíchaní nabízených odpovědí, aby se různým studentům zobrazovaly v jiném pořadí nebo přímo vkládat šablonu odpovědi či přesunující informace pro hodnotícího. Na globální úrovni testu lze pak zvolit zamíchání testových úloh pro studenty nebo nastavení časování testu (limit na vyplnění, datum zpřístupnění testu, datum uzavření testu). Odpovědi lze přímo hodnotit v systému moodle, přidávat k nim komentář, je možné je exportovat nebo si zobrazit statistiky hodnocení všech studentů.

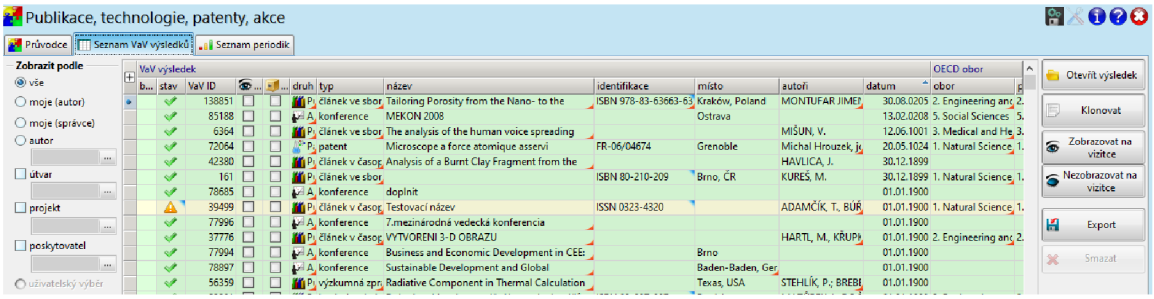
V současnosti moodle aktivně používá většina fakult, nejčastěji na sdílení výukových materiálů nebo právě vytváření testů. Největší nevýhoda systému spočívá v neumožnění automatického vkládání získaných hodnocení do elektronického indexu žáka. Není zde žádná návaznost, proto je potřeba výsledky přenést ručně. Model je provozován pouze na webu.

Vědecká část

Do vědecké sekce patří všechny moduly spojené s výzkumem na fakultě. Jedná se o především o akce spojené s projekty (vytváření, editace, financování) a vědeckými výsledky (publikace, akce, další produkty). Všechny moduly týkající se vědy a výzkumu jsou pouze implementované v systému Apollo, ale je v záměru v nejbližší době moduly implementovat na web.

VaV

Na hlavní straně modulu se nachází rozcestník na zadávání nových výsledků do katalogu vědy a výzkumu. Zde je možné vložit novou publikaci, akci, patent nebo nový produkt. Mezi publikace patří například článek ve sborníku nebo časopisu, kniha, kapitola v knize, skripta, konferenční sborník nebo zpráva. Vždy je potřeba vyplnit tento druh společně s dalšími charakteristikami publikace (název, popis, klíčovaná slova, DOI, kód WOS, URL atd.). Formulář je velmi dlouhý a obsahuje celkem 13 sekcí. Publikace může být vázána na nějakou akci nebo být přiřazena k jednomu nebo více projektům. Seznam všech VaV výsledků se nachází v sekci *Seznam VaV výsledků*, jak lze vidět na obrázku 3.8. U každého výsledku je barevně odlišen stav – zelené jsou v pořádku, žluté obsahují varování a červené jsou výsledky obsahující chyby.



Obrázek 3.8: Ukázka modulu věda a výzkum – zobrazení seznam všech VaV výsledků.

3.4 Statistika modulů VUT

Celkem se v současnosti nachází na univerzitě 4000 zaměstnanců na hlavní pracovní poměr, 66 zaměstnanců na dohodu a přes 9000 externistů, kteří mají většinou přístup do IS.

K vybrání nejpoužívanějších modulů pro zaměstnance je třeba získat statistiky ze všech IS VUT. Teoreticky moduly s největším počtem denních přístupů by měly určovat nejpoužívanější moduly. Nicméně kvůli řadě problémů vyjmenovaných v následujícím seznamu nebude výběr tak snadný.

1. **Systém je roztříštěný** na více systémů. Zaměstnanec si ve většině případů (pokud je modul implementovaný na více místech a pokud má zaměstnanec práva) může vybrat, který ze systémů bude používat.

2. **Diverzita funkcí** zaměstnanců. Uklízečka bude mít jiné potřeby přístupu do IS než učitel. Integrátor bude používat IS více než recepční.

3. Některé **moduly jsou dostupné i pro studenty**. Získané statistiky z Goodle analytics nedokáží oddělit studenta a zaměstnance. V tomto případě je možné věřit pouze statistikám, kam nemají studenti přístup, jinak budou hodnoty zkreslené.

4. **Zakázaný přístup učitelů** do určitých modulů. Učitelé (např. z FEKT nebo CESA) by mohli plně používat modul Teacher na zadávání známek a komunikaci se studenty, ale přístup mají vypnutý.

5. **Přístup do IS není rovnoměrný**. Rok je rozdělený na několik období a v každém období mohou být důležitější jiné moduly. Na začátku roku se řeší hlavně zadávání závěrečných prací a na konci roku bývá nejdůležitější vypisování termínů a hodnocení žáků.

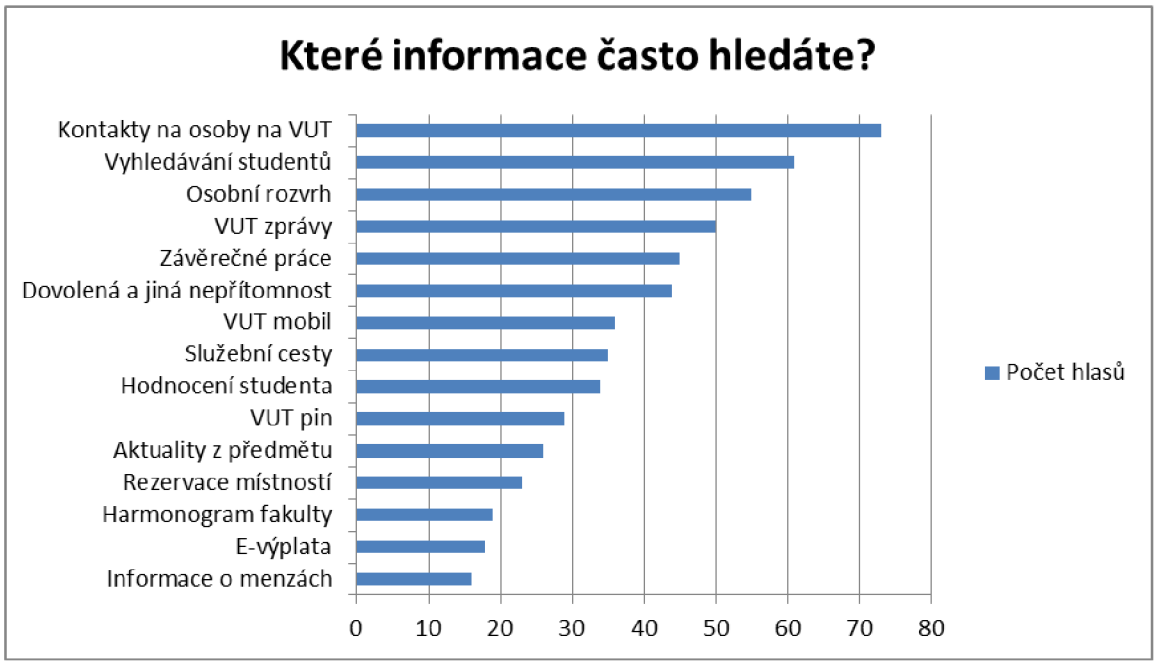
6. Data jsou z **různě dlouhých období**. Proto je potřeba analyzovat grafy IS samostatně a při porovnání stejných modulů napříč systémy normalizovat hodnoty.

Statistiky přístupů do jednotlivých částí IS lze vidět na obrázku 3.10. V případě modulu Teacher (sekce 3.4) a Intraportál (sekce 3.4) byly statistiky získány z nástroje Google analytics. Apollo (sekce 3.4) má uložená statistická data o přístupech v centrální databázi. Kvůli přechodu na nový modul Teacher2 se zapomnělo na zaznamenávání statistik, proto jsou data pouze od začátku září 2020. Moduly jsou seřazené podle počtu spuštění, který je znázorněný modrou barvou. Červená barva značí celkový počet uživatelů (Apollo) nebo počet sezení (Intaportál a Teacher), za které vstoupil uživatel minimálně jednou na danou stránku.

V souvislosti s vyhodnocením nejdůležitějších modulů jsem se ptala zaměstnanců, které informace vyhledávají nejvíce. Odpovědi jsou zobrazeny na obrázku 3.9 a analyzované v sekci 3.4. Na dotazník odpovědělo celkem 132 zaměstnanců, spíše se jednalo o učitele a vždy odpověděl alespoň jeden zástupce z každé součásti VUT. Nejvíce avšak hodnotili zaměstnanci z FEKTu.

Apollo

Data z **Apolla** jsou brána za období roku 2019. Největší počet uživatelů získal modul **Dialog pro výběr osoby** s počtem přes 2000 uživatelů, což tvoří přes 50 % z celkového množství zaměstnanců (a přes 30 % všech uživatelů Apolla). Dialog je otevřen vždy při přidání osoby do formuláře například při přidávání dalšího řešitele k požadavku. Ostatní moduly na prvních 10 místech mají všechny přes 1000 uživatelů, tedy jsou za rok důležité pro každého 4. zaměstnance. V počtu spuštění je nepoužívanější modul Předměty. Jelikož někteří učitelé nemají přístup k IS na webu, jedná se pro ně o jedinou možnost, kde zadávat



Obrázek 3.9: Odpovědi od zaměstnanců na otázku: „Které informace hledáte často?“. Celkem odpovědělo 132 zaměstnanců.

známky, hodnotit studenty a posílat jim zprávy, proto jsou také VUT zprávy na třetím místě. Z dalších modulů pro učitele jsou nejpoužívanější **Závěrečné práce** a **Rozvrhy**.

Teacher

Závěrečné práce a rozvrhy se umístili dobře i v modulu Teacher na webu. Osobní rozvrh vyhledává nejvíce uživatelů a má proto největší počet spuštění. Položky jsou seřazeny podle počtu sezení, ve kterém spustili daný modul. Jsou tímto vyfiltrovány počty spuštění, kdy uživatel neustále načítal webovou stránku. Důležité jsou i moduly spojené se studentem a to přesněji **Hodnocení studenta**, **Vypisování termínů** a **Posíláním zpráv studentům**.

Intraportál

Data z intraportálu jsou částečně ovlivněny tím, že zde mají přistup i studenti, kterých je na VUT kolem 20 000. Jedná se hlavně o E-maily, Průkazy, VUT pin, VUT software a částečně VUT zprávy. Ze zaměstnaneckého pohledu jsou nejvyužívanější (kromě VUT zpráv) moduly E-výplata, VUT mobil, VUT disk, Plánování nepřítomnosti a Požadavky. Výsledky nejsou překvapivé, jelikož výplatu dostává zaměstnanec každý měsíc a často si chce ověřit, jestli mu přišla na účet správná částka. VUT mobil má čím dál více zaměstnanců, kteří ho mají nejen pro sebe, ale i pro své známé. Modul jim poskytne přehled volání a celkové vyúčtování, proto do něho vstupují zaměstnanci často.

Dotazník

Většina dotazovaných na otázku „Které informace hledáte nejčastěji?“ odpověděla **Kontakty na osoby VUT** a **Vyhledávání studentů**. Souvisí to s faktem, že osob na VUT je přes 20 000 a tak není možné si všechny pamatovat. Proto je vyhledávání zásadní funkcí

IS. Další důležité moduly z pohledu zaměstnanců jsou VUT zprávy, Osobní rozvrh, Závěrečné práce, Nepřítomnost a VUT mobil. Tyto informace odpovídají statistickým údajům. Odpověď E-výplata byla to dotazníku vložena trochu později, proto je možné, že nedostala tolik hlasů.

3.5 Shrnutí

VUT používá více informačních systémů. V této sekci byly analyzovány pouze moduly z centrálního systému, který je implementovaný jako desktopová aplikace Apollo a některé moduly jsou znovu implementované i na webu. Největší výhodou webového přístupu je možnost spuštění modulů na všech operačních systémech, jednodušší přístup k IS a ovládání přizpůsobené pro běžného uživatele, proto se jedná o často preferovanou volbu mezi zaměstnanci. Na některých fakultách jsou ale jisté moduly zakázány na webu, nebo zatím moduly nejsou na web implementovány, proto zaměstnancům nezbývá než použití systému Apollo. Skutečnost existence více systémů přináší nutnost analyzovat statistiky pro jednotlivé části systému.

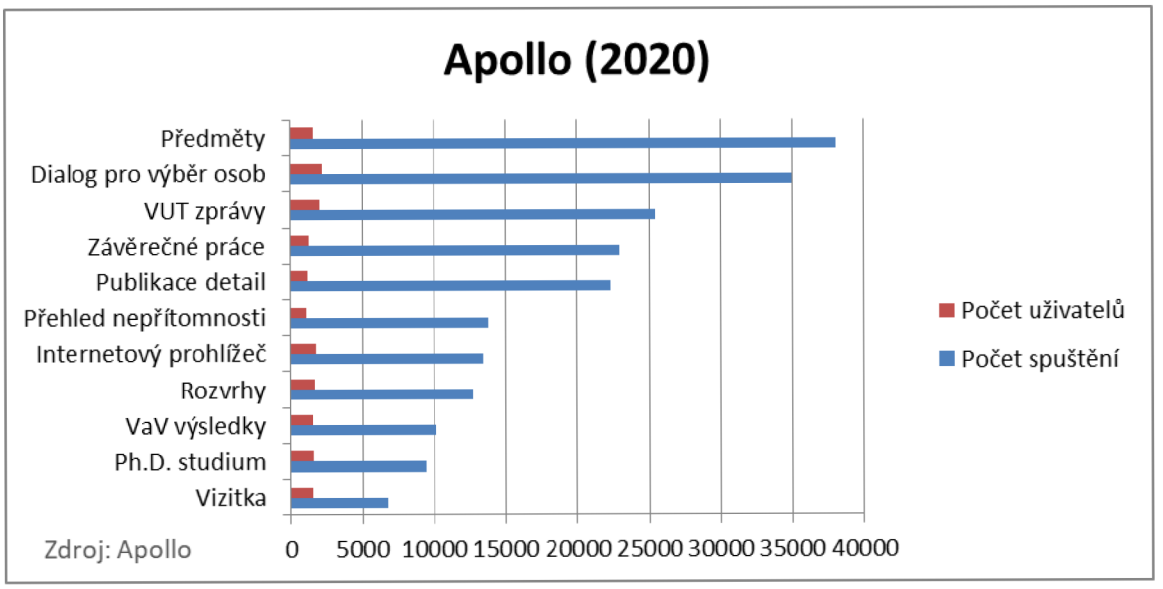
Statistiky byly získány pouze pro moduly, které používají zaměstnanci. Některé moduly mohou používat i studenti a proto mohou být někdy výsledky zkreslené. Navíc jsou některé moduly zaměstnancům zakázané, i když by do nich mohli mít přístup a mohli je používat. Napříč těmito problémy jsou vybrány nejpoužívanější moduly z informačních systémů pro zaměstnance.

Po vyhodnocení statistických dat byly jako nejpoužívanější moduly vybrány:

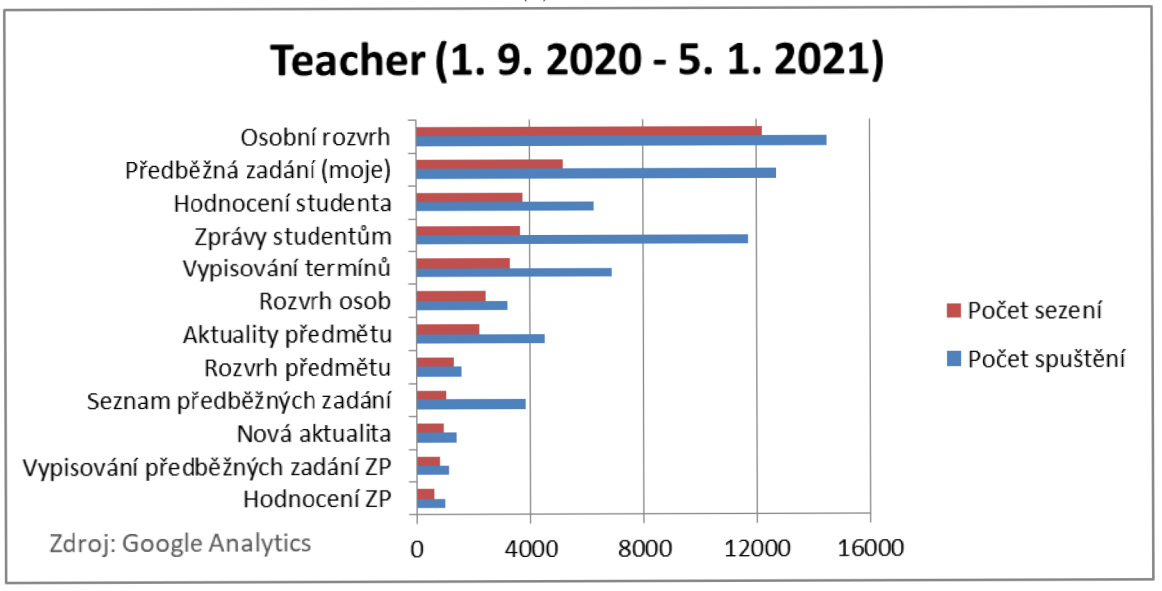
1. VUT zprávy,
2. E-výplata,
3. VUT mobil,
4. Osobní rozvrh,
5. Předměty a komunikace se studentem (zprávy studentům, vypisování termínů a hodnocení studenta),
6. Závěrečné práce.

Výsledky statistik byly ověřeny dotazníkem pro zaměstnance, který potvrdil významnost těchto modulů. Okrajově by sem mohla být přidána i Evidence nepřítomnosti, jelikož byla zvolena zaměstnanci jako jedna z hlavních modulů, které zaměstnanci vyhledávají.

