**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra systémového inženýrství**

# Bakalářská práce

## Matice sazeb pro problém obchodního cestujícího

**Richard Jonáš**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Richard Jonáš

Systémové inženýrství

Název práce

**Matice sazeb pro problém obchodního cestujícího**

Název anglicky

**Adjacency matrix for the travelling salesman problem**

---

**Cíle práce**

Cílem práce je vytvořit aplikaci, která na základě seznamu adres zadaných uživatelem vytvoří matici sazeb pro potřeby řešení problému obchodního cestujícího. Řešení bude založeno na open softwaru tak, aby využití aplikace nepředstavovalo dodatečné nákladu pro uživatele.

**Metodika**

- Studium odborné literatury
- Volba programovacího jazyka
- Volba vhodného API
- Navrhnutí architektury aplikace
- Tvorba uživatelského prostředí
- Programování samotné aplikace
- Testování aplikace
- Funkční aplikace
- Závěr a zhodnocení

**Doporučený rozsah práce**

30-40 stran

**Klíčová slova**

matice sazeb, aplikace, problém obchodního cestujícího

---

**Doporučené zdroje informací**

GUTIN, G. – PUNNEN, A P. *The traveling salesman problem and its variations.* New York: Springer, 2007. ISBN 0387444599.

PELIKÁN, J. 1993. Praktikum z operačního výzkumu. 1.vyd. Praha: VŠE. 86 s. ISBN 80-7079-135-7

ŠUBRT, T. *Ekonomicko-matematické metody.* Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, s.r.o., 2015. ISBN 978-80-7380-563-0.

---

**Předběžný termín obhajoby**

2021/22 LS – PEF

**Vedoucí práce**

Ing. Igor Krejčí, Ph.D.

**Garantující pracoviště**

Katedra systémového inženýrství

Elektronicky schváleno dne 24. 11. 2021

**doc. Ing. Tomáš Šubrt, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 25. 11. 2021

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 15. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Matice sazeb pro problém obchodního cestujícího" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2023 _____

Poděkování

Velice rád bych tímto poděkoval vedoucímu mé bakalářské práce panu Ing. Igorovi Krejčímu, Ph.D. za jeho pomoc a velkou dávku trpělivosti. Při děkování nesmím zapomenout ani na mou rodinu, své blízké a přátele, kteří mě podporovali a věřili ve mě.

# Matice sazeb pro problém obchodního cestujícího


## Abstrakt

V moderním světě je stále důležitější efektivní využívání času a zdrojů. Jedním z klíčových aspektů logistiky a plánování je optimalizace trasy. Problém obchodního cestujícího, který hledá nejkratší cestu pro projití všech bodů, je jedním z nejznámějších optimalizačních problémů v této oblasti. Jen mezi roky 2018 a 2023 bylo na katedře systémového inženýrství vypracováno více než 35 závěrečných prací zabývajících se touto tématikou.

Z tohoto důvodu se tato bakalářská práce zaměřuje na tvorbu Windows Form aplikace napsanou v jazyce C# pomocí vývojového prostředí Visual Studio. Tato aplikace umožní uživateli rychle a zdarma vytvořit matici sazeb na základě adres zadaných uživatelem a použít ji v programu Microsoft Excel pro výpočet problému obchodního cestujícího.

Bakalářská práce se dělí na teoretickou a praktickou část. Teoretická část obsahuje základní pojmy, použité programy a technologie. Praktická část popisuje postup vývoje aplikace, včetně analýzy požadavků, návrhu GUI a popisu tříd a metod použitých k naprogramování aplikace. Součástí práce bude i testování aplikace, aby bylo ověřeno její správné chování a funkčnost.

Výsledkem práce je úspěšně vytvořená aplikace, která splňuje všechny požadavky, které byly stanoveny v zadání práce. Celkově lze říci, že aplikace uživatelům znatelně šetří čas při tvorbě matice pro hledání řešení k problému obchodního cestujícího.



**Klíčová slova:** aplikace, programování, excel, matice sazeb, problém obchodního cestujícího, vývoj, API

# Adjacency matrix for the travelling salesman problem

**Abstract**

In the modern world, efficient use of time and resources is increasingly important. One of the key aspects of logistics and planning is route optimization. The travelling salesman problem which is seeking the shortest path to pass all points is one of the most well-known optimization problems in this area. Between 2018 and 2023 alone, the Department of Systems Engineering has produced more than 35 theses dealing with this topic.

For this reason, this thesis focuses on the creation of a Windows Form application written in C# using the Visual Studio development environment. This application will allow the user to quickly and free of charge create a rate matrix based on addresses entered by the user and use it in Microsoft Excel to find a solution for the travelling salesman problem.

The bachelor thesis is divided into theoretical and practical parts. The theoretical part contains basic concepts, programs and technologies used. The practical part describes the application development process, including requirements analysis, GUI design and description of classes and methods used to program the application. The work will also include testing the application to verify its correct behaviour and functionality.

The result of the work is a successfully developed application that meets all the requirements that were set out in the beginning. Overall, the application saves the user time in creating a matrix for finding an optimal solution to the travelling salesman problem.

**Keywords**: application, programming, excel, adjacency matrix, traveling salesman problem, development, API

# Obsah

# 1. Úvod

V současném moderním světě se neustále zvyšuje důležitost efektivního využívání času a zdrojů. Jedním z klíčových aspektů v oblasti logistiky a plánování je optimalizace trasy a minimalizace času potřebného pro cestování mezi různými body. Problém obchodního cestujícího, který se snaží najít nejkratší cestu pro projití všech bodů, je jedním z nejznámějších optimalizačních problémů v této oblasti. Z univerzitního informačního systému lze dohledat, že mezi roky 2018 a 2023 bylo jen na katedře systémového inženýrství vypracováno více než 35 závěrečných prací zabývající se touto tématikou. Momentálně je pro tvorbu matice možné použít Google API, ale vzhledem k počtu požadavků potřebných na vyřešení reálného problému může cena tohoto řešení rychle přesáhnout i 100 dolarů.

Z tohoto důvodu se bakalářská práce se zabývá vývojem Windows Form aplikace, která na základě seznamu adres zadaných uživatelem rychle, jednoduše, a hlavně zdarma vytvoří matici sazeb použitelnou v programu Microsoft Excel pro výpočet problému obchodního cestujícího. Aplikace je napsána za pomoci vývojového prostředí Visual Studio v programovacím jazyce C#.

Předkládaná bakalářská práce je rozdělena do několika částí. První bude část teoretická. Zde budou charakterizovány použité programy, technologie a základní pojmy použité v bakalářské práci.

Druhá část práce popisuje vývoj a testování samotné aplikace. Nejprve se analyzují a popisují požadavky na vyvíjenou aplikaci. Dále se zaměřuje na GUI a popis jeho prvků. Poté popíše významné třídy a metody pomocí kterých bude aplikace naprogramována. Závěr druhé části se zaměřuje na testování aplikace za účelem ověření funkcionality a správného chování aplikace.

Výběr bakalářské práce vyplynul z mé osobní zkušenosti při řešení obchodního dopravního problému během studia, během které jsem zjistil, že tvorba rozměrnější matice dokáže zabrat opravdu hodně času. Rozhodl jsem se tedy vytvořit aplikaci, která by uživateli tento čas ušetřila.

# 2. Cíl práce a metodika

## 2.1 Cíle práce

Cílem bakalářské práce je vytvořit lehce ovladatelnou desktopovou aplikaci pro operační systém Windows, která uživateli znatelně šetří čas při tvorbě matice pro hledání řešení k problému obchodního cestujícího.

Aplikace uživateli umožňuje buďto kliknutím, nebo zadáním adresy do vyhledávacího pole, přidat určitý počet bodů do mapy, jejichž zeměpisné souřadnice se poté pomocí HTTP požadavku pošlou na externí API, které nám obratem vrátí vzdálenosti mezi jednotlivými body. Následně jsou vzdálenosti exportovány do souboru podporovaného programem Excel, který uživatel může dále využít v součinnosti s pluginem TSPKOSA pro výpočet problému obchodního cestujícího.

## 2.2 Metodika

Práce bude rozdělena na dvě části, teoretickou a praktickou. Teoretická část bakalářské práce se zaměří na vysvětlení pojmů jako problém obchodního cestujícího, programování, operační systém, vývojové studio, nebo GUI. Popíše také použité technologie a programy pojící se s tématem práce.

V praktické části bakalářské práce autor popíše postup vývoje samotné aplikace. Nejprve zanalyzuje a popíše požadavky na vyvíjenou aplikaci. Poté se zaměří na návrh GUI, kde popíše nejdůležitější ovládací prvky, které aplikace obsahuje a přiblíží, jak fungují. Dále popíše, významné třídy a metody pomocí kterých bude aplikace naprogramována.

Před samotným programováním aplikace bude nutné nejprve nastudovat odbornou literaturu, během čehož se autor pokusí zjistit, na jaké problémy by během vývoje aplikace mohl narazit, popřípadě jak se jim vyhnout, nebo je alespoň co nejoptimálněji vyřešit.

Dalším úkolem bude vhodně zvolit externí API, které bude vyřizovat naše HTTP požadavky. Jelikož provoz aplikace nesmí představovat dodatečné náklady pro uživatele, bude výběr vhodných API velmi omezený.

Následovat bude volba vhodného programovacího jazyka za pomoci, kterého bude aplikace následně vyvíjena. Volba programovacího jazyka nám do značné míry ovlivní také typ aplikace.

Dále následuje samotný návrh architektury aplikace a návrh uživatelského prostředí, které bude uživatel používat k ovládání aplikace. Bude potřeba vhodně rozvrhnout ovládací prvky a zvolit jakým způsobem bude uživateli umožněno s aplikací interagovat.

Nakonec přijde na řadu samotné programování, ladění a testování aplikace.

