FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	ČVUT navigátor - iOS III
Student:	Daniel Březina
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vytvoření 3. generace ČVUT navigátoru podporující vektorový formát map.
1. Analyzujte možnost navigačního systému Galileo pro lokalizaci uživatele.
2. Analyzujte předchozí verzi ČVUT Navigátoru a navrhněte změny v implementaci, aby byla podporována zařízeními s iOS 10+.
3. Implementujte určení polohy pomocí systému Galileo a pomocí rozpoznávání značek.
4. Implementujte funkcionality podporující vektorové indoor a outdoor mapy, offline režim, nalezení nejkratší cesty a rozšířenou realitu.
5. Klienta podrobte vhodným testům.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 6. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

ČVUT navigátor - iOS III

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

10. května 2018

Poděkování

V prvním řadě bych rád poděkoval Ing. Jiřímu Chludilovi za perfektní vedení práce a šťouravé dotazy, které vedly ke zlepšení práce. Dále bych rád poděkoval Michalu Maněnovi, který mě uvedl do technické stránky aplikace a Danielu Lopourovi za skvělé informace ohledně systému Galileo. V neposlední řadě bych rád poděkoval rodině a kamarádům za podporu

Prohlášení

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

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

V Praze dne 10. května 2018 .....

Odkaz na tuto práci

Březina, Daniel. *ČVUT navigátor - iOS III.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Mobilní aplikace ČVUT Navigátor slouží pro orientaci v kampusu ČVUT a k zobrazení rozvrhu hodin. Cílem práce je vytvoření 3. generace ČVUT Navigatoru podporující vektorový formát map, navigování uvnitř budov pomocí rozšířené reality a analyzovat možnosti navigačního systému Galileo. Práce analyzuje předchozí verzi ČVUT Navigátoru pro operační systém iOS a navrhuje změny, tak aby třetí verze odpovídala aktuálním požadavkům na aplikace. Součástí práce je také návrh architektury aplikace, konkrétně popis návrhové vzoru Model-View-Controller a detailně rozepsány jednotlivé složky. Další součástí je popis implementace hlavních funkcionalit, jako je nalezení nejkratší cesty a její následné zobrazení v rozšířené realitě a přehled provedených testů.

Klíčová slova ČVUT Navigátor, iOS, Klient, Vektorové mapy, Navigace, Rozvrh hodin, Rozšířená realita, Offline

Abstract

Mobile application CTU Navigator is used for orientation around CTU campus and for displaying timetable. Main goal of this thesis is to create 3rd generation of CTU Navigator application, that supports vector format of maps, navigates inside of buildings using augmented reality, and to analyze possibilities of navigation system Galileo. This thesis analyse previouse version of CTU Navigator for operating system iOS and suggests changes to be made. Thesis also includes design of software architecture, specifically it is Model-View-Controller and description of all parts. There is also description of implementation of main functions, such as finding shortest path and displaying it in augmented reality and overview of tests.

Keywords CTU Navigator, iOS, Client, Vector maps, Navigation, Timetable, Augmented reality, Offline

Obsah

Seznam obrázků

Úvod

Aplikace ČVUT Navigátor slouží pro studenty ČVUT, kteří se neorientují po kampusu. Po rozpoznání uživatelovy polohy (uvnitř budovy pomocí naskenované značky, mimo budovu pomocí signálu GPS) a zadání cílové učebny ukáže aplikace nejkratší cestu do vybrané učebny (uvnitř budovy i v rozšířené realitě). Aplikace dále zobrazí uživateli jeho rozvrh a dokáže automaticky navrhnout cílovou destinaci pro navigování. Samotný rozvrh se dá poté importovat do interního kalendáře. Pro orientaci v okolí bude aplikace znát okolní budovy a pomocí rozšířené reality bude schopna zobrazit názvy budov v reálném čase v obrazu z kamery. Jelikož aplikace bude pracovat s citlivými daty (uživatelův rozvrh hodin), může ji uživatel uzamknout a přistupovat k ní přes heslo zadávané pomocí biometricých čidel.

Rozbor zadání

Cílem práce je vytvoření 3. generace ČVUT navigátoru podporující vektorový formát map.

Analyzujte možnost navigačního systému Galileo pro lokalizaci uživatele.

V dnešní době existuje několik navigačních systémů a jedním z nich je i evropská služba Galileo[9]. Součástí rešeršní práce bude srovnání tří hlavních systémů - GPS[10], Galileo a GLONASS[11]. V praktické části práce, kterou je aplikace pro operační systém iOS, bude možnost navigování pomocí kombinace systémů GPS + Galileo + GLONASS.

Analyzujte předchozí verzi ČVUT Navigátoru a navrhněte změny v implementaci, aby byla podporována zařízeními s iOS 10+.

Rešeršní část práce se z velké části bude zabývat změnami mezi druhou verzí a třetí verzí aplikace. Časový rozdíl mezi oběma verzemi je 5 let, což je

na tak mladý obor, jakým je vývoj aplikací pro mobilní zařízení, dlouhá doba, ve které se událo plno změn.

Malou změnou zadání bude podporovaná verze operačního systému iOS - aplikace bude v době odevzdání bakalářské práce podporována zařízeními s operačním systémem iOS 11, a bude plně připravena na podporu nové verze operačního systému iOS 12, který bude představen 4. června 2018 na konferenci *WWDC*[12]. Důvodem změny je, že interní knihovnu *ARKit* podporuje pouze operační systém iOS 11 a pomocí této knihovny se řeší rozšířená realita použita v práci. Pro další vývoj aplikace je lepší použít interní knihovnu od firmy Apple, protože bude nejlépe optimalizovaná pro zařízení s operačním systémem iOS a bude mít zařízenou oficiální podporu.

Implementujte určení polohy pomocí systému Galileo a pomocí rozpoznávání značek.

Aplikace bude umět rozpoznat polohu uživatele pomocí systému Galileo. Rozpoznání polohy uživatele pomocí systému Galileo bude možné pouze na modelech iPhone 8/8Plus/X, protože jedině tyto zařízení mají čip pro příjem signálu tohoto systému[13]. Pro rozpoznání polohy uvnitř budov bude použita metoda z druhé verze aplikace - naskenování a rozpoznání značek.

Implementujte funkcionality podporující vektorové indoor a outdoor mapy, offline režim, nalezení nejkratší cesty a rozšířenou realitu

Aplikace bude fungovat bez nutnosti připojení k internetu od prvního spuštění. Tím pádem bude potřeba nalézt speciální knihovnu pro zobrazení offline vektorových map z vlastních zdrojů. Navigování uživatele pomocí rozšířené reality bude novinkou ve třetí verzi. Bude se jednat o experiment, protože knihovna ARKit je stále ve vývoji a není plně optimalizovaná. Aplikace bude podrobena uživatelský testům v přirozeném prostředí (v budově Fakulty informačních technologií) a bude obsahovat jednotkové testy modelové části aplikace.

Klienta podrobte vhodným testům

Po vytvoření klikatelného prototypu proběhne první test, zda-li jsou UI prvky rozložené tak, aby bylo možné pomocí nich se dostat ke všem funkcionalitám aplikace. Po dokončení implementace se aplikace podrobí uživatelskému testování, kdy se vyzkouší, jestli je možné se pomocí aplikace dostat z určitého bodu do konkrétní učebny, a nakonec se otestuje funkčnost jednotlivých kompoment pomocí jednotkových testů.

Kapitola 1

Analýza

V kapitole Analýza jsou analyzované změny ve vývoji aplikace pro operační systém iOS, dále zde jsou rozebrané tři různé polohové systémy a možnosti jejich využití, popsaná rozšířená realita a s ní spojená knihovna *ARKit* a nakonec jsou aktualizovány funkční a nefunkční požadavky.

1.1 Existující verze

Předchozí verze aplikace ČVUT Navigátor[14] byla vytvořena v roce 2013. Aplikace byla napsaná v *Objective-C*, což byla tehdy jediná možnost jak vytvářet aplikace pro operační systém iOS. Aplikace také podporovala pouze iPhony 3GS/4/4S/5 a 5S, který byl představen v září 2013. V dnešní době je aplikace téměř nepoužitelná. Obsahuje zastaralé knihovny, které nahrazují jednoduché úkoly jako je například hlídání klávesnice, aby nezasahovala do aktivní oblasti. Z tohoto a dalších důvodů uvedených v sekci Změny ve vývoji je potřeba aktualizovat aplikaci, tak aby byla použitelná v dnešní době a vyhovovala dnešním standardům.

Na operačním systému Android je už třetí verze vytvořena[15]. Aplikace je uzpůsobena pro operační systém Android a tím pádem se bude s verzí pro operační systém iOS shodovat pouze ve funkčních požadavcích. Aplikace pro iOS dokáže ještě navíc ukázat nejbližší budovy pomocí rozšířené reality.

1.2 Změny ve vývoji

Druhá verze ČVUT Navigátoru byla dokončena v červnu 2013. O rok později firma Apple představila svojí vizi pro budoucnost vývoje iOS aplikací v čele s novým programovacím jazykem *Swift*[16] a v září roku 2014 poprvé vydala dvě nové verze iPhonu, iPhone 6 s uhlopříčkou 4.7 palců a iPhone 6 Plus s uhlopříčkou 5.5 palců, na které bylo potřeba zareagovat změnou ve vývoji a uzpůsobit staré i nové aplikace. Každý rok je také vydaná nová verze operač-

ního systému iOS s novými funkcemi a novými knihovnami. V následujících podsekcích budou rozebrány hlavní změny ve vývoji třetí verze aplikace ČVUT Navigátor a jejich přehled je v tabulce 1.1.

Tabulka 1.1: Přehled změn ve vývoji

Změna
Programovací jazyk Swift
Podpora všech iPhonů s iOS 11
Vytváření UI
Minimalizace použití knihoven třetích stran

1.2.1 Programovací jazyk Swift

Nejzásadnější změnou ve třetí verzi aplikace bude použití nového programovacího jazyka Swift, konkrétně verze Swift 4. Firma Apple vytvářela Swift za účelem nahradit starší programovací jazyk Objective-C (vznikl v roce 1986), ale již nechtěli vycházet z jazyka C. V jazyce Swift najdeme klasické datové typy jako jsou *Int*, *String*, *Double*, *Float* nebo klasické kolekce - *Array*, *Dictionary*, *Set*[17]. Můžeme také využít i datové typy z Objective-C, jako jsou *NSString*, *NSArray* atd. Swift podporuje objektový přístup, oproti Objective-C používá prvky funkcionálního programování a přichází i s vlastním přístupem - *protokolově orientovaným programováním*.

Základní charakteristikou pro Swift je bezpečnost a lepší čtení kódu. Swift představil nový přístup k proměnným, které mohou být nealokovány (*nil*). Vytvořil nadstavbu datových typů, které se nazývají *Optional*[18] (např.: *Optional(Int)*). Datový typ *Optional* programátorovi říká, že daný typ může obsahovat *nil* a musí se k proměnné v kódu přistupovat speciálně. Tímto přístupem se minimalizuje přístup k paměti, kterou programátor nealokoval, protože kompilátor nahlásí chybu už při kompilaci. Dále je Swift silně typovaný. To znamená, že proměnná, pokud je jednou inicializovaná, už nemůže změnit svůj datový typ. Swift zároveň dokáže při inicializaci zjistit datový typ bez toho, aniž by musel být explicitně určen programátorem.

Syntax jazyka Swift má blíže k programovacímu jazyku Python než k Objective-C. Ve verzi 3.0 byl představen nový přístup k vytváření rozhraní funkcí tak, aby se co nejvíce blížili klasickému (anglickému) jazyku[19]. Ve zkratce jde o to, aby se v názvu funkce neobjevovala slova, která jsou použita pro názvy argumentů funkcí. Pro argumenty se rozlišují *labely* a názvy argumentů, např.: *func add(into array: [Int])*, kde *func* je klíčové slovo, *add* je název funkce, *into* je label a *array* je název argumentu, se kterým se bude pracovat v těle funkce. *[Int]* znázorňuje datový typ pole celých čísel. Celkově je kód v jazyce Swift kratší než v Objective-C.

Swift je rozšířen o možnosti používání *enum* - výčtových typů, dále obsahuje struktury *struct*, které se oproti třídám *class* předávají funkcím hodnotou (tzn. vytvoří se nová kopie, která zanikne s koncem těla funkce). V neposlední řadě je rozšířená funkčnost přepínače *switch*, který může rozlišovat komplexnější podmínky (např. intervaly). Všechny tyto změny se dají využít v novém přístupu k programování, kterým je *protokolově orientované programování*[20]. Protokoly jsou šablonou pro danou funkčnost, které se dají v jazyce Swift rozšiřovat a slouží k lepší znovupoužitelnosti kódu a k lepšímu pochopení kódu. Oproti objektově orientovanému přístupu se nemusí kvůli nové funkčnosti vytvářet nová třída, která konkrétně tuto funkčnost využije. Naopak se nemusí k existující třídě (ze které mohou dědit další třídy) přidávat funkčnost, kterou využije pouze daná třída a třídy z ní vycházející už funkčnost nepotřebují. Každá *třída/enum/struct* musí, pokud chce protokol adaptovat, implementovat všechny povinné funkce z protokolu. Dále mohou být funkce nepovinné. Aktuálně se dá protokolově orientovaný přístup využít ve vytváření iOS aplikací k rozšíření objektového přístupu. Knihovny potřebné k vytváření aplikací ještě plně nepodporují protokolově orientované programování.

Swift byl zpočátku vyvíjen interně ve firmě Apple, ale od začátku se počítalo s tím, že se po vypuštění přestoupí na *opensource*. Vývoj probíhá neustále a prozatím s každou novou verzí iOS vychází i nová verze jazyka Swift. Nejprve šlo o velké změny tak, aby velká část programátorů byla s jazykem spokojena. Od verze 3, kdy proběhla největší změna syntaxe, se provádějí změny hlavně uvnitř jazyka a tím pádem přechod na novou verzi není náročný. Apple dodává do svého vývojového nástroje *XCode konvertor*, který dokáže převést syntax na aktuální verzi jazyka[21].

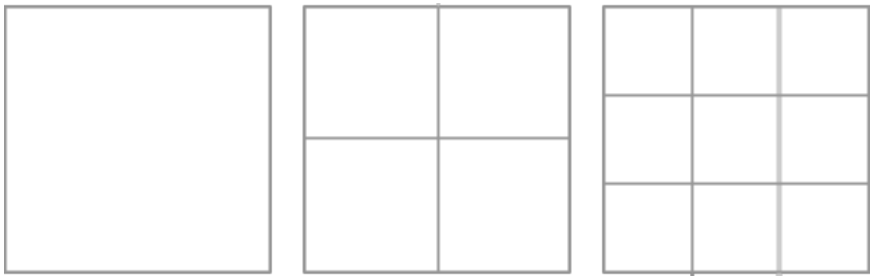
Apple dovoluje používat jazyk Swift v jednom projektu s jazykem Objective-C[22]. Pokud je potřeba použít kód napsaný v Objective-C ve Swift kódu, vytvoří se speciální hlavičkový soubor *bridging-header.h*, u kterého bude kompilátor vědět, že v něm jsou obsaženy třídy, které jsou napsané v Objecitve-C, a převede jejich syntax na syntax jazyka Swift.

1.2.2 Podpora všech iPhonů s iOS 11

Druhá verze ČVUT navigátoru byla vytvářena pro iPhony s úhlopříčkou 3.5 palce a 4 palce (iPhony 3GS, 4, 4S, 5 a 5S), bez použití nástroje pro pozicování UI kompoment *AutoLayout*. To znamená, že na větších úhlopříčkách nebude aplikace vypadat dobře a UI komponenty budou buď rozházené po displeji, nebo budou shlukovány v levém horním rohu. Třetí verze bude plně podporovat *AutoLayout* a díky tomu bude aplikace z pohledu UI plně optimalizovaná pro všechny aktuálně nabízené iPhony.

S novými velikostmi obrazovek přišly nové displeje a také přišla změna v používání obrázků. Buď se mohou použít vektorové obrázky, kterým se mohou měnit rozměry bez ztráty kvality, nebo se musí rozšířit knihovna obrázků o další velikosti. Pro jeden obrázek jsou potřeba dvě velikosti. Jedna velikost

s trojnásobným rozlišením (označováno @3x) pro iPhony 6Plus/6S Plus/7 Plus/8 Plus/X se *Super Retina* displejem a druhá velikost s dvojnásobným rozlišením (@2x) pro všechny ostatní podporované iPhony s *Retina* displejem. Základní velikost(@1x) byla historicky používaná pro původní iPhony (do modelu 3GS včetně), s iPhonem 4 přišel *Retina* displej, který se vyznačuje tím, že pro každý pixel se přidá další pixel v každém směru. Ukázka je na obrázku 1.1. S iPhonem 6 Plus, vydaným v roce 2014, se pro každý pixel přidají další dva pixely[1].



Obrázek 1.1: Znázornění jednoho pixelu na klasickém displeji, *Retina* displeji a *Super Retina* displeji[1]

Firma Apple si diktuje podmínky pro aplikace, které chtějí být umístěny v obchodě s aplikacemi *App Store*, a jednou z podmínek je, že od dubna 2018 musí všechny aplikace být napsány za pomoci *iOS 11 SDK* a musí podporovat *Super Retina* displeje, které má iPhone X[23].

1.2.3 Vytváření UI

Při vytváření iOS aplikací, konkrétně UI aplikace, se může použít speciální soubor nazvaný *Storyboard*, kam se můžou vkládat UI prvky a pozicovat pomocí nástroje *AutoLayout*, popřípadě dále upravovat část vzhledu. Výhodou *Storyboardů* je, že programátor ihned vidí, jak bude výledek vypadat. Nevýhod je ale více. Ve *Storyboardu* se dá udělat pouze pár základních úprav UI kompomenty (barva, font atd.), zbytek se musí dodělat v kódu. Vzniká tím nepořádek a programátor nemá přehled o tom, kde danou úpravu najde. Dále je nevýhodou nepřehlednost. Už u projektu střední velikosti, jako je například ČVUT Navigátor, je *Storyboard* velmi nepřehledný a také velmi pomalý při každém otevření souboru. Proto jsem se rozhodl používat při programování třetí verze druhý přístup vytváření UI, a tím je vytváření UI pouze v kódu. Kod se tím sice rozšíří, ale zároveň se budou dát určité kompomenty použít opakovaně (např.: tlačítka). V kódu nebudou použity přímo funkce pro *AutoLayout* od firmy Apple, ale využije se knihovna třetí strany *SnapKit*[24]. Knihovna *SnapKit* je průběžně vyvíjená skupinou vývojářů a má tradici od počátku nástroje *AutoLayout* (v té době se knihovna jmenovala *Masonry*, název

SnapKit přišel s jazykem Swift). Výhodou knihovny *SnapKit* je přímočařejší syntax a lepší čitelnost kódu.

