

Aplikace pro evidenci návštěv ve firmě

Kristýna Dvouletá



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav počítačových a komunikačních systémů

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Kristýna Dvouletá
Osobní číslo: A22098
Studijní program: B0688A140008 Informační technologie v administrativě
Forma studia: Prezenční
Téma práce: Aplikace pro evidenci návštěv ve firmě
Téma práce anglicky: Application for Visitor Registration in a Company

Zásady pro vypracování

- Vypracujte literární rešerši na dané téma.
- Specifikujte požadavky na aplikaci.
- Navrhněte aplikaci a představte databázi pro ukládání dat.
- Popište zvolené technologie k implementaci.
- Částečně implementujte navrženou aplikaci.

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

Seznam doporučené literatury:

1. ARLOW, Jim a NEUSTADT, Ila. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 9788025115039.
2. COCKBURN, Alistair. Use Cases: jak efektivně modelovat aplikace. Brno: CP Books, 2005, 262 s. ISBN 8025107213.
3. CONOLLY, Thomas; BEGG, Carolyn E. a HOLOWCZAK, Richard. Mistrovství – databáze: profesionální průvodce tvorbou efektivních databází. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.
4. SELLS, Chris. C# a WinForms: programování formulářů Windows : Microsoft .NET development series. Encyklopedie Zoner press. Brno: Zoner, 2005. ISBN 80-86815-25-0.
5. VIRIUS, Miroslav. Programování v C#: od základů k profesionálnímu použití. Myslíme v.. Praha: Grada Publishing, 2021. ISBN 978-80-271-1216-6.

Vedoucí bakalářské práce: **Ing. Darina Bajusová**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **4. prosince 2024**
Termín odevzdání bakalářské práce: **9. června 2025**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



Ing. Miroslav Matýsek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 4. prosince 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- 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;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- 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(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- při tvorbě této práce jsem použil/a nástroj generativního modelu AI [ChatGPT; <https://chat.openai.com>] za účelem parafrázování vybraných částí textu v teoretické části a asistence při tvorbě kódu. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne

.....

podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a částečnou implementací desktopové aplikace pro evidenci návštěv ve firmě. Cílem projektu je vytvořit přehledný a snadno ovladatelný nástroj, který umožní recepčním zaznamenávat příchody návštěv a spravovat jejich údaje. Teoretická část se věnuje rešerši existujících řešení, návrhovým principům softwarových systémů, modelování pomocí UML, návrhu databázových struktur a vývoji uživatelských rozhraní ve Windows Forms. Praktická část popisuje specifikaci požadavků, návrh databáze v prostředí Microsoft Access a implementaci základních funkčních prvků aplikace v jazyce C#. V rámci řešení byla realizována přibližně třetina plánovaných funkcionalit – přihlášení a odhlášení uživatele, registrace jednorázové návštěvy, zobrazení seznamu aktuálních návštěv, zobrazení detailu konkrétní návštěvy a její odstranění. Výsledná aplikace představuje funkční prototyp, který lze dále rozvíjet směrem k plnému pokrytí požadavků.

Klíčová slova: evidence návštěv, desktopová aplikace, C#, databáze, UML, Windows Forms

ABSTRACT

This bachelor thesis focuses on the design and partial implementation of a desktop application for managing company visits. The aim of the project is to create a clear and user-friendly tool that allows reception staff to record visitor entries and manage related information. The theoretical part covers an overview of existing solutions, key principles of software system design, UML modeling, database structure design, and the development of user interfaces using Windows Forms. The practical part outlines the specification of requirements, the database design using Microsoft Access, and the implementation of basic application features in the C# programming language. Approximately one-third of the planned functionality has been implemented—user login and logout, registration of a single visit, displaying the list of current visits, viewing the details of a selected visit, and deleting a visit. The resulting application represents a working prototype that can be further developed to fully meet the system requirements.

Keywords: visit management, desktop application, C#, database, UML, Windows Forms

Ráda bych poděkovala vedoucí mé bakalářské práce Ing. Darině Bajusové za odborné vedení, vstřícný přístup a ochotu při konzultacích během zpracování této práce.

OBSAH

ÚVOD.....	13
I TEORETICKÁ ČÁST	14
1 ÚVOD DO PROBLEMATIKY	15
1.1 EVIDENCE NÁVŠTĚV A JEJÍ VÝZNAM V ORGANIZACÍCH	15
1.2 PŘEHLED EXISTUJÍCÍCH ŘEŠENÍ V ZAHRANIČÍ	16
1.3 PŘEHLED ŘEŠENÍ VYUŽÍVANÝCH NA ČESKÉM TRHU	16
1.4 VYHODNOCENÍ REŠERŠE.....	17
2 UML.....	18
2.1 ÚVOD DO UML	18
2.2 INŽENÝRSTVÍ POŽADAVKŮ	18
2.2.1 Softwarové požadavky	19
2.2.2 Klasifikace požadavků	19
2.2.3 Strukturování požadavků	20
2.3 MODELOVÁNÍ PŘÍPADŮ UŽITÍ	21
2.3.1 Komponenty modelu případu užití.....	22
2.3.2 Specifikace případu užití.....	25
2.4 TRASOVÁNÍ POŽADAVKŮ MEZI MODELÝ	27
2.5 ENTERPRISE ARCHITEKT	27
3 DATABÁZE.....	28
3.1 CHARAKTERISTIKA DATABÁZOVÝCH SYSTÉMŮ	28
3.1.1 Systém řízení báze dat.....	28
3.2 RELAČNÍ MODEL.....	29
3.2.1 Vlastnosti relačních tabulek	29
3.3 JAZYK SQL	30
3.4 MICROSOFT ACCESS.....	31
4 C#.....	32
4.1 STRUKTURA C#.....	32
4.2 OBJEKTOVĚ ORIENTOVANÝ PŘÍSTUP	32
4.3 PRÁCE S DATABÁZÍ V C#.....	33
4.4 UDÁLOSTI A OVLÁDACÍ PRVKY.....	33
5 WINDOWS FORMS.....	34
5.1 ZÁKLADNÍ PRINCIPY WINDOWS FORMS.....	34
5.2 KOMPONENTY A JEJICH VLASTNOSTI	34
5.3 UDÁLOSTMI ŘÍZENÉ PROGRAMOVÁNÍ	35

5.4	PRÁCE S DATY VE FORMULÁŘÍCH	35
5.5	NASTAVENÍ VZHLEDU A VLASTNOSTÍ FORMULÁŘE	36
II	PRAKTICKÁ ČÁST	37
6	POŽADAVKY	38
6.1	FUNKČNÍ POŽADAVKY	38
6.1.1	Správa návštěv	39
6.1.2	Správa uživatelů	45
6.2	NEFUNKČNÍ POŽADAVKY	47
7	MODEL PŘÍPADŮ UŽITÍ.....	49
7.1	AKTÉŘI	50
7.2	PŘÍPADY UŽITÍ	51
7.2.1	Vytvoření jednorázové návštěvy	52
7.2.2	Vytvoření hromadné návštěvy	53
7.2.3	Vytvoření opakované návštěvy	54
7.2.4	Vytvoření rezervace návštěvy	55
7.2.5	Zobrazit návštěvy	56
7.2.6	Označit odchod návštěvy	57
7.2.7	Vyhledat návštěvu	58
7.2.8	Zobrazit detail návštěvy	59
7.2.9	Zrušit návštěvu/rezervaci	60
7.2.10	Přihlásit se	61
7.2.11	Odhlásit se	62
7.2.12	Vytvořit uživatele	63
7.2.13	Odstranit uživatele	64
7.2.14	Upravit uživatele	65
7.2.15	Přidat zaměstnance	66
7.2.16	Odstranit zaměstnance	67
7.3	MATICE SLEDOVATELNOSTI POŽADAVKŮ	68
8	NÁVRH DATABÁZE	69
8.1	POPIS DATABÁZOVÉHO MODELU	70
8.2	RELACE MEZI TABULKAMI	71
8.3	VZTAH K APLIKACI	71
9	IMPLEMENTACE.....	72
9.1	PŘIHLAŠOVACÍ STRÁNKA	73
9.1.1	Popis funkcionality	73
9.1.2	Popis uživatelského rozhraní	74
9.1.3	Návaznost na další části systému	74
9.2	HLAVNÍ STRÁNKA	75
9.2.1	Popis funkcionality	75
9.2.2	Ovládací prvky a jejich funkce	76
9.2.3	Popis uživatelského rozhraní	76

9.3	REGISTRACE NÁVŠTĚVY	77
9.3.1	Popis funkcionality.....	78
9.3.2	Popis uživatelského rozhraní.....	79
9.3.3	Návaznost na databázi	79
9.4	ZOBRAZENÍ NÁVŠTĚV	80
9.4.1	Popis funkcionality.....	81
9.4.2	Popis uživatelského rozhraní.....	82
9.5	ZOBRAZENÍ DETAILU NÁVŠTĚVY	83
9.5.1	Popis funkcionality.....	84
9.5.2	Popis uživatelského rozhraní.....	85
10	BUDOUCÍ VÝVOJ APLIKACE	86
	ZÁVĚR	88
	SEZNAM POUŽITÉ LITERATURY	89
	SEZNAM OBRÁZKŮ	91
	SEZNAM TABULEK.....	92
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	93
	SEZNAM PŘÍLOH.....	94

ÚVOD

V moderním firemním prostředí hraje efektivní správa návštěv důležitou roli při zajišťování bezpečnosti, přehledu o pohybu osob a profesionálního dojmu při příchodu do společnosti. Tradiční způsoby evidence návštěv, jako jsou papírové knihy, často nevyhovují současným požadavkům na rychlost, přesnost a jednoduchost obsluhy. Z tohoto důvodu se čím dál častěji uplatňují softwarová řešení, která tento proces automatizují a zpřehledňují.

Cílem práce je navrhnout a částečně implementovat aplikaci pro evidenci návštěv ve firmě, která bude sloužit recepčním k registraci příchozích osob, správě jejich údajů a zobrazení aktuálního přehledu návštěv. Aplikace je navržena s ohledem na jednoduchost použití a přehlednost, což ji činí vhodnou pro nasazení v menších firmách bez potřeby složitých systémů.

V teoretické části práce je provedena rešerše existujících systémů pro správu návštěv na českém i zahraničním trhu, analýza požadavků na systém, návrh databázové struktury a přehled použitých technologií – zejména programovací jazyk C#, databázový systém Microsoft Access a vývojové prostředí Windows Forms. Součástí návrhu je také modelování pomocí UML, které slouží k zachycení chování systému.

V rámci praktické části byla realizována přibližně třetina plánované funkcionality – konkrétně přihlášení a odhlášení uživatele, registrace jednorázové návštěvy, zobrazení seznamu aktuálních návštěv, detail záznamu a možnost jeho odstranění. Výsledkem je funkční prototyp, který tvoří základ pro další rozšiřování a úpravy podle potřeb konkrétního provozu.

I TEORETICKÁ ČÁST

1 ÚVOD DO PROBLEMATIKY

1.1 Evidence návštěv a její význam v organizacích

Systémy pro evidenci návštěv (tzv. *Visitor Management Systems – VMS*) hrají klíčovou roli v moderních firmách, které dbají na bezpečnost, efektivitu a profesionální přístup k návštěvníkům. Zatímco tradiční papírové knihy návštěv poskytovaly základní přehled o příchozích osobách, současná řešení nabízejí mnohem širší spektrum funkcí, jež odpovídají požadavkům dnešního podnikatelského prostředí. [1]

Efektivní správa návštěv představuje klíčový nástroj pro zajištění bezpečnosti, organizovanosti a profesionality v rámci firemního prostředí. Mezi hlavní přínosy evidence návštěv patří především zvýšená bezpečnost, která spočívá v kontrole vstupu do objektu, ověřování identity příchozích a zaznamenávání jejich pohybu v prostorách budovy. Další výhodou je lepší organizace a přehlednost díky možnosti rychlého vyhledání konkrétní návštěvy či analýzy historie záznamů. Evidence návštěv zároveň přispívá k souladu s legislativními požadavky – například v případě evakuace nebo bezpečnostního auditu umožňuje jednoznačně doložit, kdo se v daném okamžiku nacházel v objektu. V neposlední řadě hraje důležitou roli i z hlediska reprezentace firmy, protože profesionální přístup k návštěvníkům zanechává pozitivní první dojem a podporuje důvěru v organizaci. [1]

Moderní digitální systémy navíc rozšiřují základní evidenci o řadu pokročilých funkcí, jako je automatické informování hostitele o příchodu návštěvy, tisk návštěvnických štítků, integrace s firemními kalendáři či možnost bezkontaktní registrace prostřednictvím QR kódu nebo tabletu. Tyto technologie významně přispívají k úspoře času a zvyšují uživatelský komfort jak pro návštěvníky, tak pro zaměstnance recepcie. [2]

1.2 Přehled existujících řešení v zahraničí

Moderní systémy pro správu návštěv nabízejí široké spektrum funkcí od jednoduchého přihlášení návštěvníka až po pokročilou bezpečnostní integraci. Na trhu je k dispozici řada komerčních řešení, která se liší cílovou skupinou, funkcemi i cenou. Níže je uveden přehled pěti často používaných řešení. [3]

Systém Envoy nabízí cloudové řešení pro evidenci návštěv, vhodné především pro malé a střední podniky. Umožňuje bezkontaktní přihlášení, tisk štítků, notifikace hostitelům a přizpůsobení designu i procesu registrace. Díky propojení s dalšími nástroji zjednodušuje komunikaci v rámci firmy. [3]

Proxyclick je pokročilý systém pro větší organizace. Nabízí správu návštěv napříč lokalitami, předregistraci hostů, validaci přístupu, automatizaci procesů a soulad s GDPR. Silné jsou také jeho bezpečnostní funkce a možnosti přizpůsobení. [4]

iLobby (také známý jako VisitorOS) je zaměřený na podniky s vysokými nároky na bezpečnost. Umožňuje skenování dokladů, integraci s přístupovými systémy a detailní sledování pohybu osob v objektu. Vhodný je např. pro průmyslové provozy, sklady či datová centra. [5]

Vizito představuje uživatelsky jednoduché řešení pro malé a střední firmy. Mezi jeho funkce patří předregistrace přes e-mail, bezkontaktní přihlášení pomocí QR kódu, vícejazyčné rozhraní a notifikace. Lze jej nasadit jak v cloudu, tak lokálně. [6]

Greetly je minimalistický systém zaměřený na jednoduchost a rychlou implementaci. Umožňuje automatizované oznámení příchodu návštěvníka, vedení historie záznamů a správu vstupních formulářů. Vhodný pro firmy, které hledají jednoduché a dostupné řešení. [7]

1.3 Přehled řešení využívaných na českém trhu

Na českém trhu existuje několik specifických řešení pro evidenci návštěv, která reflektují lokální požadavky na bezpečnost, legislativu (včetně GDPR) a jednoduchost obsluhy. Zde je přehled nejčastěji používaných aplikací.

Safe4Entry je česká webová aplikace zaměřená na správu vstupů návštěvníků a dodavatelů do firemních areálů. Systém umožňuje nejen evidenci návštěv, ale také jejich plánování, ověření proškolení a testování osob před vstupem. Výhodou je také propojení se

zařízeními na místě, jako jsou brány nebo turnikety. Systém je vhodný zejména pro průmyslové podniky a firmy s důrazem na bezpečnost práce a BOZP. [8]

Granus je komplexní systém pro evidenci návštěv, který je využíván ve firmách, úřadech a institucích. Klade důraz na rychlost registrace návštěvníků, která probíhá pomocí čteček ID dokladů. Mezi funkce systému patří tisk návštěvních lístků, přehledná historie návštěv, export dat a také možnost označení nežádoucích návštěvníků. Systém lze přizpůsobit konkrétním požadavkům klienta a propojit s dalšími bezpečnostními moduly. [9]

Společnost Bosch nabízí v rámci svého přístupového systému Access Management také funkci návštěvnické správy. Řešení je určeno především větším organizacím a veřejným institucím. Nabízí předregistraci návštěvníků, samoobslužné kiosky a přehledné rozhraní pro recepční. Kromě toho je možné systém napojit na další bezpečnostní komponenty, jako jsou kamery nebo kontrola vstupu. [10]

1.4 Vyhodnocení rešerše

Na základě provedené rešerše je zřejmé, že existující systémy pro evidenci návštěv (jak zahraniční, tak i tuzemské) nabízející komplexní a často robustní řešení, která cílí na široké spektrum zákazníků a firemních potřeb. Jsou navrženy tak, aby zvládaly i rozsáhlé provozy s vysokými nároky na bezpečnost, integraci s fyzickými zařízeními a dodržování právních předpisů. V mnoha případech však tato řešení obsahují rozsáhlé funkcionality, které mohou být pro menší firmy nadbytečné nebo finančně nedostupné.

Navrhovaná aplikace v rámci této bakalářské práce se odlišuje především svou jednoduchostí, přehledností a zaměřením na konkrétní praktické využití v jedné firmě. Je navržena tak, aby pokrývala klíčové potřeby recepce bez zbytečné složitosti. Díky tomu je snadno implementovatelná a zároveň přizpůsobitelná konkrétním procesům a pracovnímu prostředí dané organizace. Projekt zároveň poskytuje prostor pro budoucí rozšiřování.