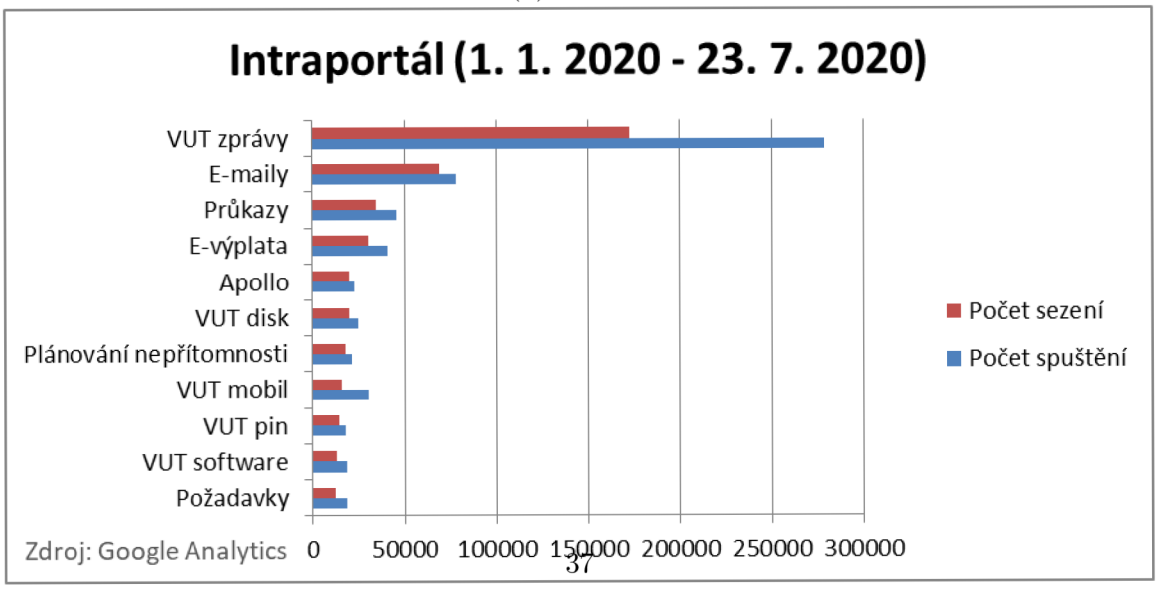
Pro tvorbu mobilní aplikace pro zaměstnance jsou důležité nejpoužívanější moduly, ale je potřeba i zvážit jejich použitelnost a využitelnost na telefonu. Je zbytečné znovu implementovat funkční rozhraní, které je snadno dostupné a přehledné na telefonu. Zároveň je třeba vzít v úvahu výhody, které nabízí nativní funkce telefonu, zaměřit se na rychlé a snadné úkony k ulehčení každodenních akcí zaměstnanci. Popis rozhraní bude detailně rozebrán v následující kapitole.



(a) Apollo



(b) Teacher



(c) Intraportál

Obrázek 3.10: Statistky z IS

Kapitola 4

Návrh aplikace

V předchozí kapitole byly podrobně popsány statisticky nejpoužívanější moduly. Nyní je potřeba určit, které z nich je vhodné mít v mobilní aplikaci, jaké výhody má aplikace přinést a jakým způsobem má ulehčit každodenní práci. Výběrem modulů se bude zabývat kapitola 4.1. V sekci 4.2 je vybrána nejvhodnější technologie na základě vybraných modulů. Sekce 4.3 se zabývá návrhem obrazovek.

4.1 Výběr modulů a funkčnosti

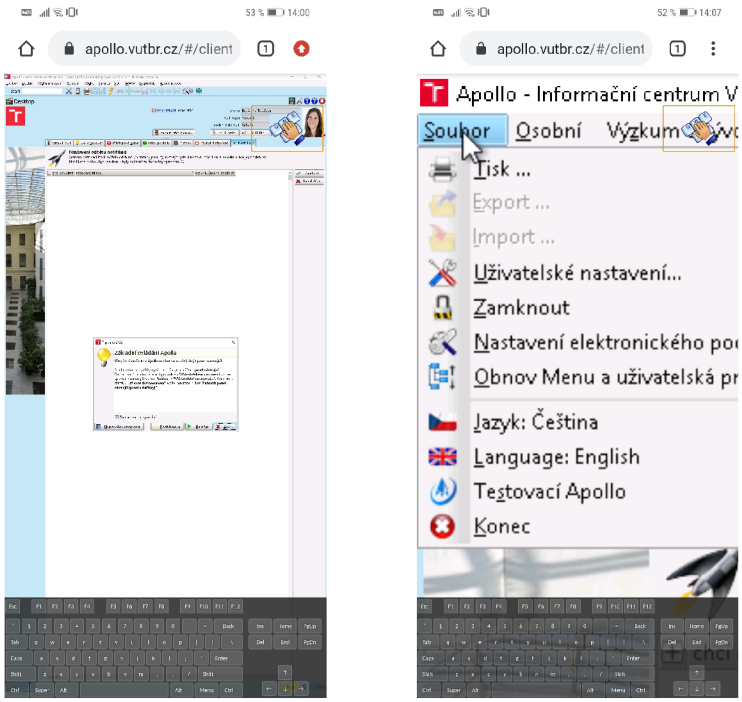
Jak bylo již dříve popsáno, VUT používá více informačních systémů, které jsou do detailu propracované, aby splnily všechny úkony, které by mohl uživatel potřebovat. Cílem mobilní aplikace není vytvořit další informační systém, který by pouze kopíroval již naimplementované moduly. Cílem je zjednodušení častých úloh, využití nativních funkcí na telefonu, přidání rychlejšího přístupu k nejpoužívanějším funkcím vedoucí ke zvýšení produktivity zaměstnanců.

Mobilní telefon má pravděpodobně většina zaměstnanců, ale jen někteří z něho přistupují do IS. Z dotazovaných uvedlo 65 % uživatelů, že vstupují do IS přes mobilní zařízení, ale pouze 16 % uvedlo, že se jim s IS pracuje dobře. Důvodů bylo uvedeno několik:

- Apollo nelze používat na telefonu.
- IS je málo přehledný na telefonu. Špatně se zde hledá.
- Rozhraní je neobratné díky velkým ovládacím prvkům a jejich nešťastnému zarovnání.
- Špatná orientace v agendě na webu. Velikosti oken IS nejsou optimalizované pro mobilní zařízení. Nepřehledné, nepohodlné.
- Problematický výběr záložek.
- Především špatné zobrazení rozvrhů a obtížné zobrazení studentů v jednotlivých hodinách v rozvrhu.
- Složité hledání rozvrhu přes mnoho stránek a formulářů.
- Tabulky jsou moc široké na telefonu.

Největší problémy byly s přístupem do systému Apollo přes mobilní zařízení. Jelikož přístup probíhá přes vzdálenou plochu a je potřeba vybrat aplikační server, na který se

klient připojí, může být přihlášení velmi pomalé a nemusí se podařit na první pokus. Zároveň se zde používá vestavěná klávesnice, která je velmi malá a psaní na ní je téměř nemožné. Na obrázku 4.1 lze vidět zobrazení Apolla na telefonu. Navigační položky jsou velmi malé, menu je rozmazané a orientace není snadná. Při testování zobrazení mého rozvrhu se mi nepodařilo zadat vyhledávací řetězec, jelikož klávesnice nereagovala z neznámého důvodu na vstupy. Testováno na HUAWEI Mate 20 PRO (Android 10).



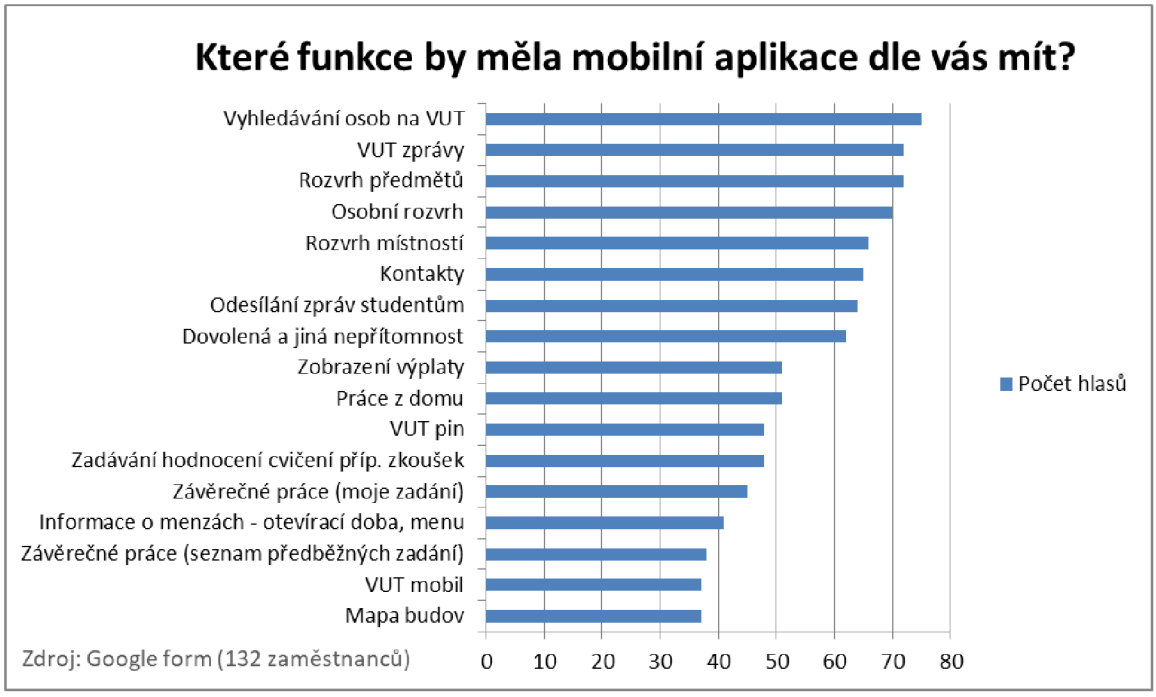
Obrázek 4.1: Náhled hlavní strany systému Apollo na telefonu.

Web je z mobilního hlediska mnohem lépe uzpůsobený, jelikož používá responsivní design. Položky se přizpůsobují tvaru obrazovky a práce je výrazně příjemnější. Problémem je zobrazení tabulek a rozvrhů, které není možné kompletně zobrazit na mobilním zařízení kvůli příliš velké šířce. Proto se musí uživatel posouvat podélně pomocí přetáhnutí prstu. Nešťastná může být pro některé uživatele velikost textu, proto si raději sednou k počítači.

Mobilní aplikace je vytvořena pro uživatele tak, aby ji rádi používali. Jejich názor a potřeby jsou velmi důležité. Ptala jsem se proto zaměstnanců, které funkce by podle nich měla aplikace mít. Výsledky zobrazuje graf na obrázku 4.2. Kromě běžných modulů přidávali zaměstnanci i různé nové zajímavé nápady. Zde jsou vybrané ty nejzajímavější nápady na rozšíření funkcionality:

- **Nahrání zaměstnanecké karty do aplikace**, aby s ní bylo možné platit obědy, otevírat dveře a vjezdy do areálu.
- **Notifikace** na přidání termínu zkoušky od kolegů, automatické připojení VUT wifi, upozornění na přicházející akci rozvrhu, hlídání deadlinů u závěrečných prací jako je například zadání komise, posudku, známek atd.
- **Zobrazení, kdo je v práci**, a navigace k dané osobě (se znalostí, které dveře jsou otevřené).

- Rozpoznání podle GPS, že se uživatel nachází na pracovišti a **automatické spuštění docházky**.
- **Práce se studenty** – diskuzní fóra, hodnocení výuky, vytváření anket a testů.
- **Čtečka QR kódů** u místností ke zjištění rozvrhu.
- **Přihlášení podle otisku prstu.**
- **Zadávání docházky studentům** k zjednodušení administrativy.



Obrázek 4.2: Odpovědi zaměstnanců na otázku „Které funkce by měla mobilní aplikace dle vás mít?“

Je zřejmé, že není možné všem vyjít vstříc. Bude potřeba udělat kompromis. Největší skupinou zaměstnanců jsou učitelé, tedy v první fázi vývoje aplikace bude cíleno na ně. Pohybují se nejčastěji kolem studentů (před, během či po přednášce, v průběhu konzultací), v kanceláři, na obědě nebo v terénu (různá sportoviště pro učitele tělocviku nebo výstavy pro pracovníky FaVu). Nemusí ale mít počítač stále u sebe a telefon jim může značně ulehčit práci.

Na základě znalosti cílové skupiny uživatelů, kontextu použití aplikace, statistik nejpoužívanějších modulů a využití nativních funkcí telefonů byly zvoleny hlavní funkce budoucí mobilní aplikace pro zaměstnance.

- **Vyhledávání** uživatelů, předmětů, a místností. Vyhledávání bude možné pomocí vložení identifikačního řetězce nebo pomocí QR kódu.
- **Osobní rozvrh**. Rozvrh bude upozorňovat na nadcházející akce a bude možné si u každého rozvrhového okna zobrazit seznam studentů.

- **Přihlášení pomocí biometrie**. Místo zadávání hesla, bude stačit otisk prstu k přihlášení do aplikace.
- **Zobrazení předmětů** vyučujících se zobrazením nejbližších termínů a aktualit.
- **Evidence docházky studentů** na výuku. Učitelé již nebudou muset evidovat ručně docházku studentů a archivovat si množství papírů spojené s touto administrativou. Docházka je důležitá hlavně pro povinné typy rozvrhových jednotek a tedy je často spojena se zápočtem. Může být využita technologie Bluetooth k urychlení zadávání přítomnosti.
- **Obědy v menzách**. Každý má právo na polední pauzu a aplikace jim umožní porovnat nabídku jídel ve školních jídelnách. Využití GPS k zobrazení vzdálenost k menzám.
- **VUT zprávy**. Náhled VUT zpráv a možnost rychle odpovědět na zprávu od kolegů nebo studentů. Přijetí VUT zprávy a obdržení notifikace.
- **Evidence docházky u zaměstnanců**. Využití GPS k zjištění aktuální polohy a nabídnutí zahájení práce, pokud práce již neprobíhá a uživatel se nachází na pracovišti.
- Rychlý přístup k **výplatě**, **závěrečným pracím**, **VUT mobilu**.

4.2 Výběr technologie

Při výběru technologie bylo nejprve zohledněno kritérium jednotné implementace na více platforem, proto byly vyloučeny nativní technologie. Navíc nebude aplikace vykonávat žádné náročné aplikace, proto výkon a optimalizace nejsou tak důležité. Důležitá bude práce s aplikací i v offline režimu tj. bez nutnosti připojení k internetu a snadné použití nativních funkcí telefonu (GPS, Bluetooth, fotoaparát). Proto byl vyloučen přístup webové aplikace. Pro svižnější reakce na uživatelské vstupy byl vyloučen i framework Ionic. Dále je třeba uvážit již existující aplikaci pro studenty Moje VUT vytvořenou v roce 2020 na základě bakalářské práce Ondreje Klinovskeho [22] a možnost budoucí integrace řešení z touto aplikací, která je psaná v frameworku React Native. Na základě popsaných skutečností jsem se rozhodla pro technologii React Native.

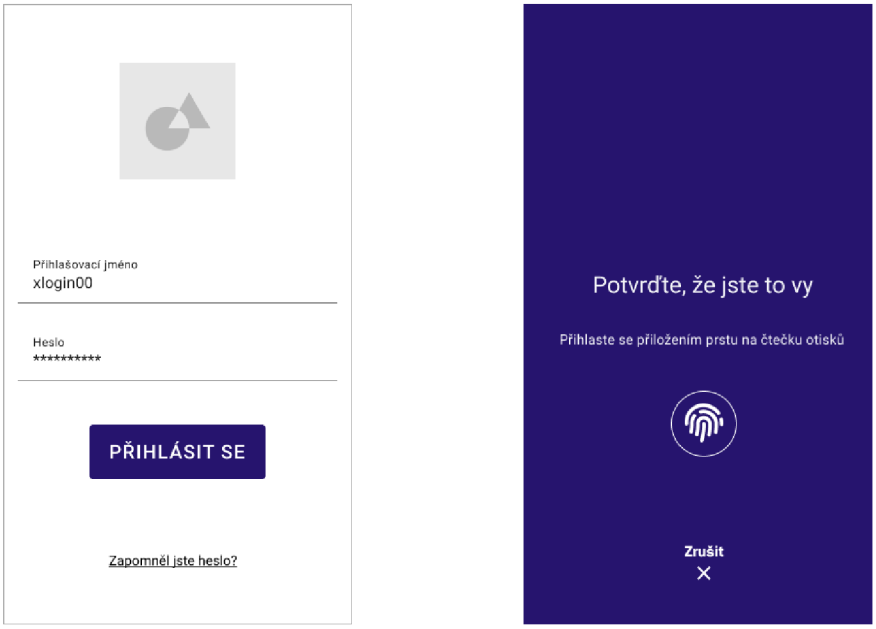
4.3 Uživatelské rozhraní

Dobré uživatelské rozhraní umožňuje uživateli pracovat s aplikací účelně a pohodlně. Pokud dodržuje předepsaná pravidla, pomáhá uživateli vytvářet správné stereotypy práce se systémem. Potom je dobré uživatelské rozhraní dobré i pro autory, jelikož nemusí psát rozsáhlé manuály k aplikacím. Uživatelům stačí pouze uplatit dobré návyky z ostatních aplikací.

Tato kapitola je věnována návrhu uživatelského rozhraní. Návrhy prošly dvěma fázemi. V první fázi byly vytvořeny černobíle wireframy, na kterých byly otestovány jednotlivé moduly a byly prodiskutovány navržené funkce. V druhé fázi byly již zařazeny připomínky do návrhů. Nejedná se o finální design aplikace. Důležité je pouze rozmístění prvků a funkčnost.

Přihlašovací obrazovka

Po otevření aplikace se zobrazí standardní přihlašovací rozhraní se dvěma položkami pro jméno a heslo. Pod nimi se bude nacházet odkaz na stránku s instrukcemi, co dělat v případě zapomenutého hesla. Uživatel potvrdí přihlášení kliknutím na tlačítko *Přihlásit se*. V případě, že bude mít nastaveno přihlašování pomocí biometrie, zobrazí se uživateli pouze obrazovka k verifikaci podle otisku prstu.



