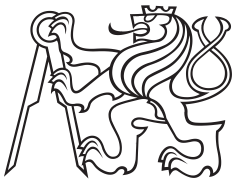
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Aplikace pro podporu hosta v restauraci

Prokop Umlášek

Vedoucí: Doc. Ing. Ivan Jelínek, CSc.
Studijní program: Sofwarové inženýrství a technologie
Leden 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Umlášek** Jméno: **Prokop** Osobní číslo: **499358**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro podporu hosta v restauraci

Název bakalářské práce anglicky:

Restaurant guest support application

Pokyny pro vypracování:

Cílem bakalářské práce je implementace aplikace pro podporu hosta v restauraci. Podpora hosta umožní rezervaci místa v restauraci, načtení aktuální nabídky restaurace pomocí QR kódu, vytvoření objednávky, přivolání obsluhy, integraci na pokladní systém restaurace, simulaci skutečné platby pro pozdější použití.
Proveďte rešerši stávajících řešení, proveďte rozbor funkčnosti těchto aplikací.
Identifikujte slabé a silné stránky těchto řešení a závěry rešerše vyhodnoťte.
Navrhněte strukturu aplikace,
Proveďte rešerši vhodných implementačních prostředků a vyhodnoťte, uvažte Technologie React Native.
Stanovte a popište postup implementace, uvažte použití technologie GraphQL, pro načtení QR kódu technologii Expo BarCodeScanner
Implementujte klientskou stranu aplikace a propojte ji se serverovou stranou (pokladní systém restaurace), zvažte použití systému Dotykačka
Navrhněte testovací scénáře, proveďte vhodné testy, testy vyhodnoťte

Seznam doporučené literatury:

Introduction to API v2 - API. Introduction to API v2 - API [online]. © 2021 Dotykačka ČR s.r.o . Dostupné z: https://docs.api.dotypos.com/
Dotykačka [online]. © 2021 Dotykačka ČR s.r.o . Dostupné z: https://dotykacka.cz/
Expo. Expo [online]. Copyright ©. Dostupné z: https://expo.dev/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Ivan Jelínek, CSc. kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.02.2023** Termín odevzdání bakalářské práce: **09.01.2024**

Platnost zadání bakalářské práce: **22.09.2024**

doc. Ing. Ivan Jelínek, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Tímto bych chtěl poděkovat všem, kteří mi pomohli při tvorbě mé závěrečné práce. Bez Vaší podpory by tato práce nikdy nemohla vzniknout.

Rád bych poděkoval Doc. Ing. Ivanu Jelínkovi, CSc., pod jehož vedením jsem tuto práci sepsal.

Dále bych chtěl poděkovat panu Tomáši Martincovi a společnosti Dotykačka ČR s.r.o. za vstřícnou komunikaci a ochotu mi poskytnout podporu u mé bakalářksé práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

Podpis autora práce .....

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací mobilní aplikace pro podporu hostů v restauracích. Cílem práce bylo vytvořit uživatelsky přívětivou aplikaci, která zefektivní procesy rezervace, objednávání a platby v restauračních zařízeních. Aplikace byla vyvíjena s využitím moderních technologií, jako jsou React Native, Apollo Client a GraphQL, a integrována s pokladním systémem Dotykačka. V práci je kladen důraz na analytický přístup k návrhu, důkladnou rešerši existujících řešení a efektivní řešení výzev spojených s integrací systémů. Testování aplikace ukázalo její spolehlivos a funkčnost.

Klíčová slova: mobilní aplikace, restaurace, rezervace a objednávání, pokladní systémy, React Native, Apollo Client, GraphQL, Dotykačka, integrace systémů

Vedoucí: Doc. Ing. Ivan Jelínek, CSc. Karlovo náměstí 13, Prague 2, 121 35

Abstract

This bachelor's thesis focuses on the design and implementation of a mobile application to support guests in restaurants. The aim was to create a user-friendly application that streamlines reservation, ordering, and payment processes in dining establishments. Developed using modern technologies such as React Native, Apollo Client, and GraphQL, the application was integrated with the Dotykačka cashier system. The thesis emphasizes an analytical approach to design, thorough research of existing solutions, and effective problem solving in system integration. Application testing demonstrated its reliability and functionality.

Keywords: mobile application, restaurant, reservation and ordering, cash register systems, React Native, Apollo Client, GraphQL, Dotykačka, system integration

Title translation: Restaurant guest support application

Obsah

1 Úvod 1

2 Rešerše existujících přístupů k řešení 3

2.1 Obecná rešerše . . . 3
2.2 Aplikace s podobnou funkcionalitou . . . 4
2.2.1 Systémy pro větší množinu různých restaurací . . . 4
2.2.2 Systémy pro konkrétní prodejny 6
2.3 Závěr rešerše . . . 6

3 Popis struktury aplikace 9

3.1 Hlavní funkce aplikace . . . 9
3.1.1 Rezervace místa u stolu . . . 9
3.1.2 Načtení QR kódu s nabídkou menu . . . 10
3.1.3 Vytvoření objednávky . . . 10
3.1.4 Přivolání obsluhy . . . 10
3.1.5 Zaplacení objednávky . . . 11
3.1.6 Integrace na pokladní systém restaurace . . . 11
3.2 Struktura aplikace . . . 11
3.2.1 Component diagram . . . 12
3.2.2 Use case diagram . . . 13
3.2.3 Flow diagramy . . . 14
3.3 Návrh uživatelského rozhraní . . . 16
3.4 Požadavky aplikace . . . 16

4 Implementační prostředky 17

4.1 Rešerše prostředků pro vývoj mobilních aplikací . . . 17
4.1.1 React Native . . . 17
4.1.2 Kotlin . . . 18
4.1.3 Flutter . . . 19
4.1.4 Ionic Framework . . . 19
4.1.5 Xamarin . . . 20
4.1.6 Cordova . . . 21
4.1.7 Závěr rešerše . . . 21
4.2 Další využité implementační prostředky . . . 22
4.2.1 Expo . . . 22
4.2.2 Expo BarCodeScanner . . . 22
4.2.3 Apollo Client . . . 22
4.3 GraphQL . . . 22

5 Postup implementace 25

5.1 Pokladní systém Dotykačka . . . 25
5.1.1 Pobočka v systému a autorizace . . . 26
5.1.2 Vystavené rozhraní systému Dotykačka . . . 26
5.2 Další kroky implementace . . . 27
5.2.1 Struktura aplikace v React Native . . . 27

6 Implementace 29

6.1 App.js . . . 29
6.1.1 Importy . . . 29
6.1.2 Konfigurace Apollo Client . . . 30
6.1.3 Navigační struktura . . . 31
6.2 GetAccessToken . . . 31
6.2.1 Importy . . . 32
6.2.2 GraphQL mutace . . . 32
6.2.3 Práce s access tokenem . . . 33
6.3 HomeScreen . . . 34
6.4 ReservationFormScreen . . . 34
6.5 SelectTableScreen . . . 35
6.5.1 Komponenta Table . . . 36
6.6 GetReservation . . . 36
6.7 ReservationSummaryScreen . . . 37
6.8 CreateReservation . . . 38
6.9 ScanCodeScreen . . . 38
6.9.1 Importy . . . 38
6.9.2 Dovolení o použití kamery . . . 39
6.9.3 Použití BarCodeScanner . . . 39
6.10 OrderMenuScreen . . . 39
6.11 CreateCustomer . . . 40
6.12 Výsledné grafické zpracování implementace . . . 41
6.13 Souhrn implementace . . . 45

7 Testování 47

7.1 Testovací scénáře . . . 47
7.1.1 Rezervace stolu . . . 47
7.1.2 Vytvoření objednávky . . . 49
7.1.3 Přivolání obsluhy . . . 51
7.2 Unit testy . . . 52

8 Závěr 55

Zdroje 57

Obrázky

3.1 Component diagram 12
3.2 Use case diagram 13
3.3 Flow diagram: Rezervace 14
3.4 Flow diagram: Vytvoření objednávky 15
3.5 Flow diagram: Přivolání obsluhy 15

6.1 Domácí obrazovka 41
6.2 Obrazovka s rezervačním formulářem 42
6.3 Obrazovka k vybrání stolu 43
6.4 Obrazovka po načtení QR kódu 44

Úryvky kódu

6.1 App.js: importy 29
6.2 App.js: konfigurace Apollo Client 30
6.3 App.js: navigační struktura 31
6.4 App.js: GetAccessToken 32
6.5 GetAccessToken.js: importy 32
6.6 GetAccessToken.js: GraphQL mutace 32
6.7 GetAccessToken.js: funkce handleSignIn 33
6.8 HomeScreen.js 34
6.9 ReservationFormScreen.js: příklad odeslaných dat 35
6.10 SelectTableScreen.js: funkce komponenty 35
6.11 SelectTableScreen.js: komponenta Table 36
6.12 GetReservation.js: GraphQL mutace 36
6.13 ReservationSummaryScreen.js: funkce komponenty 37
6.14 CreateReservation.js: GraphQL mutace 38
6.15 ScanCodeScreen.js: importy . . . 38
6.16 ScanCodeScreen.js: dovolení o použití kamery 39
6.17 ScanCodeScreen.js: použití BarCodeScanner 39
6.18 CreateCustomer.js: GraphQL mutace 40

7.1 PayOrder.test.js: test mutace placení 52
7.2 ReservationFormScreen.test.js: test v rezervačním formuláři na nevyplněné jméno 53

Kapitola 1
Úvod

Cílem této bakalářské práce je navrhnout a implementovat mobilní aplikaci, která má poskytnout návštěvníkům restaurací komplexní podporu a zároveň zefektivnit procesy rezervace, objednávky a platby. Tato aplikace však není zaměřena pouze na pohodlí hostů, ale také na optimalizaci pracovních postupů samotné restaurace a zjednodušení práce personálu. Aplikace se zaměří na podporu hostů od jejich příchodu až po odchod z restaurace.

Kromě samotné implementace aplikace se také zaměříme na rešerši stávajících řešení v oblasti podpory hostů v restauraci. Provedeme rozbor funkčnosti těchto aplikací a identifikujeme jejich slabé a silné stránky. Závěry této rešerše nám poskytnou nezbytné informace pro návrh optimální struktury naší aplikace.

V projektu se budeme věnovat rezervaci místa v restauraci, což umožní návštěvníkům vytvořit si rezervaci předem a zajistit si tak pohodlné místo s potřebným počtem míst v prostoru restaurace. Dalším klíčovým prvkem bude možnost načtení aktuální nabídky restaurace prostřednictvím QR kódu. To umožní hostům rychlý a snadný přístup k informacím o jídle a nápojích, což povede jednoduchému a srozumitelnému vytvoření objednávky.

Aplikace bude zahrnovat funkci vytváření objednávky a možnost platby přes mobilní zařízení. To nejenže urychlí proces obsluhy, ale zároveň přinese pohodlí a flexibilitu pro hosty, kteří si budou moci objednat a zaplatit za své pokrmy přímo z pohodlí svého stolu. Vytvořená aplikace bude též podporovat přivolání obsluhy prostřednictvím interaktivních prvků, což zajistí okamžitou komunikaci mezi hosty a personálem restaurace.

Další část práce bude věnována technickým aspektům implementace, kde provedeme rešerši vhodných implementačních prostředků. Uvažujeme o využití technologie React Native pro mobilní aplikaci a technologie GraphQL pro efektivní načítání dat. Pro skenování QR kódu zvolíme technologii Expo BarCodeScanner, která zajistí spolehlivý a rychlý přístup k nabídce restaurace. Aplikace bude integrována na pokladní systém restaurace Dotykačka, s ohledem na možnosti a výhody tohoto systému.

Je nutné podotknout, že se nejedná o doopravdovou pobočku restaurace, ale o simulovanou právě přes "branch"v systému Dotykačka.

Celkový postup implementace bude v práci popsaný a zdokumentovaný. Součástí práce budou také testovací scénáře a hodnotící testy, které nám

pomohou ověřit funkčnost a spolehlivost vytvořené aplikace.

Kapitola 2

Rešerše existujících přístupů k řešení

2.1 Obecná rešerše

Existuje mnoho řešení automatizovaných procesů, které se dnes používají v restauracích. Téměř každý podnik používá pokladní systém, který má mnoho funkcí. Značná část restaurací v České republice přešla k těmto systémům, když po vzoru několika dalších zemí v Evropské Unii česká vláda v roce 2016 zavedla Elektronickou evidenci tržeb (EET). To znamenalo povinnost obchodníků evidovat své tržby online a reportovat je státní správě. Automatické pokladní systémy toto podporovaly a to přestavovalo pro mnoho obchodníků důvod je zavést. [1] [2] [3] [4]

I přes to, že EET je dnes zrušena, restaurace nadále využívají těchto systémů. Hlavním důvodem je automatizace a zlepšení efektivity práce. Toto je samozřejmě i důvodem, proč čím dál tím víc obchodníků k tomuto řešení přistupuje. [1] [2] [3] [4]

Prodejci pokladních systémů nenabízejí jen software ale i hardware. Mezi jejich sortiment patří: digitální obrazovky se zabudovaným systémem, pokladní zásuvky, čtečky čárových kódů, tablety, tiskárny účtenek, mobilní platební terminály, váhy a další příslušenství používané k provozu prodejny. Tyto systémy jsou dnes již tak vyspělé, že mohou fungovat bez připojení na wifi službu.[5] [6] [10] [11] [12]

Takovéto robustní systémy podporují obrovské množství funkcionalit. Mezi ně patří například: platba, zobrazení mapy stolů, tisk účtenek, vystavování faktur, správa uživatelů, přehled tržeb, vzdálená správa aplikace, evidence plateb atd. [5] [6] [10] [11] [12]

Je zjevné, že tyto masivní systémy, které jsou běžně používané většími podniky na podporu a zlepšení efektivity, jsou orientované více na stranu použití obchodníka a jeho zaměstnanců než na stranu zákazníků. Myšleno tak, že tento systém je využíván zaměstnanci restaurace, nikoliv jejími zákazníky. Například i funkcionalita "zobrazení mapy stolů"je myšlena pro personál. Číšník díky této funkcionalitě ví, u jakého stolu má obsluhovat.

Systémy, na něž jsem se výše v této kapitole zaměřil, jsou především navrženy tak, aby splňovaly potřeby interního provozu v restauracích. Tyto pokladní systémy jsou vytvořeny s ohledem na efektivní řízení objednávek, evidenci plateb a správu zásob, což odpovídá funkcím, které by měla mít i

aplikace, na niž se má mé řešení pomocí poskytovaného rozhraní integrovat. Jejich zaměření je výrazně orientováno na poskytování nástrojů pro vnitřní operace restauračního podniku.

Na rozdíl od systémů zaměřených na interní procesy se mé řešení bude specificky orientovat na potřeby zákazníka.

2.2 Aplikace s podobnou funkcionalitou

Aplikace, kterým se tato sekce věnuje, patří do kategorie, kterou bychom obecně nazývali "aplikace pro zákazníky v restauračních zařízeních". Tyto aplikace jsou navrženy s ohledem na pohodlí klientů restaurací a slouží k interakci s obsluhou na základě jejich individuálních preferencí.

Rozdělil bych je do dvou hlavních skupin. První skupina zahrnuje aplikace, které poskytují systémy umožňující restauracím propojit své služby a zapojit se do širší sítě, která sdružuje více restaurací pod společnými službami a funkcionalitami.

Druhou skupinou jsou pak aplikace, které jsou vytvořeny na míru pro konkrétní pobočky nebo řetězce s podobnými prodejnami. Tyto aplikace mohou být navrženy s cílem plně se integrovat se specifickým prostředím dané restaurace a poskytovat personalizované služby pro zákazníky.