2 UML

2.1 Úvod do UML

Unified Modeling Language (UML), v překladu unifikovaný modelovací jazyk, představuje univerzální vizuální prostředek sloužící k modelování různých systémů. Běžně se používá při návrhu softwaru založeného na objektově orientovaném přístupu, ale jeho využití sahá i mimo tuto oblast. [11]

Hlavním cílem vzniku UML bylo sjednocení osvědčených modelovacích technik a přístupů používaných v softwarovém inženýrství. Jazyk byl navržen tak, aby jej bylo možné nasadit ve všech nástrojích typu CASE (Computer-Aided Software Engineering). UML diagramy jsou navíc snadno srozumitelné nejen pro lidi, ale i pro nástroje, které je mohou automaticky zpracovávat. [11]

Je důležité zdůraznit, že UML sám o sobě nepředstavuje metodiku modelování. Nabízí pouze grafickou notaci, kterou lze využít k vytváření modelů. Jazyk není vázán na konkrétní metodický přístup ani vývojový cyklus, a proto jej lze kombinovat s různými existujícími metodami. Za nejvhodnější metodiku k UML se však považuje Unified Process, která definuje role, činnosti a výstupy potřebné k vytvoření funkčního softwarového systému. Z tohoto důvodu je právě Unified Process metodikou, která je s UML nejvíce v souladu, ačkoliv jazyk lze uplatnit i v jiných přístupech k modelování. [11]

2.2 Inženýrství požadavků

Definování a specifikace požadavků probíhají převážně v úvodních fázích projektu, konkrétně ve fázi zahájení a rozpracování. Ještě před samotnou objektově orientovanou analýzou či návrhem je důležité mít alespoň rámcovou představu o cílech systému a o významu požadavků a jejich specifikace. Tyto požadavky by měly být formulovány jazykem, který je srozumitelný budoucím uživatelům systému. Výsledkem této fáze je základní popis toho, co má systém dělat – tedy tzv. inženýrství požadavků. [11]

Inženýrství požadavků zahrnuje určení služeb, které má vyvíjený systém poskytovat, a podmínek, za nichž musí být tyto služby realizovány. Zároveň zahrnuje shromažďování požadavků ze strany uživatelů systému a určení jejich priorit. [11]

2.2.1 Softwarové požadavky

Specifikace softwarových požadavků zahrnuje dva základní modely – model požadavků a model případů užití. Tyto modely představují odlišné, avšak navzájem se doplňující způsoby, jak vyjádřit požadavky na systém. [11]

Model požadavků obsahuje funkční požadavky, které určují, jaké činnosti má systém vykonávat, a dále nefunkční požadavky, jež vymezují omezení týkající se provozu systému. [11]

Model případů užití je tvořen množstvím balíčků, které obsahují samotné případy užití (tedy specifikace funkcionality systému), aktéry (externí role, které se systémem přímo komunikují) a jejich vzájemné vztahy. [11]

Specifikace softwarových požadavků představuje počáteční fázi celého vývojového procesu. Obvykle slouží jako vstup do objektově orientované analýzy, na kterou následně navazuje samotný návrh systému. [11]

2.2.2 Klasifikace požadavků

Požadavek lze chápat jako popis toho, co má být v systému implementováno. Obecně rozlišujeme dva hlavní typy požadavků [11]:

1. Funkční požadavky

- Vymezují, jakým způsobem by se měl systém chovat.
- Popisují konkrétní funkce a činnosti, které má systém vykonávat.

2. Nefunkční požadavky

- Definují vlastnosti systému či omezení, za nichž má být systém provozován nebo vyvíjen.

Požadavky tvoří základ každého systému. Představují formální vyjádření toho, co má systém provádět. Je důležité zdůraznit, že požadavky by měly určovat pouze co má systém dělat, nikoli jak to má být realizováno. Právě toto rozlišení je zásadní. [11]

2.2.3 Strukturování požadavků

Požadavky lze organizovat do tzv. taxonomie, což představuje hierarchické členění typů požadavků, které slouží k jejich kategorizaci. Hlavním cílem takového třídění je přehlednější uspořádání velkého množství různorodých požadavků do menších a lépe zvládnutelných skupin. To výrazně usnadňuje práci s požadavky. [11]

Základní rozdělení na funkční a nefunkční požadavky tvoří nejjednodušší formu taxonomie. Tuto základní strukturu lze dále rozšiřovat podle potřeb konkrétního softwarového projektu. Výběr konkrétních typů požadavků závisí na povaze vyvíjeného systému. Třídění požadavků podle typu je zvláště přínosné v situacích, kdy se pracuje s větším množstvím požadavků. Hloubka hierarchie přitom může být prakticky neomezená. [11]

Každý požadavek může být doplněn o sadu atributů, které poskytují dodatečné informace vztahující se k danému požadavku. Tyto atributy mají popisné názvy a přiřazené hodnoty. Při jejich definici je vhodné dbát na co nejjednodušší formulaci. Nejčastěji používaným atributem je priorita, jejíž hodnota vyjadřuje relativní důležitost požadavku vůči ostatním. Mezi další běžné atributy patří například stav, význam, náročnost, rizikovost nebo stabilita. [11]

2.3 Modelování případů užití

Případ užití slouží jako forma dohody mezi členy projektového týmu a budoucími uživateli systému ohledně toho, jak by se měl systém chovat. Zachycuje způsob, jakým systém odpovídá na požadavky uživatele – označovaného jako primární aktér – v různých situacích. Tento aktér spouští interakci se systémem za účelem dosažení určitého cíle. Reakce systému přitom bere v úvahu potřeby a zájmy všech jeho uživatelů. [12]

Na základě konkrétních požadavků či okolností, které daný požadavek doprovázejí, může systém vykazovat různé varianty chování, tzv. scénáře. Případ užití tyto různé scénáře sjednocuje a spojuje do jednoho celku. [12]

Případy užití jsou zpravidla vyjadřovány textově, ačkoliv existuje i možnost jejich zápisu pomocí vývojových diagramů, sekvenčních diagramů, Petriho sítí nebo prostřednictvím programovacích jazyků. Popis má podobu jednoduchých akčních kroků ve formě vět, v nichž aktér buď realizuje svůj cíl, nebo předává informace jinému aktérovi. [12]

Při tvorbě případů užití je nutné uplatnit tři základní principy, a to jak na jednotlivé věty popisu, tak na celý případ užití. Těmito principy jsou: [12]

1. Rozsah – Co přesně představuje systém, který je modelován?
2. Primární aktér – Kdo má objektivní cíl, který se systémem sleduje?
3. Úroveň – Na jaké úrovni podrobnosti je případ užití popsán?

2.3.1 Komponenty modelu případu užití

Modelování případů užití představuje jednu z metod inženýrství požadavků. Slouží jako nástroj pro zachycení požadavků na systém. Základní struktura modelu případu užití se opírá o čtyři hlavní prvky: [11]

1. Hranice systému

- Při navrhování softwarového systému je prvním krokem určení jeho hranic. Jinými slovy, je nutné stanovit, které prvky spadají do systému (nacházejí se uvnitř jeho hranic) a které jsou mimo jeho rámec. Vymezení hranic systému má zásadní vliv na formulaci funkčních i nefunkčních požadavků. [11]
- Hranice systému zároveň určuje, kdo bude se systémem interagovat (tedy aktéři), a jaký přínos z těchto interakcí plyne (případy užití). V modelu je hranice znázorněna jako obdélník s názvem systému. Aktéři jsou umístěni mimo tento vymezený prostor, zatímco případy užití se nacházejí uvnitř. [11]

2. Aktéři

- Aktér představuje roli, kterou přebírá určitá externí entita ve chvíli, kdy začíná přímo interagovat se systémem. Může jít o uživatele, jiný systém nebo jinou entitu, která se nachází mimo hranice systému, ale vstupuje s ním do kontaktu. [11]
- V jazyce UML jsou aktéři znázorňováni jako stereotyp třídy pomocí specifického symbolu. Nejčastěji se používá ikona postavičky pro zobrazení lidských rolí, zatímco obdélníkové tvary bývají využívány pro reprezentaci rolí jiných systémů. [11]
- Ačkoliv jsou aktéři z hlediska systému vnějšími prvky, systém může obsahovat jejich interní reprezentaci, která umožňuje s nimi efektivně pracovat. [11]
- Při určování aktérů je užitečné přemýšlet o tom, kdo systém využívá nebo s ním nějakým způsobem komunikuje. [11]
- V situacích, kdy potřebujeme modelovat události, které nastávají v konkrétním čase, ale nejsou vyvolány žádným konkrétním aktérem, je možné zavést jako aktéra čas – například v případě automaticky spuštěné zálohy systému. [11]

3. Případy užití

- Příklad užití představuje funkci, kterou systém poskytuje jednomu nebo více aktérům. Jedná se o situaci, ve které aktér očekává od systému určitou službu nebo reakci. Jinými slovy jde o konkrétní způsob, jakým aktér systém využívá. Každý případ užití je vždy vyvolán aktérem a je popsán z jeho pohledu. [11]
- Při identifikaci případů užití je vhodné zaměřit se na to, jakým způsobem budou jednotliví aktéři systém využívat a jaké funkce či služby jim systém poskytuje. [11]
- V UML je případ užití znázorněn elipsou, přičemž jeho název může být umístěn buď uvnitř, nebo pod tímto tvarem. [11]
- Jedním z nejúčinnějších přístupů k definování případů užití je projít si seznam aktérů a u každého promyslet, jakým způsobem bude se systémem pracovat. Názvy případů užití by měly být formulovány jako slovesné vazby, protože každý případ užití vyjadřuje určitou činnost. [11]
- Modelování případů užití je iterativní proces, který se rozvíjí postupným zpřesňováním. Na začátku se obvykle pracuje pouze s názvem případu užití. Postupně se k němu přidávají stručné popisy, které se dále rozvíjejí až do podoby kompletní specifikace. [11]

4. Relace

- Relace představují sémantické vazby mezi modelovými prvky a slouží k propojení jednotlivých objektů a abstrakcí v rámci jazyka UML. Jsou nástrojem, kterým lze vyjádřit významové souvislosti mezi různými částmi modelu. [11]
- Při vytváření modelů případů užití je vhodné dbát na jednoduchost. Relace by se proto měly používat pouze v případech, kdy přispívají k lepší přehlednosti a srozumitelnosti modelu. [11]
- V rámci případů užití existuje několik typů vztahů mezi aktéry nebo mezi samotnými případy užití: [11]
 - **Zobecnění aktéra**
 - Tento vztah se používá v situacích, kdy dva nebo více aktérů sdílejí stejné chování a způsob interakce se systémem. Místo jejich opakovaného popisu lze zavést zobecněného aktéra, který toto chování zastupuje. [11]
 - Konkrétní aktéři (např. osoby či systémy) přebírají role a relace definované tímto nadřazeným aktérem. Zobecněný aktér je zpravidla abstraktní, ale nemusí to být pravidlem – může jít i o konkrétní roli, kterou vykonává reálná osoba nebo systém. [11]
 - **Zobecnění případu užití**
 - Tento typ vztahu se využívá tehdy, když některé případy užití představují konkrétnější formy obecnějšího případu. Odvozený případ užití (potomek) dědí chování i vlastnosti svého nadřazeného případu (předka), ale může některé jeho části upravit nebo předefinovat. [11]
 - **<include>**
 - Relace include umožňuje vyčlenit opakující se části z více případů užití do samostatného případu užití, který je následně do těchto případů vkládán. Zahrnující případ užití (klient) volá chování obsažené v zahrnovaném případě užití (dodavatel). Dodavatelský případ dodává konkrétní chování, které se opakovaně využívá. [11]

- V rámci zahrnujícího případu užití musí být přesně určeno místo, kde je zahrnovaný případ volán. Tento způsob připomíná volání funkce ve strukturálním programování. [11]
- <extend>
 - Relace extend slouží k rozšíření existujícího případu užití o dodatečné chování. Základní případ užití poskytuje tzv. body rozšíření, ke kterým lze připojit nové scénáře. [11]
 - Zajímavostí této relace je, že základní případ užití nemá o rozšířeních žádnou přímou znalost – funguje samostatně, a to i bez připojených rozšíření. Rozšíření jsou tedy volitelná a nezávislá. [11]

2.3.2 Specifikace případu užití

UML jako takový nestanovuje žádný oficiální standard pro specifikaci případů užití. V praxi se však často využívá šablona, která zahrnuje následující části: [11]

1. Název případu užití

- Jazyk UML neurčuje přesná pravidla pro pojmenovávání případů užití. Doporučuje se však používat víceslovné názvy obsahující slovesa či slovesné vazby, přičemž každé slovo by mělo začínat velkým písmenem. Název by měl být stručný a výstižný. [11]

2. Jedinečný identifikátor případu užití

- Jedná se o neměnné číselné označení, které slouží ke snadné identifikaci případu užití, zejména v situacích, kdy by se jeho název mohl v průběhu vývoje změnit. [11]

3. Stručný popis

- Krátký odstavec, který vystihuje hlavní smysl a cíl daného případu užití. [11]

4. Zapojení aktéři

- Z pohledu konkrétního případu užití rozlišujeme dva typy aktérů: [11]
 - Primární aktéři, kteří případ užití iniciují,
 - Vedlejší aktéři, kteří do něj vstupují až po jeho spuštění.
- Každý případ užití je vždy spuštěn pouze jedním aktérem, nicméně v různých situacích jej mohou spouštět různí aktéři. [11]

5. Vstupní podmínky

- Vymezují stav systému před tím než lze případ užití aktivovat. Jedná se o podmínky, které musí být splněny, aby mohl případ užití začít. Pokud žádná taková podmínka neexistuje, do dané části specifikace se uvede „žádné“. [11]

6. Hlavní scénář

- Popisuje sled událostí představující tzv. ideální průběh – situaci, kdy vše probíhá bez problémů, chyb či přerušení. Hlavní scénář vždy začíná činností aktéra, která případ užití spouští. Dále je tvořen očíslovanými kroky uspořádanými podle časové posloupnosti. [11]

7. Výstupní podmínky

- Vymezují stav systému po ukončení případu užití. Udávají, co musí platit po jeho dokončení. Pokud neexistují žádné výstupní podmínky, do příslušného pole se uvede „žádné“. [11]

8. Alternativní scénáře

- Jedná se o varianty hlavního scénáře, které popisují chyby, výjimky, odchylky nebo přerušení běžného průběhu. Obvykle se nevracejí zpět k hlavnímu scénáři, protože řeší specifické podmínky a zpravidla končí s odlišným výsledkem. [11]

2.4 Trasování požadavků mezi modely

Po vytvoření specifikace systémových požadavků a souboru případů užití je důležité uvědomit si, že teprve jejich vzájemné propojení umožňuje odhalit případné nesrovnalosti. Tímto propojením lze zjistit, zda specifikace neobsahuje požadavky, které nejsou v případech užití dosud zachyceny. [11]

Jedním z nástrojů pro sledování návazností mezi požadavky a případy užití je matice sledovatelnosti požadavků. Jedná se o tabulku, v níž jsou na jedné ose uvedeny identifikátory jednotlivých požadavků a na druhé ose názvy případů užití. Nejčastěji se tato matice vytváří pomocí tabulkového procesoru. [11]

Matice sledovatelnosti požadavků slouží jako účinný prostředek pro ověřování konzistence modelu. Objeví-li se v matici požadavek, který není spojen s žádným případem užití, ukazuje to na to, že daný požadavek není v modelu případů užití zohledněn. Naopak, pokud existuje případ užití, který není spojen s žádným požadavkem, signalizuje to, že specifikace požadavků není úplná. [11]

2.5 Enterprise Architect

Enterprise Architect je výkonný modelovací nástroj vyvíjený společností Sparx Systems, který slouží především k návrhu, vizualizaci a správě komplexních systémů v oblasti softwarového inženýrství, podnikové architektury a systémového inženýrství. Podporuje celou řadu modelovacích standardů, jako jsou UML, BPMN, SysML a ArchiMate, což z něj činí flexibilní řešení pro široké spektrum projektů. [13]

Program umožňuje uživatelům vytvářet detailní modely systémů, procesů a struktur organizace, přičemž poskytuje nástroje pro sledování požadavků, řízení verzí, spolupráci v týmu a generování dokumentace. Díky své škálovatelnosti je vhodný jak pro malé týmy, tak pro rozsáhlé podnikové projekty. Enterprise Architect rovněž nabízí integrace s dalšími nástroji, podporu skriptování a možnosti rozšíření prostřednictvím plug-inů. [13]

Celkově lze říct, že Enterprise Architect poskytuje ucelené prostředí pro modelování a správu životního cyklu systémů, které pomáhá organizacím zefektivnit vývoj, zlepšit komunikaci mezi týmy a podpořit rozhodování na základě přesných a aktuálních modelů. [13]

3 DATABÁZE

3.1 Charakteristika databázových systémů

Databáze představuje centralizované, případně rozsáhlé úložiště dat, které mohou současně využívat různá oddělení i jednotliví uživatelé v rámci organizace. Všechna potřebná data jsou v databázi uspořádána tak, aby se minimalizovalo jejich opakování. Klíčovým aspektem je skutečnost, že databáze obvykle nebývá ve vlastnictví konkrétního oddělení či uživatele, ale slouží jako společný zdroj napříč celou organizací. [14]

Databáze neobsahuje pouze provozní data, ale rovněž informace, které tato data popisují. Právě z tohoto důvodu je často označována jako sebedopisující kolekce integrovaných záznamů. Tato popisná data, známá jako metadata (tedy „data o datech“), jsou uchovávána ve speciální části databáze nazývané systémový katalog nebo datový slovník. [14]

Schopnost databáze popisovat vlastní strukturu zajišťuje tzv. nezávislost dat. Jinými slovy, při přidání nových struktur do databáze nebo úpravách těch existujících mohou aplikace, které s databází pracují, i nadále fungovat bez nutnosti změn – za předpokladu, že nejsou přímo závislé na dané úpravě. [14]

Součástí definice databáze je také pojem logická souvislost. Při analýze informačních potřeb organizace se identifikují klíčové objekty, které je nutné v databázi reprezentovat, a současně se určují logické vazby mezi těmito objekty. [14]

3.1.1 Systém řízení báze dat

Jedním z klíčových pojmů v oblasti databázových systémů je systém pro řízení databáze (DBMS – Database Management System). Tento software slouží jako prostředník mezi samotnou databází, uživateli a aplikacemi, které s databází pracují. DBMS umožňuje provádět základní operace s daty – jejich vkládání, úpravu, mazání a vyhledávání. [14]

V případě, že jsou všechna data i jejich popis centralizována v jednom úložišti, může DBMS nabídnout uživatelům jednotný způsob práce s těmito daty prostřednictvím tzv. dotazovacího jazyka. Nejznámějším a nejpoužívanějším jazykem tohoto typu je SQL (Structured Query Language), který slouží ke komunikaci s databází a získávání požadovaných informací. [14]

3.2 Relační model

Relační model je založen na matematickém konceptu zvaném „relace“, který je v praktickém využití reprezentován tabulkovou strukturou.

Tento datový model tvoří pět základních prvků: [14]

1. Relace – tabulka složená z řádků a sloupců.
2. Atribut – název jednotlivého sloupce v rámci relace.
3. Datová n-tice – konkrétní řádek v relaci.
4. Doména – soubor hodnot, které mohou být přiřazeny jednomu či více atributům.
5. Relační databáze – kolekce relací (tabulek), které prošly procesem normalizace.