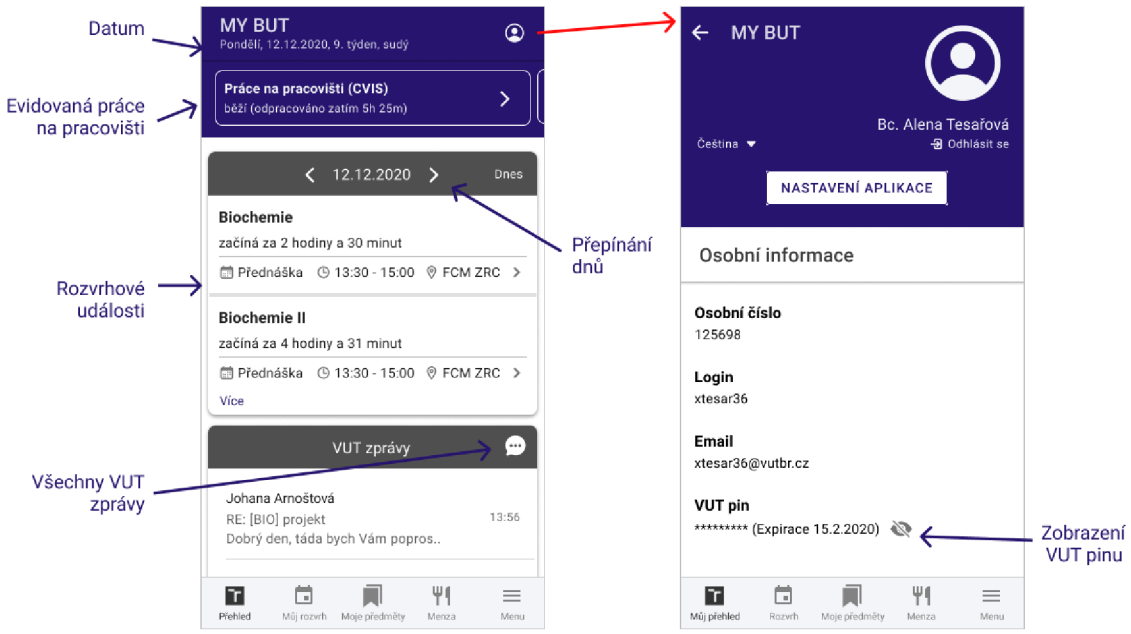
Obrázek 4.3: Obrazovky přihlášení

Hlavní obrazovka

Po přihlášení bude uživatel přesměrován na hlavní obrazovku. Hlavní strana je výchozí místo aplikace. Musí být proto přehledná, aby se v ní uživatel mohl rychle zorientovat. Zároveň by měla obsahovat nejdůležitější funkce pro uživatele. Obrazovku lze rozdělit na tři části – hlavička, tělo a spodní navigace. V hlavičce se bude nacházet dnešní datum, jelikož nejen studenti, ale hlavně učitelé, potřebují vědět, o jaký se jedná týden pro lepší orientaci během semestru. Dále bude umožněno se rychle dostat na profil kliknutím na profilovou fotku v pravém horním rohu, aby měl uživatel rychle dostupný VUT pin. Pod hlavičkou se bude zobrazovat počet dosud odpracovaných hodin za daný den. Zaměstnanec bude mít takto vždy přehled, kolik hodin již odpracoval a kdy je čas si dát přestávku.

V těle okna budou dvě karty. První obsahuje kalendář s nejbližšími rozvrhovými akcemi s možností expandovaní karty. V kalendáři bude možné se pohybovat do minulosti i do přítomnosti. Vždy bude ale možné se vrátit rychle na dnešní datum po kliknutí na tlačítko Dnes. Vždy proto bude mít učitel přehled o aktuální výuce ve svých předmětech a bude moci se vrátit zpětně na minulé hodiny, aby si zkontroloval nebo doplnil účast studentů na hodině. V druhé kartě se budou zobrazovat VUT zprávy seřazené podle data. Neunikne mu proto žádná důležitá zpráva. Ikonka v pravém horním rohu přesměruje uživatele na všechny VUT zprávy s možností v nich vyhledávat.

Hlavní obrazovka byla navržena jako výchozí bod, ze kterého se uživatel dostane do ostatních modulů, pokud mu nebudou stačit informace, které právě vidí. Nebylo žádoucí přidávat další karty na hlavní obrazovku, jelikož by nebyly na první pohled vidět a zobrazení by mohlo působit složitě.



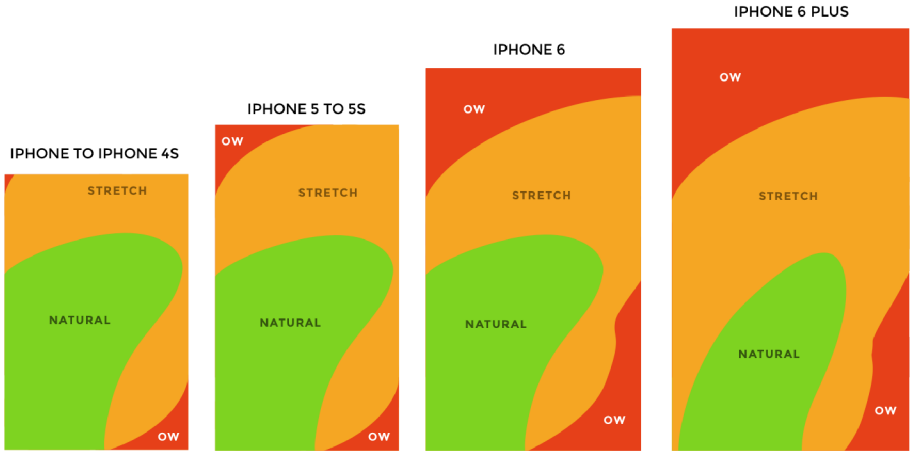
Obrázek 4.4: Obrazovka hlavní strany

Navigace

Pokud obsahuje aplikace více obrazovek, mezi kterými se potřebuje uživatel rychle pohybovat, navigace je nezbytný prvek. Při návrhu navigace byly brány v úvahu dva typy – boční menu a horní menu (tapbar). Boční menu označované někdy jako hamburger menu umožňuje v sobě zahrnout hodně navigačních odkazů, jelikož je ho možné vysunout podél celé obrazovky po kliknutí na ikonu hamburger a tak zabírá téměř celou plochu obrazovky. Jelikož se většinou ikona nachází v levém horním roku, není tak jednoduché dosáhnout prsem na tento prvek. Obrázek 4.5 ukazuje nejlépe dostupné části obrazovky pro uživatele držící telefon v pravé ruce [30].

Tapbar navigace pochází z designu pro stolní počítače. Navigační prvky jsou formou ikon zobrazené vedle sebe přímo na obrazovce, tedy na rozdíl od bočního menu není navigace skrytá. Může se nacházet buď v horní, nebo spodní části obrazovky. Dovoluje to tedy uživateli se lépe zorientovat v aplikaci, jelikož jednotlivé ikony představují většinou hlavní části aplikace. Největší nevýhodou je omezení počtu ikon na 3 až 5. Větší množství by mohlo naopak způsobit nepřehlednost, jelikož by ikony nebyly dobře vidět. [8, 31]

Při výběru navigace jsem se rozhodla použít jako hlavní navigaci tapbar navigaci, kterou jsem umístila do dolní části obrazovky, jak lze vidět na obrázku 4.4. Tapbar sice zabere více místa, avšak v současnosti se čím dál více vyrábí telefony s většími displeji, tak by s nedostatkem místa neměl být problém. Vybrala jsem pak 5 nejdůležitějších modulů na základě předchozí analýzy, které byly umístěny do navigace. Ostatní méně důležité moduly byly umístěny do menu jako jedna z položek navigace.



Obrázek 4.5: Porovnání dostupností oblastí displeje pro různě velké obrazovky telefonů. Převzato z [30].

- Rozvrh – osobní rozvrh přihlášeného uživatele,
- Moje předměty – seznam předmětů, které přihlášený uživatel vyučuje,
- Menza – seznam menz VUT společně s aktuálním jídelním lístkem,
- Menu – sekundární moduly aplikace (vyhledávání, mapa atd.).

Osobní rozvrh

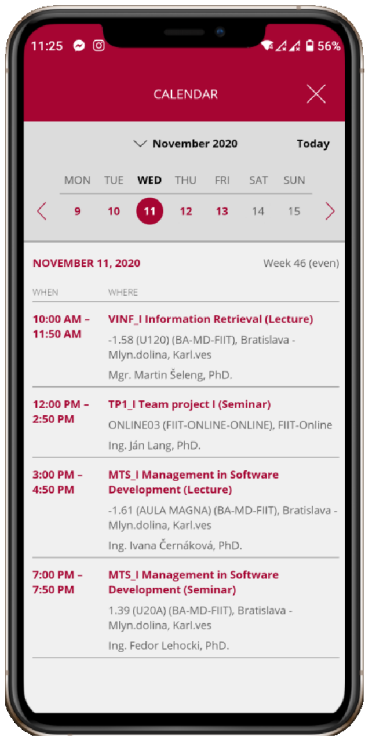
Společně s modelem Předměty by se mělo jednat o nejdůležitější obrazovku v aplikaci. Na obrázku 4.6 jsou zobrazena řešení rozvrhů od různých společností.

První variantu kalendáře zvolila firma IS4U, s.r.o., která vyvíjí informační systémy pro vysoké školy. Jedná se o jednoduché řešení. Po kliknutí na daný den se zobrazí seznam chronologicky seřazených událostí. Není zde ale vidět žádný přehled celého týdne a vizuálně není možné rychle zjistit, kolik času má student na přestávku mezi událostmi nebo jestli má student daný den kolizi. Navíc, pokud již skončil semestr, není možné se podívat do historie na výuku (pokud nemá student koupenou PREMIUM verzi).

Druhou variantu zobrazení rozvrhu si vybrala Univerzita Palackého v Olomouci. Rozhodli se pro seznam po sobě jdoucích událostí, který doplnili informací o délce přestávky mezi vyučováními. Řešení je přehledné, ale v momentě, kdy má uživatel více vyučovacích hodin za den, seznam je velmi dlouhý a může snížit přehlednost. Navíc není možné přepínat jednoduše mezi jednotlivými měsíci či dny. Chybí týdenní zobrazení.

Třetí aplikace Moje VUT má řešeno zobrazení velmi přehledně. Lze přepínat týdny semestru a navíc rozvrh podporuje i denní zobrazení. Nevýhodou je omezení zobrazení kalendáře pouze na 13 vyučovacích týdnů semestru.

Jako čtvrté řešení bylo přidáno propracované zobrazení od firmy Google. Podporuje všechny možnosti zobrazení kalendáře (denní, týdenní, měsíční, seznam) a přesun mezi dny či týdny je realizovaný potáhnutím prstem po obrazovce. Učitelé ale často neučí o víkendech a nemají výuku v 1 hodinu ráno. Tyto časové úseky by byly v navrhované aplikaci zbytečné, proto se nejedná o ideální řešení. Celkově je ale řešení velmi inspirativní.



(a) Řešení zobrazení rozvrhu v aplikaci My College od firmy IS4U

(b) Rozvrh používající na Univerzitě Palackého v Olomouci

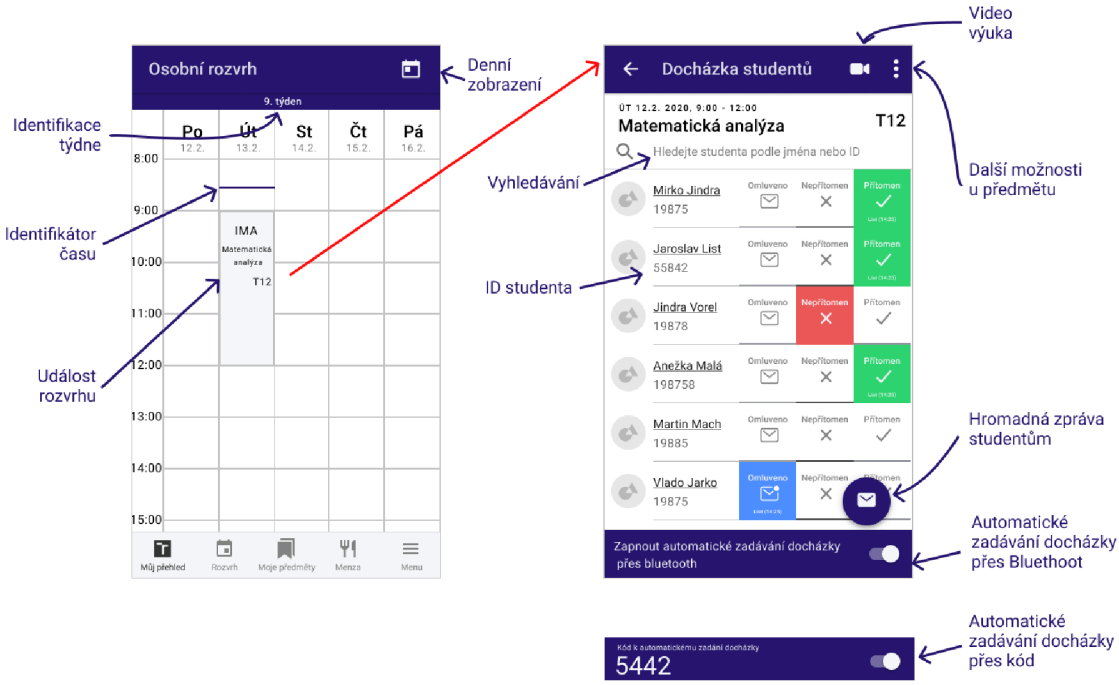
(c) Rozvrh v aplikaci Moje VUT

(d) Google kalendář

Obrázek 4.6: Existující řešení zobrazení rozvrhu v různých aplikacích.

Na základě již existujících řešení jsem navrhla rozvrh, který se nejvíce podobá řešení aplikace Moje VUT, jak lze vidět na obrázku 4.7. Rozdílem bude možnost rychlého posouvání mezi týdny horizontálním potáhnutím obrazovky místo složitého přepínání omezeného počtu týdnů. Orientace se zlepší i vysunutím kalendáře po kliknutí na číslo týdne. Vysunutí by mělo být animované, aby uživatel věděl, co se právě děje a nebyl zmatený, kam zmizel rozvrh.

Jednolité vyučovací bloky budou mít velikost v závislosti na délce rozvrhové jednotky. V případě kolize bude šířka zmenšena na polovinu. Uživatel bude mít k dispozici dva režimy zobrazení rozvrhu. V týdenním bude vidět celý týden a bude přesně vědět, kdy má pauzy mezi vyučováními. V denním zobrazení uvidí pouze konkrétní den. Bloky budou obsahovat pouze důležité informace – název, zkratka a místnost předmětu. K rychlejšímu přístupu učitele ke studentům se vyučujícímu zobrazí seznam studentů hned po kliknutí na daný blok.



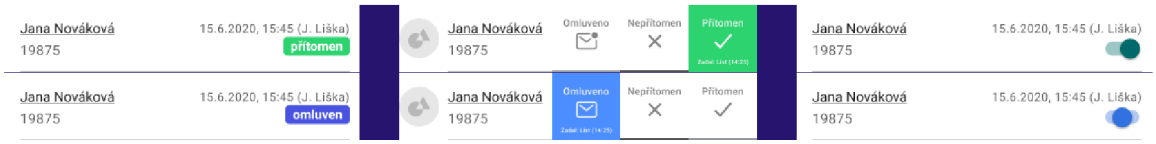
Obrázek 4.7: Návrh obrazovky rozvrhu společně se zadáváním docházky.

V případě potřeby si může učitel zaznamenat **docházku** studentů na výuku. Stavy docházky jsou 4 – přítomen, nepřítomen, omluven a nezadáno. Bude možné přidat poznámku, pokud by si chtěl vyučující poznačit bodové hodnocení nebo vlastní komentář ke studentovi.

Zaznamenání docházky bude možné buď ručně nebo automaticky. U ručního zadávání je potřeba, aby bylo rychlé. Může totiž nastat situace, kdy hodnotí vyučující větší množství studentů a není cílem strávit celou hodinu pouhým zadáváním docházky. Navíc je důležité, aby učitel při zadávání vždy věděl, o jakého se jedná studenta a měl možnost stav vždy změnit. Bylo analyzováno více možností, které jsou zobrazené na obrázku 4.8. První způsob je vybírání stavu ze seznamu možností, který se zobrazí po kliknutí na štítek se stavem. I když je zadání intuitivní, je potřeba vykonat dvě kliknutí (napřed zvolit studenta a pak vybrat stav), proto se nejedná o nejrychlejší řešení. V uživatelskému testování navíc nebylo správně pochopeno datum a čas nacházející se nad stavem.

Druhou možností je kliknutí přímo na stav u studenta. Řešení je rychlé, neztrácí se informace o studentovi a údaje o datumu a čase jsou pochopitelné, jelikož se nachází pohromadě hned u stavu. Nevýhodou je velikost tlačítek, jelikož zabírají přes polovinu šířky displeje.

Poslední testovanou možností byl přepínač se třemi stavy, na který jsou uživatelé zvyklý ve spojení se zapínáním a vypínáním určité funkce na mobilním zařízení. Bohužel neexistuje přepínač, který by umožňoval přepnutí tří stavů, a proto by v tomto případě byla potřeba vymyslet vlastní komponentu, jak lze vidět na třetím obrázku 4.8. Nové komponenty ale mohou způsobit zmatení uživatele a ztrátu jeho důvěry k modulu.



Obrázek 4.8: Různé možnosti zadávání docházky. Zobrazení stavu přítomen a stavu omluveno u jednotlivých možností.

Po uživatelském testování byla nakonec vybrána druhá možnost, jelikož připadá uživatelům nejintuitivnější. Pokud avšak je na hodině více než 20 studentů, manuální zadávání může být zdlouhavé. Navrhla jsem proto dvě možnosti automatického zadávání docházky.

