Bakalářská práce

České
vysoké
učení technické
v Praze

F3 Fakulta elektrotechnická

Administrační systém pro restaurace využívající systém Qerko

Autor: Robert Mysliveček

Vedoucí: Bc. Petr Huťťák

Studijní program: Softwarové inženýrství a technologie

Leden 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mysliveček** Jméno: **Robert** Osobní číslo: **466004**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Administrační systém pro restaurace využívající systém Qerko

Název bakalářské práce anglicky:

Administration system for restaurants using Qerko system

Pokyny pro vypracování:

Cílem projektu je vytvořit nové administrační rozhraní v ReactJS, které nahradí stávající rozhraní napsané v AngularJS. Toto nové rozhraní bude responzivní Single Page Aplikace, komunikace se serverem bude přes REST rozhraní. Web bude poskytovat podporu pro PWA (Progressive Web App).
Systém bude restauracím poskytovat následující služby:
- Správa stolů a k nim přiřazených QR kódů.
- Základní analýza využití aplikace Qerko zákazníky.
- Správa uživatelů systému.
- Historie plateb v restauraci přes aplikaci Qerko.
- Zobrazení hodnocení přes aplikaci Qerko.
- Stahování pdf vyúčtování.
- Změny profilu restaurace.
- Vytvoření a editaci věrnostního programu.

Systém bude zaměstnancům firmy poskytovat následující služby:
- Vytvoření nové restaurace.
- Výpis všech restaurací a jejich editace.
- Výpis všech uživatelů systému.
- Výpis všech plateb přes systém Qerko.
- Výpis všech objednávek přes systém Qerko.
- Výpis všech vygenerovaných QR kódů.

Seznam doporučené literatury:

[1] Artemij Fedosejev, React.js Essentials, 2015
[2] Bertoli M. React Design Patterns and Best Practices, 2017
[3] Adam Horton, Ryan Vice, Mastering React, 2016
[4] Dennis Sheppard, Beginning Progressive Web App Development, 2017

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Bc. Petr Huťťák, katedra počítačové grafiky a interakce FEL

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

Datum zadání bakalářské práce: **06.10.2020** Termín odevzdání bakalářské práce: **05.01.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Bc. Petr Huťťák podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

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, 4. ledna 2021

Robert Mysliveček

Poděkování

Velice rád bych poděkoval vedoucímu práce Bc. Petru Huťákovi za cenné připomínky a vedení práce. Dále bych rád poděkoval Ing. Lukáši Kovačovi a všem členům vývojového týmu společnosti Qerko s.r.o. za skvělou spolupráci. V neposlední řadě děkuji své rodině za podporu mého studia.

Abstrakt

Cílem této práce je vytvořit nové administrační rozhraní v ReactJS, které nahradí stávající rozhraní implementované v AngularJS. Nové rozhraní je responzivní Single Page Aplikace. Komunikace aplikace se serverem je zajištěna přes Representational State Transfer, dále jen REST rozhraní. V práci je obsaženo vysvětlení příslušných pojmů a technologií spojených s bakalářskou prací. Práce je zaměřena na problematiku frameworků ReactJS a AngularJS. Dokument obsahuje popis platformy Qerko od společnosti Qerko s.r.o. pro kterou je administrační rozhraní vyvinuto. Před vývojem tohoto projektu je analýza projektu, která je zde také uvedena. Analýza se zaměřuje na požadavky, a to jak funkční, tak business.

Klíčová slova – SPA, PWA, administrační rozhraní, React, REST, Next.js

Abstract

The aim of this bachelor‘s thesis is to create a new administration interface in ReactJS, which is intended to replace the existing interface implemented in AngularJS. The new interface is a responsive Single Page Application. Communication of the application with the server is ensured via the Representational State Transfer (REST). The thesis explains the relevant concepts and technologies associated with the bachelor‘s thesis. The work is focused on the ReactJS and AngularJS type of frameworks. The document provides a description of the Qerko platform from Qerko s.r.o. for which the administration interface is developed. The development of this project was preceded by a project analysis, which is also presented herein. The analysis is focused on both functional and business requirements.

Keywords– SPA, PWA, administration interface, React, REST, Next.js

Obsah

1 Úvod

Hlavním cílem této bakalářské práce je návrh a implementace administračního rozhraní pro restaurace, které využívají platformu Qerko, z toho vyplývá aktualizace stávajícího rozhraní, které je napsáno v Javascriptovém frameworku AngularJS. Nové rozhraní bude přepsáno do ReactJS využívající technologie TypeScript a Progressive Web Apps (dále jen PWA). Budou zde vysvětleny pojmy a technologie související s prací.

Před implementací rozhraní je potřeba zanalyzovat stav a vytvořit návrh tohoto rozhraní. Proto zde budou uvedeny a popsány požadavky na nové rozhraní. Při implementaci byly použity knihovny a další technologie, které zde budou také popsány. Webová aplikace bude responzivní Single Page aplikace s podporou více jazyků.

2 Popis současného stavu (AS-IS)

V této časti práce představím platformě Qerko, pro jejíž potřeby je administrační rozhraní navrhnuto a následně vyvinuto. Tato kapitola popisuje službu Qerko, jak po business stránce, tak po stránce technologické. Zde je uvedeno k čemu aplikace Qerko slouží a jaké má hlavní výhody, jak pro zákazníky restaurací, kaváren či jiných zařízení, tak pro personál těchto zařízení. V kapitole jsou definovány hlavní procesy, které chce firma pomocí administračního rozhraní řešit.

2.1 Platforma a společnost Qerko

Společnost byla založena na konci roku 2017 s názvem „It Is Paid s.r.o." Miloslavem Mrvíkem, Romanem Kantorem, Janem Dorazilem a Lukášem Kovačem, který je současně i jednatelem společnosti. V únoru roku 2020 se firma přejmenovala na dnes známé Qerko, které vydalo stejnojmennou aplikaci. Qerko je aplikace vytvořená pro mobilní zařízení s operačním systémem Android a IOS. [32]

Obrázek 1 Logo společnosti Qerko s.r.o. [31]

2.1.1 Výhody pro uživatele

Aplikace umožňuje zákazníkům restaurací zaplatit rychle, bezpečně a bezkontaktně útratu v restauracích bez přítomnosti obsluhy. Zákazník tak může pohodlně zaplatit prostřednictvím QR platby v aplikaci a odejít bez čekání. Aplikace umožňuje také zaplatit část útraty. Pokud je v restauraci skupinka lidí, kteří chtějí zaplatit každý svoji část útraty, a přitom sedí u jednoho stolu, není to problém. Útratou v zařízeních uživatel sbírá body do věrnostního programu, který aplikace nabízí. Veškeré bonusy jsou tak přehledně k dispozici a nejsou potřeba kartičky nebo voucher. Po zaplacení účet není vytisknut, ale zaslán elektronicky na e-mail uživatele.

Zaplacení je možné více způsoby. Platforma nabízí široký sortiment platebních metod, ke kterým patří i platba stravenkovou kartou, či věrnostní kartou. [31]

2.1.2 Výhody pro restaurace

Hlavním benefitem pro restaurace je samozřejmě úspora času obsluhy. Jelikož hosté zaplatí pomocí QR kódu, který je na stole bez nutnosti volat obsluhu, personál může získaný čas věnovat efektivněji. Obsluha se nemusí zabývat účtenkami, jelikož vše je řešeno elektronickou formou.

Z pohledu věrnostního programu zařízení nemusí zařizovat papírové kartičky a řešit administrativu. S digitálním věrnostním programem má zákazník vše u sebe a bonusy si při dosažení aktivuje sám. Díky aplikaci zařízení získají zpětnou vazbu od jejich zákazníků, které mohou být důležité pro následný rozvoj. [31]

2.2 Administrační rozhraní

V této podkapitole je stručně popsáno, proč je administrační rozhraní zapotřebí, jak pro obsluhu, tak i pro majitele a zaměstnance firmy Qerko.

2.2.1 Pro obsluhu

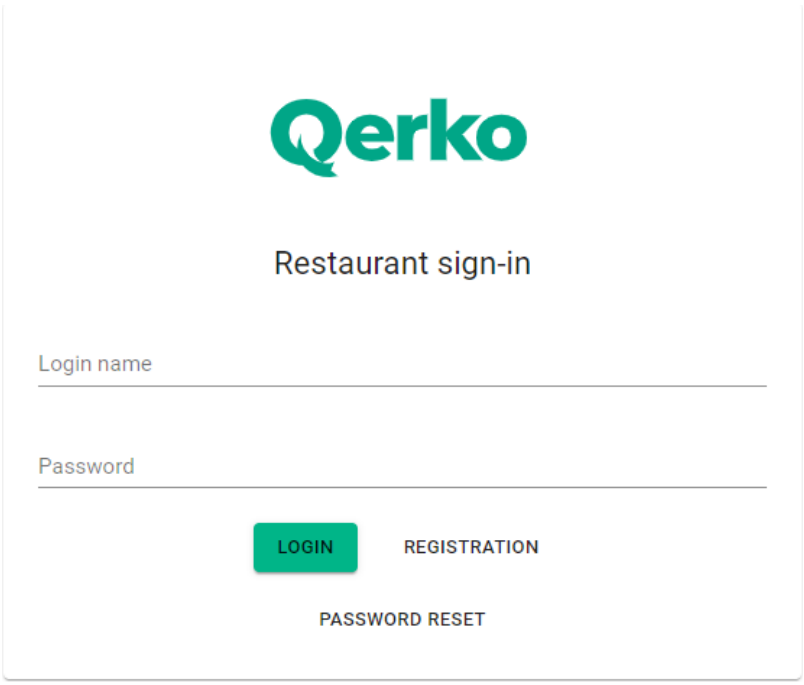
Administrační rozhraní nabízí obsluze podniků zobrazení informací o platbách za posledních 7 dní k aktuálnímu datu. Obsluha tak může vidět informace o tom, jakou částku zákazníci utratili a jaká je výše spropitného, která jim pak bude vyplácena. V případě, že nastane problém s platbou, obsluha se může podívat do administračního rozhraní, a zjistit, zda platba skutečně proběhla a dle stavu platby dále jednat.

