

**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**



## **Diplomová práce**

**Vývoj Laravel aplikace pro firmu zabývající se čištěním  
interiérů**

**Bc. Martin Dub**

**© 2024 ČZU v Praze**



# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

## ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Martin Dub

Informatika

Název práce

**Vývoj Laravel aplikace pro firmu zabývající se čištěním interiérů**

Název anglicky

**Laravel Application Development for Interior Cleaning Company**

---

### Cíle práce

Cílem teoretické části práce je popis teorie a problematiky vývoje softwaru, dále představení open-source frameworku Laravel a jeho komponent, programovacího jazyka PHP včetně historie vývoje a databáze MySQL. Teoretická část práce zanalyzuje a porovná konkurenci představeného frameworku, zejména frameworky Nette a Symfony, také jednotlivé varianty databázových a šablonovacích systémů. Dále teoretická část představení jednotlivé use-case Laravel komponenty se šablonovacími systémy Nette Latte a Laravel Blade.

Cílem praktické části je návrh, implementace a otestování webové stránky včetně nastavení administrace pro klienta. V rámci vývoje bude využit verzovací systém Git a aplikace GitHub.

### Metodika

Metodika práce bude obsahovat analýzu požadavků a možných řešení na základě literární rešerše. Řešení stránky bude responzivní s pomocí frameworku Tailwind CSS. Hosting bude zajišťovat služba Forpsi a během vývoje bude využit Laravel Artisan CLI.

Kromě dynamického kontaktního formuláře bude k dispozici na výsledné stránce i výběr varianty čištění nebo ukázka referencí společnosti. Webová stránka bude plně řízena klientem prostřednictvím zmíněné administrace.

### **Doporučený rozsah práce**

60–80 stran

### **Klíčová slova**

webová aplikace, čištění interiérů, nabídka služeb, PHP, Laravel, SQL, GitHub, VCS, Composer

---

### **Doporučené zdroje informací**

BOTROS, Silvia a Jeremy TINLEY. High Performance MySQL. 4. edice. Sebastopol, CA: O'Reilly Media, 2021. ISBN 9781492080510.

FORD, Neal, Mark RICHARDS, Pramod SADALAGE a Zhamak DEGHANI. Software architecture: The hard parts. Sebastopol, CA: O'Reilly Media, 2021. ISBN 9781492086895.

STAUFFER, Matt. Laravel: up & running: a framework for building modern PHP apps. Second edition. Sebastopol, CA: O'Reilly, [2019]. ISBN 9781492041214.

---

### **Předběžný termín obhajoby**

2023/24 LS – PEF

### **Vedoucí práce**

Ing. Dana Vynikarová, Ph.D.

### **Garantující pracoviště**

Katedra informačního inženýrství

---

Elektronicky schváleno dne 4. 9. 2023

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

---

Elektronicky schváleno dne 3. 11. 2023

**doc. Ing. Tomáš Šubrt, Ph.D.**

Děkan

V Praze dne 11. 03. 2024

## **Čestné prohlášení**

Prohlašuji, že svou diplomovou práci „Vývoj Laravel aplikace pro firmu zabývající se čištěním interiérů“ 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 31. března 2024

---

## **Poděkování**

Rád bych touto cestou poděkoval Ing. Daně Vynikarové, Ph.D. za odborné vedení a profesionální pomoc při psaní této práce, za konzultace i cenné rady. Chtěl bych také poděkovat rodině za podporu, bráchovi Štěpovi za nápad, přítelkyni Aničce za trpělivost a kolegovi Mildovi za věcné připomínky ke kódu a za motivaci psát lepší a přehlednější kód.

# Vývoj Laravel aplikace pro firmu zabývající se čištěním interiérů

## Abstrakt

Diplomová práce analyzuje možnosti řešení při tvorbě webových aplikací, jaký je vhodný framework a na co si dát pozor při výběru webhostingu. Dále popisuje překážky při vývoji, nepsaná pravidla programování a časté chyby programátorů.

V praktické části je shrnut vývoj aplikace doplněný o výstřižky kódů, výsledek v podobě obrázku i analýza návštěvnosti. Výstupem diplomové práce je hotová webová aplikace, rozdělená na sekci pro návštěvníky a administrátory, kteří mají v uživatelsky přívětivém prostředí nastaveno několik nástrojů, kterým lze webovou stránku spravovat bez přítomnosti programátora.

**Klíčová slova:** webová aplikace, čištění interiérů, nabídka služeb, PHP, Laravel, SQL, GitHub, VCS, Composer

# **Laravel Application Development for Interior Cleaning Company**

## **Abstract**

The master thesis analyzes the possibilities of solutions for creating web applications, what is the appropriate framework and what to look out for when choosing a webhosting. It also describes casual obstacles in development, unwritten rules of programming and common programmers' mistakes. The practical part summarizes the development of the application, along with code snippets, final result of website in the image form and a Google traffic analysis. The output of the thesis is a finished web application, divided into a section for visitors and administrators, whose have several additional tools set up in a user-friendly environment to manage the website without the presence of the programmer.

**Keywords:** Web application, interior cleaning, service offering, PHP, Laravel, SQL, GitHub, VCS, Composer



# Obsah

<b>1 Úvod</b> .....	<b>- 1 -</b>
<b>2 Cíl práce a metodika</b> .....	<b>- 2 -</b>
2.1 Cíl práce .....	- 2 -
2.2 Metodika.....	- 2 -
<b>3 Teoretická východiska</b> .....	<b>- 3 -</b>
3.1 Popis technologií .....	- 3 -
3.1.1 PHP .....	- 5 -
3.1.2 Laravel framework.....	- 10 -
3.1.2.1 Ekosystém.....	- 10 -
3.1.2.2 Výuka.....	- 12 -
3.1.2.3 Instalace.....	- 12 -
3.1.2.4 Schéma.....	- 12 -
3.1.2.5 Laravel Artisan .....	- 14 -
3.1.2.6 Hello world v Laravelu.....	- 14 -
3.1.2.7 Výhody používání frameworku.....	- 15 -
3.1.2.8 Nevýhody používání frameworku .....	- 17 -
3.1.3 Databázové systémy.....	- 17 -
3.1.3.1 Základní orientace tabulkou .....	- 19 -
3.1.3.2 Datové typy.....	- 20 -
3.1.4 Verzování.....	- 28 -
3.1.4.1 GitHub Actions.....	- 30 -
3.1.5 Hosting.....	- 31 -
3.1.5.1 Freehosting .....	- 31 -
3.1.5.2 Doména a hosting .....	- 31 -
3.1.5.3 Datový limit.....	- 32 -
3.1.5.4 Mail.....	- 32 -
3.1.5.5 Operační systém a software.....	- 32 -
3.1.5.6 Podpora skriptování.....	- 33 -
3.1.5.7 Logování.....	- 33 -
3.1.5.8 Certifikáty.....	- 33 -
3.1.5.9 Soubor .htaccess .....	- 34 -
3.1.5.10 Ochrana proti (D)DOS útokům .....	- 34 -

3.2	Konkurence .....	- 34 -
3.2.1	PHP .....	- 35 -
3.2.2	Laravel .....	- 35 -
3.2.3	Databáze.....	- 36 -
3.2.4	Git a GitHub.....	- 37 -
3.2.5	Hosting.....	- 37 -
<b>4</b>	<b>Vlastní práce.....</b>	<b>- 38 -</b>
4.1	Návrhové vzory a osvědčené postupy.....	- 40 -
4.2	Implementace aplikace.....	- 41 -
4.2.1	Barevné schéma .....	- 45 -
4.2.2	Obecná nastavení .....	- 45 -
4.2.3	Domovská stránka.....	- 46 -
4.2.4	Nastavení proměnných prostředí .....	- 49 -
4.2.5	Sekce s administrací.....	- 49 -
4.2.5.1	Stránka s přehledem.....	- 53 -
4.2.5.2	Zákazníci .....	- 54 -
4.2.5.3	Recenze.....	- 56 -
4.2.5.4	Faktury.....	- 57 -
4.2.5.5	Vouchery .....	- 59 -
4.2.5.6	Ostatní.....	- 60 -
4.3	Testování aplikace.....	- 65 -
<b>5</b>	<b>Závěr.....</b>	<b>- 68 -</b>
<b>6</b>	<b>Seznam použitých zdrojů .....</b>	<b>- 69 -</b>
<b>7</b>	<b>Přílohy.....</b>	<b>- 72 -</b>

## Seznam obrázků

Obrázek 1	- Logo frameworku Laravel.....	- 10 -
Obrázek 2	- Příkaz MySQL, vytvoření databáze .....	- 24 -
Obrázek 3	- Příkaz MySQL, používání databáze.....	- 24 -
Obrázek 4	- Příkaz MySQL, založení uživatele.....	- 24 -
Obrázek 5	- Příkaz MySQL, vytvoření tabulky .....	- 25 -
Obrázek 6	- Příkaz MySQL, zobrazení struktury tabulky .....	- 25 -
Obrázek 7	- Příkaz MySQL, úprava tabulky.....	- 26 -
Obrázek 8	- Příkaz MySQL, kontrola struktury.....	- 26 -
Obrázek 9	- Příkaz MySQL, odstranění sloupce.....	- 27 -
Obrázek 10	- Příkaz MySQL, vložení dat do tabulky.....	- 27 -
Obrázek 11	- Příkaz MySQL, získání dat z tabulky .....	- 28 -
Obrázek 12	- Schéma GitHub workflow.....	- 30 -

Obrázek 13 - Představení doménového systému .....	- 31 -
Obrázek 14 - Wireframe domovské stránky .....	- 40 -
Obrázek 15 - Wireframe admin panelu .....	- 40 -
Obrázek 16 - CLI instalátor Laravelu .....	- 42 -
Obrázek 17 - Dokončení CLI instalace .....	- 43 -
Obrázek 18 - Úvodní stránka Laravelu verze 11. ....	- 44 -
Obrázek 19 - Instalace závislosti přes Composer. ....	- 44 -
Obrázek 20 - Barevné schéma webu .....	- 45 -
Obrázek 21 - Výstup dat z Google Analytics. ....	- 49 -
Obrázek 22 - Admin panel, verze 1. ....	- 50 -
Obrázek 23 - Admin panel, recenze, verze 1. ....	- 50 -
Obrázek 24 - Admin panel, faktury, verze 1 .....	- 50 -
Obrázek 25 - Admin panel, vouchery, verze 1. ....	- 50 -
Obrázek 26 - Staré logo. ....	- 51 -
Obrázek 27 - Nové logo .....	- 51 -
Obrázek 28 - Admin panel, verze 2. ....	- 52 -
Obrázek 29 - Admin stránka, přehled .....	- 54 -
Obrázek 30 - Admin stránka, přehled, mobilní zobrazení. ....	- 54 -
Obrázek 31 - Admin stránka, zákazníci .....	- 56 -
Obrázek 32 - Admin stránka, zákazníci, mobilní přehled .....	- 56 -
Obrázek 33 - Admin stránka, recenze .....	- 57 -
Obrázek 34 - Admin stránka, recenze, mobilní zobrazení .....	- 57 -
Obrázek 35 - Admin stránka, faktury .....	- 59 -
Obrázek 36 - Admin stránka, faktury, mobilní zobrazení .....	- 59 -
Obrázek 37 - Admin stránka, vouchery. ....	- 60 -
Obrázek 38 - Admin stránka, vouchery, mobilní zobrazení. ....	- 60 -
Obrázek 39 - Šablona e-mailu s novou poptávkou. ....	- 63 -
Obrázek 40 - Verze webu A, počet zobrazení a počet událostí. ....	- 66 -
Obrázek 41 - Verze webu B, počet zobrazení a počet událostí .....	- 66 -
Obrázek 42 - Verze webu A, zapojení uživatele .....	- 66 -
Obrázek 43 - Verze webu B, zapojení uživatele .....	- 67 -

## Seznam objektů

Figure 1 – Výpis aktuálního data s HTML .....	- 6 -
Figure 2 – Výpis aktuálního data s PHP .....	- 6 -
Figure 3 - Jednořádkový komentář v PHP .....	- 6 -
Figure 4 - Víceřádkový komentář v PHP .....	- 7 -
Figure 5 - Ukázka PHP kódu .....	- 7 -
Figure 6 – Datové typy integer, string a array v PHP .....	- 7 -
Figure 7 - Dynamické přiřazení proměnné do PHP .....	- 7 -
Figure 8 - PHP třída a objekt .....	- 8 -
Figure 9 - Přiřazení objektu do proměnné v PHP .....	- 8 -
Figure 10 - Výpis funkce message() .....	- 8 -
Figure 11 - Příkaz fopen() .....	- 9 -
Figure 12 - Demonstrace dávkovače v PHP .....	- 9 -
Figure 13 – Příkaz na instalaci frameworku Laravel .....	- 12 -
Figure 14 – Příkaz na spuštění Laravel serveru .....	- 12 -

Figure 15 – Struktura frameworku Laravel.....	13 -
Figure 16 – Zobrazení Hello World v Laravelu, část Route.....	14 -
Figure 17 – Zobrazení Hello World v Laravelu, část Controller.....	14 -
Figure 18 – Zobrazení Hello World v Laravelu, část View.....	14 -
Figure 19 - Výpis dat z databáze, nativní PHP .....	16 -
Figure 20 - Výpis dat z databáze, Laravel framework.....	16 -
Figure 21 - Získání dat z pole, nativní PHP .....	16 -
Figure 22 - Získání dat z pole, Laravel framework.....	16 -
Figure 23 - Přepočítání velikosti souboru do standardní formy, nativní PHP .....	16 -
Figure 24 - Přepočítání velikosti souboru do standardní formy, Laravel framework.....	17 -
Figure 25 - Převod znaku do ASCII, nativní PHP .....	17 -
Figure 26 - Převod do znaku ASCII, Laravel framework.....	17 -
Figure 27 – Jednoduchý SQL příkaz.....	18 -
Figure 28 – DQL příkaz.....	18 -
Figure 29 – Objektové i skalární filtrování v DQL.....	19 -
Figure 30 - Datový typ Enumerations v PHP. ....	46 -
Figure 31 - Vytvoření controlleru pro domovskou stránku. ....	46 -
Figure 32 - Import Controlleru do souboru s cestami.....	46 -
Figure 33 - Cesta obsluhující domovskou stránku.....	46 -
Figure 34 - Metoda index() na zobrazení domovské stránky.....	47 -
Figure 35 - Pomocník config() pro výpis členů týmu.....	47 -
Figure 36 - Nastavení varianty po kliknutí na variantu ceníku.....	48 -
Figure 37 - Značka <input> s nastavením varianty. ....	48 -
Figure 38 - Framework Vue.js a předání varianty do formuláře. ....	48 -
Figure 39 - Google tag pro sledování návštěvnosti webu.....	48 -
Figure 40 - Cron spouštěný každou minutu.....	52 -
Figure 41 - Cron spouštěný jednou ročně 1. ledna ve 13 hodin 37 minut. ....	52 -
Figure 42 - Cron spouštěný v půlnoci a v poledne 1. den každý druhý měsíc v roce.....	53 -
Figure 43 - Vytvoření controlleru pro práci se zdrojem. ....	54 -
Figure 44 - Definice controlleru pro práci se zdrojem.....	55 -
Figure 45 - Metoda index() pro zobrazení zákazníků.....	55 -
Figure 46 - Laravel helper pro nastavení HTTP hlavičky. ....	55 -
Figure 47 - Kód knihovny Guzzle pro import recenzí z platformy Google Mapy. ....	57 -
Figure 48 - Generování příjmového dokladu.....	58 -
Figure 49 - Generování hashů pro jednotlivé vouchery.....	59 -
Figure 50 - Získávání aktuálních voucherů.....	60 -
Figure 51 - Validace na straně klienta. ....	60 -
Figure 52 - Validace na straně serveru.....	61 -
Figure 53 - Mailable třída na posílání e-mailu s novým zákazníkem.....	62 -
Figure 54 - Posílání e-mailu v případě výskytu chyby nebo výjimky.....	64 -

## Seznam tabulek

Tabulka 1 – Binární a desítková soustava.....	4 -
Tabulka 2 – HTTP požadavky a jejich využití .....	5 -
Tabulka 3 – Logo, název a popis několika komponent Laravel ekosystému .....	11 -
Tabulka 4 – Jednoduchá SQL tabulka. ....	18 -
Tabulka 5 - Základní příkazy MySQL.....	20 -
Tabulka 6 - Číselné datové typy, celá čísla.....	21 -

Tabulka 7 - Číselné datové typy, reálná čísla .....	- 21 -
Tabulka 8 - Datový typ CHAR a VARCHAR .....	- 22 -
Tabulka 9 - Datový typ BINARY a VARBINARY .....	- 22 -
Tabulka 10 - Datový typ BLOB.....	- 23 -
Tabulka 11 - Datový typ TEXT .....	- 23 -
Tabulka 12 - Základní git příkazy.....	- 29 -
Tabulka 13 - Přehled využívání GitHub Actions dle předplatného .....	- 30 -
Tabulka 14 - Doporučené nastavení hostingu.....	- 32 -

## **Seznam příloh**

Příloha 1 - Domovská stránka, úvodní video.....	- 72 -
Příloha 2 - Domovská stránka, služby. ....	- 72 -
Příloha 3 - Domovská stránka, sekce o nás.....	- 73 -
Příloha 4 - Domovská stránka, ceník. ....	- 73 -
Příloha 5 - Domovská stránka, recenze.....	- 74 -
Příloha 6 - Domovská stránka, tým.....	- 74 -
Příloha 7 - Domovská stránka, kontaktní formulář.....	- 74 -

## Seznam použitých zkratk

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages
AWS	Amazon Web Services
BCMath	Binary Calculator Math
CDN	Content Delivery Network
CLI	Command Line Interface
CMS	Content Management System
CSS	Cascading Style Sheets
cURL	Client URL Request Library
DB	Database
DDOS	Distributed Denial of Service
DKIM	Domain Keys Identified Mail
DOS	Denial of Service
DQL	Doctrine Query Language
EOF	End of File
FTP	File Transfer Protocol
GA	Google Analytics
GB	Gigabyte
GPT	Generative Pre-training Transformer
GTM	Google Tag Manager
HDD	Hard Disk Drive
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secured
IDE	Integrated Development Environment
IIS	Internet Information Services
INT	Integer
IP	Internet Protocol
JSON	JavaScript Object Notation

kB	Kilobyte
LDAP	Lightweight Directory Access Protocol
MB	Megabyte
MD5	Message-Digest Algorithm
MITM	Man In The Middle
MVC	Model View Controller
NPM	Node Package Manager
ORM	Object Relation Model
OS	Operation System
PDO	PHP Data Objects
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
PSR	PHP Standards Recommendation
SLD	Sub Level Domain
SQL	Standard Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SVN	Apache Subversion
TLD	Top Level Domain
Unix	UNiplexed Information Computing System
XAMPP	Cross-Platform, Apache, MySQL, PHP, and Perl
XML	eXtensible Markup Language
YAML	Yaml Ain't Markup Language

# 1 Úvod

Internet byl vytvořen v roce 1983 za účelem předávání informací mezi uživateli. Nejinak je tomu dnes a na internetu lze najít spoustu relevantních i nerelevantních informací. Osoby, společnosti i vlády jednotlivých zemí mají na internetu svá zázemí, kde předávají informace o své osobě, podnikání nebo aktualitách. Od té doby se internet přeměnil k nepoznání a stal se s odpovídající technologií dostupným po celé planetě.

Dnes internet nabízí propojení s ostatními uživateli, sdílení různých dat i inzerci vlastního podnikání. Majitelům webových stránek dokáže správně nastavený web přivést velké množství potenciálních zákazníků. K webu lze nastavit i nástroje na hlídání návštěvnosti, a tudíž se web dá dále optimalizovat pro lepší zaujetí zákazníků a jejich delší setrvání na webu.

Společnost na čištění interiérů má sídlo v Kondraci a chce primárně rozšířit povědomí blízké veřejnosti o založení živnosti. Webová stránka je primárně rozdělena na dvě části, statickou domovskou stránku a dynamický admin panel. Pro prezentaci firmy byla zakoupena doména i hosting u společnosti FORPSI spolu s e-mailovými adresami a databází.

Práce je rozdělena na teoretickou rešerši a praktickou část, kde je popsán návrh, implementace a testování aplikace. V rešerši je popsán programovací jazyk PHP, framework Laravel a databáze MySQL, verzovací systém Git s automatizačním procesem Github Actions a princip hostingu. V poslední kapitole jsou uvedeny konkurenční technologie, které dokáží vytvořit stejný výsledek.

Praktická část představuje návrhové vzory a osvědčené postupy při programování webové aplikace. V praktické části jsou zobrazeny části kódu, snímky obrazovky jednotlivých sekcí webové stránky a popis implementace funkčních požadavků.



## **2 Cíl práce a metodika**

### **2.1 Cíl práce**

Cílem teoretické části práce je popis teorie a problematiky vývoje softwaru, dále představení open-source frameworku Laravel a jeho komponent, programovacího jazyka PHP včetně historie vývoje a databáze MySQL. Teoretická část práce zanalyzuje a porovná konkurenci představeného frameworku, zejména frameworky Nette a Symfony, také jednotlivé varianty databázových a šablonovacích systémů. Dále teoretická část představuje jednotlivé use-case Laravel komponenty se šablonovacími systémy Nette Latte a Laravel Blade.

Cílem praktické části je návrh, implementace a otestování webové stránky včetně nastavení administrace pro klienta. V rámci vývoje bude využit verzovací systém Git a repozitářová služba GitHub.

### **2.2 Metodika**

Metodika práce bude obsahovat analýzu požadavků a možných řešení na základě literární rešerše. Řešení stránky bude responzivní s pomocí frameworku Tailwind CSS. Hosting bude zajišťovat služba Forpsi a během vývoje bude využit Laravel Artisan CLI.

Kromě dynamického kontaktního formuláře bude k dispozici na výsledné stránce i výběr varianty čištění nebo ukázka referencí společnosti. Webová stránka bude plně řízena administrátorem prostřednictvím zmíněné administrace.

