

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ



Katedra ekonomiky a řízení ve stavebnictví

BAKALÁŘSKÁ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Macek** Jméno: **Tomáš** Osobní číslo: **502296**
Fakulta/ústav: **Fakulta stavební**
Zadávací katedra/ústav: **Katedra ekonomiky a řízení ve stavebnictví**
Studijní program: **Management a ekonomika ve stavebnictví**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Základní využití nástrojů programování ke zvýšení efektivity standardních činností v oblasti kalkulací a rozpočtování

Název bakalářské práce anglicky:

Basic utilization of programming tools to increase the efficiency of standard costing and budgeting activities

Pokyny pro vypracování:

- Úvod, cíle práce, zvolená metodika
- Teoretický úvod do problematiky - programovací nástroje, potřeby kalkulací a rozpočtování
- Možnosti využití programování nástrojů pro potřeby kalkulací a rozpočtování
- Praktická aplikace - tvorba příkladu uplatnění
- Závěr, vyhodnocení, shrnutí dosažení cílů, diskuze

Seznam doporučené literatury:

Summerfield, M.: Python 3 Výukový kurz. Computer Press, 2021. ISBN 978-80-251-2737-7.
Schneiderová Heralová, R., Vitásek, S., Brožová, L., Střelcová, I.: Oceňování staveb, FSv ČVUT, 2020.
Schneiderová Heralová, R.; Vitásek, S.; Strnad, M.; Střelcová, I.: Kalkulace nákladů ve stavebnictví, FSv ČVUT, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Matějka, Ph.D. katedra ekonomiky a řízení ve stavebnictví FSv

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **25.02.2024** Termín odevzdání bakalářské práce: **27.05.2024**

Platnost zadání bakalářské práce: _____

Ing. Petr Matějka, Ph.D.
podpis vedoucí(ho) práce

prof. Ing. Renáta Schneiderová Heralová, Ph.D.
podpis vedoucí(ho) ústavu/katedry

prof. Ing. Jiří Máca, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studenta

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, pouze za odborného vedení vedoucího bakalářské práce *Ing. Petra Matějky, Ph.D.*

Dále prohlašuji, že veškeré podklady, ze kterých jsem čerpal, jsou uvedeny v seznamu použité literatury.

V Praze dne

Tomáš Macek

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce Ing. Petru Matějkovi, Ph.D. za jeho rady, trpělivost a čas věnovaný pomoci a konzultacím.

Dále bych chtěl poděkovat Ing. Ivetě Střelcové za poskytnutí dat na analýzu a konzultaci s problematikou v oboru kalkulací a rozpočtování.

V Praze dne

Tomáš Macek

**Základní využití nástrojů programování ke zvýšení efektivity
standardních činností v oblasti kalkulací a rozpočtování**

**Basic utilization of programming tools to increase the
efficiency of standard costing and budgeting activities**

Abstrakt

Tato bakalářská práce se zabývá využitím programování v oblasti kalkulací a rozpočtování. Cílem práce je zvážit potenciál programování pro zefektivnění činností v této oblasti. Práce v teoretické části představuje problematiku oboru kalkulací a rozpočtování a prezentuje základní možnosti využití programování v této oblasti. Následně analyzuje činnosti oboru kalkulací a rozpočtování, které je možné zefektivnit. V praktické části demonstruje práce vytvoření jednoduché aplikace s využitím programovacího jazyka Python. Aplikace identifikuje shodné položky napříč rozpočty rodinných domků. Opakující se položky zapisuje do šablony pro zefektivnění tvorby nových rozpočtů. Součástí práce je také manuál pro programování v oblasti kalkulací a rozpočtování, který popisuje použitý postup a užitečné rady pro další využití. Výsledky práce ukazují, že programování je důležitý nástroj s potenciálem pro budoucí využití v oboru kalkulací a rozpočtování. Vytvořená aplikace a manuál slouží jako praktický nástroj a zároveň inspirace pro další využití v této oblasti.

Abstract

This bachelor thesis deals with the use of programming in the field of calculations and budgeting. The aim of the thesis is to consider the potential of programming to streamline activities in this area. The theoretical part of the thesis introduces issues in the field of calculations and budgeting and presents basic opportunities for using programming in this area. Subsequently, it analyzes activities in the field of calculations and budgeting that can be streamlined. In the practical part, the thesis demonstrates the creation of a simple application using the Python programming language. The application identifies common items across family home budgets. Repeating items are recorded in a template to streamline the creation of new budgets. The thesis also includes a manual for programming in the field of calculations and budgeting, describing the procedure used and useful tips for further utilization. The results of the thesis show that programming is an important tool with potential for future use in the field of calculations and budgeting. The created application and manual serve as practical tools and inspiration for further use in this area.

Klíčová slova

Kalkulace a rozpočtování, programování, umělá inteligence, python, automatizace, BIM, vývojový proces, činnosti rozpočtáře, rozpočet, rodinný domek, manuál, aplikace.

Key words

Calculation and budgeting, programming, artificial intelligence, Python, automation, BIM, development process, quantity surveyor activities, budget, family house, manual, application.

Obsah práce

Úvod	1
Stanovení cílů práce	3
Slovníček.....	4
1. Teoretická část	6
1.1 Vymezení oblastí využití programování v kalkulacích a rozpočtování	7
1.2 Možnosti využití programování v kalkulacích a rozpočtování.....	8
1.3 Výběr programovacího jazyka	9
1.4 Počáteční příprava.....	11
1.5 Dostupné programovací nástroje a metodiky	13
1.6 Využití umělé inteligence	15
1.7 Logická mapa procesu	18
2. Praktická část.....	20
2.1 Příprava vývoje	21
2.2 Vývoj aplikace.....	22
2.3 Demonstrace výstupu.....	32
2.4 Nejčastější chyby a čemu předejít.....	33
2.5 Stručný manuál: Programování pro kalkulace a rozpočtování.....	35
Závěr	36
Vyhodnocení cílů	37
Diskuse	38
Použité zdroje.....	39
Seznam obrázků	41
Seznam tabulek	42
Seznam příloh.....	43

Úvod

Tato práce se zaměřuje na identifikaci možných oblastí v oboru kalkulace a rozpočtování, ve kterých je vhodná aplikace programovacích nástrojů pro zvýšení efektivity práce. Programování stále ještě mnoho lidí považuje za příliš složité a odborné. A to zejména ti, kteří nemají specifické vzdělání v oboru tvorby programů. S dnešní dostupností uživatelsky jednoduchých pomocných nástrojů ale můžeme využít programování pro zvýšení efektivity běžných činností i bez detailní znalosti problematiky vývoje softwaru.

Kalkulace a rozpočtování je důležitý proces, který je součástí každého stavebního projektu. Ve výsledku ovlivňuje následnou realizaci a finanční zdraví stavby. Tradičně se tyto procesy provádí převážně manuálně, tedy zadáváním vstupních dat uživatelem. To přináší riziko chyb, malou efektivitu a nedostatečnou transparentnost. Se stále výrazněji nastupující digitalizací ve stavitelství, včetně oblasti kalkulace a rozpočtování jsou programovací nástroje o to více relevantní.

Cílem teoretické části je prozkoumat možný potenciál programování jako nástroje pro zvýšení efektivity procesů v oblasti kalkulace a rozpočtování. Zaměříme se zde na vymezení oblastí, které programování zefektivní nejvíce. Představíme si dostupné programovací jazyky vhodné pro tento obor a nástroje vhodné pro pomoc při tvorbě programů. Jedním z těchto nástrojů, které si představíme bude umělá inteligence. Ukážeme si nové možnosti, které přináší a jaké je její vhodné využití v oboru kalkulace a rozpočtování.

Teoretická část poskytne ucelený pohled na možné výhody, které programování do oboru kalkulace a rozpočtování přináší. Položí také základy pro praktickou část, kde budeme poznatky, nástroje a metody aplikovat a jejich přínos budeme demonstrovat na konkrétním případě.

Praktická část práce bude demonstrovat postup, jak vyřešit konkrétní zadání se zaměřením na rozpočtování rodinných domků pomocí naprogramování aplikace. Projde krok za krokem vytvoření aplikace v programovacím jazyce Python, který společně s analýzou vstupních dat vygeneruje soubor opakujících se položek (tzv. "Common body"). Tento soubor bude možné následně importovat do rozpočtářského programu KROS 4. Soubory vytvořené výsledné aplikace včetně testovacích dat jsou součástí této práce ve formě přílohy.

Součástí praktické části je také vytvořený stručný manuál, který slouží jako návod pro čtenáře, kteří by chtěli replikovat proces vývoje aplikace, nebo chtějí začít se základy programování v libovolné oblasti.

V závěru jsou popsány výsledky práce a vyhodnocení stanovených cílů. Diskuse obsahuje návrhy na případná rozšíření řešeného tématu a diskutuje se zde o výsledcích, které byly výstupem při použití vytvořené aplikace na testovacích datech.

Bakalářská práce obsahuje poznatky, které mohou být využity pro inovaci v oblasti kalkulací a rozpočtování. Prezentuje také nástroje, které jsou vhodné pro budoucí automatizaci jednotlivých procesů a prostřednictvím názorné ukázky motivuje čtenáře k jejich využívání.

Stanovení cílů práce

1.cíl: Identifikovat potenciál programování v oblasti kalkulací a rozpočtování

Cílem bakalářské práce je na základě rešerše a praktické aplikace analyzovat možnosti využití programování v oblasti kalkulací a rozpočtování. Identifikuje konkrétní činnosti vhodné pro automatizaci, nebo zefektivnění při využití programování a zhodnotí přínosy zvoleného přístupu.

2. cíl: Vytvořit aplikaci pro demonstraci praktického využití programování v kalkulacích a rozpočtování

Druhým praktickým cílem práce je vytvořit jednoduchou aplikaci, která bude schopna analyzovat opakující se položky v sadě rozpočtů rodinných domků. Výstupem aplikace bude automaticky vytvořená šablona rozpočtu s předvyplněnými položkami, která zefektivní budoucí práci rozpočtáře.

3. cíl: Vypracovat stručný manuál pro programování v oblasti kalkulací a rozpočtování

Třetím cílem je vypracovat jednoduchý manuál, který popíše základní kroky programování aplikace a poskytne užitečné rady pro začátečníka. Manuál bude cílit na využití programování v oblasti kalkulací a rozpočtování. Součástí bude vybraný programovací jazyk, užitečné pomocné nástroje a postupy včetně zdrojů.

Slovníček

Skript –

„Skript je program zapsaný skriptovacím jazykem. Obvyklá podoba skriptu je soubor. V něm je uložený program, jehož kód je napsán ve speciálním skriptovacím jazyce. Typickými skriptovacími jazyky jsou například PHP či Python. Skript je tvořen souvislou sérií příkazů či algoritmem, který má určitý úkol, například odeslat vyplněný formulář na webu.“ (12)

Knihovna –

„Databáze hotových kódů, které v daném programovacím jazyce řeší určitou funkci, např. kompresi dat, připojení k databázi nebo vykreslování grafů. Vývojář v knihovně najde i návod, který mu usnadní její použití v praxi.“ (11)

Syntaxe –

“Soubor pravidel, která definují kombinaci symbolů, které jsou považovány za správně strukturovaný dokument nebo fragment v tomto jazyce. Platí pro programovací i značkovací jazyky.“ (12)

Software –

„Software je programové vybavení počítače – tedy programy a aplikace v počítači. Software rozdělujeme na aplikační a systémový. Systémový software zajišťuje chod počítače a aplikační software jsou programy, s nimiž pracuje uživatel.“ (12)

Aplikace –

„Aplikace jsou typem software, který slouží uživateli k okamžitému použití a pro různé práce a úkoly. Existují různé typy aplikací, například je můžeme dělit dle toho, pro jaký operační systém jsou určeny.“ (12)

Bug –

„Chyba v softwaru, kvůli které se program nebo aplikace nechová tak, jak má - např. spadne, zamrzne nebo zobrazí špatně grafiku. Odstranění bugu se nazývá bug fix.“ (11)

Kompilátor –

„Kompilátor je program, který převádí zdrojový kód na spustitelný program.“ (12)

Chatbot –

„Jde o počítačový program, který simuluje lidskou konverzaci nebo chatování skrze umělou inteligenci.“ (12)

Benchmark –

„Benchmark je v podstatě srovnávací výkonový test. Populární benchmarky se týkají především výpočetní techniky. V takovém případě se jednotlivá zařízení (počítače, smartphony, tablety nebo i jen samostatné procesory) podrobují zátěžovým zkouškám. Měřit se může rychlost provedení určitých matematických operací.“ (12)

BIM –

„BIM je organizovaný přístup ke sběru a využití informací napříč projektem. Ve středu tohoto úsilí leží digitální model obsahující grafické a popisné informace o designu, konstrukcích a údržbě objektů.“ (13)

1. Teoretická část

1.1 Vymezení oblastí využití programování v kalkulacích a rozpočtování

Kalkulace a rozpočtování je obor, který se zjednodušeně věnuje vytváření podkladů pro ocenění stavebních prací a projektů. Pracovní činnosti v tomto oboru jsou tedy činnosti prováděné rozpočtářem, které definuje databáze povolání spravovaná Ministerstvem práce a sociálních věcí České republiky:

„Pracovní činnosti

- *Zajištění všech podkladů nutných pro rozpočtování projektu.*
- *Určení druhů použitých konstrukcí patrných dle projektové dokumentace.*
- *Určení druhů výkonů dle projektové dokumentace.*
- *Určení druhů konstrukcí a výkonů, které projekt nezmiňuje, ale jsou nutné k zajištění nebo provedení projektovaných konstrukcí.*
- *Určení druhů stavebních materiálů z projektové dokumentace.*
- *Zpracování výkazu výměr a výpočet množství stavebního materiálu.*
- *Výpočet jednotkové ceny konstrukcí, výkonu a materiálu s použitím ceníku stavebních prací nebo kalkulací ceny podle vzorce nebo vlastní úvahou.*
- *Určení, výpočet nákladů a výkonů vyplývajících ze specifických podmínek konkrétní stavby vyvolaných umístěním stavby nebo požadavkem stavebníka.*
- *Sestavení podkladů k následným činnostem procesu přípravy stavby – údaje o plánované spotřebě času pracovníků, strojů a materiálu při realizaci stavby.*
- *Určení sazby DPH, rozpočtování podle aktuálně platné vyhlášky.*
- *Kontrola správnosti zpracování rozpočtu.“ (1)*

Jedná se o detailní úkony, které nemusí mít jen jedno řešení nebo správný výsledek, protože každý projekt je něčím jedinečný. Vzniká mnoho činností prováděných lidmi, které jsou náchylné k chybám a má na ně vliv lidský faktor. To může vést k chybám, nedostatečné transparentnosti a neschopnosti zpětně dohledat, jak byly stanoveny ceny, odkud pochází hodnoty ve výkazu výměr, nebo jak byla udělána konkrétní rozhodnutí při výběru položek. Můžeme také přehlédnout chyby ve vstupních datech, nesprávné výpočty a jiné nepřesnosti, které následně ovlivňují finanční výsledky a předpoklady. Většina činností v tomto procesu je v dnešní době stále prováděna manuálně, což může být zdlouhavé a často neefektivní. Právě tyto problémy může částečně eliminovat programování jako pomocný nástroj. Můžeme plně nebo částečně automatizovat činnosti, detekovat chyby, provádět analýzy,

vytvářet šablony a formuláře, sestavovat databáze, propojit výpočty s výkresy a provádět mnoho dalších kroků.