# 3. Teoretická část práce

## 3.1 Problém obchodního cestujícího

Problém obchodního cestujícího (Traveling Salesman Problem, TSP) je dobře známý problém v oblasti informatiky a optimalizace. Problém spočívá v nalezení nejkratší možné trasy, po které může prodavač navštívit danou množinu měst přesně jednou a vrátit se do výchozího města. Tento problém je považován za jeden z nejnáročnějších optimalizačních problémů, neboť je klasifikován jako NP-úplný, což znamená, že řešení nelze nalézt v polynomiálním čase. (1)

Problém obchodního cestujícího má širokou škálu uplatnění. Lze ho použít pro optimalizaci trasování, nebo logistické plánování. TSP lze například použít k optimalizaci tras pro doručování zásilek, nebo k plánování trasy pro obchodníka, který chce navštívit řadu měst, aby prodal své výrobky. Pro řešení problému obchodního cestujícího byla vyvinuta celá řada algoritmů. Mezi nejoblíbenější algoritmy patří exaktní algoritmy, heuristické algoritmy a metaheuristické algoritmy. (2)

Exaktní algoritmy zkoumají všechny možné permutace měst, což je činí nepraktické pro velké datové soubory. Heuristické algoritmy používají pravidla pro rychlé nalezení téměř optimálního řešení, zatímco metaheuristické algoritmy používají inteligentní techniky, jako jsou genetické algoritmy, simulované žíhání a optimalizace mravenčích kolonií k nalezení kvalitních řešení. Navzdory mnoha algoritmům, které byly vyvinuty pro řešení problému obchodního cestujícího, zůstává nalezení optimálního řešení náročným úkolem, zejména pro větší soubory dat. (1)

### 3.1.1 Historie TSP

Historie TSP sahá až do 19. století, kdy matematici začali zkoumat koncept nalezení nejkratší trasy, která spojuje řadu měst. Ve 30. letech 20. století byl TSP zaveden jako formální matematický problém díky průkopnické práci Karla Mengera a Hasslera Whitneyho. Od té doby se TSP stal jedním z nejrozsáhleji studovaných problémů v oblasti kombinatorické optimalizace a inspiroval vývoj mnoha algoritmů, heuristik a aproximačních metod. (2)

Vzestup TSP jako referenčního problému pro optimalizační algoritmy vedl k vývoji nových technik, jako jsou branch-and-bound a branch-and-cut algoritmy, které se osvědčily při řešení TSP a dalších příbuzných optimalizačních problémů. Mezi nejnovější pokroky ve výzkumu TSP patří využití smíšeného celočíselného programování, algoritmů branch-and-price

a technik strojového učení, které vedly k vývoji efektivnějších a přesnějších řešení stále složitějších instancí tohoto problému. (1)

### 3.1.2 Typy algoritmů

**Exaktní algoritmy**

Exaktní algoritmy jsou důležitou třídou algoritmů pro řešení problému obchodního cestujícího (TSP), které mohou zaručit nalezení optimálního řešení. V knize "The Traveling Salesman Problem: A Computational Study" se Applegate a kol. zabývají několika exaktními algoritmy pro řešení TSP, včetně metod větvení a hranic, větvení a řezu a řezné roviny. Tyto algoritmy se při určování optimálního řešení spoléhají na výčet všech možných tras nebo dílčích tras, což může být pro větší instance problému výpočetně náročné. Exaktní algoritmy však mohou poskytnout měřítko pro hodnocení kvality řešení vytvořených heuristickými a aproximačními algoritmy, stejně jako dolní meze potřebné k prokázání optimality. Navzdory výpočetní náročnosti exaktních algoritmů umožnil nedávný pokrok v algoritmech a hardwaru řešit instance TSP s tisíci městy, což z exaktních algoritmů činí důležitý nástroj pro řešení reálných problémů TSP. (1)

**Heuristické algoritmy**

Heuristické algoritmy jsou třídou algoritmů, které vyměňují kvalitu řešení za výpočetní efektivitu a často se používají k řešení velkých instancí problému obchodního cestujícího, které jsou pro exaktní algoritmy neřešitelné. Mezi heuristické algoritmy patří například algoritmy konstruktivní heuristiky, jako jsou metoda nejbližšího souseda a metoda nejlevnějšího vložení, a také zlepšovací heuristiky, jako jsou 2-opt a 3-opt. Tyto algoritmy jsou navrženy tak, aby rychle vygenerovaly proveditelné řešení, které lze poté iterativně zlepšovat prováděním malých lokálních změn v řešení, dokud není dosaženo uspokojivé úrovně kvality. I když heuristické algoritmy nezaručují optimální řešení, v praxi často poskytují dostatečně kvalitní řešení a lze je použít na širokou škálu optimalizačních problémů. (1)

**Metaheuristické algoritmy**

Metaheuristické algoritmy jsou třídou algoritmů, které se spoléhají na iterativní zkoumání rozsáhlých prostorů řešení s cílem nalézt téměř optimální řešení. Existuje celá řada metaheuristických algoritmů, včetně algoritmu simulovaného žíhání, tabu vyhledávání, genetických algoritmů, optimalizace mravenčích kolonií a optimalizace hejnem částic. Simulované žíhání funguje tak, že iterativně přijímá horší řešení s určitou pravděpodobností,

aby se zabránilo uvíznutí v lokálním optimu. Tabu search využívá paměťové struktury, aby se zabránilo opětovnému navracení dříve prozkoumaných řešení. Genetické algoritmy jsou inspirovány přirozeným výběrem a k vývoji populace řešení používají operátory křížení a mutace. Optimalizace pomocí algoritmu mravenčích kolonií je založena na chování mravenčích kolonií, kde proces hledání řídí feromonové stopy. A konečně optimalizace hejnem částic simuluje chování hejna částic, které se pohybují prostorem řešení a vzájemně komunikují, aby prostor prozkoumaly. Tyto metaheuristické algoritmy se osvědčily při řešení rozsáhlých instancí TSP a nadále inspirují nový výzkum a vývoj v oblasti kombinatorické optimalizace. (1)

**Aproximační algoritmy**

Aproximační algoritmy poskytují efektivní řešení problémů TSP tím, že obětují přesnost řešení ve prospěch výpočetní rychlosti. Mezi aproximační algoritmy patří například Christofidesův algoritmus a Lin-Kernighanovy heuristika. Již zmíněný algoritmus nejbližšího souseda lze však také považovat za aproximační algoritmus, protože poskytuje řešení, které je v závislosti na instanci problému zaručeně v určitém rozmezí od optimálního řešení. Lze jej tedy zařadit do kategorie heuristických i aproximačních algoritmů. Christofidesův algoritmus je sofistikovanější algoritmus, který vytváří řešení, jež je v rozmezí 1,5 násobku optimálního řešení kombinací minimálního rozpínacího stromu s minimální váhou dokonalé shody. Lin-Kernighanova heuristika je iterační zlepšovací algoritmus, který opakovaně upravuje dané řešení s cílem najít lepší řešení. Dále existují například Frieze-Kannanův algoritmus a Held-Karpův algoritmus, které lze použít k řešení problémů TSP s určitými omezeními. Aproximační algoritmy sice nezaručují optimální řešení, ale pro praktické účely mohou poskytnout řešení, která jsou dostatečně blízká optimálnímu řešení, a to za zlomek času, který potřebují exaktní algoritmy. (1)

## 3.2 Excel

Excel je tabulkový procesor vyvinutý společností Microsoft a je součástí skupiny produktů Office. Microsoft Excel je široce používaný pro správu, analýzu a prezentaci dat. Tento software umožňuje uživatelům vytvářet, upravovat a organizovat data ve formátu podobném tabulce, známém jako pracovní list.

Excel se skládá z několika součástí, mezi něž patří sešit, pracovní listy, buňky, sloupce, řádky a vzorce.

- Sešit je soubor, který obsahuje jeden nebo více pracovních listů.
- Pracovní list je mřížka buněk, která obsahuje data, vzorce a formátování. Jeden sešit může obsahovat více pracovních listů.
- Buňka je průsečíkem řádku a sloupce a je základní jednotkou pracovního listu. Může obsahovat data, vzorce, nebo formátování.
- Sloupec je svislá skupina buněk označená písmenem v horní části sloupce.
- Řádek je vodorovná skupina buněk označená číslem na levé straně řádku.
- Vzorec je rovnice, která provádí výpočty, nebo manipulace s daty. Lze jej použít k provádění aritmetických, logických, textových a dalších operací.

Microsoft Excel je k dispozici uživatelům na platformách Windows, macOS, Android a iOS. Mezi alternativy programu excel patří například Google Sheets, Numbers, nebo Apache OpenOffice Calc. (3)

## 3.3 Programování

Programování je proces vytváření instrukcí, které počítač chápe a provádí za účelem provedení určitého úkolu nebo vyřešení problému. Zahrnuje psaní kódu pomocí programovacího jazyka, což je soubor instrukcí používaných k vytváření softwaru, aplikací, webových stránek a dalších počítačových programů.

Programy lze psát v různých programovacích jazycích, včetně populárních jazyků, jako jsou například Python, Java, C++ a JavaScript, nebo C#. Každý jazyk má svou vlastní syntaxi neboli soubor pravidel a konvencí, které je třeba dodržovat, aby bylo možné napsat platný kód.

Programátoři mohou vytvářet aplikace pro mobilní zařízení, vyvíjet webové stránky, psát software pro firmy a vytvářet videohry. Schopnost programovat je velmi žádaná a v dnešním světě plném technologií je považována za cennou dovednost. (4)

## 3.4 Operační systém

Operační systém je program, který spravuje hardwarové a softwarové prostředky počítače a poskytuje běžné služby pro počítačové programy. Funguje jako prostředník mezi hardwarem a softwarem počítače, řídí a koordinuje počítačové zdroje, jako je paměť, výpočetní výkon, nebo uložiště. Operační systémy poskytují uživatelům platformu pro spouštění aplikací a softwaru, procházení webu a správu souborů. (5)

Mezi nejpoužívanější operační systémy pro stolní a přenosné počítače patří Windows, macOS, Linux a Chrome OS. Windows je nejčastěji používaným operačním systémem na světě, přičemž jeho podíl na trhu k lednu 2022 činil přibližně 77 %. MacOS je druhým nejoblíbenějším operačním systémem se zhruba 16% podílem na trhu, zatímco Linux má menší, ale stále významné zastoupení s 2% podílem. Operační systém Chrome OS je na trhu relativně novým hráčem, jeho tržní podíl se pohybuje okolo 0,5 %. (6)

**Windows**

Microsoft Windows je celosvětově nejrozšířenější operační systém pro počítače, který vyvinula společnost Microsoft Corporation. Nabízí grafické uživatelské rozhraní (GUI), možnost multitaskingu a širokou škálu softwarových aplikací. Existuje mnoho různých verzí systému Windows, ale některé známé jsou například Windows 10 vydaný v roce 2015, Windows 8 uvedený na trh v roce 2012, Windows 7 představený v roce 2009 a Windows Vista z roku 2007. (7)

**MacOS**

Operační systém macOS vyvinula společnost Apple Inc. a používá se ve stolních a přenosných počítačích společnosti Apple. Je známý svým uživatelsky přívětivým rozhraním, pokročilými funkcemi a bezproblémovou integrací s dalšími produkty a službami společnosti Apple. Operační systém byl vydán v několika verzích, včetně Mojave, která byla uvedena v roce 2018, High Sierra, která byla vydána v roce 2017, a Sierra, která byla představena v roce 2016. (7)

**Linux**

Linux je operační systém s otevřeným zdrojovým kódem, který lze používat a upravovat zdarma. Je proslulý svou stabilitou, bezpečností a flexibilitou. Je hodně používán vývojáři, správci systémů a dalšími IT profesionály. (7)

**ChromeOS**

Chrome OS je odlehčený operační systém vyvinutý společností Google, který je určený pro použití v levných noteboocích zvaných Chromebooky. Je založen na linuxovém jádře a do značné míry se spoléhá na cloudové aplikace a uložiště. (7)

## 3.5 Integrované vývojové prostředí

*"Integrované vývojové prostředí (IDE) je softwarový nástroj, který poskytuje programátorům komplexní vybavení pro vývoj softwaru. IDE se obvykle skládá přinejmenším z editoru zdrojového kódu, nástrojů pro automatizaci sestavování aplikace a ladicího programu."* (8, s. 82), (překlad vlastní)

### 3.5.1 Visual Studio

Visual Studio je výkonné integrované vývojové prostředí (IDE) vyvinuté společností Microsoft pro vytváření různých typů softwarových aplikací. Prostředí vývojářům poskytuje komplexní sadu nástrojů a služeb pro vytváření, testování a nasazování softwaru na široké škále platforem a zařízení. Visual Studio také podporuje řadu programovacích jazyků, včetně jazyků C#, C++, Visual Basic, F#, Python a mnoho dalších. Nabízí také pokročilé funkce pro úpravu kódu, ladění a testování, díky čemuž je oblíbenou volbou vývojářů. (9)

Mezi některé klíčové funkce prostředí Visual Studio patří (9):

- **Úprava kódu:** Visual Studio má bohatý editor kódu, který nabízí pokročilé funkce, jako je zvýrazňování syntaxe, doplňování kódu, refaktoring a úryvky kódu. Podporuje více programovacích jazyků a poskytuje přizpůsobitelné a flexibilní uživatelské rozhraní.