## 3 Teoretická východiska

V této kapitole budou postupně představeny jednotlivé komponenty potřebné k vývoji a produkci webové aplikace. Hlavní komponentou je programovací jazyk PHP, na něj navazuje framework Laravel společně s databází MySQL a asynchronní funkce zaštiťuje jazyk JavaScript a framework Vue.js. O samotný vzhled stránky se stará skriptovací jazyk HTML, který je sepsaný v rámci šablonovacího systému Nette Latte, soubor stylů CSS a frameworky Tailwind CSS pro administraci a Bootstrap pro návštěvníky webu. Aplikace bude posílat příkazy Cron – automatické zprávy administrátorovi o dalším zákazníkovi nebo měsíčním přehledu výdajů a tržeb.

### 3.1 Popis technologií

*"Počítače jsou hloupé."* (Lengstorf et al., 2022).

I když nám počítače pomáhají v každodenním životě s čím dál obsáhlejší škálou činností, veškeré funkce fungují na bázi pouhých dvou čísel: 1 a 0. Tyto dvě čísla tvoří takzvanou binární soustavu. Veškerý hardware (hmotné části počítače) i software (nehmotné části počítače, například program) je postavený kolem těchto dvou čísel. Každý počítač má nicméně svůj operační systém (OS), který umožňuje člověku (i počítači) komunikovat s hardwarem. Role OS v tomto případě je jazykový interpret – vezme informace od uživatele nebo aplikace a přeloží je do podoby, které samotný počítač rozumí.

K provedení, byť i jednoduchého úkolu, je zapotřebí nespočet jedniček a nul. Počítač má schopnost informace uchovávat, a to na interní úložiště (HDD) nebo externí úložiště (cloud). Základní jednotkou informace v informatice je jeden bit, který sestává z jedné jedničky nebo jedné nuly.  $N$  bitů se seskupuje do *bytů* a tvoří tak celek. *Byte* může mít různou velikost, obvykle platí  $1 \text{ byte} = 8 \text{ bitů}$ , tedy 256 permutací od 00000000 přes 01001101 až po 11111111 (Lengstorf et al., 2022).

Pokud uživatel zmáčkne na klávesnici například písmeno *M*, v mžiku jej vidí na obrazovce. Po zmáčknutí klávesy se totiž odešle binární vzor daného písmene přes OS do programu, který jej přeloží zpět do čitelné podoby. Tento proces je velmi zjednodušený a nezahrnuje popis paměti, ovladače, grafickou kartu ani monitor.

Pokud však chce uživatel pracovat s nějakými daty, musí je ukládat do proměnných či konstant. Proměnná v tomto případě je totožná s proměnnou z algebry, kdy vzoreček

$$A + B = C$$

jsou A, B i C proměnné a mohou reprezentovat jakékoliv číslo. V programování dokáží proměnné uchovat nejen čísla, ale i text či objekty. Správná syntaxe pro zápis proměnných v programování je reverzní, tedy

$$C = A + B$$

z důvodu funkce znaku = jakožto operátora přiřazení. Do lidské řeči by se předchozí výrok dal interpretovat jako *Do proměnné C ulož součet proměnných A a B*. Dokud nebude proměnná C přepsána, uchová tento součet, společně se zbylými proměnnými, v paměti počítače (Lengstorf et al., 2022).

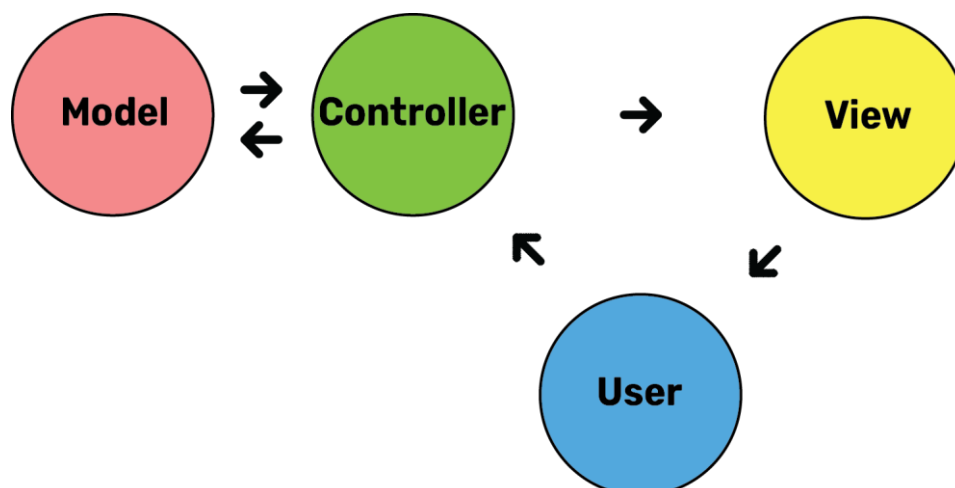
Ukládání do paměti počítače probíhá v již zmíněných bitech, a tudíž vše, čísla i písmena, se ukládají jako sousled jedniček a nul. Je proto nutné, aby se rozlišovaly datové typy proměnných. Pokud tedy chceme uložit například číslo 23, je velký rozdíl uložit jej jako text nebo jako číslo. Pokud by se s tímto číslem dále matematicky pracovalo, není možné jej uložit jako text.

Další razantní rozdíl je v číselné soustavě. Na rozdíl od počítačů využívají lidé desítkovou soustavu, tedy 0 – 9. Princip je totožný, nicméně rozsah čísel v binární soustavě je omezen na dvě čísla, tudíž se při vyčerpání možností musí přidat další cifra – v případě čísla 2 je jeho binární zápis 10 (binární čísla se čtou po jednotlivých cifrách, tedy *jedna nula*, nikoliv *deset*) (Lengstorf et al., 2022).

Tabulka 1 – Binární a desítková soustava (Lengstorf et al., 2022).

Binární	0	1	10	11	100	101	110	111
Desítková	0	1	2	3	4	5	6	7

Webové aplikace zastávají princip MVC (Model, View, Controller), kdy *Model* reprezentuje jednu databázovou tabulku, *View* představuje šablonu s daty uživateli a *Controller* se stará o HTTP požadavky, validace, úpravu dat a následnou odpověď uživateli. Na MVC je postaven i framework Laravel a obdobný příklad MVC je uveden ve Figure 16, Figure 17 a Figure 18 (Lengstorf et al., 2022).



V rámci HTTP požadavků existují způsoby, jakými jednotlivé požadavky řešit. Nejznámějšími způsoby jsou metody GET a POST, následované méně často používanými metodami PUT a DELETE. Existují ještě metody HEAD, OPTIONS a PATCH a téměř nikdy nepoužívané, alespoň v normálním vývojovém prostředí, metody TRACE a CONNECT. Následující tabulka shrnuje základní funkce každé metody:

Tabulka 2 – HTTP požadavky a jejich využití (Stauffer, 2019; MDN Web Docs, 2024).

<b>GET</b>	Vyžádání zdroje nebo seznamu zdrojů
<b>POST</b>	Vytvoří zdroj
<b>PUT</b>	Přepíše zdroj
<b>DELETE</b>	Smaže zdroj
<b>HEAD</b>	Vrátí pouze hlavičku GET odpovědi
<b>OPTIONS</b>	Zeptá se serveru, které možnosti jsou povolené na dané adrese
<b>PATCH</b>	Upraví zdroj
<b>TRACE</b>	Testuje zpětnou smyčku zpráv a poskytuje ladění
<b>CONNECT</b>	Vytvoří oboustrannou komunikaci s požadovaným zdrojem

### 3.1.1 PHP

Jazyků, které umožňují programátorovi sestavit webovou aplikaci, je nespočet. PHP (Hypertext Preprocessor) patří ke snadnějším jazykům na naučení a je vhodný pro každého začínajícího programátora (Lengstorf et al., 2022).

Aby mohl programátor používat PHP, stačí příponu souboru *.html* změnit na *.php* a má ihned přístup k celému jazyku za předpokladu, že je tento jazyk nainstalovaný na programátorově počítači. Pokud programátor chce vítat nové uživatele oznámením aktuálního dne a data, je tento problém v PHP, oproti statickému HTML, řešitelný mnohem jednodušeji:

```
<p>Dnes je středa, 11. října 20:55.</p>
```

*Figure 1 – Výpis aktuálního data s HTML (Nixon, 2021).*

```
<p>Dnes je <?php echo date('l, d. F H:i');?>.</p>
```

*Figure 2 – Výpis aktuálního data s PHP (Nixon, 2021).*

PHP bohužel v tomto případě neumí lokalizovat (přeložit) datum do požadovaného jazyka, tudíž bez pomoci překladu výstupu funkce `date()` se objeví hláška

*Dnes je Wednesday, 11. October 20:55.*

Výhodou je fakt, že tento výpis je dynamický a každou vteřinu se výpis obnovuje. Celý tento systém funguje na tzv. Unix časové známce, která se počítá od půlnoci 1. ledna 1970, kdy počáteční čas byl 1000000000. Do tohoto výpočtu se nezapočítávají přestupné vteřiny, které způsobují nechtěné časové posuny (Clewett, 2013).

S PHP má programátor kontrolu nad celým webovým serverem. Mimo editaci HTML umí i zpracovat nákup platební kartou, přidat záznam do databáze nebo dokonce získávat data z webových stránek třetích stran (Nixon, 2021).

Základem dobrého PHP kódu je dokumentace kódu. Komentář v PHP lze využít různými způsoby, záleží na počtu řádků komentáře. Jednořádkový komentář se píše se dvěma lomítky:

```
// Toto je jednořádkový komentář
```

*Figure 3 - Jednořádkový komentář v PHP (Nixon, 2021).*

zatímco víceřádkový komentář využívá jedno lomítko následované hvězdičkou (Nixon, 2021):

```
/*
  Toto
  je
  víceřádkový
  komentář
*/
```

Figure 4 - Víceřádkový komentář v PHP (Nixon, 2021).

Jedna z nejčastějších chyb při psaní PHP kódu je opomenutí středníku na konci každého řádku. Tím PHP vývojáři oznamuje, že kódu nerozumí a vrátí ParseError (Nixon, 2021):

```
echo "Hello world!";
```

Figure 5 - Ukázka PHP kódu (Nixon, 2021).

Znak dolaru (\$) se stal důležitým článkem napříč různými programovacími jazyky. V PHP plní dolar roli označení proměnné. Označení proměnné dolarem pomáhá PHP parseru být rychlejší ve čtení kódu. Ať už je proměnná číslo, text nebo pole, syntaxe by měla být následující (Nixon, 2021):

```
$integer = 1;
$string  = "Hello";
$array   = array("One", "Two", "Three");
$array   = ["One", "Two", "Three"];
```

Figure 6 – Datové typy integer, string a array v PHP (Nixon, 2021).

Dle Nixona (2021) je dobré pro lepší pochopení využít metaforu s krabičkou zápalek – každá krabička má na sobě napsaný název (proměnné) a na kousek papíru se napíše hodnota proměnné. Papírek se poté vloží do krabičky a tato hodnota zůstane přiřazená, dokud nebude papírek nahrazený.

Pokud je potřeba vytvořit novou proměnnou a rovnou ji přiřadit hodnotu nějaké již existující proměnné, je v PHP možné zapsat onu operaci takto:

```
$jiz_existujici_promenna = 100;
$nova_promenna = $jiz_existujici_promenna;
```

Figure 7 - Dynamické přiřazení proměnné do PHP (Nixon, 2021).

Pokud v proměnné `jiz_existujici_promenna` je uloženo číslo 100, tuto hodnotu bude v sobě udržovat i proměnná `nova_promenna`.

Do proměnné v PHP se dá uložit několik datových typů, mezi tři nejpoužívanější patří číslo, text a pole (Nixon, 2021). Další datové typy jsou boolean, objekt, null a resource (W3Schools, 2023).

První zmiňovaný datový typ je pojmenován podle britského matematika a filosofa George Boolea (\* 1815, † 1864), který položil základ logickým symbolům a jehož logika, dnes známá jako Booleova logika, je velmi důležitým aspektem fungování obvodů v dnešních počítačích (Encyclopaedia Britannica, 2023). Boolean v zásadě nabývá pouze dvou hodnot `true` nebo `false` a fungují jako výsledky podmínek.

Druhý datový typ `object` jde ruku v ruce s třídou. Zatímco třída vytváří prostor pro vytvoření objektů, objekt je instance třídy. Příkladem může být třída pojmenovaná *Auto* a uvnitř třídy může být nastavena proměnná na značku a barvu auta. V rámci třídy se volá funkce `__construct()`, jejíž účel je zpracovat vstupní parametry třídy, které následně předá do proměnných uvnitř objektu, viz Figure 9 níže (W3Schools, 2023).

```
<?php
class Auto
{
    public string $barva;
    public string $model;

    public function __construct($barva, $model)
    {
        $this->barva = $barva;
        $this->model = $model;
    }

    public function message(): string
    {
        return "Moje auto je " . $this->barva . " " . $this->model . ".";
    }
}
```

Figure 8 - PHP třída a objekt (W3Schools, 2023).

Pokud by se tento objekt chtěl využít v kódu, objekt by se dal zavolat následovně:

```
$moje_auto = new Auto("modrá", "Škoda Octavia");
```

Figure 9 - Přiřazení objektu do proměnné v PHP (W3Schools, 2023).

kdy *modrá* je vstupní hodnota pro barvu a *Škoda Octavia* je hodnota, která se uloží do proměnné `model`. Po zavolání tohoto kódu bude v proměnné `moje_auto` uložený celý objekt. Data z proměnné (objektu) lze dostat zavoláním funkce `message()` viz Figure 10.

```
echo $moje_auto->message();
```

Figure 10 - Výpis funkce `message()` (W3Schools, 2023).

Figure 10 tedy vrátí *Moje auto je modrá Škoda Octavia*. (W3Schools, 2023).

Null (nebo také NULL) je speciální datový typ, který reprezentuje nic. Null nastane v případě, když uživatel chce vypsát něco, co neexistuje. Výhodou je fakt, že null může také pomáhat v řešení podmínek, podobně jako boolean. Pokud něco neexistuje, nedá se s tím dále pracovat a zamezí se tak případným chybám (W3Schools, 2023; PHP, 2023).

V neposlední řadě umí PHP pracovat i se zdroji (resources). Zdrojem je myšlená proměnná, která má v sobě referenci na externí zdroj. Tímto zdrojem mohou být například cURL, FTP, LDAP, SSL, databázové systémy (MySQL, PostgreSQL, SQLite atp.) a spousta dalších. Pro práci s těmito zdroji existují v PHP speciální funkce, které jsou připravené s danými zdroji dále pracovat (PHP, 2024a).

PHP zastává, podobně jako operační systémy Linux, přístup „vše je soubor“. To znamená, že při práci se zdrojem (*resources*) je stream dat přenesen do podoby souboru, kterou PHP ukládá do dočasného úložiště. Nejčastější funkce pro práci s *resource* datovým typem jsou nativní `fopen()`, `file_get_contents()` a `file_put_contents()`. Tyto funkce pracují s dávkovačem (anglicky *buffer*), který přenos dat dávkuje. Zatímco ve Figure 11 - Příkaz `fopen()` (Mann, 2023). funkce `fopen()` vytvoří soubor a do proměnné `$fp` se uloží daný zdroj, dá se přes cyklus `while` ve Figure 12 demonstrovat, jak onen dávkovač funguje:

```
$fp = fopen('document.txt', 'r');
```

Figure 11 - Příkaz `fopen()` (Mann, 2023).

```
while (($buffer = fgets($fp, 4096)) !== false) {  
    echo $buffer;  
}  
  
fclose($fp);
```

Figure 12 - Demonstrace dávkovače v PHP (Mann, 2023).

Funkce `fgets()` čte jeden řádek souboru, dokud nenarazí na konec řádku nebo dokud nedosáhne limitu uvedeném jako druhý parametr (ve Figure 12 je limit 4096 bajtů). Pokud buffer narazí na EOF (End Of File, konec souboru), ukončí se načítání zdroje do dávkovače automaticky (Mann, 2023).



Pokud je potřeba zdroj zamknout, aby nebyly prováděny další změny, lze využít funkce `lock()` s parametry zdroje a status `LOCK_EX` nebo `LOCK_UN` (Mann, 2023).

### 3.1.2 Laravel framework

Dle webu [hostinger.com](https://hostinger.com) (2023) je Laravel momentálně nejpoužívanější framework v rámci jazyka PHP. Hodnotícími kritérii byli křivka rychlosti učení, MVC, Debuggování, Cachování a bezpečnost aplikace.

Laravel vznikl v roce 2013 Taylorem Otwellem. Mezi hlavní přednosti Laravelu patří čistý kód, dodržování moderních standardů programování, funkční balíčky, jednotné prostředí a velmi aktivní komunita programátorů. Popularity se dostává debuggovacímu panelu, který ukáže příčinu, proč něco nefunguje (Stauffer, 2023).



Obrázek 1 - Logo frameworku Laravel (Laravel, 2024a)

Laravel poskytuje mimo framework i spoustu balíčků v rámci vlastního ekosystému, které mezi sebou výborně spolupracují. Další informace o jednotlivých balíčcích a jejich funkcích popisuje následující kapitola 3.1.2.1.

#### 3.1.2.1 Ekosystém

Laravel disponuje několika balíčky, které ve spolupráci s jádrem frameworku tvoří aplikaci s jedním velkým ekosystémem funkcí. V tabulce na další stránce je představeno patnáct nejznámějších a nejoblíbenějších komponent frameworku Laravel. Vzhledem k povaze frameworku a neustálého vývoje nových komponent je obtížné zachytit trend používání Laravel balíčků.

Tabulka 3 – Logo, název a popis několika komponent Laravel ekosystému (Laravel, 2024b).

	Breeze	Odlehčená startovací sada se šablonovacím systémem Blade nebo Inertia
	Forge	Server management aplikace vhodná pro nasazení PHP aplikací na servery od DigitalOcean, Amazon nebo Hetzner
	Horizon	Přehledné uživatelské rozhraní pro sledování front řízených službou Redis v systému Laravel
	Inertia	Moderní statická stránka v jazycích React a Vue.js se standardním routováním na serveru
	Livewire	Vytvoření reaktivních a dynamických aplikací se šablonovacím systémem Blade
	Nova	Důmyslně vytvořený administrační panel pro Laravel aplikace
	Pint	Opravář stylu kódu postavený nad nástrojem PHP-CS-Fixer a zajišťuje, aby kód zůstal formátovaný a konzistentní
	Pulse	Průběžný přehled o výkonu a využití aplikace
	Reverb	Rychlý a škálovatelný protokol WebSocket
	Sail	Lokální vývoj Laravelu pomocí Dockeru
	Sanctum	Odlehčený systém ověřování pro jednostránkové aplikace, mobilní aplikace a jednoduché API založené na tokenech
	Scout	Ovladač pro přidání fulltextového vyhledávání do Eloquent modelů
	Socialite	Pohodlný způsob ověřování skrz OAuth2 – Facebook, Twitter, LinkedIn, Google, GitHub, GitLab, Bitbucket a Slack
	Spark	Plně vybavený fakturační portál
	Vapor	Platforma pro nasazení Laravelu bez serveru, od společnosti Amazon AWS

### 3.1.2.2 Výuka

Laravel pravidelně vydává svá vlastní videa Laracasts, ve kterých vysvětluje základní principy fungování frameworku. Mimo jiné lze na této stránce najít návod k instalaci, tipy a triky jednotlivých komponent a další užitečné rady, které pomohou vývojářům všech úrovní schopností (Laravel, 2024c).

### 3.1.2.3 Instalace

Podobně, jako má JavaScript svůj NPM, má PHP svůj Composer. To stejné platí i pro Laravel, který se nainstaluje v cílové složce spuštěním jediného Composer příkazu:

```
composer create-project laravel/laravel example-app
```

*Figure 13 – Příkaz na instalaci frameworku Laravel (Laravel, 2024d).*

Tímto příkazem dostane Composer příkaz vytvořit nový projekt z balíčku *laravel/laravel* a umístit celý framework do složky *example-app*. Složka nemusí být vytvořená, příkaz ji při absenci dokáže vytvořit během provádění příkazu. Po dokončení stahování se stačí přesunout do projektové složky a pustit příkaz

```
php artisan serve
```

*Figure 14 – Příkaz na spuštění Laravel serveru (Laravel, 2024d).*

který je podrobněji popsán v kapitole 3.1.2.5 (Laravel, 2024e). Je nutné mít na paměti, že Laravel ke správnému fungování vyžaduje OpenSSL, PHP, mbstring, Tokenizer, XML, ctype, JSON a BCMath rozšíření (Stauffer, 2023; Laravel, 2024f).

Pracování s Laravelem je mnohem jednodušší na Unix systémech (Linux, macOS, ...), vývojáři používající systém Windows musí své počítače trochu přizpůsobit. Dobrým krokem je instalace Dockeru a spouštění Docker kontejner s Laravel aplikací uvnitř anebo si stáhnout vývojářský nástroj XAMPP. Všichni vývojáři pak musí mít následující PHP rozšíření aktivované: OpenSSL, PDO, Mbstring, Tokenizer, XML, ctype, JSON a BCMath. Optimální je vyvíjet na nejmladších stabilních verzích PHP, tedy PHP 8.2 a 8.3 (Stauffer, 2023).

### 3.1.2.4 Schéma

Laravel obsahuje nespočet velmi propracovaných komponent, které vývojáři velmi usnadní celý proces programování aplikace. Struktura frameworku vypadá zhruba takto:

```
app/  
bootstrap/  
config/  
database/  
public/  
resources/  
routes/  
storage/  
tests/  
vendor/  
.editorconfig  
.env  
.env.example  
.gitattributes  
.gitignore  
artisan  
composer.json  
composer.lock  
package.json  
phpunit.xml  
readme.md  
vite.config.js
```

Figure 15 – Struktura frameworku Laravel (Stauffer, 2023).

