Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Rezervační systém

Bakalářská práce

Autor: Tomáš Zahradník
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.

Hradec Králové srpen 2022

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové, dne 14. 8. 2022 Tomáš Zahradník

Poděkování:

Děkuji vedoucím bakalářské práce Mgr. Daniela Ponce, Ph.D. za metodické vedení práce a poskytnuté rady a připomínky.

Anotace

Název: Rezervační systém

Bakalářská práce se zabývá analýzou, návrhem a následnou implementací rezervačního systému pro klientku, která v současné době pronajímá část svých podnikatelských prostor dalším podnikatelům s obdobnou podnikatelskou činností. Stanovení základních parametrů rezervačního systému probíhá formou diskuse s klientkou. V rámci rešerše běžně používaných českých komerčních rezervačních systémů jsou analyzovány jejich klady a zápory a konfrontovány s požadavky klientky. Následně je stanoven seznam požadavků na nový rezervační systém. Podle funkčních požadavků je pro rezervační systém sestrojen diagram případu užití.

V části práce věnující se implementaci jsou popsány postupy fungování jednotlivých částí systému. Během implementace rezervačního systému jsou již implementované požadavky na systém znovu konzultovány s klientkou a následně upravovány dle její zpětné vazby. Výsledný produkt je poté představen klientce, která je současně poučena, jak zastávat roli administrátora systému.

Annotation

Title: Reservation system

This bachelor’s thesis deals with the analysis, design and subsequent implementation of a reservation system for a client who currently rents part of her business premises to other entrepreneurs with similar business activities. Determining the basic parameters of the reservation system takes place in the form of a discussion with the client. As part of the research of commonly used Czech commercial reservation systems, their pros and cons are analyzed and confronted with the client's requirements. Subsequently, a list of requirements for the new reservation system is established. According to the functional requirements, a use case diagram is constructed for the reservation system.

In the part of the work devoted to implementation, the procedures for the functioning of individual parts of the system are described. During the implementation of the reservation system, the already implemented requirements

for the system are again consulted with the client and subsequently modified according to her feedback. The resulting product is then presented to the client, who is also instructed on how to assume the role of system administrator.

Obsah

Seznam obrázků

Seznam ukázek kódu

1 Úvod

Cílem této bakalářské práce je vytvořit funkční aplikaci rezervačního systému, který by klientka ráda využila ve svém podnikání. Klientka je v současné době majitelkou velkého Salonu, ve kterém provozuje kadeřnictví. V tomto Salonu dále pronajímá místnosti (studia) ostatním samostatným podnikatelům, kteří zde také odděleně nabízejí své služby veřejnosti, především v oblasti kosmetických, kadeřnických a holičských služeb, piercingu, manikúry a pedikúry, masáží apod.

Autor se v práci nejprve pokusí analyzovat požadavky na nový rezervační systém, který poskytne možnost spojení vytváření a správy rezervací všech podnikatelů v Salonu na jedno místo. V rámci rešerše běžně používaných komerčních rezervačních systémů českými firmami autor bakalářské práce popíše jejich klady a zápory a konfrontuje je s požadavky klientky, se kterými stávající rezervační systémy nejsou v souladu. Navržený systém rezervací vychází z požadavků klienta.

Pro uživatele rezervačního systému bude implementována registrace a přihlášení za použití jejich existujícího Google účtu. Google účet byl zvolen, protože podle průzkumu mezi zákazníky Salonu je to nejčastěji používaný účet. Rezervační systém proto nebude muset řešit problematiku šifrování a ukládání hesel, resetování hesel apod., protože tato zodpovědnost bude přesunuta na stranu Googlu. Rezervační systém zároveň po registraci zákazníka ihned obdrží osobní informace uživatele jako jméno, email apod. Z pohledu uživatele to znamená, že si nebude muset vymýšlet a pamatovat nové heslo a vyplňovat při registraci svoje osobní údaje.

2 Cíl práce a metodologie

Cílem práce je navrhnout a implementovat novou webovou aplikaci rezervačního systému, která bude funkční a přehledná pro všechny druhy uživatelů v daném Salonu. Přechod na rezervační systém zpřehlední všechny rezervace zákazníků tak, aby jejich správa umožnila zlepšit podnikatelům nabízené služby a také zlepšit komfort zákazníků při objednávání a přeobjednání poptávaných služeb. Cílem implementace nového rezervačního systému je snížit transakční náklady každého podnikatele v daném Salonu, a tím zvýšit jejich zisk, a také optimálně rozvrhnout časový průběh poskytování služeb, aby nedocházelo k časovým prodlevám. Díky rezervačnímu systému bude podnikatelská činnost více efektivní a zlepší se i přístup ke službám pro zákazníky. V neposlední řadě bude mít majitelka Salonu přehled o zákaznících vstupujících do její nemovitosti do jednotlivých studií.

Metody řešení práce vycházejí z cíle a logické struktury bakalářské práce: na základě analýzy potřeb klientky a průzkumu mezi stávajícími zákazníky Salonu je provedena prvotní analýza a rešerše stávajících řešení rezervačních systémů a jejich komparace. Následuje posouzení možnosti implementace již existujícího rezervačního systému pro potřeby klientky. Na základě diskuse s klientkou a analýzou činnosti a poskytovaných služeb je zvoleno nové řešení. Testování autorem práce vytvořeného rezervačního systému bude probíhat ve dvou fázích, první s majitelkou Salonu a poté s poskytovateli služeb ve studiích, druhá fáze bude zkušební provoz s přihlášenými i nepřihlášenými klienty.

3 Analýza a návrh

Následující kapitola se zabývá analýzou navrhovaného systému. Kapitola obsahuje seznam požadavků na systém, modely systému a vyhodnocení cílů.

3.1 Přehled existujících systémů

Klientka nejprve zvažovala využití existujícího řešení, tedy zakoupení existujícího rezervního systému. Porovnání existujících řešení bylo prováděno podle následujících kritérií. Kritéria jsou řazena sestupně od nejdůležitějšího podle subjektivního hodnocení klientky:

1. podpora mobilních zařízení
2. cena licence za měsíc
3. maximální počet rezervací
4. existence cizí reklamy
5. možnost integrace do vlastního webu

3.1.1 Reenio

Prvním posuzovaným systémem je Reenio. Údaje o cenách a poskytovaných službách a doplňující informace jsem čerpal z [1]. Reenio je rezervační systém české společnosti GARVIS Solutions s.r.o. Reenio kromě responzivity rezervačního formuláře nabízí i vlastní mobilní aplikaci pro mobilní zařízení zdarma ve všech nabízených balíčcích. Rezervační systém je nabízen ve třech různých balíčcích. Základní balíček „FREE“ je zdarma, další dva balíčky „BUSINESS“ a „PREMIUM“ vyžadují placenou licenci. Licenci lze zakoupit buď na 1, 6, nebo 12 měsíců, kdy licence na 12 měsíců nabízí nejvýhodnější cenovou nabídku, která v přepočtu na 1 měsíc činí 225 Kč bez DPH pro balíček „BUSINESS“ nebo 584 Kč bez DPH pro balíček „PREMIUM“. Balíček „FREE“ omezuje maximální počet vytvořených rezervací za měsíc na 50 rezervací, placené balíčky poté na 1.000 pro balíček „BUSINESS“ nebo 10.000 rezervací pro balíček „PREMIUM“. V případě balíčku „FREE“ se do maximálního počtu rezervací počítají i stornované rezervace. Po uplynutí prvního měsíce používání balíčku „FREE“ se na rezervační stránce může začít zobrazovat reklama, placené balíčky reklamu neobsahují. Pro kompletní vypnutí zobrazování

značky *reenio* ve formuláři je nutné mít nejdražší balíček „PREMIUM“. Reenio poskytuje možnost integrace rezervačního systému do vlastní webové prezentace pomocí HTML kódu, který je Reenio schopné vygenerovat [2].

3.1.2 SuperSaaS

SuperSaaS B.V. je holandská společnost, která provozuje a vyvíjí rezervační systém s názvem SuperSaaS. Údaje o cenách a poskytovaných službách a doplňující informace jsem čerpal z [3]. Systém plně podporuje zařízení od stolních počítačů, přes tablety, až po mobilní telefony. Licence rezervačního systému jsou nabízeny v mnoha balíčcích. Placené balíčky jsou označenými písmeny abecedy od A až po J a jejich cena se pohybuje v rozmezí od 150 Kč až do 2.850 Kč bez DPH měsíčně. SuperSaaS nabízí i balíček zdarma, ten je ale určený výhradně pro účely vyzkoušení a seznámení se se systémem. V případě využití balíčku zdarma pro komerční účely si SuperSaaS vyhrazuje možnost vyžádat navýšení licence na jeden z placených balíčků, nebo provozovaný účet omezit nebo úplně zrušit [4]. Balíček zdarma je tedy pro použití v Salonu nevhodný, a proto dále budou popisovány pouze placené balíčky. Maximální počet rezervací v nejlevnějším balíčku je stanoven na 100, v nejdražším pak na 15.000. Limit rezervací zde není nastaven na vytvořené rezervace za měsíc, ale omezuje počet všech budoucích rezervací. Existuje zde i limit na maximální počet již uplynulých rezervací, který se vždy rovná desetinásobku limitu budoucích rezervací. Žádný z placených balíčků neobsahuje reklamy. Integrace do vlastního webu je možná třemi způsoby: přesměrováním na rezervační rozvrh SuperSaaS, vložením widgetu, nebo vložením pomocí iframe [5].

3.1.3 Reservio

Reservio je produkt vlastněný a provozovaný českou společností Reservio, s.r.o. Údaje o cenách a poskytovaných službách a doplňující informace jsem čerpal z [6]. Rezervační systém Reservio je dostupný na všech zařízeních od počítačů po telefony. Reservio lze pořídit v jednom balíčku zdarma (balíček „Free“) a třech dalších placených balíčcích („Starter“, „Standard“ a „Pro“). Zakoupení licence lze buď na 6 měsíců, 1 rok nebo 2 roky. V případě nákupu na delší dobu se snižuje průměrná

cena za měsíc. Při nákupu na jeden rok vyjde cena placených balíčků na 184 Kč za „Starter“, 368 Kč za „Standard“ nebo 738 Kč za „Pro“ na jeden měsíc. Maximální počet rezervací za 30 dnů je pro balíček „Free“ 40, „Starter“ 200, „Standard“ 500 a nejdražší balíček „Pro“ limit na rezervace odstraňuje úplně. Všechny balíčky kromě balíčku „Pro“ obsahují reklamu v podobě loga Reservio. Pro použití Reservia na vlastní doméně je nutné vlastnit alespoň baliček „Standard“ a pro kompletní přístup k API systému je nutné vlastnit balíček „Pro“.

