Návrh aplikace pro evidenci pracovních cest zaměstnanců

Tomáš Ševců

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Ševců**
Osobní číslo:	**A17400**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Návrh aplikace pro evidenci pracovních cest zaměstnanců**

Zásady pro vypracování

1. Proveďte rešerši existujících řešení.
2. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
3. Proveďte rozbor a analýzu požadavků na zvolené řešení.
4. Realizujte navrženou aplikaci.
5. Věnujte pozornost zabezpečení aplikace.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. NEUSTADT, Ila; ARLOW, Jim. UML 2 a unifikovaný proces vývoje aplikací. Computer Press, Albatros Media as, 2016.
2. JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3: training guide. Redmond, Wash. Microsoft, 2013. ISBN 978-0735674387.
3. UNHELKAR, Bhuvan. Software engineering with uml. Auerbach Publications, 2017.
4. FREEMAN, Adam. Pro Asp. net Core Mvc. Apress, 2016.
5. JAKOBUS, Benjamin. Mastering Bootstrap 4. Master the latest version of Bootstrap 4 to build highly customized responsive web apps. Packt Publishing Ltd, 2018.
6. BEN-GAN, Itzik; DAVIDSON, Louis; VARGA, Stacia. MCSA SQL Server 2016 Database Development Exam Ref 2-pack: Exam Refs 70-761 and 70-762 Microsoft Press, 2017.

Vedoucí bakalářské práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **20. prosince 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Ve Zlíně dne 20. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Cílem této práce bylo vytvořit webovou aplikaci pro správu služebních cest zaměstnanců. Důležitou součástí služební cesty jsou cestující, firmy a dárky, které navštíví a darují a dopravní prostředek. Struktura schvalování, kde každá služební cesta je schvalována podle cestujících a jejich vedoucích oddělení. Uživatelé navíc mohou mít role, díky kterým mají v aplikaci další možnosti jako správa dopravních prostředků nebo správa dárků.

Pro realizaci byl zvolen programovací jazyk ASP.NET Core Razor Pages, Entity Framework, LINQ

Klíčová slova:

Služební cesty, webová aplikace, správa uživatelů, struktura uživatelů, HTML, ASP.NET Core Razor Pages, LINQ, Entity Framework

ABSTRACT

The objective of this work was to create web application for managing business trips. Important part of business trip are passengers, companies and gifts which they visit and give and vehicle. Structure of approving, where each business trip is approved according to the passengers and their supervisors of department. In addition, users can have roles that give them additional options in the application, such as vehicle management or gift management.

For implementation was chosen programming language ASP.NET Core Razor Pages, Entity Framework, LINQ.

Keywords:

Business Trips, web application, user management, user structure, HTML, ASP.NET Core Razor Pages, LINQ, Entity Framework

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Plánování služebních cest je důležitou součástí mnoha firem. Pro velké firmy s mnoha zaměstnanci je nutné, aby bylo schvalování rozděleno mezi každé oddělení, protože jediný člověk by nemohl mít přehled o všech kancelářích a odděleních.

Cílem této práce je návrh a realizace webové aplikace pro správu služebních cest a k nim patřícím dopravních prostředků a dárků. Požadavky jsou autorizace a autentizace uživatelů, vytváření služebních cest, jejich schválení příslušným nadřízeným, správa uživatelů, oddělení, dopravních prostředků a dárků.

Teoretická část obsahuje rešerši existujících řešení, jejich porovnání a rozdíl s vytvořenou aplikací. Rozbor technologií se zaměřením na ASP.NET. Popis zabezpečení a možných hrozeb webových aplikací a jak se proti nim v ASP.NET Core bránit.

Praktická část je rozdělená na dvě kapitoly. První kapitola se zaměřuje na plánování a cíle práce. Obsahuje návrh požadavků, diagram případu užití UML, E-R diagram a diagramy popisující procesy při práci se služebními cestami. Druhá kapitola popisuje vzhled aplikace, celkovou práci se služebními cestami v aplikaci, správu firem, dopravních prostředků, dárků, uživatelů.

Závěrečná kapitola shrnuje výsledky práce a jejich dosažení.

I. TEORETICKÁ ČÁST

1 ROZBOR PROBLEMATIKY

1.1 Aplikace pro služební cesty

Na internetu se nachází několik profesionálně vytvořených aplikací, které se zabývají správou lidských zdrojů (Human resources HR) v organizaci. Obsahují software pro personalistiku, mzdy, nábor, docházku, plánování směn, plánování dovolené, vzdělávání, hodnocení, stravování nebo služební cesty. Tyto aplikace jsou velmi komplexní a zajišťují správu nad všemi částmi firmy, kde je potřeba řídit zaměstnance. Kvůli své všestrannosti a složitosti jsou většinou licencované a placené měsíčně například podle počtu zaměstnanců firmy. Pro porovnání jsem vybral pět aplikací, které poskytují určitou možnost zapisování služebních cest.

Vema

Softwarová společnost Vema nabízí podnikový informační systém Vema. Nabízí kompletní řešení v oblasti informačních systémů pro řízení lidských zdrojů, ekonomi a logistiky v organizacích. Mimo jejich aplikace pro řízení kompletních lidských zdrojů nabízejí i aplikaci nazvanou Pracovní cesty. Ta umožňuje plánování, schvalování, zálohy, měnu, vyúčtování, výstupy do účetnictví a mezd. Dále obsahuje evidenci vozidel nebo možnost spolujízdy.

Vema. Plánování a evidence pracovních cest [online]. [cit. 2020-07-25]. Dostupné z: https://www.vema.cz/cs/pracovni-cesty

Kompas2

Společnost PC HELP nabízí aplikaci Kompas2. Nabízí souhrnné řízení lidských zdrojů. Obsahuje moduly pro personalistiky, hodnocení zaměstnanců, evidenci docházky, vzdělávání, zpracování mezd, stravování, uchazeče o zaměstnání a pracovní cesty. Modul Pracovní cesty umožňuje schvalování, jedno nebo vícekrokové schvalování, upozorňování na termíny, akce a činnosti, výpočet stravného, práce s různými měnami, spolucestující, použití soukromého vozidla, proplácení náhrad, zálohy na výdaje. V současné době už se nenabízí a PC HELP ji nahrazuje aplikací plusPortal.

Kompas2. Personalistika, mzdy a HR [online]. [cit. 2020-07-25]. Dostupné z: https://www.kompas2.cz/

plusPortal

Aplikace plusPortal je nová aplikace pro správu lidských zdrojů od společnosti PC HELP. Jedná se o personální systém, který lze zakoupit po jednotlivých modulech nebo jako celek. Jeho výhodou je možnost napojení na existující systémy. Obsahuje stejné moduly jako jeho předchůdce Kompas2. Modul pro pracovní cesty je zde pojmenován Cesty a výdaje. Obsahuje různé způsoby schvalování a zabývá se především vyúčtováním a jeho zpracováním účtárnou.

PlusPortal. Personální a mzdový software [online]. [cit. 2020-07-25]. Dostupné z: https://www.plusportal.cz/

PowerKey

Společnost Advent nabízí aplikaci PowerKey. Aplikace pokrývá tyto oblasti identifikačních systému: Docházkový systém, Přístupové systémy a Stravovací systémy. Docházkový systém obsahuje evidenci pracovní doby, docházkový terminál, výpočet mezd. Dále obsahuje modul Správa akcí pro evidenci opakujících akcí, jako jsou zdravotní prohlídky nebo školení zaměstnanců a modul Správa cestovních náhrad pro evidenci služebních cest, jejich schvalování a výpočtu cestovních náhrad.

Advent. Software PowerKey [online]. [cit. 2020-07-25]. Dostupné z: https://www.advent.cz/produkty/software-powerkey/

AUTOPLAN

Společnost KROBSOFTWARE nabízí aplikaci AUTOPLAN. Od ostatních zmíněných aplikacích se liší především tím, že se zabývá výhradně správou vozového parku a vedení agendy pracovních cest. Obsahuje tři aplikace Kniha jízd, Cestovní příkazy a Automapa. Kniha jízd slouží k evidenci a vyhodnocování provozu firemních i soukromých vozidel. Cestovní příkazy slouží k evidenci a vyúčtování pracovních cest, včetně schvalování. Automapa je pomocný program k dvěma předchozím sloužící k vyhledání optimálního silničního spojení mezi dvěma nebo více bodu v automapě.

AUTOPLAN. Kniha jízd | Cestovní příkazy | Automapa [online]. [cit. 2020-07-25]. Dostupné z: https://www.autoplan.cz/

Následně každou aplikaci zhodnotím podle požadavků:

1. Je to samostatná aplikace?
2. Je zdarma nebo zpoplatněná?
3. Je webová aplikace?
4. Obsahuje strukturu schvalování?
5. Možnost spolucestujících?
6. Možnost dopravních prostředků?
7. Obsahuje evidenci výdajů / vyúčtování?

Tabulka 1: Přehled aplikací SC

Software	Samostatná aplikace	Zdarma	Webová aplikace	Strukturu schvalování	Spolucestující	Dopravní prostředky	Vyúčtování
Vema	ANO	NE	ANO	ANO	ANO	ANO	ANO
Kompas2	NE	NE	ANO	ANO	NE	NE	ANO
plusPortal	ANO	NE	ANO	ANO	NE	NE	ANO
PowerKey	NE	NE	ANO	NE	NE	ANO	ANO
AUTOPLAN	ANO	NE	NE	NE	ANO	ANO	ANO

Všechny zmíněné aplikace jsou zpoplatněné. Proto jsem čerpal pouze z toho, co o sobě píšou na svých webech.

Některé obsahují rozsáhlé možnosti mimo služební cesty pro správu celé firmy a služebním cestám se věnují hlavně pro jejich snazší vyúčtování. Nejlépe vychází software Vema, splňuje nejvíce požadavků a nabízí opravdu hodně možností, jak se služební cestou a vlastnostmi kolem ní zacházet a evidovat.

1.2 Specifikace aplikace

Aplikace Služební cesty navrhnutá a vytvořená v této bakalářské práci má několik specifikací a požadavků, kvůli kterým se liší od ostatních aplikací. Její základní funkcionalita je vytvoření a schválení služební cesty nadřízeným. Služební cesta se skládá z:

- Hlavních informací jako je název, místo, datum
- Cestujících, kteří se SC zúčastní
- Firem, které se během SC navštíví
- Dárků, které se rozdají během SC jednotlivým firmám
- Dopravního prostředku, který je buď vybrán nebo později přiřazen

Z toho vyplývají další funkcionality:

- Schválení SC nadřízeným jednoho z cestujících
- Uživatel s právy pro správu a vydávání dárků
- Uživatel s právy pro správu a přidělování dopravních prostředků
- Správa uživatelů – přiřazení práv jednotlivým uživatelům, nastavení struktury schvalování
- Zápis průběhu návštěvy u jednotlivých firem po skončení SC

Všechny aktivity, úpravy a schválení se provádějí na webovém rozhraní a zaznamenávají se do databáze.

Předpokladem pro využití aplikace Služební cesty je firma, která má přes 200 zaměstnanců rozdělených do několika oddělení. Zaměstnanci během svých služebních cest navštěvují předem definované, známé firmy, kterým při své návštěvě během služební cesty mohou darovat dárky. Dále se předpokládá, že firma s tolika zaměstnanci může mít 20 a více služebních aut. Při takové počtu dopravních prostředků se o jeho přiřazování k služebním cestám stará určitý uživatel aplikace. Podobně je tomu u dárků. Dárky pro služební cesty a konkrétní firmy jsou předávány určitým uživatelem, který si o tom v aplikaci vede přehled.

Touto strukturou jako je přiřazování aut a předávání dárků ke služebním cestám je tato aplikace odlišná od ostatních a je vyvíjena podle těchto daných požadavků firmy.

1.3 Rozbor použitých technologií

1.3.1 SQL

Jazyk SQL (Structured Query Language) a relační databázové systémy založené na něm představují jednu z nejdůležitějších základních technologií v počítačovém průmyslu. Stovky databázových produktů nyní podporují SQL, běžící na počítačových systémech od sálových počítačů po osobní počítače. Databáze založena na SQL může být dokonce zabudována do mobilního telefonu nebo PDA nebo do zábavního systému automobilu. Oficiální mezinárodní standard SQL byl několikrát přijat a rozšířen. Každý hlavní podnikový softwarový produkt se při správě dat spoléhá na SQL a SQL je jádrem stěžejních databázových produktů od společností Microsoft, Oracle a IBM. [1]a.[1]

SQL je nástroj pro organizování, správu a získávání dat uložených v počítačové databázi. Ve skutečnosti SQL pracuje s jedním specifickým typem databáze, který se nazývá relační databáze. [1]a.[1]

SQL se používá k řízení všech funkcí, které DBMS (systém řízení báze dat) poskytuje svým uživatelům:

- Definice dat SQL umožnuje uživateli definovat strukturu a organizaci uložených dat a vztahy mezi uloženými datovými položkami – příkazy pro definici dat (CREATE, ALTER, DROP)
- Vyhledávání dat SQL umožňuje uživateli nebo aplikačnímu programu načíst uložená data z databáze a použít je – příkazy pro vyhledávání dat (SELECT)
- Manipulace s daty SQL umožňuje uživateli nebo aplikačnímu programu aktualizovat databázi přidáním nových dat, odstraněním starých dat a úpravou dříve uložených dat – příkazy pro manipulaci s daty (INSERT, DELETE, UPDATE)
- Řízení přístupu SQL lze použít k omezení schopnosti uživatele získávat, přidávat a upravovat data a chránit uložená data před neoprávněným přístupem – příkazy pro řízení přístupových práv (GRANT, REVOKE)
- Sdílení dat SQL se používá ke koordinaci sdílení dat současnými uživateli a zajišťuje, že změny provedené jedním uživatelem neúmyslně nevymažou změny provedené téměř současně jiným uživatelem.
- Integrita dat SQL definuje omezení integrity v databázi a chrání je před poškozením v důsledku nekonzistentních aktualizací nebo selhání systému. [1]a.[1]

Jazyk SQL má mnoho rozšíření, z nichž jsou nejznámější T-SQL od firmy Microsoft a PL/SQL od firmy Oracle.

1.3.2 ASP.NET

ASP Classic

Classic ASP nebo Active Server Pages byla první skriptovací platforma společnosti Microsoft, která zpracovávala data na straně serveru. To umožnilo uživatelům vytvářet interaktivní a dynamické webové stránky. Byla poprvé vytvořena a vydána v roce 1996. [18]

Programovací jazyky, které se u ASP používají jsou VBScript a JScript. Všechny jazyky se prolínají v jednom souboru, což vedlo k nepřehlednému kódu. [1]a.[18]

V současné době se již nepoužívá.

ASP.NET

První verze ASP.NET byla vydána počátkem roku 2002 jako součást .NET Frameworku pro tvorbu webových aplikací a služeb a nástupcem ASP Classic. Technologie ASP.NET je postavena na technologii Common Language Runtime (CLR), která umožňuje programátorům psát kód pomocí libovolného jazyka .NET, například Visual Basic, JScript, C# a další. ASP.NET proto není programovací jazyk, ale svazek technologií.

ASP.NET podporuje řadu programovacích modelů pro vytváření webových aplikací jako jsou ASP.NET WebForms, ASP.NET AJAX, ASP.NET MVC, ASP.NET Dynamic Data, ASP.NET WebAPI nebo ASP.NET SignalR. [1]a.[2]

Technologie ASP.NET WebForms zjednodušuje přechod vývojářů z vývoje aplikací systému Windows na vývoj webových aplikací tím, že nabízí možnost vytvářet stránky složené z ovládacích prvků podobných uživatelskému rozhraní systému Windows. Ovládací prvek webu, například tlačítko nebo štítek, funguje téměř stejným způsobem jako jeho protějšky Windows: kód může přiřadit své vlastnosti a reagovat na své události. Ovládací prvky vědí, jak se mají vykreslit: zatímco ovládací prvky Windows se přitahují na obrazovku, webové ovládací prvky vytvářejí segmenty HTML a JavaScript, které tvoří části výsledné stránky odeslané do prohlížeče koncového uživatele. [1]a.[3]

Další výhody jsou kompilovaný kód, díky kterému běží aplikace rychleji a více chyb návrhu je zachyceno ve vývojové fázi. Zachycování chyb při běhu a zpracovávání výjimek pomocí

bloků try-catch. Možnost používat objektově orientovaný model pro programovaní stránek a ovládacích prvků. Možnost používat Code-behind model k oddělení logiky od vizuální stránky. [1]a.[3]

Poslední oficiálně vydaná verze nese označení ASP. NET 4.8 a byla představena v dubnu roku 2019. [1]a.[4]

V roce 2016 vyšla jako ASP.NET Core 1.0, která přináší zásadní změny architektury i principů fungování celé platformy. [1]a.[4]

ASP.NET Core

V roce 2015 společnost Microsoft oznámila nový směr pro ASP.NET a MVC Framework, který by nakonec produkoval ASP.NET Core MVC.