- **Ladění:** Visual Studio obsahuje výkonný ladicí program, který pomáhá vývojářům najít a opravit problémy v kódu. Podporuje funkce, jako je krokové procházení kódu, nastavování bodů přerušení, kontrola proměnných a profilování výkonu.

- **Testování:** Visual Studio nabízí integrované nástroje pro testování jednotek, testování zátěže a testování výkonu. Podporuje oblíbené testovací rámce, jako jsou MSTest, NUnit a xUnit.

- **Integrace s nástroji pro správu projektů:** Visual Studio obsahuje funkce jako je integrace správy zdrojů se systémem Git a dalšími systémy pro správu verzí, nástroje pro agilní správu projektů a služby pro spolupráci, jako je Azure DevOps.

- **Vývoj napříč platformami:** Visual Studio podporuje vývoj aplikací pro širokou škálu platforem, včetně Windows, MacOS a Linuxu. Podporuje také mobilní vývoj pro zařízení se systémy Android a iOS.

- **Rozšiřitelnost:** Visual Studio poskytuje rozsáhlý ekosystém rozšíření a doplňků, které lze použít k přizpůsobení IDE a přidání dalších funkcí.

### 3.5.2 Visual Studio Code

Visual Studio Code je bezplatný editor kódu s otevřeným zdrojovým kódem vyvinutý společností Microsoft pro systémy Windows, Linux a macOS. Poskytuje výkonné a flexibilní vývojové prostředí pro vytváření a ladění kódu v různých programovacích jazycích. Visual Studio Code obsahuje řadu funkcí, jako je zvýrazňování syntaxe, dokončování kódu a nástroje pro ladění, které usnadňují psaní a testování kódu. Podporuje také rozšíření, která lze nainstalovat a přidat tak další funkce, například podporu jazyků a vlastní motivy.
Jednou z hlavních výhod aplikace Visual Studio Code je její odlehčený design, díky němuž se rychle a snadno používá. Podporuje také širokou škálu programovacích jazyků, což z něj činí univerzální nástroj pro vývojáře pracující na různých projektech. (10)

### 3.5.3 Porovnání Visual Studio a Visual Studio Code

Visual Studio Code i Visual Studio jsou vývojová prostředí vytvořená společností Microsoft, ale jejich funkce a cílová skupina se od sebe poněkud liší. Visual Studio je komplexnější vývojové prostředí, které poskytuje kompletní sadu nástrojů pro tvorbu aplikací, včetně funkcí pro návrh uživatelských rozhraní, správu databází a vývoj webových a mobilních aplikací. Obvykle se používá pro větší a složitější projekty a je určeno profesionálním vývojářům, kteří vyžadují výkonnější sadu nástrojů.

Naproti tomu Visual Studio Code je odlehčený editor kódu, který poskytuje jednodušší a přehlednější rozhraní pro psaní a ladění kódu. Je určen pro širší okruh vývojářů, včetně těch, kteří pracují na menších projektech, nebo těch, kteří dávají přednost minimalističtějšímu vývojovému prostředí. Visual Studio Code také podporuje širší škálu programovacích jazyků než Visual Studio, takže je univerzálnějším nástrojem pro vývojáře, kteří pracují v různých jazycích a na různých platformách. (11)

## 3.6 User Interface vs User Experience

UX a UI jsou podobné, ale velmi odlišné pojmy. UX znamená User Experience (uživatelský zážitek), zatímco UI znamená User Interface (uživatelské rozhraní).

UX označuje celkovou zkušenost, kterou má uživatel při interakci s digitálním produktem nebo službou. Zahrnuje nejen uživatelské rozhraní, ale také emoce, vnímání a chování uživatele během a po používání produktu, nebo služby. Návrh UX zahrnuje navrhování a optimalizaci digitálních produktů a služeb tak, aby poskytovaly co nejlepší uživatelský zážitek.

Uživatelské rozhraní naproti tomu označuje konkrétně prvky a komponenty, s nimiž uživatelé interagují při používání digitálního zařízení nebo softwarové aplikace. Účelem uživatelského rozhraní je poskytnout uživatelům vizuální a interaktivní způsob interakce se systémem, softwarem nebo zařízením. Návrh uživatelského rozhraní zahrnuje návrh vizuálních a interaktivních součástí digitálního produktu nebo služby, včetně ikon, tlačítek, nabídek a dalších grafických prvků. (12)

### 3.6.1 Textové uživatelské rozhraní

Textové uživatelské rozhraní je typ uživatelského rozhraní, které používá textové znaky a symboly k prezentaci informací a přijímání vstupů od uživatelů. Jedná se o jednodušší a základnější formu uživatelského rozhraní ve srovnání s grafickým uživatelským rozhraním.

Textové uživatelské rozhraní má oproti grafickému rozhraní některé výhody, například vyžaduje menší výpočetní výkon a je přístupnější pro uživatele se zrakovým postižením. Ve srovnání s grafickým uživatelským rozhraním však může být méně intuitivní a méně vizuálně přitažlivé.

Mezi běžné příklady textového uživatelského rozhraní patří rozhraní příkazového řádku v operačních systémech Windows nebo Linux a také některá programovací prostředí, například IDLE v Pythonu. V rozhraní příkazového řádku uživatelé zadávají textové příkazy pro interakci se systémem, například procházení souborového systému, spouštění aplikací nebo provádění skriptů. (13)

### 3.6.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (GUI) je typ uživatelského rozhraní, které umožňuje uživatelům komunikovat se softwarovými aplikacemi a zařízeními jako jsou počítače a chytré telefony prostřednictvím vizuálních prvků, jako jsou ikony, tlačítka a další grafické prvky.

Grafické uživatelské rozhraní bylo poprvé představeno v 80. letech 20. století jako náhrada za textová rozhraní, například rozhraní příkazového řádku, která vyžadovala od uživatelů zadávání příkazů pro interakci se systémem.

Grafické uživatelské rozhraní výrazně zlepšilo uživatelský zážitek z digitálních zařízení a softwarových aplikací, protože je intuitivnější, jednodušší na používání a vizuálně přitažlivější.

S grafickým uživatelským rozhraním se manipuluje pomocí ukazovacího zařízení, jako je myš, trackball, stylus nebo prst na dotykové obrazovce. (14)

### 3.6.3 Základní principy vytváření uživatelského rozhraní

Existuje několik základních principů návrhu uživatelského rozhraní, které mohou pomoci vytvořit uživatelsky přívětivé a vizuálně atraktivní rozhraní (15):

- **Jednoduchost:** Uživatelské rozhraní musí být jednoduché a snadno použitelné a nesmí obsahovat nepřehledné a zbytečné prvky. Díky tomu mohou uživatelé rychle najít to, co potřebují, a efektivně plnit své úkoly.
- **Konzistence:** V celém uživatelském rozhraní používejte konzistentní prvky vizuálního designu, jako jsou písma, barvy a ikony. To může pomoci vytvořit jednotné a soudržné uživatelské prostředí.
- **Přehlednost:** Používejte jasný a stručný jazyk. To může pomoci zajistit, že uživatelé pochopí účel a funkci prvků uživatelského rozhraní.
- **Zpětná vazba:** Poskytujte uživatelům zpětnou vazbu při interakci s prvky uživatelského rozhraní, například při kliknutí na tlačítko nebo vyplnění formuláře. Díky tomu se uživatelé mohou cítit jistěji při svých činnostech.
- **Flexibilita:** Umožněte flexibilitu a přizpůsobení, například umožněte uživatelům upravit velikost písma nebo barevné schéma. To může pomoci přizpůsobit se různým preferencím a potřebám uživatelů.

## 3.7 Programovací jazyk

*"Programovací jazyk je umělý jazyk určený k vyjádření výpočtů, které může provádět stroj, zejména počítač. Je to soubor pravidel pro vyjádření a reprezentaci výpočetních procesů ve formě vhodné pro provádění strojem."* (16, s. 1), (vlastní překlad)

### 3.7.1 Programovací jazyk C#

Programovací jazyk C# byl společností Microsoft navržen jako moderní, vysokoúrovňový, objektově orientovaný programovací jazyk, který se uživatel snadno naučil a zároveň poskytoval vysokou úroveň výkonu a flexibility. Byl silně ovlivněn jazyky C++ a Java se kterými má velmi podobnou syntaxi, proto je velmi jednoduché z těchto jazyků na C# přejít. Jazyk lze využít při programování široké škály aplikací, mezi některé příklady patří desktopové, mobilní, nebo webové aplikace. (17)

### 3.7.2 Porovnání C# a C++

Jazyk C# vychází z programovacího jazyka C++, který byl vyvinut v 80. letech 20. století a dodnes je hojně používán. Stejně jako C++, je i C# kompilovaný jazyk, což znamená, že napsaný kód je přeložen do strojového jazyka, který může být přímo spuštěn procesorem počítače. C# je v porovnání s C++ mnohem robustnější, ale zároveň obsahuje jednodušší syntaxi a z toho důvodu je považován za jednodušší k naučení.

Velký rozdíl lze najít v tom, jak jednotlivé jazyky spravují paměť. C# obsahuje garbage collector, který automaticky běží na pozadí, kde vyhledává a uvolňuje již nepotřebné části paměti, nebo předchází některým běhovým chybám, například úniku paměti (anglicky memory leak). C++ naopak garbage colletor neobsahuje, což znamená, že o správu paměti se musí starat sám programátor.