1.2.4 Minimalizace použití knihoven třetích stran

Ve druhé verzi je použito mnoho knihoven třetích stran, které buď řeší lehké úkoly, které stejným nebo lepším způsobem řeší i knihovny od firmy Apple, nebo již nejsou vyvíjeny a udržovány. Ve třetí verzi se použije výše zmíněná knihovna *SnapKit* na *AutoLayout*, která je dlouhodobě vyvíjena a udržována, a reaguje každý rok na novou verzi jazyka Swift[25].

Ze seznamu použitých knihoven jsem vyřadil knihovnu *Route-Me*[26], která stahovala a zobrazovala mapy z *OpenStreetMap*, a nahradil jsem jí knihovnou *Carto Mobile SDK*[27]. Knihovna *Route-Me* byla zastaralá a nesplňovala požadavek zobrazovat offline vektorové mapy. Knihovna *Carto Mobile SDK* je alternativa, která nabízí kromě placené služby i služby zdarma (mezi které patří právě zobrazení offline vektorových map) a hlavně je plně rozvíjena a má kvalitní podporu.

Pro přidání knihoven do projektu se použije *dependency manager* nazvaný *CocoaPods*[28]. *CocoaPods* je opensource projekt vyvíjený od srpna 2011. V souboru s názvem *Podfile* se specifikují jednotlivé knihovny, buď v konkrétních verzích, nebo s rozsahem verzí (použít jakoukoliv verzi, která se vydá, nebo pouze verze, které budou v aktuální verzi a bude se lišit pouze o desetiny, atd.). Knihovny se nainstalují a nakonfigurují v projektu ve vývojovém prostředí *XCode* a vytvoří se nový *projekt*, který obsahuje pouze nainstalované knihovny. Zároveň se vytvoří *workspace* (soubor obsahující více XCode projektů), který obsahuje původní projekt s iOS aplikací a nový projekt s nainstalovanými knihovnami a jejich propojení.

1.3 AutoLayout

AutoLayout je nástroj pro dynamické pozicování UI elementů na obrazovce mobilního zařízení. Představen byl v roce 2013 jako alternativa k tehdejšímu způsobu pozicování (poloha podle souřadnic) a plně rozšířen byl v roce 2014, kdy fimra Apple představila nové velikosti iPhonů a *AutoLayout* se stal standardem.

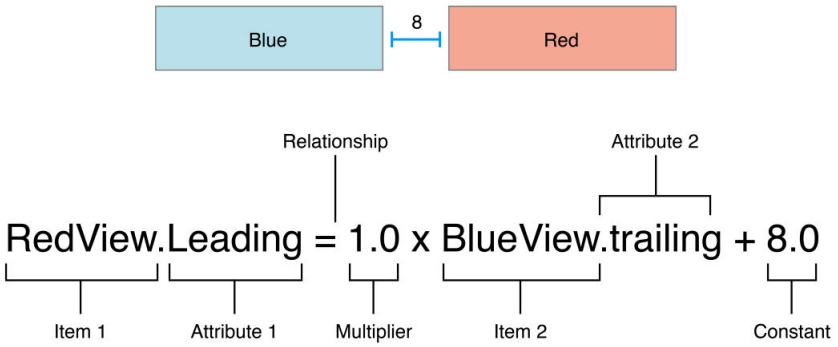
Programátor specifikuje polohu a atributy UI elementu vůči ostatním prvkům pomocí jednoduchých omezení a *AutoLayout* vyhodnotí specifikaci, která by měla mít právě jedno řešení, kterým je poloha UI elementu. Jedno omezení se skládá z levé a pravé strany, které k sobě mají určitý vztah (rovná se, větší nebo rovno, a menší nebo rovno). Levá strana se skládá z následujících částí:

- UI element, který se pozicuje
- atribut elementu, na který se vztahuje dané omezení (horní část, spodní část, střed elementu atd)

Pravá strana se skládá z následujících částí:

- násobitel, implicitně nastaven na 1x. Hodnota pravé strany je tímto násobitelem vynásobena
- UI element, vůči kterému je pozicováno
- atribut elementu, na který je pozicováno (typicky opačný k levému atributu, pokud nepozicujeme element ke kraji obrazovky)
- konstanta, která značí posunutí vůči pravému atributu

Celá rovnice je na obrázku 1.2 [2].

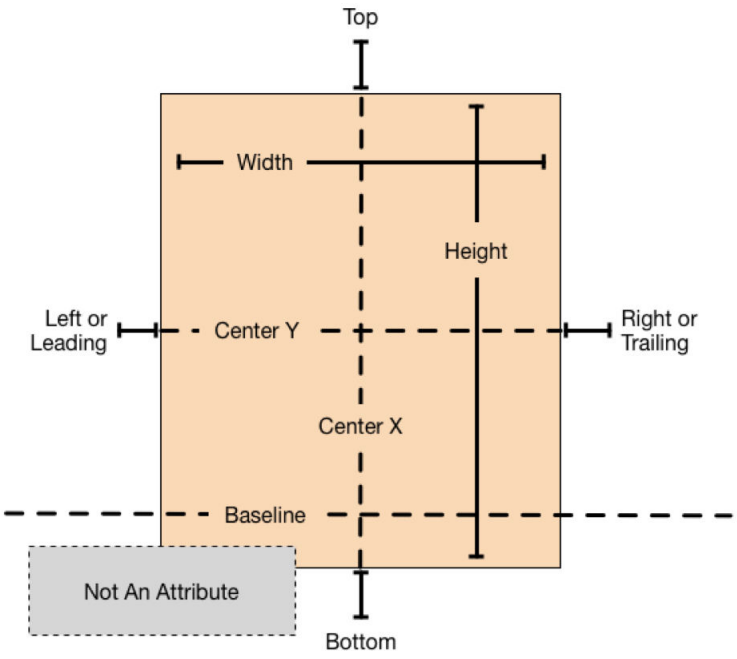


Obrázek 1.2: Pozicování v *AutoLayout* vizuálně[2]

1.3.1 Atributy

UI element se pozicuje pomocí atributů. Každý element jich má několik. Základními jsou atributy *Top*, *Bottom*, *Left* a *Right*. K *Left* a *Right* existují ještě dvě alternativy, a sice *Leading* a *Trailing*. Slouží k lokalizování aplikace pro národy, které mají odlišné způsoby čtení textu. *Leading* značí počátek a *Trailing* značí konec. Pro Českou republiku je tedy *Leading* jako *Left* a *Trailing* jako *Right*, ale například v arabských zemí je *Leading* jako *Right* a *Trailing* jako *Left*. Při použití těchto atributů se nemusí dále programátor starat o směr textu pro různé národnosti.

Mezi další atributy patří výška (*Height*) a šířka (*Width*). Existují určité UI elementy (např. *UILabel*), které si spočítají tyto atributy podle svého obsahu. Většina ostatních UI elementů ale musí mít tyto dva atributy definovány. Pokud programátor specifikuje u UI elementu atributy *Leading* a *Trailing* a zároveň i atribut *Width*, pak nástroj *AutoLayout* nedokáže jednoznačně určit polohu a vyhodí varování. Stejné to je u atributů *Top* a *Bottom* společně s atributem *Height*. Posledními atributy jsou pozice na osách X a Y (*CenterX* a *CenterY*) a základní čára (*Baseline*) [2]. Přehled atributů je na obrázku 1.3.



Obrázek 1.3: Atributy pro *AutoLayout* [2]

1.3.2 Layout Guide

UILayoutGuide je rozhraní představené s uvedením iOS9 v roce 2015. Je to ohraničená oblast na obrazovce, která se ale nerenderuje, slouží pouze k orientaci pro další elementy. Speciálními případy byly *topLayoutGuide* a *bottomLayoutGuide*. V operačním systému iOS jsou dva typy obrazovek. Jedna klasická bez navigační lišty v horní části a jedna s navigační lištou. Pozice vůči *topLayoutGuide* zaručí, že obsah obrazovky bude viditelný na obou typech a nebude překrýván lištou. Podobně je to s *bottomLayoutGuide*, kdy se spodní část chová podle toho, jestli je viditelná lišta s jednotlivými okny aplikace (*tabBar*). *TopLayoutGuide* a *bottomLayoutGuide* byly v operačním systému iOS 11, který byl vydán v roce 2017, změněny na *safeAreaLayoutGuide*, protože se velikost navigační lišty mění dynamicky (například při posouvání obrazovky). Tento důvod byl známý v době představení iOS 11. V září roku 2017 Apple představil nový iPhone X, na kterém změnil kompletně celý displej. Rohy nejsou hranaté, ale kulaté, a v horní části je obrazovka rozdělena tzv. *notchem*, výřezem, ve kterém je zabudovaná *TrueDepth* kamera a další senzory a mikrofony. Navigační lišta je tedy větší než na ostatních iPhonech. Aplikacím, které používají *AutoLayout*, stačí nastavit, aby se elementy pozicovaly k *safeAreaLayoutGuide* a aplikace se bude správně zobrazovat i na iPhone X. [**?**]

1.4 Rozšířená realita

Rozšířená realita je reálný obraz snímaný fotoaparátem rozšířený o digitálně vytvořené objekty (obrázek 1.4). V mobilních zařízeních byla plně rozvinuta v roce 2017 příchodem knihovny *ARKit* od společnosti Apple. Možnosti uplatnění rozšířené reality jsou obrovské. Od výukových aplikací, přes hry, až po podávání informací o snímaném okolí. Zobrazení cesty je ideální použití pro rozšířenou realitu. Uživateli se promítne cesta na displeji jeho mobilního zařízení a on se po ní může vydat a kontrolovat směr, aniž by musel hledat na mapě kde je a jakým směrem se dát.

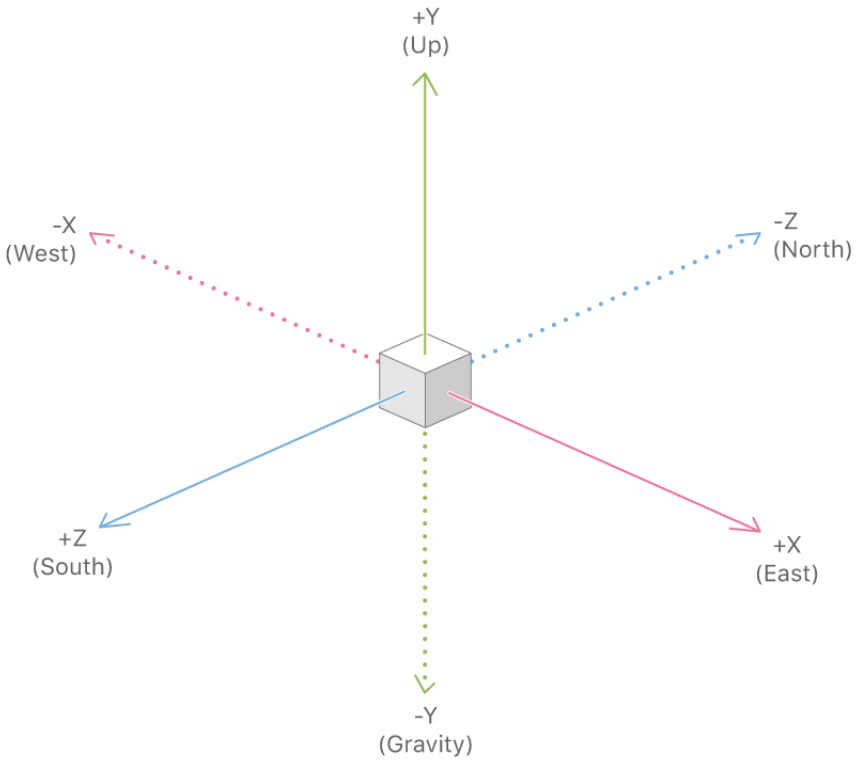
Obrázek 1.4: Ilustrační obrázek použití rozšířené reality[3]

1.4.1 Knihovna ARKit

Knihovna *ARKit*[29] byla představena v červnu 2017 na konferenci WWDC a funguje na modelech iPhone 6S/6S Plus a novějších, s nainstalovaným operačním systémem iOS 11. Pro vykreslování objektů se dají použít další grafické knihovny od firmy Apple jako je *SpriteKit*[30] (pro 2D grafiku), *SceneKit*[31] (pro 3D grafiku) a *Metal*[32] (knihovna pro komunikaci s grafickou kartou, obdoba OpenGL). Knihovna *ARKit* využívá techniku nazvanou *visual-inertial odometry*[33] ke kombinování informací ze senzorů telefonu s obrazem z kamery. Z kamery dokáže knihovna *ARKit* rozeznat spousty bodů ve scéně a následně s nimi pracovat. Základní vlastností knihovny *ARKit* je rozpoznání horizontálních a vertikálních ploch, na které je poté možno umístit digitální objekty tak, aby vypadaly, že sedí přesně na dané ploše. Pro vkládání objektu se používá třída *ARAnchor*[34], která se stará o správné pozicování objektu v průběhu času vypočítáváním transformačních matic. Jelikož je knihovna *ARKit* nová, ještě není plně optimalizovaná a digitální objekty ve scéně nemusí pevně "držet". Pro co nejlepší výsledek doporučuje firma Apple vytvářet aplikace, které počítají s dobrým světlem. V temné scéně se špatně hledají význačné body a špatně se detekují plochy. Dále je doporučováno informovat

uživatele o průběhu detekování okolí. Hledání ploch není instantní záležitost, proto velmi záleží na okolí, jak dlouho bude trvat.[33].

Jedním ze senzorů, který dodává informace, je i kompas. Při konfigurování detekování okolí, před samotným spuštění, se dá nastavit, jestli bude kompas využíván. Pokud ne, bude se počátek scény měnit s každým pohybem telefonu. Negativní část osy Y bude vždy směřovat tam, kam směřuje spodní část telefonu. Pokud bude kompas využíván, nabízí se dvě možnost. První možností je vzít v úvahu gravitaci. V tom případě bude negativní část osy Y vždy směřovat k zemi, a osa X, k ní kolmá, se odvodí při spuštění snímání okolí tak, aby směřovala napravo od mobilního zařízení. Druhou možností je vzít v úvahu gravitaci a magnetický sever. Opět negativní část osy Y směřuje k zemi. Nyní ale negativní část osy Z směřuje k severu nezávisle na pozici telefonu při spuštění detekování okolí (obrázek 1.5). K určení severu je potřeba, aby měl uživatel zapnutou tuto možnost v nastavení telefonu. [4]



Obrázek 1.5: Počátek při detekování okolí za pomoci kompasu [4]

Aplikace ČVUT Navigátor využije rozšířenou realitu k zobrazení názvů budov v okolí uživatele a hlavně k zobrazení trasy mezi dvěma body při navigování v uzavřených prostorech. Aplikace nebude potřebovat detekovat body v okolí a ani nebude hledat plochy na zemi. Díky tomu bude fungovat i v míst-

nostech se špatným osvětlením. Umístění bodů bude počítáno ze světových souřadnic a kvůli tomu bude potřeba použít kompas k určení severu. I tak je potřeba zdůraznit, že knihovna *ARKit* se stále vyvíjí, stejně jako hardware pro její použití (výsledky jsou rozdílné na různých typech iPhonů), a proto se výsledné zobrazení může chovat po každém spuštění jinak (cesta se může vychylovat ze svého směru o pár stupňů). Firma Apple 4. června představí další verzi knihovny a je dost pravděpodobné, že s další verzí budou výsledky o mnoho lepší.

1.5 3D Touch

V roce 2015 představila firma Apple nový iPhone 6S a 6S Plus s funkcí *3D Touch*[35]. Šlo o rozšíření *multi-touch* displeje. Do té doby bylo možné na *multi-touch* displejích rozpoznávat dotyk na osách X a Y. U funkce *3D Touch* se dá rozpoznávat i hloubka stisku. *3D Touch* je přítomen na všech nových modelech, kromě modelu SE. Funkce *3D Touch* je použita k usnadnění a zrychlení používání aplikace. Klasicky se používá u UI elementů, které slouží k přechodu "směrem dopředu". Můžou to být tlačítka, obrázky (zobrazení jejich plné velikosti) nebo jednotlivé buňky v seznamech.

1.5.1 Peek a Pop

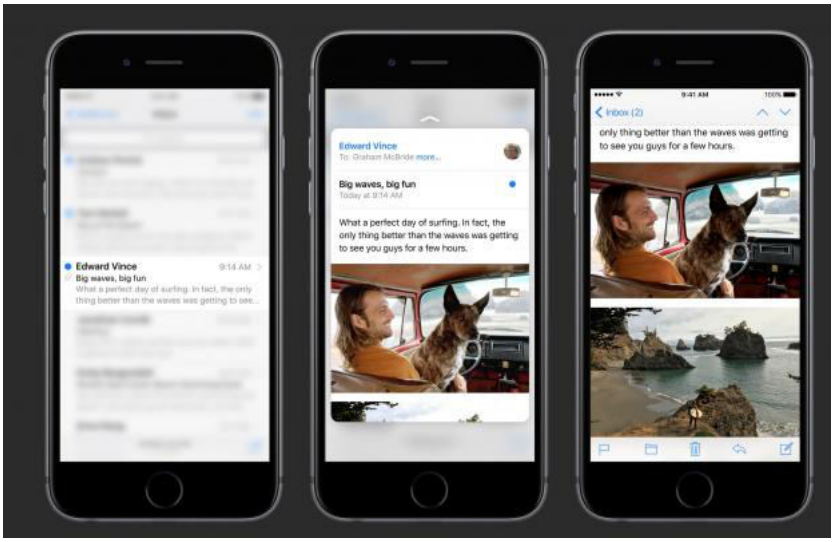
Společně s *3D Touch* byla představena dvě nová gesta - *Peek* a *Pop*[35]. *Peek*, neboli náhled, ukáže při lehkém stlačení náhled bez opouštění aktuální obrazovky. Používá se například u webových stránek, kdy se přitláčí na odkaz a vyskočí náhled stránky. Po přerušení gesta se náhled schová. Při držení gesta a následném posunu směrem nahoru může vyskočit kontextové menu s dalšími akcemi. Po dokončení stisku se uskuteční *Pop*, tedy přesun na požadovanou stránku, obrazovku atd. Ukázka gest je na obrázku 1.6.