2.2.2 Pro majitele

Toto rozhraní je určené hlavně pro majitele restaurací a account management společnosti Qerko. Pro provozovatele je velmi důležité vědět, za jaké období měl podnik, profit či ztrátu a v jaké výši. Administrační rozhraní v tomto ohledu může majiteli restaurace velmi pomoci, protože umožňuje zobrazit platby zákazníků dle stanoveného období. Provozovatel tak snadno zjistí, kolik zákazníci utratili peněz v různých časových intervalech. Dále také zjistí, v jakých hodinách je restaurace nejvíce vytížená a dle informací mohou upravit provoz restaurace, či vytvořit jiné změny. Administrační rozhraní tak dává provozovatelům restaurací přehled o chodu jejich podniků. Tyto zásadní informace jsou v dnešní době a množství konkurence velkou výhodou.

2.2.3 Account management

Pro společnost Qerko s.r.o. je rozhraní důležitou součástí nasazení Qerko aplikace do restaurací. Proto, aby restaurace podporovaly danou službu, je zapotřebí v systému restauraci zaregistrovat a přiřadit k ní svého majitele. QR kódy, které má restaurace, na stolech jsou zapotřebí vygenerovat a přiřadit k danému podniku a ke konkrétnímu stolu. Některé QR kódy mohou být zároveň NFC čipy a je nutné k danému QR kódu napsat vlastnost a detailnější popis pro zpřehlednění a praktičnost. Protože společnost Qerko nevyrábí pokladny, je zapotřebí každému podniku vygenerovat API klíč, díky kterému se služba spáruje s pokladnou v podniku. Administrační rozhraní tak slouží zaměstnancům Qerko pro zjištění případných chyb, které mohou nastat, a pro nastavení restaurací pro zákazníky.



Obrázek 2 Přihlášení do administračního rozhraní

3 Analýza

Pro správnou implementaci práce je nutné provést analýzu požadavků. V této kapitole jsou rozepsány požadavky dle zadání práce. Následně jsou požadavky rozděleny do menších funkcí systému, dle kterých je vytvořen diagram případů užití (také jako Use Case diagram).

3.1 Požadavky na systém

Níže je uveden seznam požadavků, které byly součástí zadání bakalářské práce. U každého požadavku je popis, který vystihuje hlavní funkce, které je nutné vytvořit. Dle zadání jsou požadavky rozděleny do těchto dvou kategorií.

3.1.1 Co bude systém poskytovat restauracím?

Tato kategorie obsahuje všechny funkce, které musí administrační rozhraní poskytovat restauracím.

3.1.1.1 Správu stolů a k nim přiřazených QR kódů

Každý stůl v restauraci má svůj unikátní QR kód, přes který zákazníci restaurace mohou zaplatit účet nebo jeho část. Aby restaurace měly přehled o svých stolech a jejich QR kódech, administrační rozhraní bude podporovat jejich správu. Přes REST rozhraní se načtou stoly z pokladny a lze k nim přidělit QR kód. Pokud pokladní systém nepodporuje mapu stolů, lze vytvořit ve webovém rozhraní stůl.

3.1.1.2 Základní analýzu využití aplikace Qerko zákazníky

Pro přehled zákazníků systém nabízí manažerům restaurací zobrazení statistiky jejich zákazníků. V daném období se zobrazí počet nových zákazníků, celkový pokles nebo růst a spokojenost. Systém nabízí také možnost zobrazení věku zákazníků.

3.1.1.3 Správa uživatelů systému

Pro přehled uživatelů systému nabízí administrační rozhraní manažerům restaurací výpis a editaci uživatelů systému. Manažer si může nechat vypsat jména a login uživatelů, kteří se systémem pracují v dané restauraci. Dále může přidat oprávnění pro obsluhu či pozvat a vymazat uživatele.

3.1.1.4 Historie plateb v restauraci přes aplikaci Qerko

Pro přehled manažera restaurace o útratě zákazníků prostřednictvím aplikace Qerko, systém nabízí možnost vypsání plateb a zobrazení přehledu plateb. Manažer si může zvolit období, které bude chtít vypsat, a následně zjistí informace o tržbách a spropitném v čase nebo si vypíše seznam plateb a zjistí bližší informace k nim. Tyto informace může získat i obsluha, která dostala oprávnění, ale jen za posledních 7 dní.

3.1.1.5 Zobrazení hodnocení zákazníků přes aplikaci Qerko

Manažer restaurace také může v rámci časového intervalu získat zpětnou vazbu od zákazníků. Jsou zde uvedené parametry hodnocení jídla, obsluhy a prostředí restaurace.

3.1.1.6 Možnost stažení vyúčtování

Je důležité, aby restaurace měly přehled o vyúčtování. Proto administrační rozhraní umožňuje výpis všech vyúčtování, a to buď denní nebo měsíční, které lze stáhnout. Dostupné formáty jsou pdf a csv.

3.1.1.7 Možnost editace profilu restaurace

Pokud by zaregistrovaná restaurace měla potřebu změnit základní údaje, např. adresu, otevírací dobu či logo, systém nabízí tyto základní informace kdykoliv změnit.

3.1.1.8 Vytvoření a editaci věrnostního programu

Poslední službou je nastavení věrnostního programu pro zákazníky. V rozhraní je možné si vybrat z daných věrnostních programů, ty lze upravit restauraci na míru a poté aktivovat pro zákazníky. V případě nespokojenosti nebo potřeby změnit či zrušit daný věrnostní program může manažer program deaktivovat a upravit na jiný či vybrat nový.

3.1.2 Co bude systém poskytovat zaměstnancům společnosti?

V této kategorii jsou rozepsány služby, které administrační rozhraní bude poskytovat zaměstnancům společnosti.

3.1.2.1 Vytvoření nové restaurace

Pro zaregistrování restaurace do systému je třeba ji v systému vytvořit. Při vytváření se zadají základní údaje o restauraci a přidělí se QR kódy. Manažer restaurace si pak může v rozhraní údaje přepsat, případně doplnit.

3.1.2.2 Výpis všech restaurací a jejich editace

Systém bude poskytovat přehled všech restaurací. Uživatel pak může restauraci upravit, debugovat (vidí komunikaci mezi Qerkem a pokladním systémem) či deaktivovat. Pro zúžení výběru restaurací je k dispozici filtr.

3.1.2.3 Zobrazení všech uživatelů systému

Uživatel si může zobrazit také seznam všech uživatelů, který se dá, jako seznam všech restaurací, filtrovat.

3.1.2.4 Výpis všech plateb probíhající přes platformu Qerko

V administračním rozhraní je možné vypsat veškeré platby ze všech restaurací. Kvůli nadměrnému množství dat je zapotřebí rozsáhlejší filtr. U každé platby je možné stáhnout účtenku ve formátu pdf.

3.1.2.5 Zobrazení všech objednávek, které vznikly prostřednictvím aplikace Qerko

Prostřednictví administračního rozhraní se dají získat, podobně jako platby, i všechny objednávky, které proběhly přes aplikaci Qerko. Na rozdíl od plateb, objednávky mají různý stav.

