VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÁ APLIKACE PRO VÝDEJNÍ OKÉNKA A ROZVOZY V DOBĚ KORONAVIROVÉ PANDEMIE

WEB APPLICATION FOR TAKEAWAYS IN TIMES OF COVID-19 QUARANTINE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DAVID ŽÁK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN HRUBÝ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Žák David**
Program: Informační technologie
Název: **Webová aplikace pro výdejní okénka a rozvozy v době koronavirové pandemie**
Web Application for Takeways in Times of Covid19 Quarantine
Kategorie: Elektronický obchod

Zadání:

1. Prostudujte programování webových aplikací ve frameworku Vue.js, PHP a MySQL. Prostudujte workflow rezervačních systémů v prostředí podniků veřejného stravování.
2. Navrhněte systém pro správu produktů a objednávek prodejnám postižených koronavirem formou webové aplikace. Navrhněte platformu, na které budou zákazníci objednávat z nabídky prodejny. Systém navrhněte tak, aby byl snadno ovladatelnou webovou aplikací.
3. Implementujte navržený systém. Zaměřte se na jeho snadnou instalovatelnost a správu uživatelem.
4. Systém testujte v simulovaném provozu. Podle možností ukažte nasazení systému v reálných situacích.

Literatura:

- C. Macrae: Vue.js: Up and Running: Building Accessible and Performant Web Apps, O'Reilly Media, 1. vydání, 2018.

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hrubý Martin, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 22. dubna 2020

Abstrakt

Cílem této práce je vytvořit webovou aplikaci pro správu nabídky a objednávek prodejnám ovlivněným koronavirovou pandemií. Aplikace je implementována pomocí Vue.js, PHP a MySQL. Práce obsahuje teoretickou a praktickou část. V teoretické části jsou popsány aktuální možnosti vývoje webových aplikací a použitých technologií v této práci. Praktická část obsahuje návrh, implementaci a testování aplikace. Výsledná aplikace umožňuje zákazníkům vytvořit objednávku produktů z nabídky prodejny. Jednotlivým prodejnám prostřednictvím administrace poskytuje možnost správy nabídky, objednávek a nastavení.

Abstract

The aim of this thesis is to create a web application enabling management of offers and orders for businesses affected by the Covid-19 pandemic. The application is implemented using Vue.js, PHP and MySQL. The thesis consists of a theoretical and practical part. The theoretical part focuses on current possibilities of web application development and technologies used in this thesis. The practical part describes the designing process, implementation and testing of the application. The resulting application allows customers to browse through offers and to place an order. Additionally, for individual business it provides an option of managing their offer, orders and settings through the administration.

Klíčová slova

Vue.js, webová aplikace, Javascript, MySQL, PHP, koronavir, objednávky s sebou

Keywords

Vue.js, web application, Javascript, MySQL, PHP, Covid-19, takeaways

Citace

ŽÁK, David. *Webová aplikace pro výdejní okénka a rozvozy v době koronavirové pandemie*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Hrubý, Ph.D.

Webová aplikace pro výdejní okénka a rozvozy v době koronavirové pandemie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
David Žák
28. května 2020

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce panu Ing. Martinovi Hrubému, Ph. D. za vedení při realizaci této práce. Děkuji za jeho věcné připomínky a nápady, které mi byly při implementaci předány. Dále bych chtěl poděkovat Davidovi Dvořáčkovi za vytvoření designu aplikace a agentuře Creepy Studio za podporu.

Obsah

Kapitola 1

Úvod

Koronavirová pandemie změnila ze dne na den život většině lidí. V České republice omezil provoz všem kávárnám, restauracím a mnoha dalším prodejnám. Povolené je pouze vyzvednutí v tzv. okénku, které směřuje směrem ven do ulice nebo rozvoz produktů k zákazníkovi domů. Na omezení provozu nebyly prodejny z minulosti připraveny, a tak začaly improvizovat a vyřizovat své objednávky e-mailem, zprávami na Instagramu nebo pomocí formuláře na Google Forms. Na trhu existují již funkční služby jako Dámejídlo.cz nebo Wolt, které si však berou z objednávek i desítky procent[1]. Mnoho prodejen se tak snažilo najít vlastní cestu, jak prodávat online.

Tato bakalářská práce se zabývá vývojem webové aplikace pro výdejní okénka a rozvozy. Cílem je pomoci zdarma prodejnám, které si otevřely výdejní okénko nebo rozváží své produkty svým zákazníkům do karantény a zjednodušit jim správu objednávek a nabídky. Aplikace má název Sebou.cz, jedná se o zjednodušení spisovně psaného „s sebou“ a je dostupná na doméně sebou.cz. Každé prodejně aplikace vytvoří vlastní subdoménu (např. kafo.sebou.cz). Svou adresu následně prodejny sdílejí na svých sociálních sítích nebo jinou formou propagace, aby se o nové formě prodeje dozvěděli jejich zákazníci.

Ve druhé (2) kapitole je srovnání aktuálních technologií pro tvorbu webových aplikací a aktuální konkurenční platformy na trhu. Třetí kapitola (3) popisuje technologie použité v aplikaci této práce. Ve čtvrté kapitole (4) je popsán samotný návrh řešení. Jelikož si situace žádala co nejrychlejší řešení, použil jsem k vývoji aplikace technologie, se kterými mám mnohaleté zkušenosti. Frontend aplikace je napsaný v javascriptovém frameworku Vue.js nadstavený frameworkem Nuxt.js, které jsou popsané v kapitolách 3.1.1, respektive 3.1.2. Ukládání dat je vyřešeno pomocí relační databáze MySQL popsaná v kapitole 4.4. Backend API je implementován v jazyce PHP. Pro dotazy do MySQL databáze je využita PHP knihovna Medoo (kapitola 3.2.1). Dále se zde rozebírá návrh uživatelského rozhraní a získávání zpětné vazby od prodejen. V páté kapitole (5) je popsána samotná implementace aplikace. Ta obsahuje konkrétní realizaci frontendu a backendu a propojení mezi nimi. Šestá kapitola (6) práce popisuje správu aplikace z pohledu prodejny a její užívání z pohledu uživatele. Sedmá kapitola (7) se zabývá testováním a reálným nasazením do provozu. Bakalářská práce je zakončena závěrem (8) shrnující výslednou práci, další možná vylepšení a statistiky z reálného provozu.

Kapitola 2

Současný stav webových technologií

Kapitola shrnuje aktuální možnosti pro tvorbu webových aplikací. Popisuje jednotlivé jazyky pro jejich implementaci a porovnává tři nejpoužívanější databáze ve webových technologiích. Závěr kapitoly obsahuje analýzu konkurenčních aplikací pro výdej a rozvozy produktů prodejen.

2.1 Vývoj webových aplikací

Webové technologie se neustále posouvají. To je pro vývojáře a samotné uživatele velká výhoda, jelikož mohou využívat nové funkce a možnosti. Vývoj těchto technologií však není jednotný, jelikož samotné vykreslování webu probíhá na více na sobě nezávislých webových prohlížečích od různých společností. Mezi nejpoužívanější prohlížeče dnes patří Chrome[1] vyvíjený společností Google, Safari[2] společností Apple, Explorer[3] a Edge společností Microsoft a Firefox[4] společností Mozilla[11]. Pro zmíněné prohlížeče a mnoho dalších existuje databáze caniuse.com, která detailně mapuje podporu jejich funkcí. V této databázi je tedy možné najít informace, v jakých prohlížečích a jeho verzích je dostupná konkrétní funkce, kterou potřebujeme použít.

Je důležité poznat cílovou skupinu uživatelů aplikace a získat data o prohlížečích, které její návštěvníci používají. Na základě získaných dat je potřeba aplikaci optimalizovat pro co největší možný počet uživatelů v naší cílové skupině. Pro získání těchto dat můžeme využít mnoho nástrojů jakými jsou například Google Analytics[5] nebo Hotjar[6], který umí zpětně zobrazit web přesně tak, jak jej viděl daný uživatel ve svém prohlížeči.

Cílem této práce je vytvořit dynamickou webovou aplikaci. Rozdíl dynamické webové aplikace od statické webové stránky se liší hlavně jejími funkcemi a samotnou implementací. Dynamická webová aplikace využívá ke generování obsahu programovací jazyk, který je zpravidla napojen k databázi. Z databáze daný jazyk získává data, ze kterých pak generuje HTML a CSS. Prohlížeč pak na základě vygenerovaného obsahu vykreslí uživateli výslednou stránku. Tyto stránky jsou tedy dynamicky generované programovacím jazykem. Naproti

[1] https://www.google.com/intl/cs_CZ/chrome/
[2] https://www.apple.com/safari/
[3] https://www.microsoft.com/cs-cz/edge
[4] https://www.mozilla.org/cs/firefox/
[5] https://analytics.google.com
[6] https://www.hotjar.com/

tomu statická webová stránka je napsaná přímo v HTML a CSS a web se mění jen při úpravě samotného kódu vývojářem.

2.2 Jazyky pro vývoj webových aplikací

K vývoji webových aplikací se dnes používá mnoho jazyků, ty nejčastější popisuje tato kapitola. Základními stavebními kameny webu jsou HTML a CSS. HTML slouží ke strukturované reprezentaci dat, které se pomocí CSS mohou stylovat a vytvořit přívětivé uživatelské rozhraní. Pomocí programovacích jazyků, CSS preprocesorů a databází je možné HTML a CSS generovat a vytvořit tak požadovanou dynamickou webovou aplikaci.

2.2.1 HTML

HTML [2] je jednoduchý textový značkovací jazyk, který se lze snadno naučit a podporují jej téměř všechna zařízení obsahující webový prohlížeč. Vývojem webových standardů se zabývá společenství W3C[7], vedené Timem Berners-Leem, jenž vynalezl web a jazyk HTML. Jazyk je v době tvorby této práce ve verzi 5.

Běžný uživatel obsah HTML v prohlížeči nevidí. Prohlížeč jej převede do grafické podoby, která lze upravit pomocí CSS. Jedná se tedy o předpis, kterému rozumí webový prohlížeč. Dále je velmi zásadní správné značkování obsahu kvůli SEO[8], aby byl web co nejlépe dohledatelný vyhledávačemi, jako jsou například Google nebo český Seznam.cz.

2.2.2 CSS

CSS [6] vzniklo někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Zkratka znamená Cascading Style Sheets, česky „*“kaskádové styly. Kaskádové, jelikož se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední.

CSS preprocesory

V dnešní době se pro kaskádové styly hojně využívají CSS preprocesory, které jsou v pozici nadstavby CSS. Preprocesory umožňují definovat proměnné, používat funkce, matematické výpočty a mnoho dalšího, co samotné CSS v neumí. Nejoblíbenějšími CSS preprocesory jsou Sass, Stylus a Less[9]. Ze stylů, psaných v těchto preprocesorech, následně překladač vytvoří výsledný CSS soubor, se kterým umí pracovat prohlížeč. Less a Sass mají velmi podobnou syntaxi. Hlavní výhodou Sassu je možnost uložení selektoru do proměnné, které nám může usnadnit práci v navrácení zpět v jeho potomku. V praktické části této bakalářské práci se používá preprocesor Less.

2.2.3 PHP

PHP je skriptovací jazyk, který je nejpoužívanějším jazykem pro tvorbu dynamických webových služeb. Aktuálně ho využívá 77 % všech webů r[13]. Jeho přední vlastností je jednoduchost a skvělá dokumentace. Pro zajímavost je v PHP napsané nejpoužívanější CMS na světě – Wordpress. Na tomto redakčním systému běží 33,5 % z celkového počtu webů na internetu a zabírá 60,4 % trhu na poli redakčních systémů [5]. Je tedy nesporné, že je tento

[7] W3C – World Wide Web Consortium
[8] SEO – Search Engine Optimization

jazyk populární i právě díky Wordpressu[9]. V této bakalářské práci je v jazyce PHP napsané API aplikace, popsané v kapitole 5.

2.2.4 Javascript

Javascript se dá dnes využít v backendu i na frontendu. To z toho důvodu, že ho jako jediný skriptovací jazyk podporuje prohlížeč. Tím se může velice ulehčit serveru, kdy se některé výpočty nechají vykonat až na straně klienta.

Zmíněného přístupu využívají i tzv. SPA[10], v překladu jednostránkové, aplikace. Jak už název napovídá, jedná se o aplikaci, která je tvořena jedinou stránkou. Aplikace si po úvodním získání stránky dále načítá jen data, která potřebuje k dalšímu užití aplikace například na základě interakcí uživatele. Uživateli díky tomu nemusí server například stále generovat a posílat hlavičku a patičku webu, která je na všech stránkách stejná, ale načte mu jen obsah stránky. Mezi nejpopulárnějšími frameworky pro tvorbu takových aplikací jsou React od společnosti Facebook, Angular od společnosti Google a Vue.js od vývojáře Evana You. V této práci se využívá poslední zmíněný framework Vue.js, popsaný v kapitole 3.1.1.

2.2.5 Python

Python a další jazyky jsou uvedeny jako zajímavost, nejsou v bakalářské práci použity, ale určitě je stojí za to zmínit. S jazykem Python se uživatel ve webovém prostředí potká jen zřídka. Jeho použití je méně časté, jelikož je v něm napsáno pouhých 4 % webů [13]. Bývá použitý s frameworkem [11], který se chlubí spuštěním funkční webové aplikace v řádech jednotek hodin.

2.2.6 C#

Webové aplikace v jazyce C# se vytváří v nástroji Visual Studio, který vyvíjí společnost Microsoft. S vývojem webu je potřeba zmínit framework .NET, který je potřebný pro vývoj webu. Jazyk svými vlastnostmi bývá využíván u velkých projektů, které staví na robustní databázi. Jelikož jazyk není skriptovací, musí se projekt nejdříve sestavit před jeho přesunem do produkčního prostředí.

2.2.7 Java

Ve webových technologií se můžeme setkat i s řešením, které je napsané v jazyce Java. Je to především díky vysokému počtu vývojářů a frameworků, které jsou na trhu.

2.3 Databáze

Databáze slouží ke strukturovanému uložení dat. Ve webových technologiích se používají dva druhy databáze – relační a objektové. Relační databáze funguje na principu tabulek, které obsahují sloupce definovaného datového typu. Jednotlivé řádky tvoří tzv. záznamy, které lze vkládat, mazat, aktualizovat či získávat pomocí dotazů. Mezi jednotlivými tabulkami lze

[9] https://wordpress.org/
[10] SPA – Single page application
[11] https://www.djangoproject.com/

tvořit relace a je možné pracovat s těmito relacemi přímo v dotazech. Objektová databáze má jiný přístup k ukládání dat. Jednotlivé záznamy se ukládají ve formě objektů.

2.3.1 MySQL

MySQL[12] běží jako proces na webové službě, jakou může být například Apache nebo Nginx a podporuje ji několik různých klientů včetně příkazového řádku. Jeden MySQL server může spravovat více databází pro více aplikací a každá z nich může ukládat různá data uspořádána různými způsoby [14].

MySQL je ve webých technologiích standardem, běží na něm 97 % [12] všech webů, které používají databázi. Jedná se o relační databázi a je použita v API pro ukládání dat v této bakalářské práci. Návrh databáze se nachází v kapitole 4.4.

2.3.2 Firebase

Firebase[13] je služba koupená v roce 2014 společností Google. Obsahuje soubor produktů, jehož součástí je i databáze jménem Firebase. Používá objektový přístup a je v základní variantě zdarma. Její cena roste počtem připojených zařízení, počtu přenesených dat a velikostí uložiště. Databáze je uložená v cloudu[14] na serverech Googlu. Právě díky Googlu, který vyvíjí i Javascriptový framework Angular se propojení velice nabízí. Tato databáze je však volně přístupná i třetím stranám.

2.3.3 MongoDB

MongoDB[15] je moderní a populární objektová databáze. Je to open source software a byl původně vyvinutý společností 10gen. Přednostmi MongoDB jsou možnost ukládání souborů, indexace jakéhokoliv pole v dokumentu nebo vyvažování zátěže pomocí shardingu [8].