2.2.1 Systémy pro větší množinu různých restaurací

V této podsekci se zaměřím na aplikace, které nabízejí služby pro větší množinu různých restaurací. Tyto systémy umožňují restauracím připojit se ke sdíleným platformám, což může zahrnovat služby pro online objednávky, doručování a další.

UberEats

Jedním příkladem může být velmi oblíbená a rozšířená služba UberEats, která umožňuje restauracím po celém světě rozšířit svůj dosah. Tato platforma, aktivní ve mnoha zemích, nabízí restauracím příležitost oslovit nové skupiny zákazníků na mezinárodní úrovni. Od roku 2020 společnost Uber ukončila provoz služby UberEats v České republice. Nicméně tato služba stále funguje v dalších zemích po celém světě. [13] [14]

UberEats poskytuje platformu pro online objednávky, což umožňuje zákazníkům jednoduše a rychle si objednat jídlo z mnoha restaurací v okolí. Pro restaurace představuje tato služba příležitost oslovit nové zákazníky. Možnost doručování poskytuje zákazníkům pohodlí, zatímco restauracím umožňuje rozšířit své služby o doručování jídla na pracoviště, domov nebo jiná místa podle přání zákazníka. [14]

Hodnocení a recenze od zákazníků jsou klíčové pro vytváření reputace a důvěry mezi uživateli platformy. UberEats také nabízí personalizované nabídky, které reflektují preference zákazníků a jejich historii objednávek.

Pro restaurace jsou k dispozici nástroje pro efektivní správu objednávek a sledování vývoje poptávky. [14]

Celkově UberEats přináší nejen snadný přístup zákazníků k různým restauracím, ale také nové možnosti pro rozvoj a posílení podnikání restaurací. [14]

I když se platforma UberEats v mnoha ohledech liší od aplikace, na níž bych chtěl pracovat v této práci, považuji za důležité ji zmínit, neboť hraje zásadní roli v online objednávání na trhu. Je jedním z mnoha podobných systémů, které jsou velmi moderní a často používané hlavně ve velkých městech. Mezi další známé platformy patří například Deliveroo, Just Eat, nebo Glovo, které rovněž poskytují podobné služby, ačkoliv s různými přístupy a funkcionalitami. [14] [15]

Qerko

Nyní se podívejme na další službu, Qerko, která přináší snadný způsob platby do gastronomie. Qerko umožňuje hostům pohodlně platit a rozdělit účet přímo z mobilu, což šetří čas obsluze a umožňuje efektivnější obsluhu více hostů. Pro podniky znamená Qerko také zvýšení tržeb a zisky díky rychlejšímu odbavování a vyšším spropitným od hostů. [20]

Služba také poskytuje digitální menu s fotografiemi jídel, což zvyšuje atraktivitu nabídky pro hosty. Systém funguje pomocí QR kódů, což umožňuje hostům objednávat přes mobil a zrychlit tak proces obsluhy. Qerko rovněž nabízí gastronomický průvodce a mapu podniků, což je pro restaurace další možnost, jak se prezentovat a zvýšit svou viditelnost v rámci gastronomické komunity. [20]

Qerko umožňuje restauracím efektivně využívat kapacity a naplňovat stoly pomocí integrovaného rezervačního systému. Restaurace mohou zdarma využívat online rezervační systém, který je k dispozici buď přímo v Qerko aplikaci nebo na rezervační webové stránce poskytnuté Qerkem. Pro snadnou správu rezervací slouží automatický rezervační asistent, který informuje o nových rezervacích prostřednictvím telefonních hovorů a e-mailů. Kromě toho získají restaurace od Qerka unikátní odkaz na rezervační systém, který mohou integrovat na své webové stránky nebo sdílet na sociálních sítích. Tím zlepšují přívětivost vůči zákazníkům, kteří preferují online rezervace, a zajišťují jednoduchý přístup ke stolům přes Qerko aplikaci nebo přímo na webových stránkách podniku. [20] [21]

Qerko je česká mobilní aplikace a nese výrazné znaky, kterými se podobá té, na které se zaměřuje tato práce. Tato platforma se odlišuje tím, že není zaměřena pouze na konkrétní prodejny, ale funguje na mnoha pobočkách různých restaurací. [20]

Silné a slabé stránky

Výhody aplikací v této kategorii zahrnují zvýšení dosahu a možnost oslovit nové zákazníky na globální úrovni. Dále umožňují restauracím rozšíření svých služeb o online objednávky a doručování, což může přinést další příležitosti

pro růst podniku. Díky tomu, že jsou zpravidla mobilní, je s těmito aplikacemi snazší pracovat na jakémkoli mobilním zařízení. [14] [15] [20] [21]

Naopak mezi slabé stránky těchto aplikací patří skutečnost, že nejsou stavěné přímo na konkrétní prodejnu. To znamená, že nemohou plně adresovat specifické potřeby každé restaurace a mohou mít omezenou schopnost přizpůsobit se individuálním preferencím či strategiím konkrétního podniku. [14] [15] [20] [21]

2.2.2 Systémy pro konkrétní prodejny

Aplikace podobnější té, kterou budu v této práci navrhovat, jsou v restauracích vzácnější. V době, kdy jsem začínal psát tuto práci jsem se setkal pouze se dvěmi. Konkrétně to byly: Cafe Prostoru_v Praze a Luční Bouda v Peci pod Sněžkou. Od té doby jsem se ovšem setkal s pár dalšími, což poukazuje na rostoucí popularitu této formy komunikace se zákazníky. [16] [17]

Tyto aplikace typicky umožňují platbu v mobilním telefonu, načítání menu a vytváření objednávek pomocí QR kódu umístěného na stolech v restauraci. Všechny takovéto aplikace, se kterými jsem se setkal, byly implementovány jako webové aplikace.[16] [17]

Mobilní aplikace pro podporu hosta v restauraci samozřejmě existují. Nabízejí stejné funkce jako mnou zmíněné systémy v restauracích, které jsem navštívil. Navíc jsem našel i aplikace, které podporují i další funkcionality. Například: vystavení digitální účtenky, integrace na kasové systémy, hodnocení jídel i obsluhy. Takovéto aplikace bych ale zařadil spíše do sekce 2.2.1, protože jsem se nikdy nesetkal s takovouto aplikací přímo pro jednotlivou restauraci. [18] [19]

Nepodařilo se mi také nalézt aplikaci, která by se integrovala na stejný pokladní systém, který jsem si vybral já. Je velmi pravděpodobné, že nějaká taková aplikace již existuje. Nejspíše se ale jedná právě o aplikaci navrženou přímo pro konkrétní restauraci a nejedná se o aplikaci dostupnou na trhu. Podobně to bylo i s dvěma systémy, se kterými jsem se setkal právě v Praze a Krkonoších.[16] [17]

Silné a slabé stránky

Silné stránky těchto aplikací spočívají v tom, že jsou navrženy přímo pro konkrétní prodejnu, což umožňuje lepší přizpůsobení se specifickým potřebám daného podniku. [16] [17] [18] [19]

Naopak mezi slabé stránky patří skutečnost, že většina těchto aplikací je stavěná jako webové, což znamená, že je nutné je spouštět prostřednictvím internetového prohlížeče. [16] [17] [18] [19]

2.3 Závěr rešerše

V průběhu této rešerše jsem zkoumal existující přístupy k řešení automatizovaných procesů v restauracích, se zaměřením na aplikace poskytující služby

pro zákazníky v restauračních zařízeních. Obecná rešerše ukázala, že v dnešní době jsou restaurace téměř univerzálně vybaveny pokladními systémy s různými funkcionalitami. Tyto systémy nejen usnadňují evidenci tržeb, ale i efektivní řízení objednávek, plateb a správu zásob.

V rámci aplikací s podobnou funkcionalitou, tím jsou myšleny aplikace zaměřené na klienty poodniků, jsem identifikoval dvě hlavní skupiny. První skupina zahrnuje systémy pro větší množinu různých restaurací, které umožňují propojení služeb několika restaurací pod společnými platformami. Příkladem je UberEats, který nabízí online objednávky a doručování, poskytující restauracím možnost oslovit nové zákazníky na mezinárodní úrovni. [14] [15] [20] [21]

Druhou skupinou jsou aplikace zaměřené na konkrétní prodejny. Zde jsem se zaměřil na aplikace, jako je Qerko, která umožňuje hostům snadné placení a interakci s obsluhou prostřednictvím mobilního zařízení. Tyto aplikace jsou navrženy tak, aby lépe vyhovovaly potřebám konkrétní restaurace. [16] [17] [18] [19]

Silné stránky aplikací pro větší množinu restaurací spočívají v zvýšení dosahu, možnosti oslovit nové zákazníky a rozšíření služeb o online objednávky. Naopak slabými stránkami jsou omezené možnosti přizpůsobení se specifickým potřebám každé restaurace. [14] [15] [20] [21]

Aplikace pro konkrétní prodejny mají silnou stránku v lepším přizpůsobení se specifickým potřebám daného podniku. Na druhou stranu mají omezenou dostupnost, protože jsou často navrženy přímo pro jednu restauraci. [16] [17] [18] [19]

Celkově je patrné, že existující přístupy jsou převážně zaměřeny na interní operace restaurací a efektivitu provozu. Mým cílem je navrhnout mobilní aplikaci, která bude pracovat se zákazníky restaurace a která spojuje výhody obou výše zmíněných skupin. Podobně jako aplikace pro konkrétní prodejny (sekce 2.2.2), chci, aby měla schopnost lépe se přizpůsobit specifickým potřebám určité restaurace. Současně by však měla být mobilní, což je charakteristické pro systémy pro větší mnoužinu různých restaurací (sekce 2.2.1), z čehož vychází snadnější použití pro uživatele.

Kapitola 3

Popis struktury aplikace

V následující kapitole Vás detailně seznámím se strukturou navrhované aplikace. Představím základní funkce, které tato aplikace nabídne, a rozeberu klíčové požadavky, které jsou kladeny na její správný provoz. Získáte tak ucelený pohled na celkovou koncepci a architekturu aplikace, což vám pomůže lépe porozumět jejím možnostem a přínosům.

3.1 Hlavní funkce aplikace

V této sekci se budu věnovat hlavním funkcím aplikace. Všechny tyto funkce se orientují na zákazníka v restauraci a jeho použití. Jedinou vyjímkou je sekce 3.1.6, kde Vás uvedu do základů integrace na pokladnu restaurace. Aplikace nebude využívána personálem. Zaměstnancům restaurace poslouží tím, že se zautomatizuje a zrychlí některé procesy a tím pádem jim ušetří čas.

Hlavními funkcemi aplikace jsou: rezervace místa u stolu, načtení QR kódu s nabídkou menu, vytvoření objednávky, přivolání obsluhy, zaplacení objednávky a integrace na pokladní systém restaurace.

3.1.1 Rezervace místa u stolu

Systém bude zákazníkovi, který má nainstalovanou aplikaci, umožňovat vytvořit rezervaci míst u stolu.

Bude to fungovat tak, že si uživatel bude moci v systému zamluvit pouze celý stůl. Aplikace nebude podporovat rezervaci jenom několika míst. Horní hranice počtu stolů, které si lze zamluvit nebude omezená, ale při vytváření rezervace bude možné si zamluvit pouze jeden stůl. Po dokončení procesu rezervace bude mít uživatel možnost si zarezervovat stůl další.

V aplikaci bude proces začínat vybráním možnosti "Create reservation". Uživatel do formuláře vyplní jméno, ke kterému bude rezervace zapsána. Dále vybere čas a dobu, kdy bude chtít restauraci navštívit. Také bude moci vložit nějakou poznámku k rezervaci. Následně se zobrazí mapka stolů v restauraci. U každého stolu bude zapsán počet míst, které je možno si zarezervovat. Pokud je stůl ve vybraný čas již obsazený, je znázorněn šedou barvou a svítí u něj číslice nula. Plánek bude znázorňovat rozložení sezení v restauraci tak,

aby se co nejvěrohodněji blížilo skutečnosti. Následně si uživatel vybere místa, která si chce zarezervovat tím, že klikne na stůl a potvrdí tlačítkem "Confirm".

Pokud vše proběhne v pořádku, zapíše se rezervace do pokladního systému restaurace. Zákazník může o rezervaci zažádat samozřejmě jen v termínu otevírací doby podniku a pod podmínkou, že už si stejná místa na tentýž čas nezamluvil někdo jiný.

Aplikace nebude podporovat zrušení rezervace. Důvodem je to, že do aplikace se nepřihlašuje. Tím pádem by mohl každý uživatel zrušit rezervaci jiného uživatele. Řešením by bylo telefonické zrušení rezervace.

3.1.2 Načtení QR kódu s nabídkou menu

Systém umožní zákazníkovi načíst QR kód (možnost "Scan QR-Code"), který zobrazí aktuální nabídku jídelního lístku. Kódy se budou nacházet na stolech, tak aby si jich zákazníci ihned všimli. U každého stolu bude jiný, protože při jeho naskenování si aplikace zapamatuje číslo stolu, u kterého se nachází. Tímto způsobem poté obsluha bude vědět, jaký stůl má obsluhovat. Po zobrazení menu totiž bude možnost přejít do procesu "vytvoření objednávky".

Systém bude podporovat i funkci zobrazení menu bez načtení QR kódu. Zákazník po té ovšem nebude mít možnost vytvořit objednávku.

Aktuální menu se bude načítat ze systému restaurace, takže každá změna se projeví i v aplikaci pro podporu hosta.

3.1.3 Vytvoření objednávky

Systém umožní zákazníkovi vytvořit objednávku vybráním položek z jídelního lístku. Tento proces navazuje na proces načtení QR kódu s nabídkou menu. Jinak se k němu uživatel nemá možnost dostat. Je tomu tak z důvodu toho, aby si jídlo mohli objednat pouze lidé, kteří se zrovna nacházejí v restauraci a kteří sedí u stolů.

Objednávka bude probíhat obdobně jako například nakupování na e-shopu. Zákazník bude mít něco jako internetový košík, kam bude vkládat jednotlivé vybrané položky z jídelního lístku. Když bude s výběrem spokojený, tak klikne na možnost "dokončit objednávku". Následně bude mít uživatel možnost objednávku zaplatit nebo vybere, že chce platit u pokladny.

Po dokončení objednávky se její obsah pošle do pokladního systému restaurace, tak aby zaměstnanci mohli zákazníka obsloužit.

3.1.4 Přivolání obsluhy

Aplikace bude poskytovat zákazníkovi funkci přivolání obsluhy ke svému stolu. Použít ji budou moci jen ti, kteří načetli QR kód z jednoho ze stolů. Možnost vyběhne zároveň s možností vytvoření objednávky. Je to v aplikaci proto, aby uživatelé, kteří se momentálně v restauraci nenacházejí, nemohli přivolávat obsluhu ke stolům a nemohli této možnosti zneužívat.

Díky načtení QR kódu se do systému uloží číslo stolu, u kterého zákazník přivolávající obsluhu sedí. Není tedy nutné, aby bylo do aplikace ručně vloženo uživatelem.

3.1.5 Zaplacení objednávky

Aplikace umožní zákazníkovi zaplatit své vybrané položky bezprostředně poté, co v aplikaci dokončí objednávku. Může si ale také vybrat variantu, že chce zaplatit u pokladny. Nemusí tedy k zaplacení své objednávky tuto aplikaci použít.

Protože aplikace nebude reálně využívaná, platby budou pouze simulované. Bude to vypadat tak, že pokud se zákazník rozhodne pro platbu přes aplikaci, tak se mu zobrazí 3 možnosti platby. Platba kartou, Apple Pay nebo Google Pay. Zákazník vyplní informace o kartě nebo klikne na jedno z políček Apple Pay nebo Google Pay. Následně možnost potvrdí tlačítkem "Confirm". Aplikace nepracuje s reálnými daty, takže platební údaje jsou redundantní a do pokladního systému se nepropíší. Zaplacení objednávky se ale do systému dostane. Platba může následně být viděna i v tržbách pobočky.

