ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačov

Diplomová práca

Informačný systém pre správu autoservisov

Bc. Daniel Lamper
Študijný program: Otevřená informatika, Odbor: Softwarové inženýrství

Máj 2019
Vedúci práce: Ing. Božena Mannová, Ph.D

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lamper** Jméno: **Daniel** Osobní číslo: **406414**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Informační systém pro správu autoservisů

Název diplomové práce anglicky:

Management system of car repair shops

Pokyny pro vypracování:

Navrhněte systému pro společnost, která se zabývá správou autoservisů a jejich nabídkou služeb zákazníkům. Navrhovaná aplikace by měla umožnit jednodušší komunikaci mezi zákazníky a autoservisy a usnadnit administrativu autoservisů. Seznamte se s dostupnými řešeními a proveďte jejich analýzu.
Na základě provedené analýzy specifikujte konkrétní funkční a nefunkční požadavky na navrhovaný systém. Zohledněte při specifikaci požadavky konkrétní společnosti, provozující takové služby. Navržený systém bude zákazníkům poskytovat možnost najít, vybrat a rezervovat servisní úkon ve vybraném servisu podle jeho potřeb. Systém bude navržen jako webová aplikace. Pro analýzu a návrh použijte vhodné prostředky softwarového inženýrství. Navrženou aplikaci dostatečné otestujte (včetně testů použitelnosti).

Seznam doporučené literatury:

1. Ian Sommerville: Software Engineering, Global Edition, Pearson Higher Ed, 2016, ISBN1292096144
2. Arlow, J., Neustat, I.: UML 2 a unifikovaný proces vývoje aplikací. Computer Press, 2007

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Božena Mannová, Ph.D., kabinet výuky informatiky FEL

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

Datum zadání diplomové práce: **06.02.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

Ing. Božena Mannová, Ph.D.		prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

24. 4. 2019
Datum převzetí zadání

Podpis studenta

Poďakovanie / Prehlásenie

V prvom rade by som chcel poďakovať mojej mamine, ktorá mi umožnila štúdium na vysokej škole. Najbližšej rodine (predovšetkým mamine a bratovi) za podporu pri zvládaní nástrah tohto štúdia. Veľká vďaka patrí vedúcej mojej práce, Ing. Božene Mannovej, Ph.D, za odvahu pustiť sa so mnou do tejto práce. V neposlednom rade sa chcem poďakovať mojim kamarátom a kolegom, za rady, ktoré mi pomohli pri vypracovaní tejto práce.

Prehlasujem, že som predloženú prácu vypracoval samostatne, a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 22. 5. 2019

.....

Abstrakt / Abstract

Diplomová práca obsahuje aplikáciu pre správu autoservisov, ktorá prináša možnosť vyhľadávania, vyberania a rezervácie servisného úkonu v zvolenom autoservise, prostredníctvom jednej webovej stránky.

V úvode je popísaná motivácia a ciele tejto práce. Kapitola analýzi rozoberá podobné riešenia, definuje funkčné a nefunkčné požiadavky, vypracováva prípady použitia a skúma možný výber architektúry systému. Ďalšie kapitoly sa venujú návrhu a implementácie aplikácie. Pre implementáciu bol zvolený model Softvér ako služba ktorý bol spracovaný formou webovej aplikácie. Súčasťou práce je rovnako testovanie zahŕňajú aj testy použiteľnosti. Záver práce je venovaný hodnoteniu výsledkov a možnostiam ďalšieho vývoja.

Kľúčové slová: autoservis, zákazník, webová aplikácia, rezervácia, vyhladávanie, služby

This master thesis contains an application for management of car repair shops. The application offers the possibility to search, select, and book a service at a chosen car repair shop, all at one website.

The introduction describes the motivation and goals of this work. The Analysis chapter examines similar solutions, defines functional and non-functional requirements, produces use cases, and examines the possible system architecture choices. The next chapters deal with the design and implementation of the application. For implementation, the Software as a Service model was selected and processed in the form of a web application. Part of the thesis is also testing, which includes usability tests. The conclusion is devoted to the evaluation of the results and possibilities of further development.

Keywords: car repair shop, customer, web aplication, reservation, search, services

Obsah /

Tabuľky / Obrázky

Kapitola 1
Úvod

Táto diplomová práca sa zameriava na návrh prototypu systému pre spoločnosť DriveCare s.r.o., ktorej záujmom sú dve skupiny subjektov. Do prvej skupiny patria autoservisy, pneuservisy a autoumyvárne (ďalej len prevádzky) a druhou skupinou je široká verejnosť (ďalej len zákazníci).

Aplikácia, ktorú máme za cieľ navrhnúť a vyvinúť by mala umožniť jednoduchšiu komunikáciu medzi zákazníkom a prevádzkou. Zároveň by mala zefektívniť a uľahčiť vybrané administratívne úkony pre prevádzky.

1.1 Motivácia

V dnešnej dobe sme zvyknutý zariadiť čo najviac vecí z pohodlia domova, kancelárie, kaviarne alebo akéhokoľvek iného miesta pomocou počítača, mobilu poprípade tabletu behom chvíle, bez toho, aby sme museli s niekým komunikovať na priamo alebo telefonicky. To platí aj pri riešení problémov s vozidlom, nikto nemá čas a nechce obchádzať desiatky autoservisov len kvôli výmene oleja, alebo čakať desiatky minút pred autoumývarňou len aby mal svoje vozidlo opäť lesklé a čisté.

Hlavným cieľom a motiváciou spoločnosti DriveCare s.r.o. je odbremeniť zákazníkov od týchto starostí a poskytnúť im možnosť rezervovať si termín návštevy v servise či pneuservise alebo objednať umytie svojho vozidla z pohodlia domova prostredníctvom webovej aplikácie.

Je pravda, že existujú servisy u ktorých je možné sa rezervovať telefonicky no častokrát nikto nedvíha telefón. Dokonca existujú aj prevádzky, ktoré na svojích webových stránkach poskytujú online rezervačný systém ale takých prevádzok je málo a človek musí vynaložiť veľa úsilia aby ich našiel. Preto vznikla myšlienka zjednotiť prevádzky a poskytovať ich služby na jednej webovej stránke, kde si zákazníci lahko nájdu, čo potrebujú.

1.2 Ciele

Cieľom tejto diplomovej práce je navrhnúť a vyvinúť prototyp systému, ktorý bude zákazníkom poskytovať možnosť nájsť, vybrať a rezervovať prevádzku podľa potrieb a požiadaviek. Tento systém bude navrhnutý ako responzívna webová aplikácia. Pre návrh a implementáciu budú použité štandardné nástroje a postupy softwarového inžinierstva.

1.3 O spoločnosti

Spoločnosť DriveCare s.r.o. (ďalej len spoločnosť) je startupová spoločnosť pôsobiaca na Slovenskom trhu niekoľko mesiacov. Táto spoločnosť vznikla s ambiciou vytvoriť aplikáciu, ktorá by v prvom rade uľahčila rezervácie úkonov v prevádzkach a v druhom

rade poskytovala široké spektrum možností, čo sa prevádzok a poskytovaných služieb týka. Zároveň, by táto spoločnosť chcela prevádzkam, z dlhodobého hľadiska, uľahčiť prácu a náklady, tým, že zefektívni niektoré administratívne úkony ako napríklad potvrdzovanie rezervácií.

Kapitola 2
Analýza

V kapitole analýzi hladáme, rozoberáme a skúmame existujúce riešenia, aké majú nedostatky, poprípade, čo je dobré a využiteľné. Ďalej si prejdeme celkovú analýzu problematiky z čoho nadviažeme na funkčné a nefunkčné požiadavky, definíciu prípadov použitia a analýzu možnej voľby architektúry.

2.1 Prehľad existujúcich riešení

V tejto časti sme chceli rozoberať podobné existujúce aplikácie na slovenskom trhu no bohužiaľ žiadna aplikácia ktorá by zastrešovala všetky, poprípade veľkú množinu servisov, neexistuje, alebo sa nám ju nepodarilo nájsť. Existujú však združenia servisov kde je možné nájsť viacero prevádzok. Prípadne niektoré prevádzky majú možnosť online rezervácie na svojich stránkach.

Q-Service je sieť nezávislých autoservisov, ktoré pôsobia na Slovensku a v iných krajinách Európy. Na ich stránkach je možné vyhľadávať servisy podľa krajov alebo podľa mesta, kde sa po zvolení napríklad kraja zobrazí zoznam servisov v danom kraji. Nie je úplne zrejmé, podľa čoho sú prevádzky zoradené a nie je nikde možné zvoliť spôsob radenia. O každej prevádzke sú v zozname zobrazené informácie ako adresa, kontakt na prevádzku (e-mailová adresa a/alebo telefónne číslo), názov prevádzky a zoznam poskytovaných služieb, kde zoznam služieb je viditeľný podľa ikoniek (viz obrázok 2.1).

Po rozkliknutí detailu servisu užívateľ vidí ďalšie informácie o servise, ako adresu sídla, kontaktné údaje a to konkrétne e-mailovú adresu, telefónne číslo a adresu webovej stránky prevádzky. Užívateľ ďalej môže vidieť otváracie hodiny, mapu, kde sa servis nachádza, poskytované služby a ďalšie poskytované úkony, partnerské poisťovne, poistné udalosti ktoré je prevádzka schopná riešeť a dodatkové informácie. Navyše je možné, vytvoriť online objednávku do servisu. Pre úspešné odoslanie objednávky je treba vyplniť formulár s informácie o modeli a type vozidla, kontaktnými údajmi osoby, to je celé meno/názov firmy, e-mail a telefón. Rovnako treba uviesť o aký úkon by sa jednalo, prípadne popis k úkonu a preferovaný termín príchodu s vozidlom do prevádzky. Objednávka je odoslaná na manuálne spracovanie prevádzkou. Pri výbere servisného úkonu je možné vybrať jednu z piatich možností `Servisná prehliadka`, `Mechanická oprava`, `Oprava po havárii`, `Záručná oprava` alebo `Iné`. Užívateľ si v sekcii termín môže vybrať deň a časť dňa (ráno, doobeda, poobede) no nie konkrétny čas príchodu s vozidlom do prevádzky. Pri vytváraní objednávky nemá užívateľ možnosť vedieť či a ako má servis naplnené kapacity na to, aby ho prijali.

Obrázok 2.1. Príklad zobrazenia prevádzky v systéme Q-Service

Na základne informácii získaných analýzou Q-Service nám príde ako hlavný nedostatko v tomto riešení, že pod túto sieť nespadá veľa servisov. Pre porovnanie, v aplikácii Q-Service nám našlo dva autoservisy v Banskej Bystrici a pri vyhľadávaní Autoservisov registrovaných v Banskej Bystrici cez obchodný register sme ich našli 134 [1], čo je podstatne viac. Ďalším nedostatkom, ktorý vidíme v tejto aplikácii je, že zákazník nevidí ako ďaleko je prevádzka od jeho aktuálnej polohy, napriek tomu, že sa v prehľade prevádzky nachádza mapa s jej vyznačenou polohou. Jediný spôsob ako to môže zistiť je skopírovať adresu a nájsť si prevádzku cez stránky poskytujúce mapy. Posledný nedostatok, ktorý vidíme je v tom, že užívateľ pri vytváraní objednávky nevie aké sú reálne časové možnosti a vyťaženie prevádzky, dozvie sa to až po manuálnom spracovaní objednávky a to len čiastočne.

Ďalšia aplikácie cez ktoré je možné nájsť autoservisy sú napríklad zoznam.sk, azet.sk, autovia.sk samozrejme google, ale všetky tieto systémy poskytujú len zoznam prevádzok poskytujúcich služby v oblasti servisovania vozidiel, no nie je možné vytvoriť rezervácie priamo z ich stránok, užívateľ musí prejsť na stránky prevádzky. Navyše nie všetky údaje sú aktuálne a pravdivé, napríklad sa nám stalo, že číslo, ktoré bolo u prevádzky zadané na jednom so spomínaných systémov bolo neplatné.

2.2 Analýza problému

V predošlej sekcii sme si ukázali príklady podobných existujúcich riešení, čo nám ukázalo, že na Slovenskom trhu neexistuje podobný systém tomu, ktorý sa snažíme vytvoriť. Najbližším podobným riešením sa môže zdať sieť autoservisov Q-Service, no to len zdanlivo. Na základe informácii získaných analýzov spomínaného Q-Service sme vyvodili záver, že ide skôr o združenie servisov ktoré spĺňajú kritéria potrebné pre možnosť patriť pod túto sieť. Zákazníkom to poskytuje akúsi istotu o kvalite a poskytovaných službách. Webová aplikácia spomínaného združenia poskytuje vytvorenie online objednávky, ale ako je už spomínané vyššie, tieto objednávky majú svoje nedostatky. Ostatné služby ktoré umožňujú vyhladavanie autoservisov poskytovali len jednoduchý zoznam prevádzok.

[1] táto informácia bola nájdená na stránke finstat.sk[1]

2.2.1 Navrhované riešenie

Z dôvodov uvedených vyššie, vznikla myšlienka vytvoriť systém, ktorý by riešil všetky spomínané nedostatky a omnoho viac. Cieľom je vytvoriť aplikáciu, ktorá na jednej stránke zobrazuje všetky dostupné autoservisy spĺňajúce vyhľadávacie kritéria zákazníka. Navyše by takáto aplikácia poskytovala služby pneuservisov a autoumyvární.

Hlavným účelom aplikácie je poskytovať možnosť vytvoriť rezervácie v ktorejkoľvek prevádzke. Umožňuje zvoliť voľný termín u prevádzky, zobrazuje časovú vyťaženosť prevádzky a zoznam poskytovaných úkonov a služieb. Aplikáciu, ktorá rovnako uľahčí a zefektívni prácu prevádzok zavedením rezervačného systému.

Inštalácia systému v prevádzkach nie je žiaduca preto je potrebný taký návrh aplikácie, ktorý by umožňoval používanie systému, bez nutnosti inštalácie a nasadzovaniu aplikácie u prevádzky. “Software ako Servis (SaaS) opisuje situáciu kedy si užívateľ prenajíma alebo požičia online systém namiesto toho, aby ho zaobstaral a nainštaloval na vlastných počítačoch”[2]. Ináč povedané, je to model, kedy aplikácia beží u poskytovateľa služby a zákazník ju využíva prostredníctvom internetu. V takomto prípade nie je potrebné nič inštalovať u zákazníka, čo, zo strany zákazníka, zjednodušuje náročnosť na údržbu. Jedinú požiadavku, ktorú musí zákazník spĺňať, je mať prístup na internet prostredníctvom prehliadača.

2.3 Funkčné požiadavky

V tejto sekcii spisujeme a rozoberáme funkčné požiadavky na aplikáciu, aké by mala poskytovať služby a zároveň ako by sa mala chovať v rôznych situáciách. Funkčné požiadavky boli zozbierané a zostavené na základe požiadaviek spoločnosti DriveCare a niektorých opýtaných autoservisov. Každý z funkčný požiadaviek má svoju prioritu. Množina a kritéria priorít sú definované v tabuľke 2.1.

Skratka	Hodnoty atribútov priorita	Sémantika
M	Must Have (Nevyhnuteľný)	Povinné požiadavky, tvoria základ systému
S	Should Have (Možné)	Dôležité požiadavky, je ich však možné vynechať
C	Could Have (Eventuálne)	Nepovinné požiadavky, keď na ne zostane čas
W	Want to Have (Chceli by sme)	Požiadavky, ktoré sa pravdepodobne objavia niekedy v budúcnosti

Tabuľka 2.1. Popis priorít pre požiadavky.[3].

2.3.1 Role

Priorita: M

Užívateľ s konkrétnou rolou má rôzne oprávnenia. Na základe týchto oprávnení má užívateľ prístup k rôznym úkonom a rôznym častiam aplikácie.

2.3.1.1 Prevádzka [*Shop*]

Užívateľ s rolou *Prevádzka* môže:

- Spravovať vlastnú prevádzku, informácie o nej, list a detaily o poskytovaných úkonoch.

- Vytvárať nové rezervácie, upravovať a mazať existujúce rezervácie, to zahŕňa aj rezervácie ktoré neboli vytvorené prevádzkou ale zákazníkom v prevádzke.

2.3.1.2 Zákazník [*Customer*]

Užívateľ s rolou *Zákazník* môže:

- Spravovať vlastný účet, čiže upravovať svoje kontaktné údaje a spravovať svoje autá (pridávať, upravovať a mazať).
- Vytvárať nové rezervácie na úkon v prevádzke, taktiež môže upravovať alebo mazať svoje rezervácie.
- Zobraziť vlastnú históriu vykonaných úkonov.
- Zobrazovať detail prevádzky.

2.3.1.3 Administrátor [*Admin*]

Užívateľ s rolou *Administrátor* môže:

- Vytvoriť nového užívateľa s rolou *Prevádzka*.
- Má prístup do všetkých častí systému.
- Zobraziť detail o všetkých zákazníkoch a prevádzkach.
- Mazať užívateľov alebo im blokovať prístup.

2.3.2 Autentifikácia a autorizácia

Priorita: M

Užívateľ môže navštíviť nasledujúce stránky bez prihlásenia:

- Stránka prihlásenia.
- Stránka registrácie zákazníka.
- Stránka vyhľadávanie prevádzok.
- Stránka s mapou a výberom prevádzok.
- Stránka detailu prevádzky.

Neautorizovaný užívateľ môže rovnako vytvárať nové rezervácie. Užívatelia sú autorizovaný pomocou užívateľského mena a hesla.

2.3.2.1 Prihlásenie

Systém poskytuje iba jednu prihlasovaciu stránku. Po prihlásení užívateľa s rolou *Prevádzka* je presmerovaný na stránku prehľadu vlastnej prevádzky. Po prihlásení užívateľa s rolou *Zákazník* je presmerovaný na stránku vyhľadávania prevádzok.

2.3.2.2 Registrácia

Systém poskytuje registračnú stránku, ktorá bude dostupná bez autorizácie. Nový užívatelia s rolou *Zákazník* môžu byť registrovaný cez túto stránku. Registrácia užívateľov s rolou *Prevádzka* bude možná cez inú registračnú stránku, ktorá bude prístupná len prihláseným užívateľom s rolou *Administrátor*. Po úspešnej registrácii, bude užívateľovi poslaný aktivačný email.

2.3.3 Detail užívateľa

Priorita: M

Systém bude uchovávať nasledujúce informácie o všetkých užívateľoch, to je rovnako o zákazníkoch a prevádzkach:

2.3.3.1 Užívateľské meno

Užívateľské meno je povinné pole používané pre prihlasovanie. Užívateľské meno môže obsahovať len malé/veľké písmená, číslice a špeciálne znaky. Minimálna dĺžka užívateľského mena je šesť znakov, maximálna dĺžka je tridsaťdva znakov.

2.3.3.2 Heslo

Heslo je povinné pole použité pre prihlasovanie. Ukladané bude v zahešovanej forme. Heslo môže obsahovať znaky zo štyroch skupín: malé písmená, veľké písmená, číslice a špeciálne znaky. Minimálna dĺžka hesla je šesť znakov, maximálna dĺžka je tridsaťdva znakov. Heslo musí obsahovať aspoň tri zo štyroch skupín znakov.

Heslo nepodlieha expirácii, z čoho vyplýva, že užívateľ môže používať rovnaké heslo počas celej doby užívania systému.

V prípade, že užívateľ heslo zabudne, systém poskytuje možnosť obnovy hesla. Pre vytvorenie nového hesla bude užívateľ povinný zadať emailovú adresu, kam systém odošle unikátnu URL na ktorej bude možné nové heslo vytvoriť.

Systém si bude pamätať posledné tri heslá užívateľa. Pri zmene hesla, urobí systém validáciu zhody nového hesla s uloženými heslami. Tieto heslá sa nesmú navzájom zhodovať.

2.3.3.3 Role

Každý užívateľ má práve jednu rolu. Role sú definované v sekcii 2.3.2.

2.3.3.4 Stav

Stav sa zákazníkovi zobrazovať nebude, rovnako nebude môcť toto pole meniť v nastaveniach účtu. Užívateľ môže naberať tieto stavy: **NOT_ACTIVATED** [*Neaktivovaný*], **ACTIVE** [*Aktívny*], **BLOCK** [*Blokovaný*] a **DELETED** [*Zmazaný*].

2.3.4 Detail zákazníka

Priorita: M

Systém bude uchovávať informácie o zákazníkoch. Každý zákazník má definované povinné polia a to osobné meno, priezviskom, e-mailová adresa a telefónne číslo. Ďalej budú ukladané aj nepovinné polia a to kontaktná adresa, história návštev v prevádzkach a informácie o vozidlách užívateľa.

2.3.4.1 Informácie o vozidle

Systém bude uchovávať informácie o vozidlách zákazníkov. Každý zákazník môže mať niekoľko vozidiel. Povinné polia pre vozidlo sú identifikačné číslo vozidla (takzvané VIN číslo), evidenčné číslo vozidla, kategória (vozidlo do 3.5t, malé nákladné a tak ďalej) a značka. Ďalej má definované nepovinné polia a to model, rok výroby, typ paliva, výkon motora v kW, karoséria, prevodovka, pohon vozidla a prevodovka.

2.3.5 Detail prevádzky

Priorita: M

Systém bude uchovávať informácie o prevádzkach. Povinné polia pre prevádzky sú e-mailová adresa, aspoň jedno telefónne číslo, adresa prevádzky, DIČ/IČO a typ prevádzky. Nepovinné polia, ktoré budú ukladané pre prevádzky sú URL, cena servisnej hodiny, list poskytovaných úkonov s detailom. Každý servis bude mať definovaný maximálny počet rezervácii, ktoré je možné vytvoriť na jeden čas. Tak isto bude mať definované kľúčové slová, ktoré budú určovať, aký typ úkonov prevádzka poskytuje. Tieto kľúčové slová budú použité pre jednoduchšie a presnejšie vyhľadávanie zákazníkmi.

2.3.5.1 Typ prevádzky

Systém bude definovať rôzne typy prevádzok, kde typ prevádzky bude určovať hlavné zameranie prevádzky. Každá prevádzka bude mať vyplený práve jeden typ:

- Autoservis,
- Pneuservis,
- Autoumyváreň

2.3.5.2 Informácie o úkonoch

Každý úkon bude mať špecifikované nasledujúce polia:

- Názov - názov bude definovať úkon a zároveň bude použitý ako kľúčové slovo pre vyhladávanie.
- Kategórie - úkon bude patriť do minimálne jednej z troch kategórii **A**, **B** a **C**. Existencia kategórii je spôsobená tým, že jeden úkon môže pre rôzne typy aut znamenať iný obnos práce, inú sumu náhradných dielov a inú celkovú náročnosť. Kategórie sú rozdelené podľa typu a druhu na tri skupiny:
  - **A** - štandardné vozidlá s lacnými náhradnými dielmi a s jednoduchou konštrukciou, čiže úkony spravidla nezaberú toľko času.
  - **B** - vozidlá s drahšími ako lacnými náhradnými dielmi a so zložitejšou konštrukciou, čiže úkony spravidla zaberú viac času ale stále nie sú až tak náročné.
  - **C** - drahé a prémiové vozidlá s drahými náhradnými dielmi a so zložitou konštrukciou, čiže úkony spravidla zaberú veľa času a úsilia.
- Cena - cena vykonania úkonu.
- Doba trvania - odhadovaný čas trvania úkonu.

2.3.6 Rezervácie

Priorita: M

Systém bude poskytovať možnosť rezervácie úkonu v prevádzach, ktoré sú registrované v aplikácii.

Nová rezervácia bude, buď potvrdená automaticky alebo ju bude musieť potvrdiť prevádzka manuálne. To, či sa potvrdí rezervácia automaticky alebo manuálne bude závisieť na nastaveniach u prevádzky a na základe rezervovaného úkonu. Niektoré úkony môžu vyžadovať súčiastky, ktoré daná prevádzka nemá k dispozícii a je závislá na čase dodania od svojich dodávateľov. Keď bude rezervácia potvrdená, zákazník bude informovaný e-mailom alebo notifikáciou v aplikácii.

Pri manuálnom potvrdení rezervácie, bude prevádzka povinna vyjadriť sa nasledovne:

- Ak bude rezervácia vytvorená na iný deň ako aktuálny a to v dobe medzi 8:00 a 16:00, je táto prevádzka povinná rezerváciu potvrdiť ešte v ten deň, ak tak neučiní rezervácia je automaticky zrušená a zákazník aj servis sú o tom informovaný.