Klientka zvažovala využití existujícího řešení. Preferovala by ale verze zdarma, které jsou omezené nebo nepoužitelné pro komerční využití. Pokud by za rezervační systém platila, chtěla jej mít nastavený podle svých požadavků. Samozřejmě vlastní řešení nabízí více možností pro další rozvoj podnikání podle dalších možných budoucích požadavků. V případě nespokojení s fungováním již existujícího řešení by jakákoliv zásadní změna, jako přechod na jiný systém, byla velmi obtížná.

3.2 Aktuální řešení

V současné době klientka přijímá od zákazníků rezervace buď osobně v Salonu, nebo telefonicky a jenom během otevírací doby. Každý další provozovatel služeb ve studiích v Salonu si vlastní objednávky zákazníků eviduje sám, jak osobně uzná za vhodné. Nejčastěji formou zápisu do fyzického diáře.

3.3 Diskuse zadání s klientem

Dalším krokem analýzy byla diskuse s klientkou a její seznámení s možnostmi rezervačního systému podle jejích požadavků. Výsledkem diskuse byla nahrávka hlavních požadavků klientky na první funkční prototyp systému. Autor požadavky klientky shrnul do následujícího textu:

„Primárním cílem systému je umožnit zákazníkům Salonu vytvářet rezervace bez přímého kontaktu se nájemci prostor a zaměstnanci Salonu, kdekoliv ze svých vlastních zařízení. Pro zaměstnance Salonu by systém měl sloužit jako elektronický diář, kde si budou moci zobrazit přehled o nadcházejících i uplynulých rezervacích.

Systém umožní vytvoření nových účtů a následné přihlášení. Uživatelé systému budou rozlišováni pomocí následujících čtyř rolí:

- neregistrovaní zákazníci,

- registrovaní zákazníci,

- provozovatelé služeb

- administrátoři.

Neregistrovaný zákazník má možnost prohlížet všechny studia a jejich služby, které se v Salonu nachází. Pokud má zákazník o určitou službu zájem, může si pomocí formuláře vytvořit rezervaci. Zákazník, který se rozhodne registrovat se, může vykonávat stejné akce jako neregistrovaný zákazník, zároveň ale získá vlastní trvalý profil v systému. V profilu si registrovaný uživatel bude moci zobrazit seznam nadcházejících i uplynulých rezervací, které vytvořil. Nadcházející události bude moci do 24 hodin od rezervace upravit nebo stornovat.

Systém po vytvoření rezervace odešle na zákazníkovu emailovou adresu email o úspěšném vytvoření rezervace a její shrnutí.

Provozovatel služeb je typ registrovaného uživatele, který je nájemcem studia v Salonu a správcem rezervací svých klientů. Může prohlížet nadcházející a uplynulé události. Nadcházející události může také upravovat nebo stornovat. Může vytvářet dovolenou pro své studio.

Administrátor je speciální rolí, která je udělena pouze majiteli Salonu a správci nebo správcům systému. Administrátor má přístup ke správě všech objednávek ve všech studiích v Salonu ve stejném rozsahu jako jednotliví nájemci. Může také zobrazit všechny registrované uživatele, měnit jejich atributy.

Z hlediska uživatelského rozhraní bude systém přehledný, snadno navigovatelný a použitelný na mobilním zařízení. Hlavní stránka nabídne zákazníkovi možnost vybrat si mezi nabízenými studii v Salonu v podobě odkazu. Kliknutím na odkaz se zákazník přesune na formulář pro vytvoření objednávky. Jednotlivá studia bude také možno vybrat z hlavní navigace systému v horní části. Kromě odkazů na rezervační formuláře bude navigace obsahovat i výzvu přihlásit se nebo zaregistrovat nepřihlášeným uživatelům. Uživateli, který je přihlášen, bude místo této zprávy zobrazena možnost přejít na svůj profil, zobrazit objednávky a odhlásit se.“

3.4 Funkční požadavky

Části systému přístupné zákazníkovi, jako jsou vytvoření a správa rezervací, by měly být použitelné ze stolního počítače i telefonu. Pro provozovatele služeb bude systém nabízet pohled na současné rezervace jeho zákazníků.

Systém bude rozeznávat čtyři různé role oprávnění: nepřihlášený uživatel, přihlášený uživatel, poskytovatel služeb a administrátor. V systému bude možné vytvořit uživatelský účet. Systém umožní vytvoření účtu pomocí existujícího Google účtu.

Funkční požadavky pro neregistrovaného uživatele:

- Vytvoření nové rezervace
- Uživatel obdrží email po vytvoření, úpravě nebo stornování rezervace
- Vytvoření nového uživatelský účet

Funkční požadavky pro registrovaného uživatele:

- Vytvoření nové rezervace
- Uživatel obdrží email po vytvoření, úpravě nebo stornování rezervace
- Do 24 hodin od začátku rezervace možnost úpravy a stornování rezervace
- Zobrazení vlastního profilu
- Zobrazení vlastních nadcházejících i uplynulých rezervací

Funkční požadavky pro provozovatele služeb:

- Vytvoření nové rezervace
- Uživatel obdrží email po vytvoření, úpravě nebo stornování rezervace
- Vytvoření a správa dovolené
- Vytvoření a správa nabízených služeb studia
- Úprava a stornování všech budoucích rezervací do vlastního studia
- Zobrazení vlastního profilu

Funkční požadavky pro administrátora:

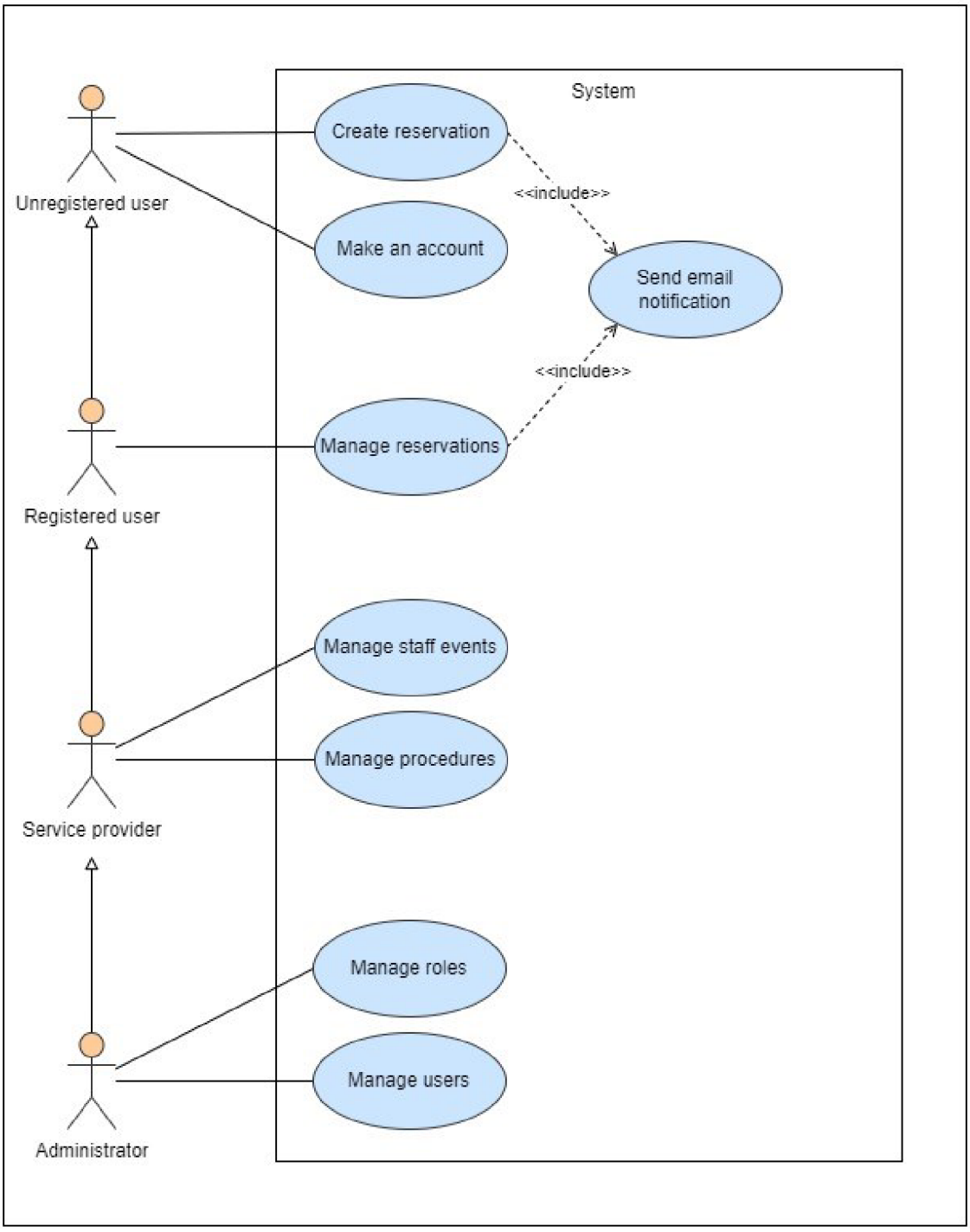
- Vytvoření nové rezervace
- Uživatel obdrží email po vytvoření, úpravě nebo stornování rezervace
- Správa studií nabízených v systému
- Správa administrací všech studií v Salonu
- Správa registrovaných uživatelů v systému
- Zobrazení vlastního profilu

3.5 Nefunkční požadavky

- Podpora mobilních zařízení
- Implementování uživatelského rozhraní pomocí knihovny React
- Použití frameworku Express pro NodeJS na backendu
- Použití NoSQL databáze

3.6 Diagram případu užití

Diagram případu užití byl vytvořen s ohledem na funkční požadavky na systém definované v kapitole 3.4 Funkční požadavky.



Obrázek 1: Diagram případu užití
Zdroj: Autor

4 Implementace

Následující kapitola se zabývá implementací systému navrhnutého v předchozí kapitole. Kapitola obsahuje popis finální implementace požadavků na systém. Lze ji také použít jako dokumentaci implementační části systému.

Při implementaci jsem využíval obecné postupy a informace popsané v následujících dokumentacích: webovou dokumentaci javascriptu od společnosti Mozilla [7], dokumentace knihovny mongoose [8], dokumentaci frameworku Express [9], dokumentaci knihovny React [10] a dokumentaci frameworku React Bootstrap [11].

4.1 Validace požadavků odeslaných do systému

Odesláním požadavku do systému z něj lze získat informace (např. seznam poskytovaných služeb), nebo v systému vyvolat akci (např. vytvoření objednávky). Aby se požadavek správně provedl a zamezilo se případnému zneužití systému, musí se vstupy požadavku validovat. V případě popisovaného systému je kladen zvýšený důraz na validace vstupů příkazů přístupným veřejnosti jako vytvoření objednávky apod.

