Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informační inženýrství

Bakalářská práce

Návrh a implementace webové aplikace pro správu vozového parku

Marek Škalda

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Marek Škalda

Informatika

Název práce

Návrh a implementace webové aplikace pro správu vozového parku

Název anglicky

Design and implementation of a web application for fleet management

---

Cíle práce

Cílem této bakalářské práce je navrhnout a implementovat webovou aplikaci sloužící ke správě vozového parku firmy. Dílčími cíli je přestavit možnosti vývoje pomocí ASP.NET Core a související technologie pro tvorbu front-endu aplikace.

Metodika

Práce sestává ze dvou částí, teoretické a praktické. Metodika zpracování teoretické části je založena na studiu odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou formulována východiska pro zpracování praktické části.

Praktická část bude spočívat v návrhu a implementaci webové aplikace pomocí ASP.NET Core. Aplikace bude sloužit ke správě vozového parku vybrané firmy. Výsledná aplikace bude nasazena a otestována. Závěrem budou shrnuty poznatky získané při tvorbě aplikace a na základě zpětné vazby z testování budou navrženy možnosti případných dalších úprav a možných dalších budoucích rozšíření aplikace.

Doporučený rozsah práce

35-40 stran

Klíčová slova

C#, ASP.NET Core, informační systém, webová aplikace, Bootstrap, evidence vozidel, SQL

Doporučené zdroje informací

FREEMAN, Adam. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 8th edition. New York: Apress, 2020. ISBN 978-1-4842-5439-4.
https://docs.microsoft.com/cs-cz/aspnet/core/?view=aspnetcore-5.0
https://docs.microsoft.com/cs-cz/dotnet/csharp/
https://getbootstrap.com/docs/5.0/

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 11. 03. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh a implementace webové aplikace pro správu vozového parku" jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2022 _____

Poděkování

Rád bych touto cestou poděkoval mému vedoucímu Ing. Jiřímu Brožkovi Ph. D. za jeho vedení, užitečné rady a připomínky k práci, dále patří velké díky mému bratrovi Danielovi za konzultace týkající se vývoje výsledné aplikace.

Návrh a implementace webové aplikace pro správu vozového parku

Abstrakt

Práce se zabývá návrhem a implementací webové aplikace pro správu vozového parku firmy. V teoretické části je přímé uvedení do technologií spojených s tvorbou webových aplikací. Vzhled webové aplikace se implementuje pomocí značkovacího jazyka HTML a CSS frameworku Bootstrap. Funkcionalita webové aplikace je zaopatřena jazykem C# a jeho frameworkem pro tvorbu webových stránek ASP.NET Core. Aplikace čerpá data z databázového souboru pomocí SQLite. Největším přínosem této bakalářské práce je představení a následný vývoj pomocí výše uvedených technologií pro tvorbu webové aplikace.

Klíčová slova: C#, ASP.NET Core, informační systém, webová aplikace, Bootstrap, evidence vozidel, SQL, LINQ, web

Design and implementation of a web application for fleet management

Abstract

This Bachelor's thesis is focused on designing and implementing web application for fleet management. In theoretical part is direct introduction into technologies connected to development of web applications. Web application design is being implemented by markup language HTML and CSS framework called Bootstrap. Functionality of web application is done by using C# language and its framework used to create web applications called ASP.NET Core. Application is getting data from database file using SQLite technology. Biggest benefit of this bachelor's thesis is the introduction and subsequent development by using technologies above to create a web application.

Keywords: C#, ASP.NET Core, information system, Bootstrap, vehicle evidence, SQL, LINQ, web application, web

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

Zkratka	Význam
HTML	HyperText Markup Language
CSS	Cascading style sheets
LINQ	Language integrated query
SQL	Structured query language
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
IDE	Integrated development environment

1 Úvod

V dnešní době se opravdu hodně často využívají webové aplikace. A to hned z několika důvodů, kterými je například dosažitelnost pro klasického uživatele. Webovou aplikaci může využívat téměř z jakéhokoliv místa pokrytého internetovým připojením. Výhodou webových aplikací je jednoznačně nevyžadující instalace čehokoliv na systém, výhodou je to jak pro klasické uživatele, tak i pro vývojáře. Vývojář totiž webovou aplikaci vyvíjí už pro několik platforem najednou, zatímco klasické aplikace se vyvíjejí většinou zaměřené na konkrétní platformu. Klasický uživatel má zase výhodu v tom, že na webovou aplikaci se dostane téměř ze všech moderních zařízeních – především jde o tablet, mobilní telefon a počítač, protože webový prohlížeč je v dnešní době součástí všech moderních operačních systémů.

Tato práce se zabývá problematikou tvorby webových aplikací a front-endových technologií. Je také zaměřena na vytvoření webové aplikace využívající frameworku pro tvorbu webových aplikací ASP.NET Core. Ve výsledné aplikaci se také využívají front-endové záležitosti pro úpravu uživatelského rozhraní. Pro uživatelské rozhraní je využíván jazyk HTML v kombinaci s CSS a jeho frameworkem Bootstrap.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této bakalářské práce je navrhnutí a implementace webové aplikace sloužící ke správě vozového parku firmy. Dílčími cíli je představení možností vývoje pomocí frameworku pro tvorbu webových aplikací ASP.NET Core, nejpopulárnějších softwarových architektur využívající se pro vývoj webových aplikací a také představení souvisejících technologií pro vývoj front-endu aplikace. Dalším cílem bylo představení dvou významných vývojových prostředí – Visual Studio Code a Visual Studio 2022.

2.2 Metodika

Práce sestává ze dvou částí, teoretické a praktické. Metodika zpracování teoretické části je založena na studiu odborných informačních zdrojů týkající se vývoje webových aplikací a souvisejících front-endových a back-endových technologií.

Praktická část spočívá přímo v návrhu a implementaci webové aplikace pomocí frameworku určeného pro vývoj webových aplikací ASP.NET Core. Aplikace je navržena především z hlediska funkcionality. Výsledná aplikace slouží ke správě vozového parku firmy, aplikace je nasazena na zakoupený server a otestována jak v lokálním prostředí, tak i v ostrém provozu. Závěrem jsou shrnuty poznatky získané při tvorbě aplikace a na základě poznatků z vývoje i testování jsou dále formulovány možnosti případných dalších úprav a možných dalších budoucích rozšíření výsledné webové aplikace.

3 Teoretická východiska

3.1 Využité technologie

Tato kapitola se zabývá obecným přehledem technologií a terminologií pro vývoj webových aplikací, a to nejen pro výslednou aplikaci. V první řadě se porovnávají softwarové architektury MVC a MVVM, dále se vysvětlují technologie jako je Entity Framework Core, LINQ a Razor Pages. V další části se rozebírají frontendové záležitosti což je Bootstrap, JavaScript a jeho jeden vybraný populární framework. V neposlední části je text věnován informačním systémům. Vývojovým prostředím je text věnován až úplně naposled.

3.1.1 Framework ASP.NET Core

ASP.NET Core je open-source a cross-platform framework vyvíjený společností Microsoft. Motivací tvorby tohoto frameworku bylo, jak je již výše zmíněno open-source a cross-platform, mezi další aspekty patří udržitelnost softwarové architektury a aby byl použitelný vůči aktuálním trendům ve web developmentu pro například client-side aplikace nebo nasazení na cloudové služby. Pro dosažení těchto cílů Microsoft potřeboval platformu, která dokáže poskytnout veškeré knihovny pro tvorbu základních objektů a jednoduchých operací se soubory. [1]

Původně byl totiž ASP.NET mířen pouze na operační systém Windows. Pro ASP.NET Core Microsoft vytvořil platformu která běží na Windows, Linuxu, a i macOS nazvanou .NET Core (následně .NET #verze). Jedná se o potomka dvou frameworků, ASP.NET MVC a ASP.NET Web API. [1]

3.1.1.1 MVC architektura

MVC architektura rozděluje aplikaci do tří hlavních skupin komponentů. Těmi je Models, Views a Controllers. Tato architektura pomáhá rozložit aplikaci do srozumitelné formy, kdy i jiní programátoři, než je sám autor se dokážou orientovat v souborové struktuře dané aplikace.

Veškeré požadavky uživatele jsou směrovány na Controller, který je zodpovědný za spolupráci s Modelem. Model má na starosti různé uživatelské akce či získávání dat z databáze a tak podobně. Controller taktéž volí daný View, který se má zobrazit uživateli a poskytne mu veškerá data z komponenty Modelu, která View potřebuje pro správné zobrazení. [2], [3]

3.1.1.1.1 Model

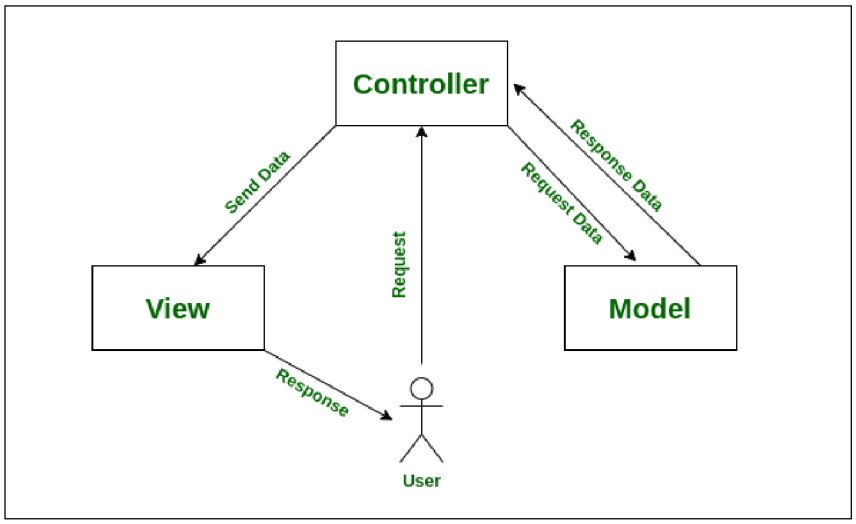
Model v této architektuře má za úkol reprezentovat veškerou logiku či operace které by měla aplikace provádět. Provádí jak matematické operace, tak i například získávání dat z databáze, a to všechno na základě volání z Controlleru. [4]

3.1.1.1.2 View

View je zodpovědný za správné zobrazování dat zprostředkované pomocí vhodného modelu. Ve View by mělo být co nejméně logiky, samozřejmě občas se bez toho nedá obejít, ale když už, tak veškerá logika zde by měla patřit čistě k zobrazování nějakého obsahu. K využívání .NET kódu v HTML se používá Razor View engine. [4]