[12] https://www.mysql.com/
[13] https://firebase.google.com/
[14] Cloud je poskytování služeb či programů uložených na serverech poskytovatele. [3]
[15] https://www.mongodb.com/

2.4 Analýza konkurenčních řešení

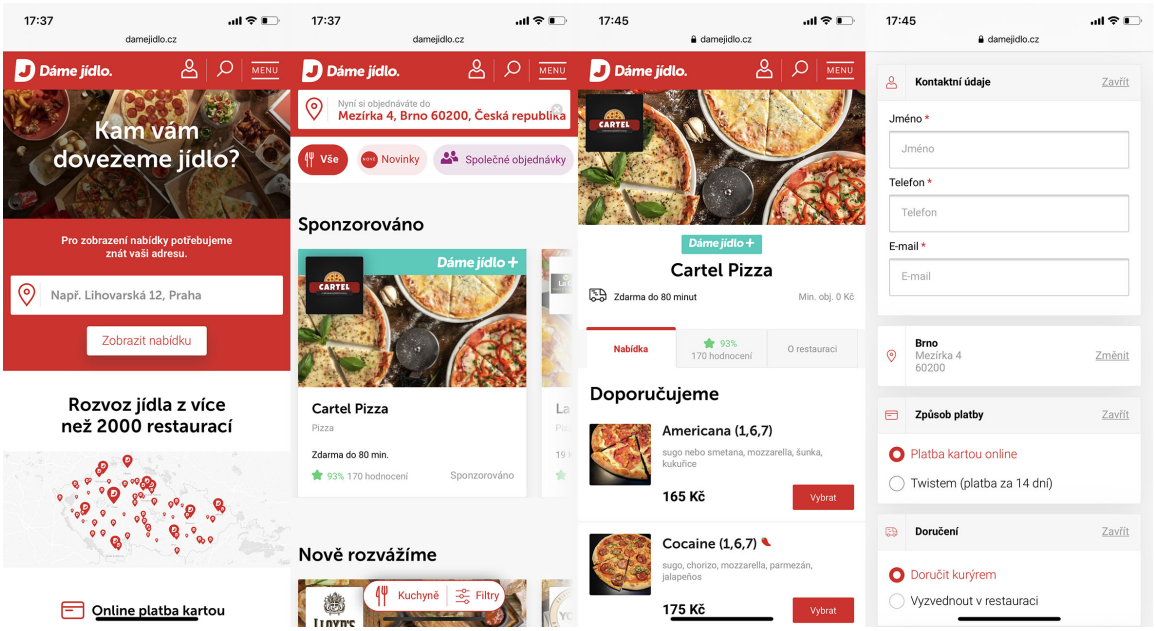
Tato kapitola analyzuje konkurenční řešení, která se zaměřují na stejnou či podobnou funkci této práce. Jedná se tedy o služby zaměřující se převážně na gastro průmysl. Služby poskytují umístění nábídky prodejny online, ze které následně zákazníci mohou vytvářet objednávky. Objednávky jsou pak připraveny k vyzvednutí v dané prodejně nebo je daná prodejna či online služba k zákazníkovi přiveze. Cílem práce není konkurovat těmto službám, ale poskytnout pomoct zdarma prodejnám postiženým koronavirovou krizí.

2.4.1 Dámejídlo.cz

Česká služba Dámejídlo.cz[16] se zaměřuje pouze na gastronomické podniky a je dostupná z webu a aplikací pro iOS a Android zařízení. V aplikaci se nachází mapa prodejen, mezi kterými lze filtrovat. Lze si vybrat, zda si zákazník objednávku vyzvedne, nebo ji chce přivézt. Může si navolit maximální cenu za dopravu a kdy nejpozději potřebuje jídlo doručit.

Prodejny musí se službou Dámejídlo.cz nejdříve podepsat smlouvu, ve které je mimo jiné definované, kolik služba získá z každé objednávky procent. To se liší podle toho, zda si objednávku prodejna vyřídí sama, nebo využije rozvozové služby Dámejídlo.cz.

Oproti Dámejídlo.cz má aplikace Sebou.cz především 2 výhody. První z nich je, že pro vytvoření profilu prodejny v aplikaci Sebou prodejna nepotřebuje podepisovat žádnou smlouvu, ale stačí si pouze založit účet. Tím ušetří čas, který v době krize hraje velice důležitou roli. Druhou výhodou jsou zmíněná procenta z objednávky. Aplikace Sebou.cz je zdarma. Celou částku objednávky získá prodejna.

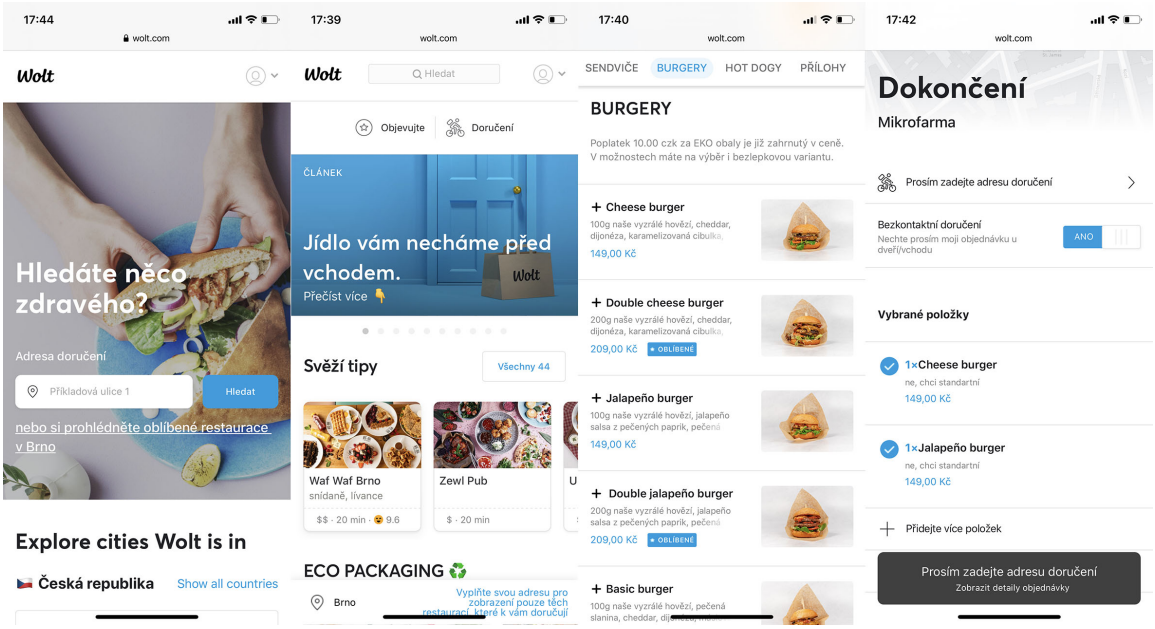


Obrázek 2.1: **Aplikace Dámejídlo.cz** – zleva: vyhledávání lokality, výpis restaurací v dané lokalitě, nabídka prodejny, objednávkový formulář

[16] https://damejidlo.cz

2.4.2 Wolt

Wolt[17] je Finská služba, která se principem podobá Dámejidlo.cz. V Česku omezuje svůj provoz pouze na Prahu a Brno. Poskytuje vyzvednutí v prodejně i rozvoz samotné prodejny, ale i vlastními vozy. Wolt si bere taktéž procenta z objednávek a je nutné se společností nejdříve podepsat smlouvu. Výhody aplikace Sebou.cz jsou tedy stejné jako u aplikace Dámejídlo.cz. Díky své působnosti jen ve 2 největších městech může aplikace pomoct pouze zlomku všech prodejen v České republice.

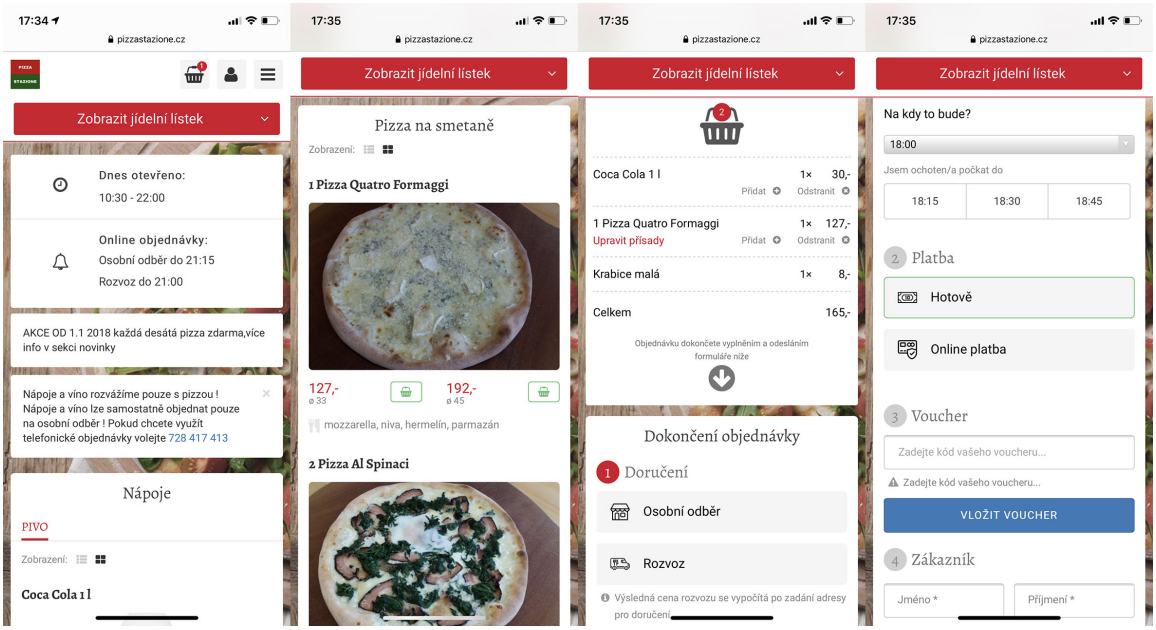


Obrázek 2.2: **Aplikace Wolt** – zleva: vyhledávání lokality, výpis restaurací v dané lokalitě, nabídka prodejny, objednávkový formulář

[17] https://wolt.cz

2.4.3 Adaptee Gastro

Služba Adaptee Gastro[18] se svou funkcí nejvíce podobá aplikaci Sebou.cz. Zaměřují se na systém pro správu objednávek online. Na rozdíl od Dámejídlo.cz a aplikace Wolt nemají vlastní rozvoz. Aplikace poskytuje prodejnám řešení s vlastním hardwarem v podobě tabletu a tiskárny účtenek. Aplikace láká prodejny na ušetření času, který jim aktuálně zabírá vyřizování objednávek telefonicky či e-mailem. Při přijetí nové objednávky prodejna musí nejdříve objednávku schválit. Po schválení se zákazníkovi pošle notifikace, že je objednávka přijata. Před samotným doručením zákazníkovi přijde SMS zpráva, že se kurýr blíží a že se má připravit k převzetí objednávky. Gastro Adaptee poskytuje také aplikaci Adaptee Driver, která sleduje polohu kurýra a zobrazuje ji zákazníkovi na mapě, aby přesně věděl, kde se jeho objednávka nachází. Služba je zatím ve zkušebním provozu zdarma a po uplynutí zkušebního provozu bude fungovat na měsíčním fixním paušálu.



Obrázek 2.3: **Gastro Adaptee** – zleva: úvodní obrazovka, nabídka prodejny, košík, objednávkový formulář

[18] https://gastro.adaptee.cz/

Kapitola 3

Použité technologie v aplikaci

Kapitola ujasňuje použité techologie v aplikaci a propojení mezi nimi. Definuje pojmy frontend a backend. Dále následující text popisuje frameworky a knihovny v projektu včetně jejich funkce.

3.1 Frontend

Za frontend se považuje vrstva aplikace, kterou vidí uživatel. Součástí této vrstvy je HTML, CSS a Javascript. HTML nám definuje obsah stránky, CSS má na starost vzhled a pomocí Javascriptu můžeme z webu udělat interaktivní aplikaci. V Javascriptu existuje mnoho knihoven poskytující řešení různých problémů, se kterými se při tvorbě webů můžeme setkat. Například AJAX[1], drag & drop[2], smooth scroll[3], validaci formulářů před samotným odeslaním a spoustu další věcí. V této kapitole si rozebereme javascriptové knihovny a jejich hlavní výhody, kvůli kterým jsou v této práci použity.

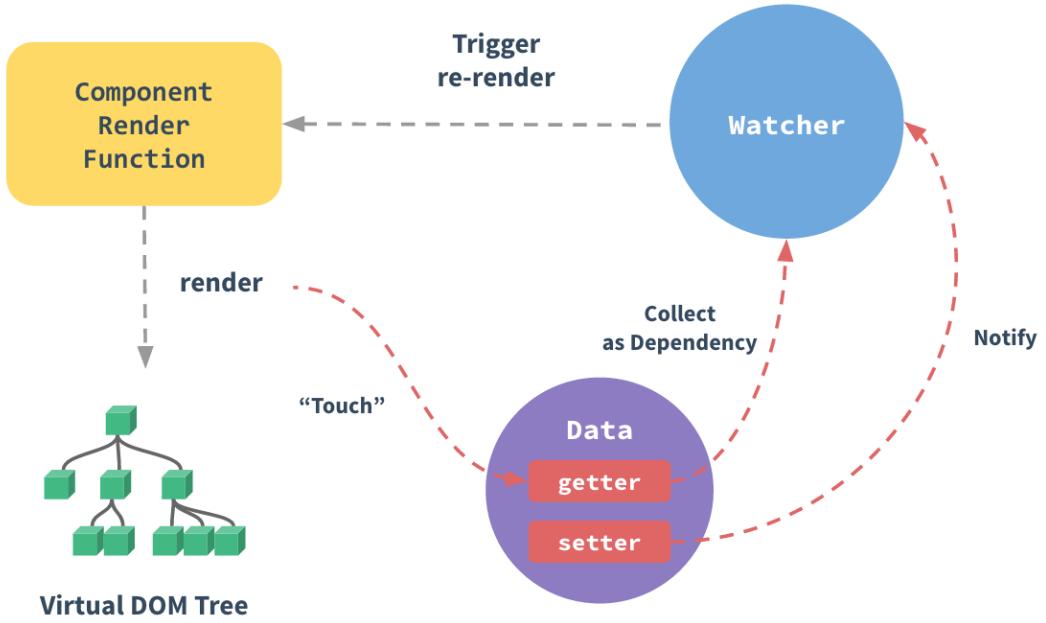
3.1.1 Vue.js

Vue.js je javascriptová knihovna pro tvorbu webových uživatelských rozhraní. Vyvinul jej bývalý zaměstnanec Googlu či Meteoru Evan You. Jeho motivací bylo právě díky práci v Googlu vzít to nejlepší z Angularu a vytvořit knihovnu, kterou by sám používal. Hlavními vlastnostmi Vue.js je bezesporu jeho velikost, důsledkem je vysoká rychlost samotných webů postavených na tomto frameworku. Evan You v dokumentu o vzniku Vue.js „Vue.js: The Documentary“ popisuje, že inspiraci pro co nejmenší velikost zdrojového kódu získal v Číně, kde se narodil. V této zemi podle jeho slov mají průměrně velmi pomalé připojení k mobilnímu internetu, a tak je velikost webu důležitým aspektem. Tento přístup se hodí i do této práce, díky kterému bude výsledná aplikace rychlá.

Další jeho předností je jednoduchost a přehledná dokumentace. Dokumentace je napsaná díky národnosti autora i v Čínštině. Tím je knihovna populární také v této zemi. Vue.js se dá vložit jako Javascript do HTML již hotového projektu. To se hodí, pokud se Vue.js použije jen pro malou část webu, například pro validaci formuláře. Druhým způsobem je použít vue-cli, nástroj pro vývoj aplikace, která běží celá ve Vue.js. Tento vývoj je přehlednější a dává smysl pro SPA aplikace. Vue-cli se vyvíjí pomoci souborů s příponou .vue, ty pak tento nástroj umí sestavit v hotovou webovou aplikaci.

[1] AJAX – Asynchronní Javascript a XML
[2] Drag & drop – umožňuje uživateli chytnou, přesunout a pustit prvek uživatelského rozhraní
[3] Smooth scroll – Plynulé posouvání stránky webu

Vue.js se řadí do skupiny takzvaných progresivních frameworků. Slovo progresivní znamená, že je implementován jako dodatečné rozšíření do jazyka HTML. Díky tomuto rozšíření můžeme přímo v HTML pracovat s datovým modelem, který je definován ve Vue.js. Tento datový model je reaktivní, to znamená, že když se datový model upraví, zobrazení se aktualizuje. Všechny definované data mají svůj getter a setter, které samotný vývojář nevidí. Je součástí Vue.js a kontroluje všechny závislosti dané hodnoty. Pokud se hodnota změní, současně s ní se aktulizují všechny její závislosti a následně i samotné zobrazení.



