Nativní iOS aplikace pro objednávání v restauraci

Jakub Josef Forman

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub Josef Forman**
Osobní číslo:	**A16095**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**prezenční**
Téma práce:	**Nativní iOS aplikace pro objednávání v restauraci**
Téma anglicky:	**A Native iOS App for Ordering in a Restaurant**

Zásady pro vypracování:

1. **Stručně popište proces vývoje mobilní aplikace v nativním kódu Swift pro platformu iOS. Věnujte se také popisu vývojářských nástrojů.**
2. **Seznamte se také s technologiemi vhodnými pro vývoj serverového pozadí mobilní aplikace, jako je např. PHP framework Lumen či Firebase.**
3. **V rámci praktické části shrňte funkční a nefunkční požadavky na aplikaci pro objednávání jídel či nápojů v restauraci.**
4. **Navrhněte wireframe obrazovek a uživatelské rozhraní reflektující výše uvedené požadavky.**
5. **Implementujte mobilní aplikaci v jazyce Swift pro platformu iOS a nasaďte a otestujte běh na reálném zařízení.**
6. **V textu praktické části shrňte nejdůležitější části implementace.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **GHAREEB, H. Application Development with Swift. Packt Publishing, 2015. ISBN 9781785282362.**
2. **MANNING, Jon, Paris BUTTFIELD-ADDISON a Tim NUGENT. Swift development with Cocoa. Sebastopol, CA: O'Reilly Media, 2014. ISBN 1491908947.**
3. **FEILER, Jesse. Exploring Swift Playgrounds: the fastest and most effective way to learn to code and to teach others to use your code. Berkeley, California?: Apress, 2017. ISBN 1484226461.**
4. **LEWIS, Rory. IPhone and iPad apps for absolute beginners. IOS 5 ed. New York: Distributed to the Book trade worldwide by Springer-Verlag New York, c2012. ISBN 1430236027.**
5. **DERICO, Steve. Introducing iOS 8. Sebastopol, CA: O'Reilly, 2014. ISBN 1491908610.**
6. **DIPPERY, Michael. Professional Swift. Indianapolis, Indiana: John Wiley, [2015]. Wrox professional guides. ISBN 1119016770**

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 14. května

Jakub Josef Forman, v. r.
podpis diplomanta

ABSTRAKT

Cílem této bakalářské práce je navrhnout a částečně realizovat aplikaci pro objednávání v restauracích. Aplikace je primárně určena pro zařízení iPhone a iPod Touch. Obsahem této práce je návrh a částečná realizace aplikace včetně vytvoření vlastní API, databáze a grafiky. V rámci práce byl vytvořen wireframe aplikace, sepsány funkční a nefunkční požadavky a následně implementován prototyp aplikace. Pro prakticky realizovanou databázi a backend rozhraní byl naprogramován funkční prototyp na platformě iOS využívající nativní funkce a pomocných frameworků.

Klíčová slova: iOS, aplikace, Swift, iPod, iPhone, restaurace, Xcode, AppCode, Lumen, Adobe XD, Firebase, PHP, Postman

ABSTRACT

The goal of this bechelor thesis is to desing and in part develop mobile application for orders in restaurant. Application is intended for iPhone and iPod Touch devices. Content of bechelor thesis is design and in part realization of the application also realization own backend API, database and graphics. Wireframe, function and non-function requiments and also prototype of the application were created as a part of this thesis. For practically implemented database and backend was created working prototype on iOS platform using native iOS functions and auxiliary frameworks.

Keywords: iOS, aplikace, Swift, iPod, iPhone, restaurant, Xcode, AppCode, Lumen, Adobe XD, Firebase, PHP, Postman

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Radku Valovi Ph.D., za odborné vedení a skvělé rady, které mi během vypracování poskytl a také za vyřešení problémů, které bych sám, řešil velice dlouho. Dále mému kamarádovi Adamu Ulrichovi za tipy spojené s návrhem databázové struktury. A také všem ostatním, kteří aplikaci testovali, nebo se jinak podíleli na samotném vývoji.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **9**
I TEORETICKÁ ČÁST **10**
1 MOBILNÍ APLIKACE **11**
1.1 MULTIPLATFORMNÍ VÝVOJ 12
1.2 PLATFORMA IOS 12
1.2.1 Vývoj na platformu iOS 13
1.2.2 Programovací jazyk Swift 14
1.2.3 Cocoapods 15
2 STRUKTURA IOS APLIKACE **17**
2.1 STORYBOARDS VS. XIB VS. KÓD 17
2.1.1 Storyboard 17
2.1.2 Xib 17
2.1.3 Kód 18
2.2 ORIENTACE V KÓDU 18
2.2.1 Delegát 19
2.2.2 Notifikace 19
2.2.3 Extension 19
2.2.4 Override 20
2.2.5 Proměnné a typy 21
2.3 APPLE APP SERVICES (NATIVNÍ NÁSTROJE) 21
2.3.1 Core Location 21
2.3.2 UIKit 21
2.4 ZÁKLADNÍ ZOBRAZOVACÍ KOMPONENTY UIKITU 22
2.4.1 UIView 22
2.4.2 UILabel 22
2.4.3 UITableView 23
2.4.4 UICollectionView 23
2.4.5 UIButton 24
2.4.6 UIAlertController 25
2.4.7 UIImageView 25
3 BACKEND **28**
3.1 SROVNÁNÍ DOSTUPNÝCH BACKENDŮ 28
3.1.1 Node.js 28
3.1.2 Laravel 29
3.1.3 Lumen 29
3.2 POUŽITÉ PHP FRAMEWORKY 30
3.3 FIREBASE 30
II PRAKTICKÁ ČÁST **31**
4 PRŮZKUM TRHU **32**
4.1 FUNKČNÍ A NEFUNKČNÍ POŽADAVKY 34
5 NÁVRH MOBILNÍ APLIKACE **36**
5.1 UI, WIREFRAME A PROTOTYP 36
6 BACKEND **38**

6.1 STRUKTURA DATABÁZE 38
6.1.1 Problémy při vytváření databáze 40
6.2 HLAVNÍ ČÁSTI BACKENDU 40
6.2.1 Část vytváření restaurací a práce s daty: 40
6.2.2 Část obsluhy aplikace: 41
6.2.3 Část odeslání objednávek: 41
6.3 TESTOVÁNÍ POMOCÍ POSTMAN 41
7 TVORBA APLIKACE 43
7.1 VYTVOŘENÍ ZÁKLADNÍ ČÁSTI UI 43
7.2 NAPOJENÍ NA BACKEND 44
7.3 NAČÍTÁNÍ QR KÓDU 45
7.3.1 Firebase 47
7.4 OBSLUHA APLIKACE PERSONÁLEM 48
7.5 MOŽNOSTI DALŠÍHO ROZŠÍŘENÍ APLIKACE 48
7.6 NASAZENÍ V RESTAURACÍCH 49
ZÁVĚR 50
SEZNAM POUŽITÉ LITERATURY 52
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 55
SEZNAM OBRÁZKŮ 56
SEZNAM TABULEK 57
SEZNAM PŘÍLOH 58

ÚVOD

Často řešíme, kde se rychle a zdravě najíst, kde mají i výběr dobrých jídel a nápojů. Problém nastává, pokud se dostaneme do restaurace např. když je plná zákazníků a číšník na nás nemá čas.

V dnešní době snad neexistuje člověk, který by neměl telefon, mnohdy mají lidé i více telefonů. Samotné telefony nás ovlivňují víc a víc. Před 7 lety byly chytré telefony teprve na počátku své cesty, ale dnes se dá říci, že je to opravdu počítač, co nosíme každý ve vlastní kapse. Dnes si už asi většina z nás nepůjde koupit noviny nebo časopis, můžeme si je koupit přímo v zařízení nebo si je přečíst v internetových novinách.

Cílem této bakalářské práce je zjednodušit proces objednání v jakékoliv restauraci nebo podniku kde obsluhuje 1 i více číšníků, a to díky zařízení s iOS. V podnicích se běžně můžeme setkat s „kiosky“ nebo tablety, které nám urychlí objednávku, ale objednávat přímo z telefonu neumožňuje žádná restaurace. Praktickým výstupem tedy bude návrh aplikace pro koncové uživatele mobilních telefonů. Volba pro iOS padla hned z několika důvodů a to: jednotný systém, jednoduchá manipulace s aplikacemi, rychlá možnost migrace na nový iOS, do budoucna Apple Pay, nebo iBeacon.

Teoretická část práce popisuje, co to je mobilní aplikace, jaké jsou možnosti vývoje, jak se pracuje s kódem v jazyce Swift, základní vlastnosti jazyka Swift, jaké jsou možnosti tvorby serverového pozadí aplikace (backendu) a jaká je struktura iOS aplikace.

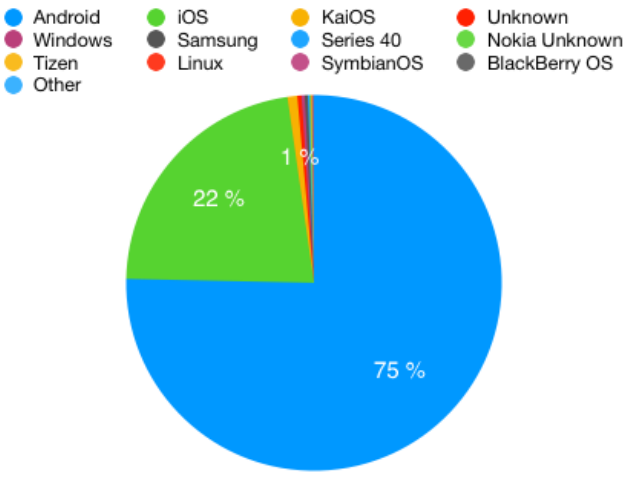
Praktická část dokumentuje průzkum trhu, postup vytváření samotné aplikace, backendu a napojení jednotlivých částí. Závěr práce je věnován testování prototypu a možnosti dalšího vývoje aplikace a samotnému nasazení v restauracích.

I. TEORETICKÁ ČÁST

1 MOBILNÍ APLIKACE

Mobilní aplikace je software určený pro mobilní zařízení typu tablet nebo telefon, v poslední době i chytré hodinky. Drtivá většina uživatelů mobilních zařízení se s nějakou aplikací setkala, nebo je denně využívá. Samotná aplikace má za úkol ulehčit práci, zabavit nudu, nebo poskytnout to, co právě hledáme. Samotné zařízení mají již od výrobce předinstalované určité aplikace, které uživatelé mohou využívat, nebo používají alternativy těchto aplikací od vývojářů třetích stran. Určitě aplikace, kterou uživatelé využívají denně je Telefon (nebo telefonní adresář), Zprávy, Hudba, Fotoaparát, Hodiny (formou budíku), Mapy, Prohlížeč a další. Většina těchto aplikací je v každém zařízení, které uživatel zakoupí. Pokud uživatel tyto předinstalované aplikace nepotřebuje, lze je odebrat. Neplatí to o všech aplikacích, asi těžko uživatel odebere hlavní aplikaci Telefon, nebo Zprávy. Uživatel má možnost si dodatečné aplikace zakoupit prostřednictvím obchodu aplikací.

Obchod s aplikacemi je součást softwaru telefonu, který má za cíl nabídnout uživateli přesně ty aplikace, které potřebuje nebo hledá. Každá taková nabídnutá aplikace má popis, hodnocení, fotky a informace pro koho je určená. Některé aplikace jsou zdarma, jiné za poplatek nebo za měsíční členství. V tento moment nastupují samotní vývojáři, kteří aplikace vytvářejí a umisťují na obchody s aplikacemi. V dnešní době jsou populární 2 mobilní operační systémy a také 2 obchody s aplikacemi. Na platformě Android je to Google Play a na iOS je to AppStore.



Obrázek 1 - Procentuální využití mobilních operačních systémů [3]

Obrázek 1 znázorňuje procentuální pokrytí celosvětového trhu operačních systému v mobilních zařízení. Data jsou aktuální z dubna 2019. Je možné si všimnout že Android je oproti

iOS daleko rozšířenější. Faktem je, že do 75 % budou patřit i zařízení typu jednoúčelové tablety pro obsluhu, případně podobná zařízení, která mají jen jednu jedinou funkci a jiné aplikace na ně nelze nainstalovat, proto je podíl tak velký.

1.1 Multiplatformní vývoj

Existuje mnoho možností, jak vyvíjet aplikace, jednou z nich je právě multiplatformní (neboli hybridní) vývoj, který slouží vývojářům jako rychlé a účinné řešení pro menší aplikace, které nepotřebují velký výkon a mají běžet na více platformách (dnes iOS a Android). Lze zde zmínit Xamarin, který slouží pro vývoj aplikací v C#, pak NativeScript nebo React Native, kde se kód píše v JavaScriptu a styluje pomocí XML nebo HTML a CSS/SCSS. A na konec IONIC nebo Flutter, kde se kód také píše v JavaScriptu a styluje se pomocí XML, HTML a CSS. Řešení existuje daleko více, zde autor uvádí jen ty, ze kterými se v praxi setkal. Většina multiplatformních kódů využívá vestavěného prohlížeče pro vykreslování XML/HTML/CSS prvků, proto je lze spustit na různých zařízeních.