3.1.1.1.3 Controller

Controller je komponenta která zpracovává veškeré uživatelské vstupy a interakce. Z jeho názvu již jde usoudit, že řídí aplikaci. Podle uživatelských vstupů a akcí je controller povinen vybrat vhodný model pro zpracování dat se kterými musí pracovat a poté je zobrazí v dané komponentě View. Controller by taktéž neměl být zbytečně moc zahlcován úkony, které může v pořádku vyřešit i model. [4]



1- MVC architektura [29]

3.1.1.2 MVVM architektura

Architektura MVVM obsahuje též tři komponenty, jimiž jsou Model, View ViewModel, každý z nich samozřejmě má rozdílné účely. Je znám také jako Model, View a Binder. Jako spoustu ostatních architektur, slouží k uspořádání kódu a rozděluje program do dejme tomu modulů, které usnadňují vývoj, úpravu kódu, a jeho znovu použitelnost. [5]

3.1.1.2.1 Model

Model obsahuje stejně jako u MVC struktury data a logiku aplikace či programu [5]. U této struktury se model bere spíše jako takzvaný doménový model, který reprezentuje model s „business-logic“ či ověřovací logikou. Modelové třídy jsou typicky využity ve sjednocení služeb či uložišť, které obsahují přístupová data a mezipaměť. [6]

3.1.1.2.2 View

View je zodpovědné za definování struktury, rozložení a toho, co vidí uživatel. Každý View by měl být definován pomocí XAML [1]s omezenou logikou. V některých případech je za potřebí využít logiku i ve View například kvůli animacím, reagováním na události (stisknutí klávesy, kliknutí na tlačítko atp.) z důvodu zbytečně těžké implementace do XAML. [6]

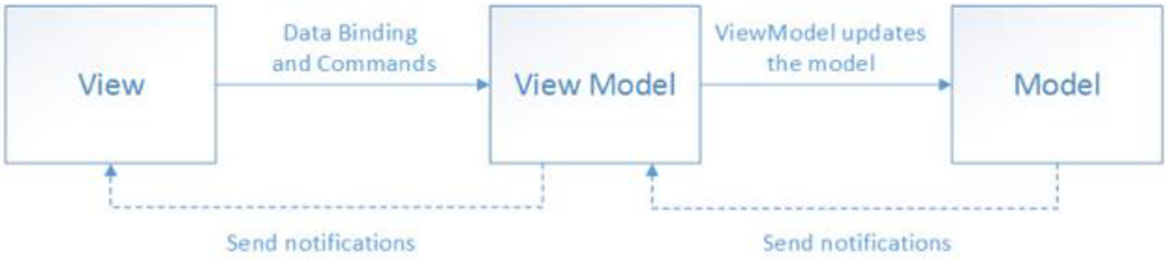
[1] XAML – eXtensible Application Markup Language, jazyk založen na XML. Jedná se o alternativu pro instancování, zakládání objektů a jejich organizaci v hierarchii.

3.1.1.2.3 ViewModel

ViewModel obsahuje vlastnosti, metody a příkazy, které pak může přiřadit ke komponentě View. Obeznamuje komponentu View o jakékoliv změně stavu, pomocí událostí. Vlastnosti a příkady, které ViewModel poskytuje definuje vlastně funkcionalitu, kterou poté nabízí uživatelské rozhraní, ovšem View rozhoduje o tom, jak je tato funkcionalita zobrazena.

Tato komponenta je také zodpovědná za koordinaci interakcí komponenty View s ostatními komponentami modelu, které jsou vyžadovány. Typicky je zde takzvaný vztah „one-to-many“ mezi ViewModelem a Modelem. ViewModel může odhalit Model přímo komponentě View, tak, že funkce ve View, můžou být přímo přiřazeny k Modelu. V tomto případě, je za potřebí navrhnout Model tak, aby podporoval přiřazování dat a také podporu události oznamování změn.

Každý ViewModel poskytuje data z modelu ve formě, které View jednoduše rozumí. K dosažení tohoto, ViewModel musí občas provést konverzi dat. Umisťování těchto datových konverzí ve ViewModelu je dobrým nápadem, protože poskytuje vlastnosti, které mohou být komponentě View přiřazeny. Například ViewModel může kombinovat hodnoty dvou vlastností, aby ulehčil komponentě View zobrazení. [6]



2- MVVM struktura [6]

3.1.1.3 Porovnání architektury MVC a MVVM

MVC	MVVM
Komponenta Controller je vstupním bodem do aplikace.	Komponenta View je vstupním bodem do aplikace.
„One-to-many“ vztah mezi Controllerem a View	„One-to-many“ vztah mezi View a ViewModelem
View nemá možnost, jak odkazovat na Controller	View má možnost odkazovat na ViewModel
MVC je zastaralý vzor	MVVM je relativně novým vzorem
Složitější ke čtení, úpravám, jednotkovým testům atp.	Debuggovací proces bude zkomplikovaný v případě, že máme komplexní datové vazby
MVC komponenta Model může být testována odděleně od uživatele	Jednoduché pro oddělení jednotkových testů a kód je řízený událostmi.

Tabulka 1- Porovnání architektury MVC a MVVM [7]

3.1.1.3.1 Výhody a nevýhody MVC [7]

Mezi výhody patří:

- vývoj různých komponent může být paralelní.
- využívá „front-controller“ vzor, tak, že procesy webové aplikace využívají jeden jediný Controller.
- Nabízí super podporu pro takzvaný „vývoj řízený testy“.
- Skvělý pro webové aplikace, na kterých spolupracuje tým web designerů a vývojářů.
- Poskytuje čisté rozdělení starostí.
- Všechny třídy a objekty jsou na sobě nezávislé, což znamená, že je možnost je testovat odděleně.
- MVC poskytuje logické uskupení relevantních akcí na Controller.

Nevýhody:

- Nepodporuje formální validaci.
- Zvýšená komplexita a neefektivita dat.
- Obtížnost použití MVC s moderním uživatelským rozhraním.
- Pro více vývojářů je potřeba řídit paralelní programování.
- Je potřeba znát více technologií.

3.1.1.3.2 Výhody a nevýhody MVVM [7]

Mezi výhody patří:

- Logika je oddělena od uživatelského rozhraní.
- Lehké na údržbu a testování
- Lehké znovu využití různých komponent.
- Jedná se o volně propojenou architekturu.
- Možnost psát jednotkové testy pro ViewModel i pro Model bez potřeby odkazovat na View.

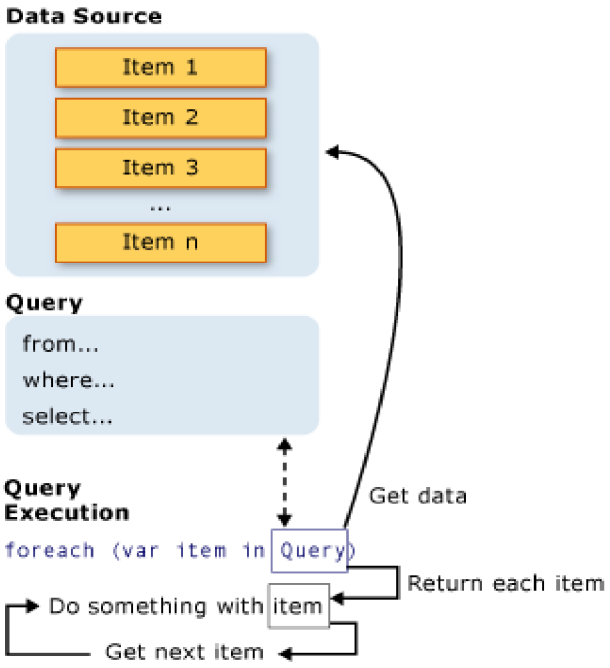
Nevýhody:

- Někteří lidé si myslí, že využití architektury MVVM pro jednoduché uživatelské rozhraní je „přestřel“.
- Nenabízí těsné spojení View a ViewModel

3.1.2 LINQ

LINQ je zkratkou „language-integrated query“ což je v překladu do českého jazyka integrované dotazování. Tato technologie integruje dotazy pro získání dat přímo do jazyka C#. Bez této technologie je nutné znát rozdílné dotazové jazyky pro každý datový typ čímž může být například: SQL databáze, XML dokumenty, různé webové služby a tak dále. LINQ poskytuje konzistentní dotazy pro objekty (LINQ to Objects), relační databáze (LINQ to SQL) a XML (LINQ to XML). [8], [9]

Pro vývojáře, který píše dotazy je nejviditelnější ta integrovaná část přímo do jazyka. Při používání dotazové syntaxe může vývojář provádět filtrování, uspořádání a seskupování dat s minimální délkou kódu. Stejné dotazové výrazy se používají pro transformaci dat SQL databáze, ADO.NET Datasets, XML dokumenty, a i .NET kolekce. [8]



3- Náhled na funkcionalitu LINQ technologie [10]

3.1.3 Entity Framework

Entity Framework nebo také EF Core je knihovna co používají vývojáři k rychlému přístupu do databáze. Je navrhnut jako objektově relační „mapper“. Mapuje vlastně dva světy, prvním je relační databáze s její vlastní API (není nutnost se připojovat na databázový server, může být použito jiné uložiště) a druhým světem je objektově orientovaný software. Hlavní silou EF Core je možnost rychle psát přístup do databáze v jazyce který může vývojář znát více než například SQL. [11]

EF Core vytváří datové modely založené na POCO (Plain Old CLR Object)[2] entitách s get a set vlastnostmi různých datových typů. Tento model se využívá při dotazování či ukládání dat entit do databáze. Také podporuje LINQ, o kterém je psáno výše. Poskytovatel databáze přeloží LINQ dotaz do jazyku dané databáze. EF také podporuje vykonávání čistých SQL dotazů přímo do databáze.

Poskytuje několik migračních příkazů, které mohou být provedeny v NuGet Package Manageru[3] či v CLI pro vytvoření či úpravy schémata databáze.