ASP.NET Core je vysoce výkonná Open-Source platforma pro různé platformy, která umožňuje vytvářet moderní cloudové aplikace připojené k Internetu. [1]a.[2]

Technologie ASP.NET Core je postavena na rozhraní .NET Core, což je multiplatformní verze rozhraní .NET Framework bez rozhraní pro programování aplikací specifických pro Windows (API). Windows jsou stále dominantním operačním systémem, ale webové aplikace jsou stále častěji hostovány v malých a jednoduchých kontejnerech v cloudových platformách a přijetím přístupu napříč platformami Microsoft rozšířil dosah .NET a umožnil nasazení ASP.NET Core aplikací do širší sady hostitelských prostředí a jako bonus umožnilo vývojářům vytvářet webové aplikace ASP.NET Core v systémech Linux a OS X / macOS. ASP.NET Core je zcela nový rámec. Je to jednodušší a je snazší s ním pracovat. A protože je založen na .NET Core, podporuje vývoj webových aplikací na řadě platforem a kontejnerů. [1]a.[2]

ASP.NET Core nabízí následující výhody [5]:

- Jednotné prostředí pro vytváření webového uživatelského rozhraní a webových rozhraní API.
- Navrženo pro testování.
- Razor Pages usnadňuje a pružně zakódovat scénáře zaměřené na stránku.
- Blazor umožňuje používat C# v prohlížeči spolu s JavaScriptem. Sdílení logiky aplikace na straně serveru a na straně klienta, které jsou napsané pomocí .NET
- Možnost vyvíjet a spouštět v systémech Windows, macOS a Linux.

- Architektura zaměřená na open-source a komunitu
- Integrace moderní architektury klienta a vývojových pracovních postupů
- Podpora hostování služeb vzdáleného volání procedur (RPC) pomocí gRPC.
- Konfigurační systém založený na prostředí, který je připravený pro cloud.
- Integrovaná injektáž závislostí.
- Odlehčený, vysoce výkonný, modulární kanál požadavků HTTP
- Možnost hostování v následujících aplikacích:
    - Kestrel
    - IIS
    - HTTP.sys
    - Nginx
    - Apache
    - Docker
- Souběžná Správa verzí.
- Nabízí nástroje, které usnadňují vývoj moderních webů.

ASP.NET Core MVC

ASP.NET Core MVC poskytuje funkce původního ASP.NET MVC Framework postaveného na nové platformě ASP.NET Core. Zahrnuje funkce, které dříve poskytovalo rozhraní Web API, zahrnuje přirozenější způsob generování komplexního obsahu a činí klíčové vývojové úkoly, jako je testování jednotek, jednodušší a předvídatelnější. [2]

Technologie ASP.NET Core MVC se řídí vzorem zvaným model-view-controller (MVC), který řídí tvar ASP.NET webové aplikace a interakce mezi komponenty, které obsahuje.

Je důležité rozlišovat mezi architektonickým vzorem MVC a implementací ASP.NET Core MVC. Vzor MVC není nový – datuje se do roku 1978 a projektu Smalltalk v Xerox PARC, ale dnes získal popularitu jako vzor pro webové aplikace z následujících důvodů [2]:

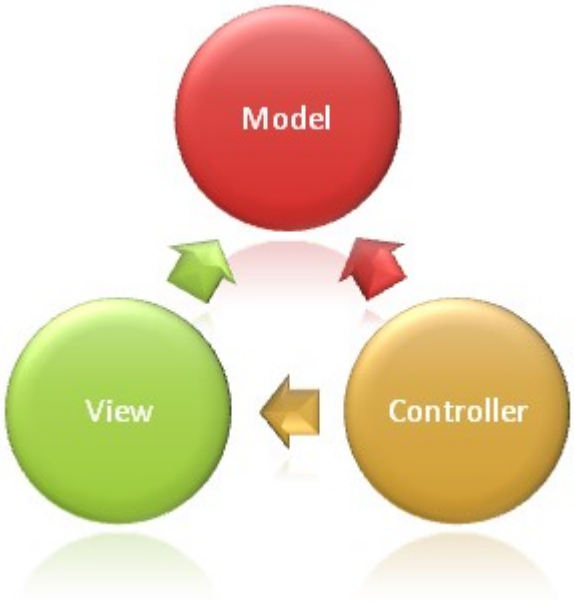
- Interakce uživatele s aplikací, která dodržuje vzor MVC, probíhá přirozeným cyklem: uživatel provede akci a v reakci na ni aplikace změní svůj datový model a doručí uživateli aktualizovaný pohled. A pak se cyklus opakuje. Toto je vhodné pro webové aplikace dodávané jako série požadavků HTTP a odpovědí.

- Webové aplikace vyžadují kombinaci několika technologií (například databází, HTML a spustitelného kódu), obvykle rozdělených do sady úrovní nebo vrstev. Vzory, které vznikají z těchto kombinací, se přirozeně mapují na koncepty ve vzoru MVC.

Technologie ASP.NET Core MVC implementuje vzor MVC, a přitom poskytuje výrazně lepší oddělení zodpovědností ve srovnání s webovými formuláři. [2]

Architektonický vzor architektury MVC (Model-View-Controller) odděluje aplikaci do tří hlavních skupin komponent: Model (model), View (pohled) a Controller (řadič). Tento model pomáhá dosáhnout oddělení zodpovědností. Pomocí tohoto modelu jsou požadavky uživatelů směrovány na Controller, který je zodpovědný za práci s Modelem pro provádění akcí uživatele nebo načtení výsledků dotazů. Controler zvolí View, které se zobrazí uživateli, a poskytne mu veškerá data Modelu, která vyžaduje. [6]

Následující diagram znázorňuje tři hlavní komponenty a odkazy na ostatní:



Obr. 1: Vzor MVC [6]

Tato změna odpovědnosti pomáhá škálovat aplikace z hlediska složitosti, protože je snazší kód, ladit a testovat něco (Model, View nebo Controller) s jednou úlohou.

Povinnosti modelu

Model v aplikaci MVC představuje stav aplikace a veškerou aplikační logiku nebo operace, které by měla provádět. Aplikační logika by měla být zapouzdřena do modelu spolu s jakoukoli implementační logikou pro přetrvávání stavu aplikace. [6]

Povinnosti View

Views (pohledy) jsou zodpovědná za prezentaci obsahu prostřednictvím uživatelského rozhraní. Používají pohledový stroj Razor k vložení .NET kódu do značek HTML. Ve Views (pohledech) by měla být minimální logika a jakákoli logika v nich by se měla vztahovat k prezentačnímu obsahu. [6]

Povinnosti Controller

Controllers (řadiče) jsou komponenty, které zpracovávají interakci uživatele, pracují s Model (modelem) a nakonec vybírají View (pohled), který se má vykreslit. V aplikaci MVC View (pohled) zobrazuje pouze informace; Controller (řadič) zpracovává a reaguje na vstup a interakci uživatele. Ve vzoru MVC je Controller (řadič) počátečním vstupním bodem a je zodpovědný za výběr typů Model (modelů), se kterými chcete pracovat a které zobrazení se má vykreslit (odtud jeho název – Controller controls (řadič řídí), jak aplikace reaguje na daný požadavek). [6]

ASP.NET Core Razor Pages

ASP.NET Core umožňují použít k vývoji framework ASP.NET Core Razor Pages, které mohou usnadnit a zvýšit produktivitu zaměřených na jednotlivé stránky aplikace než použití Controllers (řadičů) a Views (pohledů). [7]

Kód vypadá podobně jako ve View (pohledu) Razor používaný v aplikaci ASP.NET Core s Controllers (řadiči) a Views pohledy. Odlišuje se direktivou @page. @page vytvoří soubor, který zpracovává požadavky přímo, aniž by procházel Controller (řadičem). @page musí být první direktivou Razor na stránce. @page ovlivňuje chování ostatních konstruktů Razor. Názvy souborů Razor Pages mají příponu .cshtml. [7]

Soubor třídy PageModel má obvykle stejný název jako soubor Razor Page s připojeným .cs. Například stránka Index.cshtml, soubor obsahují PageModel Index.cshtml.cs. Slouží pro propojení stránek s daty a aplikační logikou aplikace. [7]

Framework Razor Pages je založený na ASP.NET Core MVC, jehož infrastrukturu silně využívá. Z tohoto důvodu lze Razor Pages chápat jako vrstvu abstrakce nad ASP.NET Core MVC, umožňující jednodušším způsobem a s méně obsáhlým zdrojovým kódem dosáhnout stejného výsledku. [7]

ASP.NET Core SignalR

SignalR je knihovna, která nabízí vývoj webových aplikací v reálném čase pro aplikace ASP.NET Core.

SignalR je primárně používán aplikacemi, které vyžadují push oznámení ze serveru na klienta; například aplikace jako chat, akciový trh, hry a dashboardy. Před aplikací SignalR vývojáři aplikací používali k implementaci webových funkcí v reálném čase pomocí metod sdružování, jako je dlouhé / krátké dotazování, ve kterém klient bude dotazovat server na nové informace na základě časového intervalu. Tento přístup je výkonně intenzivní, síťově vytěžující a vyžaduje silnější hardware. SignalR řeší problém poskytováním trvalého spojení mezi klientem a serverem. Používá rozhraní Hubs API k zasílání oznámení ze serveru na klienta a podporuje více kanálů, jako je WebSocket, Server-sent events a Long Polling. [8]

1.3.3 Entity Framework

Entity Framework (EF) Core je snadná, rozšiřitelná, open source a multiplatformová verze populární technologie pro přístup k datům Entity Framework. Byla součástí .NET Framework, ale od verze 6 Entity Framework je oddělena od .NET Framework. [9]

EF Core může sloužit jako objektově relační mapovač (ORM - Object Relation Mapper), který vývojářům .NET umožňuje pracovat s databází pomocí objektů .NET a vylučuje potřebu většiny kódu pro přístup k datům, který obvykle potřebují psát. [9]

EF Core podporuje mnoho databázových modulů.

U EF Core je přístup k datům prováděn pomocí modelu. Model je tvořen třídami entit a kontextovým objektem, který představuje relaci s databází, což umožňuje dotazovat a ukládat data.

EF umožňuje vygenerování modelu z existující databáze, ručně kódovaný model tak, aby odpovídal databázi, nebo pomocí EF migrace vytvořit databázi modelu a poté ji vyvíjet, jak se model v průběhu času mění. [9]

Instance tříd entit jsou získávány z databáze pomocí Language Integrated Query (LINQ).

Data se vytvářejí, odstraňují a upravují v databázi pomocí instancí tříd entit.

Entity Framework nabízí čtyři různé pracovní postupy [9]:

- Model First umožňuje vytvořit nový model pomocí nástroje Entity Framework Designer a poté vygenerovat databázové schéma z modelu. Model je uložen v souboru EDMX (přípona .edmx) a lze jej prohlížet a upravovat v nástroji Entity Framework Designer. Třídy, které pracují s aplikací, jsou automaticky generovány ze souboru EDMX.
- Database First umožňuje zpětně analyzovat model z existující databáze. Model je uložen v souboru EDMX (přípona .edmx) a lze jej prohlížet a upravovat v nástroji Entity Framework Designer. Třídy, které pracují s aplikací, jsou automaticky generovány ze souboru EDMX.
- Code First: Tento scénář zahrnuje cílení na databázi, která neexistuje, a vytvoří se Code Fisrt, nebo prázdná databáze, do které Code First přidá nové tabulky. Code First umožňuje definovat model pomocí tříd C # nebo VB.Net. Další konfiguraci lze volitelně provést pomocí atributů na třídách a vlastnostech nebo pomocí Fluent API.
- Code First From Database: Code First umožňuje definovat model pomocí tříd C # nebo VB.Net. Volitelnou další konfiguraci lze provést pomocí atributů na třídách a vlastnostech nebo pomocí plynulého rozhraní API.

1.3.4 LINQ

Language-Integrated Query (LINQ) je název pro sadu technologií založených na schopnosti integraci dotazů přímo do jazyka C#.

LINQ zajišťuje jednotnou syntaxi pro přístup k datům. Výrazy dotazů jsou zapsány v syntaxi deklarativního dotazu. Pomocí syntaxe dotazu lze provádět operace filtrování, řazení a seskupování na zdrojích dat s minimem kódu. Používají se stejné základní vzorce výrazů dotazu k dotazování a transformaci dat v databázích SQL, datových sadách ADO.NET, dokumentech XML a jakoukoliv kolekci objektů. [10]

Zdroj dat je možné dotazovat dvěma způsoby. Syntaxí imitující SQL nebo stručnější formou pomocí lambda výrazů a rozšiřujících funkcí. SQL syntaxe je pro mnohé intuitivnější a uchopitelnější. Nicméně C# si kód na pozadí vždy přetvoří právě do podoby lambda výrazů a volá rozšiřující metody jako Where(), Select(), OrderBy() a další.

LINQ rozděluje do těchto kategorií či podtypů [10]:

- LINQ to Objects (C#) – přímé použití dotazů LINQ s jakoukoli kolekcí IEnumerable nebo IEnumerable <T>

- LINQ to XML – dotazování nad XML soubory.
- LINQ to ADO.NET
  - LINQ to DataSet – umožňuje vytvářet dotazy nad DataSet
  - LINQ to SQL – převede dotazy v objektovém modelu do SQL a odešle je do databáze k provedení, když databáze vrátí výsledky, LINQ na SQL je převede zpět do objektů, se kterými lze manipulovat
  - LINQ to Entities – umožňuje psát dotazy na koncepční model Entity Framework

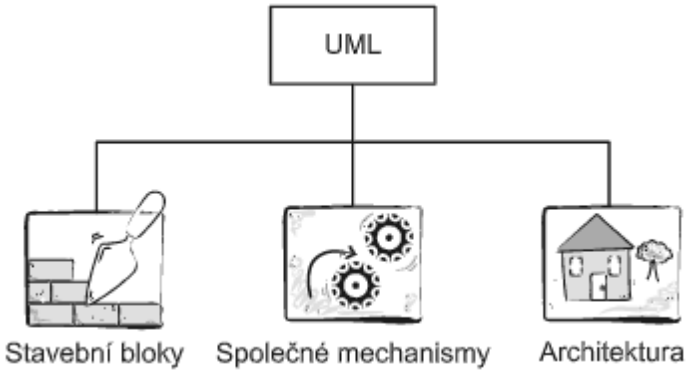
1.3.5 UML

Jazyk UML (Unified Modeling Language, unifikovaný modelovací jazyk) je univerzální jazyk pro vizuální modelování systémů, který vznikl v roce 1994. Přestože je nejčastěji spojován s modelováním objektově orientovaných softwarových systémů, má mnohem širší využití, což vyplývá z jeho zabudovaných rozšiřovacích mechanismů. [12]

Jazyk UML byl navržen proto, aby spojil nejlepší existující postupy modelovacích technik a softwarového inženýrství. Jako takový je explicitně navržen takovým způsobem, aby jej mohly implementovat všechny nástroje CASE (computer-aided software engineering). Diagramy vytvořené v jazyku UML jsou srozumitelné pro lidi, ale navíc je mohou snadno interpretovat i programy CASE. [11] [12]

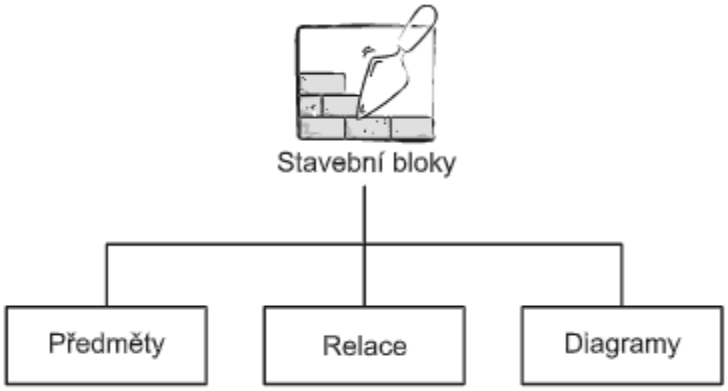
V roce 2005 byl vyvinut UML 2. V jazyce UML 2 bylo zavedeno mnoho nových prvků vizuální syntaxe. Některé z nich nahrazují a objasňují existující syntaxi verzí 1.x. Jiné jsou úplně nové a ztělesňují novou sémantiku přidanou k jazyku. Přestože je v jazyce UML 2 mnoho syntaktických odlišností vůči předchozím verzím, základní pravidla jsou velice podobná. [11]

Struktura jazyka UML se skládá ze tří částí: stavební bloky (základní prvky modelu, relace a diagramy), společné mechanismy (obecné způsoby k dosáhnutí specifických cílů), a architektura (pohled na architekturu navrhovaného systému).



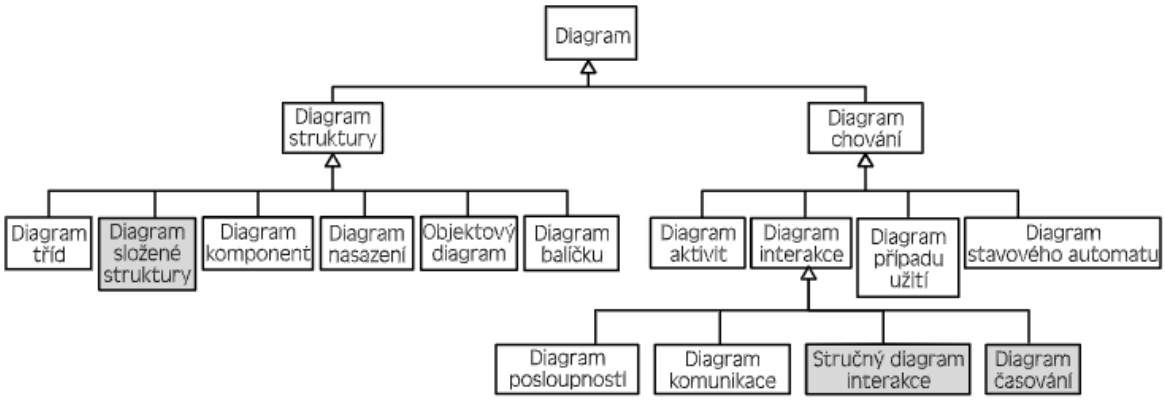
Obr. 2: Struktura jazyka UML [11]

Stavební bloky jazyka UML jsou tři základní: předměty (prvky modelu), vztahy (pojítka mezi předměty) a diagramy (pohledy na modely, ukazují, co systému bude dělat a jak to bude dělat).