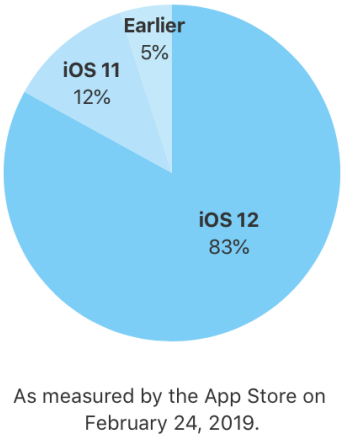
Každé z těchto multiplatformních řešení má své výhody i nevýhody. Rozhodně se jim nedá zapřít jednoduchost a rychlost tvorby aplikací, ale i toto přináší problémy. Jeden kód pro všechny zařízení zní náramně hezky, ale občas se vyskytne problém, který postihuje určitý typ zařízení. Příkladem může být, když se vyvíjí aplikace v multiplatformní kódu pro iOS, většinou vše bude fungovat stejně na všech iOS zařízeních. Opakem toho bude vývoj na Android, který má velké množství různých velikostí obrazovek, každé zařízení obsahuje jiný hardware a software, a tudíž je i jinak výkonné a tak dále. Tato problematika se nazývá *Fragmentace mobilních zařízení*. [15]

1.2 Platforma iOS

Operační systém iOS je mobilní operační systém vytvořený společností Apple v roce 2007. Původně byl tento systém pojmenován jako iPhone OS a byl předinstalovaný v iPodu Touch a iPhone vydaného roku 2007. Následně v roce 2010 došlo k přejmenování na název iOS. Během dalších let došlo k několika obměnám systému, ale základní koncept zůstal zachován. Apple dnes systém postupně integruje do dalších zařízení jako je: iPad, Apple TV (částečná verze iOS) nebo Watch. [16]

Apple systém vyvíjí velice obezřetně, tak aby hodnota celého systému, a tak i uživatelské rozhraní zůstalo co nejvíce stejné i v nových verzích. Firma podporuje svá zařízení velmi

dlouho. Podporou je myšlena možnost dostávat aktualizace i pro zařízení 5 let stará. S rychlostí, jakou se dnes zařízení vyvíjí je Apple jedničkou se samotnou podporou zařízení. [17]



Obrázek 2 - Procentuální podíl verzí systému iOS [2]

Na obrázku 2 lze jasně vidět že nejnovější verze iOS 12 je na 83 % zařízeních, což velice zjednodušuje vývoj samotné aplikace a určuje verzi iOS, na kterou se vývojář má zaměřit.

Uživatelské prostředí iOS je navrženo tak, aby bylo jednoduché a vždy bylo stejné a mohl k němu každý přistupovat. Nejde jen o rozložení prvků, ale třeba funkce pro zrakově postižené, čtení na obrazovce, diktování a další možnosti. Celý systém je ovládán pouze dotyky a u starších modelů i jedním tlačítkem, nepočítaje postranní tlačítka na ovládání hlasitosti a přepnutí tichého režimu. Velkou výhodou je, že propojenost všech Apple zařízeních je tak jednoduchá a funkční, že není problém, aby uživatel rozepsal email na Apple Watch, upravil jej na iPhone, fotky přidal z iPadu a přílohy z Macu. Navíc všechny zařízení běžící na iOS (iPhone, iPod Touch a iPad) mají totožný základ, a proto se stejně i ovládají. Nadneseně lze říct, že to co bylo v aplikaci vpravo nahoře u iPhone bude tam i u iPadu. [17]

1.2.1 Vývoj na platformu iOS

Pro vývoj na platformu iOS je potřeba vlastnit jakýkoliv počítač (Mac) značky Apple. Další podmínkou je mít nainstalované vývojové studio Xcode, které je dostupné zdarma v Mac AppStore. V jakém studiu už bude vývojář vyvíjet, je na něm, ale Xcode musí být vždy nainstalovaný, aby mohlo dojít ke kompilaci, publikaci aplikace na AppStore a dalším úkolům, které lze dělat pouze díky Xcode. Autor nejraději používá AppCode od firmy JetBrains s.r.o., avšak tento projekt je vyvíjen v Xcode. Důvodem je rychlejší indexace souborů na starším počítači.

Vývoj lze provádět ve dvou jazycích, a to Objective-C nebo moderním Swift. Aplikace bude psána v jazyce Swift, jelikož je podobný jazykům C, PHP, JavaScript, které autor používá. Kód, který vývojář napíše, lze testovat dvojím způsobem. Buď v Playground – což je místo na „hraní“, nebo jako reálná aplikace v simulátoru. Zde lze spustit i složitější operace a vizuální prvky, vyzkoušet různé šířky obrazovek, nebo rozdílné verze operačních systémů. Ale většina vývojářů testuje na poslední 2 verze systému, jelikož převážná část zařízení má nainstalovanou nejnovější verzi iOS, viz obrázek 2. Pro vývojáře je skvělé vědět, že když bude vyvíjet na nejnovější verzi systému, nic tím nepokazí, protože obsáhne přes 80 % trhu iOS zařízení. Pokud bude chtít vývojář vyvíjet pro starší verze iOS, musí mít na paměti, že určité funkce, co jsou dostupné v posledních verzích iOS, nemusejí být dostupné a bude muset hledat alternativy. Stejně tak pokud používá nejrůznější frameworky je možné, že ani ty, nelze spustit na starších verzích.

Pokud vše běží v simulátoru, tak lze otestovat aplikaci na reálném zařízení, protože ne vše je v simulátoru dostupné. To lze provést dvěma způsoby. Pokud uživatel není registrován jako vývojář, může aplikace na reálné zařízení nainstalovat, ale certifikát pro spuštění aplikace bude platit zhruba týden. Po instalaci na zařízení je vyzván k ověření vývojáře v nastavení, pokud tak učiní, může aplikaci používat na svém zařízení po dobu týdne. Druhá cesta je placená, uživatel si musí své Apple ID zaregistrovat jako Developer a následně zaplatit roční poplatek za členství. Tento poplatek zpřístupní betaverze iOS, možnost publikovat na App Store, sdílet aplikace testerům a další.

1.2.2 Programovací jazyk Swift

Programovací jazyk Swift je poměrně nový (2. června 2014), ale již teď má 5. verzi. Oproti Objective-C přináší několik výhod, které jsou i jeho charakteristickými rysy:

- Objektově orientovaný
  Swift je moderní, objektově orientovaný jazyk, kde vše existuje jako objekt. Vychází z jazyka C a Objective-C, proto je možné používat stejné typy a podobné postupy. [18]
- Přehledný
  Ve Swiftu je jednoduché programovat, a též kód číst. Má jasný způsob zápisu, je konkrétní, s minimem skrytých funkcí. [18]
- Bezpečný

Swift je silně typový jazyk, nutí tak vývojáře přemýšlet nad tím, co bude v jakém typu, a co se v daném momentu v objektu nachází. [18]

- Ekonomický
  Swift je dosti malý programovací jazyk poskytující základní typy a funkce a nic víc. Zbytek musí být naprogramován vývojářem nebo knihovnami, které vývojář používá. [1]
- Správa paměti
  Swift spravuje paměť automaticky. Vývojář se shledá se správou paměti jen velmi zřídka. [1]
- Kompatibilita s Cocoa
  Cocoa API jsou frameworky určené k fungování aplikací pro iOS. Cocoa API jsou psané v původním Objective-C a klasickém programovacím jazyku C. Swift je navržen tak, aby byl propojený s většinou těchto frameworků. [1]

Vlastnosti jazyka Swift z něj dělají ideální jazyk pro tvorbu mobilích iOS aplikací.

Swift je také od druhé verze Open Source, a díky tomu jej lze spustit na OS Linux, oficiální dokumentace uvádí OS Ubuntu.

1.2.3 Cocoapods

Je chytrý správce závislostí (dependency manager), díky kterému lze do projektu vývojáře přidat další frameworky (projekty). V základu se jedná o správce závislostí pro jazyk Objective-C, ale v případě Swiftu je totožný. Jak je popsáno v bodu 1.2.2, tak je Swift kompatibilní s Cocoa, a tím je i zachována spolupráce obou jazyků.

Na oficiální stránce lze nalézt velké množství knihoven, které vývojářům usnadňují práci. Příkladem může být SwiftyJSON knihovna, která má za účel jednoduší práci s JSON objekty dosazených z API.

Samotná instalace Cocoapods je velmi snadná a s nadsázkou řečeno bezúdržbová. Zaprvé je nutné nainstalovat cocoapods do systému, na kterém budeme vyvíjet (Unix OS). Po nainstalování přejít do složky s již vytvořeným projektem a zde vytvořit „Podfile“, nebo použít příkaz *pod init* v terminálu. Do souboru s názvem Podfile pak vývojář přidává jednotlivé knihovny a případně nastavuje jejich verze. Následně stačí tyto knihovny do projektu nainstalovat příkazem pod install a otevřít místo projektu aplikace, workspace aplikace, který se

instalací knihoven vytvořil. Využití knihoven je jednoduché, vždy je třeba danou knihovnu importovat jako hlavičku na začátku kódu (souboru).

2 STRUKTURA IOS APLIKACE

Každý vývojář se při tvorbě aplikace setká s oficiální dokumentací k jazyku Objective-C nebo Swift. Velká část této dokumentace je psána pro pokročilé, a proto existuje v IDE Xcode jednoduchý grafický editor samotné aplikace. Mezi vývojáři platí obecné pravidlo, že někdy je dobré tento editor používat a jindy nikoliv.

2.1 Storyboards vs. Xib vs. kód

Na platformě iOS existují 3 možné přístupy, jak vytvářet aplikaci, každý z nich má určité výhody, ale i nevýhody. Pokud se vývojář rozhodne postupovat určitým směrem, měl by tento směr dodržovat a nepřeskakovat mezi jednotlivými možnostmi.

2.1.1 Storyboard

Storyboards jsou nejmladší, ze zmíněních tří metod pro tvorbu uživatelského rozhraní. Lze si je představit jako kreslící plátna, na kterých vytváříme vlastní design a malujeme na ně stejně jako malíř. Každé z těchto pláten propojujeme přechody. Chyba, které se zde mnoho vývojářů dopouští je, že do jednoho Storyboard přidají extrémně moc pláten, a tím celý proces zpomalí a znepřehlední. Aplikace by se měla dělit do určitých celků: Storyboards pro obsluhu uživatelského profilu, pro nákupní košík a podobně. [4]

Největší výhodou Storyboards je jejich vizuální explicitnost. Každý, kdo se podívá do storyboardu, vidí jasnou návaznost jednotlivých obrazovek, přechody mezi nimi a co v sobě obsahují. Další výhodou je prototypování, díky čemu je možné rychle naklikat jednotlivé obrazovky a předvést tak prototyp aplikace v reálném použití. [4]

Za určitou nevýhodu lze považovat znovupoužitelnost kódu, především grafických prvků. Nedají se třeba vytvořit složitější grafické prvky typu zakulacení jen jednoho rohu tlačítka. Určitě je ale Storyboards jednoduchý, a lze s ním vytvořit statické elementy, které se snadno editují. Příkladem může být tabulka přihlášení nebo registrace uživatele. [4]

2.1.2 Xib

Xib je podobně jako Storyboards vytvořený v Interface Builderu, ale reprezentuje pouze jeden konkrétní element (view, controller, buňku tabulky atd.) naproti Storyboardu, který prezentuje celé obrazovky s elementy. Díky tomu je přehlednost horší než u Storyboardů, ale vývojář získá větší znovupoužitelnost kódu a větší kontrolu nad logikou aplikace. Ale jelikož

je vytvářen ve Interface Builderu, nese s sebou stejné špatné vlastnosti jako v případě Storyboards. [4]

2.1.3 Kód

Spousta vývojářů by o programování GUI v kódu řekla, že je zdlouhavé a nudné. Přesto má tento způsob větší výhody a více využití než vytváření GUI v Interface Builderu. [4]

Znovupoužitelnost je v případě vlastního kódu nejlepší. Je výhodné si připravit jednotlivé vizuální prvky tlačítek a používat je napříč objekty. Proces přidávání kontrolérů do kontroléru je v případě kódu zdlouhavý a v Interface Builderu naopak jednoduchý a rychlý. [4]

Upravitelnost kódu je další nespočetnou výhodou, změna barvy, fontu, velikosti písma se provádí rychle a bez jakýchkoliv komplikací na jednom místě. Nezáleží pak už na tom, jestli se vyvíjí multibrandingová aplikace nebo se vyvíjí agilně a mění se požadavky. Vždy je reakce na změnu velice pružná. [4]

Při práci ve více lidech na stejném projektu je kód snažší na údržbu, merge requesty, a tím také zrychluje samotný vývoj. [4]

Lokalizace se provádí jednou metodou a lokalizačním souborem. Oproti Interface Builderu, který je možné také lokalizovat nabízí tato práce rychlejší pokrok. Lze si nastavit vlastní klíče a překlady a velmi snadno je poté použít. V případě Interface Builderu je proces více chaotický. [4]

Animace a dynamika různých přechodů a animací se v Interface Builderu dělají velice komplikovaně, kolikrát i nemožně, pokud je v kódu vlastní hierarchie a vlastní kontroléry. Pokud je třeba skrýt určitý prvek v závislosti na datech, je práce opět jednodušší v kódu. [4]

Autolayout, velikosti a odsazení prvků se v dnešní době vytváří snáze v Interface Builderu, ale vznikají problémy, kdy je třeba určitý prvek dynamicky změnit. Kód je potom rozházený a neuspořádaný. Pokud využijeme některé z knihoven pro layout, je práce daleko příjemnější přímo v kódu. Aktuální verze jazyka Swift přináší mnohé vylepšení, které už tak zjednodušují práci s layoutem prvků a jsou snadné na implementaci. [4]

2.2 Orientace v kódu

Obecné doporučení přímo Applu je psát kód v MVC – Model-View-Controller (model-pohled-kontrolér) návrhovém vzoru. Jelikož je kontrolér úzce spjat s pohledem, je dobré mít

jen pohledové třídy, které implementujeme do kontroléru. Pokud by vývojář šel cestou, kde kontrolér má vlastní soubor a kompletní pohled má vlastní soubor, mohou se vyskytnout problémy při aktualizaci pohledu, případně při úpravách v závislosti na datech.

