EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103003/B/2024/36124048424963844

METÓDY VIACKRITERIÁLNEHO VYHODNOCOVANIA ALTERNATÍV: APLIKÁCIA V PROCESE KÚPY NEHNUTEĽNOSTI A VÝVOJ VLASTNÉHO SOFTVÉROVÉHO PRODUKTU

Bakalárska práca

2024 **Alex Németh**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103003/B/2024/36124048424963844

METÓDY VIACKRITERIÁLNEHO VYHODNOCOVANIA ALTERNATÍV: APLIKÁCIA V PROCESE KÚPY NEHNUTEĽNOSTI A VÝVOJ VLASTNÉHO SOFTVÉROVÉHO PRODUKTU

Bakalárska práca

Študijný program: Hospodárska informatika
Študijný odbor: Ekonómia a manažment
Školiace pracovisko: Katedra operačného výskumu a ekonometrie
Vedúci záverečnej práce: doc. Ing. Andrea Furková, PhD.

Bratislava 2024 **Alex Németh**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Alex Németh
Študijný program: hospodárska informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Bakalárska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: Anglický

Názov: Metódy viackriteriálneho vyhodnocovania alternatív: aplikácia v procese kúpy nehnuteľnosti a vývoj vlastného softvérového produktu

Anotácia: Práca je zameraná na aplikáciu metód viackriteriálneho vyhodnocovania alternatív. V práci budú objasnené dôležité súčasti teórie viackriteriálneho rozhodovania a následne vybrané metódy aplikované na reálnu rozhodovaciu situáciu, proces kúpy nehnuteľnosti, ktorý je charakteristický mnohými konfliktnými zámermi. V práci bude predstavený vlastný softvérový nástroj na riešenie úloh viackriteriálneho vyhodnocovania alternatív.

Vedúci: doc. Ing. Andrea Furková, PhD.
Katedra: KOVE FHI - Katedra operačného výskumu a ekonometrie
Vedúci katedry: prof. Mgr. Juraj Pekár, PhD.

Dátum zadania: 16.03.2022

Dátum schválenia: 13.04.2023

doc. Ing. Martin Mišút, CSc.
osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne, a že všetku použitú literatúru a ďalšie podkladové materiály, ktoré som použil, uvádzam v zozname použitých zdrojov.

Dátum: .....

(podpis študenta)

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcej mojej bakalárskej práce doc. Ing. Andrei Furkovej, PhD. za poskytnutú pomoc, rady, návrhy a odporúčania.

Abstrakt

NÉMETH, Alex: *Metódy viackriteriálneho vyhodnocovania alternatív: aplikácia v procese kúpy nehnuteľnosti a vývoj vlastného softvérového produktu.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra operačného výskumu a ekonometrie. – Vedúci záverečnej práce: doc. Ing. Andrea Furková, PhD. Bratislava: FHI, 2024, 51 s.

Cieľom bakalárskej práce je vysvetlenie procesu viackriteriálneho vyhodnocovania alternatív a procesu vývoja softvéru. Práca je rozdelená do 5 kapitol. Obsahuje 29 obrázkov a 1 prílohu. Prvá kapitola je venovaná teoretickému vymedzeniu problematiky, uvádza do dôležitosti viackriteriálneho vyhodnocovania alternatív a do zásad vývoja softvéru. V druhej časti sa charakterizuje účel vyvinutej aplikácie. Tretia kapitola popisuje prípravu na jej vývoj, architektúru a implementáciu aplikácie. Štvrtá kapitola sa zaoberá využitými dátami a výsledkami rozhodovacieho procesu. Záverečná kapitola je venovaná diskusii k prínosom aplikácie a možným rozšíreniam v budúcnosti. Výsledkom riešenia danej problematiky je odôvodnenie potenciálu používania vyvinutej aplikácie v rámci rozhodovania sa pri kúpe nehnuteľnosti a vysvetlenie jej prínosu pre bežného človeka.

Kľúčové slová: viackriteriálne rozhodovanie, nehnuteľnosť, vývoj softvéru, angular

Abstract

NÉMETH, Alex: Methods of multi-criteria evaluation of alternatives: application in the real estate purchase process and development of own software product. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of operational research and econometry. – Advisor: doc. Ing. Andrea Furková, PhD. – Bratislava: FHI, 2024, 51 p.

The aim of the bachelor thesis is to explain the process of multi-criteria evaluation of alternatives and the process of software development. The work is divided into 5 chapters. Contains 29 images and 1 appendix. The first chapter is devoted to the theoretical definition of the issue, introduces the importance of multi-criteria evaluation of alternatives and principles of software development. The second part describes the purpose of the developed application. The third chapter describes the preparation for its development, architecture and implementation of the application. The fourth chapter explains the used data and the results of the decision-making process. The final chapter is devoted to a discussion on the contribution of the application and possible improvements in the future. The result of solving the given problem is a justification of the potential of using the developed application in the decision-making process when buying real estate and an explanation of its benefits for ordinary people.