Ve složce *app* jsou všechny důležité PHP kódy. *Bootstrap* obsahuje soubory, které Laravel používá při každém spuštění. Ve složce *config* jsou všechna nastavení, *database* uchovává databázové migrace, seedy a továrny (factories). Do *public* nahlíží server, který spouští soubor *index.php* a tudíž i celou *Bootstrap* složku. Do této složky je dobré ukládat i datové soubory – obrázky, videa, šablony stylů, skripty atp. V *resources* se nacházejí zejména front-end soubory včetně zdrojových stylů a skriptů. Z *routes* se berou definice HTTP cest, ale dají se zde nastavit i Artisan příkazy a konzolové cesty. Do *storage* se ukládá mezipaměť, zkompileované systémové soubory a logy (chybové i informační). Integrovaní a unit testy se pouští ve složce *tests* a všechny Composer balíčky a závislosti jsou umístěny ve složce *vendor*. Co se týče souborů mimo složky, tak *.editorconfig* dává vývojářovu IDE instrukce, jak například správně odsazovat řádky, znakovou sadu atd. Oba soubory *.env* a *.env.example* obsahují proměnné prostředí, kde se nastavují citlivé informace, které server potřebuje: heslo do e-mailu, údaje do databáze, klíč aplikace aj. Soubor *artisan* spouští Artisan příkazy, viz 3.1.2.5. Soubory *composer.json* a *composer.lock* jsou konfigurační soubory pro Composer, nicméně *composer.lock* není editovatelný a dává vhled do aktuálního nastavení všech závislostí. To samé dělá

i *package.json*, akorát místo PHP spolupracuje s NPM a JavaScriptem. Soubor *phpunit.xml* se používá na testování „out of the box“, *readme.md* obsahuje základní informace a *vite.config.js* slouží jako konfigurační soubor pro Vite (Stauffer, 2023).

### 3.1.2.5 Laravel Artisan

Laravel Artisan CLI (Command Line Interface) je prostředí příkazové řádky, který je součástí Laravelu. Již v základu obsahuje spoustu užitečných příkazů, od databázových migrací, přes správu mezipaměti až po příkazy vytvoření komponent.

Příkazem ve Figure 14 se spustí lokální server na portu 8000 (Laravel, 2024e).

### 3.1.2.6 Hello world v Laravelu

Aby programátor zobrazil větu *Hello world!* na stránce, potřebuje naprogramovat minimálně tři části: Route, Controller a View. Nejjednodušší nastavení všech tří komponent může vypadat přibližně takto:

*routes/web.php*

```
Route::get('/hello', [HelloWorldController::class, 'index']);
```

Figure 16 – Zobrazení Hello World v Laravelu, část Route (Laravel, 2024g).

*app/Http/Controllers/HelloWorldController.php*

```
class HelloWorldController extends Controller
{
    public function index(): View
    {
        return view('hello');
    }
}
```

Figure 17 – Zobrazení Hello World v Laravelu, část Controller (Laravel, 2024g).

*resources/views/hello.blade.php*

```
<p>Hello World!</p>
```

Figure 18 – Zobrazení Hello World v Laravelu, část View (Laravel, 2024g).

Pro zobrazení textové zprávy je potřeba mít nastavený base64 klíč v proměnné prostředí *APP\_KEY* a uložené v souboru *.env*. Tento klíč se vygeneruje příkazem

`php artisan key:gen`, a pokud je potřeba klíč pouze zobrazit, stačí přidat za konec příkazu přepínač `--show` (Stauffer, 2023; Laravel, 2024).

### 3.1.2.7 Výhody používání frameworku

Laravel má kromě přehlednosti a jednoduchosti také spoustu nástrojů, které vývojářům usnadňují práci. Jak již bylo zmíněno výše, Laravel má soubor s proměnnými prostředí `.env`, do kterých vývojář píše citlivá data, zejména hesla. S tímto souborem pak pracuje konfigurační složka `config/` a poskytuje vývojáři přístup k údajům v `.env` přes metodu `config()` (ačkoliv existuje i metoda `env()`, kterou vývojář může taktéž využít). Dále obsahuje chytrou práci s importovanými soubory skrz metodu `asset()`, kdy za soubor přidá čas poslední modifikace importovaného souboru tak, aby při změně nemusel vývojář mazat mezipaměť. S poslední zmiňovanou metodou se hodí používat zkratky do jednotlivých složek frameworku, Laravel má nadefinované `base_path()`, `app_path()`, `config_path()`, `database_path()`, `lang_path()`, `public_path()`, `resources_path()` a `storage_path()`. V rámci controllerů ve Figure 17 lze využít metody `abort()`, která vynutí spadnutí aplikace a zobrazení chybové stránky, dále metoda `dd()`, která je vhodná na debugování chyb, `today()` na zobrazení dnešního data, `session()` pro přístup do session, `request()` pro práci s http požadavky, `response()` spolu s `redirect()`, `to_route()`, `route()`, `url()` pro práci s URL, například při chybném/úspěšném zpracování požadavku (Laravel, 2024h).

Laravel má mimo helpery i třídy, které rozšiřují své pole působnosti na vyšší úroveň. V každé sekci frameworku – třídy, šablony, databáze, konfigurace, příkazy aj. – lze použít helper třídy. Laravel svým helper třídám vyhradil namespace `Illuminate/Support`, který se dále větví na jednotlivé kategorie. Díky Laravelu tak pomocí jedné třídy lze zjistit veškerý obsah tabulky z databáze, pracovat s poli, čísly i textovými řetězci, pomáhat se zpracováním formulářů a tak dále. Některé příkazy z nativního PHP a z Laravelu jsou uvedené v jednotlivých Figures níže. U všech výstřižků kódu jsou předpokladem správně nakonfigurované přístupy (Laravel, 2024h):

*Přístup do databáze*

```
$db = new PDO('mysql:host=localhost;dbname=test', $user, $password);
$stmt = $db->query('SELECT * FROM foo');
```

Figure 19 - Výpis dat z databáze, nativní PHP (PHP.net, 2024).

```
$foo = DB::table('foo')->get();
```

Figure 20 - Výpis dat z databáze, Laravel framework (Laravel, 2024h).

### Práce s polem (vypsát jména všech vývojářů)

```
$array = [
    ['developer' => ['id' => 1, 'name' => 'Martin']],
    ['developer' => ['id' => 2, 'name' => 'Milda']],
];

$names = [];
foreach ($array as $item) {
    $names[] = $item['developer']['name'];
}

print_r($names);

// ['Martin', 'Milda']
```

Figure 21 - Získání dat z pole, nativní PHP (PHP.net, 2024).

```
use Illuminate\Support\Arr;

$names = Arr::pluck($array, 'developer.name');
// ['Martin', 'Milda']
```

Figure 22 - Získání dat z pole, Laravel framework (Laravel, 2024h).

### Zobrazení velikosti souboru

```
function formatFileSize($bytes) {
    $units = ['B', 'KB', 'MB', 'GB', 'TB', 'PB'];
    $index = 0;
    while ($bytes >= 1024 && $index < count($units) - 1) {
        $bytes /= 1024;
        $index++;
    }
    return round($bytes, 2) . ' ' . $units[$index];
}

$size = formatFileSize(1024);

echo $size; // 1 KB
```

Figure 23 - Přepočítání velikosti souboru do standardní formy, nativní PHP (PHP.net, 2024).

```
use Illuminate\Support\Number;

$size = Number::fileSize(1024);
echo $size; // 1 KB
```

Figure 24 - Přepočet velikosti souboru do standardní formy, Laravel framework (Laravel, 2024h).

### Převod znaku do kódu ASCII (American Standard Code for Information Interchange)

```
$slice = iconv('UTF-8', 'ASCII//TRANSLIT', 'û');

echo $slice; // u
```

Figure 25 - Převod znaku do ASCII, nativní PHP (PHP.net, 2024).

```
use Illuminate\Support\Str;

$slice = Str::ascii('û');
echo $slice; // u
```

Figure 26 - Převod do znaku ASCII, Laravel framework (Laravel, 2024h).

Ačkoliv poslední operace nezkracuje samotný kód, je přinejmenším intuitivnější.

#### 3.1.2.8 Nevýhody používání frameworku

Laravel, podobně jako ostatní frameworky, funguje jako jeden velký ekosystém. To v praxi znamená, že pokud vývojář nefunguje podle nastavených předpisů, ztěžuje si práci. Některé funkce při nahrazení určité části frameworku nemusí plně fungovat, nebo dokonce mohou úplně chybět.

Další nevýhodou jsou jednoduché projekty, využití frameworku v těchto případech postrádá větší smysl, jelikož má kostra Laravelu velikost zhruba 60 MB (Megabajtů) (Laravel, 2024a).

Omezujícím faktorem je také fakt, že Laravel je open-source. Framework je sice zdarma k využívání (nebo je jeho použití omezeno licenci), jeho vývoj a aktualizace závisí na vývojářích, kteří jej vyvíjejí ve svém volném čase.

#### 3.1.3 Databázové systémy

Databáze dává webovým aplikacím zcela nový rozměr. Funguje jako strukturovaná kolekce záznamů uložená v počítači nebo na vzdáleném serveru tak, aby se v datech dalo rychle hledat či filtrovat.



Společné jádro většiny databází, SQL (Structured Query Language), je postavené na jednoduchosti syntaxe jednotlivých příkazů, které se do databáze posílají. Příkladem je příkaz na výpis všech uživatelů, kterým je přesně 24 let:

```
SELECT * FROM users WHERE age = 24;
```

Figure 27 – Jednoduchý SQL příkaz (Nixon, 2021).

Jedna databáze může mít jednu a více tabulek a jedna tabulka může mít jeden nebo více řádků (záznamů). Příklad jednoduché tabulky *osoby* je uveden níže:

Tabulka 4 – Jednoduchá SQL tabulka (Nixon, 2021).

Jméno	Věk	Bydliště	Povolání
Fresh Prince	22	West Philadelphia	rapper
Carlton Banks	23	Bel-Air	student
Jazzy Jeff	22	Compton	DJ

Databázových systémů existuje celá řada. Mezi nejrozšířenější databáze patří MySQL, která byla odkoupena společností Oracle, a tudíž přestala být open-source (zdrojový kód přístupný veřejnosti). Poslední veřejná verze MySQL byla naklonovaná a vznikl nový systém MariaDB, jenž je open-source doted' (Nixon, 2021).

Oba zmíněné systémy využívají Structured Query Language, ale existují i databáze postavené na jiném jádru. Mezi jiná jádra patří ORM (Object Relation Mapping) spolu se systémy Eloquent a Doctrine (Doctrine, 2023). První zmíněný systém používá i framework Laravel, ačkoliv není prioritou použití (Laravel, 2024i). Modely ORM využívají takzvaný DQL (Data Query Language), který je sice méně čitelnější než SQL, na druhou stranu je vhodnější pro komplexnější příkazy (Doctrine, 2023).

Na Figure 28 je vidět stejný příkaz jako na Figure 27 pro výpis všech uživatelů, kterým je přesně 24 let:

```
<?php
$query = $em->createQuery('SELECT u FROM App\Model\User u WHERE
u.age = 24');
$users = $query->getResult();
```

Figure 28 – DQL příkaz (Doctrine, 2023).

DQL závisí na EntityManageru, jenž má přístup ke všem entitám vytvořených v rámci mapování. Dalším krokem je klauzule SELECT, která definuje, co chce

programátor s tabulkou dělat (v tomto případě vybrat vhodné řádky). Do aliasu *u* třídy `App\Model\User` se uloží všechny instance třídy `User`, kteří vyhovují filtru definovaném po klauzuli `WHERE`. Dle příkazu výše se hledají všechny `User` instance, jejichž sloupec *age* má hodnotu přesně 24. V případě Tabulka 4 – Jednoduchá SQL tabulka (Nixon, 2021 by tedy DQL příkaz z Figure 28 – DQL příkaz (Doctrine, 2023) vrátil prázdnou množinu výsledků (Doctrine, 2023).

Výhodou DQL je, že si můžeme z tabulky filtrovat nejen celé entity, ale také skalární data nebo dokonce kombinace obou, viz Figure 29:

```
SELECT u, p.quantity FROM MyProject\Model\User u ...
```

*Figure 29 – Objektové i skalární filtrování v DQL (Doctrine, 2023).*

Ve výsledné práci je zvolena kombinace obou zmíněných variant, a to využití nativního Eloquent ORM ve spolupráci s MySQL.

### 3.1.3.1 Základní orientace tabulkou

Jelikož tato práce zahrnuje operace v rámci relační databáze MySQL, jsou veškeré budoucí popisy zaměřeny na tento systém.

V prostředí MySQL existuje mnoho příkazů a v následující tabulce je vypsáno několik nejdůležitějších z nich:

Tabulka 5 - Základní příkazy MySQL (Nixon 2021).

Příkaz	Akce
ALTER	Upravit tabulku
BACKUP	Zálohovat tabulku
CREATE	Vytvořit tabulku
DELETE	Smazat řádek tabulky
DESCRIBE	Popsat sloupce tabulky
DROP	Smazat tabulku
EXIT nebo QUIT	Odejít z prostředí MySQL
GRANT	Změnit uživatelská nastavení
HELP	Zobrazit nápovědu
INSERT	Vložit data
LOCK	Zamknout tabulku
RENAME	Přejmenovat tabulku
SHOW	Vypsát data o objektu
SOURCE	Spustit soubor
STATUS	Zobrazit status
TRUNCATE	Vyčistit tabulku od dat

### 3.1.3.2 Datové typy

Při zakládání nové tabulky v databázi je nutné určit datový typ každého sloupce. Nejzákladnější typy jsou číslo, desetinné číslo, text, dlouhý text a binární data, která jsou postupně představena v následujících odstavcích.

MySQL umí velmi dobře pracovat s čísly. V rámci databázového systému se rozlišují dva hlavní číselné typy – celá čísla a reálná čísla. Celá čísla se dále rozdělují na čtyři podrobnější typy dle velikosti, viz následující tabulka. Tato čísla mimo jiné mohou využít speciální atribut `UNSIGNED`, který eliminuje použití záporných čísel a rozsah kladných čísel se zhruba zdvojnásobí. Opačnou hodnotou je `SIGNED`, která je zároveň výchozím stavem. Jako příklad lze využít fiktivní sloupec `ID` s typem `TINYINT`, do kterého chceme ukládat čísla 1 až  $n$ . Při přidání atributu `UNSIGNED` se změní rozsah z -128 až 127 na 0 až 255 (Botros a Tinley, 2021).

Tabulka 6 - Číselné datové typy, celá čísla (Nixon, 2021).

Datový typ	Použito bajtů	SIGNED		UNSIGNED	
		MIN	MAX	MIN	MAX
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8.38e + 6	8.38e + 6	0	1.67e + 7
INT	4	-2.15e + 9	2.15e + 9	0	4.29e + 9
BIGINT	8	-9.22e + 18	9.22e + 18	0	1.84e + 19

Nixon (2021) v kategorii reálných čísel zmiňuje další dva typy DOUBLE a FLOAT. Jediný rozdíl je pouze v přesnosti desetinného čísla (anglicky floating-point precision). Tyto typy se používají, pokud je potřeba pracovat s desetinnými čísly s velkou přesností několika cifer za desetinnou čárkou. Stojí za zmínku, že datové typy FLOAT i DOUBLE nemají atribut UNSIGNED, jejich hodnota je tedy v Tabulka 7 - Číselné datové typy, reálná čísla (Nixon, 2021). vynechána.

Tabulka 7 - Číselné datové typy, reálná čísla (Nixon, 2021).

Datový typ	Použito bajtů	SIGNED		UNSIGNED	
		MIN	MAX	MIN	MAX
FLOAT	4	-3.4e + 38	3.4e + 38	N/A	N/A
DOUBLE	8	-1.80e + 308	1.80e + 308	N/A	N/A

Autoři Botros a Tinley (2021) udávají ještě jeden datový typ DECIMAL. Tento datový typ se používá, pokud ukládané číslo překračuje rozsah typu BIGINT. Zároveň zabírá více místa v paměti než FLOAT a DOUBLE, na druhou stranu je velmi precizní v počítání zlomkových čísel. Autoři doporučují používat DOUBLE pro ukládání peněžních výsledků, které mohou mít až pět desetinných míst. Zároveň dodávají, že je taktéž možné každé číslo vynásobit číslem 10 000 a dále s ním pracovat ve formátu BIGINT. Syntax zápisu je DECIMAL ( $M$ ,  $D$ ), kde  $M$  určuje počet cifer v rozsahu 1-65, pokud se vynechá, nastaví se výchozí hodnota 10. Druhé písmeno  $D$  je počet cifer za desetinnou čárkou, má rozsah od 0 do 30 a nesmí být větší než  $M$ . Pokud se  $D$  vynechá, použije se výchozí hodnota 0.

Dalším hojně používaným datovým typem je text. MySQL rozlišuje dva typy, CHAR a VARCHAR. Hlavní rozdíl mezi nimi je v používání počtu bitů. Následující tabulka zobrazuje, jaký vliv má zvolení typů CHAR, případně VARCHAR na celkovou velikost tabulky.

Tabulka 8 - Datový typ CHAR a VARCHAR (Nixon, 2021).

Datový typ	Použito bajtů	Příklad
CHAR( <i>n</i> )	$n (\leq 255)$	CHAR(4) "Ahoj" = 4 bajty CHAR(57) "Dobrý den" = 57 bajtů
VARCHAR( <i>n</i> )	1 až $n (\leq 65535)$	VARCHAR(4) "Ahoj" = 4 bajty VARCHAR(57) "Dobrý den" = 9 bajtů

Zatímco VARCHAR může být lepší volbou v drtivé většině případů, datový typ CHAR se může použít v případě, kdy je jisté, že každý záznam bude mít fixní délku, například u šifry MD5 (v tomto případě CHAR (40)). Zároveň je CHAR výhodnější, pokud je třeba ukládat pouze jeden znak – CHAR (1) zabere 1 bajt paměti, na druhou stranu typ VARCHAR (1) zabírá 2 bajty kvůli jednomu bitu vyhrazenému na určení velikosti (Botros a Tinley, 2021).

Do databáze lze také ukládat binární data, k tomu jsou opět určené dva typy BINARY a VARBINARY. Tento typ se používá, pokud je potřeba uložit dlouhý binární řetězec, například obrázek ve formátu GIF. Binární řetězce jsou velmi podobné konvenčním (textovým) řetězcům, ale namísto znaků ukládají bajty. Zároveň je binární porovnávání snazší než znakové porovnávání, a tudíž je i rychlejší (Botros a Tinley 2021).

Tabulka 9 - Datový typ BINARY a VARBINARY (Nixon, 2021).

Datový typ	Použito bajtů	Příklad
BINARY( <i>n</i> )	$n (\leq 255)$	Stejně jako CHAR, ale ukládá binární data
VARBINARY( <i>n</i> )	1 až $n (\leq 65535)$	Stejně jako VARCHAR, ale ukládá binární data

Další datové typy jsou uzpůsobené pro ukládání velkého množství dat, a proto se u nich nepíše maximální povolená délka. Jedná se o datové typy BLOB a TEXT, přičemž oba jmenování se dále dělí, podobně jako číselné datové typy, na TINYBLOB, SMALLBLOB, BLOB, MEDIUMBLOB a LONGBLOB, potažmo TINYTEXT, SMALLTEXT, TEXT, MEDIUMTEXT a LONGTEXT. Ve většině případů fungují typy SMALLBLOB a BLOB jako synonyma, stejně tak tomu je i v druhém případě a v pozdějších referencích bude uváděn pouze kratší zápis. Faktem je, že různé databázové enginy se k uloženým hodnotám chovají jako k objektům s vlastní identitou. To v praxi znamená, že například InnoDB engine ukládá tyto objekty na externí úložiště a každá hodnota vyžaduje 1 až 4 bajty úložiště v řádku, plus dostatek místa v onom externím úložišti pro uložení samotné hodnoty.

Tabulka 10 - Datový typ BLOB (Nixon, 2021).

Datový typ	Použito bajtů	Příklad
TINYBLOB	$n (\leq 255)$	Binární data, bez znakové sady a porovnání
BLOB	1 až $n (\leq 65535)$	
MEDIUMBLOB	1 až $n (\leq 1.67e + 7)$	
LONGBLOB	1 až $n (\leq 4.29e + 9)$	

Oba datové typy jsou téměř identické s tím rozdílem, že typ TEXT využívá ještě znakovou sadu a porovnávací sadu, která umožňuje ukládat například text s diakritikou.

Tabulka 11 - Datový typ TEXT (Nixon 2021).

Datový typ	Použito bajtů	Příklad
TINYTEXT	$n (\leq 255)$	Textový řetězec se znakovou sadou a porovnáním
TEXT	1 až $n (\leq 65535)$	
MEDIUMTEXT	1 až $n (\leq 1.67e + 7)$	
LONGTEXT	1 až $n (\leq 4.29e + 9)$	

V dalších příkladech budou zobrazena využití jednotlivých příkazů. Pro simulátor databáze byl využit jazykový model ChatGPT verze 3.5 od společnosti OpenAI dostupný na adrese <https://chat.openai.com>.

Nixon jako první příkaz uvádí `CREATE DATABASE` název s poznámkou, kdyby byl uživatel omezený jedinou databází, což není tento případ. V případě úspěšného provedení příkazu by databáze měla vrátit `Query OK` a počet ovlivněných řádků.

```
CREATE DATABASE master;
```



Obrázek 2 - Příkaz MySQL, vytvoření databáze (Nixon, 2021).

Následně nastavíme používání práv vytvořené databáze:

```
USE master;
```



Obrázek 3 - Příkaz MySQL, používání databáze (Nixon, 2021).

A vytvoří se nový uživatel `xdubm006` s příslušnými právy

```
CREATE USER 'xdubm006'@'localhost' IDENTIFIED BY 'heslo';  
GRANT PRIVILEGES ON database.object TO 'xdubm006'@localhost;
```



Obrázek 4 - Příkaz MySQL, založení uživatele (Nixon, 2021).

Nyní je možné vytvořit tabulku. Příkaz `CREATE TABLE` umožní vytvořit tabulku s daným názvem následovaný závorkami, kde jsou určeny všechny sloupce a jejich datové typy. Poslední dvě slova, `ENGINE InnoDB`, naznačují, jaký engine použít v rámci databáze. Od verze 5.6 se InnoDB stává výchozí možností zejména proto, že umožňuje fulltextové vyhledávání (Nixon, 2021):

```
CREATE TABLE books (author VARCHAR(128), title VARCHAR(128),  
type VARCHAR(16), year CHAR(4)) ENGINE InnoDB;
```

### ChatGPT

```
sql Copy code  
  
Query OK, 0 rows affected
```

Obrázek 5 - Příkaz MySQL, vytvoření tabulky (Nixon, 2021).

