**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Diplomová práce

## Informační systém pro servis osobních vozidel

**David Rigl**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. David Rigl

Informatika

Název práce

**Informační systém pro servis osobních vozidel**

Název anglicky

**Information system for personal vehicle service**

---

**Cíle práce**

Cílem projektu je vytvoření informačního systému pro servis vozidel zaměřený na průběžnou komunikaci servis – zákazník. Systém umožní dokumentaci veškerých zásahů a oprav pro dané vozidlo a plánování budoucích oprav a prohlídek vozu.

**Metodika**

V teoretické části budou popsány možnosti realizace jednotlivých součástí systému. Stručně bude probrána bezpečnostní stránka systému. Dále budou rozebrány a popsány nástroje sloužící pro návrh a vývoj systému.

V praktické části dojde na analýzu požadavků na tento informační systém. Dále bude proveden jeho návrh a příprava backendové a frontendové části systému. Dalším krokem bude jeho implementace s následným testováním aplikace a otestování veškerých funkcí. Po otestování bude aplikace připravena k nasazení u vybraného podniku v provozu.

**Doporučený rozsah práce**

50-60 stran

**Klíčová slova**

Servis vozidel, Server, MySQL, HTML5, NodeJS, Linux, SCSS, Typescript

---

**Doporučené zdroje informací**


CASTRO, Elizabeth; HYSLOP, Bruce. *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek.* Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

KARVINEN, Tero, Kimmo KARVINEN a Ville VALTOKARI. Make: sensors. Sebastopol, CA: Maker Media, 2014. ISBN 9351106373

LUBBERS, Peter; ALBERS, Brian; SALIM, Frank. *HTML5 : programujeme moderní webové aplikace.* Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

ROSEBROCK, E, FILSON, E. Linux, Apache, MySQL a PHP – Instalace a konfigurace prostředí pro pokročilé webové aplikace. 2005. ISBN 80-247-1260-1.

ULLMAN, L. PHP – pokročilé programování pro World Wide Web. 2003. ISBN: 80-86497-36-4

WILLIAMS, Hugh E.; KRÁSENSKÝ, David; LANE, David. *PHP a MySQL : vytváříme webové databázové aplikace : podrobný průvodce tvůrce WWW stránek.* Praha: Computer Press, 2002. ISBN 80-7226-760-4.


---

**Předběžný termín obhajoby**

2023/24 LS – PEF

**Vedoucí práce**

Ing. Marek Pícka, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 30. 11. 2023

**doc. Ing. Tomáš Šubrt, Ph.D.**

Děkan

V Praze dne 24. 03. 2024

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Informační systém pro servis osobních vozidel" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.03.2024 _____

**Poděkování**

Rád bych touto cestou poděkoval panu Ing. Marku Píckovi Ph.D. za vstřícný přístup při realizaci této diplomové práce, za odborné rady, příkladné vedení a především za neskutečnou trpělivost při konzultacích.

# Informační systém pro servis osobních vozidel


## Abstrakt

Cílem práce je návrh a realizace informačního systému sloužícího pro servis motocyklů a osobních vozidel se zaměřením na malé soukromé podnikatele a firmy. Tento systém má za cíl zjednodušit komunikaci mezi servisem a zákazníkem a umožnit zákazníkovi vidět zdokumentovaný průběh a výsledek servisních zákroků na daném vozidle v podobě fotografií a k nim přiřazeným komentářům od mechanika. V systému bude ze strany mechanika možné jednoduše dohledávat historii oprav a plánovat budoucí opravy či kontroly. Ze strany zákazníka bude v systému umožněno si vést aktuální údaje o svém vozidle či vozidlech a navíc bude systém automaticky včas upozorňovat zákazníka na blížící se termín konce platnosti technické kontroly vozu.

Práce je rozdělena do dvou částí, a to teoretické a praktické. V teoretické části budou popsány všechny součásti systému a jejich varianty, dále rozebrány možnosti použití technologií, na kterých může systém fungovat a jejich výhody či nevýhody. Následovat bude stručný rozbor bezpečnostní stránky systémů v podobě hrozeb a ochrany.

V praktické části dojde na analýzu požadavků potencionálního zájemce o tento systém. Dále bude proveden návrh systému pomocí zvolených metod a diagramů. Po návrhu systému dojde na jeho postupné zprovoznění od nastavení základních služeb pro veřejně dostupný systém až po tvorbu backendu a následně frontendu systému. V projektu nebude chybět zobrazování průběhu postupné realizace celého systému a na závěr dojde na otestování všech funkcí.



**Klíčová slova:** Servis vozidel, Server, MySQL, HTML5, NodeJS, Linux, SCSS, Typescript

# Information system for personal vehicle service


## Abstract

The aim of the work is the design and implementation of an information system used for servicing motorcycles and passenger vehicles with focus on small private entrepreneurs and companies. The aim of this system is to simplify communication between the service and the customer and enable the customer to see the documented course and result of service interventions on the given vehicle in the form of photographs and comments from the mechanic. In the system the mechanic will be able to simply look up the repair history and plan future repairs or inspections.

The customer will be able to keep up-to-date data on their vehicle or vehicles in the system, and in addition, the system will automatically notify the customer in time of the approaching expiration date of the vehicle's technical inspection.The work is divided into two parts, theoretical and practical. In the theoretical part, all the components of the system and their variants will be described, the possibilities of using technologies on which the system can work, and their advantages and disadvantages will be discussed. This will be followed by a brief analysis of the security side of the systems in the form of threats and protection.

The requirements of a potential interested party for this system will be analyzed in the practical part. Furthermore, the design of the system will be carried out using the selected methods and diagrams. After the design of the system, it will be put into operation gradually, from the setting of basic services for the publicly available system to the creation of the backend and then the frontend of the system. Displaying the progress of the gradual implementation of the entire system will not be missing in the project, and at the end all functions will be tested.



**Keywords**: Vehicle Service, Server, MySQL, HTML5, NodeJS, Linux, SCSS, Typescript

# Obsah

# 1 Úvod

V dnešní moderní době, co se týče informačních technologií téměř není možné úspěšně fungovat bez jejich používání. Většina servisů vozidel již také využívá moderní technologie pro různé kontroly, evidence a plno dalších činností. Velmi často klienti ani nevědí, co již bylo na jejich vozidle provedeno za servisní práce a co bude teprve nutné provést. Tento projekt je zaměřen na menší servisy, které chtějí udržovat stálý kontakt se svými klienty a řešit veškeré potřebné situace operativně v danou chvíli, s možností poskytnutí aktuálního stavu vozidla například prostřednictvím fotografie do společného chatu s majitelem vozu a řešit s ním veškeré otázky na jednom místě.

Zrealizovaný systém a jeho databáze by měly být konstruovaný tak, aby udržovaly informace o jednotlivých vozech přehledně rozdělené podle majitelů a bylo možné je filtrovat podle vhodných parametrů či uživatelů. V systému budou následně po registraci uživatelů vytvořeny jejich uživatelské účty propojené s jejich následně vytvořenými vozidly, o kterých bude systém evidovat informace v podobě historie oprav, naplánovaných aktuálních a budoucích servisních prohlídek či zásahů a podobně. Systém bude umožňovat včasnou notifikaci zákazníkovi o blížícím se konci platnosti technické prohlídky vozu.

Tento systém by měl zefektivnit celkovou komunikaci mezi servisem a zákazníkem. Zároveň by měl umožnit dokumentaci a popis servisních činností na vozidlech klientů. Pro účely zájemce o tento systém musí být systém především přehledný a jednoduchý, aby nezpomaloval činnost servisu a musí splnit všechny požadavky, které budou stanoveny. Pro implemetaci takového systému je nutné provést vhodnou analýzu požadavků, na základě kterých bude zpracován odpovídající návrh.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Cílem projetu je vytvoření informačního systému pro servis vozidel zaměřený na průběžnou komunikaci servis - zákazník. Systém umožní dokumentaci veškerých zásahů a oprav pro dané vozidlo a plánování budoucích oprav a prohlídek vozu. Dále bude umožněno komunikovat zákazníkovi přímo v systému s mechanikem a v rámci jednotlivých funkcí systému jej využívat pro přehledné vedení aktuálních údajů o daném vozidle. Mechanikovi bude umožněna správa všech jednotlivých událostí naplánovaných na jednotlivých vozidlech a dokumentace všech servisních zákroků.

## 2.2 Metodika

V teoretické části budou popsány možnosti realizace jednotlivých součástí systému a rozebrány důvody, výhody a nevýhody jejich použití. Stručně bude probrána bezpečnostní stránka systému. Dále budou rozebrány a popsány nástroje sloužící pro návrh a vývoj systému a důvody jejich použití.

V praktické části dojde na analýzu požadavků potencionálního zájemce o tento informační systém a stanovení funkcionalit. Dále bude proveden základní návrh systému pomocí zvolených metod a dojde na přípravu backendové a frontendové části systému. Funkčnost systému se bude zaměřovat na požadavky potencionálního zákazníka, jeho základní návrh bude však zaměřen na možnost využití systému u více firem. Velký důraz při návrhu backendové části bude kladen na konstrukci databáze, na které je celý systém silně závislý. Dále bude podstatným krokem zvolit vhodné a zároveň jednoduše použitelné metody pro funkcionality v systému. Po vytvoření celkového návrhu dojde na základní nastavení serveru pro zviditelnění systému do internetu a následně na implementaci obou částí se zprovozněním veškerých funkcí požadovaných zákazníkem. Dalším krokem bude testování systému a veškerých jednotlivých funkcí jak ze strany mechanika, tak ze strany uživatelů (klientů). Po otestování všech funkcionalit bude systém připraven pro vzhledovou úpravu dle požadavků zákazníka před případným nasazení systému do podniku.

# 3 Teoretická východiska

## 3.1 Základní informace

Jak už bylo v úvodní části zmíněno, v současnosti využívají informační technologie i servisy osobních vozidel, a to kvůli evidenci zakázek, plánovací kalendář, vlastní webové stránky s propagací vlastní činnosti a mnoho dalších. Dnešní možnosti systémů či aplikací jsou již velmi rozšířené a dokáží servisu velmi zjednodušit jeho činnost i co se týče logistické sféry, nákupu nových dílů, plánování budoucích servisních úkonů na vozidlech a mnoho dalších funkcionalit.

Existuje mnoho různých firem, které nabízí hotová řešení přizpůsobená přímo pro daný servis, zabezpečující podporu a udržitelnost systému aktuální. Tato řešení však bývají velmi nákladná a pro některé soukromé podnikatele cenově nedostupná. Pro některé se dá říci až zbytečná, zároveň to však může snižovat důvěryhodnost samotného servisu, či zbytečně snižovat jeho klientelu, pokud není alespoň webovými stránkami daný servis propagován.

Podstatou vytvořeného systému je pro soukromého podnikatele, aby mu zautomatizoval manuální činnosti a zjednodušil či zrychlil činnost mimo dílnu a zároveň umožnil komunikaci se zákazníkem.

## 3.2 Webová aplikace

Webová aplikace je typ software, který je zprostředkovaný pomocí internetového prohlížeče. Díky rozvoji vysokorychlostního internetu poslední dobou tento typ aplikace bývá velmi oblíbený díky velkým možnostem automatizace, přístupnosti, nezávislosti na operačním systému a mnoho dalších. [1]

Obrovským benefitem tohoto typu aplikací je jednoduchost v používání, kdy uživatel potřebuje pouze zařízení schopné spustit internetový prohlížeč a připojení k internetu. V opačném případě by musela aplikace být nainstalována přímo na daném zařízení, což s sebou nese otázky velikosti úložiště, výkonu zařízení a celkově omezení dané aplikace vůči vlastnostem operačního systému daného zařízení. [1] [3]

**Výhody webových aplikací** [2]:

- Minimální nároky na zařízení přistupující na webovou aplikaci

- Nezávislost vůči typu zařízení a operačnímu systému
- Škálovatelnost aplikace
- Větší možnosti automatizace
- Responzivita
- Cenově vůči vývoji aplikace na každý typ OS
- Distribuce

**Nevýhody webových aplikací** [2]:

- Závislost na připojení k internetu
- Konektivita s HW a notifikace
- Bezpečnostní rizika
- Podpora a další vývoj

## 3.3 Backend

Backend webové aplikace slouží ke zpracování a implementaci dat. Lze tuto část nazývat jako část pro uživatele neviditelnou. Zastupuje databáze, aplikační logiku a vývoj. Jeho základ je tvořen HWR, případně cloudovými platformami a virtuálními servery, na který se postupně realizuje základní struktura systému.

### 3.3.1 Server

Pro realizaci systému existuje možnost využít vlastní fyzický server, který je však nákladnější, je nutné jej spravovat, udržovat a časem obměňovat jeho součásti a je potřeba jej fyzicky na určitém místě a za určitých podmínek, jako například zajištění dočasného napájení v případě výpadku apod. provozovat. Tato možnost je tedy z počátku nákladnější, zároveň umožňuje aplikaci více funkcí a zapojení více periferních zařízení zapojených například přímo do serveru a tou nejpodstatnější výhodou je, že jsou data uložena na vlastním prostoru, nikoliv u poskytovatele. Alternativním řešením je například použití klasického počítače nebo zařízení Raspberry Pi, které jsou však vhodné pro menší podniky a to v závislosti na výkonu. [4] [5]

Dále je možné zprovoznit několik dalších právě virtuálních serverů a rozdělit si tak jej podle potřeb na různá odvětví. Na druhé straně je možnost využít VPS. V této variantě je nutné si vybrat poskytovatele VPS, který bude splňovat naše podmínky a u něj si následně zprovoznit systém se všemi potřebnými funkcemi. [5] [6]

Každý poskytovatel nabízí různé funkce a možnosti využití, rozšíření, garance stability či vzdálenou podporu přímo od poskytovatele za různé finanční částky. Mezi známé české poskytovatele patří například Wedos, Hostinger, Váš hosting, ZonerCloud, Český hosting VPS apod. Velmi důležitým faktorem je zde stabilita poskytovatele a jeho dostupnost.
Tato varianta je celkově jednodušší, a i zpravidla bývá levnější, to se však odvíjí od velikosti požadavků a projektu, který by měl být na dané struktuře realizován a případně do budoucna rozšiřován. V současné době je k tomuto typu velmi přihlíženo a plno firem jej používá. [5] [6]

**Výhody fyzického serveru** [6]:

- Všechna data jsou na vlastním serveru.
- V dlouhodobém měřítku a v závislosti na velikosti projektu se může stát finančně výhodnějším.
- Nezávislost na externí firmě (dodavateli).
- Nezávislost na internetovém připojení.

**Nevýhody fyzického serveru** [6]:

- Možné poruchy hardware.
- Větší servery mohou být prostorově náročnější.
- Náročnost na údržbu a chlazení.
- V případě velkého růstu dat a zátěže nutná obměna techniky za výkonnější.
- Žádný technický support.
- Energeticky nevýhodné.

**Výhody virtuálního serveru** [6]:

- Snadné zálohování.
- Neomezený vzdálený přístup (při přístupu k internetu).
- Energeticky nenáročné.
- Škálovatelnost (možnost rozšiřovat a upravovat výkon, velikosti disků apod.).
- Snadné a rychlejší nasazení a zprovoznění.
- V případě správně sepsané smlouvy support od poskytovatele.

**Nevýhody virtuálního serveru** [6]:

- Při výpadku sítě prakticky nemožné připojení.
- Možné výkyvy stability sítě dodavatele a tím snížení dostupnosti.
- Uživatelská a firemní data na cizím úložišti využívaná po síti.
- Cenově možný velký nárůst při vysokém zatížení a rozrůstání serveru.
- Bezpečnostní rizika.

### 3.3.2 Cloud computing

V širším slova smyslu se jedná o poskytování výpočetních prostředků a služeb prostřednictvím internetu. Jedná se o servery, síťové vybavení aplikační vybavení, úložiště a analytické nástroje. V současnosti patří cloud computing k velmi rozšířenému typu technologií, který využívá celá škála firem a podniků, a to bez ohledu na jejich velikost. S menší nadsázkou se dá říci, že cloud computing nabízí neomezené možnosti pronájmu, realizace apod. v rámci veškerých informačních možností současnosti. [33][34]

V současné době je úzce spojován s umělou inteligenci, kterou většiny cloudových představitelů již umožňují využít jako službu. Nejznámější představitelé cloud computingu: Google Cloud, Amazon Web Services (AWS), Microsoft Azure, apod. Nejpodstatnější součástí v cloud computingu je tzv. distribuční model, který definuje, co je v rámci dané služby nabízeno. Jeho základní rozdělení je: [35]

1) Infrastructure as a Services (IaaS)

   Tento model je zaměřen na výpočetní, síťové a úložné prostředky. Tento případ je klasický pronájem infrastruktury jako služby. Tato služba je nejzákladnější a nejpoužívanější ze všech. Hlavní výhodou je zde rychlá dostupnost, nízké počáteční náklady a plynulé dynamické škálování. Zde je hojně využívána metoda „pay as you go“, která funguje na principu pravidelné platby za garantované zdroje. [35]

2) Platform as a Services (PaaS)

   V tomto typu si klient pronajímá prostory pro vývoj a distribuci svého hardware. Zde poskytovatel zajišťuje navíc i hardwarovou základnu s vhodnými nástroji určenými na programování vývoj zákaznických produktů. V této struktuře je vhodné zmínit IBM cloud, který nabízí obrovské spektrum nástaveb od umělé inteligence až po automatizovaná workflow. [35]

3) Software as a Services (SaaS)

Tento typ nabízí již hotová řešení, přesněji software. V tomto případě klient má v cloudu již od poskytovatele připravený funkční software, nebo si jej může aplikovat na vlastní infrastrukturu se zajištěnou podporou a správou od poskytovatele. Cílem tohoto distribučního modelu je, aby zákazník využíval SW bez jakýchkoliv starostí a poskytovatel zabezpečil funkci, chod a vývoj. [35]

Realizace cloudového řešení má více typů možností: [34]

**Public cloud** – zde se jedná o klasický cloud computing, tedy nabízení služeb prostřednictvím internetu.
**Private cloud** – tento typ je zaměřen pro vlastní využívání a přizpůsobení daného podniku s vysokou bezpečností. Správa je prováděna danou organizací.
**Hybrid cloud** – toto řešení je kombinací předešlých dvou variant. Část služeb je poskytována z veřejného zdroje a část z vlastního.
**Community cloud** – tento typ je sdílen mezi více subjekty například se společným záměrem v podnikání. Jednotlivé subjekty si stanoví podmínky pro používání.

### 3.3.3 Databáze

Databáze je nástroj pro shromáždění informací, které jsou organizovány za určitým účelem a jsou navzájem propojené pomocí klíčů. Existuje mnoho různých typů relačních databází jako je MySQL, Oracle, PostgreSQL, Microsoft SQL Server a mnoho dalších. [7]

SŘBD umožňuje přístup k jednotlivý datům a manipulaci s nimi a je označován jako samostatný software. Jejich hlavním účelem je přehledně uspořádat velké spektrum dat do velkého celku, ze kterého jsou data jednoznačně přehledná a přístupná a aby se snížila jejich redundance. [7][8]

**Jednotlivé databázové modely** [7][8]:

- Relační databáze – založená na relačním modelu, tedy na tabulkách. Jednotlivé řádky jsou zde chápány jako jednotlivé záznamy a sloupce nesou informace o jednotlivých relacích mezi záznamy.
- Objektová databáze – zde je informace reprezentována jako objekt. Používá se především u objektově orientovaného programování.

- Objektově relační databáze – kombinace relačních a objektových databází.
- Síťová databáze – V tomto typu databáze mohla mít jedna entita více otců a umožňovala rekurzi. Dnes se již nevyužívá.
- Hierarchická databáze – data uspořádána ve stromové struktuře. První datový model. Vznik již v 70. létech.

**MySQL**

MySQL se řadí mezi tzv. open source databáze, což znamená, že je možné tuto variantu bezplatně stáhnou kdekoliv na internetu a využívat jej plnohodnotně bez jakýchkoliv poplatků a zároveň je možné upravit jeho zdrojový kód tak, aby vyhovoval stanoveným požadavkům. [7]

Řadí se mezi nejpoužívanější a zároveň jako jeden z nejoblíbenějších typů databází na celém světě a to díky svému dlouhodobému vývoji. Vlastníkem je Sun Microsystem, což je společnost firmy Oracle Corporation. [8]

Jedná se o typ relační databáze, tedy na tabulkách složených z řádků reprezentujících jednotlivé záznamy a sloupce, které zastupují atributy. Zde platí, že každý sloupec může obsahovat hodnoty pouze jednoho datového typu a má svůj pevně stanovený datový typ. Oproti ostatním typům databází je snadno implementovatelná, výkonná, volně šiřitelná a velmi často se spojuje s OS Linux, Apachem a PHP. [9]

**Použití MySQL** [9][10]:

- Cloud applications – v této oblasti je velmi hojně využíván. Je mnohonásobně levnější a rychlejší než jeho konkurenti. Příkladem je Amazon Redshift, který je o poloviční cenu dražší a podle výpočtů až 6,5x pomalejší než MySQL HeatWave.
- Social platforms – V obrovském měřítku využívá MySQL databáze například Facebook.
- Content management, Ecommerce a další.

**Výhody použití MySQL** [10]:

- Open source – volně dostupný z internetu.
- Vysoká bezpečnost – zabezpečení dat a podpora transakčního zpracování.
- Nepřetržitý provoz – je navržen tak, aby byl schopen pracovat prakticky bez zastavení.

- Nízké náklady – celkově se jedná o jednu z nejlevnějších variant mezi databázovými systémy, a i jednotlivé pluginy a rozšíření jsou oproti konkurenčním software levnější.
- Historický pevný základ – díky několikaletému vývoji je MySQL vysoce optimalizovaný.

**PostgreSQL**

Začátek tohoto typu SQL se datuje k roku 1986 pod vedením Michaela Stonebrakera a je to relační databázový systém s otevřeným zdrojovým kódem. Přes 25 let vývoje zaručuje vysoce stabilní, v současnosti velmi oblíbené databáze s podporou jak SQL, tak JSON. [11]

PostgreSQL disponuje mnoho funkcemi, jako jsou user-defined types, tablespace, table inheritance, point-in-time recorvery a mnoho dalších. Je schopen fungovat na všech rozšířených operačních systémech typu Windows Linuxu a UNIXů a zároveň podporuje mnoho kódovacích jazyků jako C/C++, Ruby, Python a mnoho dalších. [12]

Ve světě databází je předním představitelem, co se týče autentizace dat a rychlosti čtení a zápisu a je velmi efektivní při hloubkových analýzách různých typů dat. Z veřejných depozitářů linuxových distribucí archivu EnterpriseDB je možné jednoduše stáhnout a nainstalovat pod licencí BSD, která umožňuje neomezené používání. Nejaktuálnější verze je ze 14. září 2023 a to 16.0. [13]

**Použití PostgreSQL** [13]:

- Primárně se používá jako back-end databáze, na které stojí dynamické weby či samotné webové aplikace.
- Podniky využívají PostgreSQL velmi často jako primární úložiště pro aplikace.
- Společně s PostGIS se používá jako úložiště geoprostorových dat pro GIS.

**Výhody použití PostgreSQL** [13]:

- Open source – volně dostupný z internetu, a především je plně zdarma s plnou podporou všech funkcí.
- Vysoká bezpečnost a stabilita díky dlouhodobému vývoji.
- Podpora ukládání binárních objektů - obrázky, zvuky, videa.
- Vícebytové kódování znaků.
- Podpora relačních i nerelačních datových typů.

**MariaDB**

Je jedním z představitelů open source relačních databází. Je škálovatelná a podporuje více úložišť. Stejně jako konkurence MySQL je pod licencí GPLv2. Verze zdarma navíc obsahuje kompletní balíček funkcí, který mnohem obsáhlejší než free verze od MySQL. Je velmi závislá na podpoře komunity a jejich řešení chyb a aktualizací. [20]

**Použití MariaDB** [20]:

- MariDB je nadstavbou MySQL a její využití je podobné, avšak oproti MySQL jsou všechny části software open source a navíc mí oproti MySQL mnoho dalších rozšíření jako je engin Aria apod.

**Výhody MariaDB** [21]:

- Kompatibilita s MySQL.
- Silná ve zpracování velkých datových skupin a přehledná.
- Kontinuální vývoj.
- Podpora enginů – Aria, TokuDB, Spider a mnoho dalšách.

**MySQL Workbench**

Jedná se o vizuální nástroj, který umožňuje datové modelování, nástroje pro správu serverů, vývoj SQL, překlápění namodelovaných databází do kódu a plno dalších. Aplikace je optimalizována pro systémy Windows, Mac OS X a Linux a je vyvíjen a podporován společností Oracle. [28]

Mezi prvních 5 hlavních výhod MySQL Workbench patří [28]:

- SQL Development – umožňuje vytvářet a spravovat konektivitu k databázovým serverům a spouštět SQL dotazy.
- Data Modeling - umožňuje tvorbu modelů databáze v grafickém prostředí včetně úprav všech aspektů databáze pomocí editoru,
- Server Administration – umožňuje zálohování databází, správu uživatelů, sledování výkonu serveru a podobně.
- Data Migration – Tato funkce je schopna zmigrovat data z většiny existujících typů databází do databáze MySQL včetně jejích předešlých verzí.