Po zvolení možnosti platby u pokladny se do systému zapisuje pouze objednávka, která čeká na zaplacení.

3.1.6 Integrace na pokladní systém restaurace

Aby byl splněn jeden z klíčových požadavků aplikace, bude nutné provést integraci s pokladním systémem restaurace. Tato integrace umožní propojení mezi klientskou a serverovou stranou. Naše aplikace bude komunikovat s pokladním systémem prostřednictvím vystaveného rozhraní, což umožní nejen odesílání dat do pokladny, ale také jejich čtení. Tímto způsobem zajistíme efektivní a bezproblémový tok informací mezi naší aplikací a centrálním systémem provozovaným v restauraci. Mějte na paměti, že restaurace je pouze simulovaná a nemá reálnou předlohu. Toto bude možné právě díky pokladnímu systému.

3.2 Struktura aplikace

Navrhovaná aplikace je koncipována tak, aby pracovala s několika klíčovými komponentami, které společně tvoří systém. Centrálním prvkem je uživatelské rozhraní, které poskytuje zákazníkům v restauraci přívětivé prostředí pro interakci a provádění různých funkcí.

Serverovou stranu aplikace, představuje restaurační systém, zajišťuje zpracování transakcí a komunikaci s databázovým modulem. Tato část systému je klíčová pro bezpečnou a efektivní interakci mezi klienty a backendem.

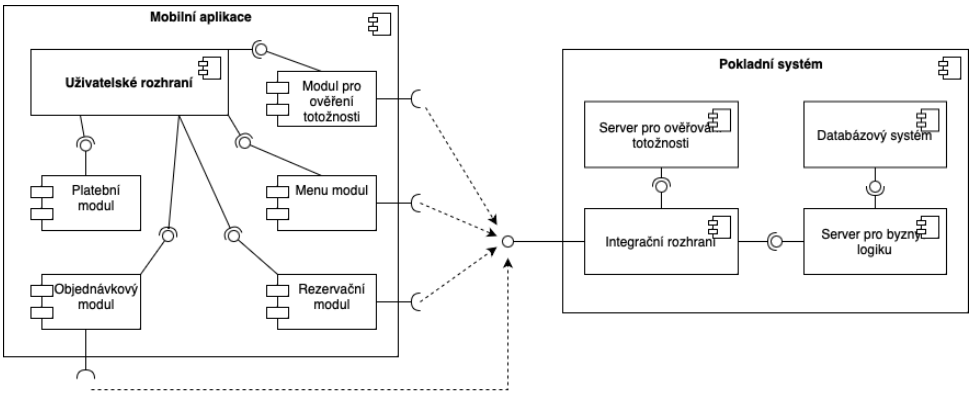
Databázový modul slouží jako úložiště pro data, včetně informací o rezervacích, aktuální nabídce menu a průběhu objednávek. Jeho úlohou je zajistit centralizovanou a spolehlivou správu dat, což je nezbytné pro sledování aktu-

álního stavu restaurace. Databázový modul je zajištěn pokladním systémem restaurace a je jeho součástí.

Integrace na pokladní systém zajišťuje propojení s centrálním pokladním systémem restaurace. Tato integrace je provedena vystaveným rozhraním sytému. Tím umožňuje přenos informací o provedených objednávkách a rezervacích mezi mobilní aplikací a centrálním systémem.

Celkově vzato, navrhovaný systém se snaží poskytnout komplexní a efektivní řešení pro restaurační provoz, které nejen usnadňuje práci personálu, ale zároveň vytváří příjemný a moderní zážitek pro zákazníky v restauraci.

3.2.1 Component diagram



Obrázek 3.1: Component diagram

Komponentový diagram je rozdělen na dvě hlavní části: mobilní aplikaci a pokladní systém s databází, přičemž každá část obsahuje klíčové komponenty a jejich vzájemné interakce.

Mobilní aplikace

- **Uživatelské rozhraní**
  Tato část zahrnuje všechny vizuální prvky, se kterými uživatel na mobilní aplikaci interaguje. To zahrnuje obrazovky pro rezervace, načítání menu, vytváření objednávek, zaplacení a další.

- **Funkční moduly**

  - **Rezervační modul:** Umožňuje uživatelům vytvářet rezervace.
  - **Menu modul:** Zahrnuje funkce načítání menu a výběru položek.
  - **Objednávkový modul:** Umožňuje uživatelům vytvářet objednávky a sledovat jejich stav.

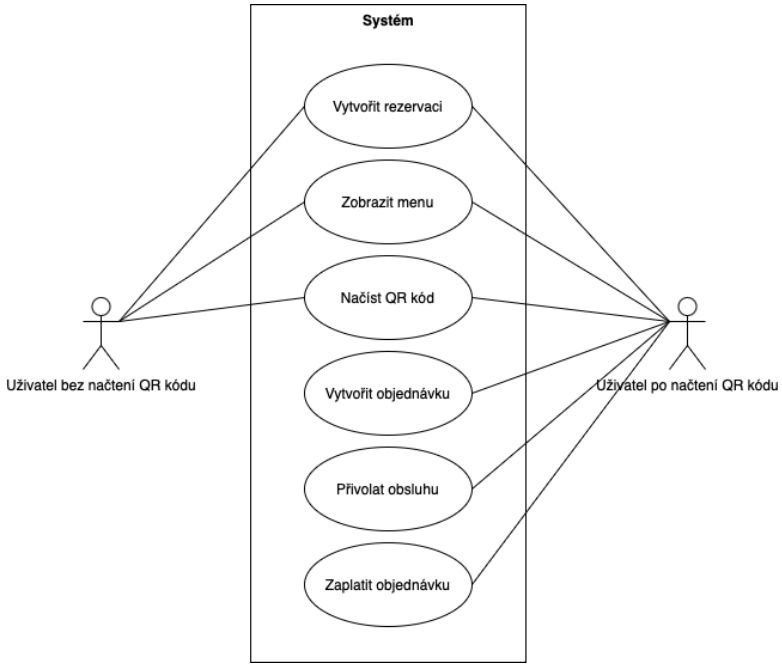
- **Platební modul:** Obsahuje funkce spojené s platbami a integrací s platebními službami.
- **Modul pro ověření totožnosti:** Umožňuje aplikaci se ověřit pokladnímu systému tak, aby do něj měla přístup.

Pokladní systém s databází

- **Server pro ověřování totožnosti:** Odpovědný za ověřování totožnosti pro připojení do pokladního systému a zajištění bezpečné komunikace.
- **Server pro byznys logiku:** Zpracovává transakce a podnikovou logiku spojenou s objednávkami a rezervacemi.
- **Databázový systém:** Úložiště pro data, zahrnující informace o rezervacích, menu, objednávkách a dalších relevantních údajích.
- **Integrační rozhraní:** Umožňuje komunikaci mezi mobilní aplikací a pokladním systémem, zahrnuje zasílání a čtení dat o rezervacích a objednávkách.

Tímto způsobem jsou klíčové komponenty jasně definovány, a rozděleny mezi mobilní aplikaci a pokladní systém s databází, což usnadňuje sledování a porozumění celkové architektuře systému.

3.2.2 Use case diagram



Obrázek 3.2: Use case diagram

Use case diagram má dva aktéry: uživatel bez načtení QR kódu a uživatel po načtení QR kódu. Každému z nich aplikace umožňuje jiné služby.

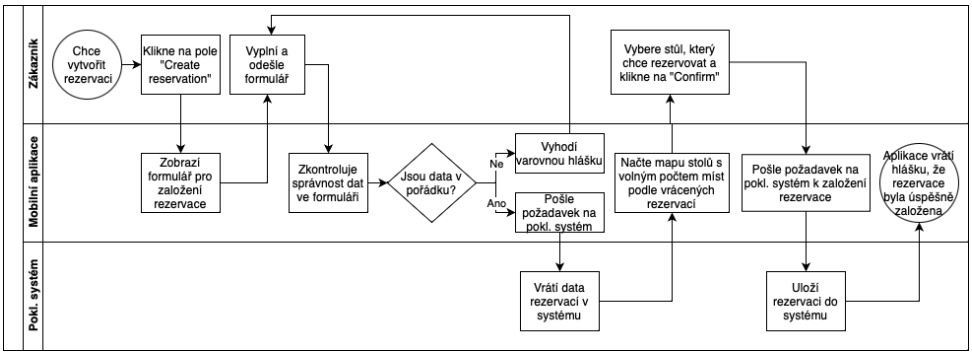
Uživatel bez načtení QR kódu:

- **Vytvořit rezervaci:** Uživatel může vytvořit rezervaci v restauraci.
- **Zobrazit menu:** Uživatel může prohlížet aktuální menu restaurace.
- **Načíst QR kód:** Uživatel může naskenovat QR kód, aby získal další informace o restauraci.

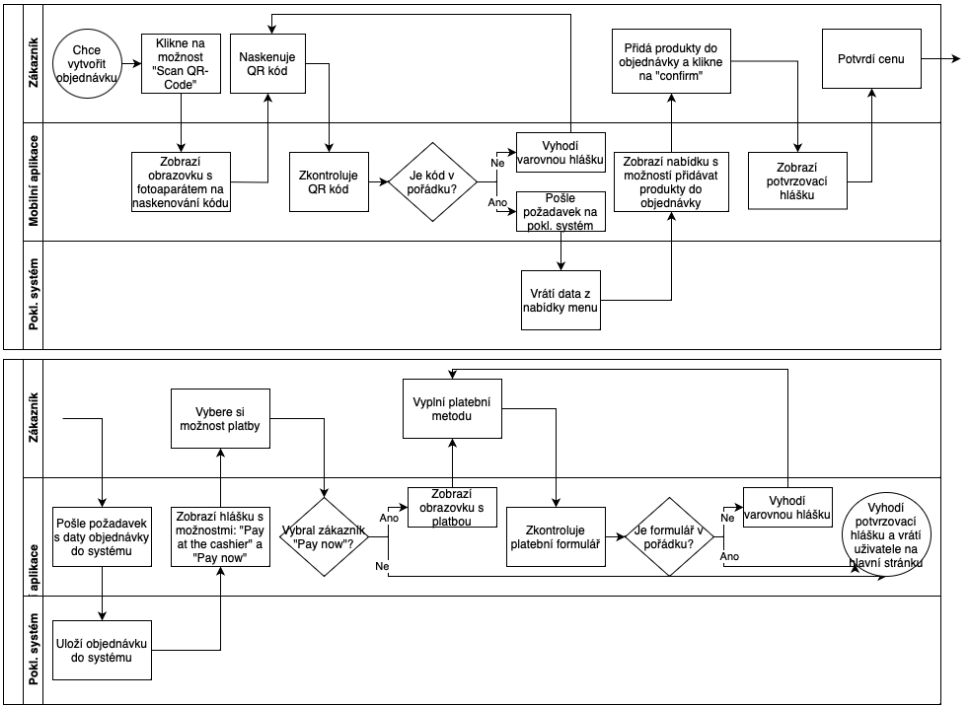
Uživatel po načtení QR kódu:

- **Vytvořit rezervaci:** Uživatel má stále možnost vytvořit rezervaci.
- **Zobrazit menu:** Uživatel může i nadále prohlížet menu.
- **Načíst QR kód:** Uživatel může naskenovat QR kód pro další akce.
- **Vytvořit objednávku:** Po načtení QR kódu může uživatel vytvářet objednávky.
- **Přivolat obsluhu:** Uživatel může přivolat obsluhu ke svému stolu.
- **Zaplatit objednávku:** Uživatel může provést platbu za svou objednávku.

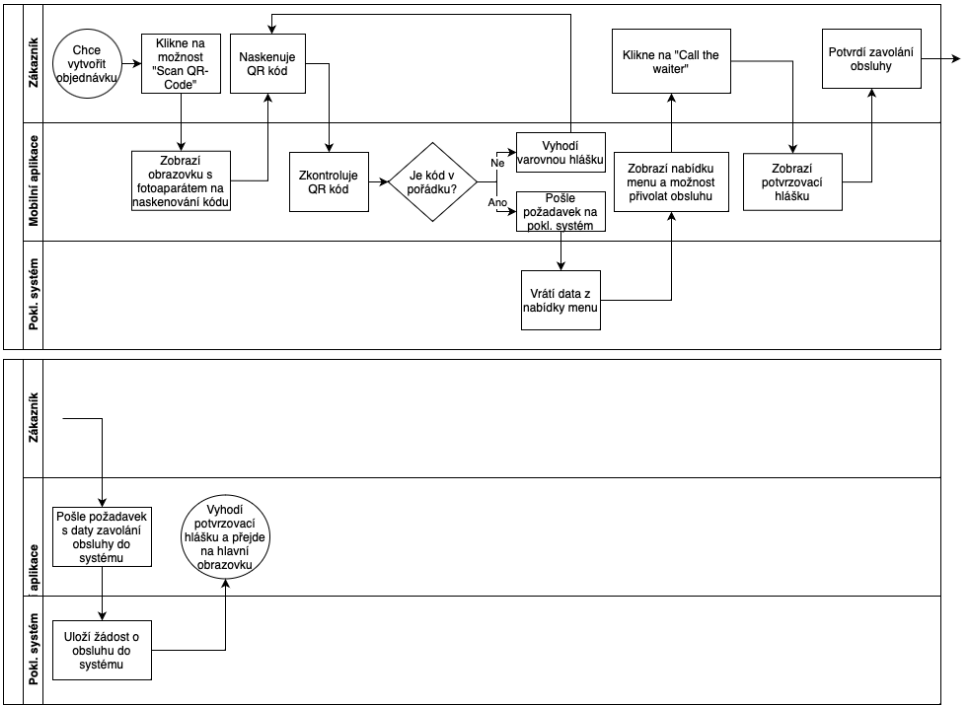
3.2.3 Flow diagramy



Obrázek 3.3: Flow diagram: Rezervace



Obrázek 3.4: Flow diagram: Vytvoření objednávky



Obrázek 3.5: Flow diagram: Přivolání obsluhy

V této kapitole jsem vytvořil tři grafické diagramy, které detailně ilustrují průběh klíčových procesů v restaurační aplikaci. Každý z těchto diagramů

mapuje specifický postup, konkrétně proces rezervace, vytváření objednávky a zavolání obsluhy. Druhé dva diagramy mají velkou část společnou, což je zřejmé z ilustrací. Chybí zde ilustrace procesu zobrazení nabídky menu bez QR kódu, ale ta je tak primitivní, že jsem ji neuváděl.

3.3 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní aplikace klade důraz na intuitivnost a jednoduchost. Hlavní cíl je poskytnout uživatelům plynulý a bezproblémový zážitek. Uživatelské rozhraní by mělo být přehledné, s jasně viditelnými tlačítky a navigací. Všechny funkce, jako jsou rezervace, objednávání jídla a platby, budou snadno přístupné a intuitivní.

3.4 Požadavky aplikace

Povinnými požadavky pro běh aplikace ze strany uživatele jsou následující:

- Připojení k internetové službě
- Zařízení se systémem iOS
- Naskenování QR Kódu v restauraci - *tento bod se vztahuje pouze na některé funkce aplikace*

Povinnými požadavky pro běh aplikace z pohledu poskytovaných služeb třetích stran jsou následující:

- Fungující pokladní systém restaurace s vystaveným rozhraním a daty

Kapitola 4

Implementační prostředky

4.1 Rešerše prostředků pro vývoj mobilních aplikací

V této sekci porovnávám různé frameworky používané pro vývoj multiplatformních mobilních aplikací. Jsou zde také jednotlivé výhody a nevýhody frameworků. Specificky se zaměřuji na Kotlin, React Native, Flutter, Ionic, Xamarin a Cordova.

