Rezervační systém sportovišť

Jakub Anděl

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Anděl**
Osobní číslo:	**A19002**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Rezervační systém sportovišť**
Téma práce anglicky:	**Sports Facility Reservation System**

Zásady pro vypracování

1. Analyzujte problematiku a vypracujte literární rešerši na dané téma.
2. Popište technologie, které budou v práci použité.
3. Proveďte analýzu požadavků pro rezervační systém sportovišť.
4. Navrhněte a realizujte databázi a aplikaci.
5. Zajistěte zabezpečení aplikace a dat.
6. Proveďte testování aplikace a zhodnoťte provedené řešení.

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

Seznam doporučené literatury:

1. MALL, Rajib. *Fundamentals of software engineering*. PHI Learning Pvt. Ltd., 2018. ISBN 978-93-88028-03-5.
2. RAJLICH, Vaclav. *Software engineering: The current practice*. Chapman and Hall/CRC, 2019. ISBN 978-1-4665-1035-7.
3. DATE, Chris J. *Database design and relational theory: normal forms and all that jazz*. Apress, 2019. ISBN 978-1-4842-5540-7.
4. HARRINGTON, Jan L. *Relational database design and implementation*. Morgan Kaufmann, 2016. ISBN 978-0-12-804399-8.
5. PRICE, Jason. *C#: programování databází*. Grada Publishing as, 2005. ISBN 9788024709826.

Vedoucí bakalářské práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské 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 bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou 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 – bakalářskou 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 bakalářské 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 23.5.2022

Jakub Anděl, v. r.
podpis studenta

ABSTRAKT

Předmětem bakalářské práce je vytvoření webové aplikace rezervačního systému sportovišť. Teoretická část se zabývá popisem jazyka UML a UML diagramů, databáze a popis použitých technologií. Praktická část řeší funkčního analýzu pomocí UML, implementaci webové aplikace a její testování. Výsledkem je webová aplikace rezervačního systému sportovišť postavená na platformě ASP.NET MVC v jazyce C#.

Klíčová slova: ASP.NET (software), rezervační systémy, C# (programovací jazyk),UML (modelovací jazyk),databáze

ABSTRACT

The subject of the bachelor thesis is the creation of a web application for a sports facility reservation system. The theoretical part deals with the description of the UML language and UML diagrams, database and description of the technologies used. The practical part deals with functional analysis using UML, implementation of the web application and its testing. The result is a web application of a sports reservation system built on ASP.NET MVC platform in C#.

Keywords: Active Server Pages.NET (software), reservation systems, C# (programming language), UML (modeling language), databases

Chtěl bych poděkovat své vedoucí bakalářské práce doc. Ing. Zdence Prokopové, CSc., Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych rád poděkoval své rodině, která by mi byla oporou během celého studia.

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

OBSAH

ÚVOD

V dnešní době není nic neobvyklého, že si lidé rezervují termíny na různé akce či setkání pomocí internetu a vyznavači sportu nejsou výjimkou. Vyberou si místo, čas a způsob platby a rezervace je na světě. Usnadňuje to práci oběma stranám, uživatel vidí obsazenost areálu a k tomu vidí svou cenu, kterou zaplatí během pár kliků a vteřin. Pak už jen stačí přijít ve zvolený čas na preferované sportoviště bez jakýkoliv starostí. Mojí úlohou je takový rezervační systém sportovišť navrhnout a implementovat jako webovou aplikaci.

Teoretická část se zaobírá popisem modelovacího jazyka UML a popisem základních diagramů, které se běžně při modelování využívají. Součástí je také charakteristika databází zaměřena na používané termíny, databázový systém, vztahy mezi tabulkami a normalizaci databází. Další část se věnuje použitým technologiím, které byly během vývoje webové aplikace použity.

Samotná webová aplikace hraje hlavní roli v praktické části této práce. V programu Enterprise Architect se prostřednictvím informací z teoretické části navrhnou funkční a nefunkční požadavky a zbylé potřebné diagramy. Implementací a vývojem webové aplikace navazuje na předchozí analýzu systému. Závěrečná část se zaobírá testováním vytvořené aplikaci a zhodnocením dosažených výsledků.

I. TEORETICKÁ ČÁST

1 ANALÝZA PROBLEMATIKY

V teoretické části se zaobírá jazykem UML a diagramy použitými později v praktické části. Následují databáze a popis použitých technologií v práci.

1.1 Požadavky na systém

Cílem každého projektu je dodat zákazníkovi finální produkt podle jeho požadavků. Když jsou zadány nejasně, vede to k prodloužení harmonogramů a rostoucím nákladům na vývoj projektu. Dělíme je na funkční a nefunkční požadavky.

1.1.1 Funkční a nefunkční požadavky

Funkční požadavky specifikují, co má systém dělat a umět. Pomáhá vývojovému týmu kontrolovat, zda systém obsahuje vše, o co bylo žádáno. Pokud systém nesplní funkční požadavek, tak selže, protože není schopen dosáhnout něčeho, co musí dělat pro správné fungování. V případě vstupů funkční požadavky definují, co musí systém v reakci na ně dělat a co musí být jeho výstupem.

Nefunkční požadavky definují chování systém a omezení jeho funkčnosti. Na rozdíl od funkčních požadavků není nutné splnění nefunkčních požadavků, ale může to být žádoucí, aby to negativně neovlivnilo uživatelský zážitek, protože správně definované a provedené nefunkční požadavky usnadňují uživatelům použivatelnost systému a zvyšují jeho výkonnost. Příklady nefunkčních požadavků systému - rychlost, dostupnost, kapacita, použitelnost a spolehlivost. [1] [2]

Tabulka 1 Rozdíl mezi funkčními a nefunkčními požadavky [1]

Funkční požadavky	Nefunkční požadavky
Pomáhají pochopit funkce systému	Pomáhají pochopit výkon systému
Vysvětlují, jaké vlastnosti by měl systém mít	Vysvětlují, jak by měl systém fungovat (jak by se měl chovat)
Identifikují, co systém musí nebo nesmí dělat	Identifikují, jak by to měl systém dělat
Umožní systému fungovat, i když nejsou splněny nefunkční požadavky	Systém nebude fungovat pouze na základě nefunkčních požadavků

Zajišťuje, aby systém splňoval požadavky uživatelů	Zajišťuje, aby systém splňoval očekávání uživatelů
Jsou nezbytné pro provoz systému	Nejsou vždy nezbytné, ale mohou být žádoucí
Je snadné je definovat a dohodnout se na nich	Je těžší je definovat a dohodnout se na nich
Splnění těchto požadavků je povinné	Splnění těchto požadavků není povinné, ale může být žádoucí
Definují systém nebo jeho prvek	Definují kvalitu systému
Obvykle definovány uživatelem	Obvykle je definují softwaroví inženýři, vývojáři, softwaroví architekti nebo jiní techničtí odborníci
Lze zdokumentovat a pochopit prostřednictvím případů užití	Lze dokumentovat a chápat jako atribut kvality

1.2 UML

Unifikovaný modelovací jazyk (UML) používáme jak při vývoji softwaru, tak i mimo něj, napříč různými odvětvími. Pomocí něj vidíme, jak daný proces či struktura vypadá, nebo jak se chová. Cílí na to, aby diagramy byly snadno pochopitelné pro každého. Dnes je UML standardní notací pro vývojáře, projektové manažery a další profese v oblasti technologií.

Vznik UML se datuje do devadesátých let minulého století, kdy tři softwaroví inženýři Grady Booch, Ivar Jacobson a James Rumbaugh chtěli vymyslet méně chaotický způsob dokumentace stále komplikovanějšího vývoje softwaru.

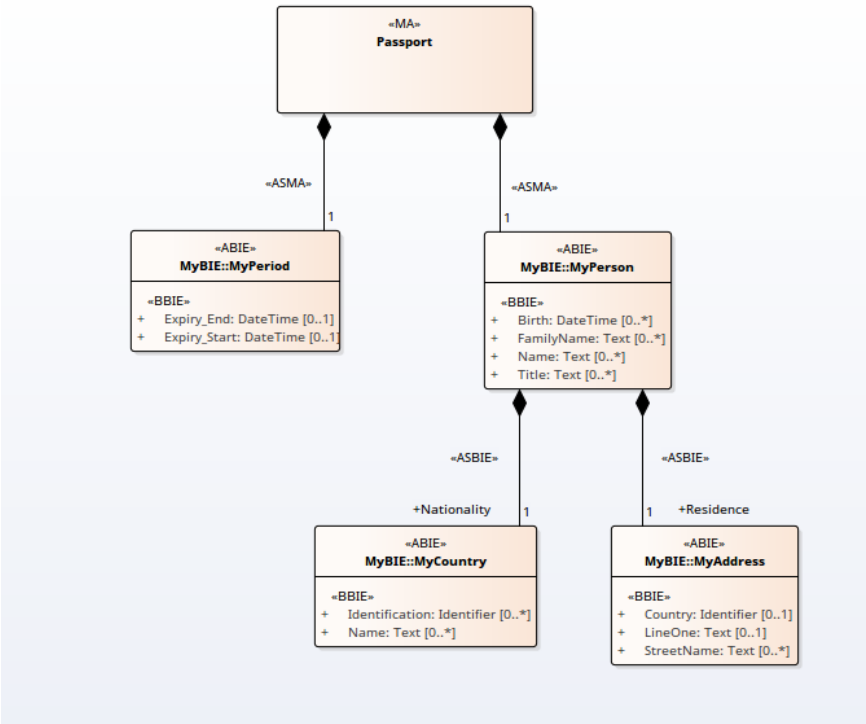
V UML rozlišujeme dva druhy diagramů - strukturální a diagramy chování. Strukturální diagramy znázorňují strukturu systému a toho jak například vypadá databáze. Nejznámějším zástupcem strukturálních diagramů je model tříd. Naopak diagramy chování se soustředí na funkcionalitu systému a všeho, co se v navrhovaném sytému musí stát. Tady najdeme model případů užití. [3]

1.2.1 Diagram tříd

Diagram tříd v jazyce UML je zástupcem strukturálních diagramů, který popisuje strukturu daného systému a zobrazuje objekty, její atributy, operace (metody) a vztahy mezi objekty. Považuje se za základní stavební kámen objektově-orientovaného modelování.

Objektem se rozumí instance třídy, která slouží jako předpis daného objektu. Samotná třída popisuje pomocí atributů, jak daný objekt bude po inicializaci vypadat a s jakými metodami budeme moct pracovat. Pro kompletnost třídy je důležité nastavit atributům a operacím viditelnost. Používáme 3 druhy modifikátory přístupů - veřejný přístup (+), soukromý přístup (-) a chráněný přístup (#). [4]

Aby mohly třídy mezi sebou spolupracovat, musí být propojeny a to pomocí vztahů. Statický vztah mezi třídami označujeme jako asociace a graficky se znázorňuje čárou mezi třídami. Kromě ní existuje na úrovni instance ještě vztah agregace, kompozice a na úrovni třídy dědičnost a realizace. V agregačním vztahu podřízená třída není silně závislá na rodičovské třídě, což znamená, že podřízená třída bude existovat i přes smazání nadřazené třídy. V diagramu se značí pomocí prázdného kosočtverce u rodičovské třídy. Kompozice se velmi podobá agregaci, ale vztah mezi třídami je silnější a při smazání rodiče zmizí i potomek. Značí se vyplněným kosočtvercem u původní třídy. Dědičností označujeme vztah, kdy podřízená třída přebírá všechny atributy a metody rodičovské třídy. Tento vztah se označuje nevyplněným trojúhelníkem u rodičovské třídy. Vztahem realizace přebírá podřízená třída všechny metody z původní třídy. V diagramu má takový vztah přerušovanou čáru s trojúhelníkem na konci u rodičovské třídy. [4] [5]



Obrázek 1 Příklad diagramu tříd v UML [6]

1.2.2 Diagram případů užití

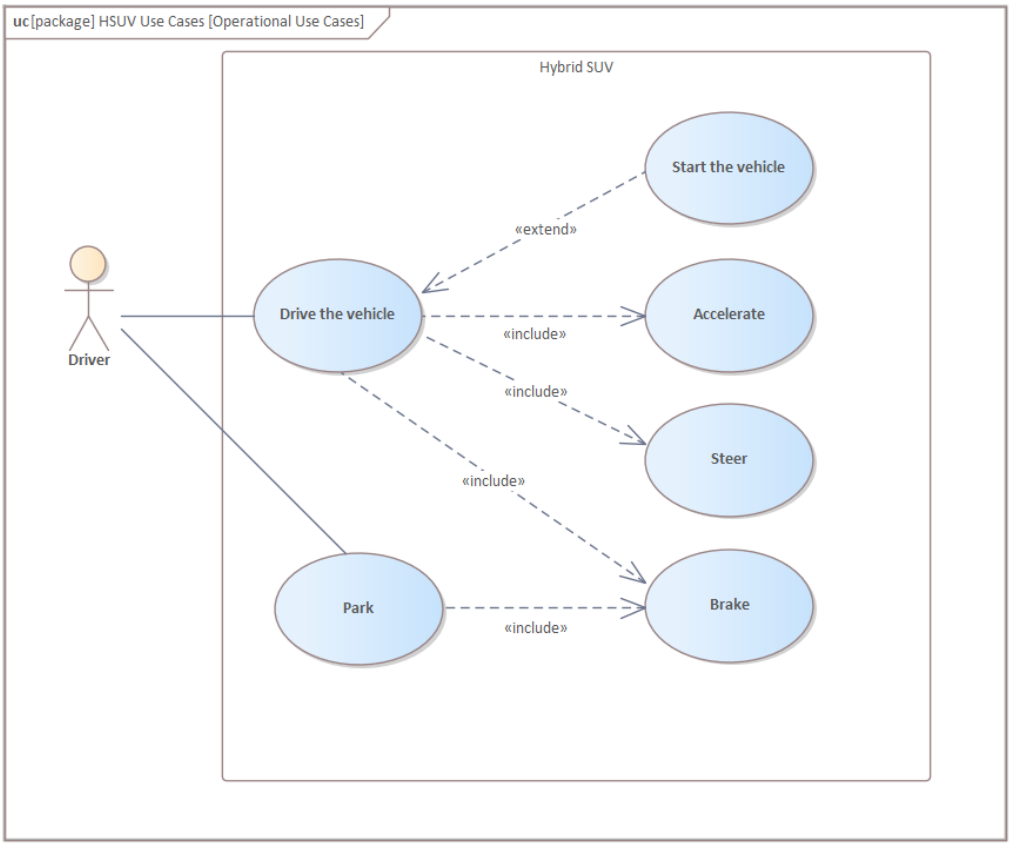
Diagram případů užití patří mezi diagramy chování jazyka UML a specifikuje chování systému, interakce mezi systémem a jeho aktéry. Popisuje tak, co systém dělá a jak ho aktéři používají, ale bez toho, jak systém funguje zevnitř. Obvykle se vytváří v raném vývoji projektu a pomáhá při určování požadavků na systém a při představě, co by měl systém umět.

