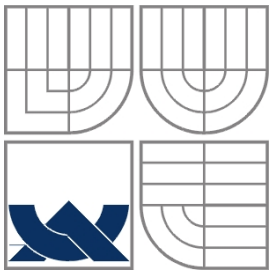
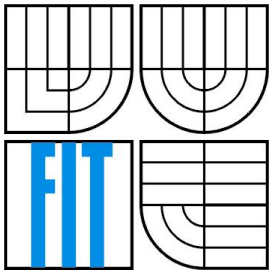


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLIENT VIRTUÁLNÍ ČEKÁRNY PRO IOS

VIRTUAL WAITING ROOM CLIENT FOR IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUDĚK BUKOVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2016

Abstrakt

Cílem této práce bylo navrhnout a implementovat uživatelsky přívětivou mobilní aplikaci pro správu termínu a přihlašování na ně pomocí rezervačního systému. Výsledkem je aplikace pro platformu iOS s příjemným designem. Testování proběhlo na uživatelích, kteří aplikaci ohodnotili kladně s jednoduchým ovládáním. Za velký přínos této práce se považuje vhodně navržené uživatelské rozhraní.

Abstract

The goal of this thesis was to develop and to implement a user friendly mobile application for users to registrate on terms with help of reservation system. The result is an application for mobile platform iOS with enjoyable look. Testing was made with users, who rated application in positive way and easy to use. As a main benefit of this work is considered fitingly developed user interface.

Klíčová slova

iOS, mobilní aplikace, uživatelské rozhraní, Swift, virtuální čekárna

Keywords

iOS, mobile application, user interface, Swift, virtual waiting room

Citace

BUKOVSKÝ, Luděk. *Klient virtuální čekárny pro iOS*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Klient virtuální čekárny pro iOS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Luděk Bukovský
10. května 2016

Poděkování

Chtěl bych poděkovat prof. Dr. Ing. Pavlu Zemčíkovi, který mi umožnil pod jeho vedením práci realizovat. Děkuji také za poskytnutí podpory a inspirace. Dále bych chtěl poděkovat panu Ing. Janu Brejchovi za jeho cenné rady ohledně vývoje.

Obsah

1 Úvod

Moderní technologie jsou všude kolem nás a jen stěží se jim dokážeme vyhnout. Ať už se jedná o digitální tabule na vlakovém nádraží, inteligentní vytápění domácnosti či virtuální čekárnu například u lékaře, kde není potřeba být ve frontě fyzicky přítomen. V dnešní době vstupuje do popředí nový trend zmíněný v posledním příkladu, a to virtuální čekárny. Návštěvy lékaře zná každý z nás a většinou si vyhradíme dostatek času ze svého dne v případě, že se bude v čekárně nacházet mnoho pacientů. Počet pacientů před námi můžeme odhadnout podle hodin. Než se dostaneme na řadu například s banálním požadavkem, jako předepsání antibiotik, musíme vystát frontu před námi. Těmto dnům může být konec díky mobilní aplikaci, se kterou si zvolíme termín v čase, který nám vyhovuje. Díky tomu budeme moci lépe pracovat se svým časem, a zorganizovat si ho co nejefektivněji. Jednoduše si budeme moci zobrazit aktuální počet pacientů v čekárně a rozhodnout, zda se vyplatí dostavit dnes, nebo je výhodnější registrace na konkrétní termín v jiný den.

Není to pouhým usnadněním pro pacienty, ale také pro lékaře. Ti budou schopni pracovat efektivněji, neboť budou vědět, s jakým zákrokem se má pacient dostavit. Trhání osmiček zabere jistě více času s veškerou přípravou zahrnující lokální umrtvení úst, než aplikace zubní plomby. Lékař bude tedy informován o budoucím zákroku a bude schopen předem připravit ordinaci k jeho potřebě a ošetřit pacienta v kratší době.

Cílem práce je navrhnout a implementovat mobilní aplikaci pro operační systém iOS s především jednoduchým a uživatelsky přívětivým rozhraním, aby se dobře používala. Měla by umožňovat zobrazení podniků, přepážek, front a v případě zájmu registraci na termín.

Součástí práce je analýza existujících řešení a výběr vhodného postupu při řešení. Další velmi důležitou součástí je definice budoucí cílové skupiny uživatelů, a dle toho zvolit vhodné uživatelské rozhraní, neboť právě to bude hrát klíčovou roli, zda se uživatel k aplikaci v budoucnu vrátí či nikoli.

Celý text je členěn do kapitol, které popisují jednotlivé části vývoje. Jako první je popsána analýza požadavků nutných pro splnění vývoje pro platformu iOS, pravidla návrhu uživatelského rozhraní, implementační prostředky, architektura systému a vysvětlení pojmu vyvolávací systém. V další kapitole proběhlo zhodnocení současného stavu a následné vytvoření cílů. Následuje analýza požadavků a návrh architektury. Samotná implementace a využité nástroje jsou popsány v kapitole 4. Předposlední kapitola se zabývá testováním a výsledky z testování. Poslední kapitolou je závěr, kde jsou shrnuty dosažené výsledky aplikace a možnosti dalšího pokračování.

2 Současný stav technologií

Tato kapitola obsahuje stručné shrnutí současného stavu technologií, aplikací se zaměřením na virtuální čekárny a prostředí iOS. Obecné shrnutí by bylo rozsáhlé a v práci by pro něj nebyl prostor a jednak by to v práci nebylo účelné. Proto jsou v této kapitole uvedeny jen poznatky s bezprostředním vztahem k bakalářské práci.

2.1 Úvod do vývoje pro operační systém iOS

Aplikace bude zaměřena pro mobilní zařízení využívající operační systém iOS a je dobré tento systém a principy pro vývoj [2] na něm stručně představit.

- **Apple a iOS**
  Operační systém iOS je vyvíjen firmou Apple Inc. a poprvé byl představen v roce 2007 pro mobilní telefon iPhone. Ve stejném roce také pro iPod Touch. V roce 2010 pro tablet iPad a Apple TV. Nakonec v roce 2012 začal používat iOS také iPad Mini. První verze iPhonu v roce 2007 obsahující iOS byla revolucí pro mobilní trh. Zkratka iOS je původně odvozena od iPhone OS, ale dnes je již dobře zaveden název iOS. V minulosti byl iOS určen pouze pro mobilní zařízení iPhone a iPod Touch. Dnes je již rozšířen na všechna zmíněná zařízení. Po operačním systému Andorid, je iOS druhým nejrozšířenějším systémem pro chytré telefony[14].

  Aktuálně je dostupná verze systému 9.3.1 a tato verze je použita i v době psaní této práce. Verze systému iOS 9 byla představena na WWDC v roce 2015 [12]. Za novinky této verze se považuje chytřejší vyhledávání v Spotlightu, propracovanější mobilní asistent Siri. Pro vývojáře jsou od této verze novinkou 3D Touch, vylepšení programovacího jazyka Swift, nový kontrolér UIStackView a zachytávání výjimek v kódu pomocí bloků try-catch-finally pro Swift. Jako poslední změnu, která se týká už od iOS 7 je nový rovný design celého systému a přívětivější vzhled.

- **Vývoj aplikace pro iOS**
  Vývoj pro Apple na platformách iOS a OS X musí splňovat několik podmínek. Jednou z nich je být oficiální vývojář a s tím souvisí zakoupení vývojářské licence. Licence pro vývojáře stojí ročne 99 dolarů. Bez licence se aplikace nikdy nedostane do obchodu App Store a byla by použitelná pouze na lokálním zařízení. Další nutností je mít svůj vlastní Apple ID účet, který slouží jako identifikátor přístupu ke službám nabízeným společností Apple Inc. Pro vývoj jako takový budeme ještě potřebovat iOS SDK (Software Development Kit), znalost programovacího jazyka Objective-C nebo Swift, počítač od společnosti Apple s označením Mac se systémem OS X a vývojové prostředí Xcode.

- **Xcode**
  Xcode je vývojové prostředí od společnosti Apple Inc., které je ke stažení zdarma. Podmínkou vývoje v Xcode je mít operační systém OS X, které jsou určeny pro počítače firmy Apple. Xcode obsahuje obdobné vlastnosti, jako další IDE (Integrated Development Enviroment) s přidanou

výjimkou simulátoru. Díky simulátoru se může aplikace testovat bez potřeby mít reálné zařízení přímo na počítači Mac.

Není třeba zacházet do větších detailů. Byly představeny ty nejdůležitější věci týkající se vývoje pod iOS.

2.2 Programovací jazyky

Původním jazykem pro vývoj pro Apple zařízení je Objective-C z roku 1986. Tento jazyk je objektově orientovaný programovací jazyk, který rozšiřuje původní jazyk C s doplněním systému pro zasílání zprav jazyka Smaltalk. Dalšími vlastnostmi Objective-C je oddělení implementace a rozhraní do samostatných souborů. Do nedávna byl Objective-C používán v operačních systémech iOS a OS X. V roce 2014 na WWDC byl však společností Apple představen zcela nový programovací jazyk Swift [4]. Tento jazyk by měl dodržovat modernější programovací paradigmata, nahradit Smaltalkový systém zpráv pomocí tečkové notace a umožňuje psát samotný kód rychleji a čitelněji. Swift můžeme chápat jako novější alternativu jazyka Objective-C. Stejně jako jeho předchůdce, je i Swift objektově orientovaný. Můžeme v něm pracovat s již existujícími frameworky jako Cocoa Touch. Nyní nejsou vyžadovány hlavičkové soubory *.h a příkazy na konci řádku nemusí být zakončeny středníkem. Swift je kompilován pomocí LLVM[1] a díky tomu může být kombinován s ostatními programovacími jazyky jako jsou C, C++, Objective-C++ a Objective-C v jedné aplikaci.

2.3 Uživatelské rozhraní a zásady v iOS

Uživatelské rozhraní umožňuje ovládat zařízení uživatelem dle interakcí, které vykoná.
V některých případech jako příchozí hovor může zařízení vyvolat interakci samo a uživatel má na výběr hovor přijmout nebo zamítnout. Uživatel má kontrolu nad tím, že když klikne na dané tlačítko, dostane daný výstup. Jako zařízení je v tomto kontextu myšlena aplikace pro systém iOS využívající potřebný hardware. Uživatelské rozhraní je v dnešní době klíčovým prvkem a pokud aplikace nezaujme ihned na první pohled, je pravděpodobné, že uživatele odradí od jejího dalšího použití [8]. Rozhraní musí být intuitivní a lehce stravitelné bez zbytečných prvků, které by mohly rušit uživatele v průběhu využití aplikace. Běžný uživatel dokáže zhodnotit, jak aplikace vypadá, ale nemá tušení, jak pracuje v pozadí. Když aplikace bude nabízet všechny funkce a požadavky, ale bude nevhodně zvolené uživatelské rozhraní, uživatel se pravděpodobně poohlédne po jiné alternativě za cenu, že aplikace nebude mít takovou funkcionalitu, ale bude se s ní dobře pracovat [8].

Aby se předcházelo zbytečnému odrazování uživatelů, Apple vydal publikaci *iOS Human Interface Guidelines* [2], kde by se měl vývojář držet těchto principů. Tato práce se bude také řídit tímto návodem, neboť výsledná aplikace bude právě pro systém iOS. Je zde popsáno, jakých způsobů se držet při navrhování uživatelského rozhraní. Dále vyzývá vývojáře, jaké formáty textu použít, jak velké ikonky tvořit a další informace.

[1] Low Level Virtual Machine - optimalizovaný překladač

Konkrétní pravidla uživatelského rozhraní pro iOS

Pravidla zmíněná v iOS Human Interface Guidelines [2] jsou uvedena podrobněji níže.

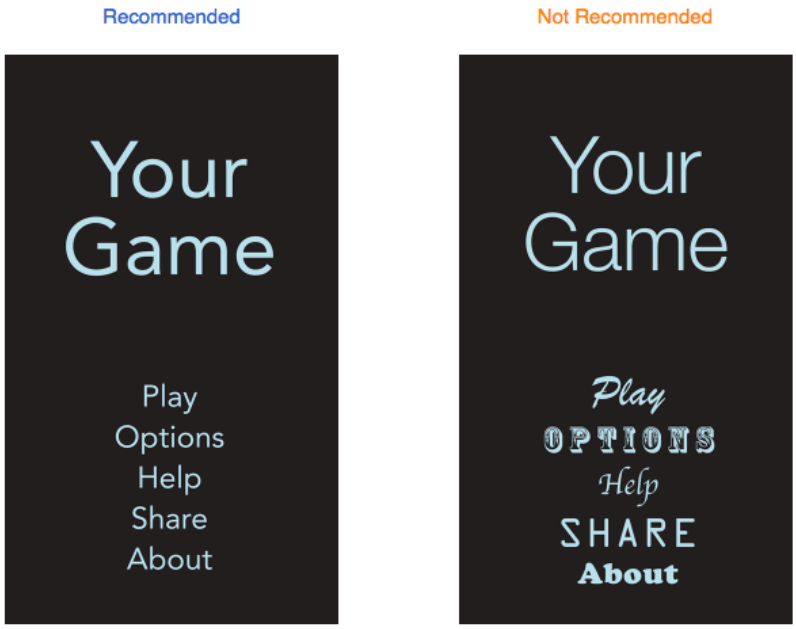
- **Přehlednost** - text by měl být čitelný v různých velikostech, ikonky by měly být přesné a jasné. Ozdoby by měly být jemné a vhodné.

- **Použití prostoru celé obrazovky**

- **Použití barev** - vhodné použití kombinací barev je v iOS klíčové a lze vidět ve vestavěných aplikacích systému iOS.

- **Konzistence** - ovládací prvky by měly splňovat zažité standardy platformy, které jsou známé z jiných aplikací a při jejich použití se očekává, že se aplikace zachová stejně či podobně.