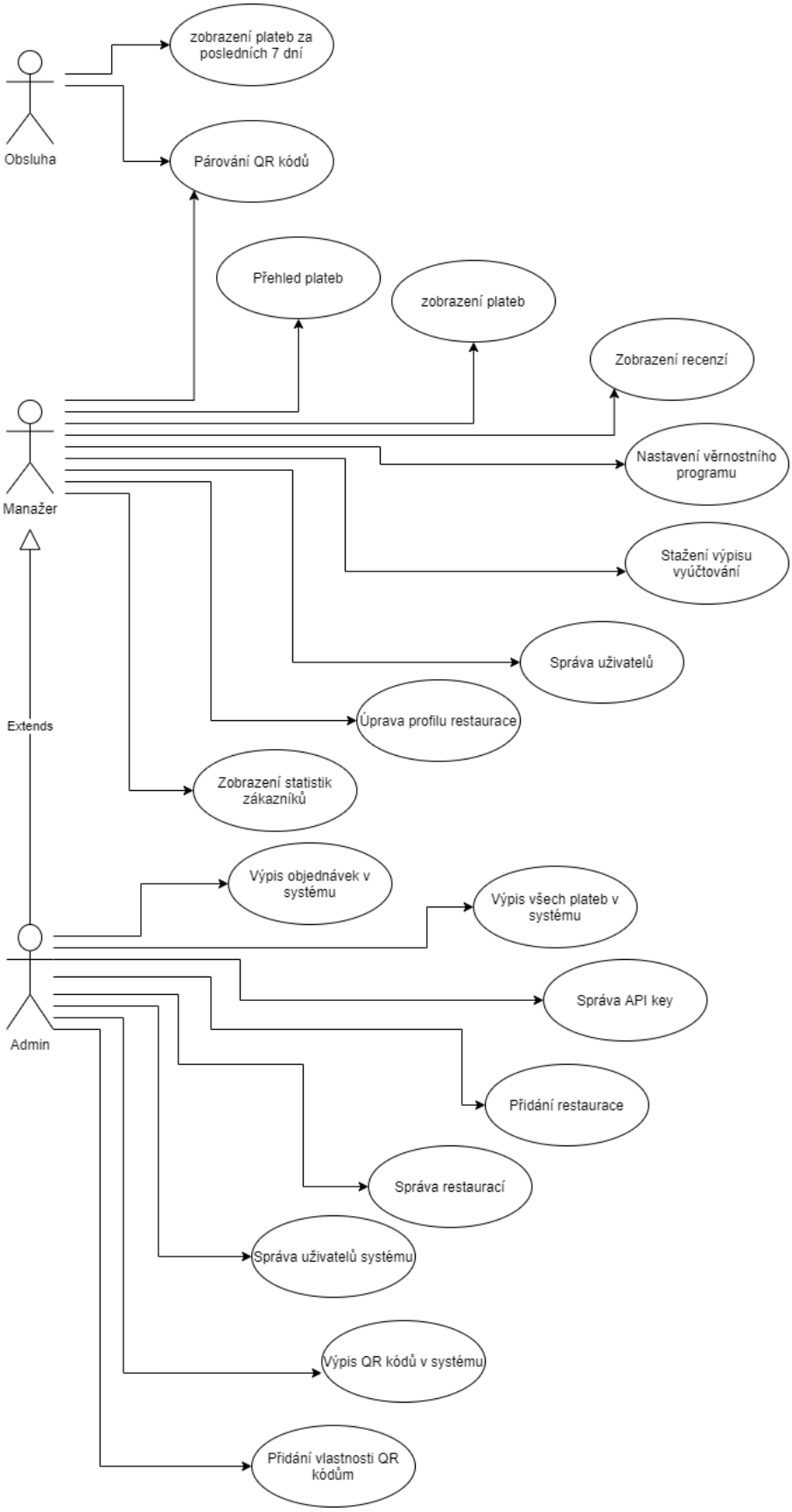
3.1.2.6 Výpis všech vygenerovaných QR kódů

Posledním požadavkem je výpis všech vygenerovaných QR kódů. Každá restaurace potřebuje pro každý stůl QR kód. Vygenerované QR kódy musí být unikátní. Pro přehled QR kódů je v systému výpis, který ukazuje všechny vygenerované QR kódy a jejich přiřazení.

3.2 Rozdělení požadavků

Dle prvotních požadavků byly vytvořeny menší služby, které musí administrační rozhraní poskytovat. Kompletní rozdělení je zachyceno na diagramu případu užití a níže jsou jednotlivé služby popsány dle funkcí.

3.2.1 Případy užití



Obrázek 3 Případy užití

3.2.2 Popis případů užití

3.2.2.1 Zobrazení plateb za posledních 7 dní

Obsluha si může zobrazit souhrn veškerých plateb včetně spropitného za určitou dobu, avšak maximálně za poslední týden zpět.

3.2.2.2 Párování QR kódů

Z pokladního systému se získá seznam stolů v restauraci, ke kterým se následně v systému přidělí QR kód z rozsahu QR kódů přidělených restauraci. Pro pokladní systémy, které nemají seznam stolů, lze stoly přidat manuálně a však ID stolu musí korespondovat s ID v pokladním systému.

3.2.2.3 Přehled plateb

Manažer společně s administrátorem mohou získat přehled o tržbách a spropitném v jednotlivém časovém horizontu. Uživatel zde zjistí průměrné hodnocení za vybrané období, celkové tržby a spropitné.

3.2.2.4 Zobrazení plateb

Za dané období, včetně informací o platbách, jako jsou platební metody nebo výše platby, informuje výpis plateb. Tyto platby lze filtrovat dle stolu, platební metody a stavu platby.

3.2.2.5 Zobrazení recenzí

Výpis hodnocení nabízí restauraci zpětnou vazbu. Hodnocení se skládá ze tří částí, hodnocení pokrmu, obsluhy a celkového dojmu z prostředí restaurace. U každého hodnocení je uvedená objednávka v restauraci a datum. V případě negativního hodnocení může zákazník přidat komentář.

3.2.2.6 Nastavení věrnostního programu

V této části je možné vytvořit individuální věrnostní program pro zákazníky restaurace či skupinu restaurací. Při vytvoření programu lze zvolit ze tří typů programu.

- Zákazník za určitý počet návštěv získá slevu na příští návštěvu a to, buď procentuální, či v absolutní částce.
- Zákazník za určitý počet nakoupených položek získá poukaz na tu samou položku zdarma či s procentuální slevou.
- Zákazník za určitý počet návštěv získá poukaz na vybraný produkt restaurace při další návštěvě.

3.2.2.7 Stažení výpisu vyúčtování

Majitel a administrátor mají možnost stažení měsíčního/denního výpisu vyúčtování ve formátu PDF a CSV. Jsou zde informace o období vyúčtování a datum vytvoření.

3.2.2.8 Správa uživatelů

Zde je možná editace a výpis správců určité restaurace. Lze zde vytvořit přístup pro obsluhu. Při výpisu správců jsou uvedena jména a login správců.

3.2.2.9 Úprava profilu restaurace

Zde je možná editace profilu restaurace a údajů o ní. Manažer zde může měnit logo restaurace, otevírací dobu, typ restaurace či přidat odkaz na webové stránky.

3.2.2.10 Zobrazení statistik zákazníků

Zde se zobrazují informace o poklesu či nárůstu zákazníků v restauraci za zvolené období. Informace se zobrazují např. v podobě grafu poměru nových a stávajících zákazníků jak za zvolené období, tak i za jednotlivé dny.

3.2.2.11 Výpis objednávek v systému

Zde je možné zobrazit výpis veškerých objednávek ve všech restauracích. Výběr lze filtrovat dle každého z uvedených parametrů. Mezi nejdůležitější parametry zde lze považovat název restaurace nebo stav objednávky.

3.2.2.12 Výpis všech plateb v systému

Zde lze zobrazit výpis veškerých plateb ve všech restauracích. Výběr lze filtrovat dle každého z uvedených parametrů. Mezi nejdůležitější parametry zde lze považovat název restaurace, platební brána a stav platby.

3.2.2.13 Správa API klíčů

Zde je funkcionalita pro vygenerování API klíče, se kterým se pokladna autorizuje v systému.

3.2.2.14 Přidání restaurace

Administrátor má možnost přidat restauraci do systému pomocí formuláře, kam zadá veškeré potřebné informace o restauraci.

3.2.2.15 Správa restaurací

Zde je možné zobrazit výpis veškerých restaurací a jejich informací, možnost editace každé restaurace a zobrazení komunikace s pokladním systémem.

3.2.2.16 Správa uživatelů systému

Zde je možné zobrazit všechny uživatele platformy Qerko.

3.2.2.17 Výpis QR kódů v systému

Zde je možné zobrazit výpis QR kódů a k nim přiřazené restaurace se stoly. Výběr lze filtrovat dle všech kritérií.

3.2.2.18 Přidání vlastnosti QR kódů

Zde je možné zobrazit výpis všech vlastností QR kódů v daném rozsahu a možnost nastavení nové vlastnosti QR kódů v zadaném rozsahu.

4 Technologie

V této časti bakalářské práce se budu zabývat výběrem technologií a následnou implementací řešení. V zadání bakalářské práce je určeno, jaké hlavní technologie má implementované řešení obsahovat. Je proto vhodné jednotlivé technologie v práci zmínit nejen prakticky ale také teoreticky.

4.1 Přístupy k tvorbě webové aplikace

Dříve se webové aplikace vyvíjely tradičním způsobem, kdy server posílal klientovi webové stránky při každém dotazu. Moderním trendem je psát aplikace tzv. jednostránkové. V následujících podkapitolách obě varianty více rozeberu.