Každý, kdo někdy zkusil vyvíjet pro platformu zjistil, že kód má svá specifika a vše se řeší různými návrhovými vzory.

2.2.1 Delegát

Návrhový vzor delegáta je snadný způsob, jak komunikovat mezi dvěma objekty přes společné rozhraní – protocol (protokol). Pro práci s delegátem je třeba vytvořit protokol, který rozdává „úkoly“ metody a objekt, který tento protokol implementuje a „úkoly“ plní. [5]

Delegátem lze také zachytit určitá data, která se mají předat mezi objekty. Například po zobrazení modalového okna s editací textu. Upravit text a vrátit onen editovaný text do původního objektu.

2.2.2 Notifikace

Jde o metodu upozorňování všech objektů, které zachytávají změny, na které mají reagovat. Zjednodušeně řečeno jde o Události (Eventy), které lze zachytit na jiné části aplikace.

Kdekoliv v kódu se vytvoří notifikace s daným jménem, a na jiné části aplikace se tato notifikace se stejným jménem odchytává pomocí objektu nazývaného Observer. Pokud kód, kde se notifikace zachytává, nebyl nikdy spuštěn, není možné tuto notifikaci zachytnout. Například pokud jsou v aplikaci implementovány dvě stránky, tak notifikace vzniklá na první stránce nemůže upozornit stránku druhou, pokud tato stránka nebyla nikdy načtena.

2.2.3 Extension

Jde o unikátní vlastnost swiftu, díky které lze jednoduše rozšířit již existující třídu bez nutnosti dědění. Například třída UIColor nemá metodu na implementaci hexadecimální barvy, kterou vývojáři znají z CSS #000FFF. Díky extension má vývojář možnost vytvořit metodu do hlavní třídy UIColor nebo přepsat konstruktor (funkce init) pro své potřeby.

```swift
import Foundation
import UIKit
extension UIColor {
    convenience init(hexString: String) {
        let hex = hexString.trimmingCharacters(in: CharacterSet.alphanumerics.inverted)
        var int = UInt32()
        Scanner(string: hex).scanHexInt32(&int)
        let a, r, g, b: UInt32
        switch hex.count {
        case 3: // RGB (12-bit)
            (a, r, g, b) = (255, (int >> 8) * 17, (int >> 4 & 0xF) * 17, (int & 0xF) * 17)
        case 6: // RGB (24-bit)
            (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)
        case 8: // ARGB (32-bit)
            (a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int >> 8 & 0xFF, int & 0xFF)
        default:
            (a, r, g, b) = (255, 0, 0, 0)
        }
        self.init(red: CGFloat(r) / 255, green: CGFloat(g) / 255, blue: CGFloat(b) / 255, alpha: CGFloat(a) / 255)
    }
}
let darkGrey = UIColor(hexString: "#757575")
//R: 0.459, G: 0.459, B:0.459 A:1
let white = UIColor(hexString: "#FFFFFF")
//R: 1, G: 1, B:1 A:1
```

Obrázek 3 - Rozšíření třídy UIColor

2.2.4 Override

Jedná se o přepisování již existujících metod nebo vlastností (property) v objektu. Při dědění se před metodu napíše klíčové slovo override. Pokud by programátor toto klíčové slovo nenapsal, nastalo by následující: jazyk Swift může mít stejnou metodu se stejným, nebo jiným počtem parametrů nebo typů parametrů tolikrát, kolikrát je třeba. Platí to i pro konstruktor. Demonstraci lze vidět na obrázku 4 níže.

```swift
class Test {
    init(a: Int){
        print("int: \(a)")
    }
    init(a: String){
        print("string: \(a)")
    }
}

let a = Test(a: "4")
// string: 4
let b = Test(a: 4)
// int: 4
```

Obrázek 4 – Prezentace totožných metod

Stejnou techniku s menší obměnou lze použít i při rozšiřování tříd o vlastní konstruktory jako je třeba konstruktor barvy pro hexadecimální kód.

2.2.5 Proměnné a typy

Jazyk Swift je silně typový jazyk, proto se používají 2 typy. Let pro vytvoření konstanty, nebo var pro vytvoření proměnné. Dále není třeba striktně definovat o jakou hodnotu se jedná, jazyk Swift si typ proměnné automaticky zjistí.

Pokud vývojář bude definovat konstantu, do které se zapíše hodnota až v konstruktoru je třeba typ konstanty určit. Je také možné nastavit, zda tato konstanta bude nastavena v základu na nil (obecně znamení null) a nebo bude vždy striktně dosazena. Demonstruje obrázek 5

```
// definovaný string
let a = "Test"
// definovaný int
let b = 4
// string s definovanou nil
var c: String?
// string s nedefinovanou hodnotou která musí být přiřazena
var d: String!
// string k definouvanou hodnotou nil
let e: String! = nil
```

Obrázek 5 - Definice proměnných

2.3 Apple App Services (nativní nástroje)

Apple pro vývoj na iOS poskytuje určité softwarové balíky, většina z nich se jmenuje KIT, ale toto pojmenování nenesou všechny. Kompletní seznam balíku je dostupný na adrese dokumentace pro vývoj na iOS zařízení.

2.3.1 Core Location

Knihovna poskytující přístup k určení geografické polohy, nadmořské výšky, orientace nebo polohy vzhledem k objektu iBeacon. Knihovna používá všechen dostupný hardware v zařízení včetně Wi-Fi, GPS, Bluetooth, magnetometru, barometru a mobilního přijímače signálu. [6]

Při prvním spuštění této knihovny se zařízení uživatele zeptá, zda mu povolí přístup k polohovým údajům. Hláška, která se zobrazí je částečně editovatelná v souboru Info.plist. Po získání potřebného oprávnění je možné přistupovat k poloze zařízení. [6]

2.3.2 UIKit

Knihovna UIKit poskytuje požadovanou infrastrukturu pro aplikaci iOS nebo tvOS. Poskytuje okna pro zobrazení obsahu, infrastrukturu pro zpracování událostí, Multi-Touch, další

typy vstupu do aplikace a hlavní smyčku potřebnou pro správu interakcí mezi uživatelem, systémem a aplikací. Mezi další funkce, které tato knihovna nabízí patří podpora animace, dokumentů, kreslení a tisku, informace o zařízení, správa a zobrazení textu, podpora vyhledávání, podpora usnadnění přístupu, rozšíření aplikací a správa zdrojů. [7]

2.4 Základní zobrazovací komponenty UIKitu

UIKit je nejpodstatnější částí celé aplikace, jelikož obsahuje veškeré grafické komponenty a spoustu dalšího. Vývojář se při práci nejčastěji setká s níže zmíněnými prvky.

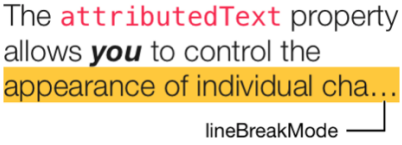
2.4.1 UIView

Lze popsat jako základní objekt většiny grafických objektů. Do objektu lze vkládat další objekty a díky tomu lze vytvářet složitější grafické prvky a přizpůsobovat tak vzhled. Samotné UIView je definované jako čtverec s barevným pozadím. S UIView se dají dělat i složitější operace, jako jsou animace, roztahování, posunování, změna barvy, nebo zmizení. UIView zprostředkovává i uživatelská gesta jako je dotyk jedním prstem, dvěma anebo překrytí dlaní. [8]

Díky tomu že do UIView lze přidávat další View tak se vytváří stromová struktura, kde dítě UIView se nazývá *subview* a rodič *superview*, a tyto UIView lze jednoduše přeskládat a měnit tak jejich zobrazení. [8]

2.4.2 UILabel

Jde o jednoduchý textový objekt, který lze upravovat pomocí atributů jako je: barva textu, velikost textu, stín, zvýraznění a spousta dalších. [9] Velkou výhodou je automatické zalomení textu, kdy je programátor schopný definovat počet řádků, na které bude text zalomen a oddělen znaky „…“, stejná funkce se také projevuje, pokud se nastaví výška objektu a text uvnitř tohoto objektu je vyšší než celková výška objektu.

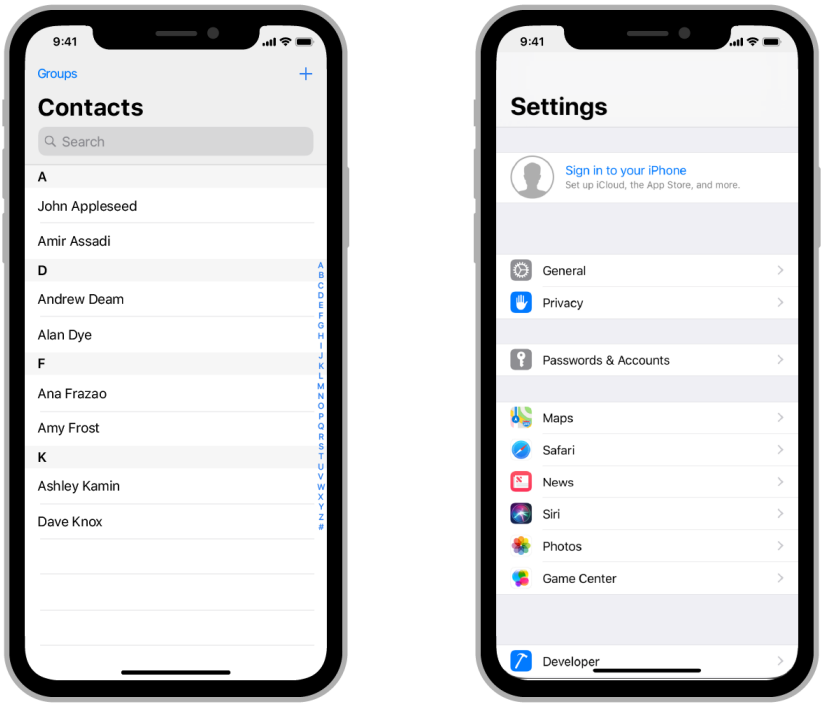


Obrázek 6 - Label zobrazující text s atributy [9]

Pokud chce vývojář použít víceřádkový text, který bude možné posunovat, místo **UILabel** využije objekt **UITextView**

2.4.3 UITableView

UIKit obsahuje různé tabulkové řešení zobrazení obsahu, UITableView je jedno z nich. Jedná se o vertikálně posuvný sloupec obsahující řádky UIView (UITableViewCell). Každý řádek v tabulce prezentuje jeden obsah. Například aplikace Kontakty zobrazuje název kontaktu v samotném řádku nebo aplikace Nastavení zobrazuje dostupné skupiny nastavení. Tabulka se dá zobrazit dvěma způsoby. Buď jako jeden dlouhý seznam, nebo dělit seznam do sekcí nebo skupin, a tím usnadnit orientaci v obsahu. [19]



Obrázek 7 – Ukázka UITableView [19]

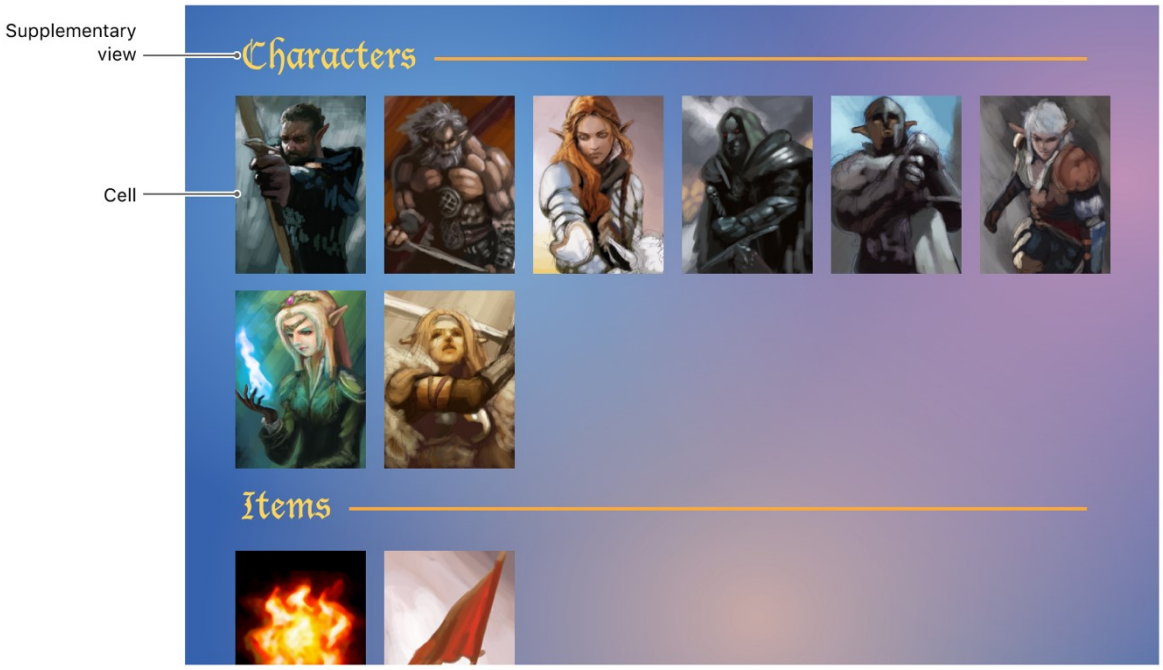
2.4.4 UICollectionView

UIKit obsahuje různé tabulkové řešení zobrazení obsahu, UIColletionView je další z nich. Je také nejčastěji využívané, protože v porovnání s UITableView má jednodušší možnosti editace a formátování obsahu.