Praktická část mé práce se následně zaměřuje na analýzu rozpočtů u běžných rodinných domků. Demonstruje, jak může být užitečné vytvořit jednoduchý program, který porovná položky z libovolného počtu rozpočtů a vytvoří statistiku o jejich výskytu. Položky, které se vyskytují ve 100 % rozpočtů, pak následně tvoří šablonu, která reprezentuje běžné, opakující se položky v rozpočtech rodinných domků.

1.2 Možnosti využití programování v kalkulacích a rozpočtování

V procesu kalkulace a rozpočtování jsou často využívány softwarové aplikace, ve kterých je sestavován a uchováván výsledný rozpočet. Je proto klíčové, aby odborníci v tomto oboru měli znalosti, jak tyto aplikace nejen používat, ale také vyvíjet a zdokonalovat. Mezi takové počítačové aplikace dle České komory autorizovaných inženýrů a techniků činných ve výstavbě (dále pouze ČKAIT) patří: KROS plus, ProfiKROS, euroCALC, BUILD Power, RTS Stavitel+, Contec, Aspe (2).

Samozřejmě se v běžné praxi používá spousta dalších aplikací a neustále se objevují další nové nástroje. Mezi takové nové nástroje patří například BIM platforma, která aplikuje propojení datových 3D modelů a rozpočtování v softwaru KROS a přináší výhody procesu BIM do rozpočtování. Jako zmíněné výhody prezentuje ČKAIT například: Snížení nerozpočtovaných změn neboli claimů až o 40 %, redukci potřebného času pro stanovení ceny výstavbového projektu až o 80 %, zvýšení přesnosti následně stanovené ceny a také detekování kolizí na projektu s vlivem na zkrácení času na přípravu a realizaci (2).

Aplikace Callida euroCALC 4 patří mezi inovativní nástroje v oboru rozpočtování. Tvůrci prezentují efektivní využití metodiky BIM a takzvaného 5D navrhování, za pomoci propojení 3D modelu stavby, který má v sobě uložené informace o jednotlivých prvcích. Jedná se o informace jak o rozměru, tak i o technické specifikaci, můžeme díky nim hovořit o 4D, neboli čtvrtém rozměru modelu. Právě tyto informace jsou následně využity pro vytvoření položkového rozpočtu a jeho ocenění. Položkový rozpočet, který se ideálně stane součástí modelu je dále prezentován jako pátý rozměr modelu neboli 5D (16).



Obrázek 1: euroCALC - BIM (zdroj: (19))

1.3 Výběr programovacího jazyka

Tato kapitola se zabývá výběrem vhodného programovacího jazyka pro aplikace v oblasti kalkulací a rozpočtování. Dále pak stanovuje kritéria, podle kterých bude tento výběr hodnocen. Důležité faktory pro hodnocení jsou:

- **Jednoduchost (J)** – Jak snadné je naučit se samotný jazyk a syntaxi a následně napsat funkční kód v tomto jazyce.
- **Nástroje (N)** – Jak moc jsou dostupné pomocné nástroje a aplikace, který vám zjednoduší samotné psaní kódu a vytváření procesů.
- **Komunita (K)** – Jak velká je online komunita uživatelů a kvalita a dostupnost podkladů k samostatnému vzdělání.
- **Univerzálnost (U)** – Jak všestranný je jazyk pro účely vytváření jednoduchých funkcí a cyklů.
- **Práce s daty (D)** – Jak moc je jazyk uzpůsobený k práci s importovanými daty.

Tyto parametry budou hodnoceny na stupnici 1–10, kde 1 znamená „nevyhovuje“ a 10 představuje maximální hodnocení, které lze vyjádřit jako „splňuje všechny požadavky“. Každý parametr bude také ohodnocen koeficientem důležitosti, který bude násobit udělené hodnocení.

Koeficienty důležitosti:

- Jednoduchost (J): 0.3
- Nástroje (N): 0.2
- Komunita (K): 0.2
- Univerzálnost (U): 0.2
- Práce s daty (D): 0.1

Zde je zhodnocení jazyků běžně používaných v oblasti kalkulací a rozpočtování:

- **Python** – Univerzální jazyk, který má jednoduchou syntaxi a širokou škálu knihoven pro práci s daty, analýzu a vizualizaci. Je považován za dobře pochopitelný i pro začátečníky.
- **R** – Jazyk více zaměřený na analýzu a vizualizaci dat, má širokou škálu nástrojů pro zpracování datových sad a vytváření například komplexních grafů.
- **Excel VBA** – Integrovaný programovací jazyk přímo v aplikaci Microsoft Excel, je jednoduchý a lehce dostupný, rozšiřuje možnosti funkcí a zpracování dat v Excelu.
- **SQL** – Jazyk zaměřený na práci s databázemi, je zaměřený na práci s rozsáhlými sady dat o velkém objemu.
- **Java** – Objektově orientovaný jazyk, který je složitý, ale je robustní a kód je následně spolehlivý a rychlý. Využívá se pro vývoj komplexních aplikací.

Jako běžný uživatel samozřejmě nemám dostatečné zkušenosti v oblasti programování, abych mohl objektivně posoudit tato kritéria. Proto jsem se rozhodl využít konverzaci s umělou inteligencí, která zajistila objektivní hodnocení při výběru programovacího jazyka.

Jazyk	J	N	K	U	D	Hodnocení
Python	8	9	10	9	9	8.76
R	7	8	9	8	10	8.32
Excel VBA	9	7	8	7	6	7.2
SQL	5	6	7	5	9	6.3
Java	4	7	9	8	7	6.8

Tabulka 1: Hodnocení programovacích jazyků (zdroj: Gemini.Ai. konverzace 16.03.2024 (5))

„Z hodnocení vyplývá, že **nejvhodnějším jazykem pro Vaši bakalářskou práci je Python**. Je to jazyk, který se snadno učí, má širokou škálu nástrojů a knihoven pro práci s daty a velkou online komunitu.

R je druhou nejlepší volbou, ale je méně univerzální a hůře se učí. Excel VBA je dobrou volbou, pokud chcete pracovat přímo v Excelu, ale má omezené možnosti práce s daty. SQL je vhodný pouze pro práci s databázemi. Java je robustní jazyk, ale je složitý a hůře se učí.“ (5)

Na základě této krátké analýzy jsem se rozhodl v praktické části pracovat s programovacím jazykem Python, konkrétně s Python verze 3.12.

1.4 Počáteční příprava

Před praktickou částí je potřeba si zodpovědět několik otázek týkajících se vybraného úkolu, jak je již definováno v první kapitole.

Zadání je vhodné rozdělit na menší úkoly nebo procesy, které budeme postupně vytvářet a následně propojovat. Tímto způsobem bude jednodušší identifikovat případné chyby v kódu a sledovat postup vývoje.

Při tvorbě praktické části je vhodné začít jednoduchým popisem jednotlivých logických kroků, tím získáme představu o požadavcích na chování aplikace. Tento popis může být ve formě textového dokumentu. Takto vypadá obecný příklad vytvořený umělou inteligencí:

1. „Spuštění aplikace:

- *Otevře se uživatelské rozhraní (např. okno, webová stránka).*
- *Uživatel má k dispozici různé možnosti interakce (tlačítka, menu).*

2. Možnost 1 (např. "Zpracování dat"):

- *Uživatel vybere soubor(y) k načtení.*
- *Aplikace načte data ze souboru(ů).*
- *Provede se zpracování dat (analýza, transformace).*
- *Výsledky se zobrazí uživateli nebo uloží do souboru.*

3. Možnost 2 (např. "Vizualizace"):

- *Uživatel zvolí typ vizualizace (graf, tabulka).*
- *Aplikace vytvoří vizualizaci na základě zpracovaných dat.*
- *Uživatel může vizualizaci uložit nebo sdílet.*

4. Další možnosti:

- *Mohou být zahrnuty další funkce podle konkrétního zadání (např. export dat, nastavení parametrů).“ (5)*

Díky předchozí kapitole již víme, že budeme program vytvářet v jazyce Python. Pro tvorbu kódu v tomto jazyce máme na výběr ze dvou možností:

1. První možností je použití textového editoru, který je používán pro programování, například Sublime text nebo Microsoft Visual Studio Code. Tyto textové editory jsou považovány za jednoduché, co se týká asistence při psaní kódu, protože poskytují pouze základní funkce, jako například číslování řádků, zvýraznění syntaxe... (3).
2. Druhou možností je využití integrovaného vývojového prostředí IDE (Integrated Development Environment), které kombinuje funkce textového editoru, kompilátoru a debuggeru do jednoho grafického rozhraní (3).

Po zvážení výhod jsem se rozhodl využít IDE prostředí PyCharm od české společnosti JetBrains, ke kterému mám jako student ČVUT přístup prostřednictvím studentské licence.

V praxi se běžně stává, že uživatel neumí psát kód v programovacím jazyce.

Je tedy potřeba získat alespoň základní znalosti, aby uživatel mohl porozumět kódu v daném jazyce. Nejjednodušší cestou je využít internetové stránky s výukovými programy a jednoduchými úkoly, které lze provádět přímo v internetovém rozhraní. Alternativně je možné přečíst knihy jako Python 3 Výukový kurz od autora M. Summerfielda, vydanou nakladatelstvím Computer Press v roce 2021 (ISBN 978-80-251-2737-7). Je také dobré absolvovat výukové kurzy v angličtině pro Python na stránce <https://www.w3schools.com/>, ale existuje spousta dalších možností, jako je <https://www.learnpython.org/> nebo česká stránka <https://python.cz/zacatecnici/>. Na internetu je nepřeberné množství zdrojů, takže není těžké najít ten, který bude vyhovovat vašim potřebám.

Poté, co jsme si stanovili obsah programu, který budeme vytvářet, jaké části budou potřeba, v jakém prostředí budeme program psát a jakým způsobem, můžeme v podstatě začít s jeho tvorbou.

1.5 Dostupné programovací nástroje a metodiky

Vzhledem k tomu, že se jedná o využití programování osobou, která nikdy předtím seriózně neprogramovala, je dobré představit a ukázat nástroje a metody, které nám pomohou s vytvořením funkčního kódu.