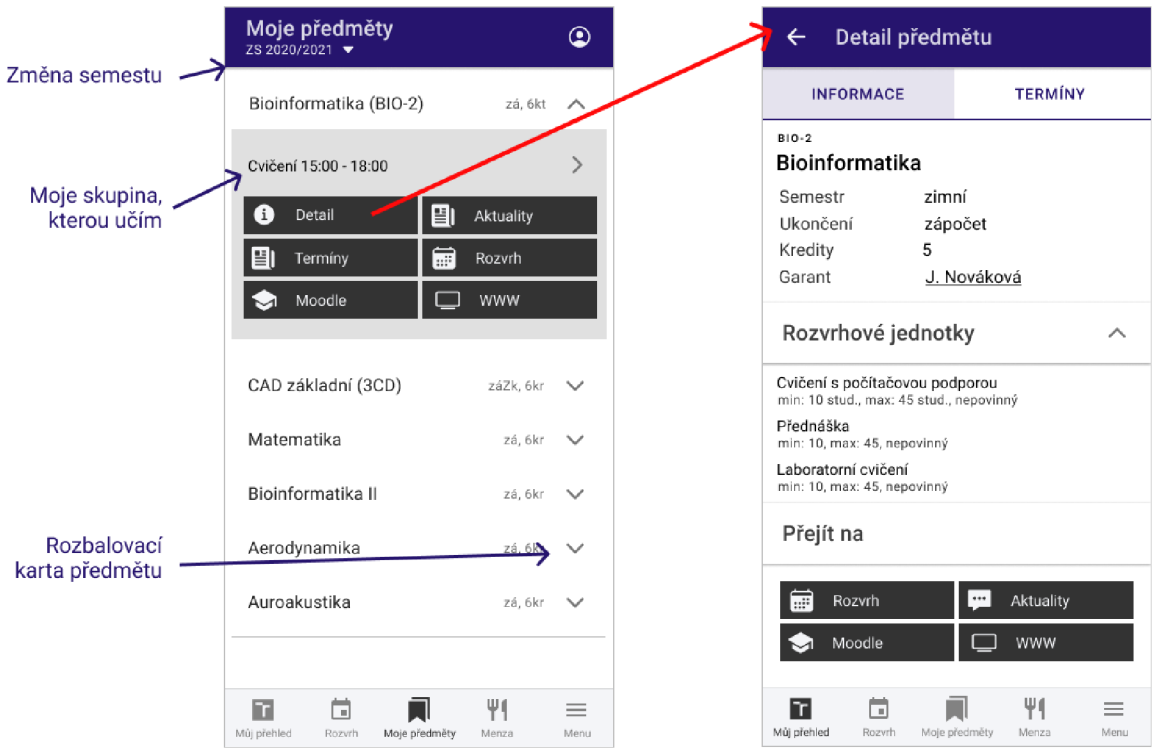
První možností bude automatické zadávání pomocí technologie bluetooth. Pokud učitel povolí tuto možnost, zobrazí se studentům v aplikaci Moje VUT upozornění, aby si zadali docházku. Technologie bluetooth pak automatiky ověří, jestli se student doopravdy nachází na hodině.

Druhou možností je zadávání docházky přes kód. Učitel sdělí studentům kód, který se mu vygeneruje aplikace. Studenti pak přes aplikaci moje VUT zadají kód do systému a nastaví se jim automaticky přítomnost. Tato varianta je jednodušší z hlediska možného podvodu, jelikož studenti mohou poslat kód spolužákům. Obě varianty však umožní zautomatizování zadávání činnosti.

Další pomocnou funkcí je napsání zprávy všem studentům dané vyučovací hodiny. Přes web i přes Apollo byla potřeba se složitě dostávat přes více stránek k této funkci. V aplikaci stačí kliknout na ikonu s poštovní obálkou v pravém dolním rohu. Každá hodina má vygenerovaný odkaz k online hodině do Microsoft Teams. Učitelé k ní budou mít rychlý přístup přes kliknutí na ikonu kamery v horní liště. Další možnosti u výuky (jako je například přístup ke kartě předmětu) se zobrazí po kliknutí na ikonu s třemi tečkami.

Moje předměty

Druhá z nejdůležitějších obrazovek jsou vyučované předměty učitele. Navrhla jsem zobrazení jako seznam předmětů, jak lze vidět na obrázku 4.9. Rozbalená karta předmětu obsahuje rychlou volbu k nejdůležitějším funkcím, aby nemusel učitel nic dlouze hledat. Každý učitel má právo si zobrazit pouze ty studenty, které učí. Učit může učit více skupin, proto hned pod názvem předmětu se bude nacházet seznam skupin, na které má učitel práva. K předmětu se dále vážou jeho aktuality, rozvrh, termíny, moodle a webové stránky. Mezi jednotlivými semestry se bude moci jednoduše přepínat pomocí šipek nebo pomocí přetáhnutí prstem po obrazovce.



Obrázek 4.9: Obrazovka mých předmětů se zobrazením detailních informací o předmětu

4.4 Použití biometrických technologií

Jak bylo zmíněno v kapitole 2.5, zabezpečení pomocí biometrie nabízí větší míru komfortu, nemůže být zapomenuto a zloděje může odradit k podvržení. Chránit je potřeba jak naše soukromé údaje, tak i pracovní záležitosti, které by mohl někdo vidět a zneužít. Univerzitní informační systém musí respektovat zásady *General Data Protection Regulation* (GDPR) a nemůže si dovolit žádný únik dat, proto je otázka bezpečnosti v mobilní aplikaci pro zaměstnance zásadní.

Výběr knihovny

V rámci návrhu byly studované existující moduly, kde by bylo vhodné využít biometrického zabezpečení. Na OS android i iOS je zabezpečení pomocí biometrie řešeno různými způsoby. Pro vývojáře to znamená, že se musí ujistit, jestli vybraná knihovna v sobě obsahuje podporu pro oba systémy. V této práci byla vybrána open-source knihovna **RNSInfo**. Pro android je zde naimplementovaná podpora pro nativní třídu *FingerPrintManager* starající se o autentizaci a šifrování dat pomocí otisku prstu. Citlivé údaje se ukládají do zabezpečeného systému nazývaný Android Keystore. Ukládání dat probíhá následujícím způsobem. [12]

1. Aplikace požádá Android KeyStore o tajný klíč (SecretKey).

2. Android Keystore vytvoří tajný klíč na zabezpečeném místě nazývající se *Strongbox* nebo *Trusted Execution Environment* (TEE).
3. Keystore vrátí aplikaci alias k tajnému klíči. Pouze keystore umí mapovat tento alias na skutečný tajný klíč.
4. Pokud potřebuje aplikace provést zakódování dat, musí o to požádat Keystore.
5. Keystore vezme data a alias a vrátí již zakódovaná data (nazývaná *ciphertext*).
6. Pokud si přeje aplikace data dekódovat, Keystore vezme *ciphertext* a alias a vrací dekódovaná data.

Pokud je v konfiguraci přidáno zašifrování dat pomocí biometrie (například otisku prstu), keystore odmítne dekódování dat bez přítomnosti uživatele, který by se ověřil otiskem prstu. Ověření probíhá ve dvou krocích.

1. Uživatel se autorizuje pomocí biometrie jako je například otisk prstu přiložením prstu na senzor.
2. Systém vygeneruje šablonu v zabezpečeném místě a porovná jej s již vloženou šablonou. Pokud se šablony shodují, vrací kladnou odpověď. V opačném případě zápornou.

Na OS iOS se k ukládání citlivých dat používá *Keychain Access control*, který specifikuje, jak mají být data zabezpečena. Umožňuje nastavení autentizace pomocí hesla nebo otisku prstu.

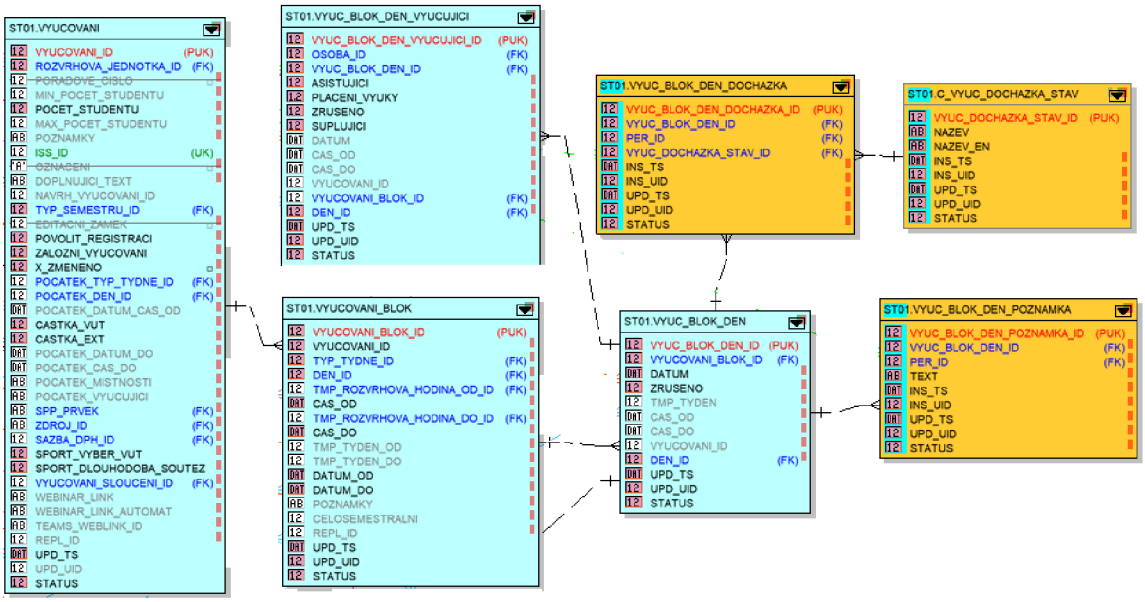
Vybrané citlivé moduly v IS

V kapitole 4.1 byly popsány vybrané nejpoužívanější moduly informačního systému. Většina modulů je velmi citlivá na ztrátu dat. Jedná se například o VUT pin, osobní číslo, modul s výplatami, fotky zaměstnanců, informace o studentech (jejich jméno, id, kontakt, fotka), proto je třeba řádně promyslet, jak bude uživatel verifikován. První způsob, který se nabízí, je verifikace uživatele na základě jeho otisku prstu nebo rozpoznání podle obličeje (podle podpory zařízení uživatele). Lidé jsou na tento způsob zvyklí a zvedne se u nich důvěra aplikaci používat.

Na základě analýzy existujících modulů byly navrženy situace, kdy by bylo vhodné ověřit uživatele v aplikaci.

- **Přihlašování** je jedním z prvních modulů, kde je vhodné biometrii využít. Uživatel si navíc často nepamatuje dlouhá vygenerovaná hesla, a proto mu automatické přihlášení může ušetřit spoustu času.
- **Použití biometrie při přechodu aplikace do popředí**. Po přihlášení aplikace obdrží přihlašovací token, který má jistou platnost (nastavena na jeden týden). Po dobu platnosti se není potřeba znovu přihlašovat do aplikace k získání dat. Je důležité proto přidat možnost ověření uživatele při každém přechodu aplikace do popředí. Nemůže se pak stát, že by se nepovolaná osoba dostala k citlivým údajům.
- **Zadávání známek a docházky** studentů do systému. V tomto případě bude biometrie sloužit ke dvěma účelům. Za prvé se bude jednat o verifikaci, jestli se jedná o správnou osobu. Za druhé bude další obrazovka sloužit jako potvrzení, že si je uživatel skutečně jistý vložením údaje do systému.

- **Poslání VUT zprávy**. Nikdo si nepřeje, aby mu cizí lidé pod jeho jménem posílali cizí zprávy. Proto je ověření důležité.
- **Podepisování dokumentů**. Klasická metoda podpisů dnes již není efektivní a čím dál více jdou do popředí elektronické podpisy. Podepisování dokumentů pomocí biometrie bude v aplikaci využito spíše pracovníky vedoucích pozic. Tato možnost přispěje k celkovému zrychlení administrativy a ulehčení papírování.
- **Přístup k výplatě**. Ve většině případů není žádoucí sdílet údaje o platu. Jedná se o velmi citlivý a osobní údaj.
- **Zobrazení VUT pinu**. Pomocí VUT pinu se můžeme přihlásit na wifi, vytisknout si nějaký dokument nebo se přes něj ověřovat do jiných aplikací. Údaj by měl být minimálně skrytý nebo přístupný pouze po biometrickém ověření.
- **Zobrazení studentů.** U studentů se nacházejí jejich jméno, fotka, osobní číslo, známky a další informace. Tyto seznamy může vidět pouze vyučující daného předmětu a neměl by je mít právo vidět nikdo jiný.



Obrázek 4.10: Nově přidané tabulky s docházkou jsou označeny žlutě

4.5 Návrh databáze

Aplikace je napojena na centrální databázi VUT obsahující velké množství schémat a tabulek. Aplikace k databázi přistupuje přes VUT API. V rámci aplikace pro zaměstnance bylo potřeba vytvořit a navrhnout nové struktury pro docházku, která v systému chyběla.

Docházka se bude zadávat na konkrétní vyučující hodině (např. Matematika, cvičení, sk A, 10.5.2021, 12:00 – 13:00), tato konkrétní hodina se nachází v tabulce *VYUC_BLOK_DEN*. Je navázána na vyučovací blok značící konkrétní cvičení nebo přednášku a na vyučující, kteří učí na dané hodině. V rámci návrhu byly uvažovány dvě varianty vkládání poznámky. První variantou bylo vložení sloupce poznámka přímo k docházce. V tomto případě by pak bylo

potřeba zadávat poznámku společně s docházkou. Znamenalo by to četné duplikování dat při každé změně docházky, jelikož by se poznámka musela zkopírovat ke každému nově vloženém záznamu. Druhým řešení bylo přidán poznámky samostatně do tabulky. V takovém případě je poznámka naprosto nezávislá od docházky. Nakonec byl zvolen druhý návrh, aby učitel nebyl nucen k docházce zadávat poznámku, ale mohl si ji zadat naprosto nezávisle u studenta na hodině.

Výsledné ERD lze vidět na obrázku 4.10. Byly přidány 3 tabulky označené žlutou barvou.

- *VYUC_BLOK_DEN_DOCHAZKA* – tabulka obsahující všechny stavy docházky studentů na hodině (VYUC_BLOK_DEN) s informací, kdo docházku vložil,
- *VYUC_DOCHAZKA_STAV* – číselník stavů docházky obsahující stavy: přítomen, nepřítomen, omluven, nezadáno,
- *VYUC_BLOK_DEN_POZNAMKA* – tabulka s poznámkami u studentů na hodině.

Kapitola 5

Implementace

Podle návrhu představeného v předchozí kapitole byla implementovaná aplikace s názvem *VUT Empee*. Vychází z již existující aplikace pro studenty *Moje VUT* a má za cíl zjednodušit práci zaměstnancům u každodenních činností. V této kapitole je popsaná struktura projektu, je zde vysvětleno ukládání dat, komunikace se serverem a obsahuje detailní popis implementace vybraných modulů (sekce 5.4). Na konci, v sekci 5.5 je ukázáno aktuální využití biometrického zabezpečení v aplikaci.

5.1 Struktura projektu

Projekt má typickou strukturu používanou v React Native (RN) projektech.

- `android` ..... zdrojové kódy pro sestavení aplikace pro OS Android
- `ios` ..... adresář s zdrojovými kódy pro sestavení aplikace pro OS iOS
- `app` ..... RN projekt
  - `assets` ..... ikony, fonty, obrázky atd.
  - `components` ..... React Native komponenty (např. VUTButton, VutInput atd.)
  - `hooks` . vlastní hooky (např. hook useStores umožňující přístup k MobX úložištím)
  - `i18n` ..... definování překladů pro český a anglický jazyk
  - `models` ..... definice datových struktur
  - `modules` ..... moduly pro práci s interním úložištěm
  - `parsers` ..... parsování odpovědí ze serveru a ukládání do vnitřních struktur
  - `screens` ..... React komponenty představující jednotlivé obrazovky aplikace
  - `services` ..... třídy pro práci s backendem
  - `styles` ..... globální styly aplikace
  - `stores` ..... MobX úložiště dat
  - `utils` ..... pomocné funkce pro práci s rozvrhy, časem, transformaci dat atd.
- `index.js` ..... vstupní soubor aplikace
- `app.json` ..... konfigurační soubor RN aplikace
- `package.json` ..... metadata projektu (např. závislosti projektu)
- `tsconfig` ..... konfigurace jazyka TypeScript
- `node_modules` ..... adresář modulů generovaný pomocí nástroje npm
- `scripts` ..... skripty na testování a automatické vytvoření APK

Složka **android** obsahuje konfigurační soubory *Gradle* pro sestavení APK souborů nebo zdrojové soubory napsané v jazyku Java. S těmito adresáři se pracuje při konfiguracemi výsledného APK souboru nebo při vytváření nativních komponent, pro které RN nenabízí podporu (například widgety).

Ve složce **iOS** se nachází zdrojové soubory jazyku Objective-C nebo Swift a konfigurační soubory pro XCode. Podobně jako u OS Android se v něm konfiguruje sestavení aplikace a integrace knihoven třetích stran.

5.2 Ukládání dat

Základem aplikace tvoří MobX úložiště nazývané Stores. Jejich hlavní úlohou je poskytovat data získaná ze serveru a uchovávat stav aplikace. Každé úložiště je zodpovědné za jednu doménu aplikace (např. úložiště pro přihlášení, menzy nebo profil osoby). Platí, že po dobu běhu aplikace musí existovat pouze jedna instance třídy. V aplikace je používáno více úložišť, a proto byl implementován RootStore, který vytváří instance všech úložišť a poskytuje na ně referenci.

- **UIStore** slouží ke správě uživatelského rozhraní.
- **AttendanceStore** slouží ke správě docházky studentů na hodině.
- **ScheduleStore** obsahuje data, operace k práci se všemi typy rozvrhů (osob, předmětů a místností).
- **TeachStore** spravuje třídní knihu učitele se všemi jeho předměty a studenty.
- **SearchStore** slouží k podpoře vyhledávání osob.
- **ProfileStore** spravuje profily osob.

Všechny úložiště používají pro získání dat třídu *VutService*, která se stará o komunikaci se serverem. Všechny data získaná ze serveru mají stejnou strukturu:

1. **data** – samotná získaná data,
2. **error** – chyba při čtení dat ze serveru,
3. **state** – stav dat (na začátku obsahuje hodnotu *INITIAL*, pak načtením se změní na *LOADING* a po načtení je nastaveno buď na *SUCCESS* nebo *ERROR* podle výsledku dotazu)

V aplikaci je důležité si udržovat stav získaných dat, aby bylo možné případně uživateli zobrazit ikonu označující načítání dat, aby věděl, že musí počkat na dokončení operace. Časový limit k získání dat je nastavený na serveru na 5 minut. Všechna data v úložištích představují pozorovatelné objekty. V případě jejich změny dojde k překreslení té React komponenty, která tyto data používá a zobrazuje je.

Data se nachází v úložišti, pouze pokud je aplikace spuštěná. Aby nebylo potřeba stahovat data do úložiště při každém zapnutí aplikace, je důležité si data ukládat přímo do zařízení. Pak se nemůže stát, že přijdeme o data při odpojení k internetu.