Jak C#, tak C++ podporují objektově orientované programování, což vývojářům umožňuje vytvářet složitější aplikace bez zvýšení nároků na údržbu. Mezi výhody objektově orientovaného programování patří vlastnosti jako dědičnost, nebo polymorfismus. Na rozdíl od C++ byl C# od počátku navrhován, jako objektově orientovaný jazyk a má mnoho funkcí, které vývojáři usnadňují práci s objekty, například automatickou správu paměti a bohatou knihovnu tříd. V dnešní době C++ také podporuje objektově orientované programování, ale zároveň je silně zaměřen na nízko-úrovňové systémové programování, což může práci s objekty ztěžovat. (18)

## 3.8 Framework

Framework je soubor předpřipravených softwarových komponent, které lze použít ke zjednodušení a urychlení procesu vytváření aplikace. Frameworky poskytují vývojářům základ pro budování aplikací tím, že nabízejí sadu nástrojů, knihoven a běhových prostředí, které abstrahují od složitostí základního operačního systému a hardwaru. Kromě toho frameworky poskytují standardizovaný způsob provádění běžných úloh, jako je zpracování vstupů a výstupů, správa paměti a systémová komunikace, a mohou vývojářům nabídnout osvědčené postupy a pokyny, kterými se mohou řídit. Celkově je framework abstrakční vrstvou, která zjednodušuje vývoj aplikací tím, že poskytuje předem připravené komponenty a standardizované metodiky. (19)

### 3.8.1 .NET

.NET je framework pro vývoj softwaru vyvinutý společností Microsoft, který poskytuje rozsáhlou a výkonnou sadu nástrojů a knihoven pro vytváření široké škály softwarových aplikací.

Jednou z klíčových vlastností frameworku .NET je podpora více programovacích jazyků, včetně C#, F# a Visual Basic. To znamená, že si vývojáři mohou vybrat jazyk, který jim nejvíce vyhovuje, a přitom využívat funkce a možnosti frameworku .NET.

Mezi ně patří široká škála nástrojů a technologií pro vytváření webových, mobilních či stolních aplikací. Vývojáři mohou například používat Windows Forms, nebo WPF pro vytváření desktopových aplikací, APS.NET nebo Blazor pro vytváření webových aplikací a Xamarin nebo Unity pro vytváření mobilních aplikací.

Framework .NET lze použít pro vývoj aplikací pro různé platformy. Od Windows přes iOS až po aplikace podporující různé typy procesorů (ARM32, ARM64, x64, x86). (20)

### 3.8.2 Windows Forms

Windows Forms (zkráceně WinForms) je framework uživatelského rozhraní, který se používá k vytváření aplikací pro počítače s operačním systémem Windows.

WinForms poskytují vývojáři širokou škálu komponentů (ovládacích prvků), které lze použít k vytvoření uživatelského rozhraní aplikace. Mezi tyto komponenty patří například tlačítka, textová pole, štítky, zaškrtávací políčka, přepínače, ale i mnoho dalších. Hlavní výhodou frameworku Windows Forms je jeho jednoduchost. Ta spočívá převážně v možnosti

použití takzvaného drag-and-drop designéru, který umožňuje vývojáři snadno a rychle uspořádat ovládací prvky příslušném daném formuláři.

Další výhodou je podpora datových vazeb. Ta umožňuje vývojářům svázat ovládací prvek se zdrojem dat, jako například databáze nebo XML soubor a automaticky jej aktualizovat při změně dat. To usnadňuje vytváření aplikací, které zobrazují a manipulují s daty. (21)

**Komponenty**

V prostředí Windows Forms je komponenta opakovaně použitelný kus kódu, který zapouzdřuje funkce a může být přidán do formuláře, nebo jiného ovládacího prvku kontejneru, aby poskytoval další funkce, nebo chování.

Komponenty lze do aplikace přidávat přetažením z okna Toolbox. Po přidání lze komponenty konfigurovat pomocí okna Vlastnosti a jejich události lze zpracovávat pomocí prostředí Visual Studio IDE nebo psaním kódu.

Windows Forms poskytuje nejenom řadu ovládacích prvků připravených k okamžitému použití, ale obsahuje také infrastrukturu pro návrh vlastních ovládacích prvků.

**Mezi základní prvky patří:**

**Form**

Komponenta Form je základním kontejnerem pro uživatelské rozhraní aplikace. Poskytuje okno, které lze přizpůsobit pomocí dalších ovládacích prvků.

**Button**

Komponenta Button slouží ke spuštění akce po kliknutí uživatele. Lze ji přizpůsobit pomocí textu, písma, barvy a dalších vlastností.

**Label**

Komponenta Label slouží k zobrazení textu v uživatelském rozhraní.

**TextBox**

Komponenta TextBox slouží k přijímání uživatelského vstupu ve formě textu.

**CheckBox**

Komponenta CheckBox slouží k tomu, aby uživatel mohl vybrat položku ze seznamu možností.

**RadioButton**

Komponenta RadioButton slouží k tomu, aby uživatel mohl vybrat jednu možnost ze seznamu. (22)

### 3.8.3 GMap.NET

Komponenta GMapControl systému Windows Forms je ovládací prvek rozhraní .NET, který umožňuje vývojářům integrovat různé zdroje map do jejich desktopových aplikací. Tato komponenta poskytuje pohodlný způsob, jak zobrazovat mapy a překrývat na nich data, jako jsou značky, polygony a trasy.

Komponenta je založena na rozhraní Google Maps JavaScript API a k zobrazení mapy používá ovládací prvek WebBrowser. (23)

## 3.9 API

API (Application Programming Interface) je rozhraní, které umožňuje interakci mezi aplikacemi nebo webovými službami. Vývojáři mohou využít API pro vytvoření nových aplikací, které využívají funkce a data poskytovaná jinými aplikacemi nebo službami. API zahrnuje soubor definic, protokolů a nástrojů pro vývoj aplikací.

API může být ve formě knihovny, což je soubor funkcí a procedur, které mohou být volány jinými aplikacemi, nebo webového rozhraní, které umožňuje volání funkcí a procedur prostřednictvím internetového protokolu.

API umožňuje vývojářům využít již existujících funkcí a dat, což snižuje náklady a zvyšuje rychlost vývoje nových aplikací. API také umožňuje vývojářům využívat výhod specializace jiných poskytovatelů služeb, což může vést ke zlepšení kvality a rozšíření funkcí nových aplikací. (24)

### 3.9.1 OpenRouteService API

OpenRouteService je open-source směrovací služba, která poskytuje vývojářům přístup k řadě směrovacích a optimalizačních algoritmů prostřednictvím rozhraní REST API. Vyvinul ji Heidelberský institut pro geoinformační technologie a je založena na OpenStreetMap datech.

OpenRouteService poskytuje širokou škálu trasovacích funkcí, včetně trasování pro automobily, pěší a cyklisty, stejně jako trasování pro veřejnou dopravu a těžká vozidla. Podporuje také optimalizační algoritmy, jako je problém obchodního cestujícího (TSP) a problém trasování vozidel (VRP), které se běžně používají v logistice a plánování dopravy.

Služba OpenRouteService je pro nekomerční účely k dispozici zdarma a pro komerční použití nabízí řadu cenových možností. Nabízí také řadu klientských knihoven a zásuvných modulů pro oblíbené programovací jazyky, jako jsou C#, Python, Java a JavaScript, což vývojářům usnadňuje integraci do jejich projektů. (25)

### 3.9.2 HTTP požadavek

HTTP požadavek je zpráva odeslaná klientem na server pomocí HTTP protokolu. HTTP protokol je používaný pro přenos dat přes internet a umožňuje klientům (například webovým prohlížečům) požadovat zdroje od serverů (například webových serverů).

HTTP požadavky se skládají z několika částí, včetně řádku požadavku, hlaviček a volitelného těla zprávy. Řádek požadavku určuje metodu HTTP (například GET nebo POST), požadovaný prostředek a použitou verzi HTTP protokolu.

Hlavičky poskytují další informace o požadavku, například agenta uživatele, který identifikuje klienta zadávajícího požadavek, typ obsahu všech dat odesílaných s požadavkem nebo informace o ověření, které jsou nutné pro zpracování požadavku. Nepovinné tělo zprávy obsahuje veškerá data odesílaná s požadavkem.

Jakmile server přijme požadavek, zpracuje jej a odešle klientovi odpověď. Odpověď obvykle obsahuje stavový kód, například 200 nebo 404. Hlavičky poskytující další informace o odpovědi a nepovinné tělo zprávy obsahující veškerá data požadovaná klientem. (26)

## 3.10 OpenStreetMaps

OpenStreetMap (OSM) je volně editovatelná mapa světa, kterou vytvářejí a spravují dobrovolníci. OSM byla založena v roce 2004 a v současnosti je jedním z největších a nejaktivnějších open-source mapových projektů na světě. Jeho cílem je vytvořit mapu, která je zdarma, přesná a dostupná všem.

Jedním z klíčových rysů OSM je, že je zcela otevřená a založená na spolupráci. Každý může přispívat do mapových dat přidáváním nebo úpravou informací, jako jsou silnice, budovy a body zájmu. Platforma umožňuje vytvářet vlastní mapy a vyvíjet lokalizační služby, což z ní činí cenný zdroj pro podniky, vlády a nevládní organizace.

OSM má různorodou komunitu přispěvatelů z celého světa, kteří ke sběru a ověřování dat používají různé mapovací techniky, například zařízení GPS, letecké snímky a průzkumy na místě. Licence otevřených dat platformy umožňuje volné použití a šíření dat, což z ní činí ideální zdroj pro výzkum, vzdělávání a inovace.

Mapové podklady v OSM jsou neustále aktualizovány a zpřesňovány, denně jsou do nich přidávány nové informace. Platforma také poskytuje řadu nástrojů a zdrojů, které pomáhají přispěvatelům, včetně mapovacího softwaru, dokumentace a podpory komunity.

Kromě mapových dat hostuje OSM také řadu aplikací založených na mapách, včetně nástrojů pro určování tras a navigaci, překryvů počasí a dopravy, a dokonce i zážitků v rozšířené realitě. (27)

## 3.11 GitHub

GitHub je webová platforma, která umožňuje vývojářům spolupracovat na softwarových projektech a společně spravovat jejich kód. Jedná se o platformu pro hostování kódu, která používá systém správy verzí Git ke správě změn v kódu a sledování problémů a chyb. GitHub umožňuje vývojářům vytvářet a spravovat repozitáře, což jsou sbírky kódu, které lze sdílet s ostatními. Uživatelé mohou spolupracovat na projektech tím, že přispívají do kódu, poskytují zpětnou vazbu a sledují problémy a chyby.

GitHub také poskytuje mnoho nástrojů a funkcí, jako jsou nástroje pro správu projektů, nástroje pro kontrolu kódu a integrace s dalšími nástroji pro vývoj softwaru. GitHub také poskytuje mnoho nástrojů a funkcí, které vývojářům pomáhají budovat jejich profesní profily, včetně možnosti prezentovat svou práci, přispívat do jiných projektů a spojovat se s dalšími vývojáři. (28)

# 4. Praktická část práce

V této kapitole jsou autorem popisovány požadavky, které má výsledná aplikace splňovat. Zmíní, jak postupuje při jejich analýze, a jak probíhá návrh a implementace programu, který je vyvíjen jako součást bakalářské práce.