Skládá se z případů užití, aktérů a vztahů mezi nimi. Případ užití popisuje samotnou funkci vybranou aktérem, kterou systém vykonává a měl by mít nějaký pozorovatelný výsledek přinášející pro něj nějakou hodnotu. Aktér představuje uživatele a další role, které pracují se systémem. Může jím být člověk, stroj nebo systém. [7]

V celém diagramu můžeme použít 5 různých vztahů:

Asociace mezi aktérem a případem užití - Tento vztah se používá v každém diagramu případů užití a je to základní vztah mezi aktérem a případem užití. Každý aktér musí být spojen minimálně s jedním případem užití, ale zároveň jich může mít více najednou. K jednomu případu užití můžeme přiřadit více aktérů.

- Generalizace aktéra - Jeden aktér může dědit roli jiného aktéra a taky jeho případy užití. Potomek má tak k dispozici jak svoje případy užití, tak případy užití svého rodiče.
- Generalizace případu užití - Vztah podobný generalizaci aktéra a používá se, když dva případy užití mají společné chování. Příklad: Případ užití s názvem Zaplacení objednávky lze rozdělit na "zaplatit platební kartou" a "zaplatit "zůstatkem na účtu".
- Vztah extend mezi dvěma případy užití - Rozšiřuje základní případ užití a přidává mu další funkce. Obvykle je nepovinný a spouští se podmíněně. Rozšířený případ užití je závislý na základním případu užití, který by měl být sám o sobě smysluplný a nezávislý.
- Vztah include mezi dvěma případy užití - Základním případem užití je chování, které se více případech užití opakuje a navazují na něj specifické případy užití. Rozšiřující případy užití je nutné vytvořit, protože bez nich je základní případ užití neúplný. [8]

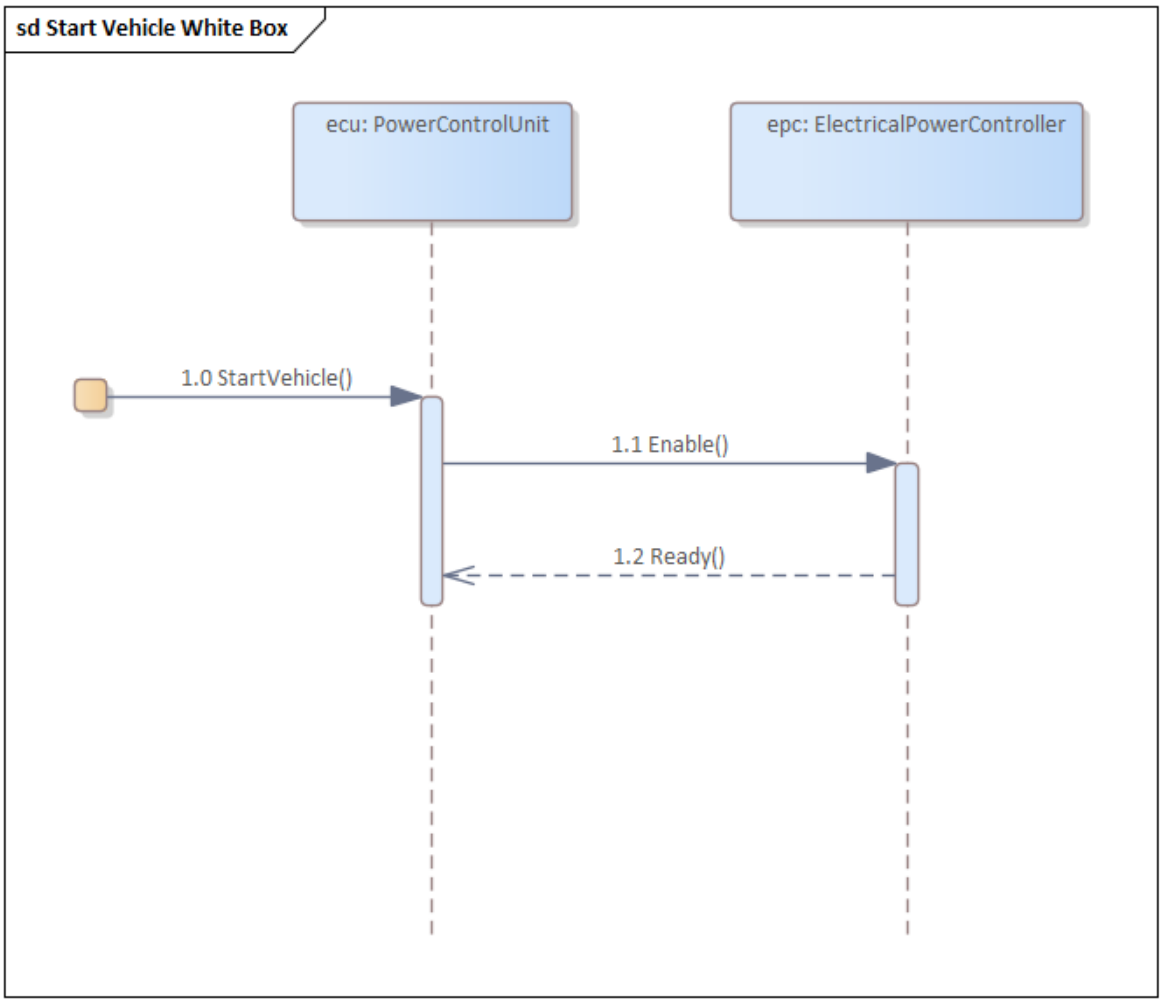


Obrázek 2 Příklad diagramu případu užití v UML [9]

1.2.3 Sekvenční diagram

Sekvenční diagram, zástupce interakčních diagramů, které se řadí do diagramů chování jazyka UML. Většinou sekvenční diagram zachycuje sekvenci jednoho případu užití a zobrazuje interakce mezi systémovými objekty v sekvenčním pořadí.

Aby bylo možné vytvořit sekvenční diagram, je zapotřebí znát všechny detaily, což pomáhá najít překážky dříve než během implementace. Samotné grafické znázornění sekvenčního diagramu se skládá z určení čár života (lifelines), které představují instance objektů. Každá čára života je umístěna v horní části diagramu a vede z ní svislá přerušovaná čára. Pro pochopení úkonů a znázornění operací se používají zprávy (např. Detaily objednávky), které znázorňujeme pomocí vodorovných šipek a textem. Aktér zastává v diagramu uživatele, kterého se celá sekvence týká - například přihlášený uživatel si chce zobrazit své objednávky. [10]



Obrázek 3 Příklad sekvenčního diagramu v UML

1.3 Databáze

Databáze slouží jako úložiště záznamů pro aplikace, se kterými se následně pracuje. S databázemi se setkáváme každý den a to nejen ve virtuálním světě. V papírově tu byly ještě před nástupem počítače, když se spousta záznamů uchovávalo v kartotékách, knihovnách a podobně. Problém ale byl, když jsme chtěli něco najít v záznamech najít nebo dokonce zálohovat, abychom předešli zničení nebo ztrátám.

Co se týče těch počítačových databází, Charles Bachman vymyslel v 60. letech 20. století první databázový systém s názvem Integrated Data Store (IDS), na který posléze navázal IBM se svým Information Management System. Obě zmíněné databáze patřily do kategorie navigačních databází. Aby uživatel našel, co hledá, musel projít celou databází.

V 70. letech článek E. F. Codda zavedl termín relační databáze, která se používá dodnes. V tomto typu databáze mohou tabulky mít mezi sebou vztah (relace) a jsou tak vzájemně propojeny. Oproti navigačním databázím nevyžadovaly tolik prostoru na disku, a tak náklady na ukládání záznamů byly nižší. Společnost IBM o pár let později přišlo se svým návrhem relační databáze s názvem System R, který jako první použil strukturovaný dotazovací jazyk SQL.

V počátku posledního desetiletí 20. století nebyly objektově-orientované databáze příliš populární, jelikož přepsání stávajících databází bylo časově a finančně náročné. Příchod World-Wide-Web napomohl databázovému průmyslu, protože vzrostla poptávka po systémech klient-server. V roce 1995 přišlo na trh MySQL, který oproti systémům od Oracle nebo Microsoft má otevřený kód.

Osmdesátá léta znamenala zánik navigačních databází a naopak růst používání relačních databází a jazyk SQL se stal standardním jazykem pro databáze. V polovině 80. letech vznikly objektově-orientované systémy, které se na záznamy dívají jako na objekty a pracují s jazyky, které umožňují objektové orientované jazyky.

Jazyk SQL není součástí všech databázových systémů. Od roku 1998 se začaly používat NoSQL databáze, které nevyužívaly jazyk SQL. Oproti relačním databázím jsou vhodnější pro větší a rozmanitější datové soubory.

V novém tisíciletí se začalo zaměřovat především na velké objemy dat a také na jejich bezpečnost kvůli novým legislativám jako je třeba GDPR. [11]

1.3.1 Terminologie

- Databáze - Kolekce dat uložená v tabulkách, která mohou být vzájemně propojené.
- Tabulka nebo entita - Složená z řádků a sloupců a obsahuje data.
- Atribut - V tabulce popisuje atribut entitu pomocí sloupce. Atributem je například sloupec Jméno v tabulce Zákazník.
- Klíč - Účelem je jednoznačná identifikace záznamu a zabránění duplicitám. Rozlišujeme tři typy klíčů - kandidátní, primární a cizí.
- Kandidátní klíč - Množina sloupců a každý z nich může jednoznačně identifikovat záznam.
- Primární klíč - Jednoznačně identifikuje záznam, většinou sloupec obsahující název ID.
- Cizí klíč - Pomocí cizího klíče spojujeme záznam v tabulce B se záznam v tabulce A. Podmínkou je, že existuje v tabulce A jako primární klíč.
- NULL - Hodnota v databázi, která určuje, zda řádek může být prázdný či nikoliv. SQL používá IS NULL (může být prázdný) a NOT NULL (nemůže být prázdný).
- ACID - Atomicity, Consistency, Isolation, Durability - 4 vlastnosti, které by měly splňovat databázové transakce.
- Atomicity (atomicita) - Každá transakce se musí splnit buď kompletně, nebo vůbec.
- Consistency (konzistence) - Všechny transakce musí splnit pravidla definované databází.
- Isolation (izolace) - Každá transakce pracuje nezávisle na dalších transakcích a pokud probíhají nějaké zároveň, nesmí se ovlivnit.
- Durability (trvalost) - Výsledek transakce bude v databázi i přes její selhání, například vlivem ztráty napájení. [12]

1.3.2 Databázový systém

Databázový systém hraje roli prostředníka mezi koncovým uživatelem a databází a umožňuje mu vytvářet, číst, aktualizovat a mazat data v databázi. Skládá se ze dvou částí - systém řízení báze dat, což je část starající se o správu databáze. Doplňuje jí báze dat, jež umožňuje manipulaci se záznamy v SŘBD.

Mezi známé relační databázové systémy patří MySQL, Microsoft Access, Microsoft SQL Server, Oracle nebo MariaDB. [13]

1.3.3 Dělení databází

Ve světě digitální techniky dělíme databáze podle toho, jak jsou organizovány a ukládaný data v ní. První tři typy zastupují nerelační databáze, které nekladou důraz na strukturu a relace mezi tabulky a daty. Mezi běžné typy patří:

1.3.3.1 Prostý databázový soubor

V tomto typu databáze jsou data uloženy sekvenčně v textovém souboru ve formátu prostého textu. Používá jen jednu tabulku bez žádných vazeb a je tak jednoduché ji vytvořit či přidávat do ní nové data, ale právě kvůli tomu ji tíží několik nevýhod jako například pomalé vyhledávání a redundance dat. Příkladem může být obyčejný CSV soubor. Pro oddělení řádků na sloupce se používají oddělovače, typicky čárka.

1.3.3.2 Hierarchická databáze

Hierarchická databáze používá stromovou strukturu se vztahem rodič - potomek a kardinalitu 1:N. Oproti prostému databázovému souboru se snaží eliminovat opakující se data a vyhledávání je tak efektivnější a rychlejší, ale za předpokladu, že známe použité vztahy. Ačkoliv se dnes už tento typ nepoužívá, setkat s ní můžeme v registru systému Windows.

1.3.3.3 Dokumentově orientované databáze

Dokumentově orientované databáze uchovává záznamy ve formě dokumentů podobných formátu JSON. JavaScript Object Notation (JSON) reprezentuje strukturované data a používá se k přenosu dat mezi serverem a klientem, aby se mohly zobrazit klientovi na webové stránce. Mezi nevýhody dokumentově orientované databáze patří možnost opakujících se záznamů, absence vazeb mezi daty a pomalé vyhledávání požadovaných hodnot. K řízení tohoto typu databází lze využít například databázové systémy Redis a CouchDB.

1.3.3.4 Relační databáze

Relační datový model se používá dnes v systémech nejvíce. Samotná databáze se skládá z tabulek tvořené sloupci (atributy) a řádky (záznamy). Jednotlivým sloupcům bychom měli před vložením záznamů nastavit název, datový typ, zda může být záznam prázdný a klíče, jestli jsou potřeba. Pro uživatelskou manipulaci s daty se používá dotazovací jazyk SQL.

Máme-li například tabulku o zákaznících, sloupce představují jména či adresy, každý řádek obsahuje údaje o jednotlivém zákazníkovi. Tabulky lze propojovat navzájem pomocí primárního klíče v původní tabulce a cizích klíčů v cílové tabulce. Vztahem primárního a cizího klíče vzniká relace. Po propojení tabulky a vložení cizího klíče vidíme v cílové tabulce na řádku index z původní tabulky, který odkazuje na primární klíč původní tabulky.

Použití tohoto modelu zpřehledňuje práci s databází, protože jednotlivé záznamy rozdělíme do kategorií, kam patří. Dále umožňuje jednoduché přidávání řádků či sloupců a tabulek, bez změny struktury celé databáze.

Naopak neustále přidávání nových sloupců, řádků či tabulek může udělat databázi udělat natolik složitou, že se tvoření vztahů mezi tabulkami může stát obtížné. To pak zpomaluje dotazování na jednotlivá data.

Mezi nejznámější relační databáze patří MySQL, PostgreSQL, MariaDB a SQLite. [14] [15]

1.3.4 Vztahy mezi tabulkami

V relačních databázích použití vztahů umožňuje předejít nadbytečným datům. Vztahy fungují na tom, že se porovnávají data v klíčových sloupcích, které jsou propojené klíči. Existují celkem 3 druhy vazeb a ty závisí na tom, jak jsou definovány sloupce a jejich tvorbu si můžeme pomoct, když si vztah představíme v reálném životě. [16]

1.3.4.1 Relace typu 1:N

Nejčastěji se setkáme se vztahem 1:M. Tento vztah říká, že jeden záznam v první tabulce odpovídá mnoha záznamům v tabulce A. V reálném světě tomu odpovídá vztah například letadlo - pasažér. V letadle pojme více pasažérů, ale pasažér je v jednu chvíli na palubě jen jednoho letadla.

1.3.4.2 Relace typu M:N

Vztah mnoho-na-mnoho reprezentuje relaci, která říká, že více záznamů z tabulky A má mnoho odpovídajících záznamů v tabulce B a naopak. Definováním třetí tabulky vytvoříme tento vztah. Primární klíč spojovací tabulky je tvořen z cizích klíčů tabulky A i B. Příklad z reálného světa - více studentů si může zapsat více předmětů. V jednom okamžiku může student mít více předmětů a zároveň každý předmět může být zapsán více studenty.