  Vzorová situácia: aktuálny deň je pracovný deň 22.11.2018, prevádzka otvára o 9:00 a zatvára o 18:00. Zákazník vytvorí rezerváciu dňa 22.11.2018 o 15:32 na 28.11.2018 o 15:00. Servis je povinný potvrdiť túto rezerváciu do 22.11.2018 23:59.
- Ak bude rezervácia vytvorená na iný deň ako aktuálny a to v dobe medzi 16:00 aktuálneho dňa a 8:00 nasledujúceho dňa, je prevádzka povinná užívateľovi túto rezerváciu potvrdiť najneskôr do 10:00 nasledujúceho dňa, ak tak neučiní, rezervácia je automaticky zrušená a zákazník aj servis sú o tom informovaný.

Vzorová situácia: aktuálny deň je pracovný deň 22.11.2018, prevádzka otvára o 9:00 a zatvára o 18:00. Zákazník vytvorí rezerváciu dňa 23.11.2018 o 7:24 na 28.11.2018 o 15:00. Servis je povinný potvrdiť túto rezerváciu do 23.11.2018 10:00.

- Ak bude rezervácia vytvorená na aktuálny deň o viac ako hodinu a pol pred požadovaným časom rezervácie, prevádzka je povinná najneskôr hodinu pred časom rezervácie potvrdiť túto rezerváciu v opačnom prípade bude automaticky zrušená a zákazník aj servis budú o tom informovaný.

  Vzorová situácia: aktuálny deň je pracovný deň 22.11.2018, prevádzka otvára o 9:00 a zatvára o 18:00. Zákazník vytvorí rezerváciu dňa 22.11.2018 o 10:15 na 22.11.2018 o 13:00. Servis je povinný potvrdiť túto rezerváciu do 22.11.2018 12:00.
- Nie je možné vytvoriť rezerváciu menej ako hodinu a pol pred požadovaným časom rezervácie.

  Vzorová situácia: aktuálny deň je pracovný deň 23.11.2018, prevádzka otvára o 9:00 a zatvára o 18:00. Zákazník požaduje vytvoriť rezerváciu dňa 23.11.2018 o 13:00 na 23.11.2018 o 14:30. Systém zákazníkovi neumožní vytvoriť takúto rezerváciu.

Pri automatickom potvrdení rezervácie bude možné vytvoriť rezerváciu najneskôr v požadovanom čase úkonu.

V prípade že zákazník vytvára rezerváciu na úkony, ktoré budú trvať niekoľko dní, rezervácia sa vytvára na čas kedy dovezie vozidlo do prevádzky a nie na celkový úkon. Na čase vyzdvihnutia vozidla sa dohodne prevádzka so zákazníkom samostatne alebo bude môcť v aplikácii označiť predpokladaný čas odovzdania auta. Tento čas sa môže meniť, zákazník bude o týchto zmenách informovaný pomocou upozornení.

Užívateľ si bude môcť v rámci jednej rezervácie zvoliť viacero úkonov, celkový čas rezervácie sa sčíta na základe predpokladanej dĺžky zvolených úkonov. Každá rezervácia bude kontinuálna. Ak nebude možné vykonať požadované úkony v jednom časovom bloku, užívateľ bude musieť vytvoriť viacero samostatných rezervácii na každý požadovaný úkon.

Vzorové situácie:

- Zákazník požaduje výmenu pneumatík a umytie auta, oba tieto úkony trvajú po hodine, prevádzka v ktorej chce vytvoriť rezerváciu má voľné tieto termíny, od 9:00 do 11:00 a od 15:00 do 16:30, zákazník v takomto prípade môže zvoliť voľný čas od 9:00 do 11:00 a vytvoriť jednu rezerváciu na oba úkony.
- Zákazník požaduje výmenu pneumatík a umytie auta, oba tieto úkony trvajú po hodine, prevádzka v ktorej chce vytvoriť rezerváciu má voľné tieto termíny, od 9:00 do 10:00 a od 15:00 do 16:30, zákazník v takomto prípade musí vytvoriť dve samostatné rezervácie a to pre každý z požadovaných úkonov. Ďalej je už na zákazníkovi ako sa dohodne s prevádzkou, či si vyzdvihne vozidlo medzi jednotlivými úkonmi alebo ho nechá v prevádzke.

Prevádzka bude mať možnosť označiť stav rezervácie, zákazník tak môže sledovať čo sa deje s jeho vozidlom. Prevádzky tieto údaje budú môcť využívať k zefektívneniu odhadovania náročnosti práce a k budúcemu plánovaniu úloh. Každý úkon bude mať nasledujúce stavy: **Zadaný**, **Potvrdený**, **Spracovávaný**, **Dokončený**. V prípade viacerých úkonov sa bude zákazníkovi zobrazovať stav každého úkonu samostatne.

Rezerváciu bude môcť upravovať zákazník ktorý ju vytvoril.

Systém nebude poskytovať možnosť vytvoriť rezerváciu menej ako hodinu pre operácie, ktoré nie sú automaticky potvrdené.

každý zákazník bude môcť vidieť svoje rezervácie v prehľade rezervácii. Na nadchádzajúcu rezerváciu bude zákazník upozornený v určený čas zvoleným spôsobom. Spô-

sob upozornenia môže byť buď pomocou notifikácie alebo emailu a zákazník si ho bude môcť nastaviť v nastaveniach účtu. Zákazník bude môcť vypnúť upozornenia. Zákazník si bude môcť zvoliť možnosť upozornenia na možné meškanie vykonania jeho rezervácie, toto bude užitočné a platné na jednoduché úkony ktoré budú vykonané na počkanie.

Zákazník bude môcť rezervaciu stornovať cez systém ale to maximálne 24 hodín pred časom uskutočnenia úkonu. Ak bude chcieť zákazník stornovať rezerváciu menej ako 24 hodín pred uskutočnením, tak bude musieť kontaktovať priamo prevádzku. Systém, nebude poskytovať možnosť zrušenia rezervácie na úkony, pre ktoré prevádzka objednávala náhradné diely, takéto rezervácie bude musieť zákazník stornovať kontaktovaním prevádzky.

Pri stornovaní rezervácie prevádzkou bude prevádzka povinná vyplniť dôvod zrušenia rezervácie.

2.3.7 Filtrovanie a vyhľadávanie prevádzok

Priorita: M

Aplikácia bude ponúkať vyhľadávanie požadovaných prevádzok podľa typu, čiže či zákazník požaduje zobraziť autoservisy, pneuservisy alebo autoumyvárne. Prevádzky bude možné vyhľadávať podľa polohy, poprípade mesta v ktorom sa prevádzky nachádzajú. Zákazník bude mať rovnako možnosť vyhľadávať podľa kľúčových slov definovaných prevádzkou ako aj podľa názvu prevádzky.

2.3.8 Recenzie

Priorita: S

Zákazník bude mať možnosť hodnotiť prevádzku ktorú navštívil, hodnotiť kvalitu úkonu ktorá bola vykonaná na jeho vozidle a zároveň ohodnotiť samotnú prevádzku ako bol spokojný s prístupom zamestnancov, rýchlosťou vybavenia a aký mal celkový dojem z návštevy. Rezencie môžu písať a dávať len registrovaný a overený užívatelia. Užívateľ je overený po aktivácii účtu cez odkaz, ktorý mu bude zaslaný na e-mailovú adresu.

2.3.9 História úkonov

Priorita: C

Systém bude ukladať históriu úkonov vykonaných na vozidle. História bude obsahovať úkon ktorý bol vykonaný, v prípade výmeny súčiastok, zoznam týchto súčiastok. Ďalej bude v histórii názov prevádzky v ktorej bol úkon vykonaný, čas a dátum od kedy do kedy bolo auto v prevádzke a cena koľko zákazník prevádzke zaplatil. Prevádzka takisto zapíše kilometre vozidla.

2.4 Nefunkčné požiadavky

2.4.1 Použiteľnosť

Systém musí byť intuitívny a jednoduchý na používanie. Nebude vyžadovaná žiadna inštalácia systému u prevádzok ani u zákazníkov. Zákazníci rovnako ako aj zamestnanci prevádzok by mali byť schopný pracovať so systémom intuitívne, bez potreby špeciálneho školenia.

2.4.2 Zabezpečenie

Systém musí byť zabezpečený proti nebezpečným a nežiadúcim útokom, ako je napríklad SQL Injection, CSRF, XSS alebo Session Hijacking. HTTP hlavičky by mali posielať so správnou konfiguráciou.

Systém umožní prístup k funkcionalitám a datam len užívateľom s náležitým oprávnením.

Užívateľské heslo bude v systéme uložené v zahešovanej forme. Na zahashovanie hesla bude použitá Bcrypt hashovacia funkcia.[4–5]

2.4.3 Spoľahlivosť

Pri migrácii a/alebo aktualizácii databáze musí systém poskytnúť obnovu (takzvaný rollback) dát pri akomkoľkev zlyhaní procesu migrácie/aktualizávie databáze.

Systém musí fungovať bez problémov počas pracovnej doby prevádzok.

2.4.4 Prístupnosť

Systém musí byť prístupný 24/7 aby si zákazníci mohli kedykoľvek vytvoriť rezerváciu. Aktualizácie a nasadzovanie systému bude prebiehať výhradne v nočných hodinách, mimo pracovnú dobu prevádzok. Aktualizácie nesmie obmedziť prístupnosť systému pre prevádzky. Aktualizácie budú plánované a zákazníci ako aj prevádzky o nich budú informovaný upozornením (notifikáciou v aplikácii alebo emailom).

2.4.5 Kompatibilita prehliadačov

Predmetná aplikácia bude realizovaná ako webová aplikácia, preto deklarujeme minimálne verzie aplikáciou podporovaných webových prehliadačov, pri ktorých je zaručená funkčnosť systému. Jedná sa o Google Chrome verzia 70.0.3538 alebo novšia, Mozilla Firefox verzia 60.0 a novšie a Opera verzie 50 a vyššie. Použitie niektorého zo spomínaných prehliadačov zaručuje funkčnosť systému.

2.4.6 Škálovateľnosť

Systém musí obsluhovať tisíce až desiatky tisíc užívateľov bez akýchkoľvek problémov.

2.5 Prípady použitia

Prípad užitia popisuje zoznam krokov vykonaných aktérom v systéme k dosiahnutiu požadovaného cieľa. Aktérom je myslená externá entita ktorá príma určitú rolu v okamžiku, kedy systém začína používať.[6] Pri popisovaní užívateľských prípadov predpokladáme, že užívateľ je neprihlásený a nachádza sa na hlavnej stránke aplikácie.

2.5.1 Aktéri

V sekcií Funkčných požiadaviek podsekcia Role 2.3.2 sme definovali tri nasledníce role: *Prevádzka*, *Zákazník* a *Administrátor*. Na základe týchto rolí identifikujeme v systéme troch rôznych aktérov. Role popisujú aktérov systému, ktorý sú v aplikácii registrovaný, systém ale môžu rovnako používať aj neregistrovaný užívatelia. Týchto užívateľov označíme ako aktérov s názvom *Neregistrovaný Zákazník*. Dokopy máme teda štyroch aktérov: *Neregistrovaný Zákazník*, *Zákazník*, *Prevádzka* a *Administrátor*.

2.5.2 Základná časť systému

2.5.2.1 Registrácia

1. Používateľ zvolí registráciu.
2. Vyplní formulár registrácie. Formulár obsahuje polia pre užívateľské meno, e-mailovú adresu, telefónne číslo, heslo a potvrdenie hesla. V ďalšom kroku aktér môže (ale nemusí) vyplniť informácie o vozidle.
3. Užívateľ odošle vyplnený formulár.

2.5.2.2 Prihlásenie

1. Používateľ zvolí prihlásenie do systému.
2. Vyplní formulár pre prihlásenie. Formulár obsahuje polia pre užívateľské meno a heslo.
3. Odošle vyplnený formulár.

2.5.2.3 Vyhľadať prevádzky

1. Užívateľ vyplní filtrovacie kritéria (môže nechať prázdne).
2. Klikne na vyhľadávanie prevádzok.

2.5.2.4 Zobraziť detail prevádzky

1. Prípad použitia začína po vyhladani prevádzok.
2. Používateľ zvolí požadovanú prevádzku a klikne na zobrazenie detailu.

2.5.2.5 Zobraziť užívateľské informácie

1. Prípad použitia začína po prihlásení do systému.
2. Užívateľ zvoli v menu položku pre zobrazenie svojich informácii.

2.5.2.6 Upraviť užívateľské informácie

1. Prípad použitia začína po zobrazení užívateľských informácii.
2. Užívateľ klikne na tlačítko upravenia informácii.
3. Zobrazí sa formulár v ktorom užívateľ upraví požadované polia (ak je tieto polia možné meniť).
4. Uloží formulár.

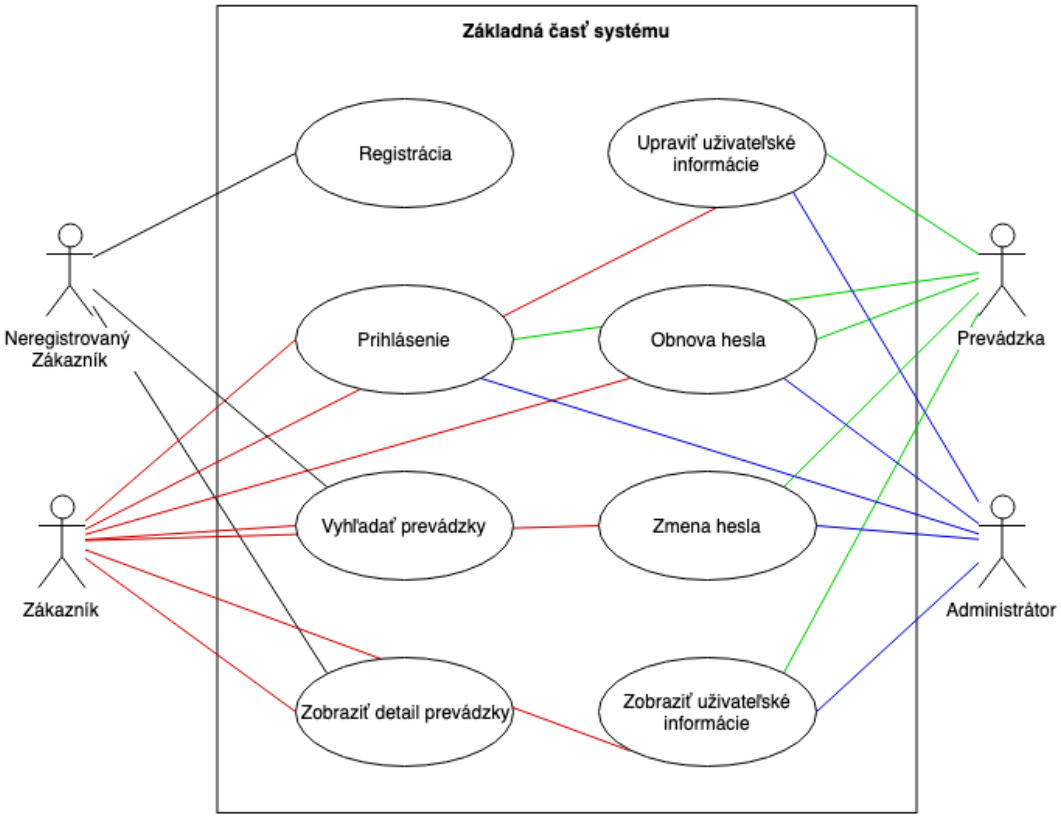
2.5.2.7 Zmena hesla

1. Prípad použitia začína po zobrazení užívateľských informácii.
2. Užívateľ klikne na tlačítko upravenia hesla.
3. Zobrazí sa formulár v ktorom užívateľ zadá stare heslo, nové heslo a potvrdenie nového hesla.
4. Odošle formulár.

2.5.2.8 Obnova hesla

1. Používateľ zvolí obnovu hesla.
2. Vyplní užívateľské meno, na ktorú mu bude odoslaný odkaz na obnovu hesla.

3. Otvorí odkaz z e-mailu, ktorý mu bol zaslaný systémom.
4. Vyplní polia pre nové heslo a potvrdenie nového hesla.
5. Odošle formulár.



Obrázok 2.2. Diagram prípadov použitia pre základnú časť systému.

2.5.3 Administračná časť systému

2.5.3.1 Registrovať prevádzku

1. Prípad použitia začína po prihlásení Administrátora do systému.
2. Administrátor zvolí v menu položku pre registráciu novej prevádzky.
3. Vyplní formulár s údajmi o prevádzke.
4. Odošle formulár.

2.5.3.2 Zmazať zákazníka

1. Prípad použitia začína po prihlásení do systému.
2. Ak je užívateľ prihlásený s rolou *Administrátor* tak
    1. Administrátor zvolí v menu položku pre zobrazenie zoznamu užívateľov.
    2. Nájde požadovaného zákazníka a zvolí zmazať.

3. Ak je užívateľ prihlásený s rolou *Zákazník* tak

    1. Užívateľ zvolí v menu položku pre zobrazenie svojich informácii.
    2. Klikne na tlačidlo vymazania účtu.

2.5.3.3 Zobraziť všetkých zákazníkov

1. Prípad použitia začína po prihlásení Administrátora do systému.
2. Administrátor zvolí v menu položku pre zobrazenie zoznamu zákazníkov.

2.5.3.4 Zobraziť všetkých prevádzok

1. Prípad použitia začína po prihlásení Administrátora do systému.
2. Administrátor zvolí v menu položku pre zobrazenie zoznamu prevádzok.

2.5.3.5 Zobraziť detail užívateľa

1. Prípad použitia začína po prihlásení do systému.
2. Ak je užívateľ prihlásený s rolou *Administrátor* tak

    1. Administrátor zvolí v menu položku pre zobrazenie zoznamu zákazníkov / prevádzok.
    2. Nájde požadovaného užívateľa a zvolí zobraziť detail.

3. Ak je užívateľ prihlásený s rolou *Zákazník* alebo *Prevádzka* tak

    1. Užívateľ zvolí v menu položku pre zobrazenie svojich informácii.

2.5.3.6 Zmazať prevádzku

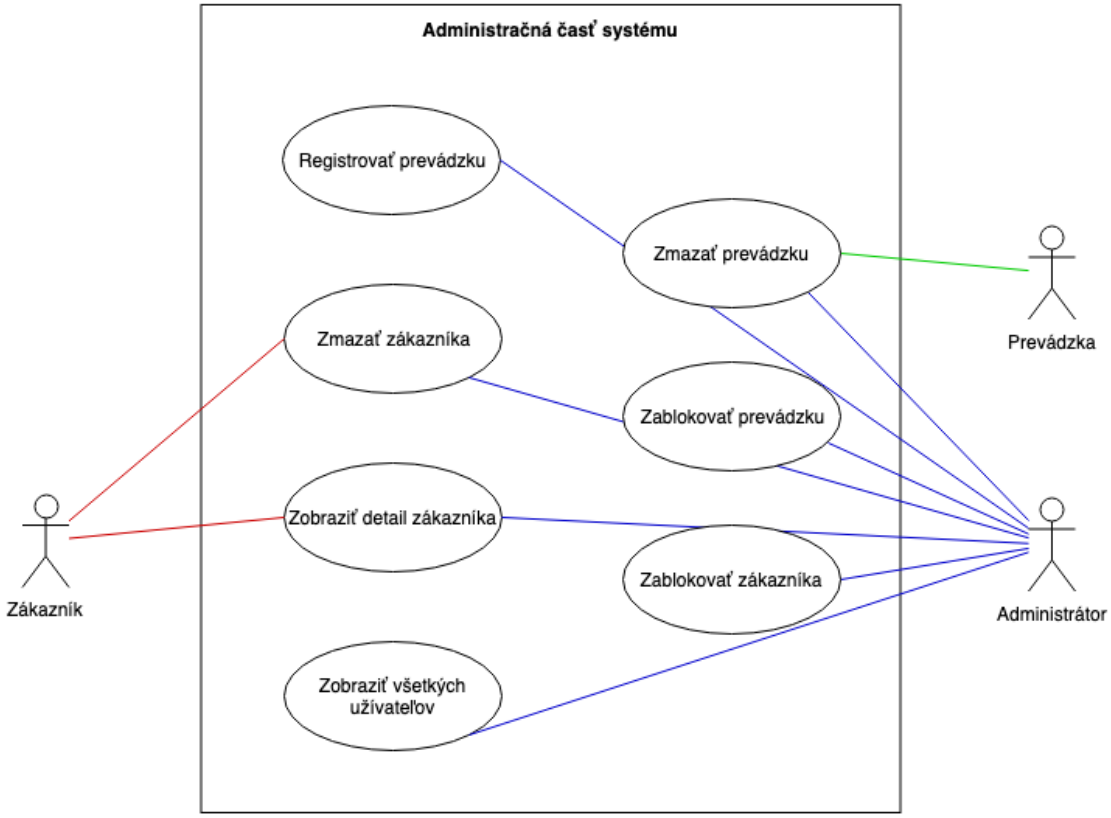
1. Prípad použitia začína po zobrazení všetkých prevádzok.
2. Administrátor nájde požadovanú prevádzku a zvolí zmazať.

2.5.3.7 Zablokovať zákazníka

1. Prípad použitia začína po zobrazení všetkých zákazníkov.
2. Administrátor nájde požadovaného zákazníka a zvolí zablokovať.

2.5.3.8 Zablokovať prevádzku

1. Prípad použitia začína po zobrazení všetkých prevádzok.
2. Administrátor nájde požadovanú prevádzku a zvolí zablokovať.



Obrázok 2.3. Diagram prípadov použitia pre administratívnu časť systému.

2.5.4 Rezervačná časť systému

2.5.4.1 Vytvoriť rezerváciu

1. Prípad použitia začína po zobrazení detailu prevádzky.
2. Zákazník klikne na tlačítko rezervácie.
3. Vyplní formulár rezervačný formulár.
    1. Prihlásený užívateľ vyplní polia pre dátum a čas, požadovaný úkon, zvolí auto na ktorom bude úkon vykonaný a má možnosť napísať poznámku pre prevádzku.
    2. Neprihlásený užívateľ musí navyše vyplniť aj informácie o aute, svoje celé meno, e-mailovú adresu a telefónne číslo.
4. Odošle formulár.

2.5.4.2 Zobraziť nadchádzajúce rezervácie

1. Ak má užívateľ rolu *Administrátor*.
    1. Prípad použitia začína po zobrazení detailu požadovaného užívateľa.

2. Ak má užívateľ rolu *Prevádzka* alebo *Zákazník*.

    1. Prípad použitia začína po prihlásení do systému.
    2. Užívateľ zvolí v menu položku pre zobrazenie rezervácii.

3. Zvolí nadchádzajúce rezervácie.

2.5.4.3 Zobraziť minulé rezervácie

1. Ak má užívateľ rolu *Administrátor*.

    1. Prípad použitia začína po zobrazení detailu požadovaného užívateľa.

2. Ak má užívateľ rolu *Prevádzka* alebo *Zákazník*.

    1. Prípad použitia začína po prihlásení do systému.
    2. Užívateľ zvolí v menu položku pre zobrazenie rezervácii.

3. Zvolí minulé rezervácie.

2.5.4.4 Zobraziť detail rezervácie

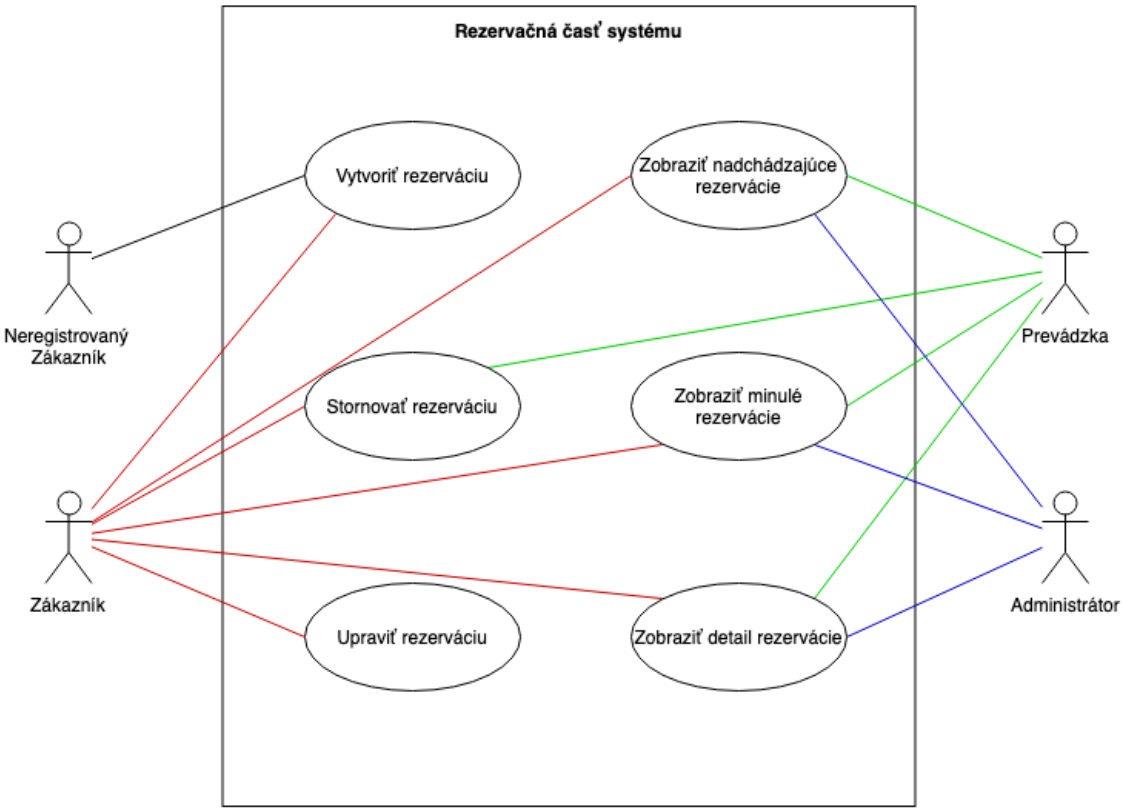
1. Prípad použitia začína po zobrazení nadchádzajúcich/minulých rezervácii.
2. V liste vyberie požadovanú rezerváciu a zvolí zobraziť detail.