EF Core podporuje multi-platform tak, že může běžet jak na Windows, Linux tak i na MacOS. [12]

3.1.4 Razor Pages

Razor Pages je nový, zjednodušený model pro vývoj webových aplikací. Každá Razor Page, která je ve složce Pages je vlastně i koncovým bodem. Razor Pages mají v C# objekt pojmenovaný „page model“, který má na starosti chování dané stránky. Každá taková stránka podporuje pouze GET a POST metody. [13]

[2] POCO – Plain Old CLR Object – jednoduchý objekt vytvořený v .NET common language runtime neovlivněný dědičností či atributy

[3] NuGet Package Manager – jedná se o takzvaný mechanismus skrze který mohou vývojáři vytvářet, sdílet a používat kód. Většinou se jedná o kód v balíčku, který obsahuje různé kompilované kódy (jako DLL soubory) podle potřeb v projektu užívající tyto balíčky. [28]

3.1.5 HTML

HTML je zkratkou pro Hypertext Markup Language. Je základním stavebním kamenem pro tvorbu webových dokumentů (stránek). Definuje význam a strukturu obsahu zobrazovaného v dokumentu. Další technologií pro stylizování stránky je **CSS** a pro vizuální efekty a různé funkce které se mohou odehrávat na straně klienta je zde **JavaScript**.

HTML využívá značky, aby označil text, obrázky anebo jiný obsah pro zobrazení ve webovém prohlížeči. HTML obsahuje speciální takzvané elementy (značky) jako je například *<head>*, *<title>*, *<body>* a spousty dalších.

Jak již bylo zmíněno, využívají se značky pro oddělení textů, obrázků a jiných částí obsahu, tyto značky jsou obklopeny znaménky < a >. Název dané značky je uveden mezi znaménky. Názvy značek je možné psát jak malým písmem, velkým tak i kombinovaně. [14]

3.1.6 CSS

CSS neboli Cascading Style Sheets – kaskádové styly dovolují autorovi definovat pravidla stylů, která jsou aplikována na HTML elementy. Pravidlo se může týkat jednoho elementu, skupině elementů anebo všech elementů. Pravidla stylů ovlivňují vykreslování elementů, včetně jejich barev, zarovnání, okrajů a mezer mezi nimi a obsahem. Pravidla stylů také můžou kontrolovat speciální elementy, jako je například element *ol* (element označeného listu), který může využívat písmena či římské číslice jako označení. Kaskádové styly definují plnou syntaxi pro přiřazování stylových atributů pro pravidla. [15]

3.1.7 Bootstrap

Bootstrap je jedním z nejvíce využívaných CSS frameworků pro vytváření webových stránek či aplikací. Pro jeho využívání je samozřejmostí znát alespoň základy **HTML** a **CSS**. [16]

Stejně jako každý framework slouží k urychlení procesu tvorby, v tomto případě jde o vzhled stránky, obsahuje totiž spousty již předpřipravených šablon pro úpravu různých formulářů, tlačítek a navigací.

Mezi důvody, proč si zvolit tento framework je, jak již bylo zmíněno nejpopulárnější z čehož lze usoudit, že má také nejrychlejší odezvu oprav na různé chyby, dalším důvodem je samozřejmě žádný poplatek za používání, jelikož se jedná o open-source, nemusí se vůbec uvádět jeho používání, stejně to všichni dřív či později poznají. A hlavním důvodem jeho využívání a jeho popularity je responzivita a samozřejmě mobile-first přístup. [17]

3.1.8 JavaScript

JavaScript je interpretovaný nebo tzv. „just-in-time“ kompilovaný programovací jazyk. Je především známý jako skriptovací jazyk pro webové stránky, ale je schopen fungovat i mimo webový prohlížeč, používá se například přes Node.JS, Apache CouchDB a Adobe Acrobat. [18]

Samotný JavaScript běží na straně klienta, což umožňuje úpravu vzhledu stránky či reakci na událost spuštěnou právě tím klientem.

Mnoho lidí často spojuje či zaměňuje samostatnou Javu s Javascriptem, jedná se ale o dva opravdu rozdílné jazyky v ohledu na syntax, sémantiku a vůbec i použití. Jediné, co mají společné je, že jsou registrované pod stejnou firmou což je Oracle.

JavaScript může fungovat jak procedurálně, tak i objektově. Objekty jsou tvořeny programaticky pomocí metod a vlastností (propert) pro vyčištění objektů při run-time, a na rozdíl definicím tříd v kompilovaných jazycích jako je C++ a Java. Jakmile je objekt vytvořen, může se dále využívat jako takzvaný blueprint pro vytváření podobných objektů. [19]

3.1.8.1 Vue.JS

Vue.JS je JavaScriptovým frameworkem a ekosystémem, který pokrývá veškeré potřebné funkce ve front-end developmentu. Je navrhnut tak aby byl flexibilní a postupně adaptabilní.

Vue může být použit v několika use-casech, čímž může být například: Jednostránkové aplikace (SPA[4]), obohacování statických HTML stránek, vkládání různých webových komponent (animace, notifikace a tak dále), zaměřování na desktopové verze, mobilní anebo taky terminál. [20]

3.1.9 Informační systém

Informační systém je set komponentů, které spolupracují mezi sebou, aby sbírali, ukládali, zpracovávali a rozšiřovali informace. Tyto informace podporují fundamentální byznysové operace, reporty dat a vizualizaci, analýzu dat, rozhodování, komunikaci a koordinaci organizace. Informační systémy zpracované na vyšší úrovni vykazují nějakou formu mechanismu zpětné vazby pro monitorování a kontrolování jeho operací. Tato zpětná vazba potvrzuje, že systém pokračuje v operacích efektivním způsobem. [21]

[4] SPA – Single Page Application – Jednostránková aplikace

3.2 Vývojové prostředí

Vývojové prostředí je kolekcí procesů a pomůcek, které jsou využity ve prospěch vývoje zdrojového kódu softwaru. Vývojová prostředí automatizují, či alespoň usnadňují rutiny jako je vytváření, testování, debuggování, aktualizování a udržování softwaru.

Časté slovní spojení, které pod vývojovým prostředím naleznete, je Integrované vývojové prostředí neboli IDE[5]. Toto spojení zaštituje vývojové prostředí, které vývojáři využívají pro psaní, tvorbu, testování a debuggování softwaru. IDE poskytuje konzistentní uživatelské prostředí, které podporuje proces psaní a následné testování kódu. Vývojové prostředí takto jednoznačně vylepšuje vývojářovu produktivitu. [22]

3.2.1 Visual Studio Code

VSCode je jedním z nejpoužívanějších editorů dnešní doby. Hlavním důvodem je jeho otevřenost vůči novým přídavným modulům, jeho rychlosti, možností úpravy uživatelského prostředí, a především možnost ho používat jak na operačním systému Windows, tak i v MacOS a samozřejmě i v Linuxu.

Jeho popularita je zapříčiněna uživatelským rozhraní, velmi užitečným „našeptávačem“ kódu a zvýrazňování správně či špatně používané syntaxe jazyka. Dalším bonusem, k již zmíněným výhodám je podpora pro ladění a refactoring aplikace a podpora technologie IntelliSense.

Přídavné moduly se dají dokoupit či doinstalovat zdarma, používají se například pro „našeptávač“ jazyků či frameworků které v základu nejsou podporovány samotným editorem. Dalším příkladem přídavných modulů je modul pro stylizaci kódu, který se po uložení formátuje do přednastaveného stylu.

Editor je nezávislý na modulech, pokud se načítá velké množství přídavných modulů, nijak to VSCode neovlivňuje ve smyslu jeho rychlosti. [23]

[5] IDE – Integrated Development Environment – Integrované vývojové prostředí

3.2.2 Visual Studio

Visual Studio je IDE taktéž od společnosti Microsoft. Oproti VSCode je více komplexní, protože je využíváno pro tvorbu počítačových programů, webových stránek, aplikací, služeb a mobilních aplikací.

Podporuje technologii IntelliSense i refactoring kódu. Jiné podporované funkce jsou například designer pro tvorbu GUI aplikací, designer tříd, schémat databází a webu. Visual Studio podporuje 36 různých programovacích jazyků a také je možnost podpory dalších jazyků které nejsou defaultně podporovány.

Toto IDE je vyvíjeno již od roku 1997, kdy vyšla jeho první verze s názvem „Visual Studio 97“, od tohoto roku je dá se říct nepravidelně vydáváno až do roku 2012, od roku 2013 je vydáváno ob rok. Nejaktuálnější verze, se kterou budeme pracovat v praktické části je dnešním dnem Visual Studio 2022. [24]

Má tři edice a jednou z nich je edice Community, kterou využíváme pro tvorbu výsledné aplikace z důvodu, že edice je zdarma a je určena pro tým s méně než pěti lidmi či takzvaným single-developerům. Edice Proffesional je další verzí Visual Studia, jedná se vlastně o lehce vylepšenou verzi Community, vylepšenou o podporované use-case. Poslední edicí je Enterprise a ta je určena pro podniky které mají 250 nebo více počítačů, anebo mají roční příjmy alespoň jeden milion amerických dolarů. [25]

4 Vlastní práce

Pro mou bakalářskou práci jsem si vybral návrh a implementaci správy vozového parku firmy. Aplikace bude zpracována pomocí ASP.NET Core frameworku, s nejaktuálnější verzí 6.0 (zkráceně .NET 6.0). Pro stylování stránky se defaultně v .NETu využívá framework Bootstrap. Pro správu databáze jsem zvolil SQLite.

Aplikace bude tvořena ve vývojovém prostředí (IDE) Visual Studio 2022, pro vedlejší účely bude využíván VSCode.

Pod pojmem správa vozového parku si představuji aplikaci, která bude obsahovat přehledy o vozidlech – komu je vozidlo přiděleno a na jakou dobu po vybraný den, závady na vozidle – případné dočasné či úplné vyřazení z provozu a evidence vozidel. Aplikace by měla také obsahovat informace o uživatelích (zaměstnancích), záznamy o jejich provedených úpravách či rezervací vozidel.

4.1 Návrh aplikace

Front-end aplikace bude jednoznačně tvořen značkovacím jazykem **HTML**, který je výchozím výstupem webových stránek a je dále zpracovávaný webovým prohlížečem. Front-end dále bude využívat kaskádových stylů ve formě responzivního frameworku **Bootstrap**, který defaultně využívá **JavaScript** pro interakci s uživatelem (klikání na elementy jako je například menu, utvoří efekt takzvaného dropdownu a tak podobně).

Back-end aplikace bude tvořen jazykem **C#** za pomoci **ASP.NET Core** frameworku a jeho v tuto dobu nejaktuálnější verzí 6.0. Pro databázi jsem zvolil SQLite, výhoda je v tom, že je databázový soubor uložen přímo na serveru. Aplikace je tvořena softwarovou architekturou **MVVM**.