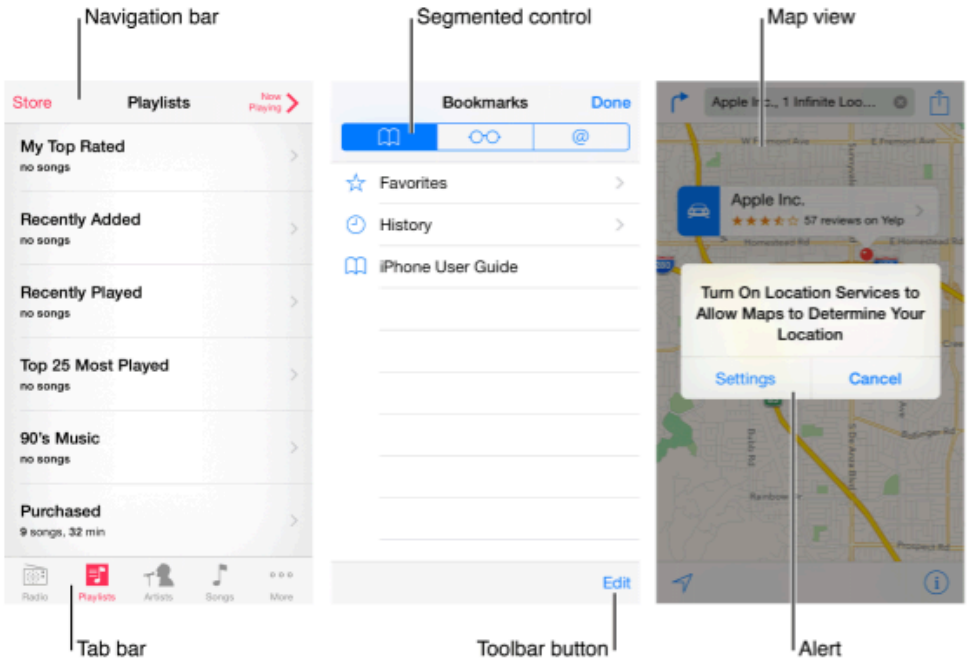
- **Použití textu a jeho velikosti** - text by měl být čitelný a iOS nabízí systémové formáty písma vhodné pro použití v aplikacích. Text by měl být čitelný ve všech velikostech použití v aplikaci. Velmi vhodné je použití jednoho formátu textu pro celou aplikaci.



Obrázek 2.1: Ukázka správného použití textu [2]

- **Integrita** - aplikace by měla mít vzhled odpovídající jejímu použití. Při pracování s aplikací kalendář by se nehodily dekorativní prvky, které by narušovaly hladké použití kalendáře, kdežto ve hrách jsou dekorativní prvky a efekty chtěné.

- **Přizpůsobení layoutu** - aplikace by měla být dobře čitelná na všech zařízeních s různým rozšířením displeje. Nemělo by záležet na tom, zda se jedná o použití aplikace na výšku či šířku v landscape módu.



Obrázek 1.2: Základní prvky systému iOS [2]

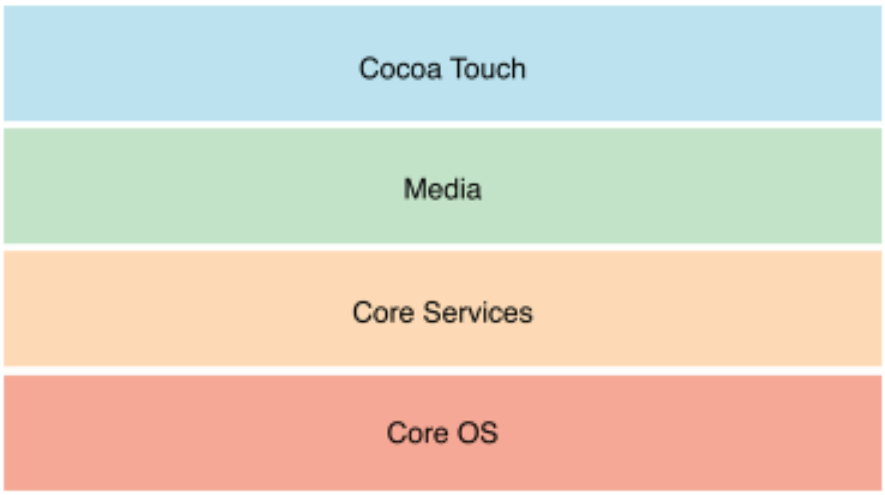
- **Zpětná vazba** - uživatel by měl dostat zpětnou vazbu například po kliknutí na položky seznamu a to třeba zvýrazněním prvku, na který bylo kliknuto. Zvýraznění může být dále v podobě animací, vibrací nebo zvukovým tónem. Při déle prováděných operacích by měl být uživatel upozorněn, že aplikace pracuje a bude vědět, že má vyčkat na dokončení operace. Dále uživatelé mohou využít známých gest, která znají z jiných aplikací. Příkladem může být obrázek a při požadavku jeho přiblížení mohu použít již dobře zažitých gest jako *pinch to zoom* či *double tap*.

- **Nevyžadovat přihlášení do aplikace, pokud není potřeba** - uživatelé často zavrhnou aplikaci, která jim nedovolí první seznámení s jejím obsahem, ale ihned po nich žádá zdlouhavou registraci. Příkladem může být AppStore, kde si uživatel může prohlížet obsah aplikací, ale přihlášení vyžaduje až v případě požadavku pro stažení aplikace.

- **Uživatel má kontrolu nad aplikací** - uživatelé, nikoli aplikace, by měly iniciovat akce. Aplikace může navrhnout postupy nebo varování o nebezpečných důsledcích, avšak neměla by rozhodovat za uživatele. Uživatel se tak cítí ve větší kontrole nad aplikací.

2.4 Architektura iOS

Operační systém iOS je systém UNIXového typu a je odlehčenou verzí operačního systému OS X pro počítače Mac. Neobsahuje veškerou funkcionalitu OS X, ale zato přidává podporu pro dotykové ovládání systému. Systém se dělí na čtyři základní vrstvy zajišťující základní funkčnost [3]. Tyto vrstvy poskytují frameworky a API pro vývojáře, které jsou potřebné k vývoji. Vrstvy lze vidět na [2.3].



Obrázek 2.2: Vrstvy iOS [3]

Popis jednotlivých vrstev může být následující.

- **Vrstva Cocoa Touch**
  Vrstva je nejvyšší vrstvou v hierarchii a obsahuje nejdůležitější frameworky pro vývoj aplikací na iOS. Technologie v této vrstvě poskytují infrastrukturu pro implementaci grafického rozhraní a interakci s uživatelem. Dále poskytuje vysokoúrovňové systémové služby jako jsou *multitasking, push notifikace, rozpoznávání gest, sdílení souborů* a mnoho dalších služeb. Při vývoji aplikací je vhodné začínat právě s touto vrstvou a nižší používat pouze v případě potřeby.
  Příklady frameworků: MapKit, GameKit, Auto Layout

- **Vrstva Media**
  Tato vrstva obsahuje technologie pro video, grafiku a audio. Tyto technologie lze v aplikaci využít a zajišťují plynulé přehrávání animací, videí a zvuků.
  Příklady frameworků: Core Image, Core Media, Metal

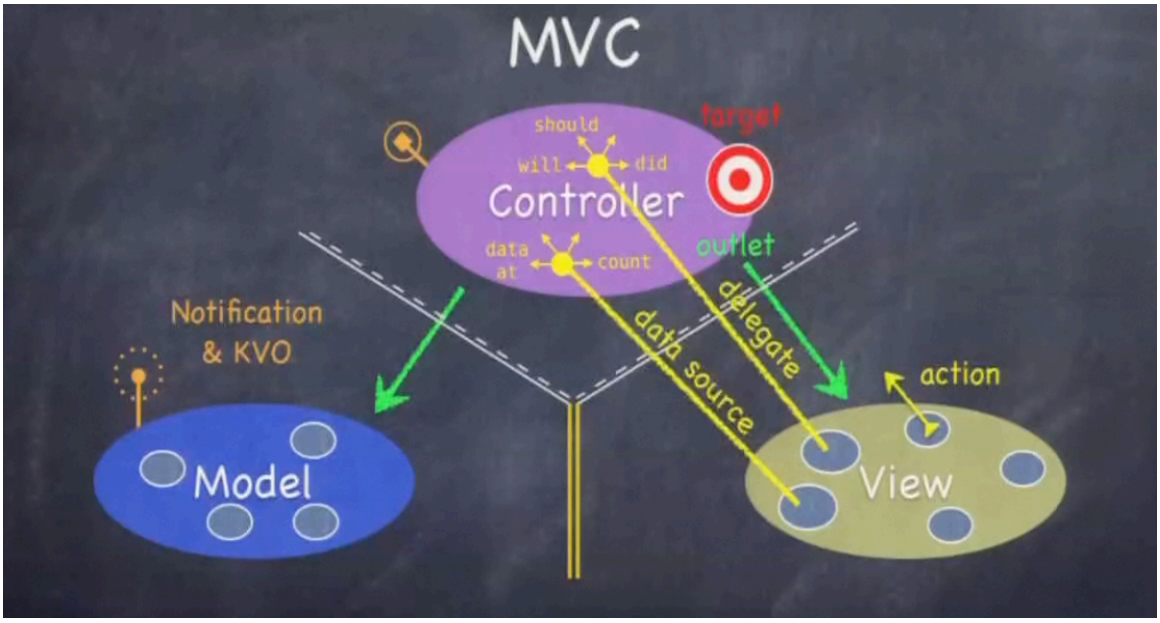
- **Vrstva Core Services**
  Vrstva obsahuje základní systémové služby pro aplikace. Nachází se zde služby jako lokalizační služby, práce s daty a databází, automatická reference objektů a jiné.
  Příklady frameworků: Core Location, Core Motion, Multipeer Connectivity

- **Vrstva Core OS**

Tato vrstva je poslední vrstvou a obsahuje nízkoúrovňové funkce ostatním technologiím, které jsou postaveny právě na ní. K této vrstvě nemají iOS aplikace přímý přístup. Najdeme zde prostředky pro alokaci paměti, matematické funkce a další.
Příklady frameworků: Security, External Accessory, Core Bluetooth

2.5 Koncepce Model-View-Controller (MVC)

Vývoj pod iOS využívá softwarovou architekturu, která rozděluje funkcionalitu aplikace do tří kategorií a změna v jednotlivé kategorii má minimální vliv na ostatní části. Model obsahuje data a operace nad nimi. View je část aplikace, kterou uživatel vidí a může s tímto objektem komunikovat pomocí interaktivních gest, jako kliknutím či přesunutím prstu z jednoho konce obrazovky do druhého. Jak bude View vypadat si můžeme sami navrhnout pomoci Storyboardu, které využívá vývojové prostředí Xcode. View zobrazuje data, která jsou uložena v Modelu. Controller je prostředník mezi View a Modelem reagující na podněty, které uživatel vyvolá ve View. Následně provádí změny v Modelu nebo View. Model a View by spolu nikdy neměly komunikovat přímo a proto je tu právě tato mezivrstva [6]. Jak funguje MVC pod iOS můžeme vidět na obrázku [2.4].



Obrázek 2.4: Model-View-Controller [1]

2.6 Vyvolávací systém

Vyvolávací systém [11] zná každý z nás a setkali jsme se s ním na místech jako banka, úřad, pošta, zdravotnictví či jiných podobných službách. Cílem tohoto systému je odbavování klientů na základě vyvolání, které se odvíjí v čase, kdy přišli. Na rozdíl od supermarketů, kde se stojí fronta na odbavení u pokladny, se klient může posadit v takzvané čekárně a vyčkat si, než bude vyvolán a vyzván, aby se dostavil na přepážku. Vyvolávací proces probíhá způsobem, kdy se klient dostaví do chtěného podniku a

při vstupu si vyzvedne lístek s přiděleným pořadovým číslem. Lístek zvolí podle oddělení, které odpovídá jeho potřebě a zařadí se do fronty.

Jako příklad uvedu Magistrát města Brna, kde je několik oddělení pro vydání průkazů. Průkazy se dělí na občanské, cestovní a řidičské. Při požadavku na vydání cestovního pasu se budu chtít zařadit do fronty klientů čekajících přede mnou. Registrace do systému může být zhotovena na místě pomocí automatu na výdej pořadových čísel nebo pomocí elektronického objednání. Při dokončení registrace o lístek žadatel vyčká, než přidělené číslo bude vyvoláno. Vyvolání může být pomocí informační tabule, hlasového vyvolání, textové zprávy na mobilní telefon nebo internetu. Je zvyklostí, že při vyvolání je klientovi sděleno, na které přepážce bude odbaven. Odbavení může být u přepážky či v místnosti. Zohledňování odbavení může být postupně, nebo pomocí algoritmu, kde se zjišťuje, kdy a který pracovník je schopen klienta obsloužit.

Za obrovskou výhodu těchto systémů se považuje efektivní využití pracovníků, podle jejich kvalifikace. Tímto je myšleno, že se oddělují klienti dle jejich potřeby. Klient žádající o vyšetření zraku by se nerad přihlásil k ortopedickému lékaři. Za nevýhodu těchto systémů se může považovat obtížná odhadnutelnost čekací doby.

Obrázek 2.5: Vyvolávací systém [11]

V současnosti lidé používají chytrý telefon mnohem častěji, než dříve [15]. Tomuto faktu se přizpůsobil trh a rozšířil možnost využití pro tablety a chytré telefony. Výsledkem je několik aplikací dostupných pro platformy iOS, Windows Phone a Android. Tyto aplikace typu rezervační systém mají jedno společné, a to specializaci na vyvolávací systém. Mezi nejrozšířenější nabízené služby patří možnosti přidání do fronty v odvětvích restauracích, úřadech, obchodních korporací jako T-Mobile a zdravotnictví. Jelikož se jedná o nový trend, současnost nenabízí mnoho řešení a tyto možnosti se objevují v posledních dvou letech. Aplikace jsou převážně v anglickém jazyce. Níže budou popsány již existující řešení.