Na ukládání dat do zařízení je použitá knihovna *AsyncStorage*[1], která poskytuje rozhraní ke vkládání dat a k získání dat z úložiště. Obě funkce vracejí typ *Promise*, což je třída

[1]Detail knihovny s možností stažení se nachází na stránkách: https://github.com/react-native-community/async-storage

na práci s asynchronními událostmi. Signatura funkcí ukazuje, že se data budou ukládat v textové formě, proto je nutné před uložením data serializovat pomocí `JSON.stringify` a po načítání deserializovat funkcí `JSON.parse`.

5.3 Komunikace se serverem

Aplikace komunikuje se serverem pomocí protokolu *WebSocket* za účelem získání dat. *Web socket* je komunikační protokol pracující na základě protokolu TCP (Transmission Control Protocol), který poskytuje point-to-point komunikaci mezi serverem a klientem. Na rozdíl od protokolu REST (Representational State Transfer) si *WebSocket* udržuje spojení mezi klientem a serverem po celou dobu komunikace. Hlavní výhodou je snížení režie na straně klienta a serveru, jelikož je spojení potřebné vykonat pouze jednou. Další výhody *WebSocketů* jsou [28]:

- Obousměrná komunikace. Komunikace může probíhat od klienta i serveru nezávisle na sobě.
- Závisí pouze na IP adrese a čísle portu (na rozdíl od REST, které je založené na posílání HTTP metod k posílání dat).
- Možné využití k real-time aplikacím.
- Udržuje si stav po celou dobu spojení.
- Nižší cena komunikace.
- Použití HTTP pouze při inicializačním spojení.

V jazyku JavaScript je protokol *WebSocket* zpřístupněný přes třídu *WebSoket*. Tato třída obsahuje 4 typy posluchačů: *onopen, onmessage, onerror* a *onclose*. **Onopen** se použije při otevření spojení, **onmessage** při přijetí zprávy. Tyto zprávy si může programátor nastavit podle sebe. Spojení se vytvoří klíčovým slovem *new* a jako parametr se vezme URL adresa serveru s portem, ke kterému se má připojit a název použitého protokolu. Pro poslání zprávy slouží metoda *send*, která posílá data v textové formě, takže je vždy potřeba data serializovat pomocí `JSON.stringify`. Spojení se ukončení zavoláním metody **close**. V případě chyby spojení se volá metoda **onerror**.

Thor

Aplikace se spojuje přímo se serverem VUT, kterému se říká *Thor*. Ve skutečnosti se jedná o dva servery. Spojení probíhá tak, že se napřed zkusí první a pokud nedojde k ustanovení spojení, tak se zkusí připojit na druhý. Server se skládá z jádra a modulů, kde jádro je zodpovědné za monitorování modulů. V případě, že by jeden z modulů neodpovídal na požadavky, nastartuje ho jádro znovu. Každý požadavek má nastavený časový limit na 5 minut. Po této době přichází na klienta chybová hláška o vyčerpaném časovém limitu. Požadavky se umisťují do fronty a jejich vyřízení je spravováno plánovačem.

Podle připojeného portu pracuje *Thor* s různými centrálními databázemi. K dispozici jsou tři:

1. **CDBX** – Produkční databáze VUT.
2. **CISD** – Testovací databáze. Přepisuje se 1x týdně.

Modul	Funkce	Popis
http	OsobniRozvrh_Teacher	Osobní rozvrh učitele.
http	RozvrhMistnost_Teacher	Rozvrh místnosti.
http	RozvrhOsob_Teacher	Osobní rozvrh konkrétního učitele.
http	RozvrhPredmet_Teacher	Rozvrh předmětu.
query	CREATE:Zam_PredDochazka	Vložení docházky studentů na hodině.
query	MODIFY:Zam_PredDochazka	Modifikace stavu docházky studentů na hodině.
query	Zam_PredmetDochazka	Číselník stavů s typy docházky.
query	NajdiOsobu_dle_QR	Vyhledá 10 nejbližších osob podle parametru. Hledá QR kódu na průkazu.
query	Zam_predmety	Seznam předmětů patřící k zadanému vyučujícímu.
query	Zam_predmety_studenti	Seznam studentů patřící k danému předmětu.
query	Zam_predmety_vyucovani	Seznam rozvrhových jednotek učitele (cvičení, přednáška) patřící k předmětu.
query	Zam_studenti_detail_hodnoceni	Detail hodnocení studenta v předmětu.

Tabulka 5.1: Koncové body vytvořené na serveru pro aplikaci pro zaměstnance

3. **CISV** – Vývojová databáze. K přepisu dochází pouze jednou za čas, proto je vhodná pro programátora, který má zde vytvořená testovací data.

Každý požadavek je typu **VutRequest** a obsahuje: ID požadavku, *payload*, časovač a typ dat (např. typ *personDetail* k získání informací o osobě). *Payload* je speciální datová struktura obsahující název funkce, kterou chceme volat na serveru, seznam parametrů a informaci, o jaký se jedná typ modulu. Rozlišují se 3 typy modulů:

1. **Query** se používá pro většinu požadavků, které aplikace pošle na server. Odpovědi jsou získané přímo z databáze. V každém dotazu na server se provede kontrola práv na daný záznam SQL.

2. **Http** se používá pro získání dat z webu. Důvody jsou dva. Prvním je nejednotnost zdroje dat, jelikož ne všechna data se nachází v centrální databázi. Jedná se například o menzy. Druhý důvod je znovupoužitelnost již vytvořených dotazů do databáze. Může se stát, že se dotaz v budoucnosti změní a je zbytečné ho měnit na více místech.

3. **File** se týká všech souborů jako jsou například profilové fotky studentů a zaměstnanců.

Kvůli aplikaci pro zaměstnance bylo na serveru vytvořeno přes 10 nových koncových bodů, které jsou z aplikace volány. Ty nejdůležitější z nich jsou zobrazeny v tabulce 5.1. Jedná se většinou o funkce vracející data bez modifikace záznamů v databázi. Pouze v případě zadávání a modifikace docházky je potřeba data měnit. V tomto případě jako odpověď nepřijde počet vrácených záznamů, ale počet modifikovaných záznamů. V případě, že nedojde ke změně záznamu, tak učitel buď nemá práva zadávat docházku u daného studenta, došlo k chybě se spojením nebo k chybě na serveru.

Přihlášení

Aplikace se na server přihlašuje pomocí jména a hesla zadaného od uživatele. Na serveru databázová funkce ověří, jestli jsou zadané údaje správně a vytvoří se záznam v tabulce

přístupů. Informace společně s tokenem a ID sezení se pošle zpátky do aplikace, kde jsou údaje uložené do interního úložiště telefonu. Tento token se následně používá pro přihlášení na server po dobu jeho platnosti, která je aktuálně nastavena na jeden týden. Po vypršení je nutné opět zadat jméno a heslo. Tento způsob je důležitý z hlediska bezpečnosti, aby se nemohlo stát, že dojde k zfalšování přihlašovacího tokenu.

Ukládání hesla

Jelikož je nutné heslo zadávat při vypršení tokenu, bude nabídnuto uživatelům si heslo uložit do telefonu. Aby ale řešení bylo co nejbezpečnější, heslo bude uloženo pod otiskem prstu na zabezpečeném místě v telefonu pomocí knihovny *RNSInfo* (více v sekci 4.4). V tomto případě se k heslu může dostat pouze majitel telefonu a nikdo jiný.

5.4 Implementované moduly

Z analýzy nejpoužívanějších modulů byly k implementaci vybrány tři moduly, které budou popsány v následujících sekcích. Jedná se o modul s rozvrhem, třídní knihou a docházkou. U rozvrhu bylo uvažováno více variant, které budou představeny v sekci 5.4.1. Modul třídní knihy nabízí učiteli přehled všech jeho předmětů, u kterých si může zobrazit studenty a přidat si k nim informace na hodině v modulu docházky.

Uživatelské rozvrhaní je implementované pomocí knihovny React Native psané v jazyku TypeScript. Komponenty využívají knihovnu *MobX*, díky které reagují na změny pozorovaných dat překreslením uživatelského rozhraní.

5.4.1 Rozvrh

Rozvrh je jedna z nejklíčovějších a nejkomplexnějších obrazovek celé aplikace. Učitelé k němu měli hlavně přístup z Apolla, kde bylo zobrazení pomalé a na telefonu nepoužitelné. Původní verze Moje VUT měla jeden velký problém, kterým byla orientace v rozvrhu. K přepnutí týdne bylo potřeba jít do menu a až tam si vybrat týden semestru. Navíc zde bylo omezení pouze na 13 týdnů, proto nebylo možné vidět zkouškové období a nebo si zobrazit již uplynulý rozvrh z minulého semestru.

Na tento problém reaguje vytvořená první verze rozvrhu podle návrhu, kterou lze vidět na obrázku 5.1. Základem je vysouvací kalendář, který se rozbalí po kliknutí na šedý posuvník nacházející se pod dny. Řešení kalendáře bylo převzato od firmy *Wix* nabízející knihovnu *React Native Calendars*[2] jako open-source. Při implementaci bylo ale nalezeno několik problémů s knihovnou:

1. Problém různého počtu vyučovacích dnů v týdnu. Může se stát, že učitel má výuku i v neděli a podle toho by se mělo zobrazení přizpůsobit.

2. Kalendář používá velké množství komponent, které se vzájemně překrývají a používají animace způsobující zpomalení celého rozvrhu. Zpomalení dosahovalo kolem jedné až tří sekund, pro aplikaci nepřijatelné.

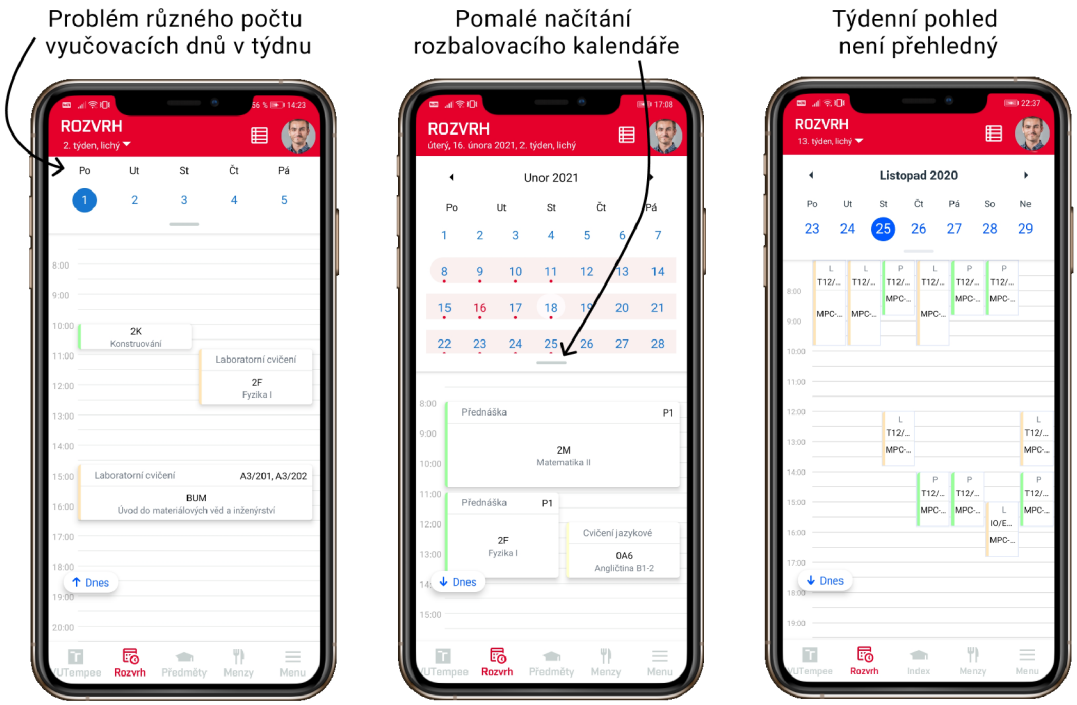
3. Týdenní zobrazení rozvrhu je velmi nepřehledné a navíc přesně nesedí zarovnání názvu dnů s časovými bloky (lze vidět na třetím obrázku 5.1).

[2] https://github.com/wix/react-native-calendars

4. Chybějící podpora pro dynamickou změnu barevného režimu. Aplikace *VUT empee* podporuje tmavý a světlý režim, proto je potřeba překreslit rozvrh k případě změny režimu.

5. Implementace v jazyku *JavaScript*, zatímco aplikace *VUT empee* je psaná v jazyku *TypeScript*.

6. Chybějící podpora posouvání se po týdnech a dnech horizontálním potáhnutím prstu po obrazovce.

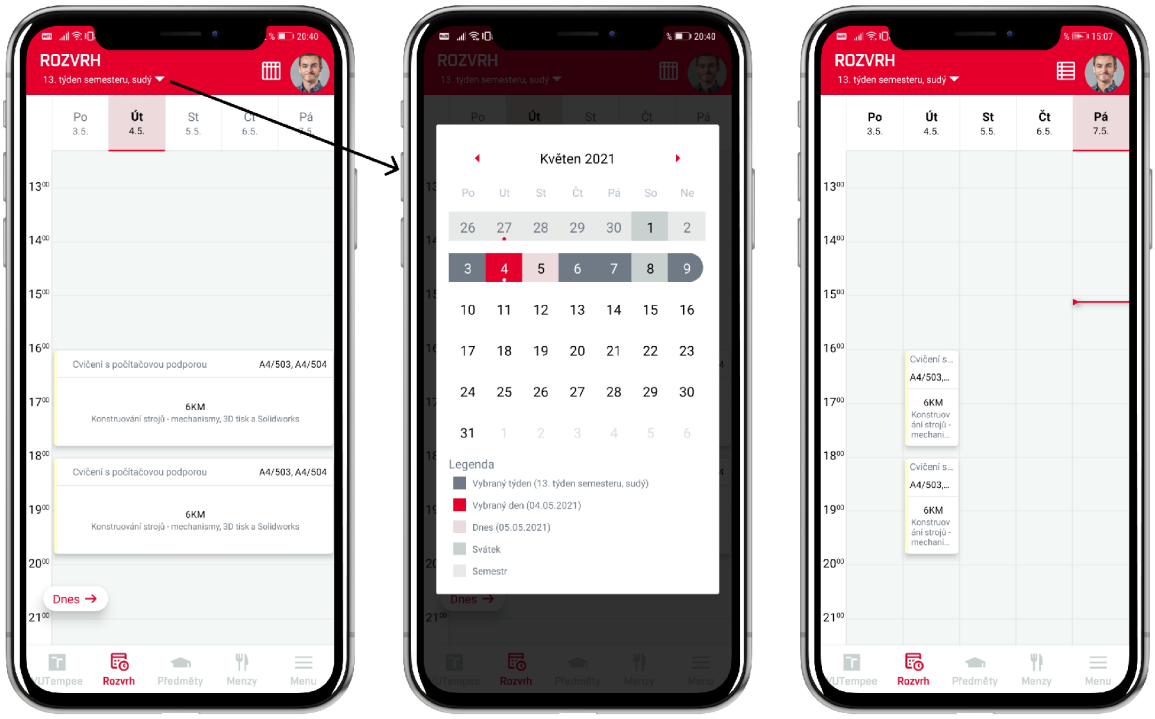


Obrázek 5.1: První verze rozvrhu a zobrazení problémů, na základě kterých bylo opuštěno od tohoto řešení.

Na druhou stranu, výhodou knihovny bylo zobrazení celého roku, včetně zkouškového období, které v aplikaci Moje VUT chybělo. Orientace se navíc zlepšila i vyznačením důležitých dní v kalendáři, kterými jsou:

- dny, kdy učitel učí (zobrazeno červenou tečkou pod dnem),
- podbarvený celý semestr jednou barvou,
- svátky,
- aktuální a vybraný den.

Některé z problémů se podařilo vyřešit částečným přepisem knihovny. Později ale bylo rozhodnuto, že bude lepší řešení použít z knihovny pouze jednoduchý kalendář a vylepšit původní řešení Moje VUT. Aktuální řešení je zobrazené na obrázku 5.2.



Obrázek 5.2: Obrazovka osobního rozvrhu a kalendáře k přepínání dnů v rozvrhu

Rozvrh podporuje týdenní a denní zobrazení s možností přesouvat se mezi jednotlivými dny i týdny potáhnutím prstu po obrazovce. Uživatel se proto může rychle dostat do budoucího nebo minulého týdne bez nutnosti chodit složitě do menu s výběrem cíleného týdne. Rozvrh se umí přizpůsobit i pro případ, kdyby učitel učil v sobotu nebo v neděli. Pouze v těchto případech by došlo k roztažení týdne. Přibylo i tlačítko v levém dolním rohu pro rychlé zobrazení dnešního dne.

Jednou z nejdůležitějších komponent je kalendář, který se zobrazí po kliknutí na číslo aktuálního týdne v rozvrhu. Kalendář je převzat z knihovny od firmy Wix. Orientace v něm je snadná. Přepínání měsíců je realizováno kliknutím na šipky vedle názvu měsíce nebo potažením prstu po obrazovce. Jsou zde vyznačené důležité dny zároveň s legendou, která vysvětluje použité barvy.

Aplikace podporuje čtyři typy rozvrhu. Viditelnost rozvrhů je ovlivněna vnitřním nastavením práv jednotlivých fakult. Jedná se o:

- **Osobní rozvrh** – zobrazí se všem, jedná o osobní rozvrh vyučujícího,
- **Rozvrh osob** – rozvrhy ostatních zaměstnanců, viditelný pouze pro fakulty s právem *teacher2RozvrhyVyucujicichStudentu*,
- **Rozvrh místností** – rozvrhy všech místností na VUT, dostupný pro všechny,
- **Rozvrh předmětu** – rozvrhy všech předmětů, dostupný všem zaměstnancům
- **Rozvrhy studentů** – dostupné pouze vyučujícím, kteří učí dané studenty a pro fakulty s právem *teacher2RozvrhyVyucujicichStudentu*.

Technicky se o logiku rozvrhů stará hlavně třída *SheduleBaseStore*, ze které dědí logiku třídy *ScheduleStore* pro osobní rozvrh a *OtherScheduleStore* pro ostatní typy rozvrhů (předmětů,

místností atd.). Třída *SheduleBaseStore* má v sobě informaci o všech týdnech semestru, prázdninách, aktuálním a vybraném dni a obsahuje funkce k přepnutí týdnů a ukládání rozvrhu do úložiště. Aplikace je aktuálně nastavená tak, že se vždy stahují data z aktuálního a následujícího semestru. Není potřeba stahovat více informací, jelikož se většinou tyto informace nenachází v informačním systému.