4.1.1 React Native

React Native je framework používající JavaScript a je známý svou schopností poskytovat vysoký výkon, který se téměř nerozlišuje od nativního prostředí. To znamená, že aplikace vytvořené pomocí React Native mají tendenci vypadat a chovat se tak, jako by byly napsány v nativním programovacím jazyce (například Swift pro iOS nebo Java/Kotlin pro Android). [33] [34] [35] [36] [22] [37] [38] [39]

Výhody

- Snadná integrace s existující infrastrukturou a kompatibilita s různými knihovnami třetích stran.
- Pohodlné psaní kódu díky důležitým chybovým zprávám, časově úsporným nástrojům a bohaté komunitní podpoře.
- Možnost psát kód jednou a nasadit ho na více platformách, včetně iOS, Android a Windows.
- Opakované využití kódu až do 90% pro různé platformy, včetně možnosti použití web app kódu pro mobilní aplikace.
- Předvyvinuté komponenty a kompatibilita s pluginy třetích stran.
- Hladší a rychlejší uživatelské rozhraní díky využití nativních API pro renderování UI.

Nevýhody

- Mírně nižší výkon než u plně nativních aplikací, zejména v graficky náročných aplikacích.
- Složitost udržování aplikace aktualizované s nejnovější verzí React Native.
- Komplikace při ladění a řešení problémů kvůli kombinaci JavaScriptu a nativního kódu.
- Omezení vývojářských zdrojů a menší komunita ve srovnání s Javou.
- Závislost na Facebooku a riziko, že podpora React Native může být ukončena.

React Native je vhodný pro širokou škálu použití, zejména pro projekty vyžadující inovace a vývoj funkčně bohatých, multiplatformních mobilních aplikací. [33] [34] [35] [36] [22] [37] [38] [39]

4.1.2 Kotlin

Kotlin je moderní programovací jazyk, který běží na Java Virtual Machine (JVM) a je kompatibilní s Javou. Je oblíbený pro svou čistotu a efektivitu. [40] [41]

Výhody

- Kotlin maximalizuje produktivitu díky svému jasnému, kompaktnímu a efektivnímu jazyku.
- Kompatibilita s Javou umožňuje snadnou integraci s existujícím Java kódem.
- Větší spolehlivost a bezpečnost, zejména ve vztahu k nulovým hodnotám.
- Podpora pro křížový vývoj platforem díky Kotlin Multiplatform Mobile.

Nevýhody

- Kotlin není tak ustálený jako Java, což může znamenat častější aktualizace a potenciální problémy.
- Omezené výukové zdroje a menší komunita ve srovnání s Javou.
- Stále existuje potřeba znalosti Javy pro efektivní využití Kotlinu.

Kotlin je vhodný pro projekty, které vyžadují vysokou produktivitu a efektivitu, zejména ve vývoji Android aplikací. Nicméně je důležité vzít v úvahu jeho omezení, zejména pokud jde o ustálenost jazyka. [40] [41]

4.1.3 Flutter

Flutter, který využívá programovací jazyk Dart, je známý pro svůj vynikající výkon a schopnost vytvářet komplexní uživatelské rozhraní. [33][35] [42] [43] [44]

Výhody

- Vysoký výkon díky kompilaci kódu do nativního ARM kódu.
- Konzistence a princip "write once, run anywhere", což vede k rychlejšímu vývoji a snazší údržbě kódu.
- Silná podpora komunity a otevřený zdrojový kód.
- Vynikající dokumentace a široké spektrum přizpůsobitelných widgetů.
- Podpora pro vývoj webových a desktopových aplikací.

Nevýhody

- Nutnost učení nového jazyka (Dart).
- Flutter je stále relativně nový framework s omezenými zdroji pro učení a menším počtem pluginů a balíčků.
- Větší velikost aplikací, což může vést k delší době stahování a většímu nároku na prostor v zařízení.
- Menší komunita Dartu ve srovnání s jazyky jako je JavaScript.

Flutter je efektivní nástroj pro rychlý multiplatformní vývoj s jednou základnou kódu, který je vhodný pro vývojáře hledající efektivní řešení pro vytváření nativního uživatelského rozhraní a vysokého výkonu. Nicméně, je důležité vzít v úvahu jeho omezení, zejména pokud jde o novost frameworku a potřebu znalosti Dartu. [33] [35] [42] [43] [44]

4.1.4 Ionic Framework

Ionic využívá webové technologie a je vhodný pro méně výkonově náročné aplikace. [34] [35] [45] [46]

Výhody

- Je postaven na standardizovaných webových technologiích.
- Díky svému základu na webových technologiích, což usnadňuje přizpůsobení vzhledu a chování aplikací, má vysokou flexibitu.
- Umožňuje vytvářet více aplikací z jedné základny kódu.

- Nabízí knihovnu komponent a pluginů, které umožňují snadné propojení s nativními API telefonů.
- Poskytuje rozsáhlou knihovnu UI komponent, které usnadňují vývoj uživatelského rozhraní.
- Umožňuje vývojářům testovat aplikaci v prohlížeči zařízení. Je to dáno tím, že operuje prostřednictvím WebView.

Nevýhody

- Nižší výkon, zejména u graficky náročných aplikací.
- Hot reloading je ve vývoji software často považován za standardní funkci, ale u Ionicu je jeho nastavení náročnější.
- Na moderních zařízeních je rozdíl ve výkonu zanedbatelný, ale na starších zařízeních může být výkon problematický.

Ionic je užitečný pro projekty, které potřebují rychlý vývoj a nevyžadují vysoký výkon. Je ideální pro vývojáře, kteří jsou již obeznámeni s webovými technologiemi, ale je třeba vzít v úvahu jeho omezení ve výkonu ve srovnání s nativními aplikacemi. [34] [35] [45] [46]

4.1.5 Xamarin

Xamarin, který používá C# a .NET, je známý pro svou schopnost poskytovat dobrý výkon a nativní uživatelské rozhraní, což jej činí vhodným pro vývojáře obeznámené s těmito technologiemi. [34] [35] [47] [48] [49]

Výhody

- Snadná integrace s existující infrastrukturou a kompatibilita s různými knihovnami třetích stran.
- Příjemný vývojový zážitek díky důležitým chybovým zprávám, časově úsporným nástrojům a bohaté komunitní podpoře.
- Jednotný vývojový prostor umožňující psaní aplikací v C# a .NET pro různé platformy.
- Vysoký výkon a nativní uživatelské rozhraní.
- Rozsáhlá knihovna a snadná integrace s Microsoftovými nástroji.

Nevýhody

- Méně robustní komunitní podpora ve srovnání s jinými populárnějšími frameworky.
- Větší velikost aplikací kvůli potřebě zahrnout .NET a další knihovny.

- Omezení v přístupu k některým specifickým API na určitých platformách.
- Potenciální zpoždění ve zveřejňování podpory pro nejnovější verze nativních platforem.

Xamarin poskytuje efektivní cestu pro multiplatformní mobilní vývoj, zejména pro ty, kteří jsou již obeznámeni s C# a .NET. Přesto je důležité pečlivě zvážit jeho nevýhody při rozhodování o jeho použití pro konkrétní projekt. [34] [35] [47] [48] [49]

4.1.6 Cordova

Cordova je založena na webových technologiích a je vhodná pro jednodušší aplikace. [35] [50] [51]

Výhody

- Snadná integrace s existujícími webovými aplikacemi.
- Vhodná pro projekty s omezeným rozpočtem. Díky využití webových technologií a snadnému nasazení mohou být projekty vyvinuté v Cordově cenově efektivnější.
- Poskytuje přístup k široké škále pluginů, které rozšiřují funkčnost aplikací a zjednodušují integraci s nativními funkcemi zařízení.

Nevýhody

- Nižší výkon a responzivita ve srovnání s ostatními frameworky.
- Méně nativní uživatelské zkušenosti.
- Funkčnost aplikace je často závislá na externích pluginech, jejichž kvalita a aktualizace mohou být nejisté.

Při výběru nejlepšího frameworku pro vývoj mobilních aplikací je důležité zvážit specifické požadavky a cíle vašeho projektu. Cordova je vhodná pro projekty, kde je prioritou snadná integrace s webovými technologiemi a multiplatformní nasazení. [35] [50] [51]

4.1.7 Závěr rešerše

Nakonec jsem se rozhodl ve své práci využít framework React Native. Toto rozhodnutí bylo motivováno několika faktory. Prvním z nich je bohatá komunitní podpora a dostupnost mocných knihoven, které React Native nabízí. Tato kombinace znamená, že mám k dispozici širokou škálu nástrojů a zdrojů, které mi usnadní vývoj.[22] [23]

Dalším důležitým faktorem byla popularita React Native vývojového prostředí. Jedná se o jeden z nejčastěji používaných nástrojů ve své kategorii, což

znamená, že se jedná o technologii, která je široce akceptována a využívaná v praxi. To mi přineslo jistotu, že se budu učit a pracovat s nástrojem, který má perspektivu a budu moci využít své dovednosti i v budoucnosti.[22] [23]

4.2 Další využité implementační prostředky

V této kapitole uvedu další prostředky, které hodlám požít na vývoj aplikace pro podporu hosta v restauraci. Tyto prostředky jsem vybral s vědomím, že chci pracovat s frameworkem React Native. A zvolil jsem je tak, aby co nejlépe sloužily potřebám aplikace.

4.2.1 Expo

Na vývoj aplikace bych chtěl použít nástroj Expo. Jedná se o platformu, která se specializuje na vytváření nativních aplikací. Expo je založeno na React Native a přidává mnoho nástrojů pro mobilní aplikace. [24] [25]

Výhodou aplikací, které používají Expo je, že nemusejí obsahovat žádný iOS nebo Android kód. Uživatel si může svůj kód ihned vyzkoušet na mobilním zařízení. Stačí si stáhnout mobilní aplikaci Expo Go a skrze ní spustit svou nativní aplikaci přímo na svém zařízení s iOS nebo Android. [24] [25] [26]

4.2.2 Expo BarCodeScanner

Expo BarCodeScanner je nástroj, který do React Native aplikace přidává možnost skenovat kódy. Jsou skenovány pomocí fotoaparátu mobilního zařízení. Podporuje Android i iOS aplikace. [27]

Toto rozšíření umožňuje skenovat obrovské množství různých čárových kódů. V aplikaci pro podporu hosta v restauraci budu využívat pouze čtečku QR kódů. Pokud se pomocí kamery mobilního zařízení naskenuje jiný kód, aplikace ho zamítne. [27]

4.2.3 Apollo Client

Apollo Client je nástroj, který umožňuje psát GraphQL querries přímo v React aplikaci. Ve své podstatě poskytuje rozraní GraphQL serveru do Reactu.[28] [29]

Uživatel může napsat přímo GraphQL query nebo mutation v kódu React aplikace a ty jsou přes rozhraní Apollo Client poslané na GraphQL server, který následně může komunikovat s další stranou. V případě této aplikace s pokladním systémem restaurace.[28] [29]

4.3 GraphQL

Pro komunikaci mezi aplikací pro podporu hosta v restauraci a pokladním systémem restaurace jsem si zvolil dotazovací jazyk GraphQL.

Jedná se o moderní technologii, která nahrazuje konkurenty jako je například velmi rozšířený REST. Výhodami GraphQL je: jednoduchost dotazování, dotazovaní se vždy na existující data, konzistence mezi platformami, hierarchické uspořádání. Jedná se o silně typový jazyk, což se dá brát jako výhoda i nevýhoda. Aplikace používající GraphQL jsou rychlejší, protože se dotazují na konkrétní data. Nedostávají výčet dat ze serveru jako je tomu například u aplikací s REST technologií. [30] [31]

V aplikaci pro podporu hosta v restauraci se bude používat GraphQL server na zapisování a získávání dat z pokladního systému restaurace. Dotazovat se bude pomocí vystaveného API systému.

Kapitola 5
Postup implementace

Pro postup implementace je nejdříve nutné se seznámit s použitými technologiemi. Ty už jsem si zvolil výše. Nyní je potřeba si zvolit pokladní systém, na který se bude má aplikace integrovat. Rozhodl jsem se pro pokladní systém Dotykačka. Hlavními důvody byly: velmi dobrá dokumentace k API systému, podpora všech funkcí, které potřebuji a vstřícnost týmu Dotykačka, který mi poskytl možnost založit virtuální restauraci. Za to bych jim chtěl moc poděkovat.

5.1 Pokladní systém Dotykačka

Pokladní systém Dotykačka bude představovat třetí stranu pro aplikaci, kterou popisuji v této práci. Jedná se o velice složitý nástroj, který umožňuje vytváření pokladních systémů v mnoha sektorech služeb. Podporuje gastronomické odvětví, které bude pro mou práci nejdůležitější. Dále podporuje pokladní systémy v klasických materiálních obchodech, službách - například kadeřnictví a v mnoha ubytovacích zařízeních. [6] [7]

Hlavními funkcemi pokladního systému pro gastro jsou správa rezervací, docházky zaměstnanců, přehled tržeb, online objednávky, přehled skladových zásob. Systém zároveň umožňuje všechny druhy plateb. Podporuje také mapu stolů, která kopíruje restauraci. Tato vlastnost je implementována pro použití zaměstnanců prodejny. [8]

Funkce tohoto nástroje obsahují všechny požadavky aplikace pro podporu hosta v restauraci. Je to tedy ideální pokladní systém, na který se bude integrovat.

Aplikace bude se systémem komunikovat přes dotazovací jazyk GraphQL a integrovat se bude na vystavené API. Pokladní systém bude simulovat byznys logiku aplikace, bude přijímat požadavky z klientské strany a bude je zpracovávat. Vystavené API poskytuje obrovské množství funkcí, které jsou velice dobře zdokumentovány. Díky tomu by komunikace s aplikací měla být přímočará a bez velkých komplikací. [9]

Pokladní systém, na který se bude integrovat aplikace, bude představovat restauraci. Nebude se ale jednat o existující prodejnu. V systému bude vytvořena pobočka, která bude pouze maketou. Budou do ní ale vložena data a pokladny tak, aby co nejvěrohodněji napodobovala reálnou restauraci.