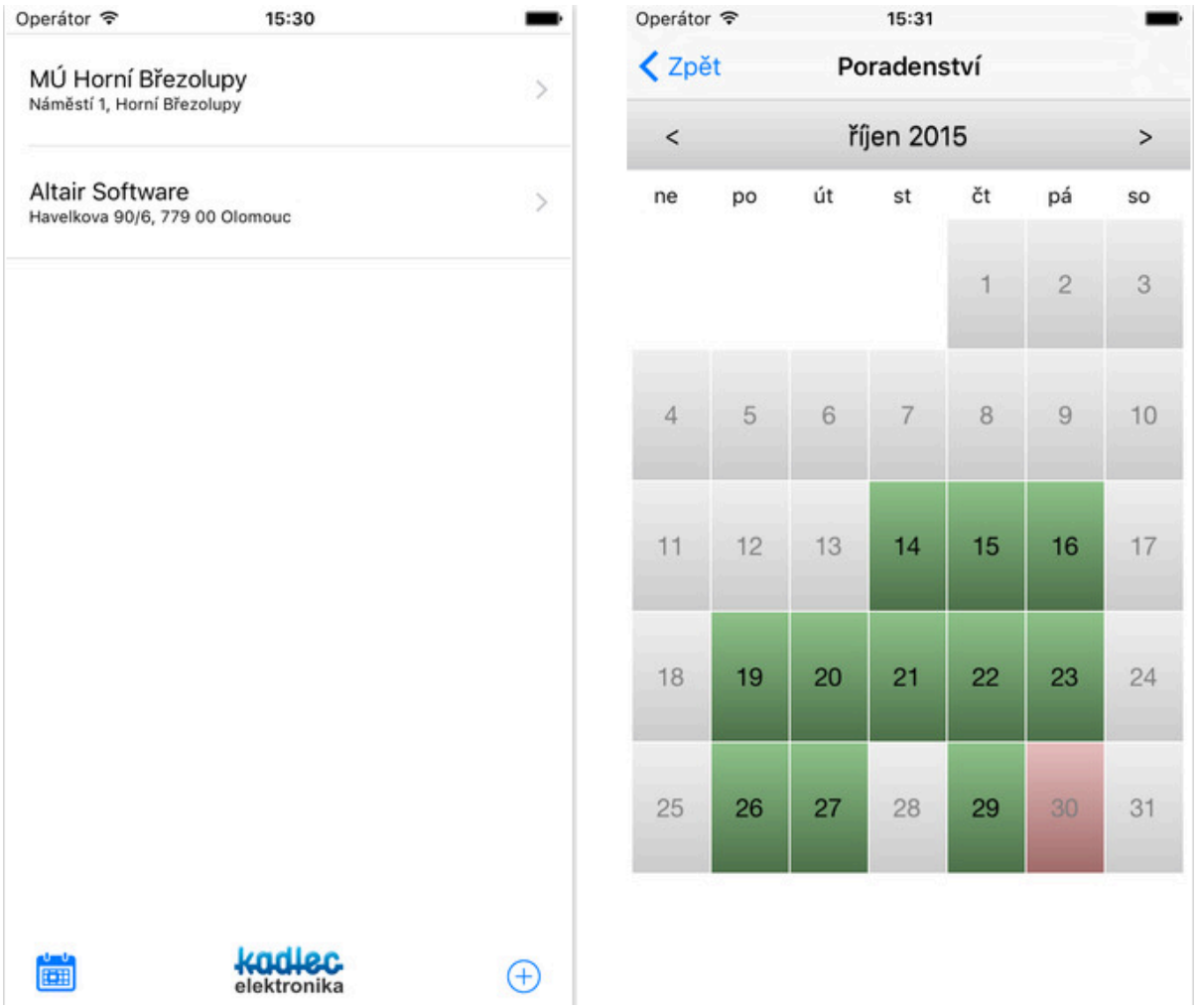
2.7 Příklady existujících vyvolávacích systémů

Zde jsou shrnuty vyvolávací systémy pro platformu iOS.

WebCall

Aplikaci na míru si nechala vyhotovit společnost Kadlec Elektronika s.r.o., která se na českém trhu soustředí výhradně na vyvolávací systémy nabízející komplexní řešení pro zákazníky ze státní a komerční sféry. Mobilní aplikace WebCall umožňuje objednání se do vyvolávacího systému na konkrétní datum a čas, kdy se klient dostaví na zvolený termín, na který se objednal. Následně po příchodu do čekárny je

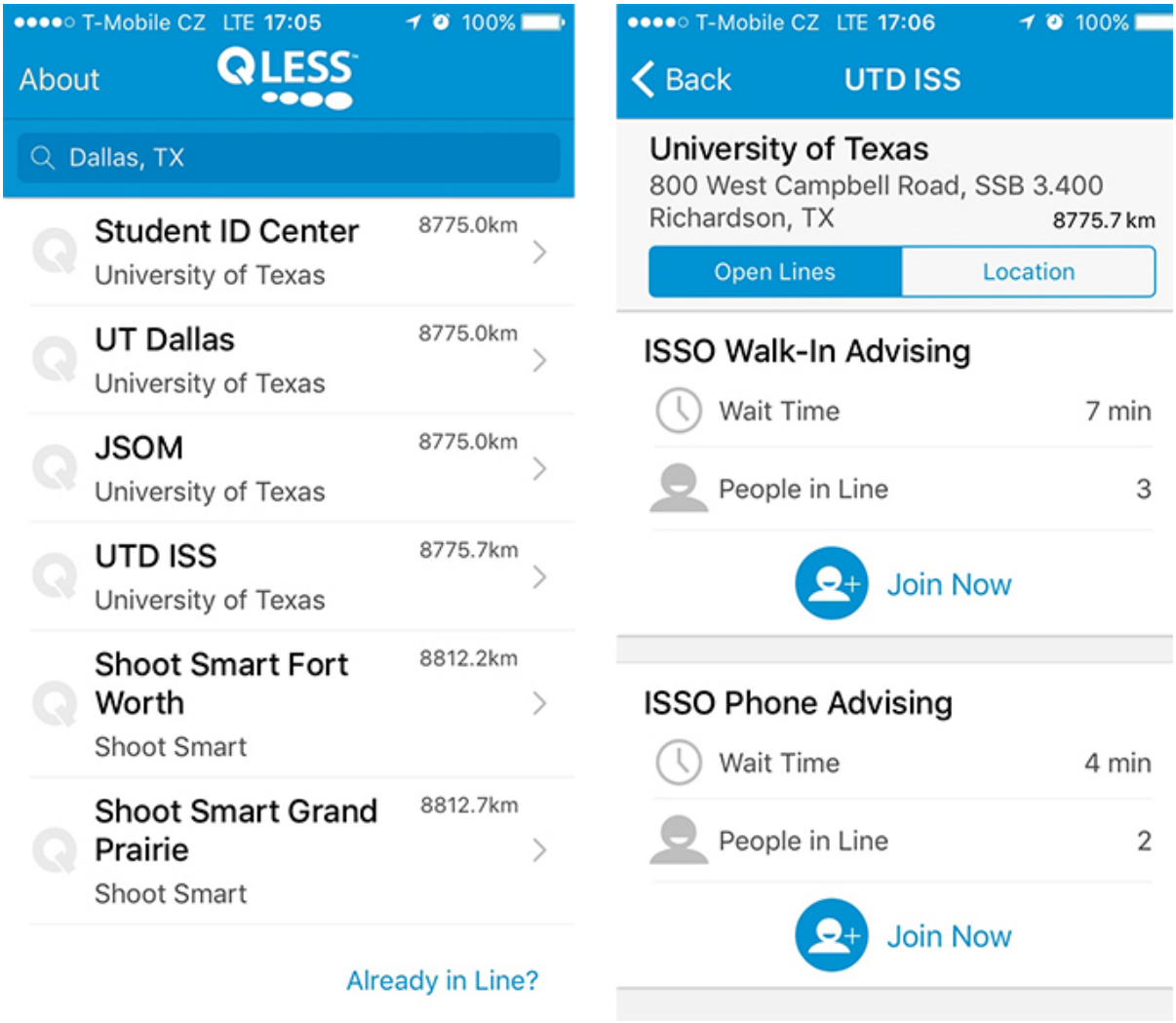
klient vyvolán s prioritním číslem, díky kterému se vyhne zdlouhavému stání fronty. Jako bezpečnostní faktor před fiktivními klienty se návštěvník po dostavení do čekárny autentizuje pomocí ověřovacího kódu vygenerovaného aplikací. Při nedostavení do čekárny není žádný postih. Příklad aplikace je zobrazen na obrázku [2.6]



Obrázek 2.6: Snímky obrazovek aplikace WebCall

QLess Concierge

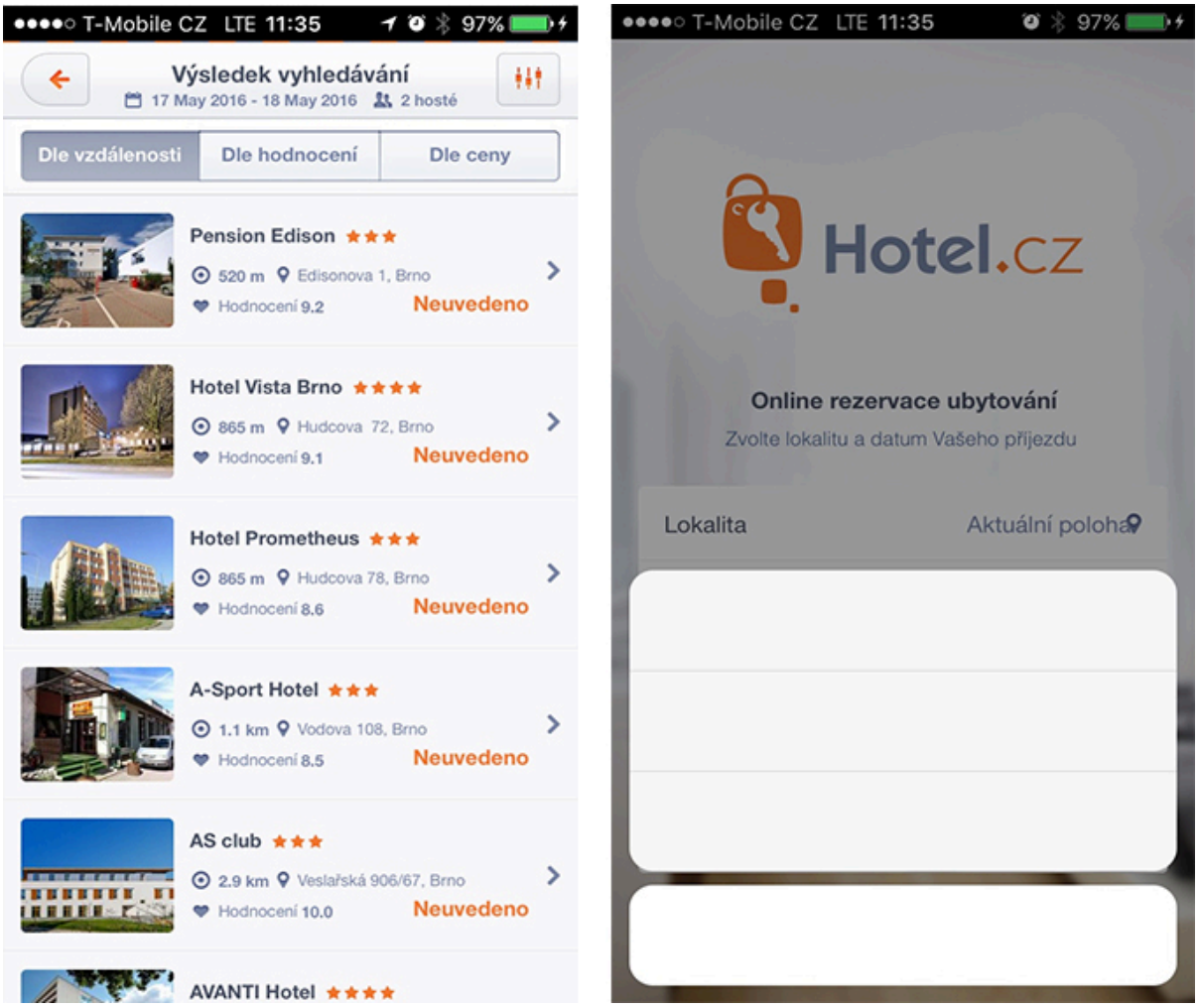
Aplikace QLess Concierge se soustředí především na americký trh a nabízí několik specifičtějších odvětví jako lékařství, úřady, maloobchodní sítě, školství a dalších odvětví. Jedná se o kapesní manažer front virtuálních čekáren pro zákazníky. Do aplikace se lze registrovat na základě telefonního čísla a následného obdržení textové zprávy o podrobnějších informacích.



Obrázek 2.7: Snímky obrazovek aplikace QLess Concierge

Hotel.cz

Aplikace Plazaro je hotelový rezervační systém ke správě ubytování. Jedná se o doplněk k online rezervačnímu systému. Hotel.cz je určen pro lidi hledající ubytování pomocí chytrého zařízení a nabízí informace o ubytování v okolí, dle hodnocení a podle ceny Aplikace. Zařízení jsou rozdělena na hotely, penziony, rekreační a lázeňská střediska.



Obrázek 2.8: Snímky obrazovek aplikace Hotel.cz

3 Analýza a návrh struktury aplikace

V této kapitole je analyzován současný stav aplikací pro virtuální čekárny a shrnut popis pomocí požadavků na aplikaci. Následně je uvedena vize aplikace. Poté je vyroben diagram případů užití, který reflektuje jednotlivé funkce aplikace

3.1 Zhodnocení současných řešení

Výše zmíněné aplikace pro rezervaci termínů provádí principielně to stejné, avšak u každé postrádám několik vlastností.

U aplikace Webcall mi schází zajistit penále za nedostavení se do čekárny, a tím nechat účastníky před námi čekat. Jistým penálem by se toto riziko snížilo. Při potvrzení termínů v aplikaci vyskočí varovné okno o termínu s informacemi přes celou obrazovku a nutí uživatele o jedno kliknutí navíc. Řešením by mohla být notifikace textovou zprávou či emailem. Jednou z věcí, která mi u tohoto řešení schází je tab bar, neboli spodní menu v aplikaci pro rychlejší orieantaci. Aplikace je prozatím ve zkušebním provozu a obsahuje tři podniky, kde dva z nich se tváří být funkčí. Dále bych se chtěl v mém řešení vyvarovat neznalosti uživatele o aktuální kapacitě čekárny.

Další aplikace QLess byla navržena tak, aby některé problémy řešila, avšak je dle mého názoru již příliš rozsáhlá a obsahuje mnoho prvků. Chybí podpora českých podniků.

	Nutná registrace	Informace o kapacitě	Příliš rozsáhlé rozhraní	Česká podpora podniků	Penále za nedostavení
WebCall	ne	ne	ne	ano	ne
QLess	ano	ano	ano	ne	ne
Hotel.cz	ne	ne	ano	ano	ano
Návrh	ne	ano	ne	ano	ano

Tabulka 3.1: Srovnání aplikací

Poslední aplikací pro srovnání byl Hotel.cz, který dopadl snad nejhůře. Není zde na možnost zvolit datum příjezdu, a tím aplikace hodně ztrácí, protože dává na výběr pouze z registraci z aktuálního dne a to na jeden den, neboť není možné nastavit ani datum odjezdu. Při zkušebním pokusu o vyplnění se aplikace vypnula. Nebyla tedy zmínka o tom, zda se rezervace dokončila nebo ne.