Keywords: multi-criteria decision-making, real-estate, software development, angular

Obsah

Úvod

S rozhodovaním sa stretávame pravidelne. Môže ísť o jednoduché a bežné rozhodnutia, ale aj o dôležité rozhodnutia zahŕňajúce značnú investíciu ako aj práve kúpa nehnuteľnosti. Pri týchto zložitých rozhodovacích situáciách, kde sa možno stretávať s rôznymi konfliktnými zámermi vieme povedať, že ide o viackriteriálne rozhodovanie, pre čo existujú rôzne metódy ktoré môžu človeku napomôcť rozhodnúť sa pre tú ideálnu možnosť.

Rozhodovací proces a implementácia týchto metód však môže byť zložitá, a preto cieľ mojej práce bol vyvinúť webovú aplikáciu, ktorá používateľa cez tento proces prevedie bez potreby akejkoľvek znalosti o týchto metódach.

Zvolená téma umožňuje spojiť môj osobný záujem o vývoj softvéru a možnosť vytvoriť potenciálne hodnotný produkt, ktorý môže byť nápomocný pre kohokoľvek, kto nad kúpou nehnuteľnosti uvažuje, čo je nemalá investícia, ktorej sa v istom bode života venuje takmer každý. Zároveň v dnešnej dobe s neustále narastajúcimi cenami nehnuteľností a vysokými hypotekárnymi úrokmi je tento predmet o to viac vecný a zaujímavý.

1 Teoretické vymedzenie problematiky

1.1 Úvod do viackriteriálneho vyhodnocovania alternatív

1.1.1 Definícia a dôležitosť

Rozhodovanie je proces, s ktorým sa stretávame v každodenných situáciách. Môže ísť o rôzne problémy, ktoré zásadný vplyv na existenciu človeka nemajú, ale zároveň aj o také, ktoré môžu mať na život značný vplyv, ako napr. výber strednej či vysokej školy, kúpa auta, alebo aj kúpa nehnuteľnosti – čo je situácia na ktorú som sa v tejto práci sústredil. (Furková, Ivanovičová, 2017)

Rozhodnutia takéhoto charakteru sú zvyčajne spojené s vynaložením väčších finančných prostriedkov, resp. Ovplyvnia profesijný život jedinca na dlhé obdobie. Človek by sa preto mal snažiť rozhodovať racionálne a maximalizovať svoj úžitok zo zvolenej alternatívy (Furková, Ivanovičová, 2017)

Rozhodovanie medzi alternatívami je jedna z najdôležitejších úloh čo sa týka plánovania v mnohých sférach. Rozhodovatelia sa musia spoliehať na výsledky týchto procesov, aby bolo možné porovnať výkonnosť alternatív, ktoré sa porovnávajú, a následne teda určiť, ktorá alternatíva by bola pre cieľ daného projektu najvýhodnejšia (Janarthanan, Schneider 1986).

Zároveň zložitosť dnešných problémov robí z rozhodovania zložitú úlohu. V mnohých sférach je pri plánovaní potrebné vyberať medzi alternatívami, ktoré obsahujú mnohé kritéria. Práve v takýchto prípadoch výraznú rolu hrajú metódy viackriteriálneho rozhodovania.
Čo sa týka reálnych príkladov, môže ísť v práci o výber projektu, alebo určovanie priorít.
V takomto prípade je potrebné sa zamyslieť akú kvantitu ľudí projekt ovplyvní, koľko času a kapitálu bude potrebné do projektu zainvestovať, a aké výnosy projekt poskytne.

Môže ísť taktiež o rozhodovanie sa pri investovaní. Pri porovnávaní investičných možností zvažujeme volatilitu, risk alebo ročné výnosy, pri výbere spôsobu investovania alebo výbere sprostredkovateľa poplatky alebo dodatočné služby, a v oboch prípadoch pri rozhodovaní je potrebné taktiež zvážiť plánovaný časový horizont investície.

Pri takýchto komplexných rozhodovacích procesoch majú metódy viackriteriálneho rozhodovania veľký potenciál pomôcť rozhodnúť sa matematicky a objektívne.

1.1.2 Metódy stanovenia váh kritérií

Pri viackriteriálnom rozhodovaní je k dispozícii niekoľko metód pre stanovenie váh kritérií a pre vyhodnocovanie alternatív, pričom každá má svoje výhody a nevýhody. Pre stanovenie váh kritérií poznáme nasledovné metódy:

Metóda poradia, pri ktorej rozhodovateľ usporiada kritéria od najdôležitejšieho po najmenej dôležité. Kritériám sa takýmto spôsobom priradzujú hodnoty podľa celkového počtu kritérií, teda v prípade napr. 10 kritérií bude mať to najdôležitejšie kritérium nadobudne hodnotu 10, druhé najdôležitejšie 9, a tak ďalej, až po to najmenej dôležité, ktoré dostáva hodnotu 1 (Furková, Ivanovičová, 2017).

Táto metóda je veľmi jednoduchá avšak v mnohých prípadoch úplne dostačujúca. Môže sa však stať, že pri rozhodovaní považujeme dve kritéria za rovnocenné. Tu metóda poradia zlyháva, a je potrebné použiť inú.