Všechny operace se dají kontrolovat příkazem DESCRIBE nebo SHOW.

```
DESCRIBE books;
```

### ChatGPT

```
sql Copy code  
  
+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| author | VARCHAR(128)  | YES  |     | NULL    |      |  
| title  | VARCHAR(128)  | YES  |     | NULL    |      |  
| type   | VARCHAR(16)   | YES  |     | NULL    |      |  
| year   | CHAR(4)       | YES  |     | NULL    |      |  
+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

Obrázek 6 - Příkaz MySQL, zobrazení struktury tabulky (Nixon, 2021).

Příkaz DESCRIBE zobrazuje vlastnosti každého sloupce v šesti variantách. *Field* ukazuje název pole tak, jak jej vidí uživatel, *Type* ukazuje datový typ, *Null* znamená, zdali pole musí obsahovat hodnotu, *Key* určuje typ klíče, *Default* výchozí hodnotu a *Extra* nějaké doplňující informace (například jestli hodnota se zvedá automaticky, tzv. *auto-increment*) (Nixon 2021).

Aby byla data takzvaně serializovaná, musí tabulka obsahovat primární klíč – unikátní identifikátor, kterým se rozliší jednotlivé řádky tabulky. K tomu slouží například sloupec ID, které využívá zmíněnou vlastnost *auto-increment*. Příkaz ALTER TABLE se využívá, pokud již máme vytvořenou, ale chceme ji dodatečně upravit. INT je další datový typ značící celé číslo (Integer), UNSIGNED znamená pouze přirozená čísla, navíc



se prodlouží maximální hodnota na zhruba 4 biliony, konkrétně 4 294 967 295 unikátních záznamů.

```
ALTER TABLE books ADD id INT UNSIGNED NOT NULL  
AUTO_INCREMENT KEY;
```

### ChatGPT

```
sql Copy code  
  
Query OK, 0 rows affected
```

Obrázek 7 - Příkaz MySQL, úprava tabulky (Nixon, 2021).

Pokud se zavolá opět příkaz DESCRIBE books, přibude nový sloupec id s veškerým nastavením.

```
DESCRIBE books;
```

### ChatGPT

```
sql Copy code  
  
+-----+-----+-----+-----+-----+-----+  
| Field | Type                | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id    | INT(10) UNSIGNED   | NO   | PRI | NULL    | auto_increment |  
| author | VARCHAR(128)       | YES  |     | NULL    | |  
| title | VARCHAR(128)       | YES  |     | NULL    | |  
| type  | VARCHAR(16)        | YES  |     | NULL    | |  
| year  | CHAR(4)            | YES  |     | NULL    | |  
+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```


Obrázek 8 - Příkaz MySQL, kontrola struktury (Nixon, 2021).

Pokud by naopak bylo potřeba nějaký sloupec odstranit, stačí opět použít příkaz ALTER TABLE.

```
ALTER TABLE books DROP type;
```

 ChatGPT

```
sql
```

 Copy code

```
Query OK, 0 rows affected
```


Obrázek 9 - Příkaz MySQL, odstranění sloupce (Nixon, 2021).

Nyní je tabulka připravená i zkontrolovaná a lze ji vyplnit daty.

```
INSERT INTO classics(author, title, type, year) VALUES
    ('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year) VALUES
    ('Jane Austen','Pride and Prejudice','Fiction','1811');
INSERT INTO classics(author, title, type, year) VALUES
    ('Charles Darwin','The Origin of Species','Nonfiction','1856');
INSERT INTO classics(author, title, type, year) VALUES
    ('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year) VALUES
    ('William Shakespeare','Romeo and Juliet','Play','1594');
```

 ChatGPT

```
sql
```

 Copy code

```
Query OK, 1 row affected
Query OK, 1 row affected
Query OK, 1 row affected
Query OK, 1 row affected
Query OK, 1 row affected
```

Obrázek 10 - Příkaz MySQL, vložení dat do tabulky (Nixon, 2021).

Vypsání všech dat lze provést příkazem `SELECT * FROM books`. Hvězdička značí vše a `FROM` říká, z jaké tabulky se vypíší data. Pokud by místo hvězdičky bylo například `author`, vypsali by se pouze autoři daných knih.

```
SELECT * FROM books;
```



ChatGPT

```
sql Copy code
```

id	author	title	year
1	Mark Twain	The Adventures of Tom Sawyer	1876
2	Jane Austen	Pride and Prejudice	1811
3	Charles Darwin	The Origin of Species	1856
4	Charles Dickens	The Old Curiosity Shop	1841
5	William Shakespeare	Romeo and Juliet	1594

5 rows in set (0.00 sec)

Obrázek 11 - Příkaz MySQL, získání dat z tabulky (Nixon, 2021).

Vše v tabulce se dá upravovat – tabulka samotná, sloupce, datové typy, vazby i klíče. Následující příkazy postupně přejmenují tabulku, poté se změní datový typ sloupce `year`, přidá se sloupec `pages` a následně se smaže, a nakonec se naindexuje sloupec `title`.

```
ALTER TABLE books RENAME classics;
ALTER TABLE classics MODIFY year SMALLINT;
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
ALTER TABLE classics DROP pages;
ALTER TABLE classics ADD INDEX(title(20));
```

Indexování sloupce znamená, že se nově vložená data zkrátí na pevný počet znaků, v případě `ADD INDEX(sloupec(20))` zkrátí automaticky text *The Adventures of Tom Sawyer* na *The Adventures of To*.

### 3.1.4 Verzování

V rámci udržení historie vývoje projektu je do výsledné práce zakomponován verzovací systém Git. Proč právě tento systém je popsáno v kapitole 3.2.4 Git a GitHub.

Git funguje v rámci repozitářových služeb, například GitHub. Uživatel má na GitHubu svůj profil a zakládá si v něm repozitáře. Do repozitáře poté přes git příkazy nahrává aktualizace projektu (Skoulikari, 2023).

Verzovací systém git má mnoho příkazů, běžný vývojář si vystačí se zhruba dvanácti příkazy, viz následující tabulka:

Tabulka 12 - Základní git příkazy (Skoulikari, 2023).

Příkaz	Akce
<i>clone</i>	Naklonuje vzdálený repozitář do vybrané složky
<i>init</i>	Inicializuje nový git repozitář
<i>add</i>	Přidá soubor(y) do git repozitáře
<i>remote</i>	Nastaví adresu ke vzdálenému repozitáři (například na GitHubu).
<i>fetch</i>	Stáhne všechny neaktuální commity do git repozitáře
<i>commit</i>	Připraví aktuální verzi projektu
<i>push</i>	Nahraje commit(y) do vzdáleného repozitáře
<i>pull</i>	Podobné jako <i>fetch</i> , všechny commity rovnou integruje
<i>merge</i>	Sjednocení pracovního prostředí podle předem určených strategií
<i>checkout</i>	Přepnutí na jinou git větev
<i>status</i>	Aktuální status git repozitáře
<i>diff</i>	Zobrazí změny od posledního commitu

V rámci vývoje se vývojáři pro založení nového git repozitáře vytvoří také složka *.git* v adresáři projektu. Tento adresář obsahuje veškeré informace o projektu z hlediska verzování, a také je klíčovým spojením se vzdáleným repozitářem (Skoulikari, 2023).

Jakmile má vývojář připravenou funkční verzi programu, může připravit commit, kterým zaznamená stav všech souborů a může pokračovat v práci dále. *Rollback* commitu znamená, že změny mohou být zahozeny. Naopak, pokud se commit(y) nahrají na vzdálený repozitář, již není tak snadné jej smazat. Poslední commit na aktuální větvi zaznamenává *HEAD* a vše, co má předponu *origin/*, je reference na vzdálený repozitář (Skoulikari, 2023).

### 3.1.4.1 GitHub Actions

V rámci verzování nabízí GitHub automatické procesy, tzv. *workflows*. Tato automatizace se nazývá GitHub Actions a je přístupná pro všechny úrovně předplatného, viz následující tabulka:

Tabulka 13 - Přehled využívání GitHub Actions dle předplatného (Laster, 2023).

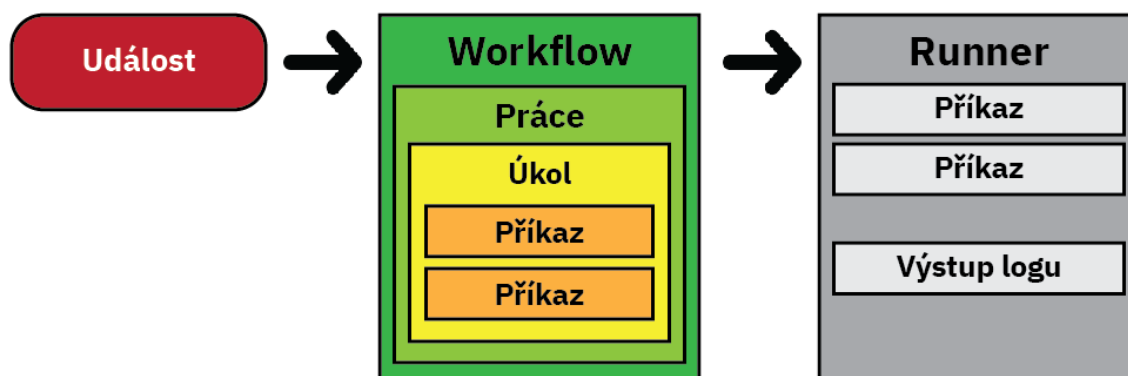
Plán	Úložiště	Minut (za měsíc)
Free	500 MB	2 000
PRO	1 GB	3 000
Free pro organizace	500 MB	2 000
Team	2 GB	3 000
Enterprise Cloud	50 GB	50 000

Workflow se spouští automaticky pokaždé, když je splněna kterákoliv z následujících podmínek:

1. Uživatel nebo proces udělá změnu v GitHub repozitáři,
2. Je aktivován externí spouštěč (mimo prostředí GitHub),
3. Je nastavený časovač, který v určitém čase či intervalu spouští workflow,
4. Workflow je spuštěn ručně,

přičemž nejčastější spouštěč je možnost první. (Laster, 2023).

Každý workflow zachovává určité schéma spouštění, viz následující Obrázek 12:



Obrázek 12 - Schéma GitHub workflow (Laster, 2023).

Workflow se nastavuje skrz soubor YAML (YAML Ain't Markup Language, rekurzivní akronym) umístěný v adresáři `.github/workflows`. Obsahuje veškeré kroky od spouštěče až po výpis hlášky o ukončení akce (Laster, 2023).

V této práci je implementován jeden workflow, který usnadňuje nahrání souborů na produkční server, více bude představen v praktické části.

### 3.1.5 Hosting

Hosting je služba, kdy si uživatel pronajímá serverové úložiště od poskytovatele pro své webové stránky. Hostingů je celá řada a závisí zejména na ceně a výhodách.

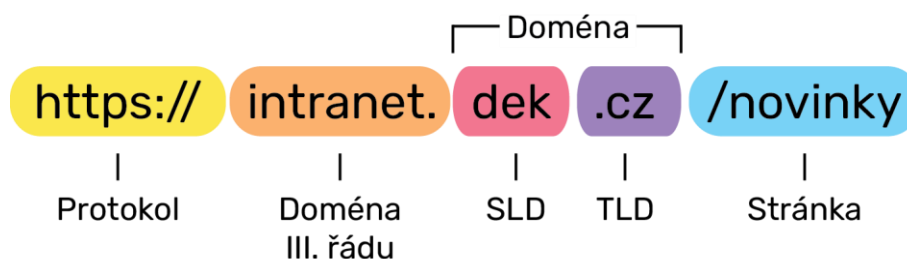
#### 3.1.5.1 Freehosting

Freehosting je nejlevnější varianta. Hosting poskytuje zdarma, ale má několik nevýhod. Jsou většinou omezené prostorem, službami, nejsou moc spolehlivé a většinou vám do stránky přidají reklamu. Pro testovací provoz vhodné, jinak spíše ne (Janovský, 2023).

#### 3.1.5.2 Doména a hosting

Je potřeba rozlišovat mezi pojmy doména a hosting. Hosting byl vysvětlený již na počátku podkapitoly 3.1.5.

Novým pojmem je doména, která slouží jako cílový bod pro návštěvníky webových stránek. Existuje několik stupňů domén a klasická doména má v drtivé většině dva řády: TLD (Top Level Domain) a subdoménu 2. řádu. Doména nejvyššího řádu je například `.cz`, `.eu`, `.com`, `.org`, `.aero` nebo třeba `.london`. Na některé domény platí restrikce, například TLD `.eu` lze využívat pouze v takových případech, pokud je majitel domény občanem v Evropské Unii. Subdoménu druhého řádu si volí uživatel a musí být unikátní. Rozložení různých domén představuje Obrázek 13 níže.



Obrázek 13 - Představení doménového systému (Janovský, 2023).

### 3.1.5.3 Datový limit

Důležitým faktorem ve vybírání optimálního hostingu je datový prostor neboli datový limit. Spousta jednoduchých webů si vystačí s 10 MB prostoru. Pokud je na webu hodně videí nebo fotek, je potřeba prostor zvýšit. Spousta poskytovatelů má dostatek místa na svých discích, může se také stát, že k hostingu bude nabízen i neomezený prostor (Janovský, 2023).

S prostorem souvisí o *traffic*, který říká, jaký objem dat se přenesl z webu na páteřní síť. Za traffic poskytovatelé samozřejmě platí a zákazníkům pak upravují ceny hostingu dle jejich trafficu (Janovský, 2023).

### 3.1.5.4 Mail

Nastavení e-mailových služeb by měla být samozřejmost a většinou je nabízena v rámci hostingu. Občas může být nastavení e-mailových adres obtížné nebo může mít poskytovatel hůře nastavené DKIM (Domain Keys Identified Mail) a odchozí e-maily mohou končit ve spamu. Některé maily v rámci hostingu mohou mít lépe vyřešené spam filtry, což se může zákazníkům také hodit (Janovský, 2023).

### 3.1.5.5 Operační systém a software

Na jakém operačním systému bude hosting fungovat, je v zásadě jedno – nijak to neovlivňuje příchozí návštěvníky na výsledný web. V zásadě se rozlišují dva (Windows a Linux), kde preference zůstává na samotných vývojářích.

Webové servery (software) jsou úzce spjaty s OS a používají se Apache nebo Nginx. Oba jsou velmi rychlé a na internetu k nim existuje velké množství návodů. Na serverech s Windows je lepší využít IIS (Internet Information Server) od společnosti Microsoft. Má lepší logování aktivity na serveru, ale je možné využít i Apache server (Janovský, 2023).

V praxi je většinou dodržováno jedno pravidlo viz tabulka níže:

Tabulka 14 - Doporučené nastavení hostingu (Janovský, 2023).

Programovací jazyk	Operační systém	Webový server
PHP	Linux	Apache
ASP nebo .NET	Windows	IIS

#### 3.1.5.6 Podpora skriptování

V neposlední řadě je dobré se porozhlédnout i po podpoře skriptování na straně serveru. Skriptování vyžadují i redakční systémy, například WordPress. Pokud na hostingu bude uložena jednoduchá statická stránka bez ukládání dat od návštěvníků, skriptování není potřeba. Je ale dobré si zjistit, jestli to daný webový server umí, aby se v případě nutnosti využití skriptování nemuselo přecházet k jinému poskytovateli (Janovský, 2023).

#### 3.1.5.7 Logování

Servery umožňují zapisovat informace o provozu do souborů, kterým se říká logy. Běžný serverový log má spoustu řádků a na každém z nich je zaznamenána většina informací, co vývojář potřebuje. Do toho patří IP adresa návštěvníka, čas, o jaký soubor požádal, kde kliknul a jaký měl prohlížeč (odborně User-Agent).

Samotné logování zahrnuje servery, a proto je dobré si dopředu zjistit, jestli je poskytovatelem nabízeno, případně za jakou cenu. Na serverech, kde je logování zapnuté, se obvykle záznamy promazávají (tzv. rotace logů). V případě potřeby je nutné si záznamy průběžně stahovat, buď přes administraci nebo přes protokol FTP (Janovský, 2023).

#### 3.1.5.8 Certifikáty

Webové stránky se dají provozovat na nezabezpečeném (HTTP) nebo na zabezpečeném (HTTPS) protokolu. Pokud stránka není provozována na HTTPS protokolu, téměř všechny prohlížeče o tom varují návštěvníky, kteří se mohou rozhodnout na stránku nevstupovat. Od roku 2016 nabízí certifikační služba Let's Encrypt SSL certifikát zdarma (SSL certifikát zaručí HTTPS protokol na webu). Zabezpečená stránka zabraňuje spoustě bezpečnostním rizikům, například MITM (Man In The Middle). Zjednodušeně to znamená, že blíže nespecifikovaná osoba vlastní jeden určitý bod v internetu, přes který putuje provoz a má nad ním absolutní kontrolu. Pokud chce, může odchyťvat hesla a veškeré citlivé údaje posílané přes nezabezpečený HTTP protokol. Protokol HTTPS v jednoduchosti zaručuje následující:

1. Jistotu komunikace se serverem, a ne s útočníkem,
2. Útočník nemůže nijak pozměnit předávaný obsah uživateli,
3. Útočník nevidí, jaké stránky uživatel navštěvuje.



Google v roce 2015 uvedl, že hodnotí stránky s HTTPS protokolem kladně (vzhledem k optimalizaci vyhledávání SEO) (Janovský, 2023).

#### 3.1.5.9 Soubor .htaccess

Na některých serverech je povoleno použití souboru *.htaccess*. Pomocí tohoto souboru si může autor webu nastavit určité věci, které v jiných případech musí nastavovat správce serveru, jež je povětšinou nedostupný. Mít povolený *.htaccess* tak přináší své výhody. Soubor *.htaccess* se dá použít, pokud na serveru běží Apache. Mezi nejjednodušší příkazy nastavitelné pomocí tohoto souboru jsou:

1. jednoduché přesměrování
2. Nastavení chybových stránek
3. Nastavení výchozího souboru
4. Nastavení procházení adresářů
5. Povolení, zákazy a ochrana heslem
6. Nastavení mezipaměti (cache)

Je dobré si dopředu zjistit, jestli poskytovatel má povolený *.htaccess* v konfiguračním souboru (Janovský, 2023).

#### 3.1.5.10 Ochrana proti (D)DOS útokům

Mezi hlavní činnosti poskytovatelů patří péče o ochranu serverů před útoky a přetížením. Oba typy napadení jsou nejčastěji cílené na úspěšné weby. Útoky DOS (Denial of Service) a potažmo i DDOS (Distributed Denial of Service) se dělají principem odeslání nezpracovatelného množství požadavků, až server spadne. Útoky mohou mít různou motivaci, buďto jde o snahu web na nějakou dobu vyřadit z provozu, můžou se ale také objevit omylem nebo se někdo pokouší o vydírání správců serveru. Řešení jsou jednoduchá, na druhou stranu velmi drahá, varianty se liší od různých softwarových programů až po hardwarové krabičky, které se serverům předřazují (Janovský, 2023).

## 3.2 Konkurence

Tato kapitola je zaměřena na obdobné aplikace a jazyky, ve kterých lze dosáhnout totožných výsledků. Již existujících řešení je celá řada a preference výběru je především na vývojáři. Může vybírat podle toho, co vyhovuje požadavkům na vývoj, jaké jsou možnosti rozšíření frameworku a flexibilitu jednotlivých částí aplikace.

### 3.2.1 PHP

PHP vzniklo v roce 1995 a od té doby se stalo jedním z nejčastěji používaných programovacích jazyků. Je také velmi dobrou volbou pro začátečníky a běží bez problémů na zařízeních s OS Windows, Linux i MacOS. PHP je také velmi rychlé, je tedy vhodné pro weby s vysokou návštěvností. Nicméně, PHP nemá striktní typování a je možné, že při větších projektech může docházet k neočekávaným chybám (Valuecoders, 2024).

V poslední době získává stále větší popularitu Node.js, založený na JavaScriptu. Node.js je open-source multiplatformní prostředí, které umožňuje používat JavaScript i na serveru, tedy mimo webový prohlížeč. K Node.js patří ještě manažer závislostí NPM, přes který je možné instalovat do webové aplikace rozšíření dle libosti. Node.js je také jednovláknové, a to může při náročnějších operacích, například při dekodování videa, zpomalovat celou stránku. Jeho ekosystém je udržován nezávislými vývojáři, a proto jsou některé knihovny stále v raných fázích vývoje a může docházet k výpadkům nebo být nestabilní (Valuecoders, 2024).

.NET vydala společnost Microsoft 2002 jako nástroj pro vývoj aplikací do OS Windows, na mobilní platformy i webové aplikace. Podobně jako PHP lze i .NET používat v rámci Windows, Linux i MacOS. Je jednoduše škálovatelný, a proto je vhodná volba pro velké organizace. Obsahuje nespočet knihoven, díky kterým je možné postavit finální aplikaci rychle a efektivně. Nevýhodou používání .NET je omezení balíčků pouze od Microsoftu. Pro prostředí .NET je pokročilé a pro začátečníky se může zdát obtížné na naučení (Valuecoders, 2024).

### 3.2.2 Laravel

Laravel se stal nejoblíbenějším PHP frameworkem, zejména pro velmi obsáhlou dokumentaci na webu a stručný popis funkce přímo v kódu frameworku. Má svůj databázový systém Eloquent a šablonovací systém Blade. Dokonce se pořádá každoroční konference LaraCon ve Spojených státech. Jedním příkazem v Composeru se nainstaluje celý počáteční projekt, což usnadní práci s nastavením (Laravel, 2019 cit. podle Štípek, 2019). Více o Laravelu a jeho výhodách je již popsáno v rámci kapitoly 3.1.2.

Dalším, dokonce českým frameworkem je Nette od Davida Grudla. Celý framework je velmi magický a stará se o spoustu nastavení, pokud je dodržena syntaxe.