1.3.4.3 Relace typu 1:1

Vztah jeden-ku-jednomu definuje relaci, která odpovídá situaci, kde jeden záznam z tabulky A odpovídá nejvýše jednomu záznamu z tabulky B. Tento vztah vzniká, když jsou oba související sloupce primárními klíči. S tímto vztahem se moc nesetkáváme, protože spousta dat by byla v jedné tabulce. Hodí se k rozdělení tabulky s mnoha sloupci, izolování části tabulky z bezpečnostních důvodů nebo k ukládání krátkodobých dat.

1.3.5 Normalizace databáze

Cílem normalizace databáze je pomocí pravidel správně vytvářet tabulky, aby se předešlo nadbytečným a nekonzistentním záznamům. Takové data plýtvají místo na diska a tvoří problémy s údržbou. Je mnohem jednodušší provést změnu záznamu v jedné tabulce, než na všech místech, kde se záznam vyskytuje.

Existuje několik pravidel pro normalizaci databáze a každé z nich se nazývá normální forma. Ačkoliv jich existuje vícero, považuje se třetí normální forma za tu nejvyšší, která je pro většinu aplikací nezbytná. [17]

1.3.5.1 První normální forma

Tabulka splňuje první normální formu, pokud každý sloupec v tabulce je unikátní, pro každou sadu souvisejících dat musí být vytvořena samostatná tabulka, musí obsahovat primární klíč a řádky ani sloupce se nesmí duplikovat. Správně by měl každý sloupec v tabulce obsahovat jen jeden záznam (atomicita). Tím docílíme správného použití relačních dotazů pro získání dat z databáze. Pokud nemáme splněnou první normální formu a data se opakují, získané záznamy se nám jeví jako nejednoznačné. [18]

1.3.5.2 Druhá normální forma

Druhou normální formou pokračuje normalizace a navazuje na první normální formu, jejíž podmínky musí být splněny. Aby byla splněna druhá normální forma, je třeba splnit, aby záznamy, které jsou ve vztahu M:N, měly vytvořené samostatné tabulky. Tyto tabulky by se měly propojit cizím klíčem a záznamy by neměly záviset na ničem jiném než na primárním klíči. Častá chyba je, že se například adresa uživatele píše neustále do tabulky objednávek místo toho, aby byla jednou a na jednom místě v tabulce Uživatelé. [19]

1.3.5.3 Třetí normální forma

Třetí normální forma navazuje na předchozí dvě normální formy a musí tak splňovat jejich podmínky a k tomu splnit pravidlo, které říká, že hodnoty záznamu, které nejsou součásti klíče daného záznamu, do tabulky nepatří a měly by rozděleny do samostatné tabulky. Uvažujme tabulku s názvem Zaměstnanci, kde najdeme sloupce Město a PSČ, které jsou ale na sebe závislé a správně bychom měli vytvořit novou tabulku Města a poté je propojit pomocí klíčů. [20]

1.3.6 Dotazovací jazyk SQL

Dnešní systémy řízení báze dat umožňují uživatelům ukládat, přistupovat, modifikovat záznamy efektivně bez napsání velkého množství kódu. Dříve byli uživatelé odkázáni na programátory, kteří byli nuceni napsat program v programovacím jazyce, například COBOL, aby mohli přistupovat k datům v databázi. Přístup k datům tehdy tedy nebyl praktický a vyžadoval služby zkušeného programátora.

Uživatelé nebyli s touto situací spokojeni, protože aby si mohli jen seřadit data podle nějakého kritéria, museli buď využít program, který již splňoval jejich požadavky, nebo služeb programátora, aby takový program napsal, což v mnoha případech kvůli jednorázovým dotazům nebylo ekonomicky výhodné. Postupem času rostl počet uživatelů, kteří požadovali snadnější přístup k záznamům.

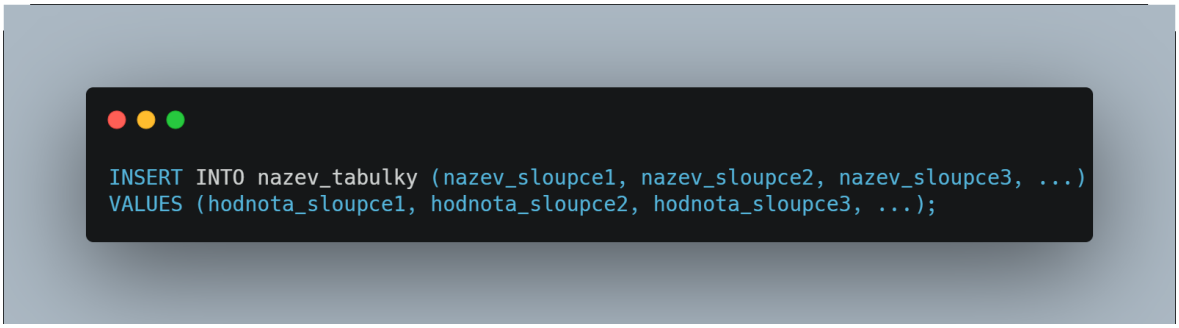
Cílem bylo vytvořit jazyk, kterým mohli své požadavky vyjádřit. Takový požadavek se definuje jako dotaz a jazyk jako dotazovací. Za tímto účelem bylo vyvinuto několik dotazovacích jazyků a nejoblíbenějším se stal jazyk SQL - Strukturovaný dotazovací jazyk. Vyvinula ho společnost IBM v 70. letech 20. století, v letech 1986 a 1987 se stal standardem ANSI a posléze ISO a dnes se používá ve spoustě databázových systémech. [21]

1.3.6.1 CRUD

Pod akronymem CRUD se skrývá čtyři anglická slova - Create, Read, Update, Delete. Tyto 4 základní funkce pro práci s daty tvoří základ jazyka SQL a patří mezi nejpoužívanější příkazy tohoto jazyka.

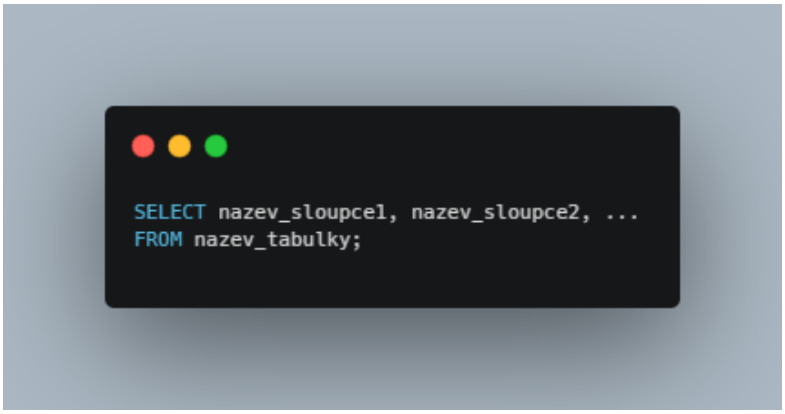
Vysvětlení jednotlivých elementů slova CRUD:

Create (INSERT) - V jazyce SQL se používá příkaz Insert a s jeho použitím vkládáme záznamy do tabulky, případně jednotlivých sloupců.



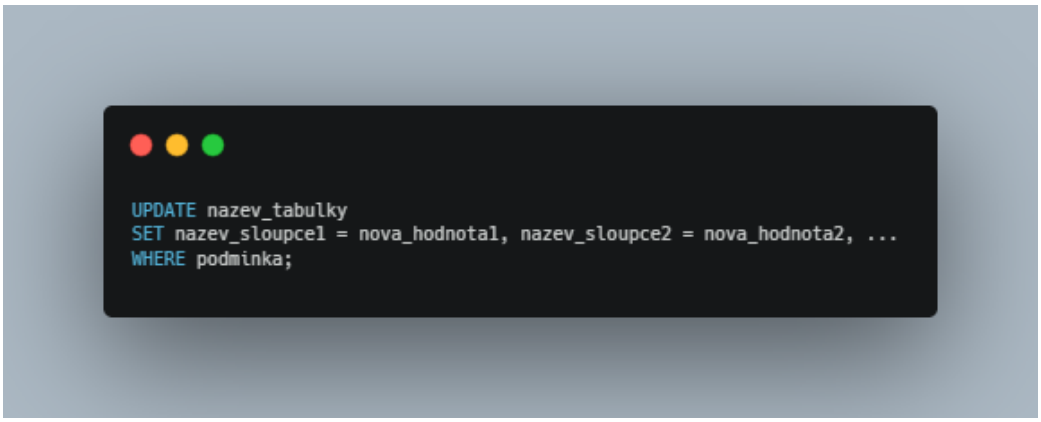
Obrázek 4 Příklad příkazu INSERT v jazyce SQL

Read (SELECT) - Pro zobrazení dat z databáze slouží příkaz SELECT, kde vybíráme požadované názvy sloupců, které chceme zobrazit z tabulky. V případě, že chceme zobrazit jen některé záznamy, použijeme ještě příkaz WHERE, kde v jazyce SQL slouží jako podmínka.



Obrázek 5 Příklad příkazu SELECT v jazyce SQL

Update (UPDATE) - Slouží k úpravě existujících záznamů v tabulce. Pro správné použití je nutné do příkazu zakomponovat podmínku, jinak se upraví všechny záznamy v tabulce.



Obrázek 6 Příklad příkazu UPDATE v jazyce SQL

Delete (DELETE) - Slouží k smazání existujících záznamů v tabulce. Stejně jako v případě UPDATE, i tady je nutné použít podmínku, jinak se smažou všechna data. Vyprázdnění celé tabulky bez jejího smazání obstará i příkaz TRUNCATE. [22]

Obrázek 7 Příklad příkazu DELETE v jazyce SQL

2 POPIS POUŽITÝCH TECHNOLOGIÍ

V této kapitole jsou popsány technologie využité při implementaci rezervačního systému sportovišť.

2.1 C#

Nebo taky C Sharp je objektově orientovaný a typově bezpečný programovací jazyk z rodiny jazyků C od společnosti Microsoft. Na rozdíl od staršího C, není třeba v tomto jazyku řešit úniky paměti, protože obsahuje automatickou správu paměti a uvolní tak úseky paměti, které program či proces nepoužívá. Dále mezi benefity jazyku patří nulovatelné typy proměnných, funkce výjimek nebo lambda výrazy. Programy a jejich zdrojové kódy v jazyce C# mohou být ukládány do více souborů, které na sebe mohou různě odkazovat. [23]

2.2 ASP.NET Core

Celý rezervační systém je postaven na platformě ASP.NET Core od Microsoftu. Tento framework umožňuje vytvářet webová uživatelská rozhraní v jazyce C# a lze tak zcela vynechat JavaScript. Celá platforma pracuje na architektuře klient-server, logiku aplikace řeší na straně serveru a poté výsledek zobrazí klientovi v podobě webové stránky za pomocí HTML a CSS. Dále pracuje se třemi hlavními prvky – Model, zobrazení (View) a kontroler (Controller). Aktuální verze je 6.0. [24]

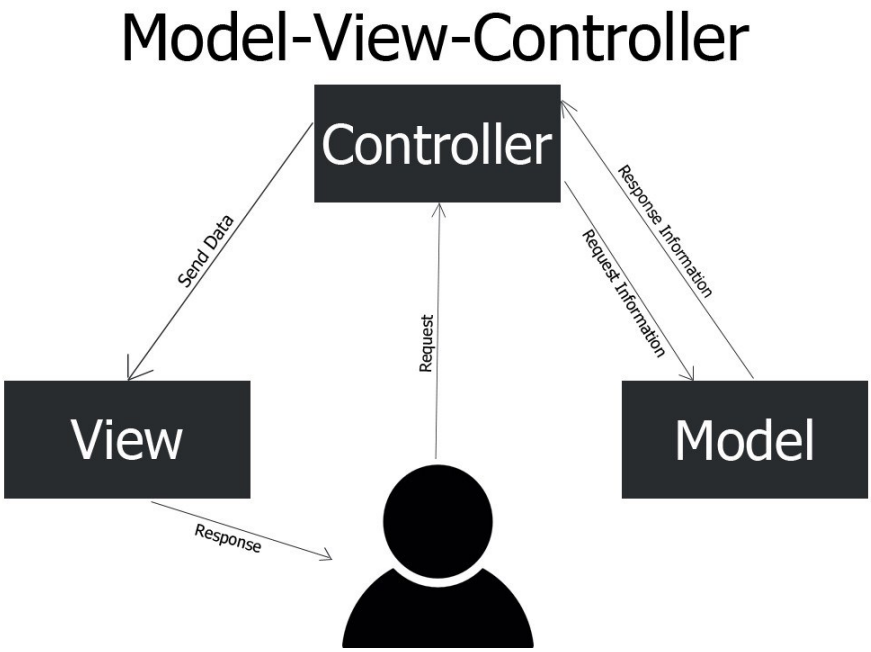
2.2.1 Architektura MVC

Model-View-Controller (MVC) architektura odděluje komponenty v aplikaci do třech nezávislých skupin. Každá z nich má svou specifickou roli a vzájemně jsou si propojeny. Aplikace s touto architekturou se pak stanou přehlednější.

Model představuje datovou strukturu, kde najdeme data, které se přenáší mezi kontrolerem a zobrazením. U ASP.NET data získaná z databáze se uchovávají v objektu modelu.

View (Zobrazení) v celé architektuře zodpovídají za zobrazení obsahu v uživatelském rozhraní. V ASP.NET tvoří zobrazení soubor s příponou .cshtml, který obsahuje HTML kód, do kterého lze vkládat syntaxi jazyka C#. Správně by se ale logiky v zobrazení mělo být ale co nejméně.