5.1.1 Pobočka v systému a autorizace

Pro založení virtuální pobočky v systému Dotykačka je prvním krokem stažení aplikace Dotykačka na mobilní zařízení s operačním systémem Android, kde následně pobočka běží. Aplikace umožňuje uživatelům efektivně spravovat nastavení a funkce restaurace. Nicméně, pro přístup k API pobočky, což je nezbytné pro integraci aplikace a získání detailních dat, je nutné mít aktuální Bearer token. [6] [9]

Bearer token, který slouží jako autorizační klíč pro přístup k API do konkrétní pobočky, lze získat prostřednictvím specifického dotazu na API Dotykačky. Tento dotaz zahrnuje identifikaci uživatele a unikátní identifikátor pobočky. Po ověření těchto informací systém Dotykačka poskytne Bearer token, což umožní uživatelům provádět požadované operace prostřednictvím API. [9]

Tento mechanismus zajišťuje, že pouze autorizovaní uživatelé mají přístup k API, a tím chrání citlivé informace a funkce systému. Správná implementace a použití Bearer tokenu je klíčové pro bezpečnou a efektivní integraci mobilní aplikace s pokladním systémem Dotykačka, což umožňuje detailní správu a analýzu dat pobočky. [9]

5.1.2 Vystavené rozhraní systému Dotykačka

V této kapitole se podrobněji zaměříme na vystavené rozhraní (API) systému Dotykačka verze 2, které umožňuje programové řízení entit, požadování datových reportů a zpracování objednávek přímo v systému Point of Sale. API v2 Dotykačky přináší rozšířené možnosti pro komplexní správu a automatizaci procesů, což je zásadní pro efektivní využití v rámci mobilních aplikací a dalších integrovaných systémů. [9]

Struktura a funkce

API v2 Dotykačky je navrženo tak, aby poskytovalo uživatelům široký rozsah funkcí pro správu různých aspektů restauračního a maloobchodního podnikání. Hlavními funkcemi tohoto rozhraní jsou správa rezervací, přehledy docházek zaměstnanců, generování tržebních reportů, online objednávání a sledování skladových zásob. Kromě toho systém podporuje správu mapy stolů, která je klíčová pro efektivní plánování obsluhy v restauracích. [9]

Bezpečnost

Pro přístup k API v2 je vyžadován proces autorizace, kde uživatelé musí získat klientské přihlašovací údaje, včetně Client ID a Client Secret. Tyto údaje jsou základem pro vytváření a udržování bezpečné komunikace mezi uživatelskými aplikacemi a API. Bezpečnost je dále posílena použitím Refresh Tokenů a Access Tokenů, které zajišťují, že pouze autorizované entity mají přístup k funkcím API. [9]

5.2 Další kroky implementace

Implementace aplikace započne založením React Native projektu skrze nástroj Expo. Následně se naimportuje knihovna Apollo Client a zprovozní se GraphQL server pomocí tohoto nástroje. Díky tomu se bude moci aplikace dotazovat skrze GraphQL server na pokladní systém Dotykačka. Další krok přestavuje zprovoznění dotazů z mobilní aplikace na pokladní systém. Do aplikace bude nutné vložit údaje o registrovaném uživateli a pobočce založené v systému Dotykačka tak, aby bylo možné získávat Bearer token, který umožňuje přístup k datům pobočky. Následně bude potřeba naimplementovat dotazy v GraphQL, aby se dalo vyzkoušet spojení a komunikace mezi jednotlivými stranami. Pokud jsou všechny předchozí body splněny, následuje naprogramování požadavků v React Native. Nutností bude přidání nástroje BarCodeScanner, který je využitý k naskenování QR kódů. Součástí toho bude také upravení povolení v operačním systému, konkrétně vyžádání použití fotoaparátu. Jako poslední bude potřeba vyřešit vzhled aplikace a její rozložení v mobilním telefonu. Komponenty budou graficky ztvárněny pomocí kaskádových stylů. Aplikace bude mít snahu o co nejjednodušší design tak, aby měl uživatel z používání co nejlepší zážitek.

5.2.1 Struktura aplikace v React Native

Klasická aplikace vytvořená v nástroji React native má konkrétní strukturu, která bude využita i na implementaci v této práci. Po vytvoření aplikace se založí rootový adresář s jejím názvem. V tomto adresáři jsou typicky tyto položky:

- adresář **assets** - Jedná se o adresář, kde jsou uloženy zdroje pro aplikaci. Jsou to například obrázky a fonty.
- adresář **node_modules** - Složka externích modulů, které konkrétní React Native aplikace využívá. Slouží jako cache pro tyto moduly.
- soubor **App.js** - Hlavní komponenta React Native aplikace. Přes ní se startuje celý její běh. Jedná se o Javascriptový soubor. Skrze něj budou do aplikace naimportovaná knihovna Apollo Client.
- soubor **app.json** - Konfigurační soubor aplikace.
- soubor **package.json** - Uschovává údaje o projektu, které jsou nutné před publikováním a spuštěním. Obsahuje například konkrétní použité verze a vstupní body aplikace.
- soubor **babel.config.js** - Používá se k práci s nástojem Babel.

[32]

Jelikož aplikace bude používat nástroj Expo, tak se v rootovém adresáři objeví navíc adresáře **/.expo**, který slouží pro konfiguraci samotného nástroje.

Do rootu aplikace přidám také adresáře, ve kterých následně budou jednotlivé Javascript soubory. Adresáře budou rozděleny podle typu soubou, tak aby se v kódu dobře orientovalo.

- adresář **requests** - Je adresář, který v sobě má komponenty, které přímo obsahují požadavky s kódem GraphQL.
- adresář **screens** - Každý Javascriptový soubor v něm představuje jednu obrazovku, ke které se může uživatel dostat.
- adresář **styles** - Adresář obsahuje soubory se styly pro komponenty React Native. Tyto soubory budou určovat vzhled aplikace.
- adresář **___test___** - Adresář obsahuje soubory s unit testy. Kopíruje strukturu adresářů v rootu.

Kapitola 6

Implementace

V této kapitole popíši samotnou implementaci aplikace a uvedu části kódu tak, aby bylo zjevné, jak aplikace funguje a jak využívá zvolené technologie.

6.1 App.js

App.js je hlavní komponenta React Native aplikace. U ní se startuje celý její běh. V mé aplikaci se stará o strukturu navigace mezi obrazovkami a nastavení komunikace přes ApolloClient.

6.1.1 Importy

```
1 import HomeScreen from "./screens/HomeScreen";
2 import ReservationFormScreen from "./screens/
    ReservationFormScreen";
3 // (dalsi importy obrazovek)
4 import { NavigationContainer } from '@react-navigation/
    native';
5 import { createNativeStackNavigator } from '@react-
    navigation/native-stack';
6 import { ApolloClient, InMemoryCache, ApolloProvider }
    from '@apollo/client';
```

Úryvek kódu 6.1: App.js: importy

Každý import obrazovky (**HomeScreen**, **ReservationFormScreen**, atd.) reprezentuje jednu obrazovku v aplikaci, zatímco importy z **@react-navigation/native** a **@apollo/client** zajišťují navigaci a otevření komunikace přes Apollo Client.

6.1.2 Konfigurace Apollo Client

```
const restLink = new RestLink({
    uri: "https://api.dotykacka.cz/v2/",
    headers: {
        Authorization: 'User ***', // Misto *** je vlozen
            identifikacni klic uzivatele k ziskani tokenu
        'Content-Type': 'application/json',
    }
});

const client = new ApolloClient({
    uri: 'localhost:4000/graphql',
    cache: new InMemoryCache(),
    link: restLink
});
```

Úryvek kódu 6.2: App.js: konfigurace Apollo Client

Tento kód ukazuje, jak je v aplikaci nastavena integrace s externím API prostřednictvím **ApolloClient** a **RestLink**. Tento přístup umožňuje aplikaci komunikovat jak s REST API, tak s GraphQL endpointy.

RestLink

RestLink je součástí **apollo-link-rest**, která umožňuje Apollo Clientu komunikovat s REST API bez nutnosti mít kompletní GraphQL server.

- **uri:** Toto je základní URL adresa REST API, v tomto případě **https://api.dotykacka.cz/v2/**.
- **headers:** Zde se nastavují HTTP hlavičky pro všechna volání přes **RestLink**. Hlavička **Authorization** obsahuje přihlašovací token, který ověřuje přístup uživatele k API. **Content-Type** nastavuje typ obsahu, kterým je v tomto případě **application/json**.

Apollo Client

ApolloClient je hlavní knihovna používaná pro interakci s GraphQL API. V tomto případě je však napasována tak, aby mohla pracovat s REST API přes RestLink. Toto je nutné proto, aby aplikace využívající GraphQL mohla komunikovat s rozhraním systému Dotykačka, které je RESTové.

- **uri:** Toto je endpoint pro GraphQL API, v tomto případě **localhost:4000/graphql**. To znamená, že server běží lokálně na počítači.
- **cache: InMemoryCache** je standardní cache mechanismus pro **ApolloClient**, který pomáhá ukládat a spravovat odpovědi od serveru pro rychlejší přístup a optimalizaci výkonu.

- **link:** Zde se používá **restLink** vytvořený výše. To umožňuje **ApolloClient** využívat REST API jako součást jeho operací.

6.1.3 Navigační struktura

```
1  const Stack = createNativeStackNavigator();
2
3  export default function App() {
4      return (
5          <ApolloProvider client={client}>
6              <NavigationContainer>
7                  <Stack.Navigator>
8                      <Stack.Screen name="Home" component={
                           HomeScreen} />
9                      // (dalsi navigacni obrazovky)
10                 </Stack.Navigator>
11             </NavigationContainer>
12         </ApolloProvider>
13     );
14 }
```

Úryvek kódu 6.3: App.js: navigační struktura

Zde **Stack.Navigator** obaluje jednotlivé **Stack.Screen** komponenty, což umožňuje navigaci mezi různými obrazovkami aplikace. Každý **Stack.Screen** reprezentuje jednu obrazovku v aplikaci a definuje název cesty a komponentu obrazovky, která se má zobrazit.

Funguje to následovně: **Stack.Navigator** je kontejner, který spravuje skupinu obrazovek v aplikaci. Je možno si to představit jako zásobník, kde každá nová obrazovka je přidána na vrchol a uživatel může navigovat zpět k předchozím obrazovkám. Když **Stack.Navigator** otevře novou obrazovku, stará obrazovka zůstává pod novou a může být znovu zobrazena, když se uživatel rozhodne "jít zpět".

6.2 GetAccessToken

Tato komponenta je určena k získání a správě přístupového tokenu pro API. Je použita hned v souboru App.js, to znamená, že je načtena při zapnutí aplikace. Viz kód níže.

```
export default function App() {
    return (
        <ApolloProvider client={client}>
            <GetAccessToken cloudId={specificCloudId}/>
            // (zbytek kodu)
            </NavigationContainer>
        </ApolloProvider>
    );
}
```

Úryvek kódu 6.4: App.js: GetAccessToken

6.2.1 Importy

```
import React, { useEffect, useState } from 'react';
import { View } from 'react-native';
import { useMutation, gql, useApolloClient } from '
   @apollo/client';
import { RestLink } from "apollo-link-rest";
```

Úryvek kódu 6.5: GetAccessToken.js: importy

- **UseEffect** a **useState** jsou React Hooks, které umožňují použití stavu a životního cyklu u komponent.
- **UseMutation**, **gql**, a **useApolloClient** jsou součástí Apollo Client knihovny pro práci s GraphQL.

6.2.2 GraphQL mutace

```
const SIGNIN_MUTATION = gql`
  mutation SignIn($cloudId: String!) {
    signin_cloudId(input: { _cloudId: $cloudId }) @rest(
       type: "SignInResponse", path: "signin/token",
       method: "POST") {
      accessToken
    }
  }
`;
```

Úryvek kódu 6.6: GetAccessToken.js: GraphQL mutace

Pomocí **gql** je definována GraphQL mutace **SIGNIN_MUTATION**. Tato mutace očekává proměnnou **cloudId** a volá REST API endpoint pro získání přístupového tokenu. **@rest** direktiva indikuje, že tato operace používá REST API místo tradičního GraphQL endpointu. Toto je typické pro celou

aplikaci a všechny její požadavky. Dále je zde možné vyčíst, že požadavek bude poslán metodou **"POST"**.

6.2.3 Práce s access tokenem

```
const [accessToken, setAccessToken] = useState("");
const client = useApolloClient();
const [signIn] = useMutation(SIGNIN_MUTATION);

const handleSignIn = () => {
        signIn({ variables: { cloudId } })
            .then(response => {
                const { accessToken } = response.data.
                    signin_cloudId;
                setAccessToken(accessToken);
                console.log('Access Token:', accessToken)
                    ;

                const updatedRestLink = new RestLink({
                    uri: "https://api.dotykacka.cz/v2/",
                    headers: {
                        Authorization: `Bearer ${
                            accessToken}`,
                        'Content-Type': 'application/json
                            ',
                    }
                });

                client.setLink(updatedRestLink);
            })
            .catch(error => { //zpracovani erroru
            });
    };
```

Úryvek kódu 6.7: GetAccessToken.js: funkce handleSignIn

V této části kódu je znázorněno, jak aplikace dále pracuje s access tokenem. Konkrétně je to funkce **handleSignIn**. Tato funkce vezme vrácený access token a upraví **RestLink** v **Apollo Clientu**. **Restlink** po úpravě posílá v autotrizační hlavičce právě access token tak, aby aplikace mohla přistupovat k rozhraní konkrétní pobočky.

6.3 HomeScreen

```
export default function HomeScreen({ navigation }) {
    return (
        <View style={CommonStyle.container}>
            <Image
                style={ImageStyle.mainLogo}
                source={require("../assets/logo-no-
                    background.png")} />
            <View style={ButtonStyle.buttonView}>
                <Pressable
                    style={ButtonStyle.button}
                    onPress={()=> navigation.navigate("
                        Reservation")}>
                    <Text style={ButtonStyle.text}>
                        Create reservation
                    </Text>
                </Pressable>
                // (dalsi tlacitka)
            </View>
            <StatusBar style="auto" />
        </View>
    );
}
```

Úryvek kódu 6.8: HomeScreen.js

Tento úryvek kódu představuje jednoduchou funkční komponentu **HomeScreen**, která slouží jako hlavní obrazovka (domovská stránka) aplikace. Je načtena hned po spuštění aplikace. Obsahuje logo restaurace a tři tlačítka (""**Reservation**", "**Scan code**"; a "**View menu**"). V **onPress** událostech tlačítek je použita **navigation** logika, která umožňuje navigaci mezi obrazovkami aplikace.

6.4 ReservationFormScreen

ReservationFormScreen představuje první obrazovku v procesu vytváření rezervace. Slouží jako formulář pro vyplnění údajů, tak aby mohla rezervace vzniknout.

Umožňuje uživateli zadat své jméno, datum a čas rezervace a vložit poznámku. Po validaci předává tato data pro další zpracování v aplikaci. Stylování a layout jsou upraveny tak, aby byly přívětivé pro uživatele a reagovaly na interakci s klávesnicí.

Komponenta obsahuje validační logiku, která uživateli neumožní odeslat vadný formulář. Pro vybrání času je použita komponenta **DateTimePicker** z knihovny **"react-native-modal-datetime-picker"**.

```
{
  "name": "Smith",
  "note": "",
  "date": "8. 1. 2024",
  "time": "18:46"
}
```

Úryvek kódu 6.9: ReservationFormScreen.js: příklad odeslaných dat

6.5 SelectTableScreen

```
export default function SelectTableScreen({ navigation,
    route }) {
    // Logika pro zpracovani casu rezervaci
    const compareTimes = (reservationData, reservations)
        => {
        //...
    };

    // Logika pro vyber stolu
    const handleTablePress = (tableId, maxSeatCount) => {
        //...
    };

    // Validace vyberu
    const validate = () => {
        //...
    };

    return (
        <View style={CommonStyle.container}>
            <GetReservations>
                // (Zobrazeni jednotlivych stolu a jako
                    tlacitek)
            </GetReservations>
            <StatusBar style="auto"/>
        </View>
    );
}
```

Úryvek kódu 6.10: SelectTableScreen.js: funkce komponenty

SelectTableScreen umožňuje uživateli vybrat stůl z dostupných možností na základě času. Stoly, které jsou v daném čase obsazené nebo nevyhovují počtu osob, jsou označeny jako nedostupné. Aplikace zjistí, které stoly jsou obsazené pomocí komponenty **GetReservations**, ta vrací všechny aktuální rezervace a jejich data.

Soubor **SelectTableScreen**.js obsahuje komponentu **Table**, která reprezentuje jednotlivé stoly.

6.5.1 Komponenta Table

```
const Table = ({tableId, selectedTableId, disabledTables,
    handleTablePress, mainStyle, maxSeatCount}) => {
    //...
};
```

Úryvek kódu 6.11: SelectTableScreen.js: komponenta Table

Přijímá různé props jako **tableId**, **selectedTableId**, **disabledTables** pro určení stavu a chování stolu. **TableId** je zcela zásadní, protože se musí shodovat s id v pokladním systému Dotykačka.

6.6 GetReservation

Komponenta **GetReservation** v React Native aplikaci slouží k načtení a zpracování dat o rezervacích ze serveru pomocí GraphQL. Tato komponenta také využívá Apollo Client pro provádění GraphQL dotazů.

```
const RESERVATIONS_QUERY = gql`
  query Reservations {
    reservations @rest(type: "ReservationList", path: "
       clouds/373657584/reservations?sort=-created") {
      current
      PageperPage
      totalItemsOnPage
      // ...
      data {
        _branchId
        _cloudId
        _customerId
        // ...
      }
    }
  }
`;
```

Úryvek kódu 6.12: GetReservation.js: GraphQL mutace

Takto vypadá kód GraphQL mutace. Zbytek kódu v této komponentě zajišťuje přijmutí a poslání dat dále.