Obr. 3: Stavení bloky jazyka UML [11]

Diagramy jsou okna nebo pohledy na model. Celkem existuje třináct různých typů diagramů UML. Abstraktní kategorie diagramu jsou napsány kurzívou, nově zavedené diagramy v UML 2 jsou se šedým pozadím.



Obr. 4: Diagramy UML [11]

1.3.6 HTML

HTML (HyperText Markup Language) je textový značkovací jazyk pro strukturování elektronické dokumenty jako textu s hypertextovými odkazy, obrázky a dalším obsahem. HTML dokumenty jsou základem World Wide Web a jsou zobrazovány webovými prohlížeči.

HTML je současně vyvíjeno organizacemi W3C a WHATWG. Současná verze je HTML 5.2 od prosince 2017, která je již podporována mnoha současnými webovými prohlížeči.

HTML se používá jako značkovací jazyk pro sémantickou strukturu textu. Vizuální reprezentace není součástí specifikací HTML a je určena webovým prohlížečem a šablonami návrhu jako je CSS. HTML může vkládat programy psané ve skriptovacím jazyce, jako je JavaScript, který ovlivňuje chování a obsah webových stránek. [13]

1.3.7 CSS a Bootstrap

CSS (Cascading Style Sheets) kaskádové styly popisují, jak mají být prvky HTML zobrazovány na obrazovce, papíru nebo na jiných médiích. Jazyk byl navržen organizací W3C, tak aby umožnil oddělení prezentace a obsahu, včetně rozvržení, barev a písmen. Toto oddělení může zlepšit přístupnost obsahu, poskytnout větší flexibilitu a kontrolu ve specifikaci prezentačních charakteristik, umožnit více webovým stránkám sdílet formátování zadáním příslušného CSS v samostatném souboru .css a snížit složitost a opakování strukturálního obsahu. [13]

Umožňuje přizpůsobit prezentaci různým typům zařízení, jako jsou velké obrazovky, malé obrazovky nebo tiskárny. CSS je nezávislý na HTML a lze jej použít s jakýmkoli značkovacím jazykem založeným na XML. Oddělení HTML od CSS usnadňuje údržbu webů, sdílení listů stylů napříč stránkami a přizpůsobování stránek různým prostředím. [13]

V současnosti je nejnovější verzi CSS 3, kde lze vytvářet i animace nebo 2D či 3D transformace.

Bootstrap je bezplatný framework CSS a JavaScript, který vývojářům umožňuje rychle vytvářet responzivní webové rozhraní. Rámec byl poprvé koncipován při práci na Twitteru v roce 2010. Od svého vydání jako open source projektu v srpnu 2011 se Bootstrap stal hlavním jménem vývojářů frontend webů. Díky jednoduchému použití a kompatibilitě s různými prohlížeči, podpoře mobilních uživatelských rozhraní a schopnosti reagovat na webový design je tento rámec základním stavebním kamenem každé moderní webové aplikace. [14]

Samotný rámec se skládá ze směsi JavaScriptu a CSS a poskytuje vývojářům všechny základní komponenty potřebné k vývoji plně funkčního webového uživatelského rozhraní.

Bootstrap už od jedné z prvotních verzí implementovat tzv. mřížkový (grid) systém. Mřížkový systém používá k rozložení a zarovnání obsahu řadu kontejnerů, řádků a sloupců. Je vytvořen pomocí flexbox (režim rozvržení v CSS3) a je plně responzivní. Další přednosti Bootstrap jsou typografie, formuláře, tlačítka, navigace a mnoho dalších. [14]

V současnosti je Bootstrap 4 nejnovější verzí Bootstrapu, s novými komponenty, rychlejšími šablonami stylů a větší responzivitou.

1.3.8 JavaScript

JavaScript (JS) je multiplatformní, objektově orientovaný skriptovací jazyk často zkompilovaný Just-in-time. Spolu s HTML a CSS je JavaScript jednou z hlavních technologií WWW. JS umožňuje interaktivní webové stránky a je nezbytnou součástí webových aplikací. Převážná většina webů ji používá pro chování stránek na straně klienta (interpretaci provádí webový prohlížeč návštěvníka stránky). JavaScript je obvykle vložen přímo do HTML stránek. [13]

1.3.9 Visual Studio

Visual Studio je integrované vývojové prostředí pro různé jazyky na vyšší úrovně nabízené společností Microsoft.

Aktuální verze pro Windows (Visual Studio 2019) podporuje 36 různých programovacích jazyků a umožňuje editor kódu a debugger podporu. Vestavěné jazyky zahrnují například Visual Basic .NET, C, C++, C#, F#, SQL Server, TypeScript a také HTML, JavaScript a CSS pro vývoj webových aplikací. Podpora dalších jazyků je k dispozici pomocí zásuvných modelů. Visual Studio používá platformy pro vývoj softwaru společnosti Microsoft, například Windows API, Windows Forms, Windows Presentation Foundation, Windows Store a Microsft Silverlight. Editor kódu podporuje IntelliSence (komponenta pro dokončení kódu) a refaktrorování kódu.

Visual Studio Community je zdarma, plně funkční IDE pro studenty, open-source a individuální vývojáře.

Do Visual Studio je možné stáhnout balíček pro vývoj webových aplikací frameworku ASP.NET Core (ASP.NET and web development). Poté stačí vytvořit projekt ASP.NET Core Web Application, který obsahuje všechny potřebné závislosti a strukturu základní webové stránky. Pomocí NuGet balíčků lze rozšířit aplikaci o Entity Frameworku Core a další rozšíření pro ulehčení vývoje webové aplikace.

1.4 Zabezpečení ASP.NET Core

ASP.NET Core umožňuje vývojářům snadno konfigurovat a spravovat zabezpečení svých aplikací. ASP.NET Core obsahuje funkce pro správu ověřování, autorizace, ochrany dat, vynucení https, tajných kódů aplikací, ochrany proti padělání a správy CORS. Tyto funkce zabezpečení umožňují vytvářet robustní a přesto bezpečné ASP.NET Core aplikace.

ASP.NET Core poskytuje mnoho nástrojů a knihoven pro zabezpečení vašich aplikací, včetně integrovaných poskytovatelů identit, ale lze použít služby identity třetích stran, jako je Facebook, Twitter nebo LinkedIn. Pomocí ASP.NET Core jde snadno spravovat tajné kódy aplikací, což je způsob ukládání a používání důvěrných informací, aniž by se musely vystavit v kódu.

ASP.NET Core a EF obsahují funkce, které pomáhají zabezpečit aplikace a zabránit narušení zabezpečení.

1.4.1 Autentizace

Autentizace (Authentication) neboli ověřování je proces, ve kterém uživatel poskytuje pověření, která jsou pak porovnány s těmi, které jsou uloženy v operačním systému, databázi, aplikaci nebo prostředku. Pokud se shodují, uživatelé se úspěšně ověří a pak mohou během procesu autorizace provádět akce, pro které jsou oprávněni. [15]

V ASP.NET Core je ověřování zpracováváno IAuthenticationService. Ověřovací služba používá registrované obslužné rutiny ověřování k dokončení akcí souvisejících s ověřováním. Příklady akcí souvisejících s ověřováním:

- Ověřování uživatele.
- Odpovídá, když se neověřený uživatel pokusí o přístup k omezenému prostředku.

1.4.2 Autorizace

Autorizace se týká procesu, který určuje, co může uživatel dělat. Uživatel s právy pro správu může například vytvořit knihovnu dokumentů, přidat dokumenty, upravit dokumenty a odstranit je. Uživatel bez oprávnění správce pracující s knihovnou má oprávnění ke čtení dokumentů. [15]

Autorizace je kolmá a nezávislá na ověřování. Autorizace ale vyžaduje mechanismus ověřování.

ASP.NET Core Authorization poskytuje jednoduchou, deklarativní roli a bohatý model založený na zásadách. Autorizace je vyjádřena v požadavcích a obslužné rutiny vyhodnocují deklarace identity uživatele proti požadavkům. Imperativní kontroly můžou být založené na jednoduchých zásadách nebo zásadách, které vyhodnocují identitu uživatele i vlastnosti prostředku, ke kterému se uživatel pokouší získat přístup. [15]

Autorizační komponenty, včetně atributů AuthorizeAttribute a AllowAnonymousAttribute, se nacházejí v oboru názvů Microsoft.AspNetCore.Authorization. [15]

1.4.3 Ochrana dat

Webové aplikace často potřebují ukládat data citlivá na zabezpečení. Systém Windows poskytuje rozhraní DPAPI pro aplikace klasické pracovní plochy, ale není vhodné pro webové aplikace. Sada ASP.NET Core Data Protection Stack nabízí jednoduché a snadno použitelné kryptografické rozhraní API, které může vývojář použít k ochraně dat, včetně správy klíčů a rotace. [15]

Sada ASP.NET Core Data Protection Stack je navržená tak, aby sloužila jako dlouhodobá náhrada prvku <machineKey> v ASP.NET 1.x-4.x. Byl navržen tak, aby využíval mnoho nedostatků starého kryptografického zásobníku a zároveň poskytovalo předem připravené řešení pro většinu případů použití moderních aplikací, které se mohou vyskytnout.

V nejjednodušším případě ochrana dat sestává z následujících kroků [15]:

1. Vytvořte ochranu dat od poskytovatele ochrany dat.
2. Zavolejte metodu Protect s daty, která chcete chránit.
3. Zavolejte metodu Unprotect s daty, která chcete převést zpět na prostý text.

1.4.4 HTTPS

Protokol Hypertext Transfer Protocol Secure (HTTPS) je rozšíření protokolu Hypertext Transfer Protocol (HTTP). Používá se pro bezpečnou komunikaci prostřednictvím počítačové sítě a je široce používán na internetu. V HTTPS je komunikační protokol šifrován pomocí Transport Layer Security (TLS) nebo dříve jeho předchůdce Secure Sockets Layer (SSL). [15]

Hlavní motivací pro HTTPS je autentizace přístupového webu , ochrana soukromí a integrity vyměněných dat během přenosu. Chrání před útoky typu „man-in-the-middle“. Obousměrné šifrování komunikace mezi klientem a serverem chrání před odposlechem a neoprávněným zásahem do komunikace. V praxi to poskytuje přiměřenou jistotu, že jeden komunikuje bez zásahu útočníků s webovou stránkou, se kterou chtěl komunikovat, na rozdíl od podvodníka.

Všeobecně se doporučuje podporovat HTTPS místo nezabezpečeného HTTP pro všechny webové stránky. [15]

V ASP.NET Core se používá middleware (UseHttpsRedirection) pro přesměrování požadavků HTTP na HTTPS.

1.4.5 Zabránění skriptování mezi weby (XSS)

Skriptování mezi weby – Cross-Site Scripting (XSS) je ohrožení zabezpečení, které umožňuje útočníkovi umístit skripty na straně klienta (obvykle JavaScript) do webových stránek. Když ostatní uživatelé načtou ovlivněné stránky, spustí se skripty útočníka, což útočníkovi umožní ukrást soubory cookie a tokeny relace, změnit obsah webové stránky prostřednictvím manipulace s modelem DOM nebo přesměrovat prohlížeč na jinou stránku. K ohrožení zabezpečení XSS obvykle dochází, když aplikace přebírá uživatelem vstup a vystupuje na stránku bez ověřování, kódování nebo uvozovacího prvku. [15]

Na základní úrovni technologie XSS funguje tak, že se podívá do vaší aplikace na vložení značky <script> do vykreslované stránky nebo vložením události On* do prvku. Vývojáři by měli pomocí následujících kroků prevence zabránit zavlečení SKRIPTOVÁNÍ do své aplikace. [15]

1. Nikdy neumisťovat nedůvěryhodná data do vstupu ve formátu HTML, pokud nejsou použity následující body.
   Nedůvěryhodná data jsou všechna data, která může být ovládána útočníkem, vstupy

formulářů HTML, řetězce dotazů, hlavičky protokolu HTTP, dokonce i data, která jsou zdrojem tohoto útočníka, může být, že by mohlo dojít k porušení zabezpečení vaší databáze, i když nemůžou porušit vaši aplikaci.

2. Před vložením nedůvěryhodných dat do elementu HTML zajistit, aby byl kódovaný HTML. Kódované HTML přebírá znaky jako < a mění je do bezpečné podoby jako je <
3. Před vložením nedůvěryhodných dat do atributu HTML zajistit, aby byl kódovaný HTML. Kódované atributů HTML je nadmnožinou kódování HTML a kóduje například znaky jako „ na '.
4. Vkládání nedůvěryhodných dat do JavaScriptu umístit data v elementu HTML, jehož obsah se načítá za běhu. Pokud to není možné, data by měla být kódovaná pomocí JavaScriptu. Kódování JavaScriptu přebírá nebezpečné znaky pro JavaScript a nahrazuje je šestnáctkově, například < by bylo kódováno jako \u003C.
5. Před vložením nedůvěryhodných dat do řetězce dotazu URL zajistit, aby byla zakódovaná adresa URL.

V ASP.NET Core modul Razor automaticky zakóduje veškerý výstup z proměnných. Používá pravidla kódování atributů HTML vždy, když se použije direktiva @.

[15]

1.4.6 Zabránění SQL injection

Injekce SQL je technika injekce kódu, která se používá k útoku na datově řízené aplikace, kde škodlivé příkazy SQL jsou vloženy do vstupního pole pro provedení (např. Výpis obsahu databáze útočníkovi). SQL injection musí využívat chybu zabezpečení v softwaru aplikace, například když je vstup uživatele buď nesprávně filtrován pro únikové znaky řetězce vložené do příkazů SQL nebo pokud není vstup uživatele silně zadán a neočekávaně proveden. Úspěšný SQL injection útok může upravovat databázi, zneužít citlivá data uložená v databázi, případně také získat přístup k administrátorskému účtu. [15]

Entity Framework Core umožňuje dotazovat se na nezpracované dotazy SQL při práci s relační databází. Nezpracované dotazy SQL jsou užitečné, pokud dotaz, který chcete nelze vyjádřit pomocí LINQ. Nezpracovaná dotazy SQL se také používají, pokud použití dotazu LINQ vede k neefektivnímu dotazu SQL. [15]

Při zavádění všech uživatelských zadaných hodnot do nezpracovaného dotazu SQL je třeba dbát na to, aby se zabránilo útokům injektáže SQL. Kromě ověření, že tyto hodnoty

neobsahují neplatné znaky, vždy využít parametrizaci, která odesílá hodnoty oddělené od textu SQL. [15]

Ostatní dotazy pomocí Entity Framework Core a LINQ jsou zabezpečeny proti SQL injection, protože předávají všechny data do databáze prostřednictvím parametrů. Dotazy LINQ nejsou složeny pomocí manipulace s řetězci nebo zřetězení, proto nejsou citlivě na tradiční útoky SQL injekcí. [15]

1.4.7 Zabránění Cross-Site Request Forgery (XSRF/CSRF)

Padělání požadavků napříč weby (označované také jako XSRF nebo CSRF) představuje útok proti aplikacím hostovaným na webu, kde může škodlivá webová aplikace ovlivnit interakci mezi klientským prohlížečem a webovou aplikací, která tento prohlížeč důvěřuje. Tyto útoky jsou možné, protože webové prohlížeče odesílají některé typy ověřovacích tokenů automaticky pomocí všech požadavků na web. Tato forma zneužití je také známá jako útok s jedním kliknutím nebo při jízdě relace, protože útok využívá dřív ověřenou relaci uživatele.

Příklad útoku CSRF [15]:

1. Uživatel se přihlásí k www.good-banking-site.com pomocí ověřování pomocí formulářů. Server ověří uživatele a vydá odpověď, která obsahuje soubor cookie ověřování. Lokalita je zranitelná vůči útokům, protože důvěřuje všem žádostem, které obdrží, pomocí platného ověřovacího souboru cookie.
2. Uživatel navštíví škodlivý web www.bad-crook-site.com. Škodlivý web, www.bad-crook-site.com, obsahuje formulář HTML podobný následujícímu:

```
<h1>Congratulations! You're a Winner!</h1>
<form action="http://good-banking-site.com/api/account" method="post">
  <input type="hidden" name="Transaction" value="withdraw">
  <input type="hidden" name="Amount" value="1000000">
  <input type="submit" value="Click to collect your prize!">
</form>
```

3. Uživatel vybere tlačítko Odeslat. Prohlížeč vytvoří požadavek a automaticky přidá ověřovací soubor cookie pro požadovanou doménu www.good-banking-site.com.
4. Požadavek běží na serveru www.good-banking-site.com s kontextem ověřování uživatele a může provádět všechny akce, které má ověřený uživatel povoleno provádět.