Kontroler zpracovává interakci uživatele, pracuje s modelem a vybírá se v něm zobrazení, která se má vykreslit. Řídí, jak aplikace reaguje na daný požadavek. Právě v této komponentě se nejvíce uplatní jazyk C#. [25]



Obrázek 8 Architektura MVC [26]

2.2.2 Entity Framework Core

Technologie Entity Framework Core slouží vývojářům jako objektově-relační mapovač a pracuje s databází za pomocí objektů, konkrétně modelů. Modely jsou tvořeny třídy, které reprezentují tabulky v databázi, a přistupujeme k nim za pomocí kontextu databáze. Daný kontext se může dotazovat na data a také je do tabulek ukládat. Použití EF Core nám usnadňuje práci s databází, jelikož není potřeba psát tolik kódu, a tak můžeme vytvořit databázi jen s využitím modelů a migrací. C# obsahuje dotazovací LINQ jazyk, který pracuje s lambda výrazy a dalšími metodami, které suplují dotazovací jazyk SQL. [27]

2.3 HTML, CSS a JavaScript

2.3.1 HTML

Pod zkratkou HTML se nachází hypertextový značkovací jazyk, kterým definujeme strukturu stránky a co má prohlížeč zobrazit uživateli. Tento jazyk používá značky, které značí určité elementy. Například <footer></footer> ohraničuje a označuje patičku dané stránky a

všechno, co se vloží doprostřed značky, se objeví uvnitř patičky. Značkám můžeme přidělovat identifikátory, kterými je lze odlišit od ostatních. Dnes se používá HTML5. [28]

2.3.2 CSS

Chceme-li změnit vzhled vytvořené HTML stránky, musíme použít kaskádové styly, které popisují, jak má prvek na obrazovce vypadat. CSS patří mezi základní jazyky otevřeného webu a najdeme ho tak v každém prohlížeči. Abychom mohli měnit vzhled prvku, musí mít element mít svůj identifikátor, přes který nastavujeme prvku jeho vlastnosti. Poslední verzí je CSS3. [29]

2.3.3 JavaScript/jQuery

JavaScript doplňuje HTML a CSS a používá se v prohlížečích na klientské straně jako skriptovací jazyk pro manipulaci s obsahem a konkrétními prvky. [30] Spolu s JavaScriptem je výhodné použít i jeho knihovnu jQuery, která zjednodušuje použití JavaScriptu v HTML stránce. Umožní vývojáři přidat animace nebo volání GET a POST akcí pomocí AJAXu. [31]

2.4 MySQL

Data v databázi v rezervačním systému obsluhuje MySQL, který je postaven na strukturovaném dotazovacím jazyce SQL, a jedná se o systém pro řízení báze dat, které tvoří rozhraní mezi programem a uloženými daty. Oproti systémům od Oracle nebo Microsoft má otevřený kód a představuje tak jejich alternativu. Pro vizualizaci databáze a tabulek používám nástroj MySQL Workbench. [32]

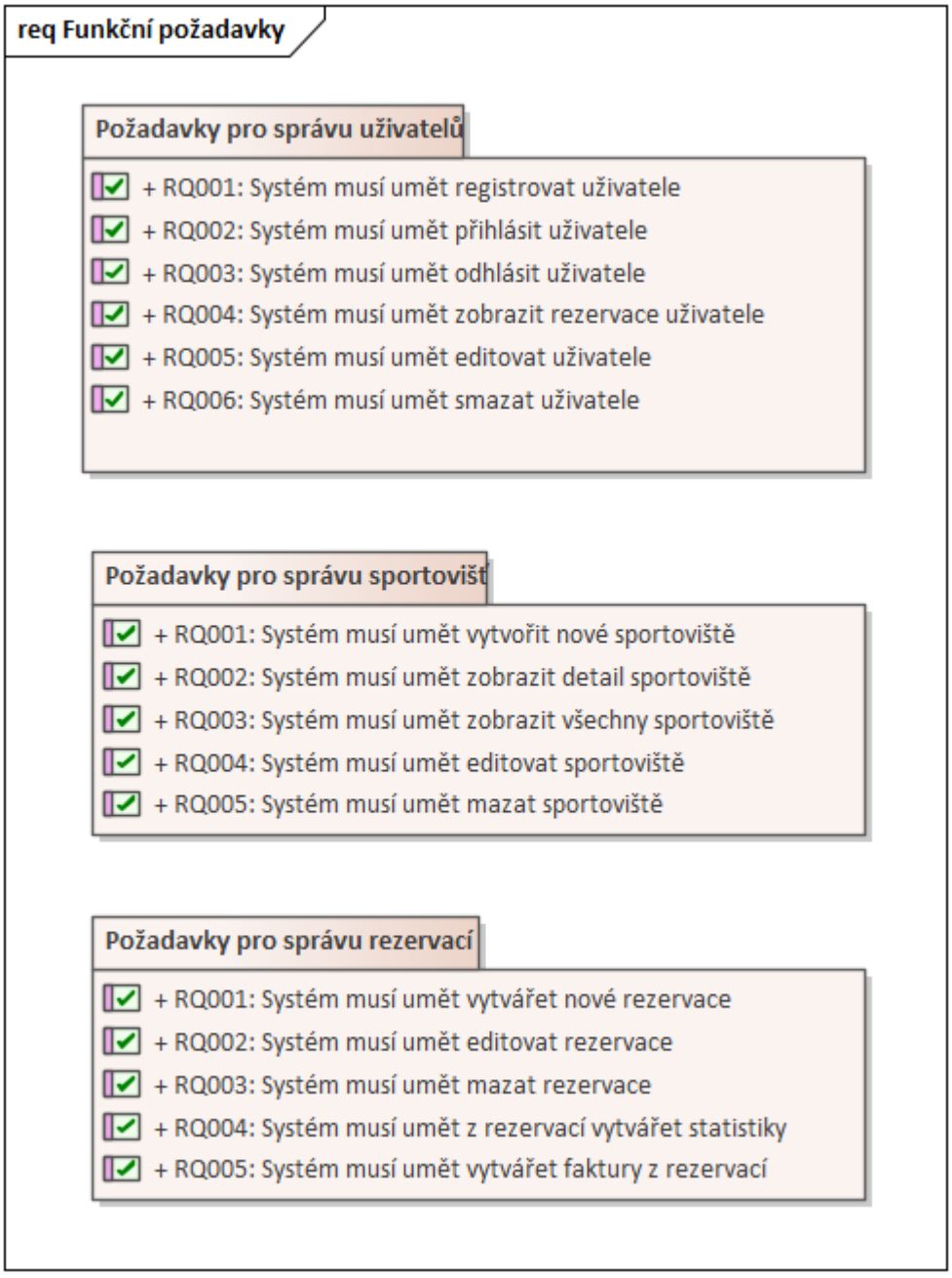
II. PRAKTICKÁ ČÁST

3 FUNKČNÍ ANALÝZA A ANALÝZA SYSTÉMU

Funkční analýza pojednává o požadavcích a využití diagramů vysvětlených v teoretické části práce na konkrétním zadání.

3.1 Funkční požadavky

Funkční požadavky jsem rozdělil do 3 částí, každá entita má své požadavky, které jsou sice podobné, ale na sobě nezávislé.



Obrázek 9 Funkční požadavky

Požadavky pro správu uživatelů:

- RQ001 - Registrace je důležitý prvek aplikace, bez které by nový uživatel neměl přístup k nejdůležitějším funkcím aplikace.
- RQ002 - Aby si zaregistrovaný uživatel mohl vytvořit rezervaci nebo si prohlédnout své rezervace, musí se nejdříve přihlásit a to systém musí umožňovat.
- RQ003 - Uživatel musí mít možnost se ze systému odhlásit.
- RQ004 - Požadavek zajišťuje, že si uživatel může prohlédnout své rezervace, které již má v systému.
- RQ005 - Systém administrátorovi umožní měnit údaje uživatele a pak změny ukládat v databázi.
- RQ006 - Smazaný uživatel administrátorem se smaže i se všemi jeho údaji včetně jeho rezervací.

Požadavky pro správu sportovišť:

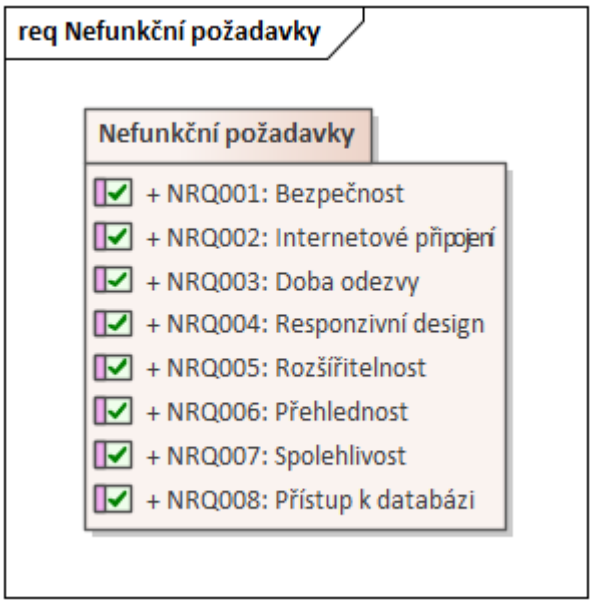
- RQ001 - V případě rozšíření služeb, administrátor má možnost vytvořit nové sportoviště.
- RQ002 - Slouží pro zobrazení popisu daného sportoviště na stránce, kde se uskutečňuje každá nová rezervace.
- RQ003 - Uživatel má možnost vidět všechny nabízené sportoviště a Administrátor má přístup do správy sportovišť.
- RQ004 - Administrátor v případě potřeby může změnit některé detaily sportoviště.
- RQ005 - Sportoviště je možné smazat, je-li to potřeba.

Požadavky pro správu rezervací:

- RQ001 - Nejdůležitější funkční podmínka, protože bez ní by systém nefungoval, jak by měl.
- RQ002 - Pomocí formuláře ve správci rezervací lze upravovat jednotlivé detaily rezervace.
- RQ003 - Mazat rezervace lze buď přímo, nebo když se smaže sportoviště či uživatel.

- RQ004 - Zobrazení jednotlivých statistik na základě rezervací.
- RQ005 - Uživatel v zobrazení svých formulářů nebo administrátor ve správci rezervací může vygenerovat fakturu ve formátu PDF.

3.2 Nefunkční požadavky



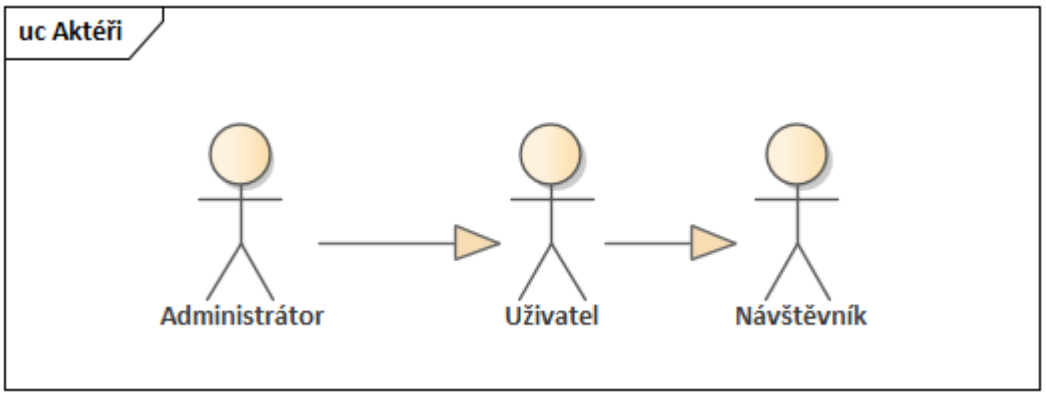
Obrázek 10 Nefunkční požadavky

Popis nefunkčních požadavků:

- NRQ001: Systém musí nabízet autentizaci uživatelů, přistupovat do různých částí na základě pravidel a rolí, hesla by měla být v bezpečném formátu a v databázi uložené zašifrované.
- NRQ002: Pro správné fungování celého systému je nutné připojení k internetu.
- NRQ003: Doba odezvy serveru by neměla být příliš dlouhá, jinak by to mohlo negativně ovlivnit uživatelský zážitek.
- NRQ004: Systém by měl být podporovat responzivní design aplikace na různých zařízeních či monitorech.
- NRQ005: Přidání nových elementů do systémů by nemělo výrazně ovlivnit zdrojový kód.
- NRQ006: I nový uživatel by se měl v aplikaci snadno zorientovat.
- NRQ007: Požadavek by měl zaručit, že systém nebude selhávat.
- NRQ008: Pro správné fungování celého systému je nutný přístup k databázi.

3.3 Aktéři

Systém a jejich případy užití mohou používat tři druhy aktérů. Návštěvník se stane každý, kdo přijde poprvé na webovou aplikaci a nemá vytvořený účet. Jakmile si vytvoří účet pomocí registrace, stane se uživatelem. Administrátor má práva předchozích dvou aktérů a zároveň má přístup do správy jednotlivých entit.



Obrázek 11 Aktéři

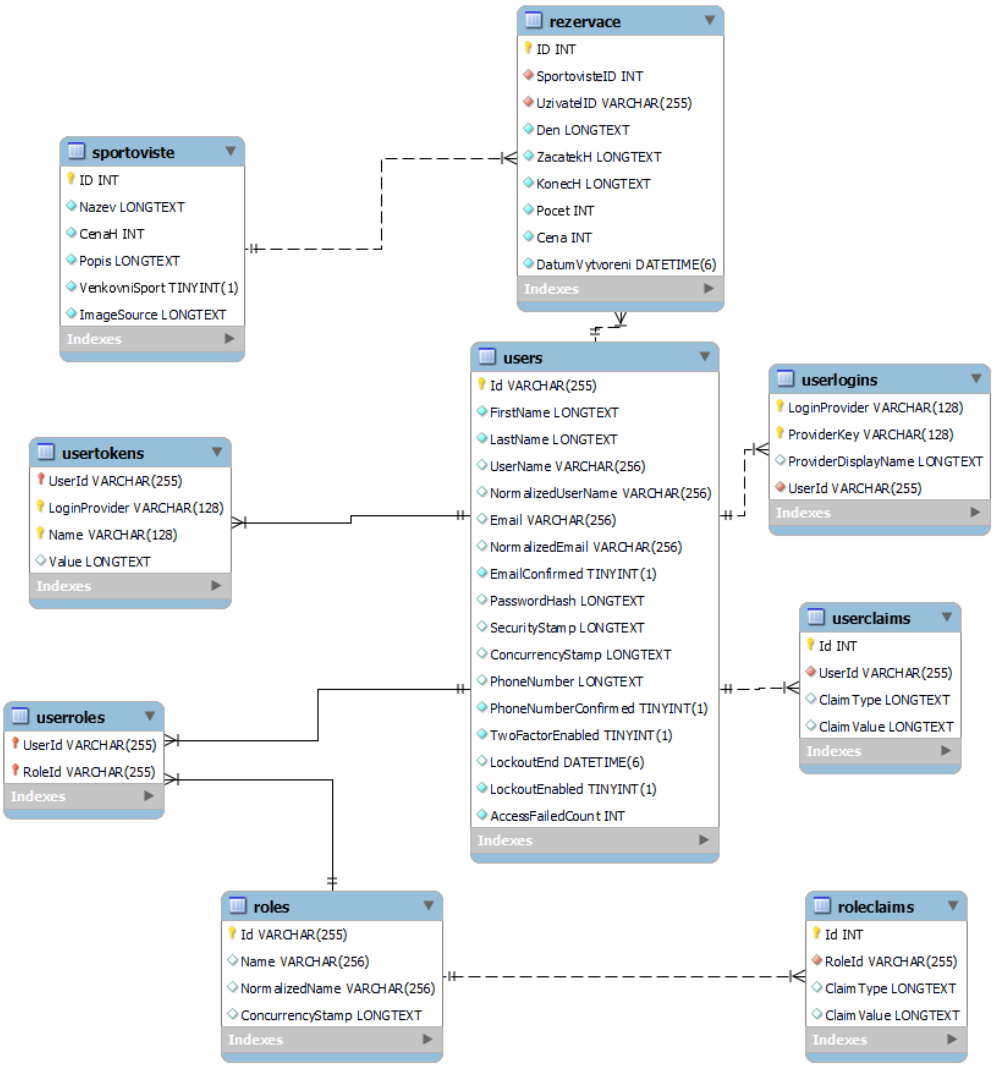
3.4 ER Diagram

ER Diagram vizuálně zobrazuje, jak jsou tabulky v databázi spolu s relacemi řešeny. V aplikaci je použitý systém ASP.NET Identity od Microsoftu, který vygeneroval tabulky users, userlogins, userclaims, roles, rolesclaims, userroles, usertokens. Zbylé tabulky (rezervace a sportoviště) jsou vytvořené pomocí modelů z architektury MVC a Entity Framework Core. Pokud je v počítači nainstalován SŘBD MySQL Workbench a spustí se program, vytvoří se automaticky všechny tabulky a k tomu vloží záznamy do tabulky sportoviště, které slouží pro výběr sportovišť, kde se dá rezervovat a taktéž účet administrátora. Popis jednotlivých tabulek:

- Users - Do této tabulky se zapisují jednotliví uživatelé a informace o nich (Křestní jméno, příjmení, Email a heslo). Tabulka obsahuje i další sloupce, které by se daly využít (telefonní číslo nebo jestli je zapnuto dvoufázové otevření).
- UserLogins - Tato obsahuje poskytovatele přihlášení a kromě použití emailu a hesla se lze přihlásit i přes sociální sítě. Jelikož v aplikaci není řešeno přihlášení přes Facebook a Google, tabulka je prázdná.