2.5.4.5 Upraviť rezerváciu

1. Prípad použitia začína po zobrazení nadchádzajúcich rezervácii.
2. V liste vyberie požadovanú rezerváciu a zvolí upraviť.
3. Upraví informácie o rezervácii.
4. Uloží zmeny.

2.5.4.6 Stornovať rezerváciu

1. Prípad použitia začína po zobrazení nadchádzajúcich rezervácii.
2. V liste vyberie požadovanú rezerváciu a zvolí zrušiť.



Obrázok 2.4. Diagram prípadov použitia pre rezervačnú časť systému.

2.5.5 Časť systému týkajúca sa auta

2.5.5.1 Zobraziť zoznam vozidiel

1. Ak má užívateľ rolu *Administrátor*.
    1. Prípad použitia začína po zobrazení detailu požadovaného zákazníka.
    2. Administrátor zvolí zobrazenie áut zákazníka.
2. Ak má užívateľ rolu *Zákazník*.
    1. Prípad použitia začína po prihlásení do systému.
    2. Zákazník zvolí v menu položku pre zobrazenie vlastných áut.

2.5.5.2 Zobraziť detail vozidla

1. Ak je prihlásený užívateľ s rolou *Zákazník* alebo *Administrátor*
    1. Prípad použitia začína po zobrazení zoznamu vozidiel.
    2. Užívateľ nájde požadované vozidlo v zozname a zvolí zobraziť detail vozidla.
2. Ak je prihlásený užívateľ s rolou *Prevádzka*

1. Prípad použitia začína po zobrazení zoznamu budúcich rezervácii.
2. Užívateľ otvorí požadovanú rezerváciu.
3. Užívateľ zvolí zobrazenie detailu vozidla.

2.5.5.3 Upraviť informácie o vozidle

1. Prípad použitia začína po zobrazení zoznamu vozidiel.
2. Užívateľ nájde požadované vozidlo v zozname a zvolí upraviť vozidlo.
3. Upraví požadované informácie o vozidle.
4. Uloží zmeny.

2.5.5.4 Zmazať vozidlo

1. Prípad použitia začína po zobrazení zoznamu vozidiel.
2. Užívateľ nájde požadované vozidlo v zozname a zvolí zmazať.

2.5.5.5 Pridať vozidlo

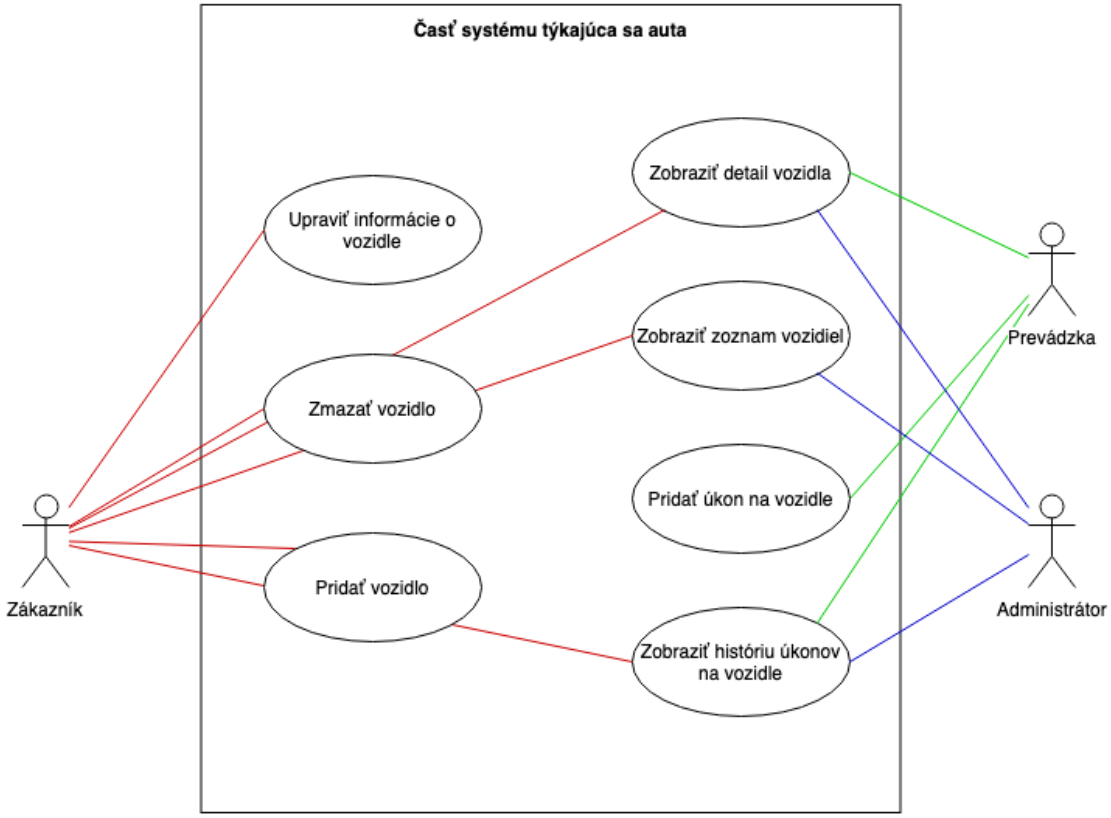
1. Prípad použitia začína po zobrazení zoznamu vozidiel.
2. Používateľ zvolí pridať nové vozidlo.
3. Vyplní formulár s informáciami o vozidle.
4. Odošle vyplnený formulár.

2.5.5.6 Pridať úkon na vozidle

1. Prípad použitia začína po prihlásení prevádzky do systému.
2. Prevádzka zvolí v menu položku pre zobrazenie rezervácii.
3. Vyberie minulé neuzavreté rezervácie.
4. Vyplní informácie o vykonanom úkone na vozidle.
5. Odošle formulár.

2.5.5.7 Zobraziť históriu úkonov na vozidle

1. Prípad užitia začína po zobrazení detailu vozidla.
2. Používateľ zvolí zobraziť históriu úkonov.



Obrázok 2.5. Diagram prípadov použitia pre časť systému týkajúcej sa aut.

2.5.6 Časť systému týkajúca sa recenzii

2.5.6.1 Vytvoriť recenziu

1. Prípad použitia začína po zobrazení minulých rezervácii.
2. Užívateľ nájde požadovanú rezerváciu a klikne na pridať recenziu.
3. Vyplní formulár pre recenziu.
4. Odošle vyplnený formulár.

2.5.6.2 Zobraziť zoznam vlastných recenzii

1. Prípad použitia začína po prihlásení užívateľa.
2. Užívateľ zvolí v menu položku pre zobrazenie zoznamu jeho recenzií.

2.5.6.3 Zobraziť zoznam recenzii prevádzky

1. Prípad použitia začína po zobrazení detailu prevádzky.
2. Detail prevádzky obsahuje recenzie o prevádzke.

2.5.6.4 Upraviť recenziu

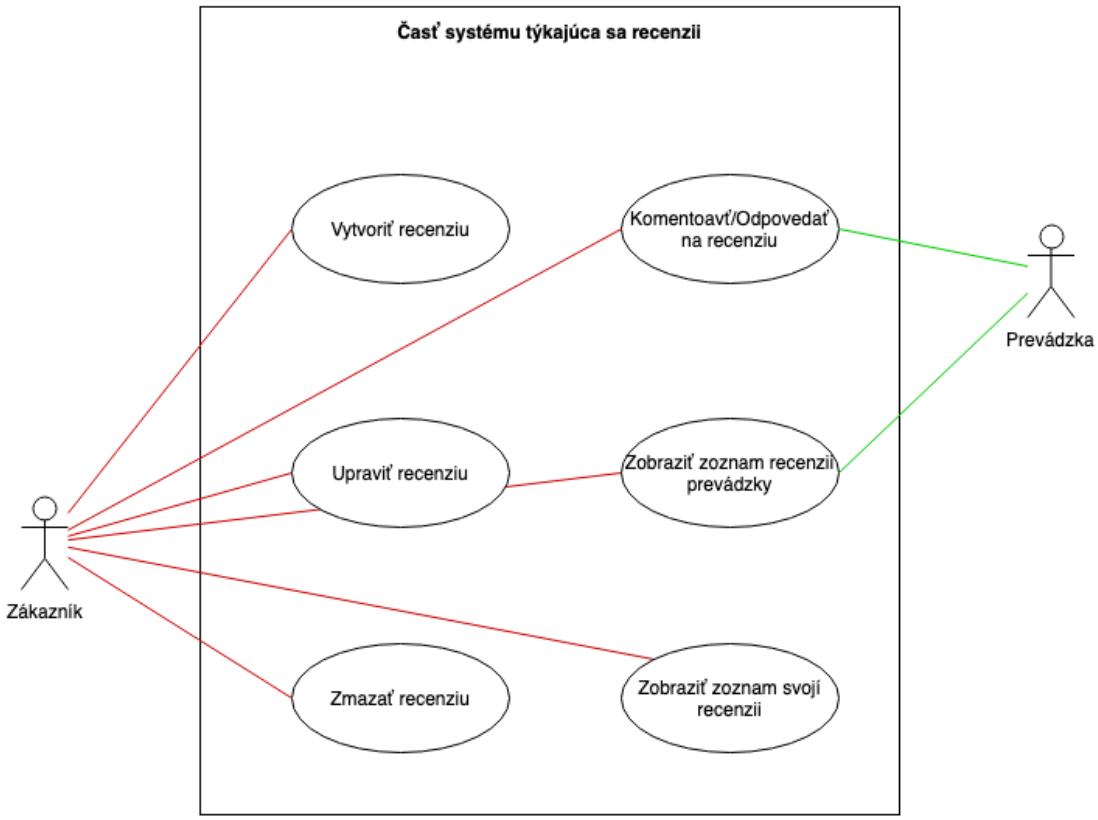
1. Prípad použitia začína po zobrazení zoznamu recenzií.
2. Užívateľ vyberie zo zoznamu požadovanú recenziu a zvolí upraviť.
3. Upraví požadované polia.
4. Uloží zmeny.

2.5.6.5 Zmazať recenziu

1. Prípad použitia začína po zobrazení zoznamu recenzií.
2. Užívateľ vyberie zo zoznamu požadovanú recenziu a zvolí zmazať.

2.5.6.6 Komentovať a Odpovedať na recenziu

1. Prípad použitia začína po zobrazení zoznamu recenzií.
2. Užívateľ vyberie zo zoznamu požadovanú recenziu a zvolí pridanie komentára/odpovedi.



Obrázok 2.6. Diagram prípadov použitia pre časť systému týkajúcej sa recenzii.

2.5.7 Časť systému týkajúca sa úkonov

2.5.7.1 Pridať nový úkon

1. Prípad použitia začína po zobrazení detailu užívateľa (prevádzky).
2. Používateľ zvolí pridať nový úkon.
3. Vyplní formulár s informáciami o úkone.
4. Odošle formulár.

2.5.7.2 Upraviť úkon

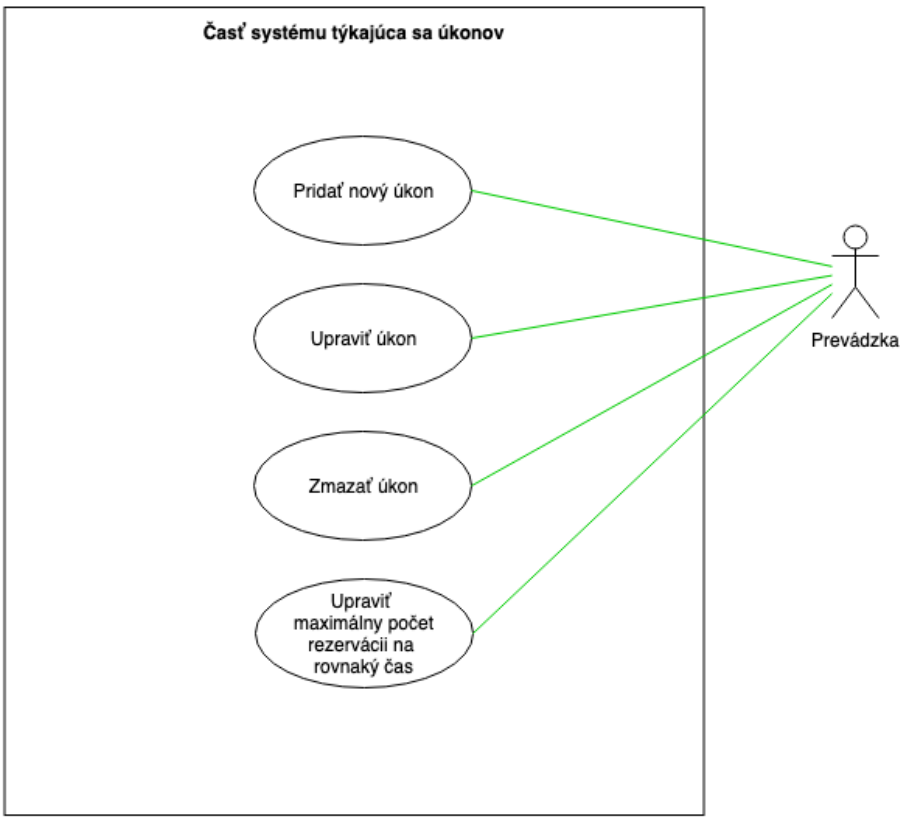
1. Prípad použitia začína po zobrazení detailu užívateľa (prevádzky).
2. Užívateľ vyberie zo zoznamu požadovaný úkonov a zvolí upraviť.
3. Upraví požadované informácie o úkone.
4. Uloží zmeny.

2.5.7.3 Zmazať úkon

1. Prípad použitia začína po zobrazení detailu užívateľa (prevádzky).
2. Užívateľ vyberie zo zoznamu požadovaný úkonov a zvolí zmazať.

2.5.7.4 Upraviť maximálny počet rezervácii na rovnaký čas

1. Prípad použitia začína po zobrazení detailu užívateľa (prevádzky).
2. Užívateľ upraví upraviť informáciu o maximálnom počte rezervácii na rovnaký čas.
3. Upraví počet.
4. Uloží zmeny.



Obrázok 2.7. Diagram prípadov použitia pre časť systému týkajúcej sa časti o úkonoch.

2.6 Analýza návrhu

Pred tým než sa pustíme do návrhu aplikácie a spracovania požiadaviek a užívateľských prípadov spracovaných v tejto kapitole, je potrebné zvážiť aký typ architektúry bude pre systém zvolený. Kniha Software Architecture Patterns s podtitulom Understanding Common Architecture Patterns and When to Use Them (čo by išlo do slovenčiny preložiť ako Vzory Softvérovej Architektúry, Porozumenie Bežným Architektonickým Vzorom a Kde ich Môžeme Použiť) od Marka Richardsa[7] popisuje a porovnáva päť vzorov architektúry softvéru a to Viacvrstvová Architektúra (ang. Layerd Architecture), Udalosťami Riadená Architektúra (ang. Event-Driven Architecture), Mikro-Jadrová Architektúra (ang. Microkernel Architecture), Mikro-Službová Architektúra (ang. Microservice Architecture Pattern) and Priestorovo Orientovaná Architektúra (ang. Space-based Architecture). Pri analýze spomínaných vzorov, sme vylúčili použitie niektorých vzorov, pretože hneď na prvý pohľad bolo vidieť prečo sa nehodia ako architektonický vzor pre našu aplikáciu.

Architektúra riadená udalosťami je prístup k tvorbe a návrhu softvérových systémov pomocou vytvárania, detekcie, spracovania a reakcii na udalosti. Aj keď sa v našej aplikácii udalosti budú vyskytovať tak ich asynchrónne spracovanie, ktoré sa pri implemen-

tácii tohto vzoru nachádza, nie je úplne žiaduce. Návrh a vývoj môže byť komplikovaný pretože komponenty spracujúce udalosti musia byť navrhnuté aby fungovali nezávisle na sebe. Rovnako je veľa aspektov ktoré treba pri návrhu brať na vedomie a počítať s nimi, to by mohol byť problém, pretože autor nemá tak veľa skúseností s návrhom systémov a mohol by niečo opomenúť.

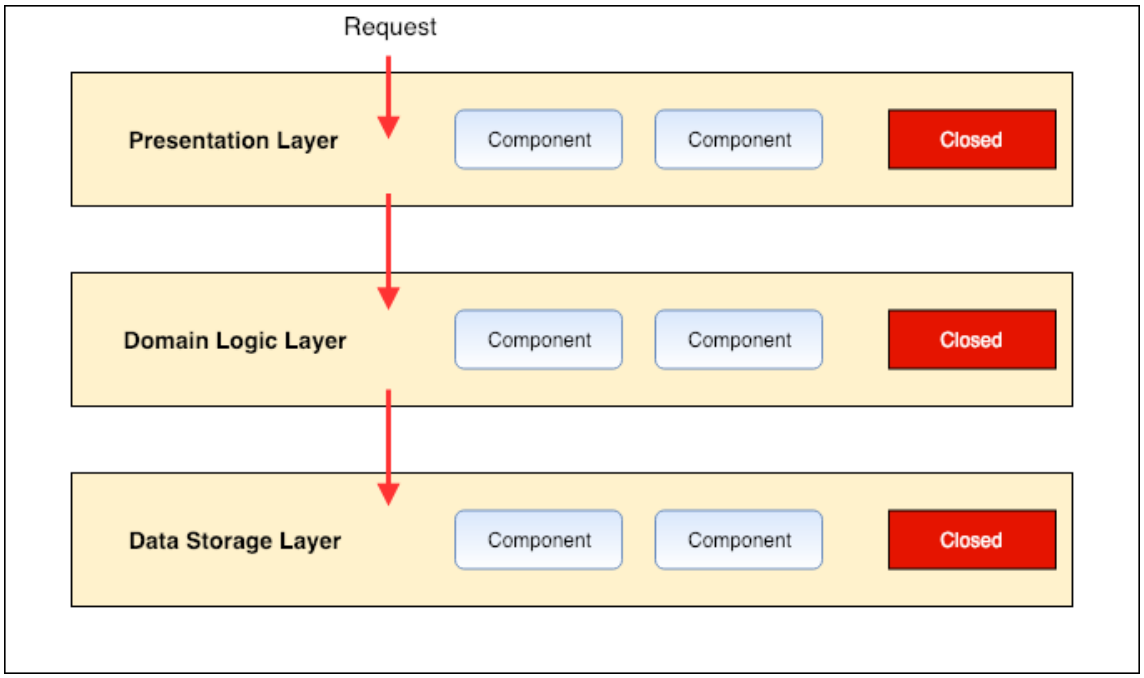
Microkernel architektúra je prirodzený vzor pre implementáciu natívnych aplikácii alebo aplikácii ktoré poskytujú pridávanie funkcionalít tretích strán, čo nie je náš prípad.

Priestorovo Orientovaná Architektúra (niekedy označovaná aj ako Cloudový Architektonický Vzor) by vďaka svojmu špecifickému návrhu riešiť problémy škálovateľnosti a súbežnému spracovávaniu mohla byť správna voľba, no absencia centralizovanej (alebo aspoň čiastočne centralizovanej) databáze nás odrádza od zvolenia takéhoto vzoru. V našej aplikácii potrebujeme uchovávať veľa informácii potenciálne počas celej existencie aplikácie a ich stratenie by bolo veľmi drahé.

Pri analýze sme preto podrobnejšie rozobrali dva vzory Viacvrstvová Architektúra a Architektúru Mikro-Služieb.

2.6.1 Layerd Architecture (*Viacvrstvová architektúra*)

Existuje niekoľko možných vzorov architektúr ktoré môžu byť pri návrh aplikácie použité. Pravdepodobne najpoužívanejšia je viacvrstvová architektúra[8] kde je aplikácia rozdelená do niekoľkých vrstiev ktoré sú fyzicky oddelené. Najčastejšie sa jedná o trojvrstvú architektúru obsahujúcu prezentačnú vrstvu, doménovo logickú vrstvu a vrstvu, zabezpečujúcu ukladanie dát.[9] Jednotlivé vrstvy sú organizované do horizontálnych úrovní, kde každá vrstva vykonána špecifickú úlohu. Jednotlivé vrstvy pozostávajú z komponent ktoré pracujú logikou ktorá náleží danej vrstve, čiže napríklad komponenta v prezentačnej vrstve obstaráva len prezentačnú logiku. Vrstvy sú označené ako uzavreté čo znamená že požiadavka sa predáva vždy len vrstve priamo pod ňou (viz obrázok 2.8). Rovnako existujú aj otvorené vrstvy ktoré umožňujú požiadavku prejsť vrstvou do ďalšej vrstvy bez akéhokoľvek spracovania.



Obrázok 2.8. Príklad uzavretých vrstiev a pristupu požiadavky.

2.6.1.1 Zváženie architektúry pre našu aplikáciu

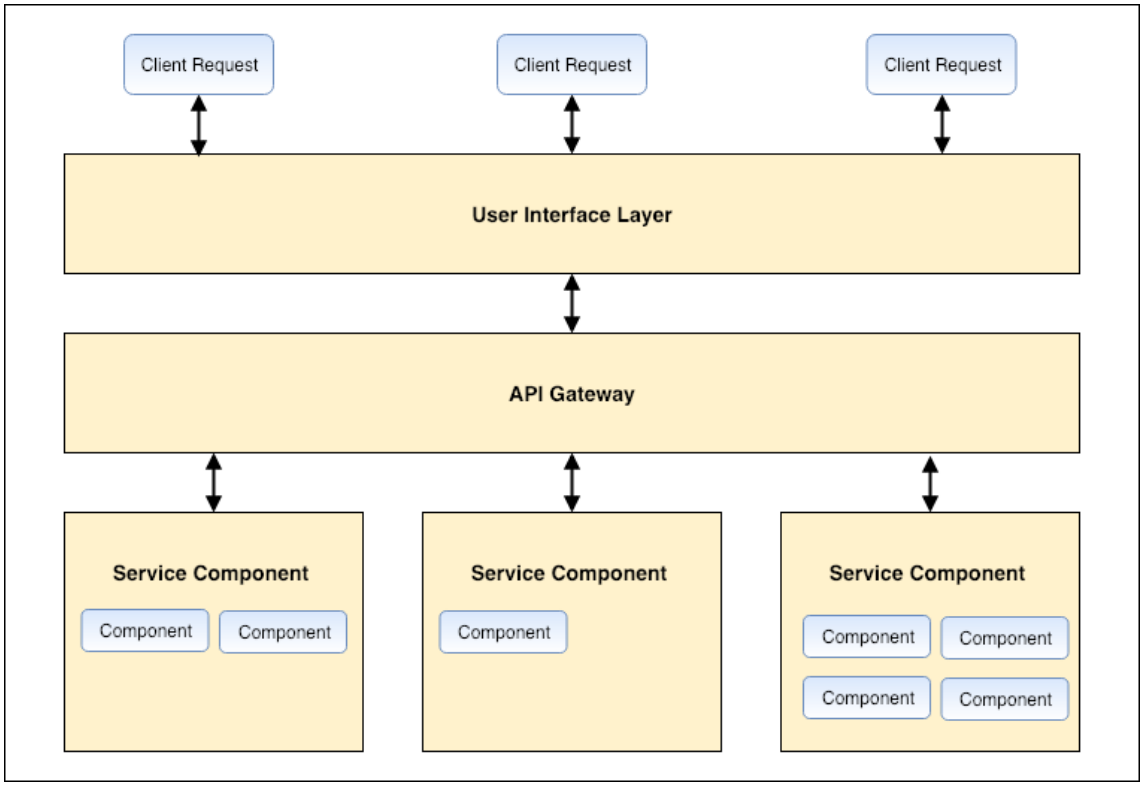
Výhody tohto vzoru sú že je pomerne jednoduchý na pochopenie a implementáciu, konzistentný naprieč rôznym projektom, garantuje oddelenie obáv a je z technického hladiska prehľadný.[10–11] Medzi ďalšie výhody tohto vzoru patrí jednoduchá testovateľnosť, kde je možné každú vrstvu samostatne testovať, závislosti alebo informácie z ostatných vrstiev je jednoduché mockovať (mokovať znamená imitovať chovanie určitého objektu, v našom prípade vrstvy).