UICollectionView je složeno z několika částí, UIColectionViewCell, UICollectionView a Layout. Zjednodušené vysvětlení, co UICollectionView vlastně je, by se dalo popsat jako

pole (UICollectionView) s čtverci/itemy (UICollectionViewCell) o základní šířce 50x50. Každý prvek se dá editovat a může vypadat odlišně. Navíc samotné UICollectionView obsahuje sekce a díky nim lze obsah dělit do uzavřených částí, které se navzájem neovlivňují. [10]

Pomocí Layoutu (UICollectionViewLayout) se definují vlastnosti Collection View jako jsou: směr pohybu, rozložení prvků, zápatí a záhlaví, nebo opakované použití určitého zobrazení. [10] Obrázek 8 znázorňuje, jak může vypadat UICollectionView se sekcemi.



Obrázek 8 - CollectionView využívající flow layout [10]

2.4.5 UIButton

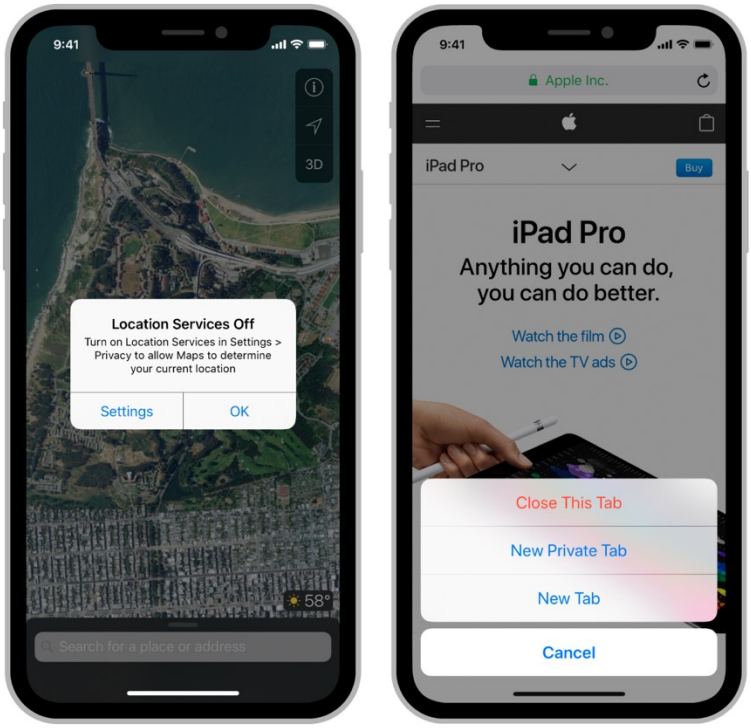
UIButton je objekt tlačítka, který reaguje na akce uživatele. Účel tlačítka je oznamován pomocí textového štítku, obrázku nebo obou. Vzhled tlačítka je konfigurovatelný, takže může vypadat přesně tak, jak je potřeba. [20]

Obrázek 9 -

Ukázka UIButton [20]

2.4.6 UIAlertController

UIAlertController reprezentuje zprávu pro uživatele formou vyskakovacího okna s potřebnou interakcí. Slouží především k zobrazování výstrah a akčních listů se zprávou. Před zobrazením tohoto elementu je třeba přidat akce pomocí metody addAction(_:), která se zavolá po interakci uživatele s akcí [21]



Obrázek 10 - Ukázka UIAlertController [22]

2.4.7 UIImageView

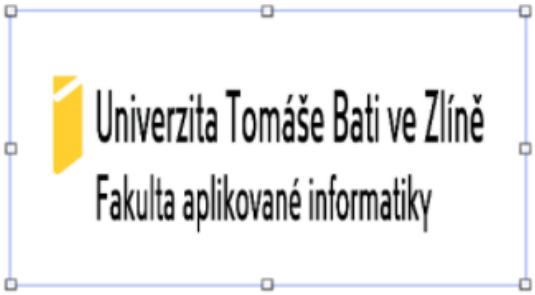
UIImageView je stejné jako UIView s možností zobrazení obrázků. Tento objekt umožňuje efektivně nakreslit libovolný obrázek, který lze zadat pomocí objektu UIImage. Apple podporuje velké množství standartních obrazových souborů, nejčastěji pak JPEG nebo PNG. Samotné PNG je doporučeno pro vše a vývojář by měl tento standart dodržovat. Obrázky lze umístit z různého zdroje. Nejlepší řešení je však umístit obrázky do katalogu *Asset*, který slouží pro soupis všech obrázků a lze je tak snadno použít v celé aplikaci. [11]

Jelikož je UIImageView podtřídou UIView lze tomuto objektu nastavit pozadí. Například červenou barvu a obrázek vykreslovat nad tímto pozadím. Pokud vývojář využije PNG soubory, které mají průhledný kanál lze tak docílit požadovaného zobrazení, občas je nutné, se této vlastnosti vyvarovat. [11]

Důležitý atribut je *contentMode*, který zobrazuje a upravuje poměr stran u obrázků. Zpravidla se stává, že obrázek zobrazený v UIImageView je větší, menší nebo má jiný poměr stran a tím pádem se může deformovat. *ContentMode* tento problém řeší, a je proto dobré si před zobrazením obrázku nastavit jakým stylem bude obrázek vykreslován. [10]

Níže jsou zobrazeny nejběžnější metody používané při vykreslování obrázků:

- **scaleToFill**
  Upravení obrázku na stejný poměr stran (výšku i šířku) stejně jako má UIImageView. Při použití této možnosti dochází k deformaci obrázku.



Obrázek 11 - Zobrazení scaleToFill [1]

- **aspectFit**
  UIImageView zobrazený na obrázku 9 má́ nastavenou fialovou barvu pozadí a přizpůsobení obrázku aspectFit. Jak je vidět, obrázek loga Fakulty aplikované́ informatiky je zobrazen správně, i když velikost samotného UIImageView je vyšší. [1]

Obrázek 12 - Zobrazení aspectFit [1]

- **aspectFill**

Tento způsob roztáhne obrázek tak, aby vyplnil celé UIImageView. Pokud bude obrázek přesahovat maximální rozměry UIImageView, lze jej oříznout pomocí parametru *clipToBounds*, který má za následek oříznutí samotného obrázku tak, aby byl správně zobrazen. [1]

Obrázek 13 - Zobrazení aspectFill s clipToBounds [1]

3 BACKEND

Samotná aplikace by byla bez beckendu statická a nic by nedokázala. Proto je třeba mít na jiném místě databázi uživatelů, příspěvky nebo další data, která se mají zobrazovat v závislosti na tom, jak aplikaci uživatelé využívají, nebo kde a kdy ji používají.

Backend je část aplikace, která není na první pohled vidět. Jde o logické funkce po odeslání požadavku na server, který vrátí určitá data. V dnešní době se jedná o REST API nebo SOAP. Menší projekty používají REST (Facebook, Instagram, Twitter) a u starších firem se využívá SOAP z důvodu XML formátu, příkladem můžou být banky, nebo EET.

Vývojář si při výběru vhodného backendu musí položit několik otázek:

1. Kde bude server?
2. Co na serveru poběží?
3. Jak bude server výkonný?
4. Jaké jsou požadavky na backend?

Otázek je vždy více, záleží na tom, jak velký projekt vývojář vytváří, v základu ale stačí tyto čtyři otázky. Z těchto otázek lze vyvodit, jestli bude backend napsán v Javě, PHP, JavaScriptu nebo na jiné platformě. Podle toho se dá usoudit, kde bude nejlepší rozběhnout tento backendový server a jaké musí mít parametry.

3.1 Srovnání dostupných backendů

Tato bakalářská práce je postavena na REST API backendu, je dobré si nějaké přiblížit. Níže autor popisuje možnosti, se kterými se setkal a pracoval s nimi. Především se jedná o vlastní řešení na vlastním serveru.

3.1.1 Node.js

Node.js je prostředí pro spouštění JavaScriptu mimo webový prohlížeč. Je postaveno na Chrome V8 enginu, který lze nalézt v prohlížeči Google Chrome. Primární účel je na psaní serverové části aplikace. Toto je společné s PHP, avšak oproti PHP je kladen důraz na obsloužení co nejvíc klientů a vysokou škálovatelnost. Jelikož je Node.js velice rychlý, je proto často používán na serverech, kde je potřeba jiných vlastností, než nabízí PHP. [12]

3.1.2 Laravel

Laravel je jeden z nejpoužívanějších PHP frameworků. Tento Framework se skládá z jednotného a přehledného kódu a vyznačuje se tím, že je optimalizovaný pro reálné webové aplikace. Vše je připravené pro start a příkazem v terminálu lze vygenerovat stránku nebo databázový model. Jde o open-source framework, který je postaven na modelu MVC.

Mezi základní vlastnosti tohoto frameworku patří [13]:

- autentizace – kontrola přístupu uživatelů
- routování – správa, směrování a zpracování dotazů na jednom místě
- databáze – veškeré nástroje pro komunikaci s databází
- mail – posílání emailu s přílohami a vloženými soubory
- sessions – zastává veškeré agendy okolo sessions
- caching – kešování používaných dat

Důvodem, proč je Laravel tak oblíbený je, že se opírá o dílčí frameworky jako jsou *Symphony, Composer, Eloquent ORM a Blade*. Tyto frameworky spojuje a tvoří jeden velký celek, díky kterému je práce mnohem rychlejší. Samotný framework má spoustu využití, od webových stránek, eshopů až po REST API. [13]

3.1.3 Lumen

Jedná se o micro-framework z frameworku Laravel. Tento framework je určen především pro tvorbu REST API, neznamená to však, že na něm nemůže fungovat web. Jelikož se jedná o stejný systém jako Laravel, je jednoduché s ním pracovat, a používat stejné metody vývoje.

Lumen je oproti Laravelu zaměřený přímo na REST, a tím i výkonnostně rozdílný. Nabízí mnoho jednoduchých metod, které se ovládají příkazy v terminálu a automaticky vygeneruje určité části databáze, modelové třídy nebo kontroléry pro zpracování požadavků.

Pokud by se našel vývojář, kterému v tomto frameworku něco chybí, je možné si jakýkoliv jiný framework doinstalovat přes *Composer*.

3.2 Použité PHP frameworky

Autor použil PHP framework Lumen, jelikož je dostačující pro práci s daty, které bude zpracovávat a lze jej spustit na klasickém webovém hostingu bez nutnosti mít VPS nebo dedikovaný server. Node.js se pro práci nehodil, jelikož by bylo nastavování komplikovanější, a Laravel využívá další knihovny, které nejsou třeba pro tvorbu REST API.