- MySQL Enterprise Supports – zabezpečuje podporu pro podnikové části například firewallu, záloh a MySQL audit.

### 3.3.4 Docker

Jedná se o tzv. kontejnerovou aplikaci, která kompletuje jednotlivé aplikace do uvedených kontejnerů. Nad těmito kontejnery nainstalovaný Docker následně umožňuje globální správu a ovládání. Tyto kontejnery obsahují veškeré potřebné funkce, knihovny, nástroje a díky nim lze rychle nasadit a spravovat mnoho různých aplikací.

Velkou výhodou Dockeru je, že ve většině případů je zdarma, a to hlavně při použití na vlastní fyzických server a konfiguracích vlastních funkcí. Docker podporuje i velká řada cloud computingových společností, jako je například od Amazonu AWS. [22]

### 3.3.5 Portainer

Je open source grafické rozhraní založené na webu, které standardně slouží pro veškerou správu kontejnerů. Podporuje Docker engine a Docker Swarm a další jako Kubernetes či Azure ACI.

Má za cíl zjednodušit správu a ovládání, a to na fyzických i vzdálených serverech. Hlavní funkce Portaineru jsou správa řízení přístupu, izolace sítě, protokolování aplikací, virtualizace clusteru, správa registru a úložiště a mnoho dalších. [23]

### 3.3.6 Raspbian

Je přímo optimalizovaný operační systém postavený na systému Debian a který je modifikovaný přímo na hardware Raspberry Pi. Patří mezi open source a v základní “Lite“ verzi bez grafického rozhraní velmi prostorově nenáročný a jednoduše ovladatelný. Jeho utility, základní programy a více než 35000 softwarových balíčků dokáží plnohodnotně ovládat kterékoliv určené zařízení od Raspberry. [24]

Je k mání jak v 32 bitové verzi, tak v 64 bitové a nejaktuálnější verze je z 10.října 2023. Pro jeho implementaci se doporučuje použít aplikaci Raspberry Pi Imager, která zjednodušuje jeho nasazení a provede instalací krok po kroku. [25]

### 3.3.7 Webový server

Úkolem webových serverů je přijmout požadavek od klientů, které představují webové prohlížeče jako Mozilla Firefox, Google Chrome apod. a odpovědět těmto klientům zasláním požadované stránky.

**1) Apache HTTP server**

Je open source software s otevřeným zdrojovým kódem, který je na trhu od roku 1995 vydán společností Apache Software Foundation. Je kompatibilní s většinou OS. Jeho služby využívají společnosti jako Cisco, IBM, HP, Siemens, eBay apod. a dodnes obsluhuje okolo 30% webových stránek. [26]

Jeho hlavním úkolem je obsluha různých webových stránek jako prostředník mezi serverem a klientským zařízením. Při jakémkoliv požadavku ze strany uživatele stahuje obsah ze serveru a následně jej doručuje na web uživatele. Díky rozšířeným modulům s mnoha funkcemi je stále velmi populární a velmi přizpůsobivý dle požadavků uživatele. Nejaktuálnější verze je 2.4.58 z 19.října 2023. [26]

**2) NGINX**

Patří mezi nejspolehlivější webové servery současnosti a je taktéž open source. První vydání proběhlo v roce 2004 od vývojářů Nginx, Inc. Jeho obrovská popularita vzrostla díky vysokému výkonu a odolnosti na zátěž. Velmi často se používá i jako loadbalancer a reverzní proxy server pro plynulost a stabilní výkon serveru. [26]

Stejně jako Apache podporuje většinu OS, ale doporučuje se především používat na OS podobné Unixu. Nejaktuálnější verze je 1.25.3 z 24.října 2023. [27]

## 3.4 Frontend

Zabezpečuje klientskou stranu systému, tedy to, co vidí uživatel v prohlížeči při příchodu na webovou stránku.

### 3.4.1 HTML

Je typ značkovacího jazyka, který vznikl na přelomu 80. a 90 let a slouží pro tvorbu obsahu webových stránek v podobě textů, multimédií, tabulek, obrázků a dalších. [14]

Základem HTML jsou značky, tedy “tagy“, které nám umožňují modifikovat text a které rozdělujeme do skupin strukturálních, popisných a stylistických. K těmto tagům se přiřazují atributy a hodnoty, které mění role prvků na stránce. Další důležitou součástí je tzv. element, který je definován právě tagy.

Existuje mnoho programů, které umožňují tvorbu webu, mezi známé editory patří například PsPad, Komodo, WordPress a mnoho dalších. [15]

V současné době se používá HTML5 z roku 2014. Oproti předešlé verzi HTML4 funguje v off-line módu, kdy je možné načíst webovou aplikaci i bez přístupu k internetu a data ukládat dočasně do lokální paměti. Po připojení k internetu dojde k synchronizaci a načtení dat do serveru. Dále stojí za zmínku i změna typu dokumentu (DOCTYPE), který byl hodně zkrácen a kde není nutné zadávat specifikace dokumentu. [14] [15]

### 3.4.2 PHP

Je jeden z nejrozšířenějších a nejznámějších open source skriptovacích jazyku na straně serveru. Aktuální verze je 8.1.24, se kterou se vývojáři zaměřují především na vývoj skriptování na straně serveru. Je nezávislý vůči platformě, řadí se mezi jednodušší typy jazyků, je schopný spolupracovat jak s relačními, tak i nerelačními databázemi a díky velmi rozsáhlé komunitě a vývojářů existují oficiální dokumentace s návody a postupy. [16]

Tento jazyk používá obrovská škála společností a firem, jako je Facebook, Wordpress, Drupal, Joomla, a i webhostingové platformy jako BlueHost, Site ground a další. Vzhledem k jeho dlouhodobé působnosti si drží okolo 80 % správy serverů webových stránek, a to i za velmi silné konkurence jiných skriptovacích jazyků jako je Python, JavaScript a podobně. [17]

### 3.4.3 HTTPS

Jedná se o zabezpečenou verzi protokolu HTTP, který slouží pro zabezpečenou komunikaci mezi webovým prohlížečem a vzdáleným www serverem. Toto šifrování je v současné době nutné, vzhledem k obrovskému přenosu dat, a především těch citlivých jako jsou přihlášení do bankovních účtů a jiné. Komunikaci přes původní http bylo možné přečíst a pomocí volných softwarových aplikací lehce napadnutelná. Původní http nebyl nijak šifrován a dnes jej již žádný prohlížeč neumožní či nedoporučí při pokusu otevřít. [18]

Šifrování je zabezpečeno pomocí SSL a TLS certifikátů, které mají za úkol šifrovat komunikaci na internetu. Díky jejich šifrování není možné přečíst komunikaci mezi klientem a vzdáleným serverem. Aktuální verze SSL je 3.3 a TLS je 1.2. Fungují na principu ověření certifikátu druhé strany, kde má každá strana 2 šifrovací klíče v podobě veřejného a soukromého. Při zašifrování veřejným klíčem může data rozšifrovat pouze příjemce s identickým veřejným klíčem a k rozšifrování použije svůj soukromý klíč. [19]

## 3.5 Ostatní součásti

Níže budou popsány další důležité součásti systému, vývojové aplikace apod.

### 3.5.1 Node.js

Je asynchronní běhové prostředí JavaScriptu s otevřeným zdrojovým kódem, které umí spouštět kód mimo webový prohlížeč. Díky struktuře JavaScript engin Chrome V8 je prostředí identické s Google Chrome webovým prohlížečem. Jeho podstatným účelem je tvorba serverové části webových aplikací. [31]

Mezi podstatné vlastnosti patří generování dynamického obsahu stránek, tvorba a veškerá administrace souborů na serveru, shromažďování dat formulářů, editovat databázi a plno dalších. Je konstruován tak, aby byl vysoce škálovatelný, tzn. byl schopen zpracovat více připojených klientů současně. pro jeho běh nutné mít zprovozněné IDE prostředí, které je možné realizovat například WebStormem, Visual Studio Code a podobně. [32]

Je kompatibilní s MacOS, Linuxem, Unixem, Windowsem apod. ve správci balíčků se nachází více než 50 000 balíčků s různými funkcemi. Má optimalizovanou synchronizaci kódu mezi klientem a serverem a díky tomu je rychlejší při načítání zvuků či videí. [32]

### 3.5.2 Visual Studio Code

Je zjednodušený editor zdrojového kódu s funkcemi vývojářských nástrojů, který je dostupný na MacOS, Linux, Windows, a to ve free verzi pro soukromé účely či komerční. Jelikož je to open source projekt, velmi často se dobře spolupracuje s Githubem a pomáhá se vyvíjet další možnosti. Vývojáři každý měsíc provádí aktualizace. [29]

Má vestavěnou podporu pro vývoj v Node.js s JavaScript a TypeScript. Popdoruje i vývoj webu za pomoci JSX/React, CSS, JSON, Less, SCSS a HTML. Podporuje i plno programovacích jazyků jako jsou C++, Python, Java, PHP apod. a pracovní prostředí jako je Docker a Kubernetes, runtimesy .NET a Unity a podporuje i většinu známých cloudových platforem například Amazon Web Services, Google Cloud, Microsoft Azure a další. [30]

Za zmínku stojí také jeho webová verze, která je sice obraná o pár funkcí s kódovými nebo jazykovými servery, ale je schopná zabezpečit většinu podstatného. Je možné jej využít například přímo na stránkách vscode.dev nebo github.dev. Oproti své plnohodnotné tzv. těžké verzi Visual Studio neumí generovat projekty ze šablon. Jsou však mezi sebou plně kompatibilní. [30]

### 3.5.3 Bezpečnost

Bezpečnost webových aplikací je v současné době naprosto nejzásadnější otázka v případě realizace a provozu webové aplikace. Díky vystavení těchto služeb na veřejném internetu a obsahu mnoha důležitých i citlivých se tato struktura stává častým cílem kyberzločinců. Prolomením skrze ochranu může útočník získat nejen firemní data a uživatelská, ale i přístupy do vnitřní infrastruktury. [36]

Kladen je velký důraz na průběžné testování bezpečnosti, a to především na aplikační vrstvě, přes kterou běží http protokol. Jako každý software i webové aplikace obsahují chyby. V případě webových aplikací se často jedná o špatné konfigurace šifrování a kryptografie aplikace a serveru. [36]

V současné době existuje mnoho možností realizace zabezpečení pomocí externích firem, rozšířeních, aplikací a mnoho dalších. Za zmínku určitě stojí například Web AppSec with Check Point, který obsahuje obrovskou škálu bezpečnostních nástrojů pro vývojáře a bezpečnostní specialisty. [37]

Příklady moderních hrozeb webových aplikací [37] [38]:

- Credential Stuffing – použití slabých či odhalených hesel pro přístup k uživatelskému účtu a jiných služeb.
- Supply Chain Attacks – knihovny a pluginy třetích stran mohou být díky možným chybám zneužity.
- DoS – prolomení infrastruktury obrovskou zátěží, jedná se o velmi starý typ bezpečnostního rizika, který je však stále obrovskou hrozbou
- Injection – záměrné posílání neplatných či špatných vstupů, které mají za následek nevypočitatelné reakce systému – SQL Injection

Možné obranné mechanismy webových aplikací [37] [38]:

- Web Application Firewall – má za úkol odfiltrovat provoz, u kterého existuje podezření, že může narušit bezpečnost aplikace. Jedná se o základní variantu ochrany, která by měla být aplikována u každé aplikace.
- DNESSEC – tento protokol má za úkol správně směrovat provoz webových aplikací na určené servery. Tím se narušuje možnost odposlechu komunikace.
- API gateways – Identifikují stínová nebo podvrhlá rozhraní API a blokují jim provoz.

- SSL/TLS šifrování – šifrovací klíče pro komunikaci, bez tohoto šifrování nelze provozovat bezpečně žádné web servery
- DDoS mitigation – použity mezi serverem a veřejným internetem se specifickým filtrem a velkou šířkou pásma pro blokaci a odlehčení zátěže serveru v případě útoku.

Rozpoznání následků v případě úspěšné bezpečnostního útoku je klíčové pro stanovení a vývoj bezpečnostních kritérií a k určení směru bezpečnosti celé infastruktury. Pro vhodné realizace bezpečnosti existují bezpečnostní testy, které jsou schopné odhalit bezpečnostní nedostatky bez jakýchkoliv následků. [38]

Možné bezpečnostní testy aplikací [38]:

Dynamic application security test – test vhodný pro vnitřně postavené aplikace
Static application security test – slouží pro identifikaci chyb
Penetration test – test vhodný pro kritické aplikace, které jsou často upravovány

Doporučené bezpečnostní postupy [36] [37]:

Validate User Input – Autentizační ochrana je podstatný prvek ochrany, cílem je tedy specifikovat vhodné požadavky pro vstup do aplikace jako jsou například silná hesla.
Automate DevSecOps – automatizace statického a dynamického zabezpečení aplikací.
Performance Regular Vulnerability Scans – pravidelné kontroly bezpečnostních chyb a systému.
Avoid Shadows APIs – provoz pouze autorizovaných a spravovaných rozhraní API.

### 3.5.4 Bootstrap

Jedná se o frontendový volně dostupný framework, který kopíruje nejnovější a nejmodernější webdesignu. Obsahuje šablony založené na HTML a CSS pro tlačítka, formuláře, tabulky, modály, karusely a mnoho dalších. Tím velmi zjednodušuje vývojářům tvorbu webu a zanáší novodobé trendy do webového designu. Mezi jeho podstatné výhody patří jednoduchost použití, responzivní funkce (přizpůsobitelnost na jednotlivé typy zařízení), kompatibilita s většinou prohlížečů (Chrome, Firefox, Safari…), Mobile first apod… [39]

Jako každý framework i Bootstrap má své alternativy, známou konkurencí je například Tailwind CSS, avšak oproti Bootstrapu neobsahuje žádné komponenty, požaduje

větší znalosti CSS a není tak přizpůsobitelný jako Bootstrap. Další známou alternativou je Bulma, která je však složitější na používání díky nutnému vytvoření skriptu pro úpravu nebo Jquery. [40]

### 3.5.5 JavaScript

Patří mezi skriptovací jazyky, které běží na straně klienta, tím je veškerá jeho činnost provozována na straně klienta a umí používat komplexní funkce na webových stránkách. Zobrazuje včasné aktualizace obsahu, videa, 2D a 3D animace a mnoho dalších. Je hojně používán i díky vlastnostem pro formování textu, možností designů a efektů, ale i pro možnost úpravy validace webových stránek tak, že dokáže klienta upozornit v případě nějaké vytvořené chyby. [41]

JavaScript je natolik univerzální, že se nepoužívá jen na straně klienta, ale i na straně serveru. Nejznámějším představitelem serverového JavaScriptu je bezpochyby Node.js. Díky svému obrovskému rozšíření, možnostem a jednoduchosti je JavaScript jedna z nejlepších možností pro tvorbu webových stránek, aplikací a mnoho dalších. V případě JavaScriptu je nutné zmínit obrovskou nevýhodu a tou je fakt, že běží u klienta, tzn. klient si může zdrojový kód zobrazit, zkopírovat, upravovat a podobně. [42]

# 4 Vlastní práce

V této části bude provedena analýza, návrh a realizace tohoto systému. Je důležité říci, že v návrhu systému bude brán ohled na možnost využití tohoto systému u různých dalších firem, a proto se může stát, že některé části v návrhu systému nebudou v realizaci provedeny nebo naopak některé funkce, které se nenachází v návrhu systému v realizaci budou implementovány. Realizace systému bude zaměřena na potencionálního zájemce o tento systém a bude tedy přizpůsobena jeho požadavkům.

## 4.1 Analýza

### 4.1.1 Seznámení s klientem

Klient je majitelem a provozovatelem soukromého malého podniku zaměřeného na servis a speciální úpravy různých typů motocyklů v malé vesnici nedaleko Prahy. Pro evidenci a zaměstnání nepoužívá žádný informační systém a ani aplikaci, pouze webovou stránku se základními informacemi o jeho činnosti. V tomto podniku spolupracuje s kolegou, který je pracovně na stejné úrovni. Díky své kvalitní a precizní práci je velmi populární nejen ve svém okolí.

### 4.1.2 Specifický problém klienta

V průběhu oprav a úprav motocyklů provádí mechanik průběžnou dokumentaci pořizováním fotografií na mobilní telefon. Dále poměrně často komunikuje se zákazníky ohledně daného problému (například, zda má danou součástku opravit či vyměnit). Tato komunikace je však různorodá v podobě telefonických hovorů, SMS, emailů a mnoho dalších.

Podstatnou problematikou je chvíle, kdy si klient vyzvedává již opravený/upravený motocykl, kdy mu mechanik ukazuje všechny pořízené fotografie z průběhu oprav a přidává k nim jednotlivé komentáře ohledně dané problematiky a jejich řešení, případně komentáře k dalším nalezeným problémům. Dále mechanik nabízí nahrání fotografií na přenosné médium, jako je například flash paměť pro vlastní využití.

V neposlední řadě si vede mechanik evidenci motocyklů v podobě fotek v počítači a jednotlivé servisní zásahy v obyčejném papírovém sešitu. Dalšími kroky jsou plánování budoucích oprav a zásahů, které se zapisují manuálně do obyčejného papírového kalendáře.

Celkově lze tedy říci, že jsou všechny tyto důležité informace poznamenávány na různá místa a komunikace se zákazníky je velmi neefektivní.

Cílem projektu je tedy navrhnout optimální řešení, které zastoupí všechny aktivity do jednoho systému a zpřehlední a synchronizuje je. Zároveň je nutné, aby systém umožňoval dohledávání historických zásahů a plánů. Podstatnou vlastností systému je jednoduchost a aby zjednodušila celkovou komunikaci se zákazníky. Hlavní pracovní zařízení bude mobilní telefon, případně tablet, přes který bude klient provádět většinu činnosti.

**Na základě problematiky a představy klienta byly stanoveny tyto následující požadavky na systém:**

- **Možnost nahrávání fotografií –** do systému by bylo vhodné k dané servisní činnosti mít možnost nahrát například 3 fotografie a k nim vložit komentář, co jednotlivé fotografie znamenají, aby klient pochopil, co je na fotografiích zaznamenáno.
- **Prostor pro vkládání servisních úkonů –** v systému by měla být možnost přiřadit nový servisní úkon k vybranému motocyklu a u něj možnost naplánovat datum provedení a zadat další důležité informace. Dále by mělo být možné filtrovat mezi jednotlivými činnostmi.
- **Poznámkový prostor –** k jednotlivým servisním plánům by mělo být možné napsat poznámky o domluvené činnosti a další detaily.
- **Komunikační prostor –** systém by měl umožňovat využití chatu s klienty, aby se zamezila komunikace na různých platformách. V ideálním případě využít jednoduchou mobilní aplikaci, aby se mechanik nemusel kvůli každé zprávě přihlašovat do systému, ale pouze si pustil chatovací aplikaci v telefonu.
- **Databáze vozidel a klientů** – mělo by být možné v systému vidět všechny zaregistrované uživatele pro jednodušší komunikaci a motocykly s potřebnými podrobnostmi včetně spojení s uživateli.
- **Flexibilní systém** – systém by měl být vizuálně přizpůsoben typu zařízení vzhledem k využívání systému i na mobilních telefonech či tabletech.
- **Jednoduché používání** – systém by neměl být složitý na využití (vstup na potřebné stránky a funkce v rámci hlavní stránky klienta).

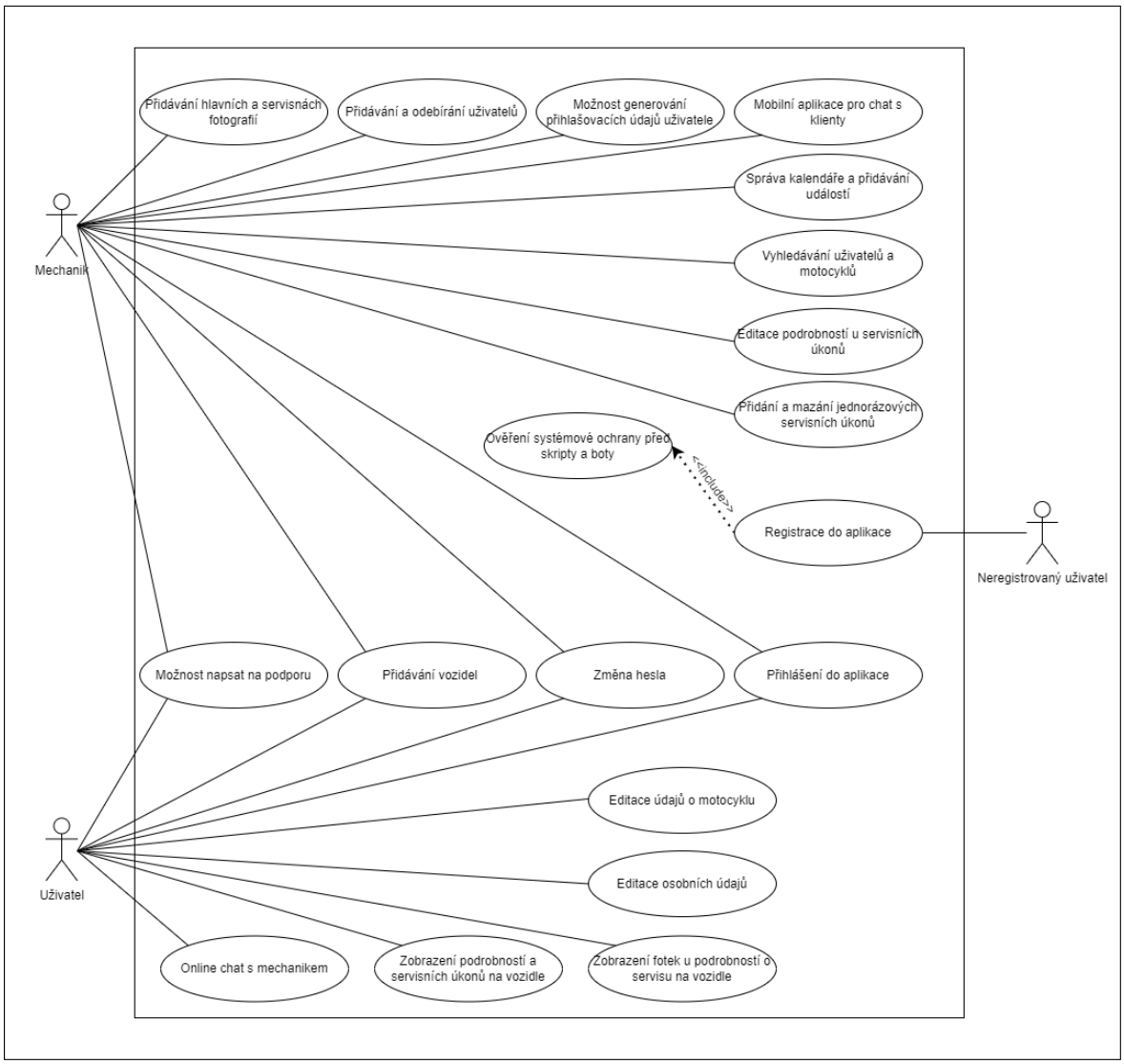
Podstatnou částí tohoto projektu je navrhnout a zrealizovat funkční prototyp, který bude následně představen potencionálnímu zákazníkovi a podle potřeb případně upraven, aby jej bylo možné zprovoznit u zákazníka. Tento prototyp bude tvořen na fyzickém zařízení a bude obsahovat 3 základní části v podobě hardware, backend a frontend.

Hardwarovou část zastoupí zařízení Raspberry Pi 4 Model B s 8 GB RAM s přidaným chlazením, zdrojem, SSD pamětí a to z důvodu jednoduššího zprovoznění a používání pro prvotní vývoj díky předešlé praxi s těmito prvky. V podobě operačního systému pro tuto vývojovou část bude vybrán systém Raspbian, který je přímo optimalizovaný pro požívání zařízení Raspberry Pi a vychází z principu systému Debian. Dále bude použit jako zástupce databázové části MariaDB. Pro správu kontejnerů a backendu bude použita volně dostupná aplikace Docker.

Frontend systému zabezpečí značkovací jazyk HTML, kaskádové styly CSS, pro skripty bude použit skriptovací jazyk PHP, případně na některé funkcionality může být použit jazyk Javascript či Python a zdrojový kód bude tvořen v aplikaci Microsoft Visual Studio Code.

### 4.1.3 Use Case Diagram

Diagram případů užití má za úkol popsat používání systému jednotlivých aktérů (účastníkům). V tomto návrhu jsou zobrazeny pouze základní možnosti použití, oproti výslednému systému se může mírně lišit. Díky tomuto návrhu je možné lépe zpracovat jednotlivé požadavky do systému a zpracovat návrh na implementaci.




*Obrázek 1 - Use Case diagram (vlastní zpracování)*

### 4.1.4 Scénáře use case

Scénáře níže slouží pro popis jednotlivých kroků aktérů mezi sebou a mají za úkol popsat, co jaký aktér musí provést, aby došlo na daný use case. Zobrazeny budou pouze některé scénáře.