- UserTokens - Na základě předchozí tabulky by se tady měly ukládat tokeny uživatelů jedinečná hodnota k uživateli) používající přihlášení skrz sociální sítě, ale taktéž nic neobsahuje.
- UserClaims - Jelikož je aplikace postavená na autorizaci pomocí rolí, deklarace nejsou využity a tabulka zůstává nevyužita.
- Roles - Přihlášený uživatel může nabývat dvou rolí - Uživatel nebo Administrátor.
- RolesClaims - Stejný případ jako u UserClaims, tabulka neobsahuje žádná data.
- UserRoles - Tabulka je spojena s tabulkou rolí (Roles) a uživateli (Users) a obsahuje uživatele a jejich role.
- Sportoviště - Popisuje jednotlivá sportoviště, kde si může uživatel rezervovat termín. Je třeba vyplnit název, cena za hodinu, popis a logickou hodnotu, jestli je sportoviště venku nebo uvnitř a cestu k fotografii.
- Rezervace - Jakmile si uživatel na detailu sportoviště vybere požadovaný termín a vyplní formulář, uloží se celá rezervace do stejnojmenné tabulky. Skládá se ze sloupců SportovisteID (cizí klíč tabulky Sportoviště), UzivatelID (cizí klíč tabulky Users), požadovaný den, požadovanou hodinu začátku a konce, počet rezervovaných osob, cena rezervace a datum vytvoření její rezervace, což je hodnota, která se rovná času, kdy systém zapsal záznam do databáze.

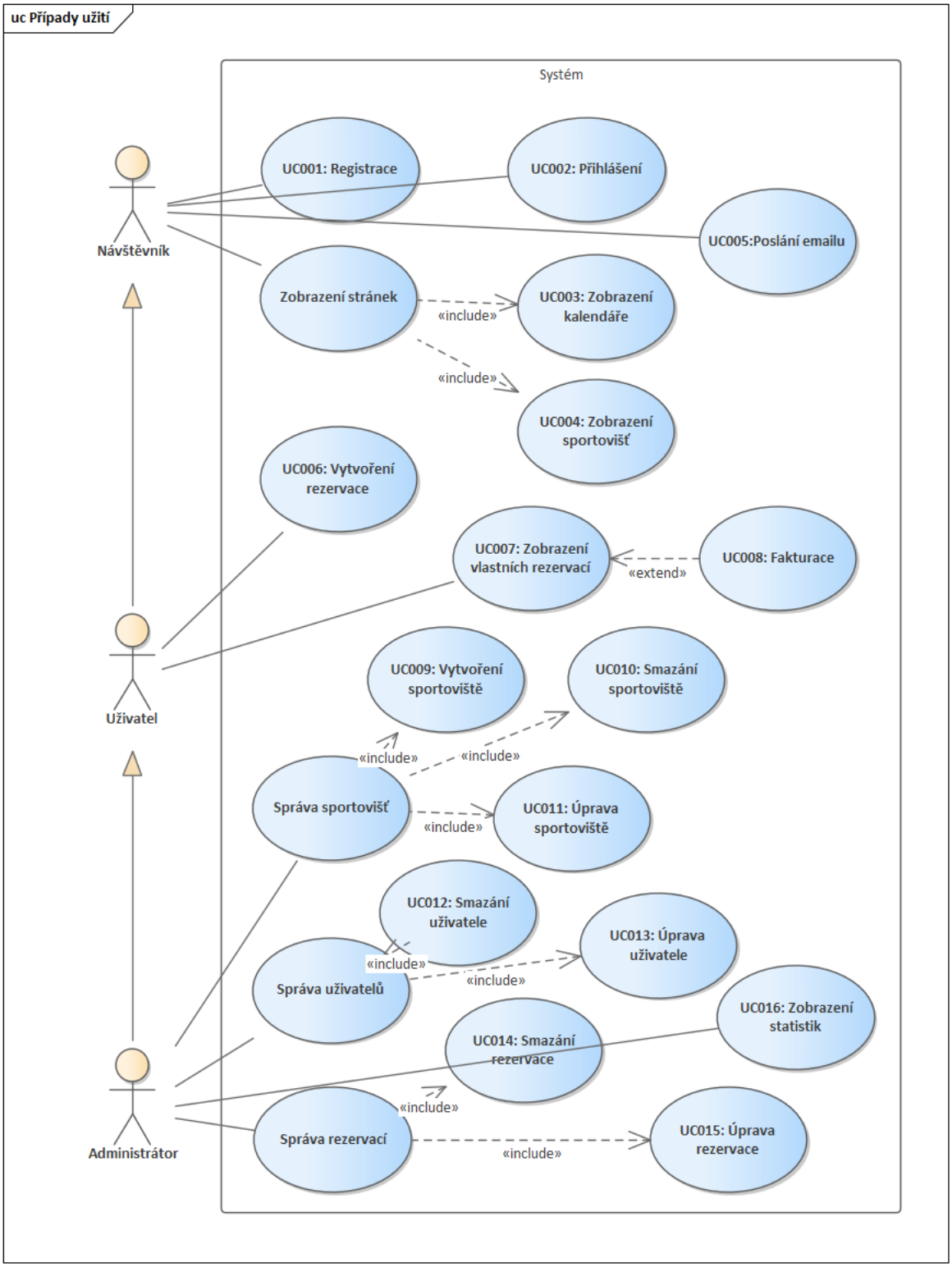
Pomocí čár mezi tabulky, jsou znázorněny jednotlivé kardinality. Přerušovaná nebo plná čára znamená vztah 1:1, přerušovaná nebo plná čára s rozvětvením vztah 1:N a oboustranně rozvětvená plná nebo přerušovaná čára značí vztah M:N, který se ale v databázi nevyskytuje.



Obrázek 12 ER Diagram

3.5 Případy užití

V modelu vidíme tři aktéry a každý z nich má své případy užití, které má možnost využít. Jednotlivé případy užití jsou specifikovány v další kapitole.



Obrázek 13 Diagram případů užití

3.6 Specifikace případů užití

Tabulky níže popisují dané případy užití a jejich scénáře. V práci je pár příkladů, všechny jsou obsažené v přiloženém Enterprise Architect projektu.

Tabulka 2 Specifikace případu užití – Vytvoření rezervace

<table>
<tr><td colspan="3">Název: Vytvoření rezervace</td></tr>
<tr><td colspan="3">ID: UC006</td></tr>
<tr><td colspan="3">Charakteristika:

Uživatel si chce vytvořit rezervaci na konkrétním sportovišti</td></tr>
<tr><td colspan="3">Primární aktér:

Uživatel</td></tr>
<tr><td colspan="3">Vedlejší aktéři: Nejsou</td></tr>
<tr><td colspan="3">Vstupní podmínky:

Uživatel je přihlášen a existuje alespoň 1 sportoviště</td></tr>
<tr><td colspan="3">Výstupní podmínky:

Vytvoří se v databázi nová rezervace a zobrazí se stránka Moje rezervace</td></tr>
<tr><td colspan="3">Hlavní scénář:</td></tr>
<tr><td>Krok</td><td>Aktér/Systém</td><td>Popis</td></tr>
<tr><td>1</td><td>Uživatel</td><td>Uživatel zmáčkne na domovské stránce u sportoviště na tlačítko Rezervovat</td></tr>
<tr><td>2</td><td>Systém</td><td>Systém zobrazí stránku s detaily sportoviště a kalendářem</td></tr>
<tr><td>3</td><td>Uživatel</td><td>Uživatel vybere v kalendáři požadované datum s časem začátku rezervace a vyplní celý formulář, který pak odešle</td></tr>
</table>

4	Systém	Systém zapíše do databáze novou rezervaci a přesměruje na stránku Moje rezervace

Tabulka 3 Specifikace případu užití – Zobrazení vlastních rezervací

Název: Zobrazení vlastních rezervací		
ID: UC007		
Charakteristika: Uživatel si chce zobrazit své vlastní rezervace		
Primární aktér: Uživatel		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen a má nějakou rezervaci		
Výstupní podmínky: Zobrazí stránku s tabulkou obsahující rezervace uživatele		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Uživatel zmáčkne tlačítko Moje rezervace v navigační liště
2	Systém	Systém zobrazí stránku s tabulkou rezervací
Alternativní scénáře: UC007a – Uživatel nemá žádnou rezervaci		

Tabulka 4 Specifikace případu užití – Alternativní scénář: Uživatel nemá žádnou rezervaci

Název – Alternativní scénář: Uživatel nemá žádnou rezervaci		
ID: UC007a		
Charakteristika: Uživatel si chce zobrazit své rezervace, ale žádnou nemá		
Alternativní scénář:		
Krok	Aktér/Sys-tém	Popis
1	Uživatel	Uživatel zmáčkne tlačítko Moje rezervace v navigační liště
2	Systém	Systém zobrazí stránku s hláškou, že zatím nemá žádnou rezervaci

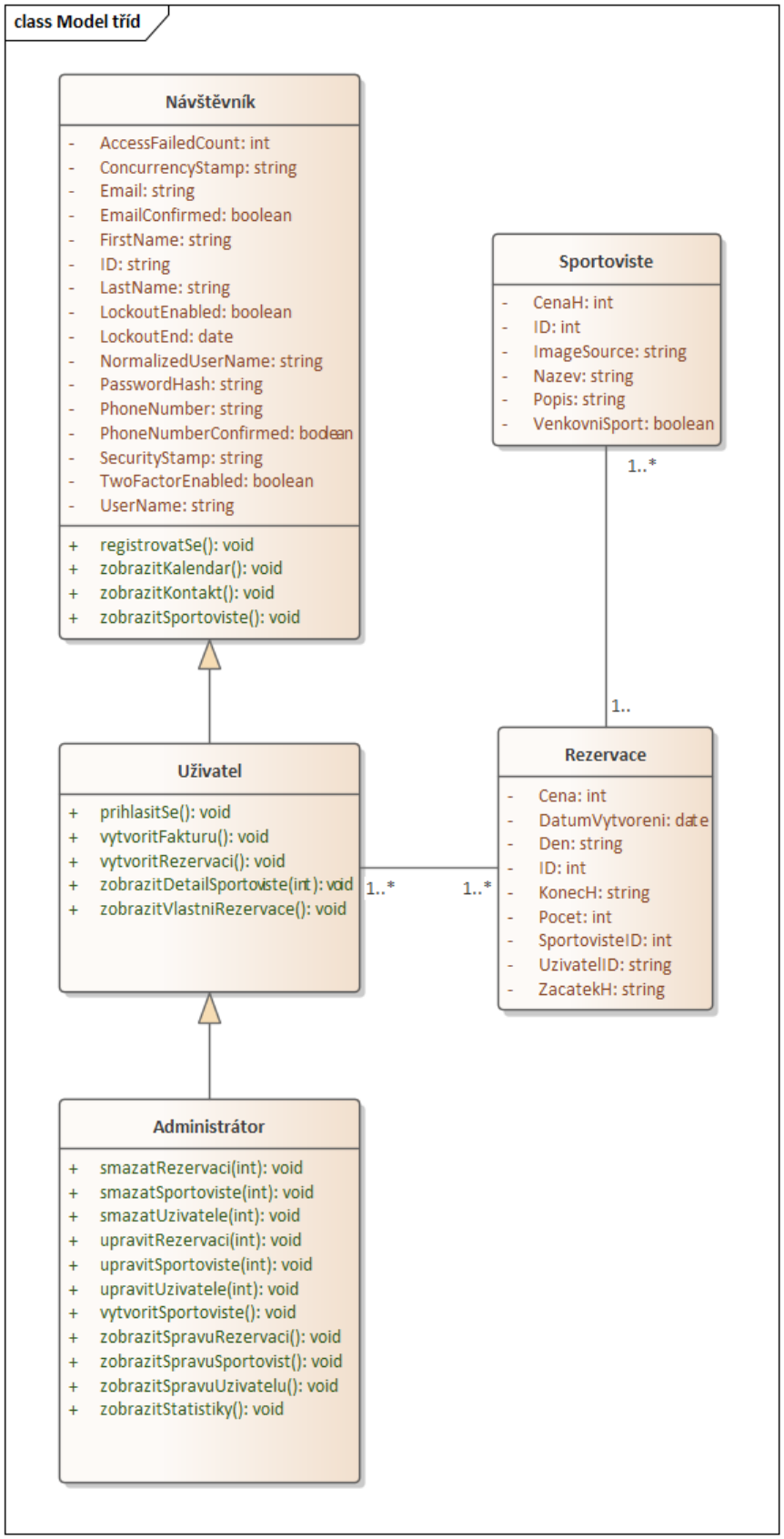
Tabulka 5 Specifikace případu užití – Smazání rezervace

Název: Smazání rezervace
ID: UC014
Charakteristika: Administrátor chce smazat rezervaci
Primární aktér: Administrátor
Vedlejší aktéři: Nejsou
Vstupní podmínky: Administrátor má roli Admin, je přihlášen a existuje v databázi nějaká rezervace

Výstupní podmínky: Rezervace se smaže z databáze		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor v navigační liště klikne na tlačítko Správa rezervací
2	Systém	Systém zobrazí stránku s tabulkou rezervací
3	Administrátor	Administrátor zmáčkne u vybrané rezervace tlačítko Smazat
4	Systém	Systém zobrazí okno, jestli si opravdu přeje smazat rezervaci
5	Administrátor	Pokud zmáčkne ano, systém smaže rezervaci z databáze, pokud ne, tak se nic nestane
6	Systém	Systém zobrazí stránku Správa rezervaci