4.1.1 Multi-page aplikace

Vícestránkové neboli multi page aplikace jsou webové stránky, které považujeme za tradiční webové stránky. Vykreslení výsledného html je vždy na serveru. Při změně k stránky dochází k přesměrování a prohlížeč znovu stahuje všechny zdroje. U tohoto typu webových aplikací může být načítání někdy obtížné, například pokud má uživatel slabé internetové připojení a potřebuje navštívit více webových stránek s médii. [1]

4.1.2 Single-page aplikace

Jedním z důvodů, proč jsou tradiční webové aplikace pomalé je, že se MVC server framework zaměřuje na poskytování statického obsahu stránku po stránce. Když se klikne na stránce na nějakou položku v navigaci, server pošle úplně celou stránku s hlavičkou, navigací i patičkou stránky. Přitom vzhledová změna byla vykonána jen na určitém obsahu webové stránky. [2]

Z těchto důvodů většina webových aplikací přechází do jednostránkových tzv. single-page aplikací, zkráceně SPA. SPA jsou aplikace, jejichž obsah je vytvořen jednou stránkou. To umožňuje rychlejší manipulaci, snadnou ovladatelnost, orientaci v aplikaci a hlavně téměř okamžitou reakci na pokyny uživatele. Jakmile uživatel načte aplikaci, data se stáhnou do lokálního úložiště tedy k uživateli. Poté dochází k dynamickému načítání jedné stránky. Toto je hlavní rozdíl mezi single page aplikacemi a tradičními webovými stránkami. [3]

4.1.3 Rozdíl mezi SPA a MPA

V této podkapitole nastíním hlavní rozdíly a výhody/nevýhody mezi oběma variantami.

4.1.3.1 Rychlost

Rychlost je důležitým faktorem. Čím delší odezvu webová aplikace má, tím jsou uživatelé méně trpěliví a web pro ně začíná být neatraktivní. V tomto ohledu mají navrch jednostránkové aplikace, protože načítají většinu komponent pouze jednou. Stránka se znovu nenačte úplně, kdykoliv uživatel požaduje novou část dat.

MPA jsou rychlejší pro prvotní zobrazení obsahu, avšak při změnách stránky je již pomalejší, protože prohlížeč musí znovu načíst celou webovou stránku, kdykoliv chce uživatel získat přístup k novým datům nebo přejít na jinou část webu. Dle cleveroad.com je optimální doba načítání webu 0,4 sekundy. [4]

4.1.3.2 Coupling

Coupling značí, jak na sobě moc závisí serverová a klientská část aplikace.

SPA jsou silně oddělené, což znamená, že front-end a back-end jsou oddělené. Jednostránkové aplikace používají ke čtení a zobrazování dat API, které je vyvinuto na straně serveru.

U tradičních webových aplikací jsou front-end a back-end na sobě více závislé. [4]

4.1.3.3 Optimalizace pro vyhledávače

V současnosti se webové stránky musí přizpůsobovat pro vyhledávače, jako je Google, či Seznam. Bohužel je toto stále slabina jednostránkových aplikací. Je to kvůli tomu, že single page aplikace jsou vyvíjeny v programovacím jazyce Javascript, který není u některých vyhledávačů podporován. Tento problém lze vyřešit pomocí serverside renderingu. Webové stránky jsou indexovány pomocí tzv. „crawling“ a „spireding“. Vyhledávače stahují soubory HTML, což usnadňuje hodnocení statických webových stránek HTML. Tradiční webové stránky umožňují lepší umístění u vyhledávačů, protože každá stránka může být optimalizována pro jiné klíčové slovo. Každá stránka má také své meta tagy, což může mít vliv ve výsledku vyhledávání. [4]

4.1.3.4 Proces vývoje

Při vývoji SPA musí být BEvrstva oddělena. API rozhraní je následně možné znovu použít, bez větších úprav, jak pro webovou aplikaci, tak například pro mobilní aplikaci. Při vývoji MPA je BE silně svázaná s FE vrstvou.

4.2 JS framework

Původní administrační rozhraní využívalo framework AngularJS, což je jeden z prvních frameworků pro tvorbu webových stránek. Nové rozhraní má být, dle požadavků, implementováno v frameworku ReactJS. Obě řešení více popíši v podkapitolách níže.

4.2.1 AngularJS

Angular byl vytvořen autory Miško Hevery a Adam Abrons v roce 2009. Je to open source javascriptový framework, který je určený pro vývoj klientské části webových aplikací.

Byl postaven na základě úvahy, že deklarativní programování je nejlepší volbou pro konstrukci uživatelského rozhraní, zatímco imperativní způsob programování je upřednostněn pro vývoj logické části aplikace. K dosažení této úvahy AngularJS zmocňuje tradiční HTML pomocí rozšíření jeho současné slovní zásoby a zjednodušení kódování. Výsledkem jsou komponenty, které se starají o části funkcionality a zjednodušují tak vývoj aplikací.

Tento framework se stal více publikovaný díky Miško Hevery, který v roce 2010 pracoval ve společnosti Google na projektu zvaném „Feedback“. Tento projekt dosáhl velikosti o více než 17 000 řádků kódu a vývojový tým nebyl spokojen s jejich produktivitou. Miško poté uzavřel se svým vedoucím sázku, že pokud použije svůj framework, dokáže projekt přepsat za dva týdny. Za pouhé 3 týdny dokázal pomocí tohoto framworku přepsat projekt na necelých 1 500 řádků kódu. [6]

AngularJS je udržovaný od společnosti Google a podporovaný robustní komunitou vývojářů po celém světě. Patří dnes mezi nejrozšířenější frameworky. Nabízí mnoho pozitiv i negativ. V dalším odstavci vypíši ty nejpodstatnější.

AngularJS umožňuje rychlejší a snažší datové úpravy, které nevyžadují zásah vývojáře. Označuje se jako „two-way data binding“ a zajišťuje, že změny provedené ve view vrstvě se okamžitě zobrazí v modelu a naopak. Podobně jako mnoho jiných populárních populárních Javascriptových frameworků Angular pohodlně zbavuje vývojáře aktivní manipulace s DOM (Document Object Model). [7]

4.2.2 ReactJS

ReactJS je stejně jako Angular knihovna JavaScriptu používaná pro vývoj webových aplikací k vytváření interaktivních prvků na webových stránkách. React byl vytvořen společností Facebook v roce 2013 autorem Jordan Walke. [8]

Stejně jako Angular vynucuje deklarativní přístup programování. Imperativní přístup k programování je možné popsat jako způsob popisu toho, jak věci fungují, na rozdíl od deklarativního programování, které funguje spíše jako popis toho, čeho je třeba dosáhnout. [9]

Tak jako AngularJS i Reactjs má mnoho výhod. Níže některé z nich zmíním a popíši.

ReactJS je open source knihovna, která umožňuje dělení dat projektů a aplikací pro MVC architekturu. React zavádí specifickou syntaxi kódu na bázi JavaScriptu a XML, která se nazývá JSX. [10]

Použití JSX není vynucené a jednotlivé elementy lze vytvářet pomoci *React.createElement.*

Pro začátečníky v psaní webových aplikací nabízí ReactJS výhodu v odstínění od DOMu. Program se píše pomocí komponent, ve kterých deklarativně vývojář nadefinuje strukturu (HTML) poskládáním JS funkcí. Popíše se tak výsledná stránka na základě příchozích dat (props) a ReactJS poté poskládá svůj vlastní virtuální DOM, který pak pomocí algoritmů porovnává se skutečným DOM. Pokud, na základě vstupních props, najde ve strukturách rozdíl, snaží se co nejefektivněji DOM aktualizovat. Programátor pak pouze dodává do jednotlivých komponent nová data a aplikace není zatěžována aktualizací skutečného DOM, jelikož to React řeší za vývojáře. [11]

Jak už bylo výše popsáno, program se skládá z komponent. Každá komponenta má svou vlastní logiku a ovládací prvky. Opakovaně použitelný kód pomáhá usnadnit vývoj a údržbu aplikací. Komponenty mohou být vnořeny do dalších komponent pro vytváření složitějších aplikací z těchto jednodušších komponent.

Svou popularitu si framework získal také díky přítomnosti praktické sady nástrojů. Pro vývojáře je tak vývoj srozumitelnější a jednodušší. Nástroje pro vývojáře React byly navrženy jako rozšíření pro prohlížeče Chrome a Firefox. Umožňují prohlížet hierarchii komponent ve virtuálním DOM, vybrat konkrétní komponenty a detailněji prozkoumat či editovat jejich aktuální rekvizity a stav.