**Use Case 1 – Autentizace do systému**

**Aktéři**

- Uživatel, mechanik a systém.

**Činnost**

- Přihlášení mechanik a uživatele do systému.

**Podmínky**

- Mechanik musí být registrován administrátorem v databázi.
- Uživatel byl zadán mechanikem či webovým formulářem do systému.

**Základní tok**

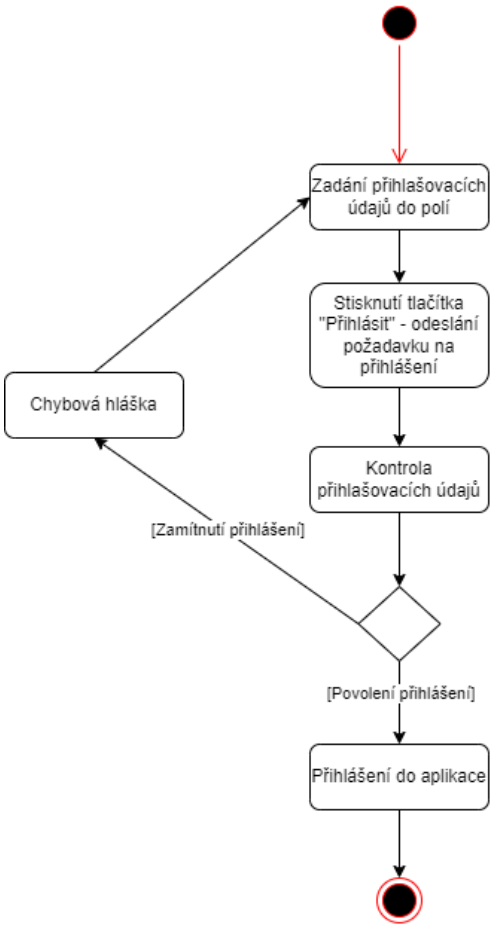
- Uživatel / mechanik se pokusí přihlásit do systému pomocí loginu a hesla.
- Systém ověří zadané přihlašovací údaje v databázi.
- Systém přihlásí uživatele / mechanika podle přihlašovacích údajů.

**Alternativní tok**

- Zadání chybných přihlašovacích údajů.
- Chybová hláška „Nesprávné přihlašovací údaje“ – možnost znovu se přihlásit.
- Opětovné zadání přihlašovacích údajů.
- Ověření přihlašovacích údajů v databázi.
- Systém přihlásí uživatele / mechanika podle přihlašovacích údajů.

**Ukončení Use Case 1**

- Úspěšná autentizace do systému.




*Obrázek 2 - Diagram aktivit - Přihlášeni (vlastní zpracování)*

**Use Case 2 – Registrace uživatele do systému**

**Aktéři**

- Neregistrovaný uživatel a systém.

**Činnost**

- Přidání uživatele do systému.

**Podmínky**

- Neregistrovaný uživatel - vyplnění formuláře a odeslání registrace.
- Systém - ověření uživatele pomocí implementované ochrany.

**Základní tok**

- Neregistrovaný uživatel vyplní přihlašovací údaje a odešle údaje.
- Systém provede kontrolu uživatele.
- Systém provede kontrolu zadaných uživatelských údajů.

- Systém vygeneruje na základě jména a příjmení login a následně náhodné heslo.
- Zápis uživatele do databáze.
- Systém odešle přihlašovací údaje na zadaný email uživatele.

**Alternativní tok 1**

- Systémová ochrana zamítne registraci.
- Systémová ochrana umožní ověření uživatele.
- Uživatel provede úspěšné ověření.
- Systém vygeneruje na základě jména a příjmení login a následně náhodné heslo.
- Zápis uživatele do databáze.
- Systém odešle přihlašovací údaje na zadaný email uživatele.

**Alternativní tok 2**

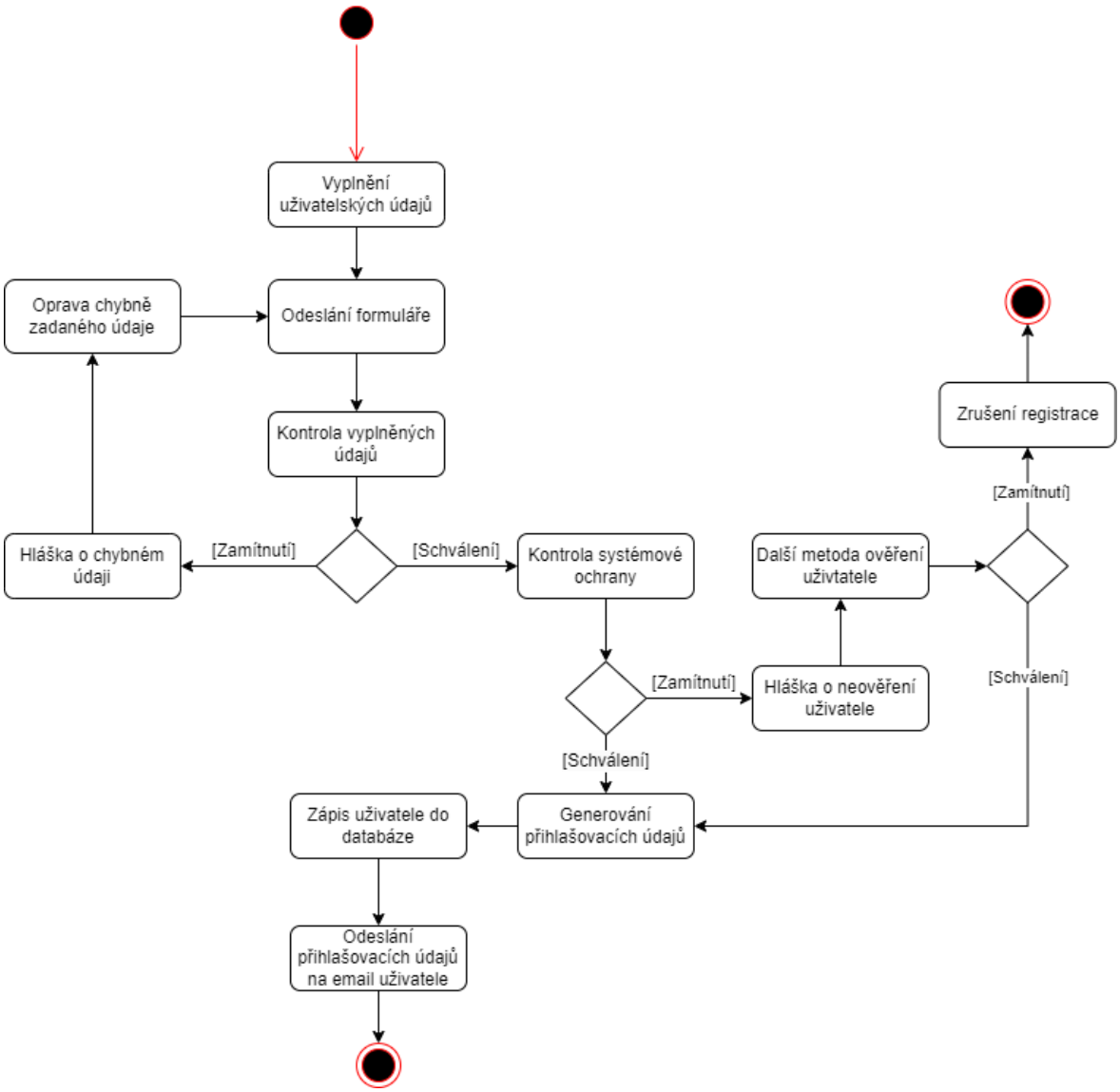
- Systémová ochrana zamítne registraci.
- Systémová ochrana umožní ověření uživatele.
- Uživatel provede neúspěšné ověření.
- Systém neumožní registraci uživatele.

**Alternativní tok 3**

- Zadání chybných parametrů u údajů uživatele.
- Vygenerování chyby v zadané části formuláře.
- Oprava chyby v zadaných údajích.
- Systém provede kontrolu zadaných údajů.
- Systém provede bezpečnostní kontrolu.
- Systém vygeneruje na základě jména a příjmení login a následně náhodné heslo.
- Zápis uživatele do databáze.
- Systém odešle přihlašovací údaje na zadaný email uživatele.

**Ukončení Use Case 2**

- Systém vygeneruje hlášku o úspěšném přidání uživatele do databáze.




*Obrázek 3 - Diagram aktivit – Uživatel - registrace (vlastní zpracování)*

### Use Case 3 – Přidání vozidla do systému mechanikem/uživatelem

#### Aktéři

- Mechanik, uživatel a systém.

#### Činnost

- Přidání vozidla uživatele do systému.

#### Podmínky

- Mechanik - Uživatel musí být zaregistrován v systému.
- Uživatel – Uživatel musí být přihlášen v systému.

**Základní tok (mechanik)**

- Mechanik se přesune na formulář pro přidávání vozidla.
- Vyplnění veškerých požadovaných údajů o vozidle.
- Výběr uživatele (majitele) vozidla načteného z databáze.
- Systém provede kontrolu zadaných údajů.
- Přiřazení ID vybraného zákazníka k vozidlu.
- Zápis vozidla do databáze.

**Základní tok (uživatel)**

- Uživatel se přesune na formulář pro přidávání vozidla.
- Vyplnění veškerých požadovaných údajů o vozidle.
- Systém provede kontrolu zadaných údajů.
- Přiřazení ID přihlášeného zákazníka k vozidlu.
- Zápis vozidla do databáze.

**Alternativní tok**

- Zadání chybných údajů o vozidle.
- Vygenerování chyby v zadané části formuláře.
- Opětovné zadání údajů o vozidle.
- Systém provede kontrolu zadaných údajů.
- Přiřazení ID přihlášeného zákazníka k vozidlu.
- Zápis vozidla do databáze.

**Ukončení Use Case 3**

- Systém vygeneruje hlášku o úspěšném přidání vozidla do databáze.

**Use Case 4 – Přidání události k vozidlu**

**Aktéři**

- Mechanik a systém.

**Činnost**

- Přidání události k vozidlu uživatele.

**Podmínky**

- Vozidlo musí být zaregistrováno v systému.

**Základní tok**

- Mechanik se přesune na formulář pro přidávání událostí k vozidlům.
- Vyplnění veškerých požadovaných údajů o události.
- Výběr vozidla načteného z databáze.
- Systém provede kontrolu zadaných údajů.
- Přiřazení ID vybraného vozidla k události.
- Zápis události do databáze.

**Alternativní tok**

- Zadání chybných údajů k události.
- Vygenerování chyby v zadané části formuláře.
- Opětovné zadání údajů k události.
- Systém provede kontrolu zadaných údajů.
- Přiřazení ID vybraného vozidla k události.
- Zápis události do databáze.

**Ukončení Use Case 4**

- Systém vygeneruje hlášku o úspěšném přidání události do databáze.

**Use Case 5 – Přidání podrobného záznamu o servisu vozidla**

**Aktéři**

- Mechanik a systém.

**Činnost**

- Přidání podrobného záznamu o servisu k vozidlu uživatele.

**Podmínky**

- Vozidlo musí být zaregistrováno v systému.
- K vozidlu musí být vytvořena odpovídající událost v systému.

**Základní tok**

- Mechanik se přesune na stránku s podrobnostmi o servisu vozidla.
- Přidání komentáře k události ohledně servisu vozidla.
- Výběr fotografií k servisu vozidla.
- Nahrání fotografií k události motocyklu.

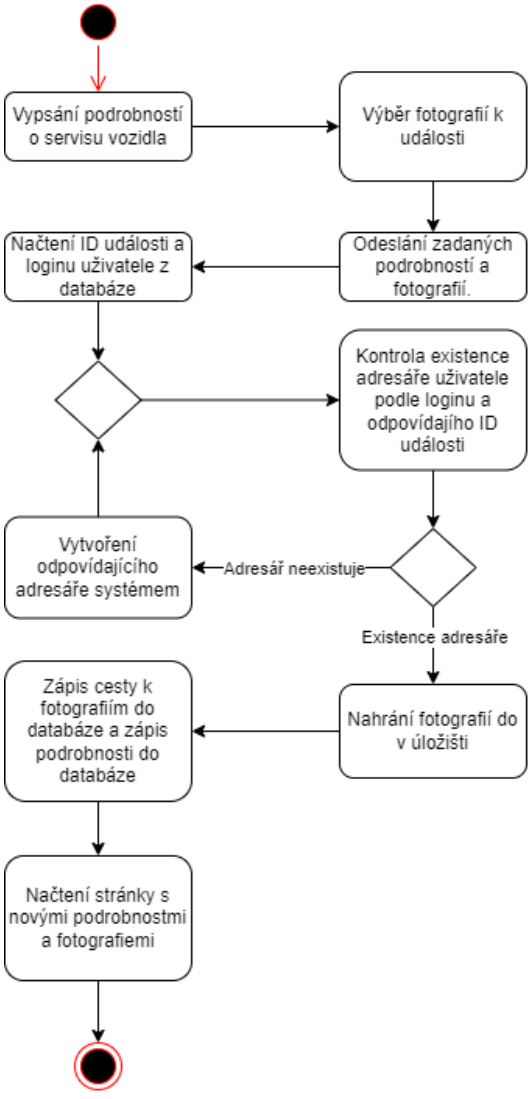
- Systém načte ID události a login uživatele.
- Kontrola existence adresáře uživatele na serveru pro fotografie.
- Nahrání fotografií do adresáře uživatele k události na serveru.
- Zápis podrobnosti a cesty k fotografiím do databáze.

**Alternativní tok**

- V případě neexistujícího adresáře uživatele či adresáře dané události pro fotografie systém automaticky vytvoří adresář pro nahrávání fotografií události.
- Nahrání fotografií do adresáře uživatele k události na serveru.
- Zápis podrobnosti a cesty k fotografiím do databáze.

**Ukončení Use Case 5**

- Systém ukončí zadávání a načte stránku s novými informacemi a fotografiemi.




*Obrázek 4 - Diagram aktivit - Přidání podrobností a fotografií k události (vlastní zpracování)*

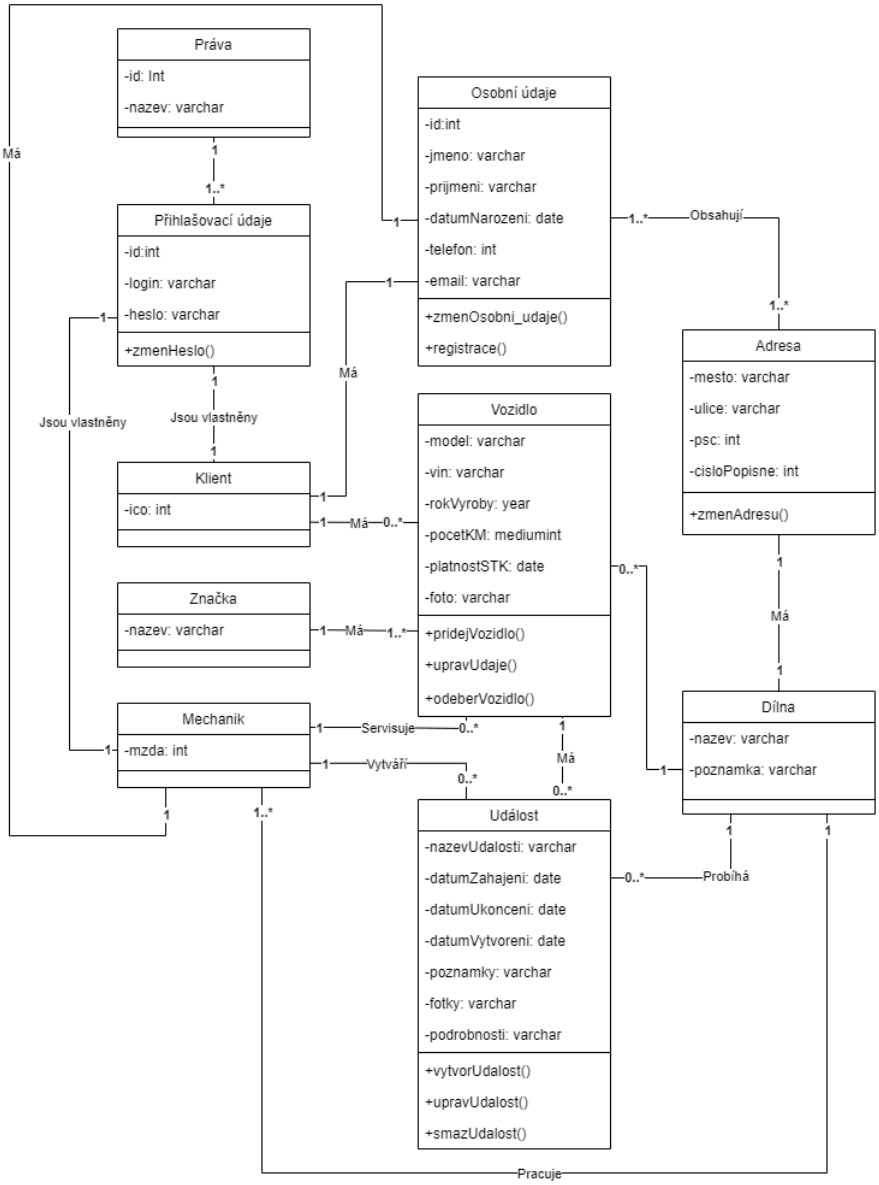
### 4.1.5 Class diagram

Diagram tříd je další z typů diagramů, který má přesněji navést programátorova při zpracovávání řešení projektu. V rámci tohoto systému bylo do diagramu vloženo 10 tříd. Nejpodstatnější třídou v tomto systému je třída „Událost“. V této třídě je ze strany mechanika spravována a přidávána jeho aktivita (servisní činnost) vůči vozidlům registrovaným v systému. V této třídě může mechanik jednoznačně ukázat klientovi daného vozu aktivitu provedenou na jeho vozidle, kterou si lze představit například jako „Generální servis“ pojmenovanu v atributu „nazevUdalosti“. U této události je následně možno nastavit datum, kdy bude zahájen tento servis pomocí atributu „datumZahajeni“. Dále se zde vloží do atributu plánované „datumUkonceni“ a „datumVytvoreni“ je vyplněno automaticky při vytvoření nové události. V této třídě se následně nachází atribut „poznamky“. Tento atribut slouží pro doplňující informace k názvu události, tedy například „Kompletní motor, výměna přední levé žárovky“. Následně se zde nachází atributy „fotky“ a „podrobnosti“. Tyto atributy slouží již pro podrobný popis servisního zákroku na vozidle a budou doplňovány později po vytvoření této události při průběhu servisní činnosti. Na tuto třídu je navázana třída „Mechanik“, která říká, že jeden mechanik nemusí vytvořit žádnou událost či jich vytvořit více. Dále je provázána na třídu „Vozidlo“ a říká, že jedno vozidlo nemusí mít žádnou událost či jich může mít více. Je důležité poznamenat, že vytvořená událost se vždy váže pouze na jedno jediné vozidlo a jediného mechanika. Následuje provázání na třídu „Dílna“ a říká, že žádná událost či více událostí může probíhat v jedné dílně. Tento princip je logický, jelikož nemůže probíhat jedna daná událost vozdila ve dvou různých dílnách.

Další důležitou třídou je zde třída „Vozidlo“. Tato třída má na sebe navázanou třídu „Značka“ a říká, že jedno vozidlo může mít vždy jen jendu značku, ale jednu značku může mít více vozidel. Nachází se zde vazba na třídu „Klient“, která říká, že klient nemusí mít žádné vozidlo registrované v systému, ale může jich mít více a zároveň dané vozidlo může mít vždy jen jednoho klienta (majitele). Spojení s třídou „Mechanik“ říká, že jeden mechanik nemusí vytvářet žádné události či jich vytvořit více, daná událost je však vždy vytvořena jedním mechanikem. Propojení vazby s třídou „Událost“ již bylo posáno a poslední spojení je zde vůči třídě „Dílna“ a říká, že se v dílně nemusí nacházet žádné vozidlo či se jich tam může nacházet více, je zde logická vazba, která říká, dané vozidlo se vždy může nacházet jen v jedné dílně.

Důležité třídy jsou zde i „Přihlašovací údaje“, „Osobní údaje“ a „Práva“. Jsou takto rozděleny z důvodu zamezení redundance dat v tabulkách „Klient“ a „Mechanik“, do

kterých by je bylo možné vložit. Třída „Práva“ říká, že tyto práva může mít přiřazeno více přihlašovacích údajů a zároveň jedny přihlašovací údaje můžou mít vždy jen jedny práva. Dále jsou zde vazby z třídy „Přihlašovací údaje“ na třídy „Mechanik“ a „Klient“ a říkají, že jedny přihlašovací údaje může mít vždy jen jeden klient nebo mechanik a obráceně. Obdobně jsou nastaveny vazby s třídou „Osobní údaje“, které mají navíc vazbu na „Adresa“. Tato vazba říká, jedny osobní údaje můžou mít jednu či více adres a obráceně (lze si zde představit, že mají registraci v sysétmu osoby z jedné bytovky). Poslední nepopsanou vazbou je vazba mezi třídou „Adresa“ a třídou „Dílna“ a říká, že danou adresu může mít jen jedna dílna  a vazba mezi Dílnou a Mechanikem, která říká, že v jedné dílně se může nacházet jeden či více mechaniků, daný mechanik se však může nacházet jen v jedné dílně.




*Obrázek 5 - Class diargam (vlastní zpracování)*

## 4.2 Návrh systému

Tato část bude na základě provedené analýzy zaměřena na návrh systému z pohledu vlastností a všech jednotlivých součástí systému pomocí datového slovníku, návrhu wireframe modelů a podle jejich konstrukce na vizuální ukázku stránek. Součástí návrhu bude diagram naznačující konstrukci databáze systému.

### 4.2.1 Datový slovník

Na základě stanovených tříd v class diagramu v předešlé části projektu dojde na vytvoření datového slovníku, který má určit jednotlivé datové typy atributů a jejich případnou délku. Zároveň zde budou nastaveny podmínky pro jednotlivé atributy.

**Osobní údaje –** tabulka osobní údaje obsahuje veškeré údaje v systému potřebné při registraci uživatele a pro následné systémové funkce. Tyto údaje jsou nastaveny i u mechanika při vytváření jejich účtu na serveru.

*Tabulka 1 - Datový slovník - "Osobní údaje"*

| Osobní údaje | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Jméno | VARCHAR (45) | NOT NULL |
| Příjmení | VARCHAR (45) | NOT NULL |
| Telefon | VARCHAR (13) | NOT NULL |
| Datum narození | DATE | NOT NULL |
| Email | VARCHAR (60) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Přihlašovací údaje –** Tato třída obsahuje přihlašovací údaje uživatelů a mechaniků v systému. Atribut „Login“ a „Heslo“ jsou automaticky generovány v systému po registraci uživatele.

*Tabulka 2 - Datový slovník - "Přihlašovací údaje"*

| Přihlašovací údaje | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Login | VARCHAR (45) | NOT NULL |
| Heslo | VARCHAR (255) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Klient –** v této tabulce se nachází pouze atribut „IČO“, který není nutné při registraci uživtale vyplnit.

*Tabulka 3 - Datový slovník - "Klient"*

| Klient | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| IČO | INT | NULL |

*Zdroj: Vlastní zpracování*

**Vozidlo –** Tato tabulka obsahuje všechny potřebné údaje o vozidle. Atribut „Fotka“ je nastaven jako varchar(300) z důvodu, jelikož se bude jednat o údaj nesoucí cestu na serveru k fotografii.

*Tabulka 4 - Datový slovník -“Vozidlo"*

| Vozidlo | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| VIN | VARCHAR (17) | NOT NULL |
| Rok výroby | YEAR (4) | NOT NULL |
| Počet KM | MEDIUMINT (7) | NOT NULL |
| Platnost STK | DATE | NULL |
| Fotka | VARCHAR (300) | NULL |
| Model | VARCHAR (45) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Událost –** V této tabulce se nachází specifické údaje o jednotlivých událostech vytvořených pro dané vozidlo. Atribut „Poznámky“ slouží pro základní informace o plánovaném servisním zákroku. Atribut „Podrobnosti“ obsahuje kompletní textovou zprávu o provedeném servisu a komentář k atributu „Fotky“. Atribut „Fotky“ je typu „VARCHAR“, jelikož se jedná o textovou cestu k adresáři obsahující nahrané fotografie k dané události. Ostatní atributy byly již popsány v předešlé části projektu.

*Tabulka 5 - Datový slovník - "Událost"*

| Událost | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Název události | VARCHAR (45) | NOT NULL |
| Datum zahájení | DATE | NULL |
| Datum ukončení | DATE | NULL |
| Poznámky | VARCHAR (45) | NULL |
| Datum vytvoření | DATE | NOT NULL |
| Podrobnosti | VARCHAR (300) | NULL |
| Fotky | VARCHAR (300) | NULL |

*Zdroj: Vlastní zpracování*

**Adresa –** Tato tabulka je provázána na dílnu a na osobní údaje uživatele a mechanika. Při registraci uživatele je ve formuláři nutné vyplnit i údaje v této tabulce.