4.1.1 Funkce aplikace

Aplikace bude mít několikero funkcí zjednodušující správu vozidel. Aplikace bude zaměřena na přehled o firemních vozidlech a jejich záznamech, čímž je myšleno poslední TK, závady, poslední servisní kontroly a tak podobně. Aplikace bude mít funkci rezervace daného vozidla po určenou dobu (aplikace bude podporovat rezervace pro každý den od rána do večera – od 06:00 do 20:00).

4.1.1.1 Přihlašování

Pro přihlašování do aplikace využijeme rozhraní ASP.NET Core Identity, které je volně přístupné. Využijeme ovšem pouze části, a těmi jsou IdentityUser a IdentityRole.

Tyto části urychlí tvorbu přihlašování do aplikace, registrace a přiřazování rolí uživatelům. IdentityUser má přednastavené databázové prvky ohledně uživatelů, tudíž můžeme přidávat pouze nezbytné věci určené pro naši aplikaci a tím bude křestní jméno a příjmení. IdentityRole má také přednastavené databázové prvky a má na starosti, jak asi z názvu může vyplynout role a tím pádem i jejich oprávnění. Tuto část využijeme ve prospěch rozdělení oprávnění jak pro funkce, tak pro přístup do jednotlivých adresářů.

4.1.1.2 Registrace uživatelů pomocí administrátorského účtu

Tato funkce bude bezprostředně zabezpečená pro roli Administrátor, jenž po nasazení bude mít pouze jeden jediný účet, který se automaticky založí se zadanými údaji. Pro administrátora je samozřejmé, že může vytvářet další účty s rolí Administrátor pro případ, kdy by si to například sama firma vyžádala.

Pro registraci bude vytvořen samozřejmě formulář s požadovanými informacemi, které po zadání využijí rozhraní IdentityUser a IdentityRole, přičemž vytvoří nový účet a přiřadí mu vybranou roli.

4.1.1.3 Přidávání a mazání vozidel

Funkce přidávání vozidel bude přístupná pro role Správce a Administrátor. Tato funkce je uzpůsobena pro vedení firmy nebo společnosti pro zjednodušené vytváření vozidel ke kterým dále mají už přístup zaměstnanci (role Uživatel v aplikaci).

Funkce mazání bude přístupná pro tytéž role. Před provedením se aplikace dotáže, zda je uživatel opravdu rozhodnutý operaci provést, vozidla totiž nebudou v této verzi nijak zálohovány a tato aplikace poté nemá možnost, jak vozidlo zpětně dohledat a případně znovu vytvořit jej musí uživatel s rolí Správce či Administrátor.

4.1.1.4 Přehledy vozidel

Přehled zobrazuje všechna vozidla zaevidovaná v databázi. V přehledu vozidel budou čtyři možnosti, co se s vozidlem dá dělat. První možností bude samozřejmě rezervace vozidla, která je popsána dále v textu. Další možností je přidat poznámku k vozidlu, kde si každý uživatel může přidat k vozidlu poznámku – například „Servisní prohlídka“ „Proběhla dne 2.3.2022, byl měněn olej a filtr, faktura číslo: 15144“. Poté přicházejí dvě možnosti, které jsou pro role Správce a Administrátor, první je úprava vozidla, která zobrazí formulář s původními hodnotami téhož vozidla a uživatel má možnost úpravy s přehledem o předchozích informacích. Poslední možností je smazat vozidlo, které se dotáže, zda opravdu chce uživatel vozidlo smazat a v případě, že uživatel potvrdí smazání tak ho nadobro smaže z databáze.

4.1.1.5 Přehledy uživatelů

Přehled uživatelů zobrazuje veškeré uživatele aplikace, je dostupný pouze pro uživatele s rolí Administrátor. Obsahuje tři akce, první akce je informace o uživateli, kde se zobrazí znovu jeho informace již zobrazené v přehledu ale hlavně, se zde zobrazí i poslední uživatelské akce (odebrání, úprava anebo přidání poznámky k vozidlu).

Po kliknutí na tlačítko upravit bude následovat formulář s možnými úpravami uživatele a jeho dat. Další možností je odebrání, což se vás opětovně zeptá, zda opravdu chcete uživatele odstranit či nikoliv.

4.1.1.6 Rezervace vozidel

Rezervace vozidel bude možná pro všechny role. Rezervace v této verzi aplikace bude dostupná pouze jednodenní s vybraným časem od 06:00 do 20:00.

4.1.1.7 Přehled o rezervacích vozidel

Přehled bude tvořen tabulkou, nad kterou bude výběr datumu, pro který má zobrazovat rezervace. Tabulka bude zobrazovat všechna vozidla zaevidovaná v databázi a jejich rezervace pro zadaný (či dnešní) datum od 06:00 do 20:00.

4.1.1.8 Záznamy o uživatelských akcích

Záznamy se automaticky vytvářejí při jakékoliv činnosti – ať už je to přihlášení, odhlášení anebo přidání nového vozidla, jeho úprava, úprava uživatele, smazání uživatele, a tak podobně. Přehled záznamů bude přístupný pro role Správce a Administrátor.

4.1.1.9 Záznamy o vozidle

Záznamy o vozidle zadává sám uživatel. Jako záznam o vozidle se bere například poslední servisní prohlídka, jakákoliv závada, která je nutná opravit a tak podobně. Záznamy se budou samozřejmě ukládat a vázat na dané vozidlo, ke kterému byly přiřazeny.

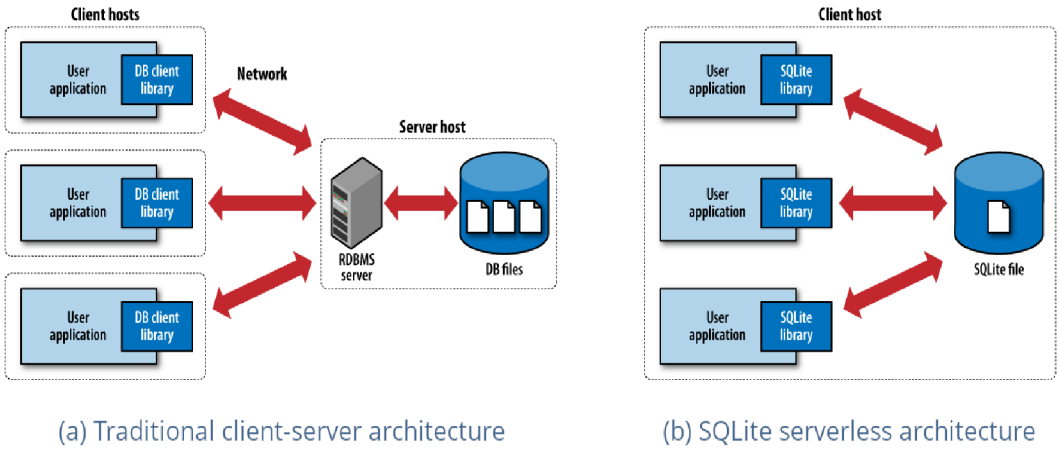
4.1.2 Adresářová struktura

Pomocí vývojového prostředí Visual Studio 2022 bude vytvořen projekt pomocí přednastavené šablony „ASP.NET Core Web Application“, tato šablona vytvoří veškeré potřebné soubory pro funkci celého frameworku v této aplikaci s exemplárním využitím Razor Pages podle vzoru Model-View-ViewModel.

4.1.3 Databáze

Pro tuto aplikaci jsem zvolil SQLite což je softwarový balíček, který poskytuje systém správy relačních databází. SQLite nevyžaduje žádný server navíc, protože se SQLite knihovna dostane do databáze přímo skrze soubor, a protože nevyžaduje server navíc, ulehčí práci tím, že se nemusí nic víc nastavovat, protože vytvořit databázovou instanci je opravdu jednoduché. [26]

SQLite jsem zvolil také z několika důvodu, jedním z nich je že databázový soubor lze otevřít na dalších platformách, nejen na Windows. Také podporuje většinu nejnovějších funkcí SQL. [26]



4- Rozdíl mezi tradiční databázovou architekturou a SQLite [27]

Databáze bude rozdělena tak, aby uspokojovala veškeré potřeby aplikace. O uživatelské účty a role se postará rozhraní ASP.NET Core Identity, a to rozhraní vytvoří několik tabulek.

- AspNetRoles
- AspNetUserRoles
- AspNetUsers

Nejpodstatnější pro nás jsou AspNetRoles, kde budou uchovávány role, dále AspNetUserRoles, kde jsou zase uchovávány role uživatelů a jako poslední a nejdůležitější je tabulka AspNetUsers, která uchovává všechna data o uživatelích jako takových.
Další tabulky vytvořené přímo pro data uchovávané ve smyslu této aplikace:

- Cars
- CarReservations
- Notes
- Logs

4.2 Implementace webové aplikace

Pro založení projektu vyžaduje šablona dodatečné informace, první z nich je verze frameworku – pro tento případ je zvolena poslední verze, a to je .NET 6. Další informací je zvolení přihlašovacího typu, kde jsou na výběr individuální účty, Microsoft Identity platform, Windows anebo žádný typ. Pro tuto bakalářskou práci jsem zvolil žádný přihlašovací typ, protože funkcionalitu přihlašování a odhlašování provedeme zvlášť sami. Dalším požadavkem je, zda se má aplikace konfigurovat na použití protokolu HTTPS a Dockeru, a to tato webová aplikace nepotřebuje.