1.5.2 Rychlé akce

Další použití *3D Touch* je na hlavní obrazovce. Po silném stisku na ikonu aplikace se objeví přehled rychlých akcí. Rychlé akce mohou být dynamické, to znamená, že se budou měnit při používání aplikace, např. v interní aplikaci *Zprávy* jsou rychlé akce seřazeny podle posledních konverzací. Nebo mohou být statické, neměnné. Pokud aplikace má widget, tak se společně s rychlými akcemi zobrazí i dostupný widget. [35]

1.6 Server

Aplikace stahuje některá data ze serveru, který byl předmětem bakalářské práce[36] v roce 2017. Server je v posledních měsících neudržovaný, nemá implementováno přihlášení pomocí ČVUT účtu a tudíž nemá data o rozvrhu



Obrázek 1.6: Ilustrační obrázek použití gest Peek a Pop [5]

hodin pro uživatele. Z tohoto důvodu nebude použit tento server, ale vytvoří se jednoduché rozhraní na *firebase.google.com*[37] pouze pro testovací účely aplikace. Tento server bude umět nabídnout testovací rozvrh hodin a několik aktuálních zpráv. Díky implementaci komunikace mezi aplikací bude potřeba použít dočasně další knihovnu třetí strany a tou bude knihovna *Firebase*. Jelikož je komunikace se službou *Firebase* odlišná od klasického posílání požadavků, bude v aplikaci také implementována služba pro klasické posílání dotazů pomocí třídy *URLSession*[38] a až se připraví serverová část aplikace, tak se pouze nahradí volání požadavků.

1.6.1 Komunikace se serverem v iOS

Ve standardní knihovně *Foundation* jsou třídy pro komunikaci se serverem. Hlavní třídou je *URLSession*. Vytvoří se s argumentem *configuration* typu *URLSessionConfiguration*[39]. V konfiguraci se dá nastavit, jestli se mohou data stahovat i na pozadí, když není aplikace aktivní. Pomocí třídy *URLSession* se volají jednotlivé požadavky na server[38]. Volání na server se konfigurují pomocí třídy *URLSessionTask*[40]. Z této třídy vycházejí další podtřídy, a sice *URLSessionDataTask*[41] pro klasické *HTTP GET* požadavky, *URLSessionUploadTask*[42] pro nahrání dat na server (metody *POST* a *PUT*), a konečně *URLSessionDownloadTask*[43] pro stahování souborů ze serveru. Celá komunikace se serverem je asynchronní a vrací data dvěma způsoby. Buď voláním delegátích metod (definovaných v *URLSessionDelegate*[44]), nebo zavoláním bloku kódu s předanými argumenty *data* (data která vrátil server), *response* (obsahující metadata jako např. *status kód*) a *error* (objekt je buď *nil*

pokud vše proběhlo v pořádku, nebo obsahuje informace, proč se požadavek nepodařil).

1.7 Polohové navigační systémy

V současnosti lidé využívají navigace pro dopravu v neznámých venkovních prostorech. Synonem pro navigaci se stalo GPS, což je zkratka pro *Global Positioning System*. GPS ale není jediný navigační systém. V následující sekci budou představeny tři polohové systémy, GPS, Galileo a GLONASS. Na obrázků 1.8 je přehled frekvencí, na kterých jsou vysílány signály ze všech tří systémů.

1.7.1 Přehled

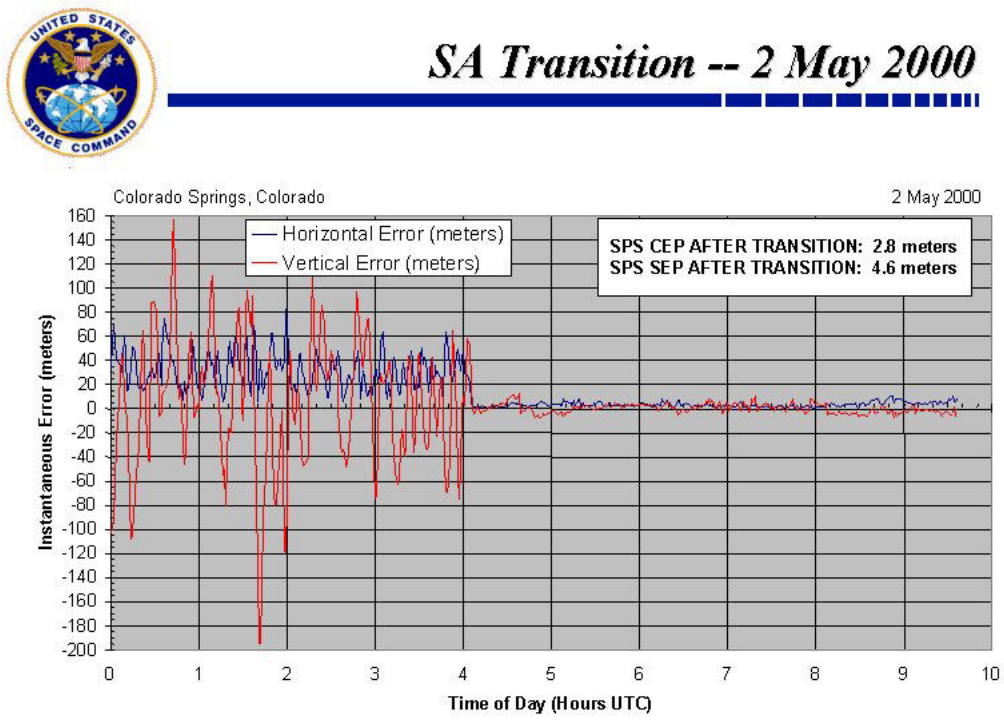
Global Positioning System(GPS) je americký armádní projekt vyvíjen od roku 1973, který je používán i v civilní sféře, a je plně funkční[45]. V dnešní době se nejčastěji připojuje k GPS pomocí mobilních zařízení. Signálem je pokrytá celá zeměkoule a zjištění polohy funguje v každém počasí.

Signál GPS je vysílán na dvou frekvencích - *L1* (1575.42 MHz) a *L2* (1227.60 MHz)[7]. První frekvence je určena pro civilisty, v kombinaci s druhou potom pro armádu. Výhodou přijímání signálu ze dvou frekvencí je, že signál může být matematickými modely očištěn od chyb, které vzniknou v atmosféře (konkrétně v ionosféře, která se nachází v mezosféře a termosféře)[46]. Výsledný signál udává přesnější polohu.

Signál GPS se připravuje na vysílání na nové frekvenci *L5* (1176.45 MHz)[7], která bude sloužit také civilistům. Bude primárně určen pro dopravu a nabídne mnohem větší přesnost signálu (podobně jako Galileo). První družice obíhají kolem Země a výrobci připravují čipy, které dokáží signál na této frekvenci přijmout.

V průběhu 90. let byla implementována *Selektivní dostupnost* (Selective Availability)[6]. Do signálu byla úmyslně zanášena náhodná chyba, která zhoršila přesnost určení polohy pro civilní sektor. Tím se zvýhodňovala americká armáda oproti ostatním zemím. Selektivní dostupnost byla zrušena v květnu roku 2000 na přímý rozkaz tehdejšího amerického prezidenta Billa Clintona. Na obrázku [**?**] je znázorněno zlepšení signálu po zrušení selektivní dostupnosti.

Galileo je evropský civilní polohový systém, který je momentálně ve vývoji, ale část služeb je již přístupná[47]. Galileo je jedinou civilní alternativou k polohovým systémům, všechny ostatní jsou provozovány armádou daného státu. Signál je vysílán na třech frekvencích - *E1* (1575.42 MHz), *E6*(1278.75 MHz) a *E5*(1191.79 MHz)[48]. Nově se začínají dávat čipy na přijímání signálu Galilea i do mobilních telefonů jako jsou nové iPhony 8/8Plus/X[13] nebo Google Pixel 2[49]. Až bude systém plně funkční, bude kolem Země obíhat 24 satelitů zajišťující provoz Galilea, dále 6 rezervních satelitů. Aktuálně obíhá



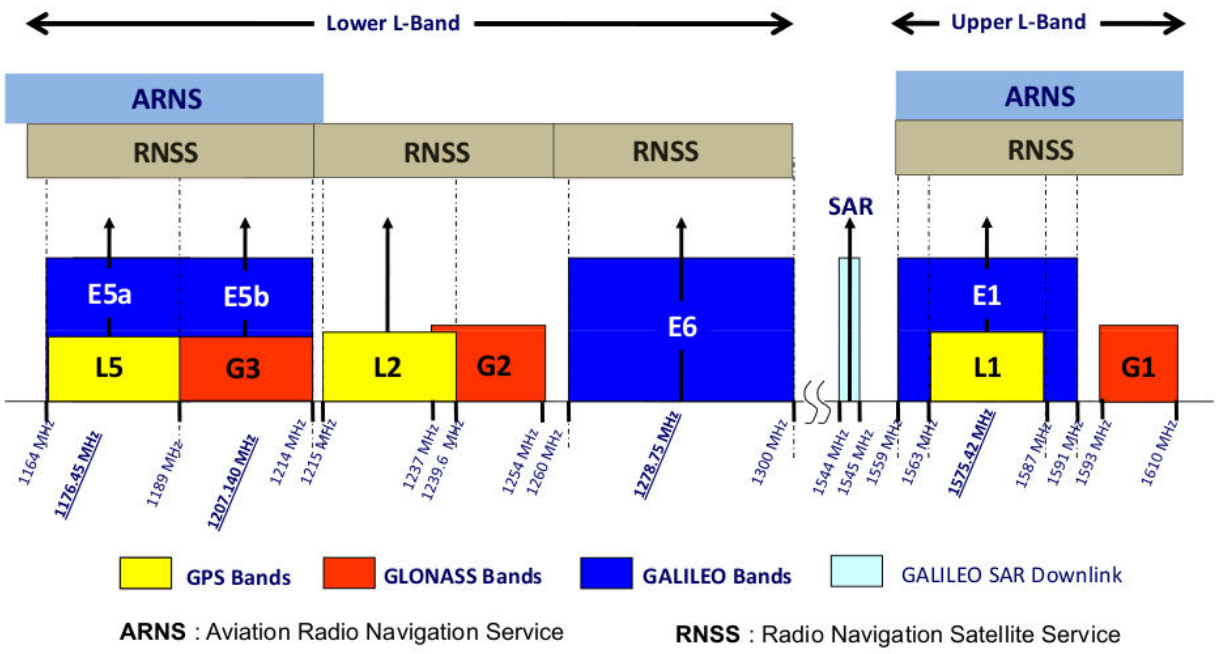
Obrázek 1.7: Graf ukázky zlepšení signálu 2. května 2000 po zrušení Selektivní dostupnosti[6]

kolem Země 22 satelitů, z nichž je 18 funkčních. Předpokládané spuštění ostrého provozu je naplánováno na rok 2020.

GLONASS (Globalnaja navigacionnaja sputnikovaja sistěma) je ruský armádní projekt, který funguje od roku 2010[11]. Podobně jako GPS, i GLONASS povoluje používání civilistům, ovšem s omezenou přesností. GLONASS se skládá ze 24 družic (3 z nich jsou záložní). Hlavním rozdílem oproti GPS (a ostatním polohovým systémům) je, že GLONASS využívá schéma *FDMA* (Frequency Division Multiple Acces)[11], kdežto GPS využívá novější *CDMA*(Code Division Multiple Access). Hlavní rozdíl mezi *FDMA* a *CDMA* je ten, že u *FDMA* se musí frekvence rozdělit do pásem a každé pásmo je přiřazeno určité stanici.

1.7.2 Kombinace polohových systémů

Navigační systém Galileo bude v kombinaci s GPS výrazně vylepšovat aktuální polohu. Podle testů, které provedli v roce 2014 *GSA*(European GNSS Agency) ve spolupráci se soukromím subjektem *RX Networks*, se ukázalo, že v té době vylepšoval systém Galileo přesnost signálu GPS o jednotky procent v těžko dostupných oblastech (silně osídlené městské oblasti, vnitřky budov). Důležité



Obrázek 1.8: Rozpětí frekvencí polohových systémů [7]

je zmínit, že tehdy fungovaly 4 Galileo satelity z avizovaných 30. V tabulkách 1.2 a 1.3 jsou k nahlédnutí výsledky testů, samotný GPS signál a zlepšení v kombinaci s ostatními polohovými systémy. Čísla v tabullkách udávají přesnost určení polohy v metrech a v závorkách je uvedeno zlepšení oproti GPS. I když je zlepšení viditelné, tak stále není (a nebude) možné určovat přesnou polohu s minimální odchylkou uvnitř budov. [50]

Tabulka 1.2: Výsledky testů od GSA a RX Networks v silně obydlených oblastech (městech)

Polohový systém	Obydlená oblast 1	Obydlená oblast 2
GPS	331.9	76.2
GPS + GLONASS	289 (13%)	68.6 (10%)
GPS + Galileo	321.2 (3%)	70.8 (7%)
GPS + GLONASS + Galileo	288.9 (13%)	51.5 (32%)

1.8 Funkční a nefunkční požadavky

V předchozí verzi aplikace ČVUT Navigátor byly definovány funkční a nefunkční požadavky. Třetí verze aplikace přejímá všechny funkční požadavky a přidává nové v reakci na aktuální trendy. U nefunkčních požadavků je více

Tabulka 1.3: Výsledky testů od GSA a RX Networks uvnitř místností

Polohový systém	Místnost 1	Místnost 2
GPS	278.7	70.3
GPS + GLONASS	210.3 (25%)	58.5 (17%)
GPS + Galileo	254.1 (9%)	60.2 (14%)
GPS + GLONASS + Galileo	214.7 (23%)	54.5 (23%)

změn z důvodu kontinuálního vývoje celého systému iOS a zařízení iPhone. Všechny funkční požadavky jsou shrnuty v tabulce 1.4, nefunkční v tabulce 1.5 a dále jsou detailně rozepsány.

Tabulka 1.4: Přehled funkčních požadavků

Funkční požadavky
Outdoor navigace
Indoor navigace
Zobrazení rozvrhu hodin
Informace o předmětech
Synchronizace se systémovým kalendářem
Prohlížení okolí v rozšířené realitě

Tabulka 1.5: Přehled nefunkčních požadavků

Nefunkční požadavky
Podpora iOS 11, iPhone aplikace
Offline použití
Vektorové mapy
Rozšířená realita pro navigování v uzavřených prostorách
Testování Galileo
Zabezpečení pomocí biometrických dat
Podpora 3D Touch
Česká a anglická lokalizace

F1 - Outdoor navigace

Uživatel bude navigován ve venkovních prostorách z výchozího bodu do cílového bodu pomocí signálu GPS. Na mobilním zařízení bude vyznačena cesta. Výchozí stav bude možné určit z aktuální polohy mobilního zařízení.

F2 - Indoor navigace

Uživatel bude navigován po vnitřních prostorách budov pomocí skenování značek u místností. Navigování bude probíhat pomocí rozšířené reality a zobrazení polohy na mapě.

F3 - Zobrazení rozvrhu hodin

Uživatel si bude moci zobrazit svůj aktuální rozvrh hodin s barevně odlišenými typy (přednášky, proseminář, cvičení, zkoušky). Mezi jednotlivými předměty bude vypsaná pauza.

F4 - Informace o předmětech

Uživatel si bude moci zobrazit informace o studovaných předmětech, které budou obsahovat učitele, místnosti, časové rozmezí, typ předmětu a možnost navigace do dané místnosti.

F5 - Synchronizace se systémovým kalendářem

Uživatel si bude moci uložit svůj rozvrh hodin do aplikace Kalendář. Tato data se poté dají zobrazit ve všech aplikacích využívající systémový kalendář.

F6 - Prohlížení okolí v rozšířené realitě

Uživatel, po namíření kamery na okolí kampusu, bude na displeji mobilního zařízení informován o bodech zájmů ve své bezprostřední blízkosti.

N1 - Podpora iOS11, iPhone aplikace

Aplikace bude podporovat všechny operační systémy s iOS 11. V dubnu 2018 tvořil podíl instalace nejnovějšího operačního systému 76 % [51]. Druhý v pořadí, operační systém iOS 10, měl pouze 19 % a ve velké míře se jednalo o uživatele iPhonu 4S, který nemá možnost aktualizace. Aplikace zároveň poběží na všech podporovaných iPhonech (iPhone SE/6s/7/8/6SPlus/7Plus/8Plus/X).

N2 - Offline použití

Aplikace bude plně funkční v offline módu od nainstalování aplikace, včetně navigování. Pouze stažení rozvrhu hodin bude vyžadovat alespoň jedenkrát připojit k internetu od nainstalování aplikace.

N3 - Vektorové mapy

Aplikace využije vektorové mapové podklady z *OpenStreetMaps*[52], které jsou méně náročné na kapacitu úložné jednotky.

N4 - Rozšířená realita pro navigování v uzavřených prostorech

Aplikace použije pro zjištění polohy v uzavřeném prostoru skenování značek, které identifikují místnost. Aktuální směr se ukáže na displeji mobilního zařízení, který bude snímat uživatelovo okolí.

N5 - Testování Galileo

Aplikace umožní testovat přesnost signálu navigačního systému Galileo v kombinaci s GPS a GLONASS uvnitř budov. Uživatel si může přepnout zjišťování polohy mezi polohovými službami a skenováním značek.

N6 - Zabezpečení pomocí biometrických dat

Uživatel si bude moci aplikaci uzamknout pomocí svého hesla k mobilnímu zařízení a poté zadávat heslo pomocí vestavěné čtečky biometrických dat (*TouchID* a *FaceID*).

N7 - Podpora 3D Touch

Aplikace bude podporovat gesta *3D touch*. Na ikoně se objeví možnosti rychlého výběru zobrazení rozvrhu hodin, navigačního formuláře a zobrazení

budov v rozšířené realitě, a v aplikaci budou využívaná nová gesta *Peek* a *Pop*.

N8 - Česká a anglická lokalizace
Aplikace bude obsahovat české a anglické překlady, v závislosti na tom, jaký jazyk má uživatel nastaven v systému iOS.