Nevýhody viacvrstvovej architektury sú nízka škálovateľnosť keďže je aplikácia tvorená jednoliatou implementáciu, aj keď je možné fyzické rozdelenie jednotlivých vrstiev ktoré ale stále môžu byť veľké a drahé na škálovanie. Ďalšia nevýhoda je nižšia výkonnosť, pridáva na komplexnosti riešenia jednoduchým aplikáciám, pri dlhotrvajúcich projektoch náročné na udržiavanie a pridávanie nových funkcionalit, rovnako ako aj slabá schopnosť prispôsobovať sa stále meniacemu prostrediu.[12–13] V neposlednom rade je tu vysoká náročnosť na nasadzovanie nových verzií, kedy je potrebné odstaviť celú vrstvu (ak nie celú aplikáciu) pri nasadzovaní, a to môže byť uskutočňované len v hodinách kedy je systém nízko vyťažený, čo sú zväčša nočné hodiny.

Chceme aby naša aplikácia bola dobre a lahko testovateľná, čo nám tento štýl ponúka. Autor tejto práce je momentálne jediný programátor a tento vzor nám ponúka jednoduchú implementáciu čo môže ušetriť čas strávený implementáciou. Na druhú stranu aplikácia by mala byť jednoducho rozšíritelná, keďže má potenciál rásť a rozširovať poskytované funkcionality, čo môže byť náročnejšie pri zvolení tohto vzoru. Rovnako dôležitá je dobrá škálovateľnosť, keďže aplikáciu môžu používať aj desiatky tisíc ľudí, na čo nie je tento štýl úplne ideálny. Z aspektu nasadenia nových verzií by tento štýl nemusel byť komplikáciou, pretože sa nepredpokladá že by v nočných hodinách (kedy by nasadenie nových verzií mohlo byť realizované) bola vysoká vyťaženosť aplikácie.

2.6.2 Microservice Architecture (*Architektúra mikro služieb*)

Architektúra orientovaná na malé služby (microservices) je metóda vývoja softvéru ktorá sa sústredí na implementáciu viacerých samostatných navzájom nezávislých modulov, miesto implementácie velkej jednoliatej aplikácie.[14] Mikroservisy riešia výzvy jednoliatych systémov ako je napríklad škálovateľnosť, pridávanie nových funkcionalít alebo upravovanie stávajúcich častí kódu. Mikroservisy nemajú formálnu definíciu, ale každý systém založený na tomto štýle spĺňa niekoľko charakteristik.[15] Decentralizácia, čiže každá modul alebo jednotka je nasadená samostatne, rovnako môže mať vlastný data storage, čo zanemá vlastné miesto kam si ukladá dáta. Keďže jednotky nie sú navzájom závislé každá môže byť implementovaná v inom jazyku a operavosť s inou databázou. Je dôležité si uvedomiť že moduly sú združené v takzvaných "komponent služieb" (ang. service component). Tieto komponenty obsahujú jeden alebo viacero modulov a reprezentujú buď jedno-účelovú funkciu alebo časť väčšej biznis logiky.



Obrázok 2.9. Príklad jednoduchého návrhu micro-servisnej architektúry.

2.6.2.1 Zváženie architektúry pre našu aplikáciu

Výhodou použitia tejto architektúry je voľnosť ktorú prináša v nezávislosti vývoja a nasadenia jednotlivých servis komponent. Nezávislosť komponent umožňuje tvorenie malých tímov (dakedy kludne len o dvoch programátoroch) pre vývoj. Tento architektonický vzor umožňuje vysokú mieru agilnosti, čiže je možné rýchle prispôsobenie zmenám prostredia. Jednoduchá a vysoká testovateľnosť komponent. Zlepšuje izoláciu chýb, kedy chyba v jednej zo servis neochromý celý systém. Umožňuje škálovateľnosť, ak je jedna zo servis vyťaženejšia ako iné, tak je inštanciovaná viackrát, naopak ak je niektorá zo servis používaná málo, málo inštancií je potrebných čo šetrí zdroje. Umožnuje jednoduché nasadzovanie nových častí, poprípade oprav existujúcich komponent, kedy iba jedna komponenta je nasadená, zatiaľčo systém ako celok môže fungovať bez odstavenia.[16–17]

Nevýhodou však je samotný štýl neposkytuje zvýšenie výkonnosti ako celku. Systémové a Akceptančné testovanie aplikácie môže byť pomerne náročné. Samotný návrh systému je náročný, pritom je veľmi dôležité aby boli správne identifikované nezávislé časti a tie správne rozdelené do komponent. Ak servisy používajú časť rovnakej funkcionality, môže vznikať duplicitný kód čo porušuje princíp DRY (z ang. Don't Repeate Yourself, v preklade Neopakuj Sa).[18]

Pre návrh našej aplikácie nám tento architektonický vzor vyhovuje z hľadiska jednoduchého unit testovania, bohužiaľ systémové testovanie môže byť pomerne výzva. Keďže chceme aby bol systém funkčný ideálne bez prerušenia, izolace chýb a jednoduchosti a nezávislosti nasadzovania komponent nám v tomto prípade splňuje túto podmienku. Aplikácia má potenciál rásť a rozširovať poskytované funkcionality a služby, čo je s týmto prístupom poskytnuté zadarmo. Aplikácia vyžaduje dobrú škálovateľnosť, čo tento štýl poskytuje.

Kapitola 3
Návrh

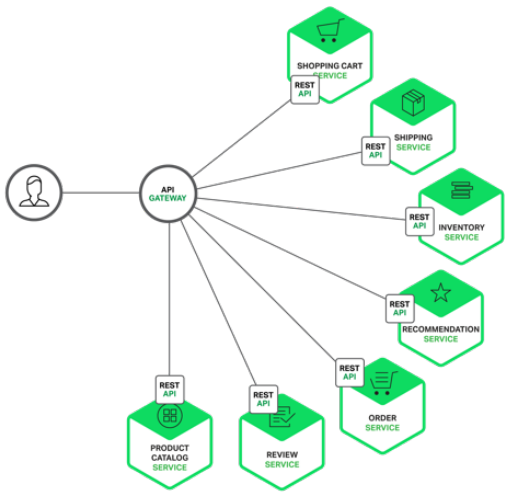
3.1 Architektúra systému

Pre návrh back-endu aplikácie bola zvolená Microservice Architecture. Táto architektúra bola zvolená z hľadiska dlhodobej stratégie. Systém vyvíjaný pre spoločnosť DriveCare s.r.o. má potenciál rozvíjať sa a rozširovať ponúkané služby a funkcionality, čo zvolená architektúra umožňuje veľmi jednoducho. Rovnako je táto aplikácia niečím novým na Slovenskom trhu, preto sa nevie aké je prostredie a čo klienti budú poprípade nebudú chcieť a potrebovať. Toto sú zmeny prostredia na ktoré vie zvolená architektúra agilne a dynamicky reágovať.

Pre frontend je zvolená MVC architektúra. Dôvodom použitia tohto štýlu je veľká podpora existujúcich javascriptových rámec ako je Angular, čo uľahčuje prácu a zrýchľuje vývoj.

3.2 Mikroslužby

Serverová časť aplikácie bude tvorená súborom komponent, ktoré budú pracovať samostatne bez ohľadu na funkčnosť a dostupnosť ostatných komponent. Každá komponenta si bude sama riadiť pripojenia k úložisku dát, rovnako bude poskytovať REST endpointy, komunikovať s API Gateway a samozrejme obstarávať logiku pre jednotku určenú. API Gateway je vrstva alebo server ktorý slúži ako jednotný vstupný bod do systému, čiže jeden bod do ktorého prichádzajú požiadavky z vonku (všade mimo systém) a sú presmerované na jednotlivé komponenty.[19]



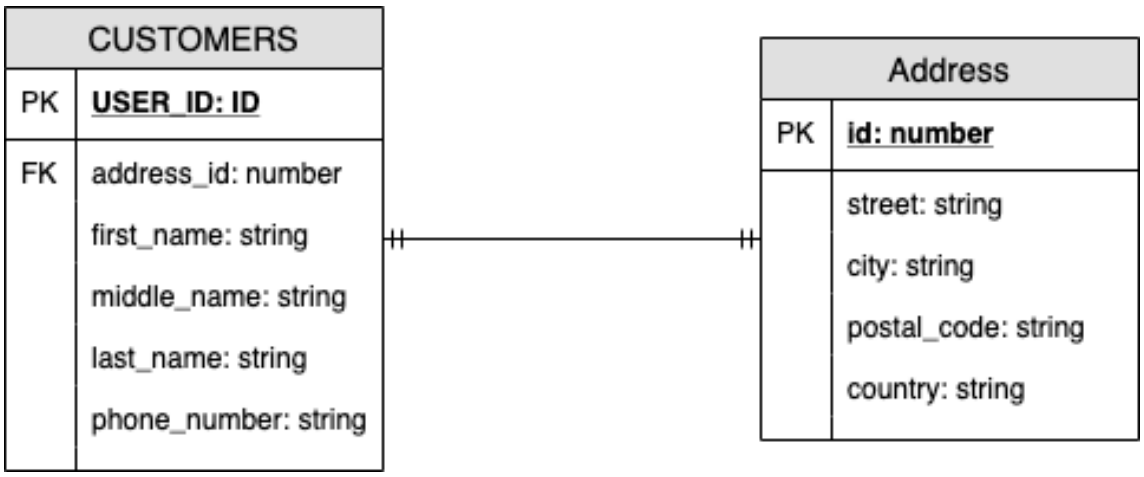
Obrázok 3.1. Príklad znázornenia na čo slúži API gateway, zdroj [20].

3.2.1 Správa zákazníkov

Služba zabezpečujúca správu zákazníkov obstaráva ukladanie a spracovanie informácii o zákazníkoch. Údaje o zákazníkoch sú uložené v relačnej databáze. Služba poskytuje REST API, na ktoré sa pripája gateway služba. Každý endpoint obsahuje prefix '/customers'.

URI	Žiadosť	Oprávnenie	Poznámka
'/find-all'	GET	Zobraziť všetkých zákazníkov	Zoznam všetkých zákazníkov v systéme.
'/detail'	GET	Zobraziť zákazníka	Informácie o zákazníkovi.
'/update'	PUT	Upraviť zákazníka	Aktualizovanie informácii o zákazníkovi.
'/register'	POST	*Bez nutnosti oprávnenia*	Registrácia nového zákazníka.
'/delete'	DELETE	Zmazať užívateľa	Zmazanie zákazníka.

Tabuľka 3.1. Endpointy poskytované službou správa zákazníkov.



Obrázok 3.2. Dátový model Správy Zákazníkov.

3.2.2 Správa vozidiel

Služba je zameraná na ukladanie a spracovanie informácii o autách. Táto služba poskytuje pridávanie nových áut alebo aktualizovanie informácii o autách ktoré sú už v databáze uložené. Tak isto poskytuje informácie o úkonoch, ktoré boli vykonané na aute. Tieto úkony boli pridané prevádzkou, ktorá úpravy, opravy alebo údržbu na vozidle vykonala. Dáta sú uložené v dokumentovej NoSql databáze. Každý endpoint obsahuje prefix '/vehicles'.

```
{
    "id": "string",
    "userId": "string",
    "info": {
        "category": "string",
        "brand": "string",
        "model": "string",
        "buildIn": "string",
```

URI	Žiadosť	Oprávnenie	Poznámka
’/vehicles/find-all’	GET	Zobraziť vozidlá	Zobraziť zoznam vozidiel.
’/vehicles/create’	POST	Vytvoriť vozidlo	Pridať zákazníkovi vozidlo.
’/vehicles/detail’	GET	Zobraziť vozidlá	Zobraziť detail vozidla.
’/vehicles/update’	PUT	Upraviť vozidlo	Upraviť informácie o vozidle.
’/vehicles/delete’	DELETE	Zmazať vozidlo	Zmazať zákazníkovi vozidlo.
’/vehicles/task/create’	POST	Pridať úkon vozidlu	Pridať servisný úkon na vozidle.
’/vehicles/task/find-all’	GET	Zobraziť históriu úkonov vozidla	Zobraziť servisné úkony na vozidle.

Tabuľka 3.2. Endpointy poskytované službou správa vozidiel.

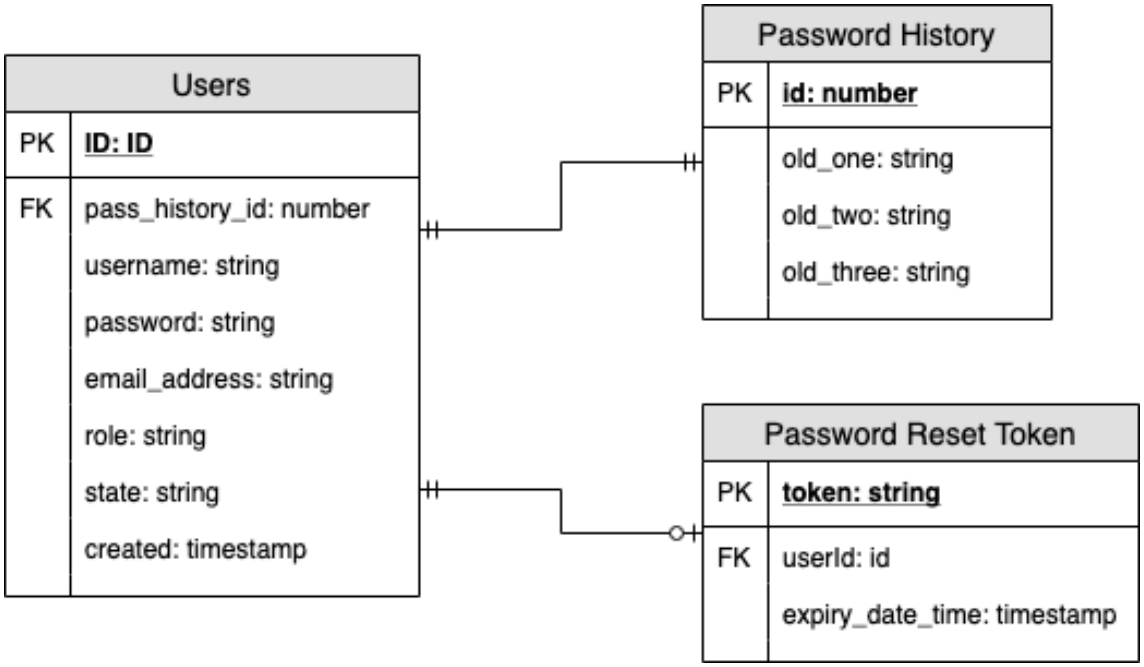
```
        "fuel": "string",
        "power": "string",
        "vin": "string",
        "vrp": "string",
        "transmission": "string",
        "bodywork": "string",
        "drive": "string"
    },
    "created": "timestamp"
}
```

3.2.3 Autentifikačná služba

Táto služba zabezpečuje autentifikáciu užívateľov do systému. Služba uchováva informácie o užívateľoch potrebné pre prihlásenie a to užívateľské meno, heslo, primárna emailovú adresu a rolu užívateľa. Každý endpoint tejto služby má prefix ’/auth’.

URI	Žiadosť	Oprávnenie	Poznámka
’/’	POST	*Bez nutnosti oprávnenia*	Prihlásenie.
’/register/customer’	POST	*Bez nutnosti oprávnenia*	Registrácia zákazníka.
’/register/shop’	POST	Registrovať prevádzku	Registrácia prevádzky.
’/change-password’	POST	Zmeniť heslo	Zmena hesla.
’/reset-password’	POST	*Bez nutnosti oprávnenia*	Obnova hesla.
’/delete’	DELETE	Zmazať užívateľa	Vymazanie užívateľa.
’/to-delete’	DELETE	K vymazaniu zákazníka	K zmazaniu zákazníka.
’/block’	PUT	Zablokovať zákazníka	Zablokovať užívateľa.

Tabuľka 3.3. Endpointy poskytované službou Autentifikačný Servis.



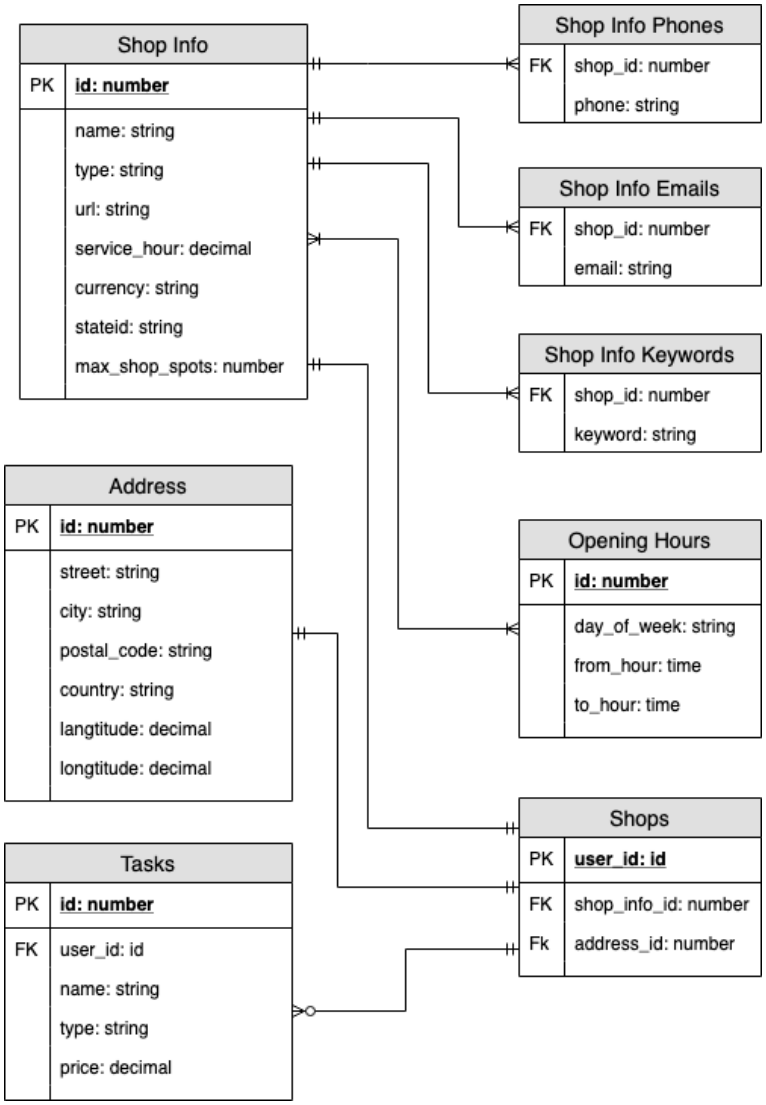
Obrázok 3.3. Dátový model autentifikačného servisu.

3.2.4 Správa prevádzok

Služba je zameraná na ukladanie a spracovanie informácii o prevádzkach. Táto služba poskytuje registráciu nových prevádzok, aktualizovanie informácii o nich, vyhľadávanie požadovaných prevádzok na základe kritérii a zobrazovanie informácii o prevádzke. Ďalej služba poskytuje správu servisných úkonov a to ich pridávanie, upravovanie alebo mazanie podľa potrieb prevádzky. Každý endpoint obsahuje prefix '/shops'.

URI	Žiadosť	Oprávnenie	Poznámka
'/shops/update'	PUT	Aktualizovať prevádzku	Aktualizovanie informácii o prevádzke.
'/register'	POST	Registrovať prevádzku	Registrácia prevádzky.
'/delete'	DELETE	Zmazať užívateľa	Zrušenie prevádzky.
'/task/create'	POST	Pridať servisný úkon	Pridať nový servisný úkon.
'/task/update'	PUT	Upraviť servisný úkon	Upraviť servisný úkon.
'/task/delete'	DELETE	Upraviť servisný úkon	Zmazať servisný úkon.
'/find-all'	GET	*Bez nutnosti oprávnenia*	Vyhladanie prevádzky.
'/detail'	GET	*Bez nutnosti oprávnenia*	Zobraziť detail prevád

Tabuľka 3.4. Endpointy poskytované službou správa prevádzok.



Obrázok 3.4. Dátový model Správy prevádzok

3.2.5 Správa rezervácii

Služba je zameraná na ukladanie a spracovanie informácii o rezerváciách. Táto služba poskytuje možnosť vytvoriť, upraviť alebo zmazať rezerváciu. Dáta sú uložené v dokumentovej NoSql databáze. Každý endpoint obsahuje prefix '/reservations'.

URI	Žiadosť	Oprávnenie	Poznámka
'/create'	POST	*Bez nutnosti oprávnenia*	Vytvoriť rezerváciu.
'/get-coming'	GET	Zobraziť rezervácie	Zobraziť nadchádzajúce rezervácie.
'/get-past'	GET	Zobraziť rezervácie	Zobraziť minulé rezervácie.
'/detail'	GET	Zobraziť rezervácie	Zobraziť detail rezervácie.
'/update'	PUT	Upraviť rezerváciu	Upraviť rezerváciu.
'/delete'	DELETE	Zrušiť rezerváciu	Stornovať rezerváciu.

Tabuľka 3.5. Endpointy poskytované službou správa rezervácii.

```
{
    "id": "objectId",
    "shopId": "string",
    "customerId": "string",
    "spotNumber": "number",
    "created": "timestamp",
    "startTime": "timestamp",
    "estimatedEndTime": "timestamp",
    "createdBy": "string",
    "status": "string",
    "reservationDetail": {
        "customerDetail": {
            "firstName": "string",
            "middleName": "string",
            "lastName": "string",
            "phone": "string",
            "vehicleInfo": {
                "category": "string",
                "brand": "string",
                "model": "string",
                "buildIn": "string",
                "fuel": "string",
                "power": "string",
                "vin": "string",
                "vrp": "string",
                "transmission": "string",
                "bodywork": "string",
                "drive": "string"
            }
        },
        "shopDetail": {
            "name": "string",
            "phones": {
                "list": "string"
            },
            "emails": {
                "list": "string"
            },
            "address": {
                "street": "string",
                "city": "string",
                "postalCode": "string",
                "country": "string"
            },
            "type": "string",
            "maxShopSpots": "number"
        },
        "task": {
            "name": "string",
            "type": "string",
            "price": "decimal",
            "executionTime": "time"
        },
```

```
        "actualEndTime": "timestamp",
        "customerNote": "string",
        "shopNote": "string",
        "estimatedPrice": {
            "price": "string",
            "currency": "string"
        }
    }
}
```

3.2.6 Správa recenzií

Služba je zameraná na ukladanie a spracovanie informácii o recenziách. Táto služba poskytuje možnosť vytvoriť, upraviť alebo zmazať recenziu. Dáta sú uložené v dokumentovej NoSql databáze. Každý endpoint obsahuje prefix '/reviews'.

URI	Žiadosť	Oprávnenie	Poznámka
'/reviews/create'	POST	Vytvoriť recenziu	Vytvoriť recenziu.
'/reviews/find-all'	GET	Zobraziť recenzie	Zoznam recenzii.
'/reviews/update'	PUT	Upraviť recenziu	Upraviť recenziu.
'/reviews/delete'	DELETE	Upraviť recenziu	Zmazať recenziu.
'/reviews/comment/create'	POST	Komentovať recenzie	Pridať komentár k recenzii.