Na straně uživatelského rozhraní systému je validace formuláře řešena pomocí atributu *required* a *type* na polích formuláře. Pomocí atributu *type* na polích, kde je očekáván vstup zákazníka, lze jednotlivé pole vymezit na specifickou hodnotu. Pokud tento atribut nastavíme např. na hodnotu *email*, internetový prohlížeč vyhodnotí pole jako validní pouze pokud bude vyplněná hodnota obsahovat znak zavináč. Ve formuláři je také použit typ *tel*, který dává prohlížeči najevo, že očekává telefonní číslo. Přesnou podobu tohoto čísla je ale nutné vyplnit do atributu *pattern* za pomocí regulárního výrazu. Atribut *required* pak zajišťuje, že pole jím opatřena musí být před odesláním formuláře vyplněna, jinak k odeslání nedojde. Tento atribut obsahují povinná pole formuláře. Protože se jedná o základní funkcionalitu nabízenou značkovacím jazykem HTML, jde o jednoduché univerzální řešení validací na uživatelském rozhraní a pro využití systému běžným uživatelem je dostačující.

Validace na uživatelském rozhraní je vhodná pro usnadnění používaní systému zákazníkům. Nelze na ní ale spoléhat jako na ochranu před pokusy o útoky na systém. Důvodem je vyhodnocování platnosti všech podmínek přímo v prohlížeči na osobním počítači zákazníka, kde lze tyto podmínky poměrně snadno upravit nebo odstranit úplně. Proto je nutné vstupy validovat i na straně serveru. Pro validaci na serveru je využit middleware express-validator [12]. Express-validator nabízí řadu předem definovaných validací, stejně tak jako možnost vytvořit si vlastní. Middleware také podporuje řetězení více různých validací jednoho atributu za sebou. Výsledek validací se kontroluje pomocí funkce *validationResult*, které je jako argument poskytnuta celá proměnná požadavku *req*. Validační funkce jako návratovou hodnotu vrátí objekt výsledku validací. Pomocí metody, které objekt výsledku obsahuje, se poté zjišťuje, zda obsahuje chybové hlášení o selhaných validacích, které lze případně přetransformovat na pole objektů. Jednotlivé objekty v poli pak obsahují jméno atributu, který neprošel validací, a také důvod selhání. Ze serveru je poté toto pole vráceno společně se zprávou o neplatném formuláři zpět na klienta.

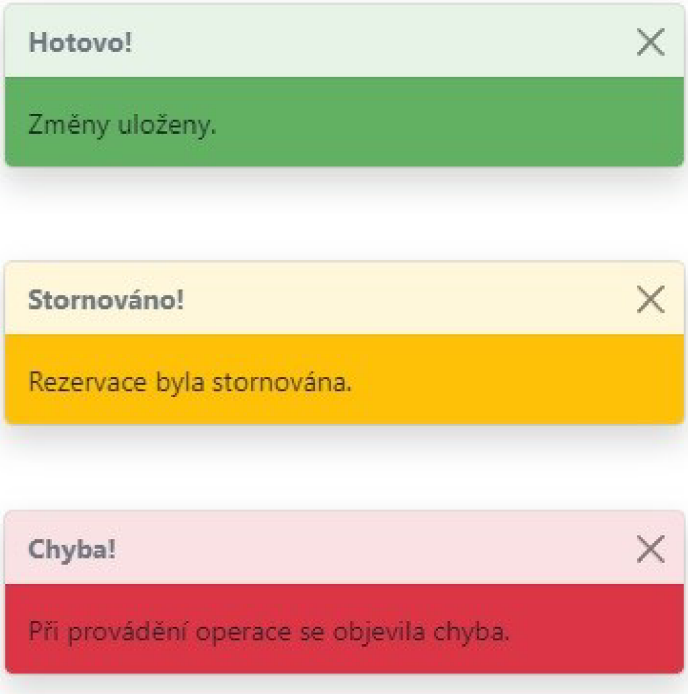
```
router.post(
  '/create-event',
  body('email').isEmail().normalizeEmail(),
  body('lastname').isLength({ min: 2 }),
  body('procedureId').isMongoId(),
  body('typeOfService').isString(),
  body('date').isISO8601(),
  body('phoneNumber').isMobilePhone('cs-CZ'),
  async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      console.error(errors.array());
      return res.status(500).json({message: 'Neplatný formulář.'});
    }
    // Algoritmus metody - vynecháno
  }
);
```

Ukázka kódu 1: Validace vstupů metody pro vytvoření rezervace
Zdroj: Autor

4.2 Zpětná vazba na uživatelském rozhraní

Během používání systému je v určité okamžiky vhodné poskytnout uživatelovi zpětnou vazbu o právě prováděné akci. Může se jednat např. o potvrzení úspěšného provedení akce, nebo informace o vyskytnutí chyby v systému. Uživatel dostane zpětnou vazbu o vyskytlé chybě objevením se takzvaného „*toastu*“ v podobě vyskakující zprávy ve spodní části obrazovky [13]. Implementace této zprávy se nachází v komponentě *ToastNotification* na uživatelském rozhraní.

V případě korektního provedení akce má zpráva zelenou barvu. Pokud akce proběhla úspěšně, ale jednalo se o akci destruktivního charakteru jako např. stornování rezervace, má zpráva oranžovou barvu. V případě, že se v systému objeví a zachytí chyba, je zpráva červeně zabarvena a obsahuje např. následující hlášení: „*Při provádění operace se objevila chyba.*“. O zachycené chybě je nutné poskytnou zpětnou vazbu i zákazníkovi. Není nutné zákazníka zatěžovat detailním popisem, proč k chybě došlo a vystavovat ho zbytečně technickým hlášením systému.



Obrázek 2: Zprávy zpětné vazby uživatelovi
Zdroj: Autor

4.3 Autentizace

Autentizace v systému obstarává middleware knihovna Passport pro NodeJS. [14]. Pro přihlášení i registraci je použita strategie využívající OAuth 2.0 API pro přihlášení pomocí účtu Google.

Výhodou tohoto řešení je jednoduchá registrace a přihlášení za využití zákazníkova Google účtu. Z pohledu implementace tento způsob autentizace odlehčuje systém o nutnost manipulování s hesly uživatelů. Systém se tak nemusí zabývat nároky na složitost uživatelských hesel, šifrování hesel v databázi, vytváření nového hesla v případě jeho zapomenutí uživatelem apod.

Zákazníkům tento způsob autentizace umožnuje registraci a přihlášení pomocí pár kliknutí za předpokladu, že jsou již přihlášeni do svého Google účtu. Při registraci nemusí zákazník vyplňovat svoje osobní jméno a vymýšlet si heslo.

Funkce *isAuthenticated* knihovny Passport nacházející se v objektu *req* je v systému obalena vlastním middleware modulem s totožným názvem isAuthenticated. Tento modul poskytuje funkci *isAuth*, která vyhodnocuje, zda je uživatel přihlášený. V případě, že přihlášený není, server terminuje vykonávání příkazu od neautorizovaného uživatele a vrátí odpověď se statusem 401 a zprávou *„You need to login.“*.

4.3.1 Registrace a přihlášení

Pro registraci nebo přihlášení do systému se zákazník musí přihlásit pomocí odkazu „*Přihlásit se pomocí Google*“ v uživatelském rozhraní do svého Google účtu a povolit sdílení svých osobních informací se systémem. Po úspěšném přihlášení předá Google serverové části systému následující informace o uživatelově profilu jako jméno, emailovou adresu apod. [15]. Z přijatých informací následně systém vytvoří nebo aktualizuje již existující záznam o zákazníkovi v databázi. Obě akce, vytvoření záznamu nebo jeho aktualizace, jsou prováděny funkcí *findOneAndUpdate* s volitelným parametrem *upsert* nastaveným na kladnou hodnotu, který zajišťuje vytvoření záznamu v případě, že nenajde odpovídající záznam, u kterého by mohl aktualizovat předložené hodnoty [16].

4.4 Autorizace

Autorizace na backendu je tvořena pomocí vlastního modulu s názvem *isAuthorized*, který exportuje tři funkce: *verifyRole*, *verifyAuthor* a *verifyRoleOrAuthor*. První funkce *verifyRole* vyhodnocuje, zda uživatel disponuje určitou rolí. Potřebnou roli a objekt uživatele na otestování přijímá funkce jako argumenty. Funkce *verifyAuthor* se využívá primárně při úpravě nebo mazání záznamů z databáze. Funkce vrátí kladný výsledek, pokud se ID uživatele shoduje s ID autora záznamu. Jako argumenty funkce přijímá objekt uživatele a ID autora záznamu. Poslední funkce *verifyRoleOrAuthor* je kombinací dvou předchozích, kdy alespoň jedna z nich musí mít kladný výsledek. Jako argumenty přijímá požadovanou roli, objekt uživatele a ID autora. Všechny tři funkce zároveň obsahují kontrolu, je-li uživatel administrátor. V takovém případě vrátí vždy kladný výsledek bez ohledu na ostatní klauzule.

Tyto funkce lze spolu kombinovat a kromě kompletního omezení přístupu je lze využít pouze pro částečnou úpravu výstupu. Například při načítání uživatelovy objednávky nejprve dojde k primární kontrole role nebo autora objednávky, při které může dojít k zabránění přístupu. Následná sekundární kontrola zabývající se pouze uživatelovou rolí už určuje pouze, zda z objektu objednávky odstranit poznámky poskytovatele služeb ve studiu, které by nebylo vhodné odeslat běžnému uživateli.

Na straně frontendu je autorizace řešena podobným způsobem jako na backendu, kontrolou role uživatele, který právě používá systém. Informace o uživatelovi se z backendu při úspěšném přihlášení zapíšou do proměnné pomocí hooku *setUser*. Proměnná *user* je následně používána při kontrolách autorizace.

Hlavním úkolem autorizace na straně frontendu je odstínění neprivilegovaného uživatele od částí systému na grafickém rozhraní, které by mohly vést k negativnímu výsledku autorizace na backendu. Například není žádoucí, aby se nepřihlášenému uživatelovi při návštěvě systému zobrazovala možnost úpravy poskytovaných služeb salonu. Uživatelům by se na grafickém rozhraní měl zobrazit vždy jen takový obsah, který je pro ně relevantní, a s chybami týkajících se nedostatečného oprávnění by neměli přijít nikdy do styku. V případě, že by se

uživatel pokusil navštívit stránku, ke které nemá přístup, např. pomocí zadání konkrétní url adresy administrace, dojde na místo vykreslení požadované stránky k vykreslení error stránky s příslušnou chybovou hláškou.

4.4.1 Vytvoření administrátorů

Před začátkem používání systému je vhodné předem určit alespoň jednoho uživatele, který dostane roli administrátora systému. Z prvotní analýzy systému lze vyvodit, že role administrátora nebude mezi uživateli nijak často přidělována ani odebírána. Dokonce by se dalo předpokládat, že systém bude s administrátory zvolenými při prvním uvedení do provozu existovat celou svoji dobu působení. Nejen z tohoto důvodu, ale i z důvodu bezpečnosti, nelze v systému vytvářet administrátory nikde v uživatelském rozhraní systému.