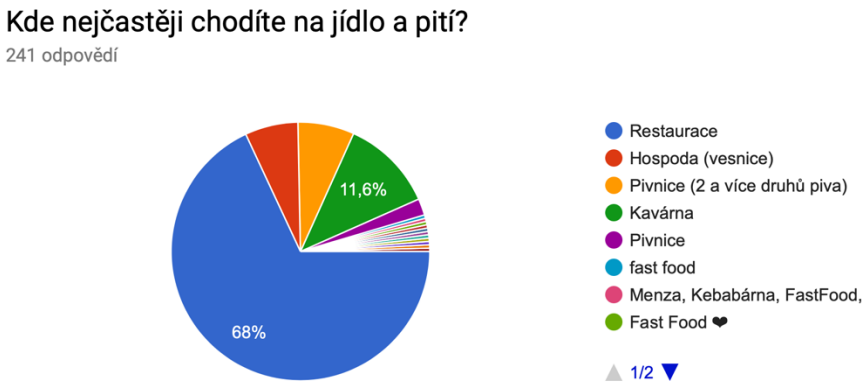
3.3 Firebase

Firebase je platforma pro vývoj mobilních a webových aplikací. Obsahuje řadu klientských knihoven pro mobilní platformy (iOS, Android), i pro web jako je JavaScript. Kromě real-time databáze nabízí i možnosti zasílání push notifikací, sledování statistik, hostingu, A/B testování, nebo datového úložiště. Její rozsah je ale daleko větší než tyto zmíněné možnosti. Většina nástrojů je dostupná zdarma, ale vždy jen do určitého počtu uživatelů, nebo počtu přenesených dat. Vše, co je nad tento počet, je placeno formou některého z tarifů, které Google nabízí. Vývojáři často používají Firebase, protože se rychle a jednoduše implementuje do jakéhokoliv systému. [14]

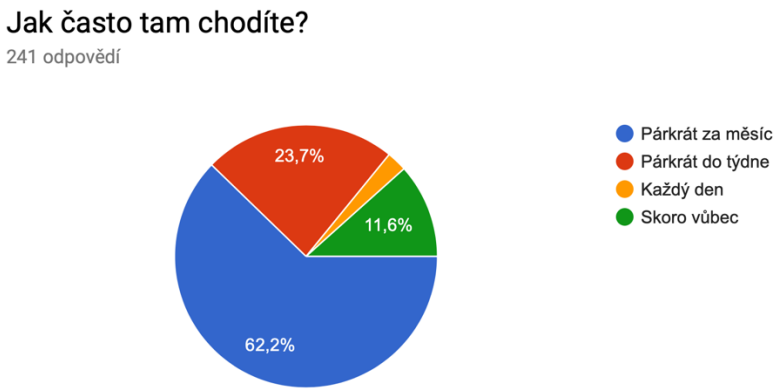
II. PRAKTICKÁ ČÁST

4 PRŮZKUM TRHU

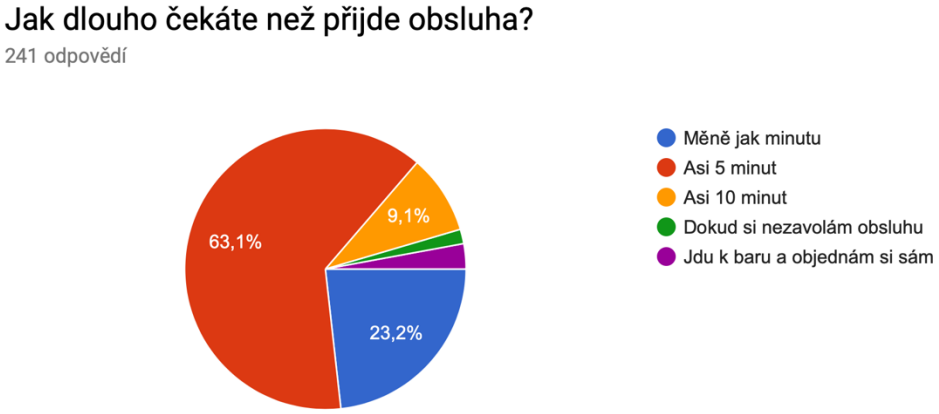
Autor si v průběhu měsíců března – května 2019 vytvořil průzkum formou internetového dotazníku, na téma **Průzkum čekání na obsluhu při objednávání v restauracích**. Tohoto průzkumu se zúčastnilo 241 dotázaných ve věku 15–45 let. *Data jsou dodána z Google Forms*. Z výsledků tohoto průzkumu lze usuzovat že 50 % dotázaných opustilo restauraci dříve, než byli obslouženi a autor tak může vyvodit, že aplikace pro objednání v restauracích může zvednout tržby majitelům restaurací až o 50 %. Během průzkumu se tato hodnota měnila +- 4 %. V průzkumu se autor dotázaných ptal i na to, kam chodí na jídlo a pití, jak často tam chodí, nebo jak dlouho na obsluhu čekají. I z těchto grafů lze vyvodit závěr, že jde o aplikaci, která by si na trhu našla uplatnění v určité sféře podniků.



Obrázek 14 - Průzkum trhu Kde

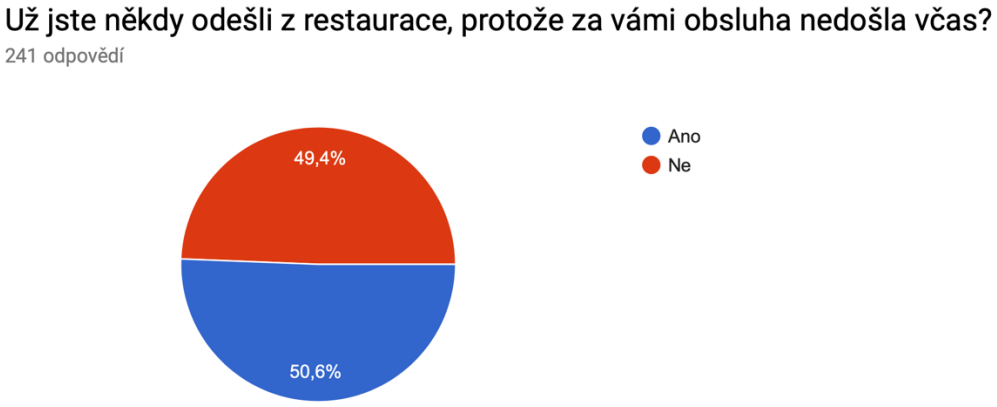


Obrázek 15 - Průzkum trhu Jak často



Obrázek 16 - Průzkum trhu Jak dlouho

Obrázek 17 - Průzkum trhu Jak moc



Obrázek 18 - Průzkum trhu Odchod

4.1 Funkční a nefunkční požadavky

Návrh mobilní aplikace musí mít dané funkční a nefunkční požadavky, které aplikace a backend musí splňovat.

Funkční požadavky na backend:

- Restaurace lze vytvořit bez adresy.
- Jedna restaurace může mít více adres (poboček).
- Každá adresa (pobočka) může mít více kontaktů.
- Každá adresa (pobočka) může mít více jídelních lístků (menu).
- Každá adresa (pobočka) může mít otevírací časy (den, od – do).
- Každá adresa (pobočka) může mít více stolů s vlastním názvem.
- Každá adresa (pobočka) může mít vlastní ilustrační fotky (fotky provozovny…).
- Každé menu musí mít vlastní kategorie s fotkou (Minutky, Hotovky…).
- Každý produkt se přidává k restauraci se základní cenou a měnou.
- Každý produkt může mít více variací (objem, váha) s rozdílnou cenou.
- Každý produkt se bude dělit do předem určených systémových kategorií (nápoje, jídlo, přílohy, omáčky, pizza, pizza-přílohy, polévky, polévky-přílohy) pro dodatečnou nabídku „co k tomu“ při zvolení položky do košíku.
- Produkt lze přidat do menu, do předem připravené kategorie se zvolenou variací a případně upravit cenu pro danou kategorii.
- V případě franšízové restaurace může být stejný jídelní lístek použit s drobnými úpravami i u jiných adres.
- Každá restaurace, menu, kategorie může mít více jazykových mutací.

Funkční požadavky aplikace:

- Aplikace umožní uživateli získat informace o dostupných restauracích v nejbližším okolí.
- Zobrazení informace restaurace nabídne jídelní lístky, kontakty, adresu, otevírací čas a popis.
- Na hlavní stránce má možnost uživatel naskenovat QR kód.
- QR kód nabídne jídelní lístky restaurace.

- QR kód musí upozornit, jestli kód patří k restauraci.
- Zákazník si může vybrat z jídelního lístku.
- Zákazník může zrušit objednávku.
- Zákazník může k produktu přidat přídavky.
- Zákazníkovi může být objednávka zamítnuta obsluhou.
- Zákazník může opakovaně přidávat produkty z jídelního lístku.
- Objednávky mohou měnit stav.
- Zákazník může dát požadavek na zaplacení lístku.
- Aplikace musí upozornit na změny u objednávek.
- Zákazník má možnost zrušit objednávku do 30 vteřin.

Všeobecné nefunkční požadavky:

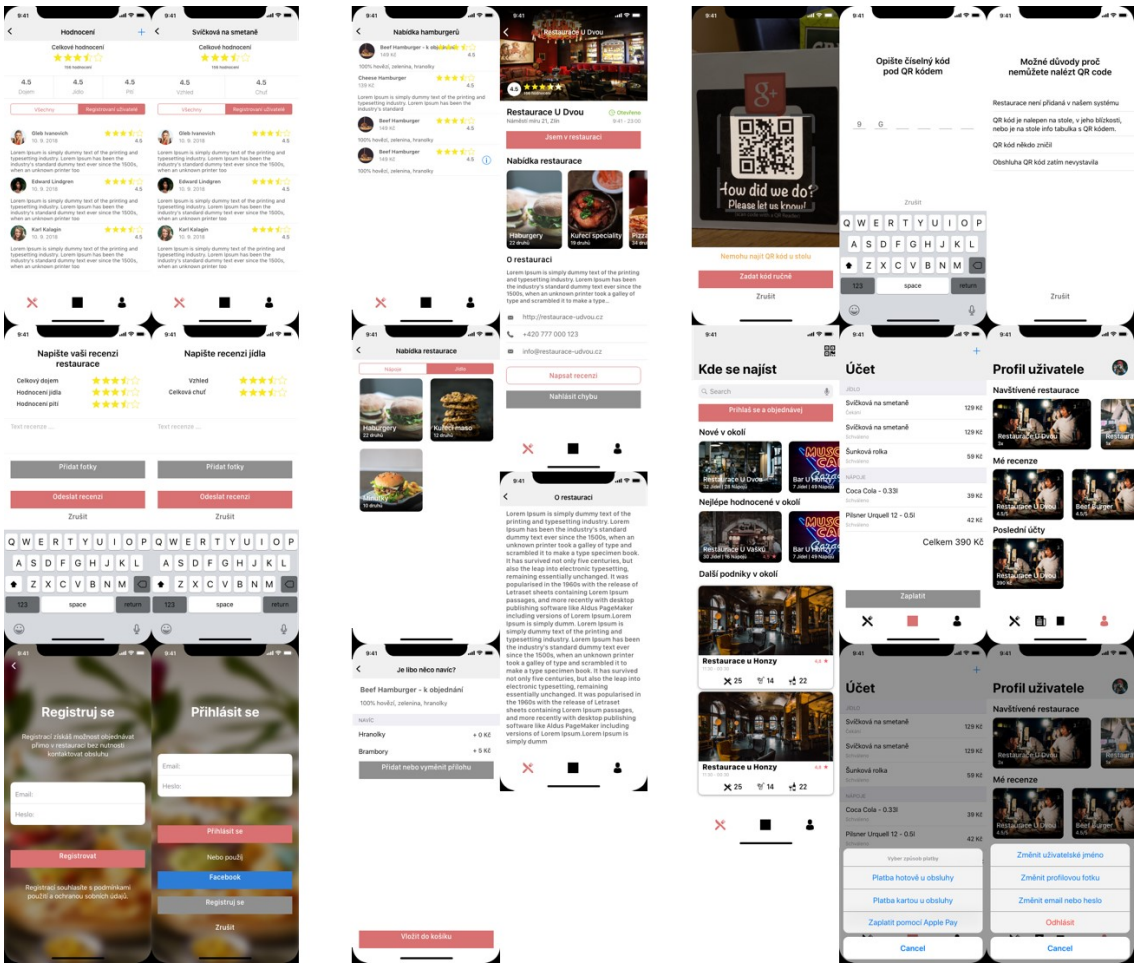
- Aplikace bude kompatibilní se zařízením: iPhone, iPod Touch s operačním systémem iOS 12.1 nebo novějším.
- Data se budou ukládat na server, jen nutně potřebná data do samotného zařízení.
- Aplikace přistupuje a zpracovává data z vlastního serveru formou REST API.
- Aplikace podporuje pouze portrétový režim zobrazení.
- Jednoduché intuitivní uživatelské rozhraní.
- Spolehlivost – při načítání QR kódů a prevence pádů aplikace.
- Robustnost – zachování dat zákazníka při náhlém ukončení aplikace.
- Realtime odezva aplikace.
- Rozšířitelnost.
- Bezpečnost.

5 NÁVRH MOBILNÍ APLIKACE

Cílem této bakalářské práce je navrhnout a vytvořit mobilní aplikaci pro platformu iOS pro zrychlené objednání v restauraci. Záměrem je nabídnout zákazníkovi v restauraci jídelní lístky s předstihem, než přijde číšník. Zákazník si tak může objednat dřív a číšník mu jeho objednávku přinese již objednanou. Po dokončení všech objednávek, může zákazník zaplatit tyto objednávky přímo v telefonu a odejít z restaurace. Aplikace samotná je velice rozsáhlá, proto se autor zaměřuje jen na určité typy problému, které mohou při objednání nastat.