*Tabulka 6 - Datový slovník - "Adresa"*

| Adresa | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Město | VARCHAR (50) | NOT NULL |
| Ulice | VARCHAR (45) | NOT NULL |
| Číslo popisné | INT (6) | NOT NULL |
| PSČ | INT (5) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Dílna –** Obsahuje dodatečné údaje o dílnách, pokud jich má podnik více.

*Tabulka 7 - Datový slovník - "Dílna"*

| Dílna | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Název | VARCHAR (45) | NOT NULL |
| Poznámka | VARCHAR (45) | NULL |

*Zdroj: Vlastní zpracování*

**Práva –** Práva obsahují v podobě názvu úroveň práv, které mají uživatelé systému přiřazené. Typy práv jsou automaticky vložena do databáze při tvorbě databáze.

*Tabulka 8 - Datový slovník - "Práva"*

| Práva | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Název | VARCHAR (10) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Značka –** Tabulka značka obsahuje názvy značek vozidel do sysétmu vložené při tvorbě databáze.

*Tabulka 9 - Datový slovník - "Značka"*

| Značka | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Název | VARCHAR (45) | NOT NULL |

*Zdroj: Vlastní zpracování*

**Mechanik –** Tato tabulka obsahuje dodatečné údaje o mechanicích.

*Tabulka 10 - Datový slovník - "Mechanik"*

| Mechanik | | |
|---|---|---|
| **Atribut** | **Datový typ (délka)** | **Podmínka** |
| Mzda | INT (6) | NOT NULL |

*Zdroj: Vlastní zpracování*

### 4.2.2 Omezení možností v systému pro uživatele

Jako každý systém musí mít i tento přesněji stanovené možnosti změn vlastních údajů a dalších paramterů u uživatelů. Návrh je udělán z důvodu zamezení změn data v databázi, která by mohla způsobovat problémy, jako je například změna příjmení apod. Mnoho omezení v systému je již naznačeno prostřednictvím diagramů a plánovanou konstrukcí systému. Níže však budou stanoveny další omezení, která zjednoduší následnou implementaci systému.

- Třída „Osobní údaje“: Uživateli bude umožněno měnit pouze email a telefon, tyto změny se následně pomocí vytvořených příkazů zapíší do databáze.
- Třída „Vozidlo“: Zde uživatel bude mít možnost měnit počet najetých kilometrů a datum platnosti technické kontroly.
- Třída „Událost“: Do této části systému uživatel nebude umožněno nijak zasahovat, bude mu pouze zobrazena v rámci provázání dat z databáze na jeho vozidla.
- Třída „Adresa“: Zde bude umožněno uživatel měnit veškeré údaje.
- Třída „Klient“: V této části uživateli bude umožněno změnit své IČO.

Do ostatních dat v databázi uživatel nebude mít možnost nijak zasahovat, všechny ostatní údaje lze měnit pouze prostřednictvím kontaktování a dohodou s administrátorem.

### 4.2.3 Struktura stránek

V této části bude velmi stručně popsána struktura některých stránek tohoto systému a to z pohledu uživatele a z pohledu mechanika. Nejprve budou popsány části, které budou přítomny na každé stránce a následně dojde na rozebrání několika vybraných stránek ze strany uživatele.

- Každá stránka bude tvořena ze 3 základních sloupců (left sidebar, middle bar a right sidebar). Následně bude obsah jednotlivých sloupců přizpůsoben typu uživatele a dané stránce.
- Jednotlivé stránky budou stylovány a responzivně designovány pomocí frameworku Bootstrep a přidaného externího css souboru pro další úpravy stylů stránky.
- Na každé stránce bude na začátku stránky aplikován php zabezpečovací mechanismus, který bude vyžadovat, aby byl uživatel nebo mechanik vždy autentizován a měl příslušná

oprávnění pro vstup na danou stránku a zároveň bude mechanismus ošetřen tak, aby provedl příslušná opatření v případě vstupu neoprávněné osoby na tuto stránku v podobě varovné hlášky a přesměrování na úvodní stránku systému.

- Všechna připojení k databázi, úpravy záznamů, vkládání a výpisy se budou provádět pomocí PDO metody ošetřené proti chybování, která potřebuje vždy pro své připojení hostitele, uživatelské jméno, heslo a název databáze.
- Použitá metoda odesílající data na server ke zpracování bude metoda „POST“ případně se výjimečně může objevit metoda „GET“.
- Proti SQL Injection budou příkazy ošetřeny pro dotazy PDO pomocí metody „prepare“, v rámci vazebních parametrů pomocí „bindParam“ a vstupy budou ošetřeny pomocí „htmlspecialchars“.

**Uživatel – Hlavní stránka**

Na této stránce se bude nacházet php kód pro získání id uživatele z relace a následně budou provedeny potřebné výpisy z databáze do profilu uživatele. Bude zde umožněna editace osobních údajů, adresy a „IČA“ uživatele. Na stránce bude zprovozněna JavaScript funkce umožňující změnu hesla v podobě modálního okna, které po vyplnění zavolá odpovídající skript pro zaheshování hesla a uložení do databáze. Na stránce bude zakomponována miniatura pro otevření online chatu pro psaní s mechanikem.

**Uživatel – Přidání vozidla**

Tato stránka bude obsahovat php kód pro zpracování formuláře po vyplnění údajů o vozidle uživatelem na přidání vozidla a odeslání formuláře. Dále se na stránce bude nacházet JavaScript kód, který bude formátovat pole s datumem platnosti STK do požadovaného formátu. Na stránce bude opět zakomponována miniatura chatu.

**Uživatel – Podrobnosti o vozidle**

Tato stránka bude prostředkem pro zobrazení a správu některých inforamcí o daném vozidle, které bude z databáze vypsáno pomocí id vozidla získaného z relace. Dále zde budou vypsány všchny události, které byly či budou vytvořené na dané vozidlo. Opět se zde naimplementuje možnost využít chat.

### 4.2.4 Online chat uživatele s mechanikem

Jako chatovací aplikace bude do systému implementována open source aplikace Tawk.to, která je zdarma a nabízí mnoho možností přizpůsobení chatu dle vlastních požadavků. Dokumentace Tawk.to obsahuje předdefinované skripty pro různé možnosti úprav chatu.

Pro tento projekt bude vhodné, aby při psaní zprávy ze strany uživatele dostal mechanik informaci, jaký uživatel ze systému píše. Taková funkcionalita je již předdefinována v JavaScript API od Tawk.to pod názvem „visitor{};“ přiložená níže, která podle nastavení vstupních údajů vypíše mechanikovi do aplikace potřebné údaje. V tomto případě se bude jednat o jméno, příjmení a email. Tato varianta chatu je zvolena z toho důvodu, jelikož realizace vlastního chatovacího serveru by byla velmi zdlouhavá a složitá.

JavaScript funkce pro identifikaci uživatele:

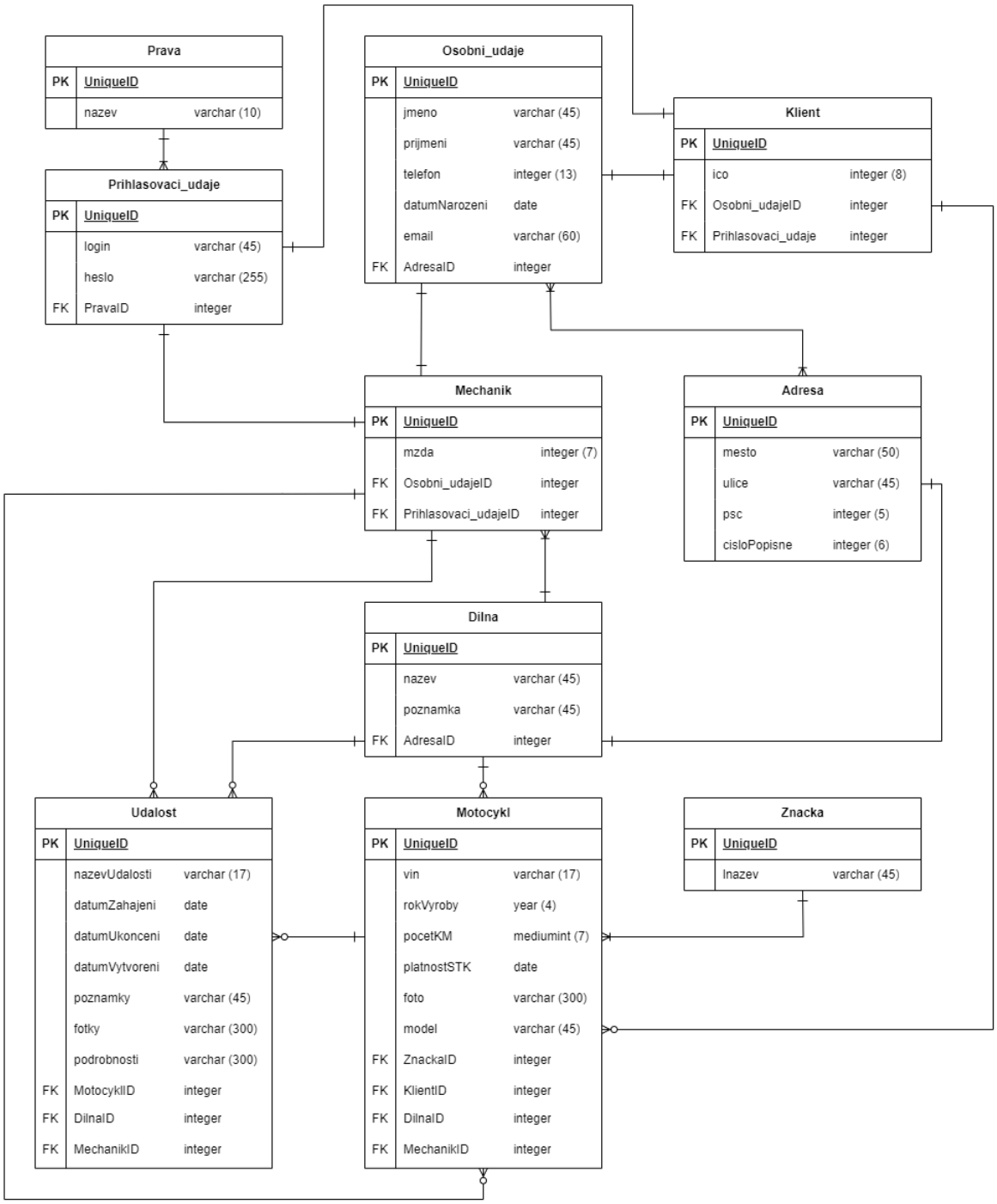
```
window.Tawk_API = window.Tawk_API || {};
window.Tawk_API.visitor = {
   name  : 'Name',
   email : 'email@email.com'
};
```

### 4.2.5 ER Diagram

ER diagram je pro návrh systému vhodný z důvodu popisu struktury databáze a jasného zvýraznění tabulek a vztahů mezi nimi. Každá tabulka obsahuje svůj primární klíč (PK) v podobě „id“ a propojení tabulek mezi sebou je provedeno pomocí cizích klíčů (FK).

Některé tabulky byly záměrně rozděleny na více částí, např. tabulka „Mechanik“ a hodnoty z „Prihlasovaci_udaje“. Je to z toho důvodu, aby nedošlo k duplicitním záznamům v jednotlivých tabulkách, tedy redundanci dat. Takto jsou stanovené přihlašovací údaje přehledně v jedné tabulce a jsou určeny v tabulce mechanik pouze pomocí jejich ID, jelikož by bylo nutné ty samé údaje zapisovat zvlášť u tabulky Klient. Obdobně je takto řešena tabulka Adresa vůči tabulce Klient a Dilna apod.

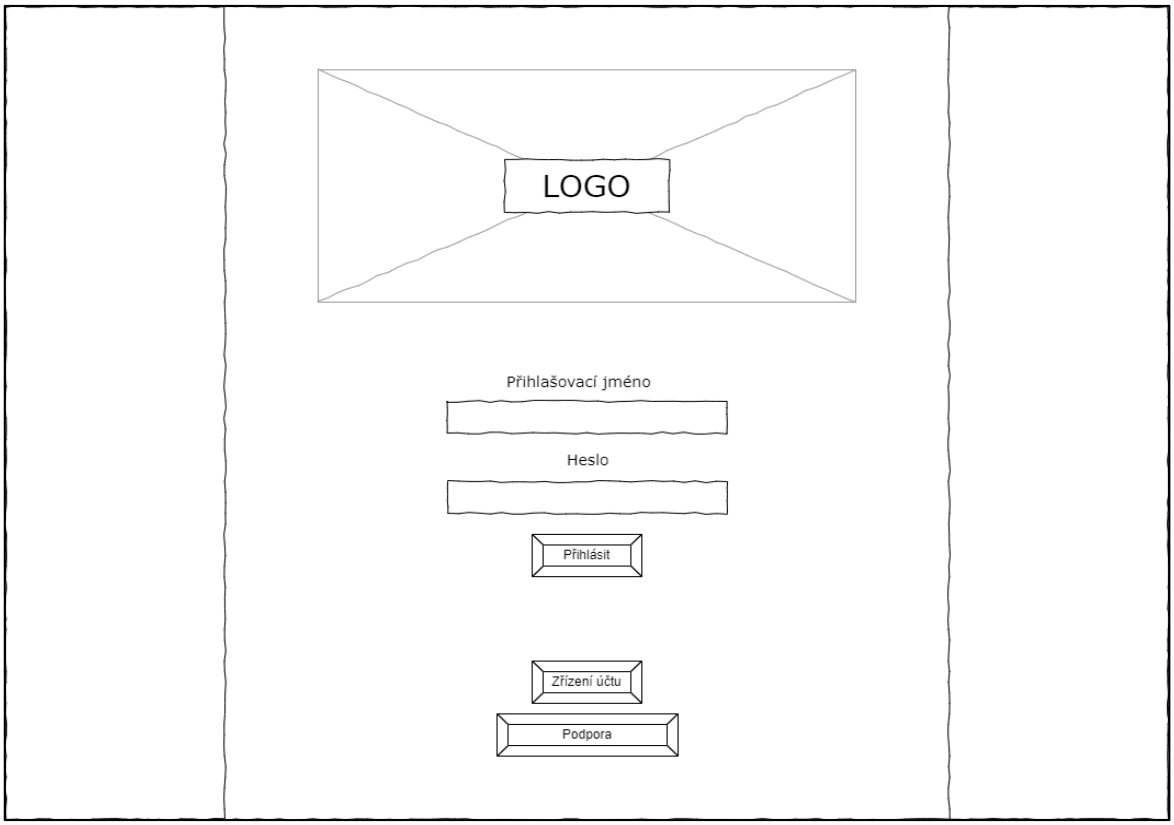
Diagram je navržený tak, aby byl pochopitelný a jednoduše čitelný a všechny vzájemné vztahy jsou zde graficky znázorněny. Struktura databáze je navržena takto s ohledem na budoucí rozšiřování a změny v požadavcích ze strany klientů na systém. Díky většímu rozdělení tabulek bude možné přidat a spárovat další tabulky bez větších změn v existující struktuře.



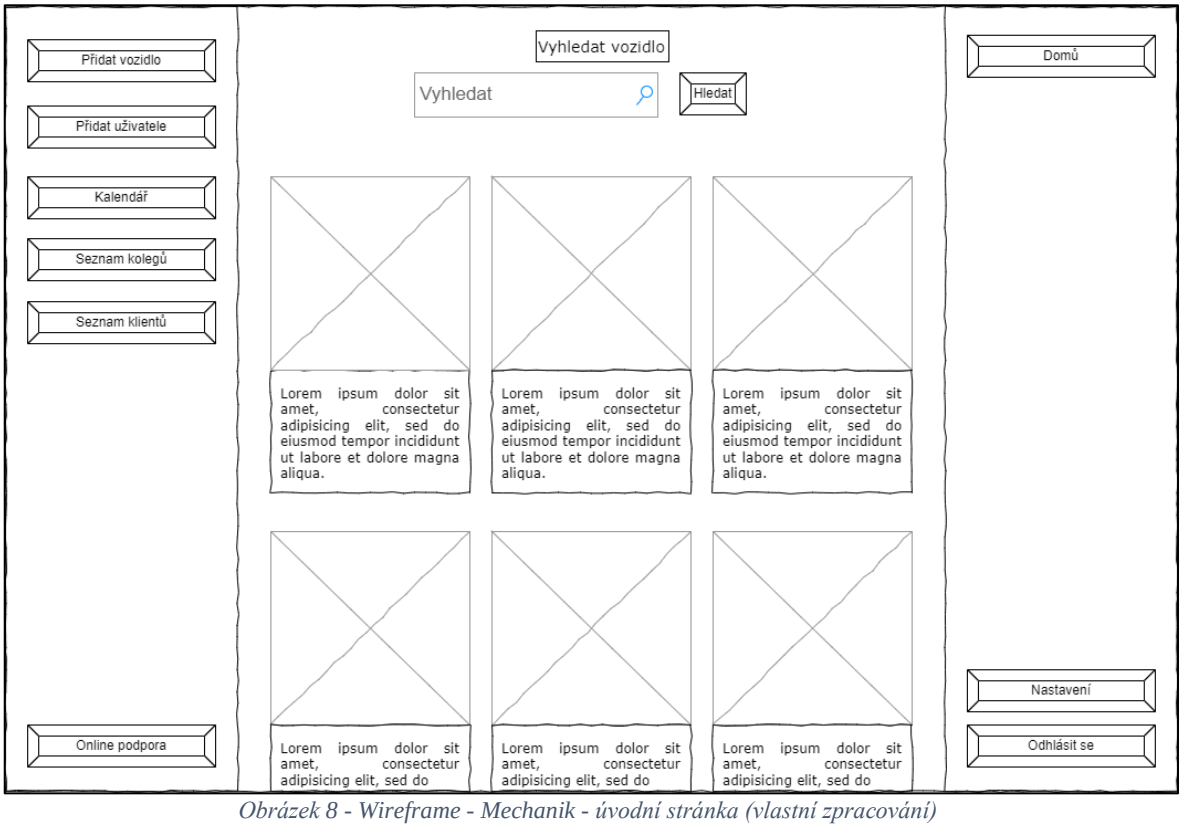

*Obrázek 6 - ER Diagram (vlastní zpracování)*

### 4.2.6 Wireframe model

Wireframe model má za úkol definovat distribuci prvků na stránce a načrtnout vzhled stránky. Pro zjednodušení konstrukce webových stránek byly provedeny dva základní návrhy přiložené níže, podle kterých se následně prováděl reálný grafický návrh.

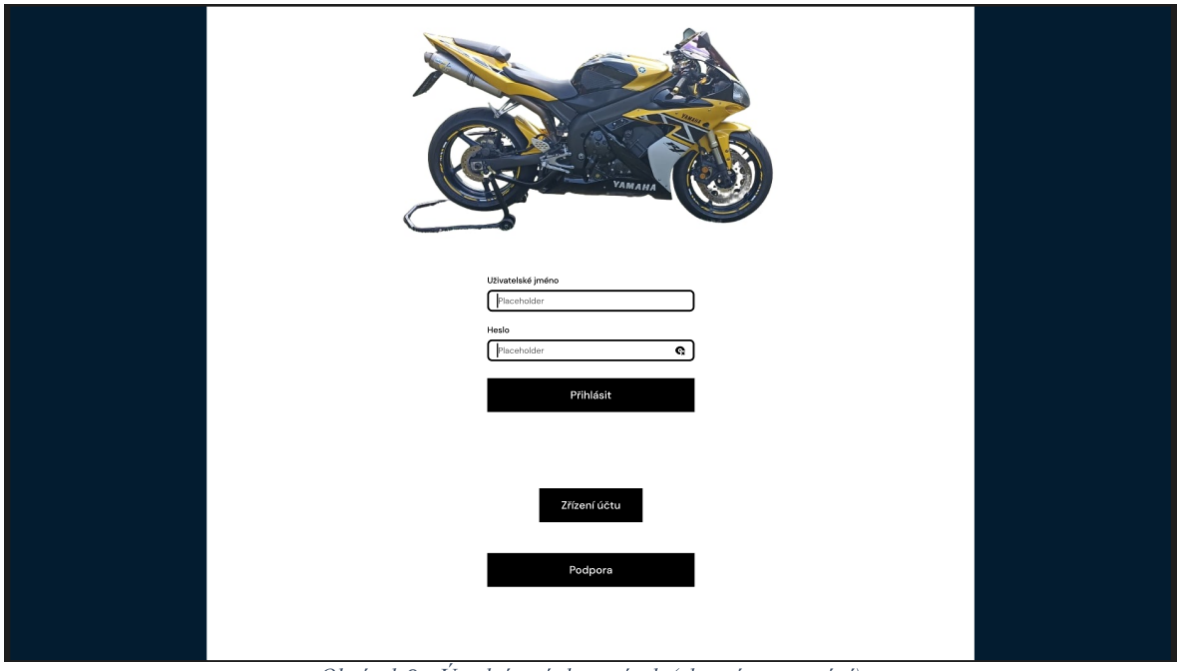



*Obrázek 7 - Wireframe - úvodní stránka (vlastní zpracování)*

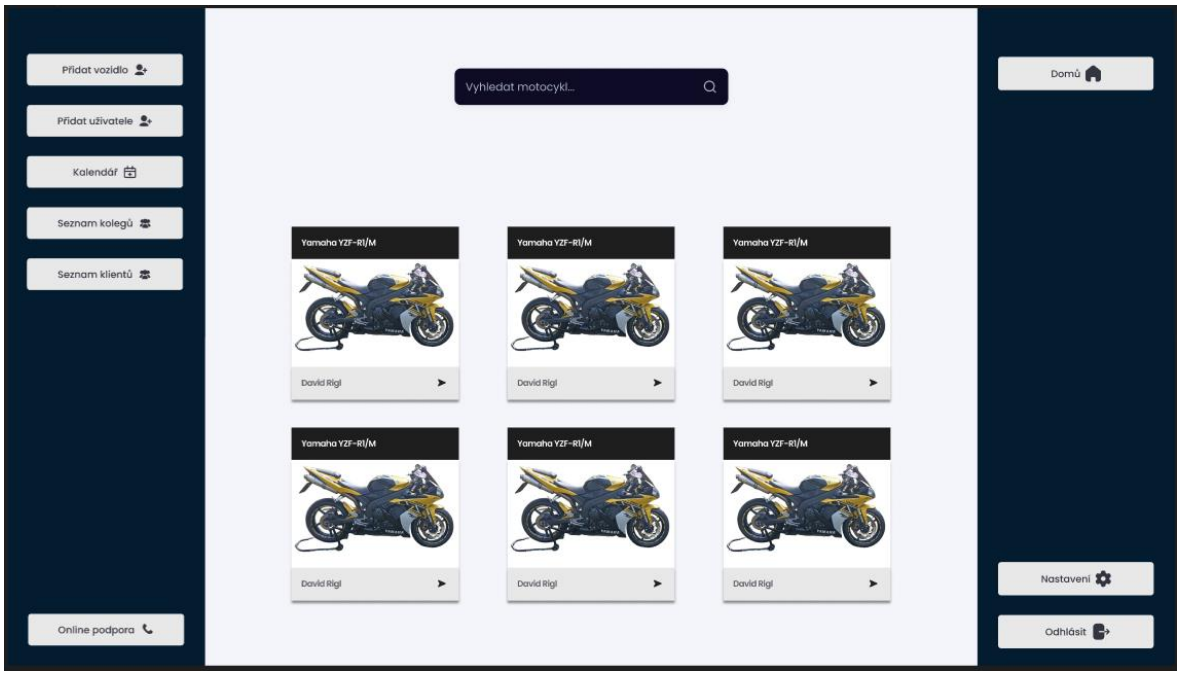



*Obrázek 8 - Wireframe - Mechanik - úvodní stránka (vlastní zpracování)*

### 4.2.7 Grafický návrh systému




*Obrázek 9 - Úvodní stránka - návrh (vlastní zpracování)*




*Obrázek 10 - Mechanik - úvodní stránka - návrh (vlastní zpracování)*

## 4.3 Implementace projektu

Prvním krokem bude pořízení a sestavení zvolených hardwarových prvků a následně dojde na jejich zprovoznění. Dále pomocí aplikace Raspberry Pi Imager dojde na spuštění instalace systému, kde bude zvolena verze 64bit, která má nižší zatížení a je bez grafického rozhraní. Dále se zvolí “choose storage“, kde dojde na výběr disku a v nastavení hostname

se zaškrtnutím SSH, dále bude nastaveno přihlašovací jméno a heslo a na závěr nastavení WIFI. Tyto kroky se uloží a zapíšou. Tímto je zprovozněn systém a je možné se tedy do systému přihlásit.

Velmi důležitou součástí je pořízení vlastní internetové domény, a to z důvodu jak důvěryhodnosti, tak k usnadnění přístupu uživatelů na vybraný web. Pro tento projekt v rámci testování bude stačit velmi levná variant domény od internetového obchodu Wedos, který patří i mezi nejlevnější v České republice. Pořízená doména pro tuto testovací verzi stála necelých 16 korun českých na rok. Zakoupená doména pro zprovoznění systému a veškerých funkcí je rigldavid.fun. Díky této doméně bude možné systém zveřejnit do internetu a otestovat její návrh ve fiktivním provozu před případným nasazením systému do vybrané firmy či podniku.