Z popisu dostupných aplikací se dá usoudit, že aplikace zabývající se rezervačními systémy nejsou úplně špatné, ale pro aplikaci, kterou chci vytvořit nejsou vhodné. Často vyskytujícím se problémem je již rozsáhlé a komplikovaně řešené uživatelské rozhraní. V některých případech chybí aktuální informace o volné kapacitě i přes mnoho dalších prvků, které nejsou dle mého důležitější, než aktuální volný termín. V tabulce [3.1] je pro grafický přehled srovnání aplikací s navrhovanou aplikací a jsou vybrány ty nejdůležitější požadavky na funkce.

První sloupec značí, zda je pro práci s aplikací nutná registrace. Druhý sloupec značí informace o tom, zda je kapacita na danou registraci volná. Prostřední sloupec podává informace o složitosti využití příliš mnoho prvků, co se týče uživatelského rozhraní. Předposlední sloupec udává, zda program

podporuje české podniky a poslední sloupec značí, zda se bude vybírat penále za nedostavení se ve zvoleném termínu.

Mým cílem je vytvořit aplikaci, která by z větší části nahradila existující řešení a doplnila komponenty, které schází. Aplikaci propojující internet a vyvolávací systém s jednoduchým a zároveň přehledným uživatelským rozhraním, aby byla práce s mou aplikací pro uživatele intuitivní a uživatelé se k ní rádi vrátili. Dále by měla mít aplikace podporu českého jazyka.

3.2 Definice cílové skupiny

Při návrhu uživatelského rozhraní je důležité mít na vědomí konečnou množinu uživatelů. Aplikace bude očekávat uživatele, kteří se chtějí zaregistrovat do virtuálních čekáren pomocí chytrého mobilního zařízení. Dnes vlastní chytré zařízení téměř většina populace [15] a uživatelské rozhraní by mělo splňovat uhlazený a jednoduchý design, aby práce s aplikací byla intuitivní pro každého. Předpokládají se uživatelé využívající chytré zařízení například pro práci s internetem, a ne pouze pro psaní textových zpráv, nastavování budíků nebo pořizování fotografií. Aplikace by měla mít takový design, že uživatelé po prvním spuštění budou naprosto přesně vědět, kde se nachází, jak se dostanou zpět a jak si zaregistrovat termín.

3.3 Analýza požadavků

Při návrhu aplikace se musí vývojář řídit požadavky zadané zákazníkem. Výsledný produkt musí tyto cíle dodržet, aby splňoval chtěnou potřebu zákazníka a mohl navíc tvořit konkurenční výhodu.

Tato kapitola se bude zabývat dvěma druhům požadavků, které lze rozdělit na funkční a nefunkční. Ve zkratce můžeme říct, že co systém dělá, spadá pod funkční požadavky, ale jak se systém chová, spadá pod nefunkční požadavky.

Nefunkční požadavky

Cílem nefunkčních požadavků [13] je vyvinout aplikaci, která bude kvalitní. Kvalita se měří podle kritérií, které zákazník ocení nejvíce. Nefunkční požadavky zmíněny v aplikaci můžeme klasifikovat následovně.

- **Použitelnost** - Požadavek značící, jak je aplikace jednoduchá na ovládání pro uživatele v průběhu, kdy s ní bude pracovat.

- **Možnost modifikace a rozšíření -** Možnost přidání nové funkcionality nebo její modifikace do aplikace bez toho, aniž by se ovlivnila funkcionalita existující aplikace. Dobrá rozšiřitelnost se uplatňuje použitím objektově orientovaného návrhu v aplikaci, jako využívání rozhraní, zapouzdření komponent a zajištění kvalitního objektového modelu.

- **Výkon -** Požadavek na aplikaci je často měřen, jako doba odpovědi aplikace na požadavek uživatele. Výkon je citlivý na hardware a verzi operačního systému, pokud se aplikace spouští. V

dalším případě se stahují data o podnicích a bude záležet na tom, jak rychlé připojení mobilní telefon bude nabízet a kolik bude k dispozici podniků pro jejich stažení.

- **Bezpečnost -** Bezpečnost je jedna z nejsložitějších kritérií, jak měřit kvalitu aplikace. Aplikace by měla také zajistit integritu dat, kdy data nebudou změněna třetí osobou v průběhu přenosu formuláře na server.

- **Spolehlivost -** Pravděpodobnost, že po určitou dobu bude aplikace fungovat bez pádu.

- **Dostupnost -** V daném časovém okamžiku jsou přístupné všechny podniky aplikace. Příkladem nedostupného systému může být příliš dlouhé zpracovávání a následné zobrazení podniků na požadavek uživatele, kdy se chce přihlásit na nějaký termín. Při dlouhém vyhodnocování uživatel označí aplikaci za nedostupnou.

- **Udržitelnost -** Schopnost v aplikaci modifikovat danou komponentu a při její změně se nemusí brát v úvahu všechny komponenty, ze kterých se celá aplikace skládá.

Funkční požadavky

Funkční požadavky [13] zahrnují hlavní složky, které uživatel očekává při práci s aplikací. Ihned po zapnutí bude zobrazen seznam všech podniků. Ve spodní části aplikace se bude nacházet tab bar menu, které bude viditelné a dostupné ze všech obrazovek. Aplikace bude obsahovat funkci pro zobrazení seznamu podniků, seznamu oblíbených podniků, více informací o aplikaci a aktuálním uživateli. Následně se bude očekávat výběr položky ze seznamu a zobrazení detailu podniku.

Detailní zobrazení by mělo zobrazit pouze důležité informace, aby uživatel nestrávil příliš dlouho rozhodováním, ale aby bylo rozhodnutí jednoznačné. Uživatel zde nalezne tlačítka odkazující na internetové stránky či zobrazení podniku na mapě. Pokud uživatel zhodnotí podnik užitečným a bude plánovat i jeho další návštěvu, bude moci pomocí dalšího tlačítka přidat podnik do oblíbených. Po přidání podniku do oblíbených obdrží uživatel dva způsoby notifikace, a to že byl podnik přidán, nebo že je podnik již v oblíbených.

Poslední možností bude dostat se na seznam oddělení, které podnik nabízí. Aplikace bude do sebe zanořovat obrazovky, na které se lze dostat a začíná od podniku, oddělení, front a končí termíny.

Při zobrazení obrazovky s termíny bude mít uživatel možnost aktualizace pomocí tažením prstu dolů. Aktualizace termínů je důležitá, protože ve stejný okamžik se bude moct registrovat jiný zákazník, zatímco druhý zvažoval registraci. Po zobrazení detailu termínu obrazovka odkazuje navíc tlačítko na registraci uživatele na termín. Pro vyplnění informací o uživateli bude k dispozici jednoduchý formulář, který navíc bude obsahovat přepínače, zda chce uživatel obdržet textovou zprávu obsahující informace o registrovaném termínu, nebo zda chce uložit událost do kalendáře. Finálním krokem bude tlačítko pro rezervaci, které uživateli oznámí dokončení termínu.

3.4 Požadavky na funkce systému

Při analýze aplikace je vhodné sestavení diagramu případů užití, který zobrazuje chování systému z pohledu uživatele. Na obrázku [3.1] je případ užití systému virtuální čekárny.

Obrázek 3.1: Diagram případů užití

Každá činnost charakterizuje použití systému uživatelem. Komunikace může být znázorněna jako *Uživatel - Zobrazení podniků*. Ohraničení udává hranice systému virtuální čekárny.

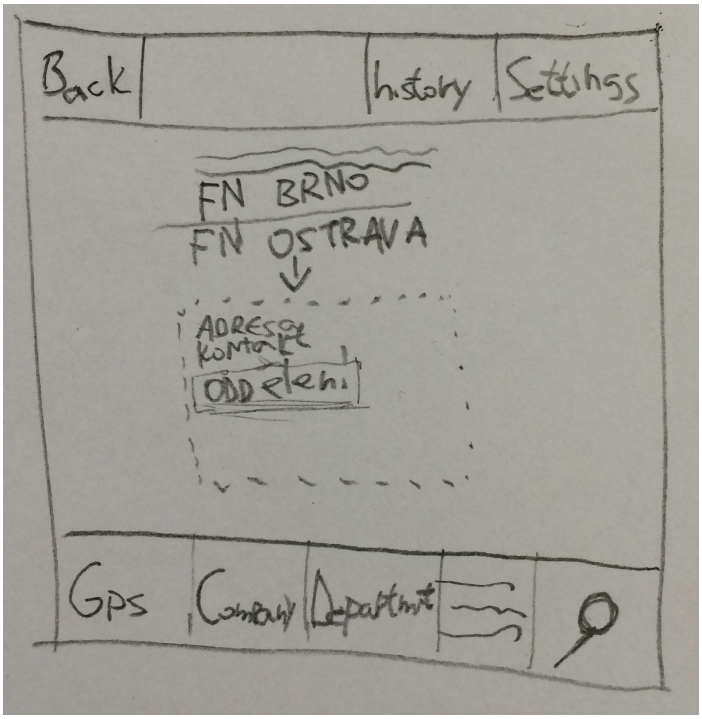
4 Implementace

V této kapitole následuje implementace jednotlivých částí programu. Popíši implementační prostředky, kterými jsem vývoj realizoval, a které byly využity. Poté je popsána realizace uživatelského rozhraní a zpracování dat ze serveru, která jsou následně rozparsována.

Jako jazyk zvolený pro vypracování této práce jsem si vybral Swift [4], protože jsem si chtěl vyzkoušet nový programovací jazyk. Dále jsem využil framework *Cocoa Touch* zmíněný v kapitole [2.4], který je nejvyšší vrstvou [2.2] v hierarchii a obsahuje nejdůležitější frameworky pro vývoj aplikací na iOS. Celá aplikace byla vyvinuta ve vývojovém prostředí *Xcode.*

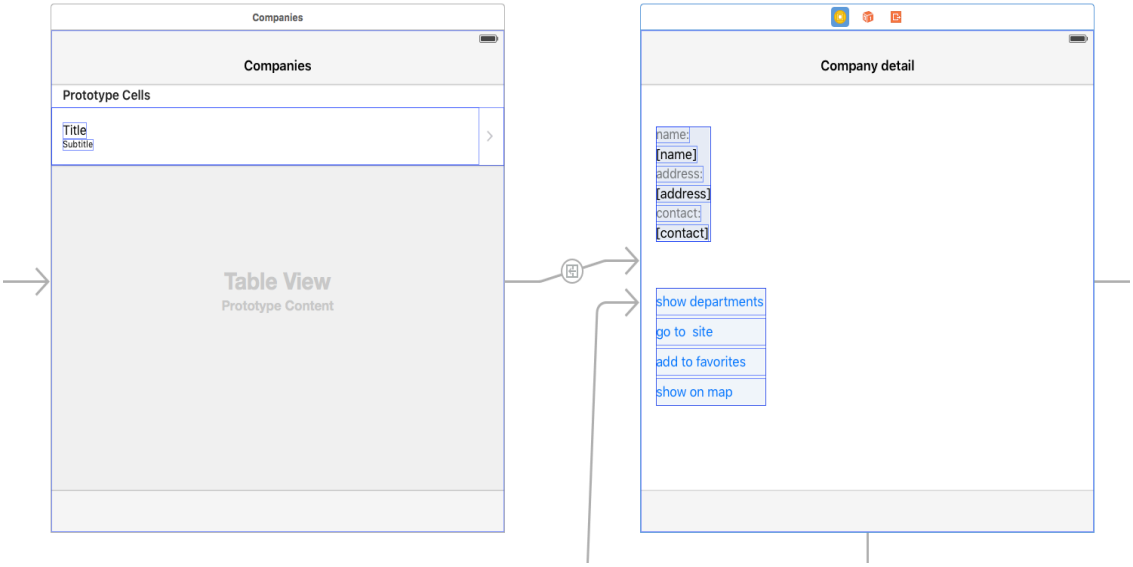
4.1 uživatelské rozhraní

Hlavní požadavek byl udělat grafické uživatelské rozhraní co nejjednodušší a aby se uživateli s aplikací dobře pracovalo. Jeden z prvních návrhů na obrázku [4.1] počítal s celkem zahuštěným spodním menu a návrh byl později upraven stylem redukce prvků, aby rozhraní nebylo zbytečně matoucí pro uživatele. Také se změnila obrazovka s detaily zobrazeného podniku, kdy se upustilo od více jmen podniků na jedné obrazovce a byla vyhrazena pro každý podnik obrazovka pouze s daným podnikem a informacemi relevantní k němu. Později byl vytvořen novější návrh rozhraní na obrázku [5.1], který nejvíce reflektuje vzhled aplikace i v elektronické podobě.