Tabuľka 3.6. Endpointy poskytované službou správa recenzií.

```
{
    "id": "objectId",
    "parentCommentId": "objectId",
    "shopId": "string",
    "customerId": "string",
    "created": "timestamp",
    "note": "string",
    "rate": "number"
}
```

3.2.7 Gateway

Gateway poskytuje endpointy ktoré sú viditeľné pre front endovú časť aplikácie. Úlohou gateway v spolupráci s load balancérom je load balancer a presmerovanie dotazov na správnu službu. Všetky dotazy z vonku na systém sú spracované touto gateway, ktorá kontroluje token a filtruje dotazy na základe oprávnení užívateľa. Každý endpoint obsahuje prefix '/api'.

3.3 Grafické užívateľské rozhranie

Pri návrhu užívateľského rozhrania predpokladáme, že aplikáciu bude používaná primárne cez počítač. Tým pádom nie je grafické prostredie zamerané na používanie v malých zariadeniach ako telefón alebo tablet.

3.3.1 Rozdelenie užívateľského rozhrania

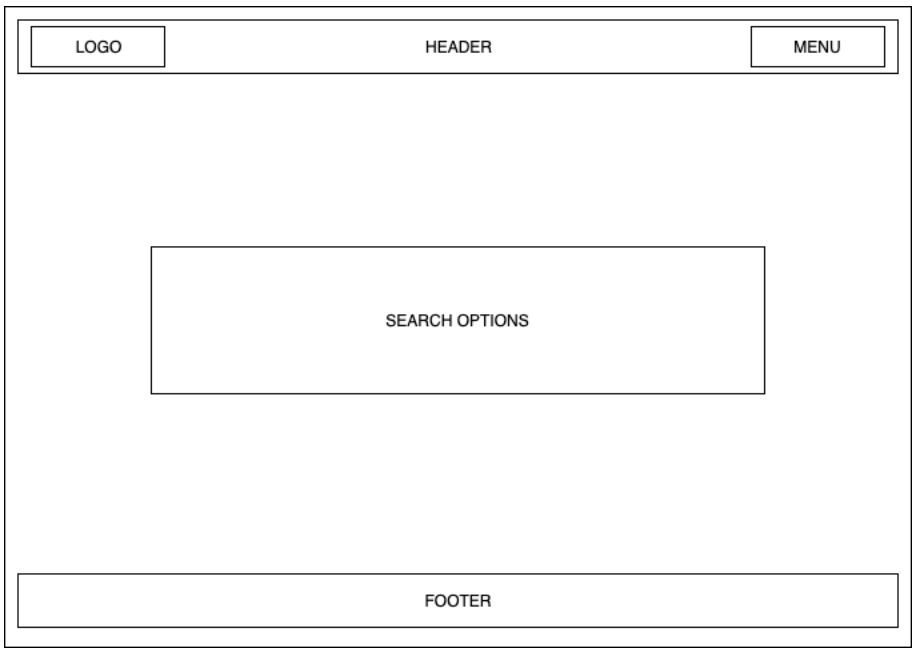
Užívateľské rozhranie bude rozdelené do štyroch častí, na základe toho, aký typ užívateľa k aplikácii pristupuje. Všetky tieto časti majú spoločnú jednu časť a tou je stránka prihlásenia, ktorá bude obsahovať len veľmi jednoduchý formulár s poliami pre zadanie užívateľského mena a hesla.

3.3.1.1 Základná časť

Prvá časť, poskytujúca zákazníkom vyhladanie, vyberanie a rezervovanie prevádzky, bude dostupná neprihláseným užívateľom a prihláseným užívateľom s rolou *Zákazník*. Základná časť rovnako poskytuje ešte stránku registrácie, ktorá bude dostupná iba neprihláseným užívateľom.

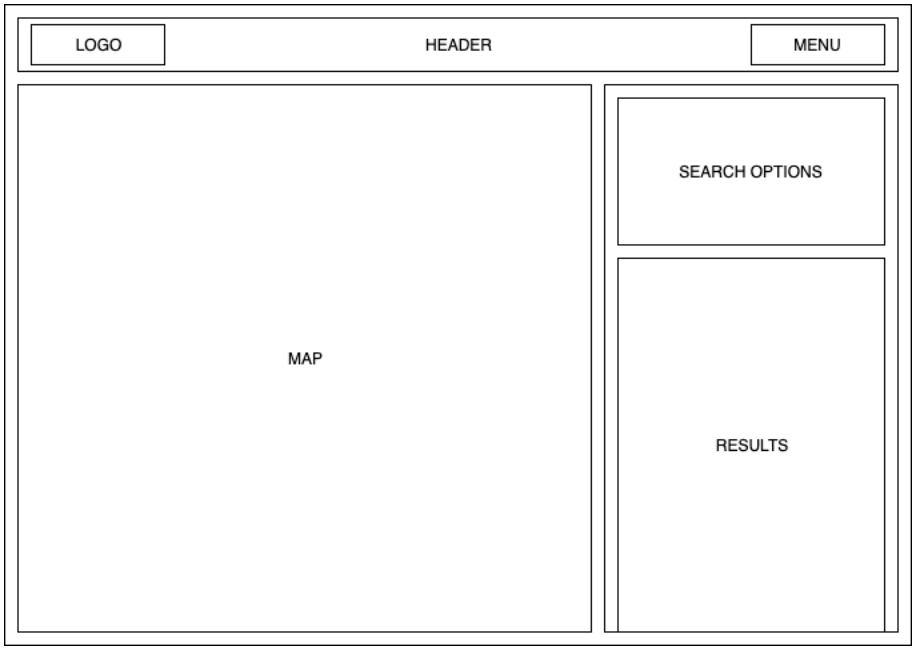
Stránka registrácie bude obsahovať registračný formulár, pre zadanie informácii o zákazníkovi rovnako ako aj informácie o vozidle zákazníky.

Súčasťou základnej časti je domovská stránka, ktorej rozloženie je znázornené na obr. (3.5).



Obrázok 3.5. Domovská stránka.

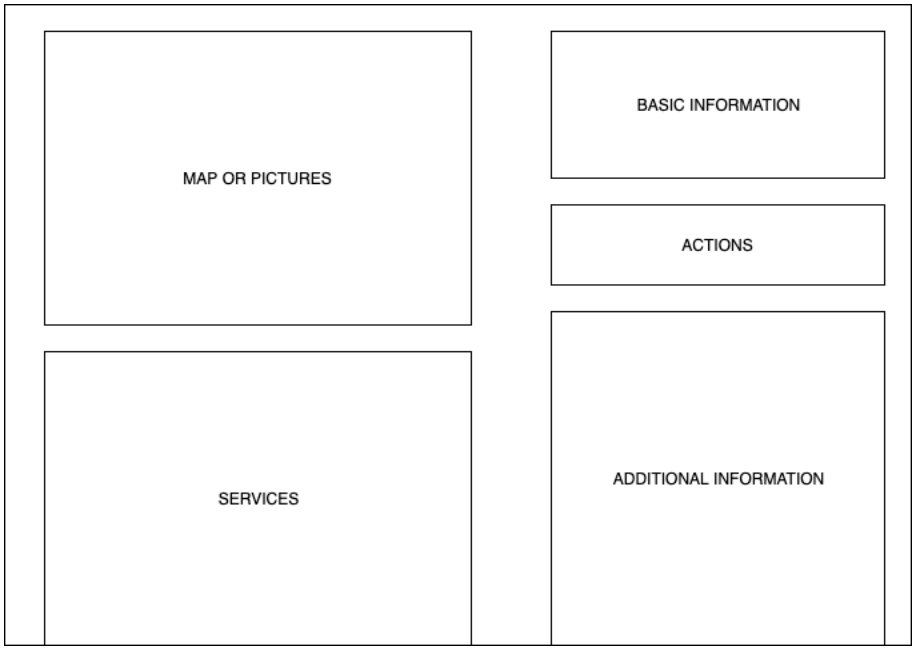
Ďalej stránku s mapou a zoznamom dostupných prevádzok spĺňajúce filtrovacie kritéria na obr. (3.6).



Obrázok 3.6. Stránka vyhľadávanie prevádzok.

Užívatelia si tiež budú môcť zobraziť detail prevádzky na samostatnej stránke, ktorej rozloženie je zobrazené na obr. (3.7).

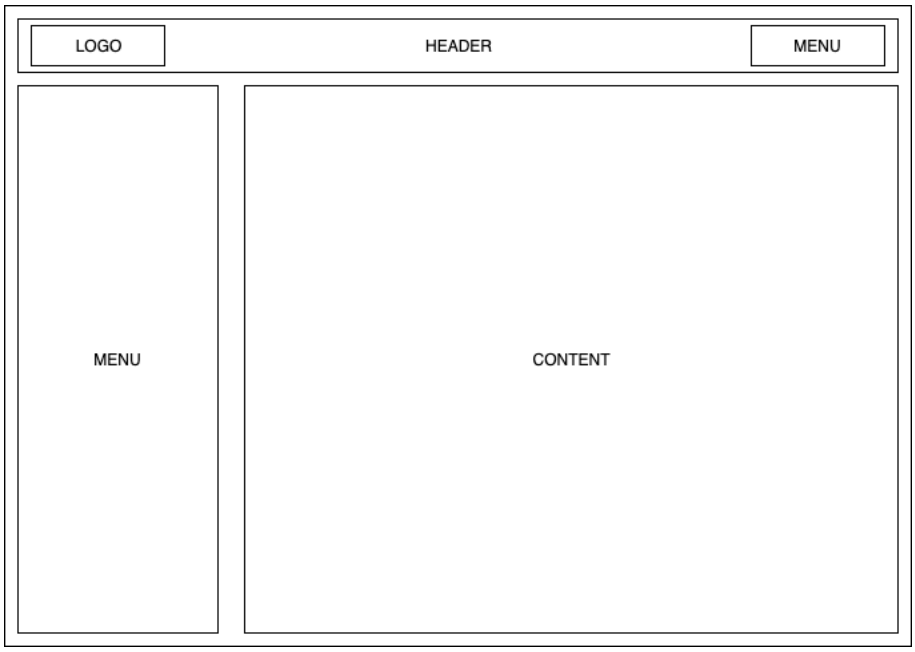
Posledná stránka dostupná aj neprihláseným užívateľom bude formulár k vytvoreniu rezervácie. Formulár bude viackrokový, kde sa v jednotlivých krokoch vyplnia rôzne údaje. Najprv si užívateľ zvolí službu ktorú by si chcel rezervovať. V ďalšom kroku, sa mu na základe vybranej služby, vygeneruje rezervačný kalendár, kde si vyberie požadovaný termín. V predposlednom kroku bude užívateľ požiadaný o zadanie osobných údajov ako informácii o aute, pre registrovaného a prihláseného užívateľa budú tieto údaje predvyplnené, poprípade vyberie požadované vozidlo ak má zákazník registrovaných viacero vozidiel. Posledný krok formulára bude vyžadovať súhlas so spracovaním osobných údajov a samotné odoslanie rezervácie.



Obrázok 3.7. Detail prevádzky.

3.3.1.2 Zákaznícke nastavenia

Druhá časť užívateľského rozhrania, pracovne nazývaná *Zákaznícke nastavenia*, bude prístupná len prihláseným užívateľom s rolou *Zákazník*. V tejto časti budú môcť užívatelia spravovať svoje osobné údaje a informácie o účte, vozidlá a rezervácie (spravovať nadchádzajúce alebo prehliadať minulé). Rozloženie stránky v časti *Zákaznícke Nastavenia* je 'Dashboard Layout' (svk. Panelové Rozdelenie) a môžeme ho vidieť na obr. (3.8). Každá odkaz v navigačnom menu odkazuje na formulár, kde je možné informácie čítať, upravovať prípadne pridávať alebo mazať.



Obrázok 3.8. Dashboard rozloženie.

3.3.1.3 Administratívna prevádzok

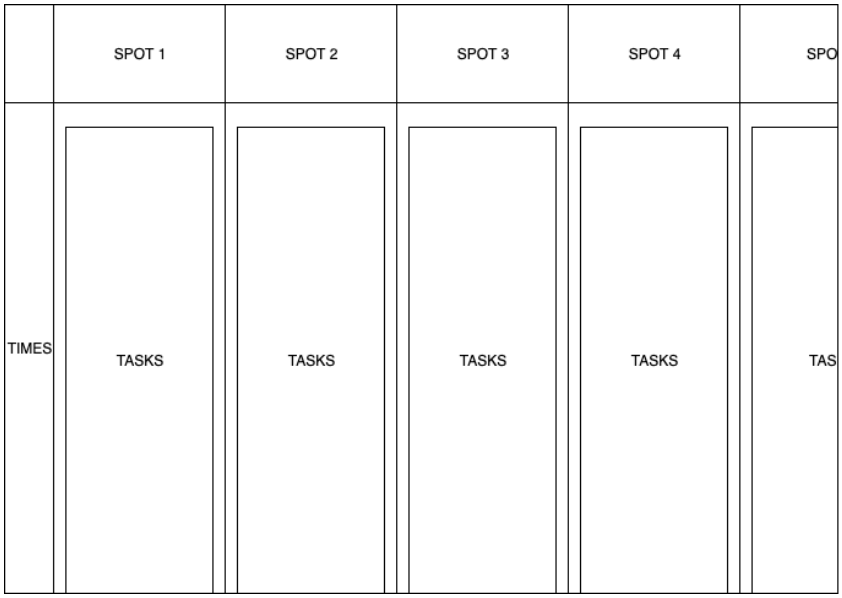
Administratíva prevádzok bude prístupná len prihláseným užívateľom s rolou *Prevádzka*.

Podobne ako predchádzajúca časť systému (*Zákaznícke nastavenia*), aj táto časť systému má rozloženie 'Dashboard Layout'. V tejto časti užívateľského rozhrania môžu prevádzky spravovať informácie o svojej prevádzke a účte, poskytované úkony, ďalej tu vidia prehľad rezervácii (minulý a budúcich), rovnako môžu vytvárať nové rezervácie, vidieť a odpovedať na recenzie od zákazníkov a v neposlednom rade spravovať rezervácie.

Jedna z najdôležitejších záložiek pre prevádzky je prehľad rezervácii. Každá prevádzka má inú maximálnu kapacitu, jednu jednotku tejto kapacity sme si pracovne pomenovali zdvihák[1][2]. Prehľad rezervácii bude vo forme rozvrhu pre jeden deň (zobrazený na obr. (3.9)), kde sa v samostatnych stĺpcoch zobrazujú rezervácie ku konkrétnym zdvihákom. Rozvrh je rozdelený diagonálne podľa hodín, čo je považovaná za najmenšiu jednotku pri rezervácií úkonu. Maximálna kapacita je dynamická, čiže aj rozvrh musí byť dynamický a meniť sa na základe toho, aký počet zdvihákov daná prevádzka má.

[1] Pretože veľká množina prevádzok budú autoservisy a pneuservisy a počet zdvihákov im určuje maximálnu kapacitu počtu zákazníkov ktorých môžu v jeden čas obsluhovať).

[2] Zdvihák nemusí byť nutne zdvihák, napríklad v prípade autoumyvárne sa jedná o jednu umývačku.



Obrázok 3.9. Rozvrh s rezerváciami.

Záložka s informáciami o prevádzke, rovnako ako aj záložka s údajmi o účte, bude obsahovať formulár, ktorý umožní tieto údaje čítať prípadne upraviť.

Záložka s úkonmy, bude obsahovať zoznam úkonov poskytovaných prevádzkov, bude poskytovať možnosť pridať nový úkon, upraviť informácie o existujúcom úkone, prípadne zmazať úkon.

Jeden z odkazov v bočnom menu poskytuje možnosť vytvorenia rezervácie, kedy sa zobrazí rovnaký formulár ako je popísané v sekcii 3.3.1 stránka vytvorenia rezervácie.

Posledná nespomenutá záložka zobrazí zoznam zákazníckych recenzií k prevádzke, prevádzka bude môcť čítať tieto recenzie a bude môcť odpovedať na ne.

3.3.1.4 Administrátorská časť

Administrátorská časť aplikácie bude dostupná len prihláseným užívateľom s rolou *Administrátor*.

V tejto časti bude môcť administrátor systému spravovať užívateľov, čiže ich mazať, prípadne blokovať. Rovnako bude môcť registrovať nové prevádzky do systému. Bude mať oprávnenie na čítanie informácii o prevádzkach, zákazníkoch, čítanie rezervácií a recenzií.

Rozloženie užívateľského rozhrania v administrátorskej časti bude rovnako ako v ostatných administračných častiach 'Dashboard Layout'.

3.3.2 Farebné schéma aplikácie

Hlavné pozadie stránok bude tvorené bielou farbou (#FFFFFF). Primárnou farbou palety je modrá (#1565C0) v svetlom odtieni (#E3F2FD) a tmavom odtieni (#0D47A1). Sekundárnou farbou je žltá (#FDD835) v svetlom odtieni (#FFEE58) a tmavom odtieni (#F9A825). Farby majú svoje odtiene na zvýraznenie prvkov pri kombinácii primárnej a sekundárnej farby. Pre upozornenia a výstrahy je použitá červená farba (#F44336). Text na svetlom pozadí je z pravidla čiernou farbou (#000000) alebo svetlo šedou (#9E9E9E) prípadne na tmavom pozadí farbou bielou (#FFFFFF).

Obrázok 3.10. Paleta farieb.

Kapitola 4
Implementácia

4.1 Použité technológie

4.1.1 Vývojové nástroje

Pri návrhu aplikácie bola využitá online aplikácia draw.io pre kreslenie užívateľských prípadov, databázových modelov a ostatných diagramov. Všetky časti systému boli vyvíjané vo vývojovom prostredí Idea od spoločnosti IntelliJ, ktorá poskytuje študentskú licenciu pre tento nástroj. Pri implementácii aplikácie bol využívaný verzovací nástroj Git. Repozitár obsahujúci zdrojové kódy aplikácie je umiestnený v aplikácii BitBucket od spoločnosti Atlassian.

4.1.2 Jazyk Java

Pre implementáciu jednotlivých mikroservis bol zvolený jazyk Java verzie 11.

Tento jazyk bol zvolený kvôli jeho popularite, podľa TIOBE indexu sa jedná o najpopulárnejší jazyk.[21] Veľká popularita jazyka znamená, že je použitý vo veľkom množstve projektoch. To väčšinou znamená že boli podobné problémy riešené veľakrát, čiže je ľahké nájsť ich riešenie v rôznych ľahko dostupných návodoch alebo na platformách akou je napríklad Stack Overflow (https://stackoverflow.com/).

Jednou z mnoha výhod jazyka Java je aj jeho prenositeľnosť medzi platformami, čo umožnilo vyvíjať a zostaviť (build) systém na operačnom systéme Mac OS a spustiť ho v dockery s operačným systémom Debian.

Pre správu použitých knihovien a build aplikácie bol použitý Maven 3[22]. Maven je softvérový projekt manager používaný pre správu, riadenie a automatizáciu buildov. Pomocou mavenu je jednoduché pridávať nové závislosti ako rôzne externé knihovny a zásuvné moduly (anglicky plugin).

4.1.3 Spring

Rozšírený jazyk ako je Java poskytuje mnoho aplikačných rámcov (anglicky framework) ktoré programátorom uľahčujú život a prácu. Aplikačný rámec je softvérová štruktúra, ktorá tvorí základnú štruktúru aplikácie a poskytuje programátorovy akúsi konstru, ktorá môže obsahovať podporné knihovny, programy, zásuvné moduly, podporu pre návrhové vzory alebo doporučené postupy pri návrhu a vývoji. Programátor do tejto štruktúry implementuje požadovanú funkcionalitu požadovaného systému.

Pri implementácie diplomovej práce bol zvolený rámec Spring Boot Framework. Spring Framework je open-source softvér (otvorený softvér) používaný pri vývoji Java Enterprise aplikácii. Jedným z hlavných dôvodov, prečo bol tento rámec zvolený pri implementácii je jednoduchosť spustenia, kedy stačí spustiť vygenerovaný .jar súbor, pretože Spring Boot má v sebe priamo zabudovaný Tomcat (popr. Jetty) web server. Ďalej poskytuje knihovny pre autentifikáciu, komunikáciu s databázou s podporou hibernate frameworku, knihovny pre jednoduchú implementáciu REST API a mnoho ďalších užitočných nástrojov a knihovní.

4.1.4 Databáza

Pre ukladanie dát aplikácie sú použité dva rôzne databázové systémy.

V relačnej databáze PostgreSQL sú ukladané dáta ktoré je potrebné mať v každom okamžiku konzistentné a pri ktorých sa neočakáva veľké vyťaženie v zmysle veľkého množstva dotazov v jeden okamžik a rýchleho spracovania požiadaviek. PostgreSQL je voľný open-source program. Tento typ RDBMS bol znovený pretože je to jeden z najpoužívanejších databázový systém s voľnou licenciou k používaniu[23]. Aj keď nie je tak populárny ako MySQL, tak má výhodu v tom že je striktnejšie dodržiava štandardu jazyka SQL a zároveň podporuje pokročilé funkcie.

NoSQL databáza MongoDB je použitá na ukladanie dát ktoré sú málo (ideálne nikdy) upravované ale často čítané a majú zložitejšiu štruktúru, čiže je praktickejšie tieto data držať v jednom dokumente a pohromade. MongoDB bola zvolená kvôli jej jednoduchému použitiu, pretože je podporované Spring Boot Frameworkom (a je implementovaná knihovna na komunikáciu v rámci tohto rámcu). Ďalším z dôvodov je jej popularita z NoSQL databázy[24] a v neposlednom rade kvôli tomu, že ukladá data ako JSON, čo umožňuje ľahkú definíciu štruktúry dát, je jednoduché človekom čitať uložené dáta a rovnako je časté používanie JSON formátu pri posielaní dát pomocou REST API, zároveň ho podporujú Front-End rámce založené na JavaScripte (napr. Angular ktorý bol použitý v tejto práci).

4.1.5 Migrácia databáze

Migrácia je spôsob, akým sa v aplikácii upravuje štruktúra databáze. Tato technika nám oproti manualnemu upravovaniu databáze cez opytovací jazyk, umožňuje správu verzii štruktúry databáze. Zmeny sú implementované v migračných skriptoch, ktoré sú poriadne označené pre rozlíšenie o ktorú verziu sa jedná. Existuje niekoľko knihovien poprípade rámcok ktoré dokážu v Jave automaticky spúšťať migračné skripty a zároveň dokážu si dokážu kontrolovať verzie a správnosť poradia skriptov (napr. aby sa nemenil dookola jeden script ale pre každú migráciu bol vytvorený nový s odpovedajúcim názvom).

V tejto práci bol pre migráciu PostgreSQL relačnej databázy zvolený Flyway migračný nástroj. Flyway poskytuje jednoduché nastavenie migrácie pomocou Maven Pluginu, kde vyžaduje len pár informácii o databáze a schéme pre správne fungovanie. Podporuje písanie migrácie buď v SQL alebo v Jave, v práci bolo zvolené písanie migračných skriptov v jazyku SQL.

4.1.6 Angular

Jednoduché a prehľadné užívateľské rozhranie je veľmi dôležité pri implementácii aplikácii s ktorou bude potenciálne pracovať veľké množstvo rôznych ľudí (rôzne vzdelaných, počítačovo zdatných, rôzneho veku a pohlavia). No je zložité takéto rozhranie navrhnúť a niekedy ešte zložitejšie implementovať. Pre uľahčenie implementácie užívateľských rozhraní vzniká veľa užitočných rámcov, niektorých menej niektorých viac, ktoré je možné bezplatne používať. Jedným z nich je aj Angular, vyvíjaný spoločnosťou Google, ktorý bol použitý v tejto diplomovej práci.

Angular je TypeSkriptový rámec používaný pre implementáciu mobilných a počítačových webových aplikácii, ktorý sa zameriava na implementáciu jedno-stránkových aplikácii (z ang. single-page)[25]. Tento rámec bol zvolený pretože je možné (a pomerne jednoduché) sa ho naučiť a používať ho v pomerne krátkej dobe. Angular poskytuje množstvo pred-implementované funkcionality, čo uľahčuje prácu a dáva programátorom priestor sústrediť sa len na požadovanú funkcionalitu.

4.1.7 Google Kubernetes

Google Kubernetes je prostredie pre nasadenie kontajnerových aplikácii. Umožňuje správu nasadených kontajnerových aplikácii, správu zdrojov (CPU, operačnej pamäte, úložnej pamäte atď), poskytuje externé IP adresy load balancing a mnoho ďalších funkcionalit.