Obrázek 3.1: Cyklus reaktivity dat [10]

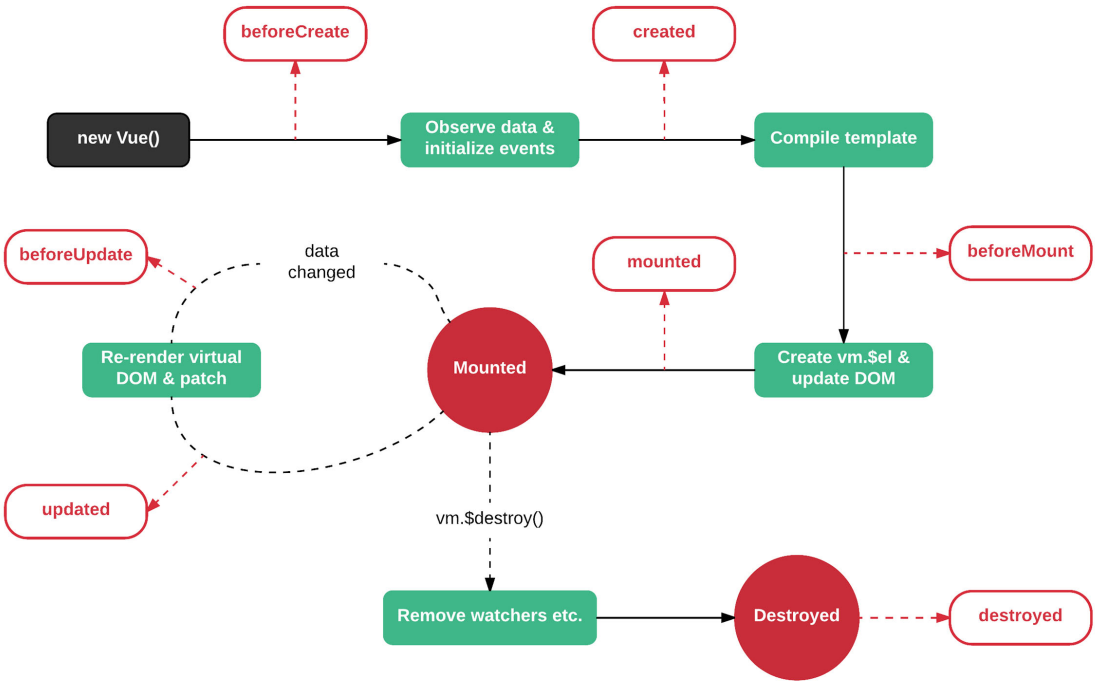
Komponenty

Komponenta je soběstačná část kódu. Má vlastní HTML, CSS a Javascript, které neovlivňují ostatní komponenty. Komponenta může obsahovat jiné komponenty a ty mezi sebou mohou komunikovat [7]. Komponentou může být například tlačítko, ale také celá část webu, která se na webu nebo na stránce opakuje. Hlavní výhodou rozdělení kódu do komponent je přehlednost a znovupoužitelnost. Všechen kód a jeho data dané části webu (komponenty) jsou na jednom místě – v jednom souboru.

Životní cyklus komponenty

Komponenty se vytváří a zanikají na základě toho, na které stránce webu a v jakém stavu se aktuálně aplikace nachází. Každá komponena má svůj životní cyklus. To je množina událostí, se kterými vývojář aplikace může pracovat. Nejdřívější událostí je beforeCreate při inicializaci komponenty, ta je však zatím prázdná a obsahuje jen její kostru. Další událostí je created, v ní už komponenta obsahuje vývojářem definovaná data, která jsou již reaktivní. Následuje beforeMount, kdy je v komponěntě sestavené HTML, ale ještě jej nevložilo vidi-

telně uživateli do aplikace. Vložením do aplikace se následně zavolá událost mounted. Tím je komponenta zcela sestavená a nyní zde existují eventy beforeUpdate a updated, které odchycují změny v dané komponentě. Před zánikem se volá event beforeDestroy, zde je stále dostupná kompletní komponenta. Poslední událostí po zániku komponenty je destroyed, kdy je již zcela uvolněna z paměti.



Obrázek 3.2: Diagram životního cyklu komponenty [4]

Definice dat v komponentě

Každá komponenta může mít definovaná data. Tahle data jsou zapsána pomocí funkce `data()`, která vrací objekt s jejich názvy a hodnotami. S hodnotami lze pak pracovat v dané komponentě i ostatních komponentách, pokud na hodnotu do jiné komponenty odkážeme. Dále je možné tato data používat progresivně přímo v HTML.

Modifikace proměnných (Computed)

Computed je objekt metod, které vrací upravená data, se kterými můžeme dále pracovat stejným principem jako u funkce `data()`. V některých případech je potřeba z definovaných dat získat jejich modifikovanou formu. Například, když je zadaná délka strany čtverce a je potřebova na více místech pracovat s jeho obsahem, můžeme si vytvořit `Computed` metodu, která umocní délku strany a vrátí výsledek. Díky reaktivitě Vue.js se pak při každé změně délky strany aktualizují i všechny `Computed` metody, které se změněnou metodou pracují.

Sledování změn proměnných (Watch)

`Watch` je objekt metod, kde jejím názvem je název proměnné, u které chceme sledovat změny. Metoda obsahuje argumenty s novou a předchozí hodnotu proměnné.

Definování vlastních funkcí (Methods)

`Methods` je objekt a slouží k definici standardních Javascriptových funkcí. Ty můžeme následně volat kdekoliv v dané komponentě.

Globální definice (Mixins)

Pomocí `Mixins` můžeme ve Vue.js definovat náležitosti komponenty globálně. Ty jsou pak přístupné z každé komponenty, stránky a šablony aplikace.

3.1.2 Nuxt.js

Nuxt.js je knihovna, která je nadstavbou Vue.js. Při vytvoření Nuxt aplikace se vytvoří struktura projektu, do které se vkládají soubory aplikace. Nuxt pak pomocí souborů a nastavením definovaném v konfiguraci `nuxt.config.js` sestaví hotovou aplikaci. Nuxt pomocí své struktury projektu definuje kostru aplikace ve Vue.js. Příkladem může být složka pages a v ní vytvořený soubor `prihlaseni.vue`. Nuxt díky umístění souboru v dané složce vytvoří v aplikaci novou stránku kontaktu přístupnou na adrese `sebou.cz/prihlaseni`. Na stejném principu funguje tvorba komponent, šablon stránek, Vuex stores[4], Middlewares[5]. Nuxt ve Vue.js sestaví konfiguraci, která vyřeší vložení a propojení vytvořených souborů v této struktuře do funkční aplikace.

3.1.3 Správce balíčků NPM

NPM slouží ke správě javascriptových balíčků (knihoven). Pomocí tohoto nástroje se do projektu jednoduše instalují knihovny, které rozšíří svými funkcemi aplikaci. Všechny údaje o balíčcích správce ukládá do souboru package.json v kmenové složce projektu. Soubory knihoven tak nemusí být na Gitu[6], jelikož se na základě dat v souboru `package.json` nainstalují v potřebné verzi na serveru či u dalšího vývojáře.

3.1.4 Použité NPM balíčky

V této části jsou uvedeny použité knihovny v projektu s popisem své funkce. Svou funkcí jednotlivé knihovky rozšířili aplikaci této práce.

Axios

Knihovna Axios ulehčuje práci s asynchronní komunikací pomocí protokolu HTTP. V tomto projektu byla využita především pro komunikaci s API aplikace.

[4] https://nuxtjs.org/guide/vuex-store/
[5] https://nuxtjs.org/api/pages-middleware/
[6] Git – systém pro správu verzí

Vue draggable

Tato knihovna umožňuje na frontendu přesouvat prvky v seznamu mezi sebou. Tato funkce je použita ve správě nabídky k přesouvání a určení pořadí přidaných položek.

Moment.js

Knihovna pro práci s časem v Javascriptu. Implementuje časové formáty, pásmy a lokalizací dat pro české prostředí.

Vue trend chart

Knihovna Vue trend chart je využita v administraci prodejny pro zobrazení statistiky tržeb pomocí grafu, který knihovna vykresluje ve formátu SVG.

Markdown to HTML

V projektu se využívá pro správu obsahu landing page a blogu redakční systém Strapi, který je blíže popsáno v kapitole 3.2.2. Strapi ukládá obsah ve formátu Markdown. Ten je potřeba převést do HTML a k tomu slouží NPM knihovna Markdown to HTML.

3.2 Backend

Backend je v této kapitole rozdělen celkem do tří částí. První z nich je návrh databáze. Ta běží na již zmíněném MySQL. Jedná se o relační databázi, která obsahuje jednotlivé tabulky. Celá relace databáze je v této kapitole znázorněna pomocí ER diagramu (obrázek č. 4.2). Druhou částí je API rozhraní. API je připojeno k databázi a komunikuje pomocí požadavků s frontendem aplikace. Třetí částí je služba Strapi poskytující hotové řešení headless CMS[7] pro vytvoření redakčního systému.

3.2.1 API

API má v tomto projektu funkci rozhraní, které komunikuje s databází a poskytuje metody a data pro fungování celé aplikace. Rozhraní je napsáno v jazyce PHP a využívá knihovny Medoo pro práci s databází, knihovnu PHP Mailer pro připojení k SMTP pro odesílání e-mailů a knihovny Image resize, která komprimuje a ořezává fotky produktů do potřebné velikosti. API obsahuje koncové body, na které se aplikace dotazuje pomocí HTTP dorazů. Každý koncový bod má jinou funkci, například get_order vrátí potřebná data o objednávce. API má dále za úkol ukládat soubory, respektive fotky produktů a loga prodejen. Implementuje zabezpečení do administrace a validuje hodnotu vkládaných dat.

3.2.2 Strapi

Součástí projektu jsou také blog a často kladené dotazy. Aby jednotlivé články a dotazy nebyly napsány přímo v kódu, musí se propojit s databází. Pomocí uživatelské rozhraní musí být možné vkládat nové články či editovat již existující. Tuhle funkci plní headless redakční systém Strapi. Jedná se o open-source aplikaci napsanou v Javascriptu pro tvorbu a správu obsahu. K obsahu je pak možno pomocí jeho vlastního API přistupovat.

[7] Headless CMS – redakční systém pro správu obsahu, kde je implementován pouze backend systému

Kapitola 4

Návrh řešení

Tato kapitola popisuje kompletní návrh řešení. Rozebírá princip aplikace, její případy užití a definuje cílovou skupinu aplikace. Dále je zde popsán návrh databáze včetně ER diagramu (Obrázek 4.2). V závěru je popsán způsob získávání zpětné vazby od prodejen.

4.1 Výběr názvu

Vymyslet správný název aplikace nebyl lehký úkol. Název musí být jednoduchý a zároveň musí dávat smysl se zaměřením aplikace. Jako jednou z variant bylo právě „s sebou“. Doména ssebou.cz však vypadá na první pohled složitě, i když je pravopisně správně. Varianta s jedním „s“ tak připadala v úvahu, ale byla nejistá právě z nespisovného pohledu. O názvu, zda „sebou.cz“, nebo „ssebou.cz“ se tedy rozhodlo v hlasování v anketě na Instagramu. Anketní otázka zněla „Co je lepší? kafo.ssebou.cz nebo kafo.sebou.cz?“, záměrně už obsahovala i název prodejny, aby dával smysl v kontextu celé adresy. V anketě hlasovalo celkem 139 lidí a vyhrála varianta „sebou“ se ziskem 76 % všech hlasů.

Aplikace má tedy název Sebou.cz, je to zjednodušení pravopisně psaného „s sebou“. Myšlenka je taková, že každá prodejna bude mít svůj název v subdoméně a tím bude jednoduše zapamatovatelná a dohledatelná zákazníky (např. kafo.sebou.cz pro zlínskou kavárnu Kafö). Důležitým aspektem při výběru názvu bylo, že je doména sebou.cz volná. A to i ve variantě ssebou.cz, kdyby zákazník napsal doménu pravopisně správně, aby ho aplikace přesměrovala na variantu s jedním „s“ v názvu.

Obrázek 4.1: Výsledky hlasování na Instagramu

4.2 Cílová skupina

Aplikace má celkově dvě cílové skupiny – majitelé prodejen a její zákazníky. Aplikace cílí na prodejny, které musely zavřít svůj běžný provoz kvůli koronavirové krizi. Chtějí však dále poskytovat své produkty online, aby si jejich zákazníci mohli produkty vyzvednout nebo jim je doručit. Zároveň nechtějí platit poplatky konkurenčních služeb. Je pro ně zásadní správa prodejny v aplikaci, přehlednost nabídky a objednávek. Aplikace musí být velice jednoduchá, mít dobré UI a zároveň musí umět zaujmout.

Pro zákazníka je důležitá rychlost aplikace, přehlednost nabídky prodejny a musí působit důvěryhodně, jelikož zde vyplňují své osobní údaje. Používá počítač či mobilní zařízení s připojením k internetu. Na zařízení zvládá základní úkony a má ideálně zkušenosti s nákupem online. Neměl by být omezován například pomalým připojením k internetu, aby ho neodradil pomalý průchod objednávkou.

4.3 Případy užití

Při příchodu na web uvidí majitel prodejny landing page, na které jsou základní informace o tom, jak aplikace funguje. Na této landing page je CTA tlačítko pro založení nové prodejny.

Pro založení prodejny stačí zadat název subdomény ve správném formátu, vyplnit e-mail a heslo. Následně se mu vytvoří účet a stránka ho přesměruje do administrace prodejny. V administraci pak nejdříve vyplní detailnější informace o svém podniku, jako je název prodejny (při registraci uváděl pouze svůj název bez diakrity, malými písmeny s možností přidat pomlčku, jako spojovník slov), adresu a logo značky. Prodejna si dále nastaví, zda má výdejní okénko nebo rozvozy. Podle typu předání si prodejna může napsat své texty, které se budou zobrazovat například na děkovné stránce a dalších částech webu. To se hodí například pro vložení instrukcí ohledně platby nebo možných časech předání.

Platby si řeší prodejna sama. Při vyzvednutí na místě můžete platit všemi způsoby, které prodejna nabízí. Doporučuje se však platba kartou, jelikož při ní dochází k nejmen-

šímu možnému kontaktu, na který je při pandemii potřeba klást důraz. Při doručení k zákazníkovi aplikace znovu prodejnu neomezuje, ale přidává možnost přidat svůj bankovní účet, aby zákazník mohl poslat peníze převodem na účet prodejny. Aplikace tedy nemá roli zprostředkovatele platby a peníze jdou přímo od zákazníka prodejně, ve které vytvořil objednávku.

Prodejna v administraci spravuje svou nabídku. Ta se skládá z kategorií, která obsahuje položky. Položkám lze zadat název, popisek, cenu, množství na skladě, alergeny a fotku produktu. Zákazník pak na webu vidí seznam těchto položek s uvedenými informacemi rozčleněný podle kategorií. Položky si zákazník přidává do košíku. U položek lze nastavit množství, které si zákazník přeje objednat. Následně přechází do pokladny, kde si vybere způsob předání – zda si objednávku vyzvedne na prodejně nebo si ji přeje doručit. Vyplní kontaktní údaje, jako jsou jméno a příjmení, e-mail, telefon, případně adresu, pokud se jedná o rozvoz. Zákazníkovi se před odesláním zobrazí souhrn objednávky, tedy všechna zboží, které objednal, počet kusů a celkovou částku objednávky. Při odeslání objednávky se zákazník dostane na děkovnou stránku, kde jsou uvedeny informace, které si prodejna nastavila v administraci. Zároveň mu na uvedenou e-mailovou adresu přijde e-mail se souhrnem objednávky.

E-mail o nové objednávce přijde i dané prodejně, pokud má aktivované notifikace ve své administraci. Výpis všech objednávek má prodejna k dispozici v administraci, kde si lze filtrovat objednávky podle jejího stavu – zda je vyřízená, nová nebo podle typu předání. Výpis objednávek obsahuje všechny informace, které zákazník vyplnil při odeslání objednávky.

4.4 Návrh databáze

Databáze se skládá z deseti tabulek. Tato kapitola prochází každou z nich a popisuje její účel a propojení s ostatními tabulkami. Pro jednodušší představu je zde vložen ER diagram (Obrázek 4.2) celé databáze.