6.7 ReservationSummaryScreen

```
export default function ReservationSummaryScreen({ route,
    navigation }) {
    // Ziskani dat z predchozi obrazovky
    const {
        //...
    } = route.params;

    // Priprava dat pro rezervaci
    const reservationData = {
        // Objekt s daty rezervace
    };

    // Callback pro vytvoreni rezervace
    const onReservationCreatedCallback = useCallback(
        (success) => {
            // Zobrazeni alertu v zavislosti na uspechu
               rezervace
        },
        [navigation]
    );

    // Efekt pro sledovani stavu rezervace
    useEffect(() => {
        if (reservationStatus) {
            onReservationCreatedCallback(
               reservationStatus === 'success');
        }
    }, [reservationStatus, onReservationCreatedCallback])
       ;

    // Zobrazeni UI
    return (
        <View style={CommonStyle.container2}>
            {reservationStatus === null && (
                <CreateReservation reservationData={
                   reservationData} onReservationCreated
                   ={onReservationCreatedCallback} />
            )}
            <StatusBar style="auto" />
        </View>
    );
}
```

Úryvek kódu 6.13: ReservationSummaryScreen.js: funkce komponenty

Komponenta **ReservationSummaryScreen** v aplikaci je určena k zobrazení souhrnu rezervace a k jejímu potvrzení. Tato obrazovka je závěrečnou částí procesu rezervace a interaguje s API pro vytvoření nové rezervace.

Obsahuje komponentu **CreateReservation**, která odesílá data do pokladního systému. Následně čeká na odpověď. Pokud se rezervace zdařila, informuje o tom uživatele a vrátí ho na hlavní stránku.

6.8 CreateReservation

Komponenta **CreateReservation** v React Native aplikaci slouží k vytváření nové rezervace prostřednictvím GraphQL mutace. Tato komponenta je součástí procesu rezervace a interaguje se na backend, aby uložila informace o rezervaci.

```
const CREATE_RESERVATION_MUTATION = gql`
  mutation CreateReservation($reservations: [ReservationInput!]!) {
    createReservation(reservations: $reservations)
      @rest(
        type: "ReservationResponse"
        path: "clouds/373657584/reservations"
        method: "POST"
        bodyKey: "reservations"
      ) {
        id
      }
}
`;
```

Úryvek kódu 6.14: CreateReservation.js: GraphQL mutace

6.9 ScanCodeScreen

Komponenta ScanCodeScreen v React Native aplikaci slouží k načítání QR kódů pomocí kamery zařízení. Tato obrazovka je určena k identifikaci stolů v restauraci pomocí QR kódů.

6.9.1 Importy

```
import React, { useState, useEffect } from 'react';
import { StatusBar } from 'expo-status-bar';
import { BarCodeScanner } from 'expo-barcode-scanner';
import { Text, View, StyleSheet, Alert } from 'react-native';
import CommonStyle from '../styles/CommonStyle';
```

Úryvek kódu 6.15: ScanCodeScreen.js: importy

Importy obsahují:

- Základní komponenty React a React Native pro vytvoření uživatelského rozhraní.
- **BarCodeScanner** z **expo-barcode-scanner** pro načítání QR kódů.
- Styly z **CommonStyle** pro aplikaci vzhledu.

6.9.2 Dovolení o použití kamery

```
1     useEffect(() => {
2         (async () => {
3             const { status } = await BarCodeScanner.
                  requestPermissionsAsync();
4             setHasPermission(status === 'granted');
5         })();
6     }, []);
```

Úryvek kódu 6.16: ScanCodeScreen.js: dovolení o použití kamery

Tento krátký úryvek kódu ukazuje, jak se aplikace dotazuje na práva k použití fotoaparátu zařízení.

6.9.3 Použití BarCodeScanner

```
1     <BarCodeScanner
2     onBarCodeScanned={scanned ? undefined :
          handleBarCodeScanned}
3     style={StyleSheet.absoluteFillObject}
4 />
```

Úryvek kódu 6.17: ScanCodeScreen.js: použití BarCodeScanner

OnBarCodeScanned je funkce, která se zavolá, když je QR kód úspěšně naskenován. V tomto případě **handleBarCodeScanned** zpracovává naskenovaná data a validuje je.

6.10 OrderMenuScreen

Komponenta **OrderMenuScreen** je navržena k poskytování funkcí pro výběr a potvrzení objednávek jídla a nápojů v restauraci. Tato obrazovka je klíčovou součástí procesu objednávání a zahrnuje interakci s produkty, výpočet celkové ceny a možnost přivolání obsluhy.

Produkty restaurace získává pomocí komponenty **GetProducts**. Ta je velice podobná komponentě **GetReservations** (viz sekce 6.6).

Z této komponenty pokračují dva procesy: vytvoření objednávky a přivolání obsluhy. Každá přechází na jinou obrazovku.

6.11 CreateCustomer

Chtěl bych se zmínit o komponentě **CreateCustomer**, která neintuitivně slouží k u procesu přivolání obsluhy. Je tomu proto, že Dotykačka neposkytuje funkci přímo pro přivolání obsluhy. Vyřešil jsem to tak, že v pokladně vytvářím zákazníky se jménem, které obsahuje čas přivolání obsluhy a číslo stolu. Pro obsluhu pokladny je snadné si zobrazit nejaktuálnější časy a stůl obsloužit.

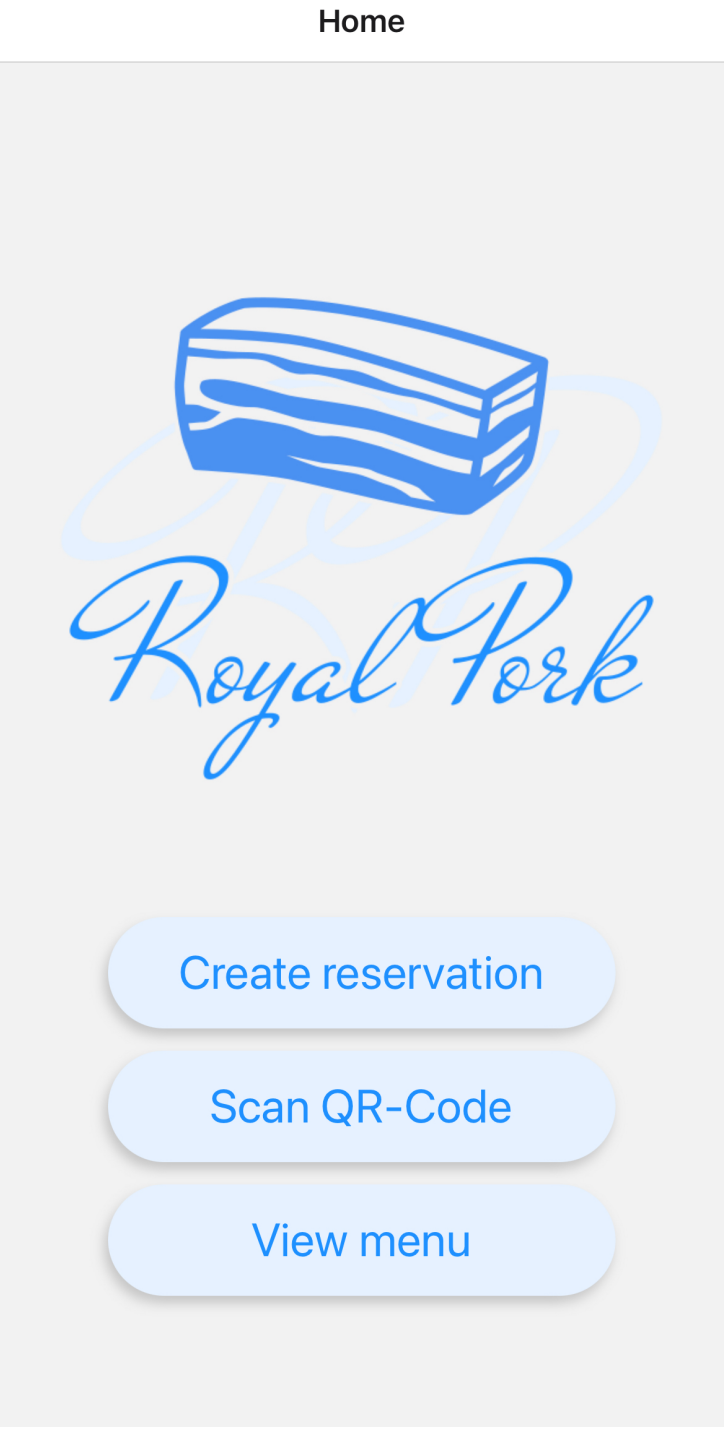
Komponenta **CreateCustomer** obsahuje GraphQL mutaci, která zapisuje zákazníky do systému Dotykačka.

```
const CREATE_CUSTOMER_MUTATION = gql`
  mutation CreateCustomer($customers: [CustomerInput!]!)
     {
    createCustomer(customers: $customers) @rest(
      type: "CustomerResponse"
      path: "clouds/373657584/customers"
      method: "POST"
      bodyKey: "customers"
    ) {
      id
      firstName //obsahuje aktualni cas
      lastName //obsahuje cislo stolu
    }
  }
`;
```

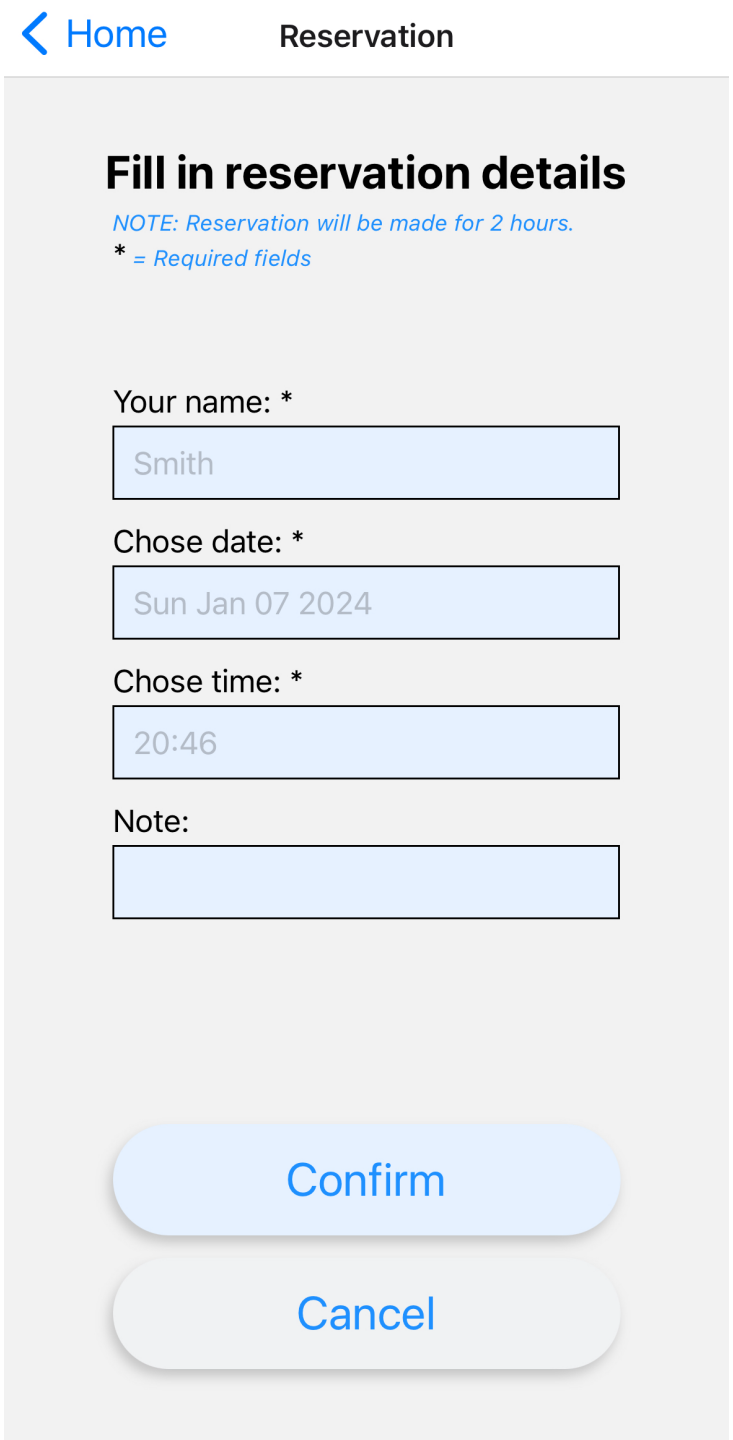
Úryvek kódu 6.18: CreateCustomer.js: GraphQL mutace

6.12 Výsledné grafické zpracování implementace

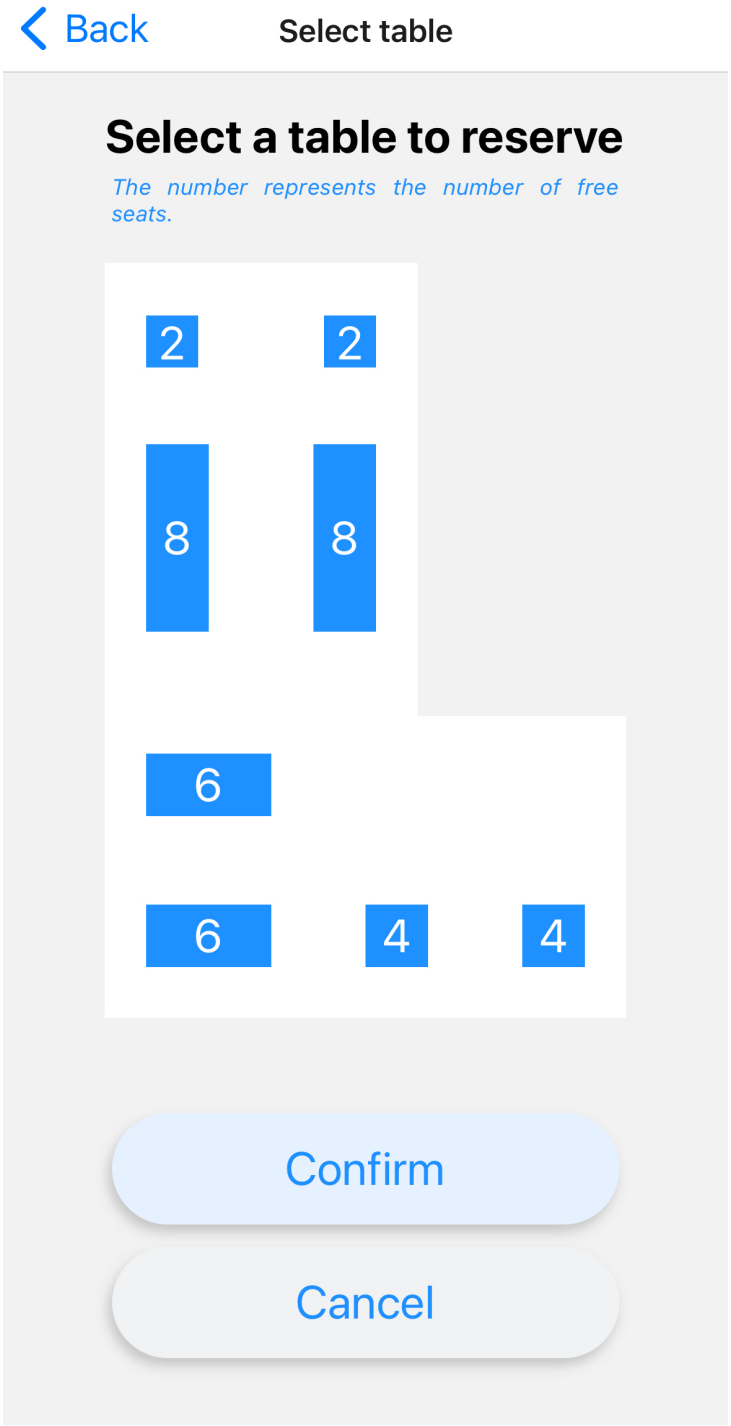
V této sekci bych chtěl předvézt ukázky některých obrazovek a jejich finálního vzhledu.