Autor také popisuje klíčové prvky aplikace, včetně funkcionality, uživatelského rozhraní a použitých technologií a nástrojů.

## 4.1 Požadavky na aplikaci

### 4.1.1 Funkční požadavky

Aplikace bude splňovat tyto požadavky:

1. Grafické zobrazení
2. Umožnuje uživatelům vkládat a mazat data.
3. Umožňuje přidání bodů pomocí interakce se samotnou mapou.
4. Umožnuje přidání bodů pomocí vyhledání adresy v textovém poli.
5. Umožnuje tisknutou a exportovat data z aplikace do formátu podporovaného programem Microsoft Excel
6. Používání aplikace nepředstavuje dodatečné náklady pro uživatele

### 4.1.2 Nefunkční požadavky

Aplikace je vyvíjena v prostředí Visual Studio 2019 Professional na počítači s operačním systémem Windows 10, na platformě .NET Framework 4.7.2 s použitím frameworku WinForms.

Pro zobrazení a práci s mapovými daty je využívána knihovna GMap.NET. Pro export výsledné matice do formátu podporovaného aplikací Microsoft Excel je používán autorem vytvořený parser, který je rychlejší.

Mezi non-funkční požadavky patří:

1. Výkonnost: Aplikace musí být dostatečně rychlá a efektivní, aby uživatelé nemuseli čekat příliš dlouho na vykonání operací.
2. Spolehlivost: Aplikace by měla být spolehlivá a stabilní, aby se minimalizovala pravděpodobnost výpadků a chyb.
3. Použitelnost: Aplikace by měla být snadno použitelná a intuitivní pro uživatele.
4. Estetika: Aplikace by měla být esteticky příjemná a přehledná.

## 4.2 Analýza požadavků

**Grafické zobrazení**

Hlavním cílem tohoto požadavku je umožnit uživatelům snadné a intuitivní používání aplikace a poskytnout jim jasný přehled o stavu aplikace a prováděných operacích.

Aplikace by měla využívat různé grafické prvky, jako jsou ikony, tlačítka a mapy, aby poskytovala uživatelům vizuální zpětnou vazbu a umožnila jim snadné ovládání aplikace.

Kvalita grafického zobrazení by měla být dostatečně vysoká, aby umožnila uživatelům snadné a přesné čtení zobrazených informací. Příliš malé písmo, nečitelné barvy nebo nevhodné rozložení prvků mohou ztížit používání aplikace.

Grafické zobrazení by mělo být jednoduché a přehledné, aby uživatelé mohli snadno pochopit, co se v aplikaci děje, a jak mohou provádět své operace. Komplexní nebo přeplněné grafické rozhraní může způsobit zmatky nebo zpomalit používání aplikace.

Grafické zobrazení by mělo být responzivní, aby uživatelé mohli rychle reagovat na změny v aplikaci a provádět své operace bez prodlev.

**Umožňuje uživatelům vkládat a mazat data**

Funkční požadavek na umožnění uživatelům vkládat a mazat data zahrnuje všechny funkce, které se týkají manipulace s daty v rámci aplikace. Hlavním cílem tohoto požadavku je umožnit uživatelům snadné a rychlé vkládání nových dat a mazání existujících dat v aplikaci.

**Umožňuje přidání bodů pomocí interakce se samotnou mapou**

Tento požadavek umožní uživateli přidávat body na mapu pomocí interakce se samotnou mapou. Zahrnuje funkci aplikace, která umožňuje uživatelům přidávat bodové záznamy na mapu kliknutím na určité místo. K tomuto účelu je potřebné používat interaktivní mapu, která umožní uživatelům snadnou navigaci a výběr místa, kde chtějí přidat bod. Tuto funkci lze implementovat pomocí různých prostředků, například prostřednictvím tlačítka na liště nástrojů, nebo přímo v rámci mapy, kdy uživatel vybere místo na mapě a následně provede přidání bodu. Po kliknutí na tlačítko „Přidat“ se bod uloží do listu hodnot, aby bylo možné s ním dále pracovat a zobrazit ho na mapě.

**Umožnuje přidání bodů pomocí vyhledání adresy v textovém poli**

Uživateli musí být umožněno zadat určitou adresu do textového pole a následně přidat bodový záznam na mapu na základě této adresy. K tomuto účelu musí být aplikace propojena s

externí službou pro geokódování, která převede zadanou adresu na souřadnice, jež se následně využijí pro vytvoření bodového záznamu na mapě, který se automaticky přidá na místo odpovídající těmto souřadnicím. Tato funkce umožňuje uživatelům rychle a snadno najít a označit určité body na mapě na základě adresy.

Uživatelské rozhraní pro tuto funkci musí obsahovat textové pole pro zadání adresy, tlačítko pro spuštění vyhledávání, a tlačítko pro potvrzení výsledku a přidání bodu na mapu.

**Umožnuje tisknutou a exportovat data z aplikace do formátu podporovaného programem Microsoft Excel**

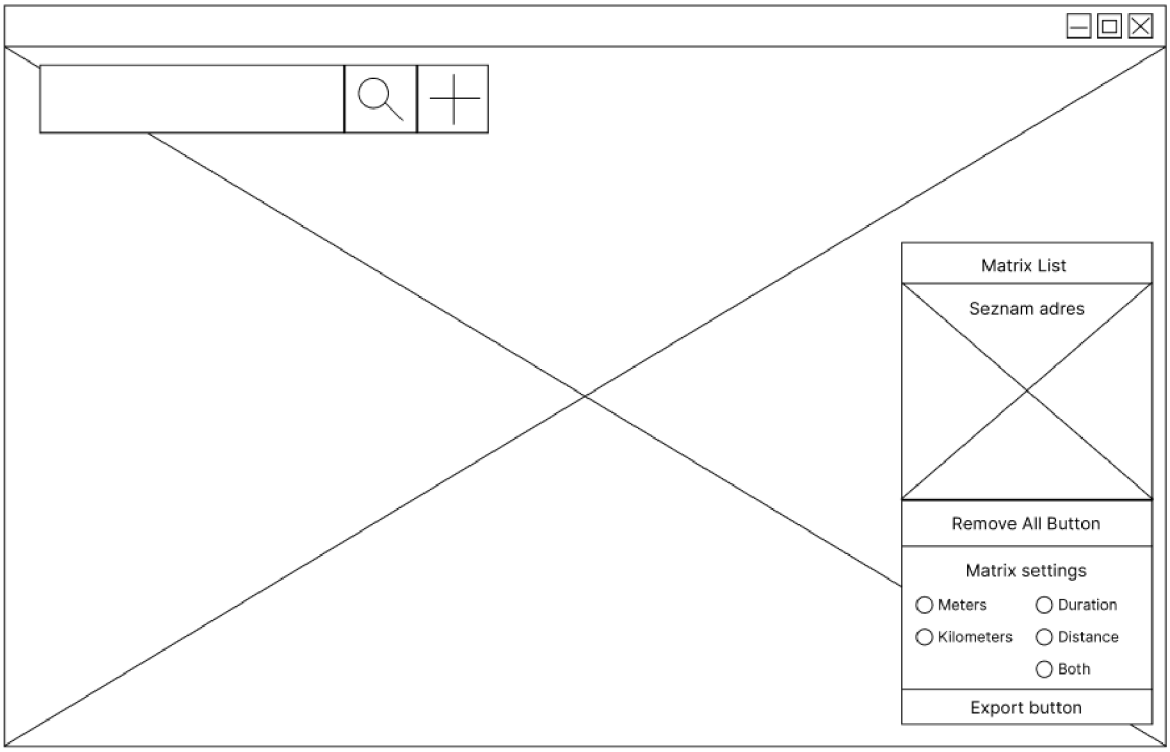
Tato funkce umožňuje uživateli tisknout nebo exportovat data z aplikace do formátu podporovaného programem Microsoft Excel. Tento formát umožňuje uživatelům dále pracovat s daty a provádět různé výpočty, zpracování a analýzy pomocí funkcí programu Excel. Exportovaná data musí být přehledně a srozumitelně organizována, aby bylo pro uživatele snadné s nimi dále pracovat.

**Používání aplikace nepředstavuje dodatečné náklady pro uživatele**

Aby pro uživatele používání aplikace nepředstavovalo dodatečné náklady, bylo potřeba využít bezplatný datový zdroj. Po srovnání více možností bylo rozhodnuto o použití služby Open Route Service, protože poskytuje všechny potřebné funkce, umožňuje vytvoření až 2500 matic a vyhledání až 5000 bodů dle adresy každý den, a hlavně je postavena na datech OpenStreetMap, ke kterým má aplikace přístup díky použití knihovny GMap.Net přístup.

## 4.3 Návrh GUI

V kapitole Návrh GUI se autor věnuje návrhu uživatelského rozhraní pro vyvíjenou aplikaci. V této kapitole autor popisuje, jak vycházel z funkčních požadavků na aplikaci a jak navrhl a implementoval uživatelské rozhraní pomocí Windows Forms v prostředí Visual Studio 2019. Autor se zaměřuje na vzhled a uspořádání ovládacích prvků aplikace, aby byla co nejvíce intuitivní a snadno použitelná pro uživatele.




Obrázek 1- Hlavní okno aplikace, vlastní implementace

### 4.3.1 Hlavní okno aplikace

Základem hlavního okna aplikace je samotný windows formulář, ve kterém se nacházejí všechny další komponenty, které uživatel používá pro ovládání aplikace. Hlavní okno je navrhnuto tak, aby byly všechny ovládací prvky ihned dostupné, ale i přes to uživateli co nejméně překáželi při práci s mapou.

Z toho důvodu jsou prvky rozděleny do tří části. První a hlavní část je samotná mapa, kterou uživatel ovládá pomocí levého, prostředního a pravého tlačítka myši. Pomocí levého uživatel označuje bod, který by chtěl přidat do seznamu. Prostřední tlačítko uživatel používá pro přiblížení, či oddálení mapy. A pomocí pravého tlačítka myši uživatel s mapou pohybuje.

### 4.3.2 Ovládací prvky pro vyhledání a přidání bodu

V levém horním rohu se nachází druhá skupina ovládacích prvků, která se skládá z textBoxu a dvou tlačítek. Účelem této skupiny je uživateli umožnit vyhledání bodu podle adresy a jeho následné přidání. Po zadání adresy do textboxu může uživatel buďto stisknout tlačítko pro vyhledání, nebo klávesu enter. Následně se na požadovaném místě objeví dočasný bod, který může uživatel potvrdit kliknutím na tlačítko Přidat, nebo přepsat vyhledáním jiné adresy, či kliknutím na jinou pozici na mapě.

### 4.3.3 Ovládací prvky pro správu a export dat

Na pravé straně hlavního okna se nachází třetí skupina ovládacích prvků, která uživateli umožňuje správu přidaných bodů a jejich export. Skládá se z labelů, dataGridu, buttonů, radioButtonů a group boxů.