Tabulka `stores`

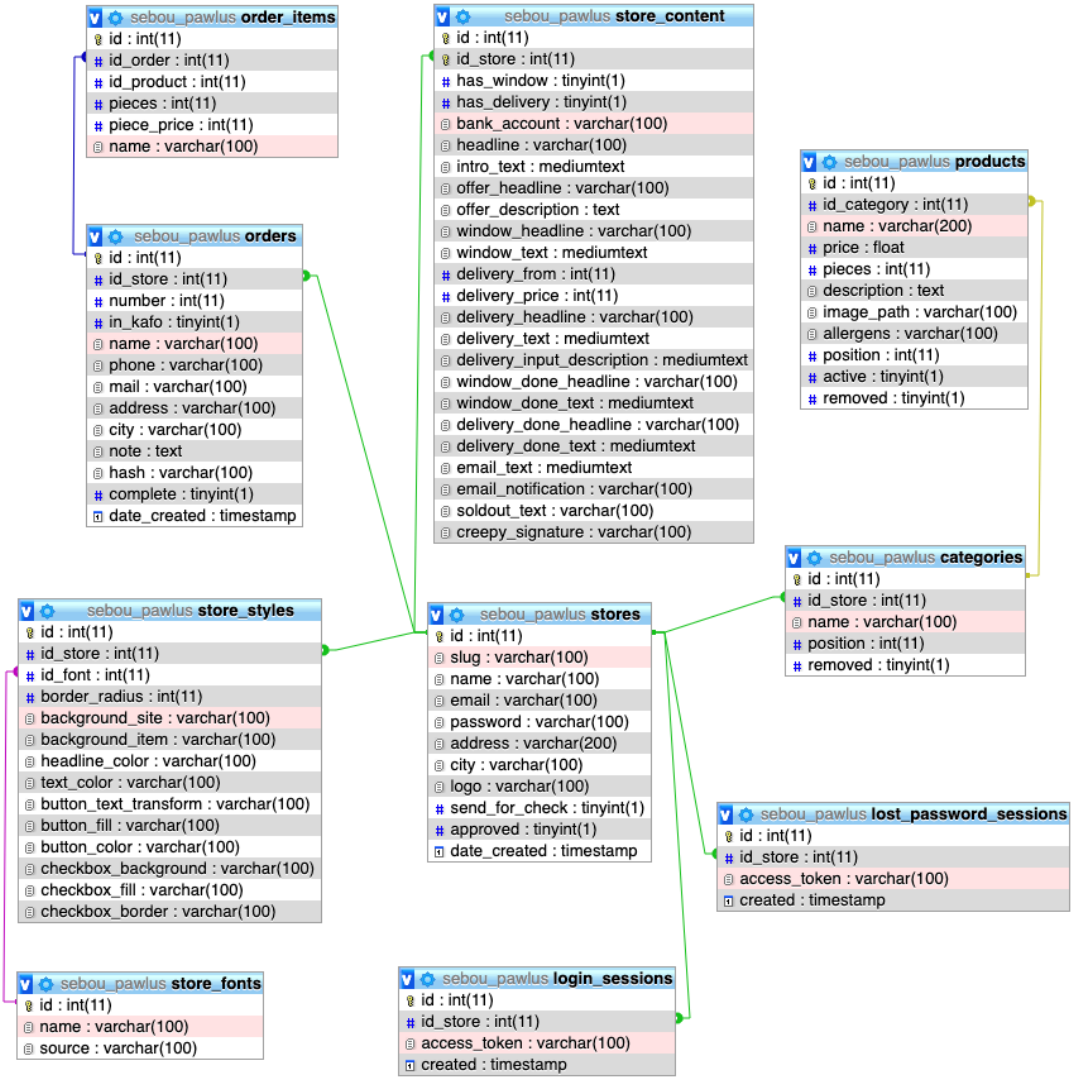
Tabulka `stores` uchovává informace o jednotlivých prodejnách. Při založení nové prodejny se v ní vytvoří nový záznam obsahující jméno subdomény, název prodejny, e-mail a heslo, které slouží pro přihlášení do administrace pro správu prodejny. Dále se v této tabulce ukládá adresa, cesta k souboru loga a informace o tom, zda se prodejna odeslala ke schválení a jestli ji administrátor aplikace schválil k provozu, čili je dostupná pro veřejnost.

Tabulka `categories`

Tabulka `categories` uchovává kategorie produktů. Obsahuje cizí klíč `id_store`, který je odkazuje na `ID` prodejny z tabulky stores. Kategorie ukládá název, pozici v nabídce, aby si prodejna mohla určovat pořadí zobrazené zákazníkům a zda je kategorie smazaná. Smazané záznamy zůstávají dále v databázi, aby se informace o kategorii mohly využívat zpětně u objednaných produktů.

Tabulka `products`

V tabulce `products` se ukládají jednotlivé produkty nabídky prodejny. Produkty jsou vždy spojené s kategorií cizím klíčem s tabulkou `categories`. Není tedy možné vytvořit produkt, který by nespadal do žádné kategorie. Jednotlivé záznamy produktu obsahují název, cenu,



Obrázek 4.2: ER diagram databáze aplikace

aktuální počet kusů na skladě, popis produktu, cestu k fotce produktu, alergeny, pozici, která stejně jako u kategorií určuje pořadí v nabídce. Dále rozlišuje, zda je produkt aktivní – produkt je vidět v nabídce a dá se objednat, nebo neaktivní – produkt není v nabídce a nedá se objednat. Zůstává však v administraci pro možné zařazení zpět mezi aktivní. Záznam uchovává stejným principem, jako u kategorií, zda je produkt smazaný.

Tabulka `orders`

Do tabulky `orders` se ukládají záznamy o přijaté objednávce. Cizím klíčem se spojena s tabulkou `stores`, díky kterému rozlišuje, které prodejně objednávka patří. Každý záznam o tabulce obsahuje osobní údaje o zákazníkovi, hodnota `in_kafo` (pojmenování pochází z první verze, kdy byla aplikace jen pro zlínskou kavárnu Kafö) rozlišuje, zda si objednávku zákazník vyzvedne na pobočce nebo si ji přeje dovézt. Dále je v záznamu uložena poznámka zákazníka, datum vytvoření, hash objednávky pro bezpečné zobrazení děkovné stránky,

popsané v kapitole 5. Záznam uchovává i hodnotu rozlišující stav objednávky, jestli už byla dokončena.

Tabulka `order_items`

Tabulka `order_items` obsahuje záznamy nakoupených produktů které cizím klíčem odkazují na `ID` objednávky. V jednotlivých záznamech se ukládá ID, název a cena produktu. Název a cena produktu nejsou redundantní k záznamu produktu, jelikož prodejna může produkty aktualizovat. Odkazování a zobrazení informací přímo z tabulky produktu by mohlo zapříčinit chybný název a cenu původně objednané položky.

Tabulka `store_content`

Tabulka `store_content` cizím klíčem odkazují na prodejnu v tabulce `stores` s kardinalitou 1:1. Tedy každý záznam v tabulce store má právě jeden záznam v tabulce `storey_content`. Záznamy jsou rozděleny do dvou tabulek především kvůli přehlednosti databáze a rozdělení celků s podobným účelem do své tabulky. Záznamy obsahují informace o prodejně a texty, které prodejna chce svým zákazníkům v objednávkovém formuláři sdělit včetně textu v e-mailech.

Tabulka `store_styles`

Tabulka `store_style` stejně jako tabulka store_content s kardinalitou 1:1 k store obsahuje CSS pravidla vzhledu formuláře prodejny. Jedná se především o definice barev v hexadecimálním tvaru, `ID` fontu z tabulky `store_fonts` propojené cizím klíčem, a další stylové atributy.

Tabulka `store_fonts`

Tabulka `store_fonts` obsahuje záznamy s názvem a zdrojem fontu, které si prodejna může zvolit v administraci pro svůj objednávkový formulář.

Tabulka `login_sessions`

Tabulka obsahuje záznamy přihlášených uživatelů, díky kterým se uživatel nemusí přihlašovat při každé návštěvě znovu, ale podle daného hashe se přihlásí ke svému účtu.

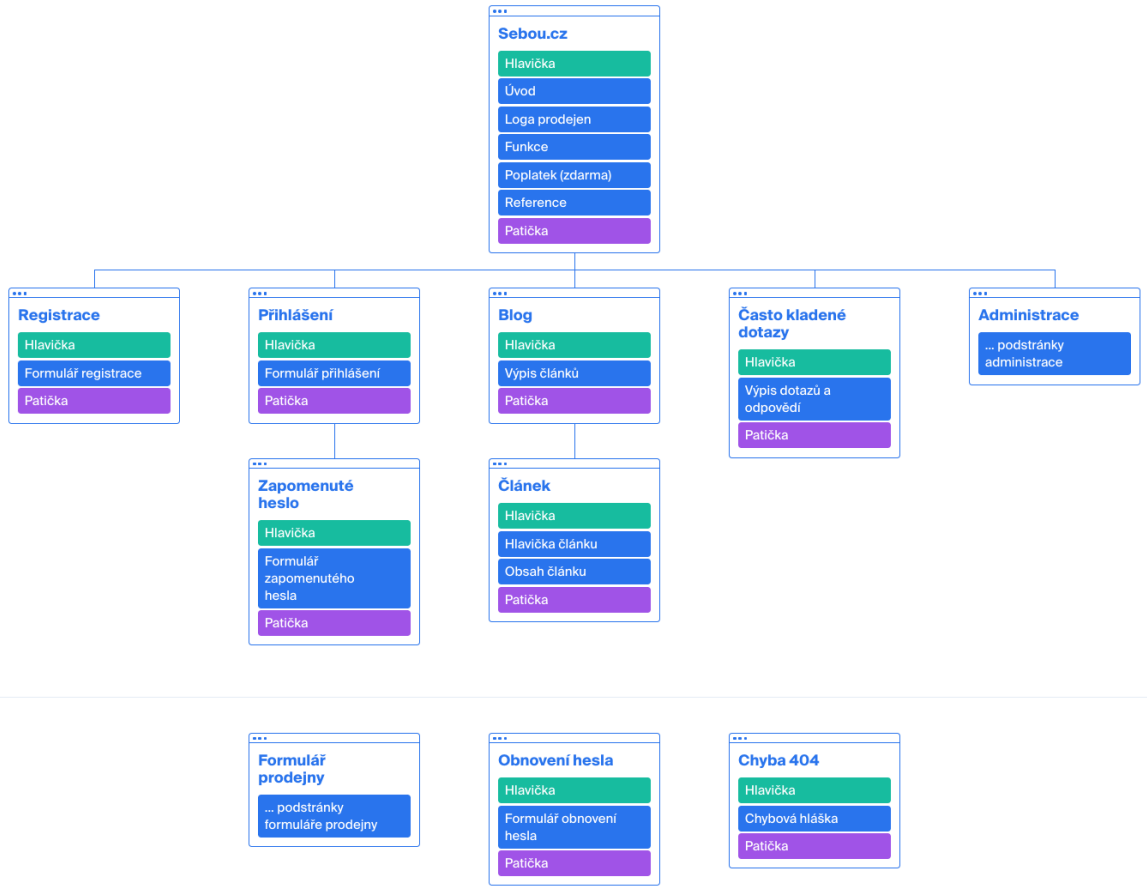
Tabulka `login_password_sessions`

Poslední tabulka uchovává záznamy pro zapomenutá hesla. Pokud uživatel zapomněl heslo, vygeneruje se hash, který se uloží do této tabulky a zároveň se odešle uživateli v odkazu zapomenutého hesla e-mailem. Při změně hesla se zkontroluje platnost hashe z této tabulky. Pokud je hash platný, uživateli se aktualizuje heslo.

4.5 Uživatelské rozhraní

V této kapitole jsou popsány jednotlivé části uživatelského rozhraní. U uživatelského rozhraní je důležitá intuitivnost rozložení prvků aplikace. Nemělo by uživatele zbytečně zatěžovat hledáním potřebných ovládacích prvků či informací v nepřehledném zobrazení. Drátěný model (wireframe), který se standardně u návrhu uživatelských rozhraní vytváří, je u této

aplikace vynechán. Situace si žádala co nejrychlejší řešení, a tak se začalo rovnou tvorbou grafického návrhu aplikace, jejímž autorem je designér David Dvořáček. Uživatelské rozhraní popisuje vytvořená mapa stránek (Obrázek 4.3), ze které designér vycházel.



Obrázek 4.3: Hlavní mapa stránek aplikace Sebou.cz

Hlavní stránka

Hlavní stránka by měla zahrnovat veškeré informace o aplikaci, návštěvník webu by se zde měl dozvědět, zda je aplikace vhodná pro jeho potřeby. Nachází se zde mockupy v reálném užití. Jsou zde loga prodejen, od známějších po méně známe podniky, aby majitele prodejny ujistil, že se aplikace reálně používá. Dále jsou zde vypsány funkce, které popisují účel celé aplikace. Hlavní stránka obsahuje odkazy na přihlášení do administrace, založení nové prodejny, blog a často kladené odkazy.

Objednávkový formulář – nabídka

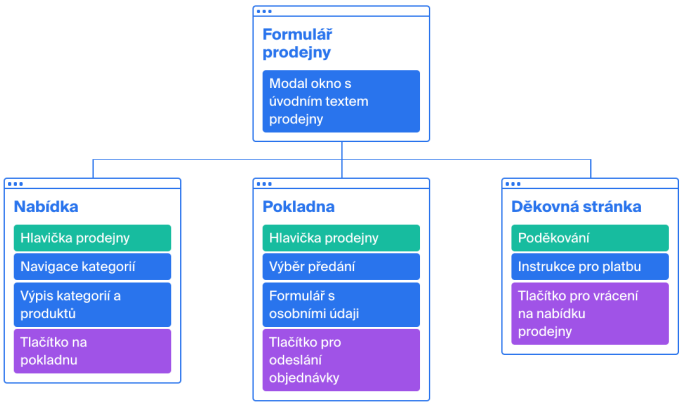
Při vstupu na nabídku prodejny by se nejdříve mělo zobrazit modalové okno, ve kterém bude informace o daném formuláři prodejny, který bude možné editovat z administrace. Při zavření okna by se zákazníkovi znovu nemělo daný den ukázat, aby při příští návštěvě mohl rovnou objednávat. Nabídka má hlavičku – nadpis a doplňující informace. Pod hlavičkou je navigace kategorií, která je fixní a zákazník ji tak má pořád k dispozici. Následuje samotný

výpis. Ten obsahuje vždy název kategorie a produkty, které do ní spadají. U produktu se pak zobrazí název, popisek, cena. Případně fotka a alergeny, pokud jsou u produktu uvedeny. Zákazník by měl stále vidět základní údaje o přidaných položkách v košíku a tlačítko pro přechod k pokladně.

Objednávkový formulář – pokladna a děkovná stránka

Na stránky pokladny pak bude formulář pro vyplnění osobních údajů a volba typu doručení. Formulář se bude dynamicky měnit na základě typu převzetí objednávky. Základní formulář obsahuje jméno, e-mail, telefon a poznámku. Pokud si zákazník vybere rozvoz, přidají se položky adresa a město, do kterého si zákazník přeje objednávku přivézt. Před samotným potvrzením údajů a odesláním objednávky by se měl zákazníkovi zobrazit její souhrn, který obsahuje položky v košíku, počty kusů a celkovou cenu objednávky. Po odeslání objednávky aplikace přesměruje uživatele na děkovnou stránku.

Děkovná stránka obsahuje potvrzení o odeslání objednávky s děkovným textem. Pokud prodejna využívá platby na účet, zobrazí se zde číslo účtu, variabilní symbol a částku, kterou má zákazník na účet poslat. Souhrn objednávky přijde zákazníkovi i e-mailem.



Obrázek 4.4: Mapa stránek formuláře prodejny

Založení nové prodejny

Na založení nové prodejny se uživatel dostane z hlavní strany. Stránka obsahuje formulář, v něm je potřeba vyplnit jméno subdomény, na kterém prodejna bude aplikaci poskytovat svým zákazníkům. Dále správce prodejny vyplní e-mail a heslo. Při odeslání formuláře uživatele aplikace přesune do administrace pro správu prodejny.

Přihlášení, zapomenuté heslo, vytvoření nového hesla

Stránky obsahují formulář, pro vyplnění potřebných údajů podle povahy dané stránky. Stránka obsahuje hlavičku a patičku webu, která je shodná s hlavní stranou, aby se uživatel mohl vrátit, či přejít na jinou stránku z navigace v hlavičce.