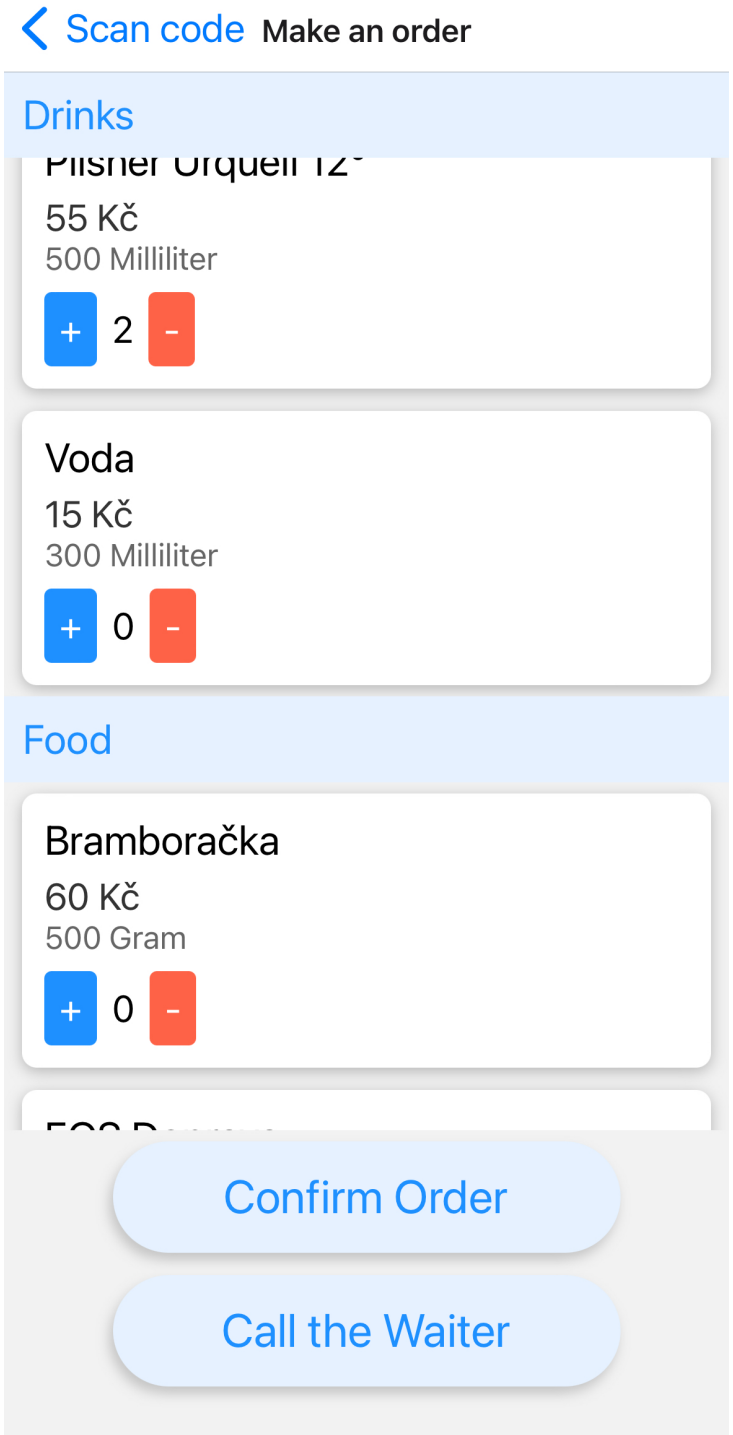
Obrázek 6.1: Domácí obrazovka



Obrázek 6.2: Obrazovka s rezervačním formulářem



Obrázek 6.3: Obrazovka k vybrání stolu



Obrázek 6.4: Obrazovka po načtení QR kódu

6.13 Souhrn implementace

Tato kapitola představuje různé aspekty implementace aplikace, od navigace a správy API až po konkrétní funkce jednotlivých komponent. Ačkoliv jsem se zaměřil na klíčové komponenty a procesy, je třeba zdůraznit, že aplikace obsahuje mnoho dalších funkcí a komponent, které jsou implementačně velmi podobné těm popsaným.

Kapitola 7
Testování

Aplikace pro podporu hosta v restauraci je v celku drobná aplikace, ale i přes to by měla být otestována. Existuje mnoho druhů testování a věcí, na které se testeři zaměřují. Některé v této aplikaci ale postrádají smysl. Například používaní nějakých složitých testovacích nástrojů je zbytečné, protože se jedná o velmi malou aplikaci.

Testování výkonnosti zde také úplně nedává smysl. Může se samozřejmě vyzkoušet, ale aplikace se v podstatě pouze dotazuje na odpovědi od pokladního systému, který je ten, kdo provádí náročnější výpočty. Testování výkonnosti by tedy bylo spíše testováním pokladního systému.

Při testování této aplikace bych se zaměřil na srozumitelnost a přehlednost uživatelského rozhraní. Dále na bezproblémový chod systému a funkčnost. Důležité je se zaměřit na scénáře, které se objeví při nějaké chybě. Například kdyby si uživatel chtěl zaregistrovat rezervaci na čas a místo, kde už rezervace existuje.

Jak bylo uvedeno výše, aplikaci nemá smysl testovat s nějakým automatickým nástrojem. Testování bude probíhat ručně. Jedná se o funkční testování. To znamená, že tester ověřuje, zda aplikace splňuje svou specifikaci. Dále zjišťuje chování aplikace při jednotlivých scénářích, které mohou v aplikaci nastat. Může například do formulářů vkládat i neplatná data a sledovat, zda aplikace tyto situace zvládne.

Dále bude aplikace obsahovat unit testy pro ověření správnosti jednotlivých drobnějších izolovaných částí kódu.

7.1 Testovací scénáře

7.1.1 Rezervace stolu

Cíl testu

Cílem je ověřit, že uživatel může úspěšně vytvořit rezervaci stolu v aplikaci a správně reagovat na neočekávané situace.

Předpoklady

- Uživatel má nainstalovanou aplikaci a přístup k internetu.
- Aplikace je propojena s pokladním systémem restaurace.

Úspěšná rezervace

1. Uživatel otevře aplikaci a přesune se na stránku pro rezervace.
2. Vybere datum, čas a počet osob pro rezervaci. Data potvrdí.
3. Vybere volný stůl s požadovaným počtem míst.
4. Potvrdí rezervaci a sleduje zpracování.
5. **Očekávaný výsledek:** Rezervace je úspěšně zapsána do systému a uživatel obdrží potvrzení ve formě vyskakovacího okna.

Neočekávané situace

Přerušení procesu rezervace.

- Uživatel neúmyslně opustí proces rezervace (např. zavřením aplikace).
- **Očekávaný výsledek:** Po opětovném spuštění aplikace se bude chovat jako by proces nebyl započat. Na pokladní systém nebude nic odesláno a uživatel bude muset začít rezervaci znovu.

Rezervace bez uvedení jména.

- Uživatel se pokusí vytvořit rezervaci bez uvedení jména, pod kterým má být rezervace vytvořena.
- **Očekávaný výsledek:** Aplikace nedovolí pokračovat v rezervaci a upozorní uživatele na neplatný čas hodící se varovnou hláškou.

Rezervace dříve než dvě hodiny dopředu.

- Uživatel se pokusí vytvořit rezervaci v kračím čase než jsou dvě hodiny od aktuálního času.
- **Očekávaný výsledek:** Aplikace nedovolí pokračovat v rezervaci a upozorní uživatele na neplatný čas hodící se varovnou hláškou.

Rezervace mimo otevírací hodiny.

- Uživatel se pokusí vytvořit rezervaci mimo otevírací hodiny restaurace.
- **Očekávaný výsledek:** Aplikace nedovolí pokračovat v rezervaci a upozorní uživatele na neplatný čas hodící se varovnou hláškou.

Rezervace na obsazený stůl.

- Ve fázi výběru stolu se uživatel pokusí rezervovat stůl, který je již zabraný.
- **Očekávaný výsledek:** Aplikace neumožní vybrat stůl, který je již obsazen. Na obrazovce je znázorněn šedou barvou a číslicí 0, která symbolizuje počet volných míst.

Vyhodnocení

Všechny testy proběhly úspěšně a potvrdily, že rezervační systém funguje správně a je odolný vůči neočekávaným situacím. Během testování jsem identifikoval několik oblastí pro zlepšení, zejména v oblasti chybových hlášek. Původní chybové hlášky nebyly dostatečně podrobné, což mohlo vést k nejasnostem pro uživatele. Po identifikaci tohoto problému jsem aktualizoval chybové hlášky, aby byly informativnější a konkrétnější, což zlepšuje uživatelskou zkušenost a pomáhá uživatelům lépe porozumět příčinám problémů při rezervaci.

7.1.2 Vytvoření objednávky

Cíl testu

Cílem je ověřit, že uživatel může v aplikaci úspěšně vytvořit objednávku z nabídky, případně ji i zaplatit, po načtení QR kódu, a zajistit, že aplikace adekvátně reaguje v případě chyb nebo neočekávaných situací během procesu objednávání.

Předpoklady

- Uživatel má nainstalovanou aplikaci, přístup k internetu, udělil aplikaci přístup k fotoaparátu a má možnost naskenovat platný QR kód.
- Aplikace je propojena s pokladním systémem restaurace.

Úspěšné vytvoření objednávky

1. Uživatel otevře aplikaci a přesune se na stránku načtení QR kódu.
2. Uživatel načte QR kód umístěný na stole.
3. Přejde k vytváření objednávky a vybere položky z menu.
4. Potvrdí objednávku a vybere způsob platby.
5. Pokud vybere platbu online, zobrazí se mu formulář platby. Následně to potvrdí.

6. **Očekávaný výsledek:** Objednávka je úspěšně odeslána do pokladního systému a uživatel obdrží potvrzení. To, zda je objednávka zaplacena, je určeno tím, jaký způsob platby uživatel zvolil. Pokud zvolil platbu u pokladny, tak je nezaplacená. Pokud však uživatel zvolil okamžitou platbu, objednávka je v systému označena jako zaplacená a je zahrnuta do tržeb.

Neočekávané situace

Naskenování vadného QR kódu.

- Uživatel naskenuje kód, který není správný v souladu se systémem.
- **Očekávaný výsledek:** Po naskenování vadného kódu aplikace vyhodí varovnou hlášku. Po odkliknutí hlášky začne aplikace znovu skenovat.

Přerušení procesu objednávky.

- Uživatel neúmyslně opustí proces objednávání v jakékoli fázi (např. zavřením aplikace).
- **Očekávaný výsledek:** Po opětovném spuštění aplikace se bude chovat jako by QR kód nebyl naskenován. Na pokladní systém nebude nic odesláno a uživatel bude muset začít objednávku znovu.

Pokus o založení prázdné objednávky.

- Uživatel se pokusí založit objednávku, aniž by do ní přidal nějaký předmět.
- **Očekávaný výsledek:** Aplikace neumožní uživateli pokračovat v objednávce. Konkrétně po kliknutí na tlačítko "Confirm"vyhodí varovnou hlášku.

Vyhodnocení

Testovací scénář proběhl v celku hladce a bez významnějších problémů. Systém byl schopen úspěšně zpracovávat objednávky a správně reagoval na různé uživatelské interakce. Během testování však byla identifikována možnost naskenování vadného QR kódu. Původně tento scénář nebyl zahrnut, ale během testování jsem si uvědomil důležitost této situace. Po identifikaci tohoto potenciálního problému jsem scénář rozšířil o testování reakce aplikace na naskenování vadného QR kódu. Tato změna pomohla zvýšit robustnost aplikace.

7.1.3 Přivolání obsluhy

Cíl testu

Cílem je ověřit, že uživatelé mohou v aplikaci úspěšně přivolat obsluhu po načtení QR kódu a zkontrolovat, jak aplikace reaguje na neočekávané situace spojené s funkcí přivolání obsluhy.

Předpoklady

- Uživatel má nainstalovanou aplikaci, přístup k internetu, udělil aplikaci přístup k fotoaparátu a má možnost naskenovat platný QR kód.
- Aplikace je propojena s pokladním systémem restaurace.

Úspěšné přivolání obsluhy

1. Uživatel otevře aplikaci a přesune se na stránku načtení QR kódu.
2. Uživatel načte QR kód umístěný na stole.
3. Přejde k možnosti přivolání obsluhy v aplikaci.
4. Využije funkci přivolání obsluhy.
5. Očekávaný výsledek: Aplikace registruje požadavek a zapíše ho do pokladního systému. Uživatel dostane potvrzovací hlášku a je navrácen na hlavní stránku.

Neočekávané situace. Tento scénář sdílí neočekávanou situaci se scénářem vytvoření objednávky. Jedná se o naskenování vadných QR kódů (viz sekce 7.2.2).

Přerušení procesu přivolání obsluhy.

- Uživatel neúmyslně opustí proces přivolání obsluhy (např. zavřením aplikace).
- **Očekávaný výsledek:** Po opětovném spuštění aplikace se bude chovat jako by QR kód nebyl naskenován. Na pokladní systém nebude nic odesláno a uživatel bude muset přivolat obsluhu znovu.

Vyhodnocení

Testování přivolání osluhy proběhlo bez zásadních problémů. Aplikace spolehlivě reagovala na pokyny uživatele a úspěšně registrovala požadavky na přivolání obsluhy do pokladního systému. Tato funkce sdílí část své funkcionality s vytvořením objednávky, konkrétně naskenování QR kódů. Zjištění z testování vytvoření objednávky", týkající se skenování vadného QR kódu, bylo relevantní i pro tento scénář a úspěšně aplikováno.

7.2 Unit testy

V této sekci bych chtěl ukázat nějaké unit testy, kterými jsem se snažil pokrýt aplikaci. Bylo to velmi obtížné, protože velká část obrazovek k načtení potřebuje poslat dotaz na pokladní systém a to je velmi obtížné namokovat.

Každý test používá **@testing-library/react-native** pro renderování komponent a **MockedProvider** z **@apollo/client/testing** pro simulaci odpovědí GraphQL serveru.

```
1  import React from 'react';
2  import {fireEvent, render} from '@testing-library/react-
       native';
3  import ReservationFormScreen from '../../screens/
       ReservationFormScreen';
4
5  describe('ReservationFormScreen', () => {
6  it('should display an error message if the name is empty
       on submit', () => {
7          const {getByText, getByTestId} = render(<
               ReservationFormScreen/>);
8          const submitButton = getByTestId('submit-button')
               ;
9
10         fireEvent.press(submitButton);
11         expect(getByText('Name must not be empty.')).
               toBeTruthy();
12     });
13
14     // dalsi testy
15 });
```

Úryvek kódu 7.1: PayOrder.test.js: test mutace placení

Tento test se zaměřuje na ověření funkcionality obrazovky **ReservationFormScreen**. Test kontroluje, zda se zobrazí chybová zpráva, pokud uživatel odešle formulář rezervace bez vyplnění jména.