Vyhledávací roboti a stroje ne vždy umí pracovat s Javascriptem, což může způsobit problémy při indexaci webu. Aplikace napsané v ReactJS mohou být spuštěny na serverové straně a vrací do prohlížeče výsledné html díky čemuž tento problém překoná. [12]

```
/*React declarative*/
class Button extends React.Component{
  this.state = { color: 'red' }
  handleChange = () => {
    const color = this.state.color === 'red' ? 'blue' : 'red';
    this.setState({ color });
  }
  render() {
    return (<div>
      <button
         className=`btn ${this.state.color}`
         onClick={this.handleChange}>
      </button>
    </div>);
  }
}

/* Imperative */
const container = document.getElementById('container');
const btn = document.createElement('button');
btn.className = 'btn red';
btn.onclick = function(event) {
 if (this.classList.contains('red')) {
   this.classList.remove('red');
   this.classList.add('blue');
 } else {
   this.classList.remove('blue');
   this.classList.add('red');
 }
};
container.appendChild(btn);
```

Ukázka kódu 1 Ukázka deklarativního přístupu Reactu a imperativního přístupu v nativním JS [29]

4.2.3 TSX/JSX

JSX je syntaxe podobná XML, která se kompiluje do spustitelného JavaScriptu. JSX získal popularitu, díky použití v Reactu a v dnešní době již nabízí i více implementací. Kompilace JSX má více módů. Pro tento projekt je použit mód *react.* JSX má rozšíření pro TypeScript, které následně zajišťuje typovou kontrolu. [27]

```
const jsxElement = (
  <h1 className='greeting'>
    Hello, world!
  </h1>
);


const reactElement = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

Ukázka kódu 2 Ukázka vytvoření h1 elementu v JSX a pomoci React.createElement[10]

4.2.4 Next.js

Next.js, framework postavený na JavaScriptové knihovně React, je dnes jednou z nejrozšířenějších knihoven pro Server side rendering. Server side rendering (zkráceně SSR) je překreslování webové stránky používající JavaScriptový kód na straně webového serveru. Ke klientovi se také pošlou javascriptové šablony, díky kterým se framework „navěsí“ na html a je s ním schopen dále pracovat. Webová aplikace, napsána například v jazyce PHP, a webový server na základě požadavků kód zpracují a pošlou uživateli již sestavenou webovou stránku. O většinu práce se tak stará serverová část, což je často nadbytečné, jelikož bez pomoci dodatečného softwaru je nucen server po každém požadavku sestavit webovou stránku znovu.

U NextJS je rendering vymyšlený tak, že uživatel obdrží nejprve částečně vykreslenou stránku. Tato stránka se uživateli zobrazí téměř okamžitě. Během toho, co se uživatel na stránce rozhlíží, pošle Next dodatečný JavaScript, který prohlížeč zpracuje a zprovozní potřebné interaktivní prvky.

Tento framework je velice populární a jeho vývoj jde prudce dopředu. Díky funkci Fast Refresh se jakákoliv úprava, kterou v kódu vývojář vykonává, projeví v prohlížeči prakticky okamžitě. Framework se stará také o vhodné načítání dat. [13]

4.3 Výběr technologie pro implementaci

Pro implementaci nového administračního rozhraní je dle zadání a požadavků společnosti vybrán ReactJS. Protože práce je zaměřená na vývoj webové aplikace, bylo rozhodnuto o návrhu aplikace jako jednostránkové aplikace. Díky JavaScriptu, do kterého je práce kompilována, je možné práci napsat jako progresivní webovou aplikaci. V implementaci je využit také framework Next.js, a to z důvodu zjednodušení routování URL adres a využití lazy loading pro jednotlivé webové stránky. Dále také kvůli úspoře datového toku a rychlosti aplikace.

5 Návrh

Jelikož nové administrační rozhraní je z hlediska funkcionalit stejné jako staré administrační rozhraní napsané v AngularJS, návrh webové aplikace zůstává stejný. Z tohoto důvodu nebylo zapotřebí vytvářet podrobný návrh stránek v podobě Wireframů či jiných ilustrací webových stránek. Mimo jiné nebyla nutnost vytvářet ani Mapu stránek, jelikož je totožná se starým systémem. Jediné, co se zde změnilo, byl vzhled aplikace, který byl flexibilní vůči vývojáři. Bylo nutné vizuální podobu připodobnit starému rozhraní, totožné být nemuselo.

6 Implementace

Na základě staré verze administračního rozhraní a vybraných technologií a požadavků, popsaných v předchozích kapitolách, implementuji nové administrační rozhraní, které více popíši v této kapitole.

6.1 Vývojové prostředí

Před zahájením etapy implementace je zapotřebí nastavit vývojové prostředí, ve kterém implementace proběhne. Pro efektivnější vývoj byly použity následující tři nástroje.

6.1.1 ESLint

ESLint je open source nástroj pro vývoj v JavaScriptu, který původně vytvořil Nicholas C. Zakas v roce 2013. Nástroj pomocí statické analýzy vyhledává v programovém kódu problematické vzory nebo části kódu, které nedodržují určité pokyny pro styl psaní kódu.[14]

Nástroj je v bakalářské práci využit pro lepší udržitelnost a čitelnost kódu.