Před samotným vytvořením účtu uživatele s rolí administrátora je nutné jeho email zapsat do systémových proměnných serveru v souboru *.env*. Konkrétně do proměnné ADMIN_EMAILS. Proměnná může obsahovat více emailů oddělené čárkou, bez mezer. Dále je nutné si dát pozor, aby emaily byly vypsány pouze v jednom nezalomeném řádku. Systém poté při registraci uživatele, jehož email se nachází ve zmíněné systémové proměnné, rozpozná, že se jedná o administrátorský účet a nastaví jeho atribut *isAdmin* na *true*, zaručující administrátorská práva.

Povýšení již existujícího uživatele na administrátora je možné přidáním jeho emailu do souboru *.env* do proměnné ADMIN_EMAILS. Server je poté nutné restartovat. Uživatelský účet pak automaticky povýší do role administrátora při jeho dalším přihlášení do systému.

Popisovaný způsob vytváření administrátorů byl vytvořen s myšlenkou jednoduchého nastavení při prvotním spuštění aplikace, kdy je nutné pouze doplnit hodnoty do souboru. Zvažovaným alternativním řešením bylo vytvoření příkazů pro nahrání a úpravu systémové konfigurace, uložené v databázi. Tato alternativa nebyla zvolena jako finální řešení, kvůli předpokladu, že se role administrátorů nebudou měnit a funkcionalita by zůstala nevyužita.

4.5 Vytvoření objednávky

Následující subkapitola má za cíl blíže popsat stěžejní funkční požadavek na systém. Pokud si zákazník přeje vytvořit rezervaci v Salonu, může k tomu využít rezervační systém, díky kterému se může objednat i mimo pracovní dobu a z pohodlí svého domova. V systému si zákazník nejprve vybere studio, ve kterém si chce vytvořit objednávku. Výběrem studia se zákazník přesune na registrační formulář.

4.5.1 První část formuláře

Frontend

První částí formuláře se zabývá výběrem požadované služby. Službu zákazník vybere ze seznamu a výběr potvrdí kliknutím. Zákazník kromě názvu služby vidí, jak dlouho bude služba trvat a její cenu.

Pokud studio nabízí i doplňkové služby, zobrazí se rozbalovací lišta nadepsaná „*Doplňkové služby…*“. Po rozkliknutí se zobrazí nabízené doplňkové služby, které je možné kliknutím přidat do rezervace.

Backend

Služby vybraného studia jsou do formuláře načteny pomocí dvou požadavků dotazovací metodou GET na adresu „*procedure/get*“. V těle metody je pod atributem *typeOfService* specifikováno studio, pro které se služby načítají, a typ služeb pod atributem *type* s hodnotou *full* pro plnohodnotné služby a *additional* pro doplňkové služby.

4.5.2 Druhá část formuláře

Frontend

S výběrem služby čeká zákazníka vyplnění další části formuláře: datum a čas. Pro zajištění zobrazení korektních volných časů musí zákazník ke všem předchozím výběrům zvolit příslušný den. Ten zákazník vybere kliknutím do komponenty kalendáře ve formuláři. V horním panelu komponenty kalendáře se na levé straně nachází jméno právě zobrazovaného měsíce společně s aktuálním rokem. Na pravé straně panelu má zákazník možnost v kalendáři listovat mezi měsíci pomocí akčních

tlačítek, označenými šipkami doprava a doleva. V případě, že se zákazník prolistuje příliš daleko do minulosti nebo do budoucnosti, může se vrátit na současné datum pomocí tlačítka „*dnes*“ nalevo od šipek.

Po výběru dne se zákazníkovi zpřístupní možnost výběru času objednávky. Časy se zobrazí v rozbalovacím menu, ze kterého si zákazník může vybrat to, které mu vyhovuje. Pole výběru času zobrazuje volné časy v 15minutových intervalech. V závislosti na vybrané službě a začátku se do nepřístupného pole *Konec* automaticky dopočítá očekávaný konec vykonání služby. Vedle očekávaného konce objednávky se v nepřístupném poli nachází i hodnota očekávané ceny vykonání všech vybraných služeb.

V případě, že uživatel během vyplňování formuláře změní datum rezervace, službu nebo jednu z doplňkových služeb, dojde k vrácení pole výběru času do výchozí neutrální hodnoty z důvodu možné změny trvání rezervace.

Backend

Po výběru požadovaného datumu na komponentě kalendáře je na serverovou adresu *calendar/get-free-time* odeslána dotazovací metoda GET pro získání volných časů. Algoritmus získání volných časů nejprve z databáze získá všechny existující události pro vybraný den. Existující události mohou být objednávky ostatních zákazníků a dovolené zaměstnanců. Dále dojde k načtení všech celodenních událostí v rozmezí 5 dní před událostí a 5 dní po události. Celodenní události označují buď plně obsazené dny, nebo dny, kdy má provozovatel služeb nahlášenou dovolenou. Načtení celodenních událostí ve zmiňovaném rozmezí zajišťuje kontrolu, že se termín rezervace nenachází v souvislé dovolené trvající více dnů. Události pro vybraný den a celodenní události z rozmezí 5 dní před a po se následné spojí do jednoho pole.

Algoritmus následně v 15minutových intervalech prochází vybraný den a zjišťuje, zda se nepřekrývá s již existující událostí. Pokud ano, algoritmus se přesune na konec nalezené události a pokračuje dál v hledání. Pokud v právě prověřovaném čase neexistuje jiná událost a zároveň žádná další neexistuje ani v intervalu od prověřovaného času do prověřovaný čas + délka služby, čas se vyhodnotí jako volný

a zapíše se do pole jako *String* ve formátu hh:mm. Nemůže se tak stát, že by se objednávka jakkoli kryla s jinou událostí uloženou v databázi.

4.5.3 Třetí část formuláře

Frontend

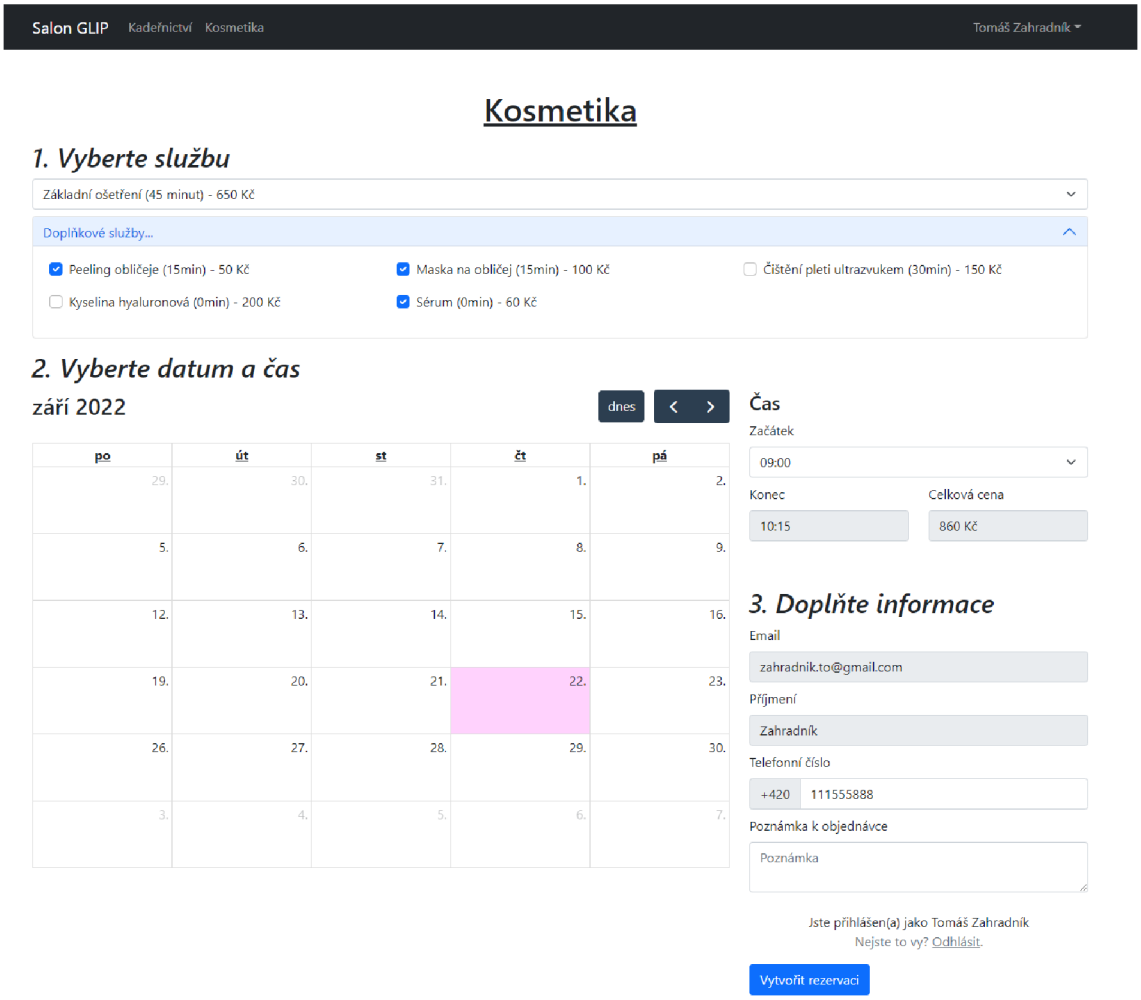
Poslední část formuláře žádá zákazníka o vyplnění osobních údajů. Jedná se o zákazníkův email, jeho příjmení a telefonní číslo. Nepovinným údajem je pole poznámky k objednávce, kde může zákazník poskytovateli služby sdělit dodatečné informace. V případě, že je zákazník zaregistrován a přihlášen, bude tato část formuláře již předvyplněna osobními informacemi uloženými v uživatelově profilu.

V případě, že zákazník chce objednávku vytvořit s odlišnými údaji než s těmi, které jsou spojeny s jeho účtem v systému, musí se odhlásit a provést objednávku s jiným vyhovujícím uživatelským účtem, nebo jako anonymní uživatel bez přihlášení.

Po vyplnění všech povinných políček je možné formulář odeslat kliknutím na tlačítko „*vytvořit rezervaci*“.

Backend

Server po obdržení požadavku metodou POST na adresu *calendar/create-event* nejprve ověří správnost vstupů v těle metody. Bližší popis validací se nachází v kapitole 4.1 *Validace požadavků odeslaných do systému*. V případě, že výsledek validace nebude chybný, objednávka se zapíše do databáze. Po zapsání objednávky algoritmus provede kontrolu zbývajícího volného času pro daný den voláním funkce *checkRemainingFreeTime*. Pokud žádný další volný čas neexistuje, dojde k vytvoření nové události v databázi v kolekci *staffEvents* s atributem *eventType* nastaveným na hodnotu *occupied* a celodenním příznakem *allDay* na kladnou hodnotu, označující plně zaplněný den. Dny označené tímto typem události jsou ve formuláři rezervace označeny zeleným podbarvením a není na ně možné vytvořit objednávku.