Nejdůležitější komponentou je *ScheduleView*, která komunikuje s třídou *ScheduleStore* a stará se o vykreslení rozvrhu. Jedná se o komponentu *FlastList*, která má dvě zobrazení nastavující se parametrem `getItemLayout` – denní a týdenní zobrazení. U každého typu je nutné zadat šířku jednoho dne a pozici dne vůči celému seznamu. U denního zobrazení se nastavuje šířka dne jako šířka obrazovky a pozice jako `šířka_obrazovky * index`. U týdenního zobrazení je konfigurace složitější kvůli možnosti nepravidelné délky dne, která nastává v momentě, kdy učitel učí i v sobotu nebo v neděli. Pozice se pak počítá jako suma prefixů jednotlivých šířek dnů v rozvrhu (metoda `prefixSumDaysOffset` v třídě *ScheduleBaseStore*). Při posouvání se mezi dny je volaná funkce `onViewableItemsChanged`, která indikuje změnu viditelné položky v seznamu a díky ní je možné zjistit index zobrazeného dne. Ten se ukládá do proměnné `dayIndex`, na kterou reaguje i `weekIndex`, jelikož při změně indexu může dojít i ke změně týdne a v rámci optimalizace je důležité si ukládat obě hodnoty a udržovat je vždy aktuální.

Widget s nadcházejícími událostmi

Rozvrh slouží zaměstnanci, aby měl vždy přehled, kdy a kde má být. K rychlejšímu přehledu nadcházejících akcí byl implementován Widget obsahující karty s událostmi pro aktuální den. Jedná se o samostatnou aplikaci, která komunikuje s aplikací VUT empee přes sdílené úložiště. Pro jeho aktivaci je ho nutné přidat na plochu zařízení přes správu widgetů. Ukázka je zobrazena na obrázku 5.3. Nachází se zde informace o předmětu, vyučujících, místnosti, času a po kliknutí na kameru dojde k přesměrování na online hodinu na platformě Teams. Widget je zatím dostupný pouze pro OS Android. React Native totiž zatím nemá na jejich vytváření žádnou podporu, proto je implementován nativně v Javě. Pokud bude o widget zájem, bude nativně doimplementovaný i pro iOS.

Obrázek 5.3: Widget s nejbližšími událostmi rozvrhu umístěný na ploše telefonu.

5.4.2 Docházka

Po kliknutí na vyučovací hodinu v rozvrhu se dostane vyučující na svůj přehled studentů, kteří jsou zapsaní na danou hodinu. U každého studenta se nachází jeho profilová fotografie, jméno a osobní číslo. U každého studenta je možné si poznačit, jestli se nachází na hodině. Tato řešení má za cíl nahradit tisíce papírů vytisknutých právě kvůli označování přítomnosti studentů na hodině. Je implementováno podle návrhu, aby bylo co nejjednodušší a pocho-

pitelné pro všechny. Na obrázku 5.4 se nachází vývoj uživatelského rozhraní k zadávání docházky.

První návrh měl za cíl být co nejvíce informativní, proto se informace o stavu a o tom, kdo stav vložil, nachází u každé buňky. V druhé fázi byla informace o stavu přesunuta do záhlaví tabulky, jelikož byla v buňce zbytečná. Výsledný vzhled je zobrazen na posledním obrázku. Designově se více hodily do aplikace kuličky než čtverce. Dále byl přidán i přehled docházky ze všech týdnů semestru dané rozvrhové jednotky u každého studenta, aby měl vyučující vždy přehled o tom, jak moc daný student chodí na hodiny.

Zadávání docházky probíhá jednoduše. IS podporuje zadávání tří stavů:

- **přítomen** – žák se nachází na hodině, stav je symbolizován ikonou s fajfkou,
- **nepřítomen** – žák není přítomen a ani se neomluvil, symbolizováno ikonou s křížkem,
- **omluven** – buď se student učiteli omlouval, nebo má zadanou omluvenku v informačním systému, nebo je zaevidována překážka ve studiu v IS, stav je symbolizován ikonou omluvenky.

Obrázek 5.4: Vývoj obrazovky se zadáváním docházky

Stav po jeho zadání je možné i zrušit opětovným kliknutím na danou ikonu. Při zadávání se bude měnit počítadlo u jednotlivých typů v záhlaví tabulky. Vpravo nahoře je vždy informace o tom, kolik studentů je celkem zaregistrováno na hodině. Zadávání je možné do minulosti i do budoucnosti, což se může hodit v případě, kdy se student dopředu omluví z hodiny. Přesouvat se mezi jednotlivými cvičeními nebo přednáškami je možné přes kliknutí na šipky vlevo a vpravo nacházející vedle titulku dané vyučovací hodiny.

Technicky se jedná o komponentu *FlatList*, která načte vždy pouze jednu vybranou hodinu a až po kliknutí na šipku si načte další data, proto je načítání seznamu tak rychlé.

Vyhledávat v seznamu je možné kliknutím na lupu přes jméno, příjmení, osobní číslo nebo číslo průkazu studenta naskenováním QR kódu. Na čtení QR kódu je použita knihovna *react-native-qrcode-scanner*, která rozšiřuje funkce knihovny *react-native-camera*. Knihovna přímo nabízí komponentu *QRCodeScanner* s parametrem `onRead`, kterých definuje, jaká akce se má vyvolat po přečtení QR kódu. Jelikož je zde nutnost použití kamery, zařízení musí požádat o povolení k přístupu ke kameře.

Doplňující informace o studentovi na hodině

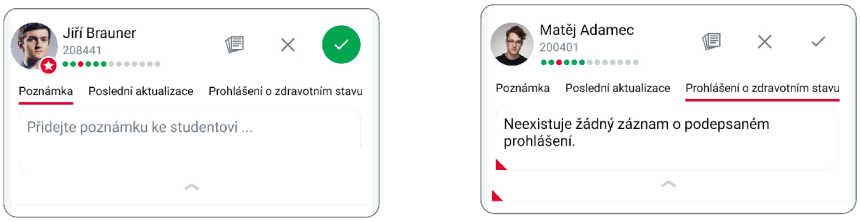
U každého studenta se nachází šipka k zobrazení dalších informacích o studentovi. Tyto informace jsou členěny v záložkách, jak je ukázáno na obrázku 5.5. Přepínání záložek funguje buď kliknutím na zvolenou záložku nebo potáhnutím prstem doleva či doprava. Často je totiž jednodušší a rychlejší použít gesto, než se strefit prstem na malou plochu, kterou zabírá název záložky. Vybraná záložka je podtržená červeně. V tento moment jsou naimplementované tři záložky:

1. **Poznámka** – Textové pole, které slouží učiteli k zaznamenání poznámky na hodině. Primárně je určeno pro učitele, kteří si píší body z jednotlivých cvičení na papír, na konci je sečtou a výsledek vloží do informačního systému. Může ale sloužit i jako osobní poznámka pro učitele (např. „student byl velmi aktivní na hodině“).

2. **Poslední aktualizace** – Jedná se o informaci, kdo naposledy docházku vložil a kdy. Není podmínkou, že cvičení musí učit pouze jeden vyučující, proto musí mít jednotliví vyučující přehled, kdo data vložil. Dále je možné, že si studenti potřebují jednotlivá cvičení nahradit u jiných vyučujících. V těchto případech je velmi důležité vědět, kdo a kdy docházku vložil.

3. **Prohlášení o zdravotním stavu** – V této záložce si učitel může ověřit, jestli má student podepsané prohlášení o zdravotním stavu. Údaj je symbolizovaný trojúhelníkem v levém dolním rohu karty.

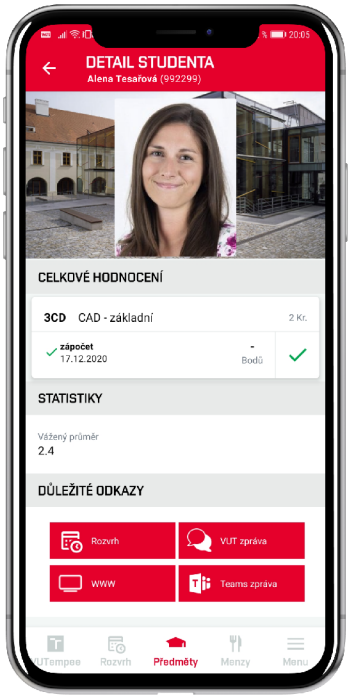
U fotografie studenta se může nacházet i ikona s hvězdičkou. Znamená to, že takový student má zaevidované **speciální potřeby** a je potřeba na něj brát ohled. Může se jednat například o přidání více času na zkoušku.



Obrázek 5.5: Záložky s dalšími informacemi u studenta, rozbalené po kliknutí na šedou šipku.

Karta studenta

Po kliknutí na jméno studenta v docházce se dostaneme do jeho karty, která je zobrazena na obrázku 5.6. Fotografie studenta již zabírá větší část obrazovky pro jeho lepší identifikaci



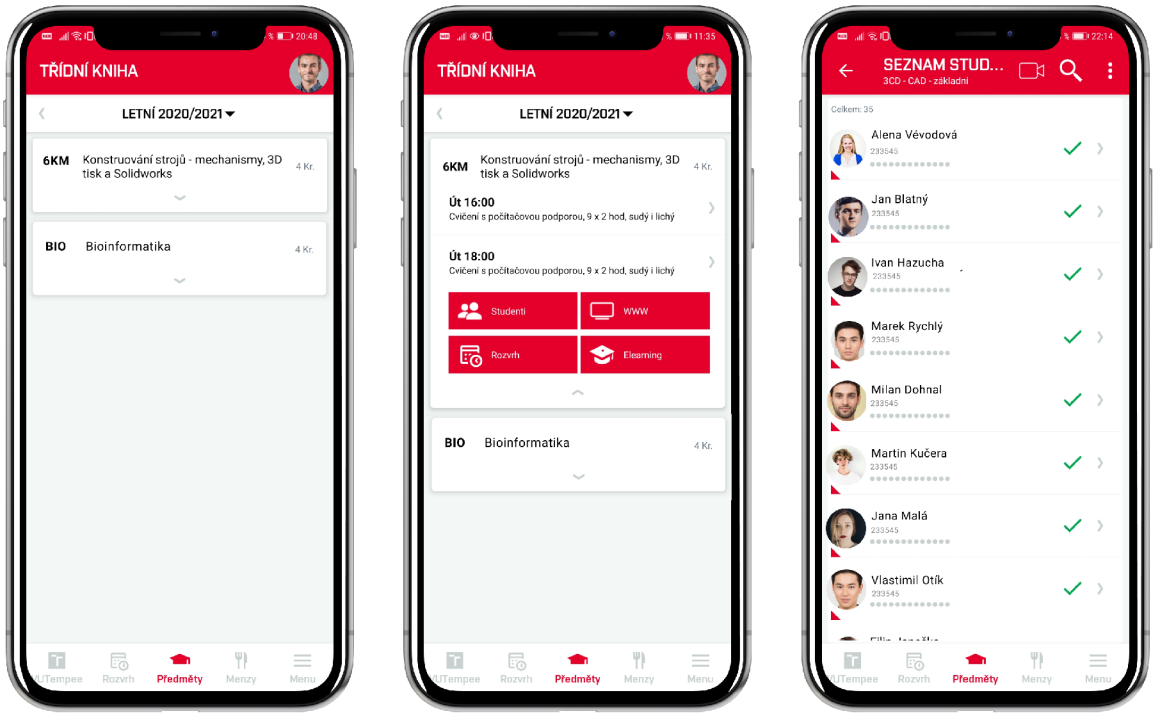
Obrázek 5.6: Karta studenta

a nachází se zde i informace o získaném hodnocení. Pokud má student hodnocení více (např. za účast na cvičení), zobrazí se další sekce **Dílčí hodnocení** společně se všemi zadanými body. Vážený průměr značí průměrnou známku za celé studium. Slouží učiteli k vytvoření představy o snaživosti studenta. V sekci důležitých odkazů se nachází základní čtyři akce, které lze provést u studenta. Jedná se o:

- **rozvrh** studenta,
- **VUT zprávu**,
- přesměrování na kartu studenta na **web**,
- **Teams zprávu**, která přesměruje uživatele do aplikace Teams a přidá studenta jako příjemce zprávy.

5.4.3 Předměty

Implementace modulů předmětů je velmi podobná návrhu a je zobrazena na obrázku 5.7. Zobrazují se zde nejen předměty, které vyučující učí, ale i ty, kde je garantem. Proto se zde může nacházet velké množství předmětů a je důležité obrazovku udržet přehlednou. U každého předmětu je vždy zobrazena pouze zkratka, název a počet kreditů. Více možností se zobrazí při rozbalení karty, jak lze vidět na druhém obrázku. V první části karty je seznam rozvrhových jednotek, které učitel vyučuje. Každá rozvrhová jednotka je identifikovaná dnem, časem, frekvencí a typem. V druhé části obrazovky se nacházejí rychlé akce u předmětu podobné odkazům popsané u karty studenta. Nalézá se zde **rozvrh** předmětu, **webové stránky** předmětu, odkaz na **elearning** a **seznam všech studentů** předmětu.



Obrázek 5.7: Přehled předmětů, které učitel učí s prokliknutím na seznam studentů

Po kliknutí na konkrétní rozvrhovou jednotku se zobrazí seznam studentů, jak lze vidět na třetím obrázku. Jedná se o stejnou obrazovku jako u zadávání docházky s tím rozdílem, že místo zadávání docházky se napravo od studenta nachází jeho výsledné hodnocení. V případě zápočtu se objeví ikona fajfky označující započteno nebo křížku symbolizující nezapočteno. Učitel má proto hned přehled o tom, kterým studentům dal zápočet, což je například vhodné u předmětů, kde se dostává zápočet pouze za účast (např. tělocvik). Zároveň je zde ponechána přehledová informace o docházce za jednotlivé týdny semestru, aby si případně mohl učitel či garant zkontrolovat, jestli je zápočet správně vložen. Pokud je předmět hodnocený zkouškou, zobrazí se výsledná známka A až F. Vkládání hodnocení zatím není aplikací podporováno, ale očekává se přidání přes odkaz na web, kde jsou již implementované veškeré kontroly, a není proto potřeba implementovat stejnou logiku dvakrát.

5.5 Nastavení zabezpečení pomocí biometrie

Na základě analýzy citlivých modulů v sekci 4.4 byly naimplementovány první dvě situace, ve kterých je vhodné použít biometrického zabezpečení. Jedná se o přihlašování a přechod aplikace do popředí. Obě možnosti bude možné zapnout i vypnout v nastavení, jak lze vidět na obrázku 5.8. V případě, že telefon nepodporuje biometrii, nebo je zakázána v aplikaci, nastavení nebude zobrazeno. V aplikaci se informace získává z knihovny *RNSInfo* přes funkci *isSensorAvailable*, která vrací podporovaný typ senzoru na daném zařízení. Jedná se o typ *RNSensitiveBiometryType* obsahující hodnoty *Face ID* v případě podpory rozpoznání podle obličeje nebo *Touch ID* v případě podpory otisku prstu.

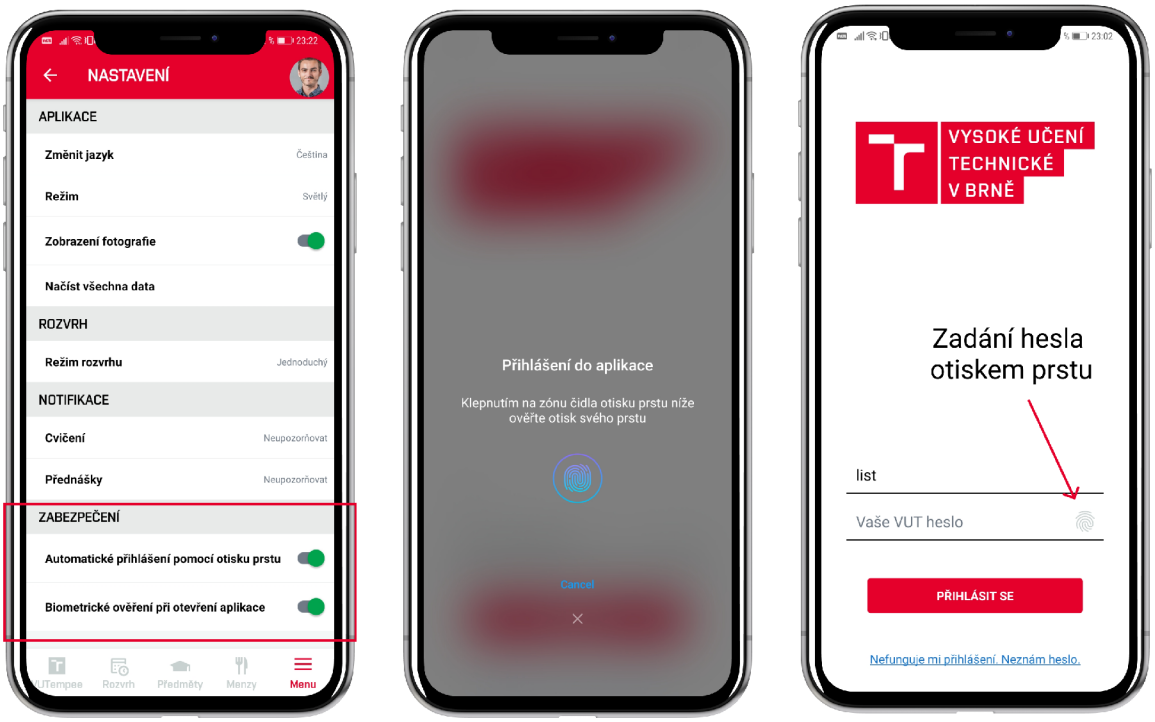
V případě zapnutí možnosti **automatického přihlášení** přes otisk prstu je potřeba se znovu odhlásit a přihlásit, jelikož v momentě změny nastavení není heslo známé (nikam se

neukládá). Až v momentě přihlášení se objeví obrazovka se souhlasem k uložení hesla pod otiskem prstu. Po odsouhlasení nebude již při příštím přihlášení heslo potřeba. Místo hesla bude možné kliknout na ikonu otisku prstu, jak je zobrazeno na třetím obrázku.