### 4.3.1 Zpřístupnění systému do internetu

V této části je nutné zabezpečit, aby byl systém viditelný v internetu. Ke zviditelnění byl zvolen globální poskytovatel DNS služeb Cloudflare, který se zaměřuje i na ochranu stránek. V této službě po registraci byla registrována zakoupená doména a nastaveny příslušné DNS a name servery. Díky Cloudflare bude systém možné otevřít pomocí https protokolu na webových stránkách.

### 4.3.2 Nastavení vzdáleného přístupu

Pro zprovoznění VPN přístupu je nutné nastavení specifikace routeru. V rámci bezpečnosti je podstatné mít zabezpečený router, kde lze nastavit firewall s oprávněním přístupů pouze za určitých podmínek a je potřeba mít veřejnou IP adresu, kterou lze získat od poskytovatele. Samotné nastavené routeru pro vzdálený přístup na HWR v nastavení směrování portu 443 na IP adresu daného serveru, která musí být přidělována staticky.

Pro instalaci aplikace Open VPN bude použit PIVPN, který má při instalaci průvodce s nastavením a zároveň vytváří certifikáty a jejich konfiguraci.

Instalační příkaz: curl -L https://install.pivpn.io | bash


*Obrázek 11 - Instalace Open VPN (vlastní zpracování)*

Dále příkaz pivpn nopass, který zajistí vytvoření certifikátu pro připojení na server. V tomto kroku bude nastaven název souboru v průběhu konfigurace. Certifikát se uloží do adresáře a tento soubor musí být nakopírován do zařízení, které bude vzdáleně na server přistupovat. Tento krok se provede příkazem:

scp *****.ovpn uživatel@ippc:/home/*****/dokumenty/*****.ovpn

Dalším krokem je stáhnutí aplikace open VPN pro vzdálené připojení k serveru a nahrání do této aplikace stažený certifikát.

Tímto je vzdálený přístup zrealizován. Samotný přístup může být zprostředkován například pomocí aplikace Putty, která umožňuje připojování k serveru pomocí SSH. Aplikaci je však nutné nastavit dle vlastních specifikací například viz obrázek níže.

### 4.3.3 Instalace aplikace Docker

Před zahájením instalace je nutné provést aktualizaci systému a softwarové verze. Tento krok se provádí standardně před každou instalací balíčků. Příkazy:

sudo apt-get update
sudo apt-get upgrade

Pro instalaci dockeru na serveru použít příkaz:
curl -fsSL https://get.docker.com -o get-docker.sh

Dále je potřeba spustit příkaz, který vytvoří administrátorské balíčky pro dané prostředí a je nutné udělit oprávnění pro přístup jednotlivým účtům do dockeru příkazy:

sudo sh get-docker.sh
sudo usermod -aG docker *****

Těmito kroky je docker zprovozněn a připraven k používání. Po tomto kroku je vhodné provést restart systému pro uložení a načtení provedených změn a instalací pomocí příkazu:

sudo reboot

**Instalace a konfigurace v prostředí Portainer.io**

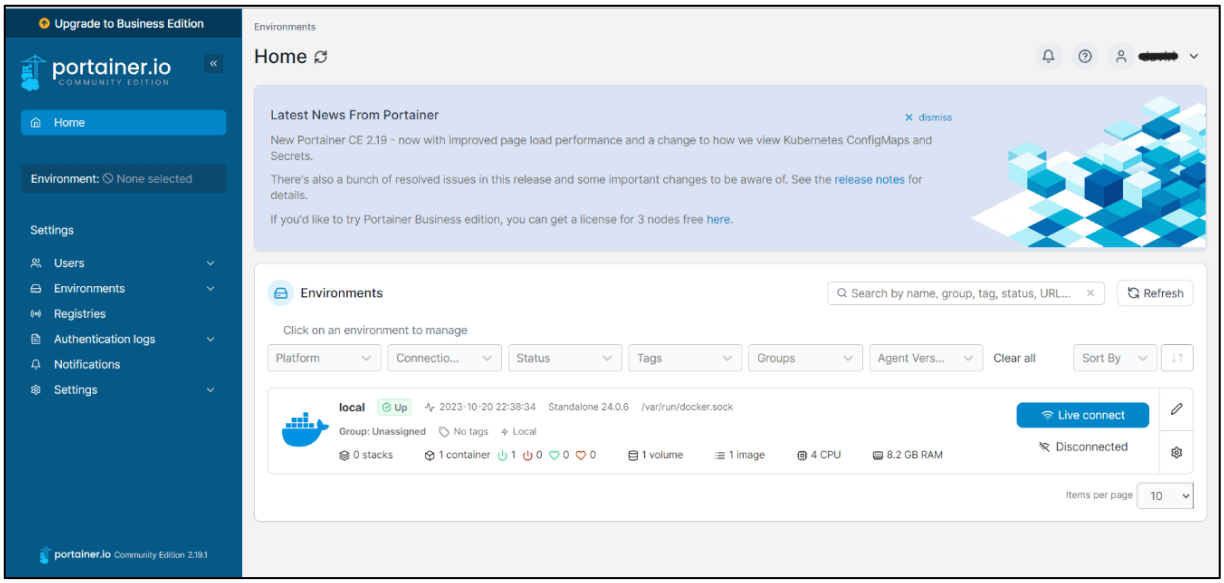
Prvním krokem bude příkaz pro stažení portaineru:

sudo docker pull portainer/portainer -ce:latest

a dále příkazem níže přímo specifikovat použití a nastavení portaineru:

docker run -d -p 8000:8000 -p 9443:9443 -p 9000:9000 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest

Dále je potřeba pro zprovoznění zadat IP adresu s portem serveru Portaineru a zde vytvořit přihlašovací údaje.

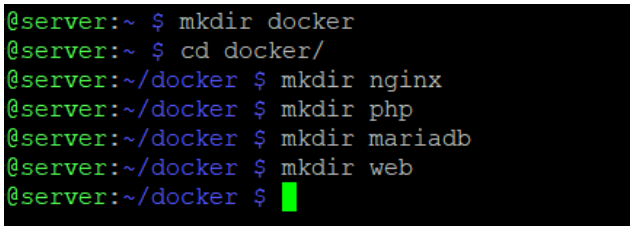



*Obrázek 12 - Správa v prostředí Portainer (vlastní zpracování)*

Následující krok je vytvoření adresářů pro jednotlivé systémové doplňky. Je nutné vytvořit adresář pro docker, tedy příkazem:

mkdir Docker

Do tohoto adresáře se přesunout. V tomto adresáři se vytvoří další adresáře obdobným způsobem pro databázi, webový server, php admina a web jako takový.




*Obrázek 13 - Vytvořené adresáře na serveru (vlastní zpracování)*

Posledním krokem je v dockeru vytvoření stacku, ve kterém je nastavená databáze MariaDB, PHPmyadmin a Apache jako web server.
Zde se vytvořil adresář „/context“, ve kterém jsou příkazy pro doinstalování php modulů a servisy do jednotlivých kontejnerů (sudo docker compose build).

Všechny tyto aplikace jsou spustitelné přes docker-compose.yaml pomocí příkazu docker compose up -d v adresáří /home/inzenyr/docker/moto_dilna.

Níže je přiložen obrázek txt kódu v Dockerfile na serveru, který má za úkol poskytnout instrukce pro vytvoření image Docker kontejneru s PHP 7.4 s Apach serverem pro běhové PHP prostředí. Následně instaluje rozšíření PDO sloužící pro MySQL pro PHP, který zajistí komunikaci PHP s MySQL databází pomocí rozhraní PDO. Další částí je aktualizace balíčků v systému a instalace dalších pro kontejner jako je git, zip, unzip, libzip-dev, cron nano a default-mysql-client. Další podstatnou instalací je Composer pro správu závislostí v PHP. Následuje instalace Pythonu 3 a pip pro balíčky Pythonu s následnou instalací „mysql-connector-python“ pro komunikaci Python skriptů s databází. Dále je nutné pro odesílání emailů nainstalovat PHPMailera knihovnu pomocí Composeru.

Následně dochází na kopírování script.py do kontejneru, který zabezpečuje automatickou kontrolu stavu technické kontroly na vozidlech a případně odesílá upozorňující email. Na tento script.py je navázán crontab pro spuštění skriptu každý den a následuje vytvoření adresáře /var/log/cron.log pro logování výstupu crontabu. Nakonec se spouští CMD instrukce pro spuštění cronu a Apach serveru v případě spuštění kontejneru.

```
  GNU nano 7.2                                                                  context/Dockerfile
FROM php:7.4-apache

# Instalace rozšíření PDO MySQL pro PHP
RUN docker-php-ext-install pdo_mysql

# Aktualizace balíčků a instalace pro Composer a další...
# Nahrazení mysql-client za default-mysql-client
RUN apt-get update && \
    apt-get install -y git zip unzip libzip-dev cron nano default-mysql-client && \
    docker-php-ext-install zip && \
    rm -rf /var/lib/apt/lists/*

# Instalace Composeru
RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');" && \
    php composer-setup.php --install-dir=/usr/local/bin --filename=composer && \
    php -r "unlink('composer-setup.php');"

# Instalace Pythonu a pip
RUN apt-get update && \
    apt-get install -y python3 python3-pip && \
    rm -rf /var/lib/apt/lists/*

# Instalace mysql-connector-python přes pip
RUN pip3 install mysql-connector-python

# Nastavení adresáře pro Composer
WORKDIR /home/

# Instalace PHPMaileru
RUN composer require phpmailer/phpmailer

# Kopírování Python skriptu do kontejneru
COPY script.py /script.py

# Crontab do root crontabu
RUN (crontab -l 2>/dev/null; echo "18 17 * * * python3 /script.py >> /var/log/cron.log 2>&1") | crontab -

# Vytvoření log souboru pro sledování cronu
RUN touch /var/log/cron.log

# Spuštění Apache a cronu v popředí
CMD cron && apache2-foreground
```

*Obrázek 14 - Kód Dockerfile (vlastní zpracování)*

Dalším podstatným krokem je konfigurace služeb pomocí Docker Compose. V přiloženém obrázku níže je kód souboru docker-compose.yml, který nejprve definuje seznam služeb, které má Compose spustit. Dochází zde k definici konfigurace pro spuštění MariaDB databázového serveru, následně na Apach webového serveru s PHP a nakonec se definuje konfigurace pro spuštění skriptu.

```
services:
  mariadb:
    image: mariadb
    container_name: mariadb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ******
      MYSQL_DATABASE: moto_dilna_db
    volumes:
      - /home/uzivatel/docker/moto_dilna/mariadb:/var/lib/mysql
    ports:
      - "3306:3306"

  web:
    build:
      context: ./context
    container_name: apache-php-web
    user: "1000:1000"
    restart: always
    depends_on:
      - mariadb
    volumes:
      - /home/uzivatel/docker/moto_dilna/web:/var/www/html
    ports:
      - "80:80"
    command: ["apache2-foreground"]

  script-runner:
    build:
      context: ./context
    container_name: script-runner
    depends_on:
      - mariadb
    command: ["cron", "-f"]
```

*Obrázek 15 - Kód docker-compose.yml (vlastní zpracování)*

### 4.3.4 Databáze

Zvolený typ databáze je MariaDBD a je implementována dle návrhu ve volně dostupné aplikaci MySQL Workbench, ze které je možné navrženou databázi vygenerovat v kódové podobě, kterou lze po určitých úpravách, aby odpovídala syntaxi SQL aplikovat přímo do systému. Je postavena na základě datového slovníku, návrhu ER diagramu a jednotlivých vytvořených sekcí jako je klient, mechanik, dílna, motocykl apod. Dalším důležitým krokem je odladění a propojení potřebných částí tak, aby databáze vhodně komunikovala a propojovala správná odvětví projektu. Podstatným pilířem databáze je sekce událost a motocykl, od kterého se odvíjí mnoho dalších částí databáze.

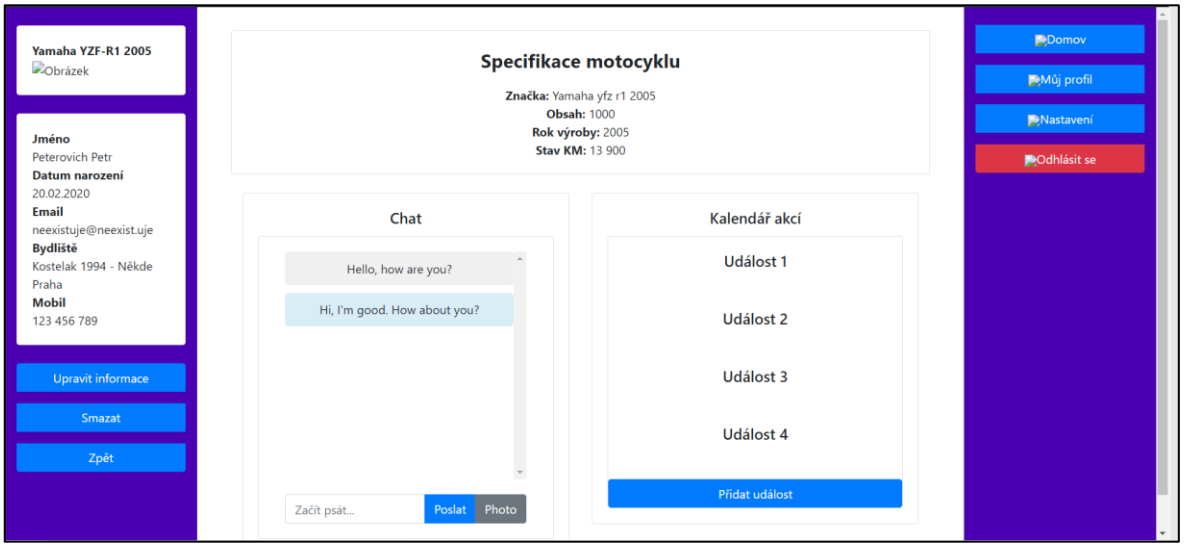
Po dokončení databáze je databáze implementována do projektu a bylo provedeno propojení s vytvořeným systémem. V rámci otestování funkčnosti databáze se vytvořily testovací data pro každou tabulku, která se v databázi nachází a bylo sledováno chování databáze pomocí příkazů „Update…“, „Delete…“, „Select…“ apod. Dalším krokem je otestování zápisu informací do databáze z webového prostředí a reakce databáze na příkazy z webu pro výpisy požadovaných údajů a případné mazání či úpravy dat v jednotlivých

tabulkách. Tato část testování databáze byla velmi důležitá před celkovou realizací systému, jelikož jsou následné úpravy databáze po realizaci již složitější a mohou s sebou nést negativní následky.

### 4.3.5 Realizace frontendu

Základem pro realizaci frontendu je tvorba webových stránek pomocí html a css. K samotné tvorbě je vhodné použít vytvořený grafický návrh v podobě framework diagramů či přímo vytvořený desgin v odpovídajích programech, jak má systém vypadat.

Tyto stránky slouží především pro následnou možnost otestování všech funkcionalit systému a optimalizace vlastností při jeho používání. Vzhled stránek tedy nemusí být odpovídající grafickému návrhu a ani není v této části cílem se zaměřit na design systému. Dále je nutné otestovat responzivitu systému z důvodu využívání systému na rozdílných zařízeních.




*Obrázek 16 - HTML stránka - Specifikace motocyklu (vlastní zpracování)*

Níže je přiložena část kódu této stránky.

```
<div class="card">
          <div class="card-body">
            <b>Yamaha YZF-R1 2005</b> <!--Odkaz na název prokliklé moto-->
            <img
src="https://images.piaggio.com/motoguzzi/vehicles/gu9032278eun00/gu9032278eun
01/gu9032278eun01-01-s.png" alt="Obrázek" class="img-fluid">
          </div>
        </div>
        <br>
        <div class="card">
          <div class="card-body">
            <b>Jméno</b><br>
            <!--Odkaz--> Petercvich Petr<br>
            <b>Datum narození</b><br>
            <!--Odkaz--> 20.02.2020<br>
            <b>Email</b><br>
            <!--Odkaz--> nikdy@sšoriš.cnie<br>
            <b>Bydliště</b><br>
            <!--Odkaz--> kostelak 1994 - Někde Praha<br>
            <b>Mobil</b><br>
            <!--Odkaz--> 728 526 323232<br>
          </div>
        </div>
        <br>
        <div class="button-container">
          <button type="button" class="btn btn-primary mb-3">Upravit
informace</button>
          <button type="button" class="btn btn-primary mb-3">Smazat</button>
          <button type="button" class="btn btn-primary mb-3">Zpět</button>
          <br>
          <button type="button" class="btn btn-primary mb-3">Chat
online</button>
        </div>
      </div>
      <div class="col-lg-8 col-md-8">
        <div class="middle-bar">
          <div class="enter">
            <div class="centered-text">
              <div class="content">
                <div class="card">
                  <div class="card-body">
                    <b><p style="font-size: 25px;">Specifikace
motocyklu</p></b>
                    <b>Značka:</b> <!--Odkaz--> Yamaha yfz r1 2005<br>
                    <b>Obsah: </b> <!--Odkaz--> 1000<br>
                    <b>Rok výroby:</b> <!--Odkaz--> 2005<br>
                    <b>Stav KM:</b> <!--Odkaz--> 13 900<br>
                  </div>
                </div>
```

*Obrázek 17 - Část zdrojového kódu stránky (vlastní zpracování)*

Jedná se pouze o vzhledový html návrh stránek bez jakékoliv funkčnosti. Následně byly do kódů stránek vkládány potřebné funkcionality a kód byl postupně předěláván dle potřeb.

### 4.3.6 Implementace online chatu

Pro implementaci chatu do systému je nutné provést registraci do webu Tawk.to a následně si podle svých potřeb přizpůsobit toto prostředí. Následným krokem je vygenerování Tawk.to skriptu pro implementaci chatu do systému a náledné přizpůsobení vlastností a vyzuální stránky chatu dle vlastních požadavků. Vygenerovaný kód chatu se musí vložit do odpovídající stránky na konec, kde má být chat funkční. V tomto případě se chat implementuje na hlavní stránku uživatele, na podrobnosti o motocyklu, přidání motocyklu a na servisní historii motocyklu.

Nyní je již chat v systému funkční a lze psát z uživatelského profilu do chatovací aplikace mechanikovi. Následuje nastavení pro identifikaci uživatele v případě napsání zprávy, které se provede pomocí určeného skriptu a v něm nastavení odpovídajích proměnných pro jméno, příjmení a email. Níže přiložený JavaScript zabezpečuje odpovídající funkcionalitu pro tento systém.

```
<script type="text/javascript">
var Tawk_API=Tawk_API||{}, Tawk_LoadStart=new Date();

// Předdefinování atributů a funkcí
window.Tawk_API = window.Tawk_API || {};
window.Tawk_API.onLoad = function() {
    Tawk_API.setAttributes({
        'name' : '<?php echo $jmeno . " " . $prijmeni; ?>',
        'email' : '<?php echo $email; ?>',
    }, function(error){});
    Tawk_API.addEvent('Informace o klientovi', {
        'Jméno': '<?php echo $jmeno; ?>',
        'Příjmení': '<?php echo $prijmeni; ?>',
        'E-mail': '<?php echo $email; ?>',
    }, function(error, success){});
};

// Načtení skriptu Tawk.to
(function(){
    var s1=document.createElement("script"),s0=document.getElementsByTagName("script")[0];
    s1.async=true;
    s1.src='https://embed.tawk.to/[redacted]';
    s1.charset='UTF-8';
    s1.setAttribute('crossorigin','*');
    s0.parentNode.insertBefore(s1,s0);
})();
</script>
```

*Obrázek 18 - Skript pro chatovací okno u účtu uživatele (vlastní zpracování)*

## 4.4 Implementovaný systém

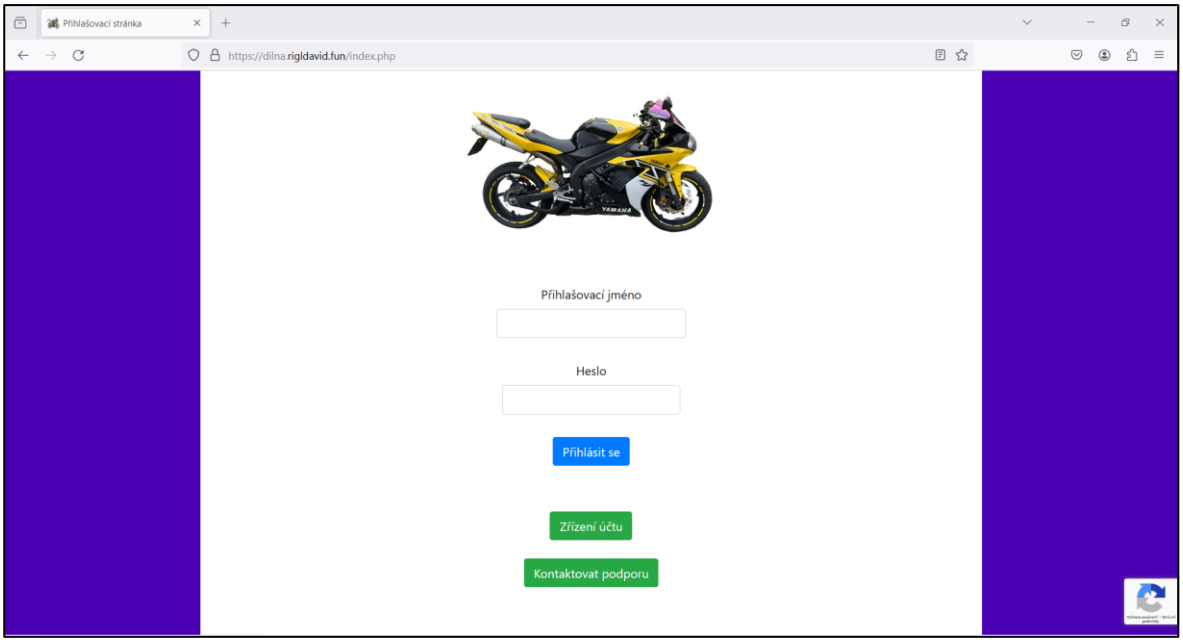
Cílem v této části je ukázat základní funkce systému a popsat jejich činnost. Veškeré fotografie motocyklů v tomto projektu jsou soukromě pořízené a v době pořízení se jedná o motocykly, které jsou či byly v osobním vlastnictví. Všechna jména a příjmení včetně všech ostatních údajů jsou smyšlená a slouží pouze pro účel testování systému. Všechny údaje motocyklů a specifikace o nich jsou smyšlené.

Po ukázce funkčnosti systému dojde na popis dalších funkcí v systému, které nemusí být na první pohled na systém patrné, jejich funkcionalita je však pro systém podstatná a u některých z nich bude ukázán a popsán skript či kód, který jejich činnost vykonává.

### 4.4.1 Ukázka systému

Níže bude ukázán zprovozněný systém pomocí printscreenů z počítače a několik ukázek z telefonu. Dále bude u každé z fotografií základní popis stránky s funkcionalitami a popis dalších vlastností tlačítek apod.

### Hlavní stránka pro autentizaci do systému

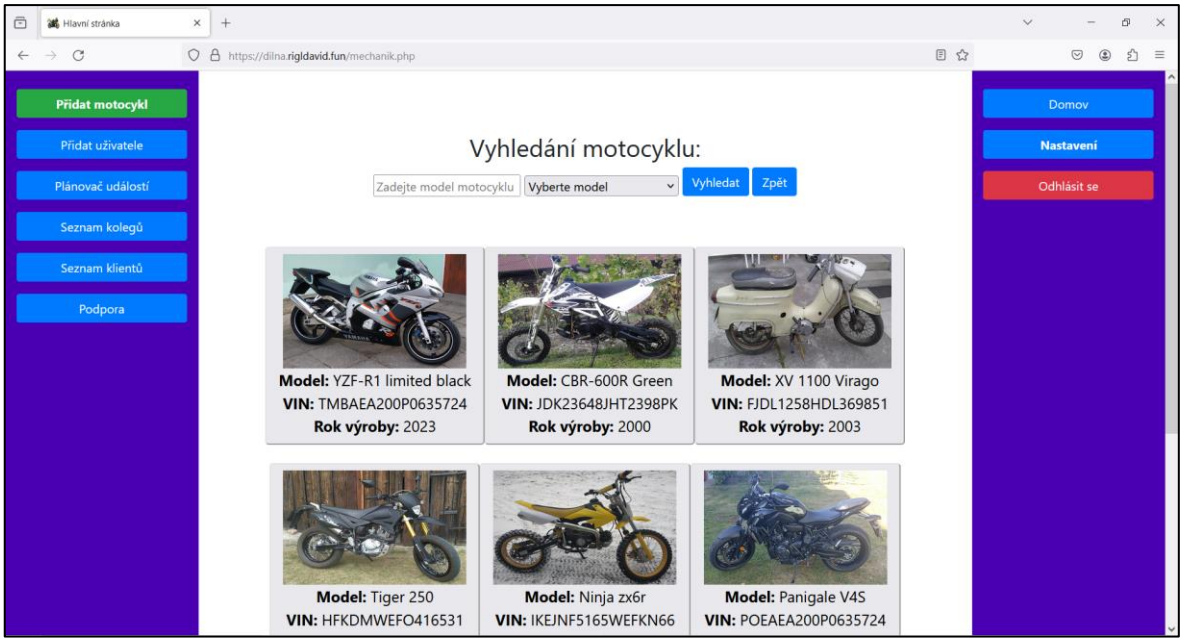



*Obrázek 19 - Hlavní stránka – PC (vlastní zpracování)*

Tato hlavní stránka slouží pro autentizaci klientů a mechaniků do systému, kdy jsou po zadání přihlašovacích údajů přesměrování na odpovídající stránky. Dále se na stránce nachází tlačítko „Zřízení účtu“, které obsahuje formulář na vyplnění potřebných údajů, které budou následně po bezpečnostní kontrole zapsány do databáze. Formulář je v backendové části ošetřen proti špatnému vyplnění údajů. Pod tímto tlačítkem se nachází tlačítko „Kontaktovat podporu“, obsahuje formulář pro odeslání dotazu na podporu.