5.1 UI, Wireframe a Prototyp

Vzhled aplikace autor navrhoval v prostředí Adobe XD. Po vytvoření kompletního návrhu vzhledu, vytvořil prototyp aplikace, která byla předlohou pro tvorbu aplikace na reálném zařízení. V průběhu vytváření autor použil dostupné zdroje komponentů systému iOS z https://developer.apple.com/design/resources/. Ikony byly vytvořeny ručně a určité prvky přizpůsobeny. Použité fotky jsou dostupné z https://unsplash.com/



Obrázek 19 - Návrh UI kompletní aplikace

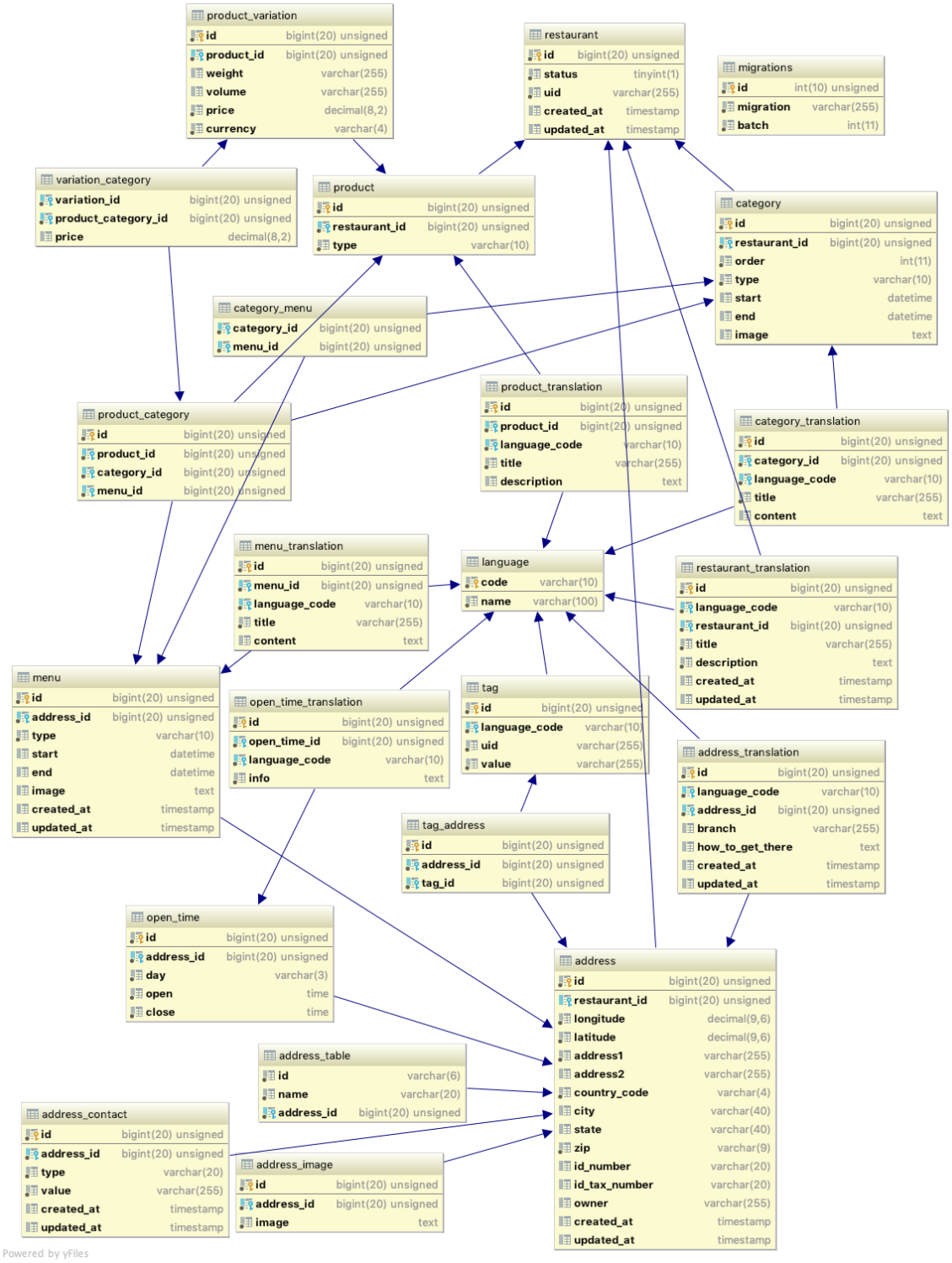
6 BACKEND

Autorská část backendu poskytuje základní zdroj dat pro samotnou aplikaci. Pro testovací účel aplikace je nezabezpečená, ale autor má v plánu vytvořit zabezpečení přes HTTP Basic Auth. Při vydání produkční verze aplikace bude zavedeno šifrované spojení přes HTTPS. Aktuální verze backendu komunikuje přes protokol HTTP a REST API ve formátu JSON.

Pro backend si autor vybral framework PHP Lumen, který vyhodnotil jako nejlepší volbu.

6.1 Struktura databáze

Autor přistupoval k databázi ve snaze minimalizovat možné duplicity v jednotlivých restauracích. Aby toho bylo docíleno, použil franšízový model restaurací, který aplikoval na celou databázi. Model databáze, který demonstruje obrázek 20, autor vytvořil na základě zjištěných požadavků na samotný backend. Při vytváření autor dodržoval migrační třídy pro vytváření tabulek a strukturu se snažil minimalizovat do co nejmenšího schématu databáze.



Obrázek 20 - Grafické znázornění databáze MySQL

6.1.1 Problémy při vytváření databáze

Při vytváření byla databáze několikrát reorganizována z důvodu nesprávného vyložení požadavků. Největší problém nastal při vytváření návaznosti samotného jídelního lístku, kdy se autor snažil zachovat koncepci franšíz.

Jako reálnou představu lze uvést franšízu Potrefená Husa, která je totožná ve všech městech (v databázi tabulka adres), ale liší se malými změnami v jídelním lístku. Obsahově se jídelní lístek liší v určitých jídlech, kategoriích i cenách. Příkladem může být tabulka znázorňující stejný produkt ve stejné kategorii, jen s jiným názvem, rozdílnou gramáží nebo objemem a cenou.

Tabulka 1 - Srovnání cen stejných produktů ve stejné restauraci

Pobočka	Kategorie	Velikost	Produkt	Cena
Zlín	Saláty	0,4kg	SALÁT S CAESAR DRESINKEM	179 Kč
Brno	Saláty	0,3kg	CEASAR SALÁT	189 Kč
Zlín	Polévky	0,33l	Hovězí vývar	59 Kč
Brno	Polévky	0,33l	Silný hovězí vývar	65 Kč

Pokud by autor srovnával jednotlivé restaurace *McDonald's* jenom na území ČR, zjistil by totožné výsledky i u totožných produktů.

Jako finální řešení si autor zvolil přidávat jídla k restauraci, následně vytvořit menu pro každou pobočku (adresu), a poté do tohoto menu vytvořit kategorie. Po vytvoření prázdného menu bez produktů, vložil do kategorií jednotlivé položky, kterým upravil cenu dle adresy.

6.2 Hlavní části backendu

Backend je rozdělen na 3 části. Jedna část se zabývá vytvářením jídelních lístků a dat, druhá pak dodává data potřebné pro aplikaci a třetí zprostředkovává realtime komunikaci s databází Firebase.

6.2.1 Část vytváření restaurací a práce s daty:

Obsahuje samotné vytváření restaurací, adres, kontaktů, otevíracích časů, jídelních lístků, produktů, párování jídelních lístků a produktů a vytváření jazykových mutací. Tato část je

dostupná formou HTTP požadavků POST a PUT, kde metoda POST zapisuje data do databáze a metoda PUT data aktualizuje.

6.2.2 Část obsluhy aplikace:

Aplikace na mobilním zařízení je schopná odesílat požadavky přes protokol HTTP a metodu GET a získat tak potřebná data k zobrazení pro uživatele. Aplikace nabízí 2 dotazy pro získání dat: *požadavek pro získání restaurací na hlavní stránku* a *požadavek pro získání jídelního lístku po načtení QR kódu*. Autor chce v budoucnu aplikaci rozšířit o vyhledávání, hodnocení, nebo komentáře uživatelů.

6.2.3 Část odeslání objednávek:

Pokud zákazník přidá objednávku do svého účtu, je tato objednávka automaticky odeslána na *Firebase realtime database server*, kde čeká na odpověď obsluhy restaurace a její schválení.

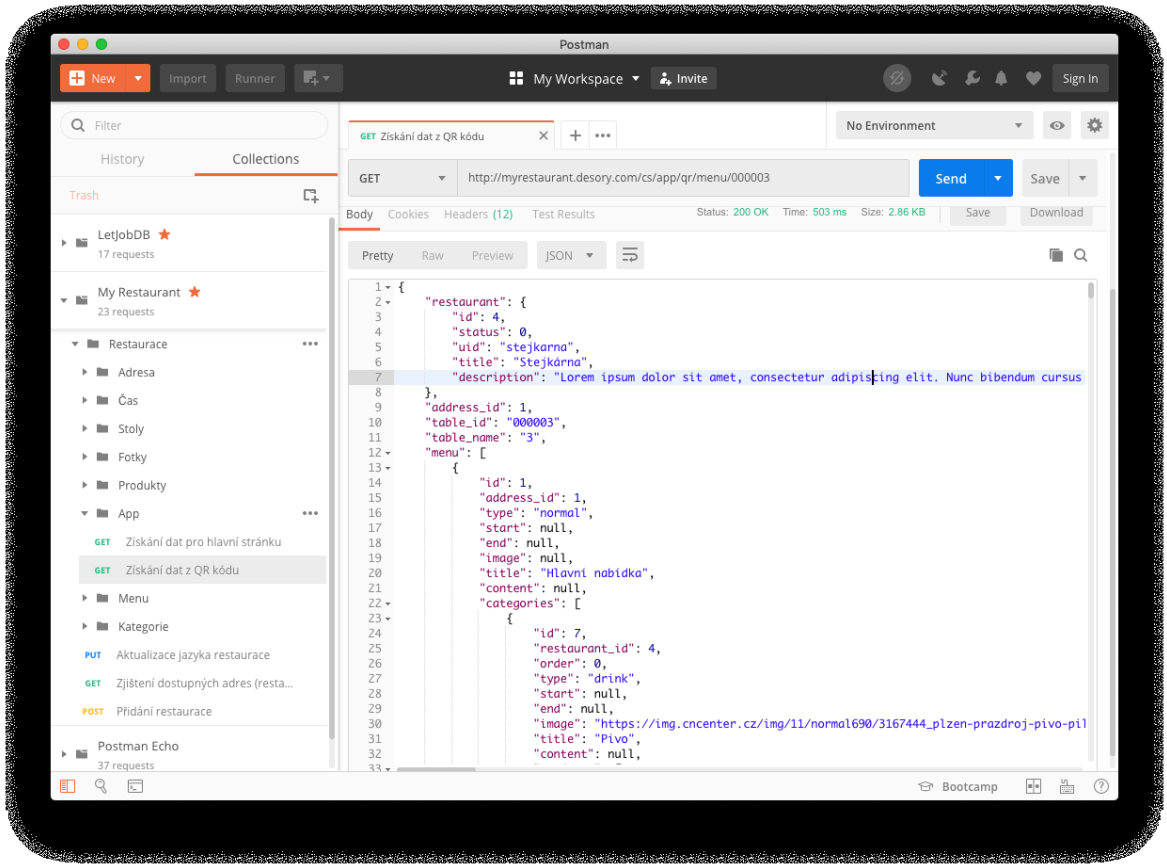
6.3 Testování pomocí Postman

Dostupné endpointy autor testoval pomocí aplikace *Postman*. Obrázek 21 představuje několik endpointů nastavených ve směřovači ve frameworku Lumen.

```
47  // vytvoření menu
48  $router->post('/{locale}/menu/add', 'MenuController@store');
49  $router->post('/{locale}/menu/add/localize', 'MenuController@addLocale');
50
51  // získání menu
52  $router->get('/{locale}/menu/a{address_id}', 'MenuController@getMenu');
53
54  // kategorie
55  $router->post('/{locale}/category/add', 'CategoryController@store');
56  $router->post('/{locale}/category/add/localize', 'CategoryController@addLocale');
57  $router->post('/{locale}/category/add/relation', 'CategoryController@addRelation');
58
59  // přidání produktu do restaurace
60  $router->post('/{locale}/product/add', 'ProductController@store');
61  $router->post('/{locale}/product/add/localize', 'ProductController@addLocale');
62  $router->post('/{locale}/product/add/relation', 'ProductController@addRelation');
63  $router->post('/{locale}/product/add/variation', 'ProductController@addVariation');
64
65
66  // specielní routy pro aplikaci
67  $router->get('/{locale}/app/home/@{latitude},{longitude}/d{radius}', 'AppController@getHomeFeed');
68  // načtení dat z QR kódu
69  $router->get('/{locale}/app/qr/menu/{table_id}', 'QRCodeController@getMenuByTable');
```

Obrázek 21 - Ukázka použitých endpointů a jejich metody

Obrázek 22 ukazuje přístup k jídelním lístkům za pomocí fiktivní funkce QR kódu, která je nahrazena přímým dosazením URL adresy. Vrácená data jsou ve formátu JSON



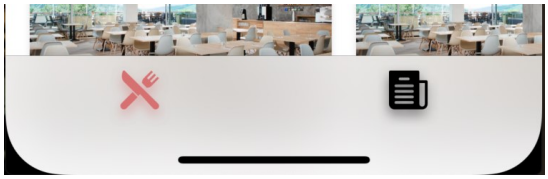
Obrázek 22 - Ukázka použití programu Postman

7 TVORBA APLIKACE

Autor se rozhodl pro tvorbu aplikace bez Storyboards z důvodu rozsáhlosti a přehlednosti aplikace jako celku. Vývoj prototypu probíhal v prostředí AppCode a následně v Xcode z důvodu rychlé indexace kódu. Vypracování aplikace začalo založením projektu, ve kterém si autor vyzkoušel základní komponenty UI a práci v prostředí Xcode a AppCode. Následovalo vytvoření projektu pro rozpoznávání QR kódu, které je popsáno v kapitole 7.3. Po otestování načítání QR kódu byl založen třetí projekt s již požadovaným prototypem aplikace, který začal odstraněním Storyboard souborů a zavedením načítání grafiky z kódu.