Nette má také vlastní šablonovací systém Latte, dále vlastní makra, ladící nástroj Tracy (v češtině Laděnka) a flash zprávy. Nette má dokumentaci v českém jazyce, což ocení zejména vývojáři nepříliš zdatní v cizích jazycích. Česká lokalizace je ostatně i nevýhodou tohoto frameworku, její rozšíření je limitováno pouze na Českou republiku a Slovenskou republiku. V zahraničí tento framework není moc známý (Grudl, 2018 cit. podle Štípek, 2019).

Symfony je z řady dalších frameworků konkurujících Laravelu. Má francouzský původ a je zároveň základem dalších frameworků, včetně Laravelu. Také má vlastní šablonovací systém Twig, databázi Doctrine ORM i flash zprávy. Symfony dokáže přidávat pouze nezbytně nutné balíčky, aplikace je pak odlehčená a neobsahuje redundantní kód. Symfony je podobně jako .NET těžký na naučení a je potřeba věnovat frameworku více času (SensioLabs, 2018 cit. podle Štípek, 2019).

### **3.2.3 Databáze**

Existuje několik druhů databází. Obecně se dělí podle toho, co se do databáze ukládá. V rámci této práce je použita relační databáze MySQL od společnosti Oracle (Codecademy, 2024).

PostgreSQL je open-source databázový systém, který neřídí žádná korporace. Je postavený na podobném základu jako MySQL, ačkoliv v rychlosti lehce zaostává a je také méně populární.

Oracle mimo MySQL vlastní i Oracle Database, která není open-source, má ale online prostředí, kde si mohou uživatelé po registraci vyzkoušet, co Oracle DB umí. Toto schéma využívá celá řada bank, protože je dobře zabezpečená a poskytuje kombinaci technologie a integrovaných aplikací pro různé účely.

Společnost Microsoft vlastní SQL Server (zkráceně MSSQL) a stejně jako Oracle je placený. Na rozdíl od předešlé databáze nabízí zdarma na zkoušku level Express, který se může velmi prodražit s postupným rozšiřováním aplikace.

Posledním konkurentem je SQLite, které dokáže uložit celou databázi do jediného souboru. Je vhodný pro nekomplexní aplikace, které ukládají malé množství dat (Codecademy, 2024).

Mezi dalšími, pro tuto práci nepříliš vhodnými typy databází, je například MongoDB, jež ukládá data ve formátu JSON, dále grafová databáze Neo4j, která zobrazuje tabulky jako jednotlivá spojení a popisuje jejich vzájemné vztahy, nebo hierarchické

databáze dodržující strukturu jeden rodič – vícero dětí. Dobrým příkladem hierarchické databáze je Editor registrů v operačním systému Windows (MongoDB, 2024).

### **3.2.4 Git a GitHub**

Ještě před vznikem systému git fungoval systém Subversion. Ten vznikl na přelomu tisíciletí a momentálně je spravován společností Apache. Výhodou je open-source systému a centralizovaný repozitář se všemi změnami. Centralizovanost je také slabina SVN a důvod, proč ji většina vývojářů nepoužívá – změny nelze ukládat lokálně, a tudíž ani není možné změny sdílet mezi vývojáři (Hackbright Academy, 2023).

Ke konceptu git se přibližuje systém Gitolite. Tento systém je postavený na gitu a umožňuje vývojářům sestavit git hosting na vlastním serveru. Na druhou stranu nemá Gitolite žádné uživatelské rozhraní a také nelze na Gitolite sledovat chyby a rozdělovat kód na projekty (StackShare, 2024).

Platforma GitHub má konkurenta s názvem GitLab. Obě platformy mají spoustu funkcí společných, ale i nemálo odlišných. GitLab vyniká v poskytování platformy vše v jednom. Funguje jako bezproblémová platforma pro malé a střední týmy a nabízí spoustu nástrojů pro údržbu repozitáře – řízení projektů, kontroly kódu i automatizace pracovních postupů. Na druhou stranu se GitLab soustřeďuje spíše na Enterprise politiku, tedy společnosti a velmi velké skupiny týmů (CircleCI, 2023).

### **3.2.5 Hosting**

Poskytovatelů hostingu je na českém trhu je nesčetně mnoho. Jak již zmiňoval Janovský (2023), je třeba vybírat dle požadavků webu spíše než dle ceny. Dobrý start zajišťuje služba Endora.cz, kde lze sehnat hosting zdarma a který nabízí 2 GB volného prostoru a 30 GB traffic (Endora, 2024), na druhou stranu je webová aplikace omezená na verzi PHP 7.4, které skončila podpora koncem listopadu 2022 (PHP, 2024b).

Oblíbenou variantou mezi uživateli je hosting od Forpsi, kde zároveň bude umístěna i praktická část této práce. Forpsi nabízí slevu na určité domény, pokud již u nich má zákazník sjednaný jiný hosting a ve výsledku jsou hostiny levnější (Forpsi, 2024).

Velmi dobrým poskytovatelem hostingu je společnost WEDOS, která působí mezinárodně. Službami i cenou je srovnatelná s Forpsi, nicméně WEDOS se zaměřuje spíše na podporu CMS, jako je WordPress a jiné podobné (Wedos, 2024).

## 4 Vlastní práce

Klient v rámci osobní schůzky předložil požadavky na stránku. Celá aplikace bude rozložena do dvou sekcí – stránku pro návštěvníky webu a stránku pro administraci. První zmíněná stránka by dle dohody měla obsahovat:

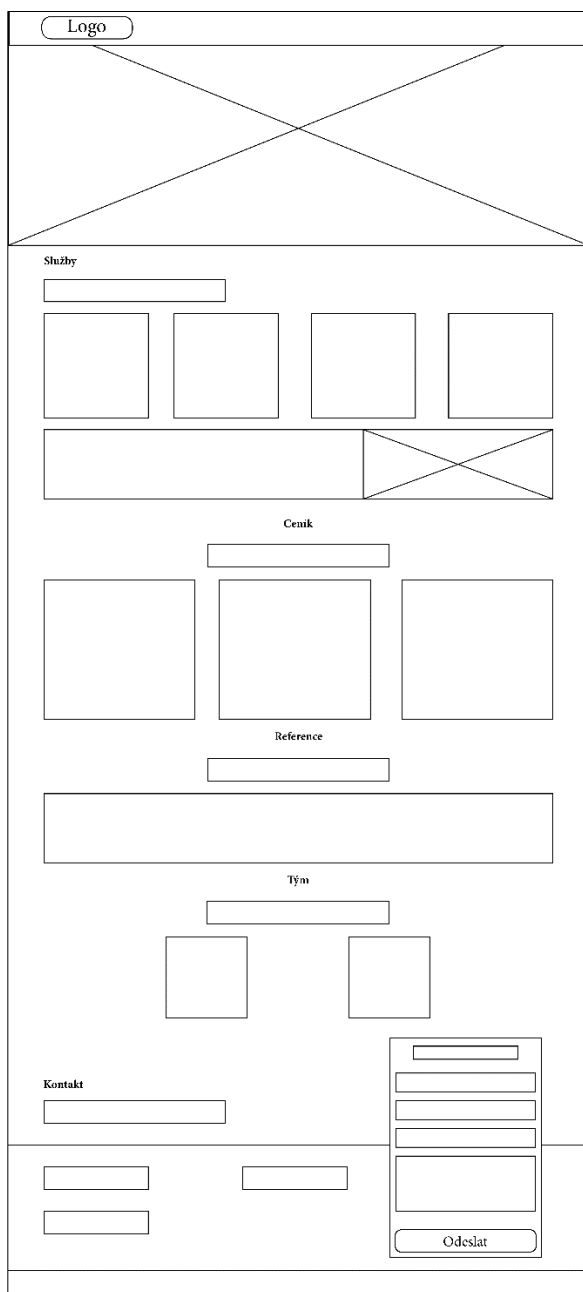
- Úvodní stránku s video prezentací čištění interiérů automobilů
- Seznam služeb
- Představení týmu
- Ceník
- Reference, včetně recenzí ze služby Google Maps
- Adresa podniku s kontaktním formulářem

Administrace pro klienta by měla po dohodě obsahovat:

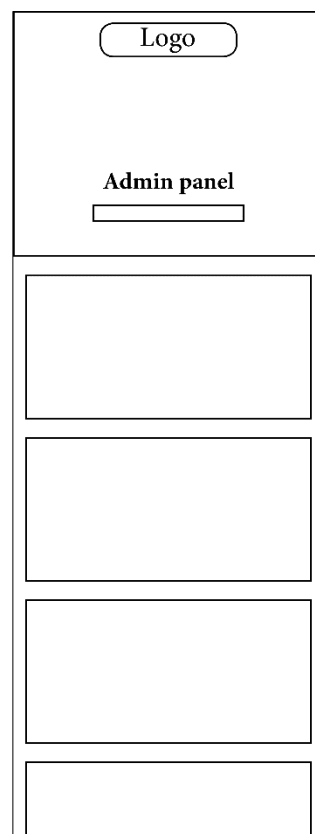
- Domovská stránka
  - Měsíční a roční statistiku tržeb
  - Celkový počet zákazníků
  - Datum příštího objednaného zákazníka
  - Poměr objednaných variant pro lepší budoucí vyvážení
- Stránka se zákazníky
  - Nový zákazník
  - Seznam aktuálních i archivovaných zákazníků
  - Kontaktní údaje
  - Možnost odeslat dotazník spokojenosti
- Stránka recenzí
  - Přehled recenzí a jejich zdroj
  - Statistika počtu a průměrného hodnocení
  - Manuální import ze služby Google Maps
- Stránka faktur
  - Seznam všech faktur
  - Typ faktury
  - Vystavení faktury
  - Generování příjmového dokladu
- Stránka voucherů
  - Generování nových voucherů

- Validace vystavených voucherů
  - Seznam vystavených voucherů
- Stránka kalendáře
  - Měsíční přehled zákazníků
  - Možnost připisovat poznámky pro podřízené
- Automatické funkce
  - Odesílání denního itineráře
  - Přehled tržeb za aktuální měsíc k poslednímu dni v měsíci
  - Notifikace administrátora při výskytu chyby na webu
  - Archivace zákazníka další den po objednaném termínu
  - Vytvoření faktury při archivaci zákazníka
  - Vytvoření faktury při vytvoření nového voucheru
  - Propojení zákazníka s recenzí

V rámci návrhu byly klientovi předloženy drátěné modely neboli *wireframe*, které reprezentují hrubou představu výsledné stránky. Po společné úpravě vypadají předběžné návrhy takto:



Obrázek 14 - Wireframe domovské stránky.



Obrázek 15 - Wireframe admin panelu.

## 4.1 Návrhové vzory a osvědčené postupy

Každý programovací jazyk má své návrhové vzory (anglicky Design patterns), které fungují jako typická řešení běžně se vyskytujících problémů při návrhu softwaru. Jsou ve formě předpřipravených plánů, které lze přizpůsobit tak, aby se vyřešily opakující se problémy návrhu v kódu. Tyto vzory nelze jednoduše zkopírovat a použít, je to spíše koncept řešení daných problémů. Je ale možné řídit se postupy vzoru a implementovat jej

do vlastního kódu. Vzory jsou často zaměňovány s algoritmy, protože oba pojmy popisují řešení problémů. Zatímco algoritmus obsahuje množinu akcí ke zvládnutí překážky, vzor je pouhý popis řešení (Refactoring Guru, 2024).

Mezi návrhovými vzory použitými ve finální práci jsou Factory (továrna), Abstract Factory (abstraktní továrna), Builder (stavitel), Facade (fasáda) a Command (příkaz). Jejich příklady a vysvětlení budou uvedeny v praktické části práce.

Dodržování osvědčených postupů (anglicky Best practices) je klíčová vlastnost programování, která udržuje kód efektivní a spolehlivý. Pro PHP platí několik postupů:

- Dodržovat PSR (PHP Standards Recommendations)
- Validovat uživatelský vstup
- Nepoužívat globální proměnné
- Využívat *namespaces*
- Odchyťávat výjimky
- Používat MVC architekturu
- Využívat předem definované příkazy pro práci s databází
- Dokumentovat a optimalizovat kód

Většinu postupů lze pochopit již z definice, vybrané PSR standardy jsou vysvětlené v dalších odstavcích.

PSR standardů je k datu 7.3.2024 celkem dvacet tři, z nichž ne všechny jsou již platné. Čísly se PSR- $n$ , přičemž  $n \leq 0$ . Mezi nejznámější standardy patří PSR-12, který rozšiřuje stylizaci psaní kódu a zároveň nahrazuje starou konvenci PSR-2, PSR-20 pracuje s časem a Unix časovou známkou, a v neposlední řadě například PSR-4 udává, jak pracovat s manažery komponent (Composer aj.). Většina těchto standardů je ve výsledné práci taktéž implementována.

## 4.2 Implementace aplikace

Jak již bylo v teoretické části definováno a popsáno, bude pro webovou aplikaci použit framework Laravel. Tento framework se dá nainstalovat skrz Composer manažer příkazem `laravel new <nazev-projektu>`. Pokud příkaz vyhazuje nějakou chybu, je možné, že Composer nezná tento příkaz a lze jej nastavit jiným příkazem `composer global require laravel/installer`. Po zadání druhého příkazu se nastaví





Po stáhnutí zdrojového kódu frameworku se instalátor doptává na výchozí databázový systém:

```
INFO Application key set successfully.

Which database will your application use? [MySQL]:
[mysql ] MySQL
[mariadb] MariaDB
[pgsql ] PostgreSQL
[sqlite ] SQLite
[sqlsrv ] SQL Server
>

INFO Application ready in [test]. You can start your local development using:
→ cd test
→ php artisan serve

New to Laravel? Check out our bootcamp and documentation. Build something amazing!
PS C:\Users\Martin\Desktop\GitHub>
```

Obrázek 17 - Dokončení CLI instalace.

Laravel je nastavený pro následné použití takto:

1. Verze 11.0.7 fungující na PHP 8.2.12
2. Bez použití startovního balíčku
3. Testování přes PHPUnit
4. Databázový systém MySQL

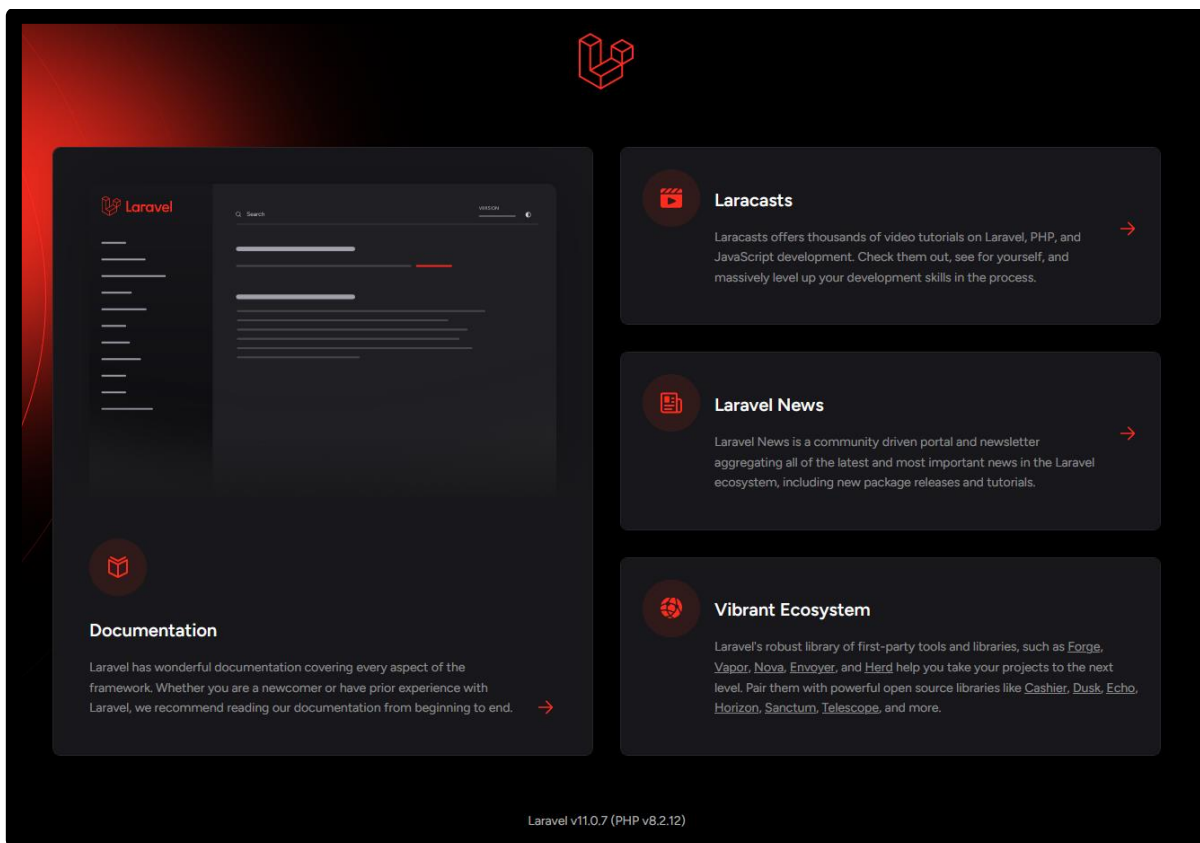
Nyní lze otevřít připravený projekt ve zvoleném editoru a v terminálu (příkazové řádce) zadat dle Figure 14 příkaz

```
php artisan serve
```

pro spuštění lokálního serveru. Tento příkaz je obdobou příkazu pro nativní PHP

```
php -S localhost:8000 -t public/
```

Pokud vše funguje, jak má, měla by být na webové adrese <https://localhost:8000> zobrazena tato stránka:



Obrázek 18 - Úvodní stránka Laravelu verze 11.

Laravel používá šablonovací systém Laravel Blade, nicméně pro tuto práci bude tento systém nahrazen systémem Latte od frameworku Nette. Pro použití Nette Latte je potřeba připravit Composer balíček, vystavit jej přes službu *Packagist* a přidat jako Composer závislost.

Balíček je připravený pod názvem `martinoak/laravel-latte` a obsahuje poskytovatele *ServiceProvider.php*, který nastavuje Latte engine jako výchozí šablonovací systém po boku Laravel Blade. Po instalaci příkazem

```
composer require martinoak/laravel-latte
```

se nainstaluje nejnovější kompatibilní verze, aktuálně 2.1. Spolu s tímto balíčkem se instalují dále všechny další závislosti, které instalovaný balíček vyžaduje. Pokud se instaluje komplexnější závislost, druhotných závislostí přibývá exponenciálně.

```
Package operations: 2 installs, 0 updates, 0 removals
- Installing latte/latte (v3.0.13): Extracting archive
- Installing martinoak/laravel-latte (2.1): Extracting archive
```

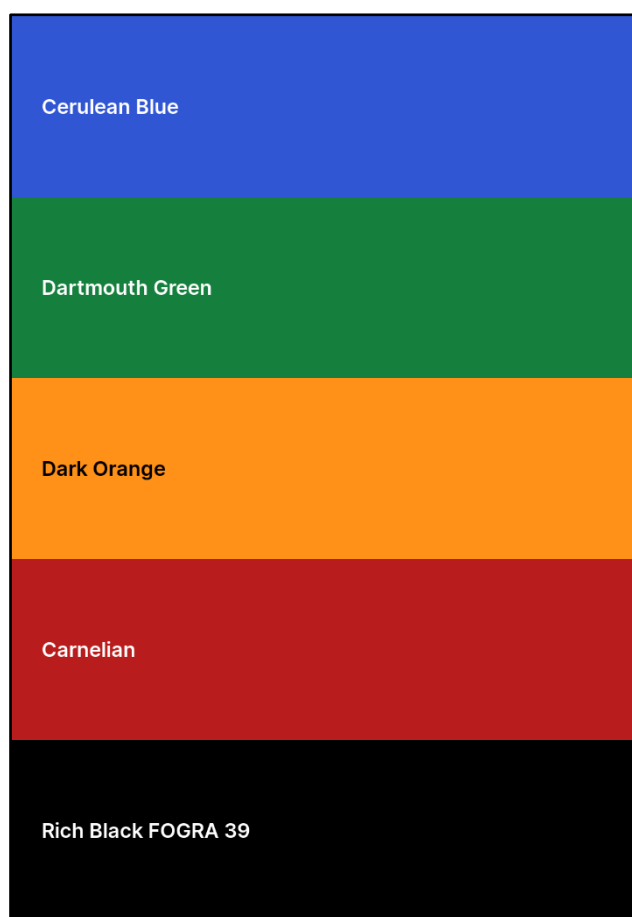
Obrázek 19 - Instalace závislosti přes Composer.

Pokud composer nenajde vhodnou verzi, informuje o nemožnosti instalace a balíček musí být upraven tak, aby byl kompatibilní.

V tento moment je možné používat jak soubory Laravel Blade s příponou *.blade.php*, tak soubory Nette Latte s příponou *.latte*.

#### 4.2.1 Barevné schéma

Pro zajištění maximálního kontrastu a zvýšení přístupnosti byly po domluvě s grafikem navrhnuty tyto barvy – primární modrá, do které je veden celý obsah webu, zelená pro úspěch, oranžová pro varování, červená pro chybu a černá jako doplňková barva:



Obrázek 20 - Barevné schéma webu.

#### 4.2.2 Obecná nastavení

Verze PHP 8.1 přinesla *Enumerations* neboli výčty, které omezují možné hodnoty pro datový typ *Enum*. Tento datový typ je vhodné použít, pokud je potřeba nastavit omezené množství možností. *Enum* má v sobě dvě vlastnosti pro získání dat – *name* a *value*. Zároveň lze využít statické metody `cases()`, která do PHP pole předá veškeré *Enum* možnosti. Tato metoda je použita i ve Figure 34. V této práci byly použity celkem

dvě *Enum* třídy pro omezení tří variant čištění a pěti druhů faktur. První výčetový typ je pro ukázkou zobrazena níže.

```
<?php

namespace App\Enums;

enum CleaningTypes: string
{
    case START = 'Lehký start';
    case MIDDLE = 'Zlatá střední cesta';
    case DELUXE = 'Deluxe';
}
```

Figure 30 - Datový typ Enumerations v PHP.