Kromě scénáře, kdy uživatel vybere tlačítko pro odeslání formuláře, může škodlivý web:

- Spustí skript, který automaticky odešle formulář.
- Odeslat odeslání formuláře jako požadavek AJAX.
- Skryjte formulář pomocí šablon stylů CSS.

Tyto alternativní scénáře nevyžadují žádnou akci ani vstup od uživatele kromě prvotního návštěvě škodlivého webu.

Použití protokolu HTTPS nebrání útoku CSRF. Škodlivý web může odeslat žádost o https://www.good-banking-site.com/ hned stejně snadno, jako by mohla odeslat nezabezpečený požadavek. [15]

ASP.NET Core implementuje ochranu proti padělání pomocí ASP.NET Core ochrany dat.

V případě, že je v Startup.ConfigureServicesvolána jedna z následujících rozhraní API, je middleware pro falšování vlastnictví přidána do kontejneru pro vkládání závislostí:

- AddMvc
- MapRazorPages
- MapControllerRoute
- MapBlazorHub

Razor Pages jsou automaticky chráněny proti XSRF/CSRF. FormTagHelper vloží tokeny antiforgery do prvků formuláře HTML. [15]

II. PRAKTICKÁ ČÁST

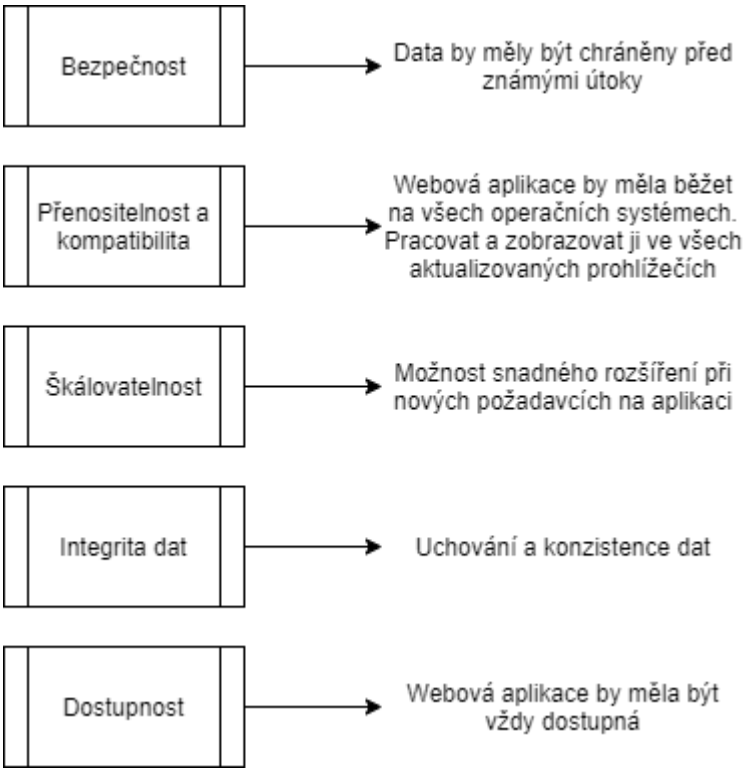
2 PLÁNOVÁNÍ A CÍLE PRÁCE

2.1 Nefunkční požadavky

V systémovém inženýrství a inženýrství požadavků je nefunkční požadavek (NFR) požadavkem, který specifikuje kritéria, která mohou být použita k posouzení fungování systému, spíše než specifická chování. Nefunkční požadavky definují, jak by to měl systém chovat a definuje, jaká omezení jsou kladena na chování systému, nespecifikují implementaci.

Typy nefunkčních požadavků: architektonické, kapacitní, účinnost, odolnost vůči chybám, soukromí, výkon, robustnost, dostupnost, integrita dat, rozšiřitelnost, ovladatelnost, udržitelnost, přenositelnost, spolehlivost, obnovitelnost, škálovatelnost, bezpečnost, použitelnost, stabilita, podpora, testovatelnost, použitelnost, dokumentace. [16]

Konkrétní nefunkční požadavky při vývoji webové aplikace pro služební cesty:



Obr. 5: Nefunkční požadavky

2.2 Funkční požadavky

V softwarového inženýrství a systémového inženýrství, je funkční požadavek definuje funkci systému nebo jeho součásti, pokud je funkce popsána jako popis chování mezi vstupy a výstupy. Funkční požadavky mohou zahrnovat výpočty, technické detaily, manipulaci a zpracování dat a další specifické funkce, které definují, čeho má systém dosáhnout. Požadavky na chování popisují všechny případy, kdy systém používá funkční požadavky, které jsou zachyceny v případech použití. Funkční požadavek popisuje, jaké funkce (tj. hardware a software) měl provádět.

Typy funkčních požadavků: obchodní pravidla, administrativní funkce, ověřování, úrovně autorizace, archivace, algoritmy, databáze, síť, infrastruktura, zálohovaní a obnova, opravy, úpravy a rušení transakcí, požadavky na vyhledávání a hlášení, strukturální. [16]

Konkrétní funkční požadavky při vývoji této webové aplikace pro služební cesty 36Obr. 6.

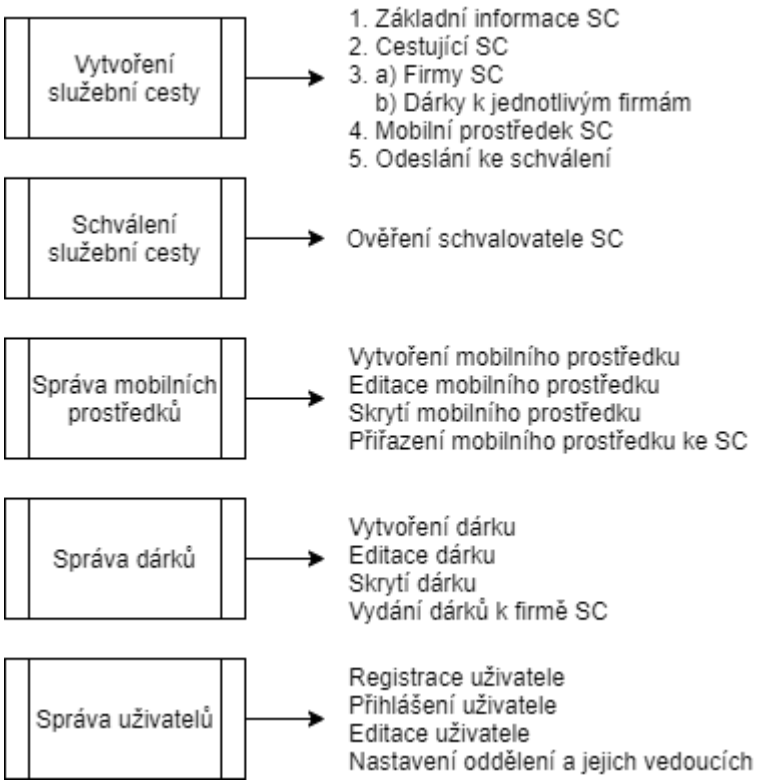
Vytvoření služební cesty probíhá v pěti krocích. V prvním kroku se zadají základní informace, jako název, místo, datum, popis. V druhém kroku se do služební cesty přidají cestující, kteří se služební cesty zúčastní. Ve třetím kroku se do služební cesty přidávají firmy, ke kterým jdou následně přidat dárky, které by se jim při návštěvě darovaly. Ve čtvrtém kroku výběr mobilního prostředku. A v pátek kroku závěrečný přehled všech vyplněných informací a odeslání SC na schválení.

Schvalovat služební cesty mohou pouze vedoucí oddělení a mohou schvalovat pouze SC, na kterých je cestující z jeho oddělení. Při ověření probíhá kontrola, zda přihlášený uživatel je vedoucím jednoho z cestujících, pokud ano, má možnost SC schválit nebo zamítnout.

Správa mobilních prostředků je omezená jen pro uživatele, kteří jsou k tomu oprávněni. Mohou vytvářen nové mobilní prostředky, editovat a skrývat existující. Mobilní prostředky jsou rozděleny na veřejné a privátní. Veřejné vidí všichni uživatelé, ale obsahují většinou pouze možnosti jako „Nepotřebuji auto“ a „Nechat si auto přidělit“. Privátní dále mohou mít ještě vlastníka, který jako jediný auto při výběru vidí. Proto dalším úkolem správců mobilních prostředků je přiřazovat mobilní prostředky ke SC, kde je zadáno „Nechat si auto přidělit“.

Správa dárků je taktéž omezena jen pro uživatele, kteří jsou k tomu oprávnění. Mohou vytvářet nové dárky, editovat a skrývat existující. Dále si vedou přehled, které dárky již byly předány. Správce dárků předává dárky uživateli, který SC vytvořil a o tom si vede záznam, aby měl přehled, které dárky ještě potřebuje vydat a které již byly vydány.

Každý uživatel má možnost se do aplikace registrovat, přihlásit a pak svůj uživatelský profil editovat. Navíc existuje Správce uživatelů, který přiřazuje uživatele do oddělení a nastavuje jim role, které je opravňují k dalším akcím.



Obr. 6: Funkční požadavky

2.3 Diagram případů užití

Modelování případů užití je jednou z forem inženýrství požadavků. Modelování případů užití se skládá z následujících aktivit:

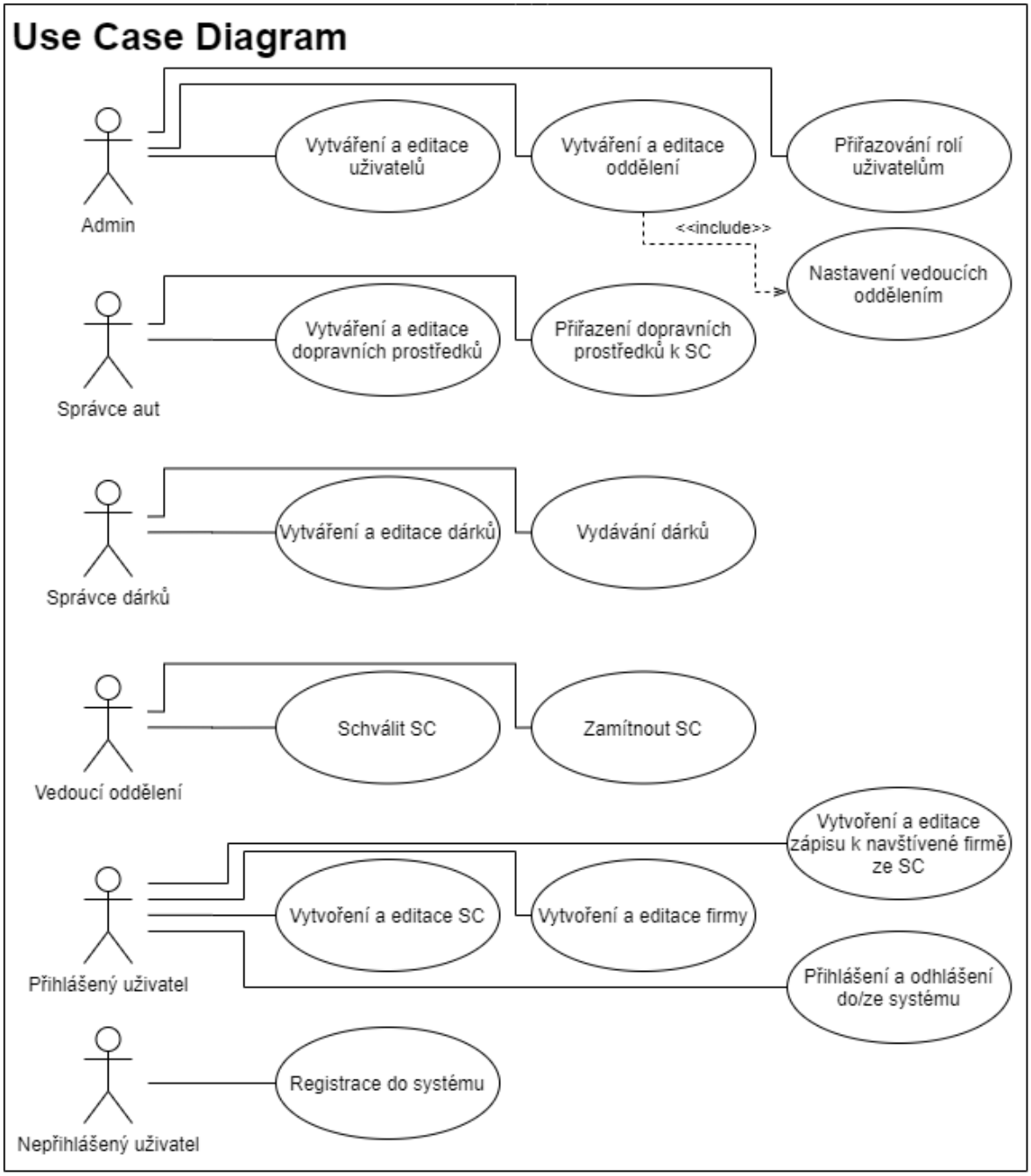
- Nalezení hranic systému
- Vyhledání účastníků
- Nalezení případů užití
- Specifikace případu užití
- Tvorba scénářů

Výstupem uvedených aktivit je model případu užití. Tento model obsahuje čtyři komponenty:

- Účastníci (actors). Jsou to role, přidělené osobám nebo předmětům používající daný systém.

- Případy užití (use cases). Činnosti, které mohou účastníci se systémem vykonávat.
- Relace (relationships). Smysluplné vztahy mezi účastníky a případy užití.
- Hranice systému (systém boundary). Ohraničení zobrazené kolem případů užití, jež je vyznačením území nebo hranic modelovaného systému.

Use Case Diagram (diagram případů užití) zobrazuje chování systému tak, jak ho vidí uživatel. [11]



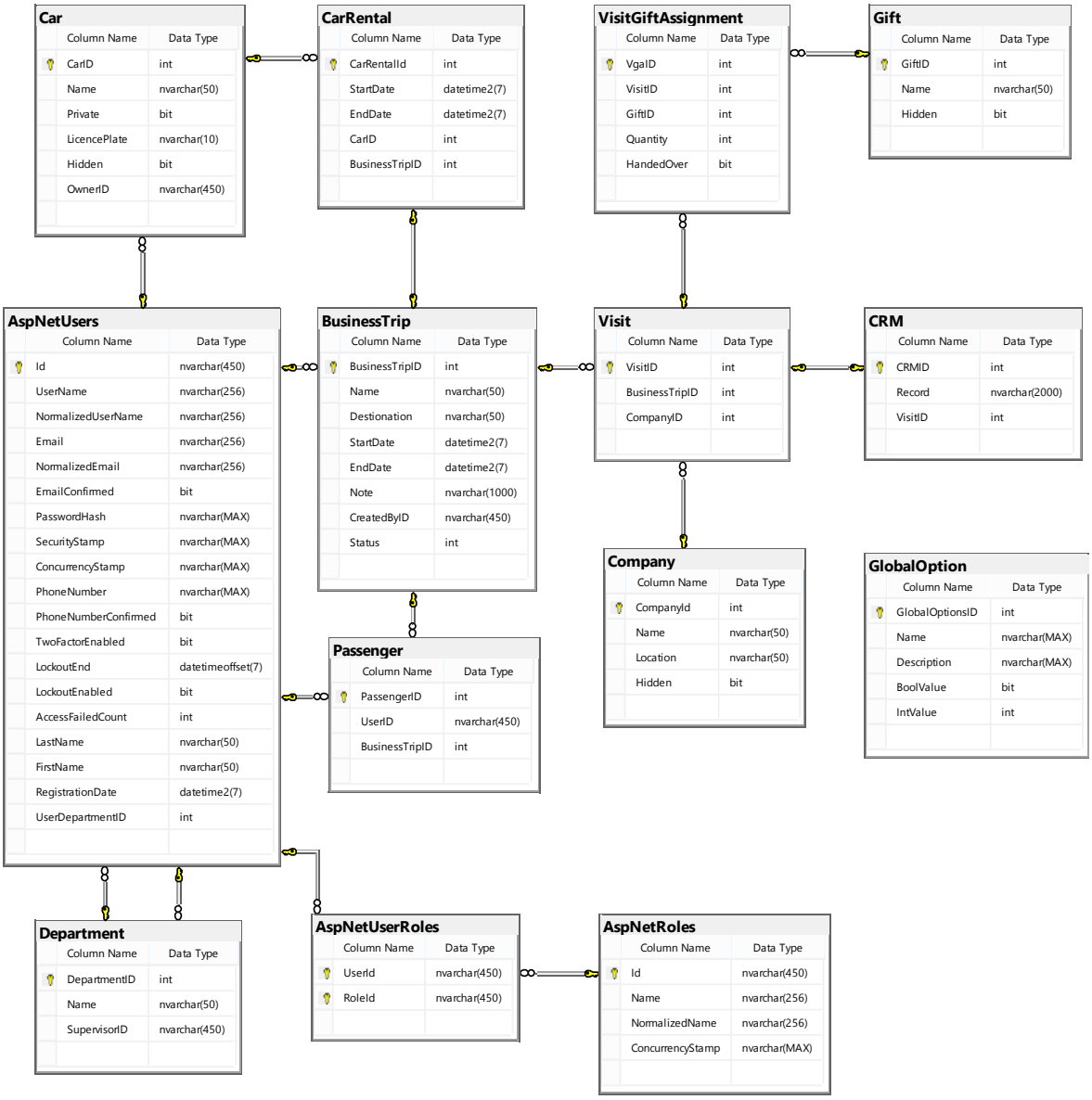
Obr. 7: Diagram případů užití

2.4 E-R diagram

Diagram vztahů entit (ERD) ukazuje vztahy sad entit uložených v databázi. Entita v tomto kontextu je objekt, součást dat. Sada entit je kolekce podobných entit. Tyto entity mohou mít atributy, které definují jeho vlastnosti.