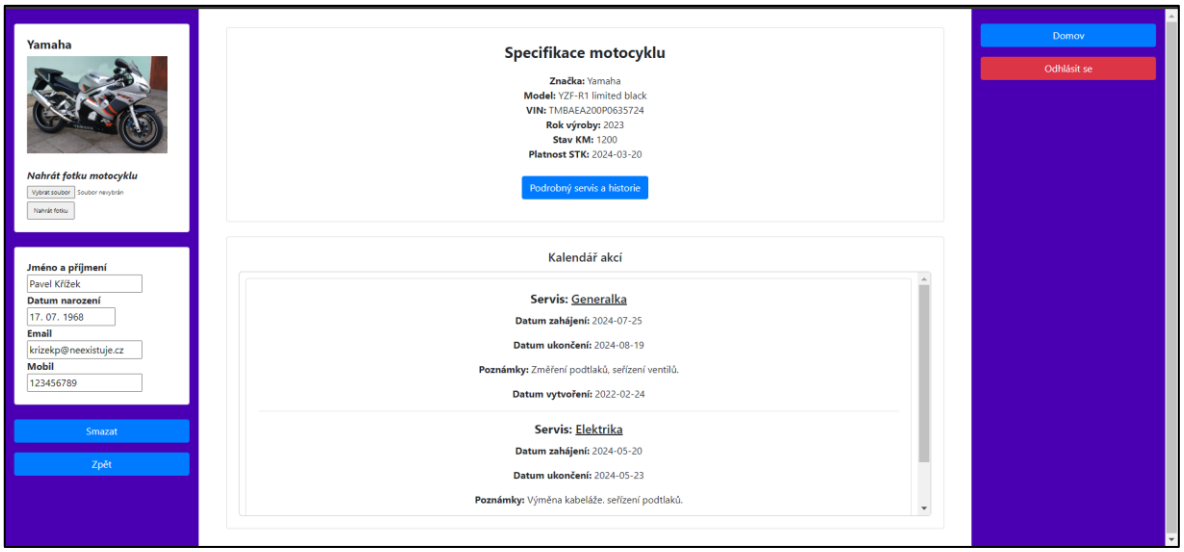

*Obrázek 20 - Hlavní stránka – telefon (vlastní zpracování)*

### Webové stránky - část mechanik

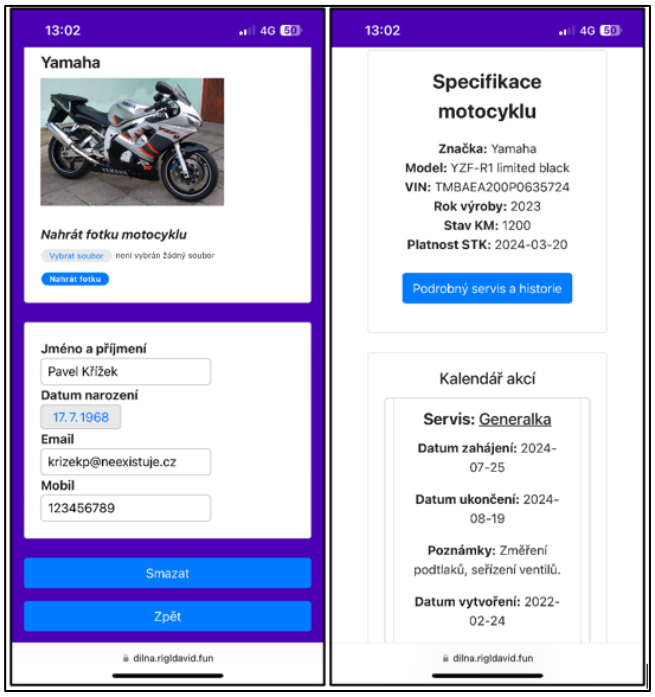



*Obrázek 21 - Mechanik - úvodní stránka (vlastní zpracování)*

Úvodní stránka po přihlášení mechanika zobrazuje automaticky všechny motocykly, které se v systému nacházejí. Je zde umožněno si motocykl vybrat z nabídky či si jej vyhledat podle názvu jednotlivých modelů zadaných při přidávání motocyklu do systému. Po kliknutí na vybraný motocykl se mechanik přesměruje na podrobnosti o motocyklu, kde je umožněno provádět další potřebné úkony. Dále se zde nachází tlačítka s odpovídajícími funkcemi pro přidání motocyklu a uživatele, plánovače událostí, seznamu kolegů, seznamu klientů, podpora, nastavení, které zabezpečuje změnu hesla, tlačítko „Domov“, které vždy odkazuje na tuto stránku a tlačítko pro odhlášení ze systému.




*Obrázek 22 - Mechanik - specifikace motocyklu - PC (vlastní zpracování)*



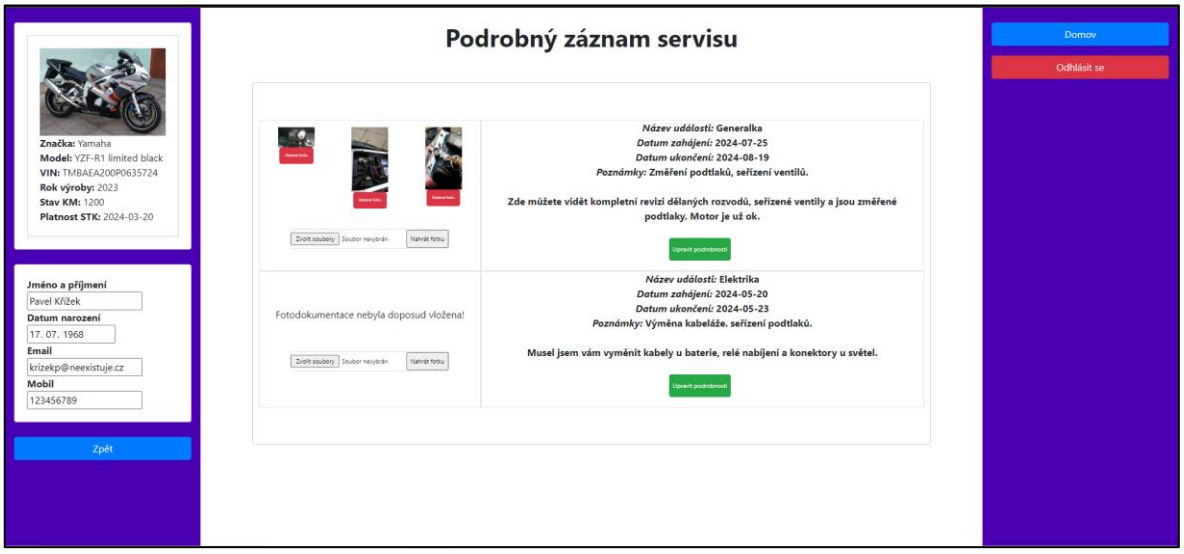

*Obrázek 23 - Mechanik - specifikace motocyklu - telefon (vlastní zpracování)*

Po výběru motocyklu je mechanik přesměrován na stránku obsahující podrobné informace o motocyklu. Dále je zde výpis z databáze o jeho majiteli. Nedílnou součástí je ve spodní částí obrazovky výpis z databáze z kalendáře akcí, které jsou naplánované, nebo byly naplánované na tento motocykl. Stránka obsahuje možnost nahrát fotku motocyklu či smazat vybraný motocykl a případně přesměrovat na stránku „Podrobný servis a historie“.

Níže je přiložena část kódu z této stránky na dotaz pro výpis informací o uživateli a daném motocyklu po zakliknutí tlačítka motocyklu. Ve chvíli použití tlačítka motocyklu se uloží do url stránky VIN motocyklu a ID uživatele, díky kterým může kód níže z url tyto údaje získat a následně provést výpis potřebných údajů o uživateli a motocyklu do stránky.

```
try {
    // Vytvoření spojení pomocí PDO
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // Nastavení režimu chybování na výjimky
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // Získání potřebných údajů pomocí idOsobni_udaje a VIN
    if (isset($_GET['idOsobni_udaje'], $_GET['VIN'])) {
        $idOsobni_udaje = $_GET['idOsobni_udaje'];
        $VIN = $_GET['VIN'];
        // Dotaz na osobní údaje
        $sql_person = "SELECT * FROM Osobni_udaje WHERE idOsobni_udaje =
:idOsobni_udaje";
```

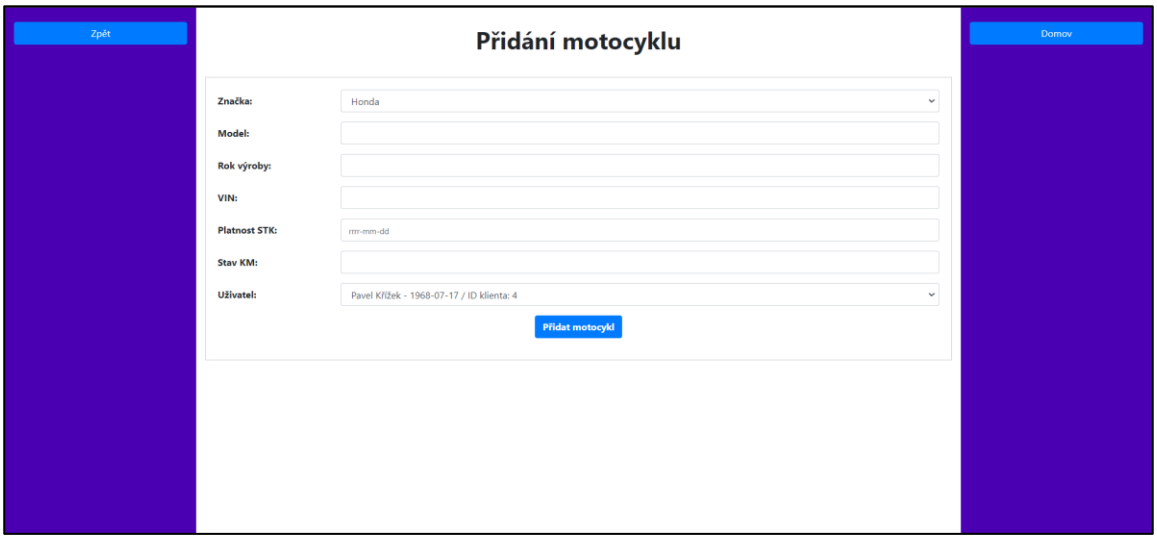
```
        $stmt_person = $conn->prepare($sql_person);
        $stmt_person->bindParam(':idOsobni_udaje', $idOsobni_udaje);
        $stmt_person->execute();
        $row_person = $stmt_person->fetch(PDO::FETCH_ASSOC);
        // Dotaz na motocykl
        $sql_motorcycle = "SELECT * FROM Motocykl WHERE vin = :vin";
        $stmt_motorcycle = $conn->prepare($sql_motorcycle);
        $stmt_motorcycle->bindParam(':vin', $VIN);
        $stmt_motorcycle->execute();
        $row_motorcycle = $stmt_motorcycle->fetch(PDO::FETCH_ASSOC);
        // Dotaz na značku motocyklu
        $sql_brand = "SELECT nazev FROM Znacka INNER JOIN Motocykl ON
Motocykl.Znacka_idZnacka = Znacka.idZnacka WHERE Motocykl.vin = :vin";
        $stmt_brand = $conn->prepare($sql_brand);
        $stmt_brand->bindParam(':vin', $VIN);
        $stmt_brand->execute();
        $row_brand = $stmt_brand->fetch(PDO::FETCH_ASSOC);
    } else {
        // Chyba - chybějící parametry v URL
        exit("Chybějící parametry v URL.");
    }
} catch (PDOException $e) {
    echo "Chyba při připojení k databázi: " . $e->getMessage();
} finally {
    // Zavření spojení s databází
    $conn = null;
}
```




*Obrázek 24 - Mechanik - podrobný záznam servisu (vlastní zpracování)*

Na této stránce jsou mechanikem nahrávány fotografie z průběhu servisu možnost přidat poznámku s podrobnostmi o daném zákroku, zároveň je zde umožněno fotky odebrat.

Jednotlivé události jsou vypsány z databáze a k nim je umožněno tyto části doplnit, není však nutné fotodokumentaci či podrobné informace o zákroku vypisovat.




*Obrázek 25 - Mechanik - přidání motocyklu (vlastní zpracování)*

Stránka „Přidání motocyklu“ umožňuje přidat mechanikovi nový motocykl do databáze. Podstatným krokem před přidáním motocyklu je však již založený účet klienta. Po vyplnění všech potřebných polí je motocykl automaticky zapsán do databáze a přiřazen k odpovídajícímu klientovi. Jednotlivá pole jsou ošetřena proti nesprávnému formátu zadání hodnot, tedy v datumu lze psát pouze čísla a to v odpovídajícím formátu, zadání VIN je omezeno na 17 znaků, značku motocyklu lze pouze vybrat z databáze, model motocyklu je bez omezení.

Stránka „Přidání uživatele“ umožňuje mechanikovi přidat nového uživatele do systému. Opět jsou zde nastavena omezení pro vhodný formát zápisu informací do databáze. Na tuto stránku je vázán skript, který automaticky podle jména a příjmení vygeneruje přihlašovací jméno uživateli a to ve tvaru celé příjmení a první písmenko ze jména. Skript ošetří příjmení odebráním diakritiky a velkých písmen a v případě shody jmen a příjmení přiřadí k přihlašovacímu jménu číslovku. Dále je skript optimalizován tak, že v případě shody jména, příjmení a telefonu či emailu neumožní přidat nový účet.

Níže je názorná ukázka části PHP skriptu, kde prvně dochází ke kontrole z databáze, zda daný uživatel již není registrován, dále je zde část skriptu pro generování uživatelského loginu na základě uživatelského jména a příjmení a skript pro generování náhodného hesla.

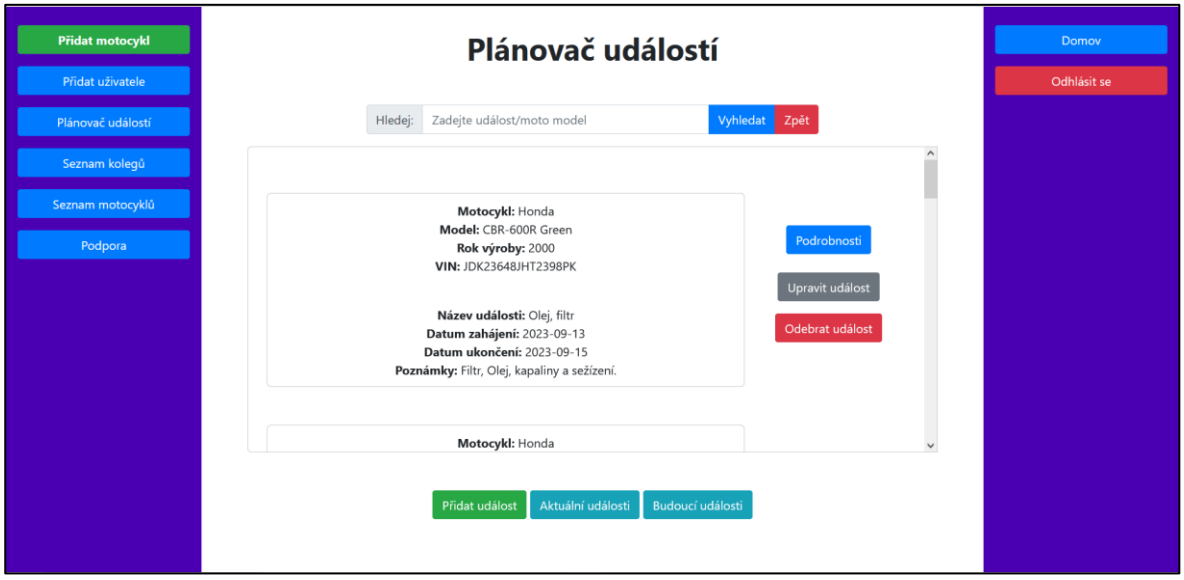
Skript pro generování loginu je konstruován tak, aby si vzal z formuláře jméno a příjmení. Z příjmení odebere diakritiku a zmenší všechny velká písmena na malá, následně začne konstruovat login v pořadí příjmení a první písmenko ze jména, následně dojde na kontrolu existence loginu a v případě detekce dojde k přiřazení čísla k loginu:

```
// Funkce pro kontrolu existence uživatele v databázi
function userExists($conn, $jmeno, $prijmeni, $datumNarozeni, $telefon,
$email) {
$sql = "SELECT * FROM Osobni_udaje WHERE jmeno = :jmeno AND prijmeni =
:prijmeni AND datumNarozeni = :datumNarozeni AND telefon = :telefon AND
email = :email";
    $stmt = $conn->prepare($sql);
    $stmt->bindParam(':jmeno', $jmeno);
    $stmt->bindParam(':prijmeni', $prijmeni);
    $stmt->bindParam(':datumNarozeni', $datumNarozeni);
    $stmt->bindParam(':telefon', $telefon);
    $stmt->bindParam(':email', $email);
    $stmt->execute();
    return $stmt->rowCount() > 0;
}
// Funkce pro generování loginu
function generateRandomLogin($jmeno, $prijmeni, $conn) {

    // Převod příjmení na malá písmena
    $prijmeniLowerCase = mb_strtolower($prijmeni, 'UTF-8');
    // Odstranění diakritiky z příjmení
    $prijmeniWithoutDiacritics = removeAccents($prijmeniLowerCase);
    // Přidání prvního písmene jména
    $login = $prijmeniWithoutDiacritics . substr(strtolower($jmeno), 0, 1);
    // Pokud login existuje, přidá číslo
    $suffix = '';
    while (loginExists($conn, $login . $suffix)) {
        if ($suffix === '') {
            $suffix = 1;
        } else {
            $suffix++;
        }
$login = $prijmeniWithoutDiacritics . substr(strtolower($jmeno), 0, 1) .
$suffix; } return $login;}
function removeAccents($string) {
    $accents = array('á', 'č', 'ď', 'é', 'ě', 'í', 'ň', 'ó', 'ř', 'š', 'ť',
'ú', 'ů', 'ý', 'ž', 'Á', 'Č', 'Ď', 'É', 'Ě', 'Í', 'Ň', 'Ó', 'Ř', 'Š', 'Ť',
'Ú', 'Ů', 'Ý', 'Ž');
    $without = array('a', 'c', 'd', 'e', 'e', 'i', 'n', 'o', 'r', 's', 't',
'u', 'u', 'y', 'z', 'A', 'C', 'D', 'E', 'E', 'I', 'N', 'O', 'R', 'S', 'T',
'U', 'U', 'Y', 'Z');
    return str_replace($accents, $without, $string);
}
```

Skript pro generování hesla je nastaven na 8 znaků, vygenerované heslo musí obsahovat minimálně jedno velké písmeno, jedno malé písmeno, jedno číslo a jeden speciální znak. Následně dochází k náhodnému doplnění zbylých znaků.

```
// Funkce pro generování náhodného hesla
function generateRandomPassword($length = 8) {
$characters =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!%&*()-_+=';
$genhesl = '';
// Ošetření pro obsah: minimálně jednoho velké, malé písmeno, čísla a
speciálního znaku
$genhesl .= substr(str_shuffle('abcdefghijklmnopqrstuvwxyz'), 0, 1);
$genhesl .= substr(str_shuffle('ABCDEFGHIJKLMNOPQRSTUVWXYZ'), 0, 1);
$genhesl .= substr(str_shuffle('0123456789'), 0, 1);
$genhesl .= substr(str_shuffle('!%&*()-_+='), 0, 1);
    // Náhodné doplnění zbylých znaků
    for ($i = 4; $i < $length; $i++) {
        $genhesl .= $characters[rand(0, strlen($characters) - 1)];
    }
return $genhesl;
}
```




*Obrázek 26 - Mechanik – plánovač událostí (vlastní zpracování)*

Tato stránka pojmenovaná „Plánovač událostí“ vychází z tabulky „Udalost“ a je jedním z hlavních pracovních prostorů mechanika. Je zde umožněno dohledat jakoukoliv událost a to podle názvu události či podle modelu motocyklu. Je zde umožněno jednotlivé události upravovat v podobě změn datumu zahájení a ukončení, názvu události a poznámky k události a je zde umožněno se i podívat, kdo je majitelem motocyklu a na jaký motocykl je tedy událost vystavena.

Zároveň stránka obsahuje tlačítko na přidání události, které po rozkliknutí otevře přímo ve stránce nový formulář pro vyplnění potřebných údajů a výpisu motocyklů. Dále se zde nachází zobrazení aktuálních událostí (probíhajících), které jsou porovnávány vůči aktuálnímu datumu a tlačítko budoucí události, které vypíše všechny události, které jsou plánovány od aktuálního datumu dále.

Níže je názorná ukázka PHP funkce v podobě vytvořeného skriptu pro smazání události v databázi v případě, že je zavolán pomocí tlačítka „Odebrat událost“. Po stisknutí tlačítka se objeví varovná hláška, která po odkliknutí způsobí, že se na stránce událostí uloží do proměnné “$row[‘idUdalost‘]“ id události, u které bylo tlačítko použito a předá jej do příkazu níže ke smazání. Volaný skript si načte id události a provede mazání přímo v databázi podle id.

**Tlačítko**

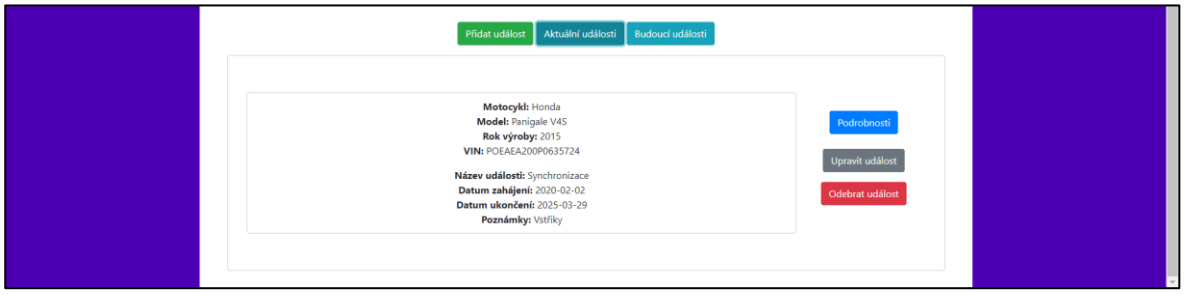
```
echo "<br><button type='button' class='btn btn-danger mb-3'
onclick='deleteEvent(" . $row['idUdalost'] . ")'>Odebrat událost</button>";
```

**Skript pro volání funkce**

```
<script>
  // Smazání události
    function deleteEvent(idUdalost) {
        if (confirm("Opravdu chcete smazat tuto událost?")) {
            fetch('S_Smazat_Udalost.php?idUdalost=' + idUdalost, {
                method: 'DELETE'
            }).then(response => {
                if (response.ok) {
                    location.reload();
                } else {
                    alert('Chyba při mazání události');
                }
            }).catch(error => alert('Chyba při komunikaci se serverem'));
        }
    }
</script>
```

**Skript „S_Smazat_Udalost.php“ pro mazání události**

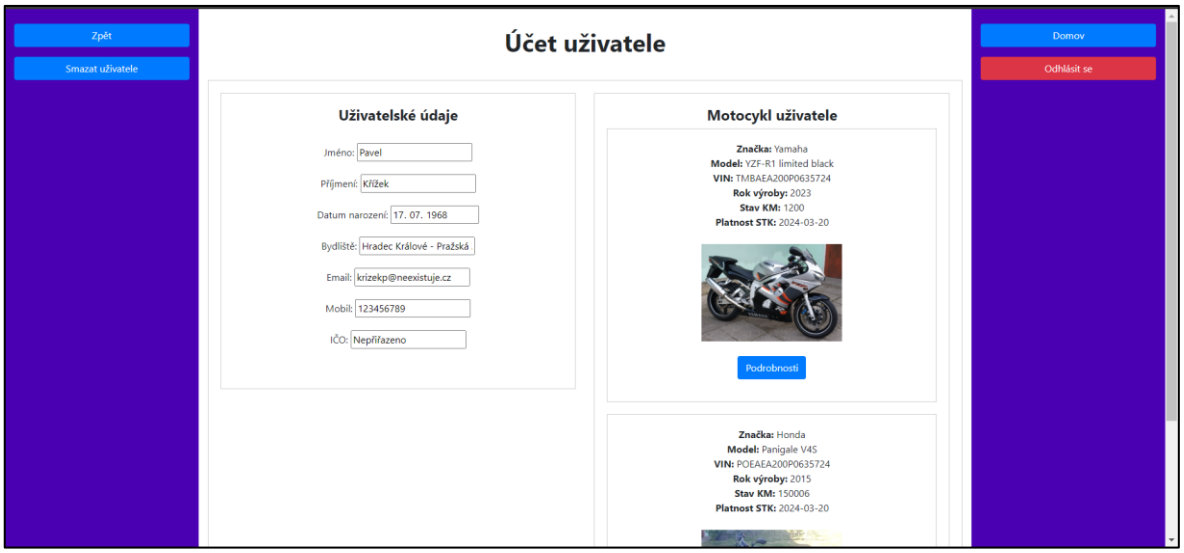
```php
<?php
// Připojení na databázi
$servername = *****;
$username = '*****';
$password = '*****';
$dbname = 'moto_dilna_db';
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // SQL dotaz na smazání události
    $stmt = $conn->prepare("DELETE FROM Udalost WHERE idUdalost =
:idUdalost");
    $stmt->bindParam(':idUdalost', $_GET['idUdalost']);
    $stmt->execute();
    echo "Událost byla úspěšně smazána";
} catch (PDOException $e) {
    echo "Chyba při mazání události: " . $e->getMessage();
}
// Uzavření připojení na databázi
$conn = null;
?>
```




*Obrázek 27 - Mechanik – plánovač událostí - aktuální události (vlastní zpracování)*

Tato část stránky události je vyvolána pomocí tlačítka „Aktuální události“ a má za úkol ukázat mechanikovi aktuálně probíhající servisní činnosti, které porovnává vůči aktuálnímu datumu. Dále je zde tlačítko „Budoucí události“, které má za úkol ukázat do budoucna naplánované servisní úkony na motocyklech od aktuálního datumu dále.