```
extends: [
        'plugin:react/recommended',
        'plugin:react-hooks/recommended',
        'plugin:jsx-a11y/recommended',
        'eslint:recommended',
        'plugin:@typescript-eslint/eslint-recommended',
        'plugin:@typescript-eslint/recommended',
        'plugin:import/typescript',
],
```

Ukázka kódu 3 Rozšíření ESLint

6.1.2 Prettier

Prettier je podobně jako ESLint nástroj pro úpravu stylu kódu. Na rozdíl od ESLintu se tento nástroj zaměřuje pouze na vzhledovou úpravu programovacího jazyka a není zaměřen pouze na JavaScript, nýbrž i na struktury jako JSON, CSS, JSX a další. [15]

6.1.3 Yarn

Yarn je nástroj, který se stará o správu modulů a jejich závislosti. Nahrazuje stávající node package manager. [16]

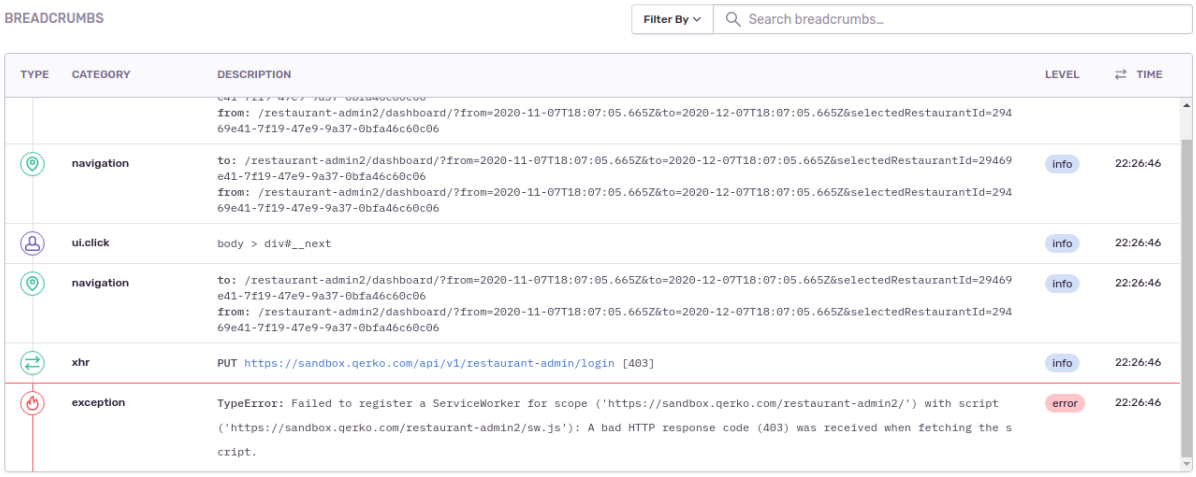
6.2 Nástroje

Pro usnadnění práce jsou použity při vývoji dodatečné nástroje, které pomohou se zpracováním chyb, se sdružením souborů a zajištění hodnotnější kompatibility s prohlížeči.

6.2.1 Sentry

Sentry je open source systém pro sledování chyb, který podporuje širokou škálu aplikací na server, prohlížeč i mobilní zařízení. [17]

V této práci je použit pro logování chyb.



Obrázek 4 Ukázka logu v Sentry

6.2.2 WebPack

Hlavním účelem WebPacku je sdružování jednotlivých souborů do jednoho task runneru. Nástroj slouží k rozdělení zdrojových kódů na více částí tzv. code splitting. WebPack zpracuje celou aplikaci a pomocí interního grafu závislostí vygeneruje jeden či více výsledných souborů. Pomáhá také aplikaci zajistit kompatibilitu se starými verzemi prohlížečů.

6.3 Programovací jazyk a syntaxe

Jak z názvu práce vyplývá, pro vývoj je zvolen framework ReactJS. ReactJS jsem podrobněji popsal v předešlých kapitolách, v této části se zaměřím na programovací jazyk.

6.3.1 JavaScript

JavaScript je multiplatformní objektově orientovaný programovací jazyk. Jedná se o nejrozšířenější programovací jazyk pro vývoj webových aplikací. Jazyk je dynamicky typovaný, což značně snižuje udržitelnost velkých projektů.

6.3.2 TypeScript

TypeScript je brán jako typová nadmnožina jazyka JavaScriptu, která se do běžného JavaScriptu kompiluje. [19]

Statický typový systém pomáhá snáze zachytit problémy. Systém statického typu znamená, že prvky jsou, až na pár situací, přesné pro typ objektu, na který odkazujeme. První vydání TypeScript bylo v roce 2012. Má nemalou podporu vývojových editorů a prakticky každý měsíc je nová aktualizace. [18] Tento přístup je v dnešní době velmi rozšířený a doporučený a díky jeho přednostem se stává neoficiálně standardem vývoje webových aplikací psaných v Reactu.

6.4 Strategie a bezpečnost

V předchozích kapitolách jsem uvedl v jakém jazyce a pomocí jakých knihoven a doplňkových nástrojů bude aplikace vyvíjena. Dále se zaměřím na strategii psaní aplikace a rozdělení adresářů v projektu pro lehkou srozumitelnost a pochopení položek uvnitř práce. Nedílnou součástí je i zabezpečení aplikace.

6.4.1 Progresivní aplikace

Progresivní aplikace nejsou vytvořeny pomocí jedinečné a specifické technologie. Není to nový framework nebo nový programovací jazyk. PWA jsou soubory strategií, různé techniky a rozhraní API, které vývojářům umožňují dát uživatelům nativní mobilní podobu aplikace, na kterou jsou zvyklí. [20]

PWA jsou rychlé a často vykreslí obsah na zařízení uživatele za méně než pár sekund. Jsou spolehlivé, a to i bez pevného datového připojení nebo na starém zařízení, a poutavé, protože povolením oznámení na aplikaci či webu uživatele lze upozornit na vše, co se v aplikaci děje.

Skutečná PWA musí mít část funkcionalit, které fungují na každém zařízení. I kdyby funkcionalita byla pouze vykreslení statické webové stránky, musí fungovat i na velmi starém mobilním zařízení s operačním systémem Android. Nemůže se stát, aby uživatel viděl pouze černou obrazovku nebo výpis chybových zpráv.

Hlavním bojem webových aplikací obecně je už dlouho správná podpora prohlížečů. Prohlížečů je mnoho a každý nemusí vždy podporovat nejnovější a nejlepší technologie. Bohužel toto je problém i pro PWA. A však jedním z nejdůležitějších principů PWA je, že by měl svým uživatelům poskytovat postupně lepší zážitek, podle toho, jak se zvyšují možnosti jejich prohlížečů. Jen proto, že prohlížeč v danou chvíli nepodporuje funkci, která je pro uživatele zajímavá, neznamená, že je vše ztraceno a ani, že prohlížeč funkci nebude podporovat v budoucnu. Samozřejmě pokud uživatel prohlíží aplikaci na prohlížeči, který podporuje většinu nebo dokonce všechny funkcionality PWA, bude mít z webové aplikace lepší zážitek.

Většina aplikací se zaměřuje na pět nejrozšířenějších webových prohlížečů, kterými jsou Chrome, Safari, Firefox, Opera a Edge. Protože společnost Google se v poslední době velmi zaměřuje na PWA, tak není překvapením, že prohlížeč Chrome má nejrobustnější podporu pro každou funkci PWA. Na obrázku číslo 5 je zobrazeno pořadí podpory prohlížečů v čele s Chromem. [20]

Obrázek 5 Podpora PWA u prohlížečů [33]

6.4.2 Adresová struktura

Práce je rozdělena do jednotlivých podadresářů se soubory, které slouží ke specifickému účelu. Takto členěná struktura projektu pomáhá k přehlednosti a zjednodušuje orientaci v projektu. Práce je členěna do následujících adresářů.

- **Pages** složka obsahuje soubor _document.tsx. Tento soubor obsahuje hlavní definici html dokumentu. Dále je v této složce také soubor _app.tsx, který je ústředním souborem celé aplikace, ve kterém je definováno přesměrování nepřihlášeného uživatele. Je zde definován aplikační kontext celé aplikace, tzv. se zde nastavuje jazyková mutace, vybraná restaurace, autorizace a uživatel. Mimo těchto důležitých souborů jsou ve složce obsaženy veškeré stránky aplikace, včetně jejich lokalizace a dle struktury je v této složce je dynamicky vytvořeno routování napříč celou aplikací.

- **Layouts** adresář slouží jako složka pro kostru celé aplikace. V adresáři je soubor obsahující definice levého menu včetně všech položek v něm se skrývajících, nastavení oprávnění a cesty k jednotlivým položkám.

- **Components** obsahují veškeré komplexnější komponenty, které jsou využity na více pozicích. V této složce se tak nacházejí komponenty, které nesou určitou funkční část. Příklad těchto komponent může být DatePicker či mapa pro vyznačení pozice dané restaurace.

- **Elements** složka sdružuje veškeré méně komplexní komponenty aplikace. Komponenty zde mají jednoduché funkce. Příklad prvků tohoto adresáře může být button nebo select s label pro vykreslení dodatečného obsahu a další prvky pro formuláře.

- **Src** sdružuje hlavní logiku na pozadí webové aplikace. V tomto adresáři se nachází jádro lokalizace webu. Dále je zde deklarace API klienta pro získávání dat ze serveru z REST API. V adresáři se deklaruje také aplikační stav, tzv. aplikační kontext, a jsou zde definovány jednotlivé routy z důvodu TypeScript kontroly při přesměrování napříč rozhraním. Složka navíc obsahuje definice Sentry, grafického tématu a logování.

- **Public** obsahuje veškeré veřejné soubory, které nejsou nijak kompilovány. Převážná většina těchto souborů jsou obrázky jako favicon, či logo, ale také například soubor robots.txt.

- **.gitlab** obsahuje konfigurační soubory pro gitlab k nasazení aplikace na vývojové a produkční prostředí.

- **node_modules** pojímá veškeré moduly stažené package managerem YARN, které slouží ke kompilaci do výsledného balíku určenému k nasazení na serverovou stranu. Tyto moduly je doporučené držet aktualizovány kvůli chybám obsahující staré verze modulů. Je zde však riziko vzniku nových chyb.

6.4.3 Zabezpečení

Ze zadaných požadavků na práci vyplývá, že aplikace může být užívána pouze registrovanými uživateli. Z tohoto důvodu je obsah aplikace dostupný pouze po přihlášení uživatele a není tak veřejný.
Pro přihlášení je nutné zadání uživatelského e-mailu a hesla. Poté uživatel získá od serverové strany autorizační token, který je součástí hlavičky každého požadavku na server.
Autorizační token je uložen do local storage a je sdílen napříč více otevřenými okny. Tokenu je při každém požadavku na server prodloužena jeho platnost a zároveň server ověří úroveň uživatelského přístupu. Pokud server detekuje token po platnosti, vrací http status 401, tzv. neautorizovaný přístup. Poté je uživatel přesměrován zpět na přihlašovací formulář a je nucen se opět přihlásit uživatelským e-mailem a heslem pro získání nového tokenu.