Definováním entit, jejich atributů a zobrazením vztahů mezi nimi ER diagram ilustruje logickou strukturu databází.

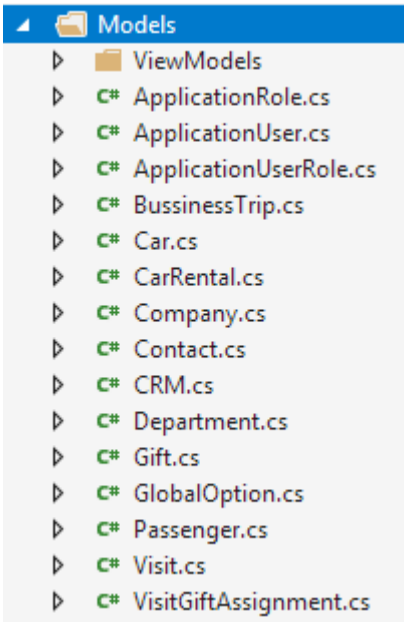
Obr. 8: Diagram vztahů entit



Obr. 9: Databázové schéma

Současný náhled na databázové schéma Obr. 9: Databázové schémaObr. 9 byl vytvořen pomocí SQL Server Management Studio. A vytvořen pomocí Entity Framework Core podle tříd modelů a jejich následné migraci do databáze.

Tento návrh je typu Code First. Code First pracovní postup začíná třídami, které popisují koncepční model, a poté Entity Framework automaticky generuje databázi z tohoto modelu.



Obr. 11: Třídy modelu

```
namespace BussinessTripsApp.Models
{
    28 references
    public class BussinessTrip : IValidatableObject
    {
        [Key]
        40 references
        public int BussinessTripID { get; set; }

        [Required]
        [StringLength(50, MinimumLength = 3)]
        50 references
        public string Name { get; set; }

        [Required]
        [StringLength(50, MinimumLength = 2)]
        29 references
        public string Destionation { get; set; }

        [Required(ErrorMessage = "The start date is required")]
        [Display(Name = "Start Date")]
        [DisplayFormat(ApplyFormatInEditMode = true)]
        39 references
        public DateTime StartDate { get; set; }

        [Required(ErrorMessage = "The end date is required")]
        [Display(Name = "End Date")]
        [DisplayFormat(ApplyFormatInEditMode = true)]
        45 references
        public DateTime EndDate { get; set; }

        [StringLength(1000)]
        24 references
        public string Note { get; set; }

        8 references
        public string CreatedByID { get; set; }

        [Display(Name = "Created By")]
        [ForeignKey("CreatedByID")]
        47 references
        public ApplicationUser CreatedBy { get; set; }
```

Obr. 10: Náhled třídy modelu BusinessTrip

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

namespace BussinessTripsApp.Migrations
{
    1 reference
    public partial class Start : Migration
    {
        7 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "BussinessTrip",
                columns: table => new
                {
                    BussinessTripID = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Name = table.Column<string>(maxLength: 50, nullable: false),
                    Destionation = table.Column<string>(maxLength: 50, nullable: false),
                    StartDate = table.Column<DateTime>(nullable: false),
                    EndDate = table.Column<DateTime>(nullable: false),
                    Note = table.Column<string>(maxLength: 1000, nullable: true),
                    CreatedByID = table.Column<string>(nullable: true),
                    Status = table.Column<int>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_BussinessTrip", x => x.BussinessTripID);
                });