Prvním pomocníkem bude vývojové prostředí, v mém případě PyCharm, které nabízí mnoho užitečných funkcí, jako je nápověda pro požadované parametry ve funkcích.

```
18  async def main():
19      var = await asyncio.gather(
20          factorial("A", 2),
21          fac name, number ,
22          factorial(3, 4),
23          factorial()
24      )
25      print(var)
```

Obrázek 2: Parametr ve funkci (zdroj: (4))

Další pomocnou funkcí je vyskakovací okénko při najetí na již napsanou funkci, které vám vysvětlí její podrobné fungování.

```
The for statement
The for statement is used to iterate over the elements of a sequence (such
as a string, tuple or list) or other iterable object:
for_stmt ::= "for" target_list "in" expression_list ":" suite
           ["else" ":" suite]

The expression list is evaluated once; it should yield an iterable object. An
iterator is created for the result of the expression_list. The suite is then
executed once for each item provided by the iterator, in the order returned
by the iterator. Each item in turn is assigned to the target list using the
standard rules for assignments (see Assignment statements), and then the
suite is executed. When the items are exhausted (which is immediately when
the sequence is empty or an iterator raises a StopIteration exception), the
suite in the else clause, if present, is executed, and the loop terminates.
A break statement executed in the first suite terminates the loop without
executing the else clause's suite. A continue statement executed in the first
suite skips the rest of the suite and continues with the next item, or with the
else clause if there is no next item.

The for-loop makes assignments to the variables(s) in the target list. This
overwrites all previous assignments to those variables including those made
in the suite of the for-loop:

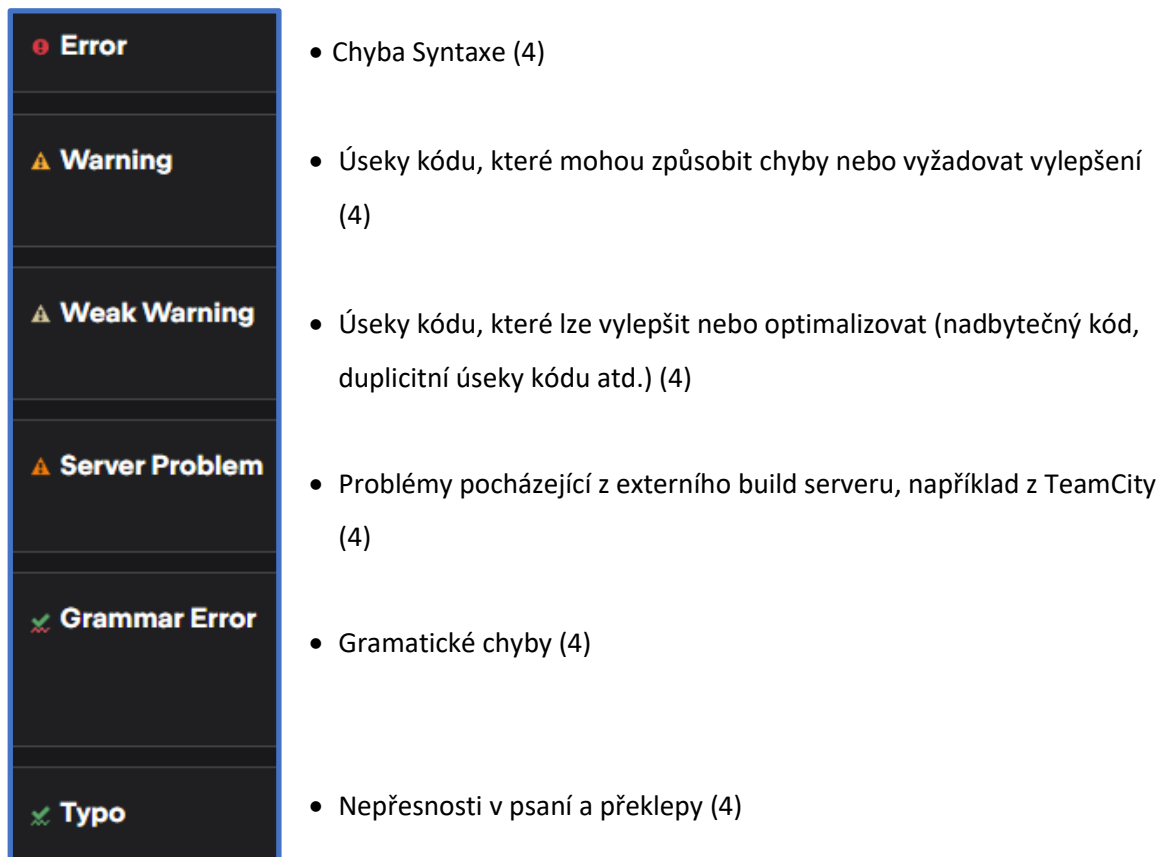
for i in range(10):
    print(i)
    i = 5           # this will not affect the for-loop
                  # because i will be overwritten with the next
                  # index in the range
```

Obrázek 3: Vysvětlení funkcí – okno (zdroj: Aplikace PyCharm)

Poslední funkcí, kterou zmíním a je součástí prostředí PyCharm, je jeho integrovaná analýza kódu. Tato analýza probíhá na pozadí, nepřetržitě kontroluje kód a vyhledává chyby a problémy. Výsledky jsou kategorizovány a zobrazeny, přičemž PyCharm rovnou navrhuje opravy nebo změny, které mohou předejít možnému zaseknutí nebo chybovým hláškám.



Obrázek 4: Upozornění na chyby (zdroj: (4))



Obrázek 5: Vysvětlení upozornění (zdroj: (4))

Mezi klíčové nástroje, které dnes usnadňují programování, patří knihovny.

Představte si programování jako stavbu domu. Namísto toho, abyste museli každou cihlu vyrobit a položit sami, máte k dispozici sady nástrojů plné hotových komponentů – oken, dveří, stěn atd. Knihovny fungují v programování na podobném principu. Jsou to sbírky předkompilovaného kódu, které zjednodušují samotný vývoj aplikace.

Místo psaní kódu od začátku pro každou opakovatelnou úlohu, jako je čtení dat z tabulky, zobrazování informací na obrazovce nebo práce s daty, můžete jednoduše načíst potřebnou knihovnu do projektu a využít již hotové funkce. To šetří čas, námahu a eliminuje nutnost opakovaně řešit stejný problém.

Knihovny tedy slouží jako základní stavební kámen pro programování, který usnadní i složitější úkoly a dělají programování jednodušší a dostupné i pro začátečníky.

Podrobnější a konkrétní využití knihoven naleznete také v praktické části práce.

Posledním nástrojem, kterým se v této kapitole budu zabývat, je webová stránka Stackoverflow. Při výběru programovacího jazyka jsem zmínil jako jeden z hodnotících parametrů komunitu, a právě na komunitě je celá stránka Stackoverflow postavena. Stackoverflow je rozsáhlá komunita vývojářů a programátorů, která slouží jako efektivní místo pro řešení problémů, sdílení znalostí a učení se od ostatních.

Stackoverflow funguje jako obří fórum zaměřené na dotazování, s velmi aktivní komunitou. Díky tomu, že jsme si vybrali programovací jazyk s velkou a silnou komunitou a zároveň patříme mezi začínající programátory, kteří se potýkají s jednoduššími chybami, je velmi pravděpodobné, že většina problémů, se kterými se setkáme, již byla někdy řešena. Řešení se dá dohledat v jednom z více než 24 miliónů dotazů, které byly již na stránce Stackoverflow zodpovězeny.

Dalším nástrojem je využití umělé inteligence, který představím v další kapitole.

1.6 Využití umělé inteligence

Umělá inteligence je v dnešní době důležitým nástrojem, který nelze opomíjet. Tato kapitola popisuje, jaký veřejně dostupný model umělé inteligence je využitý, proč byl zrovna tento model vybrán a jaký má potenciál v programování.

Nejdříve si vysvětlíme, co je to vlastně umělá inteligence (AI, z anglického Artificial Intelligence). Umělá inteligence je počítačový systém, který pomocí matematiky a logiky napodobuje způsob, jakým se lidé učí z nových informací a následně se na základě těchto znalostí rozhodují. Umělá inteligence vzniká prostřednictvím strojového učení, což je proces, při kterém se model trénuje na podmnožinách dat, aby byl schopen vytvářet a analyzovat úplné datové sady. Jakmile model dosáhne určité přesnosti a rychlosti, je označován za omezenou umělou inteligenci.

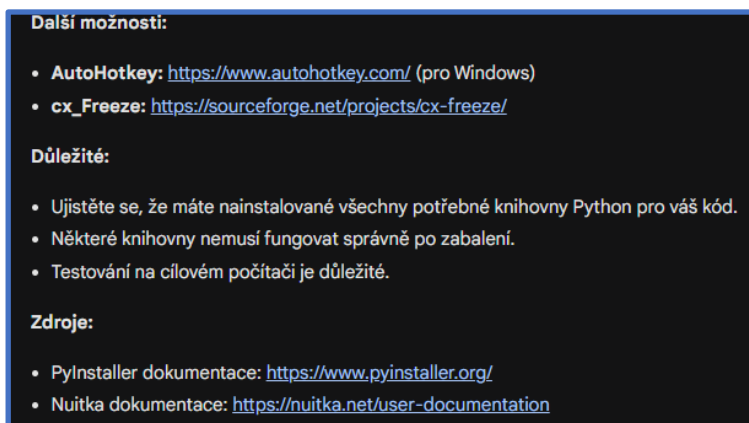
Strojové učení je základní proces, který se v pokročilejších modelech rozšiřuje na hluboké učení spíše známo anglicky jako Deep learning. Deep learning je pokročilý typ strojového učení, který využívá neurální sítě inspirované strukturou lidského mozku. Tyto sítě mají neurální uzly, kde každá otázka vede k sadě souvisejících otázek. Pro tento proces jsou potřeba rozsáhlé datové sady obsahující miliony

jednotlivých bodů. Výsledkem tréninku s těmito datovými sadami je síť schopná řešit i nejednoznačné úkoly, například generování fotorealistických obrázků s fiktivním obsahem.

Prvním krokem byla krátká rešerše dostupných modelů umělé inteligence a jejich vhodnosti pro asistenci při tvorbě kódu a získávání potřebných informací pro vývoj aplikace. Mezi zvažovanými chatboty byli nejznámější OpenAI ChatGPT, Microsoft Copilot, Anthropic's Claude a Google Gemini (dříve bard).

Existují i specifické modely umělé inteligence zaměřené na tvorbu kódu například GitHub Copilot, ale pro náš případ preferujeme univerzálnější a uživatelsky přívětivější řešení. Klíčovým kritériem pro výběr byla schopnost umělé inteligence přistupovat k informacím z internetu, aby bylo možné získávat nejnovější verze knihoven a řešení aktuálních problémů.

Testování jednoduchých otázek ve zmíněných modelech neukazuje výrazné rozdíly mezi jednotlivými modely. Rozdílem ale může být schopnost modelu dodat k odpovědi i zdroje a dodatečné informace. Toto je například užitečnou funkcí modelu Google Gemini, který uvádí další zdroje a odkazy pro další vzdělávání v dotazované oblasti.



Obrázek 6: Příklad odkazování (zdroj: Gemini.Ai. konverzace 12.05.2024 (5))

Na základě otestování jsem se pro praktickou část rozhodl využít model Google Gemini, právě kvůli jeho schopnosti poskytovat odkazy na další zdroje, možnosti využívat výkonnější placenou verzi zdarma po dobu dvou měsíců jako zkušební období a díky transparentnímu reportu, ve kterém Google porovnává svůj model v sérii nezávislých testů. Placená verze modulu Gemini Ultra 1.0 je srovnávána s dalšími volně přístupnými modely, jako jsou GPT-4, PaLM 2-L, Claude 2 a další. V tomto reportu se specificky řeší schopnost vytváření a práce s Python kódem ve 2 různých benchmarcích:

- **HumanEval**, který se skládá ze 164 ručně napsaných programovacích úloh o různých obtížnostech a v různých oblastech, například matematické operace, práce se seznamy,

manipulace řetězců a dalších. Obecně se jedná o objektivní evaluaci a nástroj pro zlepšení schopností umělé inteligence v oblasti programování. (6)

- **Natural2Code**, který je založen na stejném principu jako HumanEval, akorát se jedná o úlohy z interních data setů vývojářů Gemini. Jedná se tedy o další nástroj pro zdokonalení schopností práce s Python kódem (8).

	Gemini Ultra	Gemini Pro	GPT-4	GPT-3.5	PaLM 2-L	Claude 2	Inflection-2	Grok 1	LLAMA-2
MMLU Multiple-choice questions in 57 subjects (professional & academic) (Hendrycks et al., 2021a)	90.04% CoT@32*	79.13% CoT@8*	87.29% CoT@32 (via API**)	70% 5-shot	78.4% 5-shot	78.5% 5-shot CoT	79.6% 5-shot	73.0% 5-shot	68.0%***
GSM8K Grade-school math (Cobbe et al., 2021)	94.4% Maj1@32	86.5% Maj1@32	92.0% SFT & 5-shot CoT	57.1% 5-shot	80.0% 5-shot	88.0% 0-shot	81.4% 8-shot	62.9% 8-shot	56.8% 5-shot
MATH Math problems across 5 difficulty levels & 7 subdisciplines (Hendrycks et al., 2021b)	53.2% 4-shot	32.6% 4-shot	52.9% 4-shot (via API**)	34.1% 4-shot (via API**)	34.4% 4-shot	—	34.8% 4-shot	23.9% 4-shot	13.5% 4-shot
BIG-Bench-Hard Subset of hard BIG-bench tasks written as CoT problems (Srivastava et al., 2022)	83.6% 3-shot	75.0% 3-shot	83.1% 3-shot (via API**)	66.6% 3-shot (via API**)	77.7% 3-shot	—	—	—	51.2% 3-shot
HumanEval Python coding tasks (Chen et al., 2021)	74.4% 0-shot (PT****)	67.7% 0-shot (PT****)	67.0% 0-shot (reported)	48.1% 0-shot	—	70.0% 0-shot	44.5% 0-shot	63.2% 0-shot	29.9% 0-shot
Natural2Code Python code generation. (New held-out set with no leakage on web)	74.9% 0-shot	69.6% 0-shot	73.9% 0-shot (via API**)	62.3% 0-shot (via API**)	—	—	—	—	—
DROP Reading comprehension & arithmetic. (metric: F1-score) (Dua et al., 2019)	82.4 Variable shots	74.1 Variable shots	80.9 3-shot (reported)	64.1 3-shot	82.0 Variable shots	—	—	—	—
HellaSwag (validation set) Common-sense multiple choice questions (Zellers et al., 2019)	87.8% 10-shot	84.7% 10-shot	95.3% 10-shot (reported)	85.5% 10-shot	86.8% 10-shot	—	89.0% 10-shot	—	80.0%***
WMT23 Machine translation (metric: BLEURT) (Tom et al., 2023)	74.4 1-shot (PT****)	71.7 1-shot	73.8 1-shot (via API**)	—	72.7 1-shot	—	—	—	—

Obrázek 7: Benchmark výsledky (zdroj: (7))

Jak tedy můžeme využívat umělou inteligenci?