Google Kubernetes bol zvolený pre beh systému za účelom testovania a prevádzky aplikácie, keďže aplikácia vyvíjaná v rámci tejto diplomovej práce bola navrhnutá a implementovaná ako cloudová aplikácia. Kubernetes umožňuje paralelný beh viacerých inštancií jednej služby ako aj beh viacerých služieb. Pre tieto účely je rozdelený do niekoľkých vrstiev.

Pre spustenie aplikácie v Google Kubernetes musí byť užívateľ registrovaný a musí si vytvoriť projekt pre svoju aplikáciu. V rámci jedného projektu je možné vytvoriť niekoľko klastrov. Každý cluster má definovanú lokalitu kde beží, zdroje, počet uzlov (ktorý sa môže dynamicky meniť podla vyťaženia) a maximálnu dostupnú pamäť (s typom úložiska HDD alebo SSD). V rámci jedného clustru môže bežať niekoľko samostatných služieb, v závislosti na dostupné zdroje. Každá služba má svoju internú IP adresu, prípadne môže byť vystavená internetu a má externú IP adresu. Služby môžu medzi sebou komunikovať v rámci klasteru buď podľa interných IP adries, alebo Google Kubernetes dokáže nadviazať komunikáciu medzi jednotlivými službami na základe ich názvu. Každá služba obsahuje vrstvu deployment. Deployment vrstva obsahuje jeden alebo viaceré pods. Každý Pod by mala byť jedna inštancia bežiacej aplikácie. Deployment vrstva dokáže rozlišovať vyťaženosť pods a správne rozdeľovať spracovanie dotazov medzi ne. V rámci každého pod by mala správne behať jedna kontajnerovaná aplikácia.

Google Kubernetes poskytuje pomerne jednoduché nasadenie aplikácie do klastru. Raz ako je cluster vytvorený, je možné pomocou terminálu (alebo cez webové rozhranie, ktoré ale nebolo použité pri deployovaní aplikácie) spustiť vopred pripravené konfiguračné súbory (definované vo formáte yaml). Definovanie takýchto konfiguračných súborov zabezbečí konzistentné nasadenie služby pri každom nasaďovaní novej verzie.

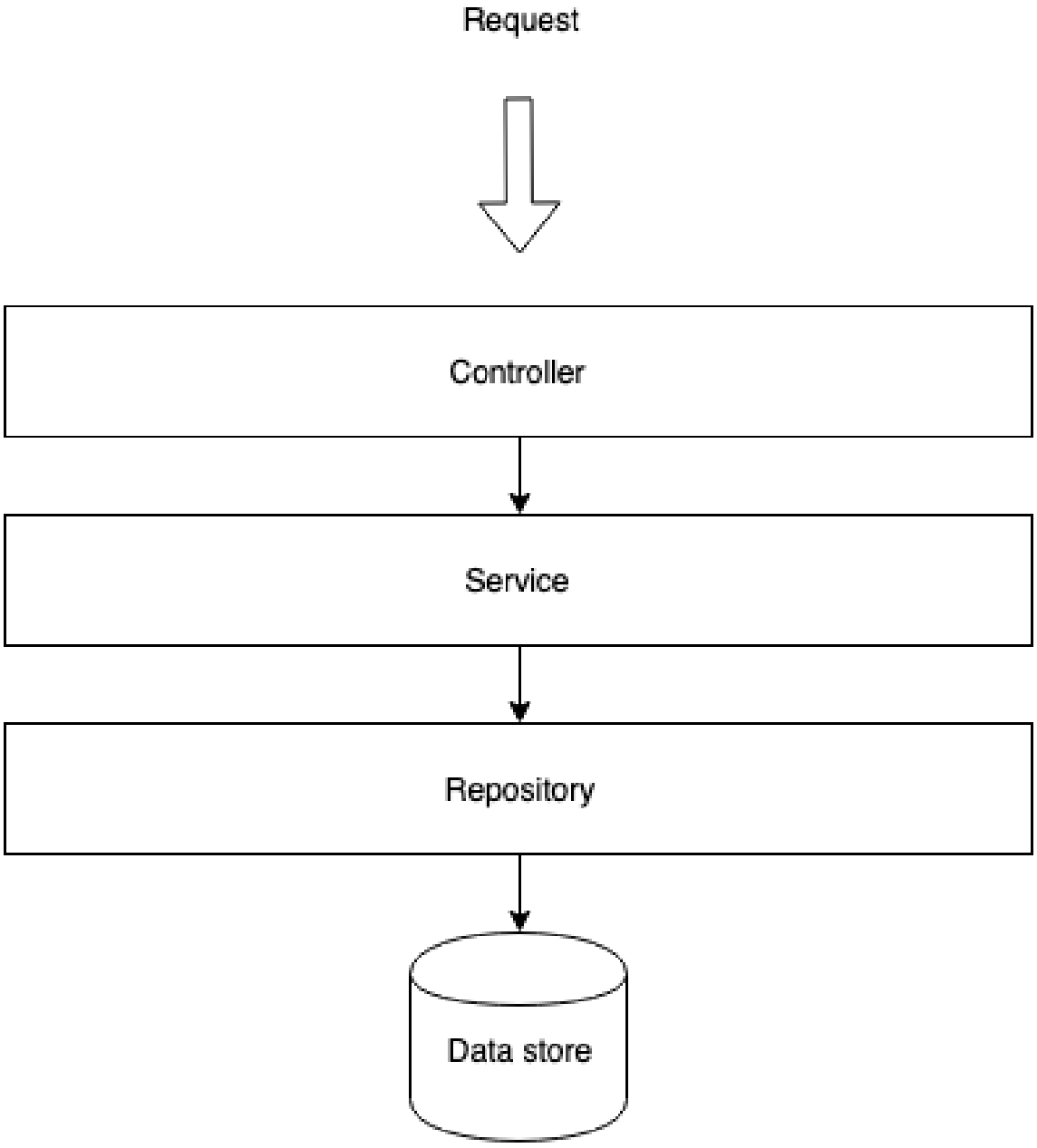
4.2 Implementácia servis

Každá služba bola implementovaná ako samostatne fungujúca jednotka, ktorá sa dá samostatne spustiť, upravovať prípadne škálovať. V rámci prototypu boli implementované nasledujúce služby:

- Správa zákazníkov
- Správa vozidiel
- Autentifikačná služba
- Správa prevádzok
- Správa rezervácii
- Gateway

Každá služba je implementovaná veľmi podobným spôsobom a to v Jave s použitím rámcu Spring Boot Framework. Všetky služby sú implementované po vrstvách, kde každá vrstva má svoju konkrétnu úlohu. Ako môžeme vidieť na obrázku 4.1) všetky požiadavky sú spracovávané Kontrolérmi. Každý endpoint ktorý obsahuje telo požiadavku implementovaný v rámci kontroleru akceptuje JSON formát tela. Tie pretransformujú Transfér Objekty, ktoré sú vygenerované z tela požiadavku, do doménových objektov, s ktorými pracuje vrstva služieb. Kontroler presmeruje pretransformované data do triedy služby (ServiceClass) v ktorej sa vykonáva všetka logika danej MikroServisy.

Mikroservisa môže obsahovať aj viacero ServisClass, záleží na veľkosti a zložitosti danej mikroservisy. Ukladanie dát do dátového úložiska je zabezpečované repository vrstvou. Tá akceptuje doménové objekty na vstupe, tie následne pretransformuje na entity objekty, ktoré sú ďalej, s využitím dostupných knihovien v Spring Boot rámci, ukladané do dátového úložiska.



Obrázok 4.1. Všeobecné znázornenie vrstiev služieb

4.2.1 Správa zákazníkov

Služba implementuje správu zákazníkov, to znamená registráciu nového zákazníka, upravu informácii o zákazníkovi, blokovanie a mazanie zákazníka. Rovnako imlpementuje možnosť vrátenia zoznamu všetkých zákazníkov spĺňajúcich filtrovacie kritéria.

Táto služba pri procese registrácie volá službu autentifikácie, ktorá po úspešnej registrácii vráti na odpovedi unikátne ID nového užívateľa, ktoré je ďalej používané v tejto službe. Toto prináša závislosti na inej službe, čo by sa pri mikro-servisnej architektúre

nemalo stávať, no autor práce sa rozhodol pre takéto riešenie, pretože inou alternatívou bolo buď:

- Pridať logiku do Gateway mikroservisy, čo ale bolo nežiaduce, pretože by to zbytočne zaťažilo gateway a mohlo by to obmedziť chod pre ostatné dotazy. Zároveň pri chybe by sa mohlo stať že by spadla celá služba a to by odstavilo celú aplikáciu.
- Druhá možnosť bola vytvoriť medzivrstvu (medzislužbu) medzi gateway a takými mikroservisami ktoré si medzi sebou potrebujú predávať informácie, toto riešenie však autorovi prišlo pre zatiaľ jednoduchý systém zbytočne komplikované, keďže v tejto mikroservise je len jeden prípad kedy je potreba volať inú službu.

Služba je pripojená k PostgreSQL databázy.

4.2.2 Správa vozidiel

Služba implementuje správu vozidiel a úkonov na vozidlách vykonaných. Je možné vytvoriť nové vozidlo, upraviť existujúce vozidlo poprípade zmazať takéto vozidlo a s ním aj všetky úkony na ňom vykonané.

Služba nie je závislá na žiadnych iných službách.

Služba je pripojená k MongoDB databázy.

4.2.3 Autentifikačná služba

Služba implementuje logiku ohľadom prihlasovanie a registráciu všetkých užívateľov v systéme. Po úspešnom prihlásení, ktoré je zaobstarané Spring Security knihovnou, služba vydá podpísaný stateless token ktorý je odoslaný v odpovedi užívateľovy.

Token obsahuje informácie a to užívateľské meno, jeho ID, oprávnenia ktoré má užívateľ a zároveň čas vytvorenia tokenu a čas jeho expirácie. Každý token je validný presne 24 hodín. Tokeny sú typu JWT, čo je kompaktný objekt, ktorý poskytuje bezpečný spôsob prenosu informácii medzi stranami (v našom prípade klientom a serverom) v JSON formáte.

Služba nie je závislá na žiadnych iných službách.

Služba je pripojená k PostgreSQL databázy.

4.2.4 Správa prevádzok

Služba implementuje správu prevádzok a to registráciu novej prevádzky, úpravu alebo zobrazenenie informácii o existujúcej prevádzke, mazanie prevádzky a zobrazenie zoznamu prevádzok ktoré spĺňajú filtrovacie kritéria. Služba takisto implementuje správu úkonov poskytovaných prevádzkami. Úkony je možné pridávať, upravovať, mazať alebo je možné zobrazenie úkonov danej prevádzky. Každý úkon patrí práve k jednej prevádzke.

Endpoint pre zoznam prevádzok podporuje stránkovanie. Stránkovanie má predvolené hodnoty na pre zobrazenie prvej stránky s 20 (slovom dvadsiatimi) záznamami. Hlavička odpovede mimo iné obsahuje odkazy (ak existujú) na prvú stránku, na predchádzajúcu stránku, na ďalšiu stránku a na poslednú stránku záznamov.

Služba pri procese registrácie volá službu autentifikácie, ktorá po úspešnej registrácii vráti na odpovedi unikátne ID novo registrovanej prevádzky, ktoré je ďalej používané v tejto službe.

Služba je pripojená k PostgreSQL databázy.

4.2.5 Správa rezervácii

V tejto službe je implementovaná logika ktorá má na starosti rezervácie a všetky úkony s tým spojené. Je možné vytvoriť novú rezerváciu, upraviť alebo zobraziť existujúcu rezerváciu poprípade zmazať rezerváciu. Rovnako je možné zobraziť všetky rezervácie ktoré buď náležia zákazníkovi alebo patria k prevádzke. Služba rovanko implementuje logiku tvorby rozvrhu prevádzky, na základe ktorého si zákazník rezervuje voľný termín.

Pri tvorbe rozvrhu je treba zohľadniť maximálny počet vozidiel ktoré môžu byť v jednom čase rezervované, čiže maximálnu kapacitu prevádzky a otváraciu dobu danej prevádzky. Ďalej je treba zistiť, načítaním z databáze, všetky úkony ktoré sa v daný čas vykonávajú. V neposlednom rade je treba pri tvorbe rozvrhu zohladniť dĺžku požadovaného úkonu, aby sme zákazníkovi neponúkli čas, ktorý nebude dostatočne dlhý na vykonanie úkonu.

Algoritmus (3.9) popisuje ako sa vypočítava rozvrh pre obdobie v prípade že je maximálna kapacita prevádzky rovná jednej. Rozvrh je možné predstaviť si ako n vektorov o m prvkoch, kde m reprezentuje počet hodín v dni (v našom prípade je m vždy rovné 24). Ak je kapacita prevádzky väčšia ako jedna, je použitý rovnaký algoritmus samostatne pre každé miesto v prevádzke. Konečný rozvrh je získaný skalárnym súčinom vektorov dňa cez všetky miesta v prevádzke.

```
Input: Dátum from, Dátum to, Číslo taskTime
Output: Rozvrh

dates := from.daysUntil(to)
scheduleDays := emptyList()

for(date : dates){
    scheduleDays.add(scheduleDay(date, taskTime))
}

Procedure scheduleDay:
{
    Input: Dátum date, Číslo taskTime
    Output: Jeden deň rozvrhu

    res := getReservations(date, this.shopId)
    openingHours := getOpeningHours(date, this.shopId)
    scheduleHours := busyMap() \\ mapa hodín v dni všetky s hodnotou BUSY

    for(hour : openingHours){
        status := res.get(hour).isFree ? FREE : BUSY
        scheduleHours.put(hour, status)
    }

    return scheduleHours
}
```

. Vytváranie rozvrhu pre jedno miesto v prevádzke.

4.2.6 Gateway

Gateway je implementovaná ako samostatná služba, ktorá zabezpečuje presmerovanie dotazov na jednotlivé služby. Táto služba zároveň validuje token a kontroluje či daný užívateľ má oprávnenie vykonať dotazovanú operáciu, presmerovanie je uskutočnené

len po úspešnej validácii a kontrole oprávnení. Ak služba, na ktorú by mal byť dotaz presmerovaný, nie je dostupná gateway vráti informáciu o nedostupnosti. V rámci tejto služby je implementovaný aj Load Balancer, ktorý zabezpečuje presmerovanie dotazov na najmenej vyťažené inštancie služieb.

Gateway nie je pripojená k žiadnej databázy. Napriek tomu že presmeruváva dotazy na všetky služby, pre plnohodnotný chod gateway nie je potrebný chod týchto služieb, a gateway sa vie vysporiadať s nedostupnosťou služieb.

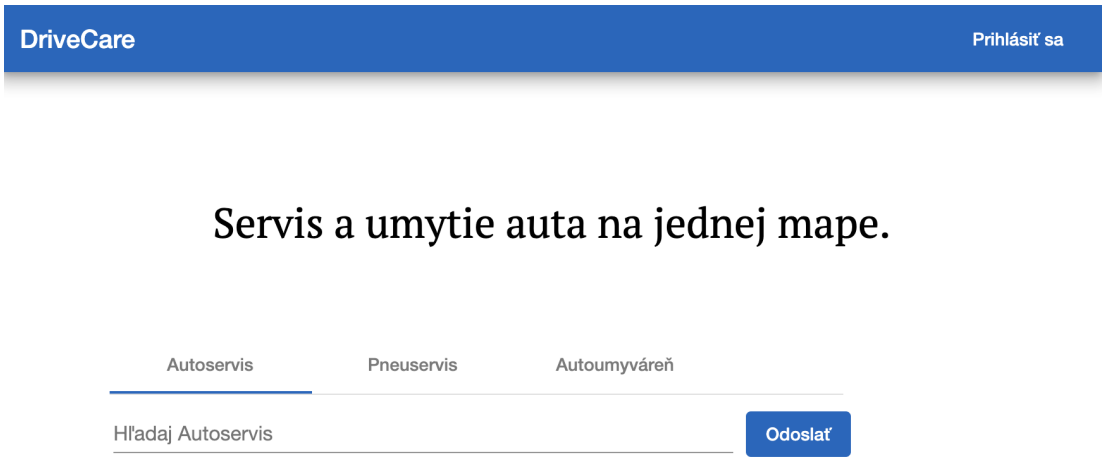
4.3 Užívateľské rozhranie

Užívateľské rozhranie bolo implementované tak, aby sa správne zobrazovalo na displejoch monitorov, čiže sa môže stať, že sa aplikácia na zariadeniach s menším rozlíšením nebude dobre zobrazovať. Pri implementácii bol využitý Angular rámec, ktorý je opísaný v sekcii 4.1.7.

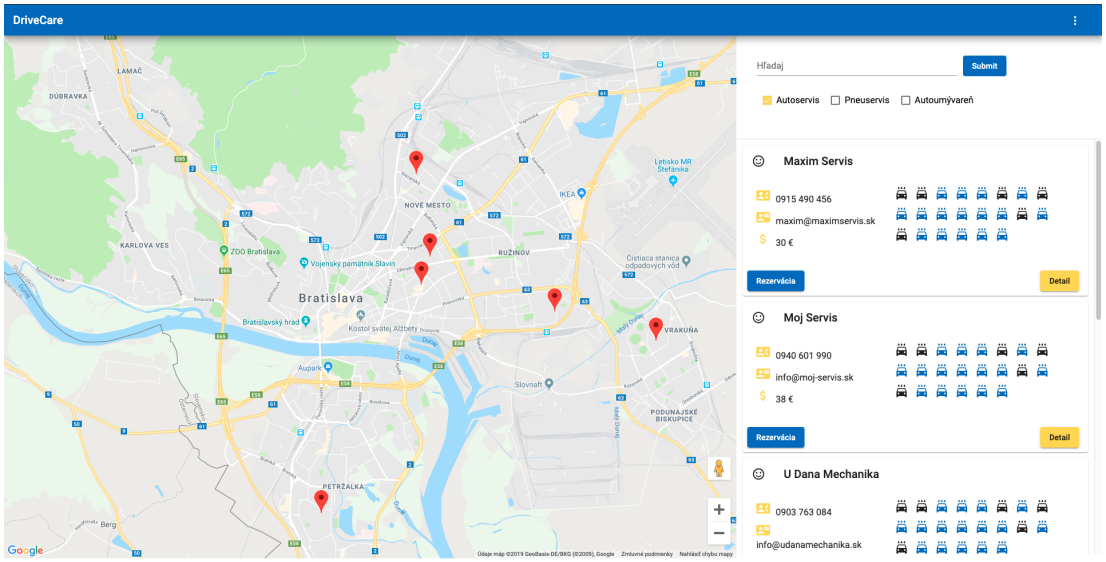
V rámci tejto práce nie je cieľom plne implementovať aplikáciu navrhnutú v sekcii Návrhu 3.1, no vytvoriť funkčný prototyp. Preto implementácia užívateľského rozhrania nepokrýva všetky štyri časti spomínané v návrhu užívateľského rozhrania 3.3.1 ale len tie, ktoré umožňujú vytvorenie rezervácie a administratívu prevádzkami.

4.3.1 Základná časť

Základná časť užívateľského rozhrania je do väčšej miery implementovaná. Na obr. 4.2 môžeme vidieť hlavnú stránku aplikácie, ďalej je spravená stránka vyhľadávanie s mapou a zoznamom prevádzok.

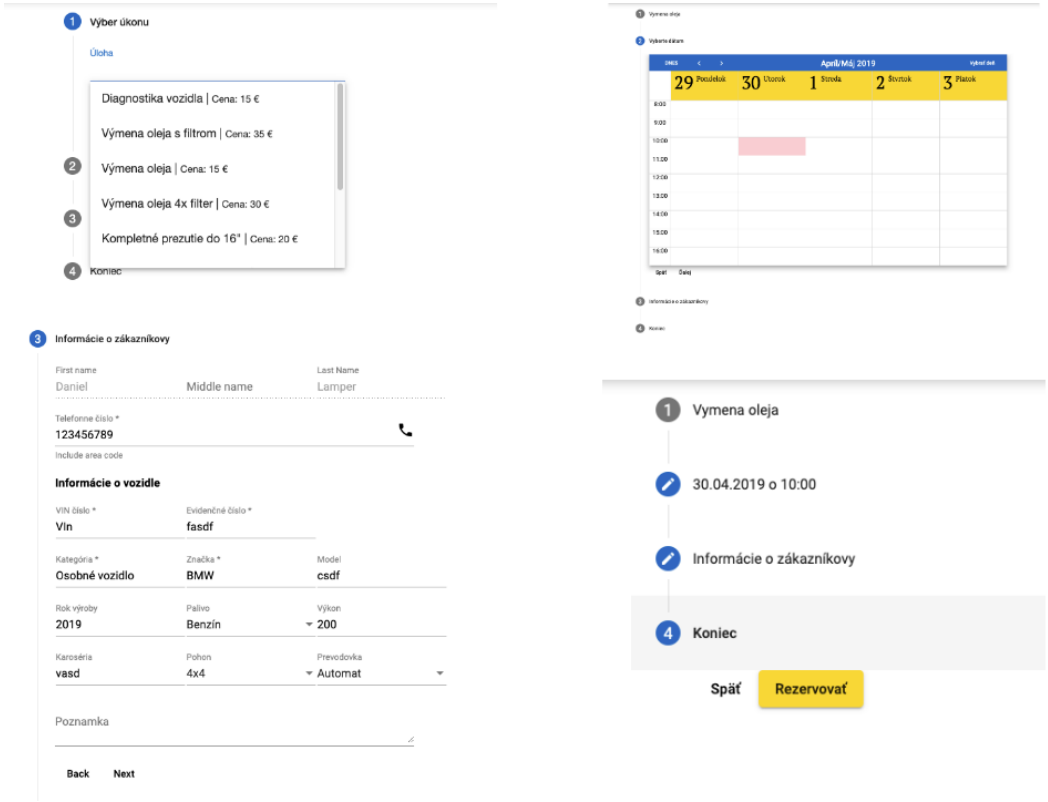


Obrázok 4.2. Snímka obrazovky domovskej stránky aplikácie.



Obrázok 4.3. Snímka obrazovky vyhľadávanie prevádzok v systéme.

Rovnako tak je možné vytvoriť si rezerváciu po zvolení prevádzky, kde užívateľ bude, po zvolení úkonu, presmerovaný na formulár rezervácie. 4.4

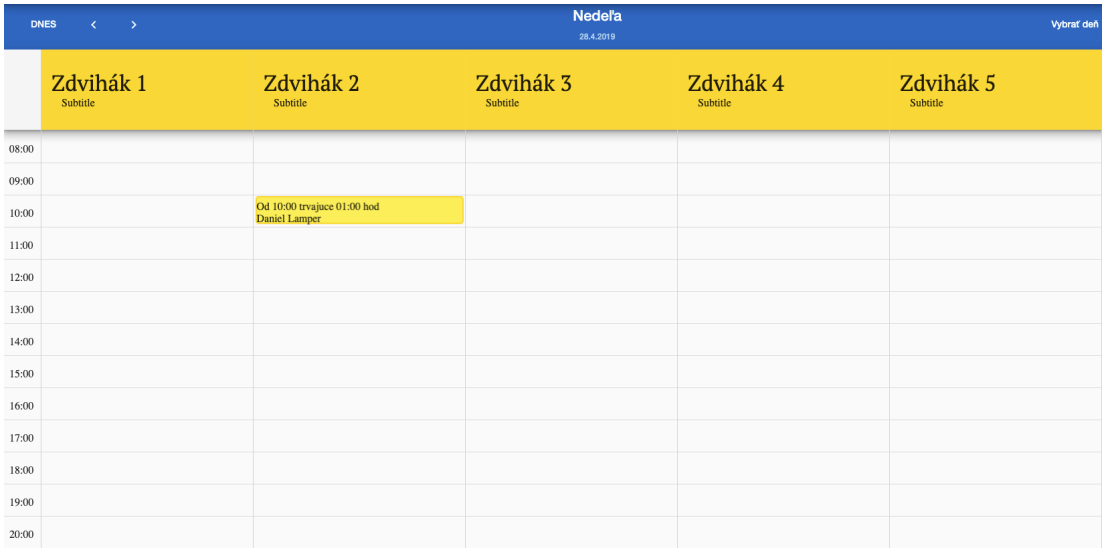


Obrázok 4.4. Snímky formuláru vytváranie rezervácie.