V rámci relačního modelu slouží relace k uchovávání údajů o objektech, které je třeba v databázi evidovat. Každá relace má podobu tabulky, kde jednotlivé řádky představují datové n-tice a sloupce odpovídají jednotlivým atributům. Pořadí atributů ve sloupcích není rozhodující – jejich změna nemění význam relace. Stejně tak nezáleží na pořadí řádků, protože jejich přeuspořádání obsah relace nijak neovlivní. Významnou roli hrají domény, které určují platné hodnoty pro jednotlivé atributy a tím i jejich význam. [14]

3.2.1 Vlastnosti relačních tabulek

Aby bylo možné efektivně spravovat data a vytvářet mezi tabulkami logické vazby, využívají se tzv. klíče – specifické sloupce, které slouží k identifikaci a propojení jednotlivých záznamů: [15]

1. Primární klíč

- Primární klíč je sloupec nebo kombinace sloupců, které jednoznačně určují každý řádek v tabulce. V databázové tabulce může být definován pouze jeden primární klíč a jeho hodnoty musí být unikátní a nesmí obsahovat hodnotu NULL. [15] [16]

2. Cizí klíč

- Cizí klíč je sloupec, který v jedné tabulce odkazuje na primární klíč jiné tabulky. Slouží k vytvoření vztahu mezi dvěma tabulkami a umožňuje jejich propojení. Na rozdíl od primárního klíče se hodnoty cizího klíče mohou opakovat. [15] [16]

Při návrhu relační databáze je klíčové správně definovat vztahy mezi jednotlivými tabulkami. Vytváření těchto relací je zásadní pro zajištění vzájemného propojení a konzistence dat. Díky nim lze uchovávat data přehledně, bez zbytečného opakování, a zároveň s nimi efektivně pracovat jako se vzájemně souvisejícími informacemi. Rozlišuje tři základní typy relací: [15]

1. Relace 1:1

- V tomto typu relace odpovídá jednomu záznamu v první tabulce právě jeden záznam ve druhé tabulce. Používá se především v případech, kdy je výhodné rozdělit tabulku s mnoha sloupci na dvě menší. V praxi se však tento typ relace využívá spíše zřídka. [15] [16]

2. Relace 1:N

- Nejčastější typ relace v relačních databázích. Jeden záznam v první (hlavní) tabulce může být propojen s více záznamy ve druhé (podřízené) tabulce. Primární klíč z hlavní tabulky je v tomto případě použit jako cizí klíč v tabulce podřízené. [15] [16]

3. Relace M:N

- Tento typ relace nelze realizovat přímo mezi dvěma tabulkami. Místo toho se využívá tzv. převodní tabulka, která propojuje záznamy obou tabulek pomocí dvou cizích klíčů. Každý z těchto klíčů odkazuje na primární klíč jedné z propojených tabulek. Převodní tabulka často obsahuje složený primární klíč z těchto dvou cizích klíčů, čímž se zajistí jedinečnost každé kombinace. [15] [16]

3.3 Jazyk SQL

Structured Query Language (SQL) představuje standardizovaný jazyk používaný pro správu a manipulaci s relačními databázemi. Umožňuje definovat strukturu databáze, manipulovat s daty a provádět dotazy za účelem získávání informací. Díky své jednoduché a srozumitelné syntaxi, která se podobá angličtině, je SQL přístupný i pro začátečníky. [17]

Relační databáze strukturuje data do tabulek, kde jednotlivé řádky představují záznamy a sloupce odpovídají atributům. SQL poskytuje prostředky pro vytváření těchto tabulek, vkládání, aktualizaci a mazání dat, stejně jako pro definování vztahů mezi tabulkami. Pro manipulaci s daty a načítání dat z databáze se používá: [17]

SELECT – používá se pro získání dat z jedné nebo více tabulek.

INSERT – slouží k přidání nového záznamu do příslušné tabulky.

UPDATE – umožňuje upravovat již existující údaje v tabulce.

DELETE – využívá se k odstranění záznamů z tabulky.

SQL je podporován většinou moderních databázových systémů, jako jsou MySQL, PostgreSQL, Oracle nebo Microsoft SQL Server. Jeho znalost je nezbytná pro efektivní práci s databázemi a tvorbu spolehlivých a výkonných databázových aplikací. [17]

3.4 Microsoft Access

Microsoft Access je databázová aplikace vyvíjená firmou Microsoft, jež je integrována do kancelářského balíku Microsoft Office. Umožňuje vytvářet a spravovat relační databáze a je určen především pro menší až středně velké projekty, kde je potřeba efektivně pracovat s daty bez nutnosti pokročilého programování. Access umožňuje ukládat, třídit, filtrovat a analyzovat data prostřednictvím tabulek, dotazů, formulářů a sestav. [18]

Jednou z hlavních výhod tohoto nástroje je jeho uživatelsky přívětivé grafické rozhraní, které usnadňuje práci i méně zkušeným uživatelům. Umožňuje rychlé vytvoření databázové aplikace pomocí předdefinovaných šablon, a zároveň poskytuje možnosti pokročilejších úprav prostřednictvím jazyka VBA (Visual Basic for Applications). Access také umožňuje napojení na jiné databázové systémy a integraci s dalšími nástroji Microsoftu. [18]

Díky kombinaci jednoduchosti a široké funkčnosti je Microsoft Access vhodným nástrojem pro jednotlivce, malé firmy či oddělení větších organizací, které potřebují spravovat a analyzovat data bez nutnosti nasazení plnohodnotného databázového serveru. [18]

4 C#

Programovací jazyk C# (C Sharp) je moderní, objektově orientovaný jazyk vyvinutý společností Microsoft jako součást platformy .NET. Jeho vývoj byl zahájen na přelomu tisíciletí a veřejně představen byl v roce 2000. C# byl navržen s cílem zkombinovat jednoduchost jazyka C a výkonnostní možnosti C++ s bezpečností a vysokou produktivitou vývoje známou z jazyka Java. V současnosti patří mezi nejpoužívanější jazyky v oblasti podnikových aplikací, vývoje desktopových systémů, mobilních aplikací (Xamarin, MAUI) i webových technologií (ASP.NET). [19] [16]

4.1 Struktura C#

Každý C# program je tvořen jednou nebo více třídami. V rámci těchto tříd jsou definovány vlastnosti (proměnné) a metody (funkce), které určují, jak se daný objekt bude chovat. Třídy mohou být rozděleny do samostatných souborů, čímž se zajišťuje přehlednost a modularita kódu. V případě aplikací s uživatelským rozhraním je běžné, že každá obrazovka nebo formulář má vlastní třídu. [19]

Metody mohou být přístupové (veřejné, soukromé) a mohou vracet hodnotu nebo být bez návratové hodnoty. Uvnitř metod se často využívají základní programátorské konstrukce jako jsou podmínky, cykly, proměnné, výčty nebo přepínače. Díky silné typové kontrole dochází k odhalování chyb již při překladu, což přispívá ke stabilitě celého systému. [19]

4.2 Objektově orientovaný přístup

C# je plně objektově orientovaný jazyk, který využívá všech klíčových principů OOP (Object-Oriented Programming). Hlavním z těchto principů je zapouzdření, které umožňuje ukrýt vnitřní stav objektu a zpřístupnit ho pouze prostřednictvím definovaných metod. Dále jazyk podporuje dědičnost, kdy jedna třída může přebírat vlastnosti a metody jiné třídy, a polymorfismus, který umožňuje přepsání nebo přetížení metod podle potřeb konkrétní implementace. [19]

Při vývoji uživatelských aplikací se typicky vytvářejí vlastní třídy, které dědí od základních tříd poskytovaných platformou .NET (např. Form). Díky tomu je možné snadno rozšiřovat funkcionalitu a přizpůsobovat chování aplikace konkrétním požadavkům. [19]

Důležitou vlastností objektově orientovaného programování v C# je práce s konstrukcemi jako jsou konstruktory, vlastnosti (get / set), statické členy nebo přetížené

metody. Tyto prostředky umožňují efektivně definovat chování objektů a přenášet mezi nimi data. [19]

4.3 Práce s databází v C#

C# umožňuje práci s relačními databázemi pomocí rozhraní ADO.NET. To poskytuje sadu tříd a metod pro navázání spojení s databází, vykonání SQL příkazů a čtení nebo zápis dat. Typické komponenty používané pro komunikaci s databází zahrnují objekty jako Connection, Command, DataReader nebo DataAdapter. [19]

Při práci s databází je klíčové využívání parametrizovaných dotazů, které zajišťují nejen správné zpracování datových typů, ale také ochranu proti útokům typu SQL injection. Připojení k databázi je možné realizovat prostřednictvím standardního připojovacího řetězce, který definuje cestu k databázovému souboru, typ databáze a další potřebné parametry. [19]

Z hlediska vývoje aplikací, které pracují s uloženými daty, je běžné otevírat spojení s databází v konkrétních momentech (např. při ukládání záznamu) a ihned po operaci spojení zavřít. Tím se optimalizuje práce se systémovými prostředky a zvyšuje výkon. [19]

4.4 Události a ovládací prvky

Jedním z nejdůležitějších principů vývoje uživatelského rozhraní v C# je událostmi řízené programování. To znamená, že program reaguje na akce uživatele, jako je kliknutí na tlačítko, výběr ze seznamu, zadání textu nebo zavření formuláře. [19]

Každý vizuální prvek (tzv. ovládací prvek) může generovat různé události, na které lze navázat příslušné obslužné metody. Například tlačítko má událost Click, která se spustí při jeho stisknutí. Události jsou definovány jako delegáti, což umožňuje volat metodu při konkrétní události. [19]

C# poskytuje bohatou knihovnu ovládacích prvků, které jsou součástí platformy Windows Forms. Mezi nejčastěji používané komponenty patří Button, Label, TextBox, ComboBox, ListBox nebo DateTimePicker. Každý z těchto prvků má své vlastnosti (např. text, velikost, barvu) a může být programově ovládán v reálném čase. [19]

Díky kombinaci událostmi řízeného modelu a rozsáhlé knihovny ovládacích prvků lze v C# vytvářet bohaté a interaktivní desktopové aplikace, které přesně reagují na potřeby uživatele. [19]

5 WINDOWS FORMS

Windows Forms (WinForms) je grafická knihovna .NET Frameworku, určená pro vývoj desktopových aplikací na platformě Microsoft Windows. Umožňuje vytvářet interaktivní aplikace s vizuálním rozhraním tvořeným okny (formuláři) a množstvím standardních ovládacích prvků. WinForms poskytuje jednoduchý model pro návrh a správu oken, který vychází z tradičního API systému Windows, ale výrazně jej zjednodušuje a integruje do objektového prostředí .NET. [20] [16]

5.1 Základní principy Windows Forms

Každá aplikace vytvořená pomocí Windows Forms se skládá z jednoho nebo více formulářů (tříd děděných z `System.Windows.Forms.Form`), které slouží jako jednotlivá okna aplikace. Formuláře mohou být hlavní (např. úvodní okno aplikace) nebo modální (např. okno detailu nebo potvrzení). Vývojář má plnou kontrolu nad jejich vzhledem, obsahem i chováním. [20]

Formulář je možné navrhnout ručně pomocí kódu, nebo vizuálně pomocí návrhového režimu ve vývojovém prostředí Visual Studio. Každý formulář má definovanou strukturu komponent, jejich vlastnosti, události a vizuální uspořádání. Tyto prvky jsou automaticky uloženy v částečném souboru (`.Designer.cs`), který se propojuje s logikou ve vlastní třídě. [20]

5.2 Komponenty a jejich vlastnosti

Windows Forms poskytuje bohatou sadu ovládacích prvků (komponent), které lze přetáhnout na formulář a přiřadit jim specifické vlastnosti. Mezi nejčastěji používané komponenty patří: [20]

- Label – pro zobrazování statického textu,
- TextBox – pro vkládání uživatelských vstupů,
- Button – pro spouštění událostí,
- ComboBox – rozbalovací seznam pro výběr z možností,
- ListBox – seznam položek pro výběr nebo zobrazení dat,
- DateTimePicker – prvek pro volbu data a času.

Každý prvek má vlastní sadu vlastností, jako je text, pozice, velikost, barva nebo zarovnání. Ovládací prvky jsou hierarchicky vnořené do formuláře a řízené pomocí rodičovských kontejnerů (např. Panel, GroupBox). [20]

Rozložení komponent na formuláři je možné řídit absolutně (pomocí souřadnic), nebo prostřednictvím ukotvení a zarovnání (Anchor, Dock), které zajišťují pružnější přizpůsobení velikosti okna. Při návrhu grafického rozhraní se běžně kombinuje statické rozmístění s dynamickým řízením velikosti a pozice. [20]

5.3 Událostmi řízené programování

Windows Forms je postaven na událostním modelu, který umožňuje reagovat na interakce uživatele s rozhraním. Každý ovládací prvek generuje události, které je možné zachytit a zpracovat ve formě tzv. obslužných metod. Nejčastějšími událostmi jsou Click (kliknutí), TextChanged (změna textu), SelectedIndexChanged (změna výběru) nebo Load (načtení formuláře). [20]

Z hlediska architektury je každá událost zprostředkována pomocí delegátů, což je typ, který odkazuje na metodu. Přiřazení události konkrétní metodě probíhá buď automaticky při návrhu formuláře, nebo ručně pomocí syntaxe jazyka C#. Tento přístup umožňuje oddělit logiku chování aplikace od vizuálního návrhu. [20]

Události tvoří základ interakce mezi uživatelem a aplikací. Většina funkcí, jako je validace vstupu, přechod mezi okny nebo zápis dat do databáze, je spouštěna právě událostmi v reakci na uživatelské akce. [20]

5.4 Práce s daty ve formulářích

Formuláře ve Windows Forms umožňují práci s uživatelským vstupem i zobrazením dat. Pro vkládání údajů se využívají komponenty jako TextBox nebo ComboBox, pro zobrazení pak ListBox nebo Label. Data lze ukládat do proměnných, předávat mezi formuláři nebo zapisovat přímo do databáze. [20]

WinForms podporuje snadné předávání dat mezi formuláři pomocí konstruktorů nebo veřejných vlastností. Každý formulář může být zobrazen jako samostatné okno (Show) nebo jako modální dialog (ShowDialog), což ovlivňuje tok řízení programu. [20]

Při načítání dat z databáze nebo jejich zápisu do formulářů je možné využívat vlastní metody pro aktualizaci komponent. To umožňuje dynamické načítání záznamů, filtrování nebo detailní zobrazení položek podle potřeby uživatele. [20]

5.5 Nastavení vzhledu a vlastností formuláře

Windows Forms umožňuje detailní nastavení vzhledu a chování samotného formuláře. Mezi klíčové vlastnosti patří: [20]

- `FormBorderStyle` – určuje, zda má formulář pevnou nebo měnitelnou velikost,
- `StartPosition` – určuje výchozí pozici formuláře na obrazovce (např. na střed),
- `MaximizeBox` / `MinimizeBox` – povolují nebo zakazují maximalizaci/minimalizaci okna,
- `Text` – název formuláře zobrazovaný v horním titulk.

Tyto vlastnosti ovlivňují, jak bude formulář vypadat a jak se bude chovat z pohledu uživatele. Jejich správné nastavení zajišťuje konzistentní uživatelský zážitek, zejména pokud se jedná o aplikaci určenou k nasazení v reálném firemním prostředí. [20]

II PRAKTICKÁ ČÁST

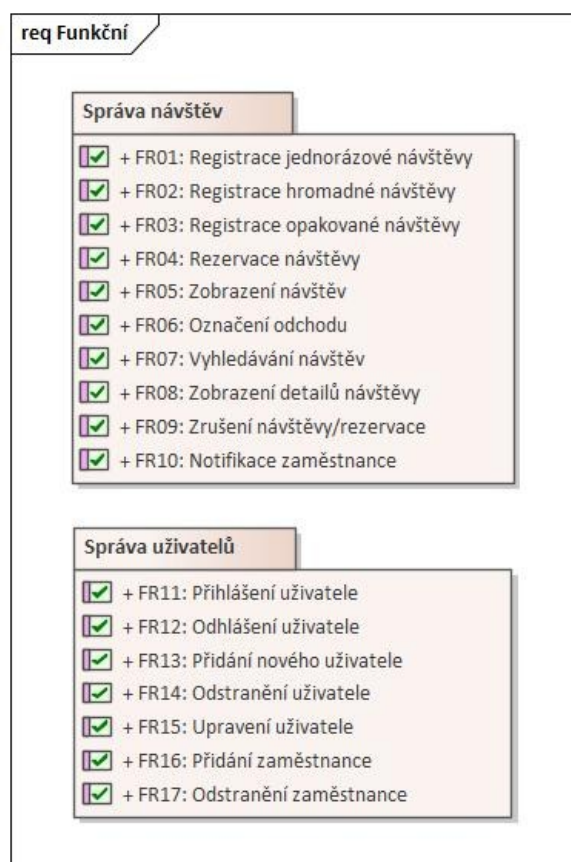
6 POŽADAVKY

V této části jsou popsány požadavky kladené na navrhovaný systém. Požadavky slouží jako základ pro následný návrh a implementaci systému. Jsou rozděleny na funkční a nefunkční, přičemž každá skupina popisuje jiný aspekt očekávaného chování systému. Cílem specifikace požadavků je přesně vymezit, co má systém umět, jak se má chovat a jaké podmínky musí splňovat, aby byl plně použitelný pro koncové uživatele.

6.1 Funkční požadavky

Funkční požadavky popisují konkrétní funkcionalitu systému, tedy co systém dělá a jaké služby poskytuje svým uživatelům. V této práci jsou funkční požadavky rozděleny do dvou hlavních oblastí: správa návštěv a správa uživatelů.

Jednotlivé požadavky definují procesy jako je registrace návštěvy, rezervace návštěvy nebo přihlášení uživatele. Tyto požadavky tvoří základ pro modelování případů užití a následný návrh rozhraní. Jsou označeny zkratkou FR a následně číslem požadavku.