Obrázek 3: Rezervační formulář
Zdroj: Autor

4.6 Administrace

Následující subkapitola se zabývá popisem privilegované části systému – administrace.

4.6.1 Administrátor vs. Provozovatel služeb

Provozovatelům služeb se přístup do administrace zobrazuje jako položka v horním menu, která vede na stránku rozdělenou na další tři podstránky: přehled, dovolená a služby. Administrátorům systému se na stejném místě v horním menu zobrazuje další rozbalovací menu s přístupem do administrací všech studií Salonu,

odkaz na stránku určenou pro správu registrovaných uživatelů a odkaz na stránku pro správu studií Salonu.

Administrátorský účet je v databázi označen atributem *isAdmin* nastaveným na kladnou hodnotu a atribut *role* na hodnotu „*admin*“. Ostatní provozovatelé služeb mají atribut *isAdmin* na výchozí zápornou hodnotu a tribut *role* na příslušnou hodnotu podle typu jejich podnikání.

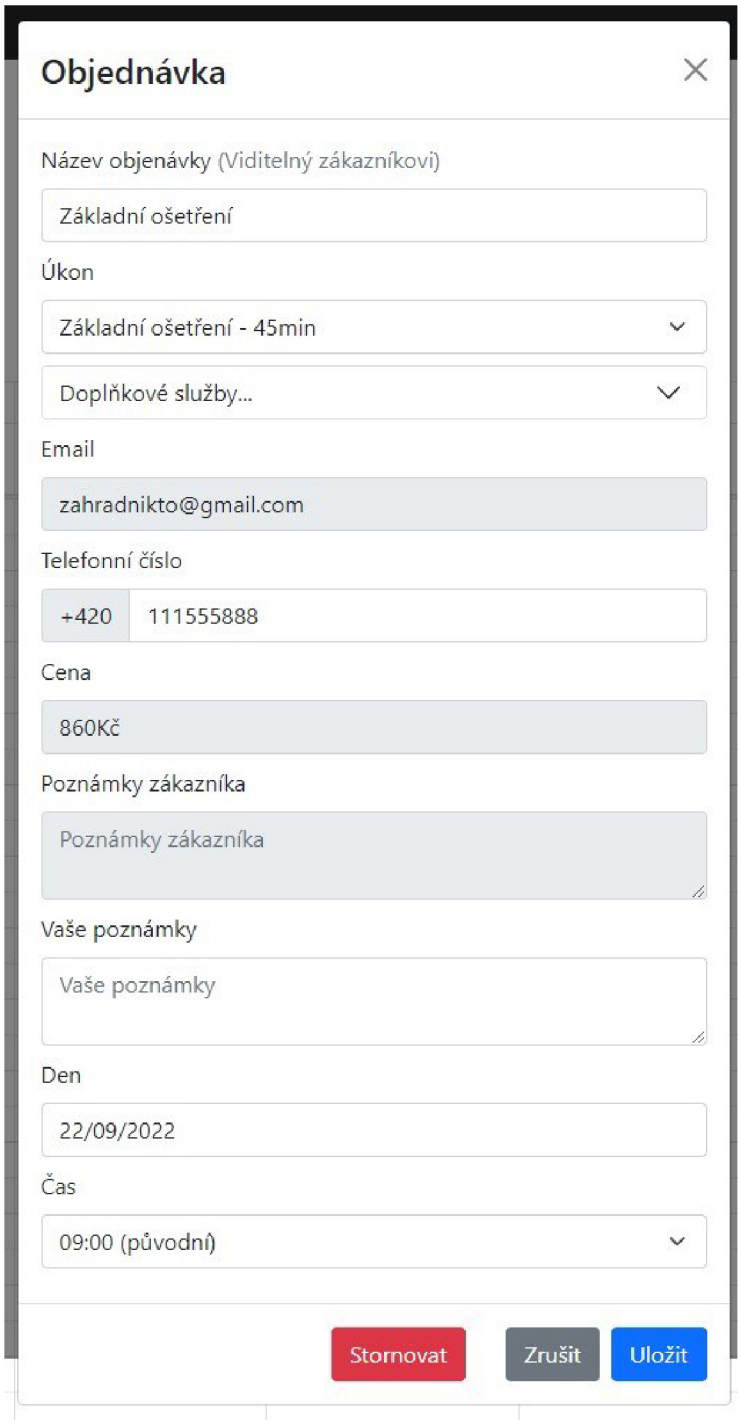
4.6.2 Administrátorské menu

Přehled

Přehled všech objednávek pro studio právě přihlášeného provozovatele služeb obsahuje komponentu kalendáře v zobrazení týdenní časové mřížky, která zobrazuje časovou osu v jednom pracovním týdnu. Na pravé straně ovládacího panelu komponenty lze kalendář přepnout do zobrazení celého měsíce.

Na jednotlivé události v kalendáři lze kliknout a zobrazit tak jejich detail a také možnost je upravit. Detail události se zobrazuje ve vlastním modal okně. K zobrazení modalu objednávek se využívá komponenta *EventModal* a k zobrazení modalu událostí provozovatelů služeb, jako např. dovolená, komponenta *StaffEventModal*.

Na rozdíl od uživatele, může provozovatel služeb ve studiu upravovat všechny budoucí objednávky bez limitace omezení úprav 24 hodin před úkonem. Zaměstnanec může měnit název objednávky, službu, telefonní číslo, vlastní poznámku k objednávce a den a čas objednávky. Možnosti výběru náhradního datumu a času se odvíjí od již existujících objednávek. Neměla by tak nastat situace, kde by si provozovatel služeb ve studiu přesunul objednávku na již obsazený termín.



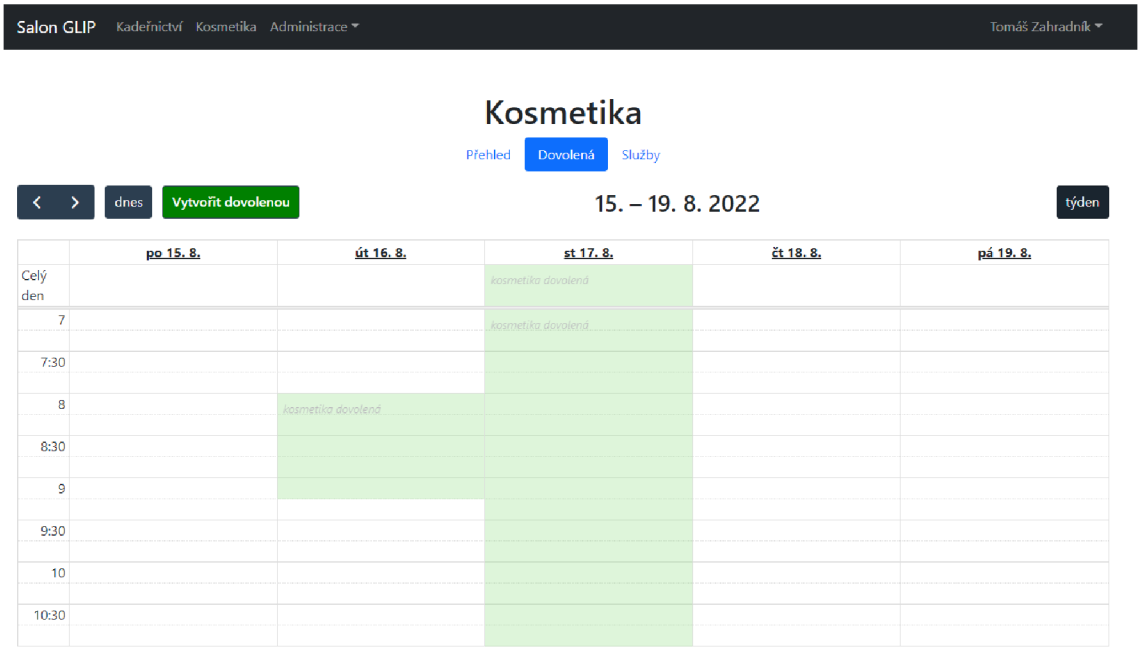
Obrázek 4: Komponenta EventModal
Zdroj: Autor

Dovolená

Stránka Dovolená obsahuje obdobnou komponentu kalendáře jako stránka Přehled. Kalendář je zde možné používat pouze v režimu zobrazení týdenní časové mřížky. Důvodem je nedostatek komponenty kalendáře, kdy události, které mají začátek a konec v rozdílných týdnech, nejsou zobrazovány v režimu zobrazení týdenní časové mřížky.

Kliknutím nebo tahem kurzoru na volné části kalendáře v budoucnosti lze vyznačit dovolenou. Dovolenou lze vytvořit buď v libovolné délce po patnácti minutových intervalech, nebo jako celodenní akci klikem na políčko označené v kalendáři jako „*Celý den*“. Vytvořit celodenní akci dovolené je možné pouze ve dnech, které neobsahují žádnou další událost. Potvrzení výběru a zapsání události do databáze se provede stisknutím tlačítka „*Vytvořit dovolenou*“ v horní části kalendáře.

Dovolená se v databázi uloží do kolekce *staffEvents* obdobným způsobem jako objednávka zákazníka a obsahuje totožné volání funkce *checkRemainingFreeTime* pro zajištění označení dne s vyčerpanými volnými termíny. Objekt dovolené je v databázi označen atributem *eventType* nastaveným na hodnotu *vacation*.