V praktické části využijeme umělou inteligenci jako asistenta při psaní kódu, při výběru vhodných knihoven, při vyskytnutí se chybových hlášek, pro které neznáme řešení, a pro optimalizaci složitějších cyklů. Kód je pořád psaný a komentovaný programátorem, ale činnosti, které by během manuálního řešení trvaly objektivně dlouhou dobu jsou řešeny prostřednictvím dotazování. Specifické ukázky využití budou prezentovány v Praktické části.

Celou dobu zde hovoříme o umělé inteligenci v podobě takzvaného LLM (Large Language Model – velký jazykový model). Jedná se o typ umělé inteligence, který je navržen pro práci s textem a jeho zpracováním a je využíván právě ve zmiňovaných aplikacích typu ChatGPT, Google Gemini atd (9).

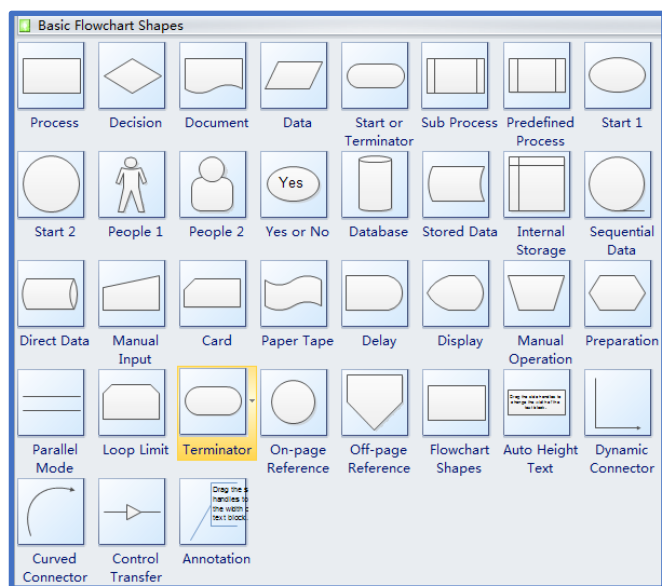
Textové dotazování LLM modelu ale není jediných způsob, jak využívat umělou inteligenci. Tyto zmiňované modely je možné implementovat do aplikací v různých podobách. Ať se jedná o asistenta, kontrolní bod, nebo automatizaci výstupu, kdy je možné integrovat modely v takzvaném API módu (Application Programming Interface – rozhraní pro programování aplikací) a to přímo za pomoci jazyku Python (10).

Pro představu právě v kalkulacích a rozpočtování by to mohlo znamenat vznik aplikace, která bude s využitím integrovaného LLM vytvářet rozpočet automaticky za pomoci informací, které získá z BIM modelu a přístupu do cenových soustav.

1.7 Logická mapa procesu

Logická mapa, vývojový diagram nebo také flowchart je jedním způsobem, jak vizualizovat procesy v aplikaci. Pomáhá vývojáři, ale především ostatním porozumět jednotlivým krokům a jejich návaznosti. Jedná se o další z důležitých kroků dokumentace vývojového cyklu. Pro vytvoření takového diagramu existují jak webové, tak přímo dedikované softwarové aplikace. Pro náš případ využijeme populární webovou aplikaci <https://app.diagrams.net/>, která má jednoduché rozhraní.

Pro zakreslování logické mapy jsou určité pravidla, které by měl uživatel dodržovat. Jedním z těchto pravidlem jsou významy jednotlivých grafických obrazců, nebo vyobrazení podmínky při rozhodování v bodě podmínek (15).



Obrázek 8: Obrazce flowchart (zdroj: (15))

Ukázkou je příloha č.1, kterou je logická mapa ve své první verzi.

2. Praktická část

2.1 Příprava vývoje

Prvním úkolem v této práci je ukázat, že je možné začít programovat a vytvořit funkční skript v oblasti kalkulací a rozpočtování. Tato demonstrace proběhne na úloze definované již v první kapitole.

Jedná se o vytvoření skriptu, který provede analýzu různých rozpočtů a následně vytvoří tzv. „Common body“ neboli soubor, který obsahuje často se opakující položky mezi analyzovanými rozpočty. Tento script by měl usnadnit vytváření dalších rozpočtů pro jednoduché rodinné domky. Výsledný soubor lze importovat do rozpočtářské aplikace. To znamená, že část položek bude již předem v rozpočtu zahrnuto, a bude tak možné pouze doplnit výměry a případně specifikovat materiál.

Vstupními daty jsou rozpočty rodinných domků. Konkrétně se jedná o 9 anonymizovaných rozpočtů vytvořených v softwarové aplikaci KROS 4, která je v České republice nejrozšířenější, zejména pro rozpočtování rodinných domků. Tyto rozpočty byly exportovány z aplikace KROS ve standardním formátu do Excelu, kde jsou dále zpracovávány jako vstupní data. Pro účely této úlohy jsou rozpočty označeny čísly 1-9 pro snadnou identifikaci. Rozpočty těchto rodinných domků byly vytvořeny v posledních pěti letech a jejich položky byly aktualizovány na verzi cenové soustavy platné v době tvorby této práce.

Skript bude vytvořen v programovacím jazyce Python 3.12 v integrovaném vývojovém prostředí PyCharm, s využitím nástrojů definovaných v teoretické části práce. Mezi tyto nástroje patří zejména umělá inteligence Google Gemini, jejíž využití je detailně rozebráno v kapitole týkající se aplikace umělé inteligence.

Druhým úkolem této práce bude využít poznatky získané během tvorby skriptu k vytvoření manuálu, který obecně popíše proces vývoje a poskytne jednoduché instrukce, jak tento proces opakovaně použít. Cílem tohoto opakování bude seznámení se základy programování a vytvoření výstupu, který by mohl potenciálně zvýšit efektivitu práce, zejména v oblasti kalkulací a rozpočtování.

Manuál by měl zahrnovat veškeré klíčové informace definované v teoretické části. Kromě toho by měl poskytovat odkazy na relevantní zdroje informací a doporučení od autora, aby usnadnil rozhodování a postup čtenářů.

2.2 Vývoj aplikace

Na začátku vývoje bylo nezbytné porozumět zadání. Bylo nutné zjistit, v jakém formátu jsou poskytnuta vstupní data – zda se jedná o formát univerzální, nebo zda je třeba implementovat dodatečné kroky pro identifikaci dat v podkladech. V tomto případě se vždy bude jednat o rozpočtovou strukturu dat aplikace KROS. Data v tomto formátu jsou exportována do excelu a následně zase ve stejné struktuře z excelu importována zpět. Import dat se provádí přímo v aplikaci KROS pomocí tzv. Inteligentních importů, které prezentuje na svých stránkách vývojář aplikace URS (15).

Sloupec	Řádky	Údaje objektu	Náhled						
CP	Kód položky	Popis	MJ	Množství	Jednotková cena	Cenová soustava			
Víkaz víměr	187	W	f02*0,5		10,482				
Víkaz víměr	188	W	f03*0,5		2,928				
Víkaz víměr	189	W	f04*0,5		22,560				
Víkaz víměr	190	W	Součet		38,439				
Konstrukce	191	B K	162751139	Příplatek k vodorovnému přemístění výkopku/sypání z horniny třídy těžitelnosti II, skupiny 4 a 5 ZKd 1000 m přes 10000 m	m3	153,754	22,80	3 505,59	CS URS 2021 01
Víkaz víměr	192	W	*50% objemu zeminy k odvozu na skládku - 14km						
Víkaz víměr	193	W	f01*0,5 * 4		9,876				
Víkaz víměr	194	W	f02*0,5 * 4		41,926				
Víkaz víměr	195	W	f03*0,5 * 4		11,712				
Víkaz víměr	196	W	f04*0,5 * 4		90,240				
Víkaz víměr	197	W	Součet		153,754				
Konstrukce	198	9 K	171201231	Poplatek za uložení zeminy a kamení na recyklační skládce (skládkové) kód odpadu 17 05 04	t	38,439	253,00	9 725,07	CS URS 2021 01
Víkaz víměr	199	W	f01*0,5		2,469				
Víkaz víměr	200	W	f02*0,5		10,482				
Víkaz víměr	201	W	f03*0,5		2,928				
Víkaz víměr	202	W	f04*0,5		22,560				
Víkaz víměr	203	W	Součet		38,439				
Konstrukce	204	10 K	174101102	Zásep v uzavřených prostorech sypánínou se ztuhnutím	m3	19,220	364,00	6 996,08	CS URS 2021 01
Víkaz víměr	205	W	f01*0,25		1,235				
Víkaz víměr	206	W	f02*0,25		5,241				
Víkaz víměr	207	W	f03*0,25		1,464				
Víkaz víměr	208	W	f04*0,25		11,280				
Víkaz víměr	209	W	Součet		19,220				
Konstrukce	210	11 K	181351104	Rozprostření omice II vrstvy do 250 mm pi do 500 m2 v rovině nebo ve svahu do 1,5 strojné	m2	494,400	66,90	33 075,36	CS URS 2021 01
Konstrukce	211	12 K	181951113	Úprava pláně v hornině třídy těžitelnosti II, skupiny 4 a 5 bez ztuhnutí	m2	19,220	15,40	295,96	CS URS 2021 01
Víkaz víměr	212	W	f01*0,25		1,235				
Víkaz víměr	213	W	f02*0,25		5,241				
Víkaz víměr	214	W	f03*0,25		1,464				
Víkaz víměr	215	W	f04*0,25		11,280				
Víkaz víměr	216	W	Součet		19,220				
Díl 2	217	p	2	Zakládání				279 070,80	
Konstrukce	218	13 K	213141111	Zřízení vrstvy z geotextilie v rovině nebo ve sklonu do 1,5 š do 3 m	m2	20,700	19,00	393,30	CS URS 2021 01
Víkaz víměr	219	W	"okapový chodník" 20,7		20,700				
Material	220	14 M	69311143	geotextilie netkaná separační, ochranná, filtrační, drenážní PP 210g/m2	m2	23,805	16,90	402,30	CS URS 2021 01
Víkaz víměr	221	W	20,7*1,15		23,805				
Konstrukce	222	15 K	271532213	Podšyp pod základové konstrukce se ztuhnutím z hrubého kamenniva frakce 8 až 16 mm	m3	29,189	1 470,00	42 907,83	CS URS 2021 01
Víkaz víměr	223	W	152,1*0,15*1,035		23,814				
Víkaz víměr	224	W	0,7*0,7*0,15*1,035		0,076				
Víkaz víměr	225	W	0,7*0,7*0,15*4*1,035		0,304				
Víkaz víměr	226	W	2,1*0,2*0,15*3*1,035		0,196				
Víkaz víměr	227	W	Základové konstrukce		24,190				
Víkaz víměr	228	W	20,7*0,1*1,035		2,142				
Víkaz víměr	229	W	Okapový chodník		2,142				

Obrázek 9: Inteligentní importy (zdroj: (14))

Známe tedy, jaká data zpracováváme a v jakém formátu. Vybrali jsme programovací jazyk a rozhraní pro tvorbu kódu a nyní je potřeba popsat a rozdělit proces na jednotlivé kroky, jak je zmíněno v kapitole Počáteční příprava. V našem případě se jedná o jednoduchý textový dokument, který vypadá takto:

Spuštění .exe souboru

* otevře se okno s tlačítky (možnostmi)

* Tlačítko Nahrát excelové soubory:

* Spustí kód který

* Načte excelové soubory

* Najde sloupec se specifickými informacemi

* Vytvoří databázi

* Lokální (SQLite) nebo jiné způsoby spojené s knihovnamí

* Struktura závislá na typu a struktuře dat v excelových souborech

* tlačítko "Statistika":

* Spustí kód který

* Kam uložit -> cesta soubory

* nový excel soubor

* vytvoří Grafické znázornění dat (histogramy, grafy)

* zapíše tabulku položek s informacemi o výskytu

(Seřazeno dle klesajícího výskytu)

* tlačítko "Šablona":

* Spustí kód který

* Kam uložit -> cesta soubory

* Otevře již vloženou šablonu

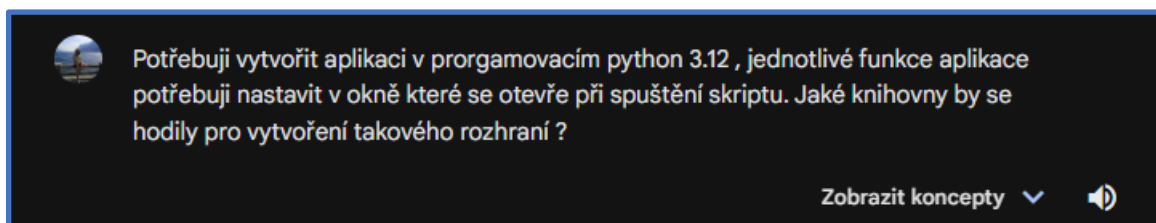
* Zapíše vybraných položek dle podmínky (možnost flexibilní podmínky).

poznámky:

Databáze nejspíše nebude třeba, moc složitá datová struktura pomocí knihoven pro čtení excelu

Pro jednotlivé kroky je následně potřeba definovat postup, cyklení a knihovny, které budeme využívat.