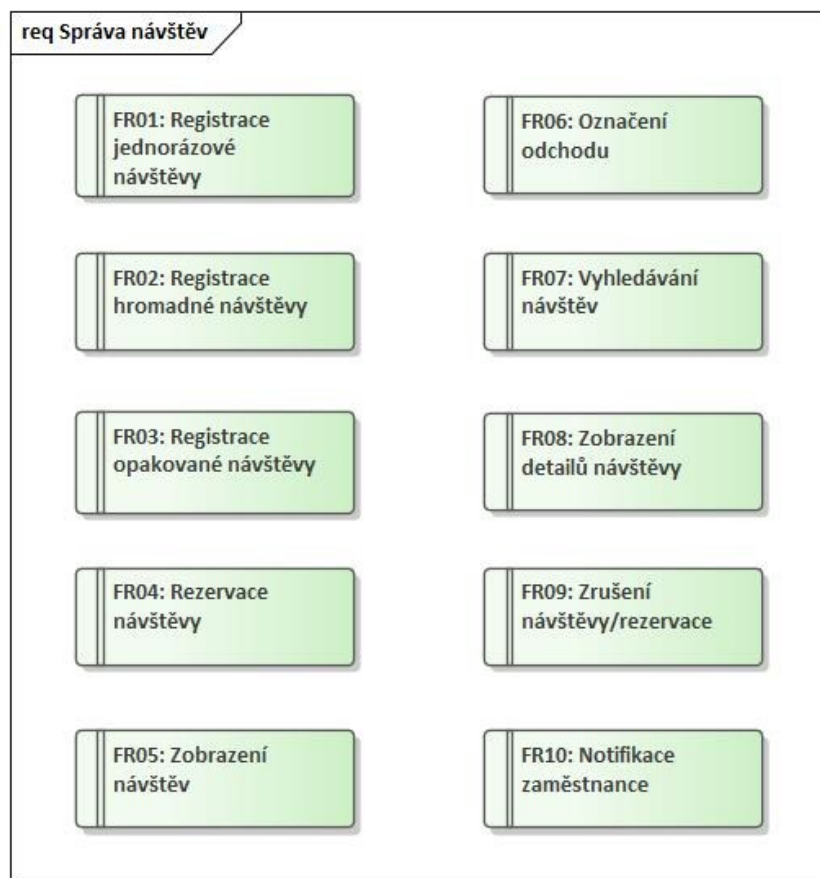


Obrázek 1: Funkční požadavky

(zdroj: vlastní)

6.1.1 Správa návštěv

Modul Správa návštěv slouží k evidenci a organizaci návštěv v rámci systému. Umožňuje uživatelům registrovat jednorázové, opakované i hromadné návštěvy, provádět jejich rezervace, notifikovat zaměstnance a sledovat příchody a odchody návštěvníků. Součástí je také zobrazení, vyhledávání a zrušení návštěv.



Obrázek 2: Funkční požadavky pro správu návštěv

(zdroj: vlastní)

FR01: Registrace jednorázové návštěvy

- Systém umožní uživateli zadat novou jednorázovou individuální návštěvu.
- Registraci je možné vytvořit pouze v den návštěvy.
- Registrace jednorázové návštěvy zahrnuje základní formulář se vstupními údaji:
 - Jméno a příjmení návštěvníka
 - Důvod návštěvy
 - Čas příchodu
 - Zaměstnanec (výběr ze seznamu)
 - Poznámka (nepovinné)
 - Seznam vybavení
- Registrace se uloží do databáze a zobrazí v seznamu Aktuální návštěvy.

FR02: Registrace hromadné návštěvy

- Systém umožní uživateli zadat více návštěvníků najednou v rámci jedné návštěvy, která je jednorázová.
- Registraci je možné vytvořit pouze v den návštěvy.
- Registrace hromadné návštěvy zahrnuje rozšířený formulář se vstupními údaji:
 - Seznam návštěvníků (jméno a příjmení každého účastníka)
 - Důvod návštěvy
 - Čas příchodu
 - Zaměstnanec (výběr ze seznamu)
 - Poznámka (nepovinné)
 - Seznam vybavení
- Registrace se uloží do databáze a zobrazí v seznamu Aktuální návštěvy.

FR03: Registrace opakované návštěvy

- Systém umožní uživateli zadat novou opakovanou individuální návštěvu.
- Registraci je možné vytvořit pouze v den návštěvy.
- Registrace opakované návštěvy zahrnuje rozšířený formulář se vstupními údaji:
 - Jméno a příjmení návštěvníka
 - Důvod návštěvy
 - Čas příchodu
 - Zaměstnanec (výběr ze seznamu)
 - Poznámka (nepovinné)
 - Seznam vybavení
 - Frekvence opakování:
 - V rámci frekvence opakování se nastaví inkrement a jednotka.
 - Denně
 - Týdně
 - Měsíčně
 - Vlastní (uživatel vybere z kalendáře konkrétní dny)
- Registrace se uloží do databáze a zobrazí v seznamu Aktuální návštěvy.
- Princip funkčnosti Opakované návštěvy:
 - V rámci Opakované návštěvy Recepční vyplňuje parametry inkrement a jednotka, díky čemuž systém ukládá Opakovanou návštěvu do databáze.
 - V okamžiku, kdy Recepční zvolí možnost odchodu, systém vytvoří na základě těchto parametrů novou rezervaci, která převezme data z registrované Opakované návštěvy a datum zvýší o inkrement opakování, který byl během této registrace vyplněn.
 - Tuto rezervaci zobrazí v seznamu Plánované návštěvy (rezervace).

- A tímto způsobem se bude vytvářet Opakovaná návštěva do doby, než Recepční zruší opakovaně vytvořenou rezervaci ze systému.
- Tento princip se netýká frekvence opakování typu „Vlastní“, protože k zakládání rezervací (podle zvolených datumů) dochází už při vytváření návštěvy (registrace).

FR04: Rezervace návštěvy

- Systém umožní uživateli předem vytvořit rezervaci návštěvy.
- Rezervace se musí vytvořit minimálně den předem.
- Rezervace jednorázové návštěvy zahrnuje formulář se vstupními údaji:
 - Jméno a příjmení návštěvníka
 - Důvod návštěvy
 - Zaměstnanec (výběr ze seznamu)
 - Datum
- Rezervace se uloží do databáze a zobrazí v seznamu Plánované návštěvy (rezervace).
- Rezervace má pouze informační charakter, v případě, že návštěva vstoupí do budovy, Rezervace se zruší a Recepční vytvoří novou návštěvu v rámci registrace návštěvy.

FR05: Zobrazení návštěv

- Systém umožní uživateli zobrazit seznam:
 - Aktuálních návštěv:
 - Jedná se o návštěvy s dnešním datem, u kterých nebyl zaznamenán čas odchodu.
 - Plánovaných návštěv (rezervací):
 - Jedná se o návštěvy, jejichž datum je > dnešní datum.
 - Historie návštěv:
 - Jedná se o návštěvy, u kterých byl zaznamenán čas odchodu.

FR06: Označení odchodu

- Systém umožní uživateli označit čas odchodu.
- Čas odchodu se zobrazí v seznamu Aktuálních návštěv v detailech návštěvy.
- V případě, že se jedná o Jednorázovou návštěvu, při zaznamenání času odchodu se tento typ návštěvy přesune do seznamu Historie návštěv.
- V případě, že se jedná o Opakovanou návštěvu (kromě typu „Vlastní“), při zaznamenání času odchodu se pro tento typ návštěvy vytvoří nová Rezervace, která se zobrazí v seznamu Plánovaných návštěv (rezervací).

FR07: Vyhledávání návštěv

- Systém umožní uživateli vyhledat konkrétní návštěvu podle zadaných kritérií:
 - Jméno a příjmení návštěvníka
 - Zaměstnanec (výběr ze seznamu)
 - Datum
- Systém provede vyhledání ve všech záznamech ze seznamu:
 - Aktuálních návštěv
 - Plánovaných návštěv (rezervací)
 - Historie návštěv

FR08: Zobrazení detailů návštěvy

- Systém umožní uživateli zobrazit detailní informace o vybrané návštěvě, které byly uloženy během Registrace nebo Rezervace.
- Detaily návštěvy bude možné zobrazit ze seznamu:
 - Aktuální návštěvy
 - Plánované návštěvy (rezervace)
 - Historie návštěv
- V rámci Zobrazení detailů návštěvy kromě zobrazení informací vyplněných během Registrace/Rezervace, bude možné zaznamenat čas odchodu nebo zrušení návštěvy/rezervace.

FR09: Zrušení návštěvy/rezervace

- Systém umožní uživateli smazat konkrétní návštěvu/rezervaci ze seznamu:
 - Aktuálních návštěv
 - Plánovaných návštěv (rezervací)

FR10: Notifikace zaměstnance

- Systém automaticky upozorní příslušného zaměstnance na nově zaregistrovanou nebo rezervovanou návštěvu.

6.1.2 Správa uživatelů

Modul Správa uživatelů umožňuje správu přístupů do systému. Obsahuje funkce pro přihlášení a odhlášení uživatelů, přidávání nových účtů, odstraňování stávajících a upravování již existujících účtů. Cílem je zajistit, aby systém používali pouze oprávnění uživatelé.



Obrázek 3: Funkční požadavky pro správu uživatelů

(zdroj: vlastní)

FR11: Přihlášení uživatele

- Systém umožní uživateli přihlásit se pomocí přihlašovacích údajů a získat přístup k aplikaci podle úrovně oprávnění.
- Vstupní údaje: Uživatelské jméno, Heslo.

FR12: Odhlášení uživatele

- Systém ukončí aktuální relaci, provede odhlášení ze systému a přesměruje uživatele na přihlašovací obrazovku.

FR13: Přidání nového uživatele

- Systém umožní administrátorovi přidat nového uživatele a přiřadit mu roli.
- Vstupní údaje:
 - Uživatelské jméno
 - Heslo
 - Jméno a příjmení
 - Role (oprávnění)
 - E-mail

FR14: Odstranění uživatele

- Systém umožní administrátorovi odstranit uživatele ze systému.

FR15: Upravení uživatele

- Systém umožní administrátorovi modifikovat údaje o uživateli.

FR16: Přidání zaměstnance

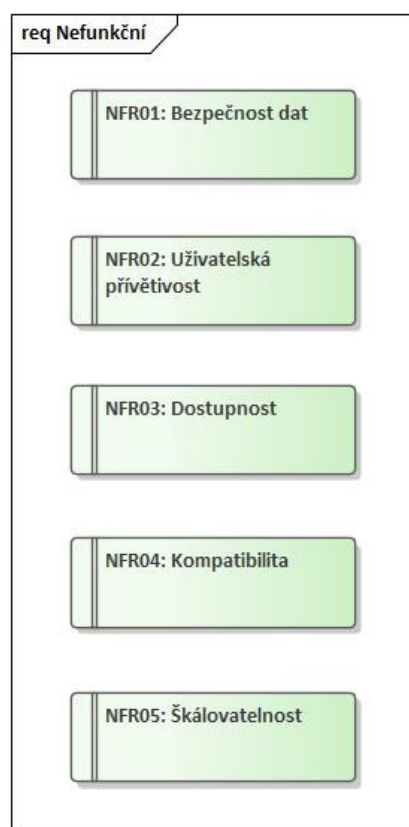
- Systém umožní administrátorovi přidat zaměstnance do systému.
- Vstupní údaje:
 - Jméno a Příjmení
 - Email
 - Telefon
 - Kancelář

FR17: Odstranění zaměstnance

- Systém umožní administrátorovi odstranit zaměstnance ze systému.

6.2 Nefunkční požadavky

Nefunkční požadavky určují kvalitativní vlastnosti systému, které nesouvisí přímo s funkcionalitou, ale výrazně ovlivňují použitelnost a provoz. Slouží k zajištění toho, že systém bude nejen funkční, ale také spolehlivý, bezpečný a efektivní z pohledu koncového uživatele. Tato kategorie zahrnuje požadavky na bezpečnost, uživatelskou přívětivost, dostupnost, kompatibilitu a škálovatelnost. Jsou označeny zkratkou NFR a následně číslem požadavku.



Obrázek 4: Nefunkční požadavky

(zdroj: vlastní)

NFR01: Bezpečnost dat

- Systém musí zajistit ochranu přístupu k datům tím, že je zpřístupní pouze autorizovaným uživatelům.
- Přístup do systému je umožněn pouze po ověření identity uživatele prostřednictvím přihlašovacích údajů (uživatelské jméno a heslo).

NFR02: Uživatelská přívětivost

- Systém musí poskytovat intuitivní, přehledné a snadno ovladatelné uživatelské rozhraní.
- Uživatelské prostředí bude navrženo tak, aby bylo srozumitelné i pro uživatele bez technických znalostí.

NFR03: Dostupnost

- Systém bude zajišťovat dostupnost služeb každý pracovní den v čase od 7:00 do 18:00.

NFR04: Kompatibilita

- Systém bude podporovat provoz v prostředí operačního systému Windows.

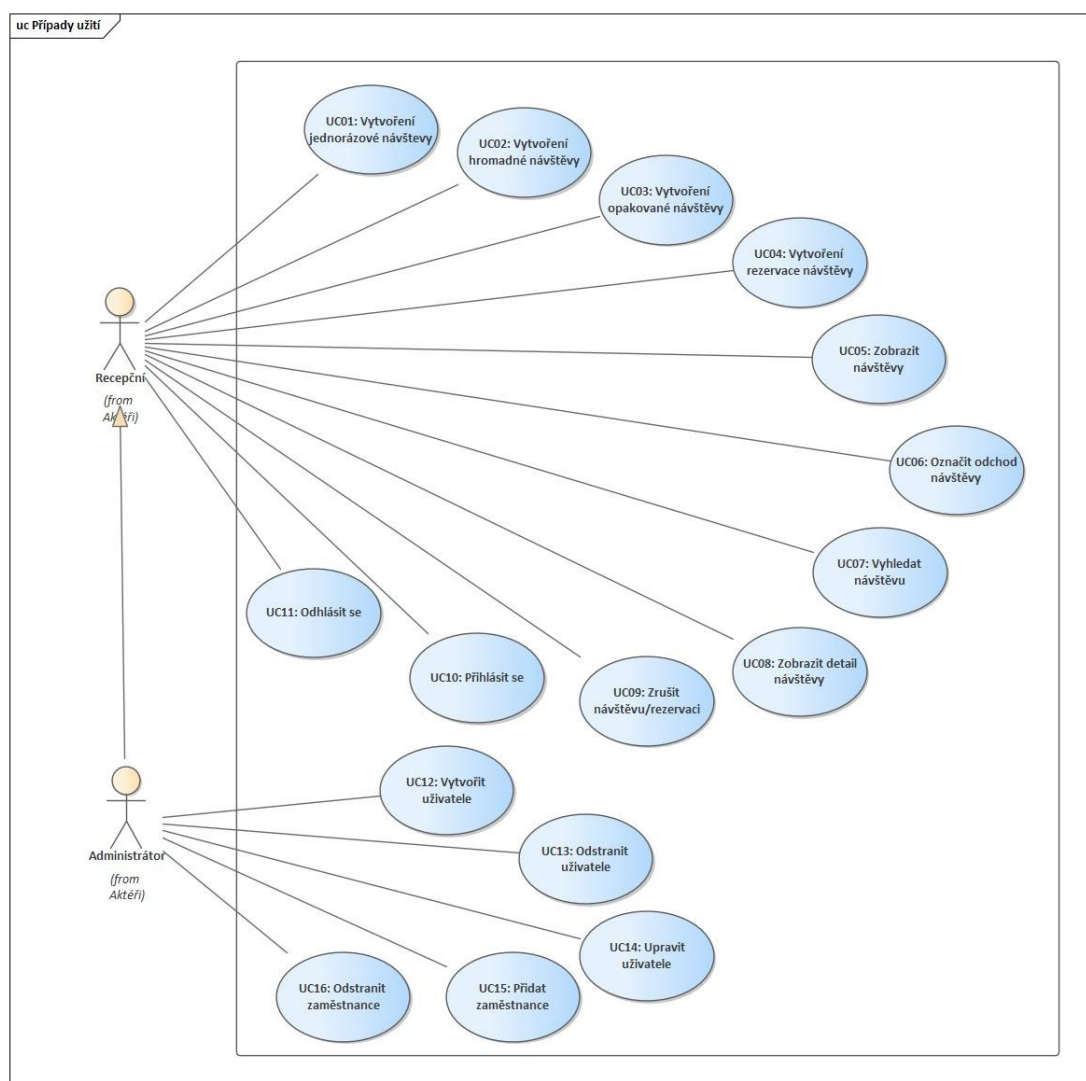
NFR05: Škálovatelnost

- Systém musí být navržen tak, aby zvládal rostoucí zátěž bez ztráty výkonu a bez nutnosti zásadních změn v architektuře.

7 MODEL PŘÍPADŮ UŽITÍ

Model případů užití představuje klíčovou část návrhu systému pro evidenci návštěv ve firmě. Na základě dříve definovaných funkčních požadavků byl vytvořen model, který systematicky popisuje interakce mezi jednotlivými aktéry a systémem. Tento model byl vytvořen s využitím jazyka UML a vizualizuje, jakým způsobem uživatelé (aktéři) využívají jednotlivé funkcionality systému – tzv. případy užití.

Cílem této kapitoly je představit aktéry systému a popsat jednotlivé případy užití. Popsané scénáře pokrývají veškeré funkce navrhované aplikace a byly detailně rozpracovány v návaznosti na definované funkční požadavky. Výsledný model slouží jako základ pro následnou implementaci.



Obrázek 5: Model případu užití

(zdroj: vlastní)

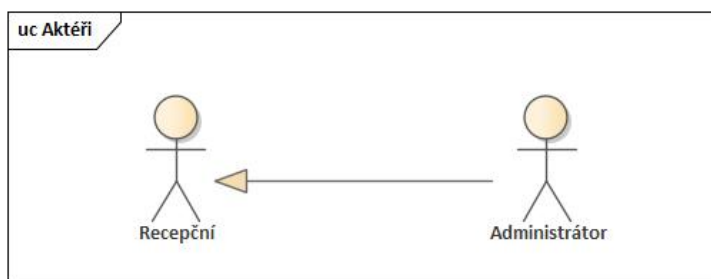
7.1 Aktéři

Systém evidence návštěv byl navržen s ohledem na jednoduchost, přehlednost a jednoznačné rozdělení rolí mezi uživateli. V systému vystupují dva hlavní typy aktérů – Recepční a Administrátor. Oba tito aktéři mají specifické role, přičemž jejich činnosti jsou jasně odděleny na základě úrovně oprávnění a zodpovědnosti.

Recepční je primární aktér systému. Jeho hlavní úlohou je každodenní práce se záznamy návštěv. Mezi jeho činnosti patří přímá registrace návštěv (jednorázových, hromadných i opakovaných), vytváření rezervací, správa příchodů a odchodů návštěvníků, stejně jako zobrazení, vyhledávání a rušení návštěv a celkový přehled nad aktuálním stavem návštěv v systému. Každý recepční má svůj vlastní uživatelský účet, pod kterým se do systému přihlašuje. Systém zaznamenává, který konkrétní recepční provedl, jakou operaci (například kdo založil určitou návštěvu). Tato role nevyžaduje hlubší správu systému, ale je zásadní pro každodenní provoz a interakci se systémem.

Administrátor má na starosti systémovou správu. Může přidávat, upravovat a odstraňovat uživatele a zaměstnance, nastavuje jejich oprávnění a zajišťuje kontrolu nad přístupem do systému. Stejně jako Recepční, může i Administrátor vykonávat operativní činnosti spojené se záznamem návštěv.