1.9 Use-case specifikace pro nové nebo upravené požadavky

Use-case specifikace popisuje chování aplikace z pohledu uživatele. Následuje přehled use-case specifikací nových funkcí, popsané jako jednotlivé kroky, které má uživatel udělat.

Synchronizace se systémovým kalendářem

1. uživatel přejde do záložky *Nastavení*
2. ze seznamu vybere položku *Synchronizace*
3. na obrazovce se objeví tlačítko *Synchronizovat*, datum poslední synchronizace a informace o synchronizaci, co se kam zapíše
4. po stisknutí tlačítka *Synchronizovat* stáhne aplikace aktuální rozvrh a zapíše ho do interního kalendáře
5. aplikace aktualizuje pod tlačítkem *Synchronizace* datum poslední synchronizace

Prohlížení okolí v rozšířené realitě

1. uživatel klikne na jakékoliv ze čtyř hlavních obrazovek (*Přehled*, *Navigace*, *Rozvrh hodin*, *Nastavení*) na pravé horní tlačítko a přejde na novou obrazovku
2. na nové obrazovce se zobrazí živý obraz z fotoaparátu a zároveň začne aplikace získávat uživatelovu polohu (obě dvě akce - přístup k fotoaparátu a získávání polohy, musí uživatel povolit při prvním zapnutí aplikace)
3. na základě polohy se začnou zobrazovat popisky budov v místech kde se reálně nacházejí

Kapitola 2

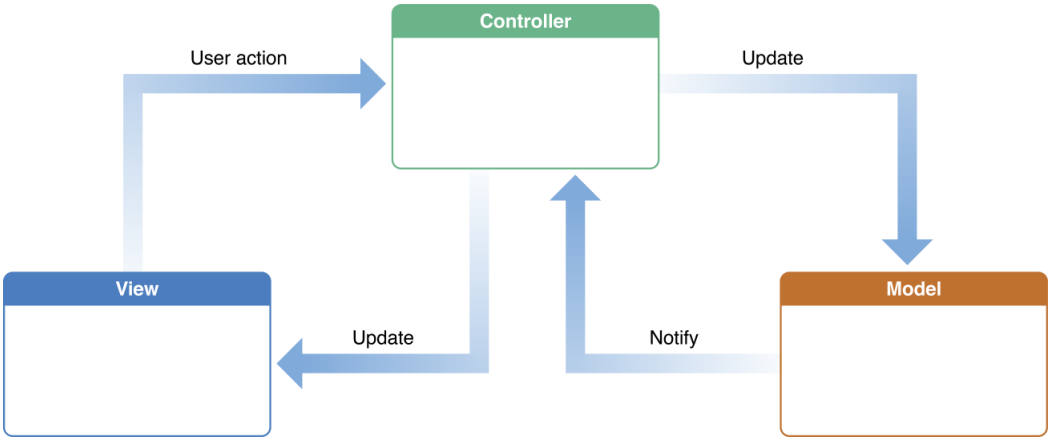
Návrh

V kapitole *Návrh* je popsaná architektura aplikace. Nejprve jsou popsány jednotlivé obrazovky, dále to jsou jednotlivé funkční modely a nakonec zobrazené UI entity.

2.1 Model-View-Controller

Model-View-Controller (zkráceně MVC) je návrhový vzor softwarové architektury oddělující od sebe datový model (*Model*), uživatelské rozhraní (*View*) a řídící jednotku (*Controller*). Komunikace mezi jednotlivými moduly je omezená. Na obrázku 2.1 je znázorněna komunikace mezi jednotlivými moduly. *Controller* je ve středu dění a je spojnicí mezi *View* a *Modelem*. Zamezí nechtěnému přístupu k datovému modelu a nebo připraví data z datového modelu na zobrazení. *View* je uživatelské rozhraní, které přijímá data od *Controlleru* a zobrazuje je uživateli. Dále reaguje na uživatelovu interakci a informuje o ní *Controller*. *Model* jsou všechna data, se kterými aplikace pracuje, které tvoří logiku aplikace. *Controller* může data v *Modelu* modifikovat.

V programování pro iOS se používá slangový výraz *Massive-View-Controller* pro zkratku MVC. Odkazuje na velký počet řádek ve zdrojovém kódu *Controlleru*. Je to způsobeno tím, že *Controller* se v iOS stará např. o zobrazení tabulek nebo jiných kolekcí nebo se stává delegátem pro různé systémové prvky (např. při lokalizaci uživatele pomocí GPS nebo při práci s textovým polem). Velkému počtu řádků kódu se dá předejít lepším návrhem modelu. Není doporučeno všechnu výše zmíněnou logiku přenést do *Modelu*, protože se tím sníží snadné pochopení kódu. Je lepší pouze eliminovat různé podpůrné funkce v Controlleru a ty převést do Modelu. [8]



Obrázek 2.1: Architektura Model-View-Controlleru[8]

2.2 Model

V následující sekci je detailní přehled celé modelové části aplikace. Model se stará o celou logiku a funkčnost aplikace. Součástí modelu je i uchování dat. V aplikacích pro iOS se pro uchování dat používá třída UserDefaults ze standardní knihovny Foundation. UserDefaults[53] se používá pro uložení jednoduchých dat, jako je například nastavení metrického systému nebo uložení příznaku. Pro práci se složitějšími daty, v podobě vlastních tříd, slouží knihovna CoreData[54]. Součástí knihovny je i ukládání/načítání dat do/z SQLite databáze.

2.2.1 Třídy reprezentující reálné objekty

Pro návrh tříd reprezentujících reálné objekty bude použit *objektově orientovaný* přístup. Do objektově orientovaného přístupu se přenášejí prvky z reálného světa. Každý objekt představuje jednu konkrétní entitu, která má vnější rozhraní, kterým komunikuje s okolním světem, a svojí vnitřní implementaci, která zajišťuje funkčnost objektu. Různé objekty spolu mohou být spojené vazbami. Rozlišují se tři druhy vazeb - 1 (entita):1 (entitě), 1:n, n:n, kde n je libovolné přirozené číslo. Pro název tříd se používá anglický výraz popisující danou entitu. Následuje přehled všech tříd reprezentující reálné objekty.

Třída Subject

Třída *Subject* znázorňuje jednu vyučovací hodinu v rozvrhu hodin. Obsahuje data jako jsou název předmětu, typ předmětu (přednáška, cvičení, proseminář, zkouška), datum kdy se předmět vyučuje, od kdy do kdy se vyučuje, kdo ho učí a kde se předmět učí. Třída *Subject* se bude inicializovat pomocí JSON dat, která se přijmou ze serveru. Všechny atributy třídy *Subject* budou veřejné pro všechny ostatní třídy a zároveň budou konstantami, to znamená,

že po inicializaci se nebude moci změnit hodnota atributu.

Třída Place

Třída Place reprezentuje jedno místo, které se zobrazí v rozšířené realitě. Obsahuje název místa, světové souřadnice místa a další pomocné atributy sloužící k ulehčení zobrazení v modelové části aplikace. Třída Place se inicializuje pomocí JSON dat ze souboru places.json.

2.2.2 Třídy zajišťující funkčnost

Následující třídy se starájí o chod aplikace a dodávání správných dat. Některé třídy jsou ve formě *singletonu*, což je návrhový vzor, který zajistí, že v celé aplikaci bude pouze jedna instance dané třídy. Názvy tříd jsou opět v anglickém jazyce.

Třída SettingsService

Třída *SettingsService* se stará o nastavení aplikace. Uchovává v sobě datum poslední synchronizace, jestli je aplikace uzamčená a jaké je barevné nastavení pro zobrazení rozvrhu hodin. Třída *SettingsService* je *singleton*, tudíž nemá veřejnou inicializaci a přistupuje se k ní přes atribut *shared*. Třída bude mít veřejné metody *syncTimetable()* pro synchronizování rozvrhu hodin, *set(color: for:)* pro nastavení konkrétní barvy pro konkrétní typ předmětu, *getColor(for:)* pro získání barvy pro daný typ předmětu, *isLocked()* vracející dvojici obsahující jestli je aplikace uzamčena a jakým biometrickým typem uzamčení (*FaceID*, *TouchID* nebo žádné), a lockApp() pro uzamčení/odemčení aplikace.

Třída TimeService

Třída *TimeService* vrací informace odvozené od aktuálního času. Přes metodu *getNextSubject()* vrací informaci o nejbližší vyučovací hodině, v metodě *getCurrentDay()* vrací informace o aktuálním dnu v podobě dvojice prvků (textová podoba data, index dne v týdnu), v metodě *getPause(between first: second:)* vrací pauzu mezi dvěma předměty a v metodě *getInterval(between first: second:)* vrací text ve formátu *první čas:druhý čas*.

Třída AroundARService

Třída *AroundARService* obsahuje pouze veřejnou metodu pro získání míst pro danou lokaci *getPlaces(for location:)* a metodu *getTransformMatrix(for place:,currentLocation)* pro vytvoření trasformační matice, která se použije v zobrazení trasy v rozšířené realitě. Více informací o fungování třídy *AroundARService* je v kapitole Implementace rozšířené reality.

Třída NavigationService

Třída *NavigationService* je stěžejní pro celou aplikaci, protože se stará o navigování mezi dvěma body. Třída obsahuje veřejné metody *getPath(between startPoi: endPoi:)*, kde *startPoi* je kód naskenovaný z QR kódu a *endPoi* je kód cílové destinace. V třídě je ještě přetížená metoda *getPath(between startLocation: endPoi:)*, kde první argument je uživatelova lokace z GPS. Obě metody vrací pole dvojic obsahující kódy jednotlivých bodů po cestě a jejich lokaci. Dalšími metodami jsou metody *getPoi(for name:)*, která vrací kód

bodu zájmu pro jeho název a *getName(for poi:)*, která vrací název pro daný kód bodu zájmu. Poslední metodou je metoda *getDistance(between point1: point2:)*, která vrací vzdálenost mezi dvěma lokacemi.

Třídy Graph a Node

Třídy *Graph* a *Node* slouží k uložení a propojení bodů zájmů a k nalezení nejkratší cesty. Třída *Node* je využívána pouze ke čtení a obsahuje informace o jednom bodu zájmu - jeho jméno, spojení s ostatními *Node* a světové souřadnice. Třída *Graph* obsahuje jednu veřejnou metodu, a sice *getPath(between start: end:)*, kde vrací pole kódu identifikující jednotlivé body zájmu. Více informací o fungování třídy *Graph* je v kapitole Navigace a nalezení nejkratší cesty.

Třída BiometricService

Třída *BiometricService* slouží k poskytování informací o biometrických čidlech a o vyhodnocení biometrických dat. Obsahuje metody *canEvaluatePolicy()*, která dává informaci, jestli je možné použít čtečku biometrických dat. Dále to jsou metody *getBiometricType()*, která vrací typ čtečky biometrických dat a nakonec to je metoda *authenticateUser()*, která vyhodnotí data ze čtečky biometrických dat.

2.3 Přehled Controllerů

V následující sekci je přehled všech controllerů a co dělají. Jelikož jeden controller reprezentuje jednu obrazovku aplikace, bude místo pojmu *controller* použit pojem obrazovka pro lepší pochopení textu. Pro pojmenování controllerů se používá následující konvence - nejprve je jednoduchý popis controlleru, hlavní činnost kterou dělá, následováno spojením *ViewController*. *UIViewController* je UI element z knihovny *UIKit*.

LoginViewController

Obrazovka obsahující dvě textová pole pro vložení username uživatele a jeho hesla. Username je použit z KOS. Po verifikaci vložených údajů se přejde na hlavní obrazovku. Pokud nemá uživatel přístupové údaje, může alespoň využít funkci navigace.

OverviewViewController

Hlavní obrazovka aplikace. Uživatel se z ní dozví následující předmět, při použití funkce *3D Touch* bude moc spustit navigování do místnosti. Dále si mmůže přečíst novinky ze školy. Tato obrazovka je součástí *TabBarControlleru*, což je UI element z knihovny *UIKit*, který dokáže udržovat více controllerů a přepínat mezi nimi pomocí záložek ve spodní části obrazovky.

NavigationViewController

Obrazovka, která zobrazí formulář pro zadání dat na navigování. Na obrazovce jsou dvě textová pole, která slouží pro zobrazení a vkládání údajů. Údaje se zadávají buď psaním do textového pole (se zobrazením nápovědy), nebo skenováním značek na obrazovce *ScanViewController*. Další možností

je předvyplnění pomocí tlačítka s místností, kde bude vyučován následující předmět.

TimetableViewController

Obrazovka zobrazující rozvrh hodin pro daný den. Mezi dny se dá pohybovat pomocí šipek umístěných v horní části obrazovky. Jednotlivé typy předmětů jsou barevně odlišeny. Po stisknutí jakéhokoliv předmětu se přejde na jeho detail, v rychlé funkci po hlubokém stisknutí se může automaticky spustit navigace.

SettingsViewController

Nastavení aplikace bude znázorněno tabulkou, kde jsou oddělené sekce. V nastavení se dá nastavit zabezpečení (pokud chce mít uživatel vstup do aplikace přes ověření biometrickými údaji z *FaceID/TouchID*), synchronizovat rozvrh s interním kalendářem, a nastavit vlastní barvy v rozvrhu (pro přednášky, cvičení, proseminář).

AroundARViewController

Obrazovka s novout funkcí ve třetí verzi aplikace. Celá obrazovka (mimo horní navigační lišty) je obraz z kamery. Zároveň se získává aktuální uživatelovo poloha. Následně se data ze souboru *places.json* zobrazí v obrazu z kamery a uživatel bude moci vidět v reálném čase jaké budovy jsou v jeho bezprostředním okolí. Na tuto obrazovku je možné dostat ze všech čtyř předchozích obrazovek a zároveň také z rychlé akce z ikony aplikace při použití funkce *3D Touch*.

RouteFromViewController

Obrazovka sloužící k získání počáteční polohy pomocí naskenování QR kódu. Po úspěšném naskenování se zobrazí jeden ze dvou typů hlášek. Buď to je název místnosti nebo budovy, u které se uživatel nachází, nebo to je pouze hláška o úspěšném naskenování, pokud se uživatel nachází u významné části cesty (schodiště, křižovatky).

RouteViewController

Obrazovka s mapou a vyznačenou cestou. V horní části se nachází tlačítko pro vypínání/zapínání polohových služeb a v dolní části se nachází vysouvací lišta s možnostmi nastavení a navigování v rozšířené realitě. V nastavení se dá nastavit zobrazení jednotlivých pater budovy, a dále se zde zobrazuje cílová destinace a tlačítko Skenovat. Po stisknutí tlačítka *Skenovat* se lišta rozšíří na celou obrazovku a zobrazí se obraz z kamery. Po naskenování značky se ukáže cesta k dalšímu bodu v rozšířené realitě.

LockViewController

První z obrazovek nastavení. Na této obrazovce si může uživatel nastavit, že chce, aby byla aplikace uzamčena kódem, a její odemčení je možné pouze přes biometrické údaje z *FaceID/TouchID*.

SyncViewController

Druhá obrazovka nastavení. Na této obrazovce si může uživatel synchronizovat svůj rozvrh s interním kalendářem. Při stisknutí tlačítka *Synchronizovat*

se vloží rozvrh hodin do interního kalendáře a aktualizuje se datum poslední synchronizace.

ColorsViewController

Třetí obrazovka nastavení. Na této obrazovce si může uživatel nastavit barevné odlišení jednotlivých typů předmětů. Po kliknutí na daný typ se objeví menší okno nad aktuální obrazovkou se třemi posuvníky (pro možnosti nastavení RGB) a náhled vybrané barvy.

ClassDetailViewController

Detail jednotlivých předmětů. Celá obrazovka se bude skládat z popisu předmětu, kdo ho učí, o jaký typ se jedná (přednáška, proseminář atd.), od kdy do kdy se vyučuje a v jaké je místnosti. Ve spodní části se nachází tlačítko s možností navigovat na tento předmět. Při stisknutí se zobrazí *Navigation-ViewController* s předvyplněnou destinací.

2.4 Průchod aplikací

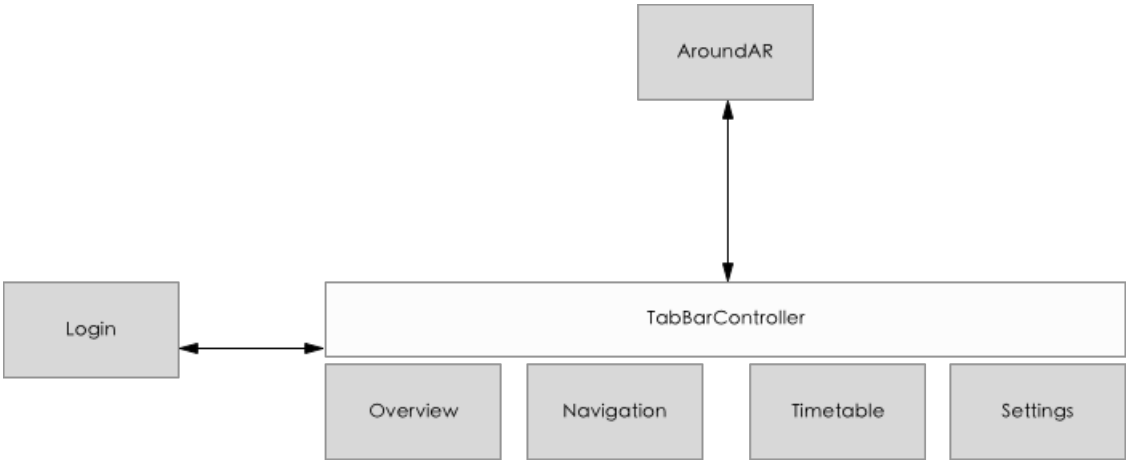
V aplikaci se nachází 12 unikátních obrazovek (1 obrazovka = 1 controller), které jsou různě propojeny mezi sebou. V této sekci je detailně rozepsán průchod celou aplikací. V textu, v závorkách a na obrázcích jsou napsány názvy obrazovek tak, jak jsou pojmenovány v kódu, jen s tím rozdílem, že v kódu je pro větší přehlednost přidáno za název ještě spojení *ViewController*.