Bodovacia metóda je taktiež relatívne jednoduchá. Pri tejto metóde sa vyžaduje, aby bol rozhodovateľ schopný kvantitatívne ohodnotiť dôležitosť kritérií na zvolenej bodovacej stupnici. Čím je teda kritérium pre rozhodovateľa dôležitejšie, tým by malo byť aj jeho bodové ohodnotenie vyššie (Furková, Ivanovičová, 2017).

Touto metódou je možné dosiahnuť podobné hodnoty váh ako za pomoci metódy poradia, avšak aj v prípadoch, kedy dve kritéria sú pre rozhodovateľa rovnako dôležité, je to v poriadku – jednoducho daným kritériám priradíme rovnakú hodnotu.

Metóda kvantitatívne párového porovnávania kritérií (Saatyho metóda) je v porovnaní s ostatnými spomenutými metódami značne komplexnejšia. Pri tejto metóde rozhodovateľ dôležitosť kritérií porovnáva navzájom. Stupne dôležitosti sa vyjadrujú na stupnici 1 až 9 (Furková, Ivanovičová, 2017).

1	Vyjadruje rovnocenné kritéria
3	Vyjadruje slabo preferované kritérium
5	Vyjadruje silno preferované kritérium
7	Vyjadruje veľmi silno preferované kritérium
9	Vyjadruje absolútne preferované kritérium

Pri tejto metóde sa všetky kritéria porovnávajú v matici párovo. Zároveň je možné využiť aj hodnoty 2, 4, 6 alebo 8, ktoré sa považujú ako medzistupne medzi vyššie zadefinovanými dôležitosťami (Furková, Ivanovičová, 2017).

	y_1	y_2	y_3	y_4	y_5
y_1	1	3	2	5	7
y_2	1/3	1	1/2	4	3
y_3	1/2	2	1	5	1/3
y_4	1/5	1/4	1/5	1	4
y_5	1/7	1/3	3	1/4	1

Obrázok 1 - Príklad Saatyho matice
Zdroj: Furková, Ivanovičová, 2017

Taktiež platí, že pri pridelení hodnoty pre jeden pár v matici, „zrkadlová“ hodnota matice sa vyjadruje ako menovateľ v zlomku s číslom 1 na mieste čitateľa. Ak teda pri porovnaní kritéria 1 s kritériom 2 pridelíme hodnotu 5, tak pre porovnanie kritéria 2 s kritériom 1 môžeme zapísať hodnotu 1/5.

Po vyplnení matice sa môže pre každé kritérium vypočítať geometrický priemer a následne aj váhové podiely.

1.1.3 Metódy viackriteriálneho rozhodovania

Po stanovení a vypočítaní váh kritérií je teda potrebné využiť niektorú z dostupných metód viackriteriálneho rozhodovania medzi alternatívami. Tieto metódy sa pri výpočtoch výsledkov dôrazne líšia, a podobne ako metódy pre stanovenie váh kritérií, všetky metódy majú svoje výhody a nevýhody. Medzi tieto metódy patria:

Metóda váženého súčtu, pri ktorej sa využíva princíp maximalizácie užitočnosti. Výsledky tejto metódy dosahujú pre každú alternatívu hodnotu 0 až 1, pričom 1 znamená, že alternatíva je najužitočnejšia, a 0 dosiahne tá najmenej výhodná (Furková, Ivanovičová, 2017).

Pre dosiahnutie výsledkov touto metódou je potrebná maximalizácia aj minimalizačných kritérií, a následný výpočet všetkých kritérií s ich príslušnými váhami pre každú alternatívu.

Metóda TOPSIS, alebo celým názvom „Technique for Order of Preference by Similarity to Ideal Solution“ sa snaží nájsť alternatívu, ktorá je geometricky najbližšie

k ideálnej, alebo najďalej od bazálnej. Ideálnu alternatívu by sme dokázali poskladať tak, že by sme z alternatív s ktorými pracujeme zobrali pre každé kritérium tú najlepšiu hodnotu. Pri bazálnej je princíp rovnaký až na to, že hodnoty vyberáme pre každé kritérium tie najhoršie (Furková, Ivanovičová, 2017).

Pri tomto postupe je najprv potrebná normalizácia dát podľa nasledovného vzťahu:

$$r_{ij} = \frac{y_{ij}}{\left(\sum_{i=1}^{n} y_{ij}^2\right)^{1/2}} \quad 1 \quad i = ,2,\ldots,n, \quad j = 1,2,\ldots,k$$

Obrázok 2 - Vzťah pre normalizáciu alternatív

Zdroj: Furková, Ivanovičová, 2017

Po normalizácií sa vypočítajú prvky váženej kriteriálnej matice, a následne sú určené ideálne a bazálne alternatívy. Ďalej sa počíta pre každú alternatívu jej vzdialenosti od týchto hodnôt, a nakoniec sa vypočíta ukazovateľ, ktorý reprezentuje relatívnu vzdialenosť alternatív od tej bazálnej.