Interakce uživatele s programem by se měla odehrávat v jednoduchém okně, kde budou tlačítka na spuštění jednotlivých implementovaných funkcí. První úkol je tedy najít vhodnou knihovnu pro požadovanou funkcionalitu. Právě výběr vhodných knihoven je jednou z činností, kterou jsem řešil převážně za pomoci konverzace s umělou inteligencí. Dotazování probíhá pomocí jednoduchých otázek, které můžou vypadat například takto:



Obrázek 10: Dotazování knihoven – Vstup (zdroj: Gemini.Ai. konverzace 25.03.2024 (5))

Umělá inteligence následně doporučí knihovny, které by mohly být vhodné a napíše i odůvodnění a připojí zdroje, kde jsou tyto knihovny zdokumentovány. Zároveň předvídá další dotazy a rovnou prezentuje, jak načíst takovou knihovnu do našeho kódu, a dokonce se snaží radit, jak přesně postupovat. Příklad takové odpovědi:

Pro vytvoření grafického uživatelského rozhraní (GUI) ve vašem Python 3.12 skriptu máte několik možností, každá s vlastními výhodami:

Doporučené knihovny:

- **Tkinter:**
 - Je to **standardní** knihovna Pythonu pro GUI, takže ji nemusíte instalovat.
 - Je **jednoduchá** pro začátečníky a vhodná pro menší aplikace.
 - Má **omezenější** možnosti přizpůsobení vzhledu než jiné knihovny.
 - Dokumentace: <https://docs.python.org/3/library/tkinter.html>
- **PyQt5:**
 - Nabízí **široké** možnosti a profesionální vzhled.
 - Je **výkonná** a vhodná i pro komplexní aplikace.
 - Vyžaduje **instalaci** a může mít strmější křivku učení.
 - Dokumentace: <https://pypi.org/project/PyQt5/>
- **Kivy:**
 - Je **moderní** a zaměřuje se na dotykové rozhraní (mobilní zařízení).
 - Umožňuje **snadno** vytvářet atraktivní a responzivní aplikace.
 - Může být **náročnější** na výkon než jiné knihovny.
 - Dokumentace: <https://kivy.org/doc/stable/>

Další možnosti:

- **wxPython:**
 - Podobný jako PyQt5, ale s trochu jinou syntaxí.
 - Dokumentace: <https://www.wxpython.org/>
- **Libavg (C++ knihovna s Python rozhraním):**
 - Zaměřuje se na vysoký výkon a hardwarovou akceleraci.
 - Vhodná pro náročné grafické aplikace.
 - Dokumentace: <https://www.libavg.de/site/>

Kterou knihovnu zvolit?

Volba závisí na vašich konkrétních potřebách a preferencích:

- **Jednoduchost a rychlý start:** Tkinter
- **Profesionální vzhled a velké možnosti:** PyQt5
- **Moderní design a dotykové rozhraní:** Kivy
- **Vysoký výkon a náročná grafika:** Libavg

Příklad s Tkinter (základní okno):

```
Python
import tkinter as tk

# Vytvoření hlavního okna
okno = tk.Tk()
okno.title("Moje aplikace")

# Spuštění aplikace
okno.mainloop()
```

Při použití kódu **buďte obezřetní**.

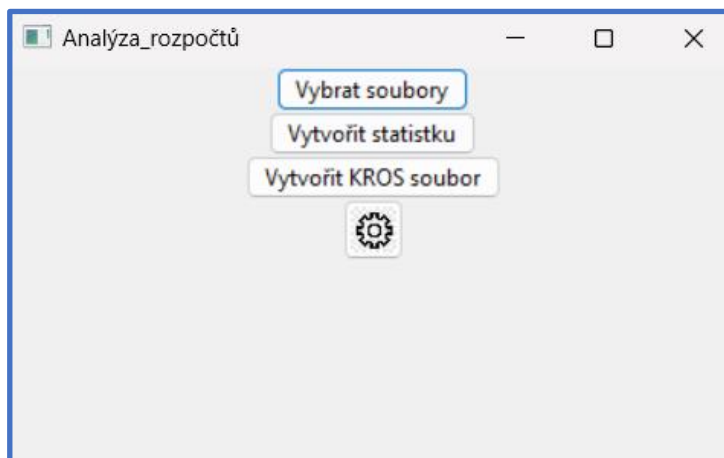
Tento kód vytvoří jednoduché okno s názvem "Moje aplikace". Můžete do něj přidávat další prvky jako tlačítka, textová pole atd.

Rada: Než se pustíte do vývoje, doporučuji prozkoumat dokumentaci a tutoriály vybrané knihovny, abyste se seznámili s jejími možnostmi a postupy.

Obrázek 11: Dotazování knihoven – Výstup (zdroj: Gemini.Ai. konverzace 25.03.2024 (5))

V mém případě jsem se na základě doporučení rozhodl pro knihovnu wxPython. Důvodem pro toto rozhodnutí byly například přehledné stránky s dokumentací, včetně videonávodů, jak knihovnu využívat. Tímto způsobem tedy bude řešen výběr knihoven v průběhu vývoje.

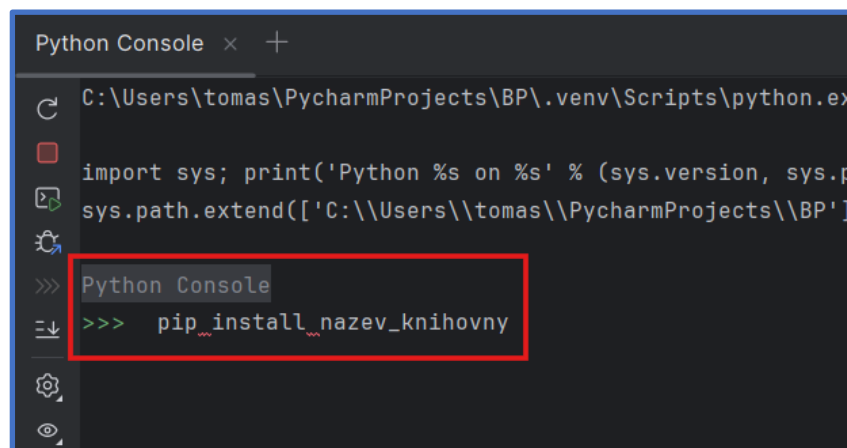
Pomocí vybrané knihovny jsem vytvořil základní kámen aplikace, komunikační okno s tlačítky, pod kterými se skrývají odkazy na funkce. Jak je možné vidět na obrázku, jsou zde tlačítka odpovídající původnímu návrhu v textovém dokumentu. Jediná funkce, která je zde oproti původnímu návrhu navíc, je tlačítko pro nastavení. Možnost nastavení jsem přidal až po zjištěních vzniklých z analýzy položek rozpočtů.



Obrázek 12: Hlavní prostředí aplikace (zdroj: Vlastní aplikace)

Po vytvoření základního komunikačního okna programu s tlačítky bylo dalším krokem vyřešení práce s daty, tedy jakým způsobem data načítat, zpracovávat a zapisovat. Potřeboval jsem najít vhodnou knihovnu na práci s daty z excelu. Pomocí konverzace s umělou inteligencí a následně rešerši jsem začal psát kód s využitím knihovny pandas. Knihovna pandas je schopná jednoduché manipulace s daty a následně pokročilé vizualizace a analýzy. V průběhu zpracování úlohy jsem se ale dostal do bodu, kdy se ukázalo, že sama knihovna pandas není ideálním řešením a začal jsem proto používat další knihovnu. Přímé důvody popisují v kapitole o chybách, kterých se vyvarovat. Na finální řešení jsem dodatečně využil i knihovnu openpyxl, která je více optimalizovaná na zpětné zapisování dat do excel souborů.

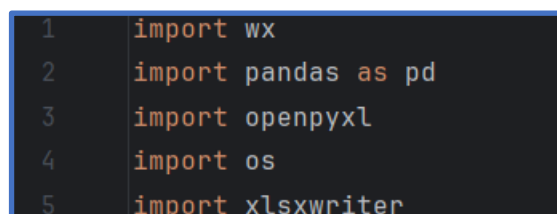
Každou knihovnu je třeba prvně nainstalovat. Nejsnadnější způsob je využití python konzole, která je lehce použitelná přímo ve vývojovém prostředí s využitím tzv. pip, což je nástroj pro instalaci knihoven. Stačí jednoduše použít tento formát:



```
Python Console x +
C:\Users\tomas\PycharmProjects\BP\.venv\Scripts\python.exe
import sys; print('Python %s on %s' % (sys.version, sys.p
sys.path.extend(['C:\\Users\\tomas\\PycharmProjects\\BP'])
Python Console
>>> pip_install_nazev_knihovny
```

Obrázek 13: Instalace knihoven (zdroj: Aplikace PyCharm)

Každou takto nainstalovanou knihovnu stačí následně importovat do kódu a její funkce budou automaticky dostupné.



```
1 import wx
2 import pandas as pd
3 import openpyxl
4 import os
5 import xlswriter
```

Obrázek 14: Import knihoven do kódu (zdroj: Aplikace PyCharm)

Samotný kód vznikl po jednoduchých malých krocích a následně jsem k němu přidával další potřebné funkce.

Prvně bylo třeba načíst data z excel souborů, než se data začnou dále zpracovávat. Data bylo následně třeba uložit, filtrovat a z filtrovaných dat vytvořit analýzu. Analýza ale zahrnuje porovnání položek z více jednotlivých souborů neboli rozpočtů. Bylo tedy třeba přidat krok, který zaznamená u každé položky informaci, z jakého souboru pochází a celý proces opakovat (zacyklit) pro načtení dat ze všech souborů.

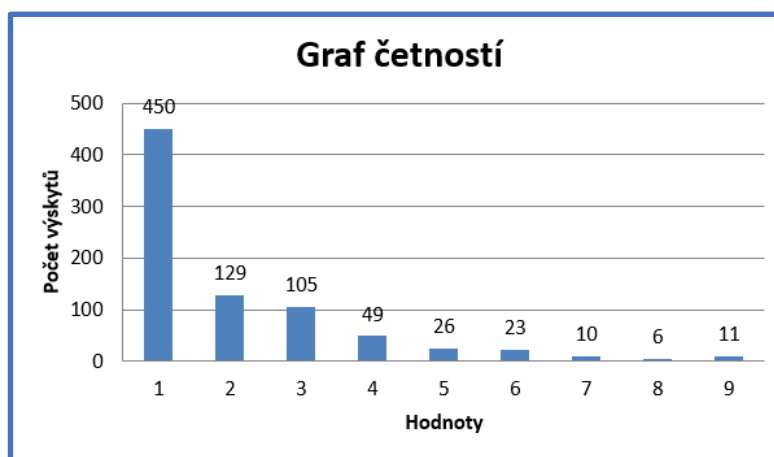
Tímto postupem vznikaly jednotlivě 3 bloky neboli funkce pro 3 tlačítka v programu.

První blok (funkci) „Vybrat soubory“ jsem již částečně popsal výše. Funkce vybere soubory a načte z nich relevantní informace z rozpočtů. Relevantní soubory uživatel označí ve výběrovém okně. Data ze souborů si program uloží do interní paměti (data frame) pro další kroky, tedy zpracování informací pro požadovanou analýzu a přípravu formátu pro další bloky, které již fungují nezávisle na sobě.

Druhý blok neboli druhé tlačítko „Stáhnout statistiku“ obsahuje odkaz na funkci, která zpracuje jednoduchou statistiku. Ta prezentuje rozložení četností jednotlivých položek v rozpočtech a pomůže nám s výběrem položek relevantních pro finální použití výstupu z programu. Výstupem statistiky je excel soubor s grafem právě zmiňovaného rozložení četností a následný seznam všech unikátních položek rozpočtů s informacemi o četnosti výskytu v jednotlivých rozpočtech.

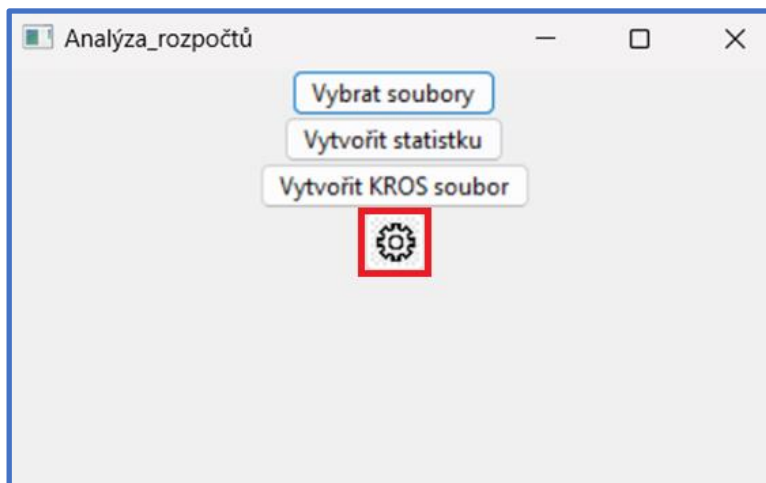
Třetí blok, který reprezentuje tlačítko „Stáhnout KROS soubor“ spustí funkci, která zapisuje do připraveného excel souboru ve formátu prázdné šablony KROS rozpočtu. Tím vytvoří soubor s předvyplněnými položkami rozpočtu („Common body“).

Původním předpokladem bylo, že na základě všech analyzovaných rozpočtů identifikujeme větší množství položek, které se opakují ve všech vstupních souborech. Vytvořená statistika na reálných rozpočtech ovšem ukázala, že četnost opakování položek je nižší, než byl předpoklad. Některé položky se opakují, ale ne ve všech rozpočtech.