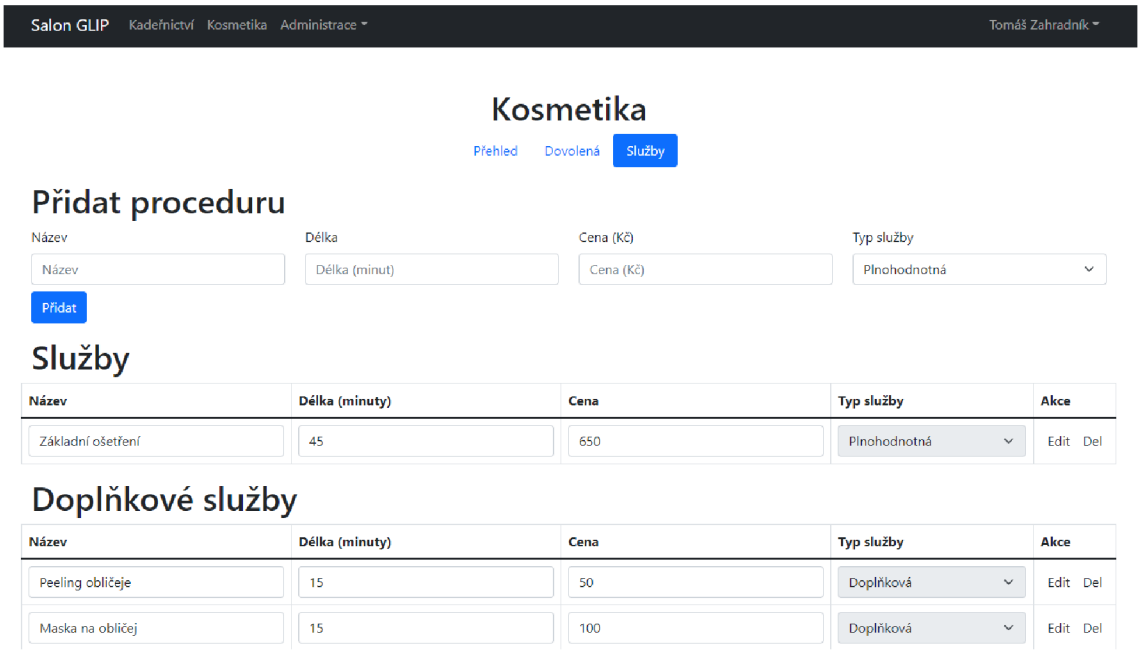


Obrázek 5: Stránka Dovolená v administraci
Zdroj: Autor

Služby

Stránka služeb obsahuje seznam všech existujících služeb pro dané studio. V horní části stránky se nachází formulář pro vytvoření nové služby. Pro vytvoření nové služby je nutné zadat její název, cenu, délku trvání, která musí být dělitelná patnácti minutami, a rozhodnout, zda služba má být plnohodnotná nebo doplňková. Doplňkové služby jsou v databázi označeny atributem *type* s hodnotou *additional*.

Služby, které již pro studio existují, jsou pod formulářem vypsané ve dvou tabulkách podle typu služby a poskytovatel služby má možnost je upravit nebo odstranit.

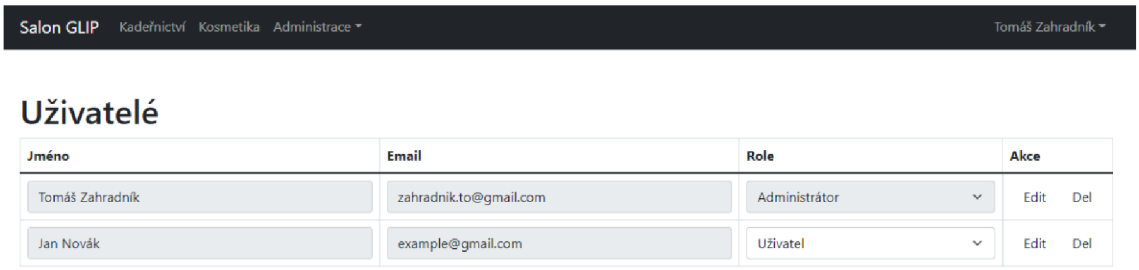


Obrázek 6: Stránka Služby v administraci
Zdroj: Autor

Uživatelé

Část administrace přístupná pouze administrátorům systému. Na této stránce se nachází tabulkový výpis všech registrovaných uživatelů systému. Údaje o uživatelích jsou zde rozděleny do sloupců jméno, email a role. Poslední sloupec obsahuje odkazy na dvě akce. Kliknutí na akci „*Edit*“ potvrdí úpravy dat uživatele a

uloží změny do databáze. Druhá akce s názvem „*Del*“ po kliknutí uživatelův účet z databáze odstraní.

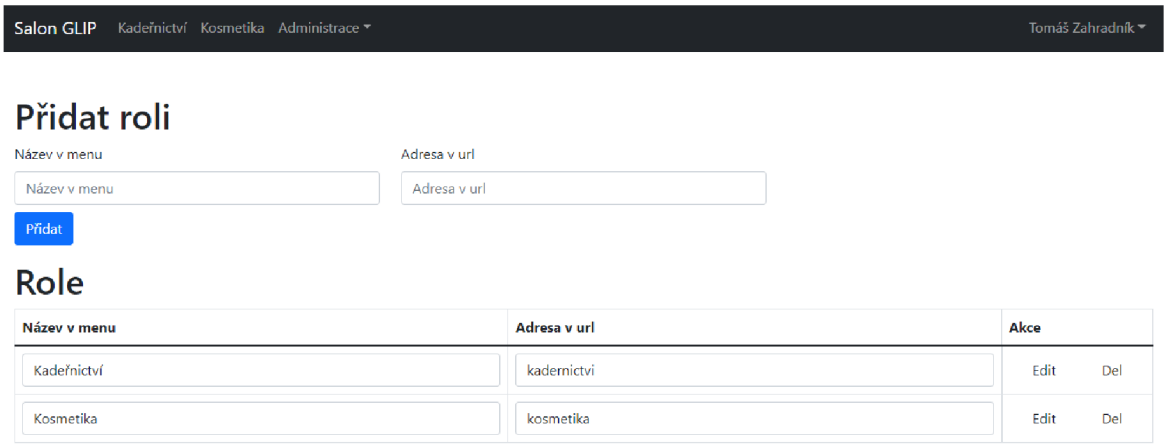


Obrázek 7: Stránka Uživatelé v administraci
Zdroj: Autor

Role

Část administrace přístupná pouze administrátorům systému. Na této stránce se nachází tabulkový výpis všech rolí systému, které v Salonu provozují služby. Pro vytvoření nové role, a tím pádem i studia, do systému, je nutné zadat název studia, který se zobrazí v horní navigaci systému, a zadat název který se bude zobrazovat v adresovém řádku studia. Název v adrese studia by neměl obsahovat žádné speciální znaky. Mezery v názvu adresy se na straně serveru automaticky převedou na znak podtržítka.

Role lze upravovat pomocí kliknutí na akci „Edit“ a mazat pomocí kliknutí na akci „Del“. Při smazání role dojde v systému ke stornování všech budoucích rezervací do daného studia, smazání všech událostí dovolené pro dané studio a všichni uživatelé s přiřazenou rolí studia obdrží roli běžného uživatele.



Obrázek 8: Stránka Role v administraci
Zdroj: Autor

4.7 Uživatelské menu

4.7.1 Uživatelský profil

Stránka uživatelského profilu slouží k zobrazení a úpravě dat uživatelského účtu. Uživatel zde má možnost změnit své telefonní číslo, které se předvyplní do formuláře pro vytvoření rezervace. Ostatní údaje není dovoleno upravovat z důvodu provázanosti účtu s uživatelovým účtem Google, který byl použit pro registraci.

Změna vlastní role

V případě, že právě přihlášený uživatel je administrátor, zobrazí se mezi osobními informacemi uživatele i rozbalovací pole pro změnu role. Tato funkčnost byla přidána pro účely testování a ukázky systémových funkcí bez nutnosti vytváření a přepínání mezi více uživateli. Pro ukázku systému z pohledu každé role by bylo nutné vlastnit minimálně tři různé Google účty, nebo měnit role v případě potřeby manuálně v databázi. Administrátor si může vlastní roli dočasně změnit na jakoukoliv další roli v systému. Změna role probíhá pouze úpravou attributu *role* na účtu administrátora a atribut *isAdmin* si ponechává svou původní hodnotu. Tím je docíleno, že uživatelské rozhraní vypadá tak, jako by ho viděl uživatel s danou rolí, a zároveň umožňuje stejným způsobem přepnutí zpět do role administrátora.

4.7.2 Moje rezervace

Stránka *Moje rezervace* dává přihlášenému uživatelovi možnost procházet a spravovat své objednávky. Výchozí zobrazení objednávek je v tabulce. Tabulka je tvořena následujícími sloupečky: den, začátek objednaného úkonu, konec úkonu, název úkonu a jako poslední se v tabulce nachází sloupec s akčním tlačítkem pro otevření modalového okna s kompletním detailem objednávky a možností úprav. Jeden řádek tabulky pak reprezentuje jednu uživatelovu objednávku. Pokud je daná objednávka stornovaná, a tím pádem neplatná, stále se ukazuje v tabulce, je však podbarvena žlutou barvou.

Tabulka umožňuje stránkování objednávek. Na jednu stránku tabulky se načte vždy maximálně deset různých objednávek. K navigaci na další stánky slouží akční tlačítka šipek a čísel umístěných pod tabulkou. V případě, že se požadované číslo stránky nezobrazuje ve výběru akčních tlačítek, může uživatel přejít na konkrétní stránku zadáním jejího čísla do pole označeného lupou a stisknutím tlačítka přejít pod akčními tlačítky stránkování.

Druhým způsobem zobrazení rezervací je zobrazení v režimu kalendáře s časovou osou. V tomto zobrazení jsou k dispozici pouze platné rezervace a stornované rezervace nejsou zobrazeny kvůli možnému vzájemnému krytí, a tím pádem snížení přehlednosti časové osy. Otevření detailového modalového okna dojde kliknutím na příslušnou událost v časové ose.

Pro načtení dat do tabulky se využívá dotaz metodou GET na serverovou adresu *calendar/get-events-page*. Do těla dotazu lze přidat parametr *page* s celočíselnou hodnotou a docílit tím načtení dat libovolné stránky. Pokud požadovaná stránka není specifikována, použije se výchozí hodnota 1. Na straně serveru se pro načtení stránkovaných dat z databáze využívá plugin pro knihovnu mongoose s názvem *mongoose-paginate-v2* [17]. Pro načtení dat do pohledu kalendáře se používá dotaz GET na serverovou cestu *calendar/get-events*, který má v těle definované parametry *start* a *end* označující začátek a konec měsíce, pro který se mají data načíst.

4.8 Odesílání emailů

Odesílání emailů je v systému řešeno pomocí modulu Nodemailer pro NodeJS [18]. Nodemailer je v systému nakonfigurován takovým způsobem, aby využíval vlastní SMTP server, vytvořený pomocným testovacím nástrojem MailHog.