Model případu užití nepočítá s žádnými dalšími typy aktérů, což odpovídá požadavku na jednoduchost systému a jeho cílovému využití v rámci jedné firmy. Přesto je v návrhu zohledněna také role Zaměstnanec, a to kvůli požadavku na automatické notifikace o příchodu návštěv. Zaměstnanec však není interaktivním aktérem systému, ale pouze evidovanou entitou, ke které se návštěva vztahuje. Pokud by v budoucnu došlo k rozšíření funkcionality, lze tuto roli snadno rozšířit na plnohodnotného aktéra s omezeným přístupem k systému.



Obrázek 6: Aktéři

(zdroj: vlastní)

7.2 Případy užití

Případy užití pokrývají kompletní sadu operací, které může uživatel v systému provádět. Každý případ užití odpovídá konkrétní funkčním požadavkům (FR01–FR10) a zachycuje interakce mezi aktérem a systémem. Pro účely této práce byly případy užití rozděleny do několika oblastí – správa návštěv, vyhledávání a zobrazení, uživatelská správa a přihlášení/odhlášení.

Každý případ užití je dokumentován formou tabulky, která obsahuje jeho ID, název, charakteristiku, primární a vedlejší aktéry, vstupní a výstupní podmínky a podrobný popis hlavního scénáře. V případě potřeby jsou také uvedeny alternativní scénáře, které rozšiřují nebo modifikují chování systému podle konkrétních situací.

Základní případy užití zahrnují:

- UC01–UC04:
 - Registrace jednorázové, hromadné a opakované návštěvy, včetně rezervace.
- UC05–UC09:
 - Zobrazení, vyhledávání, detail a rušení návštěv a označení odchodu.
- UC10–UC11:
 - Přihlášení a odhlášení uživatele, včetně alternativních scénářů.
- UC12–UC14:
 - Tvorba, úprava a odstranění uživatele.
- UC15–UC16:
 - Přidání a odebrání zaměstnance.

Všechny případy užití byly zaneseny do modelu a vizualizovány prostřednictvím příslušného diagramu. Tento diagram vizualizuje interakce mezi aktéry a systémem a zachycuje jejich vzájemné vazby.

7.2.1 Vytvoření jednorázové návštěvy

Tabulka 1: Základní scénář – Vytvoření jednorázové návštěvy

Název: Vytvoření jednorázové návštěvy		
ID: UC01		
Charakteristika: Recepční registruje nově příchozí individuální návštěvu do systému.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínka: Recepční je přihlášený do systému.		
Výstupní podmínka: Úspěšné vytvoření jednorázové návštěvy.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Vytvoření jednorázové návštěvy.
2	Systém	Systém zobrazí formulář pro zadání údajů o návštěvě.
3	Recepční	Recepční vyplní údaje o návštěvníkovi: Jméno, Příjmení, Důvod návštěvy, Čas příchodu, Zaměstnanec, Poznámka, Seznam vybavení.
4	Recepční	Recepční potvrdí vytvoření návštěvy.
5	Systém	Systém uloží návštěvu do databáze.
6	Systém	Systém automaticky odešle notifikaci zaměstnanci o příchodu návštěvy.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.2 Vytvoření hromadné návštěvy

Tabulka 2: Základní scénář – Vytvoření hromadné návštěvy

Název: Vytvoření hromadné návštěvy		
ID: UC02		
Charakteristika: Recepční registruje nově příchozí hromadnou návštěvu do systému.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínka: Recepční je přihlášený do systému.		
Výstupní podmínka: Úspěšné vytvoření hromadné návštěvy.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Vytvoření hromadné návštěvy.
2	Systém	Systém zobrazí formulář pro zadání údajů o návštěvě.
3	Systém	Systém umožní zadat seznam návštěvníků.
4	Recepční	Recepční vyplní údaje o hromadné návštěvě: Důvod návštěvy, Čas příchodu, Zaměstnanec, Poznámka, Seznam vybavení.
5	Recepční	Recepční potvrdí vytvoření návštěvy.
6	Systém	Systém uloží návštěvu do databáze.
7	Systém	Systém automaticky odešle notifikaci zaměstnanci o příchodu návštěvy.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.3 Vytvoření opakované návštěvy

Tabulka 3: Základní scénář – Vytvoření opakované návštěvy

Název: Vytvoření opakované návštěvy		
ID: UC03		
Charakteristika: Recepční registruje nově příchozí individuální návštěvu do systému, která bude opakovaně navštěvovat firmu.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínka: Recepční je přihlášený do systému.		
Výstupní podmínka: Úspěšné vytvoření opakované návštěvy.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Vytvoření opakované návštěvy.
2	Systém	Systém zobrazí formulář pro zadání údajů o návštěvě.
3	Recepční	Recepční vyplní údaje o návštěvníkovi: Jméno, Příjmení, Důvod návštěvy, Čas příchodu, Zaměstnanec, Poznámka, Seznam vybavení.
4	Recepční	Recepční vybere frekvenci opakování: Denně, Týdně, Měsíčně, Vlastní.
5	Recepční	V případě, že Recepční zvolí frekvenci opakování: Vlastní:
6	Systém	Systém zobrazí kalendář pro výběr konkrétních datumů.
7	Recepční	Recepční vybere požadované datumy.
8	Recepční	Recepční potvrdí vytvoření návštěvy.
9	Systém	Systém uloží návštěvu do databáze.
10	Systém	V případě, že Recepční zvolil frekvenci opakování: Vlastní:
11	Systém	Systém vytvoří a uloží odpovídající rezervace.
12	Systém	Systém automaticky odešle notifikaci zaměstnanci o příchodu návštěvy.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.4 Vytvoření rezervace návštěvy

Tabulka 4: Základní scénář – Vytvoření rezervace návštěvy

Název: Vytvoření rezervace návštěvy		
ID: UC04		
Charakteristika: Recepční předem rezervuje novou návštěvu do systému.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínka: Recepční je přihlášený do systému.		
Výstupní podmínka: Úspěšné vytvoření rezervace.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Vytvoření rezervace návštěvy.
2	Systém	Systém zobrazí formulář pro zadání údajů o návštěvě.
3	Recepční	Recepční vyplní údaje o návštěvníkovi: Jméno, Příjmení, Důvod návštěvy, Zaměstnanec, Datum.
4	Recepční	Recepční potvrdí vytvoření rezervace.
5	Systém	Systém uloží rezervaci do databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.5 Zobrazit návštěvy

Tabulka 5: Základní scénář – Zobrazit návštěvy

Název: Zobrazit návštěvy		
ID: UC05		
Charakteristika: Recepční zobrazí seznam všech aktuálních, plánovaných nebo historii návštěv.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínka: Recepční je přihlášený do systému.		
Výstupní podmínka: Zobrazení seznamu aktuálních, plánovaných nebo historie návštěv.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Zobrazení návštěv.
2	Systém	Systém poskytne výběr: Aktuální/Plánované/Historie.
3	Recepční	V případě, že Recepční zvolí možnost Aktuální návštěvy:
4	Systém	Systém zobrazí přehled Aktuálních návštěv.
5	Recepční	V případě, že Recepční zvolí možnost Plánované návštěvy (rezervace):
6	Systém	Systém zobrazí přehled Plánovaných návštěv.
7	Recepční	V případě, že Recepční zvolí možnost Historie návštěv:
8	Systém	Systém zobrazí přehled Historie návštěv.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.6 Označit odchod návštěvy

Tabulka 6: Základní scénář – Označit odchod návštěvy

Název: Označit odchod návštěvy		
ID: UC06		
Charakteristika: V rámci zobrazení aktuálních návštěv přítomných v budově Recepční zaznamenává čas odchodu.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční je přihlášený do systému. Recepční má zobrazený seznam aktuálních návštěv.		
Výstupní podmínka: Úspěšné zaznamenání času odchodu a v případě Opakované návštěvy (mimo typ „Vlastní“) vytvoření odpovídající rezervace.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční u konkrétní návštěvy zvolí možnost odchodu.
2	Systém	Systém zaznamená čas odchodu do databáze.
3	Systém	Systém zkontroluje, o jaký typ návštěvy se jedná.
4	Systém	V případě, že se jedná o Opakovanou návštěvu (mimo typ „Vlastní“):
5	Systém	Systém převezme základní informace ze současné návštěvy.
6	Systém	Systém na základě převzatých dat založí novou rezervaci a nastaví datum dle uložených parametrů inkrement a jednotka opakování.
7	Systém	Systém uloží rezervaci do databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.7 Vyhledat návštěvu

Tabulka 7: Základní scénář – Vyhledat návštěvu

Název: Vyhledat návštěvu		
ID: UC07		
Charakteristika: Recepční vyhledává konkrétní návštěvu z databáze podle zadaných parametrů.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční je přihlášený do systému. Recepční má zobrazený seznam aktuálních/plánovaných/historii návštěv.		
Výstupní podmínka: Úspěšné vyhledání návštěvy dle zadaných parametrů.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční zahájí proces Vyhledat návštěvu.
2	Systém	Systém umožní vyhledat záznamy podle jména návštěvy, zaměstnance nebo data.
3	Recepční	Recepční zadá hledaný výraz podle kritérií.
4	Systém	Systém zobrazí odpovídající záznamy.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.8 Zobrazit detail návštěvy

Tabulka 8: Základní scénář – Zobrazit detail návštěvy

Název: Zobrazit detail návštěvy		
ID: UC08		
Charakteristika: Recepční zobrazí detaily konkrétní návštěvy z databáze.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční je přihlášený do systému. Recepční má zobrazený seznam aktuálních/plánovaných/historii návštěv.		
Výstupní podmínka: Zobrazení detailů u zvolené návštěvy.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční klikne na konkrétní návštěvu.
2	Systém	Systém zobrazí detaily návštěvy.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.9 Zrušit návštěvu/rezervaci

Tabulka 9: Základní scénář – Zrušit návštěvu/rezervaci

Název: Zrušit návštěvu/rezervaci		
ID: UC09		
Charakteristika: Recepční odstraní konkrétní návštěvu/rezervaci z databáze.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční je přihlášený do systému. Recepční má zobrazený detail u konkrétní návštěvy/rezervace.		
Výstupní podmínka: Úspěšné vymazání záznamu z databáze.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční u vybrané návštěvy/rezervace klikne na tlačítko „Smazat“.
2	Systém	Systém odstraní návštěvu/rezervaci z databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.10 Přihlásit se

Tabulka 10: Základní scénář – Přihlásit se

Název: Přihlásit se		
ID: UC10		
Charakteristika: Recepční se přihlásí do systému.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční má spuštěnou aplikaci. Recepční má vytvořené uživatelské jméno a heslo od administrátora.		
Výstupní podmínka: Recepční je úspěšně přihlášen do systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční otevře přihlašovací stránku systému.
2	Systém	Systém zobrazí přihlašovací formulář se dvěma poli: Uživatelské jméno, Heslo.
3	Recepční	Recepční zadá své přihlašovací údaje.
4	Recepční	Recepční klikne na tlačítko „Přihlásit se“.
5	Systém	Systém ověří zadané údaje.
6	Systém	Systém umožní přístup do aplikace a zobrazí hlavní stránku.
Alternativní scénář: UC10a – Neplatné přihlašovací údaje		

(zdroj: vlastní)

Tabulka 11: Alternativní scénář – Neplatné přihlašovací údaje

Název: Neplatné přihlašovací údaje		
ID: UC10a		
Charakteristika: Alternativní scénář u kroku 6 Systém zamítne přístup v případě, že jsou přihlašovací údaje špatně zadané.		
Alternativní scénář:		
Krok	Aktér/Systém	Popis
1	Systém	Systém zjistí, že uživatelské jméno nebo heslo nejsou správné.
2	Systém	Systém zobrazí chybovou hlášku: „Neplatné přihlašovací údaje.“

(zdroj: vlastní)

7.2.11 Odhlásit se

Tabulka 12: Základní scénář – Odhlásit se

Název: Odhlásit se		
ID: UC11		
Charakteristika: Recepční se odhlašuje ze systému.		
Primární aktér: Recepční		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Recepční je přihlášen do systému. Recepční je na Hlavní stránce.		
Výstupní podmínka: Recepční je úspěšně odhlášen ze systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční klikne na tlačítko „Odhlásit se“.
2	Systém	Systém zobrazí potvrzovací dialog „Opravdu se chcete odhlásit?“.
3	Recepční	Recepční potvrdí odhlášení.
4	Systém	Systém ukončí aktivní relaci uživatele.
5	Systém	Systém přesměruje recepčního na přihlašovací stránku.
Alternativní scénář: UC11a – Nepotvrzení odhlášení		

(zdroj: vlastní)

Tabulka 13: Alternativní scénář – Nepotvrzené odhlášení

Název: Nepotvrzení odhlášení		
ID: UC11a		
Charakteristika: Alternativní scénář u kroku 3 Recepční nepotvrdí odhlášení a jeho účet zůstane aktivní na Hlavní stránce.		
Alternativní scénář:		
Krok	Aktér/Systém	Popis
1	Recepční	Recepční nepotvrdí odhlášení.
2	Systém	Systém neukončí aktivní relaci uživatele.
3	Recepční	Recepční zůstane aktivní na Hlavní stránce.

(zdroj: vlastní)

7.2.12 Vytvořit uživatele

Tabulka 14: Základní scénář – Vytvořit uživatele

Název: Vytvořit uživatele		
ID: UC12		
Charakteristika: Administrátor vytvoří nového uživatele systému.		
Primární aktér: Administrátor		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen v roli Administrátora. Uživatel se nachází na Hlavní stránce v sekci Správa uživatelů.		
Výstupní podmínka: Úspěšně vytvoření nového uživatele do systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor zahájí proces Vytvoření nového uživatele.
2	Systém	Systém zobrazí formulář pro vytvoření uživatele, kde administrátor vyplní: Jméno, Příjmení, E-mail, Uživatelské jméno, Heslo, Role.
3	Administrátor	Administrátor potvrdí vytvoření uživatele kliknutím na tlačítko „Uložit“.
4	Systém	Systém uloží nového uživatele do databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.13 Odstranit uživatele

Tabulka 15: Základní scénář – Odstranit uživatele

Název: Odstranit uživatele		
ID: UC13		
Charakteristika: Administrátor odstraní již existujícího uživatele ze systému.		
Primární aktér: Administrátor		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen v roli Administrátora. Uživatel se nachází na Hlavní stránce v sekci Správa uživatelů.		
Výstupní podmínka: Úspěšné odstranění uživatele ze systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor zahájí proces Odstranění uživatele.
2	Systém	Systém zobrazí seznam uživatelů.
3	Administrátor	Administrátor vybere konkrétního uživatele ke smazání.
4	Systém	Systém zobrazí potvrzovací dialog, kde administrátor musí akci potvrdit.
5	Administrátor	Administrátor potvrdí smazání.
6	Systém	Systém odstraní uživatelský účet z databáze.
Alternativní scénář: Nemá		

(zdroj: vlastní)

7.2.14 Upravit uživatele

Tabulka 16: Základní scénář – Upravit uživatele

Název: Upravit uživatele		
ID: UC14		
Charakteristika: Administrátor upravuje údaje o existujícím uživateli systému.		
Primární aktér: Administrátor		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen v roli Administrátora. Uživatel se nachází na Hlavní stránce v sekci Správa uživatelů.		
Výstupní podmínka: Úspěšné aktualizování údajů o uživateli.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor otevře seznam uživatelů.
2	Administrátor	Administrátor zvolí konkrétního uživatele, jehož údaje chce upravit.
3	System	System zobrazí formulář s aktuálními údaji vybraného uživatele.
4	Administrátor	Administrátor provede požadované změny.
5	Administrátor	Administrátor potvrdí úpravu kliknutím na tlačítko „Uložit“.
6	System	System ověří správnost zadaných údajů a uloží změny.
7	System	System zobrazí potvrzení o úspěšném uložení změn.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.15 Přidat zaměstnance

Tabulka 17: Základní scénář – Přidat zaměstnance

Název: Přidat zaměstnance		
ID: UC15		
Charakteristika: Administrátor přidá nového zaměstnance do systému.		
Primární aktér: Administrátor		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen v roli Administrátora. Uživatel se nachází na Hlavní stránce v sekci Správa zaměstnanců.		
Výstupní podmínka: Úspěšné přidání nového zaměstnance do systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor zahájí proces Přidání nového zaměstnance.
2	Systém	Systém zobrazí formulář pro přidání zaměstnance, kde administrátor vyplní: Jméno, Příjmení, E-mail, Telefon, Kancelář.
3	Administrátor	Administrátor potvrdí přidání zaměstnance kliknutím na tlačítko „Uložit“.
4	Systém	Systém uloží nového zaměstnance do databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.2.16 Odstranit zaměstnance

Tabulka 18: Základní scénář – Odstranit zaměstnance

Název: Odstranit zaměstnance		
ID: UC16		
Charakteristika: Administrátor odstraní již existujícího zaměstnance ze systému.		
Primární aktér: Administrátor		
Vedlejší aktéři: Nejsou		
Vstupní podmínky: Uživatel je přihlášen v roli Administrátora. Uživatel se nachází na Hlavní stránce v sekci Správa zaměstnanců.		
Výstupní podmínka: Úspěšné odstranění zaměstnance ze systému.		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Administrátor	Administrátor zahájí proces Odstranění zaměstnance.
2	Systém	Systém zobrazí seznam zaměstnanců.
3	Administrátor	Administrátor vybere konkrétního zaměstnance ke smazání.
4	Systém	Systém zobrazí potvrzovací dialog, kde administrátor musí akci potvrdit.
5	Administrátor	Administrátor potvrdí smazání.
6	Systém	Systém odstraní zaměstnanecký účet z databáze.
Alternativní scénář: Není		

(zdroj: vlastní)

7.3 Matice sledovatelnosti požadavků

Následující matice znázorňuje vztah mezi funkčními požadavky a případy užití systému. Každý řádek odpovídá jednomu funkčnímu požadavku (FR), zatímco sloupce představují jednotlivé případy užití (UC). Vzájemná propojenost je vyjádřena pomocí relace typu „Realize“, která označuje, že daný případ užití realizuje konkrétní požadavek.

Tato matice slouží k zajištění trasovatelnosti a úplnosti návrhu systému. Je tak možné snadno ověřit, zda jsou všechny funkční požadavky pokryty odpovídajícími funkcemi systému, případně identifikovat požadavky, které dosud nemají přiřazený případ užití.