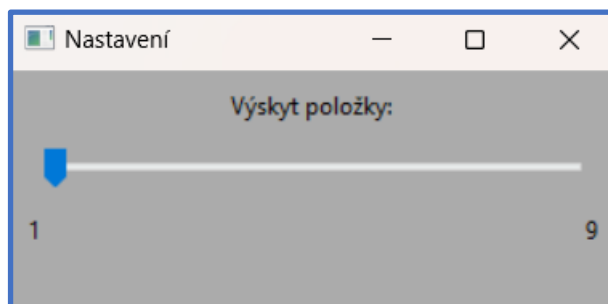
Obrázek 15: Graf analýzy četnosti (zdroj: Výstup Statistika (Příloha č.3))

Rozhodl jsem se tedy implementovat proměnnou funkci, která umožní nastavit, jak častý musí být výskyt jednotlivé položky, aby se následně zapsala do šablony, která se dá zpětně nahrát do rozpočtářské aplikace KROS. Právě na tomto nastavení budu prezentovat detailněji části kódu a vysvětlím využití postupy. Samotné nastavení parametru se provádí v bloku s tlačítkem s ozubeným kolečkem, které reprezentuje nastavení:



Obrázek 16: Tlačítko nastavení (zdroj: Vlastní aplikace)

Po stisknutí tlačítka pro nastavení se otevře okno s posuvnou lištou, na které může uživatel nastavit zmiňovanou proměnnou pro výběr položek do výstupního souboru. Maximální počet výskytu je omezen na počet vstupních souborů.



Obrázek 17: Okno s nastavením (zdroj: Vlastní aplikace)

K vytvoření okna pro nastavení je použit tento kód, který definuje rozměry, popisky a pozice jednotlivých prvků v okně. Zde je ukázka právě pro tyto parametry:

```
263     def on_but4(self, event): # Spuštění při zmáčknutí tlačítka pro nastavení
264         # Vytvoření nového okna pro nastavení
265         nastaveni_okno = wx.Frame(*args: self, title="Nastavení")
266         nastaveni_okno.SetSize((320, 160))
267
268         # Vytvoření posuvníku (slideru)
269         self.slider = wx.Slider(*args: nastaveni_okno, value=1, minValue=1, maxValue=pocet_souboru, style=wx.SL_HORIZONTAL)
270         self.slider.Bind(wx.EVT_SLIDER, self.on_slider_change)
271
272         # Popisky osy
273         label_min = wx.StaticText(*args: nastaveni_okno, label="1")
274         label_max = wx.StaticText(*args: nastaveni_okno, label=str(pocet_souboru))
275
276         # Vysvětlující text
277         label_vysvetleni = wx.StaticText(*args: nastaveni_okno, label="Výskyt položky:")
278
279         # Rozvržení prvků pomocí sizeru
280         sizer = wx.BoxSizer(wx.VERTICAL)
281         sizer.Add(*args: label_vysvetleni, 0, wx.ALIGN_CENTER | wx.TOP, 10)
282         sizer.Add(*args: self.slider, 0, wx.EXPAND | wx.ALL, 10)
283         sizer_horizontal = wx.BoxSizer(wx.HORIZONTAL)
284         sizer_horizontal.Add(*args: label_min, 0, wx.ALIGN_CENTER_VERTICAL | wx.LEFT, border=10)
285         sizer_horizontal.AddStretchSpacer(1)
286         sizer_horizontal.Add(*args: label_max, 0, wx.ALIGN_CENTER_VERTICAL | wx.RIGHT, border=10)
287         sizer.Add(*args: sizer_horizontal, 0, wx.EXPAND)
288         nastaveni_okno.SetSizer(sizer)
289
290         # Zobrazení okna nastavení
291         nastaveni_okno.Show()
```

Obrázek 18: Kód pro okno s nastavením (zdroj: Aplikace PyCharm)

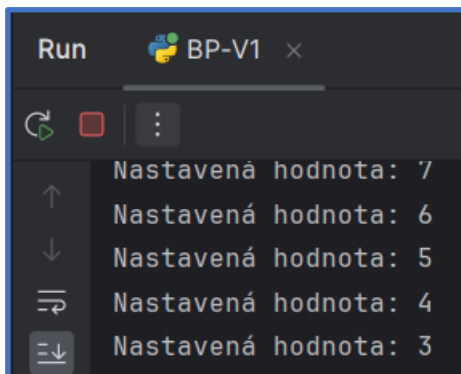
Jednotlivé části kódu jsou komentované, text psaný za symbolem # popisuje danou část kódu, ale nezasahuje do jeho fungování. Pouze slouží jako informace pro vývojáře, nebo kohokoliv, kdo by chtěl editovat, nebo pochopit fungování kódu.

Dalším krokem bylo přiřadit proměnnou k posouvání již zmiňovaného slideru. Právě tuto proměnnou, kterou jsem v kódu pojmenoval „hodnota“, definuje tato část kódu:

```
292
293     def on_slider_change(self, event):
294         global hodnota # Deklarace globální proměnné hodnota
295         # Získání aktuální hodnoty posuvníku
296         hodnota = self.slider.GetValue() # Získá aktuální hodnotu z posuvníku
297         print(f"Nastavená hodnota: {hodnota}") # Vypíše aktuální hodnotu do konzole
298
```

Obrázek 19: Kód pro slider (zdroj: Aplikace PyCharm)

V této části kódu je také vidět další vývojářská pomůcka, kterou jsem cíleně ponechal v kódu a to jednoduchá základní funkce print(). Ta se využívá pro kontrolu dat, která jednotlivé části kódu generují a zapisují se pouze v interním terminálu vývojářského prostředí. Při posunutí slideru je informace prezentována tímto způsobem:



Obrázek 20: Kontrola skrze terminál (zdroj: Aplikace PyCharm)

Proměnná „hodnota“ je následně využita pro filtrování dat získaných a upravených pro statistiku. Musel jsem ale prvně řešit situaci, kdy uživatel nebude chtít upravovat nastavení. V tomto případě aplikuji úvodní myšlenku, kdy zapisuji do šablony pouze položky, které se opakují ve všech rozpočtech. Toho docílím pomocí jednoduchého If příkazu, který vypadá takto:

```
220
221     global hodnota # Deklarace globální proměnné hodnota
222     # Podmínka pro úpravu proměnné "hodnoty"
223     if hodnota == 0:
224         hodnota = pocet_souboru
225         print(hodnota)
226     else:
227         hodnota = hodnota
228
```

Obrázek 21: If příkaz pro proměnné (zdroj: Aplikace PyCharm)

Následuje krok, kdy z dat filtruji pouze položky, které chceme zahrnout do výsledného výstupu. Potřebujeme zahrnout ty položky, jejichž výskyt je v rozmezí od uživatelem stanovené hodnoty až po maximální možný výskyt. To provádím pomocí jednoduché funkce knihovny Pandas .loc, jak je možné vidět v další ukázce kódu společně s dalšími řádky pro kontrolu správného fungování kódu:

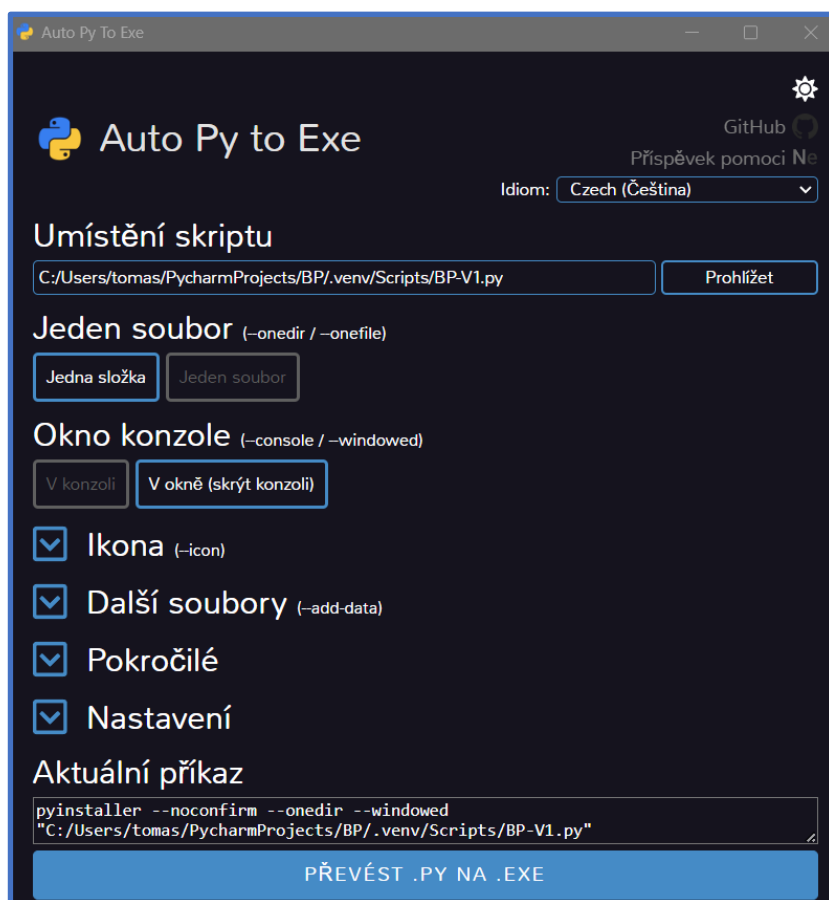
```
228
229     # Vyfiltrování dat pro KR0S soubor
230     KR0S = Data.loc[(Data["četnosti"] >= hodnota) & (Data["četnosti"] <= pocet_souboru)]
231     KR0S = KR0S.reset_index(drop=True)
232     print(KR0S)
233     print(KR0S.columns)
234
```

Obrázek 22: Podmínka filtrace (zdroj: Aplikace PyCharm)

Vyfiltrovaná data následně doplním o potřebné informace pro zapsání do excel šablony. Pomocí knihovny openpyxl zapisuji do připravené šablony (prázdný excel rozpočtu ve formátu KROS). Šablona se nakonec uloží do místa vybraného uživatelem pomocí průzkumníka souborů pomocí knihovny os.

V této kapitole jsem se snažil popsat fungování částí aplikace a metody, které jsem využíval v průběhu vývoje. Jedná se zjednodušené představení aplikace, které je vhodné pro vysvětlení jednotlivých kroků vytváření aplikace. Pro názornější představení je objektivně lepší využít logickou mapu procesů. Logická mapa procesů byla již ve své první verzi zmiňována v teoretické části. Příloha č.2, kterou tvoří druhá verze logické mapy, je doplněna o přidaný blok nastavení parametrů. V logické mapě jsou graficky vysvětleny návaznosti procesů v aplikaci.

Aby mohl budoucí uživatel aplikaci užívat, je potřeba kód exportovat do obecného formátu, který se dá spustit na libovolném počítači. K tomuto účelu jsem nainstaloval do vývojového prostředí knihovnu pro vytvoření spustitelného exe souboru. Jedná se o knihovnu auto-py-to-exe, která zjednoduší export .py (Python) kódu do formátu .exe.



Obrázek 23: auto-py-to-exe (zdroj: Knihovna Auto-py-to-exe)

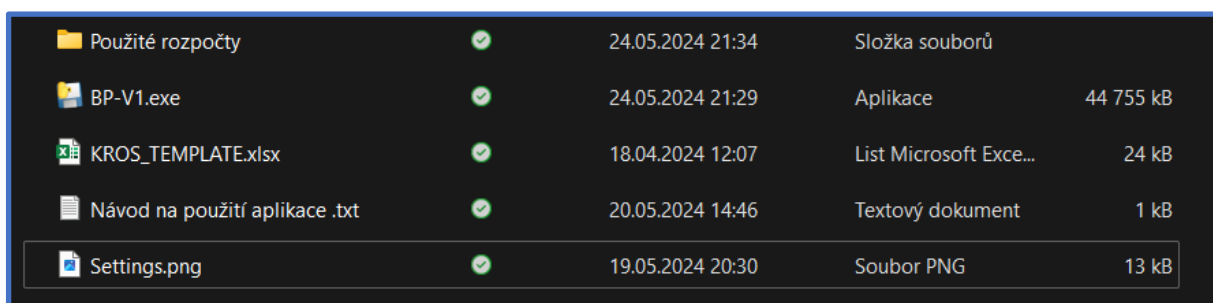
Aby uživatel věděl, jak aplikaci používat, přikládá se k jednoduchým aplikacím textový dokument (standardně s názvem ReadMe), který obsahuje instrukce/jednoduchý manuál, jak správně využívat











program. V mém případě se jedná o jednoduchý .txt soubor s názvem „Návod na použití aplikace“ a obsahuje tyto kroky:

Návod na použití aplikace

- 1. Nahrání souborů: V prvním kroku nahrajte všechny potřebné soubory najednou pomocí tlačítka "Vybrat soubory".*
- 2. Stažení statistiky: Poté stáhněte statistiku kliknutím na tlačítko "Stáhnout statistiku".
Zvolte umístění, kam se má soubor "Statistika" uložit.*
- 3. Analýza grafu: Otevřete soubor "Statistika" a prohlédněte si graf. Na základě zobrazených dat se rozhodněte, jak často by se měla daná položka opakovat mezi rozpočty.*
- 4. Nastavení opakování: Klikněte na tlačítko "Nastavení" (ozubené kolečko) a pomocí posuvné škály zvolte požadovaný počet opakování.*
- 5. Stažení souboru KROS: Nakonec stáhněte soubor "KROS", který je kompatibilní s aplikací KROS 4.
Tento soubor můžete importovat do KROS 4 pomocí funkce "Inteligentní import".*

Aplikace je pro budoucího uživatele dostupná stažením vytvořeného .zip souboru, který bude jednou z příloh této práce. Soubor obsahuje .exe soubor, který spustí aplikaci. Obsahuje také návod na použití aplikace, šablonu pro zapsání výstupu a .png soubor, který v aplikaci reprezentuje ikonu nastavení. Pro účely vyzkoušení fungování aplikace jsou přiloženy i anonymizované rozpočty, na kterých bylo prováděno testování.