Administrace

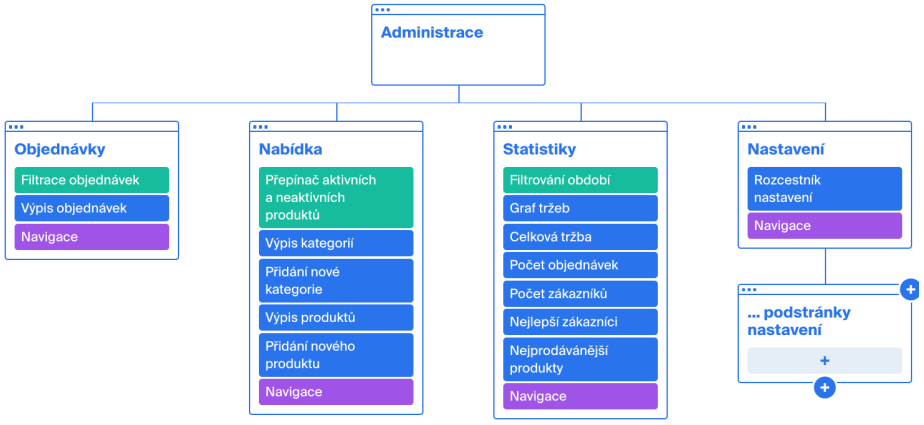
Administrace, jak jde vidět z mapy stránek, je největší částí aplikace. Díky menu ve spodní části obrazovky, které se často používá u nativních mobilních aplikací, administrátor bude

vždy vědět, v jaké části se nachází. Spodní menu by mělo obsahovat nejdůležitější položky. Nachází se zde „Objednávky“, „Nabídka“, „Statistiky“ a „Nastavení“. Měla by být graficky odlišená aktuální stránka. U každé položky by pak neměla chybět ikonka pro zvýšení orientace uživatele.

Stránka objednávek musí obsahovat filtraci podle stavu a typu doručení. Filtrace by měla být umístěna v horní části a fungovat formou přepínače. Filtrace musí obsahovat i možnost zobrazit všechny objednávky. Jednotlivé objednávky pak musí obsahovat všechny osobní údaje, které zákazník vyplnil, včetně objednaných produktů, jejich množství a celkové ceny objednávky. U objednávky je potřeba mít možnost rychle zavolat zákazníkovi, aby jej kurýr informoval, že je objednávka již na cestě. To se dá vyřešit pomocí tlačítka, které při kliknutí spustí hovor na uvedené číslo při objednávce.

Ná stránce nabídky je potřeba rozlišit aktivní a neaktivní nabídku. Aktivní nabídka je vidět na webu a je možné z ní objednávat. V neaktivní nabídce si prodejna může odkládat produkty nebo si chystat budoucí nabídku, která nyní nelze objednávat a není vidět na webu. Rozlišení můžeme vyřešit filtrací, stejně jako na stránce objednávek. Nabídka dále obsahuje kategorie, do kterých spadají jednotlivé produkty. Na stránce nabídky by mělo být možné přidávat nové kategorie a produkty, případně je upravovat. Jednotlivé kategorie a produkty musí být možné pomocí přetáhnutí přesouvat a určovat tak pořadí, ve kterém produkty uvidí zákazníci.

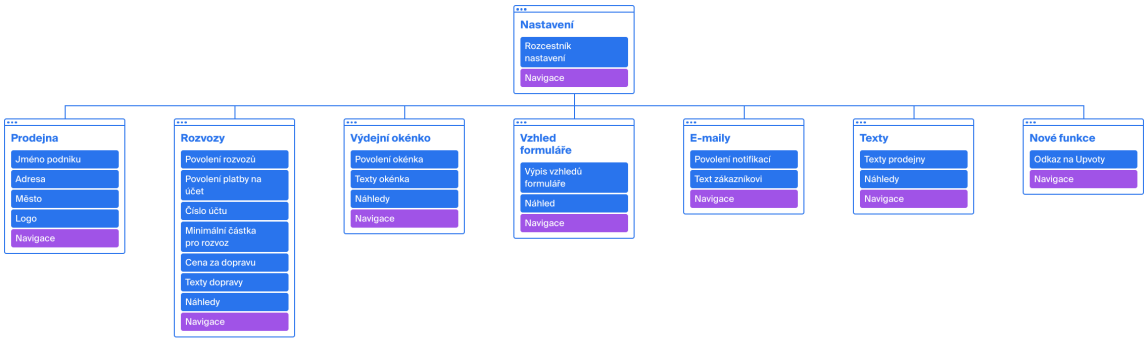
Statistiky by měly přehlednou formou pomocí grafů, číselných hodnot a seřazených seznamů zobrazit data objednávek a produktů. Musí být možné zvolit období, pro které se statistiky zobrazí. V grafu je potřebné mít možnost pomocí kliknutí získat přesnou hodnotu. Statistiky by měly obsahovat vývoj tržby pomocí grafu, celkový počet objednávek a zákazníků formou číselných hodnot. Dále nejprodávanější produkty a nejčastější zákazníky formou seznamu.



Obrázek 4.5: Mapa stránek administrace prodejny

V nastavení je potřeba zobrazit základní údaje o prodejně. Kvůli velkému počtu položek, které se dají nastavit je potřeba umístit na úvodní stránku nastavení nejprve rozcestník, který bude odkazovat na podstránky a seskupí tím podobné položky nastavení. První podstránkou by mělo být nastavení základních údajů prodejny. Ve formuláři zde administrátor uvidí předvyplněné inputy, pokud již byly dříve vyplněny. Na této stránce je také možnost nahrát logo prodejny. Další podstránkami je nastavení rozvozů a výdejního okénka. Zde má administrátor možnost aktivace a vyplnění textů a podmínek pro daný typ předání. Administrátorovi je potřeba poskytnout náhled, jak v daném nastavení uvidí formulář zákazníci.

Další podstránkou nastavení je výběr vzhledu, zde lze nastavit, v jaké typografii a kombinaci barev se objednávkový formulář zobrazí. Ve výpisu musí být každá možnost zobrazena jejím písmem a barvami. Aktuálně zvolená možnost bude zobrazená v náhledu formuláře. Další podstránkou je nastavení textů. Ty jsou bude možné pomocí formuláře měnit a znovu mít možnost v náhledu zobrazit tak, jak je uvidí zákazníci. Následuje podstránka e-mailů, kde si administrátor může pomocí checkboxu zapnout notifikace o nové objednávce a text, který se pošle zákazníkovi spolu se souhrnem objednávky. Dále je v rozcestníku návrh na přidání nových funkcí, jedná se o informativní stránku s odkazem na aplikaci Upvoty, která je blíže popsaná v kapitole 4.6. Poslední položkou v rozcestníku je možnost odhlášení, tady se nejedná o odkaz na podstránku, ale pouze o tlačítko, které při kliknutí uživatele odhlásí z administrace.



Obrázek 4.6: Mapa stránek nastavení prodejny v administraci

Blog

Blog je stránka, na které jsou vypsány všechny články. Jednotlivá karta článku obsahuje náhledovou fotku, název, nadpis, autora, přibližná doba čtení a urývek z textu. Stránka blogu neobsahuje stránkování, jelikož se nepředpokládá, že zde bude vysoký počet článků. Pokud by počet článků byl vyšší, bylo by potřeba implementovat stránkování či lazyloading.

Článek blogu

Jednotlivý článek blogu obsahuje mimo informace uvedené ve výpisu článků celý jeho obsah. Jelikož se v projektu využívá Strapi a obsah blogu je psán ve WYSIWYG editoru využívající jazyk Markdown, můžou se zde objevit jakýkoliv jeho formát.

Často kladené dotazy

Často kladené dotazy by měl být seznam nadpisů otázek. Odpověď dotazu by se měla ukázat až při kliknutí na nadpis dané otázky, aby uživatel viděl, co nejvíce často kladených dotazů zaráz. Dále je dobré dotazy rozdělit do skupin podle typu otázky.

4.6 Získávání zpětné vazby

Pro získávání zpětné vazby je použita aplikace Upvoty. V ní mohou prodejny přidávat návrhy na vylepšení. Jednotlivé návrhy jsou pak ostatním prodejnám a vývojářům aplikace dostupné v seznamu, ve kterém je možné filtrovat. Návrhy obsahují aktuální stav, který

určuje vývojář aplikace. Ostatní prodejny mohou daný návrh podpořit hlasováním. Podle počtu hlasů se pak mohou nové funkce implementovat prioritně. Hlasování také určuje, kolik prodejen novou funkci potřebuje. Zpětná vazba je důležitá k určování směru dalších verzí aplikace.

Kapitola 5

Implementace

V této kapitole se popisuje samotná implementace aplikace. Má vystihnout především zajímavé části vývoje, neobsahuje tedy detailní popis řešení implementace. Kapitola je rozdělena na implementaci frontendu a backendu. Databáze je již popsána dříve v návrhu aplikace. Čtenář by se měl dozvědět základní principy vývoje aplikací v javascriptovém frameworku Vue.js a vývoje API.

5.1 Implementace webové aplikace ve Vue.js a Nuxt.js

Nejdříve je potřeba vytvořit nový Nuxt.js projekt. Je potřeba mít nainstalováno `npx`, která je součástí instalace `NPM`. Projekt je pak možné vytvořit pomocí příkazu v terminálu `npx create-nuxt-app <project-name>`. Instalace obsahuje pár základních kroků o povaze aplikace, zde je potřeba zvolit, že se jedná o SPA aplikaci. Instalace nám vytvoří strukturu projektu a nainstaluje všechny potřebné závislosti včetně frameworku Vue.js. Ve struktuře projektu je potřeba vyzdvihnout soubor `nuxt.config.js`, který obsahuje konfiguraci projektu. Zde je možné definovat informace o aplikaci, aktuální verze používaných pluginech, atd. Struktura projektu dále obsahuje složky. Tyto složky monitoruje právě framework Nuxt.js a podle jejich obsahu vytváří kostru aplikace ve Vue.js. To nám ulehčí mnoho monotónní práce.

5.1.1 Směrování

Jak už je v této práci dříve zmíněno, jedná se o jednostránkovou aplikaci, která mění svůj obsah na základě interakcí uživatele. Každá podstránka tedy vede do souboru `index.html`, ve které se zobrazí pomocí Javascriptu potřebný obsah. O zobrazení správného obsahu se stará `Vue router`, který je generován frameworkem Nuxt.js na základě souborů ve složce projektu `pages`.

5.1.2 Komunikace s API

Pro komunikaci s API je využitý framework Axios. Pomocí něj získáme potřebná data z API, které je potřeba na frontendu zobrazit. Jednotlivé dotazy jsou promise-based. To znamená, že jsou posílány asynchroně a můžeme definovat, co se stane po přijetí odpovědi. V praxi to znamená, že uživatel může dál s aplikací pracovat a komunikace s API probíhá na pozadí. Axios je dále využíván k odeslání vyplněných formulářů včetně nové objednávky.

Validace probíhá na straně API, kdy vrací kód `200`, nebo jiný pokud nastala chyba. Na základě chybového kódu se uživateli zobrazí hláška, která blíže specifikuje chybový stav.

5.1.3 Stránky

Každý soubor ve složce pages je konkrétní stránka v aplikaci. Pomocí dalších složek můžeme vytvořit víceúrovňově stránkování. Pokud složka nebo soubor začíná podtržítkem (např. `/blog/_single.vue`), jedná se o dynamickou stránku, kde na pozici `_single` může být jakákoliv hodnota v URL adrese. Hodnotu pak můžeme ve Vue.js získat a na základě ní zobrazit potřebný obsah. Stránky mají tedy následující adresářovou strukturu, která vychází z mapy stránek:

```
/pages
├── /admin
│   ├── /nastaveni
│   │   ├── emaily.vue
│   │   ├── index.vue
│   │   ├── okynko.vue
│   │   ├── podpora.vue
│   │   ├── prodejna.vue
│   │   ├── rozvozy.vue
│   │   ├── texty.vue
│   │   └── vzhled.vue
│   ├── /statistiky
│   │   └── index.vue
│   ├── nabidka.vue
│   └── objednavky.vue
├── /alzheimer
│   ├── _resetpasword.vue
│   └── index.vue
├── /blog
│   ├── _single
│   │   └── index.vue
│   └── index.vue
├── faq.vue
├── index.vue
├── pokladna.vue
└── prihlaseni.vue
```

5.1.4 Šablony stránek

Jelikož mnoho stránek obsahuje stejné rozložení prvků, můžeme definovat šablony stránek. Například všechny stránky v administraci mají vždy ve spodní části stejné menu, které jen zvýrazňuje aktuální stránku. Stačí tedy vytvořit šablonu stránky pro administraci, která se přiřadí všem stránkám a podstránkám administrace. V těchto souborech díky tomu není potřeba vkládat zvlášť komponentu či celý kód stejného menu. Logiku zvýraznění je pak definováno přímo v této šabloně. Zároveň je v jediném souboru šablony možné měnit rozložení všech stránek administrace. To zvyšuje přehlednost a rychlost při úpravách. V projektu

jsou definovány přesně čtyři šablony. První z nich je již zmíněná šablona pro administraci, druhá z nich je pro samotný formulář prodejny, třetí slouží pro zobrazení stránek, pro zobrazení obsahu na větších šířkách zařízení. Poslední šablonou je pro chybové stavy jakým je například chyba `404`, kdy není požadovaná stránka nalezena. Ve frameworku Nuxt.js se chybová stránka vždy definuje pomocí šablony, nikoliv pomocí stránek.

5.1.5 Komponenty

V celé aplikaci je definováno celkem 53 komponent. Komponenta v tomto projektu můžou obsahovat pokročilou funkci. Zároveň může být definována jen pro zpřehlednění kódu aplikace. V této kapitole jsou zmíněny nejzajímavější komponenty aplikace.

První z nich je komponenta produktu, kterou vidí zákazníci ve formuláři prodejny. Do ní se pomocí props posílají data daného produktu. Komponenta pak rozlišuje celkem dva stavy podle toho, zda je produkt vložen v košíku nebo není. Produkt přidaný v košíku je zvýrazněn a tlačítko pro přidání do košíku se vymění za tlačítka pro zvýšení či snížení počtu kusů. Pokud data obsahují odkaz na fotku produktu, zobrazí se i fotka. Ta je možná kliknutím zvětšit pro zobrazení na celou šířku obrazovky formou modalu. Logiku zobrazení a schování definuje proměnná typu boolean `showDetail`. Při hodnotě `true` se modal s fotkou zobrazí a při jeho zavření se nastaví na hodnotu `false` a tím se schová. Dále komponenta obsahuje proměnnou, jejíž hodnota je aktuálně vložený počet kusů v košíku. Na počet kusů je nastavená metoda `watch` (popsána dříve v kapitole 3.1.1), která při její změně předá novou hodnotu `Vuex store` košíku. Komponenta dále obsahuje CSS pravidla, která definují vzhled položky produktu.

Druhou popsanou komponentou je položka přijaté objednávky v administraci prodejny. Obsahuje výpis jedné objednávky. Data jsou předána komponentě stejně jako u produktu pomocí props. Datum a čas objednání je formátován pomocí knihovny Moment.js. Komponenta obsahuje tlačítko pro zařazení objednávky mezi vyřízené respektive vrácení zpět mezi nové. Pro zařazení je v této komponentě definována metoda `complete()`, ve které se pomocí knihovny Axios pošle dotaz do API aplikace. HTML obsahuje tlačítko pro rychlé vytočení telefonního čísla zákazníka. To je implementováno pomocí značky `<a>`, kde se před telefonní číslo přidá zkratka `„tel:"`. Komponenta dále vypisuje pomocí odrážkového seznamu objednané produkty a počet kusů. Důležité v této komponentě je zobrazení celkové ceny za objednávku. Cena se počítá přímo v API, stačí ji tak pouze vypsat.

Třetí zmíněnou komponentou je prvek formuláře `inputWrapper`. Jedná se o nejsložitější komponentu v celém projektu. Implementuje jeden prvek formuláře, který může být jakéhokoliv typu značky HTML `<input />`. Mimo něj jej tahle komponenta rozšiřuje další o možné typy. Konkrétně o výběr data, nahrání fotky s jejím náhledem pro kontrolu ještě před odesláním formuláře. Dále `checkbox`, který je kvůli stylu definován vlastní komponentou místo klasické definice pomocí značky `<input />`. Komponenta podporuje i typ textového pole, který vloží standardní značku `<textarea>`. Požadovaný typ se posílá pomocí `props`.

- Pomocí props je možné nastavit i další vlastnosti komponenty, konkrétně:
  - `value` Výchozí hodnota. Může být definována staticky nebo dynamicky pomocí hodnoty získanou s API. Například pro editaci již vyplněných údajů.
  - `placeholder` Vloží klasický HTML atribut do značky `<input>`.
  - `label` Přidá před input jeho název pomocí HTML značky `<label>`.
  - `isRequired` Hodnota boolean, která určuje, zda je pole povinné vyplnit. Pole se označí červeně, pokud ho uživatel nevyplní.