Metódy triedy PROMETHEE

Metódy tejto triedy patria do princípu vyhodnocovania na základe preferenčnej relácie, ktoré zohľadňujú informácie ako preferencia, indiferencia či neporovnateľnosť medzi dvojicami alternatív vzhľadom na ich kritéria. Párové relácie ovplyvňujú aj prahové hodnoty. Tieto metódy sa líšia v tom, že pri ich implementácií nie je potrebná normalizácia (Furková, Ivanovičová, 2017).

Metódy týchto tried však vzhľadom na ich potrebu dodatočných informácií od rozhodovateľa neboli implementované, nakoľko jeden z cieľov vyvíjanej aplikácie je minimalizácia práce používateľa pre to, aby nejaké výsledky dostal.

1.2 Zásady vývoja softvéru

1.2.1 Úvod do metodológie vývoja

Vývoj softvéru obvykle taktiež nasleduje isté metodológie, najmä čo sa týka spôsobu určovania priorít, a spôsobu organizovania projektu alebo deľby práce v prípade že na produkte pracuje niekoľko ľudí súčasne. Medzi najznámejšie metódy patria napr.:

***Waterfall (vodopádový) model*,** ktorý je jeden z najstarších a najtradičnejších prístupov k vývoju softvéru. Ide o lineárny proces, v ktorom sa prechádza cez niekoľko krokov alebo fáz

vývoja, pričom každá nasledujúca fáza je závislá na tej predošlej. Vodopádový model je najčastejšie využívaný v projektoch, kde potreby zákazníkov sú správne chápané od začiatku, a je malá šanca že tieto podmienky sa v budúcnosti budú meniť (GeeksforGeeks, 2024). Fázy vodopádového modelu sú:

Analýza

- Projekt sa začína analýzou, pri ktorej sa podrobne zbierajú podmienky, potreby a predstavy zainteresovaných strán. Súčasťou tejto fázy je zároveň detailná dokumentácia týchto podmienok.

Dizajn systému

- Po analýze nastáva fáza dizajnu, kde podľa potrieb zadefinovaných v prvej fáze sa navrhuje architektúra a dizajn systému. Do tejto fázy spadá všeobecná štruktúra a špecifikácia dizajnu produktu.

Implementácia

- Vo fáze implementácie sa môže začať s programovaním a vývojom podľa rozhodnutej štruktúry.

Testovanie

- Vyvinutý softvér sa testuje, aby sa uistilo, že spĺňa zadefinované potreby a podmienky, a že všetko funguje tak ako má.

Nasadzovanie

- Po úspešnom testovaní sa môže softvér nasadiť. To znamená sprístupnenie softvéru pre užívateľov alebo zákazníkov.

Údržba

- Záverečná fáza vodopádového modelu zahŕňa dlhodobú údržbu a podporu softvéru. V tejto fáze sa ošetrujú akékoľvek nové chyby a poruchy a zároveň často sa naďalej v rámci softvéru vyvíja nová funkcionalita a iné vylepšenia podľa spätnej väzby užívateľov či zákazníkov (GeeksforGeeks, 2024).

Agile metodológia je modernejší prístup k softvérovému vývoju, ktorý zdôrazňuje flexibilitu, spoluprácu a dlhodobý alebo opakovaný vývoj. Agile metodológia uprednostňuje prispôsobivosť a neustále vylepšovanie softvéru. Zvyčajne sa priority a úlohy zadávajú na krátkodobé obdobia - inak nazývané aj šprinty – s cieľom každý šprint dodať nejaké

vylepšenia a nové funkcionality, umožňujúc neustále softvér prispôsobovať podľa spätnej väzby používateľov a zákazníkov. Agilné tímy sa skladajú z členov špecializujúcich sa v rôznych disciplínach, ako napr. vývojári, dizajnéri, testeri a podobne. Celý tím počas vývoja úzko komunikuje a spolupracuje. Zároveň sa vyžaduje väčší dôraz na zákazníkov, ktorých potreby sa časom môžu rozvíjať alebo meniť, a sú taktiež zahrnutý v celom procese, prinášajúc okrem spätnej väzby aj svoje názory o tom čo by sa pri vývoji malo do budúcnosti uprednostňovať, aké funkcionality očakávajú, aké zmeny by privítali a podobne (Brush, Silverthorne, 2022).

1.2.2 Dôležitosť vývoju softvéru v moderných aplikáciách

Vývoj softvéru hrá dnes dôležitú rolu v moderných aplikáciách a to v rôznych sférach. V dnešnej digitálnej dobe sa firmy a organizácie stále viac spoliehajú na automatizáciu procesov, zefektívňovanie práce a prínos hodnoty a služieb pre zákazníka práve pomocou aplikácií. Softvérový vývoj umožňuje týmto firmám zostať konkurencieschopnými, a inovatívnymi. Umožňuje neustále vytvárať nové produkty a služby, a ponúka priestor na experimentovanie s novými nápadmi a konceptami a rýchle sprostredkovanie riešení a funkcionality pre neustále nové potreby.

Správne navrhnuté softvérové riešenie môže drasticky zlepšiť užívateľské rozhranie a ich skúsenosť pri využívaní sprostredkovávaných služieb, čo môže viesť k zvýšeniu spokojnosti zákazníkov a k udržaniu ich vernosti.