2.4.1 Základní obrazovky

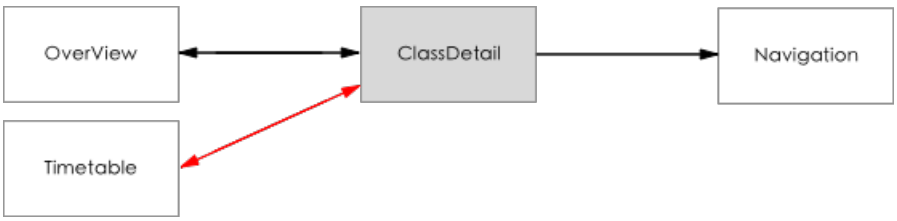
Když uživatel poprvé zapne aplikaci, objeví se na obrazovce přihlášení (*Login*). Po přihlášení se uživatel objeví na hlavním rozcestí aplikace, konkrétně na obrazovce aktuálního přehledu (*Overview*) (ilustrováno na obrázku 2.2). Tato obrazovka, společně s dalšími třemi obrazovkami (*Navigation, Timetable a Settings*), je spojena dohromady entitou z knihovny *UIKit*, *UITabBarController*[55] (na obrázku jako *TabBarController*). *TabBarController* v sobě drží předem daný počet controllerů a uživatel mezi nimi může přepínat pomocí tlačítek v dolní liště. Z každé ze čtyř základních obrazovek se dá dostat na prohlížení okolí pomocí rozšířené reality - *AroundAR*. Dále se dá z těchto obrazovek odhlásit.

2.4.2 Možnosti průchodů z obrazovky Overview a Timetable

Průchody z obrazovek *Overview* a *Timetable* jsou totožné (průchody jsou ilustrovány na obrázku 2.3). Ze základní obrazovky *Overview* se dá dostat, po kliknutí na následující předmět, na obrazovku *ClassDetail*, kde budou zobrazeny další informace o daném předmětu. Z obrazovky *Timetable* se na stejnou obrazovku dá dostat po kliknutí na jakýkoliv předmět, který je zobrazen v rozvrhu hodin. Z obrazovky *ClassDetail* se dá dostat na obrazovku *Navigation* (na obrázku 2.3 znázorněna bílou barvou).



Obrázek 2.2: Grafické znázornění základních obrazovek aplikace

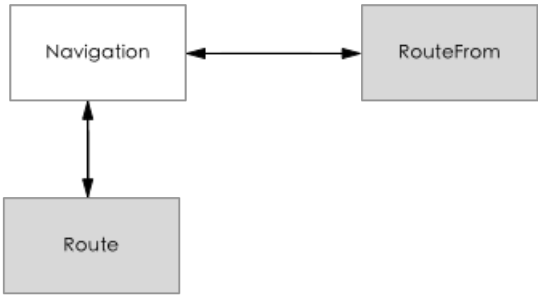


Obrázek 2.3: Grafické znázornění možností průchodů z obrazovky *Overview* a *Timetable*

2.4.3 Možnosti průchodů z obrazovky Navigation

Z obrazovky *Navigation* se uživatel může dostat na několik různých obrazovek. Tato obrazovka slouží k zadání výchozího bodu uživatelovo cesty, cílového bodu a spuštění samotné navigace. Průchod je popsán ve zmíněném pořadí. Nejprve si uživatel zvolí svůj výchozí bod pro navigaci. Buď se nachází mimo budovu, a automaticky se načte jeho poloha, nebo se nachází v budově a musí zvolit nejbližší učebnu. Začne tedy psát text do textového pole a podle napsaného textu se zobrazí nápověda, nebo klikne na šipku v textovém poli *Odkud*. Dostane se na obrazovku *RouteFrom*. Na této obrazovce je možnost naskenovat nejbližší učebnu. Po úspěšném naskenování se zobrazí hláška, že je kód naskenován. Z obrazovky *RouteFrom* se poté vrátí na obrazovku *Navigation*. Všechny zmíněné průchody jsou ilustrovány na obrázku 2.4.

Pokud se uživatel dostane na obrazovku *Navigation* z obrazovky *ClassDetail* (viz. podsekce Možnosti průchodů z obrazovky Overview a Timetable), tak textové pole *Odkud* již bude předvyplněné. Jinak se v průběhu vyplňování textového pole opět zobrazuje aktuální nápověda. Pod textovým polem se nachází tlačítko s místností, kde se vyučuje následující předmět z uživate-

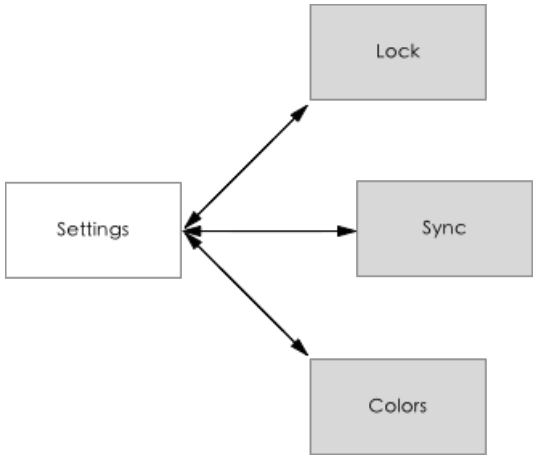


Obrázek 2.4: Grafické znázornění možností průchodů z obrazovky *Navigation*

lovo rozvrhu. Po stisknutí tlačítka *Navigovat* se uživatel dostane na obrazovku *Route*. Na ní probíhá samotné navigování a zobrazení cesty v rozšířené realitě.

2.4.4 Možnosti průchodů z obrazovky Settings

Z obrazovky *Settings* se dá dostat na tři další obrazovky, kde si může uživatel nastavit aplikaci nebo synchronizovat data. První obrazovkou je obrazovka *Lock*, kde si uživatel může nastavit uzamčení aplikace svým systémovým heslem a zadávat ho pomocí svých biometrických údajů (otisk prstu nebo obličej). Další obrazovkou je *Sync*, kde si může uživatel synchronizovat svůj rozvrh hodin s kalendářem v telefonu. Poslední obrazovkou je obrazovka *Colors*, kde si může uživatel nastavit barevné odlišení jednotlivých typů výuky v rozvrhu hodin. Průchody jsou ilustrovány na obrázku 2.5.



Obrázek 2.5: Grafické znázornění možností průchodů z obrazovky *Settings*

2.5 Přehled View

V následující sekci je přehled významných *View* z *Model-View-Controller*. Všechny vypsané View se buď vyskytují v aplikaci často (jsou znovupoužitelné) nebo jsou komplexní a slouží k danému úkolu.

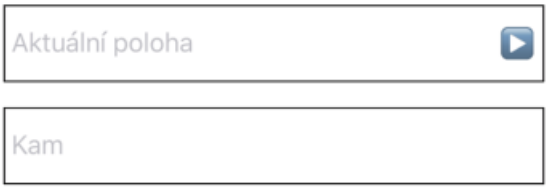
Tlačítka

Aplikace obsahuje tři druhy tlačítek - tlačítko akce, tlačítko doplnění a tlačítko uzavření. Tlačítko akce (Obrázek 2.6) je potvrzovací tlačítko spouštějící akci jako je přechod na další obrazovku. Je odlišené jasnou modrou barvou, aby uživatele zaujal na první pohled a dal mu indície, že je možné něco spustit. Tlačítko doplnění je použito pro vyplnění textového pole předdefinovanou hodnotou. Od tlačítka akce je odlišeno průhlednou barvou a doprovodným textem. Tlačítko uzavření je použito v *ColorView*. Speciální kategorií jsou tlačítka pro přepínaní mezi obrazovkami ve spodní a horní části obrazovky. U nich se předpokládá, že uživatelé ví co dělají (přepínají obrazovky), protože to jsou základní prvky pro tvorbu UI doporučené firmou Apple.

Obrázek 2.6: Tlačítko akce

Textová pole

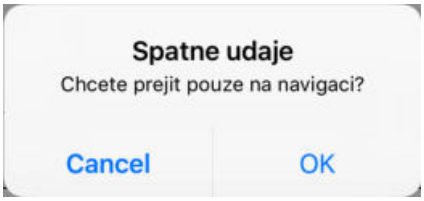
Textová pole slouží pro vyplnění krátkého textu. V aplikace se vyskytují dva druhy textového pole (Obrázek 2.7), jedno je klasické textové pole pro vyplnění textu, druhé navíc obsahuje v pravé části tlačítko naznačující možnost předvyplnění. Pokud jsou pole logicky poskládaná za sebou, je možné mezi nimi přepínat klávesou enter. U posledního pole se pomocí klávesy enter spustí očekávaná akce.



Obrázek 2.7: Textová pole

Upozornění

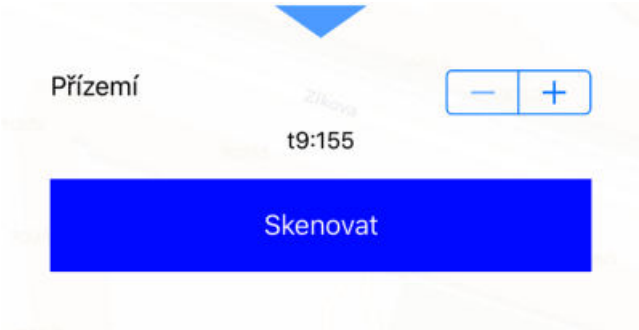
Pro zobrazení upozornění (obrázek 2.8) se používá nativní kompomenta z knihovny *UIKit UIAlertController*[56]. Upozornění může obsahovat název, doprovodný text a tlačítka pro potvrzení a ukončení, popřípadě pouze jedno tlačítko. Samotné upozornění může obsahovat i textové pole, ale to v této aplikaci není použito.



Obrázek 2.8: Upozornění

MapOptionsView

MapOptionsView (Obrázek 2.9) je nejkomplexnější view z celé aplikace. Nachází se na obrazovce *RouteViewController* a slouží k akcím na mapě. Obsahuje UI element *UIStepper*[57] k přepínání zobrazení mapových podkladů pro jednotlivá patra. Dále obsahuje název cílové destinace a tlačítko na naskenovaní značky. Po stisknutí se celé view rozšíří, zmizí ukazatel pater a zbytek prvků se posune do horní části. Ve zbylé části se objeví obraz z kamery. Celé je to doprovázeno jednoduchou animací.



Obrázek 2.9: MapOptionsView

Buňky v seznamech

Spousta obrazovek v aplikaci obsahuje seznamy a jejich buňky. Hlavní seznam v aplikaci je na obrazovce *TimetableViewController* a zobrazuje buňky *SubjectCell* (Obrázek 2.10) se základními informacemi o předmětu jako je název, místnost a čas. Na první obrazovce OverviewViewController je seznam zobrazující zprávy pomocí buňky *NewsCell*. *NewsCell* obsahuje *UITextView*[58], což je UI element z knihovny *UIKit* pro zobrazení delšího textu.

ColorView

ColorView (Obrázek 2.11) slouží k nastavení barev pro jednotlivé typy předmětů. Obsahuje tři posuvníky (*UISlider*[59]) pro nastavení jednotlivých složek RGB (Red, Green, Blue). Dále obsahuje náhled vybrané barvy a tlačítko pro uzavření.

Obrázek 2.10: Buňky SubjectCell

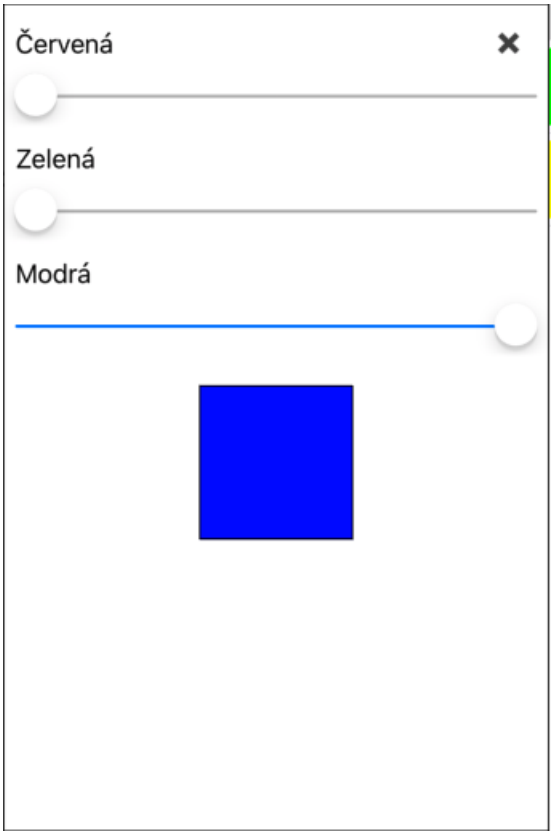
2.6 Delegáti

2.6.1 Komunikace mezi controllery

Komunikace mezi dvěma controllery je rozdělena na dva způsoby. Jeden je na "následující" controller, kdy aktuální controller A vytvoří instanci controlleru, na který chce uživatel přejít, controlleru B. Controller A ví o controlleru B, vše co má controller B veřejné. Při komunikaci směrem "zpět" (uživatel se chce vrátit na controller A) nastává problém. Aktuáně zobrazený controller B neví nic o svém předchůdci, controlleru A. Tento problém se dá vyřešit za použití delegace. Než controller A zmizí z obrazovky a objeví se aktuání controller B, nastaví se controller A jako delegát pro controller B. To, že je nějaký controller delegát (v obecném případě může být delegátem jakákoliv třída nebo struct), znamená, že musí implementovat konkrétní protokol (předpis povinných funkcí). Controller B v sobě má referenci na delegáta. To ale neznamená, že ví vše o controlleru A, ví pouze, že skrze zmíněnou referenci může zavolat funkce z protokolu delegáta. Komunikace je ilustrována na obrázku 2.12.

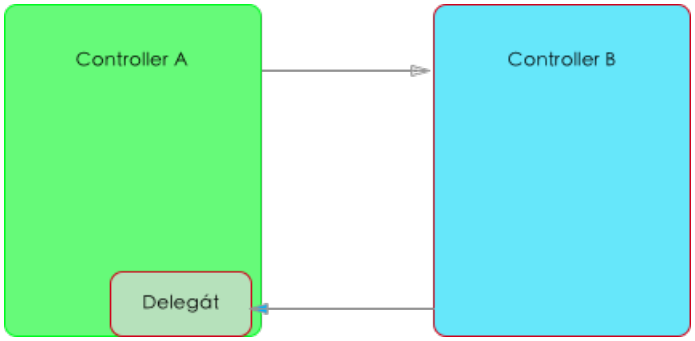
2.6.2 Další použití

Delegáti slouží ke komunikaci mezi třídami a strukturami. Entita, která se ustanoví delegátem musí implementovat předepsané funkce z delegátího protokolu. Druhá entita, která chce komunikovat se svým delegátem, může volat pouze funkce z delegátího protokolu. Jedno využití delegátů je popsáno v části



Obrázek 2.11: ColorView

Komunikace mezi controllery. Ve frameworku UIKit se pomocí delegátů komunikuje mezi některými prvky, jako je například *UITableView*, popřípadě *UICollectionView*, dále například pole pro vkládání textu - *UITextField*. V prvním případě, v *UITableView*, slouží delegát k nastavení zobrazovaného seznamu. Dostupné funkce nastaví počet buněk, počet sekcí, nebo obsah jednotlivých buněk. U textového pole *UITextField* se zavolají metody po každém vloženém písmenu, nebo při schování klávesnice.



Obrázek 2.12: Komunikace mezi obrazovkami

Kapitola 3

Implementace

V následující kapitole bude popsáno vývojové prostředí, styl psaní kódu nebo implementace zajímavých částí aplikace, ať už z hlediska implementace (zobrazení názvů budov a navigace v rozšířené realitě) nebo z algoritmického hlediska (hledání nejkratší cesty). O tom jak se dá samotná aplikace nainstalovat a používat pojednává část Instalační a uživatelská příručka.

3.1 Vývojové prostředí

Pro vývoj aplikace bylo použito vývojové prostředí *XCode*[60] od firmy Apple ve verzi 9.3. Vývojové prostředí *XCode* je dostupné pouze pro operační systém macOS a je to jediné vývojové prostředí vyvíjené a podporované firmou Apple, ze kterého se poté dá aplikace posílat do obchodu s aplikacemi pro operační systém iOS, nazvaného App Store (ilustrační obrázek 3.1). Pro testovací účely se dá také aplikace poslat do programu *TestFlight*[61], sloužící k testování pro předem určené uživatele.

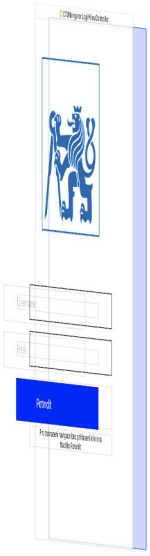
```
8
9  import UIKit
10 import SnapKit
11
12 /// Users can login via their CTU Username and password
13 class LoginViewController: BaseViewController {
14 //  UI
15     weak var logoImageView: UIImageView!
16     weak var usernameTextField: SimpleTextField!
```

Obrázek 3.1: Vývojové prostředí XCode - editor

Vývojové prostředí *XCode* plně podporuje programovací jazyk Swift, včetně následného ladění. Pro nacházení chyb a ladění kódu se používá program nazvaný *LLDB*[62]. Program *LLDB* se může používat buď z konzole, která je součástí vývojového prostředí, nebo pomocí vizuálních značek umístěných v

kódu. Pomocí nástroje nazvaného *View Debugger* (obrázek 3.2) se dá ladit vizuální stránka aplikace - rozložení jednotlivých prvků. Z aplikace se sejme aktuální obrazovka a zobrazí se ve 3D pohledu, tak jak jsou na sobě různé elementy poskládány ve vrstvách. Po výběru elementu se zobrazí jeho vlastnosti.

Aplikace se dají spustit buď na registrovaném iPhonu a přes kabel nebo přes síť wifi se dají posílat informace do vývojového prostředí *XCode*, nebo v programu *iOS Simulator* (zkráceně simulátor), který je součástí vývojového prostředí XCode. Simulátor obsahuje všechny dostupné verze iPhonů, které podporují daný operační systém iOS. Možnosti simulátoru jsou omezené, ale pro základní přehled, že rozložení prvků se zobrazuje správně, je dostačující. Simulátor například neumí zobrazit obraz z kamery počítače pro rozšířenou realitu, nebo nedokáže určit aktuální lokaci. Pro určení lokace se dají použít vlastní světové souřadnice nebo souřadnice už předem definované, jako je hlavní sídlo firmy Apple. Pro testování pohybu se dají opět použít předem nadefinované cesty, nebo je možné si vytvořit vlastní.