### 4.2.3 Domovská stránka

Pro nastavení zobrazení jiné domovské stránky je třeba založit nový soubor *home.latte* a upravit soubor *routes/web.php* tak, aby se o domovskou stránku staral nový, vhodně pojmenovaný *Controller*. Ten se založí příkazem

```
php artisan make:controller HomepageController
```

Figure 31 - Vytvoření controlleru pro domovskou stránku.

a do souboru *routes/web.php* se importuje napsáním

```
use App\Http\Controllers\HomepageController;
```

Figure 32 - Import Controlleru do souboru s cestami.

V souboru *web.php* přibude nová cesta, která specifikuje controller a funkci uvnitř něj. Dodatečný parametr *name* je velmi důležitý údaj každé cesty, jelikož se na každou jednotlivou cestu dá odkazovat odkudkoliv z frameworku. Název cesty by měl být unikátní a každá cesta by jej měla mít nastavená.

```
Route::get('/', [HomepageController::class, 'index'])->name('homepage');
```

Figure 33 - Cesta obsluhující domovskou stránku.

Všechny controllery dědí od abstraktního controlleru a jsou umístěny v adresáři *app/Http/Controllers*. Nově vytvořený controller by měl dle Figure 33 obsahovat minimálně metodu *index()* a v ní definovaný postup, co má controller při vstupu uživatele na domovskou stránku provést. Jelikož chceme zobrazit domovskou stránku, metoda *index()* by měla vracet funkci *view()* s parametrem cesty k Latte souboru *home.latte*. Všechny stránky musí být uloženy v adresáři *resources/views* a tento adresář

lze dále větvit, například založením adresáře *admin* nebo *partials*. Metoda `index()` ještě před zobrazením domovské stránky stahuje recenze z databáze, ceník z Enumerate souboru a data předává do domovské stránky:

```
public function index(): View
{
    $pricelist = [];

    foreach (CleaningTypes::cases() as $case) {
        $pricelist[$case->value] = [
            'price' => $case->getVariantPrice(),
            'description' => $case->getVariantDescription(),
        ];
    }

    return view('home', [
        'feedbacks' => Feedback::all(),
        'pricelist' => $pricelist,
    ]);
}
```

Figure 34 - Metoda `index()` na zobrazení domovské stránky.

Pro domovskou stránku byl použit framework Bootstrap, jelikož tento framework je velmi rychlý a je často využíván i jinými weby. Protože je velká pravděpodobnost, že tento framework je v rámci CDN (Content Delivery Network) uložen v mezipaměti prohlížeče, je načítání domovské stránky o to rychlejší.

Celkový koncept domovské stránky je rozdělený do sedmi částí – úvodní video, rozpis služeb, sekce O nás, ceník, reference, tým a kontakt. Všechny snímky jednotlivých částí jsou uvedené v sekci Přílohy, konkrétně Příloha 1 až Příloha 7.

Většina údajů na stránce je dynamická, data se načítají z konfiguračního souboru skrz Laravel pomocníka `config()` a s parametrem názvu souboru. Výpis členů týmu tak může vypadat zhruba takto:

```
{foreach config('web.team') as $worker}
    <!-- ... -->
{/foreach}
```

Figure 35 - Pomocník `config()` pro výpis členů týmu.

Na domovské stránce je použit také framework Vue.js, který se stará o aktualizaci obsahu webu bez nutnosti obnovy stránky. Tento framework byl použit pro kontaktní formulář ve dvou případech – nastavit zvolenou variantu do příslušného pole, pokud zákazník klikne na tlačítko volby varianty v ceníku; a při hlášece o úspěšně odeslaném e-mailu:

```
<a href="#kontakt" v-on:click="setVariant" data-variant="{ $item }">
  To беру!
</a>
```

Figure 36 - Nastavení varianty po kliknutí na variantu ceníku.

```
<input
  n:syntax="off"
  type="hidden"
  name="variant"
  :value="variant"
  placeholder=""
/>
```

Figure 37 - Značka <input> s nastavením varianty.

```
<script>
  (function() {
    let vue = new Vue({
      el: '#vue-control',
      data: {
        variant: null,
      },
      methods: {
        setVariant: function (e) {
          this.variant = e.target.dataset.variant;
        }
      }
    });
  })();
</script>
```

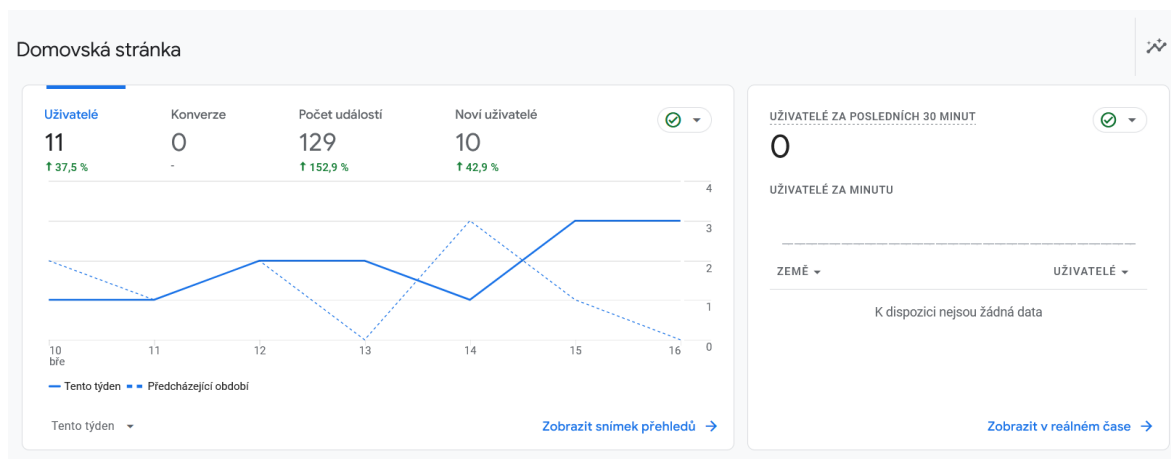
Figure 38 - Framework Vue.js a předání varianty do formuláře.

Do úvodní stránky byl implementován nástroj GA (Google Analytics), který hlídá návštěvnost a data o prohlížení předává do GA portálu. GA fungují na principu GTM (Google Tag Manager), který pro každou stránku vygeneruje unikátní kód a při návštěvě webu se tento skript zavolá a předají se informace do nástroje.

```
<script src="https://googletagmanager.com/gtag/js?id=G-CH9KTSQLV2"></script>
```

Figure 39 - Google tag pro sledování návštěvnosti webu.

Ve výsledku může vypadat přehled o návštěvnosti za minulý týden z nástroje Google Analytics takto:



Obrázek 21 - Výstup dat z Google Analytics.

#### 4.2.4 Nastavení proměnných prostředí

Proměnné prostředí se konfiguruje v souboru `.env` v kořenovém adresáři. Tento soubor obsahuje citlivá data a neměl by být ani verzován, ani přístupný zvenčí na produkčním serveru. Zde se nastavuje jak samotná aplikace, tak i logování, databáze, mezipaměť, Redis, e-mailový klient a AWS (Amazon Web Services).

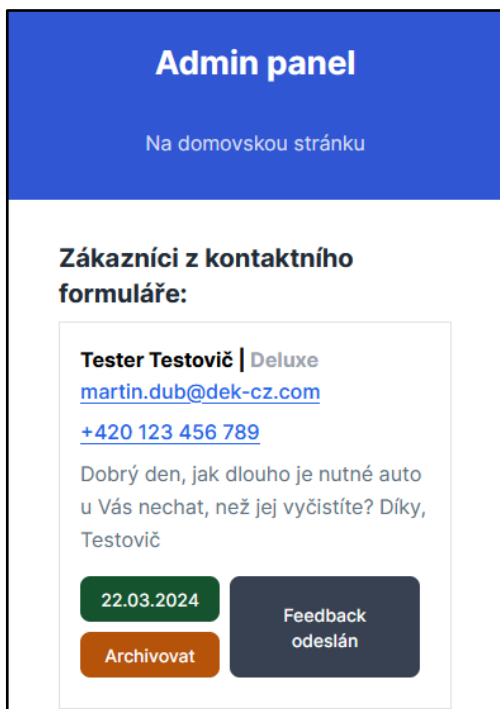
Pokud je potřeba verzovat soubor proměnných prostředí, lze buďto využít soubor `.env.example` nebo založit nový adresář `dist/` s proměnnými prostředí pro jednotlivá vývojová prostředí. Tímto způsobem jde kontrolovat a potažmo i přepínat mezi jednotlivými prostředími, nicméně je potřeba stále myslet na vynechání hesel a dalších citlivých nebo jinak zneužitelných údajů.

#### 4.2.5 Sekce s administrací

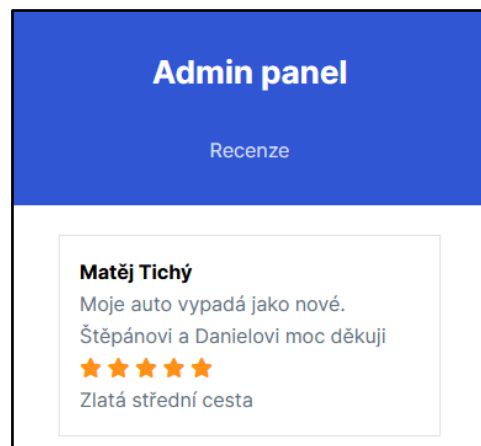
Účelem administrace je zařídit klientovi, aby byl schopen spravovat aplikaci bez pomoci programátora. Klient by měl být schopný upravovat data na domovské stránce a mít přehled o svém podnikání – recenze, archiv zákazníků, správa voucherů a přehled faktur.

Za celou dobu prošla administrace třikrát kompletním refaktoringem, který snížil vytížení serveru a zrychlil načítání aplikace. První verze administrace byla, stejně jako domovská stránka, jedna statická stránka a jednotlivé sekce byly rozděleny vertikálně. Chyběla statistika recenzí, faktury se generovaly pouze na vyžádání a vzhled byl strohý a uzpůsobený pouze pro mobilní telefony. Náhledy první verze zobrazují následující obrázky:

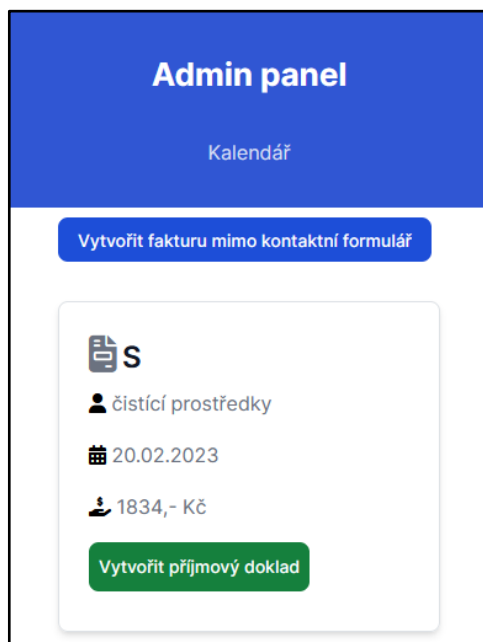




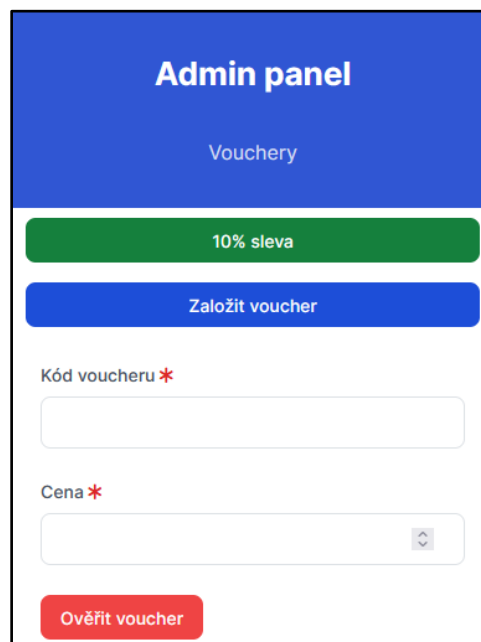
Obrázek 22 - Admin panel, verze 1.



Obrázek 23 - Admin panel, recenze, verze 1.



Obrázek 24 - Admin panel, faktury, verze 1.



Obrázek 25 - Admin panel, vouchery, verze 1.

V rámci prvního refaktoringu a druhé verze admin panelu byl proveden i zásah do grafické stránky. Bylo změněno logo a vytvořilo se barevné schéma pro lepší

sjednocení barev a zajištění dostatečného kontrastu. Změny jsou k prohlédnutí na obrázcích níže:



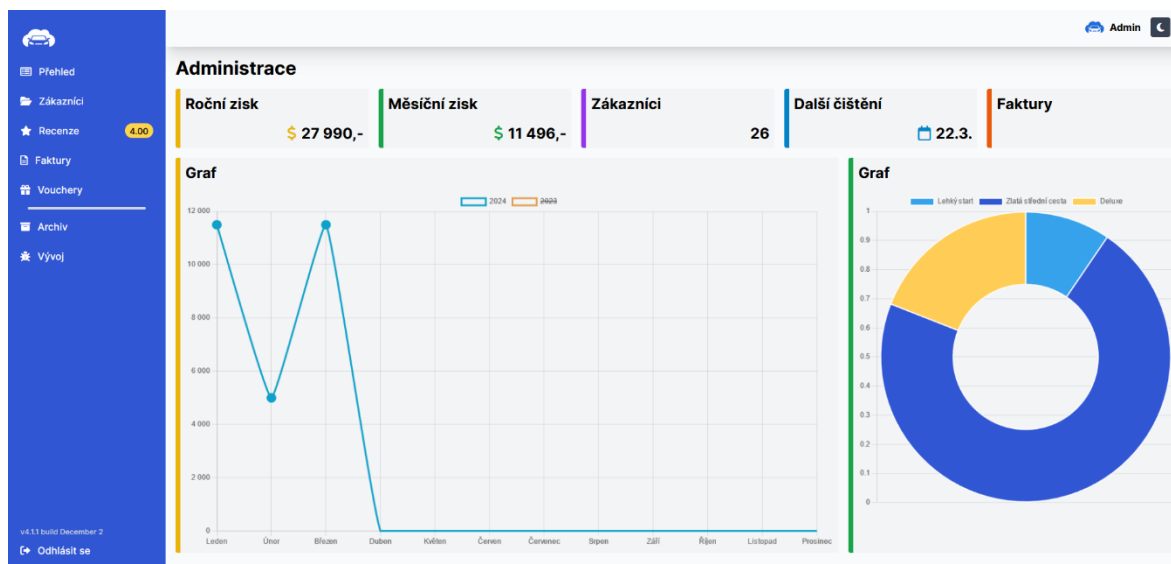
Obrázek 26 - Staré logo.



Obrázek 27 - Nové logo.

Druhá verze administrace byla přístupná již z desktopových zařízení, ale stále vyvíjena principem mobile-first. V rámci tohoto přechodu byla upravena i databáze, která nyní používala nativní Eloquent ORM namísto přístupu přes fasádu a napřímo. Tento postup koliduje s principem fasády jakožto návrhového vzoru, na druhou stranu je fasáda vhodná pro sjednocený přístup do databáze, který v tomto případě není potřeba. Každá entita je provázaná cizím klíčem, např. zákazník a feedback, zákazník a faktura, voucher a faktura apod. Propojením dvou tabulek zmizí duplicita dat a zároveň se hlídá i integrita dat. Není potřeba tedy ukládat jméno zákazníka v tabulce *customer*, *feedback*, případně ještě *voucher* a *invoice*. K získání dat tak může být použita tranzitivita.

Hlavní změnou ve druhé verzi je postranní panel, který slouží jako navigace mezi jednotlivými stránkami administrace. Na domovské stránce jsou připravené grafy s údaji o tržbách za jednotlivé měsíce a poměr objednávaných variant, včetně počítadel zisků, případně ztrát za aktuální měsíc, aktuální rok a celkové období. Dále byl implementován přepínač světlého a tmavého režimu a s tím byly upraveny i jednotlivé stránky.



Obrázek 28 - Admin panel, verze 2.

V této verzi byl implementován skript na získávání recenzí z Google Map. Pro správné fungování skriptu musel být nastaven API klíč pro autorizovaný přístup k datům a připravený skript na transformování dat pro uložení do databáze. Celý postup je popsán podrobněji v sekci 4.2.5.3.

Do webu byl dále nasazen automatický časovač *cron*. Všechny *crony* se soustřeďují v souboru *crontab* a zapisují se sekvencí pěti čísel nebo znaků, dále uživatelem s právem spuštění, a nakonec cesta ke skriptu.

Na prvním místě je zapisuje minuta spuštění, na druhém místě se zapisuje hodina, na třetím místě den, čtvrtý znak značil měsíc a na posledním místě den v týdnu (počítaný od neděle). Pokud na jakémkoli místě je umístěna místo čísla hvězdička \*, funguje tento znak jako substitute za všechna čísla.

Nejkratší doba mezi jednotlivými spuštěními je jedna minuta. Pro kratší dobu než je 60 vteřin hrozí, že se minulý *cron* nestihne ukončit a celá automatizace by přestala fungovat. Opačný případ, tedy nejdelší možná pauza mezi spuštěními, je jeden rok. Oba případy včetně zajímavosti jsou vypsány níže.

```
* * * * * root /usr/local/bin/backup.sh
```

Figure 40 - Cron spouštěný každou minutu.

```
37 13 1 1 * root /usr/local/bin/backup.sh
```

Figure 41 - Cron spouštěný jednou ročně 1. ledna ve 13 hodin 37 minut.

```
0 0,12 1 */2 * root /usr/local/bin/backup.sh
```

Figure 42 - Cron spouštěný v půlnoci a v poledne 1. den každý druhý měsíc v roce.

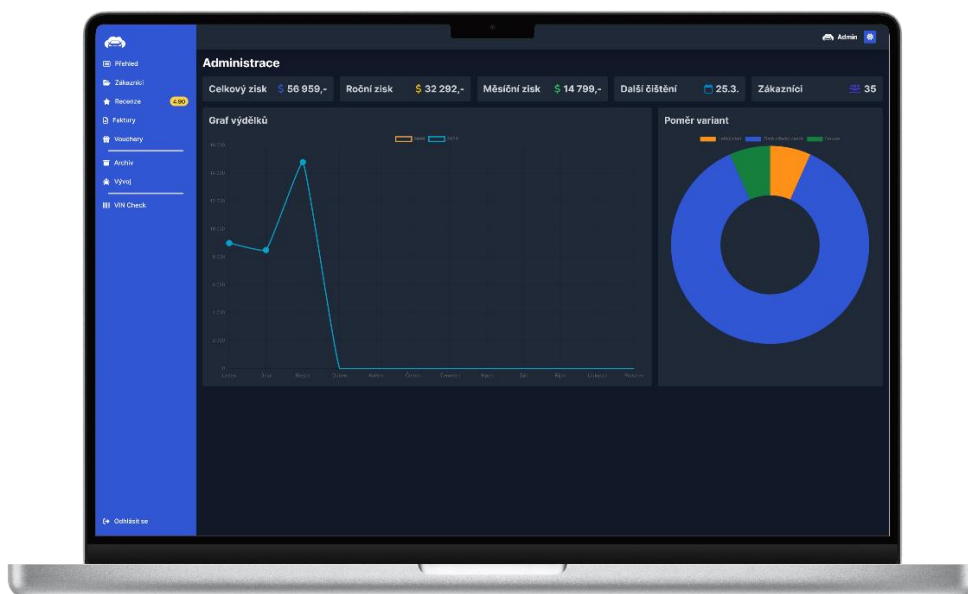
*Crony* spouští definované automatizace uvedené v úvodu kapitoly 4. Pro tyto automatizace byly navrženy e-mailové šablony, které kopírují designový návrh webu a předávají nutné informace.

Všechny *crony* ve webové aplikaci jsou spouštěny s pomocí Artisan CLI a Laravel Command rozhraním. Toto nastavení umožňuje spouštět automatizace i ručně pomocí URL. Každý příkaz má nastavený zápis do Cron logu v úložišti frameworku, aby se monitoroval průběh automatizace, jelikož příkaz sám o sobě nevrací žádná data. Cron log spolu s Laravel logem jsou k dispozici v administraci, aby se vývojář, v případě problému, nemusel připojovat na produkční server skrz protokol SSH (Secure Shell).

Při prezentaci druhé verze klientovi byla domluvena další změna administrace. Hlavním důvodem bylo špatné zobrazení na telefonu, redundantní barevná označení a složité postupy při pracování s webovou administrací. Zároveň se s předěláním administrace uskutečnilo i pár úprav na domovské stránce, zejména zlepšení zobrazení na mobilních zařízeních.

#### 4.2.5.1 Stránka s přehledem

Po úspěšné autentizaci do administrace je klient přesměrován na stránku s přehledem, kde je ihned vidět nejdůležitější statistika. Klient má k dispozici tři náhledy *cashflow*, a to měsíční, roční a celkový přehled zisků či ztrát. Tato statistika je závislá na správné historii faktur, ze kterých se sumy počítají. Logiku počítání obstarává *AdminController* včetně předání dat o jednotlivých měsících do šablony, s pomocí nichž se generují grafy z knihovny Chart.js verze 4.3.0.



Obrázek 29 - Admin stránka, přehled.



Obrázek 30 - Admin stránka, přehled, mobilní zobrazení.

#### 4.2.5.2 Zákazníci

Zákazníci se přidávají buď automaticky z kontaktního formuláře, nebo manuálně v administraci. Varianta může být doplněna dodatečně při výběru termínu čištění. Zákazník může být editován po celou dobu až do jeho archivace. Pokud je zákazník archivován, je možné pouze odeslat recenzi (pokud má nastavenou e-mailovou adresu). Pokud nemá vybranou variantu, je potřeba ji doplnit. Při archivaci probíhá automatizace vytvoření faktury s údaji zákazníka.