3.7 Diagram tříd

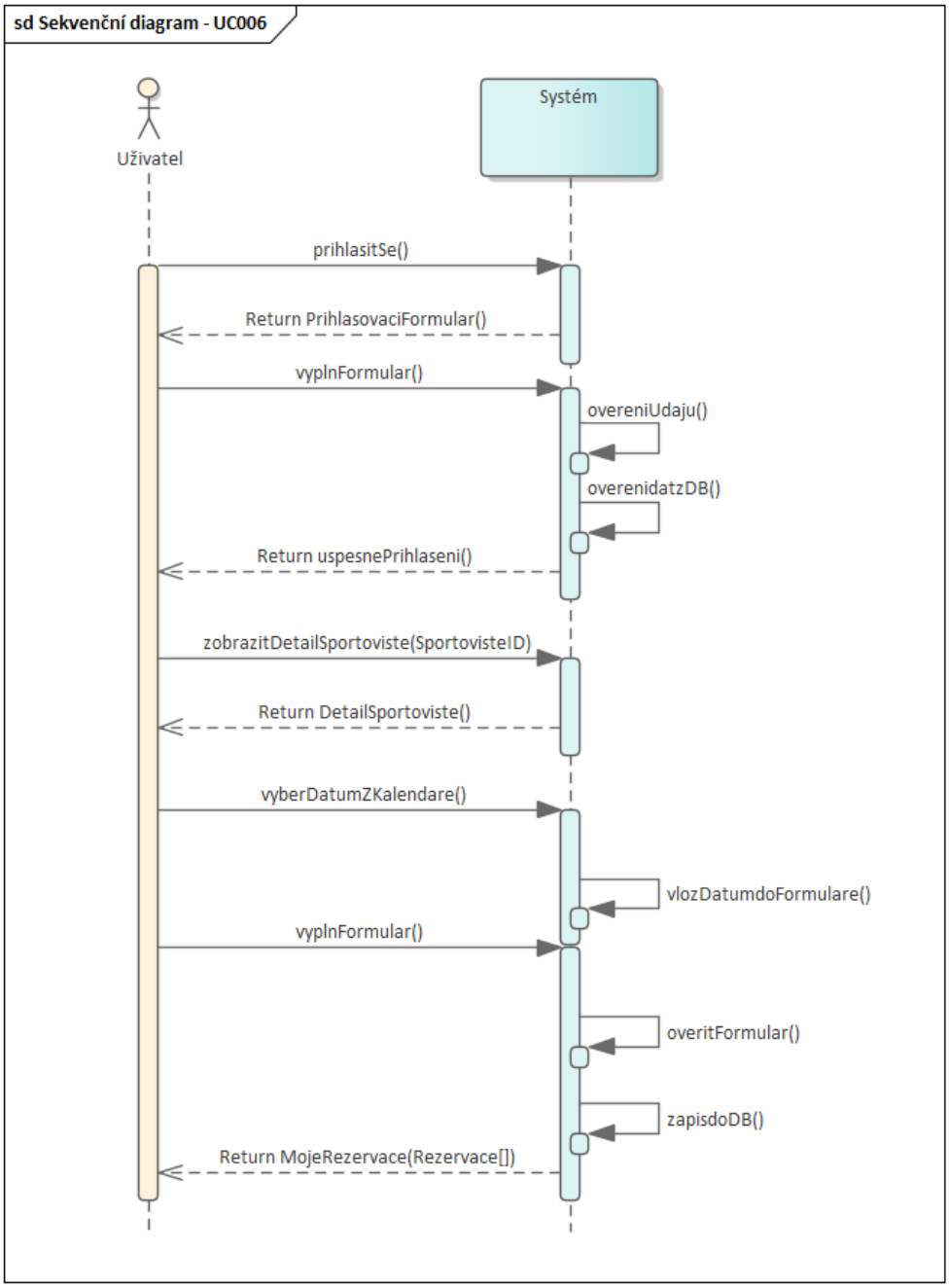
Diagram tříd se skládá ze tří aktérů, kteří reprezentují tabulku uživatelů a používá tak jejich sloupce s datovými typy. Každý aktér má své unikátní funkce a skrze vztah globalizace si vlastnosti a funkce od podřazeného aktéra přebírají. Tito tři aktéři mohou pracovat s třídami sportoviště a rezervace.



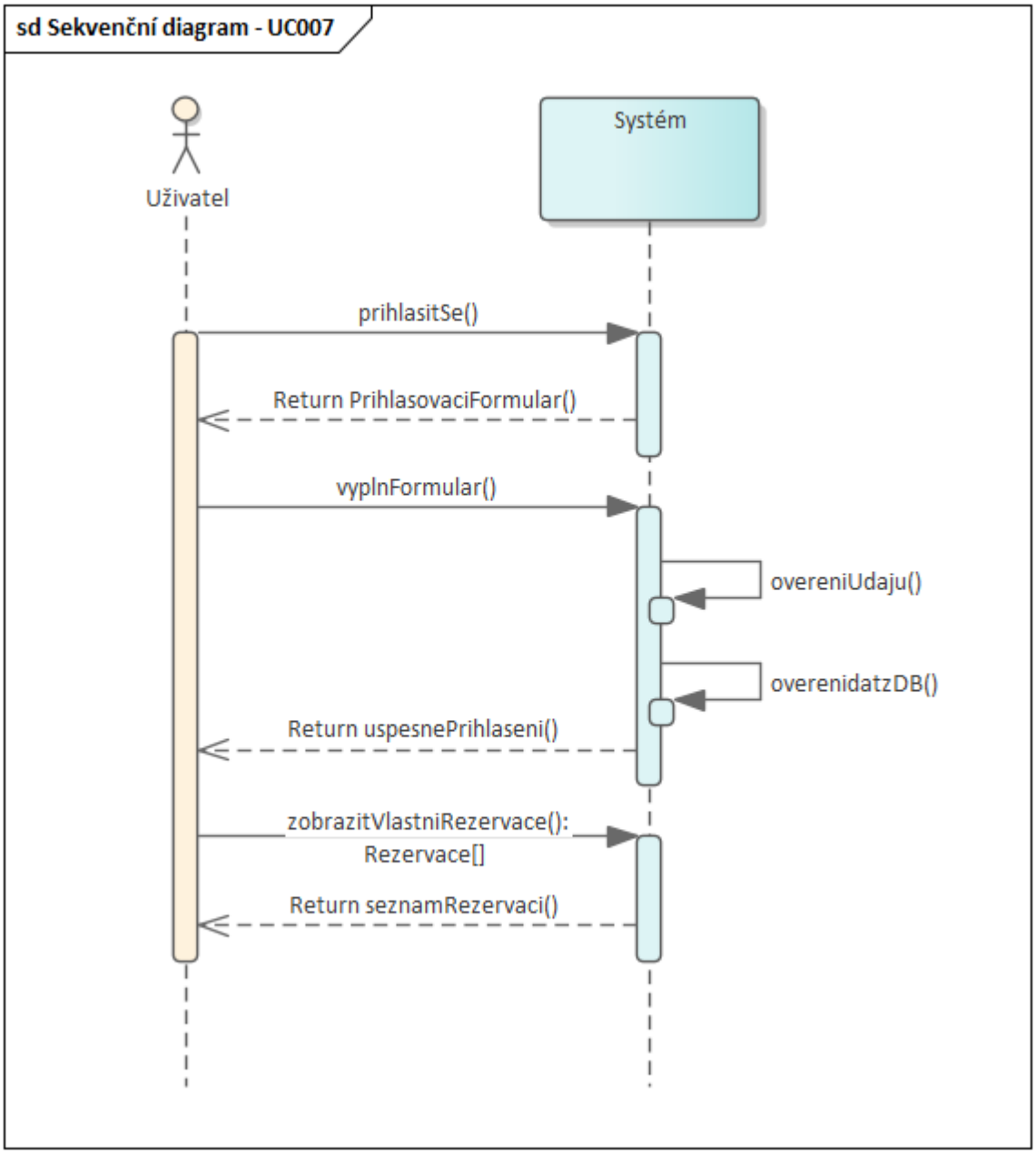
Obrázek 14 Model tříd

3.8 Sekvenční diagramy

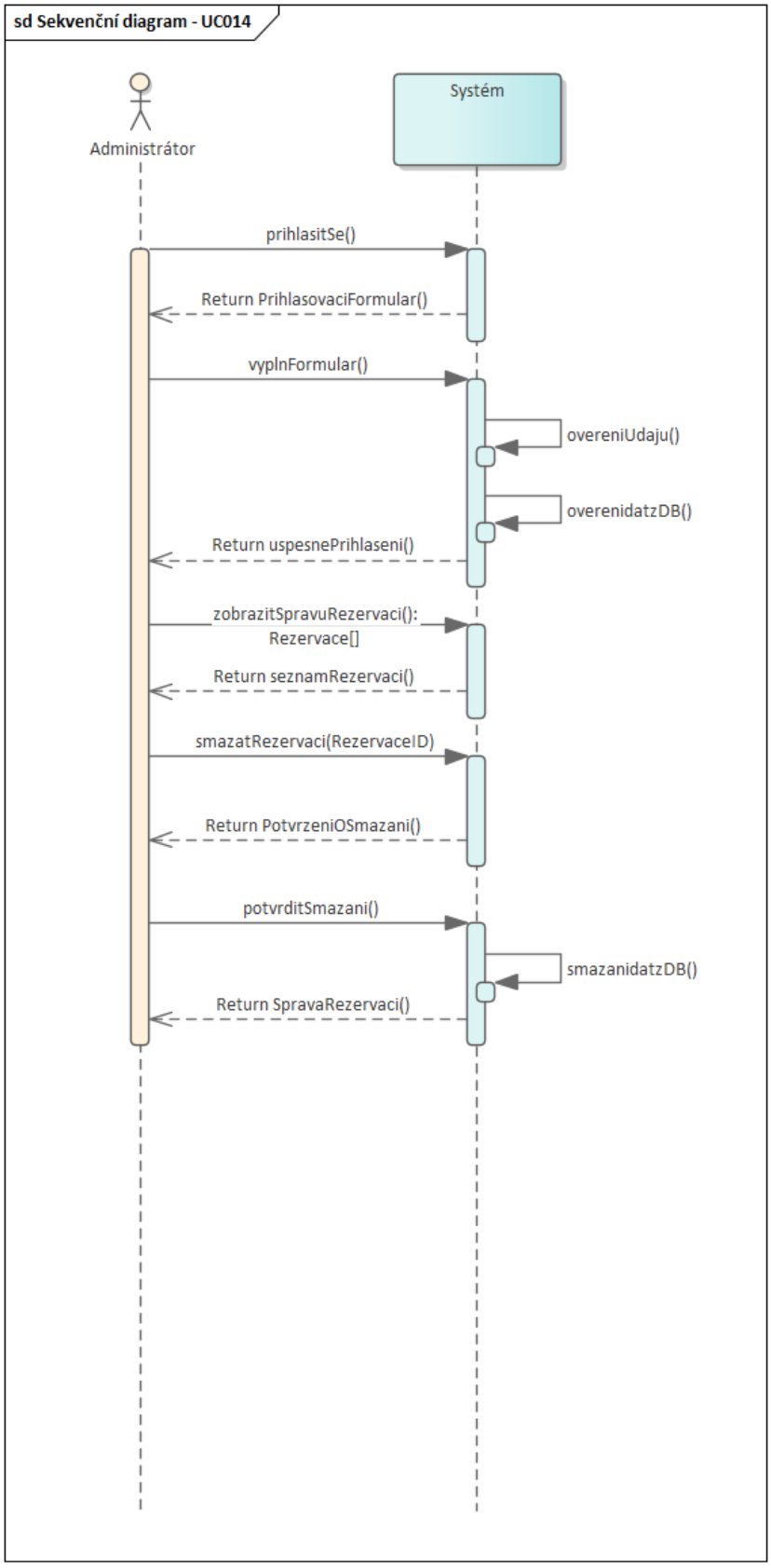
Pro každý případ užití jsou vytvořeny sekvenční diagramy, které popisují, jak daný případ užití sekvenčně pracuje. Součástí diagramu jsou aktéři (Návštěvník, Uživatel, Administrátor) a Systém, kterým se rozumí samotná aplikace spolu s databází. Jednotlivé diagramy ukazují případy užití, které byly popsány v kapitole specifikace případů užití. Zbylé sekvenční diagramy obsahuje přiložený Enterprise Architect.



Obrázek 15 Sekvenční diagram – Vytvoření rezervace



Obrázek 16 Sekvenční diagram – Zobrazení vlastních rezervací



Obrázek 17 Sekvenční diagram Smazání rezervace

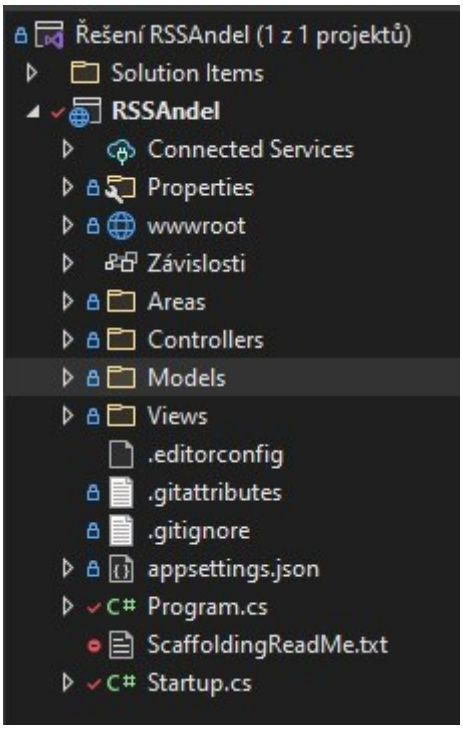
4 IMPLEMENTACE

Webová aplikace je vytvořena ve vývojovém prostředí Microsoft Visual Studio 2022 na platformě ASP.NET Core 6.0 v jazyce C#.

4.1 Souborová struktura aplikace

Popis důležitých složek a souborů v kořenovém adresáři:

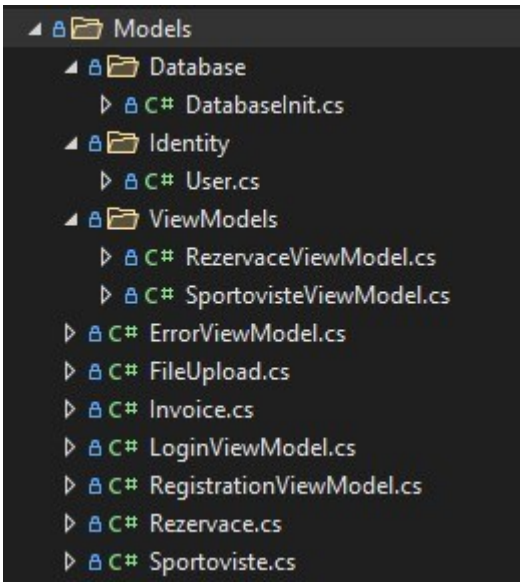
- wwwroot - Obsahuje ikonu projektu, podsložky využívající prohlížeč pro vzhled a chování stránky (obrázky, kaskádové styly, skripty) a také se tady ukládá faktura jednotlivých rezervací.
- Areas - V překladu oblasti a uchovávají se tam kontrolery a pohledy jednotlivých oblastí.
- Controllers, Models a Views - V těchto složkách se nachází komponenty architektury MVC a každá má svou vlastní funkci.
- Program.cs a Startup.cs - Jedná se o vygenerované soubory, kde kódy se vykonají během spuštění projektu. Řeší se tam připojení aplikace k databázi, definování autentizace metody Identity od Microsoftu a dalších služby, které chceme do projektu zakomponovat.