Obrázek 3.2: View Debugger - 3D pohled z boku

3.2 Styl psaní kódu a rozložení projektu

Aplikace ve vývojovém prostředí *XCode* se programují v tzv. *projektu*. Pro daný projekt se dají tvořit *targety* (aplikace a její rozšíření v podobě aplikace pro hodinky atd.) nebo spravovat možnosti aplikace (přidávat jazykové mutace nebo operační systémy, na kterých bude aplikace fungovat, nastavovat název a ikony). V projektu je také adresářová struktura pro zdrojové kódy. Pro svojí práci jsem zvolil následující adresářovou strukturu zobrazenou na obrázku 3.2.

- `Supporting Files` ..... konfigurační soubory aplikace
  - `Resources` ..... JSON soubory pro navigování
  - `Utilities` ..... Rozšíření pro existující třídy z knihoven Foundation a UIKit
- `Model` ..... Modelová část z MVC
  - `Classes` ..... Třídy reprezentující reálné objekty a podpůrné třídy
  - `Services` ..... Třídy zajišťující funkčnost aplikace
  - `CoreData` ..... Třídy pro práci s knihovnou CoreData
- `View` ..... Vzhled aplikace
  - `UI Elements` ..... Jednotlivé UI prvky
  - `Cells` ..... Jednotlivé buňky pro seznamy
  - `Map` ..... Vzhled mapy
  - `Other` ..... Ostatní UI
- `Controller` ..... Jednotlivé obrazovky
- `Assets.xcassets` ..... Knihovna obrázků

Obrázek 3.3: Adresářová struktura projektu

Kod psaný v programovacím jazyce Swift, konkrétně implementace třídy, není rozdělený do hlavičkového souboru a souboru pro implementaci, jako je kód napsaný v jazyce Objective-C, ale je psán do jednoho souboru. Dokonce neexistuje ani nic jako rozhraní. Přehlednost je ponechána na programátorovi. Já jsem se rozhodl využít možnosti rozšíření (extension) tříd pro lepší přehlednost. Rozšíření lze použít pouze pro metody tříd, nikoliv pro atributy. Atributy musí být vypsány v hlavní části kódu. Rozšíření mohou být veřejná (jiná třída může využít její metody) nebo privátní (pouze rozšiřovaná třída může využít vypsané metody). Celá třída má následující rozložení:

1. hlavní deklarace třídy s atributy a veřejnými metodami (rozhraní třídy)
2. metody z adaptovaných protokolů (pro každý protokol jedno rozšíření)
3. partrivátní metody třídy (může být rozděleno do více logických celků)
4. metody pro AutoLayout

Pro vytváření controllerů byla vytvořena třída *BaseViewController*, ze které dědí všechny ostatní controllery. Třída *BaseViewController* se stará o zobrazení klávesnice. Při zobrazení/schování klávesnice pošle systém lokální notifikaci (upozornění) aplikaci. Třída *BaseViewController* má zaregistrováno v notifikačním centru (třída *NotificationCenter*[63]), aby tyto dvě notifikace přijímala a následně provedla danou akci. Tyto dvě akce jsou dvě abstraktní metody *keyboardHiding(notification: Notification) a keyboardAppearing(notification: Notification)* a každá třída, která dědí z *BaseViewController* si je implementuje sama. Dále je v třídě *BaseViewController* vytvořeno

snímání gesta pohybu prstu od shora dolů, které schová klávesnici (aplikace upozorní systém, že chce schovat klávesnici). Nakonec se třída *BaseViewController* stará o horní tlačítka, levé tlačítko odhlásí uživatele z aplikace a vrátí se na první obrazovku s přihlášením a pravé tlačítko zobrazí popisky budov v rozšířené realitě.

3.3 Implementace rozšířené reality

Základním problémem v zobrazení cesty nebo popisků budov v rozšířené realitě je převod zeměpisných souřadnic budov nebo míst v budově do souřadnic prostoru snímaného kamerou. Zeměpisná souřadnice se skládá ze zeměpisné délky (úhlová vzdálenost od nultého poledníku) a šířky (úhlová vzdálenost od rovníku) a udává jednoznačně polohu místa na Zemi. Pro zobrazení místa v rozšířené realitě je, stejně jako v 3D grafice, potřeba matice typu *4x4* [64]. První tři prvky prvních tří sloupců udávají rotaci souřadnicového systému a čtvrtý sloupec matice udává posun od počátku. Prvky na diagonále značí poměr velikosti implicitně nastavené na 1. Výsledná matice se získá maticovým násobením s maticemi rotace a posunutí. Je důležité je násobit v daném pořadí, jinak bude pro každou kombinaci rozdílný výsledek. V aplikaci se matice násobí v následujícím pořadí:

$$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

kde první matice je matice rotace podle osy Y, a druhá matice je matice posunutí. Úhel θ se získá jako úhel mezi uživatelovou aktuální lokací a cílovou lokací pomocí *Haversinovi formule* (rovnice 3.1). [65]

$$\begin{aligned} &c_1 = uživatelova\_lokace \\ &c_2 = cílová\_lokace \\ &\theta = (X, Y) \\ &X = \cos(c_2.lat) * \sin(c_2.lng - c_1.lng) \\ &Y = \cos(c_1.lat) * \sin(c_2.lat) - \sin(c_1.lat) * \cos(c_2.lat) * \cos(c_2.lng - c_1.lng) \end{aligned} \tag{3.1}$$

Vzdálenost mezi dvěma lokacemi se určí pomocí funkce *distance(from:)* z interní knihovny *CoreLocation*. Pro matici posunutí potřebujeme pouze posunutí od uživatele, tedy na negativní části osy Z.

Pomocí interní knihovny *ARKit* se oba body zobrazí pomocí třídy *ARAnchor*, které se v inicializaci předá výše zmíněná transformační matice. Při každém překreslení si už knihovna sama přepočítává transformační matici. Knihovna *ARKit* je nová a není ještě vyladěná, objekty tedy mohou v prů-

běhu času různě kmitat po prostoru. Očekává se, že vylepšená verze knihovny bude představen v červnu 2018 na konferenci WWDC firmy Apple.

Pro zobrazení bodů a cesty mezi nimi je použita knihova *SceneKit*. Knihovna *SceneKit* zobrazuje 3D objekty. Základním prvkem knihovny je třída *SCNNode*[66], která představuje bod v prostoru. Hlavnímu bodu scény se říká *rootNode* a je atributem scény *SCNScene*[67]. Každý další bod je přímým či nepřímým potomkem bodu *rootNode*. Třída *SCNNode* sama o sobě žádný vizuál nemá. O vizuální stránku bodu *SCNNode* se starají jednotlivé geometrie (třída *SCNGeometry*[68]). V knihovně *SceneKit* je několik základních geometrií připravených a v aplikaci ČVUT Navigátor se využívají koule (*SCNSphere*[68]) pro zobrazení počáteční a cílové destinace, a válec (*SCNCylinder*[68]) pro zobrazení cesty mezi body. Každé geometrii se může nastavit její vizuál pomocí různých materiálů nebo textur.

3.4 Navigace a nalezení nejkratší cesty

Aplikace umožňuje navigovat mezi dvěma zadanými body. Seznam bodů a cest mezi nimi je v souboru *waypoints.json*, resp. v *routes.json*. Struktura jednoho bodu má následující tvar:

```
"168": {
                "lat": "50.10385",
                "lon": "14.38974"
}
```

Klíčem je jednoznačný identifikátor daného místo používaný i v dalších souborech. Hodnotami jsou světové souřadnice. Struktura jedné cesty mezi dvěma body je následující:

```
{
                "a": "173",
                "b": "172"
}
```

Cesta je pokaždé obousměrná a je mezi dvěma identifikátory, které jsou specifikované v předchozí struktuře. V souboru *pois.json* je seznam bodů, které se zobrazí v nabídce možných cílových destinací. Struktura jednoho zobrazitelného bodu v souboru je následující:

```
"ntk": {
                "name": "NTK",
                "keywords": [],
                "waypoint": "171",
                "type": "building"
}
```

Nutno dodat, že všechny tři soubory jsou prozatím testovací a cesta mezi nimi není přesně optimalizovaná, tudíž se může stávat, že cesta vede skrz zdi (jako například mezi přednáškovými místnostmi t9:105 a t9:155).

Po načtení ze souboru jsou body reprezentovány třídou *Node*, která obsahuje název bodu, reference na přímé sousedy, a jeho rodiče, který je určen až dodatečně při hledání nejkratší cesty. Vzdálenosti mezi body jsou vypočítávány pomocí již zmíněné funkce *distance(from:)* z interní knihovny *CoreLocation*. Třída *Node* ještě obsahuje atribut ukládající celkovou délku cesty. Všechny vytvořené třídy *Node* jsou uloženy v třídě *Graph*, v datové struktuře *Set* (datová struktura neseřazených a jedinečných prvků). Pro nalezení nejkratší cesty je ve třídě *Graph* implementován *Dijkstrův algoritmus*. Dijkstrův algoritmus byl vymyšlen v roce 1959 Edsgerem Dijkstrou. Nejprve se ve všech bodech nastaví rodič na nedefinovanou hodnotu nil a celková délka cesty na 0. Poté se začne procházet každý prvek od počátečního prvku, dokud se neprojdou všechny prvky. Při nalezení nejkratší cesty mezi dvěma body, se nastaví počáteční bod jako rodič následujícího bodu a u následujícího bodu se spočítá aktuální délka cesty. [69]

Navigace využívá ke zjištění polohy buď data z GPS (popřípadě kombinace GPS + Galileo + GLONASS u novějších iPhonů) nebo se zjistí poloha pomocí naskenování QR Kodu. Když se naskenuje poloha pomocí QR kódu (i v případě zadání počáteční destinace), tak se vypne zjišťování polohy pomocí dat z GPS. Každý QR kód obsahuje jako hodnotu jednoznačný identifikátor místa. Po naskenování bodu a zjištění polohy se přepočítá cesta a následne se zobrazí na mapě.

Pro zobrazení mapy se používá knihovna *Carto Mobile SDK*, která nabízí alternativu k interní knihovně MapKit, a dokáže vykreslit vektorové mapy z vlastního zdroje dat. V aplikace je celý mapový podklad pro hlavní město Praha a pro přízemí Fakulty informačních technologií. Funkce zobrazení vektorových map z vlastních zdrojů je stále ve vývoji a nezvládá správně zobrazit data ze souborů typu *GeoJSON*. V aplikaci se to projevuje špatně zobrazeným přízemím Fakulty informačních technologií. Knihovna *Carto Mobile SDK* je v dnešní době jedinou alternativou, která nabízí možnost zobrazit vektorové mapy z vlastních zdrojů zdarma.

Kapitola 4

Testování

V poslední kapitole budou popsány jednotlivé testy, které byly na aplikaci ČVUT Navigátor provedeny. Jedná se o uživatelské testování, což je testování prováděné na vzorové skupině uživatelů aplikace a dává vývojáři zpětnou vazbu co se týče uživatelského rozhraní, snadného pochopení ovládání a navíc může odhalit funkční chyby v aplikaci, kterých si vývojář nevšiml. Druhou kategorií testů, které byly provedeny na aplikaci, jsou jednotkové testy. Jednotkové testy testují, už podle názvu, jednu konkrétní jednotku aplikace. Jednotkou může být funkce, třída zajišťující funkčnost, či algoritmus nebo proměnné. Jednotkové testy slouží k ověření funkčnosti dané jednotky i po jejím rozšíření.

4.1 Uživatelské testování

Na začátku vývoje aplikace byl vytvořen klikatelný prototyp aplikace, ve kterém byl návrh rozložení jednotlivých UI elementů. Prototyp byl vytvořen pomocí aplikace pro operační systémy macOS a iOS Keynote, a díky podpoře operačního systému iOS mohl být prototyp otestován na iPhone X. Výsledkem testování, které bylo provedeno na třech lidech, bylo konstatování, že prototyp je z pohledu UI dobře navrženo, a uživatelé nemají problém dostat se v aplikaci kam potřebují. Základní úkoly pro test byly přihlásit se a zadat cestu z bodu A do bodu B, zobrazit rozvrh, detail jedné vyučovací hodiny a z detailu spustit navigaci, a nakonec nastavit uzamčení aplikace pomocí biometrické čtečky. Jednou z testovaných osob byl programátor iOS aplikací. Byl zvolen za účelem získání názoru někoho, kdo se pohybuje v oboru a zná tvorbu UI pro aplikace na operační systém iOS. Další testovací osobou byl student z Fakulty informačních technologií, který se nevěnuje vývoji mobilních aplikací, a poslední osoba neměla s programováním nic společného, pouze využívala mobilní zařízení s operačním systémem iOS.

Po vytvoření hlavních funkcí aplikace proběhlo další testování, nyní už zaměřeno na hlavní funkci aplikace - navigování. Pro testování byly vybrány čtyři

osoby - studentka Vysoké školy ekonomické, která používá iPhone hlavně na komunikaci, student Univerzity Karlovy, který pracuje s operačním systémem iOS od jeho počátku, a dva studenti Fakulty informačních technologií, kteří se zabývají vývojem aplikací pro operační systém iOS. Podstatou testování bylo zjistit, jak se dokáží testované osoby orientovat po budově za pomoci aplikace a zda-li trefí do cíle. Úkolem pro testované osoby bylo přihlásit se do aplikace (pokud byly studenty ČVUT), popřípadě spustit aplikaci bez přihlášení (pro studenty mimo ČVUT), a dostat se od vstupu budovy Fakulty informačních technologií do přednáškové místnosti t9:155. Na hlavních bodech cesty byly vyvěšeny QR kódy pro zjištění polohy.

Ukázalo se, že hlavním úskalím jsou data pro navigování. Data o bodech v budově jsou pouze testovací a nejsou optimalizovaná. Dále rozložení QR kódů nesouhlasilo přesně s body zobrazenými na mapě. Ve výsledku se uživatelé dostali k cíli po dlouhé cestě, kdy se ztráceli. V té době se aplikace chovala tak, že po načtení bodu se automaticky zobrazila aktualizovaná mapa bez správného natočení. Testovací osoby byly dezoritentované a potřebovaly čas na to, se zpátky zorientovat. Díky výsledkům testů se navigování předělalo do aktuální podoby, kdy se cesta mezi dvěma body zobrazí pomocí rozšířené reality a ne pomocí šipky udávající směr. Dále se změnilo načítání bodů pomocí QR kódu. Nyní se po načtení bodu zobrazí daný úsek cesty v rozšířené reality a případně se zobrazí popisek, pokud se uživatel nachází u místnosti. Aplikace se sama od sebe nepřepne zpět na mapu.

Testované osoby ale měly poznámky i k uživatelskému rozhraní, konkrétně k zobrazení mapy a možnosti práce s mapou. Tlačítko pro vypínání/zapínání polohových služeb bylo špatně umístěné a nebylo patrné k čemu slouží. Mapa nebyla přes celou obrazovku a špatně se ovládala klasickými gesty (posunutí prstu = posunutí mapy, roztažení dvou prstů od sebe = přiblížení mapy). Aktuální UI obrazovky s mapou bylo inspirováno interní aplikací Mapy. Celá mapa se roztáhla na celou obrazovku, do pravého horního rohu se přesunulo tlačítko k vypínání/zapínání polových služeb a do spodní části obrazovky se přidalo tlačítko, které zobrazí lištu s možnostmi. Samotná lišta do poloviny obrazovky překryje mapu. V liště je možnost přepínat zobrazení jednotlivých pater budovy, dále je zde informace o cílové destinaci a tlačítko na zobrazení obrazu z kamery na naskenování QR kódu. Po stisknutí tlačítka se skryjí nedůležité prkvy (pro zobrazení jednotlivých pater), tlačítko a text o destinaci se přesune výše a zobrazí se obraz z kamery.

4.2 Jednotkové testy

Díky zvolené architektuře *Model-View-Controller* je možné jednoduše otestovat kompletně celý model aplikace. V aplikaci se nachází celkem šest různých tříd reprezentující model v MVC - hledání nejkratší cesty v třída *Graph*, třídy *TimeService, NavigationService, SettingsService, BiometricService a Aroun-*

dARService. Z těchto šesti modelů se nedá otestovat třída *BiometricService*, protože potřebuje informace o čteckách biometrických údajů, které se nachází na zařízeních s operačním systémem iOS.

Jednotkové testy by měly být plně automatizované, vstupem pro test jsou vždy testovací data a výstupem ověření výsledků (obsah výsledné proměnné). V části Styl psaní kódu a rozložení projektu bylo popsáno, že aplikace může mít různé targety. Jedním typem targetu je i target pro jednotkové testy. Testy se provádí v třídách, které dědí z hlavní třídy *XCTest*[70]. Jednotlivé třídy implementují dvě abstraktní metody z rodičovské třídy - *setUp()* a *tearDown()*. Třída *setUp()* je volaná před každým jednotlivým testem a slouží k nastavení počátečních hodnot. Naopak třída *tearDown()* je volaná po každém testu a je dobré v ní uvolňovat paměť, aby byla jisté, že každý test začne s novou alokací.

Ověřování hodnot probíhá pomocí testovacích ověřování (*assertion*). Testovacím ověřením jsou makra v třídě *XCTest* a je jich celkem pět. *Boolean Assertions* testují, jestli výsledek vrací hodnoty pravda (*true*)/nepravda (*false*), *Nil and Non-nil assertions* kontrolují, jestli výsledek má alokovanou paměť či nikoliv, *Equality/Inequality assertions* kontrolují rovnost mezi dvěma hodnotami, *Comparable Value assertions* porovnává dvě hodnoty (větší/menší) a konečně *NSException assertions* kontroluje jestli test vyhodil vyjímku. Seznam a více informací na webové stránce dokumentace [70]. Pro jednotlivé testy se připraví jednoduchá testovací data, která by měla reprezentovat celou škálu vstupních dat. Testovat by se mělo i to, jak si testovaný subjekt poradí s nečekanými situacemi. Testovanými subjekty jsou všechny veřejné metody z dané třídy.

Závěr

Cílem práce bylo vytvořit třetí generaci aplikace ČVUT Navigátor podporující vektorový formát map. Aplikace splňuje všechny zadané úkoly a je připravena jak k napojení na serverovou část (která není funkční) tak na podporu připravovaného operačního systému iOS 12.