```
setUser({
        id: auth.user.id,
        email: auth.user.email ?? undefined,
        username: auth.user.username,
});
localStorage.setItem('data', JSON.stringify(auth));
localStorage.setItem('accessToken', auth.accessToken);
```

Ukázka kódu 4 Uložení uživatele do local storage

6.5 Využité knihovny

Při vývoji aplikace bylo zapotřebí využít knihovny pro správnou validaci dat a manipulaci s nimi. Z projektu bylo vybráno těchto 6 knihoven, o kterých je vhodné se zmínit.

6.5.1 Zod

Zod se v aplikaci využívá k validaci vstupu ve formulářích, otypování a validaci získaných dat ze serveru a validaci odesílaných dat na server.

6.5.2 Formik

V React jsou formuláře velmi podrobné a vykonávají mnoho operací, které nesou značné náklady na výkon. Formik je malá knihovna, která vývojáři pomůže se získáním hodnot do formuláře a z formuláře, ověřením, chybovými zprávami a zpracováním odeslání formuláře. [21]
V práci je knihovna využita zejména pro zpracování formulářů jako je věrnostní program a jeho detail. Dále pro úpravu či vytvoření restaurace a jejího profilu. Slouží také pro výpis chybových hlášek, které získá z validace pomocí knihovny Zod.

6.5.3 Bignumber

Je použit pro správné numerické operace s čísly, nejčastěji s částkami/cenami.
Projekt je zaměřen na zobrazování a vykonávání operací s čísly. Tato čísla často představují peněžní hodnotu zisku restaurace, obsluhy a další. Pro správné numerické operace těchto čísel je v projektu využita knihovna Bignumber. Knihovna je nejčastěji využívána pro operace s částkami peněz a cenovými nabídkami.

6.5.4 React-Query

Tato knihovna se stará o načítání a ukládání v mezipaměti a aktualizaci asynchronního stavu v Reactu. Považuje se za jednoduché malé API, které lze použít ihned bez dodatečné konfigurace. Je definována jako tzv. agnostic protokol, což znamená, že je možné použít například REST nebo GraphQL. [22]
V práci je React-Query využitý pro ukládání požadavků na server pod klíčem. Toto je odůvodněno tím, že není žádoucí, aby klientská strana vytvářela stejné požadavky pořád dokola. Není potřeba poté tolik zatěžovat serverovou stranu aplikace.

6.5.5 Axios

Při vývoji projektu je potřeba spolupracovat s rozhraním REST API. Axios je odlehčený klient http podobný nativnímu API pro načítání JavaScriptu. Je založen na tzv. promise-base. To dává možnost využít výhody asynchronního JavaScriptu a čekat na čitelnější asynchronní kód. Je možné také zrušit žádosti a na klientské straně je integrovaná ochrana proti zfalšování požadavků mezi weby. [23]
V práci je využit pro komunikaci mezi serverovou a klientskou částí tedy pro požadavky na server a odpovědi.

6.5.6 Date-fns

Date-fns poskytuje nejkomplexnější, přesto jednoduchou a konzistentní sadu nástrojů pro manipulaci s daty JavaScriptu v prohlížeči a Node.js. [24]
V projektu slouží pro výpis správného formátu datumu a času. Pro různé kultury a jazyky se může tento formát lišit například prohozením měsíce a dne nebo jinak zapsaným časovým formátem.

6.5.7 Material-UI

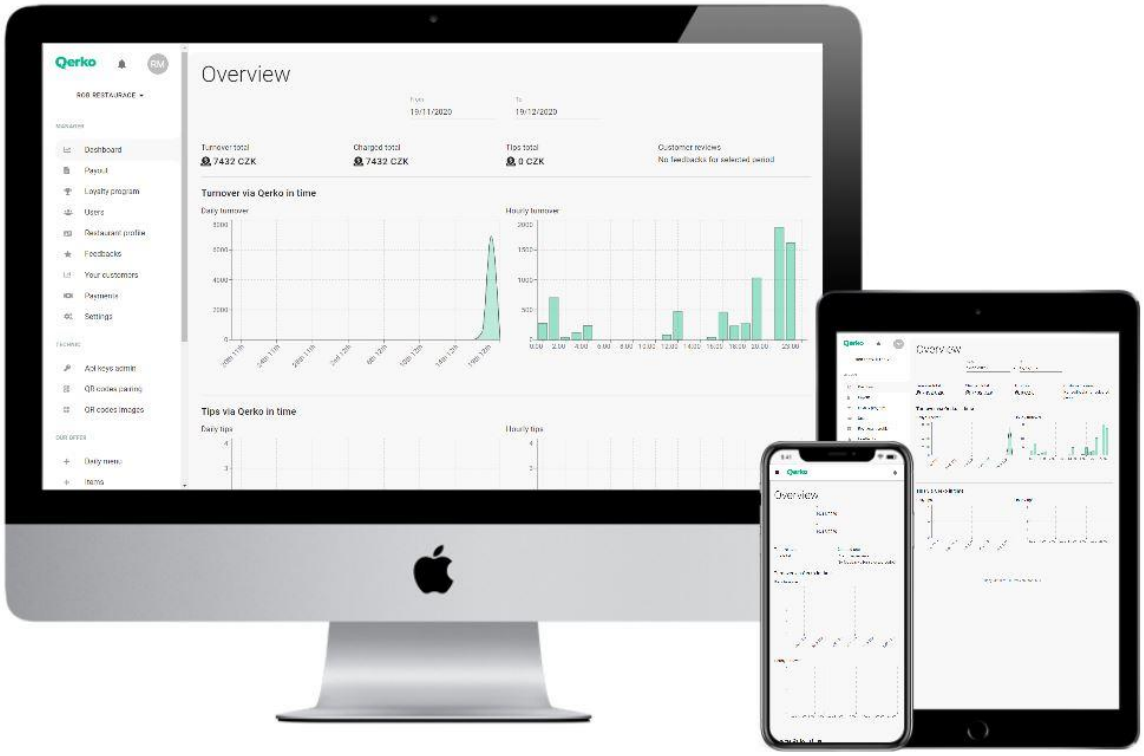
Material-UI je open-source projekt, který obsahuje komponenty Reactu. Ty implementují Material Design od společnosti Google. První verze knihovny začala v roce 2014, krátce po publikaci Reactu. Dnes je tato knihovna jednou z nejlepších knihoven uživatelského rozhraní pro React. [25]
Z hlediska vizuální stránky je rozhraní postaveno na knihovně Material-UI. Jsou zde využity komponenty jako date-picker, select a další.

6.6 Další funkcionality aplikace

Poslední podkapitolou implementační části práce jsou dodatečné funkce webového rozhraní. Některé souvisí i s technologiemi, které byly popsány výše.

6.6.1 Responzivní zobrazení

Z požadavků je známo, že aplikace bude naimplementována jako responzivní aplikace určená pro mobilní i počítačové zobrazení. Responzivita aplikace je implementována pomocí knihovny Material-UI.



Obrázek 6 Responzivní zobrazení aplikace [30]

6.6.2 Podpora více jazyků

Aplikace má podporu více jazyků. Doposud jsou nahrány slovníky v českém a anglickém jazyce. Jednotlivé slovníky jsou pomocí TypeScript validovány, aby obsahovaly stejné prvky. Tuto funkci zajišťuje uvedený kód níže.