4.3.2 Administratívna prevádzok

Administratíva prevádzok má implementovaných niekoľko častí. Tou najhlavnejšou je rozvrh s rezerváciami, ktorý pre jednotlivé zdviháky ukazuje úkony k nim rezervované. Rezerváciu je možné rozkliknuť a vidieť jej detail. Tento detail nie je plne implementovaný a momentálne sa zobrazí text v JSON formáte v surovom stave v akom prišiel zo

servera. Ďalej je možné vytvoriť novú, veľmi jednoduchú rezerváciu pre konkrétny čas na konkrétnom zdviháku.



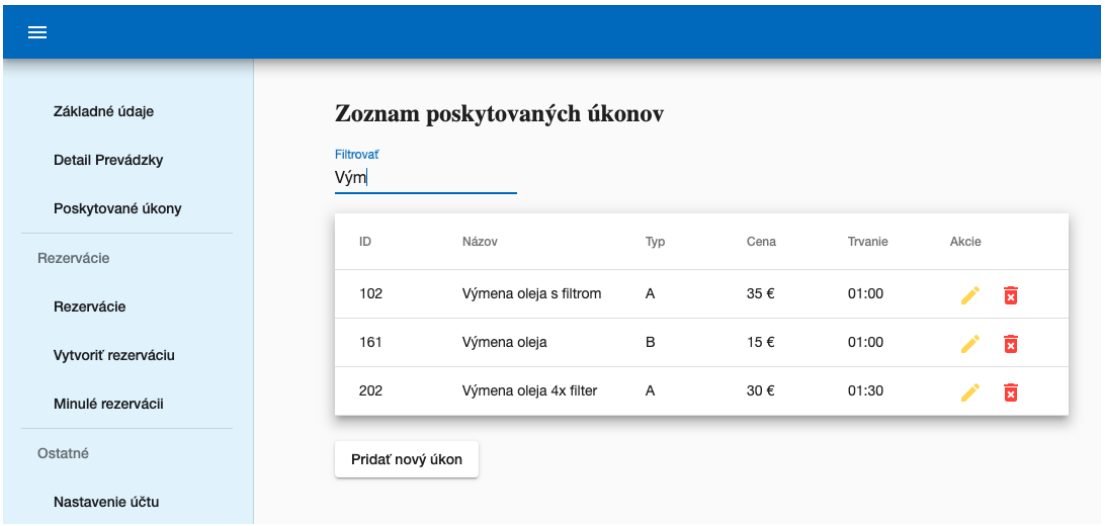
Obrázok 4.5. Snímka obrazovky rozvrhu.

Ďalšie obrazovky implementované v rámci tejto časti užívateľského rozhrania sú stránky pre zobrazenie a spravovanie informácii o prevádzke.

Pre upravovanie a zobrazovanie údajov o užívateľovi je implementovaná samostatná stránka. Na tejto stránke je možné zmeniť len heslo, momentálne systém nepodporuje zmenu užívateľského mena ani emailu.

Pre zadávanie rezervácií s komplexnejšími informáciami je implementovaná samostatná stránka, ktorá vyzerá podobne ako rezervačná stránka implementovaná v základnej časti systému.

V neposlednom rade je rovnako spravená stránka poskytuje možnosť pridávania, upravovania a zobrazovania zoznamu poskytovaných úkonov prevádzkou. Na tejto stránke sa nachádza tabuľka s týmito poskytovanými úkonmy, kde je podporované filtrovanie úkonov. Filtrovanie je jednoduché, kde sa vo všetkých stĺpcoch tabuľky hľadá zhoda s vyhladávacím paternom.



Obrázok 4.6. Snímka obrazovky s tabuľkou úkonov poskytovaných prevádzkou spolu s filtrovaním úkonov.

Kapitola 5
Testovanie

Testovanie je dôležitou súčasťou vývoja softvéru, kedy overujeme správnosť a kvalitu softvéru. Testovanie aplikácii prebieha v iteráciách na rôznych úrovniach.

V rámci tejto práce prebehlo testovanie na štyroch rôznych úrovniach, a to testovanie jednotiek (z anglického Unit Testing), integračné testovanie, manuálne testovanie užívateľského rozhrania a nakoniec akceptačné testovanie v prevádzkach.

5.1 Testovanie jednotiek

V objektovo orientovanom programovaní, jednotkové testovanie predstavuje testovanie jednotlivých tried a metód. Testovacou jednotkou nazývame takú časť aplikačného programu, ktorá je samostatne testovatelná. Tieto testy sú z pravidla písané programátorom ktorý danú jednotku vyvinul.

V tejto práci bol, na písanie jednotkových testov, použitý rámec JUnit. Tento rámec je implementáciou architektúry xUnit v jazyku Java[26].

Každý JUnit test je implementovaný ako samostatná metóda. Každá takáto metóda, testuje jeden prípad v ktorom môže byť testovaná funkcionalita použitá. K väčšine metód pristupujeme ako k black boxu, čiže nevieme, akým spôsobom metóda funguje, vieme len očakávané výsledky pre dané vstupy, ktoré v rámci každého testu kontrolujeme pomocou porovnávania (takzvanými assert funkciami).

V rámci jednotkových testov nechceme byť ovplyvnený fungovaním iných častí systému, na ktorých môže byť testovaná trieda alebo metóda závislá, preto takéto závislosti falšujeme (z anglického mock). Mockovacie objekty sú simulované objekty ktoré napodobňujú chovanie skutočných objektov, s tým, že programátor vie ovládať ich chovanie. V tejto práci bol na mockovanie použitý rámec mockito[27].

Pri písaní jednotkových testov, je jedným z dôležitých aspektov kontrolovať či vytvorené testy pokrývajú všetok kód danej funkcionality. Samozrejme nie je praktické, snažiť sa pokryť testami 100% kódu aplikácie, ale v servisných triedach, v ktorých je implementovaná biznis logika, chceme mať pokrytie čo najväčšie. V Jave je najznámejším a najpoužívanejším nástrojom na meranie pokrytia kódu JaCoCo[28–29]. Jacoco bolo použité aj v tejto práci, výsledky pokria kódu JUnit testami môžete vidieť na obr. 5.1.

Element	Missed Instructions	Cov.	Missed Branches	Cov.
svk.drivecare.authservice.model		0%		0%
svk.drivecare.authservice.domain		43%		21%
svk.drivecare.authservice.api.dto		0%		0%
svk.drivecare.authservice.api		0%		0%
svk.drivecare.authservice.security		0%		n/a
svk.drivecare.authservice		0%		n/a
svk.drivecare.authservice.repository.user		0%		n/a
svk.drivecare.authservice.service.user		96%		87%
svk.drivecare.authservice.repository.password		0%		n/a
svk.drivecare.authservice.service.password		96%		100%
svk.drivecare.authservice.service		100%		n/a
svk.drivecare.authservice.domain.enums		100%		n/a
svk.drivecare.authservice.exceptions		100%		n/a
Total	2,297 of 5,189	55%	312 of 396	21%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
svk.drivecare.customer.management.api.dto		0%		0%
svk.drivecare.customer.management.domain		47%		28%
svk.drivecare.customer.management.listeners		0%		0%
svk.drivecare.customer.management.api		0%		0%
svk.drivecare.customer.management.model		0%		0%
svk.drivecare.customer.management.repository		0%		n/a
svk.drivecare.customer.management.events		0%		n/a
svk.drivecare.customer.management		0%		n/a
svk.drivecare.customer.management.service		97%		100%
svk.drivecare.customer.management.exceptions		100%		n/a
Total	2,670 of 3,254	17%	396 of 424	6%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
svk.drivecare.vehicle.management.api.dto		0%		0%
svk.drivecare.vehicle.management.model		0%		0%
svk.drivecare.vehicle.management.domain		30%		16%
svk.drivecare.vehicle.management.listeners		0%		0%
svk.drivecare.vehicle.management.api		0%		0%
svk.drivecare.vehicle.management.repository		0%		n/a
svk.drivecare.vehicle.management.events		0%		n/a
svk.drivecare.vehicle.management.service		94%		n/a
svk.drivecare.vehicle.management		0%		n/a
svk.drivecare.vehicle.management.exceptions		100%		n/a
Total	3,186 of 3,936	19%	528 of 558	5%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
svk.drivecare.shopmanager.api.dto		0%		0%
svk.drivecare.shopmanager.domain.shop		32%		12%
svk.drivecare.shopmanager.model		0%		0%
svk.drivecare.shopmanager.domain.common		31%		12%
svk.drivecare.shopmanager.model.common		0%		0%
svk.drivecare.shopmanager.repository		0%		0%
svk.drivecare.shopmanager.listeners		0%		0%
svk.drivecare.shopmanager.api		0%		0%
svk.drivecare.shopmanager.domain.enums		66%		n/a
svk.drivecare.shopmanager.events		0%		n/a
svk.drivecare.shopmanager.service		97%		100%
svk.drivecare.shopmanager		0%		n/a
svk.drivecare.shopmanager.exceptions		100%		n/a
Total	6,443 of 8,462	23%	1,066 of 1,138	6%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
svk.drivecare.reservation.management.api.dto		1%		0%
svk.drivecare.reservation.management.model		0%		0%
svk.drivecare.reservation.management.domain		48%		28%
svk.drivecare.reservation.management.api.dto.input		0%		0%
svk.drivecare.reservation.management.domain.customer		22%		1%
svk.drivecare.reservation.management.domain.schedule		28%		5%
svk.drivecare.reservation.management.domain.shop		22%		2%
svk.drivecare.reservation.management.domain.common		22%		1%
svk.drivecare.reservation.management.api		0%		n/a
svk.drivecare.reservation.management.repository		0%		n/a
svk.drivecare.reservation.management.service		98%		94%
svk.drivecare.reservation.management		0%		n/a
svk.drivecare.reservation.management.domain.enums		100%		100%
svk.drivecare.reservation.management.exceptions		100%		n/a
Total	9,902 of 13,428	26%	1,782 of 1,906	6%

Obrázok 5.1. Graf pokrytia kódu testami u jednotlivých služieb.

5.2 Integračné testy

Keď chceme otestovať funkčnosť služby komplexnejšie, jednotkové testy nie su dostačujúce. V takom prípade prichádza na radu integračné testovanie aplikácie, kedy sa testuje funkčnosť systému na úrovni integrácie spolu s ostatnými systémami, ktoré softvér využíva, to je napríklad databáza, triedy rámcu a podobne.

Integračné testovani je možné automatizovať, alebo môže byť testované manuálne. Pri tejto práci prebiehalo integračné testovanie manuálne na úrovni REST API ponúkané službami. REST API bolo dotazované pomocou nástroja Postman[30]. V aplikácii Postman, je jednoduchým spôsobom možné vytvoriť REST dotaz, s definovaním hlavičiek, tela a parametrov dotazu. Rovnako odpoveď obsahuje hlavičky a telo odpovede, takže je vizuálne jednoduché skontrolovať, či sú hodnoty v odpovedi správne.

5.3 Testovanie grafického užívateľského rozhrania

V predchádzajúcich sekciách sme hovorili len o testovaní back-endovej časti systému. Lenže tá časť systému komunikuje s užívateľmi len nepriamo a vo veľkej väčšine prípadov spracováva dotazy vygenerované inou aplikáciou, v našom prípade sa tým myslí front-endova časť aplikácie. Tým chceme povedať, že väčšina dotazov a užívateľských prípadov je pomerne jednoducho predpovedateľná, čiže správne otestovaný systém nie je až tak náchylný na zlyhania. Iný prípad nastáva pri Grafickom Užívateľskom Rozhraní (ďalej len GUI), kde s aplikáciou interaguje množstvo rôznych ľudí, ktorých chovanie je častokrát neracionálne a ťažko predpovedateľné. Preto je komplexné testovanie GUI veľmi dôležité a pri vývoji systému pre verejnosť častokrát kritickým bodom vývoja aplikácie.

GUI je možné testovať manuálne alebo automaticky. Na automatické testovanie užívateľského rozhrania je jeden z najznámejších a najpoužívanejších nástrojov Selenium[31]. Selenium sa používa na testovanie webových aplikácii v prehliadači. Inými možnosťami na ako testovať webové aplikácie, je použitie Jasmine testovacieho rámca[32]. Jasmine testovací rámec testuje JavaScriptový kód, takže to nie je úplne to isté ako Selénium, ale je s ním možné otestovať funkcionálne časti front-endu.

Automaticke testovanie GUI je však veľmi náročné na udržateľnosť, pretože takéto testy sú hrozne neflexibilné a náchylné na nefunkčnosť i pri malých zmenách užívateľského rozhrania[33]. Preto v rámci tejto diplomovej práce neboli vyvíjané žiadne GUI automatické testy. Rozsah vyvíjaného prototypu aplikácie nie je zatiaľ tak veľký aby bolo náročné, pre jednu osobu, prejsť testovacie scenáre manuálne v priebehu desiatok minút, maximálne však pár hodín.

5.3.1 Testovacie stratégie

V rámci manuálneho testovania boli testované nasledujúce funkcionality systému:

- Prihlásenie
  - Zákazník
  - Prevádzka
- Registrácia zákazníka
- Vyhladávanie prevádzok
- Vytvorenie rezervácie
  - Neregistrovaným zákazníkom

  - Registrovaným zákazníkom
  - Prevádzkov

- Úprava informácii o prevádzke
- Zmena hesla prevádzkov
- Pridanie/Mazanie/Úprava úkonov prevádzky
- Prehlad rezervácii prevádzkou
- Odhlásenie

Tabuľky prioritizácii testovacích oblastí, intenzity testov a testovacích techník sú k nahliadnutiu v prílohách.

5.3.2 Testovacie scenáre

Pri vytváraní testovacích scenárov potrebujeme rozlišovať či chceme testovať procesy, alebo kontrolovať chovanie systému na vkladane dát od užívateľov.

Pri vytváraní testovacích scenárov kontrolujúcich chovanie pri vkladaní dát je najprv potrebné správne si určiť medzné podmienky, triedy ekvivalencie a definovať kombinácie vstupných dát.

Testovanie ekvivalencie rozdelenie si vstupov do aplikácie do tried, u ktorých sa očakáva rovnaký výsledok[34]. Ináč povedané, každá hodnota z triedy ekvivalencie má rovnakú šancu detekovať chybu[35].

5.3.2.1 Prihlasovanie

Pri procese prihlasovanie, je užívateľovi zobrazený formulár s dvoma poliami a to pole pre užívateľské meno a heslo. Pre tieto dve polia definujeme triedy ekvivalencie ktoré je možné vidieť v tabuľke 5.1.

Názov poľa	Triedy Ekvivalencie
Užívateľské meno	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Existujúce meno, Neexistujúce meno
Heslo	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Správne heslo, Nesprávne heslo

Tabuľka 5.1. Triedy ekvivalencie pre Prihlasovanie.

5.3.2.2 Registrácia

Pri procese registrácie nového zákazníka, je užívateľovi zobrazený komplexný formulár obsahujúci rôzne polia. Pre tieto polia definujeme triedy ekvivalencie, ktoré je možné vidieť v tabuľke 5.2.

Počas registrácie je možné, ale nie potrebné, vyplniť informácie o vozidle, triedy ekvivalencie pre vozidlo sú definované v tabuľke 5.3

5.3.2.3 Vyhladávanie

Domovská stránka a stránka vyhľadávania obsahuje pole pre zadanie vyhladávacieho reťazca. Tento reťazec má definované nasledovné triedy ekvivalencie: `Škodlivý skript` a `Validný reťazec znakov`.

5.3.2.4 Rezervácia

Po zvolení prevádzky, v ktorej si chce zákazník vytvoriť rezerváciu, je užívateľovi zobrazený rezervačný formulár. Tento formulár obsahuje polie pre výber úkonu, ďalej

Názov poľa	Triedy Ekvivalencie
Užívateľské meno	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Nevalidná dĺžka reťazca, Registrované meno, Validný vstup
E-mailová adresa	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Nevalidný formát, Registrovaná e-mailová adresa, Validný vstup
Heslo	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Nevalidná dĺžka reťazca, Nevalidný počet podporovaných skupín znakov, Validný vstup
Potvrdenie hesla	Heslo sa nezhoduje, Heslo je správne
Osobné meno	Škodlivý skript, Prázdny reťazec, Validné meno
Priezvisko	Škodlivý skript, Prázdny reťazec, Validné meno
Telefónne číslo	Nevalidné znaky, Nevalidná dĺžka reťazca, Validné telefónne číslo
Polia Adresy	Škodlivý skript, Validný vstup

Tabuľka 5.2. Triedy ekvivalencie pre registráciu.

Názov poľa	Triedy Ekvivalencie
VIN číslo	Škodlivý skript, Prázdny reťazec, Validný vstup
Evidenčné číslo	Škodlivý skript, Prázdny reťazec, Validný vstup
Kategória	Škodlivý skript, Prázdny reťazec, Validný vstup
Značka	Škodlivý skript, Prázdny reťazec, Validný vstup
Model	Škodlivý skript, Valdiný vstup
Rok výroby	Nevalidné znaky, Škodlivý skript, Valdiný vstup
Palivo	Hodnota mimo rozsah, Hodnota z rozsahu
Výkon	Škodlivý skript, Valdiný vstup
Karoséria	Škodlivý skript, Validný vstup
Pohon	Hodnota mimo rozsah, Hodnota z rozsahu
Prevodovka	Hodnota mimo rozsah, Hodnota z rozsahu

Tabuľka 5.3. Triedy ekvivalencie pre vozidlo pri procese registrácie.

kalendár pre výber času a dátumu rezervácie, kontaktné údaje o zákazníkovi a informácie o vozidle ku ktorému sa rezervácia viaže. Pre registrovaného a prihláseného užívateľa sú tieto informácie vyplnené automaticky, ak má zákazník viacero vozidiel, tak má možnosť vybrať vozidlo pre rezerváciu. Neregistrovaný/Neprihlásený užívateľ je, pre úspešné dokončenie rezervácie, nútený vyplniť formulár s kontaktnými údajmi a informáciami o vozidle. Polia pre kontaktné informácie a vozidlo podliehajú rovnakej validácii ako tie pri registrácii a preto aj triedy ekvivalencie sú rovnaké ako pre totožné polia v registrácii. Triedy ekvivalencie pre spomínané hodnoty môžeme nájsť v tabulke 5.2 a 5.3.

Triedy ekvivalencie pre ostatné hodnoty formuláru rezervácie sú definované v tabuľke 5.4.

Názov poľa	Triedy Ekvivalencie
Úkon	Hodnoty mimo zoznam, Hodnoty zoznamu
Kalendár	Rezervovaný čas, Voľný čas

Tabuľka 5.4. Triedy ekvivalencie pre rezerváciu.

5.3.2.5 Úkony

Formulár prítomný pri vytváraní nového úkonu alebo úprave úkonu obsahuje polia pre zadanie názvu úkonu, výber kategórie, zadanie ceny a čas trvania úkonu. Triedy ekvivalencie, pre tieto hodnoty, sú definované v tabuľke 5.5.

Názov poľa	Triedy Ekvivalencie
Názov úkon	Prázdny reťazec, Škodlivý reťazec, Validný reťazec
Kategória	Prázdna hodnota, Hodnoty mimo zoznam, Hodnoty zoznamu
Cena	Prázdny reťazec, Nevalidné znaky a hodnoty, Validné hodnoty
Doba trvania	Prázdny reťazec, Nevalidné znaky a hodnoty, Validné hodnoty

Tabuľka 5.5. Triedy ekvivalencie pre úkon.

5.3.2.6 Informácie o prevádzke

Front-endova časť systému zatiaľ neposkytuje možnosť rezervácie novej prevádzky. Takáto prevádzka môže byť vytvorená len pomocou správneho dotazu na REST rozhranie. Systém však poskytuje upravu informácii o prevádzke. Formulár s informáciami o prevádzke obsahuje polia ktoré sú spolu s triedami ekvivalencie definované v tabuľke 5.6, nie však všetky hodnoty je možné editovať, preto spomínaná tabuľka tieto hodnoty neobsahuje.

Názov poľa	Triedy Ekvivalencie
E-mailová adresa	Nevalidné znaky, Škodlivý skript, Prázdny reťazec, Nevalidný formát, Registrovaná e-mailová adresa, Validný vstup
Telefónne číslo	Nevalidné znaky, Prázdny reťazec, Nevalidná dĺžka reťazca, Validné telefónne číslo
Typ prevádzky	Prázdna hodnota, Hodnoty mimo zoznam, Hodnoty zoznamu
Webová adresa	Škodlivý skript, Validné hodnoty
Servisná hodina	Nevalidné znaky a hodnoty, Validné hodnoty
IČO/DIČ	Prázdny reťazec, Škodlivý skript, Nevalidné znaky a hodnoty, Validné hodnoty
Počet zdvihákov	Prázdny reťazec, Nevalidné znaky a hodnoty, Validné hodnoty
Polia Adresy	Škodlivý skript, Validný vstup
Otváracie hodiny	Prázdny reťazec, Nevalidné znaky a hodnoty, Validné hodnoty

Tabuľka 5.6. Triedy ekvivalencie pre informácie o prevádzke.

Na základe tried ekvivalencie môžeme vygenerovať rôzne testovanie scenáre, podla zvolenej techniky dostaneme rôzne počty testov. Napríklad kombináciou vstupných dát pomocou techniky parwise dostaneme testy kde sa kombinujú každé dve hodnoty. To je užitočné pretože niekedy v aplikácii môžu vznikať chyby spôosbené kombináciou vstupných dát. Kombinácie vygenerované touto technikou môžeme nájsť v prílohe.

Iným spôsobom akým dostať vytvoriť testovacie scenáre sú techniky: MC/DC, CC, DC, C/DC. MC/DC je technika vytvárania testov kedy kombinujeme všetky vstupy

ovplyvňujúce výsledok rozhodovacieho výrazu. CC testy pokrývajú výsledok všetkých podmienok aspoň raz. DC testy pokrývajú výsledok všetkých rozhodnutí aspoň raz. C/DC testy pokrývajú výsledky všetkých podmienok rozhodnutia aspoň raz a každý z výsledkov rozhodnutia sa testuje aspoň raz.

Kombinácie testov týchto techník je možné nájsť v prílohe.

5.4 Akceptačné užívateľské testovanie

V rámci testovania prototypu, sme sa dostali aj k testovaniu u prevádzok. Pre tieto testovacie účely bola aplikácia nasadená na cloudovom prostredí Google Kubernetes. Testovanie prebehlo v troch rôznych Bratislavských servisoch, a to konkrétne v Auto-Drgi, MaximServis a Môj-Servis.

Testovanie v prevádzkach MaximServis a Môj-Servis prebehlo v druhom aprílovom týždni.

V MaximServise boli pri testovaní prítomní dvaja majitelia servisu, ktorí si spolu prešli základný scenár prihlásenia do systému. Ďalej si vyskúšali vytvoriť novú rezerváciu a následne zobraziť túto rezerváciu v rozvrhu. Pri vytváraní novej rezervácie mali pár poznámok a otázok na budúce podporované funkcionality systému. Napríklad by im vyhovovalo, kebyže je pri zadávaní novej rezervácie možné vyhľadať konkrétne vozidlo zákazníka podľa evidenčnej značky, čo systém momentálne nepodporuje. V čase testovania v MaximServise, systém podporoval vytváranie nových rezervácií prevádzkou jedine na konkrétnej adrese, na ktorú sa bolo možné dostať kliknutím na správnu záložku v bočnom menu, a to prevádzke prišlo troch nepraktické. Na základe tohto komentára sa pri ďalšom vývoji doplnila možnosť vytvorenia rezervácie priamo kliknutím na voľný termín v rozvrhu.

Nasledujúcou prevádzkou, v ktorej prebehlo testovanie bol Môj-Servis, toto testovanie prebehlo v rovnaký deň ako testovanie v MaximServise. Priebeh testovania bol dosť podobný ako v predchádzajúcej prevádzke, kedy si majiteľ, ktorý je zároveň prímacím technikom [1] v prevádzke, otestoval aplikáciu. Rovnako ako v MaximServise, si prešiel prihlásenie do aplikácie, vytvorenie novej registrácie, zobrazenie rozvrhu. Navyše si skúsil zmeniť informácie o testovacej prevádzke a pridať nové servisné úkony. Navyše si prešiel aj užívateľskú časť aplikácie, aby videl s čím budú pracovať ľudia na druhej strane. K aplikácii mal podobné poznámky ako predchádzajúca prevádzka.