Moderné aplikácie taktiež umožňujú uchovávanie veľkého počtu dát a štatistík zo spôsobu, akým zákazníci aplikácie využívajú, ktoré možno analyzovať a spätne ich využiť pri nových nápadoch a rozhodnutiach.

Zároveň s neustále narastajúcim rizikom kybernetických hrozieb, zaručenie bezpečnosti je taktiež súčasťou toho vývoja softvéru. Vývojári musia implementovať komplexnú autentifikáciu a autorizáciu a dodržiavať prísne požiadavky na ochranu citlivých informácií aby sa zákazníci cítili pri využívaní týchto aplikácií bezpečne.

1.2.3 Prehľad relevantných programovacích jazykov a nástrojov

Programovacích jazykov dnes existuje už nespočetne, preto táto kapitola slúži ako prehľad predovšetkým jazykov relevantných pre vývoj webových aplikácií, ktorý sa skladá z troch základných jazykov:

HTML (HyperText Markup Language) je „šablónový“ jazyk, ktorý určuje štruktúru a rozloženie webovej stránky. Je to takmer úplný základ vývoja webových aplikácií, a jednoduchú statickú webovú stránku je možné vyvinúť pomocou iba HTML kódu (Moyers).

CSS (Cascading Style Sheets) pracuje práve s HTML šablónami, a je to jazyk zodpovedný za ich vizuálnu reprezentáciu. Pomocou CSS je možné HTML elementy dizajnovať a „štýlovať“. V CSS vieme detailnejšie určovať rozloženie stránky, farby a veľkosti elementov, druh písma a mnoho iných detailov (MOOC, 2021).

JavaScript je dynamický skriptovací jazyk pomocou ktorého je možné vyvíjať funkcionalitu ktorá vylepšuje interaktivitu stránok. JavaScript zvyčajne predstavuje kód, ktorý vo webových aplikáciách reprezentuje logiku, a teda mení ich zo statických na dynamické (MOOC, 2021).

TypeScript je programovací jazyk vyvinutý spoločnosťou Microsoft, ktorý rozširuje JavaScript o statické typy. Je navrhnutý predovšetkým budovanie veľkých aplikácií a jeho kód je počas „runtime“ (behu programu) prekompilovaný do štandardného a čitateľného JavaScript kódu, aby ho mohol webový prehliadač prečítať (Ars Technica, 2012).
Je to tzv. „superset“ JavaScriptu, čo znamená že každý funkčný JavaScriptový kód je zároveň funkčný TypeScriptový kód, a prináša vlastnosti ako triedy, moduly a iné rozšírenia, ktoré boli eventuálne pridané taktiež do JavaScriptu.

TypeScript je zvyčajne obľúbený hlavne pre bezpečnosť kódu a schopnosť nájsť možné chyby ešte pred kompiláciou. V prieskumoch z roku 2020 až 78% respondentov prehlásilo, že využíva tento programovací jazyk (State of JS, 2020) a bol zahlasovaný za druhý najobľúbenejší jazyk na svete (Stack Overflow, 2020).

Vo vývoji webových aplikácií je zároveň dnes už bežné, že sa vyvíjajú pomocou webových frameworkov alebo knižníc, a to najznámejšie napr. **React**, **Angular**, **Vue** alebo **Svelte**. Tieto frameworky poskytujú prehľadnejšiu štruktúru a lepšie organizovaný kód, čo napomáha k urýchleniu vývoja (Davidson, 2023).

Tieto webové frameworky poskytujú predpripravené komponenty, šablóny a knižnice, umožňujúc vývojárov sústrediť sa viac na logiku aplikácie namiesto písania opakujúceho sa kódu. Presadzujú modularitu pomocou architektúry na báze komponentov, kde časti užívateľského rozhrania sú zapuzdrené do opakovateľne použiteľných

komponentov, využívajú rôzne techniky ktoré optimalizujú rýchlosť a odozvu aplikácie, a zahŕňajú praktické riešenia pre manažment a synchronizáciu dát a tým uľahčujú narábanie so zložitými dátovými tokmi (Davidson, 2023).

Detaily implementácií a konkrétnych funkcionalít sa môžu medzi týmito frameworkami líšiť, ale všetky majú spoločný cieľ optimalizácie procesu vývoja a výkonu aplikácie.

Čo sa týka nástrojov, neoddeliteľnou súčasťou vývoja aplikácií sú aj systémy na kontrolu verzii – najznámejšie **Git**.

Git umožňuje vývojárom udržať si prehľad o všetkých priebežných zmenách vykonaných v kóde, zahŕňajúc informácie o tom kto zmeny vykonal, kedy ich vykonal, ako aj čo konkrétne bolo zmenené. Ide o zabezpečenie integrity kódu a zároveň možnosť ktorékoľvek zmeny v prípade potreby vrátiť do predošlého stavu. Tento systém umožňuje bezprostrednú spoluprácu v tíme vývojárov. Poskytuje centralizované úložisko, ktoré je prístupné a modifikovateľné pre celý tím, zároveň však bráni situáciám, kedy niekoľko vývojárov vykonáva zmeny v rovnakom kóde naraz. V takomto prípade systém zablokuje zápis zmien, kým sa konflikty v zmenách manuálne nevyriešia (MoldStud, 2024).