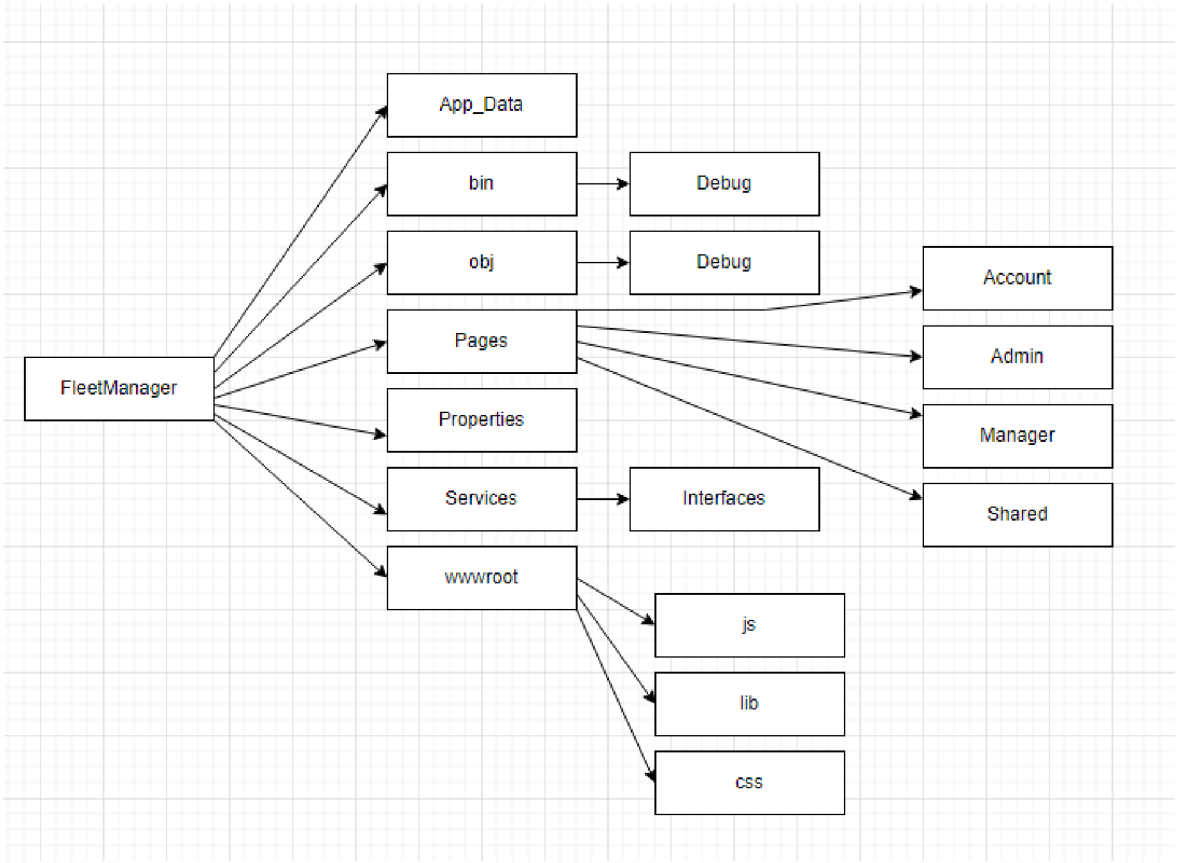
4.2.1 Souborová a adresářová struktura

Celá webová aplikace je rozdělena na dva „projekty“, kde jeden projekt – „FleetManager“ se stará o prezentační vrstvu a druhý – „FleetManager.Data“ se stará o data.

Projekt FleetManager

- App_Data
  - Adresář obsahuje databázový soubor, se kterým pracuje prezentační vrstva.
- Pages
  - Obsahuje všechny stránky aplikace, které může aplikace nějakým způsobem zpracovat a spustit. Složka Shared obsahuje sdílené layouty, které využívají defaultně všechny Razor Pages.
- Properties
  - Tento adresář obsahuje soubor se jménem „launchSettings.json“, který upřesňuje, jak se má aplikace spouštět, případně jestli má spustit zároveň i prohlížeč, na jaké adrese a tak podobně.
- Services
  - V adresáři Services se nachází podadresář Interfaces
  - Interfaces obsahuje vytvořená rozhraní – definici pro služby, které aplikace využívá.
  - Nalezneme zde naprogramované služby, které byly definovány v rozhraních.

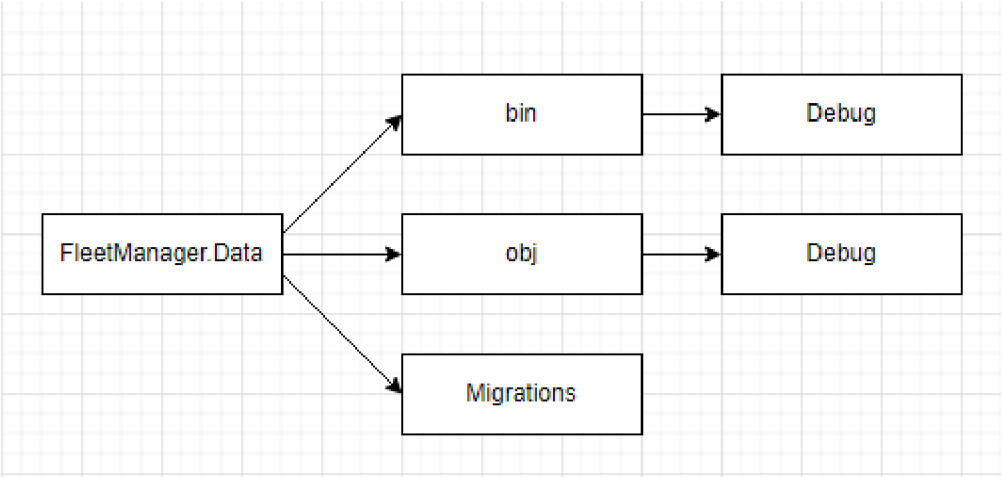
- Wwwroot
    - Veřejný adresář pro klienta při přístupu na server. V tomto adresáři se nacházejí složky pro kaskádové styly (css), JavaScript (js) a složka pro další knihovny využívané v aplikaci (lib).
- Kaskádové styly (css)
    - Jednoduchý css soubor pro základní styly.
- JavaScript (js)
    - Také jednoduchý základní JavaScriptový soubor.
- Knihovny (lib)
    - Obsahuje složku pro CSS framework bootstrap, kde nalezneme licenci a jeho soubory jak pro CSS, tak i pro JavaScript.
    - JQuery a jeho podknihovny (validation, validation-unobtrusive) je další JavaScriptovou knihovnou usnadňující interakci mezi JavaScriptem a HTML.
- Bin
    - Obsahuje adresář Debug, ve kterém se nachází design databáze a potřebné soubory pro debuggování aplikace.
- Obj
    - Obj taktéž obsahuje adresář debug, a veškeré informace z přídavných modulů. Oba tyto adresáře (bin a obj) spolupracují v debuggování.
- Soubory
    - V adresáři jsou podstatné soubory jako je „appsettings.json“, ve kterém se například nastavuje takzvaný DbContext, neboli cesta ke zdroji dat. Dalším hodně podstatným souborem je Program.cs, který nastavuje všechny potřebné služby a vytváří defaultní admin účet.



5- Adresářová struktura projektu FleetManager – vlastní tvorba

Projekt FleetManager.Data

- Bin
  - Stejně jako u prvního projektu obsahuje nástroje pro debuggování.
- Obj
  - Platí to samé jako u první projektu.
- Migrations
  - V tomto adresáři se nacházejí soubory týkající se migrací do databáze pomocí Entity Frameworku.
- Soubory
  - Mimo adresáře jsou třídy, které obsahují takzvané property (vlastnosti), ze kterých se poté pomocí Entity Frameworku dělají tabulky do databáze.



6 - Adresářová struktura projektu FleetManager.Data – vlastní tvorba

4.2.2 Databáze

V projektu FleetManager.Data jsou vytvořeny třídy podle jejich zaměření. Pomocí Entity Frameworku a jeho migrací jsem vytvořil jednoduše databázové tabulky. Pro příklad je tabulka vytvořena pomocí takovéto třídy.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FleetManager.Data
{
    public class Car
    {
        public int Id { get; set; }

        public string Brand { get; set; }

        public string Model { get; set; }

        public DateTime AssembledAt { get; set; }

        public DateTime TechnicalInspectionDate { get; set; }

        public string VIN { get; set; }

        public string Identificator { get; set; }

        public List<CarReservation> CarReservations { get; set; }
    }
}
```

Zdrojový kód 1 – Projekt FleetManager.Data třída Car

V Package Manager Console[6] ve Visual Studiu 2022 se provede pouze příkaz na přidání migrace „add-migration #NÁZEV-MIGRACE“ a tím se vytvoří soubor:

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace FleetManager.Data.Migrations
{
    public partial class CreateNewTableForCars : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Cars",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Brand = table.Column<string>(type: "TEXT", nullable:
false),
                    Model = table.Column<string>(type: "TEXT", nullable:
false),
                    AssembledAt = table.Column<DateTime>(type: "TEXT",
nullable: false),
                    TechnicalInspectionDate = table.Column<DateTime>(type:
"TEXT", nullable: false),
                    VIN = table.Column<string>(type: "TEXT", nullable: false),
                    Identificator = table.Column<string>(type: "TEXT",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Cars", x => x.Id);
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Cars");
        }
    }
}
```

Zdrojový kód 2 – Migrační soubor CreateNewTableForCars.cs

4.2.2.1 Tabulka Cars

Pro příklad s migrací, byla vybrána tabulka Cars, která poté byla vytvořena pomocí příkazu v Package Manager Console „update-database“. Tabulka obsahuje primární klíč ID,

[6] Package Manager Console – je konzole, kterou využívá Visual Studio k interakcím s NuGet [28].

který jednoznačně identifikuje vozidlo. Dále obsahuje značku, model vozidla, rok a měsíc výroby, poslední technickou kontrolu, VIN kód a SPZ.

Název	Typ	Poznámka
ID	Integer	Identifikátor pro vozidlo
Brand	Text	Značka vozidla
Model	Text	Model vozidla
AssembledAt	Text	Datum vozidla, které je převedeno poté na formát „MM / YY“
TechnicalInspectionDate	Text	Datum poslední technické kontroly, převedeno na stejný formát jako AssembledAt
VIN	Text	VIN od vozidla (identifikátor)
Identificator	Text	SPZ od vozidla

Tabulka 2 - Databázová tabulka – Cars

4.2.2.2 Tabulky AspNetRoles, AspNetUsers a AspNetUserRoles

Jak už bylo zmíněno, tyto tabulky jsou vytvořeny pomocí rozhraní Identity. Tabulka AspNetRoles uchovává čtyři prvky, tím je samozřejmě primární klíč ID role, název role, jeho název s velkým písmem a ConcurrencyStamp který chrání před konfliktem úprav.

Název	Typ	Poznámka
ID	Integer	Identifikátor role
Name	Text	Název role
NormalizedName	Text	Název role velkým písmem
ConcurrencyStamp	Text	Prevence před konfliktem úpravy

Tabulka 3 - Databázová tabulka – AspNetRoles

Tabulka AspNetUsers je tvořena sedmnácti prvky, uchovává informace o uživateli, některé prvky aplikace vůbec nevyužívá z důvodu nevyužitelnosti v tomto směru. V další verzi by bylo možné, že se pro některé prvky využití najde.