Analyzujte možnost navigačního systému Galileo pro lokalizaci uživatele

Navigační systém Galileo nepřináší a ani nepřinese do budoucna možnost přesné polohy uvnitř budovy, ale v kombinaci s navigačními systémy GPS a GLONASS určuje už teď, před oficiálním dokončením, přesnou polohu ve problémových lokalitách, jako jsou ulice velkoměst. Více informací je v kapitole Kombinace polohových systémů.

Analyzujte předchozí verzi ČVUT Navigátoru a navrhněte změny v implementaci, aby byla podporována zařízení s iOS 10+

V úvodu práce bylo řečeno, že z důvodu použití rozšířené reality bude možná podpora pouze operačního systému iOS 11. Aplikace využívá již zmíněnou rozšířenou realitu (za pomoci knihovny *ARKit*), čtečku biometrických dat k přihlášení, nástroj AutoLayout pro pozicování UI prvků a funkci *3D Touch* pro nový rozměr ovládání.

Implementujte určení polohy pomocí systému Galileo a pomocí rozpoznávání značek

Operační systém iOS podporuje u určitých typů iPhonů určení polohy pomocí kombinace systémů GPS+Galileo+GLONASS. V neuzavřených prostorách se poloha určuje právě pomocí této trojice (nebo pouze pomocí GPS u ostaních přistrojů). V uzavřených prostorách se poloha určuje pomocí skenování QR kódů, které obsahují jednoznačnou identifikaci místa.

Implementujte funkcionality podporující vektorové indoor a outdoor mapy, offline režim, nalezení nejkratší cesty a rozšířenou realitu
Kompletní aplikace je připravená na přiloženém CD nebo na platformě *TestFlight* k vyzkoušení. Knihovna podporující zobrazení vektorových map z vlastních zdrojů potřebuje ještě vyladit některé funkce, v současnosti nedokáže správně zobrazit mapy budov. Navigace dokáže najít nejkratší cestu a uvnitř budov jí dokáže zobrazit pomocí rozšířené reality.

Klienta podrobte vhodným testům Po prvním otestování klikatelného prototypu bylo rozhodnuto, že aplikace je jednoduchá na ovládaní a je možné se dostat jednoduše a intuitivně ke všem možnostem. Po uživatelském otestování hotové aplikace se musela předělat celá obrazovka sloužící k navigování, protože byla pro uživatele nepochopitelná. Aplikace dále obsahuje jednotkové testy kontrolující, zda-li je vše funguje správně.

Další vylepšení

Aplikace není zdaleka kompletní a potřebuje další vylepšení. Vedle napojení na serverovou část je potřeba vytvořit grafický návrh aplikace grafikem, tak aby vypadala moderně ale funkčně. Další možnosti rozšíří funkčnosti aplikace. Vhodným rozšířením jsou upozornění uživatele, že začne další předmět a cesta bude trvat určitý čas. Dalším rozšířením je widget zobrazující další předmět na obrazovce widgetů.

Aplikace je lokalizovaná do českého a anglického jazyka a je možné vytvořit další lokalizační soubory pro další jazyky ve formátu *klíč:hodnota klíče;*. Idelálním řešením by bylo zanalyzovat počty zahraničních studentů mířících na ČVUT a podle toho připravit lokalizační soubory.

V červnu 2018 bude představena další verze operačního systému iOS s pořadovým číslem 12, a bude velmi zajímavé sledovat, jaké novinky budou představeny. Aplikace je momentálně připravena na podporu a rozšíření na další roky dopředu.

Literatura

[1] Apple WWDC 2010 Keynote. [cit. 1. 5. 2018]. Dostupné z: https://itunes.apple.com/us/podcast/apple-keynotes/id275834665?mt=2

[2] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/AnatomyofaConstraint.html#//apple_ref/doc/uid/TP40010853-CH9-SW1

[3] [cit. 10. 5. 2018]. Dostupné z: https://www.macrumors.com/2017/08/29/big-companies-show-off-arkit-apps/

[4] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/arkit/arconfiguration.worldalignment

[5] [cit. 10. 5. 2018]. Dostupné z: http://www.knowyourmobile.com/apple/apple-iphone-6s/23316/iphone-6s-3d-touch-display-explained-ios-9-iphone-7-ios-10/

[6] [cit. 1. 5. 2018]. Dostupné z: https://www.gps.gov/systems/gps/modernization/sa/

[7] [cit. 1. 5. 2018]. Dostupné z: https://www.gps.gov/systems/gps/modernization/civilsignals/

[8] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/library/content/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html

[9] [cit. 1. 5. 2018]. Dostupné z: http://galileognss.eu

[10] [cit. 1. 5. 2018]. Dostupné z: https://www.gps.gov

[11] [cit. 1. 5. 2018]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/gnss-mimo-evropu/rusky-glonass/

[12] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/wwdc/

[13] [cit. 1. 5. 2018]. Dostupné z: https://www.apple.com/iphone/compare/

[14] Kohout, T.: *Mobile iOS client for the CTU Navigator*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2013.

[15] Janička, P.: *ČVUT navigátor III - Android klient*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[16] [cit. 1. 5. 2018]. Dostupné z: https://swift.org

[17] *The Swift Programming Language*, ekniha A Swift Tour. Apple Inc., 2014, s. 4–6.

[18] *The Swift Programming Language*, ekniha The Basics. Apple Inc., 2014, s. 39–43.

[19] Apple: What's New in Swift. Video. Dostupné z: https://developer.apple.com/videos/play/wwdc2016/402/

[20] Apple: Protocol-Oriented Programming in Swift. Video. Dostupné z: https://developer.apple.com/videos/play/wwdc2015/408/

[21] [cit. 1. 5. 2018]. Dostupné z: https://swift.org/migration-guide-swift4/

[22] *Using Swift with Cocoa and Objective-C*. Apple Inc., 2014, str. 80.

[23] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/ios/submit/

[24] [cit. 1. 5. 2018]. Dostupné z: http://snapkit.io

[25] [cit. 1. 5. 2018]. Dostupné z: https://github.com/SnapKit/SnapKit

[26] [cit. 1. 5. 2018]. Dostupné z: https://github.com/route-me/route-me

[27] [cit. 1. 5. 2018]. Dostupné z: https://carto.com/docs/carto-engine/mobile-sdk/

[28] [cit. 1. 5. 2018]. Dostupné z: https://cocoapods.org

[29] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/arkit/

[30] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/spritekit

[31] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/scenekit

[32] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/metal

[33] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/arkit/about_augmented_reality_and_arkit

[34] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/arkit/aranchor

[35] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/ios/human-interface-guidelines/user-interaction/3d-touch/

[36] Kodera, J.: *CTU Navigator III - Backend*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2017.

[37] [cit. 1. 5. 2018]. Dostupné z: https://firebase.google.com

[38] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsession

[39] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsessionconfiguration

[40] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsessiontask

[41] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsessiondatatask

[42] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/nsurlsessionuploadtask

[43] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsessiondownloadtask

[44] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/urlsessiondelegate

[45] [cit. 1. 5. 2018]. Dostupné z: https://www.gps.gov/systems/gps/

[46] [cit. 1. 5. 2018]. Dostupné z: https://www.gps.gov/systems/gps/performance/accuracy/#difference

[47] [cit. 1. 5. 2018]. Dostupné z: http://www.czechspaceportal.cz/3-sekce/gnss-systemy/galileo/

[48] [cit. 1. 5. 2018]. Dostupné z: http://www.czechspaceportal.cz/gnss-systemy/galileo/komponenty-systemu-galileo/galileo-signaly-a-prijimace/

[49] [cit. 1. 5. 2018]. Dostupné z: https://store.google.com/us/product/pixel_2_specs?hl=en-US

[50] [cit. 1. 5. 2018]. Dostupné z: https://www.gpsbusinessnews.com/Rx-Networks-GSA-Tests-Show-that-Galileo-Increases-LBS-Accuracy_a4838.html

[51] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/support/app-store/

[52] [cit. 1. 5. 2018]. Dostupné z: http://www.openstreetmap.org/about

[53] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/userdefaults

[54] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html#//apple_ref/doc/uid/TP40001075-CH2-SW1

[55] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/uikit/uitabbarcontroller

[56] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/uikit/uialertcontroller

[57] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/uikit/uistepper

[58] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/uikit/uitextview

[59] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/uikit/uislider

[60] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/xcode/

[61] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/testflight/

[62] [cit. 1. 5. 2018]. Dostupné z: https://lldb.llvm.org

[63] [cit. 1. 5. 2018]. Dostupné z: https://developer.apple.com/documentation/foundation/notificationcenter

[64] Petr Felkel, J. S. B. B., Jiří Žára: *Moderní počítačová grafika*, kapitola 21. Computer Press, 2005, str. 542.

[65] [cit. 1. 5. 2018]. Dostupné z: https://www.movable-type.co.uk/scripts/latlong.html

[66] [cit. 1. 5. 2018]. Dostupné z: `https://developer.apple.com/documentation/scenekit/scnnode`

[67] [cit. 1. 5. 2018]. Dostupné z: `https://developer.apple.com/documentation/scenekit/scnscene`

[68] [cit. 1. 5. 2018]. Dostupné z: `https://developer.apple.com/documentation/scenekit/scngeometry`

[69] Martin Mareš, T. V.: *Průvodce labyrintem algoritmů*, kapitola 6. CZ.NIC, 2017, s. 146–148.

[70] [cit. 1. 5. 2018]. Dostupné z: `https://developer.apple.com/documentation/xctest`

Příloha **A**

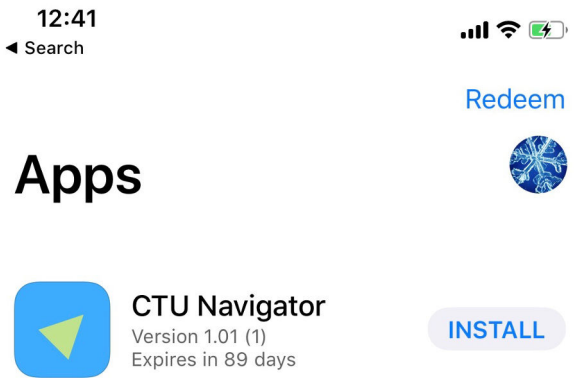
Instalační a uživatelská příručka

Aplikace ČVUT Navigátor se dá spustit pomocí aplikace TestFlight. Aplikace TestFlight slouží pro testování aplikací před jejich následným vypuštěním do obchodu App Store. Aplikace v konkrétní verzi je dostupná 90 dní, poté se musí aktualizovat nebo změnit verze aplikace. Aplikace TestFlight se dá stáhnout z obchodu App Store. Po přihlášení pomocí Apple ID *navigatorctu@gmail.com* a heslem *CTUtest2018* se zobrazí hlavní stránka s přehledem všech dostupných aplikací (obrázek A.1). Po kliknutí na aplikaci ČVUT Navigátor (popřípadě CTU Navigator podle jazykového nastavení iPhonu) se zobrazí detail aplikace (obrázek A.2). V informacích se dá nalézt kompatibilita a datum uvedení do testu. Po kliknutí na tlačítko se aplikace nainstaluje do telefonu a u názvu aplikace na domací obrazovce se objeví oranžové kolečko značící, že se jedná o testovanou aplikaci.

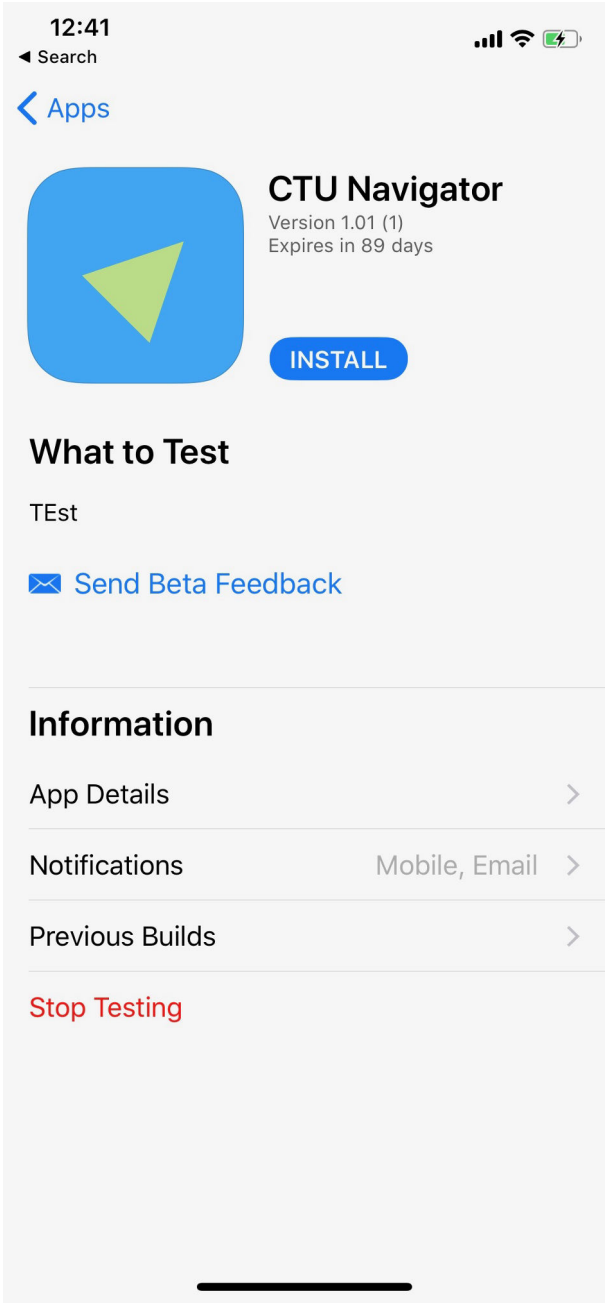
Po spuštění aplikace se objeví přihlašovací obrazovka. Jelikož není aplikace napojená na serverovou část, provádí se přihlášení na straně klienta. Pro přihlášení slouží username *Progtest* a heslo *fit* (záleží na velikosti písmen). Bez přihlašovacích údajů se dá dostat pouze na navigování. Po přihlášení se objeví obrazovka přehled s následujícím předmětem a s novinkami z Fakulty informačních technologií. V dolní části se nachází lišta se záložkami (Přehled, Rozvrh hodin, Navigace, Nastavení). V rozvrhu hodin se zobrazí denní přehled předmětů studenta, po kliknutí se dostane na detail předmětu. V navigaci se nejprve musí vyplnit jednoduchý formulář odkud a kam se bude navigovat. Při psaní textu se aktualizuje nabídka místností podle psaného textu. Po stisknutí tlačítka navigovat se objeví mapa s nakreslenou trasou. Pro testovací účely fungují dvě trasy. První trasa je kombinace určení polohy pomocí systému GPS (popřípadě kombinace GPS + Galileo + GLONASS pokud je podporována zařízením) a určení polohy pomocí značek. Jedná se o cestu od stanice metra Dejvická (v aplikaci nazváno *Metro A*) do přednáškové místnosti *t9:155*. Druhá trasa je pouze v budově a je to trasa mezi vstupem do budovy (nazváno *FA FIT: hlavní vchod*) a přednáškovou místností *t9:155*. QR kódy pro určení polohy se nachází v příloze v části QR kódy pro navigování uvnitř budovy a

jsou seřazeny ve směru cesty od vchodu do budovy (*FA FIT: hlavní vchod*) až po přednáškovou místnost *t9:155*.

Aplikace umí i synchronizovat rozvrh hodin do interní aplikace *Kalendář*. Na obrazovce *Nastavení* se nachází možnost *Synchronizace*. Po stisknutí tlačítka se synchronizuje rozvrh hodin do aplikace *Kalendář*. Jelikož se jedná opět pouze o testovací data, tak jsou všechny předměty synchronizovány na den *23. dubna 2018*.



Obrázek A.1: Aplikace TestFlight - přehled dostupných aplikací



Obrázek A.2: Aplikace TestFlight - detail aplikace

Příloha B

Seznam použitých zkratek

GPS Global Positioning System

GLONASS Globalnaja navigacionnaja sputnikovaja sistěma

UI User Interface

JSON JavaScript Object Notation

MVC Model-View-Controller

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `app` ..... zdrojové kódy aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `extras`
  - `prototype.pdf` ..... první klikatelný prototyp aplikace
  - `qrcodes.pdf` ..... QR kódy pro určení polohy v budově
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha D

Prototyp aplikace - wireframy

Login

Heslo

Zapomenuté heslo

Přihlásit

Přehled

Dnes je
Pondělí 1. 10. 2018

Nasledujici

BI-PA1

9:15 - 10:45 **T9:350**

Zprávy

Zprava 1

Zprava 2

Navigace

Odkud

Kam

Navigovat

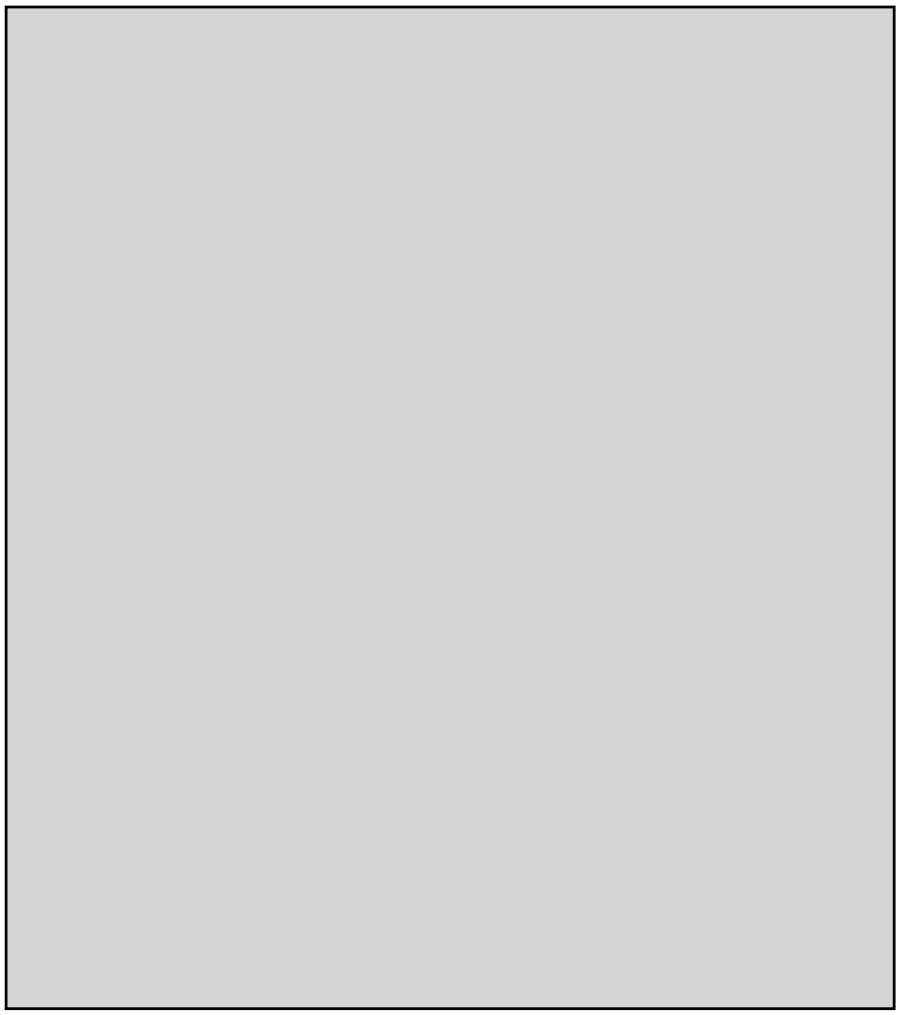
Poslední

T9:350

TH:A-924

Odkud

Naskenovat



Seznam

Zvolit

Seznam

Budova

T9:350

TH:A-924

Zvolit

Navigace

Odkud

T9:350

Navigovat

Posledni

T9:350

TH:A-924

Cesta

Mapa

Vlevo

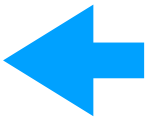
Pro naskenování značky na určení použij horní tlačítko