            migrationBuilder.CreateTable(
                name: "Visit",
                columns: table => new
                {
                    VisitID = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    BussinessTripID = table.Column<int>(nullable: false),
                    CompanyID = table.Column<int>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Visit", x => x.VisitID);
                    table.ForeignKey(
                        name: "FK_Visit_BussinessTrip_BussinessTripID",
                        column: x => x.BussinessTripID,
                        principalTable: "BussinessTrip",
```

Obr. 12: Entity Framework migrace

Tabulky (entity) začínající AspNet jsou navrhnuty a vytvořeny pomocí ASP.NET Core Identity, což je rozhraní API, které podporuje funkce přihlášení uživatelského rozhraní a správu uživatele, hesla, data profilu, role, potvrzení e-mailu a další. Tyto entity se dají následně rozšířit o vlastní požadované atributy.

V návrhu se vyskytují tyto entity, které uchovávají data systému:

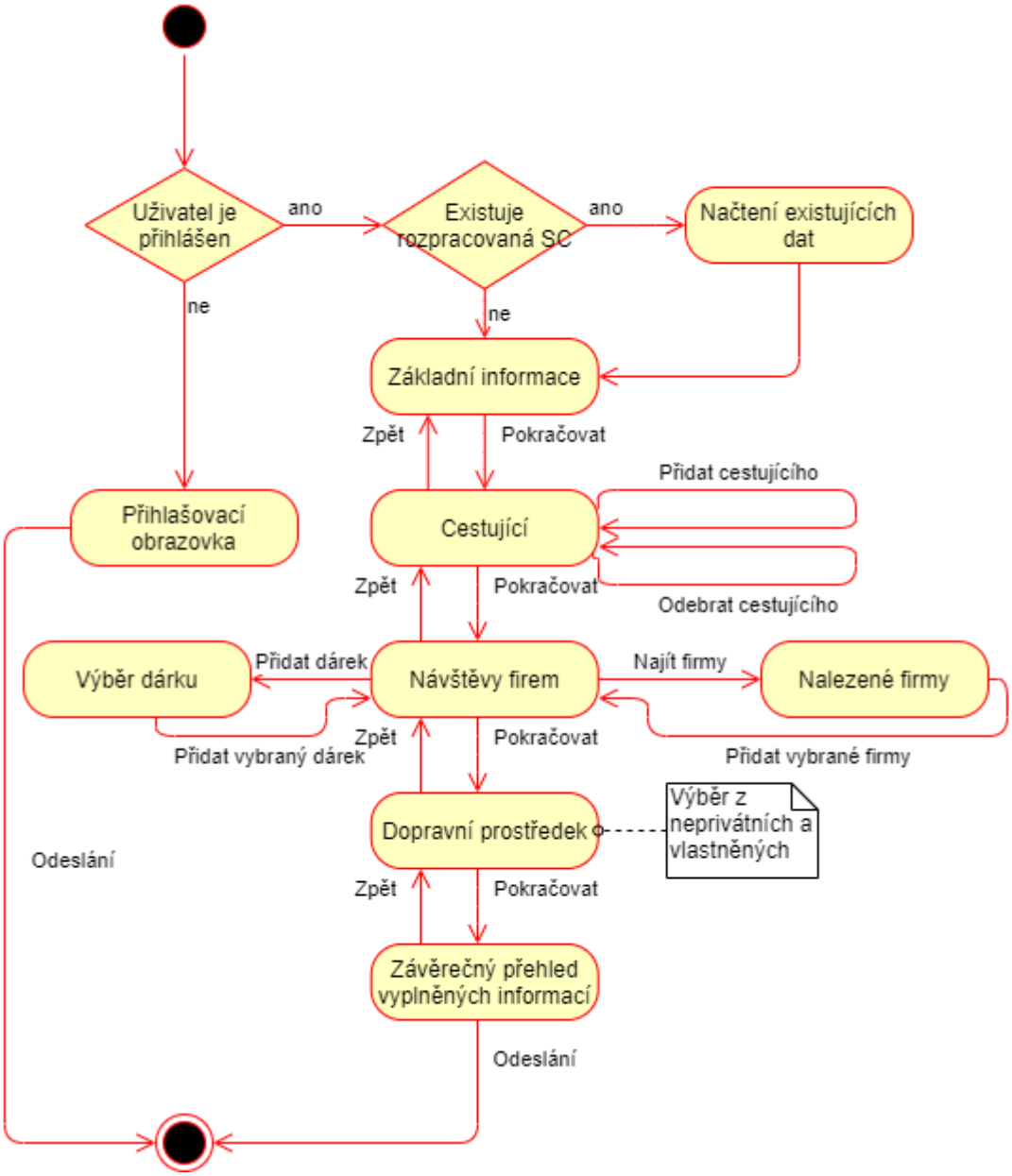
- **AspNetUsers** – uživatelé registrovaní do systému.
- **AspNetRoles** – uživatelské role, které opravňují uživatele k určité akci.
- **AspNetUserRoles** – slouží k propojení uživatele a jeho rolí.
- **Department** – oddělení, které má svého vedoucího.
- **BussinessTrip** – tabulka modelující služební cesty.
- **Passenger** – (cestující) uživatelé, kteří se účastní konkrétní služební cesty. Propojení mezi uživatelem a služební cestou.
- **Visit** – (návštěva) firmy, které jsou navštíveny během konkrétní služební cesty. Propojení mezi firmou a služební cestou.
- **Company** – tabulka modelující firmy (seznam firem, které mohou být na služební cestě navštíveny).
- **CRM** – zápis z konkrétní návštěvy firmy ze služební cesty.
- **VisitGiftAssignment** – (přiřazení dárku návštěvě) dárky, které se darují během návštěvy firmy během služební cesty.
- **Gift** – dárky (seznam dárků, které mohou být darovány při návštěvě firmy během služební cesty).
- **CarRental** – (vypůjčení dopravního prostředku) dopravní prostředek, který je vypůjčen pro konkrétní služební cestu. Propojení mezi mobilními prostředky a služebními cestami.
- **Car** – tabulka modelující mobilní prostředky
- **GlobalOption** – tabulka modelující globální nastavení systému

2.5 Vytvoření služební cesty

Služební cesty smí vytvořit jakýkoliv přihlášený uživatel.

Vytvoření služební cesty se skládá z několika kroků. Uživatel nejprve zadá základní informace o služební cestě jako je název, datum od do, místo a poznámku. Dál se dostane na stránku s cestujícími. Zadavatel služební cesty je vždy automaticky jako první cestující vybrán a má možnost přidávat další cestující, ze seznamu uživatelů. Další krok je přidávání firem, které se během služební cesty navštíví. Uživatel má pole pro vyhledávání firem podle názvu. Po vyhledání pomocí checkboxu vybere firmy, které chce přidat a přidá je. Následně ke každé firmě může přidat dárky. Posledním krokem je vybrat si dopravní prostředek. Uživatel smí vybírat pouze z veřejných (neprivátních) dopravních prostředků nebo vlastněných. Zároveň zadá i datum od do, kdy chce prostředek převzít a vrátit. Nakonec se mu zobrazí stránka s přehledem všech vyplněných informací. Pokud by chtěl ještě nějaké informace upravit, může jít zpět a cokoliv upravit a znova se vrátit na přehled všech informací. Jakmile má vše správně vyplněno, tak odešle služební cestu se všemi informacemi.

Celý proces vytvoření služební cesty je znázorněný na následujícím Obr. 13: Vytvoření služební cesty.



Obr. 13: Vytvoření služební cesty

2.6 Schválení služební cesty

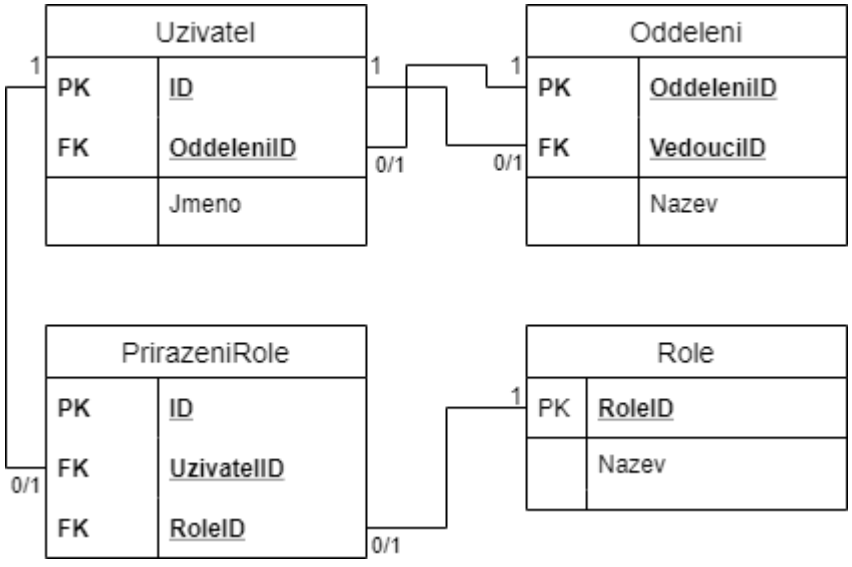
Důležitou součástí služebních cest je jejich schválení. Schvalovat smí pouze vedoucí účastníků cestky. Proto je nutné vytvořit strukturu uživatelů a jejich vedoucích. Toto je dosáhnuto pomocí tabulky oddělení. Admin má právo vytvářet a upravovat oddělené. Každé oddělení má svého vedoucího. Jakmile admin vytvoří strukturu firmy pomocí oddělení a jejich vedoucích, přiřazuje jednotlivé uživatele do daných oddělení. Tím se vytvoří struktura firma. Například uživatel X, Y a Z jsou v oddělení A1, které má vedoucího V1. Jakmile je jeden z těchto uživatelů účastníkem služební cesty, tak vedoucí V1 má právo tuto služební cestu schválit.

Dále existuje role Autoschválení, kterou může nastavit také admin u uživatele. Pak nezáleží na tom, v jakém oddělení uživatel je a jeho služební cesty se automaticky schválí. Platí pouze pro toho, kdo služební cestu vytváří. Pokud si přidá uživatel, který nemá roli Autoschválení, spolucestujícího do své služební cesty, který má roli Autoschválení, tak služební cesta bude muset být schválena jedním z vedoucích.

Vztahy mezi uživatelem, oddělením a rolemi je na následujícím obrázku.

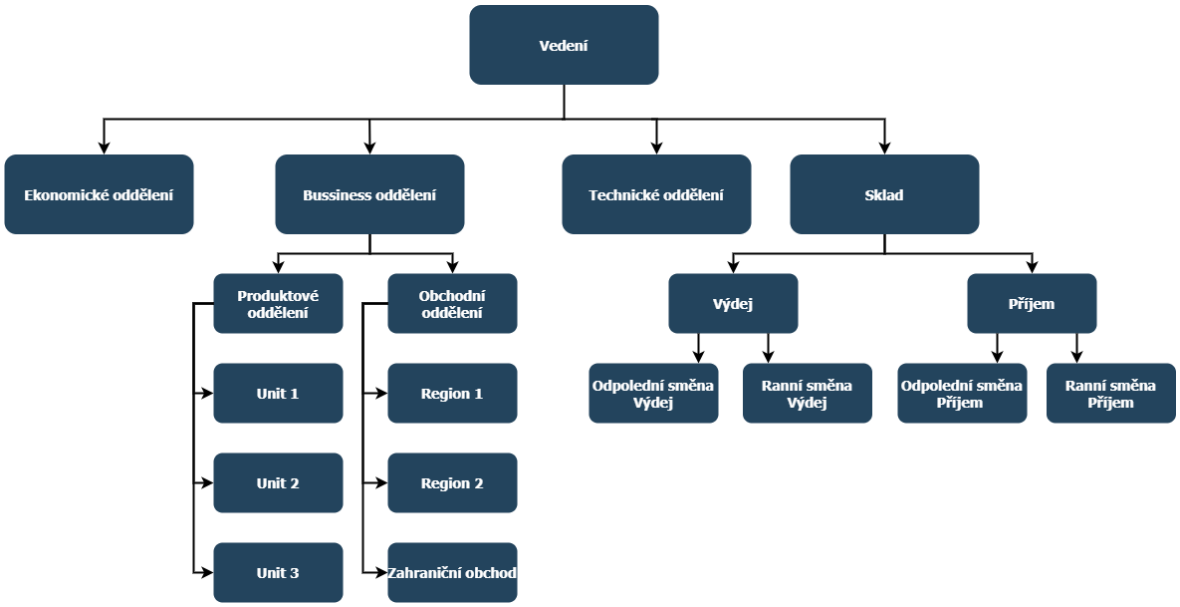
Příklad rozvržení oddělení je na Obr. 14: Vztahy mezi uživateli, odděleními a rolemi.

Popis schválení je na Obr. 16: Schválení služební cesty.



Obr. 14: Vztahy mezi uživateli, odděleními a rolemi

Obrázek ukazuje na vztahy mezi uživatelem, oddělením a rolemi. Každý uživatel může být přiřazen do jednoho oddělení. Každé oddělení může mít jednoho vedoucího uživatele. Uživatel může mít několik rolí.



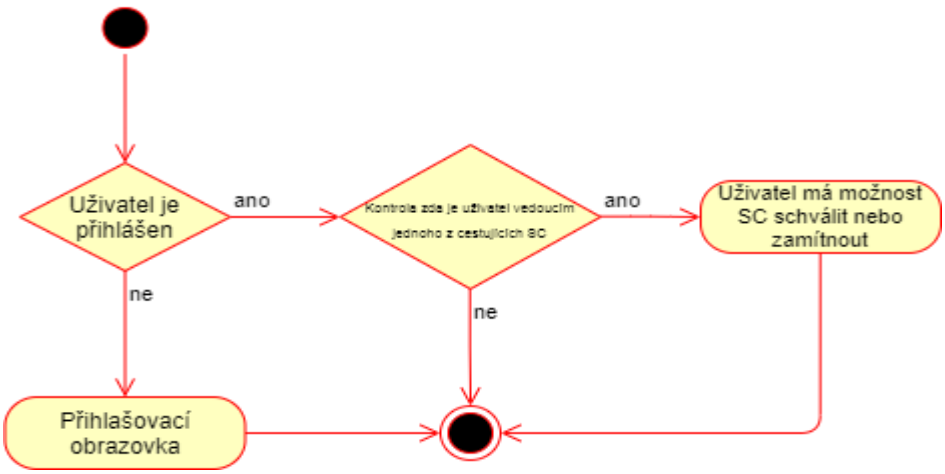
Obr. 15: Příklad struktury oddělení

Rozvržení oddělení může vypadat jako stromová struktura. V tomto příkladu rozvržení je 19 oddělení. Každé oddělení by mělo mít nastavené svého vedoucího, které pro uživatele v daném oddělení bude schvalovat služební cesty. Vedoucí oddělení je zařazen do vyššího oddělení, které má svého vedoucího.

Například uživatelé v oddělení Unit 1 schvaluje vedoucí Unitu 1. Ale vedoucí Unitu 1, 2 a 3 schvaluje vedoucí Bussiness oddělení.

Uživatelé v oddělení Vedení mohou mít nastavenou roli Autoschválení nebo by je mohl například schvalovat jeden uživatel, který by byl vedoucí celé firmy.

Rozvržení může vypadat i jako kruhové, záleží jen na adminovi, jak strukturu navrhne a přiřadí do ní uživatele a nastaví vedoucí.



Obr. 16: Schválení služební cesty

3 REALIZACE APLIKACE

Aplikace byla realizována s ohledy na plány, požadavky a návrh systému aplikace.

3.1 Náhled na aplikaci

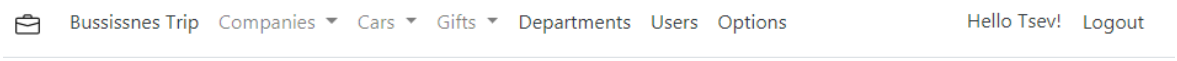
3.1.1 Vzhled aplikace

Jedná se o HTML stránku, kterou lze rozdělit na tři části.

1. Hlavička, menu

   Odkazy v menu se mění v závislosti na přihlášeném uživateli a jeho rolích.

   Konkrétní náhled na obrázku je s rolí admin, který má právo na vše.

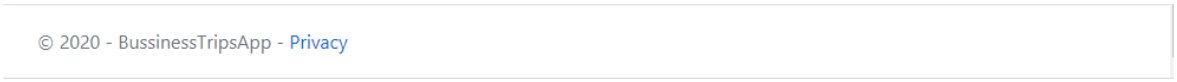


Obr. 17: Hlavička aplikace

2. Tělo stránky

   Tělo stránky se mění podle zvoleného odkazu. Primární stránka je přehled služebních cest Business Trip.

3. Zápatí, patička stránka



Obr. 18: Zápatí aplikace

3.1.2 Přihlášení a registrace do aplikace

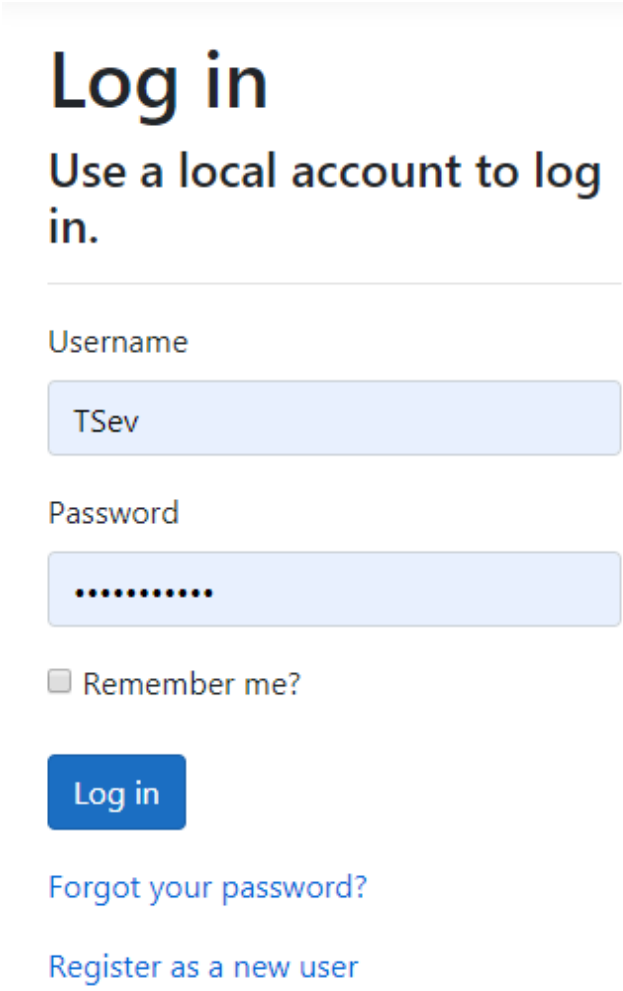
Přihlášení je nutné, aby mohl uživatel vidět jakékoliv data nebo s nimi jakkoliv manipulovat - tvořit, editovat. O registraci a přihlášení se stará ASP.NET Core Identity.

Přihlašuje se pomocí přihlašovacího jména (Username), které se může lišit od jména nebo emailu a pomocí silného hesla. Silné heslo je vyžadováno při registraci.

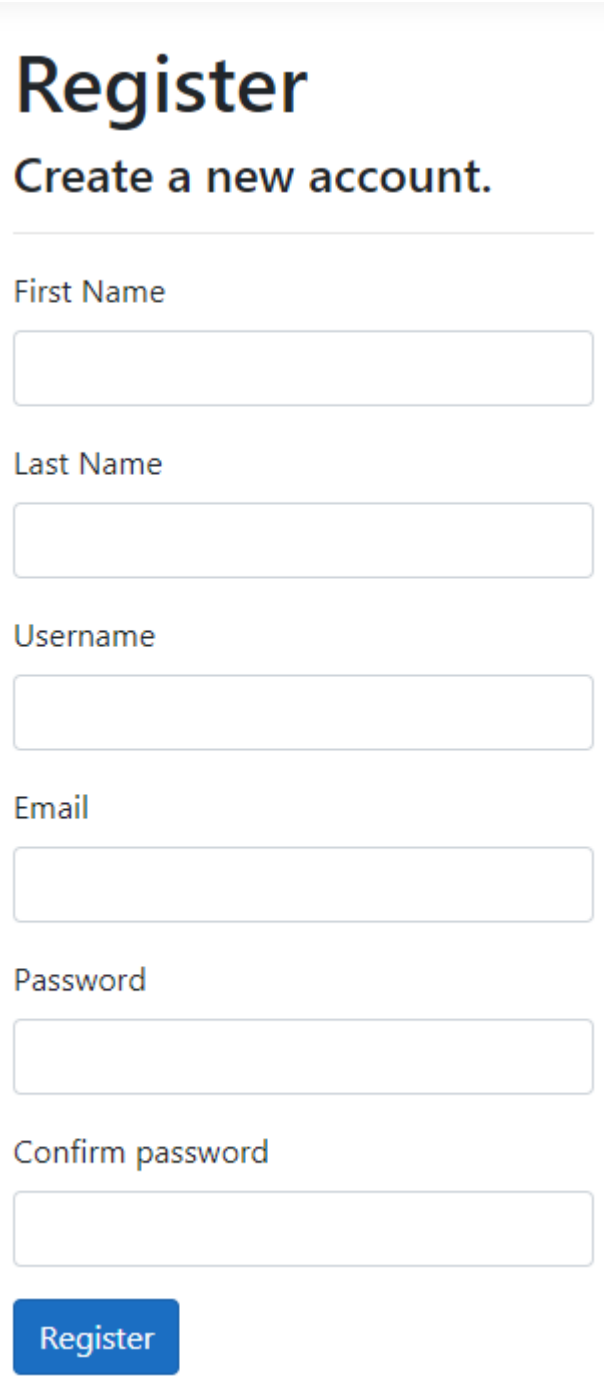
Registrace do aplikace externími (novými) uživateli může být vypnuta adminem pomocí globálního nastavení. Pak může nové uživatele tvořit pouze admin. Admin může také deaktivovat stávající uživatele a ti se pak do aplikace už nemohou přihlásit.

Po přihlášení je uživatel přesměrován na hlavní stránku s výpisem služebních cest.

Přihlášení a registraci znázorňují další dva obrázky.



Obr. 19: Přihlášení do aplikace



Obr. 20: Registrace do aplikace

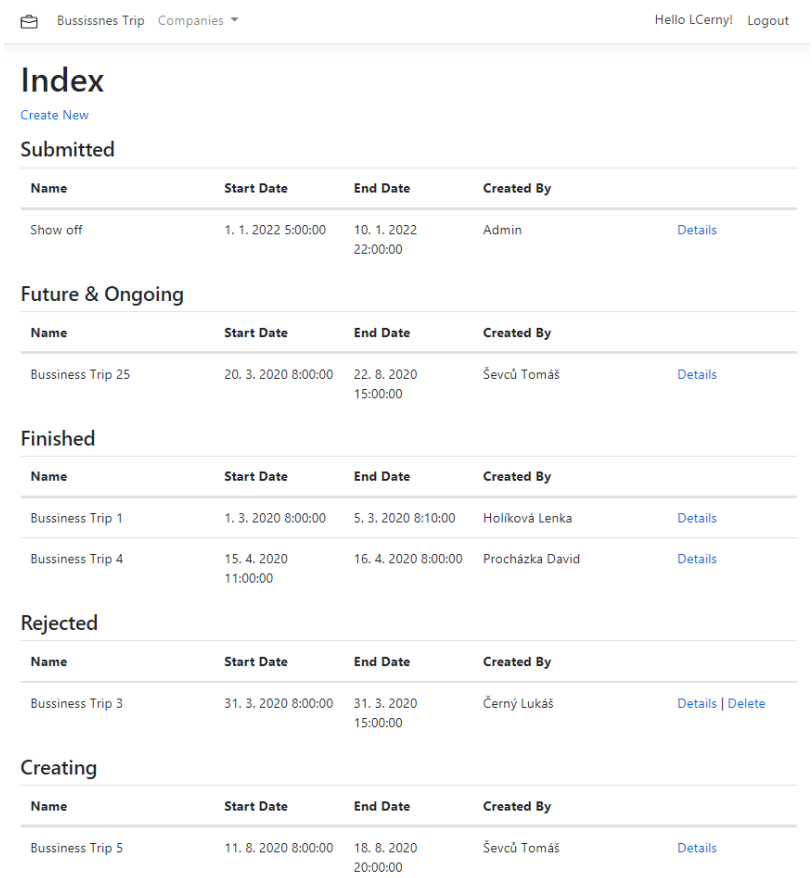
3.2 Služební cesty

3.2.1 Přehled

Služební cesty – hlavní stránka aplikace vypisuje seznam všech služebních cest. Všichni uživatelé se mohou podívat na detail kterékoliv služební cesty a vytvářet nové služební cesty.

Stránka je rozdělená do pěti kategorií podle toho, v jakém stádiu se služební cesta nachází.

1. Submitted – služební cesty odeslané a čekající na schválení
2. Futere & Ongoing – schválené služební cesty, které probíhají nebo teprve budou probíhat