Název	Typ	Poznámka
Id	Integer	Identifikátor uživatele
UserName	Text	Uživatelské jméno
NormalizedUserName	Text	Uživatelské jméno velkým písmem
Email	Text	Email uživatele
EmailConfirmed	Integer	Potvrzení emailu, v této verzi aplikace se nevyužívá
PasswordHash	Text	Zašifrované uživatelské heslo
SecurityStamp	Text	Sledování změn uživatelského účtu
ConcurrencyStamp	Text	Prevence před konfliktem upravování uživatele
PhoneNumber	Text	Telefonní číslo uživatele
PhoneNumberConfirmed	Integer	Potvrzení telefonního čísla, v této aplikaci se nevyužívá
TwoFactorEnabled	Integer	Dvoufázové ověřování, v této aplikaci se nevyužívá
LockoutEnd	Text	Uzamčení účtu
LockoutEnabled	Integer	Platnost uzamčení účtu
AccessFailedCount	Integer	Počet neúspěšných přihlášení
FirstName	Text	Křestní jméno uživatele
LastName	Text	Příjmení uživatele

Tabulka 4 - Databázová tabulka – AspNetUsers

Tabulka AspNetUserRoles uchovává data, která určují přiřazení rolí uživatelům pomocí cizích klíčů.

Název	Typ	Poznámka
UserId	Cizí klíč typu Integer	Určuje, pro kterého uživatele platí role v druhém políčku
RoleId	Cizí klíč typu Integer	Určuje roli, kterou má přiřazen uživatel

Tabulka 5- Databázová tabulka – AspNetUserRoles

4.2.2.3 Tabulka CarReservations

V této tabulce se nachází informace, které potom tvoří přehled rezervací v aplikaci. Pomocí této tabulky se také zamezuje duplicitě, kdy například Uživatel A vytvoří rezervaci na vozidlo X na datum 2.2.2022 od 10:00 do 17:00 a Uživatel B by chtěl vytvořit rezervaci také na vozidlo X v čase, kdy už má rezervaci provedenou Uživatel A – aplikace rezervaci nevytvoří, právě získáním dat z této tabulky.

Název	Typ	Poznámka
Id	Integer	Identifikátor rezervace
UserId	Cizí klíč typu Integer	Ukazuje na uživatele, pro kterého je rezervace vytvořena
CarId	Cizí klíč typu Integer	Ukazuje na vozidlo, pro které je vytvořena rezervace
From	Text	Datum, od kterého platí rezervace
To	Text	Datum, do kterého platí rezervace

Tabulka 6- Databázová tabulka – CarReservations

4.2.2.4 Tabulka Notes

Tabulka obsahuje čtyři prvky, které se využívají pro záznam k určitým vozidlům (servisní prohlídky, závady, a tak podobně). Obsahuje primární klíč identifikace poznámky, dále cizí klíč pro přiřazení k určitému vozidlu, titul poznámky (Závada, Servisní prohlídka, a tak podobně), popis poznámky (Provedena výměna zadního světla, svítí pouze levé potkávací světlo, a tak dále).

Název	**Typ**	**Poznámka**
Id	Integer	Identifikátor rezervace
CarId	Cizí klíč typu Integer	Ukazuje na vozidlo, pro které byla poznámka vytvořena
Title	Text	Titul závady
Description	Text	Popis poznámky vozidla
Date	Text	Datum, kdy byla poznámka vytvořena

Tabulka 7- Databázová tabulka – Notes

4.2.2.5 Tabulka Logs

Tato tabulka byla vytvořena za účelem uchovávání záznamů o uživatelích, které akce provedli, co upravili a tak podobně.

Název	**Typ**	**Poznámka**
Id	Integer	Identifikátor rezervace
UserId	Cizí klíč typu Integer	Ukazuje na uživatele, který provedl akci
Type	Integer	Typ záznamu
Text	Text	Text záznamu (přímé informace co uživatel provedl)
Date	Text	Datum, kdy se akce stala

Tabulka 8 - Databázová tabulka – Logs

4.3 Úprava layoutu stránky

Aplikace již defaultně využívá kaskádové styly s pomocí frameworku **Bootstrap**, proto není za potřebí vkládat vlastní styly.

Při vytvoření projektu je vytvořen přímo i soubor s názvem _Layout ve složce Pages/Shared. V layoutu se nachází veškeré věci, co se týče základních prvků, které chceme na každé stránce. Mezi těmito prvky nalezneme například header, což může být například název stránky anebo také navigační menu, které určitě musí být upraveno z důvodu dosažitelnosti pro klasické uživatele.

Už v _Layout souboru se pracuje s rolemi, aby se v navigačním menu nezobrazovalo to, k čemu určitá role nemá přístup. Jedná se třeba o přehledy uživatelů, vozidel a jejich tvorba.

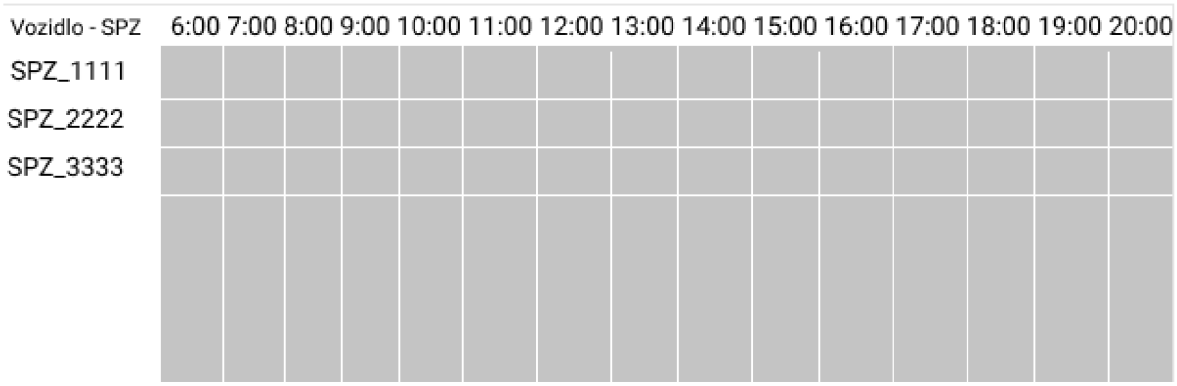
Téměř na konci souboru _Layout v elementu <*main*> se nachází metoda @RenderBody(), která vlastně vykreslí aktuálně požadovanou stránku (například http://127.0.0.1/Reservations).

4.4 Tvorba přehledu rezervací

Tato webová aplikace má za úkol spravovat firemní vozidla, tím pádem i jejich informace a využití zaměstnanci. Proto aplikace obsahuje i přehledy tohoto využití. K jeho vytvoření využíváme Razor Pages.

4.4.1 Vzhled přehledu rezervací

Přehled bude v tabulce, která bude obsahovat časy (od 06:00 do 20:00) které budou moci být zabrány uživateli. Do této tabulky budeme dodávat data z databáze (tabulka CarReservations) pomocí cyklů for a foreach. Náhled na přehled je v příloze číslo 6.



7- Návrh tabulky pro přehled rezervací – vlastní tvorba

4.4.2 ReservationsModel přehledu

V ReservationsModelu vytvoříme list tvořený z dat třídy Car. Datum musíme zohlednit též, protože přehled bude odvíjený od vybraného data nad tabulkou. Přidáváme také _carService, kterou jsme pro účely manévrování s daty vytvořili.

Pomocí _carService poté v metodě OnGet získáme data pro tabulku, která vozidla má přidat a získáme vazbu mezi tabulkami CarReservations a Cars.

4.4.3 Kód pro zobrazení přehledu

Reservations bude zobrazeno se souborem _Layout, tak jako každá stránka v této aplikaci. Tabulka bude vytvořena manuálně (časy) a poté do ní jsou vkládány získaná data z tabulky. Použití cyklu foreach nám dodává data získané v metodě OnGet do proměnné car, ze které pak čerpáme. V cyklu foreach je vytvořen další cyklus for, který vykresluje již zabrané hodiny pomocí vyplněných elementů *<td>*, obsahující jméno a příjmení uživatele, který má pro danou dobu vozidlo již rezervované, v tomto cyklu for je další cyklus, který má na starosti prázdné elementy *<td>*, z důvodu, aby tabulka vypadala přehledněji.

4.4.4 Využití LINQ

Pro získání rezervace pro dané vozidlo používáme LINQ metodu Where. V této metodě zadáváme takzvanou arrow function, která musí odpovídat parametrům, které potřebujeme získat pro správné zobrazení.

Výsledku, který požadujeme docílíme tím, že v metodě Where, budeme porovnávat den získaný z databáze s dnem zadaným nad tabulkou, zároveň s tím porovnáváme rok zadaný nad tabulkou s rokem rezervace z databáze. To vše setřídíme vzestupně pomocí další LINQ metody OrderBy dle prvku From z databáze. Tento výsledek převedeme do pole.

Dalším cyklem se už dostáváme do dosazování dat do tabulky. Cyklus vytváří další cyklus, který má na starosti vyplnit prázdná políčka. Během toho, co se zadávají prázdná políčka, původní cyklus zadává políčka s šířkou odpovídající délce trvání dané rezervace.

Po skončení cyklu zadávající data do tabulky je zde ještě jeden výplňkový cyklus, který doplní tabulku, aby nebyla ochuzena o prázdná políčka po rezervovaných polích.

```
@foreach (var car in Model.Cars)
{
      <tr>
      <th scope="row">@car.Identificator</th>
      @{

           int lastHour = 6;
           var reservation = car.CarReservations.Where(w => w.From.DayOfYear
== Model.Date.DayOfYear && w.From.Year == Model.Date.Year).
OrderBy(o => o.From).ToArray();

       }
       @for (int i = 0; i < car.CarReservations.Where(w => w.From.DayOfYear ==
Model.Date.DayOfYear && w.From.Year == Model.Date.Year).Count(); i++)
        {
            @for (int x = 0; x < reservation[i].From.Hour - lastHour; x++)
            {
                 <td></td>
            }

            <td colspan="@(reservation[i].To.Hour-reservation[i].From.Hour)"
class="table-active">@reservation[i].User.FirstName
@reservation[i].User.LastName</td>
             lastHour = reservation[i].To.Hour;
         }
         @for (int i = lastHour; i <= 20; i++)
         {

              <td></td>
         }

        </tr>
}
```

Zdrojový kód 3 – Dodávání dat do tabulky pro zobrazení rezervací pro vybraný den

4.5 Tvorba formulářů pro zakládání nových účtů a vozidel

Zakládání nových účtů a vozidel je dejme tomu na stejném principu. Rozdíl je, že vozidla může vytvářet role Správce a účty Administrátor. Další rozdíl je akorát v tom, že účty jsou tvořeny rozhraním Identity, tudíž využívají již vytvořené metody. Obě operace jsou zaznamenány pomocí vytvořené service LogService.