Vztah mezi požadavky a případy užití byl modelován pomocí nástroje Relationship Matrix. Z této matice byl následně vytvořen export do formátu PNG.

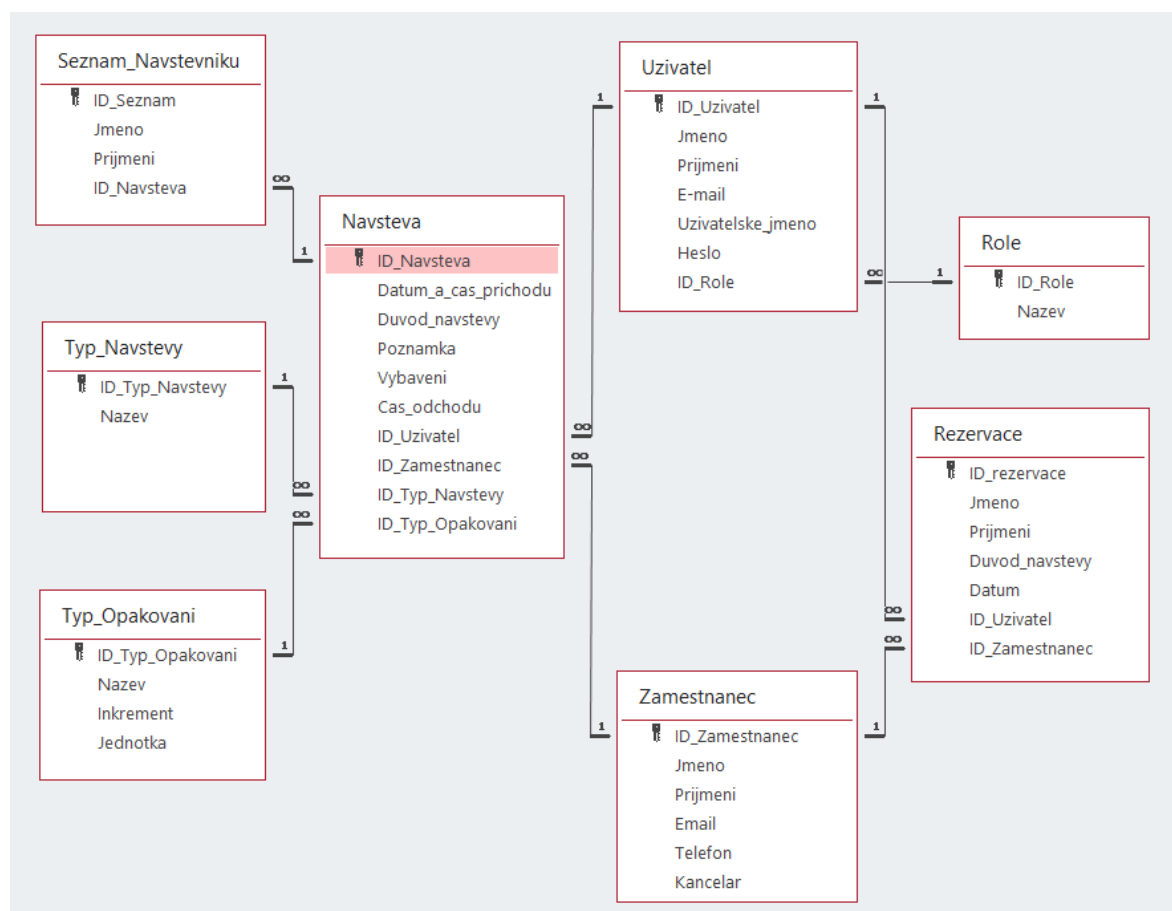
Target Source	Případy užití::UC01: Vytvoření jednorázové návštěvy	Případy užití::UC02: Vytvoření hromadné návštěvy	Případy užití::UC03: Vytvoření opakované návštěvy	Případy užití::UC04: Vytvoření rezervace návštěvy	Případy užití::UC05: Zobrazení návštěvy	Případy užití::UC06: Označit odchod návštěvy	Případy užití::UC07: Vyhledat návštěvy	Případy užití::UC08: Zobrazení detail návštěvy	Případy užití::UC09: Zrušit návštěvu/rezervaci	Případy užití::UC10: Přihlásit se	Případy užití::UC11: Odhlásit se	Případy užití::UC12: Vytvořit uživatele	Případy užití::UC13: Odstranit uživatele	Případy užití::UC14: Upravit uživatele	Případy užití::UC15: Přidat zaměstnance	Případy užití::UC16: Odstranit zaměstnance
Správa návštěv::FR01: Registrace jednorázové návštěvy	↑															
Správa návštěv::FR02: Registrace hromadné návštěvy		↑														
Správa návštěv::FR03: Registrace opakované návštěvy			↑													
Správa návštěv::FR04: Rezervace návštěvy				↑												
Správa návštěv::FR05: Zobrazení návštěv					↑											
Správa návštěv::FR06: Označení odchodu						↑										
Správa návštěv::FR07: Vyhledávání návštěv							↑									
Správa návštěv::FR08: Zobrazení detailů návštěvy								↑								
Správa návštěv::FR09: Zrušení návštěvy/rezervace									↑							
Správa návštěv::FR10: Notifikace zaměstnance	↑	↑	↑	↑												
Správa uživatelů::FR11: Přihlášení uživatele										↑						
Správa uživatelů::FR12: Odhlášení uživatele											↑					
Správa uživatelů::FR13: Přidání nového uživatele												↑				
Správa uživatelů::FR14: Odstranění uživatele													↑			
Správa uživatelů::FR15: Upravení uživatele														↑		
Správa uživatelů::FR16: Přidání zaměstnance															↑	
Správa uživatelů::FR17: Odstranění zaměstnance																↑

Obrázek 7: Matice sledovatelnosti požadavků

(zdroj: vlastní)

8 NÁVRH DATABÁZE

Návrh databáze tvoří nedílnou součást aplikace pro evidenci návštěv ve firmě, neboť veškerá data zadávaná uživateli (např. informace o návštěvách, rezervacích nebo uživatelských účtech) musí být bezpečně a přehledně uložena. Vytvořený databázový model v prostředí Microsoft Access byl navržen s důrazem na jednoduchost, přehlednost a logické oddělení jednotlivých entit. Návrh zároveň odpovídá definovaným funkčním požadavkům systému.



Obrázek 8: ERD diagram

(zdroj: vlastní)

8.1 Popis databázového modelu

Návrh databáze se skládá z osmi hlavních tabulek: Navsteva, Rezervace, Uzivatel, Role, Zamestnanec, Typ_Navstevy, Typ_Opakovani a Seznam_Navstevniku. Tyto tabulky jsou mezi sebou vzájemně propojeny pomocí relací typu 1:N, přičemž každá tabulka má přesně definované atributy sloužící k uchovávání informací, které aplikace využívá.

Hlavní tabulkou je Navsteva, která slouží k evidenci všech uskutečněných návštěv. Obsahuje základní informace o návštěvě, jako je datum a čas příchodu, důvod návštěvy, případné poznámky, vybavení a čas odchodu. Je spojena s tabulkou Seznam_Navstevniku, což umožňuje zaznamenat jména a příjmení všech účastněných osob v rámci návštěvy. Záznamy v této tabulce jsou také propojeny s uživatelem (ID_Uzivatel), který návštěvu vytvořil, a se zaměstnancem (ID_Zamestnanec), ke kterému návštěva směřuje. Dále je návštěva specifikována typem návštěvy (ID_Typ_Navstevy) a v případě opakovaných návštěv také frekvencí (ID_Typ_Opakovani).

Tabulka Rezervace slouží k předběžné evidenci budoucích návštěv. Uchovává údaje jako je jméno a příjmení návštěvníka, důvod návštěvy a místo konkrétního času příchodu obsahuje datum plánovaného příchodu. Rezervace je rovněž propojena s uživatelem a zaměstnancem. Tato tabulka umožňuje správu plánovaných návštěv a je klíčová pro realizaci případů užití týkajících se rezervací (UC04: Vytvoření rezervace, UC09: Zrušit rezervaci).

Tabulka Uzivatel obsahuje záznamy o všech uživateli systému – tedy recepčních a administrátorech. Uchovává přihlašovací údaje, osobní data a referenci na roli uživatele (ID_Role). Každý recepční má svůj vlastní účet, což umožňuje identifikovat, kdo kterou návštěvu nebo rezervaci vytvořil. Tato vazba je využita v rámci systému i pro správu přístupových práv.

Tabulka Role definuje jednotlivé typy uživatelů systému. V rámci návrhu jsou rozlišeny dvě role: Recepční a Administrátor. Tato tabulka umožňuje snadno rozšířit systém o nové role, pokud by bylo potřeba upravit oprávnění v budoucnu.

Tabulka Zamestnanec uchovává informace o osobách, ke kterým návštěvy přicházejí. Tyto záznamy nejsou interaktivními aktéry systému, ale slouží jako reference v návštěvách a rezervacích. Obsahuje základní kontaktní údaje jako jméno, příjmení, e-mail, telefon a kancelář. Jejich přítomnost v systému je zásadní zejména kvůli automatickým notifikacím o příchodu návštěv (FR10).

Tabulka `Typ_Navstevy` definuje klasifikaci návštěv podle jejich charakteru: jednorázová, hromadná, opakovaná. Slouží jako referenční číselník pro tabulku `Navsteva`.

Tabulka `Typ_Opakovani` je používána výhradně u opakovaných návštěv a specifikuje způsob jejich opakování pomocí atributů `Inkrement` a `Jednotka` (např. 1x týdně, 2x měsíčně). Tento typ návštěvy je řízen případem užití UC03: Vytvoření opakované návštěvy.

Tabulka `Seznam_Navstevniku` slouží ke správě osob, které jsou součástí registrace návštěvy. Každý záznam obsahuje jméno a příjmení návštěvníka a je navázán na konkrétní návštěvu přes `ID_Navsteva`. Tato tabulka je aktivní při realizaci případů UC01–UC03.

8.2 Relace mezi tabulkami

Databázový model je navržen tak, aby zajistil jednoznačné vazby mezi entitami a zároveň minimalizoval redundanci dat. Všechny důležité relace jsou realizovány pomocí primárních a cizích klíčů.

Na ERD diagramu je znázorněno, jak jednotlivé tabulky spolu souvisejí:

Tabulka `Navsteva` je ve vztahu M:1 s tabulkami `Uzivatel`, `Zamestnanec`, `Typ_Navstevy` a `Typ_Opakovani`. To znamená, že každá návštěva je vytvořena jedním uživatelem (recepčním), směřuje k jednomu zaměstnanci a má přidělený typ i opakovací vzor. Uživatel může vytvořit více návštěv, ale každá návštěva je přiřazena právě jednomu uživateli.

Tabulka `Rezervace` má obdobné relace jako `Navsteva` – M:1 k `Uzivatel` a `Zamestnanec`. Zajišťuje, že každá rezervace má jednoho autora a příjemce návštěvy.

Tabulka `Uzivatel` je v relaci M:1 s tabulkou `Role`, což umožňuje každému uživateli přidělit právě jednu roli (např. Recepční nebo Administrátor).

Tabulka `Seznam_Navstevniku` je ve vztahu M:1 k `Navsteva`. Každý záznam o návštěvníkovi je přiřazen právě jedné návštěvě, přičemž jedna návštěva může obsahovat více osob (typicky u hromadných návštěv).

8.3 Vztah k aplikaci

Tento model umožňuje bezpečné a přehledné ukládání dat o všech aktivitách v systému. Současně vytváří vhodný základ pro generování přehledů, vyhledávání návštěv podle různých kritérií, trasování historie a evidenci opakujících se událostí. Je navržen tak, aby odpovídal funkcionalitě systému popsané v jednotlivých případech užití a umožňoval další rozšíření systému bez zásahu do jeho základní struktury.

9 IMPLEMENTACE

Hlavní funkcionalitu desktopové aplikace pro evidenci návštěv ve firmě tvoří rozhraní navržené ve Windows Forms a trvale uložená data jsou spravována v relační databázi Microsoft Access. Aplikace je vytvořena v jazyce C# s využitím vývojového prostředí Visual Studio, přičemž celé řešení je samostatně spustitelné bez nutnosti instalace externích serverů nebo služeb.

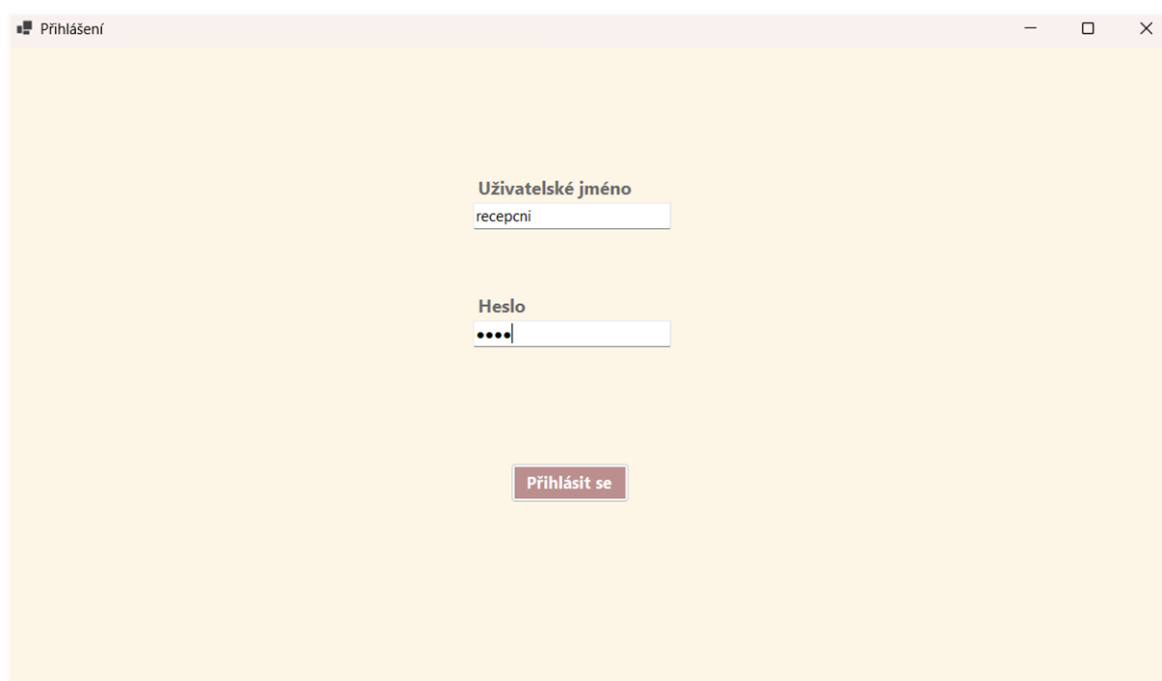
Uživatelská část aplikace je tvořena několika navzájem propojenými formuláři, z nichž každý má jasně definovanou funkci – od přihlášení uživatele, přes registraci nové návštěvy, až po správu a zobrazení detailů jednotlivých záznamů. Datová logika je řešena pomocí komponenty System.Data.OleDb, která umožňuje čtení a zápis dat přímo do databázového souboru (.accdb).

Celý návrh klade důraz na jednoduchost, přehlednost a funkčnost, což odpovídá požadavkům běžného provozu recepce v menší nebo středně velké firmě. Vzhled i chování aplikace bylo vytvořeno s ohledem na uživatelskou přívětivost, logické rozložení prvků a rychlý přístup k potřebným funkcím.

9.1 Přihlašovací stránka

Formulář LoginForm slouží jako vstupní brána do desktopové aplikace. Jeho hlavní funkcí je autentizace uživatele před zpřístupněním ostatních částí systému. Přihlašovací okno se zobrazuje jako první při spuštění aplikace a umožňuje bezpečný přístup pouze autorizovaným uživatelům.

Cílem tohoto formuláře je jednoduché a rychlé přihlášení, které zároveň zajišťuje základní úroveň ochrany před neoprávněným přístupem. Po zadání správného uživatelského jména a hesla je uživatel přesměrován do hlavního rozhraní aplikace (MainForm).



Obrázek 9: LoginForm

(zdroj: vlastní)

9.1.1 Popis funkcionality

Formulář obsahuje dva vstupní prvky typu TextBox, do kterých uživatel zadává:

- Uživatelské jméno (txtUsername)
- Heslo (txtPassword)

Pole pro heslo má aktivovanou vlastnost UseSystemPasswordChar, což znamená, že při psaní nejsou znaky viditelné – zvyšuje to důvěrnost přihlašovacích údajů.

Po kliknutí na tlačítko „Přihlásit se“ (btnLogin) dojde k vyhodnocení údajů. Kód v metodě btnLogin_Click kontroluje, zda jsou zadané hodnoty přesné.

Pokud přihlašovací údaje odpovídají, je vytvořena instance MainForm, které je předáno jméno přihlášeného uživatele jako parametr. Nové okno se zobrazí metodou .Show() a přihlašovací formulář se skryje metodou .Hide() – uživatel tak plynule přechází do hlavního menu.

V případě chybného jména nebo hesla se zobrazí chybová hláška pomocí MessageBox, která uživatele upozorní na neplatné přihlašovací údaje. Tím je zajištěno základní ověření a zpětná vazba pro uživatele.

9.1.2 Popis uživatelského rozhraní

Vizuálně je přihlašovací formulář navržen jako jednoduché, přehledné a centrálně zarovnané rozhraní. Rozměry formuláře jsou pevně nastaveny na 1200 x 700 pixelů a okno je zarovnáno na střed obrazovky (StartPosition = CenterScreen). Formulář má nastavený styl FormBorderStyle = FixedSingle, což znamená, že uživatel nemůže měnit jeho velikost.

Barevná paleta je laděna do odstínu OldLace (světle krémové), který tvoří klidné a neutrální pozadí. Použité fonty jsou Segoe UI, velikost 10.8 pt, s Bold stylem pro nadpisy polí.

Hlavní ovládací prvek – tlačítko „Přihlásit se“ – je zarovnáno do spodní části formuláře, zvýrazněno růžovo-hnědým odstínem RosyBrown, bílým písmem a stylizací s ohraničením (FlatStyle = Standard). Tím se vizuálně odděluje jako akční prvek pro odeslání údajů.

Popisky pro uživatelské jméno a heslo jsou zarovnány nad vstupními poli a vizuálně odlišené pomocí tmavšího odstínu (Color.DimGray) a tučného písma. Rozložení prvků je centrální, vertikálně vyvážené a navrženo pro rychlé a intuitivní použití.