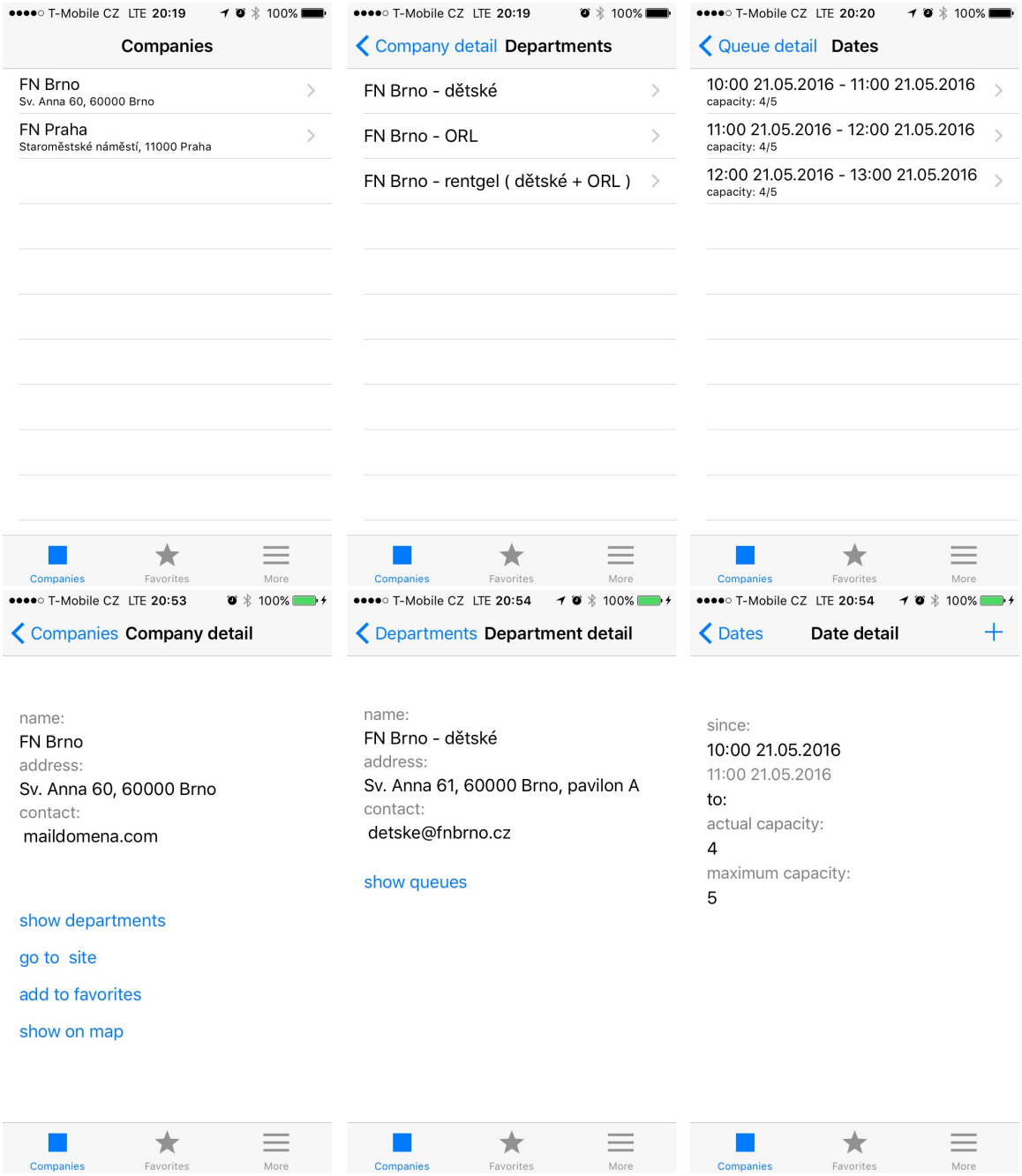
Obrázek 4.1: Návrh uživatelského rozhraní

Uživatelské rozhraní je vytvořeno ve vývojovém prostředí Xcode, které používá Storyboard. Ten je grafickým editorem a přetahují se zde komponenty jako tlačítka, obrazovky a další prvky na pracovní plochu, kde se z těchto komponent bude skládat výsledné grafické rozhraní. Storyboard označuje jednotlivé obrazovky jako *View* a přechody mezi těmito obrazovkami jako *Segue.* Všechny použité obrazovky aplikace, kromě té pro vyplnění formuláře používají společné *Tab Bar* menu. Jedná se o spodní přepínací menu, viditelnou také na obrázku výše [4.1], sloužící pro rychlejší orientaci mezi obrazovkami. Nahoře obrazovky lzde vidět navigační tlačítka, o které se stará *Navigation Controller,* díky kterému je možné se dostat jednoduše zpět. Jednoduchým se má na mysli, že jednotlivé přechody jsou ukládány na zásobník přechodů, a tím je možné jejich navrácení. V horní liště je možnost přidání popisku o aktuálně se nacházené obrazovce a další elementy, jako například tlačítka. Slovní popis výše a ukázka Storyboardu je na obrázku [4.2], kde je z obrázku patrné, že se nejedná o velikost displeje ani rozměry obrazovky displeje. Apple vytvořil *Auto layout*, díky kterému je možné vytvořit jednu aplikaci, která bude zobrazitelná na všech velikostech displejů stejně. K tomuto dosažení je nutné využít ještě *Constraints,* diký kterému lze definovat vzdálenost prvků od okrajů, a umístění se dopočítá automaticky v závislosti na velikosti displeje. *Constraints* jsou vyžadovány vývojovým prostředím Xcode a při každé manipulaci komponenty, která měla nastavené tyto *Constraints*, se musí znova aktualizovat. V Xcode má každá obrazovku svůj vlastní controller, které se odvozují od třídy *UIViewController* viditelný vpravo na obrázku [4.2], nebo *UITableViewController* viditelný vlevo na obrázku [4.2].



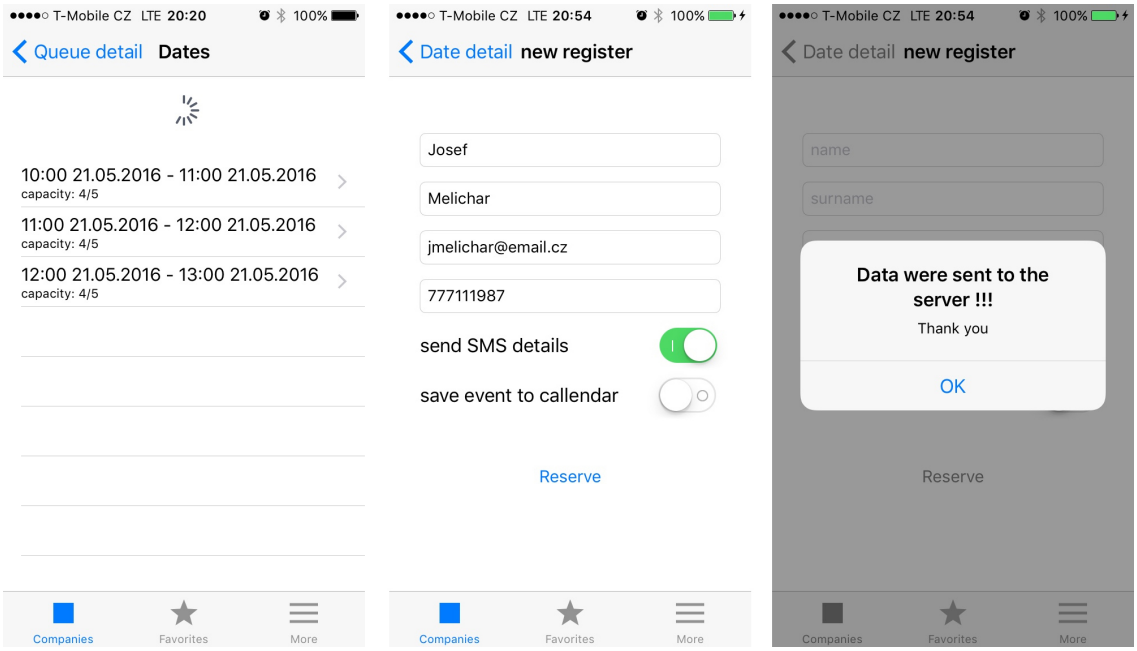
Obrázek 4.2: Ukázka Storyboardu v Xcode

- **Zobrazení dat v tabulce *UITableViewController***
  Při vytvoření tabulky *TableView* ve Storyboardu, musí být k ní vytvořen také příslušný controller. V tomto případě to bude *UITableViewController*, který obsahuje několik povinných metod pro správné zobrazení dat. Jednou z těchto metod je *numberOfRowsInTableView*, která vrací počet řádků tabulky. Například každý řádek značí jeden konkrétní podnik, který je uložen v poli podniků. Na obrázku [4.3] lze vidět za sebou několik příkladů, jak je využíván *UITableViewController* a *UIViewController*.



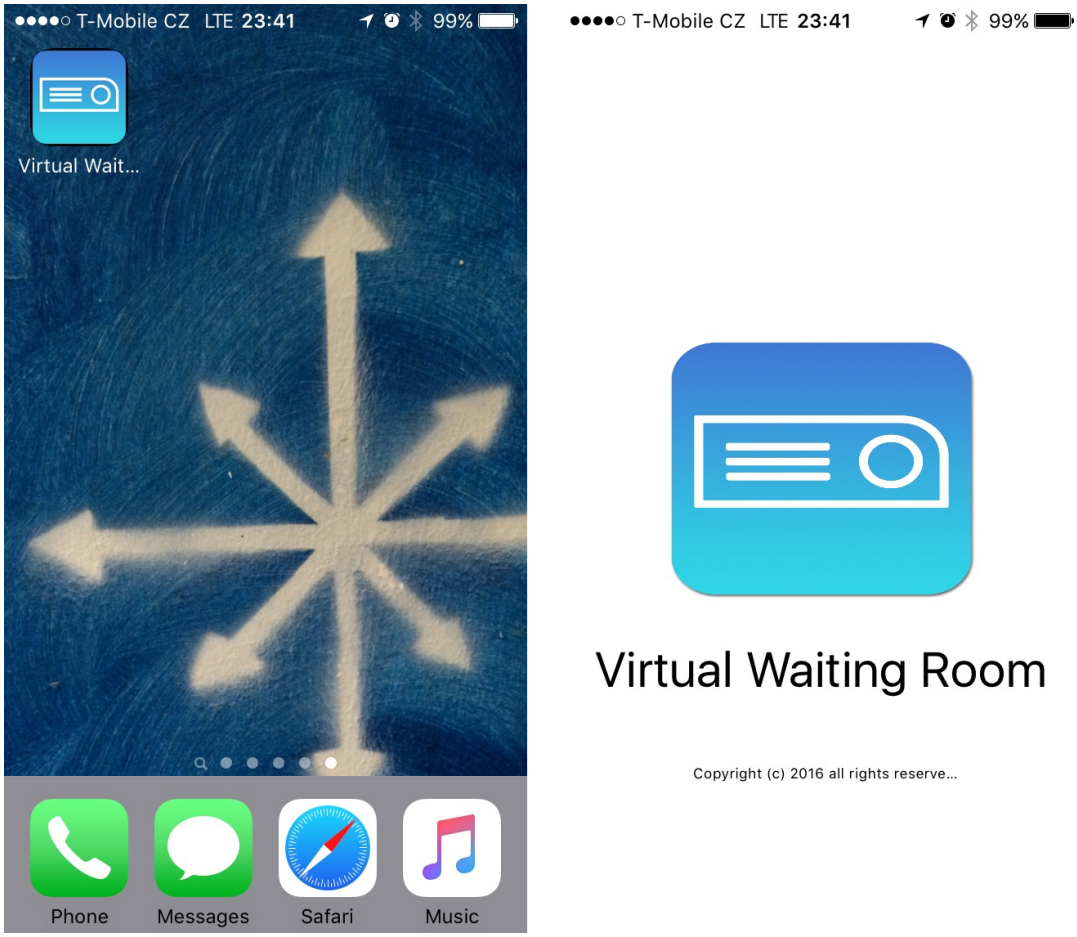
Obrázek 4.3: Ukázka obrazovek jdoucích za sebou při zobrazení detailu termínů

Obrázek znázorňující obrazovky aplikace, na které se lze dostat postupným pokusem o registraci na termín. Na obrazovce s termíny je možné aktualizace, zda se nezměnila kapacita pomocí přetažením prstu dolů a následného zvolení termínu. Tyto akce popisuje obrázek níže [4.4]



Obrázek 4.4: Kroky před registrací na termín

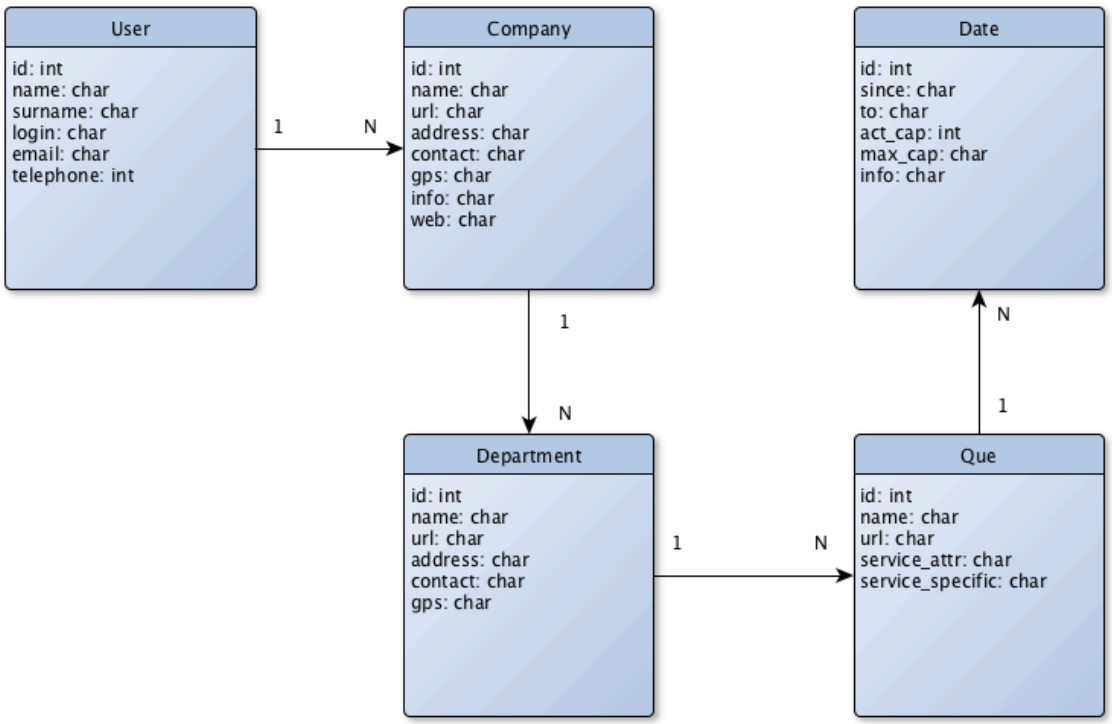
Data se na server prozatím nazasílají, neboť na něm není implementace pro *POST* požadavky, ale vše je v programu připraveno pro budoucí možnost serveru tyto požadavky přijímat.