Label neboli popisek je autorem používán k popisu a vysvětlení účelu ovládacích prvků vedle nichž se nachází.

DataGrid autor používá pro zobrazení a správu bodů, které uživatel přidá během používání aplikace.

Buttony jsou používány pro spouštění funkcí jako je například vymazání všech přidaných bodů, nebo tvorba a export matice.

RadioButtony slouží k nastavení požadovaného exportu. Dávají uživateli volbu mezi hodnotami, ve kterých jsou jedna, či více matic exportovány.

GroupBoxy autor využívá k rozdělení jednotlivých RadioButtonů do skupin, aby tak zajistil, že půjde označit vždy jen jeden ze skupiny.

## 4.4 Popis vybraných tříd

V této části autor popisuje fungování významných tříd aplikace.

### 4.4.1 MainForm.cs

MainForm.cs je třída frameworku .NET pro vytvoření formulářů v aplikaci Windows Forms. Formuláře jsou základními prvky v aplikaci Windows Forms a zahrnují všechny prvky uživatelského rozhraní aplikace, jako jsou tlačítka, textová pole, seznamy a další. MainForm.cs třída obsahuje kód pro zpracování událostí a interakcí s uživatelem pro správnou funkci aplikace.

### 4.4.2 MapPoint.cs

MapPoint.cs je třída sloužící k reprezentaci geografického bodu na mapě. Obsahuje tři vlastnosti: ID, Lat a Lon. Vlastnost ID je celé číslo, které slouží jako jedinečný identifikátor bodu. Vlastnosti Lat a Lon jsou desetinná čísla, která představují souřadnice zeměpisné šířky a délky bodu.

```
public class MapPoint
{
        public int ID {get; set;}
        public double Lat {get; set;}
        public double Lon {get; set;}
}
```

Ukázka kódu 1 - Třída MapPoint.cs, vlastní implementace

### 4.4.3 MatrixResponse.cs

MatrixResponse.cs je třída, která se v aplikaci používá k reprezentaci odpovědi z rozhraní API služby OpenRouteService. Obsahuje tři veřejné vlastnosti: destinations, durations a distances, které jsou všechny seznamy. Vlastnost destinations představuje seznam cílových bodů v matici, zatímco vlastnosti durations a distances představují vzdálenosti a délky trvání cesty mezi jednotlivými body v matice. Třída Destinations je definována jako vnořená třída v rámci třídy MatrixResponse a obsahuje jedinou veřejnou vlastnost snapped_distance, která představuje vzdálenost mezi místem vybraným uživatelem a přichyceným místem v síti. Třída MatrixResponse poskytuje způsob ukládání a správy informací z API služby

OpenRouteService, což umožňuje aplikaci snadno přistupovat k těmto informacím a využívat je podle potřeby.

```
class MatrixResponse
{
        public List<Destinations> destinations { get; set; }
        public List<double[]> durations { get; set; }
        public List<double[]> distances { get; set; }
        public class Destinations
        {
                public double snapped_distance { get; set; }
        }
}
```

Ukázka kódu 2 - Třída MatrixResponse.cs, vlastní implementace

### 4.4.4 GmapMarkerWithLabels.cs

GmapMarkerWithLabel je třída vytvořená autorem, která umožňuje uživateli přidat na mapu značku se štítkem pro jednodušší identifikaci.

Třída rozšiřuje třídu GMarkerGoogle, což je vestavěná třída značek poskytovaná knihovnou GMap.NET. Rozšířením této třídy dědí třída GmapMarkerWithLabel všechny funkce třídy GMarkerGoogle, včetně možnosti přidávat ke značkám vlastní ikony a animace. Třída GmapMarkerWithLabel však umožňuje přidávat ke značkám popisky tím, že poskytuje způsob vykreslení řetězce pod ikonou značky.

Konstruktor třídy GmapMarkerWithLabel přijímá tři parametry: umístění, popisek a typ značky. V konstruktoru třída nastaví písmo popisku a vytvoří nový objekt GMarkerGoogle se zadanou polohou a typem.

V metodě OnRender se do značky přidá popisek. Tuto metodu volá knihovna GMap.NET, aby vykreslila značku na mapě. V metodě OnRender třída nejprve zavolá základní metodu OnRender, aby vykreslila ikonu značky. Poté vypočítá velikost řetězce popisku a umístí jej pod ikonu značky pomocí lokální pozice značky. Nakonec třída vykreslí řetězec popisku na mapu pomocí vypočtené pozice a písma.

Třída také obsahuje metodu Dispose, která slouží k uvolnění všech nespravovaných prostředků používaných objektem. Metoda zkontroluje, zda je vnitřní objekt značky nulový, a pokud ne, zlikviduje jej. Třída také implementuje rozhraní ISerializable, které umožňuje serializaci a deserializaci objektů třídy.

```
public class GmapMarkerWithLabel : GMarkerGoogle, ISerializable {
       private Font font;
       private GMarkerGoogle innerMarker;
       public string Caption;
       public GmapMarkerWithLabel(PointLatLng p, string caption, GMarkerGoogleType
type): base(p, type) {
              font = new Font("Arial", 14);
              innerMarker = new GMarkerGoogle(p, type);
              Caption = caption;
       }
       public override void OnRender(Graphics g) {
              base.OnRender(g);
              var stringSize = g.MeasureString(Caption, font);
              var localPoint = new PointF(LocalPosition.X - stringSize.Width / 2,
LocalPosition.Y + stringSize.Height);
       g.DrawString(Caption, font, Brushes.Black, localPoint);
      }
      public override void Dispose() {
             if (innerMarker != null) {
              innerMarker.Dispose();
              innerMarker = null;
             }
             base.Dispose();
      }
      #region ISerializable Members
      void ISerializable.GetObjectData(SerializationInfo info, StreamingContext
context) {
              base.GetObjectData(info, context);
      }

      protected GmapMarkerWithLabel(SerializationInfo info, StreamingContext context):
base(info, context){}

     #endregion
}
```

Ukázka kódu 3 - Třída GmapMarkerWithLabel.cs, vlastní implementace

## 4.5 Popis vybraných metod

V této části autor popisuje fungování vybraných metod aplikace.

### 4.5.1 Vyhledání bodů na mapě

V aplikaci existují dva způsoby, pomocí kterých může uživatel vyhledat požadovaný bod na mapě. Prvním způsobem je vyhledání požadovaného bodu pomocí adresy. Druhým způsobem je manuální nalezení bodu pohybem v mapě a jeho následným výběrem pomocí levého tlačíka myši.

Pro umožnění prvního způsobu byla vytvořena metoda searchAddress(). Tato metoda slouží k získání zeměpisných souřadnic na základě adresy zadané v uživatelském rozhraní. Pokud je textové pole prázdné, metoda se ukončí. Obsahuje-li pole nějaký vstup, je adresa odeslána jako požadavek na OpenRouteService API, které poskytuje geokódovací funkce. Adresa zahrnuje API klíč a textový vstup adresy.

Odpověď ze serveru je uložena do proměnné response a následně zpracována pro další použití. Pokud je odpověď úspěšná, jsou získané zeměpisné souřadnice uloženy do příslušných textových polí v uživatelském rozhraní a mapa je přiblížena na základě těchto souřadnic. Dále je zavolána metoda AddDot(), která na mapu přidá bod, reprezentující hledanou adresu. V opačném případě, pokud se vrátí odpověď s chybovým stavem, metoda vytvoří dialogové okno s chybovou zprávou, která informuje uživatele o chybě při získávání zeměpisných souřadnic pro danou adresu.

```
private async void searchAddress(){
	if (textBox_Address.Text == "") {
		return;
	}
	else {
	string Address = this.textBox_Address.Text;
	var baseAddress = new
Uri($"https://api.openrouteservice.org/geocode/search?api_key=API-
KEY&text={HttpUtility.UrlEncode(Address)}&focus.point.lon=14.418540&focus.point.lat=50
.073658&sources=openstreetmap&layers=address&size=1");
	var httpClient = new HttpClient { BaseAddress = baseAddress };
	httpClient.DefaultRequestHeaders.Clear();
	httpClient.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
	var response = await httpClient.GetAsync(baseAddress);
	var JSONresponseData = response.Content.ReadAsStringAsync().Result;
	var data = JsonConvert.DeserializeObject<GeoSearchResponse>(JSONresponseData);
		if (response.StatusCode.ToString() == "OK") {
		metroTextBox_latitude.Text = data.bbox[1].ToString();
		metroTextBox_longitude.Text = data.bbox[0].ToString();
		LoadMap(new PointLatLng(data.bbox[1], data.bbox[0]));
		map.Zoom = 15;
		AddDot(new PointLatLng(data.bbox[1], data.bbox[0]));
		}
		else {
			MessageBox.Show("Vyskytla se chyba: " +
response.StatusCode.ToString());
		}
	}
}
```

Ukázka kódu 4 - metoda searchAddress(), vlastní implementace

Pro potřeby druhého způsobu existuje metoda map_MouseClick(), která slouží k zpracování události kliknutí myši na mapu. Pokud je na mapu kliknuto levým tlačítkem myši, metoda získá souřadnice místa, kde bylo kliknuto, a uloží je pro další použití. Následně zavolá funkci pro vytvoření objektu, který je poté použit pro vytvoření nového bodu na mapě.

```
private void map_MouseClick(object sender, MouseEventArgs e)
{
	if (e.Button == MouseButtons.Left){
		var point = map.FromLocalToLatLng(e.X, e.Y);
		double lat = point.Lat;
		double lng = point.Lng;
		metroTextBox_latitude.Text = lat.ToString();
		metroTextBox_longitude.Text = lng.ToString();
	}
	var mapPoint = getMapPoint();
	AddDot(mapPoint);
}
```

Ukázka kódu 5 - metoda map_mouseClick(), vlastní implementace

### 4.5.2 Přidání bodů do seznamu

Po nalezení požadovaného bodu přichází na řadu jeho uložení do seznamu. Z tohoto důvodu existuje metoda addPoint(), která získá aktuální zeměpisné souřadnice z metody getMapPoint(). Pokud jsou souřadnice platné, metoda vytvoří nový objekt třídy MapPoint s přiděleným unikátním identifikátorem a získanými zeměpisnými souřadnicemi. Tento objekt je poté přidán do datové kolekce _model a _points. Nakonec je mapa vycentrována metodou LoadMap() a je na ní přidána značka pomocí metody AddMarker().

```
private void addPoint(){
	PointLatLng point = getMapPoint();
	if (this.IsValid())
	{
		MapPoint MP = new MapPoint();
		MP.ID = ++maxID;
		MP.Lat = Convert.ToDouble(this.metroTextBox_latitude.Text);
		MP.Lon = Convert.ToDouble(this.metroTextBox_longitude.Text);
		_model.Data.Add(MP);
		_points.Add(point);
		textBox_Address.Clear();
		LoadMap(point);
	}
	LoadMap(point);
	AddMarker(point);
	textBox_Address.Clear();
}
```