Druhá možnost se týká použití biometrického ověření při **přechodu aplikace do popředí**. Pokud je v nastavení možnost povolena, po každém otevřením aplikace bude vyžadováno biometrické ověření uživatele. V případě chybných pěti pokusů dojde k odhlášení uživatele. Toto nastavení je doporučeno nechat zapnuté pro zaměstnance, kteří učí nějaké studenty. V aplikaci se totiž nachází jejich osobních údaje jako je například osobní číslo či fotografie a nebylo by dobré, kdyby tyto informace mohl vidět někdo jiný než samotný učitel v případě odcizení zařízení. Ke zjištění přechodu aplikace do popředí se používá objekt *AppState* z API přímo od knihovny React Native. K aktuálnímu stavu se přistupuje přes `AppState.currentState`. Základní stavy jsou tři:

1. **active** – aplikace běží v popředí,
2. **background** – aplikace běží v pozadí a uživatel je buď v jiné aplikaci, nebo na domovské obrazovce,
3. **inactive** (pouze proiOS) – stav nastává při přechodů stavů *active* a *background* nebo při přijatém telefonickém hovoru.

V implementaci je použit posluchač na změnu tohoto stavu a v případě přechodu do stavu *active* dojde k ověření, jestli je zapnuté nastavení biometrického ověření při přechodu aplikace do popředí. Pokud je nastavení zapnuto, dojde k verifikaci uživatele.



Obrázek 5.8: Nastavení bezpečnosti pomocí biometrie

Kapitola 6

Testování a nasazení aplikace

Testování je nedílnou součástí vývoje mobilní aplikace. Jeho cílem je odladit chyby v návrhu uživatelského rozhraní a jeho implementaci. Tyto chyby představují neintuitivnost nebo neefektivnost uživatelského rozhraní, které můžou být vývojáři skryté pro jeho dobré znalosti systému. K otestování aplikace je potřeba, aby si ji uživatelé mohli stáhnout do svého zařízení. Jelikož aplikace obsahuje několik souborů, je potřeba je zabalit do jednoho souboru, kterému se říká APK. Zabalení aplikace a distribucí se zabírá sekce 6.1. V sekci 6.2 jsou pak shrnuty výsledky uživatelského testování, na jejichž základě jsou navrženy další kroky budoucího vývoje.

6.1 Distribuce aplikace

Aplikace je zatím ve vývojovém stádiu, a proto není žádoucí ji hned zveřejňovat přes *Google Play* nebo *App Store*. Má sloužit pouze testovací skupině uživatelů, a tedy není zatím určena pro širokou veřejnost. V tomto případě je potřeba najít způsob, jak zabalit aplikaci a jednoduše ji poslat, aby byla funkčních na všech typech zařízení a instalace nebyla příliš složitá. Nejnižší podporovatelná verze OS Android je 4.4, pro iOS je nejnižší verze 9.0.

Nejjednodušší možnost distribuce je přes **Expo** klienta[1]. Jedná se o framework pro React Native obsahujících sadu nástrojů a služeb sloužící ke zjednodušení vývoje, sestavení a nasazení aplikace na OS Android, iOS i web. Dokumentace je velmi přehledná a obsahuje kompletní tutoriál a příklady použití. Expo nabízí dva přístupy vývoje aplikací:

- **managed workflow** – Jedná se o postup, kdy se Expo klient stará o všechny služby a vývojář pouze píše kód v JavaScriptu. Soubory android i ios jsou vývojáři skryté. Není potřeba vyvíjet v Android Studiu (pro OS Android) ani v XCode (pro OS iOS). Přístup má i několik nevýhod jako je například nedostupnost některých nativních modulů, chybějící podpora některých úloh běžící na pozadí nebo minimální verze OS Android 5+ a iOS verze 10+.
- **bare workflow** – V tomto případě má vývojář plnou kontrolu nad každým aspektem svého projektu a Expo nástroje jsou limitované. Pokud byl projekt vytvořen na počátku bez Expo nástrojů, je nutné použít tento přístup. Výhodou je možnost přidání vlastního nativního kódu a plná kontrola nad aplikací. Nevýhodou je složitá konfigurace přes soubor *app.json*.

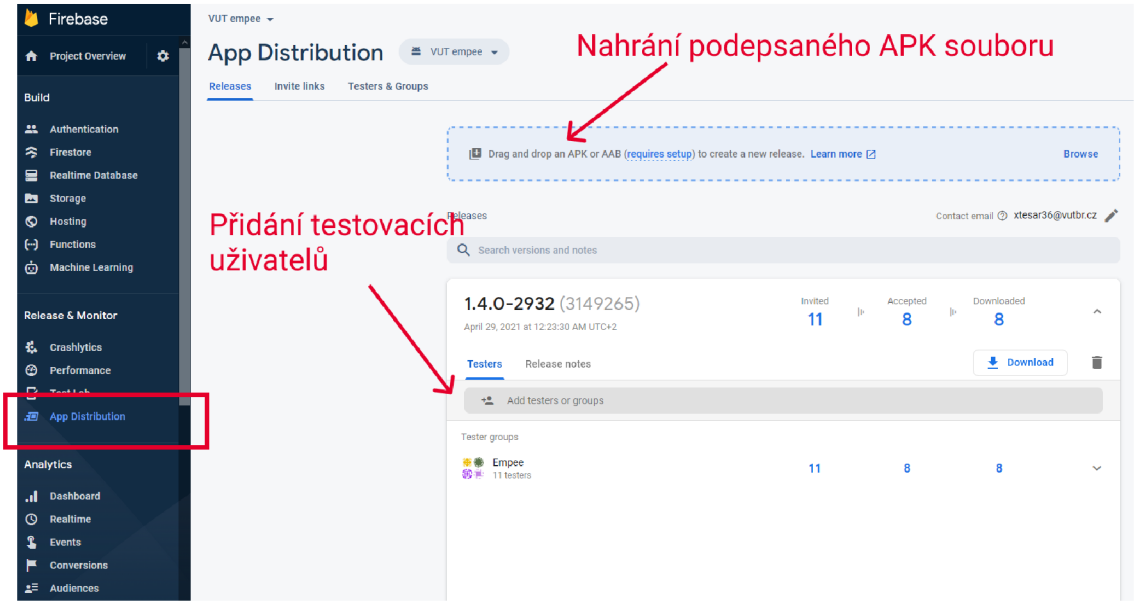
[1] Více infromací a dokumentace se nachází na stránkách `https://expo.io/`.

Jelikož vytvořená aplikace je funkční bez nástrojů Expo, byla by konfigurace pouze kvůli distribuci příliš složitá. Proto bylo řešení pomocí Expo zamítnuto.

Jednodušší možnost je nasazení aplikace přes webový nástroj **Firebase** od společnosti Google. Nasazení je možné pouze pro OS Android, což v první fázi testování není problém. Testována bude především intuitivnost uživatelského rozvrhaní než funkčnost a implementační chyby aplikace. Distribuce probíhá v následujících krocích:

1. Vytvoření účtu na stránkách `https://console.firebase.google.com/`.
2. Vytvoření nového projektu.
3. Nahrání podepsaného APK souboru přes záložku **Distribuce aplikace** (obrázek 6.1).
4. Vložení testovacích skupin uživatelů.

Po přidání skupiny do seznamu se pošle vybraným testovacím uživatelům e-mail, ve kterém bude odkaz na stažení aplikace. Jedná se o velmi pohodlnou volbu pro testovacího uživatele, jelikož nemusí instalovat nic navíc. Úlohou vývojáře je pouze vytvoření podepsaného APK souboru. Postup vytvoření shrnuje následující sekce.



Obrázek 6.1: Nahrání APK souboru a distribuce testovacím uživatelům přes Firebase

Vytvoření APK souboru

APK je zkratka pro *Android Package Kit*, který se používá v OS Android k distribuci a instalaci aplikací. Nachází se v něm zabalené všechny zdrojové soubory a může obsahovat i citlivé údaje jako jsou například přístupy k databázi. Není proto žádoucí, aby si mohl kdokoliv kód otevřít a zneužít jeho obsah. Obsah musí být zabezpečený heslem.

Nejjednodušší postup vytvoření podepsaného APK nabízí program Android Studio. Je potřeba kliknout na *Build* → *GenerateSignedBundle/APK* → *NewKeyStore* a vyplnit potřebné údaje o nově vytvořeném klíči, který bude sloužit k podepisování certifikátu. Vždy je potřeba vyplnit:

- **Alias** – Textová reprezentace klíče.
- **Heslo** – Heslo k vygenerovanému klíči.
- **Platnost** – Délka platnosti klíče udávaná v letech. Doporučeno udávat minimálně 25 let [19], aby bylo možné podepisovat aktualizace aplikace stejným klíčem.
- Dodatečné informace o společnosti, jméno zodpovědné osoby, město vydání, stát a kód země.

Dále je potřeba uvést cestu k nově vytvořenému souboru a heslo, které by mělo být různé od hesla klíče. Tento nově vytvořený soubor se nazývá *keystore* a obsahuje privátní a veřejný klíč. Slouží k podepsání aplikace. Po vyplnění údajů je potřeba nakonec uvést místo, kam se má vytvořené podepsané APK vložit a zvolit typ sestavení (*release* nebo *debug*). Výsledné APK se vytvoří na zvolené lokaci a je možné aplikaci distribuovat uživatelům na testování.

6.2 Uživatelské testování

Principem uživatelského testování je definování úloh, které mají uživatelé v aplikaci splnit. Tyto úlohy představují základní práci s aplikací a jsou navržené tak, aby ověřily intuitivnost navrženého řešení. Pro testování byla vybrána skupina uživatelů, která bude reálně aplikaci využívat a má základní znalosti práce s informačním systémem VUT. Průběh testování kvůli hygienické situaci v České republice musel proběhnout online přes platformu *Microsoft Teams* následujícím způsobem:

1. Uživateli byla poslána aplikace *VUT empee* na e-mail (přes Firebase).
2. Po instalování aplikace byl uživatel požádán o sdílení obrazovky na jeho zařízení, aby mohl být sledován každý jeho pohyb po obrazovce.
3. Uživatel byl poučen, že po celou dobu testování má komentovat své myšlenky.
4. Uživatel vykonal 15 úkolů k nalezení problémů spojených s aplikací.
5. Diskuze s uživatelem k budoucímu vývoji aplikace, nápady ke zlepšení.
6. Ukončení hovoru a poslání finálního dotazníku k vložení zpětné vazby k aplikaci. Počítá se s tím, že dotazník vyplní uživatel až několik minut po hovoru. Má proto čas si v klidu uspořádat myšlenky k aplikaci a do dotazníku může uvést ještě další nápady či připomínky, na které si vzpomněl.

Sdílení obrazovky je velmi důležité, jelikož si tester může všimnout každého pohybu v aplikaci a hned vidí, kde je problém. Čím méně kroků uživatel vykoná k nalezení řešení, tím je aplikace intuitivnější. Pokud uživatel přejde celou aplikaci a stále se mu nepodaří splnit daný úkol, například nalezení VUT pinu, umístění není intuitivní a je potřeba ho změnit. Zároveň je dobré vysvětlit uživateli simulovanou situaci, ve které se nachází, aby se mohl lépe vžít do role. Úkolů bylo celkem 15:

1. Otevřete aplikaci a přihlaste se.
2. Začíná semestr a Vás zajímá, kolik budete učit studentů. Kde tuto informaci naleznete?

3. Semestr skončil a Vás zajímá, kolik procent studentů dostalo z vašeho předmětu A a jestli předmět skončil alespoň tak dobře jako minulý rok.
4. Chcete se přihlásit na wifi. Potřebujete zjistit svůj VUT pin.
5. Potřebujete se ozvat panu Jaroslavu Listovi, jak mu rychle napíšete e-mail nebo zavoláte?
6. Student má velmi složité příjmení a pro vás je jednodušší si ho vyhledat podle průkazu. Jak ho vyhledáte přes QR kód průkazu?
7. Který z vašich studentů je dnes na hodině a nemá podepsané prohlášení o zdravotním stavu?
8. Blíží se konec roku a s ním spojené zadávání zápočtů. Zadal jste již všem svým studentům zápočet nebo někdo chybí?
9. Tento rok učíte v předmětu Matematika 50 studentů, je to více či méně než minulý semestr?
10. Máte nějakého studenta se SVP? Jak byste ho v aplikaci našel?
11. Chcete zapnout zabezpečení pomocí otisku prstu, kde byste nastavení hledal?
12. Jaký je aktuální týden semestru?
13. Jak zjistíte, kdy začíná a končí semestr a jaké dny budete mít volno?
14. Student dostal v hodině 3 body, kam byste si tento údaj poznačili?
15. Student si stěžuje, že mu nebyla zadána docházka minulý týden. Jak byste se rychle přepnul na minulý týden cvičení, aby jste si údaj ověřil a docházku mu doplnil?

Vyhodnocení výsledků

Aplikace byla testována na zaměstnancích různých fakult VUT s Android zařízeními. První nejasnosti nastaly u druhého úkolu, který měl ověřit, jestli uživatelé budou volit spíše cestu na studenty přes rozvrh nebo předměty. Z 80 % byla zvolena cesta přes předměty. Informace lze ale nalézt na obou místech, proto je pouze otázka zvyku, co bude pro daného uživatele nejpohodlnější. Uživatelé zároveň dodali, že by chtěli vidět studenty na obou místech, jak je to nyní.

Největší problém dělalo nalezení statistik předmětu, které se nachází záložce předměty po kliknutí na název předmětu. Uživatelé automaticky rozevřeli záložku s předmětem a hledali v odkazech, kde se nachází detail předmětu. Zkoušeli dokonce hledat i na webových stránkách předmětu. Řešením by bylo přidání dalšího odkazu **Detail předmětu** do vysunující karty, kde by nalezli statistiky z posledních let.

Druhý největší problém bylo pochopení červeného trojúhelníku v levém dolním rohu u studenta znamenající (ne)podepsané prohlášení o bezinfekčnosti v úloze č. 7. Ve většině případů si uživatelé mysleli, že tato informace patří k nezískanému zápočtu, někteří nevěděli vůbec. V diskuzi pak navrhovali použití jiné ikony jako je ikona viru používající se na webu, což by mohlo problém vyřešit.

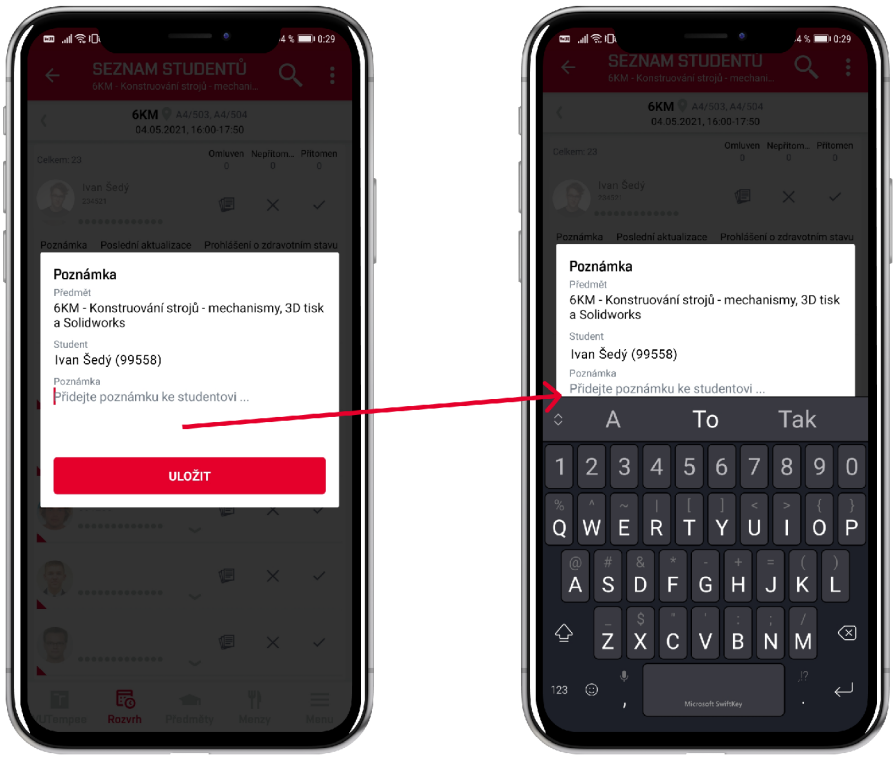
Dále bylo zjištěno, že při procházení zápočtů a známek u studentů není jasné, co má hodnota symbolizovat. Jednalo se o případy, kdy ještě studenti neměli zadaný zápočet,

a proto se zobrazila pouze pomlčka, ze které informace není jasná. Řešením by bylo přidání záhlaví do tabulky s titulkem „Hodnocení“.

Úlohy s rozvrhem nedělaly žádné velké problémy a uživatelé hned věděli, jaký je aktuální semestr (úloha 12) a dokázali se v rozvrhu rychle orientovat díky přepínání týdnů přetáhnutím prstu po obrazovce. Ve většině případů preferovali tohle řešení nad otevřením kalendáře, kde by byla orientace ještě jednodušší. Problém byl ten, že název aktuálního týdne se šipkou dolů nenaznačuje zobrazení kalendáře, proto uživatele nenapadlo na něj kliknout. Navrženým řešení je přidání tlačítka s ikonou kalendáře do vrchní lišty rozvrhu. Ikona bude jasnější a text s identifikací semestru zůstane pouze textem bez další funkcionality.

Vyhledávání studentů přes QR kód (úloha 6) nedělalo žádný problém. Uživatelé by avšak uvítali globální vyhledávání všech svých studentů, nyní totiž mohou vyhledávat studenty pouze v seznamech studentů konkrétního předmětu.

U zadávání počtu bodů studentovi na cvičení (úloha 14) byl zjištěn problém s vkládáním textu, který demonstruje obrázek 6.2. Vkládání je realizováno modálním oknem, které je umístěno na středu obrazovky. V případě zadávání textového vstupu se objeví klávesnice, která tlačítko *Uložit* překryje. Následně je uživatel zmatený, jak má hodnotu uložit, protože mu tlačítko zmizí. Pokud klikne mimo okno, změny se neuloží a okno se zavře, což se stalo 1x při uživatelském testování. Řešeních daného problému je více. Buď je možné posunout modální okno nad klávesnici, aby bylo vidět tlačítko *Uložit*, nebo předělat modální okno na samostatnou obrazovku. Na samostatné obrazovce by se dalo lépe ošetřit, jestli si uživatel přeje poznámku uložit a mohla by zde být zároveň informace o historii změn poznámek.