Obrázek 4.5: Aplikace obsahující navrhnutou ikonku a spouštěcí obrazovku

4.2 Navržení hlavních komponent

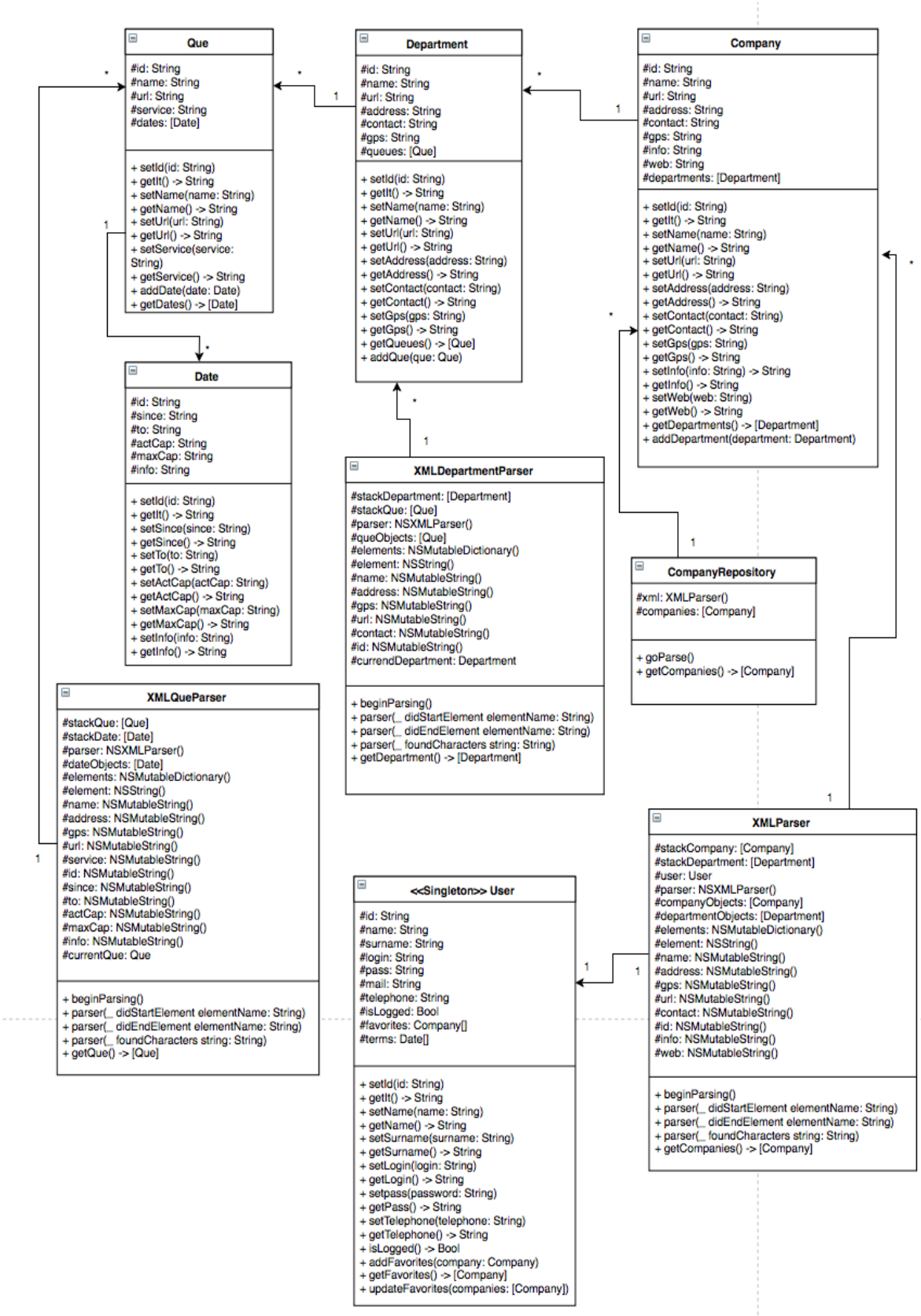
Zde prezentuji návrh datového modelu systému vycházejícího z diagramu případu užití [3.1]. Hlavní komponenty aplikace jsou podnik, oddělení, fronta, termín a uživatel. Jak lze vyčíst z diagramu, jednotlivé prvky se rekurzivně zanořují do sebe. Uživatel přihlášený v aplikaci prozatím nebude implementován úplně, neboť je potřeba spolupráce s databázovým serverem. Nyní simuluje funkcionalitu do budoucího rozšíření. Pro testovací účely bude uživatel implementován lokálně a v momentě, kdy bude server připraven pro práci s uživatelem a bude zprovozněna možnost přihlášení či odhlášení uživatele v aplikaci, tak bude funkcionalita zprovozněna malou úpravou repozitáře. Podrobněji popsáno obrázkem níže [4.5].



Obrázek 4.5: Datový model navržených komponent

4.3 Návrh struktury tříd

Návrh systému, který jsem implementoval jsem formalizoval pomocí diagramu tříd obrázkem [4.6]. Jelikož vývoj pod iOS používá architekturu MVC zmíněnou v kapitole 2.4, tak zde pro zjednodušení nejsou zahrnuty tabulky pro kontroléry. Jsou zde zahrnuty všechny metody a atributy, se kterými bude aplikace pracovat. Následuje popis některých tabulek, aby bylo pochopitelné, proč byly navrženy a k čemu slouží.



Obrázek 4.6: Diagram tříd aplikace

Třída User
Tabulka obsahuje v době používání aplikace vždy jednoho přihlášeného uživatele, o kterém uchovává informace důležité pro splnění registrace na vybraný termín či udržuje seznam oblíbených podniků.

Třída CompanyRepository
Zde jsou uloženy veškeré informace o tabulkách Company, Department, Que, Date. Vystačí zde pouze jedna tabulka způsobu použití *repository*, neboť tabulka Date nemůže existovat bez tabulky Que. Que nemůže existovat bez tabulky Department a stejným způsobem Department nemůže existovat bez tabulky Company. Z toho plyne vzájemné zanoření dat do sebe, kde hlavním prvkem je Company obsahující dílčí části. CompanyRepository v sobě volá metody pro parsování dat ze všech tabulek souvisejících s parsováním. Obrázek níže [4.7] popisuje implementaci v pseudokódu, kde se provádí nejdůležitější parsování a logika aplikace.

```
*** pseudo code ***

function returning companies with all dependency elements

for company in companies
    for department in company.getDepartments()
        start parsing data for department

        for que in department.getQueues()
        start parsing data for que and dates

return parsed data contains companies, departments, queues and dates

*** end of pseudo code ***
```

Obrázek 4.7: Pseudo kód zobrazující hlavní logiku aplikace

Třída Company
V této tabulce jsou uchovány informace o podniku. Podnik obsahuje položky jako jméno, kontakt, adresa a seznam oddělení. Pro naplnění těchto informací jsou připraveny metody jako *setName, setUrl, addDepartment.* Pro získání informací jsou metody jako *getDepartments, getInfo, getId*. Zajímavější metodou je zde přidání oddělení do podniku, kdy se ještě nepředávají všechny informace o oddělení, ale pouze URL adresa, na kterou je nutno se přesměrovat, a až poté naplnit specifičtější informace. Tabulky *Department, Que* a *Date* pracují na obdobném principu, a proto je zde pro lepší pochopení uvedena pouze tabulka *Company*.

Třída XMLParser
V tabulce se provádí parsování celku, který je následně rozložen na menší části. Menší části jsou naplněny do objektu, pomoci výše zmíněných metod. Parser rozlišuje dva případy.

- **Informace o podniku -** Jedná se o všechny potřebné informace pro kompletní naplnění podniku jako název, adresa, kontakt.

- **informace o odděleních -** Zde je pouze ID a URL adresa[2]. Tyto položky jsou vloženy do datového typu kolekce a následně přiřazeny ke konkrétnímu podniku. Kolekce z důvodu vztahu, kdy jeden podnik může obsahovat více oddělení a není předem znám počet oddělení.

Obdobným způsobem jsou implementovány další tabulky z této kategorie, ale pro zjednodušení je uvedena pouze tato tabulka.

Užité vzory

Aplikace používá dva návrhové vzory [9] a řídí se jejich postupy. Při použití návrhových vzorů je snadnější orientace v kódu a do budoucnosti a sníží se riziko chyb. Níže jsou popsány využité návrhové vzory.

- **Singleton -** Singleton neboli jedináček je třída, jejíž instance je vytvořena pouze jednou a poskytuje globální přístup k ní. V této práci se využívá vzoru pro aktuálního uživatele v aplikaci a jsou implementovány metody pro přístup k datům [4.1], které mohou vrátit či nastavit jméno uživatele, emailovou adresu nebo spravování oblíbených položek uživatele.
- **Repository pattern -** Tento návrhový vzor pomáhá k přehlednému ukládání dat a redukuje duplikaci kódu. Využit je ve třídě *CompanyRepository*, kde uchovává data o konkrétním podniku zahrnující jeho oddělení, fronty a termíny pro objednání.

4.4 Tvorba virtuální čekárny

Práce se skládá z více tématicky odlišných celků a aktuálně pracuje se serverem, který je naprogramován v jazyce PHP[3].

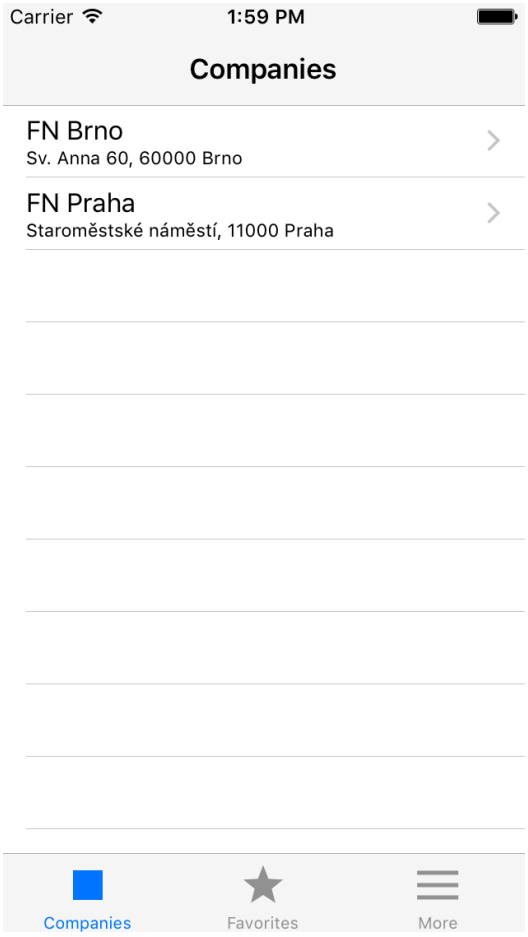
Mobilní aplikace virtuální čekárny využívá koncept objektově orientovaného programování a pracuje se vzorem MVC [6], který je zmíněn ve 2 kapitole. Díky využití MVC bude práce dobře udržovatelná do budoucna, na což se v dnešní době klade velký důraz. Aplikace se dnes rozšiřují ve větší míře a kvůli drobné funkcionalitě by bylo zbytečné předělávat celou aplikaci. Pokud se bude do budoucna změní například URL adresa, tak se hlavní logika změní pouze v oddělené části kontroléru. View a model zůstane naprosto beze změny a nemá tušení, že se něco změnilo.

Popis činností aplikace

Jako první se po zapnutí aplikace zobrazí seznam podniků, které obsahuje *CompanyTableViewControlller*. Ten v sobě vytvoří instanci *CompanyRepository* a zavolá její metody *goParse* a *getCompanies*. První metoda připraví všechna data a druhá metoda vrátí získané informace o společnostech do kolekce typu *Company*. Předané informace se zpracují pomocí připravených metod pro *tableView*, aby se správně zobrazily. Nyní se předaly informace z modelu do kontroléru, které lze vidět na obrázku [4.8] níže.

[2] Uniform Resource Locator - slouží k přesné specifikaci umístění dokumentu na internetu

[3] Skriptovací programovací jazyk



Obrázek 4.8: Seznam dostupných podniků

Za první zmíněnou metodou *goParse* v *CompanyRepository* se odehrává více volání, které budou nyní rozepsány podrobněji. Tato metoda volá třídu *XMLParser*, kterému se předá URL adresa obsahující všechny podniky. Parsování probíhá od začátku dokumentu směrem dolů a jsou zde připravené metody pro vytažení důležitých částí z dokumentu. Pro práci s těmito metodami musí *XMLParser* podědit *NSXMLParserDelegate* protokol. Tato práce využívá tři metody, které se přetěžují.

- *parser(_didStartElement elementName: String)*
- *parser(_didEndElement elementName: String)*
- *parser(_foundCharacters elementName: String)*

První metoda projede prvky na stránce, a pokud najde ty důležité, připraví si proměnné, do kterých se budou v dalších krocích ukládat data. V této metodě probíhá inicializace objektu *Company*, pokud se jedná o prvek značící podnik a následné uložení na zásobník podniků. V druhém případě se jedná o prvek oddělení, kde se vytvoří objekt *Department* a následně je vložen na zásobník oddělení.

Druhá metoda se podívá na prvek stránky, a pokud se jedná o informace o podniku, vytáhne ze zásobníku podniků poslední položku a naplní předem připravené proměnné hodnotami. Po naplnění se objekt opět přidá na zásobník. V druhém případě se jednalo o prvek oddělení a vytáhne se ze zásobníku oddělení poslední položka a opět se naplní proměnné. Po naplnění je objekt vložen zpět na zásobník oddělení a navíc se na zásobník podniků na poslední prvek přiřadí informace o naplněném oddělení a zajistí se, že každé oddělení má svůj konkrétní podnik.

Poslední metoda zajistí, že hledaný prvek bude k dispozici v podobě proměnné pro práci s metodami zmíněnými výše.

Na podobném principu fungují i ostatní třídy parserů, kdy se opět využívá ukládání a předávání dat pomocí zásobníku.

Stahování dat

Existují dvě různá řešení, která se nabízejí použít a budou rozepsány níže. Nakonec bude vybráno a zdůvodněno jedno z nich.

- **Stáhnutí všech dat najednou**

  První řešení si stáhne informace o podniku, kde je známo jméno, adresa, kontakt a další položky. K oddělení existuje pouze identifikátor a odkaz na URL adresu, přes kterou lze další informace doplnit. Adresa oddělení obsahuje položky jako jméno, emailovou adresu, kontakt a obsahuje informace o frontě v podobě identifikátoru a odkazu.

  Z toho je patrné, že se položky do sebe zanořují a vždy uchovávají informace o aktuálně zpracovávaném objektu a podrobnější informace dalšího objektu jsou vždy v dalším odkazu. Tato práce řeší stahování dat najednou ve třídě *CompanyRepository* a postup je zobrazen na obrázku [4.7], který popisuje pseudo kód hlavní logiky. Data se v průběhu používání aplikace již stahovat nemusí.