V kódu je Nodemailer obalen vlastní Javascript třídou s názvem *GlipMailer*, pomocí které se emaily odesílají. Třída nabízí metodu sendMail, která jako parametry přijímá emailovou adresu adresáta, předmět a samotný text emailu. Vygenerování textu mailu obstarává vlastní pomocný modul *glipMailerHelper*. Pomocný modul exportuje objekt, který obsahuje konstanty předmětů emailových zpráv a funkci na vygenerování textu emailu. Funkce na vygenerování textu emailu přijímá jako argument objekt rezervace.

```
const glipMailHelper = {
  reservation: {
    create: {
      subject: 'Potrvrzení rezervace',
      message: (reservation) =>
        getReservationCreateMessage(reservation),
    },
    update: {
      subject: 'Úprava rezervace',
      message: (reservation) =>
        getReservationUpdateMessage(reservation),
    },
    cancel: {
      subject: 'Stornování rezervace',
      message: (reservation) =>
        getReservationCancelMessage(reservation),
    },
  },
};
```

Ukázka kódu 2: Exportovaný objekt modulu glipMailHelper
Zdroj: Autor

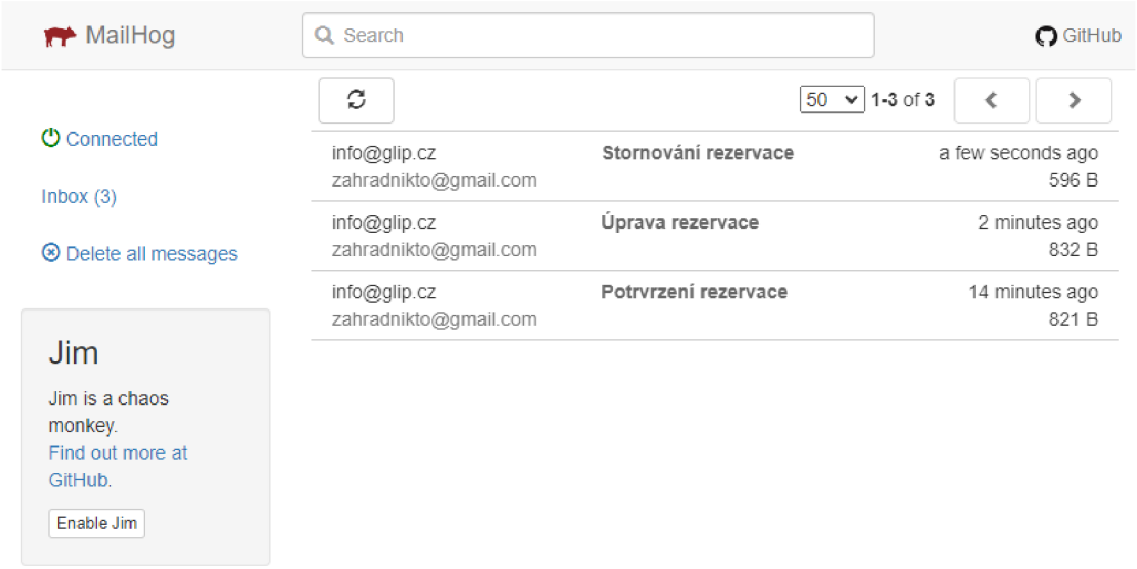
```
await GlipMailer.sendEmail(
  event.email,
  glipMailHelper.reservation.create.subject,
```

Ukázka kódu 3: Odeslání emailu
Zdroj: Autor

4.8.1 MailHog

MailHog je nástroj pro testování odesílání emailů. Nástroj obsahuje vlastní SMTP server, který zachytí odeslané emaily a zobrazí je ve webovém uživatelském rozhraní [19]. Mailhog je v systému využíván ve formě Docker kontejneru.

Výchozí port vytvořeného SMTP serveru je 1025. Tento port byl využit při konfiguraci modulu Nodemailer z předešlé kapitoly. Po spuštění docker kontejneru s nástrojem Mailhog, je možné přejít na jeho webové uživatelské rozhraní na adrese: http://localhost:8025/.



Obrázek 9: Uživatelské rozhraní nástroje MailHog
Zdroj: Autor s použitím webového rozhraní nástroje MailHog, [19]

5 Testování

První fáze testování systému probíhalo formou konzultací s majitelkou Salonu, kde jí byl systém představen, společně s předvedením všech požadavků na systém. Do systému byly následovně zapracovány nové požadavky či byly upraveny stávající podle dalších pokynů majitelky Salonu. Při testování byly nalezeny grafické odlišnosti v uživatelském rozhraní napříč různými prohlížeči.

Druhá fáze testování proběhne po sjednocení vzhledu uživatelského rozhraní mezi nejpoužívanějšími prohlížeči. Druhá fáze testování bude spočívat v představení a předvedení funkcí systému ostatním poskytovatelům služeb v Salonu. Těm bude následně po zaškolení do systému umožněn přístup. Systém se bude nadále vyvíjet podle obdržené zpětné vazby.

6 Spuštění rezervačního systému

Pro stažení potřebných závislostí systému je potřeba mít na počítači nainstalovaný runtime NodeJS a správce balíčků npm. Podmínkou spuštění potřebných Docker kontejnerů je nainstalovaný program Docker [20] s pluginem Compose [21].

Návod spuštění rezervačního systému je popisován pro operační systémy Windows 10 s nainstalovanými veškerými potřebnými programy. Postup pro ostatní operační systémy se může mírně lišit.

Pro nejlepší zkušenost s uživatelským rozhraní rezervačního systému je doporučené použít webový prohlížeč Google Chrome.

Stažení projektu

Pro stažení aktuální verze systému lze použít verzovací systém git nebo stáhnout jako zip přímo z webu Github. Systém je rozdělen na dvě části – server a klient, neboli uživatelské rozhraní. Pro optimální zkušenost s užíváním systému je nutné stáhnout obě časti.

Server: https://github.com/zahradnik-to/glip-server
Uživatelské rozhraní: https://github.com/zahradnik-to/glip-client

Instalace závislostí systému

Po úspěšném stažení obou částí systému je potřeba provést dodatečnou instalaci všech potřebných závislostí, které systém potřebuje ke svému fungování. Postup instalace je stejný na straně uživatelského rozhraní i serveru a vypadá následovně:

1. Otevřete příkazovou řádku v kořenové složce jedné z částí systému (uživatelské rozhraní nebo server)
2. Do příkazové řádky napište příkaz *npm install* a stisknutím enteru příkaz spusťte
3. Vyčkejte, než se stáhnou potřebné závislosti
4. Opakujte pro druhou část systému

Vytvoření souboru s proměnnými systému

V kořenové složce serveru je nyní nutné vytvořit soubor s proměnnými systému. Tento soubor není z bezpečnostních důvodů součástí staženého kódu z veřejného repozitáře na githubu. Soubor neobsahuje jméno, skládá se pouze z tečky přípony *env*. Soubor by měl po vytvoření vypadat následovné „.env“. Soubor je poté nutné otevřít v textovém editoru a vložit následující text:

```
APP_PORT=5000
MONGO_DB=mongodb://127.0.0.1/glip
SECRET=UHK_FIM_2021_8FEC4A09841BF9B658E07169FF044BBD2159420E4C5E20858DD
GOOGLE_CLIENT_ID=(hodnota poskytnuta emailem vedoucí práce a oponentovi)
GOOGLE_CLIENT_SECRET=(hodnota poskytnuta emailem vedoucí práce a oponentovi)
ADMIN_EMAILS=example@gmail.com,example2@gmail.com
GLIP_EMAIL=info@glip.cz
```

V souboru proměnných systému je důležité, aby každá proměnná byla na svém vlastním nezalomeném řádku. Proměnná Google client secret a Google client ID je z bezpečnostních důvodů poskytnuta pouze vedoucí práce a oponentovi, případně na vyzvání bude sdělena členům komise při státní závěrečné zkoušce a je sdílená výhradně za účelem testování rezervačního systému popisovaného v této práci a bude změněna po obhajobě práce.

Do proměnné ADMIN_EMAILS je nyní vhodné vložit email vlastního Google účtu, který bude používán při prohlížení systému.

Spuštění Docker-compose

Kontejnery obsahující dodatečné programy potřebné k běhu systému lze nainstalovat díky využití funkce Docker-compose. Složení kontejnerů docílíme navigací do složky *server/docker/glip* se souborem *docker-compose.yml* v příkazové

řádce. V příkazové řádce pak spustíme příkaz *docker-compose up*. Tímto příkazem dojde ke stažení a spuštění kontejnerů. Příkaz *docker-compose up* se sám od sebe neukončí ani po stažení a úspěšném spuštění kontejnerů, mylně se tak může zdát, že se zacyklil nebo zasekl. K ukončení tohoto příkazu je třeba použít klávesové zkratky zrušení právě prováděného příkazu příkazové řádky stisknutím tlačítek Ctrl+C. Pro opětovné spuštění kontejnerů není potřeba je znovu skládat příkazem *docker-compose up*, vystačíme si tedy s příkazem *docker-compose start* znovu ve složce serveru obsahující soubor *docker-compose.yml*. Vypnutí kontejnerů lze dosáhnout spuštěním příkazu *docker-compose stop* v příkazové řádce ve stejné složce

Spuštění systému

Po dokončení všech předchozích kroků je nyní systém připravený ke spuštění. Pro spuštění systému autor doporučuje využít dvou instancí příkazové řádky, v jedné se následně přesunout do kořenové složky serveru a ve druhé do kořenové složky uživatelského rozhraní. V obou příkazových řádkách nyní spustíme příkaz *npm start*. Na pořadí startu částí nesejde, nicméně před používáním systému je nutné vyčkat na dokončení spuštění obou částí.

Průběh spouštění lze sledovat v otevřených příkazových řádcích. Spuštěný server lze poznat podle následujících zpráv:

- Express listening on http://localhost:5000
- Database ready!

Po úspěšném spuštění uživatelského rozhraní by se mělo v internetovém prohlížeči otevřít nové okno na adrese http://localhost:3000. V případě, že obě situace již nastaly, je nyní systém připraven k použití.

Používání systému

Po úspěšném spuštění systému je nyní vhodné přihlásit se do uživatelského účtu s rolí administrátora. Následně je možné v administraci vytvořit novou uživatelskou roli, a tím i konkrétní studio v Salonu. V administraci studia je pak

možné vytvořit nabízené služby. Po vytvoření služeb je systém připraven na vytvoření první rezervace.

7 Pojmosloví

Pro uživatelský komfort a přehlednost práce obsahuje následující kapitola znalosti a pojmy potřebné k pochopení praktické části práce.

7.1 Informační systém

Podle slovníku pojmů na webu sprava-site.cz lze informační systém definovat následovně:

„Informační systém (z anglického Information System – IS) se dá zřejmě jednoduše definovat jako softwarové firemní vybavení, které podle přijatých informací pomáhá řídit jednotlivé interní procesy a dál předává informace lidským pracovníkům pro efektivnější kontrolu a veškerou další pracovní činnost.“ [22]

7.2 Definování požadavků na systém

Jim Arlow ve své knize UML 2 a unifikovaný proces vývoje aplikací k definování požadavků píše následující:

„Požadavek lze definovat jako specifikaci toho, co by mělo být implementováno. V podstatě rozlišujeme dva typy požadavků:

- *Funkční požadavky (functional requirements), jež určují, jaké chování bude systém nabízet*
- *Nefunkční požadavky (non-functional requirements), které specifikují vlastnosti nebo omezující podmínky daného systému.“* [23, s. 78]

7.3 Diagram případu užití

James Rumbaugh ve své knize „The Unified Modeling Language Reference Manual“ popisuje diagram případu užití jako: *„Graf aktérů, případů užití nacházejících se v hranicích systému (obdélník), vztahy mezi aktéry a případy užití, vztahy mezi případy užití a generalizace mezi aktéry. Diagram případů užití zobrazuje elementy z modelů případů užití (případy užití a aktéry).“* [24, s. 494], (překlad autora).