Postup testu

1. Komponenta **ReservationFormScreen** je vykreslena.
2. Pomocí **getByTestId** se najde tlačítko pro odeslání formuláře.
3. Tlačítko je aktivováno pomocí **fireEvent.press**.
4. Test ověřuje, zda se zobrazí chybová zpráva s textem **"Name must not be empty."** (Jméno nesmí být prázdné).

```
import React from 'react';
import { render, waitFor } from '@testing-library/react-native';
import { MockedProvider } from '@apollo/client/testing';
import PayOrder, { PAY_ORDER_MUTATION } from '../../requests/PayOrder';

const mockOrderData = { ... };

const mockPayOrderResponse = { ... };

const mutationMock = { ... };

const onOrderPaidMock = jest.fn();

describe('PayOrder', () => {
    it('calls the onOrderPaid callback with true on successful mutation', async () => {
        const { getByText } = render(
            <MockedProvider mocks={[mutationMock]} addTypename={false}>
                <PayOrder orderData={mockOrderData} onOrderPaid={onOrderPaidMock} />
            </MockedProvider>
        );

        expect(getByText('Paying order...')).toBeTruthy();

        await waitFor(() => {
            expect(onOrderPaidMock).toHaveBeenCalledWith(true);
        });
    });
});
```

Úryvek kódu 7.2: ReservationFormScreen.test.js: test v rezervačním formuláři na nevyplněné jméno

Tento test ověřuje funkčnost komponenty **PayOrder** v React Native aplikaci, která je propojená s GraphQL pomocí Apollo Client. Test kontroluje, že když je mutace pro platbu objednávky úspěšná, zavolá se zpětné volání **onOrderPaid** s hodnotou **true**.

Postup testu

1. Komponenta **PayOrder** je vykreslena v testovacím prostředí. Pro simulaci GraphQL mutace je použit **MockedProvider** s předdefinovaným **mutationMock**.

2. Test nejprve ověřuje, zda se na obrazovce zobrazuje text **"Paying or-**

der...", což naznačuje, že proces platby je v průběhu.

3. Pomocí **waitFor** se čeká na dokončení asynchronní operace, tedy na vykonání GraphQL mutace.

4. Test následně kontroluje, zda bylo zavoláno zpětné volání **onOrderPaid** s argumentem **true**, což indikuje úspěšné dokončení mutace.

Kapitola 8

Závěr

V této bakalářské práci jsem se věnoval návrhu a implementaci mobilní aplikace pro podporu hostů v restauraci. Hlavním cílem bylo vytvořit aplikaci, která zefektivní procesy rezervace, objednávky a přivolání obsluhy a zároveň poskytne komfortní zážitek pro hosty restaurace. Tento projekt vyžadoval důkladnou rešerši stávajících řešení a trendů v oblasti mobilních aplikací pro restaurace, aby bylo možné lépe pochopit potřeby uživatelů a technologické možnosti. Na základě těchto informací byla aplikace navržena s důrazem na jednoduchost a intuitivitu, přičemž byla postavena na moderních technologiích React Native, Apollo Client a GraphQL.

Během implementace byl kladen důraz na interakci s pokladním systémem Dotykačka, který poskytoval základní platformu pro správu dat a transakcí v aplikaci. Během implementace jsem narazil na drobné komplikace spojené se sytémem a jeho rozhraním. Někdy to vedlo k drobnějšímu obcházení a to vedlo k malému zvýšení komplexity aplikace. Nakonec byl tento systém úspěšně integrován a řešení testováno, což potvrdilo funkčnost aplikace.

Důležitým aspektem práce bylo také testování aplikace, které se zaměřovalo na ověření funkčnosti klíčových scénářů a zajištění robustnosti aplikace v různých situacích. Testování potvrdilo, že aplikace je schopna správně reagovat na nejrůznější požadavky uživatelů.

V závěru lze konstatovat, že cíl práce byl úspěšně naplněn. Aplikace pro podporu hostů v restauraci byla vytvořena v souladu s požadavky a její funkčnost byla ověřena. Implementovaná řešení demonstrují efektivní využití moderních technologií.

Zdroje

[1] Portál MOJE daně. [online]. Generální finanční ředitelství, 2020 [cit. 2023-01-05]. Dostupné z: https://adisspr.mfcr.cz/pmd/home

[2] etržby - elektronická evidence tržeb. etržby - elektronická evidence tržeb [online]. Ministerstvo vnitra, 2020 [cit. 2023-01-05]. Dostupné z: https://www.etrzby.cz/

[3] Evidence tržeb: Metodický pokyn k aplikaci zákona o evidenci tržeb. [online]. Praha: Generální finanční ředitelství, 2016 [cit. 2023-01-05]. Dostupné z: https://www.etrzby.cz/assets/cs/prilohy/Metodika-k-evidenci-trzeb_v1.0.pdf

[4] Elektronická evidence tržeb: Formát a struktura údajů o evidované tržbě. [online]. Praha: Generální finanční ředitelství, 2016 [cit. 2023-01-05]. Dostupné z: https://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

[5] Srovnání pokladních systémů 2023 - srovnání, zkušenosti. Recenze, testy, zkušenosti a hodnocení produktů a služeb | 5nej.cz [online]. [cit. 2023-01-07]. Dostupné z: https://www.5nej.cz/srovnani-eet-pokladen/

[6] Dotykačka [online]. © 2021 Dotykačka ČR s.r.o [cit. 2023-01-07]. Dostupné z: https://dotykacka.cz/

[7] Odvětví | Dotykačka [online]. © 2021 Dotykačka ČR s.r.o [cit. 2023-01-07]. Dostupné z: https://dotykacka.cz/odvetvi/

[8] Pokladní systém pro gastro | Dotykačka [online]. © 2021 Dotykačka ČR s.r.o [cit. 2023-01-07]. Dostupné z: https://dotykacka.cz/pokladni-system-pro-gastro

[9] Introduction to API v2 - API. Introduction to API v2 - API [online]. © 2021 Dotykačka ČR s.r.o [cit. 2023-01-07]. Dostupné z: https://docs.api.dotypos.com/

[10] HELLOCASH - Online registrační pokladna zdarma. HELLOCASH - Online registrační pokladna zdarma [online]. Copyright © 2023

mRaP GmbH. Všechna práva vyhrazena. [cit. 2023-01-10]. Dostupné z: https://hellocash.cz/

[11] EET pokladna pro spokojené podnikání - miniPOS. EET pokladna pro spokojené podnikání - miniPOS [online]. Copyright © 2020 QUITEC [cit. 2023-01-10]. Dostupné z: https://www.minipos.cz/

[12] KASA FIK | Platební systém pro akce, koncerty a festivaly - KASA FIK. [online]. KASA FIK s.r.o. [cit. 2023-01-10]. Dostupné z: https://www.kasafik.cz/web/cs/platebni-systemy/

[13] UBER. Uber Eats v České republice. Online. 2020. Dostupné z: https://www.uber.com/cs-CZ/newsroom/uber-eats-v-ceske-republice/. [cit. 2024-01-06].

[14] UBER TECHNOLOGIES INC. UberEats [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://www.ubereats.com

[15] MONOUSO BLOG. 15 nejlepších platforem pro doručování do vaší restaurace [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://blog.monouso.cz/nejlepsich-platforem

[16] Cafe Prostoru_. Cafe Prostoru_[online]. Copyright © 2015 Cafe Prostoru [cit. 2023-01-10]. Dostupné z: https://cafe.prostoru.cz/

[17] Hotel Luční bouda - ON-LINE rezervace. [online]. Copyright © Hotel Luční bouda 2020 [cit. 2023-01-10]. Dostupné z: https://www.lucnibouda.cz/cs/

[18] Inteligentní stůl pro restaurace - iKelp POS Mobile. Reštauračný systém iKelp POS Mobile - iKelp POS Mobile [online]. Copyright © 2009 [cit. 2023-01-10]. Dostupné z: http://www.ikelp.com/cz/funkce/inteligentni-stul?epid=1783&sett=1317

[19] Qerko | Nejpohodlnější způsob, jak platit v restauracích. Qerko | QR payments for restaurants [online]. Qerko® [cit. 2023-01-10]. Dostupné z: https://www.qerko.com/cs

[20] QERKO. Qerko [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://www.qerko.com

[21] CZECH NEWS CENTER A.S. A DODAVATELÉ OBSAHU. Aplikace Qerko dostává novou funkci. Můžete si s ní rezervovat místo v oblíbené restauraci. Online. 2024. Dostupné z: https://mobilmania.zive.cz/clanky/aplikace-qerko-dostava-novou-funkci-muzete-si-s-ni-rezervovat-misto-v-oblibene-restauraci/sc-3-a-1355384/default.aspx. [cit. 2024-01-06].

[22] React Native · Learn once, write anywhere. React Native · Learn once, write anywhere [online]. Copyright © 2023 Meta Platforms, Inc. [cit. 2023-01-16]. Dostupné z: https://reactnative.dev/

[23] 1. What Is React Native? - Learning React Native [Book]. O'Reilly Media - Technology and Business Training [online]. Copyright © 2023, O [cit. 2023-01-16]. Dostupné z: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

[24] Expo. Expo [online]. Copyright © [cit. 2023-01-16]. Dostupné z: https://expo.dev/

[25] GitHub - expo/expo: An open-source platform for making universal native apps with React. Expo runs on Android, iOS, and the web.. GitHub: Let's build from here · GitHub [online]. Copyright © 2023 GitHub, Inc. [cit. 2023-01-16]. Dostupné z: https://github.com/expo/expo

[26] Should I Use Expo For React-Native (5 Minute Read) | Upstack. Hire Skilled Remote Developers | Risk Free | UpStack [online]. ©2021 UpStack Technologies, Inc [cit. 2023-01-16]. Dostupné z: https://upstackhq.com/blog/should-i-use-expo-for-react-native

[27] BarCodeScanner - Expo Documentation. Expo Documentation [online]. Copyright © [cit. 2023-01-16]. Dostupné z: https://docs.expo.dev/versions/latest/sdk/bar-code-scanner/

[28] Get started with Apollo Client - Apollo GraphQL Docs. Apollo GraphQL | Supergraph: unify APIs, microservices, & databases in a composable graph [online]. Copyright © Apollo Graph Inc. [cit. 2023-01-16]. Dostupné z: https://www.apollographql.com/docs/react/get-started/

[29] Apollo Client GraphQL Setup | GraphQL React Native Apollo Tutorial. Instant GraphQL APIs on your data | Built-in Authz & Caching [online]. Copyright © [cit. 2023-01-16]. Dostupné z: https://hasura.io/learn/graphql/react-native/apollo-client/

[30] GraphQL | A query language for your API. GraphQL | A query language for your API [online]. Copyright © [cit. 2023-01-16]. Dostupné z: https://graphql.org/

[31] GraphQL vs REST – Difference Between APIs | GURU99 [online]. © Copyright - Guru99 2023 [cit. 2023-01-18]. Dostupné z: https://www.guru99.com/graphql-vs-rest-apis.html

[32] Lekce 2 - React Native - První aplikace. itnetwork.cz - Učíme národ IT [online]. Copyright © 2023 itnetwork.cz. Veškerý obsah webu [cit. 2023-01-18]. Dostupné z: https://www.itnetwork.cz/javascript/react/native/react-native-prvni-aplikace

[33] REDSWITCHES PTE. LTD. Flutter vs React Native: The Best Mobile App Development Tool in 2024. Online. 2023. Dostupné z: https://www.redswitches.com/blog/flutter-vs-react-native/. [cit. 2024-01-06].

[34] FINGENT. React Native, Flutter, Ionic, Xamarin - A Comparison Between The Top Mobile App Development Frameworks. Online. 2024. Dostupné z: https://www.fingent.com/blog/react-native-flutter-ionic-xamarin-a-comparison-between-the-top-mobile-app-development-frameworks/. [cit. 2024-01-06].

[35] BIZ4SOLUTIONS LLC. Comparing React Native To Other Trending Cross-Platform App Development Technologies!. Online. 2021. Dostupné z: https://www.biz4solutions.com/blog/comparing-react-native-to-other-trending-cross-platform-app-development-technologies/. [cit. 2024-01-06].

[36] NETSET SOFTWARE SOLUTIONS. Top Cross-Platform Mobile Frameworks' Comparison: React Native vs Xamarin vs Flutter. Online. 2022. Dostupné z: https://www.netsetsoftware.com/insights/top-cross-platform-mobile-frameworks-comparison-react-native-vs-xamarin-vs-flutter/. [cit. 2024-01-06].

[37] PAGEPRO LTD. React & React Native Development Agency. Online. 2024. Dostupné z: https://pagepro.co. [cit. 2024-01-06].

[38] MINDK INC. Mindk. Online. 2024. Dostupné z: https://www.mindk.com. [cit. 2024-01-06].

[39] THIRDROCKTECHKNO. Thirdrocktechkno. Online. 2020. Dostupné z: https://www.thirdrocktechkno.com. [cit. 2024-01-06].

[40] KOTLIN™. Kotlin Concise. Multiplatform. Fun. [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://kotlinlang.org

[41] INNOVINS. Kotlin Programming Language Advantages and Disadvantages. Online. 2023. Dostupné z: https://www.innovins.com/kotlin-programming-language-advantages-and-disadvantages/. [cit. 2024-01-06].

[42] FLUTTER. Flutter| Build for any screen. Online. 2024. Dostupné z: https://flutter.dev. [cit. 2024-01-06].

[43] LOGROCKET BLOG. LogRocket Blog. Online. 2024. Dostupné z: https://blog.logrocket.com/. [cit. 2024-01-06].

[44] WAVERLEY SOFTWARE INC. WHY USE FLUTTER: PROS AND CONS OF FLUTTER APP DEVELOPMENT. Online. 2024. Dostupné z: https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/. [cit. 2024-01-06].

[45] IONIC. The mobile SDK for the Web. Online. 2024. Dostupné z: https://ionicframework.com. [cit. 2024-01-06].

[46] SOFTJOURN, INC. Ionic App Development: Advantages and Disadvantages. Online. 2023. Dostupné z: https://softjourn.com/insights/ionic-app-development-advantages-and-disadvantages. [cit. 2024-01-06].

[47] MICROSOFT. Xamarin. Online. 2024. Dostupné z: https://dotnet.microsoft.com/en-us/apps/xamarin. [cit. 2024-01-06].

[48] MICROSOFT. Visual Studio Tools for Xamarin. Online. 2023. Dostupné z: https://visualstudio.microsoft.com/cs/xamarin/. [cit. 2024-01-06].

[49] MICROSOFT. Xamarin documentation. Online. 2023. Dostupné z: https://learn.microsoft.com/en-us/xamarin/. [cit. 2024-01-06].

[50] THE APACHE SOFTWARE FOUNDATION. Apache Cordova. Online. 2024. Dostupné z: https://cordova.apache.org. [cit. 2024-01-06].

[51] GITHUB, INC. Apache Cordova CLI. Online. 2024. Dostupné z: https://github.com/apache/cordova-cli. [cit. 2024-01-06].

[52] UMLÁŠEK, Prokop. Aplikace pro podporu hosta v restauraci. Praha, 2023. Semestrální projekt. České vysoké učení technické v Praze.

[53] JGRAPH LTD. Draw.io. Online. 2023. Dostupné z: https://www.drawio.com. [cit. 2024-01-06].

[54] LOGOMASTER.AI. Logomaster.ai. Online. 2024. Dostupné z: https://logomaster.ai/cs/. [cit. 2024-01-06].

[55] Apple Pay. Online. In: ICONS8 LLC. ICONS8. 2024. Dostupné z: https://icons8.com/icon/61469/apple-pay. [cit. 2024-01-08].

[56] Google Pay. Online. In: ICONS8 LLC. ICONS8. 2024. Dostupné z: https://icons8.com/icon/vizert0k77Jn/google-pay. [cit. 2024-01-08].

[57] CITACE.COM, S.R.O. Citace PRO. Online. 2023. Dostupné z: https://www.citacepro.com/. [cit. 2024-01-08].