1.2.4 Zosúladenie s cieľmi vývoju webovej aplikácie

Pre vývoj webovej aplikácie ako nástroj pre viackriteriálne rozhodovanie pri kúpe nehnuteľnosti je dôležité nasledovať správne praktiky.
Pri navrhovaní aplikácie proces dramaticky uľahčí správne nastavenie architektúry aplikácie – so správnymi rozhodnutiami pri návrhu je možné opätovné využívanie raz napísaného kódu a funkcionality (Lee, Lee, 2005), a zjednoduší spôsob implementácie rôznych metód viackriteriálneho rozhodovania sa medzi alternatívami. Práve takúto architektúru umožní využitie ktoréhokoľvek zo spomínaných webových frameworkov.

Rovnako dôležité je sústrediť sa na to, aby aplikácia bola jednoducho používateľná pre rozhodovateľov, preto pri implementácií je dobré sa rozhodnúť pre využitie takých metód, ktoré umožnia jednoduchú prácu s aplikáciou, a plynulý chod od začiatku rozhodovania až po výsledky.

1.3 Terminológia

1.3.1 Kľúčové pojmy rozhodovacieho procesu

Alternatívy – Rozličné porovnávané možnosti medzi ktorými sa rozhoduje

Kritéria – Atribúty alebo iné metriky na základe ktorých sa porovnávajú a vyhodnocujú jednotlivé alternatívy.

Váhy – Relatívna dôležitosť pridelená pre všetky kritéria v číselnej forme. Vyjadrujú preferencie rozhodovateľa ako aj dôležitosť všetkých kritérií v procese rozhodovania.

Rozhodovacia matica – Reprezentácia alternatív a kritérií v tabuľkovom formáte

1.3.2 Kľúčové pojmy vývoja softvéru

Frontend – Používateľské rozhranie a interakcia používateľa s aplikáciou

Backend – Logika na strane serveru a interakcia databázy ktorá podporuje aplikáciu

Databáza – Štruktúrovaná kolekcia dát organizovaná pre efektívny výber a manipuláciu

Open-source – Označenie typu softvéru - softvér je open-source ak má softvér voľne dostupný zdrojový kód ktorý môže prehliadať alebo redistribuovať ktokoľvek bez akýchkoľvek obmedzení. Open-source softvér je často preferovaný vzhľadom na to že užívateľ môže vidieť presne čo na svoj počítač inštaluje, ako aj z dôvodu možnosti vlastného prispôsobovania.

Cross-platform – Dostupnosť na rôznych platformách – cross-platformový softvér je dostupný pre rôzne operačné systémy a zariadenia ako napr. Windows, MacOS, Linux, Android, iOS a podobne.

Framework – Rozšírenie programovacieho jazyka, ktoré slúži na urýchlenie a zjednodušenie vývoju aplikácií.

Komponent – Kus kódu, ktorý zahŕňa špecifickú časť užívateľského rozhrania alebo funkcionalitu. Komponenty sú základ moderného prístupu k vývoju webových aplikácií.

Dependency injection – Programovací princíp, pri ktorom namiesto toho aby si komponenty či súbory manažovali závislosti sa do nich „vstrekujú“ z externých súborov. Vďaka tomuto spôsobu môže byť veľa častí logiky aplikácie udržiavaných vo vlastných súboroch a na prácu s nimi ich môže importovať akýkoľvek komponent.

URL – Adresa webovej stránky alebo aplikácie.

Smerovanie (Routing) – Vo webe reprezentuje koncept mapovania rozličných častí aplikácie pre rozličné URL adresy.

State management – proces manažmentu stavu aplikácie ktorý zahŕňa údržbu a ovládanie rôznych komponentov, tokov dát a nastavení čím sa zabezpečuje správny a efektívny chod aplikácie. Je nevyhnutný najmä vo veľkých a komplexných aplikáciách kde sa dáta neustále nejakým spôsobom aktualizujú.

Tech stack – Súbor programovacích jazykov, knižníc, frameworkov a nástrojov, ktoré sa spolu využívajú pre budovanie jadra aplikácie.

2 Cieľ práce

Cieľom práce je vývoj webovej aplikácie ktorá umožní bežnému používateľovi aplikovať a využiť metódy viackriteriálneho rozhodovania medzi alternatívami pri kúpe nehnuteľnosti.

Aplikácia by mala cieľového používateľa sprevádzať definíciou kritérií rozhodovania, odhadom váh zvolených kritérií a definíciou samotných alternatív, medzi ktorými sa rozhoduje. Nakoniec by aplikácia mala poskytnúť zhrnutie alternatív aj s príslušnými hodnotami, ktoré budú výsledkom rozhodovacieho procesu, na základe ktorého bude možno vidieť, ktoré z alternatív sú podľa zvolených kritérií a zadaných váh najefektívnejšie alebo najvýhodnejšie.