Pro práci se zákazníky byl zvolen speciální typ controlleru – Laravel jej označuje jako *Resource Controller*. Tento controller sestává ze sedmi předdefinovaných metod `index()`, `create()`, `store()`, `show()`, `edit()`, `update()` a `destroy()`, které usnadňují práci s daným zdrojem (v tomto případě zákazníky). Tento controller se založí příkazem

```
php artisan make:controller CustomersController --resource
```

Figure 43 - Vytvoření controlleru pro práci se zdrojem.

a v souboru `routes/web.php` lze nadefinovat omezení pro používání nebo vyjmutí jednotlivých funkcí. K tomu lze využít metodu `only()`, případně `except()`. Se zdrojem zákazníci je potřeba dělat vše kromě metody `show()`, protože zákazníci jsou vypsaní v metodě `index()`.

```
Route::resource('customers', CustomersController::class)->except('show');
```

Figure 44 - Definice controlleru pro práci se zdrojem.

Aby se zákazníci nemuseli v šabloně filtrovat podle archivace, je tato logika vyřešena již v controlleru a do šablony se předávají již seřazení zákazníci.

```
public function index(): View
{
    return view('admin.customers.index', [
        'customers' => Customer::where('archived', 0)
            ->get(),
        'archived' => Customer::where('archived', 1)
            ->orderBy('id', 'desc')
            ->get(),
    ]);
}
```

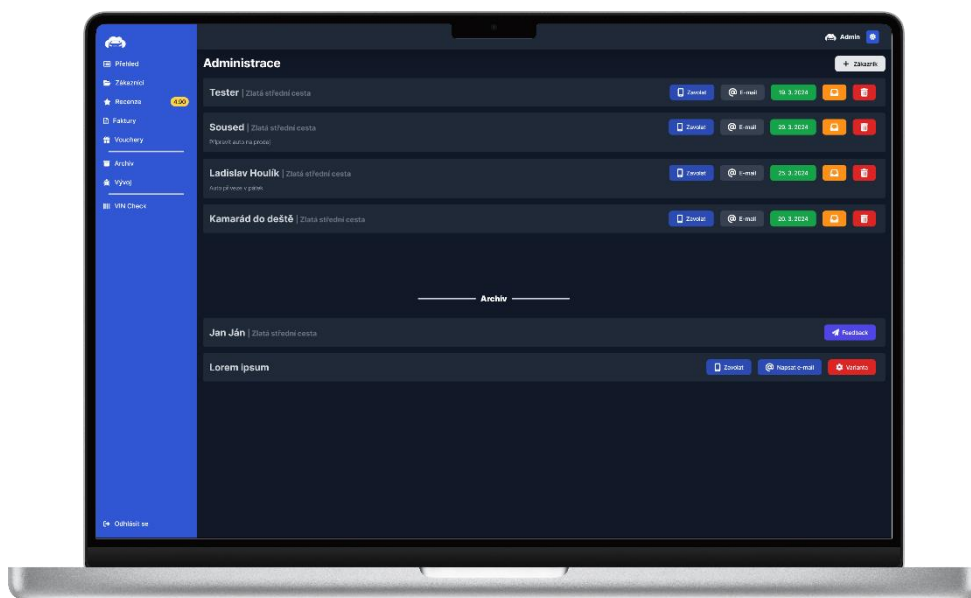
Figure 45 - Metoda index() pro zobrazení zákazníků.

Některé metody mají různé http hlavičky, kromě klasických GET a POST hlaviček používá tento controller i hlavičky PUT/PATCH a DELETE. Formulář v HTML přijímá v atributu `method` pouze hodnoty GET a POST. Laravel pro tyto účely vytvořil helper funkci:

```
method_field('PUT')
```

Figure 46 - Laravel helper pro nastavení HTTP hlavičky.

Při špatném nastavení HTTP hlavičky totiž formulář nefunguje správně a Laravel zobrazí chybu.



Obrázek 31 - Admin stránka, zákazníci.



Obrázek 32 - Admin stránka, zákazníci, mobilní přehled.

#### 4.2.5.3 Recenze

Do sekce recenzí byl implementován skript, který importuje recenze z Google Map přes PHP knihovnu Guzzle. V tomto případě bylo potřeba nastavit vývojářský účet v Google administraci. Skript se nespouští automaticky, je nutné jej pustit kliknutím na tlačítko *Import*. Pokud je recenze importována, je příslušně označena v databázi a na domovské stránce se zobrazuje logo Google Map. Pro klienta je zobrazen celkový počet recenzí, které rotují na domovské stránce, a průměrné hodnocení. Druhý ukazatel se taktéž zobrazuje v postranním panelu. Kód níže zobrazuje metodu získání recenzí. Bylo potřeba nastavit unikátní ID místa, API klíč pro oprávnění vytvářet požadavky, možnost nepřekládat cizojazyčné recenze a výčet kýžených polí.

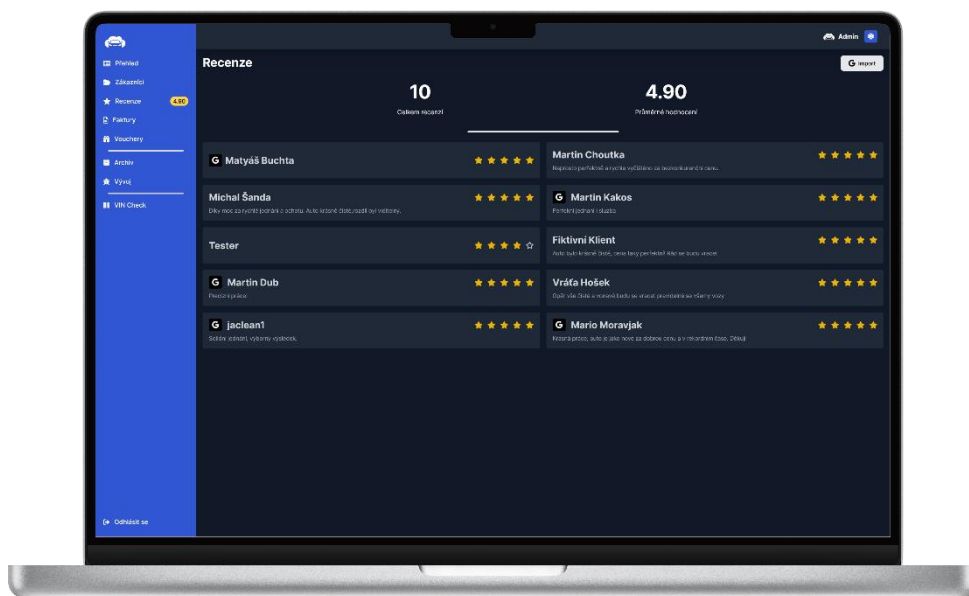
```

public function getGoogleMapsReviews(): array
{
    $response = $this->guzzleClient->get(
        'https://maps.googleapis.com/maps/api/place/details/json', [
            'query' => [
                'placeid' => 'ChIJ56G-hWSJDEcREctIevUbpDo',
                'key' => env('GOOGLE_MAPS_API_KEY'),
                'reviews_no_translations' => true,
                'fields' => 'review'
            ],
        ],
    );

    return json_decode($response->getBody()->getContents(), true);
}

```

Figure 47 - Kód knihovny Guzzle pro import recenzí z platformy Google Mapy.



Obrázek 33 - Admin stránka, recenze.



Obrázek 34 - Admin stránka, recenze, mobilní zobrazení.

#### 4.2.5.4 Faktury

Stránka faktur se v šabloně stránky generuje do tabulky. V tabulce je několik sloupců – jméno zákazníka, typ faktury (pro toto byl využit opět datový typ *Enumerations*), datum na faktuře, cena a tlačítka akcí. Každá faktura se dá stáhnout, upravit (například v případě individuální změny ceny) a zobrazit nebo stáhnout. Pro lepší čitelnost v tabulce jsou řádky v jiných odstínech. Na mobilním zařízení je zobrazení tabulky oseknuté a klient může v případě nutnosti přesunout tabulku přejetím prstu doprava pro zobrazení zbytku informací.



Na práci s fakturami byl vytvořen také resource controller, v tomto případě nebude využita metoda `destroy()`, protože se faktura nebude nikdy mazat. Metoda `index()` zobrazuje úvodní stránku se všemi fakturami a metoda `show()` jednotlivou fakturu. Každá faktura je *de facto* příjmový doklad, který se generuje z dat v databázi a klient může onen doklad nechat vytisknout či dále sdílet.

```
public function show(string $id): BinaryFileResponse
{
    $data = Invoice::find($id);
    // Šablona
    $img = imagecreatefromjpeg(public_path('invoice/template.jpg'));
    // Barva
    $c = imagecolorallocate($image, 0, 0, 0);
    // Font
    $f = public_path('fonts/Rubik.ttf');

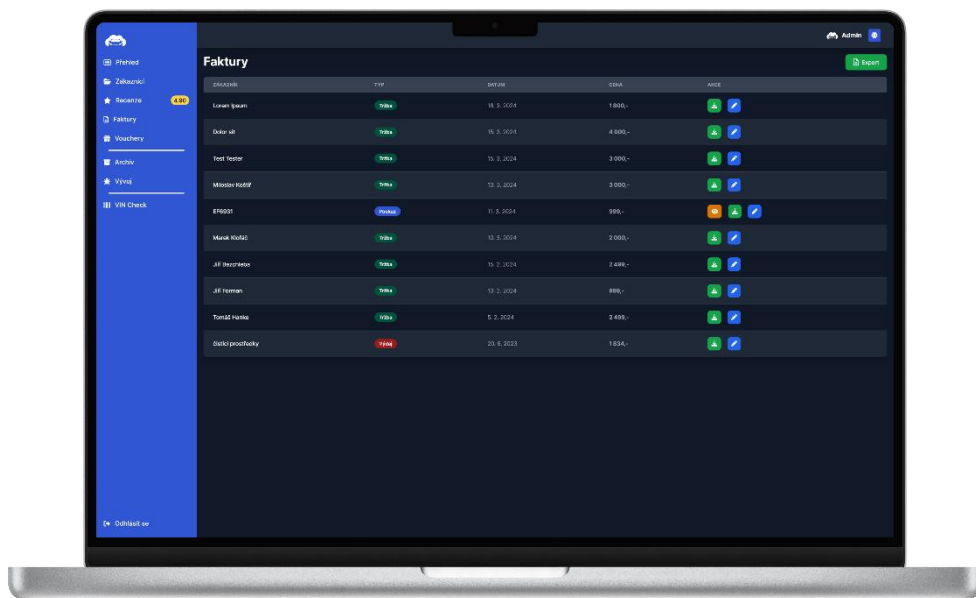
    imagettftext($img, 20, 0, 800, 100, $c, $f, utf8_decode($data->id));
    imagettftext($img, 20, 0, 500, 150, $c, $f, $data->date->format('d. n. '));
    imagettftext($img, 20, 0, 800, 150, $c, $f, $data->date->format('Y'));
    imagettftext($img, 20, 0, 250, 200, $c, $f, utf8_decode($data->name));
    imagettftext($img, 20, 0, 250, 340, $c, $f, 'Čištění interiéru auta');
    imagettftext($img, 20, 0, 200, 410, $c, $f, $data->price);
    imagettftext($img, 20, 0, 500, 540, $c, $f, 'Štěpán Dub, '.date('d. n. Y'));

    imagepng($image, storage_path('app/public/invoice/'.$id.'.png'));

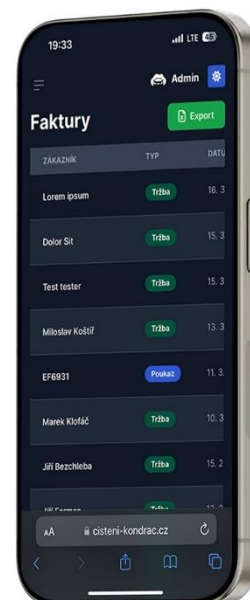
    return response()->
        download(storage_path('app/public/invoice/'.$id.'.png'));
}
```

Figure 48 - Generování příjmového dokladu.

Výstupem kódu je pak vygenerovaný příjmový doklad ve formátu PNG. Doklad se dá vytvořit i manuálně, stačí zadat do příslušného formuláře název dokladu (jméno zákazníka), typ dokladu (celkem 5 – tržba, poukaz, výdaj, mzda, ostatní), částka a datum na faktuře. Tento způsob zapisování umožňuje klientovi mít přehled o finanční bilanci, pokud nemá některé zákazníky registrované přes kontaktní formulář.



Obrázek 35 - Admin stránka, faktury.



Obrázek 36 - Admin stránka, faktury, mobilní zobrazení.

#### 4.2.5.5 Vouchery

Stránka s poukazy je rozdělena na tři části, a to generování poukazu, generování desetiprocentní slevy na další čištění a ověření vouchery. Rozdíl mezi oběma vouchery je pouze v počátečním písmenu. Zatímco oba typy jsou šest znaků dlouhé pro lepší použití, desetiprocentní voucher začíná na písmeno *x*. Oba typy jsou generované jako prvních pět (v případě slevového vouchery) nebo šest (v případě poukazu) znaků jednosměrné šifry MD5 aktuálního času. Šifra MD5 byla sice již překonána metodou hash kolize a nedoporučuje se používat pro ukládání hesel. Pro generování náhodných znaků do kódů vouchery je šifra více než dostačující. Pravděpodobnost, že dojde ke kolizi (budou vygenerovány dva totožné kódy vouchery) je  $16^5$  nebo  $1:1048576$ , respektive  $16^6$  nebo  $1:16777216$ .

```
// Desetiprocentní voucher
$hash = 'x'.strtoupper(substr(md5(rand()), 0, 5));

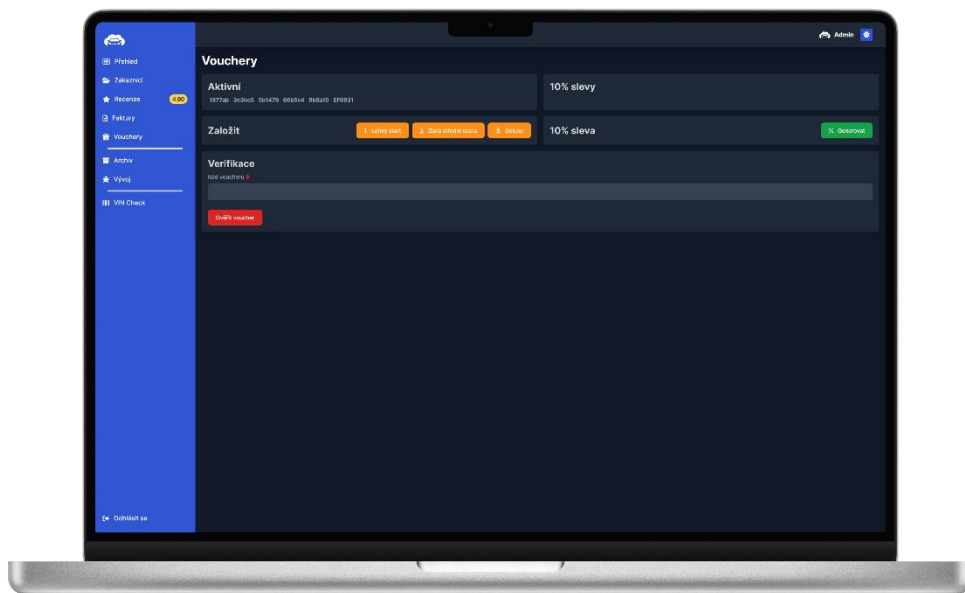
// Poukaz na čištění
$hash = strtoupper(substr(md5(rand()), 0, 6));
```

Figure 49 - Generování hashů pro jednotlivé vouchery.

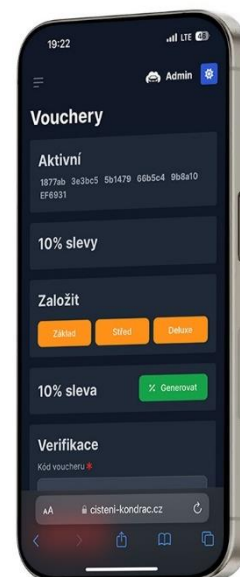
Akceptovatelné nebo nearchivované vouchery se získávají skrz Eloquent ORM model `Voucher` a předávají se do šablony do příslušných polí:

```
Voucher::where(['accepted' => 0, 'expired' => 0])->get();
```

Figure 50 - Získávání aktuálních voucherů.



Obrázek 37 - Admin stránka, vouchery.



Obrázek 38 - Admin stránka, vouchery, mobilní zobrazení.

#### 4.2.5.6 Ostatní

Každý HTTP požadavek je validován na straně klienta i serveru. Pokud by jedno nebo druhé chybělo, hrozí bezpečnostní riziko, např. SQL Injection. Pro validaci na klientově straně stačí zvolit správný typ u formulářových polí, případně přidat atribut `required` nebo skrz regulární výrazy omezit `pattern`. Správně ošetřený vstup na straně klienta může vypadat zhruba takto:

```
<input type="number" inputmode="numeric" pattern="[0-9]*" required>
```

Figure 51 - Validace na straně klienta.

Co se týče validace na straně serveru, ta zamezuje manipulaci se vstupy a ověřuje data ještě jednou. Kontaktní formulář na domovské má takovouto serverovou validaci:

```

$validator = Validator::validate($request->all(), [
    'name' => 'required',
    'email' => 'required|email',
    'telephone' => 'required|numeric|min:9',
], [
    'name.required' => 'Jméno je povinné pole.',
    'email.required' => 'Email je povinné pole.',
    'email.email' => 'Email musí být ve správném formátu.',
    'telephone.required' => 'Telefon je povinné pole.',
    'telephone.numeric' => 'Telefon musí být číslo.',
    'telephone.min' => 'Telefon musí mít alespoň 9 znaků.',
]);

if ($validator->fails()) {
    return back()->withErrors($validator)->withInput();
} else {
    Customer::create($request->all());
    Mail::to('stepan@cisteni-kondrac.cz')
        ->send(new FormEmail($request->all()));

    return to_route('homepage')->with('success', 'Email odeslán!');
}

```

*Figure 52 - Validace na straně serveru.*

Třída `Validator` vezme všechny požadavky z formuláře, zkontroluje je dle zadaných pravidel a v případě nesplnění jedné nebo vícero podmínek se uživateli zobrazí chybové hlášky ve formuláři. Zobrazení chyb probíhá metodou `withErrors()` a staré hodnoty v jednotlivých polích zůstanou zachovány, pokud je přidána metoda `withInput()`. Formulář musí mít nastavený atribut `value` s Laravel pomocníkem `old()`. Ten funguje se dvěma parametry, v prvním parametru má název vstupu (měl by souhlasit s hodnotou v atributu `name`) a v druhém parametru výchozí hodnotu, která se použije, pokud je metoda `old()` prázdná.

Po úspěšné validaci se odesílá e-mail skrz fasádu `Mail`, kde se nastaví příjemce a třída `Mailable`, která upravuje další parametry, jako je předmět e-mailu, šablona e-mailu, odesílatel a (skrytá) kopie:

```

<?php

namespace App\Mail;

use Illuminate\Mail\Mailable;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Mail\Mailables\Envelope;

class FormEmail extends Mailable
{
    public array $details;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct($details)
    {
        $this->details = $details;
    }

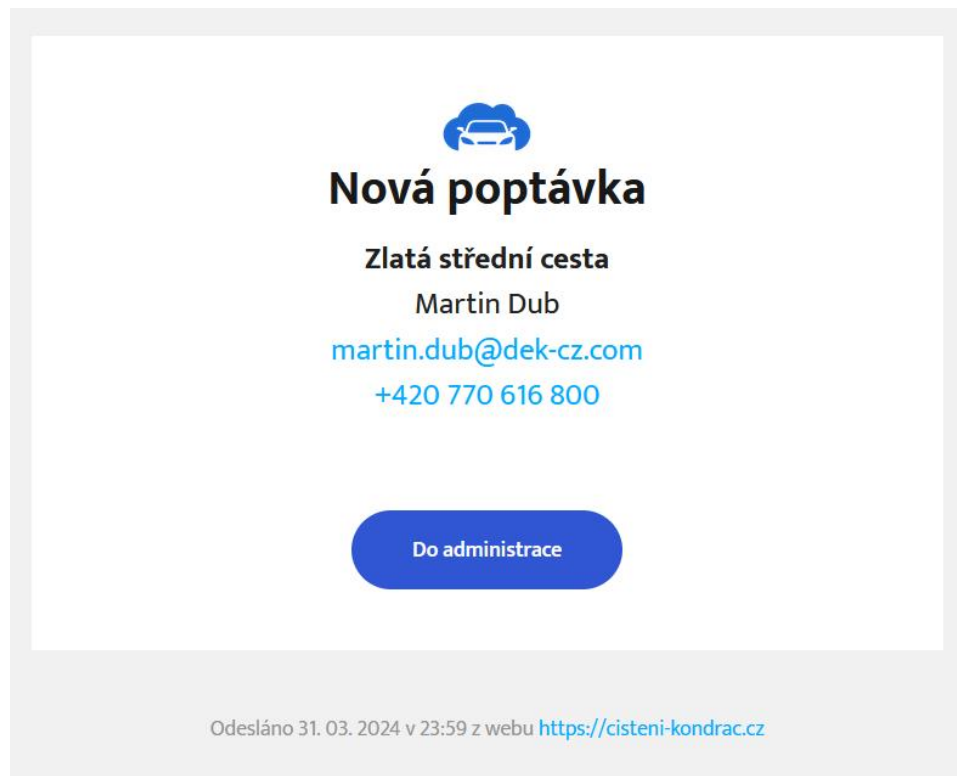
    public function envelope(): Envelope
    {
        return new Envelope(
            from: config('mail.from.address'),
            replyTo: $this->details['email'],
            subject: 'Nová poptávka'
        );
    }

    public function content(): Content
    {
        return new Content(
            view: 'emails.newDemand',
            with: [
                'name' => $this->details['name'],
                'email' => $this->details['email'],
                'telephone' => $this->details['telephone'],
                'message' => $this->details['message'],
            ]
        );
    }
}

```

*Figure 53 - Mailable třída na posílání e-mailu s novým zákazníkem.*

Šablona e-mailu vyplní data z pole \$details a výsledný e-mail s informováním o nové poptávce vypadá takto:



Obrázek 39 - Šablona e-mailu s novou poptávkou.