- `uploadInfo` Textové instrukce u nahrávání souboru, pokud je input typu `file`.
- `pattern` Vloží standardní pattern atribut, díky kterému můžeme pomocí regulárního výrazu definovat požadovaný formát vyplněné hodnoty. Prohlížeč pak neumožní odeslání formuláře, pokud není formát dodržen.
- `previewURL` Definuje adresu již nahrané fotky u inputu typu `file`.
- `maxSize` Definuje maximální velikost souboru u inputu typu `file`. Pokud je velikost překročena, uživateli se zobrazí upozornění.
- `autocomplete` Název typu pro automatické předvyplnění formuláře, které definovala společnost Google[1].

Vyplněná hodnota formuláře je vrácena zpět nadřazené komponentě. To je možné pomocí funkce Vue.js `emit()`, která umí komponentě poslat událost včetně hodnoty. Tato událost je pak v nadřazené komponentě odchycena a hodnota uložena do objektu dat formuláře, který se pošle pomocí knihovny Axios do API při odeslání formuláře.

Čtvrtou popsanou komponentou je mockup pro náhled aktuálního nastavení prodejny před potvrzením nových změn v administraci. Vkládá další komponentu `cssVars`, která na základě používaného vzhledu prodejny nadefinuje CSS pravidla. Tato pravidla se pomocí požadavku získají z API. Samotná komponenta mockupu pak obsahuje kromě CSS pravidel rámeček, který znázorňuje obrazovku telefonu. Veškerý obsah pak do komponenty předáme pomocí slotu. Výsledkem tedy bude náhled v aktuálně používaném vzhledu prodejny.

5.1.6 Objednávkový formulář prodejny

Při příchodu na objednávkový formulář se z API nejdříve získají všechna potřebná data. Ta zahrnují kategorie, produkty a samotné nastavení prodejny. Požadavek odesílá aktuální jméno subdomény prodejny. To je potřeba pomocí Javascriptu získat z aktuální URL adresy z objektu `location`. Pokud API vrátí, že daná prodejna nebyla nalezena, aplikace uživatele přesměruje na chybovou stránku `404`. Získávání dat prodejny je implementováno pomocí `middleware`. Middleware knihovny Nuxt.js umožňuje definovat kód, který se vykoná před načtením stránky. Definuje se v adresáři projektu middleware javascriptovým souborem. U stránek, kde chceme middleware spustit stačí jen definovat název tohoto souboru. Na všech stránkách formuláře objednávky je definovaný middleware `store_initial`, který se stará o načtení dat případně přesměrování na chybovou stránku.

V prvním kroku objednávky se zobrazují produkty, které je možné přidávat do košíku. Ve druhém kroku se pak vyplňují osobní údaje zákazníka a následuje odeslání objednávky. Po odeslání se nejdříve vloží do objektu všechna vyplněná data a košík ve formátu, které potřebuje API. Následuje odeslání pomocí knihovny Axios do API. Kde proběhne validace všech údajů a vrátí v odpovědi kód `200` pokud všechno proběhlo v pořádku. Chybu `400` pokud validace neproběhla úspěšně a chybu `406`, pokud validace proběhla úspěšně, ale zákazníka někdo předběhl a zboží v košíku již není k dispozici. Po úspěšném dokončení objednávky je uživatel přesměrován na děkovnou stránku.

5.1.7 Košík

Obsah košíku se spravuje pomocí `Vuex store`. To je místo kde můžeme definovat různá data a metody, která jsou přístupná z jakékoliv komponenty aplikace. Košík obsahuje celkem 3

[1] https://developers.google.com/web/updates/2015/06/checkout-faster-with-autofill

proměnné. První z nich je název subdomény aktuální prodejny, pomocí kterého přiřadíme produkty v košíku správné prodejně. To je potřeba z důvodu, kdy zákazník bude na Sebou.cz objednávat u více prodejen. Druhou proměnnou je pole samotných položek produktů. Třetí je pole poplatků, do kterého se vkládá cena za dopravu. Košík dané prodejny se ukládá souběžně i do `localstorage` prohlížeče. Pokud by se obsah košíku nikam neukládal, při obnovení stránky by byl znovu prázdný. Jméno klíče, pod kterým je košík v `localstorage` uložen obsahuje název a aktuální datum. Díky tomu načteme vždy správný obsah košíku, který je platný jen pro daný den. To je z důvodu, že většina prodejen má denní nabídku. Následující den by pak ve formuláři nebyly vložené produkty v košíku.

Dále jsou implementovány metody, díky kterým spravujeme obsah košíku. Metoda `init_cart` načte při příchodu na formulář dříve vložené produkty z `localstorage`. Metoda `add_product` vloží nový produkt do košíku. V případě, že již produkt existuje, se provede smazání a následně se vloží produkt znovu s aktuálním počtem kusů. Metoda `remove_product` smaže daný produkt z košíku a `localstorage`. Metody poplatků `add_fee` respektive `remove_fee` fungují na stejném principu jako produkty.

Vuex store košíku obsahuje i metody `getters`. Ty umožňují definovat funkce vracející data z aktuálního stavu. Například implementovaný getter `total()` vrací celkovou částku objednávky. Vychází z vložených produktů a poplatků v košíku. Getter `itemsCount` vrací celkový počet položek produktů v košíku, který se zobrazuje zákazníkovi ve formuláři objednávky.

5.1.8 Autorizace administrátora

Při přihlášení do administrace prodejny se odešle požadavek s přístupovými údaji do API. Odpověď vrací, zda autorizace proběhla úspěšně. Při neúspěšném pokusu se uživateli zobrazí chybová hláška. Pokud je všechno v pořádku, API vrátí session hash. To je vygenerovaná hodnota, která obsahuje 32 znaků. Tato hodnota se uloží do `localstorage` prohlížeče, aby uživatel zůstal přihlášený po obnovení stránky i při jeho příští návštěvě.

Zároveň je v aplikaci vytvořený plugin. Ten vkládá hash do hlavičky dotazu v knihovně Axios. Pomocí hashe se pak autorizují jednotlivé administrátorské dotazy. Dále se pomocí něj rozlišují data jednotlivých prodejen, jelikož jsou všechny hashe spojené s daným `ID` prodejny v tabulce databáze `login_sessions`.

Autorizace při obnově stránky či vrácení na web probíhá pomocí dalšího `middleware` názvem `adminAuth`. Ten zkontroluje, zda `localstorage` obsahuje hash. Dále pošle dotaz na svá data prodejny, které jsou potřeba zobrazovat v administraci. Pokud localstorage již hash neobsahuje, přesměruje aplikace uživatele na stránku přihlášení.

5.1.9 Zapomenuté heslo

Pokud administrátor zapomněl své heslo do administrace, může si požádat o jeho obnovení. Obnovení probíhá pomocí formuláře. V něm uživatel vyplní e-mailovou adresu, na který byla prodejna registrována. Na adresu se následně pomocí API odešle e-mail, který obsahuje odkaz zpátky na web Sebou.cz pro obnovu hesla. Zde uživatel zadá nové heslo, vyplní jej znovu pro ověření, že neudělal chybu. Po odeslání formuláře s obnovením hesla se může ihned uživatel s novými údaji přihlásit.

5.1.10 Administrace

Před přístupem do administrace se vždy kontroluje, zda je daný uživatel přihlášený. To nám zajistí již zmíněný middleware adminAuth. Jelikož aplikace obsahuje mnoho administračních stránek, definujeme `middleware` globálně pro všechny stránek webu. V samotném `adminAuth` pak budeme kontrolovat zda URL adresa aktuální stránky obsahuje „/admin“. Díky tomu rozlišíme, kdy se má autorizace v `middleware` provést.

Dotaz autorizace nám vrátí data prodejny, pokud proběhla úspěšně. Data obsahují informace o objednávkách, nabídce a nastavení prodejny. To má za výhodu, že procházení stránek v administraci již nepotřebují žádné další dotazy pro zobrazení obsahu, a tak je administrace velice rychlá za předpokladu. To však pouze za předpokladu nižšího počtu objednávek.

Na úvodu stránky výpisu objednávek se nachází filtrování. To je implementováno v komponentě `switchNav.vue`. Pomocí funkce `emit` se odchytí změna filtru. Podle změny se pak zobrazí požadované objednávky. Na stejném principu funguje zobrazení aktivních a neaktivních produktů a výběr časového intervalu na stránce statistik.

5.2 Implementace API

Tato kapitola popisuje implementace API v jazyce PHP. API komunikuje s databází a frontendem aplikace. Dále implementuje odesílání e-mailů, ukládání a ořez nahraných obrázků. Jsou zde popsány jednotlivé třídy objektově orientovaného programování v PHP. Kapitola ujasňuje pojem „koncový bod“ včetně konkrétního užití. Na závěr kapitoly jsou vysvětleny chybové stavy, které API vrací v odpovědi, pokud nastala chyba.

5.2.1 Třídy

API obsahuje celkem šest tříd. Samotné funkce pomocí knihovny Medoo komunikuje pomocí dotazů s databází. Hlavní funkcí je třída `API`. Má na starost připojení k databázi, jsou v ní definovány funkce, které se používají i v dalších třídách. Je v ni implementována funkce pro získání dat dotazu a jejich validaci.

- Popis funkcí třídy `API`:
  - `connection` Připojení k databáze pomocí knihovny Medoo.
  - `request` Jednotlivý dotaz do databáze.
  - `is_localhost` Vrací hodnotu, zda API právě běží na localhostu.
  - `get_request_data` Vrací data zaslaná v požadavku do API.

Drou třídou je `Store`. Definuje funkce pro práci s daty prodejny.

- Funkce třídy `Store` pro získání dat:
  - `get_id_from_slug` Vrátí ID prodejny na základě jména její subdomény.
  - `get_store_data` Vrátí všechny údaje o prodejně.
  - `get_public_store_data` Vrátí základní údaje o prodejně, které jsou veřejné.
  - `get_public_product_categorie` Vrátí všechny vytvořené kategorie prodejny, které nebyly smazány.

  - `get_store_products` Vrátí všechny produkty prodejny, které nebyly smazány.
  - `get_public_store_products` Vrátí všechny produkty prodejny, které nebyly smazány a jsou aktivní.
  - `get_store_content` Vrátí všechna data prodejny z databázové tabulky store_content.
  - `get_store_orders` Vrátí všechny objednávky prodejny.
  - `get_store_initial` Vrátí všechna potřebná data pro objednávkový formulář prodejny.
  - `get_admin_session` Vrátí všechna potřebná data pro administraci prodejny.
  - `get_css` Vrátí CSS pravidla, která definují vzhled prodejny.
  - `get_store_style_slug` Vrátí ID stylu prodejny.

- Funkce třídy `Store` pro vložení nových dat:
  - `set_category` Vloží novou kategorii prodejny.
  - `set_product` Vloží nový produkt.
  - `set_category_positions` Vloží případně aktualizuje pořadí kategorií.
  - `set_product_positions` Vloží případně aktualizuje pořadí produktů.
  - `set_style_from_presets` Vloží hodnoty zvoleného vzhledu formuláře.

- Funkce třídy `Store` pro aktualizaci dat:
  - `set_order_complete` Nastaví objednávku prodejny jako dokončenou.
  - `set_order_uncomplete` Nastaví objednávku prodejny mezi nové.
  - `update_store` Aktualizuje základní informace prodejny.
  - `update_store_content` Aktualizuje nastavení prodejny.

Třetí třídou je `StoreAuth`. Obsahuje funkce pro registraci nové prodejny, přihlášení do administrace a zapomenuté heslo.

- Popis funkcí třídy StoreAuth:
  - `get_session_id_store` Vrátí ID prodejny podle `authentication`, který je zasílán v hlavičce požadavku.
  - `login_with_email` Validuje přihlašovací údaje a vrací session přihlášení.
  - `create_login_session` Vytváří `session` pro přihlášení.
  - `signup` Vytváří nový účet prodejny.
  - `has_already_account` Kontroluje, jestli daná subdoména nebo e-mailová adresa již neexistuje.
  - `lost_password` Odesílá zákazníkovi e-mail s odkazem pro zapomenuté heslo.
  - `reset_password` Aktualizuje zapomenuté heslo.
  - `create_lost_password_session` Vytváří `session` pro nastavení nového hesla.

Čtvrtou třídou je `StoreStats`. Obsahuje funkce pro získání statistik prodejny. Celkově jsou zde implementovány funkce pro tržbu, celkový počet objednávek a zákazníků, nejčastějších zákazníku podle celkové útraty a nejprodávanější produkty podle počtu kusů.

- Popis funkcí třídy StoreStats:
  - `get_turnover_data` Vrátí tržby z databáze seskupené podle hodin, dnů, týdnů či měsíců.
  - `get_turnover_formated` Vrátí tržby formátovaně. Období, kdy nebyla žádná tržba vyplní hodnotou 0.
  - `get_total_orders` Vrátí celkový počet objednávek za dané období.
  - `get_total_customers` Vrátí celkový počet zákazníků za dané období.
  - `get_best_customers` Vrátí nejlepší zákazníky za dané období podle útraty.
  - `get_best_products` Vrátí nejlepší produkty za dané období podle počtu kusů.
  - `get_stats` Vrátí všechny zmíněné statistiky za dané období.

Pátou třídou je `Email`. V jejím konstruktoru je implementováno připojení k databáze pomocí knihovy `PHPMailer` přes protokol SMTP.

- Popis funkcí třídy Email:
  - `send_order_confirm` Odesílá e-mail zákazníhovi o přijetí objednávky. Vloží do kopie e-mail administrátora, pokud má prodejna aktivní notifikace.
  - `send_new_store_notification` Odesílá e-mail správci aplikace Sebou.cz o regisraci nové prodejny.
  - `store_send_for_check_notification` Odesílá e-mail správci aplikace Sebou.cz pro schválení nové prodejny.
  - `reset_password` Odesílá e-mail s odkazem pro obnovu hesla.

Šestou třídou je `Responses`. Ta obsahuje chybové stavy API. Do chybového stavu se dotaz dostane v momentě, kdy nastalo neočekávané chování. Tím může být chyba při validaci formuláře, chyba v přípojení k databázi, nesprávné přihlašovací údaje k přístupu do administrace, atd.

- Popis funkcí třídy Response:
  - `bad_request` Vrací chybu `400`. Jedná se o špatný formát dotazu. Například když chybí nějaká požadovaná hodnota.
  - `unauthorized` Vrací chybu `401`. Chyba, kdy je přijat dotaz, který potřebuje autorizaci.
  - `forbidden` Vrací chybu `403`. Daný uživatel nemá k tomuto dotazu potřebná práva.
  - `not_found` Vrací chybu `404`. Dotazovaný předmět nebyl nalezen.
  - `not_available` Vrací chybu `406`. Dotazovaný předmět není k dispozici.
  - `server_error` Vrací chybu `500`. Nastala chyba na serveru.

5.2.2 Koncové body API

Koncový bod (anglicky endpoint) je v tomto projektu URL cílové adresy. Na tuto adresu se dotazuje frontend aplikace. Každý koncový bod má na starost svou funkci. Například koncový bod `set_order` zpracuje novou objednávku. Všechny koncové body jsou implementovány v adresáři `requests`. API aplikace Sebou.cz obsahuje celkem 21 cílových bodů. Každý cílový bod obsahuje informace o tom, která data v dotazu očekává a jakou formou se mají validovat. K validaci slouží funkce PHP `filter_var`. V koncových bodech jsou inicializovány potřebné třídy APi pro vykonání dotazu.