4.5.1 Vzhled formulářů

Návrh vzhledu formulářů není až tak potřebný, protože jdeme jednoznačně po nejvíce jednoduchém stylu formuláře. Informace, které formulář bude vyžadovat jsou:

- Uživatelské jméno
- Heslo
- Email
- Křestní jméno
- Příjmení
- Role

Pro vozidla budou informace samozřejmě odlišná:

- Značka
- Model
- Datum výroby
- Poslední technická kontrola
- VIN kód
- SPZ

Náhled na formuláře je v příloze číslo 2 a 3.

4.5.2 Model vytváření

Pro vozidla je vytvořena služba, která dokáže manipulovat s daty v databázi. Proto v modelu využíváme tuto službu (CarService) a její metodu Create, do které pouze zadáme parametry vozidla.

Účty jsou tvořeny pomocí rozhraní Identity, tím je i jejich správa zaopatřena službou, která se nazývá UserManager. Služba UserManager využívá i asynchronních metod, například metoda CreateAsync, kterou využíváme pro zakládání účtů. Tato metoda přidá uživatele do databáze.

Oba modely vytváření jsou zaznamenány pomocí služby LogService, tudíž je možné vytrasovat jejich původ (účet, ze kterého byly akce provedeny).

4.6 Přehledy vozidel a uživatelů

Přehled vozidel je tvořen „kartičkami“, ve kterých nalezneme informace o vozidlech. Mezi informacemi bude zahrnuta SPZ, model, značka, poslední technická kontrola, VIN a kdy bylo vozidlo vyrobeno.

Přehled uživatelů je vytvořen do tabulky, kde budou základní informace o uživateli. Obsahuje tři možné akce – informace, upravit a smazat.

4.6.1 Model přehledů

Model vozidel je opět jednoduchý. V metodě OnGet předává data do vytvořeného listu.

Model uživatelů je stejně jednoduchý v tom, že také získává data do vytvořeného listu uživatelů.

4.6.2 Zobrazení přehledů

Přehledy se zobrazují za použití šablony _Layout, která je pro všechny stránky této aplikace stejná.

Přehled uživatelů je zobrazen v tabulce, do které jsou dosazeny data získané v modelu, jsou zde další tři funkce – zobrazující informace o uživateli, úprava (téměř stejný formulář) a smazání (aplikace se dotáže, zda jste si opravdu jisti tím, co děláte).

Přehled vozidel je zobrazen v „kartičkách“, kde se nachází data získaná z databáze. U každého vozidla jsou možnosti, které vidí obyčejní uživatelé anebo možnosti, které vidí i role Správce a Administrátor. Uživatelé vidí pouze možnost Rezervace (další formulář pro vytvoření rezervace) a Poznámky k vozidlu (závady, servisní prohlídky a tak podobně). Vyšší role vidí upravit vozidlo (formulář stejný jako přidávání) a smazat vozidlo (též se dotáže, zda jste si opravdu jisti tím, co právě děláte).

Náhled na přehled vozidel a uživatelů je v příloze číslo 4 a 5.

4.7 Záznamy o uživatelských akcích

Aplikace obsahuje základní verzi uživatelských záznamů. Zapisuje téměř všechny možné akce, které lze v aplikaci provádět.

Pro vytvoření této funkce záznamů byla vytvořena služba se jménem LogService. Služba obsahuje metody na založení záznamu a jejich výpis pro vybraného uživatele.

4.7.1 Metoda GetUserLogs

Metoda přijímá parametry uživatelského id a count (počtu, který chce získat). Tato metoda využívá LINQ metodu Where, ve které se vyhledávají vhodné záznamy, dle kritéria že prvek z tabulky UserId se musí rovnat parametru id. Výsledek se roztřídí dle datumu, z toho se vybere pouze zadaný počet v parametru count a převede se do listu, který pak metoda vrací.

4.7.2 Metoda Log

Metoda Log přijímá tři parametry, prvním je typ (aktuální verze aplikace zaznamenává pouze typ INFO), druhým je zpráva záznamu a třetím je ID, které určuje, k jakému uživateli se záznam přiřazuje.

4.8 Testování

Při implementaci takto relativně malé aplikace, která má pouze pár funkcí, si myslím, že vhodné testování je proveditelné i samotným autorem. Proto v průběhu tvorby aplikace testuji veškeré možné chyby, které by mohly nastat. Z důvodu menšího rozsahu aplikace není třeba žádné více rozšířené testování. Samozřejmostí je, že se aplikace musí otestovat i v ostrém provozu po nasazení celé aplikace.

4.9 Nasazení webové aplikace

Pro nasazení webové aplikace jsem využil služeb společnosti DigitalOcean, kde jsem zakoupil takzvaný droplet[7]. Nasazení bylo poměrně složité, nicméně jakmile člověk zjistí, jak to vlastně funguje, znovu nasazení už není tak těžké. Za použití vývojového prostředí Visual Studio 2022 je nasazování o něco lehčí, protože vlastně připraví celou složku projektu na přesun. Visual Studio podporuje několik možných typů nasazování ať už přes cloudovou službu Azure, přes FTP[8], webový server (IIS[9]) anebo přes složku, což jsem provedl já.

Po nasazení jsem vyčistil celou databázi a vytvořil pár základních prvků pro případné zobrazení nějakou třetí osobou.

[7] Droplet – virtuální stroj založený na linuxu.
[8] FTP – File Transfer Protocol
[9] IIS – Internet Information Service – software pro hostování webových aplikací

5 Zhodnocení výsledků

Po náležitém otestování a nasazení webové aplikace jsem dospěl k několika možným vylepšením. Tyto vylepšení jsou stoprocentně reálné, a myslím si, že významně změní celkový potenciál aplikace:

- Záloha jednotlivých vozidel (umožnění smazání vozidla a následného dohledání znovu v databázi pro zpětné obnovení).
- Změna stavu rezervací pro role Správce či Administrátor (zamítnout, upravit či úplně vymazat ze systému).
- Zprávy mezi uživateli
- Přiřazení rezervací určitému uživateli k vybranému vozidlu
- Rezervace delšího trvání než pouze jeden den od 06:00 do 20:00
- Zobrazování rezervací pro automobily už v přehledu vozidel.

Tvorba webových aplikací za použití frameworků jako je ASP.NET Core velmi usnadňuje celkový proces vývoje. Existuje kvanta technologií, které opravdu usnadní vývoj. Jedním z příkladů je v této aplikaci využití rozhraní Identity. V aplikaci s ním pracujeme a opravdu ulehčilo spoustu času s tvorbou celkové služby pro správu uživatelů.

Výsledkem práce je aplikace, která má za úkol evidovat a spravovat vozový park firmy. Aplikace je zpracována v relativně jednoduchém stylu, ale myslím si, že své využití by určitě našla. Připojení k databázi je jednoduché, kvůli Entity Frameworku jsme schopni jednoduše přidávat další rozšíření databáze.

6 Závěr

První část teoretické části se týká představení frameworku určeného přímo pro tvorbu webových aplikací – ASP.NET Core (kapitola 3.1.1). V této části je také popsána softwarová architektura MVC a MVVM (kapitola 3.1.1.1 a 3.1.1.2), následně je zde popsán i rozdíl jejich porovnání (kapitola 3.1.1.3). Dále byly popsány technologie, které jsou velmi spjaté s vývojem webové aplikace v tomto frameworku (k nalezení v kapitolách 3.1.2 až 3.1.4) – LINQ, Entity Framework a Razor Pages. Další popsané technologie se týkají front-endu – HTML, JavaScript, CSS a jejich frameworky (Vue.JS a Bootstrap). Front-endové technologie jsou popsány v kapitole 3.1.5 až 3.1.8.1. Druhá část teoretické práce je věnována vývojovým prostředím, ve kterém je možné tvořit webové aplikace a jejich front-endové součásti. Vývojové prostředí jako takové, je popsáno v kapitole 3.2. Pro výslednou aplikaci se především využívalo vývojové prostředí Visual Studio 2022, pro vedlejší účely bylo využito Visual Studio Code.

První část praktické části práce spočívá v návrhu aplikace. Návrh aplikace se drží zjištěných poznatků z teoretické části, a proto byla zvolena softwarová architektura MVVM za použití technologie Razor Pages a pro datovou strukturu je využitý Entity Framework. Pro front-end aplikace byl zvolen samozřejmě značkovací jazyk HTML a kaskádové styly ve formě frameworku Bootstrap. Uchovávání dat je řešeno pomocí SQLite. V druhé části je v kapitole 4.2 popsána implementace webové aplikace. Kapitola 4.2.1 popisuje souborovou a adresářovou strukturu a její zobrazení na obrázku, v kapitole 4.2.2 je popsáno jak se pomocí Entity Frameworku a jeho migracím dají jednoduše vytvořit databázové tabulky pro účely aplikace. Dále v této kapitole jsou ukázány veškeré tabulky, které se využívají pro navrhnuté funkce aplikace. V kapitole 4.4 je ukázka tvorby jedné z předních funkcí celé výsledné aplikace – přehledu rezervací. Přiložený zdrojový kód ukazuje tvorbu takovéhoto přehledu. Další funkce jsou aplikace jsou tvořeny v kapitolách 4.5 až 4.7.2. Během testování v kapitole 4.8 nebyly zjištěny žádné závažné chyby z důvodu přímého odlaďování aplikace při vývoji. Ani po nasazení aplikace do ostrého provozu se nevyskytli žádné potíže.

Během lokálního i hostovaného testování mě napadlo několik možných dalších rozšíření a jsou zmíněny v kapitole 5 – Zhodnocení výsledků. Při vývoji jsem zjistil, že velmi užitečnou věcí je tvorba takzvaných služeb, které je možné si naimportovat do jakékoliv RazorPage a dále s nimi pracovat. Další užitečnou věcí je možnost využívat již vytvořená rozhraní (například rozhraní Identity pro přihlašování a správu uživatelských účtů).

Zdrojový kód k výsledné webové aplikaci se nachází na přiloženém CD. K zobrazení zdrojových kódu či spuštění webové aplikace musíte mít vhodné vývojové prostředí – nejideálněji Visual Studio 2022. Pro nahlédnutí do databáze je možné využít volně přístupný software DB Browser for SQLite.