```
export const createDictionary = <
       A extends Record<string, unknown>
       >(en: A,cs: Record<keyof A,unknown> & A):
(() => A) => () => useMessages(({ en, cs }));
```

Ukázka kódu 5 Vytvoření slovníku

7 Nasazení

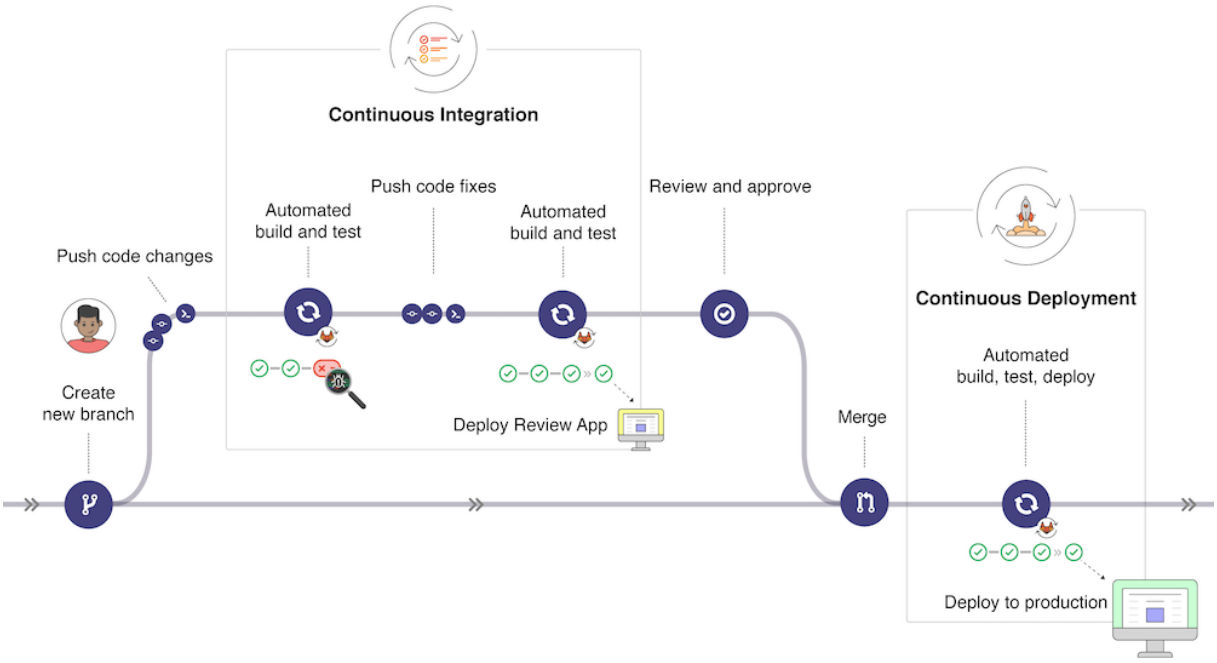
Nasazení se ovládá skrze Gitlab, kde celý vývoj probíhá. Je použit koncept *Continuous Integration*, *Continuous Delivery*, a *Continuous Deployment.* V tomto projektu jsou použity pouze první dva uvedené.[28]

7.1 Continuous Integration

Při nahrání nové části kódu a následném vytvoření merge requestu se automaticky spouští scripty pro ESLint kontrolu a typovou kontrolu. Typová kontrola může odhalit závažné chyby v kódu a jedná se o důležitý bod v této fázi. Dále by v této fázi mohly být unit testy.

7.2 Continuous Delivery

Pokud CI proběhlo úspěšně, lze změny nahrát na server. Tato akce musí být manuálně spuštěna.



Obrázek 7 Gitlab CI/CD [28]

8 Závěr

Cílem práce bylo vytvořit administrační rozhraní s použitím nejnovějších technologií, které nahradí stávající rozhraní v AngularJS. Nové rozhraní obsahuje všechny funkce staré administrace a doplňuje je o responzivní chování a podporu PWA pro lepší práci s aplikací.

Grafický design využívá framework Material-UI, který obsahuje komponenty použitelné v Reactu, využívající Material Design od společnosti Google. Framework zároveň obsahuje systém pro tvorbu CSS. Toto řešení umožňuje snadno měnit vzhled stránky na základě uživatelské interakce. Zároveň použitím layout systému z Material-UI je mnohem jednodušší vytvořit web responzivní na více zařízeních. Material-UI zároveň obsahuje definice interface v Typescriptu, což ve spojení s chytrým editorem usnadňuje implementaci. Celý projekt, kromě komponenty vykreslující mapu, je napsaný v Typescriptu. To vytváří jistotu, že funkce se volají se správnými formáty dat, a eliminuje to chyby z důvodu nullability, undefined apod. Data ze serverového API se validují přes knihovnu Zod. Zod je také využit pro validaci dat formulářů, což ve spojení s knihovnou Formik vytváří skvělé prostředí pro tvorbu formulářů v ReactJS. Na pozadí celého webu je využit framework Next.js. Ten se stará o routování napříč celým webem a rozdělení souborů do více balíčků. To zajišťuje rychlost při prvotním zobrazení stránky, protože se nestahuje jeden velký soubor, ale stahují se jen soubory, které jsou nutné pro zobrazení a funkčnost stránky, na které se právě nacházíte. PWA je použito pro cachování souborů a pro možnost nainstalovat aplikaci na zařízení. Budoucí výhledy jsou použít také notifikace.

Nové rozhraní řeší některé nedostatky bývalého rozhraní jako např. nutnost se znovu přihlásit v každém dalším otevřeném okně webového prohlížeče. Díky použití Typescriptu je projekt přehlednější a snáze udržovatelný než původní řešení.

Administrace pro restaurace od 1. prosince běží v testovacím provozu souvisle se stávajícím rozhraním. Nové i staré rozhraní jsou aktuálně spuštěna vedle sebe. Nové rozhraní obsahuje všechny funkcionality starého rozhraní a nyní je ve fázi testování a opravovaní chyb.

9 Literatura

[1] VARAKSINA, Svitlana, 30 September 2020n. l. *Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps* [online]. [cit. 2020-11-10]. Dostupné z: https://themindstudios.com/blog/spa-vs-mpa/#whatsanmpa

[2] MIKOWSKI, Michael S. a Josh C. POWELL, ©2014. *Single-Page Web Applications: Javascript end-to-end*. United States of America: Manning Publications Co. ISBN 9781617290756.

[3] ULIČNÝ, Vít, 2020. *Rascasone: CO JE JEDNOSTRÁNKOVÁ APLIKACE (SPA) A KDY JI VYUŽÍT?* [online]. 15.11.2020 [cit. 2020-12-19]. Dostupné z: https://www.rascasone.com/cs/blog/jednostrankova-webova-aplikace-spa

[4] Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches, 2019. *Asper Brothers* [online]. 12 Nov 2019 [cit. 2020-12-19]. Dostupné z: https://asperbrothers.com/blog/spa-vs-mpa/

[5] Single page vs multi-page websites: Design battle!, 2018. *JustinMind* [online]. July 10, 2018 [cit. 2020-12-19]. Dostupné z: https://www.justinmind.com/blog/single-page-vs-multi-page-websites-design-battle/

[6] BRANAS, Rodrigo, 2014. *AngularJS Essentials: Design and construct reusable, maintainble, and modular web applications with AngularJS*. Birmingham: Packt Publishing. ISBN 978-1-78398-008-6.

[7] RAJPUT, Mehul, 2016. The pros and cons of choosing AngularJS. *Jaxenter* [online]. March 18, 2016 [cit. 2020-12-10]. Dostupné z: https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html

[8] MORRIS, Scott. WHAT IS REACT JS? *Skillcrush* [online]. [cit. 2020-12-10]. Dostupné z: https://skillcrush.com/blog/what-is-react-js/#used

[9] BERTOLI, Michele, 2017. *React Design Pattens and Best Practices: Build modular applications that are easy to scale using the most powerful components and design patterns that React can offer you right now*. Birmingham: Packt Publishing. ISBN 978-1-78646-453-8.

[10] Introducing JSX. *React* [online]. [cit. 2020-12-19]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[11] React-Úvod. *DžejEs* [online].[cit.2020-12-19].Dostupné z: https://www.dzejes.cz/react-uvod.html

[12] Pros and Cons of ReactJS. *JavaTpoint* [online]. [cit. 2020-12-19]. Dostupné z: https://www.javatpoint.com/pros-and-cons-of-react

[13] ZLATKOVSKÝ, Michal, 2020. Next.js frontend: Jak tvoříme weby příští generace. *FG Forrest* [online]. 1. 7. 2020 [cit. 2020-12-19]. Dostupné z: https://www.fg.cz/cs/deje-se/next.js-frontend-jak-tvorime-weby-pristi-generace-12837

[14] About - ESLint - Pluggable JavaScript linter. *ESLint* [online]. [cit. 2020-12-19]. Dostupné z: https://eslint.org/docs/about/

[15] SEBHASTIAN, Nathan. *What is Prettier?* [online]. [cit. 2020-12-19]. Dostupné z: https://www.educative.io/edpresso/what-is-prettier

[16] NAKAZAWA, Christoph, Sebastian MCKENZIE a Jamie KYLE, 2016. Yarn: A new package manager for JavaScript. *FACEBOOK Engineering* [online]. OCT 11, 2016 [cit. 2020-12-19]. Dostupné z: https://engineering.fb.com/2016/10/11/web/yarn-a-new-package-manager-for-javascript

[17] BUCKLER, Craig, 2018. Getting Started with Sentry.io Error Tracking. *Sitepoint* [online]. November 22, 2018 [cit. 2020-12-19]. Dostupné z: https://www.sitepoint.com/getting-started-with-sentry-io-error-tracking/

[18] RIPPON, Carl, 2018. Why TypeScript with React? *Building SPAs* [online]. December 12 2018 [cit. 2020-12-19]. Dostupné z: https://www.carlrippon.com/why-typescript-with-react/

[19] PREVITE, Joe, 2020. React with TypeScript: Best Practices. *Sitepoint* [online]. September 14 2020 [cit. 2020-12-06]. Dostupné z: https://www.sitepoint.com/react-with-typescript-best-practices/

[20] SHEPPARD, Dennis, ©2017. *Beginning Progressive Web App Development: Creating a Native App Experience on the Web*. Tinley Park, Illinois, USA: Apress Media. ISBN 978-1-4842-3090-9.

[21] Getting Started | Formik. *Formik docs* [online]. [cit. 2020-12-12]. Dostupné z: https://formik.org/docs/overview

[22] AFONSO, Daniel. Caching made easy with React Query. *Medium: JavaScript in Plain English* [online]. [cit. 2020-12-19]. Dostupné z: https://medium.com/javascript-in-plain-english/react-query-fbefc608ed95

[23] HALLIDAY, Paul, 2020. How To Use Axios with React. *DigitalOcean: Community* [online]. August 26, 2020 [cit. 2020-12-19]. Dostupné z: https://www.digitalocean.com/community/tutorials/react-axios-react

[24] Date-fns - modern JavaScript date utility library. *Date-fns* [online]. [cit. 2020-12-19]. Dostupné z: https://date-fns.org/docs/Getting-Started#introduction

[25] Meet Material-UI — your new favorite user interface library, 2018. *FreeCodeCamp* [online]. APRIL 28, 2018 [cit. 2020-11-20]. Dostupné z: https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c/

[26] SO 9001 Processes, Procedures and Work Instructions. In: 9000 Store [online] [cit. 15.05.2020]. Dostupné z: https://the9000store.com/iso-9001-2015-requirements/iso-9001-2015-context-of-the-organization/processes-procedures-work-instructions/

[27] TypeScript: Documentation – JSX [online] [cit. 21.12.2020]. Dostupné z: https://www.typescriptlang.org/docs/handbook/jsx.html

[28] Introduction to CI/CD with GitLab [online] [cit. 21.12.2020]. Dostupné z: https://docs.gitlab.com/ee/ci/introduction/

[29] Declarative vs Imperative Programming | by Ian Mundy | codeburst [online] [cit. 31.12.2020]. Dostupné z: https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2

[30] Multi Device Website Mockup Generator [online] [cit. 31.12.2020] Dostupné z: http://techsini.com/multi-mockup/index.php

[31] Qerko [online] [cit. 31.12.2020] Dostupné z: https://qerko.com/

[32] Qerko s.r.o. Praha IČO 06678815 - Obchodní rejstřík firem | Kurzy.cz [online] [cit. 31.12.2020] Dostupné z: https://rejstrik-firem.kurzy.cz/06678815/qerko-sro/

[33] Browsers [online] [cit. 31.12.2020] Dostupné z: https://everydownload.net/category/browsers/

10 Seznam použitých zkratek

API – Application programming interface

BE – Backend

CD – Continuous Delivery

CI – Continuous Integration

CSS – Cascading Style Sheets

DOM – Document Object Model

FE – Frontend

HTML – Hypertext Markup Language

JS – JavaScript

JSON – JavaScript Object Notation

JSX – JavaScript XML

MPA – Multi-page Application

PWA – Progressive Web Apps

REST – Representational state transfer

SPA – Single-page Application

TSX – Typescript JSX

11 Seznam obrázků a ukázek programu

12 Příloha A

Obsah přiloženého CD

Na přiloženém CD je zdrojový kód aplikace a tento dokument.