Používanie aplikácie by malo byť jednoduché pre kohokoľvek. Navigácia by mala byť jednoduchá a zrozumiteľná, a všetok zložitej logiky by mal bežať na pozadí.
Aplikácia po spracovaní dát od používateľa vyhodnotí všetky alternatívy od najvýhodnejšej po najhoršiu niekoľkými rozhodovacími metódami zvlášť a zároveň výsledky zo všetkých metód porovná a zhodnotí do zjednoteného hodnotenia.

Aplikácia by mala byť dostupná v slovenskom aj anglickom rozhraní, pričom v oboch jazykoch bude dostupný aj dataset pre rýchle odskúšanie alebo demonštráciu chodu aplikácie.

3 Vývoj webovej aplikácie

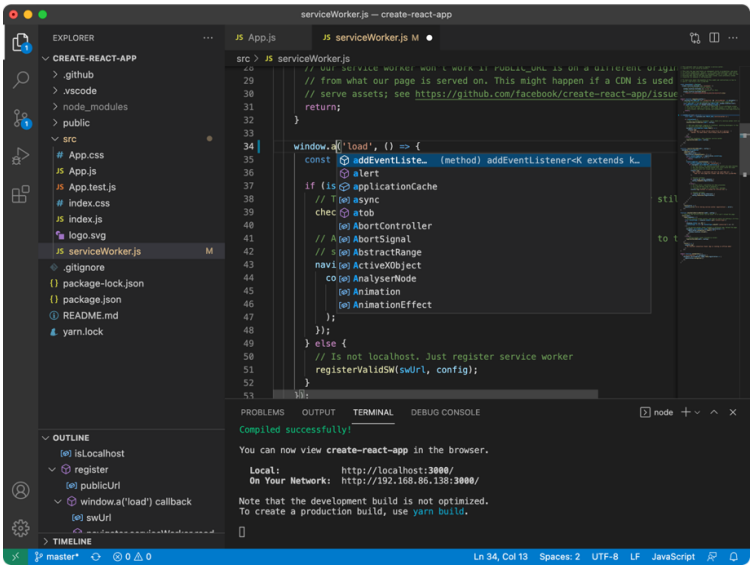
3.1 Výber jazykov, knižníc a nástrojov

Ako integrované vývojárske prostredie bol využitý Visual Studio Code. Aplikácia bola napísaná v TypeScripte v Angular frameworku, za pomoci Angular Material knižnice. Pre systém na kontrolu verzií bol využitý Git.

3.1.1 Visual Studio Code

Visual Studio Code je populárny, bezplatný, open-source a cross-platform editor kódu vyvíjaný spoločnosťou Microsoft od roku 2015 dostupný pre Windows, macOS a Linux. Podporuje inštaláciu veľkého množstva rozšírení napr. pre podporu rozličných jazykov alebo frameworkov či integráciu rôznych nástrojov a poskytuje veľa funkcií často vyhľadávaných v komplexných integrovaných vývojárskych prostrediach ako napr.:

- Syntax Highlighting
- Inteligentné doplňanie kódu
- Refaktoring kódu
- Debugging (ladenie chýb)
- Integrácia version control (Git)



Obrázok 3 - Visual Studio Code

Zdroj: https://github.com/microsoft/vscode

Napriek všetkému, čo Visual Studio Code poskytuje, je navrhnutý tak aby bol rýchly a mohol byť používaný aj na slabších počítačoch – s odporúčanou rýchlosťou procesora len 1.6 GHz a 1 GB pamäte RAM (Visual Studio).

3.1.2 Angular

Taktiež bezplatný a open-source projekt, Angular je jeden z najpopulárnejších frameworkov pre vývoj webový aplikácií vyvíjaný spoločnosťou Google.

Angular poskytuje veľmi obsiahlu škálu funkcionalít a nástrojov pre budovanie zložitých webových aplikácií, ako napr. vlastný routing systém, manažment foriem, dependency injection a podobne. Je to preto dobrá možnosť predovšetkým pre veľké podnikové projekty, avšak pre jeho striktnejšiu architektúru a pravidlá, ktoré zaručujú organizovanú a prehľadnú štruktúru projektu môže byť atraktívnou možnosťou aj pre jednoduchšie aplikácie.

Angular pracuje primárne s TypeScriptom, ktorý ako bolo už vysvetlené poskytuje statické typovanie premenných a funkcií, čo taktiež napomáha k skorému chytaniu a riešeniu možných chýb a zlepšuje udržateľnosť zdrojového kódu.