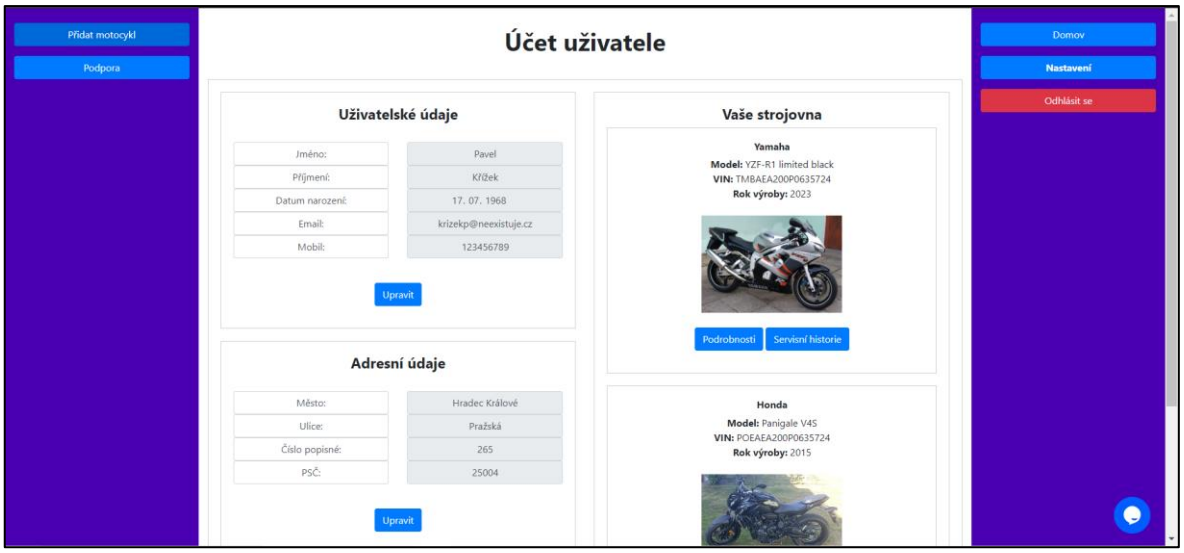
Dále se na hlavní stránce mechanika nachází tlačítko „Seznam klientů“, které umožňuje vyhledat uživatele pomocí jména či příjmení, nebo pomocí výběrového pole, které vypisuje uživatele z databáze. Po vyhledání se zobrazí proklikávací okno se základními údaji o klientovi a jeho přiřazených motocyklech.




*Obrázek 28 - Mechanik - podrobnosti o uživateli (vlastní zpracování)*

Účet uživatele je vyvolán proklikem vyhledaného uživatele. Obsahuje veškeré uživatelské údaje, seznam jeho motocyklů a možnosti přesunutí na jejich specifikaci. Dále je zde umožněno smazat daného uživatele. Toto smazání provede odebrání i všech událostí a motocyklů daného uživatele.

## Webové stránky - část uživatel



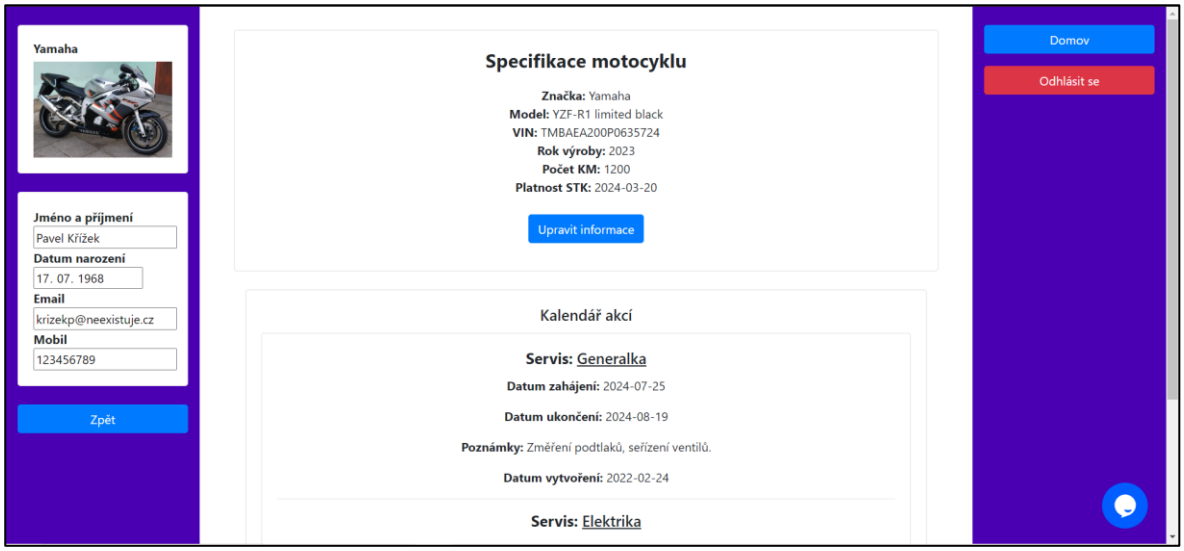

*Obrázek 29 - Uživatel – úvodní stránka (vlastní zpracování)*

Po úspěšné autentizaci uživatele je uživatel přesměrován na tuto stránku s jeho motocykly a informacemi. Uživatel si zde může změnit pomocí tlačítka „Upravit“ pod uživatelskými údaji email a mobil. Dále je zde opět funkce pro změnu hesla. Nedílnou součástí je možnost si přidat další vlastní motocykl. V sekci „Vaše strojovna“ jsou

vyobrazeny všechny motocykly uživatele s možností přesunu na stránku s podrobnostmi o motocyklu či na stránku s celkovou servisní historií motocyklu od zadání do systému.


*Obrázek 30 - Uživatel - podrobný záznam servisu (vlastní zpracování)*

Na tuto stránku je uživatel přesměrován pod tlačítkem „Servisní historie“ a jsou zde záznamy o provedení zákroků mechanikem na daném motocyklu. Uživateli je umožněno si fotky zobrazit v plném rozlišení.




*Obrázek 31 - Uživatel - specifikace motocyklu (vlastní zpracování)*

Na tuto stránku je uživatel přesměrován pomocí tlačítka „Podrobnosti“ ze stránky „Účet uživatele“. Uživatel zde vidí všechny zaznamenané informace o motocyklu v systému s možností editace počtu kilometrů a platností „STK“. Dále je zde výpis kalendáře všech

akcí týkajících se daného motocyklu a v neposlední řadě je zde implementovaný chat pro komunikaci s mechanikem v podobě miniaturní ikony.


*Obrázek 32 - Otevření chatu s mechanikem na stránce (vlastní zpracování)*

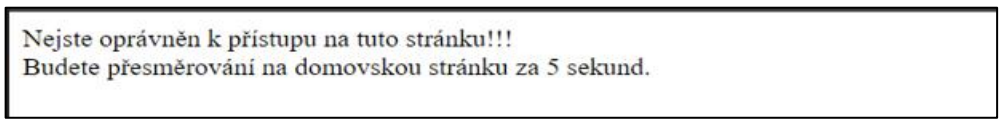
Toto okno je vyvoláno kliknutím na ikonku v chatu v pravém dolním rohu obrazovky. Aplikace Tawk.to si automaticky ukládá konverzace, díky tomu je možné si zpětně zobrazit všechny zprávy, které byly psány v chatu.

### 4.4.2 Další vlastnosti a funkce systému.

V této části budou popsány další vlastnosti a funkce v systému, které nemusí být z předešlých částí patrné. Jsou však velmi důležité pro správné fungování systému.

**Kontrola přihlášení a odpovídající práva**

Podstatnou součástí všech jednotlivých stránek je PHP skript, který na každé stránce kontroluje, zda je uživatel přihlášen a zda odpovídají jeho práva přiřazená k účtu k přístupu na tuto stránku. V případě pokusu vstupu na jakoukoliv stránku bez odpovídajících práv či bez přihlášení bude zobrazena bezpečnostní hláška a dojde na přesměrování na úvodní stránku.




*Obrázek 33 - Vyrovná hláška (vlastní zpracování)*

**Heshování hesel**

Dále je v systému aplikována funkce „heshování“ hesel. Toto heshování je zabezpečeno pomocí funkce Bcrypt, která je aplikována přímo v jazyce PHP. Tato funkce je podstatnou součástí ze stránky bezpečnosti každého systému. V backendové části systému je připraven skript na případné zaheshování hesla v případě manuální editace hesla u uživatele a je aplikován i ve skriptu pro vytvoření nového zákazníka či při editaci vlastního hesla uživatele a mechanika.

**Vytváření účtu nového uživatele**

V případě tvorby uživatele ze strany mechanika je vždy automaticky nastaveno oprávnění přístupu na úroveň „3“. Tato úroveň je uživatelská a umožňuje přístup pouze do uživatelského prostředí. Přidání účtu pro mechanika lze pouze ze strany administrátora přímo na serveru, a to s úrovní „2“. Úroveň „1“ je určena pro administrátora a je připravena pro případ rozšíření systému.

**Automatické odesílání pomocí emailu**

V systému je zahrnuta funkce automatického odesílání emailu. Tato funkce je aplikována na úvodní stránce pro odeslání formuláře při žádosti o vytvoření účtu neregistrovaného uživatele. Dále je funkce aplikována při vytváření účtu na straně mechanika pro uživatele. Ve chvíli odeslání formuláře s vyplněnými údaji skript automaticky z formuláře zjistí email uživatele a odešle na tento email vygenerované heslo a login.

Posledním místem, které využívá tento skript je automatické odesílání upozornění v případě končící platnosti technické kontroly vozidla uživatele. Odpovídající skript každý den kontroluje databázi (datum platnosti STK) a v případě zjištění datumu nacházejícího se 30 dní a 5 dní před skončením platnosti tomuto uživateli odešle odpovídající email s informacemi o konci platnosti kontroly.

Níže je názorná ukázka skriptu odesílajícího přihlašovací údaje pomocí PHPMaileru, nejprve dojde na spuštění funkce pro odesílání emailů, následně se nastaví informace pro SMTP server, do proměnné „$to“ se zapíše email uživatele z formuláře, do proměnné „$subject“ se nastaví nadpis emailu a do proměnné „$message“ se nastaví předdefinovaná zpráva, která v sobě nese vygenerované proměnné loginu a hesla pro uživatele.

```php
// Funkce pro odeslání e-mailu
function sendEmail($to, $subject, $message) {
$mail = new PHPMailer(true);
try {
// Nastavení serveru SMTP
$mail->isSMTP();
$mail->Host = 'smtp.seznam.cz';
$mail->SMTPAuth = true;
$mail->Username = 'infomoto@rigldavid.fun';
$mail->Password = '*********';
$mail->SMTPSecure = 'ssl';
$mail->Port = 465;
$mail->CharSet = 'UTF-8';
$mail->setFrom('infomoto@rigldavid.fun', 'Moto dilna DP');
$mail->addAddress($to);
// Obsah emailu
$mail->isHTML(true);
$mail->Subject = $subject;
$mail->Body = $message;
// Odeslání emailu
$mail->send();
return true;
} catch (Exception $e) {
error_log($mail->ErrorInfo);
echo "Chyba při odesílání emailu: " . $mail->ErrorInfo;
return false;
    }
}
```

**Python skript pro odsílání informativního emailu o konci platnosti STK**

Na serveru je vytvořen skript přiložený níže, který porovnává aktuální datum s datumy platností STK u všech motocyklů v databázi, kdy v případě, že je platnost končící 30 dní a 5 dní před odpovídajícím datumem pošle skript informující email uživateli s informací o konci platnosti STK. Skript si z databáze zjistí email, jméno a příjmení uživatele, model motocyklu, vin a potřebnou platnost STK a s těmito údaji informuje majitele motocyklu. Skript je nastaven na automatické spuštění každý den v 17 hodin a 18 minut večer UTC.

Spuštění tohoto skirptu je provedeno pomocí cronu, který je definován již Dockerfile na serveru a byl nastaven při vytváření kontejnerů.

RUN (crontab -l 2>/dev/null; echo "18 17 * * * python3 /script.py >> /var/log/cron.log 2>&1") | crontab –

```
# Nastavení pro SMTP server
smtp_server = 'smtp.seznam.cz'
smtp_port = 465  # Port pro SSL
smtp_user = 'infomoto@rigldavid.fun'
smtp_password = '*******'

# Připojení k databázi
conn = mysql.connector.connect(**db_config)
cursor = conn.cursor()

# SQL dotaz
sql_query = '''
SELECT Motocykl.platnostSTK, Motocykl.model, Motocykl.vin, Osobni_udaje.jmeno, Osobni_udaje.prijmeni, Osobni_udaje.email
FROM Motocykl
INNER JOIN Klient ON Motocykl.Klient_idklient = Klient.idKlient
INNER JOIN Osobni_udaje ON Klient.Osobni_udaje_idOsobni_udaje = Osobni_udaje.idOsobni_udaje;
'''

cursor.execute(sql_query)

# Funkce pro odeslání e-mailu
def send_email(recipient, subject, body):
    message = MIMEMultipart()
    message['From'] = smtp_user
    message['To'] = recipient
    message['Subject'] = subject
    message.attach(MIMEText(body, 'plain'))

    with smtplib.SMTP_SSL(smtp_server, smtp_port) as server:
        server.login(smtp_user, smtp_password)
        server.sendmail(smtp_user, recipient, message.as_string())

# Zpracování výsledků dotazu a odeslání e-mailů
for row in cursor:
    platnost_stk, model, vin, jmeno, prijmeni, email = row
    dny_do_konce_stk = (platnost_stk - datetime.now().date()).days

    if dny_do_konce_stk in [30, 5]:
        subject = f'Upozornění na končící platnost STK pro {model}'
        body = f'Vážený {jmeno} {prijmeni}, u Vaší motorky {model} s VIN {vin} končí za {dny_do_konce_stk} dnů platnost technické prohlídky.'
        send_email(email, subject, body)

# Uzavření připojení k databázi
cursor.close()
conn.close()
```

*Obrázek 34 - Python skript - kontrola platnosti STK (vlastní zpracování)*

### reCAPTCHA

Poslední funkcí, kterou je vhodné zmínit je implementovaná reCAPTCHA pro zamezení tvorby nežádoucích uživatelů pomocí skriptů či automatických botů. Tato funkce je pro veřejně dostupné stránky s možností registrace do systému nevyhnutelně důležitá. Je implementována přímo na úvodní stránce systému.

## 4.5 Testování funkcionalit

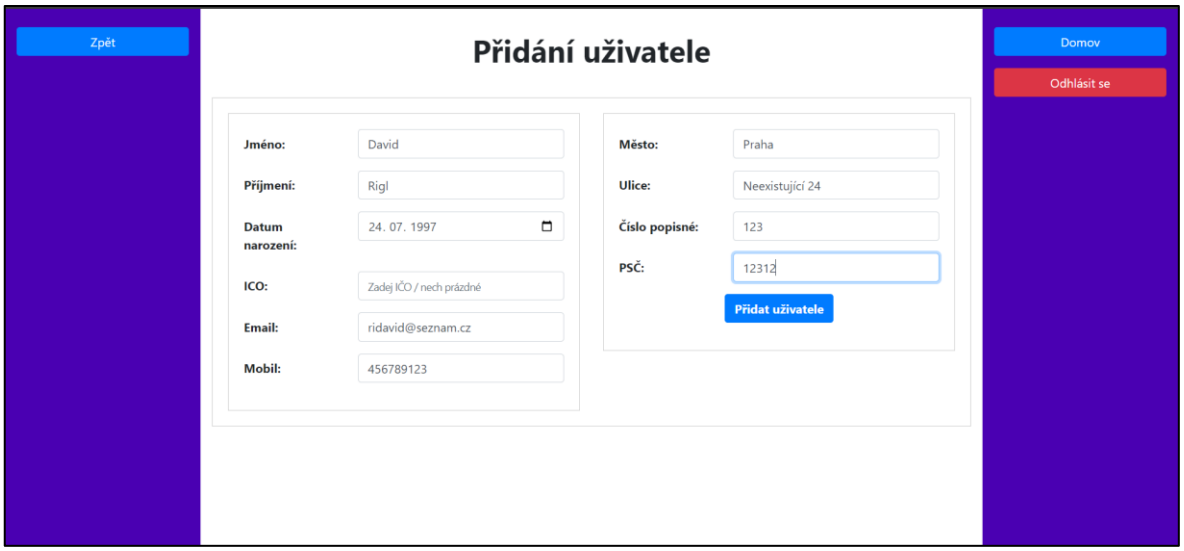
Testování jednotlivých funkcionalit systému bylo prováděno především při tvorbě jednotlivých částí systému a v případě jakékoliv chyby docházelo ihned k jejich opravě. Na závěr po kompletním zprovozněním testovací verze však byl proveden několikrát opakovaný celkový test, který probíhal ve 2 základních částech.

První částí byla autentizace mechanika a změna hesla. Následovalo otestování žádosti o vytvoření účtu ze strany uživatele a přidání uživatele do systému s otestováním odesílání emailu s přihlašovacími údaji. Dále se k uživateli přiřadilo několik motocyklů, k nim nahrály úvodní fotky a vytvořily události a podrobné záznamy s fotkami. Následně bylo testováno vyhledávání v kalendáři, mezi uživateli a motocykly. Na konci této části bylo otestováno mazání událostí, motocyklů a uživatele. Tyto kroky jsou patrné z předešlých obrázků u

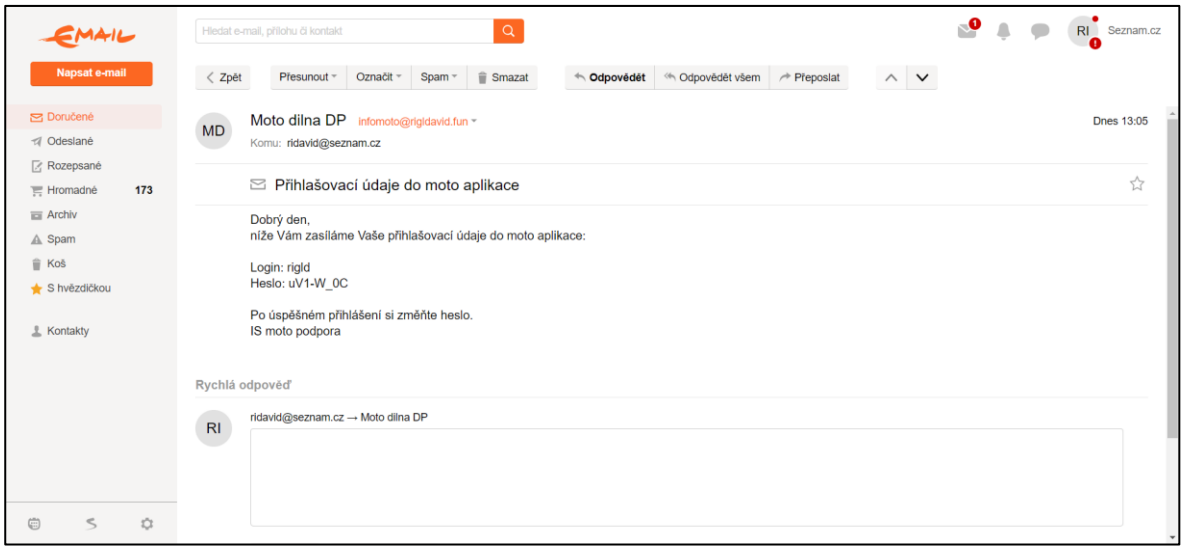
ukázek funkčnosti systému. Všechny jednotlivé příkazy upravující databázi byly kontrolovány v databázi přímo na serveru, zda daný krok ihned provedly.

V druhé části bylo provedeno testování uživatelské strany. Po přidání uživatele mechanikem bylo otestováno po přijetí přihlašovacích údajů přihlášení do systému, následným krokem byla změna hesla, editace osobních údajů v podobě telefonního čísla a emailu a byla provedena změna adresních údajů. Následovalo přidání nového motocyklu do systému, změny údajů u motocyklu v podobě platnosti STK a najetých kilometrů. Posledním krokem bylo sledování v systému u motocyklu, zda přibývají události a podrobný záznam servisu ze strany mechanika u vybraného motocyklu uživatele.

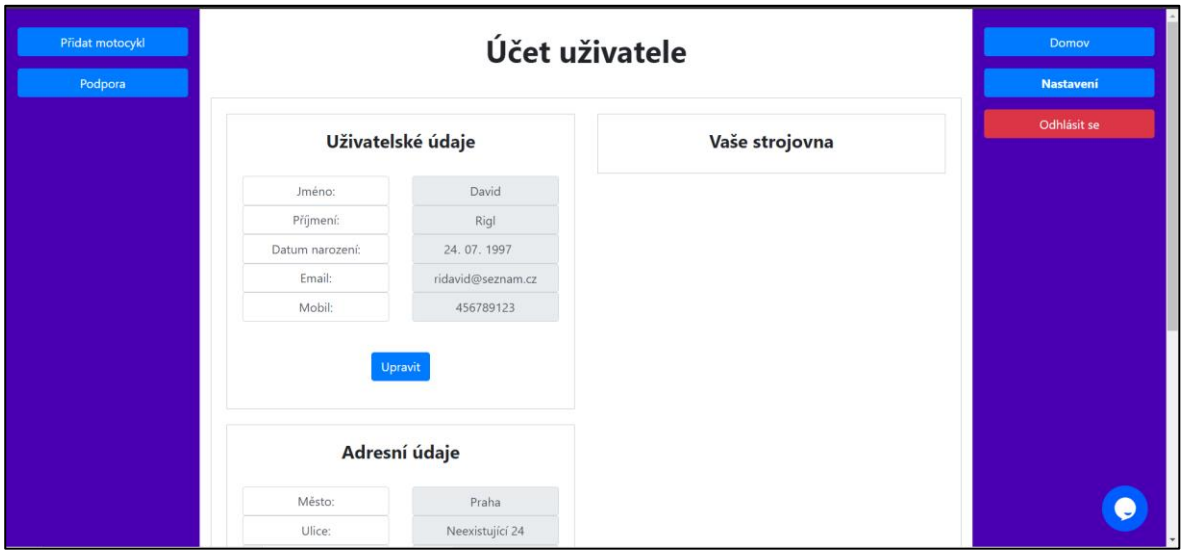
**Ukázka části testovacího postupu:** Přidání nového uživatele a jeho motocyklu.
**Postup:** V účtu mechanika se přejde na stránku přidání uživatele, kde se vyplní potřebné údaje o uživateli a uživatel se po stisknutí tlačítka přidá do databáze. Následně dorazí uživateli přihlašovací údaje v emailu vygenerované automaticky systémem. Po úspěšné autentizaci došlo na přidání motocyklu u uživatele. Po vytvoření motocyklu se v účtu mechanika přiřadila fotografie k novému motocyklu (uživatel nemá oprávnění fotografie přidávat). Dále bylo upraveno datum platnosti technické kontroly pro otestování python skriptu pro odeslání informativního emailu o konci platnosti STK. Na závěr byla otestována funkce pro změnu hesla. Níže jsou přiloženy fotografie z průběhu testování.