- **Stáhnutí dat při navigaci na zvolenou obrazovku**

  Druhým způsobem je stahování dat při přechodu na konkrétní obrazovku, kdy stažení informací může mít prodlevu a projeví se to na straně uživatele. Při zobrazení seznamu podniků aplikace nebude mít další informace o seznamu obsahující oddělení, ale při přechodu na seznam oddělení se nechají stáhnout potřebná data.

Tato práce využívá prvního řešení z důvodu ne velkého počtu podniků, a proto se může dovolit stáhnout vše najednou, aby uživatel nepoznal v průběhu používání žádná zpomalení aplikace při načítání dat. Testováno bylo několik podniků, avšak pro hlubší testování je třeba mít podniků několik desítek a různé rychlosti mobilního připojení. Poté kombinovat tyto hodnoty. Druhé řešení se nabízí v případě počtu podniků kolem jednoho sta, kdy by bylo stahování tak velkého počtu podniků neefektivní a stahovaly by se pouze informace zajímavé pro uživatele.

5 Testování

V této kapitole jsou shrnuty metody a postupy často užívané a osvědčené pro testování mobilních aplikací. Jedná se i o postupy, kterými byla aplikace testována a následně jsou uvedeny i zjištěné výsledky.

5.1 Testování grafického uživatelského rozhraní

Testování grafického uživatelského rozhraní je velmi důležitou součástí při vývoji aplikací. Bez testování by zákazník zadal vizi o požadovaném produktu a vývojáři by jej stvořili. Následně by byl zhotovený produkt představen, avšak ne vždy se podaří reflektovat představu zákazníka a v případě nepochopení obou stran, by musel být produkt předělán.

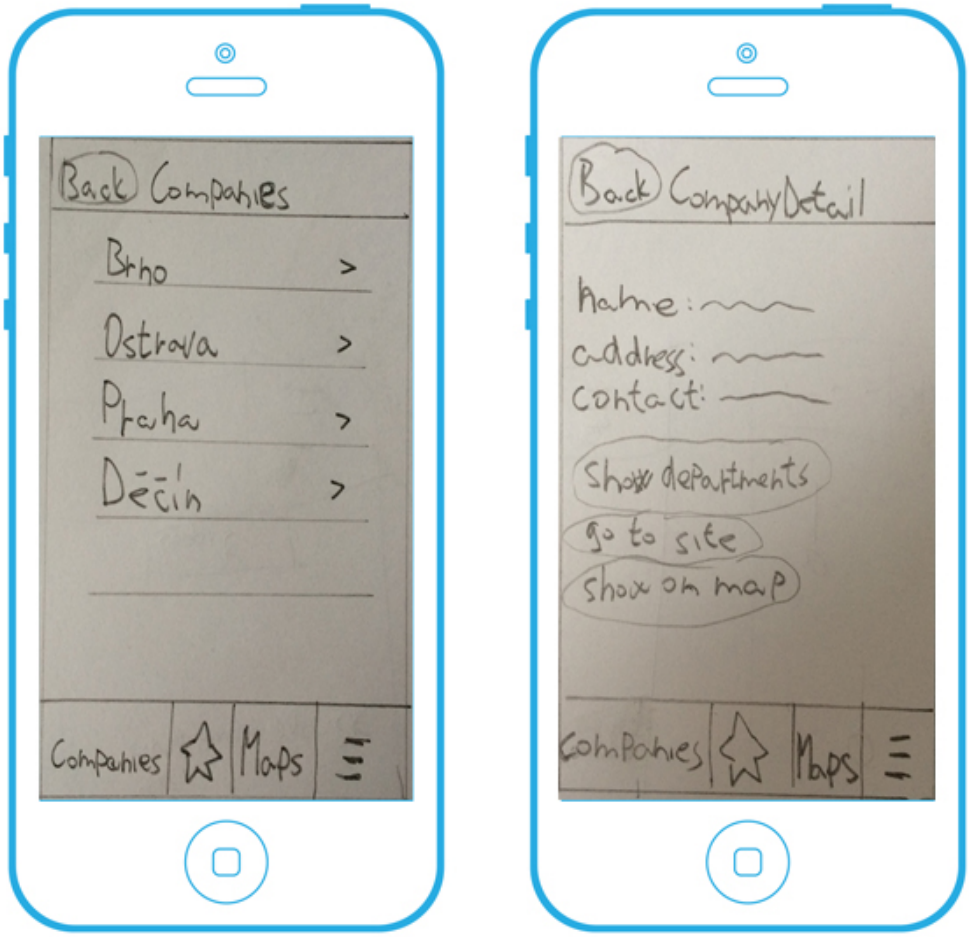
Předělání již finálního produktu je většinou velmi náročné jak časově, tak finančně. Naopak předělání návrhu není nic složitého, neboť se prozatím nejedná o funkční výrobek. Vyplatí se tedy věnovat čas a udělat si představu o tom, jak by měla aplikace vypadat, zda je pro uživatele použitelná a pokud se bude snadno ovládat. Návrhy můžeme rozdělit do tří skupin.

Náčrt na papír
Jedná se o rychlou variantu návrhu, kdy nám postačí papír a tužka. Nezachází se do detailů a počítá se s tím, že se jedná o pracovní verzi. Z nakresleného náčrtku se můžeme dotázat uživatelů, zda rozumí jednotlivým prvkům. Náčrtek také nemusí odrážet budoucí podobu aplikace.

Wireframe
Přezdívaný také jako "drátěný model" je vhodný pro náhled nového řešení, kde návrh definuje funkci a obsah aplikace pro lepší pochopení uživatelem. Výhodou je možnost prezentace tohoto návrhu elektronicky na displejích a oproti náčrtkům jsou úpravy velmi snadné.

Prototyp aplikace
Tato část se nejvíce podobá výslednému produktu. Prototyp nabízí možnost kliknutí na jednotlivé komponenty, přecházení mezi různými obrazovkami aplikace, animace přechodů a dalších prvků zaměřených na uživatelské rozhraní. Testování prototypu uživatelem dle předem daných úkolů je cenným přínosem. Vývojář aplikaci vidí jednoduše a orientace v ní je podle něj jednoznačná, protože aplikaci vyvinul a přesně ví, kde se jaké prvky nachází. Pro uživatele je první seznámení s aplikací zcela nové a musí sám zjistit, přes které komponenty se dostane na požadovaný úkol. Sledování uživatele v průběhu používání aplikace je velmi přínosné a vývojář by si měl zaznamenat odpovědi testovaného. Vývojář by si měl také všímat řeči těla, která může prozradit více, než slovní popis.

Testování je vhodné provést s cílovou skupinou budoucích uživatelů, pokud se jedná o aplikaci specifičtějšího charakteru. Aplikaci určenou pro odborníky některého z oboru by nebylo žádoucí testovat s jakýmkoliv uživatelem. Avšak aplikaci více obecnějšího charakteru je vhodné testovat větším okruhem náhodných uživatelů.



Obrázek 5.1: Návrh aplikace

5.2 Průzkum návrhu testování

V této podkapitole se jako první prování uživatelský průzkum rozhraní, kde jsou uživatelům předloženy úkoly, ze kterých se zjišťuje, jak na uživatele působí design aplikace. Poté se v této kapitole nachází moderovaný test. Zde si uživatelé vyzkouší práci s aplikací pomoci zadaných úkolů.

Návrh rozhraní s uživatelem

Postupy níže byly čerpány z knihy [8] zabývající se designem .

V určité části vývoje budeme mít několik prototypů aplikace, které budou reprezentovat potenciální vzhled výsledné aplikace. Při průzkumu je vhodné mít dva nebo více návrhů budoucího vzhledu aplikace, které nechám otestovat uživateli. Pouze s jedním návrhem je zřejmé, že na aplikaci bude od uživatelů kladný posudek, protože lidé nechtějí znít příliš kriticky k prototypům, a tím zranit návrháře či designéra. S více než dvěma návrhy se budou uživatelé cítit pohodlněji sdělit kritiku, neboť se budou více soustředit na porovnávání designu, než jeho přímé kritizování.

Uživatelům testujícím design byl předložen každý návrh jednotlivě pomocí displeje a následně byly položeny otázky. Příklad otázek byl položen stylem, "*podívejte se na každý návrh po dobu jedné minuty a následně vyberte alespoň tři možnosti ze seznamu, které nejlépe popisují grafický návrh.*" Uživatelé měli označit jejich volbu na seznamu obsahujícím několik položek. Možnosti, které se vyskytly na seznamu lze vidět níže.

- moderní
- konzervativní
- nudné
- více grafických prvků
- uživatelský přívětivé
- neuspořádané
- matoucí
- příliš mnoho prvků
- nevyhovující styl písma
- přehledné
- jednoduché
- čisté

Dalším příkladem bylo shromažďování dojmu na design pomoci otevřených otázek. Uživatelé dostali dotazník s otázkami, pod kterými měli prostor pro svou odpověď. Otázky byly kladeny ve formě na seznamu níže.

- animace při přechodu na následující obrazovku nejsou rušivé
- vždy je k dispozici informace o aktuálně prohlížené obrazovce
- souhlas s požadovanými informacemi v položeném formuláři

Testování použitelnosti aplikace

Pokud je cílem porozumět a zlepšit schopnost uživatele úspěšně dokončit klíčové akce v aplikaci, je nutné se zaměřit na usability testing[4] [10]. Akcemi je myšleno zadat uživateli úkoly, jako *přidat podnik do oblíbených*, *zobrazit oblíbené podniky*, *zaregistrovat se na termín* a jiné. Usability testing je jednou z

[4] Testování použitelnosti - v oboru se využívá anglický termín a bude využit také v této práci. Technika zaměřená na uživatele interagující s aplikací pro zhodnocení produktu.

nejčastěji používaných metod, co se týče testování uživatelských zkušeností[5]. Koncept této metody je jednoduchý a může být popsán těmito kroky.

- **vytvořit sadu úloh pro aplikaci** - příprava detailních scénářů navržených pro konkrétní aplikaci.

- **výběr cílové skupiny uživatelů** - následné požádání o pokus úlohy splnit

- **realizace testování** - vybrané skupině uživatelů se předloží mobilní aplikace se sadou úloh, které by měli splnit. Při plnění sady úloh budeme uživatele důkladně pozorovat a zaznamenávat jejich reakce.

- **analýza dat a vyhodnocení** - poznamenat si, kde měli uživatelé problém a kde naopak byli úspěšní. Bude potřeba analyzovat získaná data a opravit zjištěné nedostatky.

Moderovaný test

Test se skládá ze tří hlavních částí, které budou podrobněji rozepsány níže[5].

Screener
Před testováním bylo potřeba zvolit budoucí reálnou skupinu uživatelů, kteří budou aplikaci používat. Z podkapitoly 3.1 byla definována cílová skupina uživatelů a nyní bude potřeba tento výběr vhodně vyfiltrovat pomocí takzvaného screeneru[6]. Zhotovený screener, který byl poskytnut testovaným uživatelům je zahrnut v příloze [A]. Uživatelé, jejichž odpověď bude stejná s předpokládanou odpovědí, mohou postoupit k dalšímu dotazníku, neboť spadají do skupiny cílové skupiny. Podmínkou screeneru byla platforma operačního systému iOS, protože aplikace je určena právě na ní. Bylo vhodné nesdělit uživatelům předem, co se testuje, neboť by mohli odpovědět podle sebe a rozhodnout se, zda chtějí být další součástí testování či nikoli. Příklad některých otázek položených ve screeneru lze vidět na seznamu níže.

- Používám chytré zařízení běžící pod systémem iOS ?
- Používám toto zařízení pouze pro telefonování a posílání textových zpráv ?

Pre-test dotazník
Po následném vyfiltrování uživatelů vhodných pro další testování byl připraven takzvaný pre-test dotazník. Dotazník sloužil jako seznámení se s uživatelem. Tímto dotazníkem se získaly obecné informace o zkušenostech testovaného a případného odhalení problémů, pokud uživatel nespadal do definované cílové skupiny. Jako moderátor testu bylo nutné uživatelům sdělit průběh testu. Uživatelé dostanou úkoly, které by měli vypracovat samostatně. Já jako moderátor jsem byl při průběhu testu k dispozici, ale můj zásah do testování byl pouze v nezbytně nutných případech, kdy si uživatel nevěděl rady apod. Při testování bylo vhodné zvolit klidné prostředí a sdělit uživateli, že se nebude testovat uživatel, ale samotná aplikace. Tímto se zamezila možnost zklamání uživatele při nesplnění některé z

[5] UX neboli user experience

[6] Screener neboli první dotazník pro testované uživatele

úloh, protože zklamat mohla pouze aplikace z důvodu možného špatného navržení. Otázky položeny na pre-testu lze vidět na seznamu níže.

- Jak se cítíte před testem ?
- Máte nějaké zkušenosti s testováním a máte zájem o testování aplikace ?
- Těšíte se na samotné testování ?

Post-test dotazník
Jako poslední dotazník byl uživatelům představen post-test dotazník pro vyplnění. Testování se nejčastěji nahrává na mikrofon či video záznam, avšak v tomto případě byly vytvořeny poznámky. Ověřoval se vliv používání aplikace na uživatelích během testu. Otázky byly následující a jsou viditelné na seznamu níže.

- Připadalo Vám uživatelské rozhraní intuitivní ?
- Myslíte si, že zadané úkoly byly jednoduché na vypracování ?
- Přemýšlíte o budoucím využití aplikace ?
- Jak se Vám s aplikací pracovalo ?

5.3 Vyhodnocení testování

Tato podkapitola popisuje výsledky naměřené při testování a reaguje na některé poznatky uživatelů při plnění úkolů přiložených v příloze [B], díky kterým se aplikace posunula o něco kupředu.