Galileo test

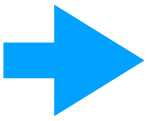
Pro naskenovani
znacky na urceni
pouzij horni tlacitko

Skenování

Rozvrh



Pondělí 1. 10. 2018



BI-PA1

9:15 - 10:45 **T9:350**

Pauza 15 minut

BI-PS1

11:00 - 12:30 **T9:350**

Pauza 5 hodin 30 minut

BI-CAO

18:00 - 19:30 **T9:105**

Nastavení

Zabezpečení

FaceID

Synchronizace

Rozvrh

Barvy

Přednášky

Cvičení

Proseminář

Zkoušky

Odhlásit

Zabezpečení

FaceID 

Žádné

Synchronizovat

Poslední 30. 9 . 2018

Text o synchronizaci

Zabezpeceni

FaceID

Sy

Ro

Ba

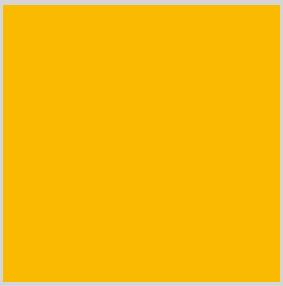
Pr

Cv

Pr

Zk

Přednášky



Red

Green

Blue

Nastavit

Detail

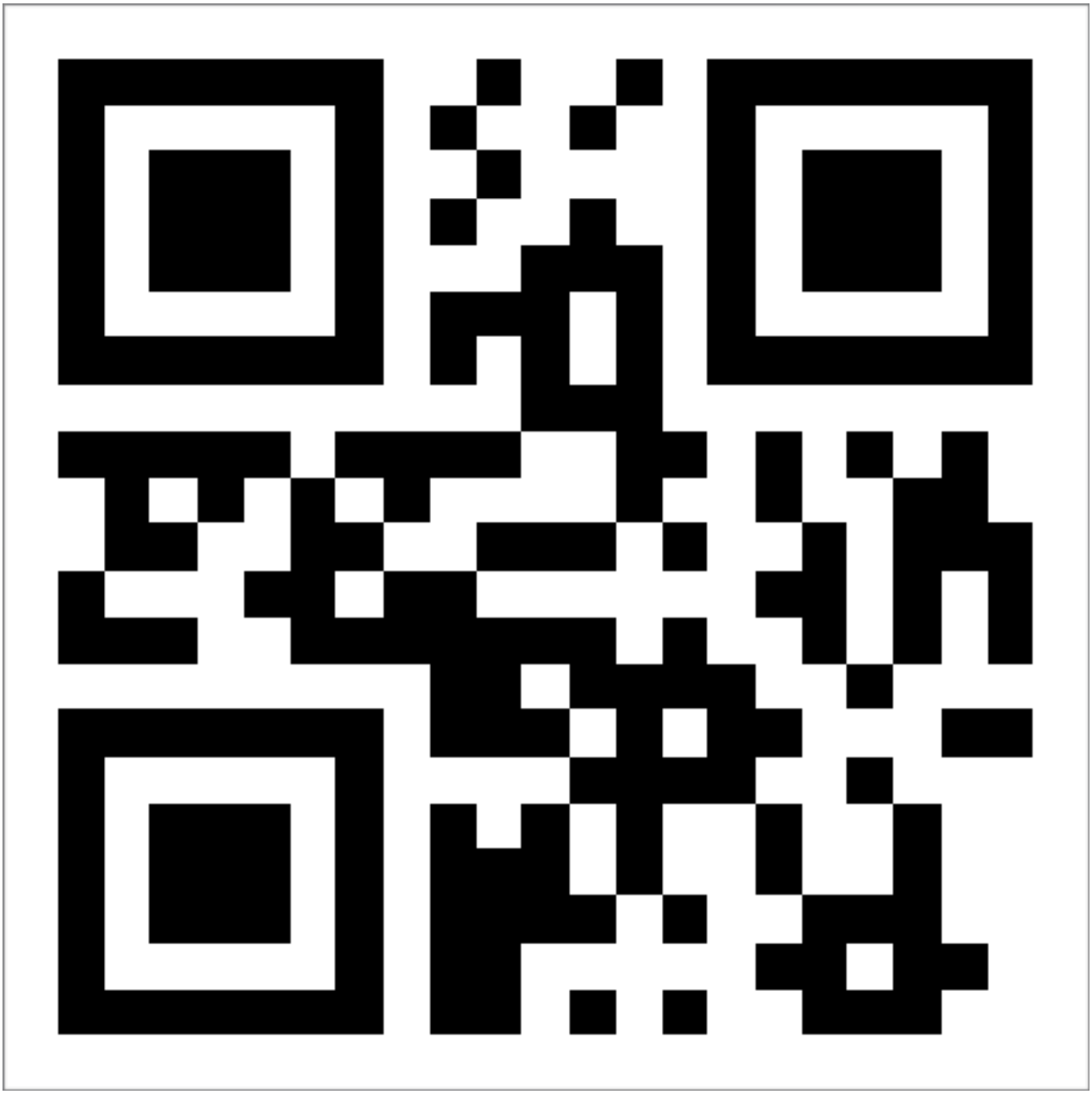
Předmět	BI-PA1
Vyučující	Láďa Vágner
Místnost	T9:350
Začátek	9:15
Konec	11:00
Typ výuky	Cvičení

Navigovat

Příloha E

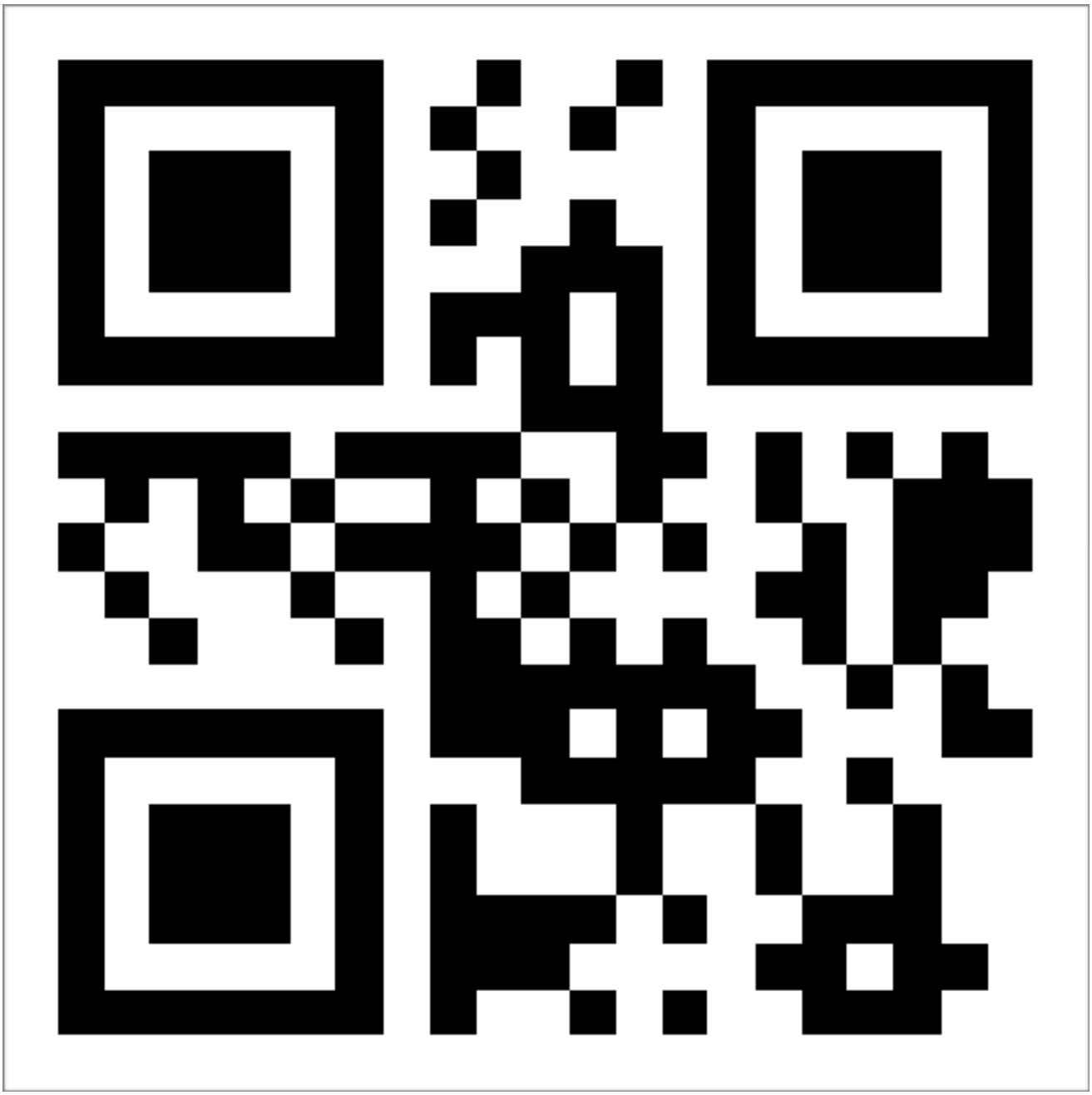
QR kódy pro navigování uvnitř budovy

Před turnikety

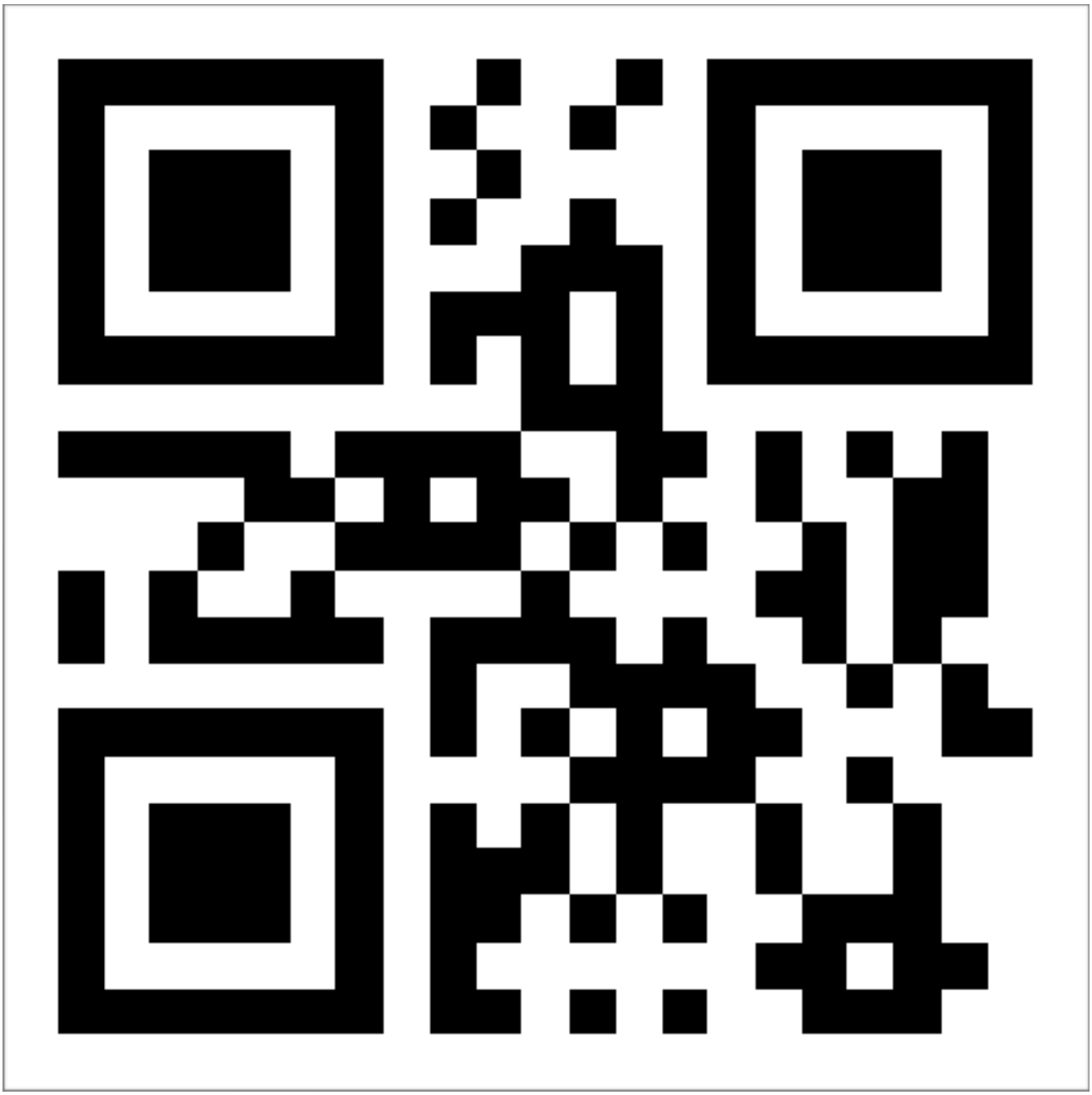


Rozcestí před výtahy

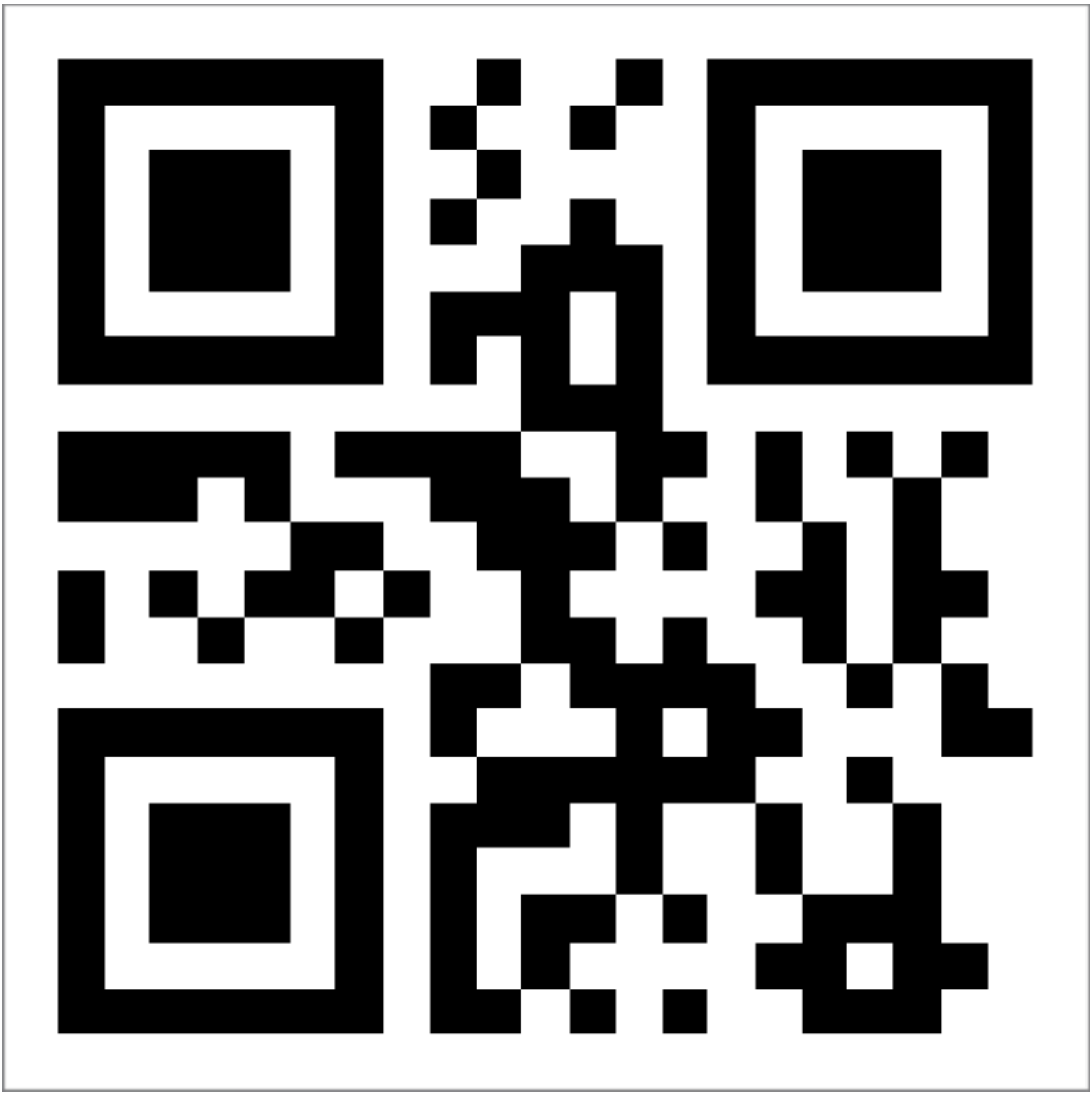
Schody



Rozcestí před přednáškovýma místnostma



t9:105



t9:155