7.4 Autorizace a autentizace

Knopová ve své práci „Bezpečnost dat v informačních systémech“ popisuje rozdíl mezi autentizací a autorizací následovně:

„Autorizace je proces, při němž se ověřují (server či jiná entita) dostatečná práva pro přístup do určité oblasti pro vykonání akce. Dalším problémem je autentizace dat, které jsou ochranou jejich integrity, která spočívá v tom, že zabráníme neautorizované nebo nepatřičné modifikaci dat jiným než autorizovaným subjektům. A tedy tím zabráníme např. modifikaci stavu na účtech jiným než autorizovaným subjektům.“ [25, s. 24]

8 Shrnutí výsledků

V rámci bakalářské práce vznikl rezervační systém podle požadavků klientky, stanovené při prvotní analýze systému.

Funkční požadavky pro neregistrované uživatele byly splněny. Neregistrovaný uživatel může vytvořit objednávku výběrem služby, termínu a zadáním svých osobních údajů. Může se v systému zaregistrovat pomocí svého Google účtu.

Funkční požadavky pro registrované uživatele byly splněny. Uživatel může po přihlášení vytvořit novou rezervaci výběrem služby a termínu. Uživatelovo příjmení, email, a pokud ho již někdy zadával, telefonní číslo jsou v rezervačním formuláři předvyplněné. Registrovaný uživatel může na stránce *Moje objednávky* prohlížet všechny své objednávky a ty budoucí upravovat. Na žádost klientky je uživatelům umožněno upravovat pouze telefonní číslo objednávky a pro další úpravy je nutné vytvořit novou objednávku nebo kontaktovat personál studia. Registrovaný uživatel si může zobrazit svůj profil a upravit telefonní číslo, které bude předvyplněné v rezervačním formuláři. Úpravy a storno objednávek jsou registrovaným uživatelům umožněny maximálně 24 hodin před začátkem rezervované služby.

Funkční požadavky pro provozovatele služeb byly splněny. Provozovatel služeb může vytvořit novou rezervaci, buď pro vlastní účely, nebo jako zaevidování rezervace do svého studia, která přišla např. po telefonu nebo při osobní návštěvě klienta. Osobní informace zákazníka lze v takovém případě doplnit do poznámek objednávky. Provozovatel služeb může vytvářet plnohodnotné a doplňkové služby do vlastního studia. Provozovatel služeb může zobrazovat všechny nadcházející i předešlé rezervace ve svém studiu. Ve svém studiu může provozovatel služeb měnit nebo stornovat budoucí rezervace služeb, včetně těch, které začínají za méně než 24 hodin. Provozovatel služeb může vytvořit ve svém studiu dovolenou, tedy období, kdy služby nejsou systémem nabízeny k rezervaci. Provozovatel služeb si může zobrazit a měnit svůj profil.

Všichni uživatelé systému obdrží na email specifikovaný u jejich rezervace informace o vytvoření, úpravě nebo stornování rezervace.

Funkční požadavky pro administrátory systému byly splněny. Administrátor může vytvářet nové rezervace. Administrátor má přístup do administrací všech studií v Salonu, kde může provádět stejné akce, jako by byl provozovatelem služeb v daném studiu. Administrátor může spravovat registrované uživatele v systému. Administrátor může spravovat, jaká studia budou dostupná v rezervačním systému. Administrátor může zobrazit vlastní profil, ve kterém může dočasně změnit svou roli.

Uživatelské rozhraní systému bylo implementováno pomocí knihovny React. Systém je použitelný na desktopových zařízeních i na mobilních zařízeních díky využití frameworku React Bootstrap. Backend systému je implementován s využitím NodeJS a frameworku Express, který komunikuje s NoSQL MongoDB databází.

9 Závěry a doporučení

Cílem této bakalářské práce bylo vytvořit funkční aplikaci rezervačního systému, který by klientka mohla využit ve svém podnikání. Cílem implementace nového rezervačního systému je snížit transakční náklady každého podnikatele v daném Salonu, a tím zvýšit jejich zisk, a také optimálně rozvrhnout časový průběh poskytování služeb, aby nedocházelo k časovým prodlevám.

Klientka zvažovala využití již existujícího rezervního systému, ale preferovala free verze, které jsou pochopitelně omezené. Pokud by za rezervační systém platila, požadovala jej upravit podle svých požadavků. Výhodou vlastního řešení je více možností pro další rozvoj podle budoucích požadavků.

Na základě analýzy vznikla řada funkčních a nefunkčních požadavků na systém, podle kterých probíhala následná implementace. Jako stěžejní technologie pro vývoj aplikace byl zvolen technologický stack MERN, tedy MongoDB, Express, React a NodeJS. Využití těchto technologií znamenalo, že se na frontendu i backendu používal jazyk JavaScript, což významně usnadňovalo vývoj.

Testování systému probíhalo formou představení funkcí systému klientce. Nalezené nedostatky systému a zpětná vazba klientky byli následně zapracovány do systému.

Výsledná aplikace splňuje všechny předem určené požadavky a lze konstatovat, že cíl bakalářské práce byl splněn.

10 Seznam použité literatury

[1] Webová stránka společnosti GARVIS Solutions, s.r.o., [online, cit. 10. 6. 2022], dostupné z https://reenio.cz/cs/balicky#PricingPlan

[2] VENCOUROVÁ, Monika. *Rezervační systém reenio jako doplněk moderní prezentace.* Mioweb, [online, cit. 14. 8. 2022], dostupné z https://www.mioweb.cz/blog/rezervacni-system-reenio-jako-doplnek-moderni-webove-prezentace/

[3] Webová stránka společnosti SuperSaaS, B.V., [online, cit. 10. 6. 2022], dostupné z https://www.supersaas.cz/info/cenik

[4] Webová stránka společnostiSuperSaaS, B.V., [online, cit. 10. 6. 2022], dostupné z https://www.supersaas.cz/info/terms

[5] Webová stránka společnosti SuperSaaS, B.V., [online, cit. 10. 6. 2022], dostupné z https://www.supersaas.cz/info/doc/integrace

[6] Webová stránka společnosti Reservio, s.r.o., [online, cit. 10. 6. 2022], dostupné z https://www.reservio.com/cs/cenik

[7] JavaScript reference. Webová stránka mdn web docs., [online, cit. 14. 8. 2022], dostupné z https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

[8] Dokumentace knihovny mongoose. Webová stránka mongoose., [online, cit. 14. 8. 2022], dostupné z https://mongoosejs.com/docs/guide.html

[9] Dokumentace frameworku Express 4.X. Webová stránka Expressjs., [online, cit. 14. 8. 2022], dostupné z https://expressjs.com/en/4x/api.html

[10] Dokumentace knihovny React. Webová stránka společnosti Meta Platforms, Inc., [online, cit. 14. 8. 2022], dostupné z https://reactjs.org/docs/getting-started.html

[11] Dokumentace frameworku React Bootstrap. Webová stránka React Bootstrap, [online, cit. 14. 8. 2022], dostupné z https://react-bootstrap.github.io/

[12] Dokumentace middlewaru express-validator. Webová stránka express-validator, [online, cit. 26. 6. 2022], dostupné z https://express-validator.github.io/docs/

[13] Dokumentace komponenty Toast. Webová stránka React Bootstrap, [online, cit. 14. 8. 2022], dostupné z https://react-bootstrap.github.io/components/toasts/

[14] Dokumentace middleware knihovny Passport. Jared Hanson, Webová stránka Passport, [online, cit. 8. 6. 2022], dostupné z https://www.passportjs.org/docs/

[15] Google Sign-In JavaScript client reference, Google, LLC, [online, cit. 5. 6. 2022], dostupné z https://developers.google.com/identity/sign-in/web/reference#users

[16] db.collection.findOneAndUpdate(), MongoDB, Inc. [online, cit. 8. 6. 2022], dostupné z https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#db.collection.findoneandupdate

[17] Repozitář s návodem k použití pluginu mongose-paginate-v2, Aravind NC, mongoose-paginate-v2. [online, cit. 31. 5. 2022], dostupné z https://github.com/aravindnc/mongoose-paginate-v2#readme

[18] Návod k použití modulu Nodemailer, Andris Reinman, Nodemailer, [online, cit. 1. 6. 2022], dostupné z https://nodemailer.com/smtp/

[19] Repozitář s návodem k použití nástroje MailHog, [online, cit. 1. 6. 2022], dostupné z https://github.com/mailhog/MailHog

[20] Get Docker, Docker, Inc., [online, cit. 15. 6. 2022], dostupné z https://docs.docker.com/get-docker/

[21] Install Docker Compose, Docker, Inc., [online, cit. 15. 6. 2022], dostupné z https://docs.docker.com/compose/install/

[22] Co je informační systém, Aira GROUP, s.r.o., [online, cit. 10. 7. 2022], dostupné z https://www.sprava-site.eu/informacni-system/

[23] ARLOW, Jim, NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Dotisk prvního vydání. Brno: Computer Press, 2011. ISBN 978-80-251-1503-9.

[24] RUMBAUGH, James, JACOBSON, Ivar, BOOCH, Grady. *The Unified Modeling Language Reference Manual.* Reading, Mass.: Addison-Wesley, 1999. ISBN 0-201-30998-x.

[25] DOSEDĚL, Tomáš. *Počítačová bezpečnost a ochrana dat*. 1. vyd. Brno: Computer Press, 2006, ISBN 8025101061, citováno dle KNOPOVÁ, Martina. Bezpečnost dat v informačních systémech. *Ikaros* [online]. 2011, ročník 15, číslo 6. ISSN 1212-5075. [online, cit. 11. 5. 2022], dostupné z http://ikaros.cz/node/13714

11 Přílohy

1. Zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Tomáš Zahradník**
Studium:	I1800244
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Rezervační systém**
Název bakalářské práce AJ:	Reservation system

Cíl, metody, literatura, předpoklady:

Cíl: Navrhnout a implementovat rezervační systém pro firmy.

Osnova:

1. Úvod
2. Analýza existujících řešení
3. Návrh vlastního řešení
4. Vyhodnocení dostupných technologií
5. Implementace vlastního řešení
6. Kritická diskuze
7. Závěr

Mantyla, Dan. *Functional Programming in JavaScript*. Packt Publishing Ltd, 2015.

CHIANG, Chun-Fang; JANG, SooCheong Shawn. The effects of perceived price and brand image on value and purchase intention: Leisure travelers' attitudes toward online hotel booking. Journal of Hospitality & Leisure Marketing, 2007, 15.3: 49-69.

NEWMAN, Greg, et al. User-friendly web mapping: lessons from a citizen science website. International Journal of Geographical Information Science, 2010, 24.12: 1851-1869.

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Mgr. Daniela Ponce, Ph.D.
Oponent:	Ing. Tomáš Nacházel, Ph.D.
Datum zadání závěrečné práce:	21.10.2019