Poslednou prevádzkou, kde bola aplikácia testovaná, bol servis AutoDrgi. Testovanie v tomto servise prebehlo začiatkom mája, kedy systém obsahoval viac funkcionalít, ako pri testovaní v predchádzajúcich dvoch prevádzkach. Táto prevádzka bola s aplikáciou veľmi spokojná, páčilo sa im ako vyzerá a akým spôsobom funguje, zhodovalo sa to s ich predstavou takéhoto systému. Jediná vec, o ktorú mali starosť bola zabezpečenie stability systému, aby neprišli o rozvrh nadchádzajúcich rezervácií. Rovnako, ako pri testovaní, tak aj pri produkčnom nasadení by mal byť systém nasadený na Google Kubernetes prostredí, čiže stabilita systému bude závislá na stabilite prostredia kubernetes, čiže nepredpokladáme, že by mali byť veľké problémy po tejto stránke.

Akceptačné testovanie prebiehalo na počítačoch v prevádzkach.

[1] Prímací technik je osoba, ktorá by používala aplikáciu kebyže je nasadená na produkcii.

Kapitola 6
Záver

Cieľom práce bolo navrhnúť systém, ktorý sa zaoberá správou autoservisov a ich ponukou služieb zákazníkom. Navrhnutý systém mal uľahčiť komunikáciu medzi zákazníkom a prevádzkou. Zároveň mala zefektívniť vybrané administratívne úkony prevádzky.

6.1 Zhodnotenie

Zadaním práce bolo i skúmanie existujúcich riešení. Na základe získaných informácií bola urobená analýza problematiky, z ktorej vyplynuli funkčné a nefunkčné požiadavky na požadovanú aplikáciu. Pri vypracovávaní požiadaviek boli zohľadnené požiadavky od spoločnosti ako aj požiadavky získané od konkrétnych autoservisov, s ktorými je firma v kontakte. Rovnako boli zohľadnené technické aspekty podľa vedomostí a skúseností autora práce.

V kapitole analýzy sa takisto rozoberali užívateľské prípady a možný výber softvérovej architektúry, ktorú by bolo vhodné zvoliť pri návrhu aplikácie.

Pre aplikáciu bola zvolená architektúra mikroservis a v kapitole Návrh môžeme vidieť navrhnutú aplikáciu, ktorá rešpektuje zvolený architektonický vzor. V kapitole návrhu je súčasne rozobraný aj návrh a dizajn grafického užívateľského rozhrania.

Jedna časť z práce bola implementácie navrhnutej aplikácie. Implementácie bola rozdelená na dva celky, kedy autor práce nezávisle na sebe vyvíjal back-endovú a front-endovú časť aplikácie. Serverová časť aplikácia je rozdelená do niekoľkých samostatne fungujúcich častí, servis, ktoré boli programované v jazyku Java, a to, čo najviac nezávislé na sebe. Implementované servisy pokrývajú špecifikácie definované v kapitole Návrh. Súčasťou implementácie bola tvorba JUnit testov.

Užívateľské prostredie, ktoré bolo vyvíjané samostatne a nezávisle na servisnej časti aplikácie, bolo napísané v TypeScript s využitím rámca Angular. Implementácia užívateľského bohužiaľ nepokrýva tak veľkú časť návrhu ako serverová časť, zadanie však splnené bolo. Menšie pokrytie je hlavne spôsobené nie príliš dobrým odhadom práce od autora práce a autorovým nedostatkom skúseností ako s vývojom grafickej časti aplikácie tak so samotným rámcom použitím v práci.

Testovanie užívateľského rozhrania prebiehalo manuálnym testovaním autora práce a užívateľským testovaním v troch prevádzkach. Testovanie v prevádzkach ukázalo, že aplikácia potrebuje niektoré vylepšenia pred zavedením do každodenného používania.

Ničmenej, úlohou bolo implementovať webovú aplikáciu, na ktorej by išlo odskúšať či je možné vytvoriť takýto systém. Zároveň či pretrvá záujem o aplikáciu zo strany prevádzok po predvedení funkčného systému.

Tieto ciele boli splnené, keďže aplikácia momentálne podporuje základný scenár odrážajúci hlavnú myšlienku aplikácie, a to nájsť prevádzku a vytvoriť si u nej rezerváciu. Prevádzky, v ktorých bola aplikácia ukazovaná a testovaná vyjadrili záujem o takýto systém a boli naklonené k ďalšej spolupráci. Pri testovaní v prevádzkach bola aplikácia spustená a nasadená na Google Kubernetes prostredí.

6.2 Možnosti budúceho vývoja

Prototyp aplikácia aktuálne poskytuje len časť funkcionalít definovaných v kapitolách návrhu a analýzy. Obsahuje však základný scenár, ktorí by sme od nej očakávali, a to vytvorenie rezervácie v prevádzke. Rovnako poskytuje prevádzkam definovať poskytované úkony a rozvrh s rezerváciami. Zákazník sa môže do aplikácie registrovať, pridať si vozidlo (ako súčasť registrácie) a vytvoriť rezerváciu.

Pre možné spustenie systému do produkcie by bolo potrebné rozšíriť implementáciu užívateľského rozhrania o funkcionality aktuálne podporované back-endom, a to menovite zobrazenie detailu prevádzky zákazníkmi (aktuálne, zákazník vidí len malý náhľad). Nastavenia zákazníka, kde by si mohol spravovať údaje o sebe, svoje vozidlá, kde by mohol vidieť svoje nadchádzajúce rezervácie a kde by mohol vidieť ako históriu rezervácii tak históriu úkonov vykonaných na jeho vozidlách. Užívateľské rozhranie zatiaľ nemá implementovanú žiadnu časť administrátorského rozhrania, čo znamená, že nie je možné registrovať nové prevádzky cez užívateľské rozhranie, ich pridávanie je možné len priamym dotazom na REST rozhranie. Systém zatiaľ nepodporuje pridávanie rezervácii, posielanie emailov alebo notifikácii v rámci aplikácie.

V dnešnej dobe kedy vieme skoro všetko vybaviť cez mobilný telefón by bol ďalším logickým krokom vytvoriť aplikáciu pre mobilné platformy, ako sú mobilné telefóny a tablety. Tento krok bol pri návrhu celkovej aplikácie zohľadnený a odráža sa v kompletnom oddelení návrhu a implementácie serverovej a užívateľskej časti aplikácie.

Literatúra

[1] Info@finstat.sk. *Databáza slovenských firiem a organizácií: banskobystrický kraj*. https://finstat.sk/databaza-firiem-organizacii?sknace=45000,45200,22110&Region=banskobystrický&District=banská bystrica&Sort=creation-date-desc. Stav k 16.12.2018.

[2] Paul Gil. *SaaS: What Is Software as a Service?* https://www.lifewire.com/what-is-saas-software-2483600. Stav k 20.1.2019.

[3] Jim Arlow a Ila Neustadt. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Computer Press, 2007.

[4] Pavel Kutáč. *Přechod na bezpečnější bcrypt*. 2017. https://www.kutac.cz/blog/weby-a-vse-okolo/prechod-na-bezpecnejsi-bcrypt/. Stav k 29.1.2019.

[5] *Securing Passwords with Bcrypt Hashing Function*. 2014. https://thehackernews.com/2014/04/securing-passwords-with-bcrypt-hashing.html. Stav k 29.1.2019.

[6] Ian Sommerville. *Software Engineering*. TENTH vydanie. Pearson, 2016.

[7] Mark Richards. *Software Architecture Patterns*. O'Reilly Media, Inc., 2015. https://www.oreilly.com/programming/free/software-architecture-patterns.csp.

[8] Peter Wayner. *How to choose the right software architecture: The top 5 patterns*. https://bit.ly/2WQ49Jh. Stav k 27.1.2019.

[9] Heiko Schuldt. *Multi-Tier Architecture*. In: *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009. 1862–1865. ISBN 978-0-387-39940-9. https://doi.org/10.1007/978-0-387-39940-9_652.

[10] Grzegorz Ziemonski. *Layered Architecture Is Good - DZone Java*. 2017. https://dzone.com/articles/layered-architecture-is-good. Stav k 27.1.2019.

[11] *Explain the benefits of layered architecture?* http://www.answers.com/Q/Explain_the_benefits_of_layered_architecture. Stav k 27.1.2019.

[12]

[13] *What are the advantages and disadvantages of a layered architecture*. 2017. http://venkataspinterview.blogspot.com/2011/03/what-are-advantages-and-disadvantages.html. Stav k 27.1.2019.

[14] Davide Taibi, Valentina Lenarduzzi a Claus Pahl. *Architectural Patterns for Microservices: A Systematic Mapping Study*. In: 2018.

[15] Chris Richardson. *What are microservices?* https://microservices.io/. Stav k 27.1.2019.

[16] Tom Huston. *What is Microservices Architecture?* https://smartbear.com/learn/api-design/what-are-microservices/. Stav k 27.1.2019.

[17] Stackify. *What are Microservices? Code Examples, Best Practices, Tutorials and More*. 2017.
https://stackify.com/what-are-microservices/. Stav k 27.1.2019.

[18] Guy Hummel. *Microservices Architecture: Advantages and Drawbacks*. 2015.
https://bit.ly/2Js3uL0. Stav k 27.1.2019.

[19] Margaret Rouse. *What is API gateway? - Definition from WhatIs.com*. 2018.
https://bit.ly/2ANOgOw. Stav k 28.1.2019.

[20] Chris Richardson. *Building Microservices Using an API Gateway*. 2015.
https://www.nginx.com/blog/building-microservices-using-an-api-gateway/.
Stav k 28.1.2019.

[21] *We Measure Your Software Code Quality*.
https://www.tiobe.com/. Stav k 14.4.2019.

[22] Brett Porter, Jason van Zyl a Olivier Lamy. *Maven – Welcome to Apache Maven*.
https://maven.apache.org/. Stav k 20.4.2019.

[23] *Engines Ranking*.
https://db-engines.com/en/ranking. Stav k 20.4.2019.

[24] Amey Varangaonkar. *Why MongoDB is the most popular NoSQL database today*. 2018.
https://hub.packtpub.com/mongodb-popular-nosql-database-today/. Stav k 20.4.2019.

[25] *Angular*.
https://angular.io/docs. Stav k 20.4.2019.

[26] *The new major version of the programmer-friendly testing framework for Java*.
https://junit.org/. Stav k 9.5.2019.

[27] *Mockito framework site*.
https://site.mockito.org/. Stav k 9.5.2019.

[28] *Java Code Coverage*.
https://java.libhunt.com/categories/347-code-coverage. Stav k 9.5.2019.

[29] *JaCoCo Java Code Coverage Library*. 2019.
https://www.jacoco.org/jacoco/index.html. Stav k 9.5.2019.

[30] *API Development Environment*.
https://www.getpostman.com/. Stav k 10.5.2019.

[31] *Selenium Training in Chennai - Best Selenium Training Institute in Chennai*.
https://www.credosystemz.com/training-in-chennai/best-selenium-training-in-chennai/. Stav k 10.5.2019.

[32] *Jasmine Documentation*.
https://jasmine.github.io/. Stav k 10.5.2019.

[33] A. R. *Automated GUI testing will kill you.* 2014.
http://arbitraryreason.com/automated-gui-testing-will-kill-you/. Stav k 10.5.2019.

[34] *Testování ekvivalence*.
http://test.swtestovani.cz/index.php?option=com_content&view=article&id=28:testovani-ekvivalence&catid=13:testovaci-techniky&Itemid=29. Stav k 11.5.2019.

[35] DOC. ING. Miroslav Bureš Ph.D. *Software Quality Assurance*. 2017. Absolvovanie predmetu B4M36ZKS v Zime 2017.

Príloha A
Slovník

API	Application Programming Interface
C/DC	Condition/Decision Coverage
CC	Condition Coverage
CPU	Central Processing Unit
CSRF	Cross-site request forgery
DC	Decision Coverage
DIČ	Daňové Identifikačné Číslo
GUI	Graphical User Interface
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
IČO	Identifikačné Číslo Osoby
ID	Identifier
IP	Internet Protocol
JaCoCo	Java Code Coverage
JSON	JavaScript Object Notation
JWT	JSON Web Token
kW	kilo Watt
MC/DC	Multiple Condition/Decision Coverage
MVC	Model View Controller
OS	Operating System
RDBMS	Relational Database Management System
REST	Representational state transfer
SaaS	Software as a Service
SQL	Structured Query Language
SSD	Solid-State Drive
TIOBE	The Importance of Being Earnest
URL	Unified Resource Locator
VIN	Vehicle Identification Number
XSS	Cross-site scripting

Príloha B
Testovacie Oblasti

Vysvetlenie skratiek použitých v nasledujúcich tabuľkách:

- MP - Možné poškodenie
- H - High (Vysoké)
- M - Medium (Stredné)
- L - Low (Nízke)
- PZ - Pravdepodobnosť Zlyhania
- TR - Trieda Rizika
- UAT - User Acceptance Testing
- FE - Front-End

Proces	Podproces	Požiadavka	MP	Vysvetlenie
Prihlásenie	Zákazník	Prihlásenie do systému	L	Bez prihlásenia zákazník nemôže využívať funkcionality poskytované registrovaným užívateľom.
Prihlásenie	Prevádzka	Prihlásenie do systému	L	Bez prihlásenia prevádzka nemôže vidieť jej rezervácie.
Registrácia zákazníka	-	Registrácia nového zákazníka	M	Nefunkčnosť spôsobí že nebudú pribúdať nový zákazníci, čo môže viesť k nižšej popularite aplikácie.
Vyhľadanie prevádzok	-	Vyhladanie zoznamu prevádzok	M	Nefunkčnosť znamená strata popularity stránky, čo vedie k strate klientov.
Vytvorenie rezervácie	Neregistrovaný zákazník	Vytvorenie novej rezervácie	H	Nefunkčnosť znamená strata popularity stránky, čo vedie k strate klientov.
Vytvorenie rezervácie	Registrovaný zákazník	Vytvorenie novej rezervácie	H	Nefunkčnosť znamená strata popularity stránky, čo vedie k strate klientov.
Vytvorenie rezervácie	Prevádzka	Vytvorenie novej rezervácie	H	Nefunkčnosť znamená strata ochromenie prevádzky a možnú stratu financií.
Úprava informácii o prevádzke	-	Úprava informácii	L	Nefunkčnosť znamená nemožnosť úpravy informácii ako napríklad otváracia doba čo môže viesť k nepresnosti vytvorených rezervácii.

Proces	Podproces	Požiadavka	MP	Vysvetlenie
Spracovanie úkonov	Pridanie nového	Pridanie nového úkonu	M	Nefunkčnosť znamená nerozšírenie poskytovaných služieb prevádzkou, čo môže viesť k finančným stratám.
Spracovanie úkonov	Úprava	Úprava existujúceho úkonu	M	Nefunkčnosť znamená nepresnosť v úkonoch čo môže viesť k nesprávnym odhadom ceny práce.
Spracovanie úkonov	Mazanie	Mazanie existujúceho úkonu	M	Nefunkčnosť znamená že zákazník môže rezervovať úkon ktorý nie je ďalej poskytovaný prevádzkou, a viesť k nespokojnosti zákazníka.
Odhlásenie	-	Odhlásenie zo systému	M	Pri nefunkčnosti môže dôjsť k úniku informácií.

Tabuľka B.1. Prioritizácia testovacích oblasti.

Proces	Podproces	PZ	Vysvetlenie
Prihlásenie	Zákazník	L	Pravdepodobnosť je nízka, zlyhanie môže byť spôsobené buď databázou alebo komunikáciou mezi gateway a servisou.
Prihlásenie	Prevádzka	L	Pravdepodobnosť je nízka, zlyhanie môže byť spôsobené buď databázou alebo komunikáciou mezi gateway a servisou.
Registrácia zákazníka	-	M	Formulár obsahuje veľa vstupov ktorých kombinácia by mohla spôsobiť chybu.
Vyhľadanie prevádzok	-	M	Chyba môže nastať pri divnom vyhladávacom výraze.
Vytvorenie rezervácie	Neregistrovaný zákazník	H	Formulár obsahuje pomerne veľa vstupov, kalendár sa môže vygenerovať nesprávne.
Vytvorenie rezervácie	Registrovaný zákazník	M	Kalendár sa môže vygenerovať nesprávne.
Vytvorenie rezervácie	Prevádzka	H	Formulár obsahuje pomerne veľa vstupov, kalendár sa môže vygenerovať nesprávne.
Úprava informácii o prevádzke	-	M	Formulár obsahuje pomerne veľa vstupov.
Spracovanie úkonov	Pridanie nového	L	Malý formulár, jednoduchý úkon.
Spracovanie úkonov	Úprava	L	Malý formulár, jednoduchý úkon.
Spracovanie úkonov	Mazanie	L	Jednoduchý úkon.
Odhlásenie	-	L	Jednoduchý úkon, vykonávaný len na FE (zmazanie tokenu).

Tabuľka B.2. Prioritizácia testovacích oblasti.

Možnosť poškodenia / Pravdepodobnosť zlyhania	H	M	L
H	A	B	B
M	B	B	C
L	B	C	C

Tabuľka B.3. Triedy rizika.

Proces	Podproces	TR	UAT
Prihlásenie	Zákazník	C	Nízka
Prihlásenie	Prevádzka	C	Nízka
Registrácia zákazníka	-	B	Stradná
Vyhľadanie prevádzok	-	B	Stradná
Vytvorenie rezervácie	Neregistrovaný zákazník	A	Vysoká
Vytvorenie rezervácie	Registrovaný zákazník	B	Stradná
Vytvorenie rezervácie	Prevádzka	A	Vysoká
Úprava informácii o prevádzke	-	C	Nízka
Spracovanie úkonov	Pridanie nového	C	Nízka
Spracovanie úkonov	Úprava	C	Nízka
Spracovanie úkonov	Mazanie	C	Nízka
Odhlásenie	-	C	Nízka

Tabuľka B.4. Intenzita testov.

Proces	Podproces	TR	Zvolená technika
Prihlásenie	Zákazník	C	Akceptačné FE testy
Prihlásenie	Prevádzka	C	Akceptačné FE testy
Registrácia zákazníka	-	B	Akceptačné FE testy
Vyhľadanie prevádzok	-	B	Akceptačné FE testy
Vytvorenie rezervácie	Neregistrovaný zákazník	A	Akceptačné FE testy
Vytvorenie rezervácie	Registrovaný zákazník	B	Akceptačné FE testy
Vytvorenie rezervácie	Prevádzka	A	Akceptačné FE testy
Úprava informácii o prevádzke	-	C	Akceptačné FE testy
Spracovanie úkonov	Pridanie nového	C	Akceptačné FE testy
Spracovanie úkonov	Úprava	C	Akceptačné FE testy
Spracovanie úkonov	Mazanie	C	Akceptačné FE testy
Odhlásenie	-	C	Akceptačné FE testy

Tabuľka B.5. Testovanice techniky.

Príloha C
Testovacie Scenáre

Tabuľky s pairwise scenármi sú príliš veľké, aby sa prehľadne zmestili na jednu stranu, preto sú umiestnené v súbore pairwise.xlsx v priloženom CD.

Prihlásenie:

- Podmienky
  - A - Užívateľské meno je validné a existuje v systéme
  - B - Heslo je validné a existuje v systéme
- Rozhodnutie
  - A & B - Je prihlásenie validné?

R = A and B	1	0
A	11	01
B	~~11~~	10

Tabuľka C.6. MC / DC testovacie scenáre Prihlasenia.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.7. C / DC testovacie scenáre Prihlasenia.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.8. CC testovacie scenáre Prihlasenia.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.9. DC testovacie scenáre Prihlasenia.

Registrácia:

- Podmienky
  - A - Užívateľské meno je validné a neexistuje v systéme
  - B - Emailová adresa je validná a neexistuje v systéme
  - C - Heslo je validné
  - D - Potvrdenie hesla je validné
  - E - Osobné meno je validné
  - F - Priezvisko je validné
  - G - Telefónne číslo je validné
  - H - Adresa je validná
- Rozhodnutie
  - A & B & C & D & E & F & G & H - Je registrácia validná?

R = A and B	1	0
A	11111111	01111111
B	11111111	10111111
C	11111111	11011111
D	11111111	11101111
E	11111111	11110111
F	11111111	11111011
G	11111111	11111101
H	11111111	11111110

Tabuľka C.10. MC / DC testovacie scenáre Registrácie.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0

Tabuľka C.11. C / DC testovacie scenáre Registrácie.

Registrácia - Vozidlo:

- Podmienky
  - A - VIN číslo je validné
  - B - Evidenčné číslo je validné
  - C - Kategória je validná
  - D - Značka je validná
  - E - Model je validný
  - F - Rok výroby je validný

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0

Tabuľka C.12. CC testovacie scenáre Registrácie.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0

Tabuľka C.13. DC testovacie scenáre Registrácie.

- G - Palivo je validné
- H - Výkon je validný
- I - Karoséria je validná
- J - Pohon je validný
- K - Prevodovka je validná

- Rozhodnutie
  - A & B & C & D & E & F & G & H & I & J & K - Je registrácia vozidla validná?

R = A and B	1	0
A	11111111111	01111111111
B	~~11111111111~~	10111111111
C	~~11111111111~~	11011111111
D	~~11111111111~~	11101111111
E	~~11111111111~~	11110111111
F	~~11111111111~~	11111011111
G	~~11111111111~~	11111101111
H	~~11111111111~~	11111110111
I	~~11111111111~~	11111111011
J	~~11111111111~~	11111111101
K	~~11111111111~~	11111111110

Tabuľka C.14. MC / DC testovacie scenáre Registrácie vozidla.

Rezervace:

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0
K	1	0

Tabuľka C.15. C / DC testovacie scenáre Registrácie vozidla.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0
K	1	0

Tabuľka C.16. CC testovacie scenáre Registrácie vozidla.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0
K	1	0

Tabuľka C.17. DC testovacie scenáre Registrácie vozidla.

- Podmienky
  - A - Zvolený úkon je vylidný a existuje
  - B - Zvolený dátum a čas je vaildný a volný
- Rozhodnutie

- A & B - Je možné vytvoriť rezerváciu?

R = A and B	1	0
A	11	01
B	~~11~~	10

Tabuľka C.18. MC / DC testovacie scenáre Rezerváciu.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.19. C / DC testovacie scenáre Rezerváciu.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.20. CC testovacie scenáre Rezerváciu.

Premenná / Výstup	1	0
A	1	0
B	1	0

Tabuľka C.21. DC testovacie scenáre Rezerváciu.

Úkon:

- Podmienky
  - A - Názov je validný
  - B - Kategória je validná
  - C - Cena je validná
  - D - Doba trvania je validná
- Rozhodnutie
  - A & B & C & D - Je vytvorenie/úprava úkonu validná?

R = A and B	1	0
A	1111	0111
B	~~1111~~	1011
C	~~1111~~	1101
D	~~1111~~	1110

Tabuľka C.22. MC / DC testovacie scenáre Úkonu.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0

Tabuľka C.23. C / DC testovacie scenáre Úkonu.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0

Tabuľka C.24. CC testovacie scenáre Úkonu.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0

Tabuľka C.25. DC testovacie scenáre Úkonu.

Prevádzka:

- Podmienky
  - A - Emailová adresa je validná a v systéme neexistuje
  - B - Tefefónne číslo je validné
  - C - Typ prevádzky je validný
  - D - Webová adresa je validná
  - E - Servisná hodina je validná
  - F - IČO/DIČ je validný
  - G - Počet zdvihákov je validný
  - H - Adresa je validná
  - I - Otváracie hodiny sú validné
- Rozhodnutie
  - A & B & C & D & E & F & G & H & I - Je úprava prevádzky validná?

R = A and B	1	0
A	111111111	011111111
B	111111111	101111111
C	111111111	110111111
D	111111111	111011111
E	111111111	111101111
F	111111111	111110111
G	111111111	111111011
H	111111111	111111101
I	111111111	111111110

Tabuľka C.26. MC / DC testovacie scenáre pre Prevádzku.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0

Tabuľka C.27. C / DC testovacie scenáre pre Prevádzku.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0

Tabuľka C.28. CC testovacie scenáre pre Prevádzku.

Premenná / Výstup	1	0
A	1	0
B	1	0
C	1	0
D	1	0
E	1	0
F	1	0
G	1	0
H	1	0
I	1	0

Tabuľka C.29. DC testovacie scenáre pre Prevádzku.

Príloha D
Obsah pribaleného CD

app.zip - zdrojový kód aplikácie
pairwise.xlsx – excel s vygenerovanými testovacími prípadmi
thesis.zip - LaTeX kód diplomovej práce