	Použité rozpočty		24.05.2024 21:34	Složka souborů	
	BP-V1.exe		24.05.2024 21:29	Aplikace	44 755 kB
	KROS_TEMPLATE.xlsx		18.04.2024 12:07	List Microsoft Exce...	24 kB
	Návod na použití aplikace .txt		20.05.2024 14:46	Textový dokument	1 kB
	Settings.png		19.05.2024 20:30	Soubor PNG	13 kB

Obrázek 24: Soubor s aplikací (zdroj: Příloha č.3)

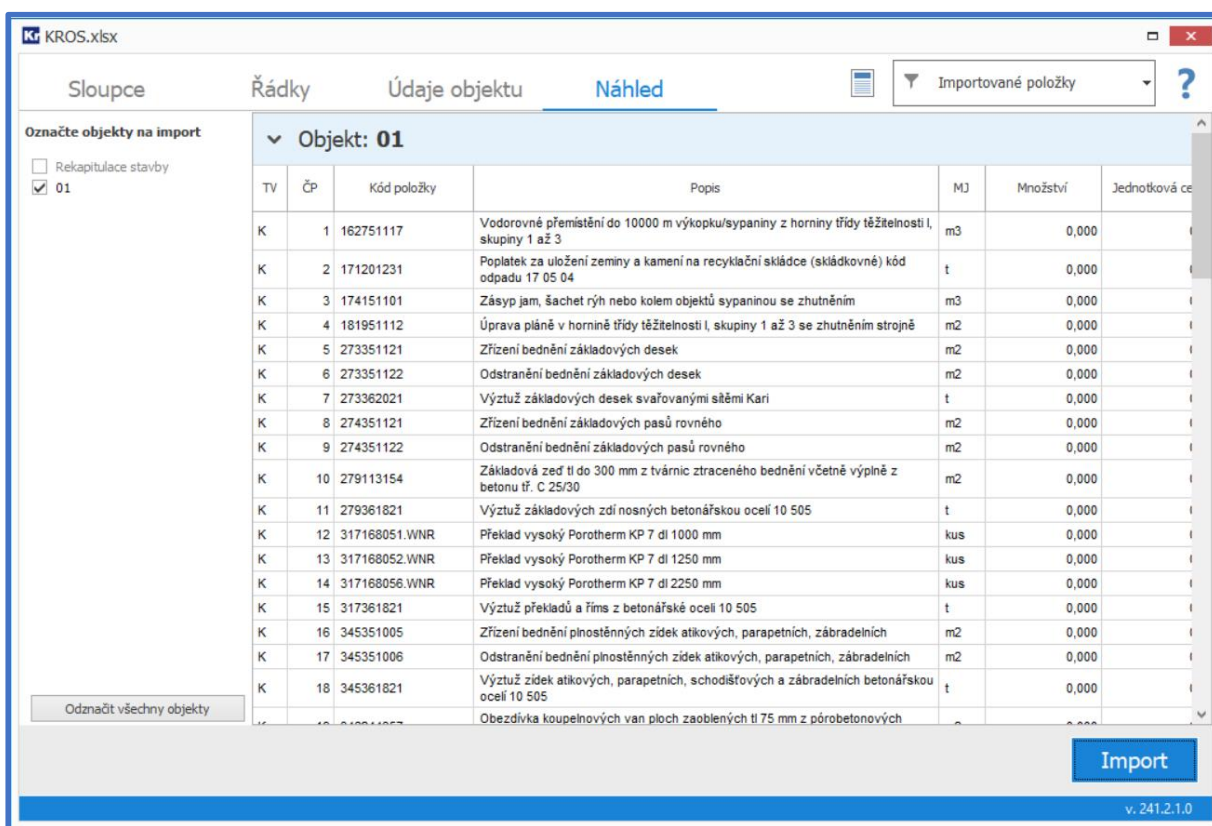
2.3 Demonstrace výstupu

Výstupem funkční aplikace jsou tedy 2 soubory ve formátu MS Excel.

První soubor obsahuje statistiku, který zobrazuje informace o opakujících se položkách v rozpočtech (dále uváděn jako soubor statistika).

Druhý soubor je šablona rozpočtu ve formátu aplikace KROS. Soubor obsahuje položky rozpočtu, které aplikace identifikovala pomocí uživatelem nastaveného parametru. Takto identifikované položky jsou zapsané do připravené excel šablony.

Soubor s vybranými opakujícími se položkami je kompatibilní pro zpětné nahrání do aplikace KROS, za pomoci funkce Inteligentního importu.



TV	ČP	Kód položky	Popis	MJ	Množství	Jednotková cena
K	1	162751117	Vodorovné přemístění do 10000 m výkopku/sypaniny z horniny třídy těžitelnosti I, skupiny 1 až 3	m3	0,000	
K	2	171201231	Poplatek za uložení zeminy a kamení na recyklační skládce (skládkovně) kód odpadu 17 05 04	t	0,000	
K	3	174151101	Zásyp jam, šachet rýh nebo kolem objektů sypaninou se zhutněním	m3	0,000	
K	4	181951112	Úprava pláně v hornině třídy těžitelnosti I, skupiny 1 až 3 se zhutněním strojně	m2	0,000	
K	5	273351121	Zřízení bednění základových desek	m2	0,000	
K	6	273351122	Odstranění bednění základových desek	m2	0,000	
K	7	273362021	Výztuž základových desek svařovanými sítěmi Kari	t	0,000	
K	8	274351121	Zřízení bednění základových pasů rovného	m2	0,000	
K	9	274351122	Odstranění bednění základových pasů rovného	m2	0,000	
K	10	279113154	Základová zeď tl do 300 mm z tvárnice ztraceného bednění včetně výplně z betonu tř. C 25/30	m2	0,000	
K	11	279361821	Výztuž základových zdí nosných betonářskou ocelí 10 505	t	0,000	
K	12	317168051.WNR	Překlad vysoký Porotherm KP 7 dl 1000 mm	kus	0,000	
K	13	317168052.WNR	Překlad vysoký Porotherm KP 7 dl 1250 mm	kus	0,000	
K	14	317168056.WNR	Překlad vysoký Porotherm KP 7 dl 2250 mm	kus	0,000	
K	15	317361821	Výztuž překladů a říms z betonářské ocelí 10 505	t	0,000	
K	16	345351005	Zřízení bednění pinostěnných zidek atkových, parapetních, zábradelních	m2	0,000	
K	17	345351006	Odstranění bednění pinostěnných zidek atkových, parapetních, zábradelních	m2	0,000	
K	18	345361821	Výztuž zidek atkových, parapetních, schodišťových a zábradelních betonářskou ocelí 10 505	t	0,000	
K	19	345361822	Obezdvíka koupelňových van ploch zaoblených tl 75 mm z párobetonových	m	0,000	

Obrázek 25: Import výstupu (zdroj: Aplikace KROS 4)

Inteligentní import v aplikaci KROS automaticky rozpozná pole jednotlivých položek a založí novou zakázku, nebo případně přidá položky do stávající zakázky, kterou je možné specifikovat. Následně stačí pouze aktualizovat rozpočet a položkám se přiřadí jednotkové ceny z aktuální cenové soustavy podle kódu položek.

Program v této formě může pomoci při vytvoření šablony pro rodinné domky. Předvyplnění často používaných položek šetří čas a udrží konzistenci při vytváření více rozpočtů (hromadná výstavba).

Program je ale také možné využít na případné sledování trendů. Porovnáním rozpočtů vytvořených v minulosti s aktuálními rozpočty je možné identifikovat rozdíly v četnosti používání určitých položek

rozpočtů při stavbách. Je možné provádět benchmarking v čase, případně porovnávat svůj projekt s projektem konkurence na veřejných zakázkách.

V tomto programu jsem pracoval pouze s výskytem samotných položek. Jednotlivé položky mohou mít specifikovaný materiál a u všech položek jsou zadávány výkazy výměr. Je možné také ponechat ceny u jednotlivých položek. Pokud by se program rozšířil o načítání těchto dalších dat, bylo by možné provádět daleko komplexnější analýzy. Například právě u veřejných zakázek by šlo automaticky identifikovat, na jakých položkách se liší parametry jednotlivých nabídek.

V případě rodinných domků by šlo s informacemi o materiálu následně vytvořit rozlišné rozpočty s různou mírou kvality materiálu a prezentovat, jaké prvky činí největší rozdíl ve výsledné ceně.

V úvodu jsem již zmiňoval chybovost u manuálního vstupu a lidský faktor.

Za pomoci programu s daty o výkazech výměr by šlo provádět kontroly formátu výkazů.

Dále by šlo provádět kontrolu u souvisejících položek, které by měly vzájemně odpovídat celkovým objemem. Příkladem může být vykopaná zemina se znovu použitou zeminou a odvezenou na skládku, dveře a doplňky ke dveřím typu kování, nebo roletové překlady a samotné rolety apod.

Při manuálním zadávání a velkém množství položek se statisticky zvyšuje možnost chyby, které by taková kontrola dokázala automaticky odhalit a eliminovat.

Program ve své podstatě provádí jednoduché kroky, ale právě jejich realizace může vést k dalším nápadům a postupnému rozšiřování možností.

V neposlední řadě je možné využít pomocný program pro sběr anonymizovaných dat a díky tomu vytvořit dostatečně velké sady dat pro strojové učení, které jsem zmiňoval v kapitole o umělé inteligenci. Opakující se činnosti rozpočtáře jsou v principu ideální kandidát na budoucí implementaci umělé inteligence. A to i s přihlédnutím k čím dál tím více populární návrhovou metodikou BIM.

2.4 Nejčastější chyby a čemu předejít

Jak jsem již zmiňoval v dřívějších kapitolách, určitým faktorem je lidská chybovost a jednou z nejvíce efektivních cest, jak eliminovat chyby je kriticky zhodnotit vlastní postup. Postupem programování se podrobněji zabývám v praktické části, tady ale zmíním určité kroky, které se v důsledku ukázaly neefektivní, či rovnou špatné a vedly k potřebným změnám.

První taková chyba proběhla při dotazování se na relevantní knihovny. Hledal jsem knihovnu, která bude schopná číst data z excelu a následně s nimi pracovat.

Umělá inteligence na základě mého požadavku doporučila jako první volbu knihovnu pandas. Vytvořil jsem na základě jejích funkcí extrakci dat, následné filtrování i cyklaci pro jednotlivé dokumenty. Vše fungovalo bez problémů. Když jsem však začal vytvářet část kódu pro export dat do šablony, zjistil jsem, že knihovna neumí editovat existující excel soubory, ale pouze vytvářet nové. Řešení komplikoval fakt, že dřívější verze knihovny tuto funkci uměla, ale v poslední dostupné verzi již není k dispozici. Musel jsem tedy najít a využít knihovnu openpyxl, která je na editaci daleko více přizpůsobená. Následně jsem využil tuto knihovnu i pro optimalizaci původního cyklu pro načítání dat z excelu. Této chybě bych předešel, kdybych lépe specifikoval požadavky na knihovnu při prvotním dotazu a podrobněji si prostudoval fungování knihovny pandas.

Opakovanou chybou, které jsem se nevyhнул, je pojmenování proměnných. Každou funkci, datovou strukturu nebo proměnnou musí programátor v kódu přesně pojmenovat a následně ji volat s přesným, jednoznačným názvem, pokud ji znovu chce využít. Jmenná konvence proměnných je citlivá i na velká a malá písmena, nebo diakritiku. Příkladem, kde jsem se trápil, byla hodnota celkového počtu souborů získaná při nahrávání souborů. Proměnnou jsem pojmenoval „Pocet_Souboru“. Neintuitivně jsem použil velká písmena. Tato malá chyba mě stála čas při hledání chyby, než jsem si uvědomil, že nově funkci volám ve formě „Pocet_souboru“.

Chyba nemusí být pouze v kódu, ale třeba i ve způsobu, jak zpracováváme data. V aplikaci, kterou jsem vytvořil, analyzuji položky rozpočtu. Původní způsob, kterým jsem identifikoval duplicitní položky, bylo porovnání názvu položky, namísto kódu položky. Kód položky se ukázal jako více unikátní parametr a spíše se upravuje při vytváření takzvaných R položek neboli položek upravených rozpočtářem. Tomu jsem musel přizpůsobit podmínky hledání duplicitních položek. Jednalo se chybu, která měla příčinu ve špatné prvotní přípravě.

Jakákoliv chyba se naštěstí dá nakonec identifikovat a opravit. Právě proto je dobré kód průběžně testovat a následně optimalizovat. Častokrát se jedná o malou chybu, která ale způsobí většinu problémů. V této oblasti lze určitě aplikovat Paretův princip, který říká, že 80 % problémů vzniká z 20 % nejčastějších chyb. Právě malé chyby, překlepy a přehlídnuté maličkosti vám dokáže odhalit během několika sekund umělá inteligence. Proto ji prezentuji jako jeden z nejužitečnějších nástrojů pro programátora – začátečníka.

2.5 Stručný manuál: Programování pro kalkulace a rozpočtování

Proč programovat?

Automatizace analýz, výpočtů a dalších rutinních činností ušetří čas a sníží riziko chyb. Vlastní skript, algoritmus, nebo aplikaci je možné přizpůsobit vašim potřebám a zvýšit tím efektivitu práce v dlouhodobém horizontu.