Obrázek 18 Souborová struktura aplikace

4.2 Modely

Modely v programování znázorňují pomocí tříd datové struktury, se kterými můžeme po inicializaci pracovat. V ASP.NET je možné využít modely jako předpis tabulky a pomocí Entity Framework Core a migrací vytvořit tabulky bez využití jazyka SQL nebo práce přímo v databázovém systému.



Obrázek 19 Složka Models

Pro účely testování a fungování aplikace je nutné mít při startu v databázi již nějaké záznamy. To má na starost třída DatabaseInit, která se spouští během začátku programu a zajišťuje, že v tabulkách Rezervace, Sportoviště, Uživatelé a Role jsou zapsány potřebné data.

Modul ASP.NET Identity implementuje do systému autorizaci s potřebnými tabulkami a User.cs dědí vlastnosti původního modelu a dodává dva nové sloupce s názvy Křestní jméno a příjmení.

Soubor FileUpload.cs nereprezentuje tabulku databáze, ale řeší funkcionalitu vkládání fotek do systému. Využití najde uživatel, konkrétně administátor při vytváření či editaci sportoviště.

Rezervace a Sportoviště definují pomocí datových typů a klíčů předpisy stejnojmenných tabulek v databázi.

```
namespace RSSAndel.Models
{
    [Table(nameof(Sportoviste))]
    Počet odkazů: 42
    public class Sportoviste
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Computed)]
        Počet odkazů: 21
        public int ID { get; set; }

        [Required]
        Počet odkazů: 24
        public string Nazev { get; set; }

        [Required]
        Počet odkazů: 15
        public int CenaH { get; set; }

        [Required]
        Počet odkazů: 14
        public string Popis { get; set; }
        [Required]
        Počet odkazů: 11
        public bool VenkovniSport { get; set; }

        [NotMapped]
        Počet odkazů: 9
        public IFormFile Image { get; set; }

        Počet odkazů: 15
        public string ImageSource { get; set; }
    }
}
```

Obrázek 20 Kód třídy Sportoviště.cs

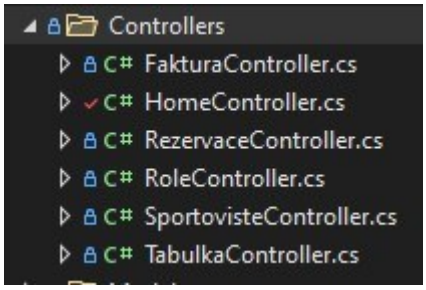
4.2.1 Viewmodely

ASP.NET MVC platforma neumožňuje v jednom pohledu inicializovat více modelů najednou a vyřešit se to dá prostřednictvím viewmodelů, které obsahují dvě a více inicializací objektů jednotlivých modelů.

RezervaceViewModel a SportovisteViewModel obstarávají inicializace potřebných tříd nebo jejich kolekce pro zobrazení všech rezervací či sportovišť.

ErrorViewModel, LoginViewModel a RegistrationViewModel vygeneroval program pro potřeby registrace, přihlášení a zobrazení chybové hlášky aplikace. Autorizační viewmodely se skládají z proměnných použitých ve formulářích. Každé proměnná se nastavují atributy jako je například datový typ nebo délka řetězce.

4.3 Kontrolery



Obrázek 21 Složka Controllers

O vygenerování faktury se stará FakturaController a metoda GenerujPDF, kde se nejprve získá data vybrané rezervace a knihovna PdfPig vytvoří fakturu ve formátu PDF.

V HomeControlleru se nacházejí akční metody, jejichž výsledkem je zobrazení požadované stránky, v případě potřeby i s viewmodelem pro zobrazení na stránce. Kromě toho kontroler definuje metodu GetCalendarInformation, díky které má kalendář záznamy z databáze. Při zobrazení stránky jQuery AJAXem zavolá zmíněnou metodu, která do kolekce uloží všechny rezervace. Z kolekce se vytvoří JSON objekt s názvem calendarEvents, který se jako pole vrátí zpátky do pohledu.

Vytvoření rezervace probíhá v kontroleru RezervaceController. Tato metoda navazuje na odeslání formuláře na stránce Detail a formulářem se do parametru přenese model Rezervace. V kódu se nejprve zjistí email přihlášeného uživatele, aby se k němu mohla rezervace přiřadit, a pak se do hodnoty Datum vytvoření přidá aktuální den a čas. Kontextem spolu s instancí modelu se uloží do tabulky a posléze se potvrdí změny v databázi. Na závěr funkce se z databáze do proměnné uloží všechny rezervace daného uživatele a spolu s ní metoda přesměřuje uživatele na stránku jeho rezervací, kde již vidí všechny své rezervace včetně té poslední vytvořené.

Logika zobrazení stránky Detail daného sportoviště je vyřešena v SportovisteController. Metoda detail dodává pohledu viewmodel Rezervace a GetCalendarInformation funguje úplně stejně jako v HomeControlleru.

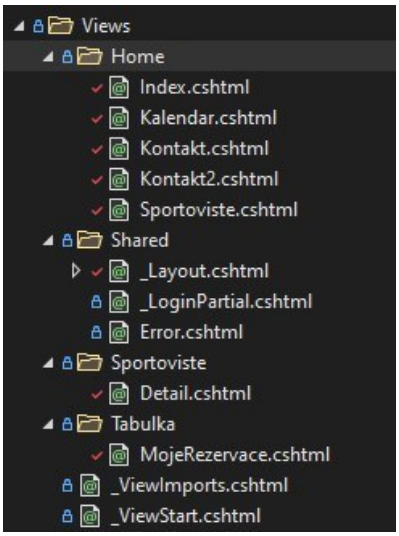
TabulkaController má jedinou funkci a to získání rezervací konkrétního uživatele a vrácení zpátky do pohledu.

```
Počet odkazů: 0
public IActionResult Select()
{
    IList<Rezervace> rezervace = _rssDbContext.Rezervace.ToList();

    return View();
}


[HttpPost]
Počet odkazů: 0
public async Task<IActionResult> Create(Rezervace rezervace)
{

        var user = await _userManager.FindByEmailAsync(User.Identity.Name);
        rezervace.UzivatelID = user.Id;
        rezervace.DatumVytvoreni = DateTime.Now;

        _rssDbContext.Rezervace.Add(rezervace);

        _rssDbContext.SaveChanges();

    var r = _rssDbContext.Rezervace.Include(s => s.Sportoviste).Include(u => u.Uzivatel).Where(
    s => s.Uzivatel.Id == user.Id);
    return View("~/Views/Tabulka/MojeRezervace.cshtml", await r.ToListAsync());
}
```

Obrázek 22 Kód pro vytvoření zobrazení a vytváření rezervací

4.4 Pohledy

Pohledem se rozumí soubor ve formátu cshtml a jeho kód definuje, jak se stránka uživateli v prohlížeči zobrazí. Oproti html souborům dovolují do kódu zahrnout kód programovacího jazyka. Jednotlivé jména složek odkazují na jména použitých kontrolerů.

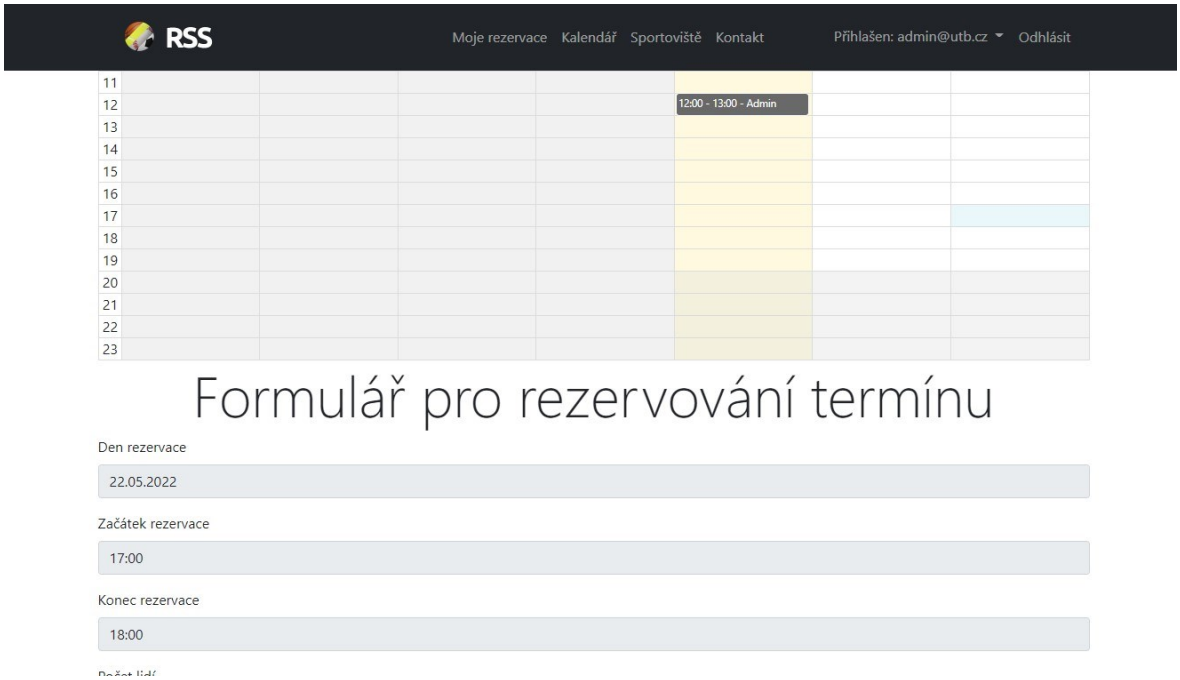


Obrázek 23 Složka Views

Vývojové prostředí při vytvoření projektu vygenerovalo složku Shared, kde se nacházejí soubory s funkcí šablon, které lze využít v celém projektu. Šablona _Layout je základem každé stránky.

Nepřihlášený uživatel může navštívit jen ty stránky, které se nacházejí ve složce Home. Index zobrazuje úvodní stránku aplikace a načte se jako první při zapnutí. Dále může vidět kalendář všech rezervací, přehled všech sportovišť a kontaktní formulář na správce webu

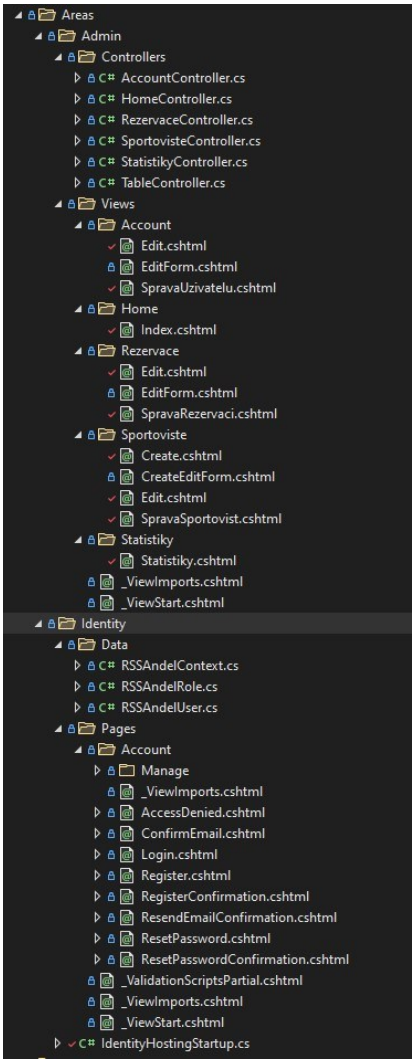
Přihlášený uživatel dál krom výše zmíněných stránek má přístup k detailu sportoviště, kde se mu zobrazí jeho vlastnosti s možností rezervovat si termín, a k tabulce svých rezervací. Při vývoji kalendáře byla použita javascriptová knihovna FullCalendar.



Obrázek 24 Kalendář a formulář pro rezervování termínu

4.5 Oblasti

Oblastmi lze zpřehlednit a rozdělit soubory na více menších částí. V případě tohoto rezervačního systému oblasti oddělují soubory administrátorské části od uživatelské. Kódy všech souborů potřebné pro stránky administrátora najdeme v této oblasti.



Obrázek 25 Složka Areas

Administrátor má oproti uživateli právo vidět tabulky všech rezervací, sportovišť a uživatelů. Oproti uživatelským kontrolerům se stejným názvem disponují metodami pro vytvoření, úpravu a smazání jednotlivých záznamů z databáze. Kromě toho si může zobrazit statistiky v grafech na základě dat z databáze a stejně jako u kalendářů, i zde se volá AJAXem metoda kontroleru, která vrací JSON objekt.

V pohledu každého kontroleru najdeme stránky s tabulkami všech záznamů z dané entity a editovací formulář. Statistiky v grafech vykresluje javascriptová knihovna C3.

Identitní oblast je vygenerována službou ASP.NET Identity a řeší nejen autorizaci a autentizaci uživatelů, nýbrž i připojení k databázi (RSSAndelContext). Tato oblast disponuje i vlastní třídou, která se vykoná při spuštění a nastavuje se zde autorizační pravidla.

4.6 Zabezpečení aplikace

Aplikace byla opatřena několika prvky zabezpečení. Pro konkrétní funkce se musí uživatel přihlásit a to obstarává vestavěný systém ASP.NET Identity, který dovoluje vygenerovat všechny potřebné prvky nejen pro autorizaci, ale taky změnu hesla nebo potvrzení emailu.

Hesla uživatelů by nikdy nikdo neměl vidět v textové podobě, a tak v databázi je uložen zašifrovaný řetězec (hash). Šifrování probíhá během asynchronní metody CreateAsync, jenž je součástí uživatelského manažera (userManager).

```
await _userManager.CreateAsync(user, Input.Password);
```

Obrázek 26 Příkaz pro registraci a zašifrování hesla

Dále se rozlišují role mezi uživateli a přístup na některé stránky je dovolen jen uživateli s určitou rolí. Politika rolí se nastavuje IdentityHostingStartup.cs a každá politika vyžaduje určitou roli. Atributem Authorize s názvem politiky nad názvem kontroleru se omezují stránky jen pro vybrané uživatele. Atribut Authorize bez dalších názvů říká, že uživatel musí být přihlášený, aby mohl danou stránku navštívit, a přesměruje ho to na přihlašovací formulář.

```
[Authorize(Policy = "admin")]
Počet odkazů: 3
public class RezervaceController : Controller
```

Obrázek 27 Politika role Admin nad kontrolerem

I databáze potřebuje předejít neodpovídajícím záznamům a to během vkládání hodnot do formuláře na klientské části. V HTML5 elementy formuláře jako vstupy nabízejí různé verifikační atributy, kterými lze donutit uživatele vyplnit pole nebo vyplnit pole podle formátu vstupu. Složitější ověřování probíhá v JavaScriptu. Pokud formulář nesplňuje všechny požadovaná pravidla, záznamy se do databáze nezapíšou.