Ukázka kódu 6 - metoda addPoint(), vlastní implementace

### 4.5.3 Tvorba a export matice

Hlavním cílem této aplikace je vytvoření matice pro řešení problému obchodního cestujícího. Jelikož se o tvorbu matice stará externí API, je nutné nejprve složit požadavek. O vytvoření a odeslání požadavku se stará hned několik metod. První uvedená je metoda getLocationsString(), která vytváří řetězec souřadnic, které mají být zahrnuty v žádosti o matici. Metoda projde seznam uložených bodů a vytvoří řetězec souřadnic ve formátu očekávaném API serverem.

```
private string getLocationsString()
{
	string locationsString = "";
	foreach (var a in _points)
	{
		locationsString += $"[{a.Lng.ToString()},{a.Lat.ToString()}],";
	}
	locationsString = locationsString.Remove(locationsString.Length - 1);
	return locationsString;
}
```

Ukázka kódu 7 - metoda getLocationsString(), vlastní implementace

Druhá metoda je getDistance(), která dle zaškrtnutého radioboxu v uživatelském rozhraní získává jednotku vzdálenosti, která má být použita v žádosti. Metoda vrací řetězec reprezentující jednotku kilometrů nebo metrů.

```
private string getDistanceUnit(){
	string distanceUnitString = "";
	if (radioButton_km.Checked){
		distanceUnitString = "\"km\"";
	}
	else if (radioButton_m.Checked){
		distanceUnitString = "\"m\"";
	}
	return distanceUnitString;
}
```

Ukázka kódu 8 - metoda getDistanceUnit(), vlastní implementace

Třetí je metoda getMetrics(), která získává metriky, jež mají být použity v požadavku. Metoda vrací řetězec reprezentující jednu nebo obě metriky, časovou dobu a vzdálenost, v závislosti na výběru uživatele.

```
private string getMetrics()
{
	string text = "";
	if (radioButton_both.Checked){
		text = "\"metrics\":[\"duration\",\"distance\"],";
	}
	else if (radioButton_duration.Checked){
		text = "\"metrics\":[\"duration\"],";
	}
	else if (radioButton_distance.Checked){
		text = "\"metrics\":[\"distance\"],";
	}
	else{
		text = "";
	}
	return text;
}
```

Ukázka kódu 9 - metoda getMetrics(), vlastní implementace

Tyto metody jsou dále využívány v metodě GetMatrixRequestAsync(), která slouží k získání výsledků vzdálenostní matice z externího API serveru. V metodě se nejprve vytvoří instanci třídy Uri do které se vloží adresa API serveru. Dále se pomocí metod getLocationsString(), getMetrics() a getDistanceUnit() získají hodnoty pro řetězce locations, metrics a DistanceUnit, ze kterých se sestaví řetězec s parametry požadavku. Poté se vytvoří instance třídy HttpClient, která zajišťuje komunikaci s API, a na tuto instanci se nastaví hlavičky požadavku. Následuje vytvoření proměnné StringContent, která obsahuje data požadavku ve formátu JSON. Dále se zavolá metoda PostAsync, která odešle požadavek na API. Po obdržení JSON odpovědi se provede její deserializace na instanci třídy MatrixResponse, která je následně metodou vrácena.

```
private async Task<MatrixResponse> GetMatrixRequestAsync()
{
        var baseAddress = new Uri("https://api.openrouteservice.org");
        string locationsString = getLocationsString();
        string metricsString = getMetrics();
        string DistanceUnitString = getDistanceUnit();

        System.Net.ServicePointManager.ServerCertificateValidationCallback = delegate {
return true; };
        var httpClient = new HttpClient { BaseAddress = baseAddress };
        httpClient.DefaultRequestHeaders.Clear();
        httpClient.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
        httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer",
"5b3ce3597851110001cf624862b9d0b7adb740dca7164b009ae72f6d");

        var contentString = "{\"locations\":[" + locationsString + "]," + metricsString
+ "\"units\":" + DistanceUnitString + "}";
        var content = new StringContent(contentString, Encoding.UTF8,
"application/json");

        HttpResponseMessage httpResponseMessage = await
httpClient.PostAsync("/v2/matrix/driving-car", content);
        var response = httpResponseMessage;
        var JSONresponseData = response.Content.ReadAsStringAsync().Result;
        var data = JsonConvert.DeserializeObject<MatrixResponse>(JSONresponseData);

        return data;
}
```

Ukázka kódu 10 - metoda GetMatrixRequestAsync(), vlastní implementace

Po obdržení odpovědi je nutné data exportovat do formátu podporovaného aplikací Excel. Jelikož všechny knihovny, které se autor během vývoje pokusil použít pro usnadnění tvorby a zápisu do souboru byly závislé na přítomnosti aplikace Excel na počítači, byl autor nucen vytvořit vlastní parser pro zpracování MatrixResponse objektů a jejich konverzi do souboru podporovaného aplikací Excel. Výhodou vlastního parseru je to, že je možné naprogramovat jen ty funkcionality, které jsou v aplikaci potřebné a tím snížit náročnost aplikace.

Vytvořený parser se skládá z několika metod. První volaná je metoda ParseMatrixToExcel(), která převádí objekt MatrixResponse obdržený ve vstupu na XML řetězec podporovaný aplikací Excel.

```
private string ParseMatrixToExcelXML(MatrixResponse data)
{
	string beginning = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><?mso-application
progid=\"Excel.Sheet\"?><Workbook xmlns=\"urn:schemas-microsoft-
com:office:spreadsheet\" xmlns:x=\"urn:schemas-microsoft-com:office:excel\"
xmlns:ss=\"urn:schemas-microsoft-com:office:spreadsheet\"
xmlns:html=\"https://www.w3.org/TR/html401/\">";
	string end = "</Workbook>";
	string middle = CreateMatrixSheets(data);
	string input = beginning + middle + end;
	return input;
}
```

Ukázka kódu 11 - metoda ParseMatrixToExcelXML(), vlastní implementace

K vytvoření tohoto řetězce slouží metoda CreateMatrixSheets(), ve které se nejprve vytvoří konstanty distanceSheetName a durationsSheetName, což jsou názvy listů, které slouží pro uložení dat o vzdálenostech a časech cestování. Poté se zkontroluje, zda chce uživatel exportovat jen jednu, nebo obě matice a zavolá se metoda CreateSheet().

```
private string CreateMatrixSheets(MatrixResponse data)
{
	string output = "";
	const string distanceSheetName = "Distance Sheet";
	const string durationsSheetName = "Durations Sheet";

	if (radioButton_both.Checked)
	{
		List<double[]> distanceArray = data.distances;
		List<double[]> durationsArray = data.durations;
		output = CreateSheet(distanceArray, distanceSheetName) +
CreateSheet(durationsArray, durationsSheetName);
	}
	else if (radioButton_distance.Checked)
	{
		List<double[]> distanceArray = data.distances;
		output = CreateSheet(distanceArray, distanceSheetName);

	}
	else if (radioButton_duration.Checked)
	{
		List<double[]> durationsArray = data.durations;
		output = CreateSheet(durationsArray, durationsSheetName);
	}

	return output;
}
```

Ukázka kódu 12 - metoda CreateMatrixSheets(), vlastní implementace

Tato metoda slouží k vytvoření jednoho listu s daty v Excel XML formátu. CreateSheet() přijímá jako vstupní parametry seznam dat a název listu, který má být vytvořen. Metoda obsahuje cyklus, který projde jednotlivé řádky dat v seznamu a pro každý řádek se vytvoří XML značky pro vytvoření řádku v tabulce. V dalším vnořeném cyklu se pro každý sloupec v řádku vytvoří XML značky pro buňku v tabulce, do kterých se vloží hodnota konkrétního prvku v seznamu.

```
private string CreateSheet(List<double[]> data, string sheetName){
	string start = "<Worksheet ss:Name=\"" + sheetName + "\"><Table>";
	string middle = "";
	for (int arrayDurationRows = 0; arrayDurationRows < data.Count;
arrayDurationRows++){
		middle += "<Row>";
		for (int arrayDurationCols = 0; arrayDurationCols <
data[arrayDurationRows].Length; arrayDurationCols++){
			middle += "<Cell><Data ss:Type=\"Number\">"
data[arrayDurationRows][arrayDurationCols] + "</Data></Cell>";
		}
		middle += "</Row>";
	}
	string end = "</Table></Worksheet>";
	string output = start + middle + end;
	return output;
}
```

Ukázka kódu 13 - metoda CreateSheet(), vlastní implementace

ExportMatrixArrayToXML() následně zapíše získaný XML řetězec jako soubor na místo vybrané uživatelem, které je jí předáno po stisknutí tlačítka s názvem getMatrix.

```
private void ExportMatrixArrayToXML(string fileName, MatrixResponse data)
{
	try
	{
		using (FileStream fs = File.Create(fileName))
		{
			string input = ParseMatrixToExcelXML(data);
			byte[] info = new UTF8Encoding(true).GetBytes(input);
			fs.Write(info, 0, info.Length);
		}
	}
	catch (Exception ex)
	{
		Console.WriteLine(ex.ToString());
	}
}
```

Ukázka kódu 14 - metoda ExportMatrixArrayToXML(), vlastní implementace

## 4.6 Testování

### 4.6.1 Cíl testování

Cílem testování aplikace je ověřit, zda aplikace splňuje všechny požadavky a funkcionality, které byly definovány v první části bakalářské práce. Testování zahrnuje ověření správného chování aplikace, její funkčnosti a uživatelské přívětivosti.

### 4.6.2 Použité testovací metody

Při testování aplikace byly použity různé metody testování, aby bylo zajištěno pokrytí co nejvíce scénářů a situací. Mezi použité metody patří manuální testování a testování jednotek.

**Manuální testování**

Manuální testování bylo prováděno s cílem ověřit chování aplikace z pohledu běžného uživatele. Testování zahrnovalo ověření všech funkcionalit, které byly popsány. Byly testovány například následující funkce:

- Přidávání bodů do seznamu adres
- Odstranění bodů ze seznamu adres
- Tvorba a export matice sazeb do Excelu

Testování probíhalo na různých verzích operačního systému Windows 10 a různých verzích programu MS Excel, včetně počítače bez MS Excel nainstalovaného. Dalším faktorem v testování bylo různé jazykové nastavení systému, nebo například verze přítomného .NET frameworku.

**Testování jednotek**

Testování jednotek bylo použito k ověření správnosti jednotlivých metod a funkcí aplikace. Testování bylo prováděno pomocí testovacích knihoven pro jazyk C#, které umožňují automatizované testování jednotek kódu.

Byly testovány například následující metody:

- Metoda pro vytvoření matice sazeb
- Metoda pro export matice sazeb do Excelu
- Metoda pro import výsledné trasy z Excelu

### 4.6.3 Výsledky testování

Během testování byly identifikovány dva problémy. Prvním z nich byla závislost knihovny používané k exportu matice na přítomnosti programu MS Excel na počítači, což zamezilo uživatelům, kteří nemají Excel nainstalovaný na svých počítačích, úspěšně exportovat matici sazeb. Druhým problémem bylo rozdílné jazykové nastavení systému, což způsobilo, že místo desetinné tečky byla do API požadavku vkládána desetinná čárka.