3. Finished – proběhlé schválené služební cesty
4. Rejected – zamítnuté služební cesty
5. Creating – služební cesty, které se právě vytváří (uživatel mohl například vytvářet novou služební cestu, ale nedostal se až k odeslání, okno se stránkou zavřel a jeho služební cesta je ve stavu vytváření a při dalším vrácení se na stránku v její tvorbě bude moci pokračovat)



Obr. 21: Přehled služebních cest

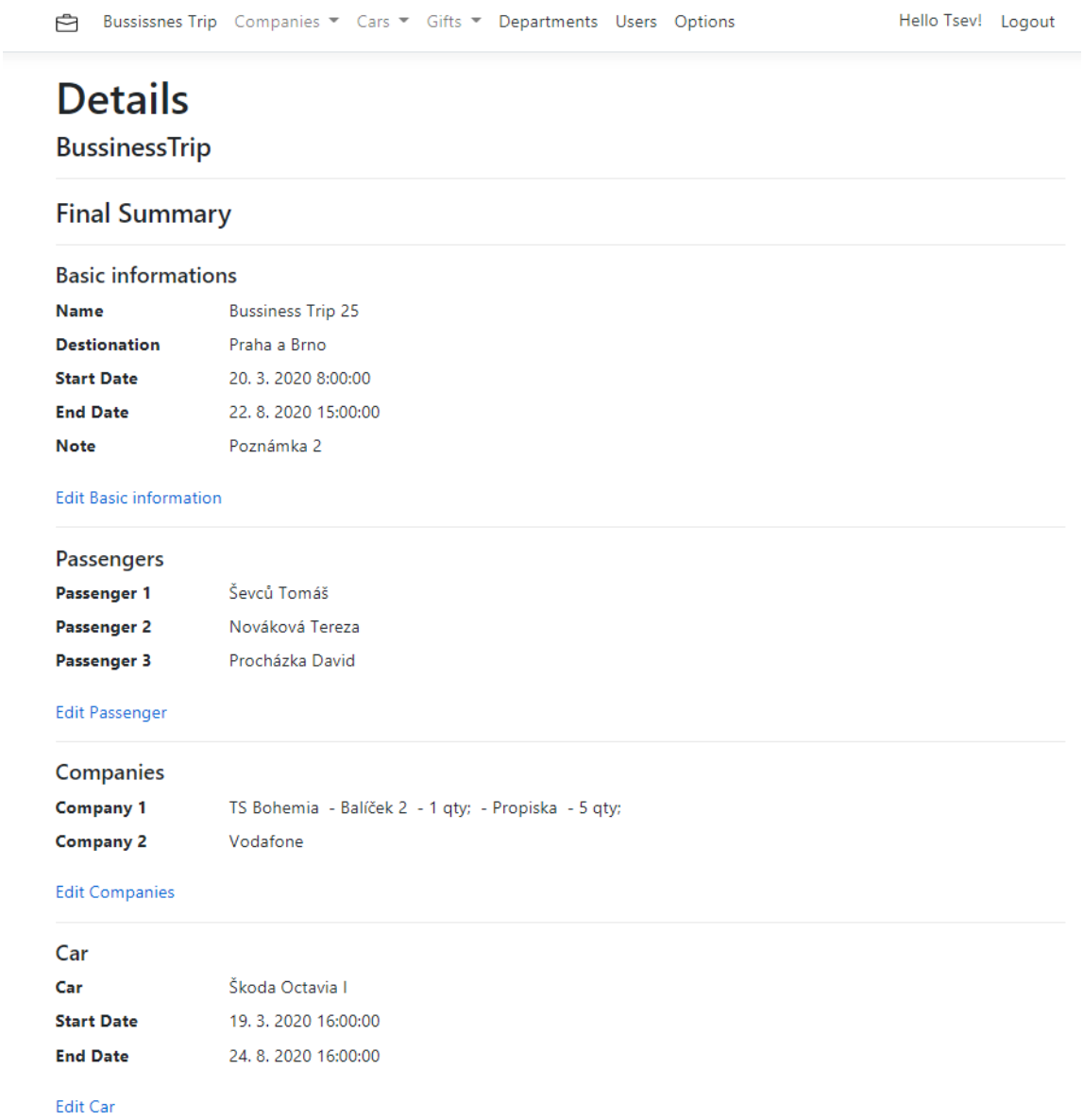
Na obrázku je přihlášen LCerny (Černý Lukáš) a proto má možnost smazat svoji zamítnutou služební cestu „Bussiness Trip 3“ a ze stránky detailu by ji mohl i editovat.

3.2.2 Detail

Detail služební cesty obsahuje všechny informace vyplněné uživatelem při vytváření.

Jsou rozdělené do 4 částí.

1. Základní informace – Název, Místo, Datum od do, poznámka
2. Cestující – jména uživatelů, kteří se SC zúčastní
3. Firmy – názvy firem, které se během SC navštíví a dárky, které se darují
4. Auto – název dopravního prostředku a datum od do jeho vypůjčení



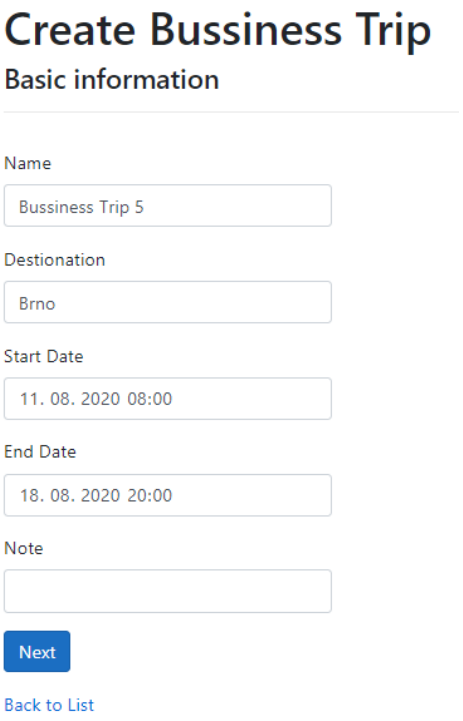
Obr. 22: Detail služební cesty

3.2.3 Vytvoření

Nová služební cesta se vytváří pomocí odkazu Create New na stránce Business Trip. Po kliknutí na odkaz se nejprve zkontroluje, zda daný uživatel nemá SC ve stavu Creating tedy rozpracovanou, kterou nedokončil a nedal ji odeslat. Pokud se taková nalezne, tak se z ní načtou data a uživatel pokračuje v její tvorbě.

Tvorba SC je rozdělena stejně jako detail SC do stejných 4 částí, ale navíc obsahuje pátou část, která je jen závěrečný přehled vyplněných informací a podobá se detailu SC a z té se nakonec odešle SC ke schválení.

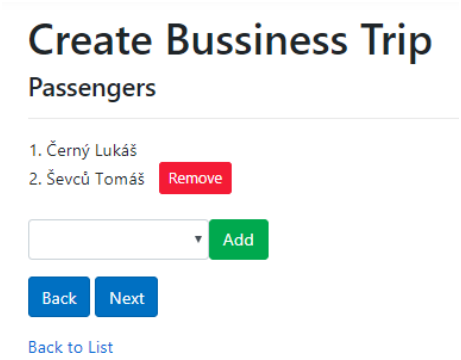
1. Základní informace (Basic information)
   Název (Name), Destination (Místo), Start Date (Datum začátku), End Date (Datum konce) jsou povinné pole. Pole s daty zobrazují kalendář pro snazší vybraní data.
   Note (Poznámka) je nepovinné pole a může zůstat prázdné.
   Správnost vyplnění je kontrolována při zmáčknutí tlačítka Další (Next) a na chybějící pole je upozorněno.



Obr. 23: Základní informace

2. Cestující (Passengers)

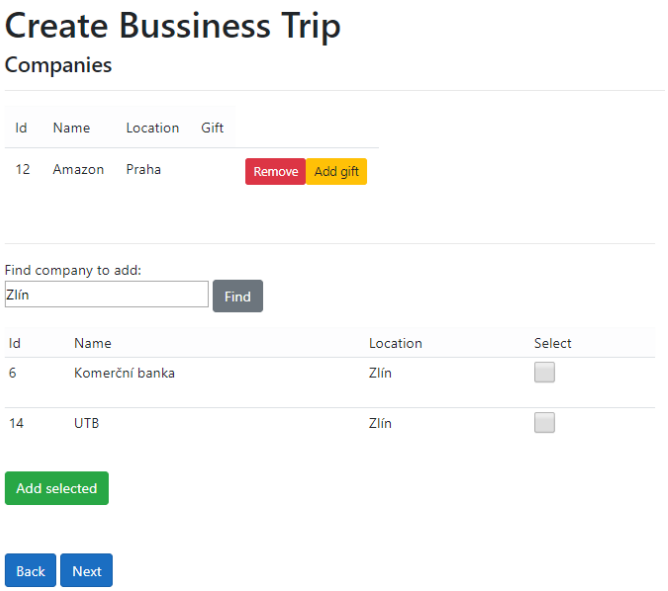
   Jako první cestující je vždy automaticky zvolen uživatel, který SC tvoří a ten nelze odebrat. Dál může přidávat (Add) další cestující ze seznamu uživatelů a ty následně může i odebírat (Remove).



Obr. 24: Cestující

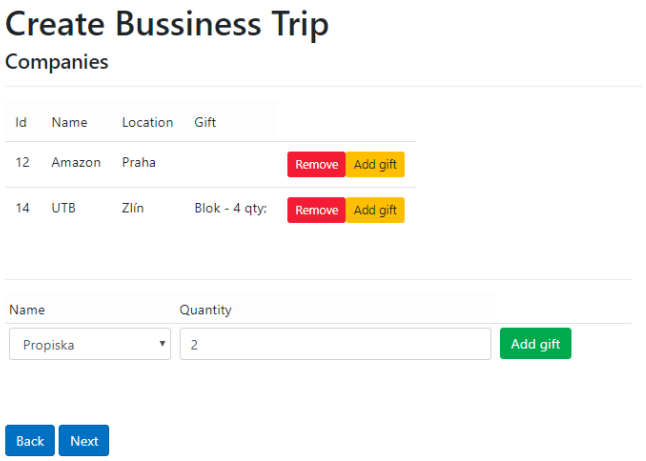
3. Firmy (Companies)

   Stránka pro přidávání firem, které se během SC navštíví. Uživatel může mezi firmami vyhledávat (pomocí názvu nebo místa firmy). Po vyhledání se mu zobrazí všechny firmy, které zadaný řetězec obsahují a pomocí checkboxu může přidat několik firem zároveň.



Obr. 25: Firmy

4. Dále k jednotlivým firmám může přidávat dárky, které jim při návštěvě předá a uživatel s rolí Gift vydá.



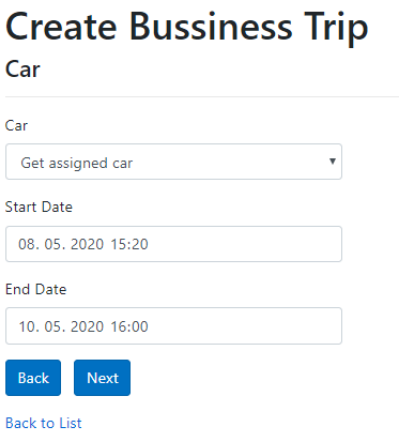
Obr. 26: Dárky

5. Dopravní prostředek (Car)

   Jako poslední část vyplňuje informace o dopravním prostředku. Většina uživatelů nemá privátní firemní auto, proto má volbu pouze „Nechat si přidělit auto“ (Get assigned car) nebo „Auto nepotřebuji“ (No car needed).

   Pokud zvolí možnost Nechat si přidělit auto, dopravní prostředek mu bude následně přidělen uživatelem s právem Car.

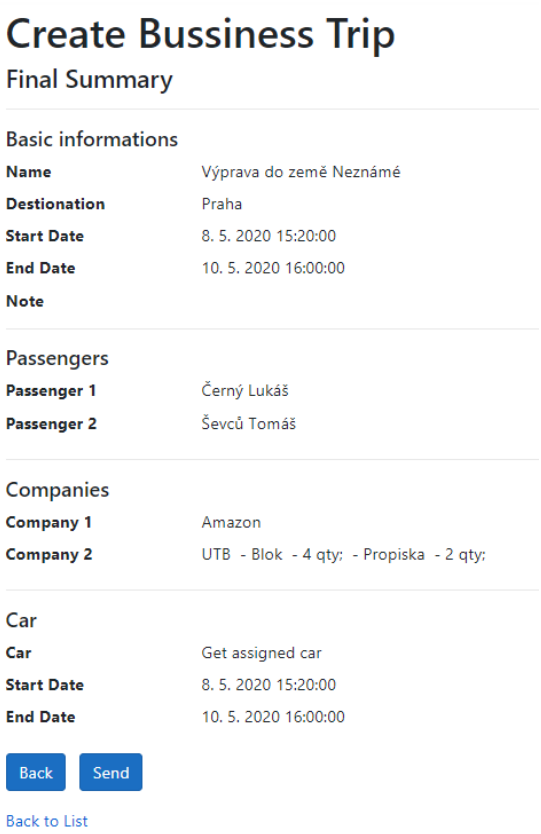
   Datum je automaticky vyplněný stejně jako datum začátku a konce SC, ale uživatel může chtít dopravní prostředek vypůjčit například o den dřív nebo vrátit až v pondělí, když mu končí SC v pátek.



Obr. 27: Dopravní prostředek

6. Závěrečný přehled (Final Summary)

   Obsahuje všechny informace vyplněné uživatelem. Takto si může uživatel zkontrolovat, zda vyplnil vše správně, případně se vrátit (Back) do jakékoliv části zpět a upravit ji.



Obr. 28: Závěrečný přehled

7. Následně SC odešle (Send) a tím je vytvořená. Admin může pomocí globálního nastavení zapnout mailing, v tom případě by se odeslal email všem vedoucím daných cestujících SC.

3.2.4 Editace

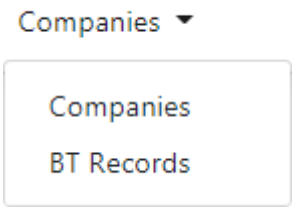
Editace SC se provádí z detailu SC. Editovat SC může její vlastní nebo admin. Editace je rozdělena stejně jako detail a vytváření do 4 částí, kde uživatel vždy edituje pouze příslušnou část – Základní informace, Cestující, Firmy, Dopravní prostředek.

Editace prvních tří částí změní schválenou SC na SC, která čeká znova na schválení. Stejně jako při vytváření, pokud je zapnutý v globálním nastavení mailing jde upozornění na úpravu a nutné znovu schválení vedoucím daných cestujících SC.

Editace dopravního prostředku nemá vliv na schválení SC, ale také informuje mailem, pokud je mailing zapnutý, všechny uživatele s rolí Car o změně dopravního prostředku.

3.3 Firmy

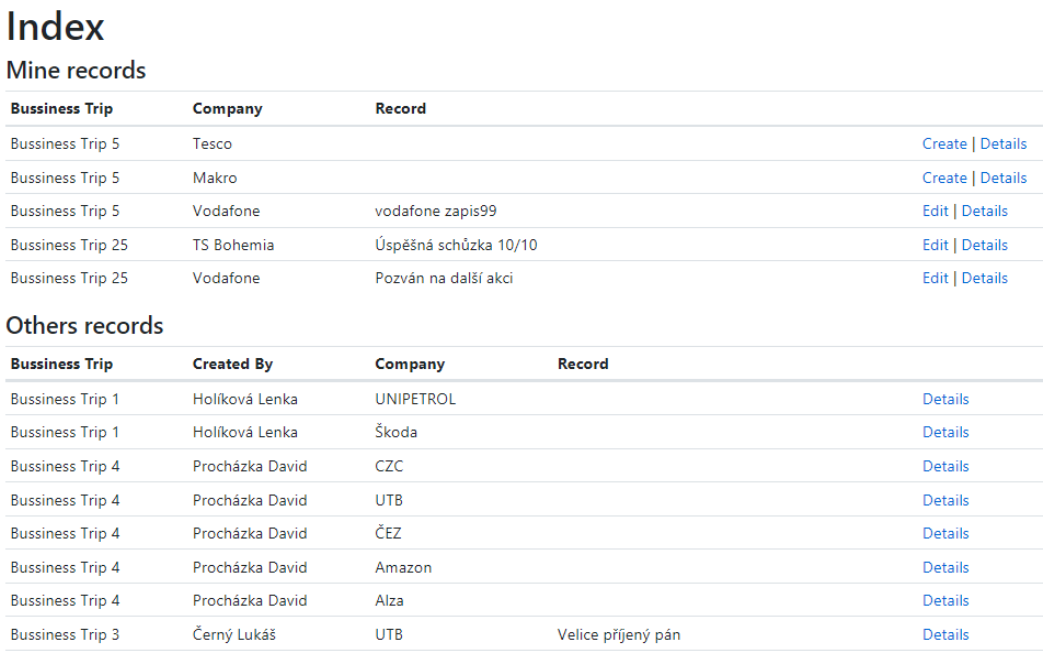
Firmy jsou rozděleny na dvě části.



Obr. 29: Firmy

- Firmy (Companies), kde všichni uživatelé mohou vytvářet, upravovat a skrývat firmy, které pak mohou přidávat do SC jako návštěvy. Detail navíc zobrazuje statistiku. Kolikrát firma byla navštívená a jednotlivé SC a odkaz na jejich detail.
- Zápisy ze SC z navštívených firem (BT Records).
  Stránka je rozdělena na dvě části. První jsou Mé zápisy (Mine records), kde se zobrazují návštěvy firem ze SC, které daný uživatel vytvořil. Zde může tvořit nové zápisy, kde nejsou a editovat existující. Druhá část jsou Zápisy ostatních (Others records), kde uživatel vidí zápisy ostatních.

Tlačítkem Detail (Details) se uživatel dostane na detail SC.



Index

Mine records

Bussiness Trip	Company	Record	
Bussiness Trip 5	Tesco		Create | Details
Bussiness Trip 5	Makro		Create | Details
Bussiness Trip 5	Vodafone	vodafone zapis99	Edit | Details
Bussiness Trip 25	TS Bohemia	Úspěšná schůzka 10/10	Edit | Details
Bussiness Trip 25	Vodafone	Pozván na další akci	Edit | Details

Others records

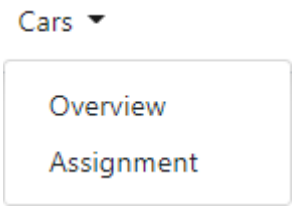
Bussiness Trip	Created By	Company	Record	
Bussiness Trip 1	Holíková Lenka	UNIPETROL		Details
Bussiness Trip 1	Holíková Lenka	Škoda		Details
Bussiness Trip 4	Procházka David	CZC		Details
Bussiness Trip 4	Procházka David	UTB		Details
Bussiness Trip 4	Procházka David	ČEZ		Details
Bussiness Trip 4	Procházka David	Amazon		Details
Bussiness Trip 4	Procházka David	Alza		Details
Bussiness Trip 3	Černý Lukáš	UTB	Velice příjený pán	Details

Obr. 30: Zápisy ze služebních cest

3.4 Dopravní prostředky

Dopravní prostředky může spravovat pouze uživatel s rolí Car nebo Admin.

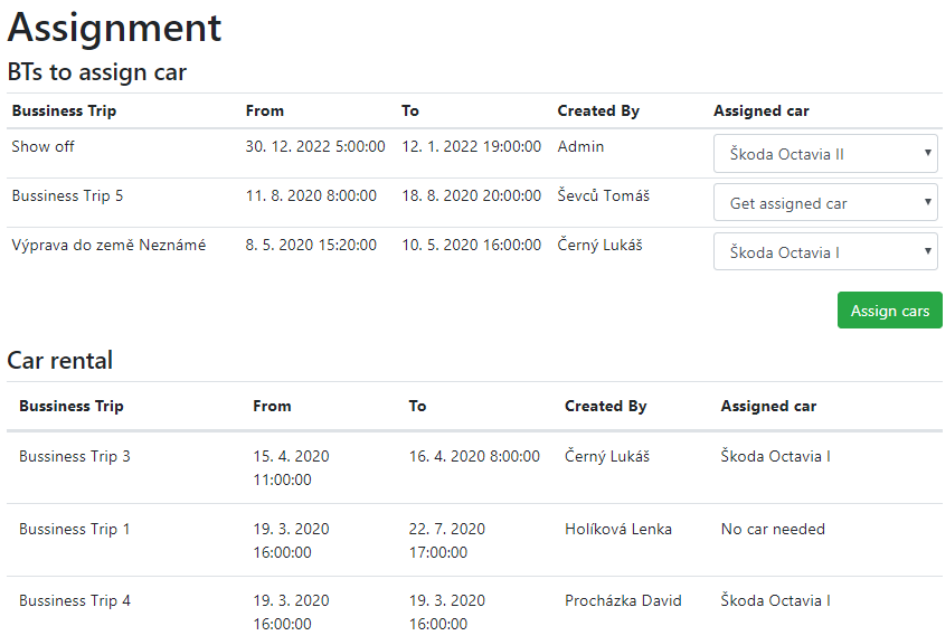
Jsou rozdělené na dvě části.



Obr. 31: Dopravní prostředky

- Přehled (Overview) slouží pro vytváření, editaci a skrývání dopravních prostředků. Dopravní prostředky mohou být privátní, tzn. mohou být přiděleny pouze uživatelem s rolí Car (Neprivátní dopravní prostředky si mohou uživatelé sami vybrat při tvoření SC). Dále mohou mít vlastníka, tzn. privátní dopravní prostředek s nastaveným vlastníkem si vlastník může vybrat při tvoření SC, aniž by do toho uživatel s rolí Car musel nějak zasahovat. Detail navíc zobrazuje statistiku. Kolikrát dopravní prostředek byl vypůjčen, kým a pro které SC a odkaz na jejich detail.

- Přidělení (Assignment) slouží pro přidělení dopravního prostředku služebním cestám, kde si uživatel zvolil možnost „Nechat si přidělit auto“ (Get assigned car). Stránka je rozdělená na dvě části. První jsou Služební cesty, které chtějí přidělit dopravní prostředek. Zde uživatel s právem Car zvolí dopravní prostředek, který chce vybrané SC přidělit a zvolí Přidělit auta (Assign cars). Zároveň může vybrat a přidělit několik dopravních prostředků. Druhá část již jen zobrazuje přidělené DP.

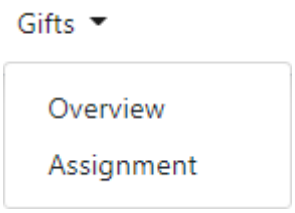


Obr. 32: Přidělení dopravního prostředku

3.5 Dárky

Dárky může spravovat pouze uživatel s rolí Gift nebo Admin.

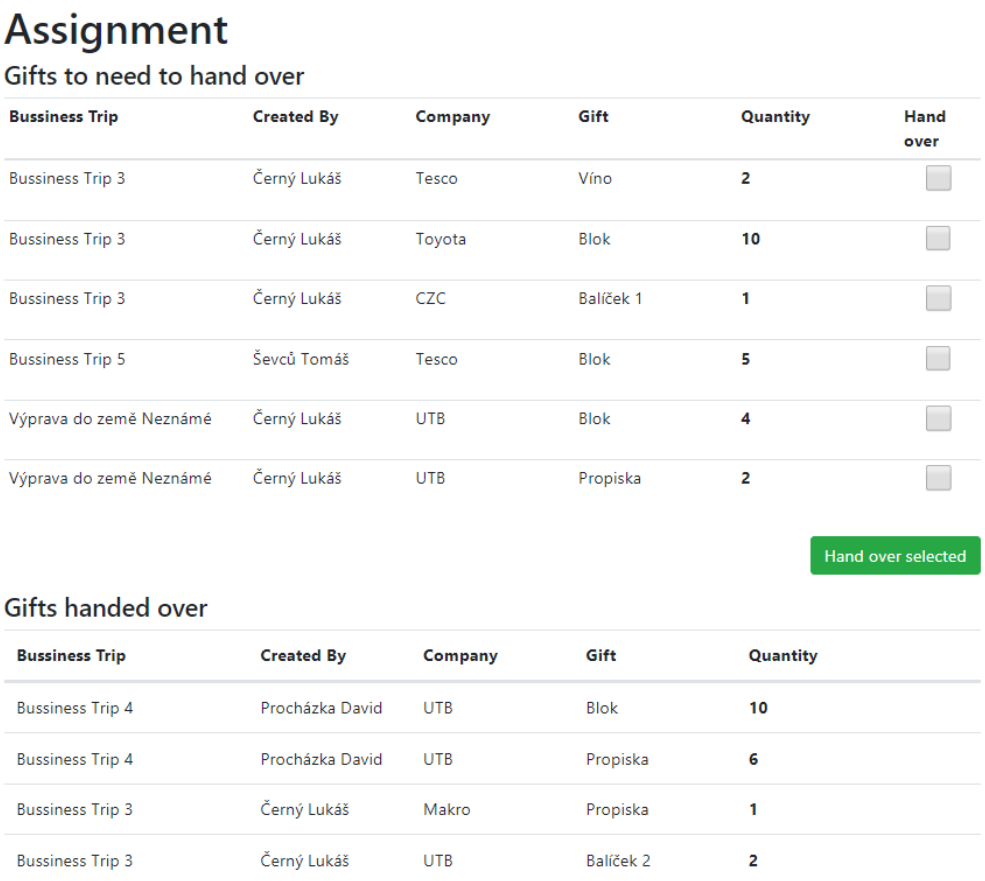
Jsou rozdělené na dvě části.



Obr. 33: Dárky

- Přehled (Overview) slouží pro vytváření, editaci a skrývání dárků. Detail navíc zobrazuje statistiku. Kolikrát byl dárek předán, kolikrát byl dárek zažádán a přehled jednotlivých firem s počtem daného dárku, který jim byl darován.

- Přidělení (Assignment) slouží pro přidělování / předávání dárků pro jednotlivé návštěvy SC. Jakmile správce fyzicky dárky předá