Obrázek 6.2: Po kliknutí na textové pole k vložení poznámky zmizí tlačítko uložit za klávesnici

Nastavení zabezpečení pomocí biometrie bylo opět v pořádku, uživatelé tuto možnost velmi vítali a potvrdili užitečnost této funkce. Někteří měli použití biometrie zakázané na jejich zařízeních, proto se jim obrazovka s potvrzením otisku prstu nikdy neukázala. V tomto případě je potřeba opravit aplikaci tak, aby zapnula biometrické ověření uživatele přímo v nastavení telefonu. Problém by se tím měl vyřešit.

Celkově byli uživatelé s aplikací velmi spokojeni. Zvlášť ocenili přidání nové funkcionality zadávání docházky studentů, kterou do teď museli řešit pouze papírově. Všichni dotázaní uživatelé dále uvedli, že je aplikace užitečná a že jim ulehčí spoustu času minimálně s hledáním výuky v rozvrhu.

6.3 Budoucí vývoj

Díky uživatelskému testování bylo objeveno několik chyb navrženého rozhraní aplikace, proto bude první fáze budoucího vývoje věnována opravě zjištěných chyb. Jedná se o:

- přidání více popisků do záhlaví tabulek,
- změny textových odkazů na samostatná tlačítka (v rozvrhu se jedná o kalendář a v předmětech o detail předmětu),
- změna ikony u potvrzení o zdravotním stavu studenta,
- zapnutí biometrické podpory v nastavení zařízení,
- opravení modálního okna u zadávání poznámky,
- přidání odkazu na web k zadání hodnocení u studenta,
- přidání globálního vyhledávání studentů.

Kromě oprav chyb je plánováno rozšíření funkcionality aplikace vycházející z analýzy nejpoužívanějších modulů a z debaty vedené se zaměstnanci. Jako první věc je potřeba zautomatizovat vkládání docházky. Většinou je spíše potřeba označit studenty, kteří chybí, proto by bylo vhodné přidat tlačítko **Vložení docházky všem** a pak pouze odklikat chybějící studenty. Další možnost automatizace je přes kód, který by učitel sdělil studentům a ti by ho pak zadali do aplikace Moje VUT. Možnosti jsou i použití Bluetooth (případně NFC). Dále je v plánu přidat více přehledových obrazovek, kde lze souhrnně vidět celou docházku třídy a zadané poznámky. Je zvažována i možnost sečtení bodů, pokud je nalezeno v poznámce číslo.

Ve spojení s docházkou je často potřeba si studenty rozdělit do skupin. Může totiž nastat situace, že jsou studenti rozděleny do dvou místností každá s jiným učitelem, i když se jedná pořád o stejnou vyučovací hodinu. Z tohoto důvodu je v plánu přidat možnost označení studentů a jejich následné umístění do skupin.

V souvislosti s hledáním bude implementováno vyhledávání v celé aplikaci, které pomůže s orientací v aplikaci. Bude možné vyhledávat podle osob (i studentů), předmětů a místností a zároveň budou přidány odkazy k nejdůležitějším modulům aplikace jako je například VUT pin nebo E-learning, aby učitel nemusel přemýšlet, kde se nacházejí.

Do základního menu bude přidáno více odkazů na web na nejpoužívanější moduly jako je Evýplata, VUT mobil, Závěrečné práce a Přehled nepřítomnosti. Jedná se o moduly, které používá uživatel pouze jednou za čas a není potřeba implementovat jejich logiku do aplikace, když již existuje na webu. Jedinou výjimkou je modul přehledu docházky

na pracovišti. Jelikož zaměstnance často zajímá, jestli si na daný den ukončil přítomnost nebo kolik hodin za celý den odpracoval. Tento modul je plánovaný k brzké implementaci.

Aplikace má velký potenciál. Může v budoucnu nahradit zaměstnaneckou kartičku použitím technologie NFC, aby bylo možné si pomocí telefonu otevírat dveře na chodbě nebo platit jídlo v menze. Už nebude potřeba vytahovat peněženku, ale bude stačit pouze telefon. Dále je v plánu s využitím technologie Bluetooth přidat možnost navigace do konkrétní místnosti. Často totiž navštěvuje učitel i jiné fakulty a orientace není vždy být snadná. Telefon by ho dokázal přesně lokalizovat, kde se nachází a dovést ho do místnosti, kde má právě učit (i s detekcí otevřených/zavřených dveří). Už by nemusel kvůli špatné orientaci po fakultě přijít pozdě na hodinu.

Kapitola 7

Závěr

V rámci diplomové práce byla navržena aplikace pro zaměstnance VUT, která má zjednodušit nejčastější denní rutiny při používání centrálního IS a sloužit k zefektivnění jejich práce a snížení administrativy. Aplikace byla vyvinuta pomocí multiplatformního frameworku React Native pro OS Android a iOS.

Na začátku byly představeny a porovnány základní technologie pro vývoj mobilních aplikací. U každé byly vždy uvedeny výhody a nevýhody společně s jedním zástupcem technologie. V analýze současného systému se podařilo popsat používané moduly IS a vybrat z nich statisticky ty nejdůležitější. Statistika byla brána ze tří zdrojů – *Teacher*, *Intraportál* a *Apollo*, proto byla data navíc sesbírána z dotazníku, který byl rozeslán zaměstnancům k rozhodnutí, který modul je pro ně nejdůležitější. V celkovém hodnocení nejlépe dopadly VUT zprávy, E-výplata, VUT mobil, Osobní rozvrh, Předměty a Závěrečné práce.

Dále se podařilo na základě této znalosti navrhnout uživatelské rozhraní mobilní aplikace, kde byl kladen důraz hlavně na všední situace zaměstnance, ve kterých se může nacházet při používání aplikace a možnost využití nativních funkcí telefonu jako je například použití *GPS*, *Bluetooth*, *NFC* nebo o otisku prstu k verifikaci. Na základě analýzy potřeb zaměstnance se podařilo navrhnout strukturu a rozhraní nového modulu docházky studentů, který v informačním systému neexistoval.

Výsledná mobilní aplikace komunikuje se školním serverem za účelem získání dat, které jsou ukládány do zařízení k poskytnutí *offline* režimu. Kromě osobního rozvrhu, vyučovaných předmětů, rozvrhu předmětů a seznam menz s aktuálním jídelníčkem, obsahuje také aplikace seznamy studentů na konkrétních hodinách s možností poznačení jejich přítomnosti na hodině nebo vložení poznámky. Každého studenta je možné identifikovat jeho fotkou, osobním číslem nebo jménem. Do aplikaci byl i přidán odkaz na poslání rychlé zprávy přes platformu *Teams* nebo odeslání VUT zprávy. U každého studenta učitel vidí všechna jeho hodnocení předmětu a hned ví, jestli mu už byl udělen zápočet.

Kvůli výskytu citlivých dat v aplikaci je v aplikaci implementováno zabezpečení pomocí biometrického ověření uživatele. V aktuální verzi aplikace je použito ve dvou případech a uživatel si ho může kdykoliv vypnout v nastavení. Biometrické ověření je ve výchozím stavu nastaveno při každém přechodu aplikace do popředí. Ve druhém případě se používá u přihlášení do aplikace.

Aplikace byla testována na malé skupině zaměstnanců se zařízeními Android. U testování byly například objeveny chyby jako je skryté tlačítko Uložit při ukládání poznámky nebo neintuitivnost textového odkazu. Celkově byli ale uživatelé s aplikací velmi spokojeni. Další vývoj se zaměří na opravu chyb uživatelského rozhraní, dále pak na automatické zadávání docházky, lepší práce se studenty, vyhledávání v celé aplikaci a přidání dalších modulů,

jako je například E-výplata, VUT mobil, Závěrečné práce a Přehled nepřítomnosti. Aplikace má velký potenciál do budoucna. Může například nahradit zaměstnaneckou kartičku nebo může sloužit jako interaktivní navigace po fakultě.

Literatura

[1] Abramov, D., Alison, N., Kakkar, A. a Monica Powell, S. *React* [online]. Srpen 2020 [cit. 2020-12-26]. Dostupné z: https://reactjs.org/docs.

[2] Apple. *Apple documentation – Start Developing iOS Apps (Swift)* [online]. Prosinec 2016 [cit. 2020-12-26]. Dostupné z: https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/BuildABasicUI.html#//apple_ref/doc/uid/TP40015214-CH5-SW1.

[3] Avola, G. a Raasch, J. *Smashing Mobile Web Development*. Wiley, 2012. Smashing Magazine Book Series. ISBN 9781118348130. Dostupné z: https://books.google.cz/books?id=gaNwl2azgcsC.

[4] Baldha, K. *Difference Between Native App VS Hybrid App* [online]. Duben 2020 [cit. 2020-12-26]. Dostupné z: https://medium.com/techvoot-solutions/difference-between-native-app-vs-hybrid-app-e33606fee545.

[5] Bolle, R., Connell, J., Pankanti, S., Ratha, N. a Senior, A. *Guide to Biometrics*. Springer-Verlag New York, 2004. ISBN 978-1-4757-4036-3.

[6] Bontrager, P., Roy, A., Togelius, J., Memon, N. a Ross, A. *DeepMasterPrints: Generating MasterPrints for Dictionary Attacks via Latent Variable Evolution*. 2018.

[7] Brewster, T. *We Broke Into A Bunch Of Android Phones With A 3D-Printed Head* [online]. Prosinec 2018 [cit. 2021-5-5]. Dostupné z: https://www.forbes.com/sites/thomasbrewster/2018/12/13/we-broke-into-a-bunch-of-android-phones-with-a-3d-printed-head/?sh=3656b7be1330.

[8] Budiu, R. *Basic Patterns for Mobile Navigation: A Primer* [online]. Listopad 2015 [cit. 2021-1-9]. Dostupné z: https://www.nngroup.com/articles/mobile-navigation-patterns/.

[9] Chirillo, J. a Blaul, S. *Implementing Biometric Security*. Wiley Publishing, 2003. ISBN 978-0-764-52502-5.

[10] Cuello, J. a Vittone, J. *Designing Mobile Apps*. Smashing Media, 2013. Dostupné z: https://books.google.cz/books?id=nQBJAQAAQBAJ.

[11] CVIS. *Apollo(Apollo)* [online]. [cit. 2021-1-10]. Dostupné z: https://www.vutbr.cz/wiki/Apollo(Apollo).

[12] Damier, I. *Using BiometricPrompt with CryptoObject: how and why* [online]. Únor 2020 [cit. 2021-5-5]. Dostupné z: https://medium.com/androiddevelopers/using-biometricprompt-with-cryptoobject-how-and-why-aace500ccdb7.

[13] Domes, S. *Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase*. Packt Publishing, 2017. ISBN 9781788296137. Dostupné z: https://books.google.cz/books?id=8RhKDwAAQBAJ.

[14] Drahanský, M. *Biometrické systémy* [online]. 2006 [cit. 2021-5-4]. Dostupné z: https://www.fit.vutbr.cz/study/courses/BIO/private/BIO_Studijni_opora.pdf.

[15] Drexlerová, J. J. S. *Evidence pracovní doby* [online]. Leden 2020 [cit. 2021-1-5]. Dostupné z: https://www.praceamzda.cz/clanky/evidence-pracovni-doby-0.

[16] Dytrych, J. *Android* [online]. Prosinec 2020 [cit. 2020-12-26]. Dostupné z: https://www.fit.vutbr.cz/study/courses/GJA/public/slides/GJA-10-Android.pdf.

[17] Eisenman, B. *Learning React Native: Building Native Mobile Apps with JavaScript*. O'Reilly Media, 2015. ISBN 9781491929070. Dostupné z: https://books.google.cz/books?id=274fCwAAQBAJ.

[18] Google. *Platform Architecture* [online]. Červenec 2020 [cit. 2020-12-26]. Dostupné z: https://developer.android.com/guide/platform.

[19] Google. *Generate an upload key and keystore* [online]. Březen 2021 [cit. 2021-5-9]. Dostupné z: https://developer.android.com/studio/publish/app-signing#generate-key.

[20] Hamedani, M. *React Lifecycle Methods — A Deep Dive* [online]. Listopad 2018 [cit. 2021-1-10]. Dostupné z: https://programmingwithmosh.com/javascript/react-lifecycle-methods/.

[21] Hayward, J., Fedosejev, A., Prusty, N., Horton, A., Vice, R. et al. *React: Building Modern Web Applications*. Packt Publishing, 2016. ISBN 9781786462848. Dostupné z: https://books.google.cz/books?id=jUvZDQAAQBAJ.

[22] Klinovsky, O. *Mobilní aplikace pro studenty VUT*. 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing. Tomáš Hruška, CSc.

[23] Lee, W. *Beginning Android Application Development*. Wiley, 2011. Wrox beginning guides. ISBN 9781118087800. Dostupné z: https://books.google.cz/books?id=hlygR8lJTzUC.

[24] Maj, W. *React Lifecycle Methods diagram* [online]. 2021 [cit. 2021-1-10]. Dostupné z: https://github.com/wojtekmaj/react-lifecycle-methods-diagram.

[25] McWherter, J. a Gowell, S. *Professional Mobile Application Development*. Wiley, 2012. ITPro collection. ISBN 9781118240687. Dostupné z: https://books.google.cz/books?id=S7CIlnNkg8UC.

[26] Panhale, M. *Begininning Hybrid mobile Application Development*. New York: Apress, 2016. ISBN 978-1-14842-1314-8.

[27] Pavel Mičeк, J. V. *Předměty(Apollo)* [online]. 2020 [cit. 2020-12-13]. Dostupné z: https://www.vutbr.cz/wiki/P%C5%99edm%C4%9Bty(Apollo).

[28] Pedamkar, P. *WebSocket vs RES* [online]. 2020 [cit. 2021-5-7]. Dostupné z: https://www.educba.com/websocket-vs-rest/.

[29] Rauschmayer, A. *Exploring ES6* [online]. 2015 [cit. 2021-5-4]. Dostupné z: https://exploringjs.com/es6.html.

[30] Ridha, R. R. *3 Good Reasons Why You Might Want to Remove that Hamburger Menu from Your Product* [online]. Únor 2018 [cit. 2021-1-9]. Dostupné z: https://medium.muz.li/3-good-reason-why-you-might-want-to-remove-that-hamburger-menu-from-your-product-69b9499ba7e2.

[31] Secord, A., Banes, C. a Pierre Louis, S. *Understanding navigation* [online]. [cit. 2021-1-9]. Dostupné z: https://material.io/design/navigation/understanding-navigation.html#lateral-navigation.

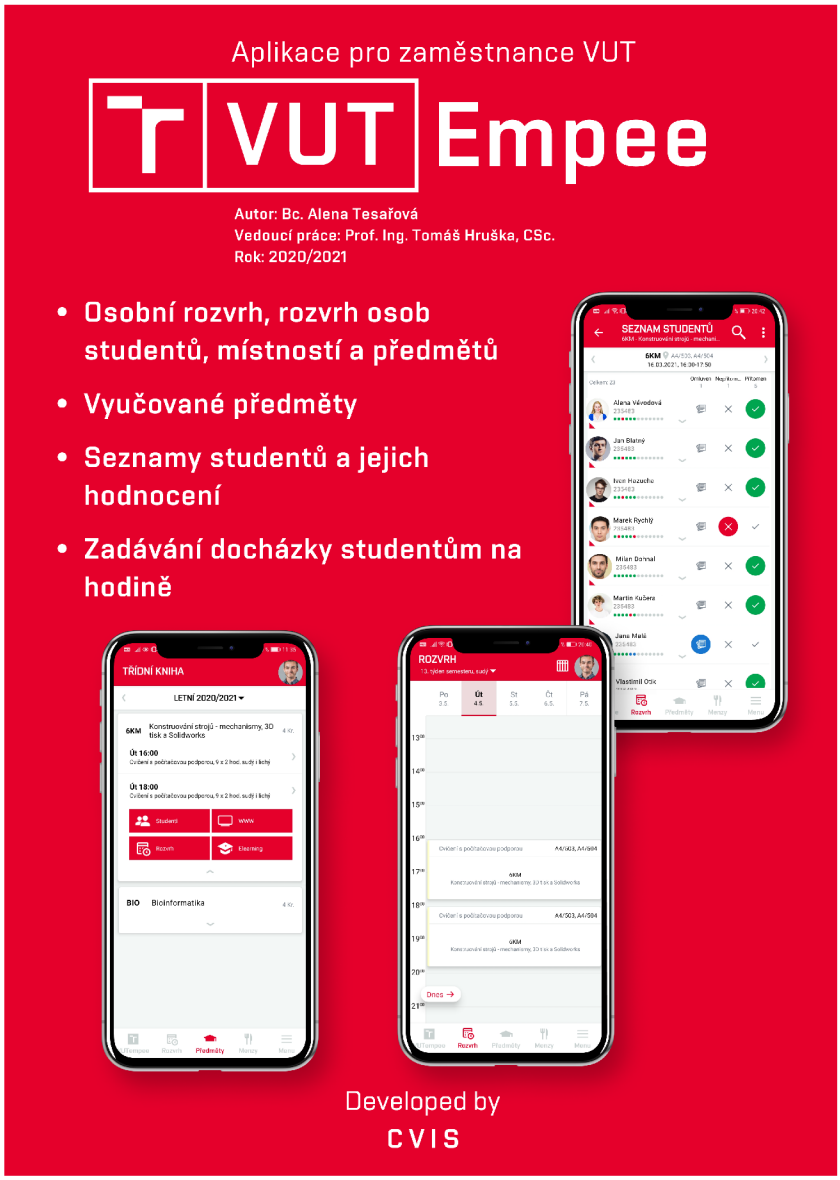
[32] Shjut, K. *States in React Native Components* [online]. 2017 [cit. 2021-5-4]. Dostupné z: http://rationalappdev.com/state-in-react-native-components.

[33] VUT. *Statut VUT* [online]. 2016 [cit. 2020-12-13]. Dostupné z: https://www.vutbr.cz/uredni-deska/vnitrni-predpisy-a-dokumenty/-d128667/statut-vut-p120424.

[34] Wikipedia. *React (web framework)* [online]. Wikipedia, prosinec 2020 [cit. 2020-12-26]. Dostupné z: https://en.wikipedia.org/wiki/React_(web_framework).

Příloha A

Plakát



Obrázek A.1: Plakát VUT empee.