Obrázek 28 Formulář, který nebyl správně vyplněn

5 TESTOVÁNÍ

Hlavní roli v testování webové aplikace hrály manuální testy v klientské části, které se prováděly během vývoje a po něm a spočívaly v tom, že se pomocí "proklikávání" elementů ve spuštěném prohlížeči testovalo, zda segment kódu pracuje podle požadavků aplikace a autora. Nejvíce času trvalo otestovat funkce javascriptových knihoven, jelikož během implementace se u nich objevovalo spoustu chyb, jako například nedodané data z kontroleru do jQuery pomocí AJAXu.

V serverové části se testovalo a hledalo chyby v kontrolerech a databázových operací pomocí debuggeru (ladící program) ve vývojovém prostředí, kdy se nastavil breakpoint (zarážka) a program se zastavil na místě zarážky a poskytl všechny dostupné proměnné a jejich hodnoty. V případě testování správného zapsání databáze se využil databázový systém MySQL Workbench, který nabízí grafické zobrazení záznamů v jednotlivých tabulkách. Pokud se pokusíme otestovat funkčnost vytvoření rezervace se zápisem do databáze a ověříme si posléze tabulku v databázovém systému, tak vidíme, že se přidal do tabulky nový řádek a jako kontrola může posloužit sloupec Datum vytvoření, který má datový typ Datetime a jeho hodnota se rovná aktuálnímu času v době zápisu do databáze.

	ID	SportovisteID	UzivatelID	Den	ZacatekH	KonecH	Pocet	Cena	DatumVytvoreni
▶	1	2	8e445865-a24d-4543-a6c6-9443d048cdb9	18.05.2022	15:00	16:00	3	120	2022-05-18 16:36:10.547803

Obrázek 29 Záznam v databázi

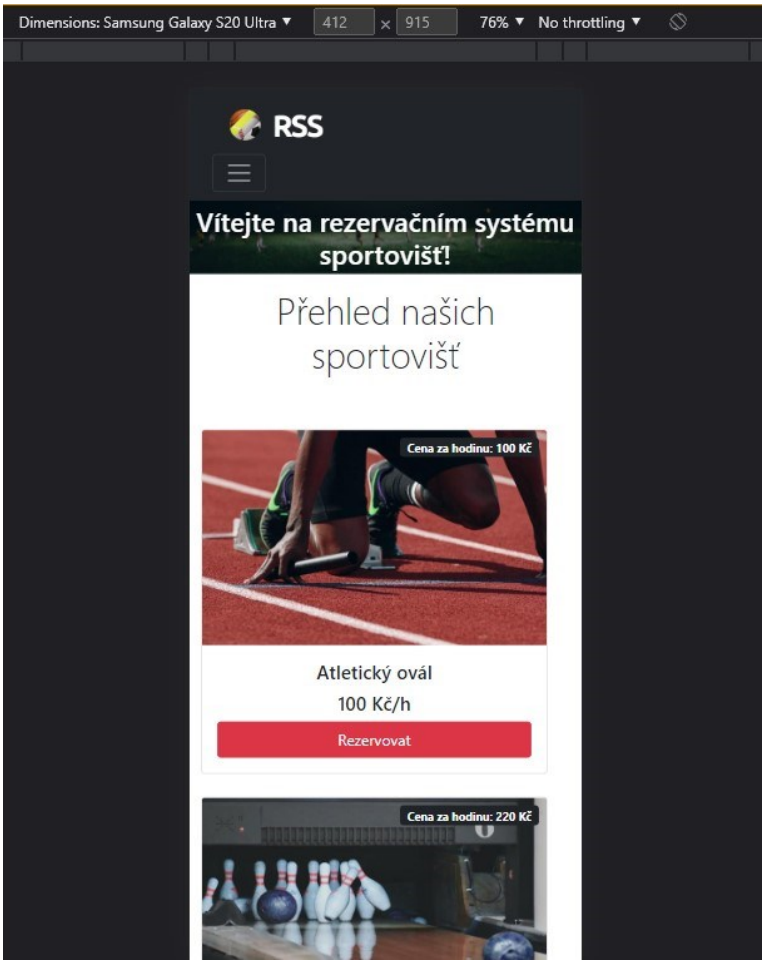
Kromě funkčnosti je ideální otestovat také dobu odezvy aplikace v prohlížeči, jelikož příliš dlouhé načítání stránky může odradit uživatele od používání aplikace. K tomu posloužilo rozšíření do Google Chrome Page load time, které stopuje čas při načítání konkrétní stránky a zastaví se, jakmile se stránka plně načte. Z výsledků vyplynulo, že všechny stránky se načtou zhruba do 2 až 2,5 vteřin. Jedinou výjimkou je úvodní stránka po zapnutí projektu na lokálním serveru (localhost), kdy se průměrné časy pohybují kolem 25 vteřin, poté se ale už další stránky načítají mnohem rychleji.

Event	Start	Duration	End
Redirect	0	0	0
DNS	10	0	10
Connect	10	110	120
Request	120	44	164
Response	164	23	187
DOM	187	1622	1809
Parse	187	532	719
Execute Scripts	719	1	720
Content loaded	720	10	730
Sub Resources	730	1079	1809
Load event	1809	34	1843
Total			**1843**

Timings are based on Navigation Timing Level 2 Spec

Obrázek 30 Časy načítání stránky v ms pomocí Page load time

V dnešní době uživatelé nepoužívají k prohlížením webových stránek počítače, ale také mobilní telefony, které mají různé rozlišení displejů. Responzivitu zajišťuje framework Bootstrap a testování proběhlo přímo v prohlížeči, kde se nacházejí vývojářské nástroje dovolující simulovat různé obrazovky mobilních zařízení. Na větších obrazovkách vzhled vypadá normálně a neobjevují se žádné velké designové chyby, které by ovlivňovaly funkčnost aplikace. Na menších displejích jsou některé elementy hůře čitelné, nicméně to na funkčnosti nic neubírá.



Obrázek 31 Simulace obrazovky Samsungu Galaxy S20

Poslední fází testování bylo vyzkoušet aplikaci napříč různými prohlížeči. Ve 4 prohlížečích (Vivaldi, Google Chrome, Mozilla Firefox a Microsoft Edge) se otestovaly kromě vzhledu 4 funkce - registrace, přihlášení, vytvoření rezervace a automatické odhlášení, pokud je uživatel 5 minut nečinný. Všechny zmíněné prohlížeče provedly první tři funkce úplně stejně, lišil se až výsledek automatického odhlášení v nečinnosti. Během přihlášení se do prohlížeče uloží cookies o uživateli a pokud 5 minut neprovede žádnou akci, prohlížeč automaticky cookies smaže a uživatele odhlásí. Všude kromě Google Chrome tato funkcionalita pracuje správně a uživatele odhlásí po uplynulé době, prohlížeč od Googlu z neznámého důvodu nedovolí cookies smazat a uživatele odhlásit.

ZÁVĚR

Bakalářská práce měla za cíl navrhnout a implementovat rezervačních systém sportovišť.

V teoretické části jsem popsal modelovací jazyk UML a jeho diagramy, které později byly použity, dále rozebral databáze a související věci téměř do detailů a charakterizoval použité technologie během vývoje aplikace a architekturu MVC, která posloužila jako základ celého systému.

Na základě analýz, modelů a diagramů jsem vytvořil webovou aplikaci, která umí vytvářet rezervaci, sportoviště a uživatele a posléze je zobrazit, mazat či modifikovat. Z výsledků testů lze vyvodit závěr, že se aplikace dá považovat za zdařilou, i když ji tíží několik systémových omezení, které ale nikterak vážně neovlivňují funkčnost či vzhled aplikace. Pro případné nasazení do ostrého provozu by aplikace potřebovala notnou dávku úprav, ale i za současného je použitelná.

V práci jsem využil znalostí v oblasti UML a programování nabité během studia a spousta nových jsem získal.

SEZNAM POUŽITÉ LITERATURY

[1] GORBACHENKO, Pavel. What are Functional and Non-Functional Requirements and How to Document These. In: *Enkonix* [online]. Enkonix, 2022 [cit. 2022-05-19]. Dostupné z: https://enkonix.com/blog/functional-requirements-vs-non-functional/

[2] Why is the difference between functional and Non-functional requirements important?. In: *ReQtest* [online]. ReQtest, 2022 [cit. 2022-05-19]. Dostupné z: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

[3] PEREZ, Marin. Guide to UML diagramming and database modeling. In: *Microsoft* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://www.microsoft.com/en-us/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

[4] UML Class Diagram Tutorial. In: *Visual Paradigm* [online]. Visual Paradigm, 2022 [cit. 2022-05-19]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

[5] UML Class Diagram Relationships Explained with Examples. In: *Creately* [online]. Cinergix Pty. Ltd., 2008-2022 [cit. 2022-05-19]. Dostupné z: https://creately.com/blog/diagrams/class-diagram-relationships/

[6] Class Diagrams: Generating schema from Class diagram. In: *Sparx Systems* [online]. Sparx Systems Pty Ltd., 2000-2022 [cit. 2022-05-19]. Dostupné z: https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_domains/sc_class_diagram.html

[7] Use-case diagrams. In: *IBM* [online]. New York: IBM Corporation [cit. 2022-05-19]. Dostupné z: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

[8] Use Case Diagram Relationships Explained with Examples. In: *Creately* [online]. Cinergix Pty. Ltd., 2008-2022 [cit. 2022-05-19]. Dostupné z: https://creately.com/blog/diagrams/use-case-diagram-relationships/

[9] SysML Use Case Models: Use Case Diagram. In: *Sparx Systems* [online]. Sparx Systems Pty Ltd., 2000-2022 [cit. 2022-05-19]. Dostupné z: https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/sysml_use_cases.html

[10] UML Sequence Diagram Template: About the UML Sequence Diagram Template. In: *Miro* [online]. Miro, 2022 [cit. 2022-05-19]. Dostupné z: https://miro.com/templates/uml-sequence-diagram/

[11] The history of databases. In: *ThinkAutomation* [online]. Parkersoftware [cit. 2022-05-19]. Dostupné z: https://www.thinkautomation.com/histories/the-history-of-databases/

[12] CHAPPLE, Mike. Glossary of Common Database Terms: Master the basic lingo of database management. In: *Lifewire* [online]. New York: Dotdash Meredith [cit. 2022-05-19]. Dostupné z: https://www.lifewire.com/databases-glossary-1019603#toc-constraints

[13] PETERSON, Richard. What is DBMS (Database Management System)? Application, Types & Example. In: *Guru99* [online]. Guru99, 2022 [cit. 2022-05-19]. Dostupné z: https://www.guru99.com/what-is-dbms.html

[14] BORONCZYK, Timothy. *Jump Start MySQL*. 1. Austrálie: SitePoint Pty. Ltd., 2015. ISBN 978-0-9924612-8-7.

[15] What Is a Relational Database?. In: *What Is a Relational Database? {Examples, Advantages & Disadvantages}* [online]. Phoenix: phoenixNAP, 2022 [cit. 2022-05-19]. Dostupné z: https://phoenixnap.com/kb/what-is-a-relational-database

[16] How to define relationships between tables in an Access database. In: *How to define relationships between tables in an Access database - Office| Microsoft Docs* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://docs.microsoft.com/en-us/office/troubleshoot/access/define-table-relationships

[17] Description of the database normalization basics. In: *Microsoft Docs* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description

[18] First Normal Form (1NF). In: *Techopedia* [online]. Techopedia Inc., 2022 [cit. 2022-05-19]. Dostupné z: https://www.techopedia.com/definition/25955/first-normal-form-1nf

[19] Second Normal Form (2NF). In: *Techopedia* [online]. Techopedia Inc., 2022 [cit. 2022-05-19]. Dostupné z: https://www.techopedia.com/definition/21980/second-normal-form-2nf

[20] Third Normal Form (3NF). In: *Techopedia* [online]. Techopedia Inc., 2022 [cit. 2022-05-19]. Dostupné z: https://www.techopedia.com/definition/22561/third-normal-form-3nf

[21] Structured Query Language (SQL). In: *Microsoft Docs* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?view=sql-server-ver15

[22] YAO, Jeffrey. CRUD Operations in SQL Server. In: *MSSQLTips* [online]. Edgewood Solutions, LLC, 2006-2022 [cit. 2022-05-19]. Dostupné z: https://www.mssqltips.com/sqlservertip/5569/crud-operations-in-sql-server/

[23] *A tour of the C# language* [online]. In: . Microsoft, 2022 [cit. 2022-05-16]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[24] What is .NET? Introduction and overview. In: *Microsoft Docs* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction

[25] SMITH, Steve. Overview of ASP.NET Core MVC. In: *Overview of ASP.NET Core MVC | Microsoft Docs* [online]. Microsoft, 2022 [cit. 2022-05-19]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0

[26] SPINELLI, Joseph. MVC Overview. In: *MVC Overview. MVC is a design patten used to help... | by Joseph Spinelli | Medium* [online]. Medium, 2018 [cit. 2022-05-19]. Dostupné z: https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

[27] Entity Framework Core. In: *Microsoft Docs* [online]. Microsoft, 2021 [cit. 2022-05-19]. Dostupné z: Entity Framework Core

[28] HTML: HyperText Markup Language. In: *HTML: HyperText Markup Language | MDN* [online]. Mozilla, 2022 [cit. 2022-05-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[29] CSS: Cascading Style Sheets. In: *CSS: Cascading Style Sheets | MDN* [online]. Mozilla, 2022 [cit. 2022-05-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[30] About JavaScript: What is JavaScript?. In: *About JavaScript - JavaScript | MDN* [online]. Mozilla, 2022 [cit. 2022-05-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[31] JQuery: What is jQuery. In: *JavaTpoint* [online]. India: JavaTpoint, 2011-2021 [cit. 2022-05-19]. Dostupné z: https://www.javatpoint.com/what-is-jquery

[32] SEDLÁČEK, Bohuslav. Co je MySQL?: Co je MyQSL – MySQL pro začátečníky. In: *Co je MySQL? - inetio.cz* [online]. inetio s. r. o., 2020 [cit. 2022-05-19]. Dostupné z: https://inetio.cz/co-je-mysql/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UML unifikovaný modelovací jazyk

IDS Integrated Data Store

IBM International Business Machines Corporation

SQL strukturovaný dotazovací jazyk

MVC model-view-controller

NoSQL databáze bez užití jazyku SQL

GDPR obecné nařízení o ochraně osobních údajů

ACID Atomicita, Konzistence, Izolovanost, Trvalost

SŘBD systém řízení báze dat

CSV hodnoty oddělené čárkami

JSON JavaScript Object Notation

CRUD Create, Read, Update, Delete

ISO Mezinárodní organizace pro normalizaci

CSS Kaskádové styly

HTML hypertextový značkovací jazyk

LINQ Language Integrated Query

EF Core Entity Framework Core

AJAX Asynchronous JavaScript and XML

RQ identifikátor funkčního požadavku

NRQ identifikátor nefunkčního požadavku

PDF přenosný formát dokumentů

UC případ užití

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM

Přiložené CD obsahuje:

- Zdrojový kód webové aplikace zabalený ve formátu .zip
- Enteprise Architekt projekt ve formátu .eapx
- Bakalářskou práci ve formátu .pdf