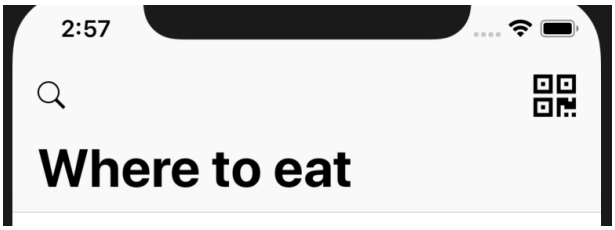
7.1 Vytvoření základní části UI

Pro vytvoření základní logiky aplikace autor vytvořil základní TabBarController, který vytváří jednoduché menu s dvěma položkami: *Hlavní stránka* a *Účet*.



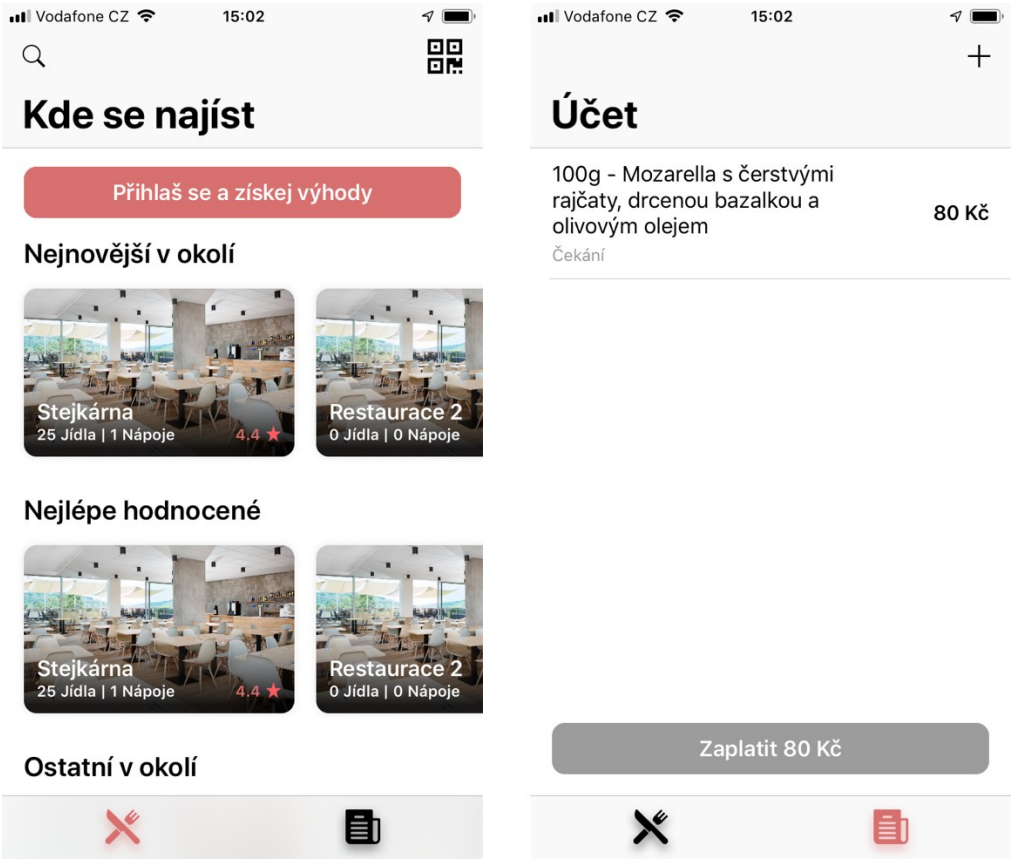
Obrázek 23 - Ukázka hlavního TabBarControlleru

Do jednotlivých Tabů se vytvořil NavigationController, který vytvořil záhlaví stránky s potřebnými tlačítky.



Obrázek 24 - Ukázka navigačního kontroleru

Obsahy stránek jsou vytvořeny za pomocí *UITableViewController* nebo *UICollectionViewController*. Na hlavní stránce se jedná o UICollectionView ve kterém jsou obsažené další UICollectionView a na stránce s účtem se jedná o UITableView.



Obrázek 25 - Ukázka grafiky obou stránek

Při tvorbě grafiky se autor snažil vycházet z návrhu a prototypu v prostředí Adobe XD. Při testování prototypu na reálném zařízení se určité části ukázaly jako nepraktické, a proto je autor upravil tak, aby byly pro uživatele při používání příjemné. Prototyp na reálném zařízení obsahuje jen základní logiku objednávání v restauraci. Do budoucna by měla být aplikace rozšířena o uživatelský profil, hodnocení a recenze restaurací nebo platby přímo v mobilu.

7.2 Napojení na backend

Při vývoji aplikace byl využit framework CoreData, který je v prototypu kompletně nahrazen Firebase databází, která je popsána v bodě 7.3.1. Aplikace vyžívá 2 druhy backendu. Jeden slouží k dodání dat do aplikace jako jsou restaurace a jídelní lístky. Druhý pak ke spojení s obsluhou restaurace a je popsán v bodě 7.3.1. Pro připojení k vlastnímu backendu je dočasně využíváno HTTP spojení bez zabezpečení. Toto řešení je pouze pro testovací účely.

V aplikaci je využíván framework TRON, který zjednodušuje připojení k serveru a získání dat jako objektu z JSON řetezce. Využívá také známé frameworky jako jsou Alamofire nebo SwiftyJSON. Důvodem využívání TRON frameworku je možnost použít generické třídy.

Data získaná ze serveru jsou obsáhlá a je třeba je rozčlenit. V aplikaci proto bylo vytvořeno několik modelových tříd, které reprezentují data obsažená v JSON řetězci. Tato data jsou dále předávána a reprezentována v jednotlivých kontrolérech jako grafické objekty restaurací, telefonního čísla do restaurace a další. Aplikace využívá načtení kompletních dat restaurace i s jídelními lístky, aby nedocházelo k dalšímu načítání dat po otevření samotné restaurace.

7.3 Načítání QR kódu

Pro načítání QR kódu byl vytvořen vlastní projekt, kde se autor seznámil s nativními funkcemi načítání QR kódu. Demonstrace samotné aplikace je na obrázku 26.

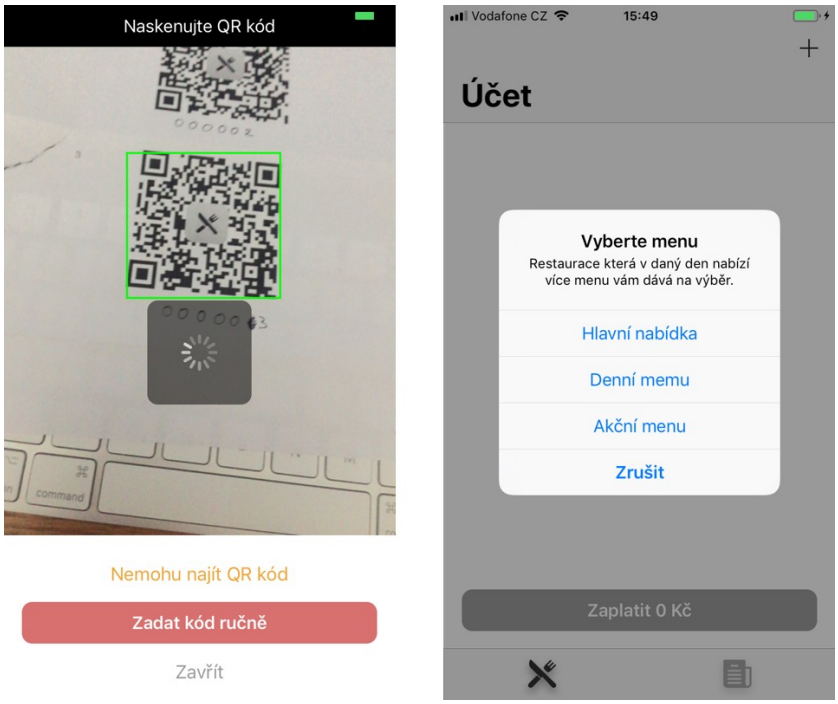
Obrázek 26 - Vlastní aplikace
pro načtení QR kódu

V prototypu aplikace je grafika načtení přizpůsobena původnímu návrhu a samotný QR kód lze načíst dvěma způsoby.

1. Prvním způsobem je načíst QR kód na hlavní stránce kliknutím na ikonu QR kódu v pravém horním rohu. Aplikace sama vyhodnotí, zda je QR kód přidružený aplikaci, ověří, jestli má restaurace aktivní registraci a následně stáhne jídelní lístky restaurace, které nabídne zákazníkovi. Informace k získání jídelního lístku jsou uloženy

v databázi *UserDefaults*. V případě pádu aplikace, uzavření aplikace, je vždy stahován nový jídelní lístek.

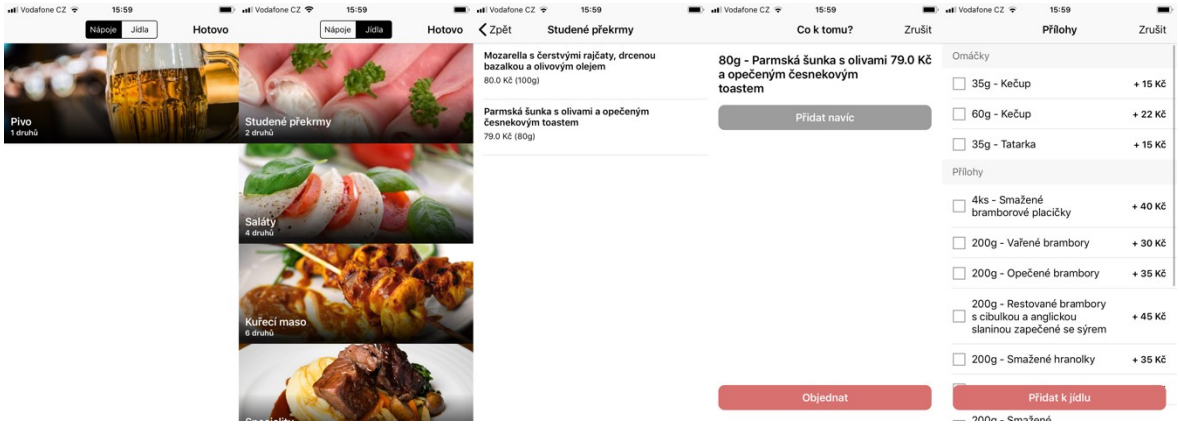
2. Druhým způsobem je načíst QR kód ze stránky Účet, a to kliknutím na tlačítko + v pravém horním rohu. Pokud nemá zákazník naskenovaný kód již dříve. Pokud zákazník QR kód naskenoval, může si kliknutím na tlačítko + vybrat produkt, který si chce objednat. Postup ověření a následující funkce jsou stejné jako v prvním bodě.



Obrázek 27 - Načtení QR kódu a nabídka jídelních lístků

V případě že má restaurace pouze jeden jídelní lístek, otevření tohoto jídelního lístku proběhne automaticky, nebo po kliknutí na tlačítko + na stránce Účet.

Připojení k restauraci a otevření nového účtu u stolu nastává v případě, že zákazník objedná alespoň jednu položku. Pokud uživatel žádný produkt neobjedná, může dále skenovat QR kódy přes hlavní stránku, nebo po ukončení aplikace postupovat podle obou bodů.



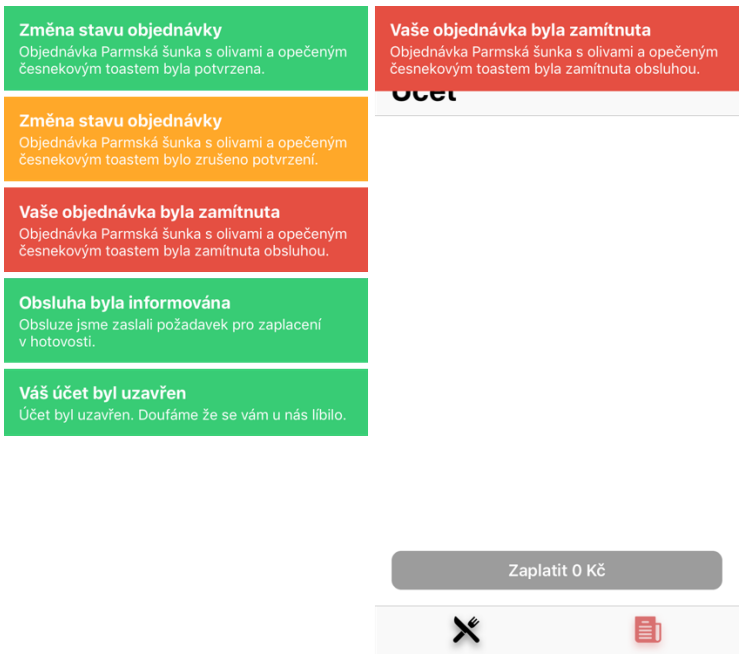
Obrázek 28 - Obrazovky při objednávání

7.3.1 Firebase

Po kliknutí na tlačítko Objednat (obrázek 28) se odesílá požadavek na vytvoření účtu u stolu v restauraci do databáze Firebase. Pokud tento požadavek byl úspěšný, do zařízení se uloží potřebné informace k získání dat v případě pádu nebo náhlého ukončení aplikace a uživatel je informován, že obsluha objednávku vyřizuje.