Vyhodnocení návrhu rozhraní s uživatelem
Při testování návrhu rozhraní s uživatelem v kapitole 5.2 proběhlo testování s 6 uživateli, kde jejich nejčastější odpovědi shrnuje tabulka níže.

možnosti	přehledné	jednoduché	uživatelsky přívětivé	více grafických prvků
počet výskytů	4	3	5	3

Tabulka 5.1: Nejčastěji volené možnosti uživateli

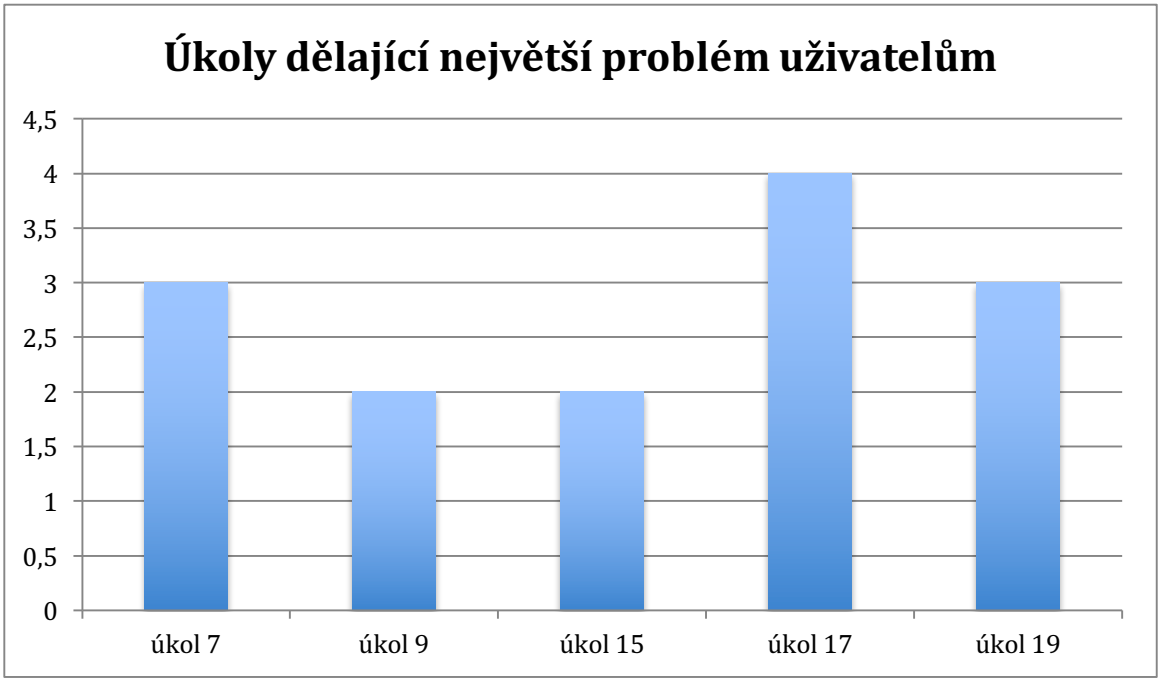
Z těchto hodnot plyne, že rozhraní aplikace je použitelné a srozumitelné. S odpověďmi ohledně více grafických prvků byly přidány u každého seznamu indikátory, že se lze po kliknuti na položku přesměrovat na její detail. Následně ve spodním tab menu byly přidány ikonky. Další zásah co se týče přidání grafických prvků by mohl narušit ostatní odpovědi uživatelů, kde mnohem více než zvyklost grafických prvků je důležitější jednoduchost aplikace.

Jako další část ve stejné kapitole bylo zjišťování dojmu uživatelů na design ve formě otevřených otázek. Testovaní uživatelé odpověděli kladně na všechny tři otázky i v případě souhlasu s požadovanými informacemi, což mě mile překvapilo, neboť nikdo neměl problém se zadáním telefonního čísla.

Vyhodnocení moderovaného testu
Pomocí moderovaného testu v kapitole 5.2 proběhlo komplexnější testování aplikace, kde si testovaní uživatele vyzkoušeli práci s aplikací. Pomocí screeneru byli vybráni uživatelé používající platformu iOS,

se kterou mají zkušenosti a orientují se v ní. Pre-test dotazník odhalil, že nikdo z uživatelů nemá zkušenost s přímým testováním. Uživatelé se tak těšili vyzkoušet si něco nového.

Vypracování připravených úkolů přiložených v příloze [B] lze demonstrovat grafem níže.



Graf 1 - Otázky, při kterých se uživatelé nejčastěji zasekli

Z grafu výše bylo zjištěno několik poznatků, které jsem si při vývoji neuvědomil a pomohly tím aplikaci posunout o něco kupředu. Níže budou rozepsány úkoly, při kterých se uživatelé nejvíce zdrželi.

- **Úkol 7 - přidejte podnik do oblíbených podniků**
  Někteří uživatelé po kliknutí na přidání podniku do oblíbených nedostali žádnou zprávu o tom, zda se přidání podařilo či nikoliv. Na moment se zde zasekli, chvíli čekali a poté kliknuli na přidání podniku do oblíbených znovu.

  Řešení
  Přidána vyskakující hláška o tom, že byl podnik přidán do oblíbených. Při vybrání podniku, který již v oblíbených existuje vyskočí hláška, že se tam už podnik nachází

- **Úkol 9 - smažte podnik z oblíbených položek**
  Pár uživatelů chvíli váhalo, jak smazat podnik z oblíbených položek. Podle zvyklostí iOS se mazání položek provádí pomoci přetáhnutí prstu z pravého konce. Tohle ale zřejmě nevěděli všichni. Po chvíli hraní s aplikací se jim podnik podařilo odstranit z oblíbených.

  Řešení
  Bylo přidáno tlačítko pro editaci oblíbených podniků, pro lepší orientaci v aplikaci

- **Úkol 15 - vyberte jinou frontu**
  Pokud oddělení mělo pouze jednu frontu, nebyla možnost vybrat frontu jinou. Toto nepovažuji za chybu, neboť tato situace může nastat a je to korektní. Některé uživatele to ale zmátlo.

- **Úkol 17 - aktualizujte terminy**
  V tomto úkolu se někteří uživatelé zbrzdili nad tím, jak aktualizaci provést. Není k dispozici žádné tlačítko, pouze seznam podniků. Zde jsem v jednom případě pomohl uživateli vyřešit tento úkol nepřímo, kdy jsem se ho zeptal, zda ví, jak aktualizovat mailovou schránku. Po tomto zásahu si uživatel vzpomněl, že se obrazovky v systému iOS aktualizuji pomocí přetažením prstu z horní části obrazovky směrem dolů.

  Řešení
  Zde by bylo možným řešením přidání tlačítka značící aktualizaci podniků, avšak jsem jej neimplementoval kvůli zvyklostem systému iOS. Do budoucna by bylo vhodné vypozorovat, zda s tím mají problém i ostatní uživatelé. V tomto případě by bylo tlačítko pro aktualizaci na místě.

- **Úkol 19 - zaregistrujte se na termín**
  V tomto úkolu si někteří uživatelé nebyli jistí tlačítkem, které slouží pro přesměrování na registrační formulář. Uživatelé váhali kliknout na toto tlačítko. Po zjištění, že zde není žádné další tlačítko na něj kliknuli.

  Řešení
  Pro jednoznačné určení, že se jedná o tlačítko pro přidání se na termín jsem zvolil styl tlačítka vypadajícího jako plus (+).

Celkem testování dopadlo pozitivně, kde uživatelé do post-dotazníku zodpověděli kladně. Podle uživatelů byly úkoly jednoduché na vypracování a pouze v některých případech jim to zabralo o něco déle úkol dokončit, neboť to bylo jejich první seznámení s aplikací. Práce s aplikací byla zhodnocena jako jednoduchá bez zbytečných rušivých prvků, což mě potěšilo a potvrdilo, že se vyplatí držet zásad při návrhu grafického uživatelského rozhraní zmíněných v kapitole 2.2.

6 Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat mobilní aplikaci, která bude umožňovat objednání uživatelů na určitý čas a přiřadit je do fronty ve virtuální čekárně, kde není potřeba být fyzicky. Tento cíl byl úspěšně splněn.

K dosažení výsledků bylo potřeba nastudovat a provést průzkum existujících řešení rezervačních systémů, kde na základě provedeného výzkumu a vyzkoušení několika v té době dostupných aplikací, bylo navrhnuto nové řešení. Společnou nevýhodou těchto aplikací je používání mnoho grafických prvků, které působí rušivě. Poté byla vypracována analýza požadavků na aplikaci a definice cílové skupiny. Z toho byly navrženy vlastnosti a funkce budoucí aplikace Virtuální čekárny. Pro následné zhotovení mobilního programu byla potřeba nastudovat pravidla pro tvorbu uživatelského rozhraní na platformě iOS, pochopit využívané vzory a funkce v systému. To vedlo k navržení uživatelského rozhraní splňující tyto pravidla a držení se zvyků platformy iOS. Při testování se práce s aplikací promítla na uživatelích pozitivně, neboť byli zvoleni uživatelé používající stejnou platformu a práce s programem jim nedělala žádné velké potíže. Přesně na to byl celou dobu kladen důraz, aby aplikace byla intuitivní a měla jednoduchý design. Tyto požadavky byly hlavním předmětem testování na uživatelích, které dopadlo kladně.

Všechny části programu byly implementovány v jazyce Swift a výsledná práce uživateli nabízí zobrazení seznamu podniků, přes které se lze dostat na jednotlivé obrazovky oddělení, front a termínů. Uživatel má možnost přidat podnik do oblíbených a registraci na termín a zobrazení registrovaných termínů. To vše s minimalistickým designem.

Vypracování tohoto programu bylo zajímavým projektem, který mne přiblížil k rezervačním systémům a s platformou iOS, které bych se chtěl dále věnovat. V průběhu vypracování tvorby jsem si vyzkoušel veškerý proces vývoje softwarového programu se vším, co obnáší. Otevřelo mi to oči při testování s uživateli a změnilo to můj postoj v průběhu vývoje k pozitivnímu.

Do budoucna by se dalo v práci pokračovat a bylo by zajímavé propojení aplikace s HealthKitem, které uchovává zdravotní data. Bylo by dobré mít v nabídce více podniků, a poté by bylo vhodné implementovat jejich vyhledávání.

Literatura

[1] Paul Hegarty: *Developing iOS 8 Apps with Swift* [iTunes] Apple Inc. [online]. 2015 [cit. 2015-04-10]. Dostupné z: https://itunes.apple.com/us/course/developing-ios-8-apps-swift/id961180099

[2] Apple Inc.: *IOS Human Interface Guidelines* [online]. 2015 [cit. 2015-04-17]. Dostupné z: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG

[3] Apple Inc.: *About the iOS Technologies* [online]. 2015 [cit. 2015-04-22]. Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html.

[4] Apple Inc.: *The Swift Programming Language* [online]. 2015 [cit. 2015-03-28]. Dostupné z: https://itunes.apple.com/us/book/swift-programming-language/id881256329?mt=11.

[5] KUNIAVSKY, Mike., Elizabeth GOODMAN a Andrea. MOED. *Observing the user experience: a practitioner's guide to user research.* 2nd ed. Morgan Kaufmann, Boston, c2012. ISBN 9780123848697.

[6] Paul Hegarty: *Developing iOS 7 Apps,* [Youtube kanál], Stanford University, California, 2015 [cit. 2015-04-25]. Dostupné z: https://www.youtube.com/playlist?list=PLzvKpLTmBmzkeSIxxaswFqMhL3j_rSBtp.

[7] RICHARDSON, Leonard a Sam. RUBY. *RESTful web services*. Farnham: O'Reilly, 2007. ISBN 0596529260.

[8] UNGER, Russ. a Carolyn. CHANDLER. *A project guide to UX design: for user experience designers in the field or in the making*. Berkeley, CA: New Riders, c2009. ISBN 0321607376.

[9] ERICH, Gamma. *Design patterns: elements of reusable object-oriented software*. 2., rev. ed. S.l.: Pearson Educ, 2000. ISBN 9780201485370.

[10] KAIKKONEN, Anne, KEKÄLÄINEN, Aki, CANKAR, Mihael, KALLIO, Titti a KANKAINEN Anu. 2005. *Usability testing of mobile applications: a comparison between laboratory and field testing. J. Usability Studies,* ročník 1, č. 1 (Listopad 2005), s. 4-16, [online]. 2015 [cit. 2015-05-12]. ISSN 1931-3357. Dostupné z: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=4384163

[11] *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2015-05-10]. Dostupné z: https://cs.wikipedia.org/wiki/Vyvolávac%C3%AD_systém.

[12] Apple Inc.: *iOS SDK Release Notes for iOS 9* [online]. 2015 [cit. 2015-05-11]. Dostupné z: https://developer.apple.com/library/ios/releasenotes/General/RN-iOSSDK-9.0/.

[13] GLINZ, Martin. *15th IEEE International Requirements Engineering Conference*, Delhi, 2007, Pp 21-26 [Online]. 2007 [cit. 2015-05-17]. ISSN 1090-705X. Dostupné z: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=4384163

[14] EDWARDS, Jim. *IPhone lost market share to Android in every major market except one* [online]. 2016 [cit. 2016-05-17]. Dostupné z: http://uk.businessinsider.com/apple-ios-v- android-market-share-2016-1

[15] *Smartphone Users Worldwide Will Total 1.75 Billion in 2014* [online]. 2014 [cit. 2016-05-17]. Dostupné z: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

Příloha A

Screener pro mobilní aplikaci virtuální čekárna

1, Vyberte operační systém Vašeho chytrého zařízení
- Android
- iOS
- Windows Phone
- Jiné

2, Používáte toto zařízení po dobu alespoň 1 roku ?
- Ano
- Ne

3, Využíváte toto zařízení s internetem ?
- Ano
- Ne

4, Převážně používám zařízení pro zasílání textových zpráv a telefonování
- Ano
- Ne

Příloha B

Sada úloh pro uživatele

1. Spusťte aplikaci Virtuální čekárna
2. Seznamte se s uživatelským rozhraním
3. Zobrazte více informací o některém podniku
4. Zobrazte podnik na mapě a poté se vraťte zpět
5. Přejděte na internetové stránky podniku a poté se vraťte zpět
6. Zkontrolujte, že v oblíbených položkách není žádný podnik
7. Přidejte podnik do oblíbených podniků
8. Ověřte, že se podnik nachází v oblíbených
9. Smažte podnik z oblíbených položek na informace o podniku
10. Zobrazte Oddělení podniku
11. Vyberte oddělení
12. Zobrazte konkrétní fronty z oddělení
13. Vyberte frontu
14. Přejděte zpět na seznam front

15. Vyberte jinou frontu
16. Zobrazte volné termíny z fronty
17. Aktualizujte termíny
18. Zvolte některý termín
19. Zaregistrujte se na termín
20. Odešlete formulář
21. Zjistěte aktuálně přihlášeného uživatele
22. Ukončete aplikaci