9.1.3 Návaznost na další části systému

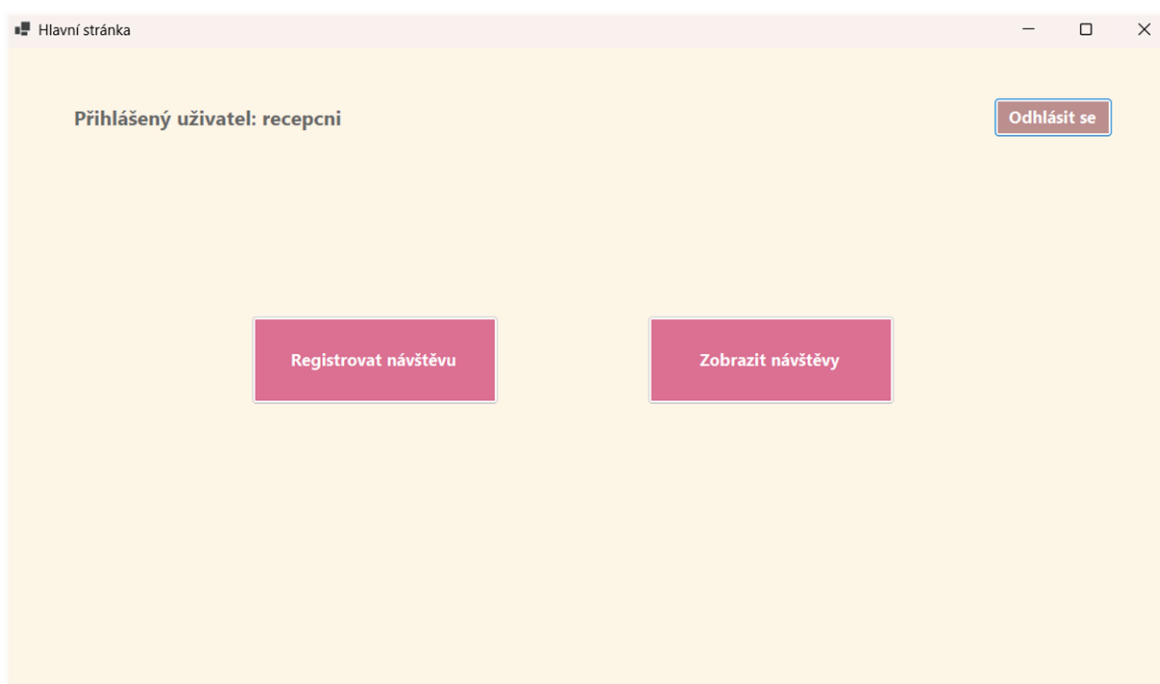
Po úspěšném přihlášení aplikace přesměrovává uživatele do MainForm, kde má uživatel k dispozici další možnosti práce se systémem. Přihlašovací formulář se tím stává vstupním bodem celého systému a zároveň jeho bezpečnostním filtrem.

Tato konstrukce je vhodná nejen pro praktické použití, ale i pro případné rozšíření – např. implementaci více rolí, ověřování z databáze nebo více uživatelských účtů v budoucnu.

9.2 Hlavní stránka

MainForm představuje centrální přehledové rozhraní aplikace, které je zobrazováno po úspěšném přihlášení uživatele. Slouží jako vstupní rozcestník pro navigaci k hlavním funkcím systému – registraci nové návštěvy a zobrazení aktuálně evidovaných návštěv.

Funguje jako základní pracovní plocha uživatele, kde má k dispozici dvě klíčová akční tlačítka. Kromě toho se zde zobrazuje přihlašovací identita uživatele a nachází se zde i možnost odhlášení z aplikace. Tímto formulářem začíná běžné denní používání systému v reálném provozu recepce.



Obrázek 10: MainForm

(zdroj: vlastní)

9.2.1 Popis funkcionality

MainForm je konstruován tak, aby umožnil uživateli přejít na dvě základní části systému – registraci návštěvy a přehled návštěv. Po spuštění je do formuláře z konstruktoru předáno přihlašovací jméno (uzivatelskeJmeno), které se uloží do privátní proměnné prihlasenyUzivatel.

V metodě MainForm_Load je následně přizpůsoben titulek okna (this.Text = "Hlavní stránka") a zobrazen text s označením přihlášeného uživatele v prvku lblUzivatel: „Přihlášený uživatel: recepni“.

9.2.2 Ovládací prvky a jejich funkce

Formulář obsahuje následující prvky:

- Label lblUzivatel
 - Textový prvek, který uživatele informuje o tom, kdo je přihlášen. Je zarovnán do levé části obrazovky a zvýrazněn tučným písmem v barvě DimGray.
- Tlačítko btnRegistrace
 - Po jeho stisknutí se spustí metoda btnRegistrace_Click, která otevře formulář RegisterVisitForm pro zadání nové návštěvy. Formulář se otevírá jako modální (ShowDialog()), čímž se zajistí, že uživatel nejprve dokončí akci, než se vrátí zpět na hlavní stránku.
- Tlačítko btnZobrazit
 - Spouští se metoda btnZobrazit_Click, která otevře VisitListForm – přehled aktuálních návštěv. I tento formulář je otevřen jako modální.
- Tlačítko btnOdhlásit
 - Kliknutím dojde k zavření MainForm a znovuotevření LoginForm, čímž je uživatel odhlášen z aplikace a vrácen zpět na přihlašovací obrazovku.

9.2.3 Popis uživatelského rozhraní

Vzhled odpovídá stylu zvolené barevné palety – béžový základ (BackColor = OldLace) a akcenty v odstínech růžové a hnědé.

Tlačítka btnRegistrace a btnZobrazit mají světlý růžový podklad (PaleVioletRed) a bílé písmo. Jsou výrazná, snadno dostupná a umístěna vodorovně na střed formuláře – btnRegistrace v levé polovině a btnZobrazit v pravé.

Tlačítko btnOdhlásit je umístěno v pravém horním rohu, má hnědorůžové pozadí (RosyBrown) a opět bílé písmo. Je lehce oddělené od ostatního obsahu, aby bylo jasné, že slouží k návratu zpět (odhlášení).

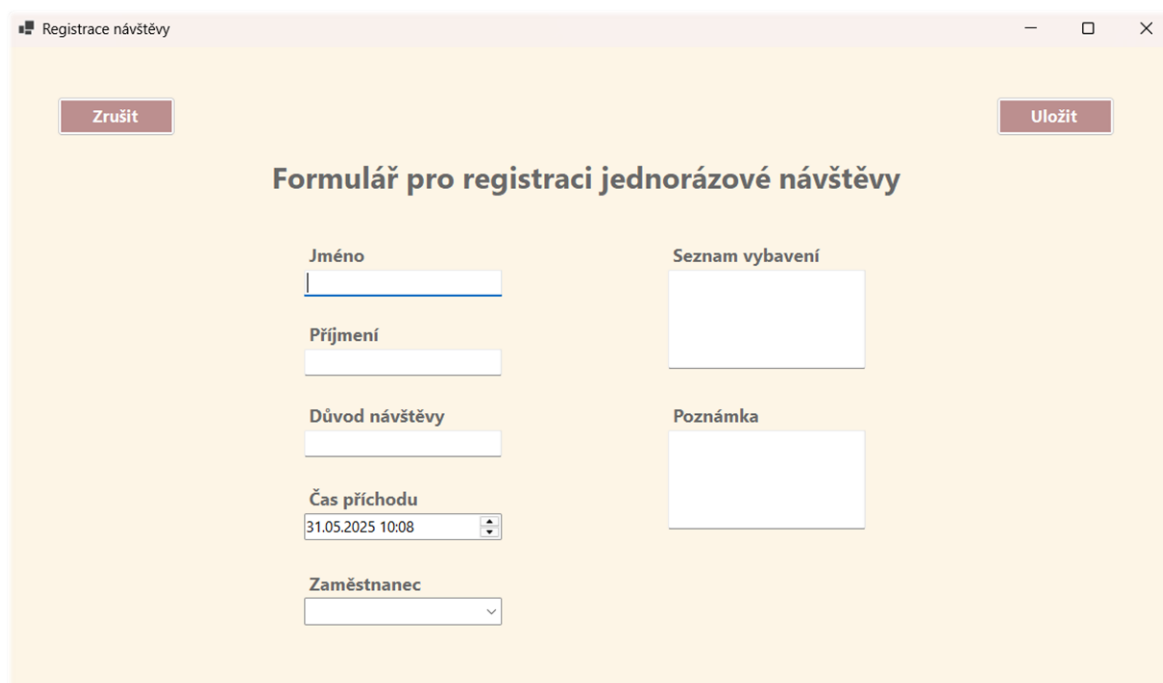
Formulář má pevnou velikost (FormBorderStyle = FixedSingle, Size = 1200 x 700 pixelů) a je centrován na obrazovce (StartPosition = CenterScreen).

Font všech prvků je Segoe UI, velikost 10.2 pt. Zvýraznění nadpisů (lblUzivatel) je řešeno pomocí větší velikosti písma (12 pt) a tučného stylu (Bold).

9.3 Registrace návštěvy

Formulář RegisterVisitForm představuje jeden z nejdůležitějších funkčních bloků celé aplikace. Jeho účelem je umožnit uživateli vytvořit nový záznam o návštěvě, která přichází do firmy. Formulář je navržen s důrazem na přehlednost, jednoduché ovládání a správu všech potřebných údajů o návštěvě. Uživatel zadává jméno, příjmení návštěvníka, důvod návštěvy, čas příchodu, jméno navštíveného zaměstnance a volitelné údaje jako poznámku a seznam vybavení.

Součástí tohoto formuláře je napojení na databázi Access, kde jsou informace o návštěvách ukládány prostřednictvím technologie OleDb. Celý proces je doprovázen validací vstupních dat a uživatelskou zpětnou vazbou.



Obrázek 11: RegisterVisitForm

(zdroj: vlastní)

9.3.1 Popis funkcionality

Formulář obsahuje dva hlavní akční prvky – tlačítko „Uložit“ a „Zrušit“. Tato tlačítka jsou umístěna v horní části obrazovky, vizuálně oddělená a barevně stylizovaná.

- Validace vstupních údajů:
 - Před samotným uložením dat dochází k důkladné validaci. Uživatel musí vyplnit: Jméno, Příjmení, Důvod návštěvy a Výběr zaměstnance.
 - Pokud některý z těchto údajů chybí, aplikace na to uživatele upozorní pomocí MessageBox s varovnou hláškou a proces uložení se přeruší.
- Ukládání do databáze:
 - Po úspěšné validaci se data zapisují do databázového souboru Navstevy.accdb, který je přiložen přímo k aplikaci. K databázi se připojuje přes OleDbConnection s využitím připojovacího řetězce Provider=Microsoft.ACE.OLEDB.12.0.
 - Data se ukládají do tabulky Navsteva, a to pomocí parametrizovaného SQL příkazu INSERT INTO. To zajišťuje bezpečnost proti SQL injection a správné zpracování různých datových typů. Pole jako poznámka nebo seznam vybavení jsou volitelná – pokud nejsou vyplněna, do databáze se ukládá DBNull.Value.
 - Po úspěšném uložení se zobrazí potvrzující zpráva o úspěšném vložení a formulář se automaticky uzavře.
- Zrušení registrace:
 - Tlačítko btnZrusit poskytuje uživateli možnost registraci zrušit. Po kliknutí se zobrazí potvrzovací dialog, zda chce uživatel opravdu ukončit registraci. Pokud ano, formulář se zavře bez uložení jakýchkoli údajů.

- Omezení výběru času:
 - Formulář obsahuje DateTimePicker pro zadání času příchodu. Pomocí nastavení MinDate a MaxDate je omezen výběr pouze na aktuální den, čímž je zajištěno, že není možné registrovat návštěvu zpětně ani na budoucí dny. Používá se formát dd.MM.yyyy HH:mm a je povoleno upravovat čas přes tzv. up/down ovladač.

9.3.2 Popis uživatelského rozhraní

Design formuláře RegisterVisitForm je uzpůsoben pro intuitivní používání s pozadím ve světlém odstínu OldLace.

Rozložení je dvousloupcové:

- V levém sloupci se nachází vstupní pole pro jméno, příjmení, důvod návštěvy, čas příchodu a výběr zaměstnance.
- V pravém sloupci se nachází větší víceřádková pole pro poznámku a seznam vybavení.

Všechny popisky (Label) jsou formátovány pomocí písma Segoe UI, 10.8 pt, Bold, a mají šedý odstín (DimGray). Tlačítka „Uložit“ a „Zrušit“ jsou navržena s barvou RosyBrown, bílým písmem a ohraničením (FlatStyle = Standard).

Na horní části formuláře se nachází nadpis „Formulář pro registraci jednorázové návštěvy“, který je zarovnán na střed a vizuálně vyvažuje celou obrazovku.

Formulář má pevnou velikost (1200 x 700 pixelů), umísťuje se vždy na střed obrazovky (StartPosition = CenterScreen) a jeho velikost nelze měnit kvůli omezení FormBorderStyle = FixedSingle.

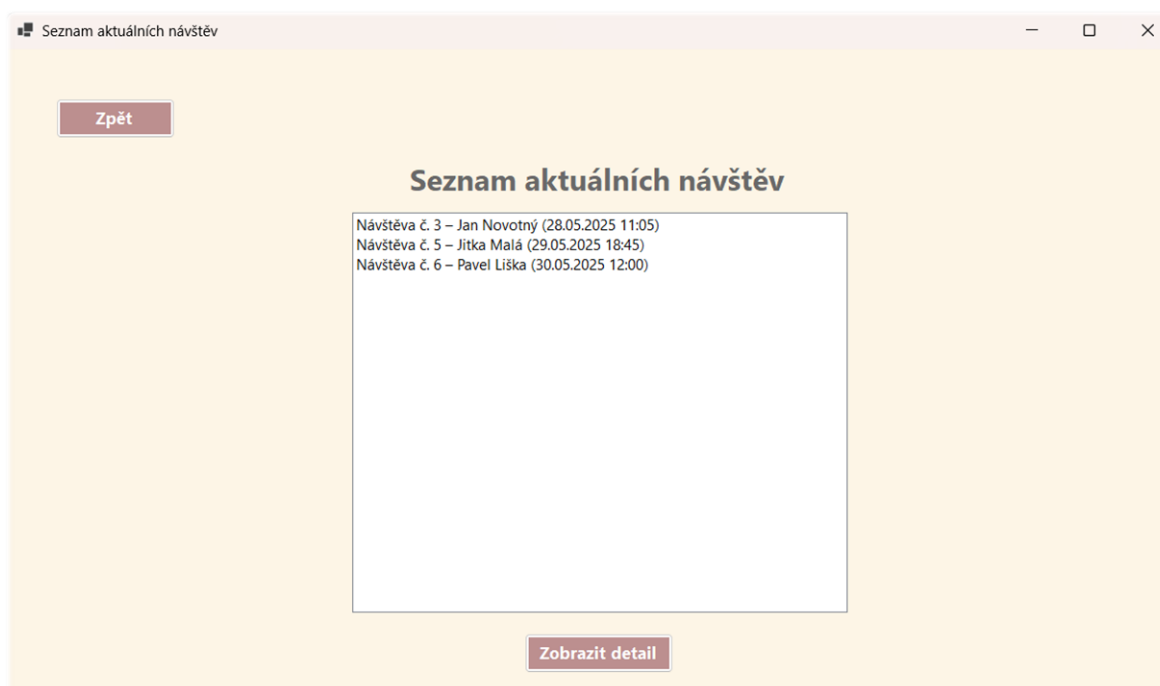
9.3.3 Návaznost na databázi

Data z tohoto formuláře se ukládají přímo do databáze Navstevy.accdb, konkrétně do tabulky Navsteva. Díky přímému spojení mezi rozhraním a databází dochází k uložení každé registrace okamžitě po potvrzení. Struktura této tabulky odpovídá jednotlivým vstupním polím formuláře: Jmeno (krátký text), Prijmeni (krátký text), Duvod_navstevy (krátký text), Cas_prichodu (datum a čas), Zamestnanec (krátký text), Seznam_vybaveni (dlouhý text) a Poznamka (dlouhý text).

9.4 Zobrazení návštěv

Formulář VisitListForm představuje seznamovou obrazovku, která uživateli zprostředkovává přístup k aktuálně evidovaným návštěvám ve firmě. Slouží jako prostředník mezi registrací a správou již existujících záznamů. Uživatel zde může přehledně procházet záznamy návštěv a následně zobrazit podrobnosti o konkrétní z nich.

Formulář zobrazuje výpis návštěv uložených v databázi a poskytuje uživateli náhled na aktuální evidenci. Jeho využití je ideální v každodenním provozu recepce, kde je potřeba rychle reagovat na dotazy, identifikovat příchozí osoby nebo potvrdit návštěvu.



Obrázek 12: VisitListForm

(zdroj: vlastní)

9.4.1 Popis funkcionality

- Načtení dat z databáze:
 - Hlavní funkcí formuláře je načíst všechna data z databázové tabulky Navsteva. To se provádí ihned po otevření formuláře ve výchozím konstruktoru, kde je volána metoda NacistData().
 - Databázové připojení je řešeno přes OleDbConnection, přičemž cesta k databázi je nastavena relativně (Navstevy.accdb), což umožňuje snadné nasazení aplikace na jiných počítačích bez potřeby pevné cesty.
 - Metoda NacistData() provádí SQL dotaz `SELECT * FROM Navsteva` a výsledky zpracovává pomocí OleDbDataReader. Pro každý záznam se vytvoří instance vnitřní třídy NavstevaZaznam, která uchovává všechny potřebné informace (jméno, příjmení, důvod, zaměstnanec, poznámka, vybavení, čas příchodu).
 - Textově zformátovaný řádek (Zobrazit detail) se přidává do ListBoxu (lstNavstevy), který představuje vizuální seznam všech aktuálních záznamů.
- Zobrazení detailu návštěvy:
 - Po výběru konkrétní položky v seznamu může uživatel kliknout na tlačítko „Zobrazit detail“. To spustí událost btnDetail_Click, která:
 - zjistí index vybrané položky
 - načte odpovídající objekt ze seznamu
 - otevře nový formulář VisitDetailForm, kterému předá veškeré detaily o zvolené návštěvě
 - Formulář VisitDetailForm se otevře jako modální (ShowDialog()), což znamená, že uživatel musí detail uzavřít, než se vrátí zpět k přehledu.
 - Pokud uživatel v detailu návštěvu smaže, formulář vrátí DialogResult.OK, a původní seznam je automaticky znovu načten pomocí volání NacistData(), čímž je zajištěna aktuálnost údajů.