- Jednotlivé koncové body API:
  - `get_admin_session` Vrací data potřebná pro administraci prodejny. Kotroluje session v hlavičce dotazu, zda je validní, podle kterého získá ID přihlášené prodejny.
  - `get_css Vrací` CSS pravidla aktuálně zvoleného stylu prodejny. Pravidla obsahují definice CSS proměnných, které jsou použité na frontendu.
  - `get_order` Vrací způsob převzetí a celkovou částku daného `ID` objednávky. Dotaz je využíván na děkovné stránce po dokončení objednávky.
  - `get_store_initial` Vrací data potřebná pro objednávkový formulář. Odpověď obsahuje produkty, kategorie a nastavení prodejny.
  - `login` Koncový bod pro přihlášení do administrace. Vrací `session hash` přihlášené prodejny.
  - `lostpassword` Koncový bod pro zaslání odkazu pro změnu hesla.
  - `remove_category` Smaže kategorii podle `ID`. Konroluje, zda má prodejna právo spravovat tuhle kategorii.
  - `remove_product` Smaže produkt podle `ID`. Konroluje, zda má prodejna právo spravovat tento produkt.
  - `resetpassword` Aktualizuje zapomenuté heslo. Kontroluje `session hash` pro zapomenutá hesla.
  - `set_category` Vkládá novou kategorii.
  - `set_category_positions` Nastavuje pořadí kategorií.
  - `set_order` Ukládá novou objednávku.
  - `set_order_complete` Nastavuje objednávku jako dokončenou.
  - `set_order_uncomplete` Nastavuje objednávku jako novou.
  - `set_product` Vkládá nový produkt prodejny.
  - `set_product_positions` Nastavuje pořadí produktů.
  - `set_store` Vytváří profil nové prodejny.
  - `set_store_content` Ukládá obsah a nastavení prodejny.
  - `set_store_defaults` Ukládá základní informace o prodejně.
  - `set_store_for_check` Zařadí prodejnu na seznam ke schválení správcem aplikace.
  - `set_store_preset_style` Nastaví vzhled prodejny.

5.2.3 E-maily

Jak je již v této práci dříve zmíněno, e-maily se odesílají pomocí knihovny PHPMailer. Tato knihovna se pomocí protokolu `SMTP` připojí k SMTP serveru. V tomto projektu je využitý server slovenské společnosti Websupport. Na serverech této společnosti je i celá aplikace Sebou.cz. Přes tento server následně aplikace odesílá e-maily doménového jména sebou.cz. K připojení k SMTP serveru jsou potřebné přihlašovací údaje, host a port serveru. Pomocí knihovny PHPMailer lze definovat předmět, samotná zpráva, příjemci, skryté kopie, adresa pro odpověď a jiné.

Kapitola 6

Správa aplikace

Tato kapitola dokumentuje správu aplikace z pohledu prodejny v administraci i z pohledu uživatele ve formuláři objednávky. Konkrétně popisuje možná nastavení prodejny a průchod objednávky zákazníkem prodejny.

6.1 Správa aplikace z pohledu prodejny

V této části je popsáno založení nové prodejny a správa administrace. Popisuje jednotlivá možná nastavení, správu kategorií a produktů v nabídce a výpis objednávek a statistik.

6.1.1 Založení prodejny v aplikaci Sebou.cz

Na vytvoření nové prodejny v aplikaci se majitel prodejny dostane z hlavní stránky. Je potřeba vyplnit formulář, který obsahuje název subdomény prodejny. Ten smí obsahovat pouze malá písmena, číslice případně pomlčku. Dále majitel vyplní e-mail, heslo a potvrzení hesla. Heslo se vyplňuje dvakrát pro ujištění, že uživatel neudělal při vyplnění chybu.

Po odeslání formuláře aplikace uživatele přenese do administrace, kde jsou potřeba vyplnit základní údaje o prodejně. Jedná se o název podniku, adresu a město pobočky a logo prodejny. Logo prodejny není povinné vkládat. Administrátor prodejny jej může vyplnit později. Vyplněním základních údajů je založení nové prodejny hotové.

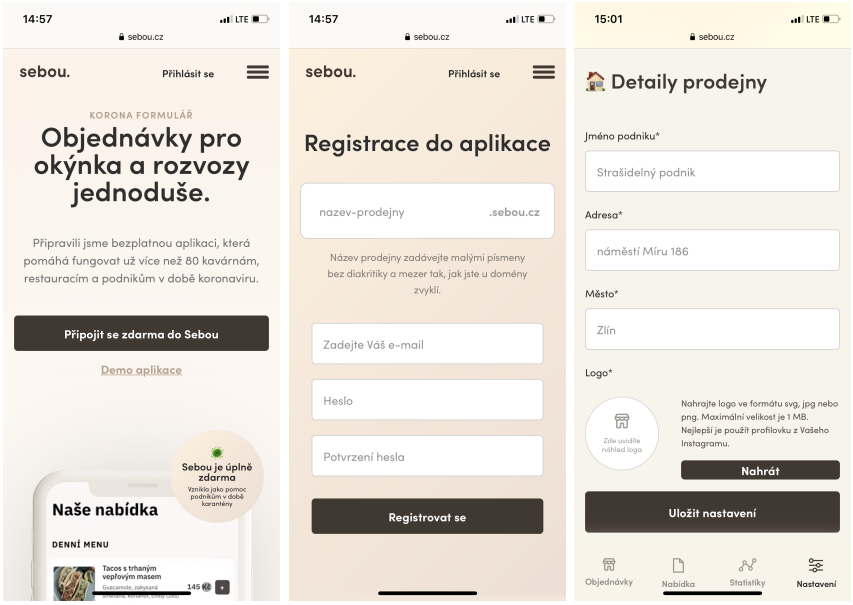
6.1.2 Nastavení rozvozů

Rozvozy lze spravovat v administraci webu pod záložkou menu „Nastavení“ dále „Rozvozy“. Pokud prodejna rozváží své produkty, je potřeba nejdříve rozvozy zapnout kliknutím na tlačítko „Povolit rozvoz“. Prodejna může zákazníkům povolit platby předem na účet kliknutím na „Umožnit platbu na účet předem“. Následně se zobrazí kolonka pro uvedení čísla bankovního účtu, na který budou zákazníci odesílat platby.

Dále si prodejna může natavit částku, od které rozváží objednávky. Od této částky bude možné dokončit objednávku, pokud si zákazník vybral předání touto formou. V případě, že zákazník nepřekročil uvedenou částku, zobrazí se mu upozornění. U rozvozů je možné určit cenu za dopravu, pokud je hodnota nastavena na nulu, zobrazí se zákazníkovi informace, že je doručení zdarma.

Následně si prodejna může zvolit texty s informacemi, které se zobrazí zákazníkovi. Zde si může určit další podmínky či upřesnění lokalit, do kterých objednávky rozváží. Všechna

zvolená nastavení jsou vidět v náhledu. Administrátor tak před uložením nových změn vidí dané změny z pohledu uživatele.



Obrázek 6.1: **Založení nové prodejny** – zleva: hlavní strana aplikace, formulář založení prodejny, formulář zákldních informací o prodejně

6.1.3 Nastavení výdejního okénka

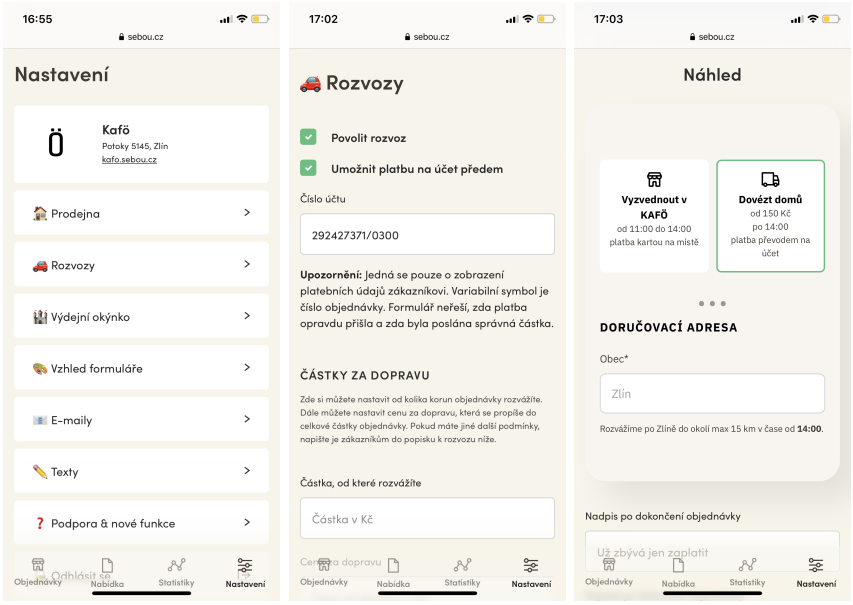
Výdejní okénko lze spravovat v administraci webu pod záložkou menu „Nastavení“ dále „Výdejní okýnko“. Pokud prodejna umožňuje předání výdejním okénkem, je potřeba nejdříve okénko zapnout kliknutím na tlačítko „Povolit okýnko“. Následně si prodejna může zvolit texty s informacemi, které se zobrazí zákazníkovi. Zde si může určit další podmínky vyzvednutí.

6.1.4 Nastavení vzhledu formuláře

Každá prodejna si může přizpůsobit vzhled objednávkového formuláře. Tím může formulář přiblížit používanému vizuálu prodejny. Vzhled formuláře lze změnit v administraci pod záložkou menu „Nastavení“ dále „Vzhled formuláře“. zde má administrátor na výběr z předvoleb. Z výběru lze zvolit konkrétní vzhled formuláře kliknutím na danou variantu. Následně se právě zvolený vzhled zobrazí v náhledu, který obsahuje objednávkový formulář. Kliknutím na tlačítko „Uložit styl“ se vybraná varianta aplikuje na objednávkový formulář prodejny.

6.1.5 Nastavení e-mailů

E-maily lze spravovat v administraci webu pod záložkou menu „Nastavení“ dále „E-maily“. Zde si může administrátor prodejny zapnout notifikace o nové objednávce. Jedná se o skrytou kopii zprávy e-mailu, která se odesílá zákazníkovi. Dále má možnost přidat do zprávy o nové objednávce vzkaz zákazníkovi. Vzkaz může obsahovat poděkování, otevírací dobu prodejny, či instrukce pro platbu.



Obrázek 6.2: **Nastavení aplikace** – zleva: rozcestník nastavení, nastavení rozvozů, náhled aktuální nastavení před jeho uložením

6.1.6 Nastavení textů ve formuláři

Texty ve formuláři lze spravovat v administraci webu pod záložkou menu „Nastavení“ dále „Texty“. Zde si prodejna nastaví úvodní nadpis a popis prodejny. Ten se uživateli zobrazí při prvním příchodu na objednávkový formulář. Dále se v tomto nastavení spravuje text vyprodané položky. Všechny texty se zobrazují v náhledu, aby administrátor viděl dané nastavení z pohledu zákazníka.

6.1.7 Správa objednávek

Správa objednávek je přístupná z menu administrace aplikace pod záložkou „Objednávky“. Nachází se zde filtrování objednávek. Podle zvolené filtrace se zobrazí jednotlivé objednávky. Každá objednávka obsahuje všechny informace, které se o objednávce ukládají. Objednávku je možné zařadit mezi dokončené, kliknutím na tlačítko vedle čísla objednávky. Při kliknutí na stejné tlačítko se objednávka zařadí zpátky mezi nové.

- Filtrování objednávek:
  - `Vše` Zobrazí všechny objednávky.
  - `Nové` Zobrazí objednávky, které nebyly označeny jako dokončené.
  - `Vyřízené` Zobrazí objednávky označené jako dokončené.
  - `Nové rozvoz` Zobrazí objednávky pro rozvoz, které nebyly označeny jako dokončené.

6.1.8 Správa nabídky

Správa objednávek je přístupná z menu administrace aplikace pod záložkou „Nabídka“. Nejdříve obsahuje možnost editace hlavičky, kde si lze zvolit název a popis nabídky prodejny.

Následuje přepínač mezi aktivní a neaktivní nabídkou. Na základě aktivního přepínače se zobrazí kategorie a produkty nabídky respektive možnost o přidání nových.

Nová kategorie se dá vytvořit po kliknutí na tlačítko „Přidat novou kategorii“. Zobrazí se formulář, kde se vyplní název nové kategorie. Ta lze následně upravit pomocí ikonky pera na pravé straně názvu kategorie. V úpravě je možné kategorii smazat. Při smazání se smažou i všechny její produkty. Administrátorovi je nejdříve zobrazena hláška s upozorněním, kterou je potřeba potvrdit.

Pro vytvoření nového produktu je potřeba nejdříve vytvořit kategorii, do které patří. Ve výpisu vytvořených kategorií se pak zobrazí tlačítko „Přidat novou položku“. Kliknutím na tlačítko se zobrazí formulář pro vytvoření nového produktu.

- Formulář položky obsahuje následující pole:
  - `Aktivní` Přepínač, zda je produkt aktivní v nabídce.
  - `Název` Povinné textové pole názvu položky.
  - `Cena` Povinné pole ceny položky za kus.
  - `Kusů na skladě` Povinné pole aktuálního počtu kusů položky na skladě. Pro nekonečno se vkládá hodnota -1.
  - `Fotka produktu` Nepovinné pole pro nahrání fotku položky do velikosti 10 MB ve formátech JPG nebo PNG.
  - `Popis produktu` Nepovinné pole popisu položky.
  - `Alergeny` Nepovinná zaškrtávací tlačítka pro určení obsažených alergenů položky.

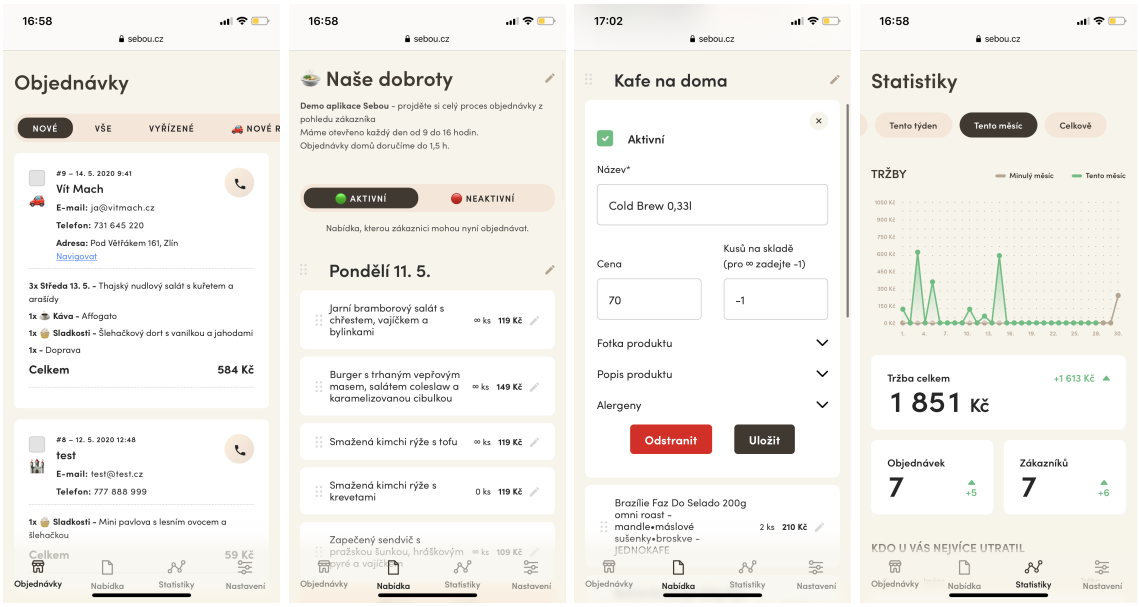
Produktům je možné měnit pořadí. Při stisknutí a následném držení šesti teček na levé straně produktu se můžou produkty mezi sebou přesouvat. Produkt je možné upravit kliknutím na ikonku pera v pravé části produktu. Editace produktu obsahuje formulář položky s aktuálními hodnotami. V editaci je taktéž možné produkt odstranit.

6.1.9 Odeslání ke schválení

Prodejna není přístupná veřejnosti, dokud prodejnu neschválí správce aplikace. Ten kontroluje, zda prodejna provedla nastavení správně. Pokud ne, zkontaktuje prodejnu a poradí jí, co je potřeba doladit. Ke schválení slouží tlačítko, které je přístupné z každé stránky administrace prodejny. Po schválení přijde správci aplikace notifikace, která jej upozorní o žádosti ke schválení. Pokud prodejna nastavila všechno správně, schválí prodejnu zpravidla do 24 hodin.

6.1.10 Statistiky

Statistiky jsou přístupné z menu administrace aplikace pod záložkou „Statistiky“. Ve vrchní části je výběr z předvolených časových období. Na základě zvoleného intervalu se zobrazí statistiky objednávek, produktů a zákazníků v daném časovém období.