Pro přihlášení do webové aplikace je vytvořen speciální účet s nejvyššími právy. Přihlašovací údaje naleznete na přiloženém CD.

7 Seznam použitých zdrojů

[1] A. Lock, ASP.NET Core in Action, Second Edition, United States: MANNING, 2021. ISBN: 978-1617298301

[2] S. Smith, „Overview of ASP .NET Core MVC," 30 9 2017. [Online]. Available: https://studreadywork.ru/wp-content/uploads/2021/09/Overview-of-ASP.NET-Core-MVC.pdf. [Přístup získán 12 2 2022].

[3] M. Contributors, „MVC," 18 2 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/MVC. [Přístup získán 21 2 2022].

[4] S. Smith, „Overview of ASP.NET Core MVC," 18 2 2022. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0. [Přístup získán 23 2 2022].

[5] TechTarget, „What is Model-View-ViewModel(MVVM)?," 9 8 2020. [Online]. Available: https://whatis.techtarget.com/definition/Model-View-ViewModel. [Přístup získán 10 2 2022].

[6] Microsoft, „The Model-View-ViewModel Pattern," 7 8 2021. [Online]. Available: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. [Přístup získán 12 2 2022].

[7] M. Martin, „Guru99," 5 2 2022. [Online]. Available: https://www.guru99.com/mvc-vs-mvvm.html. [Přístup získán 13 2 2022].

[8] BillWanger, zspitz a a další, „Language Integrated Query (LINQ) (C#)," 18 2 2022. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [Přístup získán 20 2 2022].

[9] TutorialsTeacher, „What is LINQ?," 25 7 2021. [Online]. Available: https://www.tutorialsteacher.com/linq/what-is-linq. [Přístup získán 20 2 2022].

[10] Microsoft, „Introduction to LINQ Queries (C#)," Microsoft, 15 9 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries. [Přístup získán 7 2 2022].

[11] Microsoft, „Entity Framework Core," Microsoft, 25 5 2021. [Online]. Available: https://docs.microsoft.com/en-us/ef/efcore-and-ef6/. [Přístup získán 15 2 2022].

[12] EntityFrameworkTutorial.net, „What is Entity Framework?," 13 11 2021. [Online]. Available: https://www.entityframeworktutorial.net/what-is-entityframework.aspx. [Přístup získán 20 2 2022].

[13] D. B. a. K. L. Rick Anderson, „Introduction to Razor Pages in ASP.NET Core," 18 2 2022. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio. [Přístup získán 24 2 2022].

[14] MDN Web Docs, „HTML: HyperText Markup Language," Mozilla, 18 2 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Přístup získán 21 2 2022].

[15] D. Goodman, Dynamic HTML: The Definitive Reference, 2. editor, Sebastopol: O'Reilly & Associates, Inc., 2002, pp. 9-10. ISBN: 978-0596003166

[16] D. Čápka, „ITNetwork.cz," 22 4 2021. [Online]. Available: https://www.itnetwork.cz/html-css/bootstrap/kurz/uvod-do-css-frameworku-bootstrap. [Přístup získán 15 2 2022].

[17] Bootstrap, „Bootstrap v5.1,“ 4 8 2021. [Online]. Available: https://getbootstrap.com/docs/5.1/getting-started. [Přístup získán 18 2 2022].

[18] MDN Web Docs, „JavaScript,“ 28 7 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Přístup získán 8 2 2022].

[19] MDN Web Docs, „About JavaScript,“ 26 7 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Přístup získán 7 2 2022].

[20] Vue.js, „Introduction,“ 7 2 2022. [Online]. Available: https://vuejs.org/guide/introduction.html. [Přístup získán 18 2 2022].

[21] R. Stair a G. Reynolds, Fundamentals of Information Systems, Boston: Cengage Learning, 2011. ISBN: 978-1305082168

[22] Suse, „Development Environment,“ Suse, 2022. [Online]. Available: https://www.suse.com/suse-defines/definition/development-environment/. [Přístup získán 7 3 2022].

[23] Microsoft, „Documentation for Visual Studio Code,“ 13 2 2022. [Online]. Available: https://code.visualstudio.com/docs. [Přístup získán 17 2 2022].

[24] Psypherium, NerveFalcon a a další, „Microsoft Visual Studio,“ 7 1 2022. [Online]. Available: https://www.wikiwand.com/en/Microsoft_Visual_Studio. [Přístup získán 18 2 2022].

[25] Microsoft, „Porovnání edicí sady Visual Studio 2022,“ 9 2 2022. [Online]. Available: https://visualstudio.microsoft.com/cs/vs/compare/. [Přístup získán 26 2 2022].

[26] J. Kreibich, Using SQLite, Sebastopol: O'Reilly Media, Inc., 2010, pp. 1-3. ISBN: 978-0596521189

[27] Devopedia, „SQLite,“ 15 2 2022. [Online]. Available: https://devopedia.org/sqlite. [Přístup získán 3 3 2022].

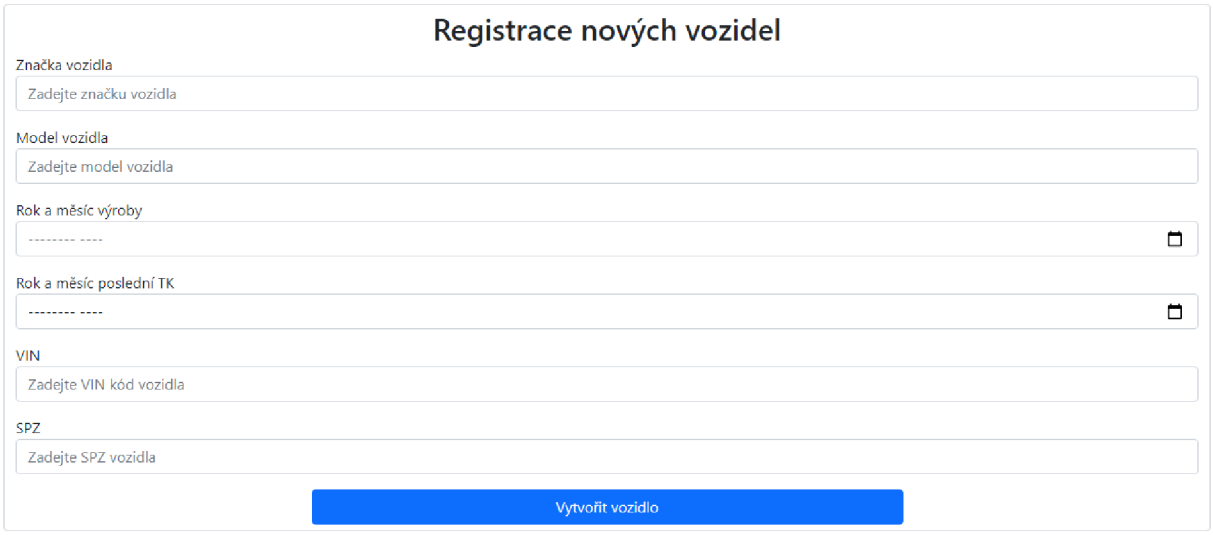
[28] Microsoft, „What is NuGet and what does it do?,“ Microsoft, 2 2 2022. [Online]. Available: https://docs.microsoft.com/en-us/nuget/what-is-nuget. [Přístup získán 6 3 2022].

[29] Geeksforgeeks, „Benefit of using MVC,“ Geeksforgeeks, 30 6 2021. [Online]. Available: https://www.geeksforgeeks.org/benefit-of-using-mvc/. [Přístup získán 14 2 2022].

8 Přílohy

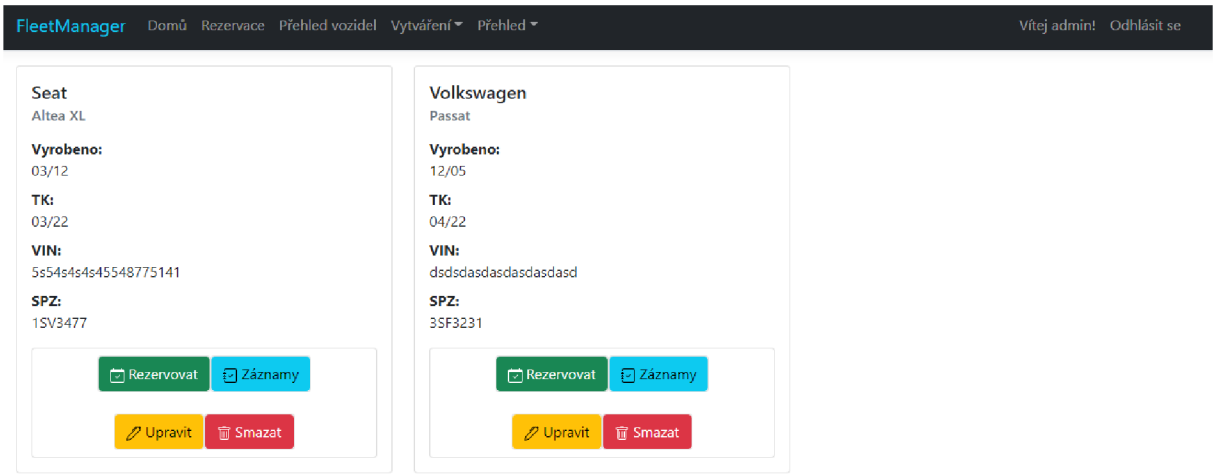
Příloha 1 – CD se zdrojovým kódem

Příloha 2 – Formulář tvorby vozidel

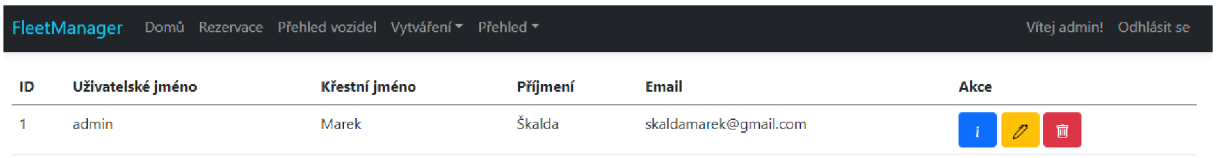


Příloha 3 – Formulář tvorby nových uživatelských účtů

Příloha 4 – Přehled vozidel



Příloha 5 – přehled uživatelů



Příloha 6 – přehled rezervací na vozidla