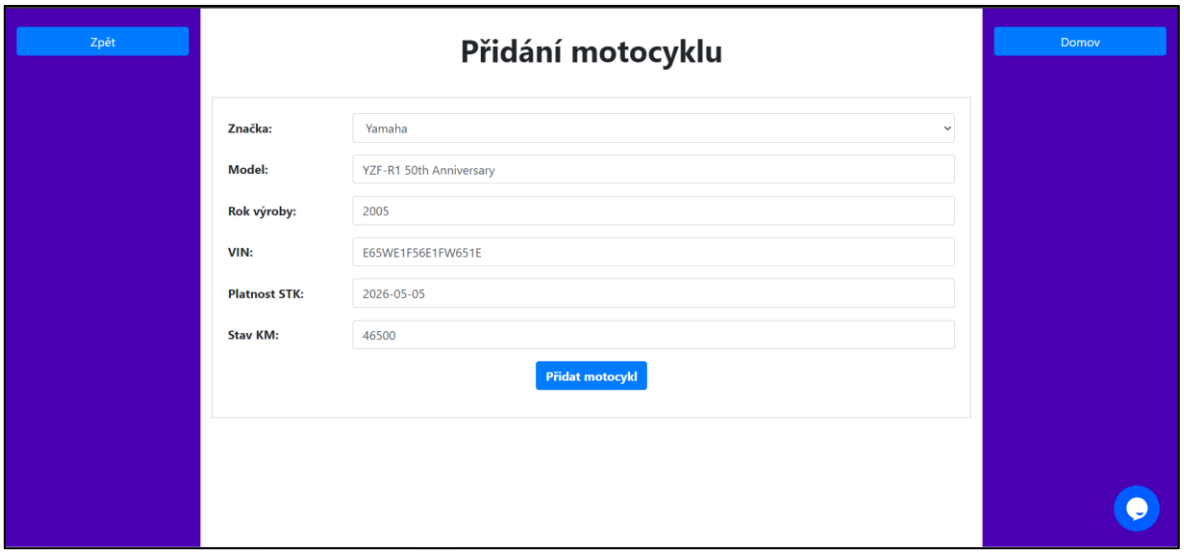

*Obrázek 35 - Testování - přidání uživatele (vlastní zpracování)*



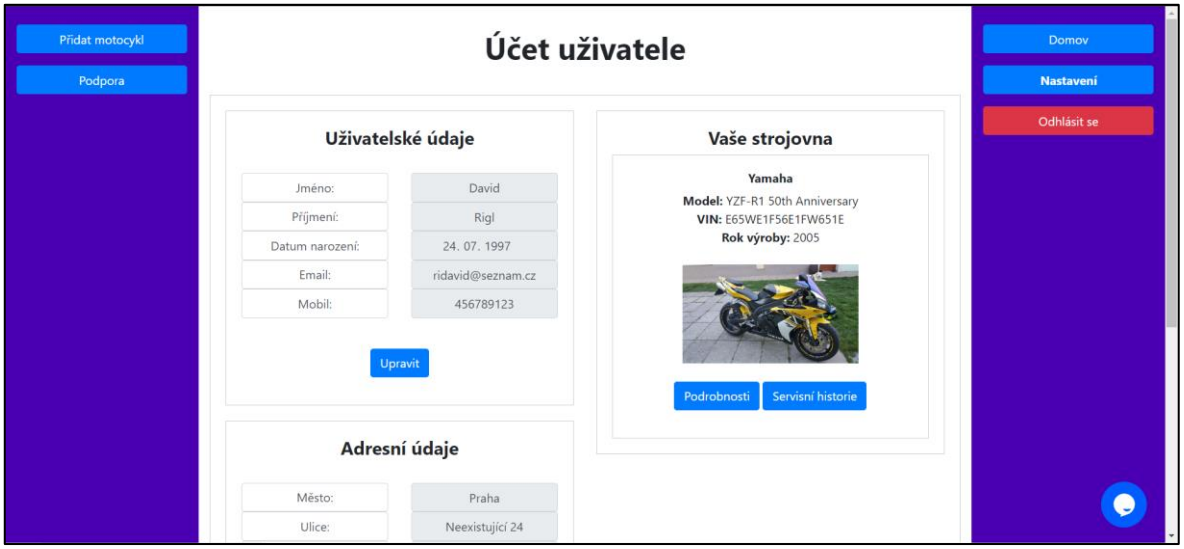

*Obrázek 36 - Testování - doručení přihlašovacích údajů (vlastní zpracování)*



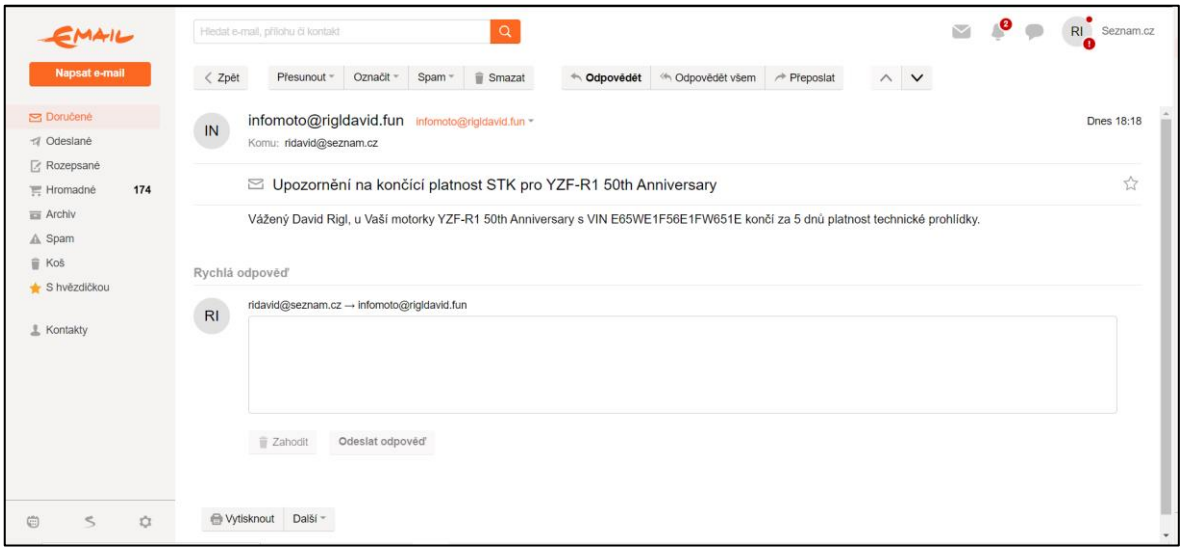

*Obrázek 37 - Testování - autentizace uživatele (vlastní zpracování)*




*Obrázek 38 - Testování - přidání motocyklu uživatele (vlastní zpracování)*




*Obrázek 39 - Testování - úspěšné přidání motocyklu a fotografie (vlastní zpracování)*

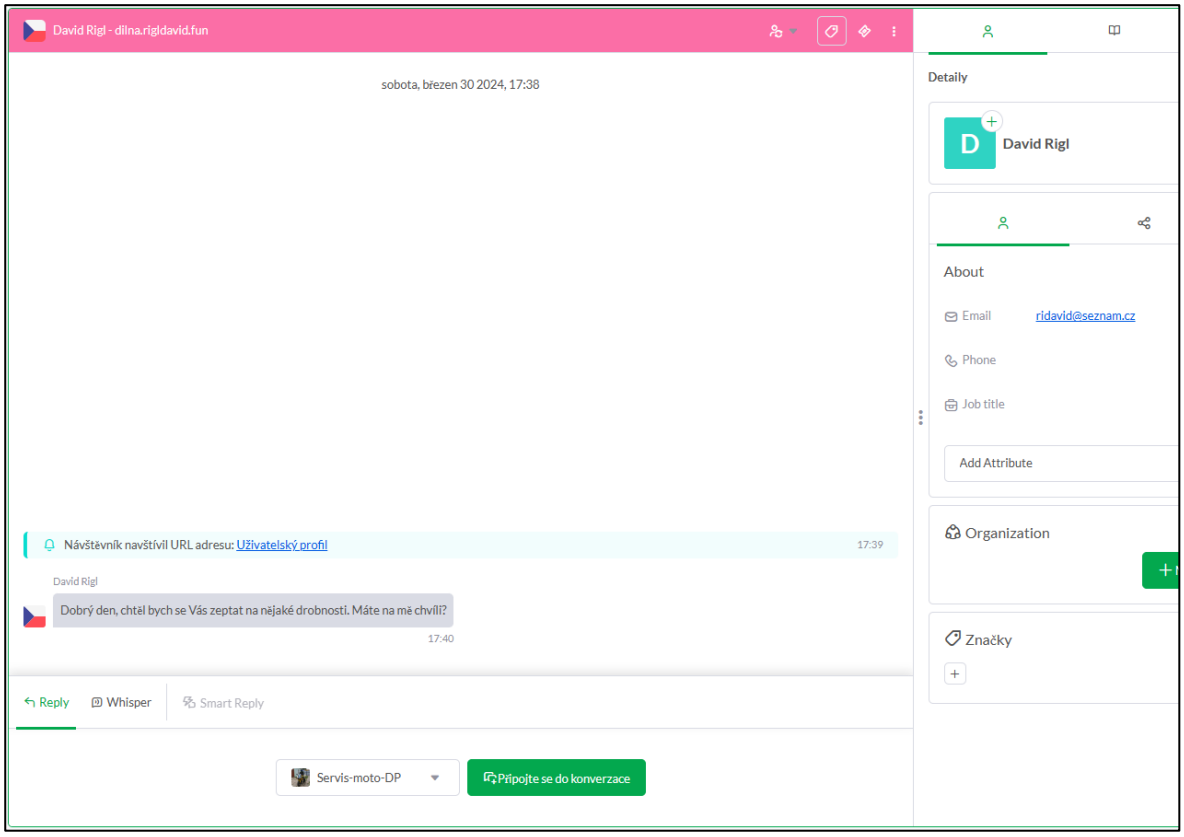



*Obrázek 40 - Testování - informativní email o konci platnosti STK na motocyklu (vlastní zpracování)*

**Ukázka otestování funkčnosti chatu**

Pro otestování funkčnosti chatu v systému a získání odpovídajících údajů o uživateli byla z uživatelského účtu „David Rigl“ odeslána zpráva v chatu mechanikovi. Přiložený obrázek níže ukazuje příchozí zprávu do aplikace chatu na straně mechanika.

V horní levé části obrázku lze vidět uživatele, od kterého byla zpráva odeslána, tento údaj je i na pravé straně obrázku a níže se nachází získaný email uživatele.




*Obrázek 41 - Otestování funkčnosti chatu v systému (vlastní zpracování)*

## 4.6 Chyby v systému

Největší potíže při tvorbě systému způsobovala možnost smazání uživatele a všech jeho součástí ze systému pomocí jednoho tlačítka „Smazat uživatele“. Toto tlačítko má za úkol smazat uživatele, všechny jeho motocykly a jim přiřazené události z databáze. Problém však byl v propojení tabulek v databázi a provázání id cizích klíčů mezi tabulkami, kdy v případě, že došlo k mazání uživatele se vždy automaticky smazalo uživatelské ID, bez kterého však nebylo možné mazat další tabulky. Mazat další tabulky dříve než tabulku s ID nebylo kvůli cizím klíčům možné provést. Řešením nakonec bylo uložení přidružených údajů uživatele z daných tabulek do proměnných a na jejich základě již bylo možné uživatele kompletně ze systému vymazat i po smazání uživatelského ID.

Další nepříjemnosti způsoboval skript, který má za úkol generovat novému uživateli heslo, toto heslo uložit do proměnné a zapsat jej v této podobě do emailu na automatické odeslání. Následně zde byl skript na heshování hesla a zápis do databáze. Po příchodu emailu však heslo pro přístup nefungovalo. Tato chyba byla velmi dlouho dobu hledána, až bylo zjištěno, že se příkaz pro hesh hesla: „$heslo_hash = password_hash($genhesl, PASSWORD_DEFAULT);“ nachází na stránce ve skriptu dvakrát s tím, že u druhého hashování bylo provedeno volání na příkaz pro generování nového hesla, tím docházelo k rozdílnému zápisu hesla do databáze a odeslání uživateli.

# 5 Výsledky a diskuse

## 5.1 Výsledky práce

Výsledkem práce je naprogramovaný informační systém obsahující všechny požadované funkce stanovené na tento systém. Cíl práce byl splněn v celém rozsahu od komunikace uživatele s mechanikem, po správu událostí, podrobné záznamy jednotlivých servisních úkonů u motocyklů a možnost nahrávání fotografií s poznámkovým prostorem, dále funkční plánování událostí s různými možnostmi zobrazení a vyhledáváním, různé výpisy z databáze uživatelů a motocyklů, flexibilita systému z důvodu využívání na mobilním telefonu, jednoduchost používání a omezená oprávnění pro změny ze strany uživatelů. Všechny naimplementované funkce a skripty fungují na základě provedeného testu na všechny funkcionality v systému v podobě přidání uživatele a motocyklů, změny parametrů, skriptů pro odesílání emilů apod. Architektura systému byla navržena s ohledem k dlouhodobé udržitelnosti a škálovatelnosti, což umožňuje systém dále upravovat a rozšiřovat dle potřeb klienta. Systém je připraven pro úpravu designu dle požadavků zákazníka a následnou implementaci do vybraného prostředí.

## 5.2 Diskuse

K diskusi je vhodné zmínit použití jiného hardwaru pro systém. Pravděpodobnost, že by chtěl zájemce systém provozovat pomocí Raspberry PI zařízení je velmi malá. Je tedy pravděpodobné, že se tento systém bude muset integrovat na vybranou cloud computingovou variantu či případně na fyzický server v daném podniku. Tato varianta by byla mnohem rychlejší a efektivnější, jelikož cloudcomputing nabízí nespočet funkcí, možností zabezpečení a analytických nástrojů, které by tvorbu a vývoj tohoto systému pozitivně podpořily. Díky celkovému návrhu a postupu, který se v této práci nachází však implementace na jiné prostředí nebude nijak zvlášť problémová. Raspberry PI bylo vybráno pro svou jednoduchou správu, finanční nenáročnost a zkušenosti s používáním této varianty.

Dále by bylo vhodné namísto vlastní tvorby vzhledu stránek v tomto typu systému vybrat vhodný webový framework, který nabízí předpřipravené komponenty a

knihovny. Tento krok by výrazně urychlil a zefektivnil vývoj systému. Tyto frameworky velmi často obsahují zabudované bezpečnostní mechanismy, které zvyšují bezpečnost nejen webu, ale celkového systému.

# 6 Závěr

Podstatnými kroky pro realizaci tohoto projektu bylo provést analýzu požadavků, provést návrh všech součástí systému a možností jeho použití a následně implementovat navržené části systému od zprovoznění základních funkcionalit potřebných pro systém funkční v internetu, implementaci navržené databáze do struktury systému, návrh webových stránek a jejich implementace do systému až po implementaci skriptů a provázání SQL databáze pro splnění všech funkcí očekávaných od tohoto systému. Velmi důležitým krokem bylo také následné testování všech funkcionalit, které v nepatrném množství ukázalo na některé chyby v systému, které byly následně opraveny. Výsledkem je funkční systém, který splňuje všechny stanové požadavky a funkcionality.

V teoretické části byly popsány všechny důležité součásti týkající se systému i možnosti realizace tohoto systému na jiná prostředí a rozebrány důvody, výhody a nevýhody jejich použití. Stručně byla probrána bezpečnostní stránka tohoto typu systémů a byly rozebrány a popsány nástroje sloužící pro návrh a vývoj systému a důvody jejich použití.

V praktické části došlo na stanovení požadavků potencionálním zájemcem o tento systém, následně byla provedena podrobná analýza těchto požadavků a poté následoval postupný návrh systému. Dále došlo na postupnou realizaci systému, zprovoznění funkcí, které byly tvořeny z velké části pomocí jazyka PHP a některé specifické funkce byly realizovány pomocí programovacího jazyka JavaScript či Python. Po implementaci návrhu byly opraveny vzniklé nedostatky. Následovalo vytvoření testovacích účtů, motocyklů a fiktivních servisních úkonů pro otestování všech funkcionalit. Systém je nyní připraven pro úpravy designu dle požadavků zákazníka a případné zprovoznění na vybrané infrastruktuře.

V projektu se nachází veškerá dokumentace z průběžné implementace systému a z testování některých funkcionalit. Realizaci projektu podpořily z velké části zkušenosti ze studijních předmětů tohoto oboru.

# 7 Seznam použitých zdrojů


1. *Jaký je rozdíl mezi webem a webovou aplikací?* [online]. 2023, 01.03.2023 [cit. 2023-10-02]. Dostupné z: https://www.coreapp.cz/blog/jaky-je-rozdil-mezi-webem-a-webovou-aplikaci
2. *Typy webových aplikací, kterou zvolit pro vlastní vývoj?* [online]. KOĎOUSKOVÁ, Barbora. 2023, 09.04.2023 [cit. 2023-10-02]. Dostupné z: https://www.rascasone.com/cs/blog/typy-webovych-aplikaci
3. *Informace o webových aplikacích* [online]. 2021, 19.05.2021 [cit. 2023-10-02]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html
4. *Rozdíl mezi fyzickým a virtuálním serverem* [online]. 2019, 16.07.2019 [cit. 2023-10-02]. Dostupné z: https://www.stable.cz/support/blog-news/rozdil-mezi-fyzickym-a-virtualnim-serverem/
5. *Cloud vs. vlastní server vs. virtuální server: Jaké řešení se vaší firmě nejvíc vyplatí* [online]. 2021, 22.01.2021 [cit. 2023-10-02]. Dostupné z: https://www.webstep.net/cloud-vs-vlastni-server-vs-virtualni-server-jake-reseni-se-vasi-firme-nejvic-vyplati/
6. *Virtuální vs. fyzický server* [online]. 2023, 25.04.2023 [cit. 2023-10-02]. Dostupné z: https://www.levnapc.cz/virtualni-vs-fyzicky-server.html
7. *Co je to databáze?* [online]. [cit. 2023-10-02]. Dostupné z: https://tech-lib.eu/tema/188/co-je-to-databaze
8. *Základní informace o databázích* [online]. [cit. 2023-10-02]. Dostupné z: https://support.microsoft.com/cs-cz/office/z%C3%A1kladn%C3%AD-informace-o-datab%C3%A1z%C3%ADch-a849ac16-07c7-4a31-9948-3c8c94a7c204
9. *8 Major Advantages of Using MySQL* [online]. VYAS, Kashyap. 2023, 03.02.2023 [cit. 2023-10-02]. Dostupné z: https://www.datamation.com/storage/8-major-advantages-of-using-mysql/
10. *MySQL advantages and disadvantages* [online]. [cit. 2023-10-02]. Dostupné z: https://www.w3schools.blog/mysql-advantages-disadvantages
11. *What is PostgreSQL?* [online]. [cit. 2023-10-02]. Dostupné z: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/
12. *What is PostgreSQL?* [online]. [cit. 2023-10-02]. Dostupné z: https://www.ibm.com/topics/postgresql
13. *What is PostgreSQL?* [online]. [cit. 2023-10-02]. Dostupné z: https://aws.amazon.com/rds/postgresql/what-is-postgresql/
14. *Lekce 1 - Úvod do HTML* [online]. HARTINGER ČÁPKA, David. [cit. 2023-10-02]. Dostupné z: https://www.itnetwork.cz/html-css/webove-stranky/jak-psat-moderni-web-html-tutorial-uvod-do-html
15. *HTML pro začátečníky aneb jak začít psát web* [online]. KOĎOUSKOVÁ, Barbora. 2021, 15.07.2021 [cit. 2023-10-02]. Dostupné z: https://www.rascasone.com/cs/blog/html-pro-zacatecniky-jak-psat-web
16. *What is PHP? The PHP Programming Language Meaning Explained* [online]. KOLADE, Chris. 2021, 30.08.2021 [cit. 2023-10-02]. Dostupné z: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/
17. *What is PHP?* [online]. [cit. 2023-10-02]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php
18. *What is HTTPS?* [online]. [cit. 2023-10-02]. Dostupné z: https://www.cloudflare.com/learning/ssl/what-is-https/

19. *Protokol HTTPS* [online]. [cit. 2023-10-02]. Dostupné z: https://www.ssls.cz/https.html
20. *What Is MariaDB?* [online]. JALLI, Artturi. 2022, 21.12.2022 [cit. 2023-10-02]. Dostupné z: https://builtin.com/data-science/mariadb
21. *MariaDB* [online]. [cit. 2023-10-02]. Dostupné z: https://viptrust.com/technologie/databaze/mariadb
22. *What is Docker?* [online]. [cit. 2023-10-02]. Dostupné z: https://aws.amazon.com/docker/
23. *Portainer* [online]. [cit. 2023-10-02]. Dostupné z: https://products.containerize.com/cs/devops/portainer/
24. *Operating system images* [online]. [cit. 2023-10-02]. Dostupné z: https://www.raspberrypi.com/software/operating-systems/
25. *Welcome to Raspbian* [online]. [cit. 2023-10-02]. Dostupné z: https://www.raspbian.org/
26. *NGINX vs Apache – Choosing the Best Web Server in 2023* [online]. , Richard B. & Aris B. 2023, 26.09.2023 [cit. 2023-10-02]. Dostupné z: https://www.hostinger.com/tutorials/nginx-vs-apache-what-to-use/
27. *Nginx* [online]. [cit. 2023-10-02]. Dostupné z: https://nginx.org/en/
28. *MySQL Workbench* [online]. [cit. 2023-10-02]. Dostupné z: https://www.javatpoint.com/mysql-workbench
29. *What is Visual Studio Code? Microsoft's extensible code editor* [online]. HELLER, Martin. 2022, 08.07.2022 [cit. 2023-10-02]. Dostupné z: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html
30. *Why did we build Visual Studio Code?* [online]. [cit. 2023-10-02]. Dostupné z: https://code.visualstudio.com/docs/editor/whyvscode
31. *About Node.js®* [online]. [cit. 2023-10-02]. Dostupné z: https://nodejs.org/en/about
32. *What is Node.js: A Comprehensive Guide* [online]. [cit. 2023-10-02]. Dostupné z: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs
33. *Thomas Erl, Zaigham Mahmood. Cloud Computing: Concepts, Technology & Architecture (The Pearson Service Technology Series from Thomas Erl) 1st Edition. s.l. : Pearson, 2013. ISBN 9780133387520.*
34. *Co je cloud computing? [online]. [cit. 2023-11-12]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-cloud-computing*
35. *IaaS, PaaS a SaaS aneb V čem se liší služby „as a Service"* [online]. MŮČKA, Jan. 2021, 29.09.2021 [cit. 2023-11-12]. Dostupné z: https://www.master.cz/blog/iaas-paas-a-saas-aneb-v-cem-se-lisi-sluzby-as-a-service/
36. *What is Web Application Security?* [online]. [cit. 2023-11-12]. Dostupné z: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-application-security-appsec/what-is-web-application-security/
37. *What is Web Application Security?* [online]. [cit. 2023-11-12]. Dostupné z: https://www.cloudflare.com/learning/security/what-is-web-application-security/
38. *Web Application Security* [online]. [cit. 2023-11-12]. Dostupné z: https://www.synopsys.com/glossary/what-is-web-application-security.html
39. *Bootstrap Get Started* [online]. [cit. 2024-11-12]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

40. *9 Bootstrap Alternatives for 2023 and Beyond (Reviewed)* [online]. 2022.10.11 [cit. 2024-11-12]. Dostupné z: https://www.creative-tim.com/blog/educational-tech/bootstrap-alternatives/
41. *JAVASCRIPT PRO ZAČÁTEČNÍKY: CO TO JE A JAK FUNGUJE* [online]. 2022, 28.01.2022 [cit. 2024-11-12]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky
42. *What is JavaScript?* [online]. [cit. 2024-11-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

# 8 Seznam obrázků, tabulek, grafů a zkratek

## 8.1 Seznam obrázků

## 8.2 Seznam tabulek



## 8.3 Seznam použitých zkratek

VPS – Virtuální privátní server

OS – Operační systém

HW - Hardware

SW - Software

SŘBD - Systém řízení báze dat

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

HTML – Hypertext Markup Language

IS – Informační systém

PHP – Hypertext Preprocessor

CSS – Cascading Style Sheets

UML – Unified Modeling Language

SSL – Secure Socket Layer

TLS – Transport Layer Security

GB - Gigabyte

DoS – Denial of Service

# Přílohy

Příloha 1 – Kompletní zdrojový kód tohoto systému včetně textových souborů ze serveru je nahrán do zip souboru v příloze práce pod názvem: Zdrojovy_kod.zip.