V rámci aplikace je nastaven notificační e-mail vývojáři v případě výskytu chyby nebo výjimky na webu. Toto nastavení se provádí v `app/Exceptions/Handler.php`. Zároveň se při výskytu chyby zapíše chyba do Laravel logu, kde je chyba zachována.

```

<?php

namespace App\Exceptions;

...

class Handler extends ExceptionHandler
{
    ...

    public function register(): void
    {
        $this->reportable(function (Throwable $e) {
            $this->sendEmail($e);
        });
    }

    protected function sendEmail(Throwable $exception): void
    {
        try {
            $content['message'] = $exception->getMessage();
            $content['file'] = $exception->getFile();
            $content['line'] = $exception->getLine();
            $content['trace'] = $exception->getTrace();
            $content['url'] = request()->url();
            $content['body'] = request()->all();
            $content['ip'] = request()->ip();

            Mail::to(config('mail.dev'))
                ->send(new ExceptionOccuredMail($content));

        } catch (Throwable $exception) {
            Log::error($exception);
        }
    }
}

```

*Figure 54 - Posílání e-mailu v případě výskytu chyby nebo výjimky.*

Admin panel nemá registrační formulář, protože jej nepotřebuje. Pro účely administrace stačí mít dva účty – pro vývojáře a pro klienta. Pro zaznamenání údajů v databázi je použit Eloquent ORM model `User`, který obsahuje data o jméně, loginu (přihlašovacímu jménu) a heslu. Heslo je z bezpečnostních důvodů šifrované `bcrypt` šifrou a při pokusu o přihlášení se používá fasáda `Hash` na ověření zadaného hesla v přihlašovací formuláři a uloženým heslem v databázi. Posílání přihlašovacích údajů je provedeno metodou `POST` tak, aby při odesílání formuláře heslo nebylo vidět v adresním řádku.

### 4.3 Testování aplikace

Testování aplikace probíhalo dvěma metodami – A/B testování (anglicky A/B split-run testing) a demo verze aplikace. První způsob umožňuje velmi účinně porovnat vícero rozložení marketingového nástroje, například webové stránky. A/B testování sloužilo pro porovnání dvou verzí domovské stránky, jedna verze (později vítězná) s videem a druhá bez videa a pouze s barvou. Výsledky testování zpracovával nástroj Google Analytics, který skrz implementovaný skript na domovské stránce sledoval chování návštěvníků stránky. Během testování bylo zjištěno, že uživatelé strávili u verze s videem delší dobu na webu, přesněji 1 minutu a 26 sekund v průměru oproti staré verzi, kde průměrný čas strávený na stránce byl 58 vteřin. Signifikantní nárůst se projevil i v průměrné době zapojení na relaci, který stoupl ze 41 vteřin na 52 vteřin. Vzrostla také relace se zapojením na uživatele. Tímto pojmem se označuje relace, která trvala déle než 10 vteřin a došlo při ní k nějaké aktivitě na webu.

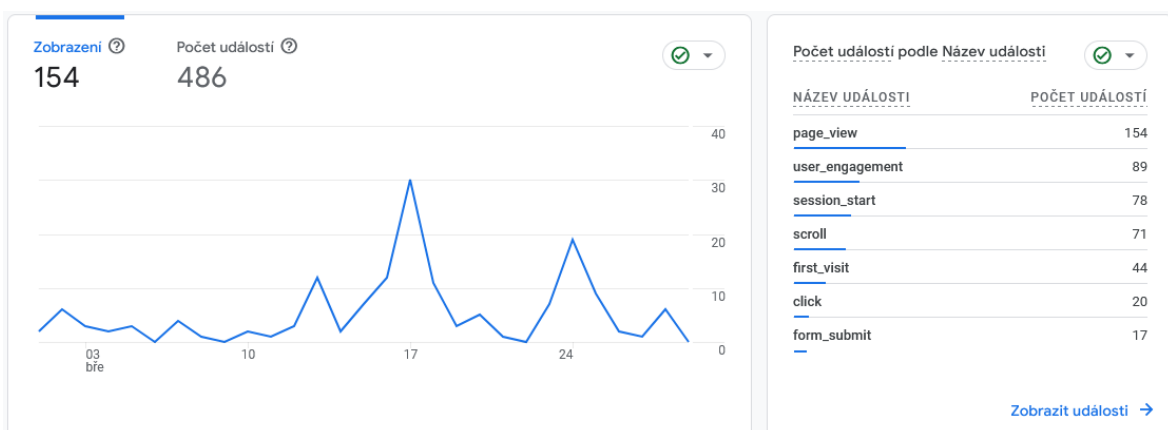
Po přidání videa na domovskou stránku se zvýšil počet zobrazení i počet událostí na webu. Uživatelé se v nové verzi častěji zapojovali, například vyplněním formuláře, na druhou stranu došlo k poklesu aktivity *scroll*. Daný pokles se dá vysvětlit tím, že uživatelé na první pohled vidí, jakou službu stránka nabízí a nemusí tyto informace hledat na webu.

Statistiky s údaji z nástroje Google Analytics jsou zobrazeny níže. Nutno dodat, že pro obě období bylo vymezeno přesně 29 dní, kdy byla nejdříve sledována varianta se starým vzhledem (model A) v termínu 1.2.2024 – 29.2.2024, a poté byla nasazena nová varianta s videem (model B) v termínu 1.3.2024 – 29.3.2024.





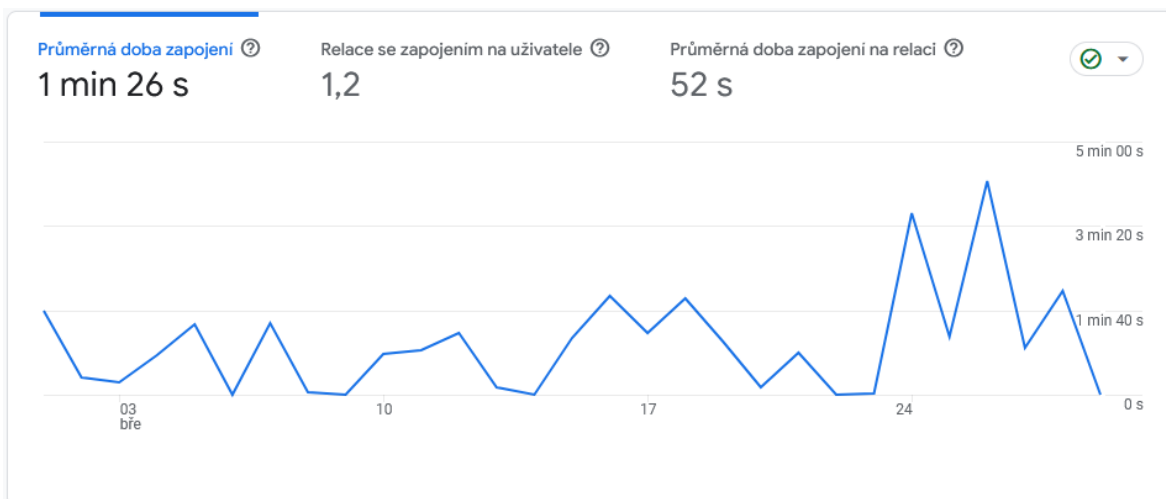
Obrázek 40 - Verze webu A, počet zobrazení a počet událostí.



Obrázek 41 - Verze webu B, počet zobrazení a počet událostí.



Obrázek 42 - Verze webu A, zapojení uživatele.



Obrázek 43 - Verze webu B, zapojení uživatele.

## 5 Závěr

Předmětem této práce bylo kompletní sestavení webové stránky včetně administrace pro firmu zabývající se čištěním interiérů aut. Teoretická práce představila použité technologie včetně obdobných nástrojů a byla provedena rešerše na existující řešení. Praktická část pojednávala o vytvoření drátěných modelů a postupném vývoji aplikace, včetně ukázek kódů a snímků obrazovky. Klientovi byl předložen drátěný model a po společné konzultaci byla vytvořena domovská stránka. Admin panel jakožto komplexnější část práce prošla celkem dvakrát refaktoringem. Jako nadstavba admin panelu byly vytvořeny automatické skripty, které informují klienta o stavu tržeb, novém zákazníkovi nebo denním itineráři. Stránka byla vyvíjena na verzovacím systému Git a zdrojový kód je uložen na platformě GitHub. Kód se díky GitHub workflow automaticky nahrává na hostingový server a dodržuje se tak aktuální verze. Byla také implementována služba na odchyťování výjimek a v případě výskytu chyby nebo výjimky se pošle mail vývojáři. Domovská stránka prošla A/B testováním kvůli vložení videí s prezentací firmy, dále prošla kontrolou na přístupnost, SEO optimalizaci a výkonnost. Klient měl po celou dobu přístup k admin panelu a během vývoje byly pořádány pravidelné konzultace jednotlivých stránek.

## 6 Seznam použitých zdrojů

11 Best PHP Frameworks for Beginners, 2023. Hostinger [online]. [cit. 2024-03-31].  
Dostupné z: <https://hostinger.com/tutorials/best-php-framework>

BOTROS, Silvia a TINLEY, Jeremy. High Performance MySQL. 4. edice. Sebastopol, CA: O'Reilly Media, 2021. ISBN 978-149-2080-510.

CLEWETT, James, 2013. End of Time (Unix) – Numberphile [online]. [cit. 2024-03-31]. Dostupné z: <https://www.numberphile.com/videos/end-of-time-unix>

*Encyclopaedia Britannica*, 2023. George Boole (British mathematician) [online]. [cit. 2024-03-31]. Dostupné z: <https://www.britannica.com/biography/George-Boole>

Endora by Webglobe: *Vlastnosti webhostingu*, 2024. In: *endora.cz* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.endora.cz/vlastnosti>

FORD, Neal, Mark RICHARDS, Pramod SADALAGE a Zhamak DEGHANI. Software architecture: The hard parts. Sebastopol, CA: O'Reilly Media, 2021. ISBN 978-149-2086-895.

Forpsi: *Webhosting*, 2024. In: *forpsi.com* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.forpsi.com/webhosting>

Getting Started with Doctrine, 2023. Doctrine [online]. [cit. 2024-03-31]. Dostupné z: <https://www.Doctrine.org/projects/doctrine-orm/en/current/tutorials/getting-started.html>

Git vs Gitolite | What are the differences? 2024. In: *StackShare* [online]. [cit. 2024-03-31]. Dostupné z: <https://stackshare.io/stackups/git-vs-gitolite>

GitLab vs. GitHub: Choosing the right version control service, 2023. In: *CircleCI* [online]. [cit. 2024-03-31]. Dostupné z: <https://circleci.com/blog/gitlab-vs-github>

HTTP request methods, 2024. In: *MDN Web Docs* [online]. [cit. 2024-03-31]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

JANOVSKÝ, Dušan. Hosting – rady, podle čeho vybrat. *Jak psát web* [online]. [cit. 2024-03-31]. Dostupné z <https://www.jakpsatweb.cz/hosting.html>

Laravel – The PHP Framework for Web Artisans, 2024a. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs>

Laravel – The PHP Framework for Web Artisans: Ecosystem Overview, 2024b. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/authentication>

Laravel – The PHP Framework for Web Artisans: Support Questions, 2024c. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/contributions>

Laravel – The PHP Framework for Web Artisans: Installation, 2024d. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/installation>

Laravel – The PHP Framework for Web Artisans: Digging Deeper, 2024e. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/artisan>

Laravel – The PHP Framework for Web Artisans: Deployment, 2024f. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/deployment>

Laravel – The PHP Framework for Web Artisans: Routing, 2024g. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/routing>

Laravel – The PHP Framework for Web Artisans: Helpers, 2024h. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/helpers>

Laravel – The PHP Framework for Web Artisans: Eloquent ORM, 2024i. In: *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: <https://laravel.com/docs/eloquent>

MASTER, Brent, 2023. *Learning GitHub Actions*. Sebastopol, CA: O'Reilly Media. ISBN 978-1098131074.

LENGSTORF, Jason, Thomas BLOM HANSEN a Steve PRETTYMAN, 2022. *PHP 8 for Absolute Beginners: Basic Website and Web Application Development*. 3. edice. New York: Apress. ISBN 978-148-4282-045.

MongoDB: *Types of Databases*, 2024. In: *MongoDB* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.mongodb.com/databases/types>

NIXON, Robin, 2021. *Learning PHP, MySQL & JavaScript*. 6. vydání. Sebastopol, CA: O'Reilly Media. ISBN 978-149-2093-824.

PHP: List of Resource Types, 2024a. In: *php.net* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.php.net/manual/en/resource.php>

PHP: Unsupported Branches, 2024b. In: *php.net* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.php.net/manual/en/resource.php>

*Refactoring Guru* [online], 2024. [cit. 2024-03-31]. Dostupné z: <https://refactoring.guru>

SKOULIKARI, Anna, 2023. *Learning Git: a hands-on and visual guide to the basics of Git*. Beijing: O'Reilly. ISBN 978-109-8133-917.

STAUFFER, Matt, 2023. *Laravel: Up & Running*. 3. edice. Sebastopol, CA: O'Reilly Media. ISBN 978-1-098-15326-7.

ŠTÍPEK, Patrik. *Porovnání PHP frameworků* [online]. Hradec Králové, 2019 [cit. 2024-03-31]. Dostupné z: <https://theses.cz/id/yzzixn/>. Bakalářská práce. Univerzita Hradec Králové, Fakulta informatiky a managementu. Vedoucí práce Mgr. Daniela Ponce, Ph.D.

Valuecoders, 2024. *Node.js vs .NET vs. PHP: Choosing the Right Stack for Software Development* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.valuecoders.com/blog/app-development/node-js-vs-net-vs-php-choosing-the-right-stack-for-software-development>

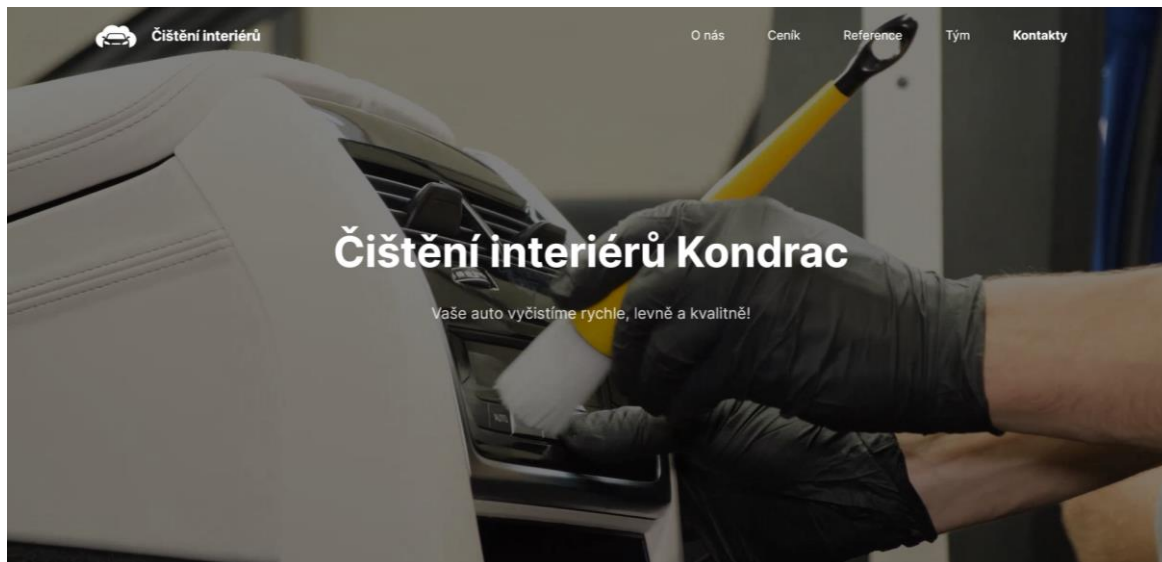
Version Control Systems: Subversion vs Git. In: *Hackbright Academy* [online]. 5. června 2023 [cit. 2024-03-31]. Dostupné z: <https://hackbrightacademy.com/blog/version-control-subversion-vs-git>

W3 SCHOOLS, 2023. *PHP Data Types* [online]. [cit. 2024-03-31]. Dostupné z: [https://www.w3schools.com/php/php\\_datatypes.asp](https://www.w3schools.com/php/php_datatypes.asp)

Wedos, 2024. In: *wedos.cz* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.wedos.com/cs>

What is a Relational Database, 2024. In: *codecademy.com* [online]. [cit. 2024-03-31]. Dostupné z: <https://www.codecademy.com/article/what-is-rdbms-sql>

## 7 Přílohy



Příloha 1 - Domovská stránka, úvodní video.

Služby

### Tohle vše dokážeme zařídit



#### Vnitřní úklid auta

Detailní luxování včetně zavazadlového prostoru a koberečků



#### Hlubkové čištění sedaček

Výkonným tepovačem a účinnými přípravky



#### Oživení plastů

Očištění plastů v interiéru a jejich následné oživení



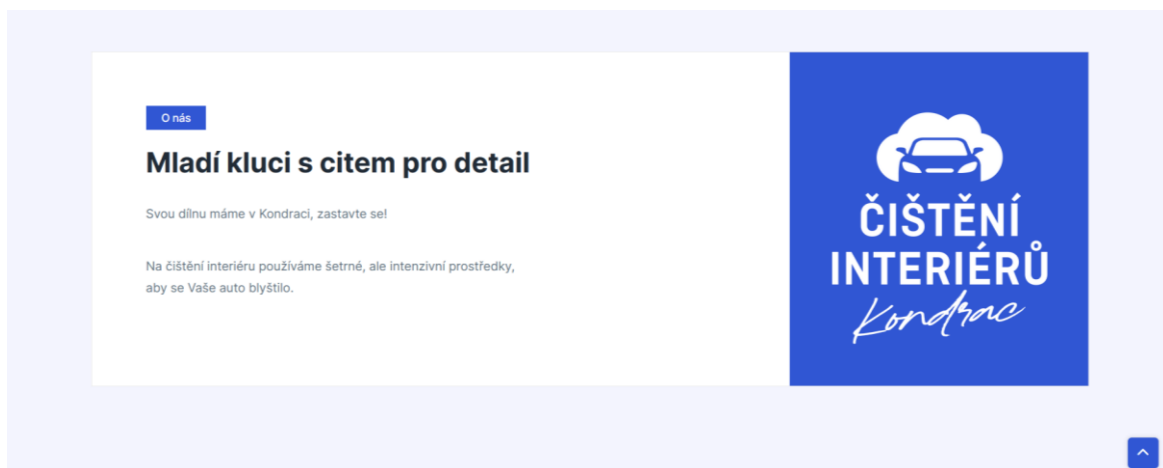
#### A mnohem více!

Stačí kliknout na tlačítko níže a podívat se na ceník

[Na ceník](#)



Příloha 2 - Domovská stránka, služby.



Příloha 3 - Domovská stránka, sekce o nás.

Ceník

## Tak jak to dneska bude?

<p><b>LEHKÝ START</b> <b>999,- Kč</b></p> <ul style="list-style-type: none"> <li>Základní luxování interiéru</li> <li>Vyluxování zavazadlového prostoru</li> <li>Vyluxování koberečků</li> <li>Ošetření plastů proti poškrábání</li> <li>Vyleštění čelního skla</li> </ul> <p><a href="#">To беру!</a></p>	<p><b>Populární</b></p> <p><b>ZLATÁ STŘEDNÍ CESTA</b> <b>2 499,- Kč</b></p> <ul style="list-style-type: none"> <li><b>Detailní luxování interiéru</b></li> <li>Vyluxování zavazadlového prostoru</li> <li>Vyčištění koberečků</li> <li>Ošetření plastů proti poškrábání</li> <li>Důkladné vyčištění párou</li> <li>Vyleštění oken a zrcátek</li> <li>Tepování sedaček a koberečků</li> <li>Desinfekce klimatizace</li> </ul> <p><a href="#">To беру!</a></p>	<p><b>DELUXE</b> <b>3 999,- Kč</b></p> <ul style="list-style-type: none"> <li><b>Detailní luxování interiéru</b></li> <li>Vyluxování zavazadlového prostoru</li> <li>Vyčištění koberečků</li> <li>Ošetření plastů proti poškrábání</li> <li>Důkladné vyčištění párou</li> <li>Vyleštění všech oken z obou stran</li> <li>Tepování sedaček a koberečků</li> <li>Základní čištění kožených částí</li> <li>Vyčištění klimatizace s vůní po citrónu</li> <li>Navoskování čelního skla</li> </ul> <p><a href="#">To беру!</a></p>
--	--	--

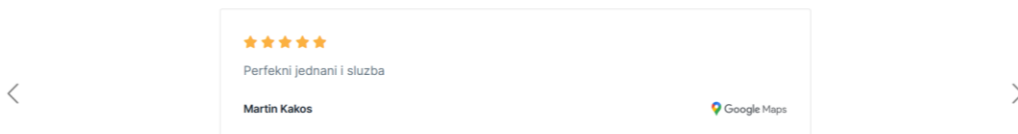
Za extra znečištěné auto může být účtován příplatek.  
Ke každému čištění malý dárek.

Příloha 4 - Domovská stránka, ceník.



Reference

## Co o nás píší zákazníci



Příloha 5 - Domovská stránka, recenze.

Tým

## Ahoj!



Příloha 6 - Domovská stránka, tým.

Kontakt

**Máte dotazy?  
Napište nebo zavolejte! Jsme  
tu pro Vás!**

**Adresa**  
Kondrac 115, 258 01 Vlašim  
[Pustit GPS](#)

**IČO:** 18026087  
**Sídlo:** Kondrac 115

**Spojme se!**  
[info@cistení-kondrac.cz](mailto:info@cistení-kondrac.cz)  
+420 602 352 402

**Napište nám!** ✎

**Celé jméno \***  
Jan Novák

**E-mail \***  
jan@novak.cz

**Telefon \***  
+420 123 456 789

**Text zprávy**  
Dobrý den...

Odeslat poptávku

Příloha 7 - Domovská stránka, kontaktní formulář.