První problém byl odstraněn vytvořením vlastního parseru, který je nezávislý na přítomnosti programu MS Excel na počítači uživatele.

Druhý problém byl odstraněn vynucením vkládání desetinné tečky v kódu.

# 5. Výsledky a diskuse

Výsledkem bakalářské práce je úspěšně vytvořená aplikace, která splňuje všechny požadavky, které byly stanoveny v počátečním zadání práce. Během vývoje a testování se však autor setkal s několika výzvami a problémy, které bylo třeba překonat. Mezi tyto problémy patřila například závislost knihoven na přítomnosti programu Excel na počítači, nebo jazykové nastavení systému, které měnilo desetinné tečky na čárky, kvůli čemuž aplikace špatně odesílala API požadavek. Tyto problémy byly však úspěšně vyřešeny. Jedním z řešení byla tvorba vlastního parseru, který převádí objekt matice do formátu XML podporovaného aplikací Excel. Díky tomu již uživatel nemusí mít program Excel nainstalován na svém počítači a může využívat aplikaci bez omezení. Tento přístup také umožňuje větší flexibilitu a nezávislost na specifických verzích programu Excel a na rozdíl od externích knihoven tolik nezatěžuje aplikaci.

Z výsledků testování vychází, že výsledná aplikace je snadno ovladatelná a uživatelsky přívětivá. Aplikace uživatelům umožňuje buď kliknutím, nebo zadáním adresy do vyhledávacího pole, přidat určitý počet bodů do mapy, jejichž zeměpisné souřadnice se poté pomocí HTTP požadavku pošlou na externí API, které nám obratem vrátí vzdálenosti mezi jednotlivými body. Tyto vzdálenosti jsou následně exportovány do souboru podporovaného programem Excel, který uživatel může dále využít v součinnosti s pluginem TSPKOSA pro nalezení optimálního řešení problému obchodního cestujícího.

V budoucnu by autor aplikaci rád rozšířil o další funkce, které by uživatelům usnadnily řešení problému obchodního cestujícího. Jednou z možností by mohla být implementace funkce automatického vyhledání nejkratší cesty mezi body, což by uživatelům usnadnilo hledání optimální trasy. Další možností by mohla být funkce načítání dat z různých zdrojů, jako jsou například databáze s geografickými informacemi nebo soubory ve formátu KML/KMZ. Tím by uživatelům umožnila snadněji pracovat s velkým množstvím dat a rychle vytvářet matice pro řešení problému obchodního cestujícího.

V neposlední řadě by se dalo uvažovat o rozšíření aplikace na jiné operační systémy, například pro MacOS nebo Linux.

Jelikož se na rozdíl od Google API jedná o bezplatné řešení, má aplikace také určitá omezení. V aktuální podobě může aplikace exportovat maximálně 2500 matic za jeden den a vyhledat 5000 bodů pomocí zadání jeho adresy.

Celkově tedy aplikace uživatelům znatelně šetří čas při tvorbě matice pro hledání řešení k problému obchodního cestujícího.

# 6. Závěr

Bakalářská práce se zaměřila na vývoj aplikace, která by umožnila rychlé a efektivní vytváření matic pro výpočet problému obchodního cestujícího. Dříve to byl časově náročný proces, který byl spojen s ručním zadáváním jednotlivých bodů, což znamenalo, že vytvoření matice mohlo trvat i desítky minut a uživatele stát i více než 100 dolarů.

Autorova aplikace však tento proces výrazně urychlila, zjednodušila a zlevnila. Uživatelé nyní mohou pohodlně používat jedinou aplikaci, která během pár kliknutí vygeneruje rozsáhlé matice. To jim umožňuje rychleji a efektivněji řešit problém obchodního cestujícího a věnovat se dalším úkolům.

Během vývoje aplikace se autor práce setkal s mnoha výzvami, jako například s tvorbou intuitivního uživatelského rozhraní, optimalizací výkonu a správným řešením chyb a chybových stavů. Jedna z větších výzev byla tvorba vlastního způsobu exportu dat do Excelu, který byl donucen vyvinout, protože během testování zjistil, že ne každý má v dnešní době nainstalovaný Excel, a že rozšíření, které původně používal bylo na přítomnosti Excelu na počítači závislé.

Lze říci, že výsledná aplikace je užitečným nástrojem pro všechny, kteří se zabývají řešením problému obchodního cestujícího. Tato práce autorovi umožnila rozšířit své znalosti a zkušenosti v oblasti softwarového vývoje a přispěla k jeho osobnímu i profesionálnímu růstu. Jsem rád, že má aplikace najde praktické uplatnění a usnadní práci ostatním.

# 7. Seznam použitých zdrojů


1. Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press. 2006. 592 s. ISBN 9780691129933.
2. COOK, William J. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton*, Princeton University Press, 2012. 224 s. ISBN 978-0691152707.
3. LAMONT, Ian. *Excel Basics In 30 Minutes: The quick guide to Microsoft Excel and Google Sheets*. 2. vydání. i30 Media Corp., 2018. 104 s. ISBN 9781939924353.
4. Petzold, C. Code: *The Hidden Language of Computer Hardware and Software*. Microsoft Press., 2000. 400 s. ISBN 9780735634688. - zkonrolovat
5. HOFFMAN, Chris. What is an operating system? *How-To Geek - We Explain Technology* [online]. 8.8.2018. [cit. 26.2.2023]. Dostupné z: https://www.howtogeek.com/361572/what-is-an-operating-system/
6. Autor neznámý. Desktop Operating System Market Share Worldwide. *Statcounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share* [online]. [cit. 27.2. 2023]. Dostupné z: https://gs.statcounter.com/os-market-share/desktop/worldwide
7. MUCHMORE, Michael. Windows, macos, Chrome OS, or linux: Which operating system is best? *PCMAG* [online]. 9.1.2023. [cit. 27.2.2023]. Dostupné z: https://www.pcmag.com/picks/windows-vs-macos-vs-chrome-os-vs-ubuntu-linux-which-operating-system-reigns
8. Pressman, Roger S. Software Engineering: A Practitioner's Approach. 8. vydání, McGraw-Hill Education, 2014. 965 s. ISBN 9781260423310.
9. MICROSOFT. Integrované Vývojové prostředí (IDE) a editor Kódu Pro Vývojáře softwaru a týmy. *Visual Studio* [online]. [cit. 27.2.2023]. Dostupné z: https://visualstudio.microsoft.com/cs/
10. MICROSOFT. Documentation for visual studio code. *Visual Studio Code - Code Editing. Redefined* [online]. 3.11.2021. [cit. 27.2.2023]. Dostupné z: https://code.visualstudio.com/docs
11. KOLADE, Chris. Visual Studio vs Visual Studio Code – What's The Difference Between These IDE Code Editors? *freeCodeCamp Programming Tutorials: Python, JavaScript, Git & More* [online]. 2019. [cit. 26.2.2023]. Dostupné z: https://www.freecodecamp.org/news/visual-studio-vs-visual-studio-code/
12. STEVENS, Emily. What is Ui Design? A complete introductory guide. *UX Design Institute* [online]. 22.3.2022. [cit. 26.2.2023]. Dostupné z: https://www.uxdesigninstitute.com/blog/what-is-ui-design/
13. Granor, T. E., McPeak, J., & Perry, M. W. Text-Based User Interfaces. Manning Publications. 2019. 328 s. ISBN 1617295344.
14. Cooper, A., Reimann, R., & Cronin, D**.** A Guide to Graphic User Interface Design. John Wiley & Sons. 2014. 720 s. ISBN 978-1-118-41664-2.

15. SOEGAARD, Mads, DAM, Rikke Friis and ZUSCHKE, Michael, 2021, What is User Interface (UI) design? *The Interaction Design Foundation* [online]. 23.9.2021. [cit. 27.2.2023]. Dostupné z: https://www.interaction-design.org/literature/topics/ui-design
16. Pratt, T. W., & Zelkowitz, M. V. Programming Languages: Design and Implementation (6. vydání). Pearson. 2019. 895 s. ISBN: 9780133943023.
17. Harbison, J., Albahari, B. C# 10 in a Nutshell: The Definitive Reference (7. vydání), O'Reilly Media, 2021, 1064 s. ISBN: 978-1098101602.
18. HEGDE, Jayashree, 2022, C# vs C++: Difference between C# and C++. *InterviewBit: Coding Interview Questions* [online]. 19.7.2022. [cit. 27.2.2023]. Dostupné z: https://www.interviewbit.com/blog/c-sharp-vs-cpp/
19. MICROSOFT, .NET Core Introduction. [online] [cit. 27.2.2023]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction.
20. MICROSOFT, Dokumentace Pro .NET, *Microsoft Learn* [online], [cit. 27.2.2023]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/fundamentals/
21. MICROSOFT, Přehled používání ovládacích prvků – windows forms .NET. *Přehled používání ovládacích prvků - Windows Forms .NET | Microsoft Learn* [online]. [cit. 27.2.2023]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/desktop/winforms/controls/overview?view=netdesktop-7.0&viewFallbackFrom=netdesktop-6.0
22. SELLS, Chris, Weinhardt, Michael, Windows Forms 2.0 Programming. Addison-Wesley Professional, 2006, 528 s. ISBN: 9780321267962.
23. LABBÉ, Jérôme, Dokumentace Pro GMAP.NET Framework. *GitHub* [online]. 26.12.2018. [cit. 27.2.2023]. Dostupné z: https://github.com/judero01col/GMap.NET
24. FIELDING, Roy T., a Richard N. TAYLOR. *Principy pro navrhování moderních webových aplikací*. 1. vydání. Praha: Grada, 2019. ISBN 978-80-271-0003-6.
25. OPENROUTESERVICE, OpenRouteService. *OpenRouteService* [online]. [cit. 28.2.2023]. Dostupné z: https://openrouteservice.org/
26. GeeksforGeeks, *Understanding HTTP Basics and HTTP Request Methods*. GeeksforGeeks [online] 21.10.2021. [cit. 27.2.2023]. Dostupné z: https://www.geeksforgeeks.org/understanding-http-basics-and-http-request-methods/.
27. OpenStreetMap. (n.d.). About OpenStreetMap. [online] [cit. 28.2.2023]. Dostupné z: https://www.openstreetmap.org/about

28. JAKUBEC, Jan. *GitHub: platforma pro spolupráci na softwarových projektech* ITnetwork. [online]. 2022 [cit. 28.2.2023]. Dostupné z: https://www.itnetwork.cz/software/agile-cms/github-platforma-pro-spolupraci-na-softwarovych-projektech

# 8. Seznam obrázků, tabulek, grafů a zkratek

## 8.1 Seznam obrázků



## 8.2 Seznam ukázek kódu