  Stránka je rozdělena na dvě části. První jsou dárky, které je potřeba předa. Správce dárků zde má přehled dárků, které má vydat, jakmile je fyzicky vydá, na této stránce pomocí checkboxu zaškrtne vydané a zvolí Vydat vybrané (Hand over selected). Druhá část jsou již vydané dárky.



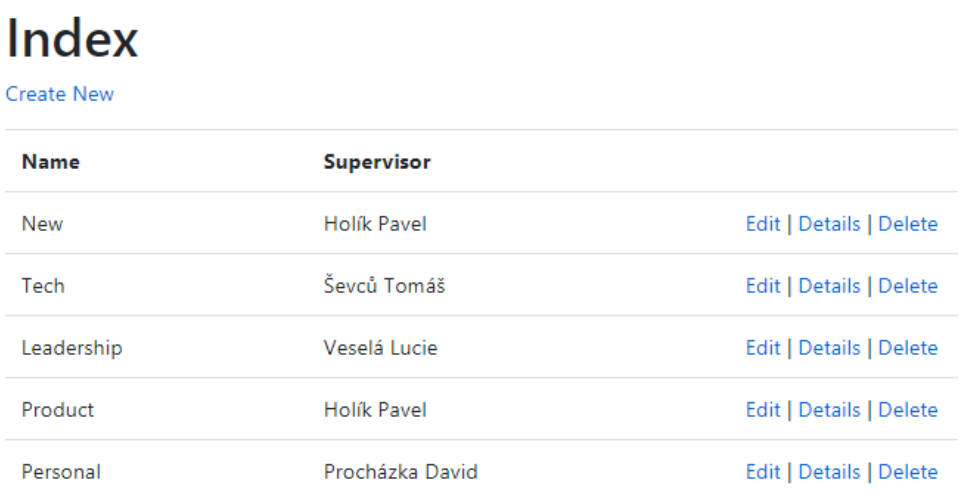
Obr. 34: Vydávání dárků

3.6 Správa uživatelů

Správa uživatelů obsahuje správu oddělení a jejich vedoucích a správu jednotlivých uživatelů. Toto nastavení má na starosti uživatel s rolí admin.

3.6.1 Oddělení

Oddělení (Department) slouží pro vytvoření struktury schvalování. Každé oddělení má název a svého vedoucího (Supervisor). Admin může vytvářet, editovat a mazat oddělení. Detail navíc zobrazuje všechny uživatele, kteří do daného oddělení patří.

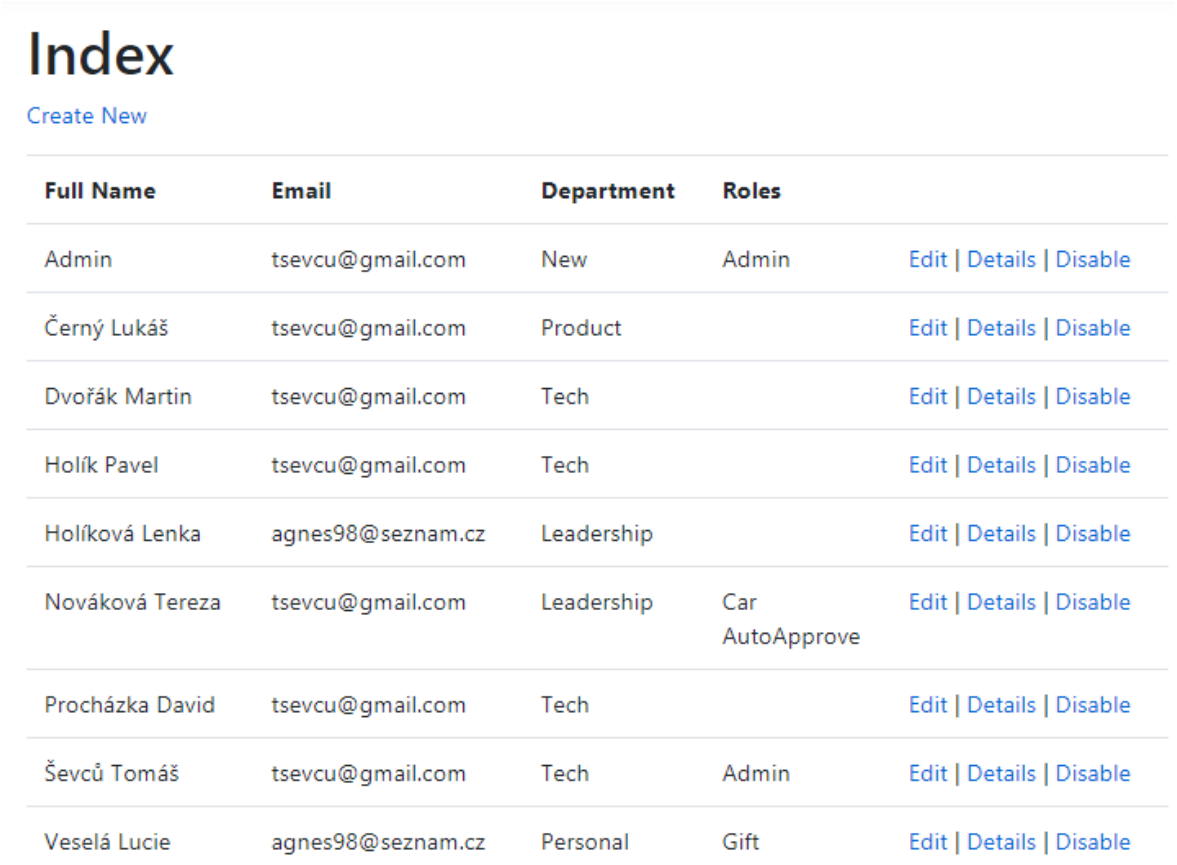


Obr. 35: Oddělení

3.6.2 Uživatelé

Uživatelé (Users) slouží pro správu registrovaných uživatelů nebo pro registraci nových. Admin zde může uživatelům měnit jejich vlastnosti – jméno, příjmení, přihlašovací jméno, email, a hlavně oddělení a nastavení jejich rolí. Role jsou Car, Gift, AutoApprove a Admin a každá má své jedinečná práva. Jejich názvy vypovídají o tom, co umožňují a jsou popsány výše v jednotlivých kapitolách.

Zároveň může vytvářen nové uživatele, což je nutné, pokud je externí registrace zakázána pomocí globálního nastavení.



Obr. 36: Uživatelé

3.7 Globální nastavení

Globální nastavení může nastavovat pouze uživatel s rolí Admin.

Globální nastavení slouží pro nastavování vlastností ovlivňující celou aplikaci. V současnosti jsou k dispozici dvě nastavení.

Nastavení externí registrace, které umožňuje nebo zamezuje registrace externích uživatelů. Pokud je externí registrace zakázána, tak nové uživatele smí vytvářet pouze Admin.

Nastavení mailingu, které zajišťuje zasílání informačních mailů po vytvoření SC, po editaci SC, po editaci DP. K rozesílání emailů je SendGrid, což je cloudová e-mailová služba, která poskytuje zasílání e-mailů. [17]

3.8 Prvotní spuštění

Pro spuštění aplikace je nutné mít například nainstalované Visual Studio, kde IIS Express reprezentuje server. Po otevření projektu se stáhnou všechny potřebné závislosti a balíčky. Aplikace pracuje s daty, které jsou uložené v databázi, konkrétně Microsoft SQL server databázi. Databáze se vytvoří automaticky při prvním spuštění a zároveň se naplní testovacími daty. Vytvoří se několik uživatelů, oddělení, služebních cest, dopravních prostředků, dárků pro testování a snadnou vizualizaci toho, jak aplikace funguje. Uživatelé, kteří se vytvoří mají přihlašovací jména: admin, Tsev, PHolik, LHolikova, TNovakova, MDvorak, LCerny, DProch, LVessela a všichni mají nastavené stejné heslo bez uvozovek „faiUTB2020*“.

Pro správnou funkčnost posílání emailů přes SendGrid je nutné nastavit uživatele a jeho klíč. Nastavení se provádí v Users Secrets projektu:

"SendGridUser": "BusinessTripsApp",

"SendGridKey": "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"

Současné omezení s tímto nastavením je 100 emailů denně.

ZÁVĚR

Realizací této bakalářské práce bylo dosaženo vytvoření webové aplikace pro správu služebních cest firmy se specifickým zadáním, které ke služebním cestám přidává správu nad přidělováním automobilních prostředků, přidělování dárků a správu uživatelů a strukturu schvalování.

V první části jsem se zaměřil na existující řešení, kde jsem zjistil, že aplikací pro správu služebních cest již existuje mnoho, ale žádná nesplňuje přesné zadání firmy.

Následně jsem vypracoval popis o všech použitých technologiích. Hlavní technologií je webový framework ASP.NET Core Razor Pages, pomocí které byla celá aplikace vytvořena. Doposud jsem měl zkušenosti pouze se skriptovacím jazykem ASP Classic, který je v dnešní době už velice zastaralý. Díky ASP.NET Core jsem si pak uvědomil spoustu bezpečnostních rizik při vývoji webové aplikace v ASP Classic, a také výhody jako mnohem snadnější udržitelnost a rozšiřitelnost díky objektovému programování. Na ASP.NET Core navazují technologie jako Entity Framework a LINQ, které usnadňují práci a propojení dat. Celá aplikace byla vytvořena ve vývojovém prostředí Microsoft Visual Studio.

Dále popisuji zabezpečení a zabránění v útoku v ASP.NET Core.

V druhé části se věnuji plánování a realizaci webové aplikace. Nejprve jsem definoval požadavky na systém rozdělené na funkční a nefunkční. Vytvořil jsem diagram použití pro všechny role uživatelů a pomocí tříd a Entity Frameworku navrhl diagram vztahů entit. Dále jsem definoval hlavní funkcionality aplikace jako vytvoření služební cesty, schválení služební cesty a strukturu uživatelů, jejich oddělení a vedoucích.

Realizovaná aplikace má základní velice jednoduchý vzhled zaměřený na funkcionalitu. Popisuji zde jednotlivé části aplikace. Hlavní částí jsou služební cesty, které jsou zároveň i úvodní stránkou po přihlášení, zobrazují seznam vytvořených služebních cest. Další části jako dopravní prostředky, dárky nebo správa uživatelů jsou pro požadovanou funkčnost také podstatné.

Při realizaci jsem se setkal s mnoha obtížemi, kvůli předchozí neznalosti ASP.NET Core. Ale ASP.NET Core je velice intuitivní a rád bych s ním pracoval nadále. S výslednou aplikací jsem spokojený, její tvorba mě bavila, a nakonec jsem do ní přidal i další funkcionality jako je globální nastavení nebo posílání emailů.

SEZNAM POUŽITÉ LITERATURY

[1] Groff, James a Weinberg, Paul. SQL: The Complete Reference. 3rd Edition. New York : McGraw-Hill Education, 2009. 978-0071592550.

[2] Freeman, Adam. Pro ASP.NET Core MVC. 6th ed. Edition. New York : Apress, 2016. 978-1484203989.

[3] What is Web Forms. Microsoft Docs. [Online] 21. 02 2014. [Citace: 11. 07 2020.] https://docs.microsoft.com/en-us/aspnet/web-forms/what-is-web-forms.

[4] ASP.NET overview. Microsoft Docs. [Online] 10. 08 2019. [Citace: 11. 07 2020.] https://docs.microsoft.com/en-us/aspnet/overview.

[5] Úvod do ASP.NET Core. Microsoft Docs. [Online] 17. 04 2020. [Citace: 11. 07 2020.] https://docs.microsoft.com/cs-cz/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1.

[6] Overview of ASP.NET Core MVC. Microsoft Docs. [Online] 12. 02 2020. [Citace: 11. 07 2020.] https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-3.1.

[7] Introduction to Razor Pages in ASP.NET Core. Microsoft Docs. [Online] 02. 12 2020. [Citace: 11. 07 2020.] https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=vi-sual-studio.

[8] Vemula, Rami. Real-Time Web Application Development: With ASP.NET Core, SignalR, Docker, and Azure. New York : Apress, 2017. 978-1484232699.

[9] Entity Framework Core. Microsoft Docs. [Online] 27. 10 2016. [Citace: 11. 07 2020.] https://docs.microsoft.com/en-us/ef/core/.

[10] Language Integrated Query (LINQ). Microsft Docs. [Online] 02. 02 2017. [Citace: 11. 07 2020.] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/.

[11] Arlow, Jim a Neustadt, Ila. UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky. Dotisk prvního vydání. Brno : Computer Press, 2011. 978-80-251-1503-9.

[12] Unhelkar, Bhuvan. Software Engineering with UML. Boca Raton : CRC Press/Taylor & Francis Group, 2018. 978-1138297432.

[13] Johnson, Glenn. Programming in HTML with JavaScript and CSS3. Redmond : Microsoft Press, 2013. 978-0-7356-7438-7.

[14] Jakobus, Benjamin. Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps. Second Edition. Birmingham : Packt Publishing, 2018. 978-178883490-2.

[15] Přehled zabezpečení ASP.NET Core. Microsoft Docs. [Online] 24. 10 2018. [Citace: 11. 07 2020.] https://docs.microsoft.com/cs-cz/aspnet/core/security/?view=aspnet-core-3.1.

[16] Koelsch, George. Requirements Writing for System Engineering. Herndon : Apress, 2016. 978-1484220986.

[17] SendGrid. [Online] https://sendgrid.com.

[18] ASP Tutorial. W3schools.com [online]. [cit. 2020-07-28]. Dostupné z: https://www.w3schools.com/asp/asp_introduction.asp

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SC	Služební cesta
DP	Dopravní prostředek
SQL	Structured Query Language
ASP	Active Server Pages
EF	Entity Framework
LINQ	Language Integrated Query
UML	Unified Modeling Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	Javascript
IDE	Integrated Development Environment

SEZNAM OBRÁZKŮ

SEZNAM TABULEK