Obrázek 6.3: **Administrace aplikace** – zleva: přijaté objednávky, správa nabídka, editace produktu, přehled statistik

6.2 Užívání aplikace z pohledu zákazníka

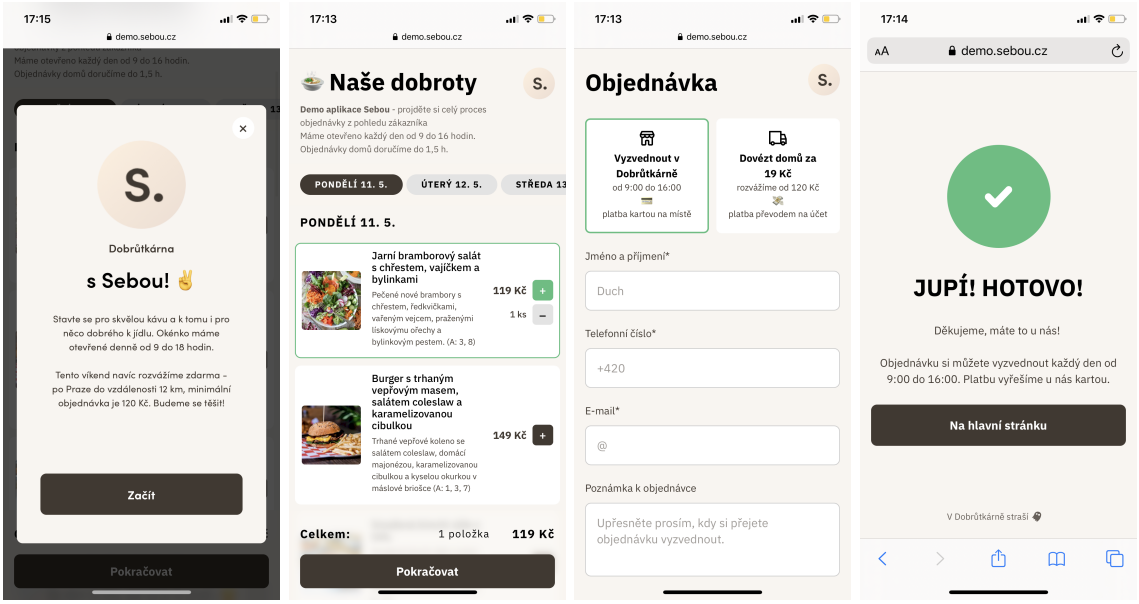
V této části práce je popsáno užívání aplikace z pohledu zákazníka. Jedná se o souhrn potřebných kroků k vytvoření nové objednávky. Uživateli se při příchodu zobrazí úvodní texty prodejny. Pokud se uživatel na formulář vrátil, úvodní obrazovka se uživateli již znovu nezobrazuje. Po přečtení úvodních textů se uživatel dostane na nabídku prodejny. Ta obsahuje hlavičku, menu kategorií a samotné produkty.

6.2.1 Přidání produktu do košíku

Jednotlivé produkty prodejny lze přidat do košíku kliknutím na ikonku „+“. Po přidání do košíku se uživateli zobrazí tlačítka pro zvýšení či snížení počtu kusů a celý produkt se zvýrazní. Pokud chce zákazník produkt odstranit z košíku, je potřeba mu pomocí tlačítka „-“ nastavit počet kusů na 0 a produktu zmizí zvýraznění. Po přidání produktů zákazník pomocí tlačítka „Pokračovat“ přechází na pokladnu.

6.2.2 Pokladna

V pokladně má zákazník na výběr z variant možného převzetí objednávky, které si prodejna nastavila. Pokud umožňuje vyzvednutí v okénku i rozvozy, uživatel si kliknutím na danou možnost vybere formu převzetí. Pokud má prodejna aktivní pouze jednu z možností, je ve výchozím stavu již zvolena. Následně uživatel vyplní osobní údaje a kliknutím na tlačítko „Odeslat objednávku“ dokončí objednávku.



Obrázek 6.4: **Objednávkový formulář** – zleva: úvodní modalové okno prodejny, nabídka prodejny, pokladna objednávky, děkovná stránka

Kapitola 7

Testování

Celý vývoj aplikace obnášel četné testování aktuálně implementovaných funkcí. V této kapitole se rozebírají nástroje a možnosti testování webových aplikací. Obsahuje způsoby testování frontendu pomocí DevTools v prohlížeči Chrome. Dále popisuje rozšíření prohlížeče pomocí plugin Vue.js devtools, testování API včetně ladění dotazů do databáze. Pro webové aplikace je dále stěžejní testování a funkčnost aplikace na co nejpoužívanějších webových prohlížečích. Jakmile byla implementace dokončena, bylo ji potřeba otestovat v reálném provozu. V následující části textu jsou všechny způsoby testování popsány.

7.1 Aplikace v simulovaném provozu

Při každém posunu v implementaci je potřeba novou funkčnost otestovat, zda funguje správně. Pro vývoj webových aplikací se využívá mnoho různých metod, díky kterým je můžeme odladit do produkčního provozu.

7.1.1 Testování a ladění frontendu

Výsledný frontend je zobrazován ve webovém prohlížeči. Ty nejpoužívanější z nich poskytují nástroje pro vývojáře. Díky nim může vývojář pomocí takzvaného inspektoru zjistit vše, co se v prohlížeči odehrává na pozadí a běžný uživatel nevidí. Základními funkcemi je zobrazení HTML a CSS pravidel jednotlivých elementů. Dále jsou zde vypsány všechny HTTP dotazy, které prohlížeč potřeboval k zobrazení konkrétní stránky aplikace. Pro testování a ladění Javascriptu nástroje nabízí javascriptovou konzoly včetně debuggeru. Tahle aplikace byla laděna pomocí nástrojů prohlížečů Chrome a Safari, které jsou nepoužívanějšími mobilními webovými prohlížeči na trhu.

7.1.2 Testování a ladění API

K testování API byla použita aplikace z internetového obchodu Chrome – Advanced REST client. Tato aplikace umožňuje pomocí uživatelského rozhraní vytvořit HTTP dotaz. Vytvořené dotazy pak směřovaly na koncové body API. Aplikace po dokončení dotazu zobrazí jeho celou odpověď. Můžeme tak otestovat správný formát dotazů, ale i chybný pro ověření, že API vrací správné chybové kódy.

7.1.3 Testování a ladění databáze

K testování databáze byl použitý nástroj phpMyAdmin, který umožňuje správu databáze pomocí webového rozhraní. PhpmyAdmin dále pomocí uživatelského prostředí zobrazuje všechny tabulky databáze a jejich data. V nástroji je možné psát SQL dotazy a zobrazit odpověď dotazu. Při propojení databáze s PHP můžeme pomocí funkce `debug()` knihovny Medoo získat požadovaný dotaz. Jelikož je Medoo objektově orientovaná knihovna pro práci s databází, můžeme si tím ověřit, zda má objektově sestavený SQL dotaz správný formát.

7.1.4 Testování v mobilním zařízení

Aplikaci je možné testovat souběžně při vývoji v mobilním zařízení. Je potřeba mít připojené zařízení k síti a na této síti spustit vývoj Nuxt.js aplikace. To je možné nastavením dané IP adresy lokální sítě do konfiguračního souboru `nuxt.config.js`. Následně je potřeba k této síti připojit i zařízení, na kterém budeme aplikaci testovat. V prohlížeči zařízení zadáním IP adresy a portu, který jsme uvedli v konfiguračním souboru, zobrazíme aplikaci.

7.2 Aplikace v reálném provozu

Tato kapitola popisuje způsob testování aplikace v reálném provozu. Při spuštění aplikace stále obsahovala chyby, které byly odhaleny až při přijetí prvních objednávek prodejny. Při psaní této části práce aplikace v reálném fungovala 49 dnů. Během uvedeného období si v aplikaci vytvořilo účet celkem 161 prodejen. Zákazníci prodejen vytvořili 5 385 objednávek obsahujících 14 239 prodaných položek. Díky aplikaci prodejny utržily za toto období celkem 1 892 805 Kč.

Z celkového počtu 161 vytvořených prodejen odeslalo svou prodejnu ke zveřejnění 101 z nich. Aplikaci tedy reálně používalo 62,7 % registrovaných prodejen. Prodejny se postupně registrovaly na základě referencí již přidaných. Zároveň se o aplikaci psalo také v médiích. Konkrétně na serverech iDnes.cz[1] a Mediář[2]. Aplikace se dostala do podvědomí veřejnosti i díky známému gastronomickému bloggerovi Lukášovi Hejlíkovi, který o aplikaci napsal článek na svém facebookovém profilu[3].

Před samotnou aplikací Sebou.cz existovala první verze aplikace. Ta obsahovala jen formulář a jednoduchou administraci, která byla dostupná vždy jen pro jednu prodejnu. Další prodejna tak měla stejný kód, ale svou databázi a hosting, na kterém aplikace fungovala. Pro každou prodejnu se tak musela aplikace zprovoznit zvlášť. Vznikl tedy nápad aplikace, kde si každá prodejna vytvoří svůj profil sama. Vznikla aplikace s názvem Sebou.cz. Zde byla potřeba upravit databázi, která bude fungovat pro všechny prodejny. Základ databáze zůstal stejný, ten obsahoval v první verzi, ve které se produkty upravovaly bez uživatelského rozhraní, pouze tři tabulky – objednávky a objednané položky a samotné produkty. Dále byla potřeba vytvořit administraci prodejny, kde si prodejna bude moct nastavit vše včetně vzhledu a textů bez pomoci vývojáře.

Další nové funkce a úpravy navrhovaly samotné prodejny v aplikaci Upvoty. Navržený nápad nové funkce pak získává hlasy a nápady s nejvíce hlasy se pak přednostně implementuje. Prodejny do aplikace přidaly za prvních 49 dnů fungování celkem devatenáct nápadů a devět z nich jich bylo již implementováno. Prodejny potřebovaly mít například možnost

[1] https://www.idnes.cz/zlin/zpravy/sebou-aplikace-zlin-creepy-studio-restaurace-kavarny-koronavirus.A200421_544241_zlin-zpravy_ras

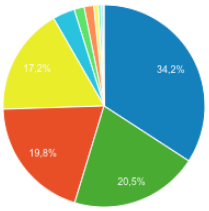
[2] https://www.mediar.cz/rozvoz-jidla-nove-nabizi-bolt-food-nabidky-vsech-rozvozu-sdruzuje-mealio/

[3] https://www.facebook.com/gastromapalh/posts/2235629646744335

přidat fotku a alergeny k produktům. Dále chtěly znát statistiky svých prodejů nebo připočítat cenu dopravy k celkové částce za objednávku.

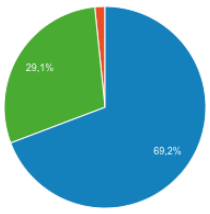
Aplikaci použilo celkem 22 020 uživatelů, kteří dohromady navštívili aplikaci celkem 47 365 krát. Nejpoužívanějšími prohlížeči byly s 40,33 všech uživatelů Safari a 34,16 % Chrome (Obrázek 7.1). K aplikaci přistupovali uživatelé nejčastěji pomocí mobilního zařízení, celkově 69,25 % uživatelů. Na druhém místě je počítač s 29,12 % a pomocí tabletů si aplikaci otevřelo 1,63 % uživatelů (Obrázek 7.2).

Prohlížeč	Uživatelé	Uživatelé	Příspěvek k celkovému výsledku: Uživatelé
	22 020 Podíl z celku v %: 100,00 % (22 019)	22 020 Podíl z celku v %: 100,00 % (22 019)	
1. Chrome	7 565	34,16 %	
2. Safari	4 548	20,53 %	
3. Safari (in-app)	4 385	19,80 %	
4. Android Webview	3 805	17,18 %	
5. Firefox	785	3,54 %	
6. Samsung Internet	355	1,60 %	
7. Edge	338	1,53 %	
8. Opera	158	0,71 %	
9. Internet Explorer	99	0,45 %	
10. Seznam	78	0,35 %	



Obrázek 7.1: Statistiky používaných prohlížečů z aplikace Google Analytics

Kategorie zařízení	Uživatelé	Uživatelé	Příspěvek k celkovému výsledku: Uživatelé
	22 019 Podíl z celku v %: 100,02 % (22 015)	22 019 Podíl z celku v %: 100,02 % (22 015)	
1. mobile	15 212	69,25 %	
2. desktop	6 398	29,12 %	
3. tablet	358	1,63 %	



Obrázek 7.2: Statistiky používaných zařízení z aplikace Google Analytics

Kapitola 8

Závěr

Cílem bakalářské práce bylo vytvořit webovou aplikaci pro výdejní okénka a rozvozy, která pomůže podnikům ovlivněným koronavirovou pandemií. Návrhu a implementaci předcházelo studium webových technologií a podobných řešení veřejného stravování na trhu. Výsledná aplikace umožňuje vytvoření objednávkového formuláře prodejnám, které otevřely výdejní okénka nebo poskytují rozvoz. Produkty a objednávky je možné spravovat z administrace prodejny. Z aktuální nabídky prodejny si zákazníci mohou objednávat.

Implementace aplikace byla napsána v javascriptovém frameworku Vue.js. K API aplikace je použitý jazyk PHP a databáze MySQL. Při tvorbě byl kladen důraz na bezpečnost a funkčnost celé aplikace. Řešení bylo odladěno simulovaným testováním pomocí dostupných nástrojů pro vývoj webových aplikací. Autorem designu finálního řešení je David Dvořáček. Komunikaci s prodejnami a zákazníky obstarávají zaměstnanci agentury Creepy Studio. Kompletní vývoj aplikace byl implementován autorem této práce.

Aplikace funguje v reálném provozu a pomohla celkem 101 prodejnám. O aplikaci se psalo v médiích a byla zmíněna v televizi. Za prvních 49 dnů v reálném provozu ji navštívilo celkem 22 020 uživatelů. Prodejny přes aplikaci utržily dohromady 1 892 805 Kč. Splnila tak nad očekávání autora svůj prvotní záměr.

Dalším rozšířením do budoucna by mohla být platba kartou přímo v aplikaci. Její zařízení je však časově náročná záležitost z důvodu fakturace, poplatku platebních bran a schválení společnostmi Visa a Mastercard. Dalším možným rozšířením je implementace kapacity prodejny. Pomocí kapacity by si mohly prodejny určit, kolik jsou schopné za určitý časový interval obsloužit objednávek. Tím by se předešlo situaci, kdy prodejna obdrží více objednávek se stejným časem vyzvednutí a před prodejnou se tak tvoří řada.

Literatura

[1] Adamcová, P. *Byznys s rozvozem jídla má zlaté časy. Třetinu jim dáme na provizi, naříkají podniky* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://zpravy.aktualne.cz/ekonomika/rozvozy-restaurace-koronavirus/r~4ddb5f5c891e11eab115ac1f6b220ee8/.

[2] Castro, E. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[3] Cloudforce. *Tajemství Cloudu Odhaleno* [online]. 2016 [cit. 2020-05-14]. Dostupné z: https://cloudforce.cz/tajemstvi-cloudu-odhaleno/.

[4] Andersen, B. *Vue Instance Lifecycle & Hooks* [online]. 2017 [cit. 2020-05-14]. Dostupné z: https://codingexplained.com/coding/front-end/vue-js/vue-instance-lifecycle-hook.

[5] Team, E. *Top 10 Popular CMS by Market Share (to Start a Website)* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://www.isitwp.com/popular-cms-market-share/.

[6] Janovský, D. *CSS styly - úvod* [online]. [cit. 2020-05-10]. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html.

[7] Macrae, C. *Vue.js: Up and Running : Building Accessible and Performant Web Apps*. Place of publication not identified: O'Reilly Media O'Reilly Media, 2018. ISBN 978-1-491-99724-6.

[8] *Sharding* [online]. [cit. 2020-05-12]. Dostupné z: https://docs.mongodb.com/manual/sharding/.

[9] *What are the best CSS preprocessors/postprocessors?* [online]. 2020 [cit. 2020-05-10]. Dostupné z: https://www.slant.co/topics/217/~best-css-preprocessors-postprocessors.

[10] Vue.js. *Reactivity in Depth* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://vuejs.org/v2/guide/reactivity.html.

[11] W3Counter. *Browser & Platform Market Share* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://www.w3counter.com/globalstats.php.

[12] Wappalyzer. *Databases* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://www.wappalyzer.com/technologies/databases.

[13] Wappalyzer. *Programming languages* [online]. 2020 [cit. 2020-05-14]. Dostupné z: https://www.wappalyzer.com/technologies/programming-languages.

[14] Williams, H. *Web database applications with PHP and MySQL*. Sebastopol: O'Reilly, 2004. ISBN 0-596-00543-1.

Příloha A

Obsah přiloženého paměťového média

- *bakalarska_prace.pdf* – soubor této bakalářské práce
- *bakalarska_prace.zip* – archiv se zdrojovými soubory této bakalářské práce
- *zdrojove_kody.zip* – archiv zdrojových kódů. Obsahuje adresáře `frontend`, `API` a soubor `db_structure.sql` se strukturou databáze
- *README.md* – soubor popisující obsah paměťového média