- Tlačítko „Zpět“:
 - Tlačítko btnZpet jednoduše uzavírá VisitListForm a vrací uživatele zpět na MainForm. Neprovádí se žádné mazání ani zápis – formulář pouze ukončí svůj běh pomocí this.Close().

9.4.2 Popis uživatelského rozhraní

Styl formuláře je navržen tak, aby odpovídal barevnému ladění celé aplikace – pozadí je nastaveno na barvu OldLace. Formulář má pevně nastavenou velikost 1200 x 700 pixelů a je na obrazovce zarovnán na střed díky nastavení vlastnosti StartPosition = CenterScreen. Dále má formulář nastavenou vlastnost FormBorderStyle = FixedSingle, která znemožňuje uživateli měnit jeho velikost. Hlavní nadpis „Seznam aktuálních návštěv“ je umístěn ve středu v horní části a formátován pomocí fontu Segoe UI, 18 pt, Bold, s barvou DimGray.

- ListBox (lstNavstevy):
 - Centrálním prvkem formuláře je ListBox, ve kterém se zobrazují jednotlivé návštěvy v textové podobě – např. Návštěva č. 3 – Jan Novotný (28.05.2025 11:05).
 - Položky jsou zarovnány vertikálně a komponenta má dostatečnou velikost (500 × 404 px), aby bylo možné zobrazit více návštěv najednou. V návrhu je umístěna do středu obrazovky.
- Tlačítko „Zobrazit detail“:
 - Tlačítko je umístěno pod seznamem a je zarovnáno na střed. Barevně je sladěno do jemně růžového odstínu RosyBrown, s bílým písmem, tučným řezem (Bold) a ohraničením (FlatStyle = Standard). Slouží jako hlavní akční prvek pro interakci s daty.
- Tlačítko „Zpět“:
 - Umístěné vlevo nahoře, se stejným stylem jako ostatní tlačítka. Je záměrně oddělené, aby bylo jasné, že neprovádí změny, ale pouze ukončuje formulář.

9.5 Zobrazení detailu návštěvy

Formulář VisitDetailForm slouží k zobrazení všech údajů o konkrétní návštěvě, která již dříve byla zaregistrována v systému. Uživatel se k tomuto formuláři dostane skrze přehled návštěv (VisitListForm) výběrem konkrétní položky v seznamu a kliknutím na tlačítko „Zobrazit detail“.

Hlavními funkcemi formuláře jsou:

- Zobrazit všechny uložené informace o vybrané návštěvě.
- Umožnit smazání návštěvy z databáze.
- Poskytnout možnost návratu zpět bez jakéhokoli zásahu do dat.

Formulář je určen výhradně k pasivnímu náhledu a správě již existujících dat, nikoli k jejich úpravě.

The screenshot shows a web application window titled "Detaily návštěvy". It features a light yellow background and a white border. At the top left is a "Zpět" button, and at the top right is a "Smazat" button. The main heading is "Údaje o návštěvě". The form is organized into two columns. The left column contains fields for "Jméno" (Pavel), "Příjmení" (Liška), "Důvod návštěvy" (Příprava na konferenci), "Čas příchodu" (30.05.2025 12:00), and "Zaměstnanec" (Tomáš Malý). The right column contains a "Seznam vybavení" (Notebook, Dataprojektor) and a "Poznámka" (Od 12:00 do 14:00).

Údaje o návštěvě	
Jméno	Pavel
Příjmení	Liška
Důvod návštěvy	Příprava na konferenci
Čas příchodu	30.05.2025 12:00
Zaměstnanec	Tomáš Malý
Seznam vybavení	Notebook Dataprojektor
Poznámka	Od 12:00 do 14:00.

Obrázek 13: VisitDetailForm

(zdroj: vlastní)

9.5.1 Popis funkcionality

- Zobrazení dat:
 - Při spuštění formuláře je pomocí konstruktoru předáno sedm parametrů – odpovídajících jednotlivým sloupcům v databázi: ID záznamu (celé číslo), Jméno, Příjmení, Důvod návštěvy, Čas příchodu, Zaměstnanec, Seznam vybavení a Poznámka.
 - Všechny tyto hodnoty jsou zobrazeny ve formuláři v jednotlivých TextBox prvcích, které jsou nastaveny jako ReadOnly, aby nedocházelo k neúmyslnému přepisování. Zvláštní pozornost je věnována formátování času (dd.MM.yyyy HH:mm) pro zachování čitelnosti.
- Tlačítko „Zpět“:
 - Tlačítko btnZpet slouží k návratu zpět do seznamu návštěv. Neprovádí žádnou manipulaci s daty – pouze zavře aktuální formulář metodou this.Close().
- Tlačítko „Smazat“:
 - Tlačítko btnSmazat umožňuje uživateli odstranit z databáze konkrétní záznam. Po kliknutí na tlačítko je zobrazen potvrzovací dialog (MessageBox), který ověřuje, zda si uživatel opravdu přeje záznam smazat.
 - V případě potvrzení (DialogResult.Yes) je navázáno připojení k databázi (Navstevy.accdb) přes OleDbConnection a pomocí příkazu SQL DELETE FROM Navsteva WHERE ID = ? je příslušný záznam odstraněn.
 - Po úspěšném smazání se zobrazí informační hláška a formulář se uzavře s návratovou hodnotou DialogResult.OK, což umožní předchozímu formuláři (VisitListForm) znovu načíst aktuální data ze seznamu.
 - Pokud během operace dojde k chybě (např. nedostupnost souboru), je uživatel informován pomocí chybového dialogu s detailním popisem výjimky.

9.5.2 Popis uživatelského rozhraní

Formulář má pevně nastavenou velikost 1200 x 700 pixelů a na obrazovce se zobrazuje zarovnaný na střed díky nastavení `StartPosition = CenterScreen`. Velikost formuláře nelze uživatelsky měnit, neboť je použito nastavení `FormBorderStyle = FixedSingle`, které zamezuje jeho přetahování či maximalizaci.

Uživatelské rozhraní tvoří základní ovládací komponenty jako `Label`, `TextBox` a `Button`, přičemž všechna pole jsou pouze pro čtení a slouží k zobrazení již uložených údajů. Formulář přijímá veškerá data prostřednictvím konstruktoru při svém otevření, sám je však nijak nemění – umožňuje pouze jejich zobrazení nebo případné smazání záznamu z databáze.

Rozhraní formuláře je rozdělené na dvě části: levý sloupec (`TextBoxy`) – Jméno, Příjmení, Důvod návštěvy, Čas příchodu, Zaměstnanec a pravý sloupec (víceřádkové `TextBoxy`) – Seznam vybavení, Poznámka.

V horní části formuláře je nadpis „Údaje o návštěvě“, formátovaný velkým tučným písmem a zarovnaný na střed.

Všechny textové popisky (`Label`) jsou formátovány pomocí písma Segoe UI, 10.8 pt, Bold, s barvou DimGray. Formulář má pozadí v barvě OldLace, což odpovídá zvolenému stylu celé aplikace. Tlačítka Zpět a Smazat mají odstín RosyBrown, bílé písmo a tučný řez.

10 BUDOUCÍ VÝVOJ APLIKACE

Aplikace v současné podobě pokrývá základní funkčnost potřebnou pro evidenci návštěv ve firmě, ale systém má značný potenciál pro další rozvoj a rozšíření. V rámci této práce byla implementována přibližně třetina ze všech definovaných funkčních požadavků, což odpovídá základnímu provozu recepce, a zároveň otevírá prostor pro další vývoj. Následující body shrnují možnosti budoucího rozšíření aplikace.

Registrace hromadné návštěvy:

- V současné aplikaci je implementována pouze registrace jednorázové návštěvy. Hromadná návštěva by vyžadovala úpravu formuláře tak, aby bylo možné zapsat více návštěvníků současně.

Registrace opakované návštěvy:

- Tato funkce by umožnila automatizovanou správu pravidelných návštěv. Systém by podle zadané frekvence (denně, týdně, měsíčně, vlastní) automaticky vytvářel nové záznamy o plánovaných návštěvách.

Rezervace návštěvy:

- Rezervace návštěv by umožnila plánovat návštěvy dopředu. V současnosti aplikace eviduje pouze příchody ve stejný den. Implementace rezervací by znamenala rozšíření databáze o samostatnou tabulku a nový formulář pro registraci plánovaných návštěv s předdefinovaným datem.

Zobrazení návštěv (rozšíření):

- Aplikace v současné verzi umožňuje zobrazit pouze aktuálně zaevidované návštěvy, ale zobrazení návštěv by šlo rozšířit i o plánované návštěvy (rezervace) a historii návštěv. K tomu by bylo nutné implementovat rozlišení typu návštěvy podle stavu a rozšířit zobrazení o filtrování nebo seskupení podle těchto kategorií. Z pohledu recepčního pracovníka by tím vznikl přehled nad celým životním cyklem návštěvy – od plánování po ukončení.

Označení odchodu:

- Tato funkce by umožnila zaznamenat čas, kdy návštěva objekt opustí. Její zavedení by přineslo přesnější přehled o délce pobytu návštěvníků.

Vyhledávání návštěv:

- Možnost vyhledávat v seznamu návštěv podle různých parametrů (jméno návštěvníka, datum nebo navštívený zaměstnanec) by výrazně zvýšila přehlednost a uživatelský komfort. Tato funkce by vyžadovala implementaci fulltextového vyhledávání nebo filtrů v seznamu návštěv.

Přidání, úprava a odstranění uživatelů:

- Zavedení těchto funkcí by znamenalo vytvoření administračního rozhraní, kde by bylo možné spravovat přístupové účty. Každý uživatel by měl přiřazenou roli (recepční, administrátor) a tomu odpovídající oprávnění.

Přidání a odstranění zaměstnanců:

- V databázi se v současnosti nachází pevně daný seznam zaměstnanců. V budoucnu by bylo vhodné implementovat rozhraní pro správu zaměstnanců, čímž by se odstranila nutnost manuální editace databáze a zvýšila by se flexibilita systému.

ZÁVĚR

Tato bakalářská práce se zabývala návrhem a částečnou implementací desktopové aplikace pro evidenci návštěv ve firmě. Cílem bylo vytvořit jednoduchý a přehledný nástroj, který umožní recepčním efektivně spravovat záznamy o návštěvnících. V úvodních částech práce byla provedena analýza dostupných řešení, definice požadavků a návrh softwarové architektury včetně databázového modelu a UML diagramů.

Na základě těchto návrhů byla vyvinuta funkční desktopová aplikace v jazyce C# s využitím knihovny Windows Forms a databáze Microsoft Access. V rámci implementace byla realizována základní funkcionality: přihlášení a odhlášení uživatele, registrace jednorázové návštěvy, zobrazení seznamu aktuálních návštěv, jejich detailů a možnost odstranění záznamu. Tyto funkce představují přibližně třetinu všech definovaných požadavků.

Přestože aplikace zatím neobsahuje pokročilé funkce, jako je správa uživatelů, rezervace návštěv nebo opakované návštěvy, vytvořený základ umožňuje její další rozšiřování. Návrh systému byl koncipován tak, aby byl flexibilní a snadno upravitelný podle potřeb konkrétní firmy.

Získané poznatky a zkušenosti z oblasti návrhu, vývoje a testování softwaru lze dále využít při rozšiřování systému nebo při tvorbě podobných aplikací v jiných oblastech. Výsledná práce tedy splnila stanovený cíl a představuje využitelný základ pro budoucí vývoj plnohodnotného systému pro evidenci návštěv.

SEZNAM POUŽITÉ LITERATURY

- [1] *A definitive guide to visitor management*. Online. Eptura. Dostupné z: <https://eptura.com/resource/visitor-management-guide/>. [cit. 2025-04-29].
- [2] *What is a visitor management system*. Online. Pitney Bowes. C1996–2025. Dostupné z: <https://www.pitneybowes.com/uk/what-is-visitor-management.html>. [cit. 2025-04-29].
- [3] *Envoy Visitors*. Online. Envoy. C2025. Dostupné z: <https://envoy.com/products/visitors>. [cit. 2025-04-29].
- [4] *Proxyclick*. Online. Eptura. Dostupné z: <https://eptura.com/proxyclick/>. [cit. 2025-04-29].
- [5] *The Leading Visitor Management System for Security & Compliance*. Online. FacilityOS. C2025. Dostupné z: <https://www.facilityos.com/visitoros>. [cit. 2025-04-29].
- [6] *The Visitor Management System that Optimizes Your Workplace*. Online. Vizito. C2025. Dostupné z: <https://www.vizito.eu/>. [cit. 2025-04-29].
- [7] *Visitor Management System*. Online. Greetly. C2014–2025. Dostupné z: <https://www.greetly.com/>. [cit. 2025-04-29].
- [8] *Digitální aplikace pro řízení vstupu dodavatelů a návštěv*. Online. Safe4entry. C2025. Dostupné z: <https://safe4entry.com/>. [cit. 2025-04-29].
- [9] *Komplexní systém evidence návštěv*. Online. Granus. C2021. Dostupné z: <https://granus.cz/wp-content/uploads/2021/11/komplexni-systemy-evidence-navstev-1.pdf>. [cit. 2025-04-29].
- [10] *Visitor Management*. Online. Bosch. C2025. Dostupné z: <https://www.boschsecurity.com/cz/en/solutions/access-control-systems/visitor-management/>. [cit. 2025-04-29].
- [11] ARLOW, Jim a NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2. aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.
- [12] COCKBURN, Alistair. *Use Cases: jak efektivně modelovat aplikace*. Brno: CP Books, 2005. ISBN 80-251-0721-3.

- [13] *Enterprise Architect in 30 minutes*. Online. Sparx Systems. C2000–2025. Dostupné z: <https://sparxsystems.com/enterprise-architect/>. [cit. 2025-04-11].
- [14] CONOLLY, Thomas; BEGG, Carolyn E. a HOLOWCZAK, Richard. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7. Dostupné také z: <http://kramerusndk.nkp.cz/search/handle/uuid:a539aa20-072d-11e4-b1a4-005056827e52>.
- [15] LAURENČÍK, Marek. *SQL: podrobný průvodce uživatele*. Průvodce. Praha: Grada Publishing, 2018. ISBN 978-80-271-0774-2. Dostupné také z: <http://www.grada.cz>.
- [16] OPENAI. *ChatGPT* Online. San Francisco: OpenAI, 2025. Dostupné z: <https://chat.openai.com> [cit. 2025-05-30]
- [17] *Access SQL: základní koncepty, slovník a syntaxe*. Online. Microsoft. C2025. Dostupné z: <https://support.microsoft.com/cs-cz/topic/access-sql-z%C3%A1kladn%C3%AD-koncepty-slovn%C3%ADk-a-syntaxe-444d0303-cde1-424e-9a74-e8dc3e460671>. [cit. 2025-04-11].
- [18] *Lekce 1 - Úvod a prostředí programu MS Access*. Online. ITNetwork. C2025. Dostupné z: <https://www.itnetwork.cz/ms-office/microsoft-access-zaklady/uvod-a-prostredi-programu-ms-access>. [cit. 2025-04-11].
- [19] VIRIUS, Miroslav. *Programování v C#: od základů k profesionálnímu použití*. Knihovna programátora. Praha: Grada Publishing, 2021. ISBN 978-80-271-1216-6.
- [20] SELLS, Chris. *C# a WinForms: programování formulářů Windows : Microsoft .NET development series*. Encyklopedie Zoner press. Brno: Zoner, 2005. ISBN 80-868-1525-0.

SEZNAM OBRÁZKŮ

Obrázek 1: Funkční požadavky	38
Obrázek 2: Funkční požadavky pro správu návštěv	39
Obrázek 3: Funkční požadavky pro správu uživatelů.....	45
Obrázek 4: Nefunkční požadavky.....	47
Obrázek 5: Model případu užití	49
Obrázek 6: Aktéři.....	50
Obrázek 7: Matice sledovatelnosti požadavků	68
Obrázek 8: ERD diagram.....	69
Obrázek 9: LoginForm.....	73
Obrázek 10: MainForm.....	75
Obrázek 11: RegisterVisitForm	77
Obrázek 12: VisitListForm	80
Obrázek 13: VisitDetailForm.....	83

SEZNAM TABULEK

Tabulka 1: Základní scénář – Vytvoření jednorázové návštěvy	52
Tabulka 2: Základní scénář – Vytvoření hromadné návštěvy	53
Tabulka 3: Základní scénář – Vytvoření opakované návštěvy	54
Tabulka 4: Základní scénář – Vytvoření rezervace návštěvy	55
Tabulka 5: Základní scénář – Zobrazit návštěvy	56
Tabulka 6: Základní scénář – Označit odchod návštěvy	57
Tabulka 7: Základní scénář – Vyhledat návštěvu	58
Tabulka 8: Základní scénář – Zobrazit detail návštěvy	59
Tabulka 9: Základní scénář – Zrušit návštěvu/rezervaci	60
Tabulka 10: Základní scénář – Přihlásit se	61
Tabulka 11: Alternativní scénář – Neplatné přihlašovací údaje	61
Tabulka 12: Základní scénář – Odhlásit se	62
Tabulka 13: Alternativní scénář – Nepotvrzené odhlášení	62
Tabulka 14: Základní scénář – Vytvořit uživatele	63
Tabulka 15: Základní scénář – Odstranit uživatele	64
Tabulka 16: Základní scénář – Upravit uživatele	65
Tabulka 17: Základní scénář – Přidat zaměstnance	66
Tabulka 18: Základní scénář – Odstranit zaměstnance	67

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADO.NET	ActiveX Data Objects for .NET
AI	Artificial Intelligence
API	Application Programming Interface
ASP.NET	Active Server Pages .NET
BOZP	Bezpečnost a ochrana zdraví při práci
BPMN	Business Process Model and Notation
C#	C Sharp
CASE	Computer-Aided Software Engineering
DBMS	Database Management Systém
ERD	Entity-Relationship Diagram
FR	Funkční požadavek
GDPR	General Data Protection Regulation
MAUI	Multi-platform App UI
NFR	Nefunkční požadavek
OOP	Object-Oriented Programming
SQL	Structured Query Language
SysML	Systems Modeling Language
UML	Unified Modeling Language
UC	Use Case
VBA	Visual Basic for Applications
VMS	Visitor Management Systém
WinForms	Windows Forms

SEZNAM PŘÍLOH

Příloha P I: prilohy.zip

PŘÍLOHA P I: PRILOHY.ZIP

Soubor ve složce .zip obsahuje:

Zdrojový kód v C# (složka BP).

Návrh databáze (soubor Databáze.accdb).

Požadavky a UC model (soubor EA.eapx).