Využití Umělé inteligence (chatovací model)

Umělá inteligence pro vás bude hlavním asistentem v procesu programování. V průběhu tvorby programu nastanou chvíle, kdy se zaseknete, a právě v takovou chvíli vám může pomoci umělá inteligence. Ptát se chatu na řešení je dobré v případech, kdy například existuje více možností, jak pokračovat, něco nefunguje a nerozumíte chybové zprávě, výstup vypadá jinak, než byste očekávali, chtěli byste efektivnější cyklus a nevíte, jak toho dosáhnout, nejste si jisti, jak implementovat knihovnu, či v podobných situacích.

Základy programování

- **Jazyk:** je třeba si vybrat jazyk ve kterém budete programovat. Velmi doporučuji Python, jakožto populární jazyk pro začátečníky. Pokud chcete pracovat výhradně s daty v excelu, tak excel VBA. Nebo SQL pro správu a vytváření databází. Nejjednodušší cestou pro výběr je konzultace s umělou inteligencí.
- **Syntaxe jazyka:** naučit se základní pojmy a pravidla je nutné pro každý jazyk. Bez toho není možné porozumět fungování kódu. Nejlepší cestou jsou online výukové kurzy. Mezi populární patří placený codecademy anebo volně přístupný W3schools. Na internetu je možné najít ale spoustu dalších kurzů, toto jsou pouze příklady.

Proces vývoje

1. **Problém:** Co potřebuji vyřešit a jak to chci řešit.
2. **Návrh:** Krátký návrh, jak by měl program logicky řešit problém.
3. **Konzultace:** Konzultovat jednotlivé kroky s AI a najít postupy a knihovny.
4. **Kód:** Začít psát kód pro jednotlivé kroky za pomoci umělé inteligence.
5. **Test:** Každý krok testovat a kontrolovat pomocí funkcí pro zobrazení na výstupní konzoli nebo jiné cesty doporučené od umělé inteligence
6. **Kompletace:** jednotlivé kroky, či části programu spojovat a znovu testovat.
7. **Optimalizace:** Umělá inteligence vám může pomoci zefektivnit jednotlivé kroky a poradí, jak přidat případně. další funkce.

Odkazy

Tutoriály: <https://www.codecademy.com/>, <https://www.w3schools.com/>

Umělá inteligence: <https://chat.openai.com/>, <https://gemini.google.com/>, <https://copilot.microsoft.com/>

Fóra: <https://stackoverflow.com/>,

Pamatujte: Začněte jednoduše, experimentujte a učte se z chyb. Umělá inteligence je asistent, ne náhrada. Kód kontrolujte. Můžete se ptát i na nejmenší maličkosti.

Závěr

Tato bakalářská práce se zaměřila na možnosti využití programování v oboru kalkulací a rozpočtování. Prvním cílem bylo identifikovat konkrétní činnosti v rámci kalkulací a rozpočtování, které by mohly být zefektivněny za pomoci programování.

Druhým cílem bylo vytvořit funkční aplikaci, která bude demonstrovat využití programování v této oblasti.

Třetím cílem bylo zpracovat získané znalosti do jednoduchého manuálu, který bude sloužit jako návod pro začátky programování v oblasti kalkulací a rozpočtování.

Nejprve byly v práci vymezeny činnosti rozpočtáře a identifikovány vhodné oblasti pro využití programování. Následně byl představen teoretický základ programování, zejména v kapitolách pojednávajících o výběru programovacího jazyka, o pomocných nástrojích pro programování a v kapitole o využití umělé inteligence jako asistenta při vývoji aplikace.

Teoretické poznatky byly následně využity v praktické části. Zde byly popsány kroky vytvoření aplikace, která na sadě rozpočtů analyzovala společné položky. Tyto identifikované položky aplikace následně zapisuje do vytvořené šablony pro zpětné nahrání do rozpočtářského softwaru. Způsob přípravy a vytvoření této aplikace demonstruje, jak lze aplikovat programování v oblasti kalkulací a rozpočtování. Součástí demonstrace jsou také další identifikované oblasti na rozšíření aplikace. Znalosti získané vytvářením aplikace jsou sepsány do jednoduchého manuálu, který bude sloužit jako praktický návod pro každého, kdo chce i s minimem zkušeností začít s programováním aplikací pro oblast kalkulací a rozpočtování.

Vyhodnocení cílů

1.cíl: Identifikovat potenciál programování v oblasti kalkulací a rozpočtování

Tento cíl byl v rámci bakalářské práce splněn. V práci byly identifikovány činnosti v oblasti kalkulací a rozpočtování, pro které je možné využít programování jako důležitý nástroj pro zefektivnění. Mezi potenciální identifikované přínosy patří úspora času, snížení chybovosti a mimo jiné i zvýšení transparentnosti.

2. cíl: Vytvořit aplikaci pro demonstraci praktického využití programování v kalkulacích a rozpočtování

Tento cíl byl také splněn. V praktické části byla úspěšně vytvořena jednoduchá aplikace, která na sadě souborů analyzuje společné položky vyplněných rozpočtů. Položky aplikace následně dle nastavených parametrů zapisuje do vytvořené šablony pro zpětné nahrání do rozpočtářského softwaru. Tato aplikace tedy demonstruje praktické využití programování pro zefektivnění práce v oboru rozpočtování a kalkulací.

3. cíl: Vypracovat stručný manuál pro programování v oblasti kalkulací a rozpočtování

Tento cíl byl rovněž dosažen. Praktické poznatky z tvorby aplikace a využití pomocných nástrojů jsou sepsány ve vytvořeném stručném manuálu, který popisuje základní kroky při přípravě a programování aplikace. Součástí manuálu jsou i relevantní zdroje.

Všechny stanovené cíle byly splněny. Práce přinesla teoretické poznatky o možnostech využití programování v oblasti kalkulací a rozpočtování. Využití těchto poznatků vedlo k úspěšnému vytvoření funkční aplikace a návodu, jak při tvorbě takové aplikace postupovat. Identifikované další nápady mohou vést v budoucnu k rozšíření aplikace o nové funkce.

Diskuse

Původní předpoklad pracoval s očekáváním, že při porovnávání položek rozpočtů podobných staveb (zde rodinných domků) bude míra opakování položek rozpočtu relativně vysoká. Úspěšně vytvořená aplikace a z ní získané výstupní analýzy tento předpoklad potvrdili jen částečně. Otázkou je, zdali je skutečně příčinou rozdílnost položek u tohoto typu jednotlivých staveb, nebo zda sada testovacích dat obsahovala objekty s různorodým technickým standardem. Dá se předpokládat, že v případě jiné sady dat může být míra opakování vyšší.

Nicméně potenciál aplikace není jen v původním navrženém využití, tedy automatizaci zadání opakujících se položek. Jak je již uvedeno v kapitole „Demonstrace výstupu“, nabízí se i další oblasti k využití. Jde o možnost využít program pro sběr anonymizovaných dat a díky tomu vytvořit dostatečně velké sady dat pro strojové učení, které je již zmíněno v kapitole o umělé inteligenci. Opakující se činnosti rozpočtáře jsou v principu ideální kandidát na budoucí implementaci umělé inteligence. A to i s přihlédnutím k čím dál tím více populárnější návrhovou metodikou BIM.

Program je ale také možné využít na případné sledování trendů. Porovnáním rozpočtů vytvořených v minulosti s aktuálními rozpočty je možné identifikovat rozdíly v četnosti používání určitých položek rozpočtů při stavbách. Je možné provádět benchmarking v čase, případně porovnávat svůj projekt s projektem konkurence na veřejných zakázkách.

V případě většího rozsahu práce by šlo aplikovat automatizaci i na další parametry rozpočtů. Zde program pracoval pouze s výskytem samotných položek. Jednotlivé položky ale mají specifikovaný materiál a u všech položek jsou zadávány výkazy výměr. Je možné také ponechat ceny u jednotlivých položek. Pokud by se program rozšířil o načítání těchto dalších dat, bylo by možné provádět daleko komplexnější analýzy. Například právě u veřejných zakázek by šlo automaticky identifikovat, na jakých položkách se liší parametry jednotlivých nabídek.

Použité zdroje

1. Rozpočtář staveb. <https://nsp.cz> [Online] 15. 8. 2016 [Citace: 03.04.2024]
<https://nsp.cz/jednotka-prace/rozpochtar-staveb>
2. Ing. Luboš Krejčí, ČKAIT profesní informační systém PROFESIS – rozpočtování staveb
<https://profesis.ckait.cz/> [Online] 2018 [Citace: 03.04.2024]
<https://profesis.ckait.cz/dokumenty-ckait/tp-3-1/>
3. Marián Hurta, Textové editory a IDE <https://engeto.cz/> [Online] 23.5.2023 [Citace: 26.04.2024]
<https://engeto.cz/blog/programovani/textove-editory-a-ide/>
4. Příručka k Pycharm <https://www.jetbrains.com> [Online] 24.4.2024 [Citace: 27.04.2024]
<https://www.jetbrains.com/help/pycharm/>
5. Umělá inteligence Gemini.AI. konverzace
6. GitHub repozitář HumanEval <https://github.com> [Online] 2024 [Citace: 02.05.2024]
<https://github.com/openai/human-eval>
7. Google gemini report [Online] 8.2.2024 [Citace: 02.05.2024]
<https://goo.gle/GeminiPaper>
8. Google Gemini release blog <https://blog.google> [Online] 6.12.2024 [Citace: 02.05.2024]
<https://blog.google/technology/ai/google-gemini-ai/>
9. Amazon článek o LLM <https://aws.amazon.com/> [Online] 2023 [Citace: 02.05.2024]
<https://aws.amazon.com/de/what-is/large-language-model/>
10. IBM edukativní článek o API <https://www.ibm.com/> [Online] 9.5.2024 [Citace: 3.05.2024]
<https://www.ibm.com/topics/api>
11. Slovník IT pojmů <https://www.coex.cz> [Online] [Citace: 12.05.2024]
<https://www.coex.cz/blog/slovník-it-pojmu/>
12. IT slovník <https://it-slovník.cz/> [Online] [Citace: 12.05.2024]
13. BIM Industry Working Group. [Online] (2011). Strategy Paper for the Government Construction Client Group. Centre for Digital Built Britain. [Citace: 12.05.2024]
<https://www.cdbb.cam.ac.uk/news/2011BIMStrategyPaper>
14. Inteligentní importy <https://www.urs.cz/> [Online] [Citace: 12.05.2024]
<https://www.urs.cz/software-a-data/kros-4-ocenovani-a-rizeni-stavebni-vyroby/inteligentni-importy> [Citace: 12.05.2024]

15. Pravidla pro flowcharts <https://www.edrawsoft.com> [Online]
<https://www.edrawsoft.com/flowchart-rules.html> [Citace: 18.05.2024]
16. BIM v aplikaci Callida euroCALC 4 <https://callida.cz> [Online]
<https://callida.cz/cs/bim> [Citace: 24.05.2024]

Seznam obrázků

Obrázek 1: euroCALC - BIM (zdroj: (19))	9
Obrázek 2: Parametr ve funkci (zdroj: (4))	13
Obrázek 3: Vysvětlení funkcí – okno (zdroj: Aplikace PyCharm)	13
Obrázek 4: Upozornění na chyby (zdroj: (4))	14
Obrázek 5: Vysvětlení upozornění (zdroj: (4))	14
Obrázek 6: Příklad odkazování (zdroj: Gemini.Ai. konverzace 12.05.2024 (5))	16
Obrázek 7: Benchmark výsledky (zdroj: (7))	17
Obrázek 8: Obrazce flowchart (zdroj: (15))	18
Obrázek 9: Inteligentní importy (zdroj: (14))	22
Obrázek 10: Dotazování knihoven – Vstup (zdroj: Gemini.Ai. konverzace 25.03.2024 (5))	23
Obrázek 11: Dotazování knihoven – Výstup (zdroj: Gemini.Ai. konverzace 25.03.2024 (5))	24
Obrázek 12: Hlavní prostředí aplikace (zdroj: Vlastní aplikace)	24
Obrázek 13: Instalace knihoven (zdroj: Aplikace PyCharm)	25
Obrázek 14: Import knihoven do kódu (zdroj: Aplikace PyCharm)	25
Obrázek 15: Graf analýzy četnosti (zdroj: Výstup Statistika (Příloha č.3))	26
Obrázek 16: Tlačítko nastavení (zdroj: Vlastní aplikace)	27
Obrázek 17: Okno s nastavením (zdroj: Vlastní aplikace)	27
Obrázek 18: Kód pro okno s nastavením (zdroj: Aplikace PyCharm)	28
Obrázek 19: Kód pro slider (zdroj: Aplikace PyCharm)	28
Obrázek 20: Kontrola skrze terminál (zdroj: Aplikace PyCharm)	29
Obrázek 21: If příkaz pro proměnné (zdroj: Aplikace PyCharm)	29
Obrázek 22: Podmínka filtrace (zdroj: Aplikace PyCharm)	29
Obrázek 23: auto-py-to-exe (zdroj: Knihovna Auto-py-to-exe)	30
Obrázek 24: Soubor s aplikací (zdroj: Příloha č.3)	31
Obrázek 25: Import výstupu (zdroj: Aplikace KROS 4)	32

Seznam tabulek

Tabulka 1: Hodnocení programovacích jazyků (zdroj: Gemini.Ai. konverzace 16.03.2024 (5))

Seznam digitálních příloh

- Příloha 1: Flowchart V1
- Příloha 2: Flowchart V2
- Příloha 3: Soubor s výstupem
- Příloha 4: Manuál
- Příloha 5: Kód skriptu