Obrázek 29 - Struktura databáze Firebase

Pokud obsluha objednávku schválí, uživatel je následně upozorněn přímo v aplikaci o změně objednávky. Tyto in-app notifikace využívají framework NotificationBannerSwift a jsou tak dostupné na všech částech aplikace, bez ohledu na to, v jaké obrazovce se uživatel nachází.



Obrázek 30 – Ukázka In-App notifikací

Notifikace se řídí dle těchto pravidel:

- Změna stavu objednávky z čekání na schváleno
- Změna stavu objednávky ze schváleno na čekání
- Zamítnutí objednávky
- Informace žádosti o platbu
- Uzavření účtu obsluhou po žádosti zákazníka
- Uzavření účtu obsluhou bez žádosti zákazníka

Notifikace mimo aplikaci, budou přidány před uvedením aplikace na AppStore.

7.4 Obsluha aplikace personálem

Před uvedením do AppStore se aplikace rozroste i o verzi pro mobilní Android OS, také bude uvedena aplikace pro obsluhu v restauraci, která je momentálně nahrazena přímým zásahem v databázi Firebase.

7.5 Možnosti dalšího rozšíření aplikace

- **Platba v aplikaci** – prototyp aplikace disponuje možností požádat obsluhu o zaplacení kartou, nebo hotově. Samotné placení touto formou je výhodné pro zákazníky, kteří chtějí urychlit proces placení bez čekání na okolojdoucí personál. V budoucích aktualizacích přibude funkce Apple Pay. Tuto funkci aplikace momentálně nenabízí,

a to z důvodu komplikovanosti napojení všech restaurací a jejich účtů. Platba kartou přes internetovou bránu je naopak zdlouhavá a zbytečně komplikovaná pro zákazníky, proto aplikace tuto možnost nabízet nebude ani v nejbližší době.

- **Registrace uživatelů** – registrace uživatelů přes Facebook nebo vlastní systém je samozřejmostí každé aplikace. Registrovaným uživatelům poté může aplikace nabízet různé slevové akce, hodnocení nabídky nebo seznam účtů. Nedílnou výhodou je záznam uživatele při nedodržení podmínek aplikace a možnost tohoto uživatele kontaktovat v případě problému přímo v aplikaci.
- **Hodnocení podniku a produktů** – každý člověk dobře ví, kam by rád zašel na dobrý oběd, či večeři, v příjemném prostředí. Hodnocení restaurací dá uživatelům jasně najevo kde je co lepší, a pro samotné restaurace přinese zpětnou vazbu, díky které se mohou zlepšit.
- **Ukládání všech mých účtů** – mít výdaje pod kontrolou je problém většiny lidí, a ne všem se daří své výdaje hlídat. Aplikace by mohla ukládat všechny zaplacené a uzavřené účty a uživatel by tak měl přehled o tom kdy a kde kolik utratil. Pokud by se tato funkce skloubila s platbou přímo v telefonu, dokáže šetřit lesy i planetu.
- **Rezervace** – někteří zákazníci si do podniku zatelefonují, jiní se tam staví osobně. Ale rezervace přes telefon kliknutím na jedno tlačítko je pohodlná, rychlá a uživatel nemusí nikam chodit ani volat. Ideální, pokud je náročný pracovní den.
- **Slevy a akce** – slevy má rád každý, proto by akce přímo v aplikaci mohli přilákat do restaurací další zákazníky a zvednout tak tržby.

7.6 Nasazení v restauracích

Nasazení v restauracích autor vyjednává. Momentálně jedná se čtyřmi restauracemi ve Zlíně a samotné nasazení plánuje nejpozději do konce roku 2019.

ZÁVĚR

Cílem této bakalářské práce byl návrh a realizace mobilní aplikace na platformu iOS od společnosti Apple. V úvodní části je popsána tvorba mobilních aplikací pro platformu iOS a multiplatformní vývoj. V další části následuje popis platformy iOS a její charakteristické vlastnosti. Čtenář se také může dočíst informace o používaných backend nástrojích jako jsou: PHP, JavaScript nebo platforma Firebase.

Praktická část se zabývá průzkumem trhu, proč by tato aplikace mohla být užitečná a ve kterých podnicích by mohla najít uplatnění. Zároveň s průzkumem trhu bylo nutné vytvořit seznam funkčních i nefunkčních požadavků, které charakterizují backend a aplikaci. Při návrhu těchto požadavků bylo třeba kontaktovat majitele několika podniků, aby byl systém navržen správně a kvalitně.

Po návrhu funkčních a nefunkčních požadavků byl vytvořen grafický návrh aplikace se všemi funkcemi, které budou postupně do aplikace implementovány. Vytvoření prvního prototypu probíhalo v prostředí Adobe XD, kde si autor vyzkoušel jednotlivé návaznosti s jednoduchou logikou aplikace, aby zamezil co nejvíce vadám při využívání aplikace dalšími uživateli.

Když byl návrh grafické části dokončen, bylo potřeba vytvořit backend pro základní funkčnost aplikace. Autor si pro backend vybral framework Lumen, ve kterém byly vytvořeny migrační třídy, kontrolery, modelové třídy a endpointy pro REST API. Databáze byla upravena na vícejazyčnou a otestována přes vlastní SQL dotazy. Během vývoje byla striktně dodržována pravidla frameworku, aby se dala aplikace rychle a efektivně rozšiřovat. Každý z vytvořených endpointů byl otestován v prostředí Postman, aby nedošlo k omylům při implementaci API v aplikaci a proces vývoje byl rychlejší.

Aplikace byla vytvářena z několika částí, při kterých si autor vyzkoušel několik možných postupů a implementací kódu. Nejdříve byla vytvořena testovací aplikace pro seznámení se s jazykem Swift a základními grafickými elementy. Následovala aplikace pro načítání QR kódu, která využívá nativní funkce kamery a dále byl založen projekt se samotným prototypem objednávací aplikace, ve které autor použil získané znalosti z předchozích dvou aplikací a aplikaci načítáni QR kódu do této aplikace integroval.

Po vytvoření aplikace, ale i při vytváření, byla aplikace testována. Testování probíhalo formou simulátoru v počítači na několika virtuálních zařízeních. Především pak na samotném

zařízení iPhone, ve kterém se testovaly finální funkce využívající QR kód. Testování se zúčastnilo 20 dotázaných, s převahou rodinných příslušníků. Během testování byly zjištěny některé nedostatky, díky kterým bylo možné aplikaci upravit tak, aby byla pro většinu testovaných přívětivá.

V samotném závěru je pojednáno o dalších možnostech rozšíření aplikace a jejím využití. Rozsah aplikace je natolik obsáhlý, že všechny naplánované funkce zde nešly zahrnout a budou postupně přidávány po nasazení v App Store, stejně jako vytvoření aplikace pro platformu Android. Sám autor věří že tato aplikace má budoucnost a na tomto projektu bude dále pokračovat.

SEZNAM POUŽITÉ LITERATURY

[1] ŽAMPACH, Michal. Mobilní aplikace pro děti s poruchou učení. Zlín: Univerzita Tomáše Bati ve Zlíně, 2017, 69s. Dostupné také z: http://hdl.handle.net/10563/41127. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav počítačových a komunikačních systémů. Vedoucí práce Vala, Radek.

[2] Podpora Apple [online]. [cit. 2019-04-28]. Dostupné z: https://developer.apple.com/support/app-store/

[3] Statcounter: GlobalStats [online]. [cit. 2019-04-28]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201903-201903-bar

[4] VESELÝ, Dominik. Programování uživatelského rozhraní na iOS v Ackee (Storyboards vs. Xib vs. kód). Ackee.cz [online]. 12. května 2015 [cit. 2019-04-29]. Dostupné z: https://www.ackee.cz/blog/programovani-uzivatelskeho-rozhrani-na-ios-v-ackee-storyboards-vs-xib-vs-kod/

[5] Swift delegate design pattern. Theswiftdev.com [online]. 27. června 2018 [cit. 2019-04-29]. Dostupné z: https://theswiftdev.com/2018/06/27/swift-delegate-design-pattern/

[6] Core Location: Obtain the geographic location and orientation of a device. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/corelocation

[7] UIKit: Construct and manage a graphical, event-driven user interface for your iOS or tvOS app. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/uikit

[8] UIView: An object that manages the content for a rectangular area on the screen.. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/uikit/uiview

[9] UILabel: A view that displays one or more lines of read-only text, often used in conjunction with controls to describe their intended purpose. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/uikit/uilabel

[10] UICollectionView: An object that manages an ordered collection of data items and presents them using customizable layouts. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/uikit/uicollectionview

[11] UIImageView: An object that displays a single image or a sequence of animated images in your interface. Apple [online]. [cit. 2019-04-29]. Dostupné z: https://developer.apple.com/documentation/uikit/uiimageview

[12] Lekce 1 - Úvod do Node.js. Itnetwork.cz [online]. [cit. 2019-04-30]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs

[13] Laravel. Wikipedia.org [online]. [cit. 2019-04-30]. Dostupné z: https://cs.wikipedia.org/wiki/Laravel

[14] MIŠTOVÁ, Jaroslava. Firebase. Ackee.cz [online]. 29. října 2018 [cit. 2019-04-30]. Dostupné z: https://www.ackee.cz/blog/glossary/firebase/

[15] Software Engineering, Business Continuity, and Education: International Conferences, ASEA, DRBC and EL 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, in Conjunction with GDC 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings [online]. 2. 12. 2011. Springer, 2011 [cit. 2019-05-06]. ISBN 9783642272073. Dostupné z: https://books.google.cz/books?id=IK4SBwAAQBAJ&dq=mobile+device+fragmentation&hl=cs&source=gbs_navlinks_s

[16] IPhone History: A Timeline From 2007-2019. History cooperative [online]. [cit. 2019-05-06]. Dostupné z: https://historycooperative.org/the-history-of-the-iphone/#First_iPhone_Release_Date_June_29_2007

[17] IOS (Apple). Wikipedia [online]. [cit. 2019-05-06]. Dostupné z: https://cs.wikipedia.org/wiki/IOS_(Apple)

[18] The Basics. Swift.org [online]. [cit. 2019-05-07]. Dostupné z: https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html

[19] UITableView: A view that presents data using rows arranged in a single column. Apple [online]. [cit. 2019-05-07]. Dostupné z: https://developer.apple.com/documentation/uikit/uitableview

[20] UIButton: A control that executes your custom code in response to user interactions. Apple [online]. [cit. 2019-05-07]. Dostupné z: https://developer.apple.com/documentation/uikit/uibutton

[21] UIAlertController: An object that displays an alert message to the user.. Apple [online]. [cit. 2019-05-07]. Dostupné z: https://developer.apple.com/documentation/uikit/uialertcontroller

[22] UIAlertController: An object that displays an alert message to the user.. Apple [online]. [cit. 2019-05-07]. Dostupné z: https://developer.apple.com/documentation/uikit/uialertcontroller

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
GUI	Graphical User Interface
JSON	JavaScript Object Notation
REST	Representational state transfer
SOAP	Simple Object Access Protoco
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
OS	Operační Systém
VPS	Virtuální privátní server

SEZNAM OBRÁZKŮ

Obrázek 1 - Procentuální využití mobilních operačních systémů [3] 11
Obrázek 2 - Procentuální podíl verzí systému iOS [2] 13
Obrázek 3 - Rozšíření třídy UIColor 20
Obrázek 4 – Prezentace 20
Obrázek 5 - Definice proměnných 21
Obrázek 6 - Label zobrazující text s atributy [9] 22
Obrázek 7 – Ukázka UITableView [19] 23
Obrázek 8 - CollectionView využívající flow layout [10] 24
Obrázek 9 - Ukázka UIButton [20] 24
Obrázek 10 - Ukázka UIAlertController [22] 25
Obrázek 11 - Zobrazení scaleToFill [1] 26
Obrázek 12 - Zobrazení aspectFit [1] 26
Obrázek 13 - Zobrazení aspectFill s clipToBounds [1] 27
Obrázek 14 - Průzkum trhu Kde 32
Obrázek 15 - Průzkum trhu Jak často 32
Obrázek 16 - Průzkum trhu Jak dlouho 33
Obrázek 17 - Průzkum trhu Jak moc 33
Obrázek 18 - Průzkum trhu Odchod 33
Obrázek 19 - Návrh UI kompletní aplikace 37
Obrázek 20 - Grafické znázornění databáze MySQL 39
Obrázek 21 - Ukázka použitých endpointů a jejich metody 41
Obrázek 22 - Ukázka použití programu Postman 42
Obrázek 23 - Ukázka hlavního TabBarControlleru 43
Obrázek 24 - Ukázka navigačního kontroleru 43
Obrázek 25 - Ukázka grafiky obou stránek 44
Obrázek 26 - Vlastní aplikace pro načtení QR kódu 45
Obrázek 27 - Načtení QR kódu a nabídka jídelních lístků 46
Obrázek 28 - Obrazovky při objednávání 47
Obrázek 29 - Struktura databáze Firebase 47
Obrázek 30 – Ukázka In-App notifikací 48

SEZNAM TABULEK

Tabulka 1 - Srovnání cen stejných produktů ve stejné restauraci.....40

SEZNAM PŘÍLOH

P1 CD s diplomovou prací a soubory obsahující zdrojové kódy