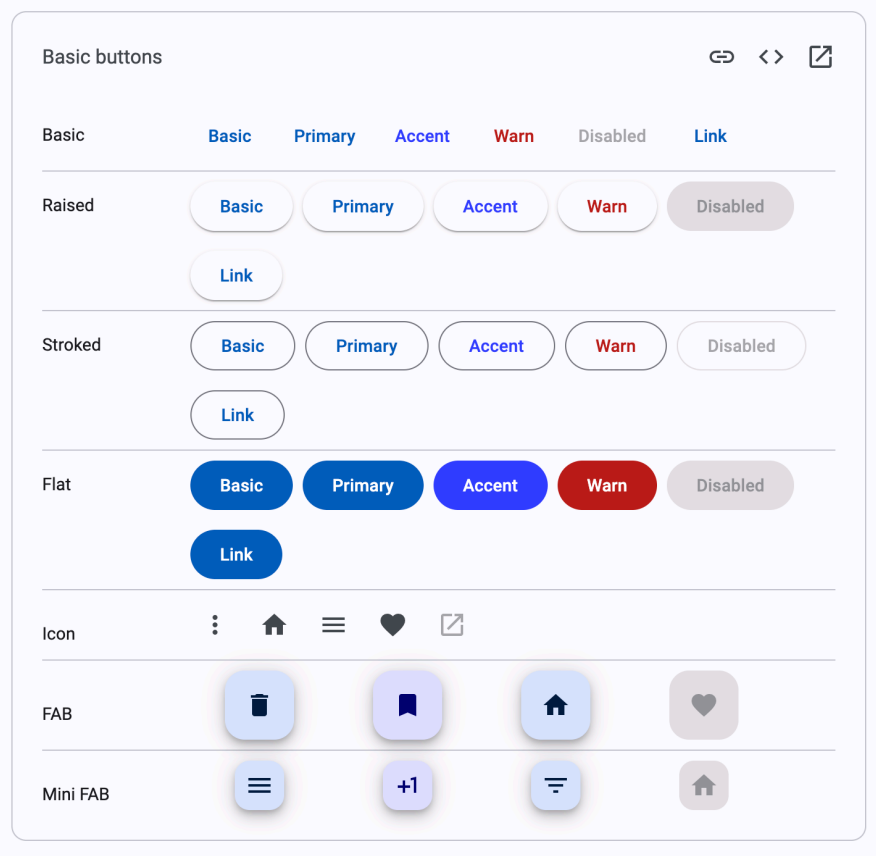
Tento webový framework v porovnaní s ostatnými môže byť spočiatku ťažší na používanie práve pre jeho striktnosť a obsiahlosť, čo však zároveň limituje potrebu využívania dodatočných knižníc pre riešenie špecifických problémov. Angular poskytuje oveľa viac nástrojov pre riešenie rozsiahlych problémov než ostatné webové frameworky (Singh, 2023). Ak však k potrebe dodatočných externých knižníc príde, vďaka jeho popularite je ekosystém knižníc pre Angular rovnako rozsiahly a nájsť potrebné knižnice kompatibilné s Angularom zvyčajne problém nie je.

3.1.3 Angular Material

Je to knižnica vopred navrhnutých a postavaných komponentov pre webové aplikácie podľa dizajnu Google. Material Design existuje ako knižnica vo viacerých formách, no Angular Material je vyvíjaný vývojármi Angularu špecificky na použitie spolu s týmto frameworkom. Angular Material poskytuje komponenty ako napr.: Polia s automatickým dopĺňaním textu, tlačidlá, začiarkavacie políčka, polia s výberom, kalendáre, ikony, animácie, tabuľky, karty, dialógové okná, formuláre a viac.

Komponenty tejto knižnice sa dajú ďalej jednoducho modifikovať. Je možné využiť jednu z predpripravených farebných paliet, ktoré obnášajú spravidla niekoľko farebných prevedení:

- Basic (základná)
- Primary (hlavná)
- Accent (detaily)
- Warning (Varovanie alebo chyba)
- Disabled (zablokovaná)
- Link (odkaz)



Obrázok 4 - Variácie základných Angular Material tlačidiel

Zdroj: https://material.angular.io/components/button/overview

Ako možno na obrázku vidieť, takýmto spôsobom je možné pre väčšinu komponentov nie len si vybrať jeden zo zadefinovaných štýlov, ale aj jednu z menovaných farieb, ktoré závisia od zvolenej témy. Konkrétne farby, či štýl písma sa dajú ďalej prispôsobiť aj s vlastnými farbami.

3.1.4 Tailwind

Tailwind (alebo TailwindCSS) je taktiež štýlovacia knižnica ktorá dokáže proces navrhovania urýchliť. Umožňuje štýlovať elementy v aplikácií priamo v HTML, a to pomocou preddefinovaných tried. Síce spôsob fungovania Tailwindu nevyhovuje každému vývojárovi, no tento spôsob ktorým elementy vieme štýlovať má potenciál byť oveľa rýchlejší než využívanie tradičného CSS štýlovania. Namiesto vytvárania špecifických tried pre rozličné elementy, vieme štýly ktoré by sme tej triede chceli priradiť písať priamo ako samotné triedy.

Pomocou Tailwindu možno použiť takýto zápis tried pre element:

```
<div class="flex flex-row gap-2 pt-3">
```

Obrázok 5 - Príklad Tailwind štýlovania

Zdroj: Vlastné vypracovanie

Pričom taký zápis je ekvivalentný v rámci štandardného štýlovania vytvoreniu a prideleniu triedy s nasledovnými parametrami v CSS:

```
<div class="className">
```

```
.className {
    display: flex;
    flex-direction: row;
    gap: 2px;
    padding-top: 3px;
}
```

Obrázok 6 - Príklad štandardného štýlovania

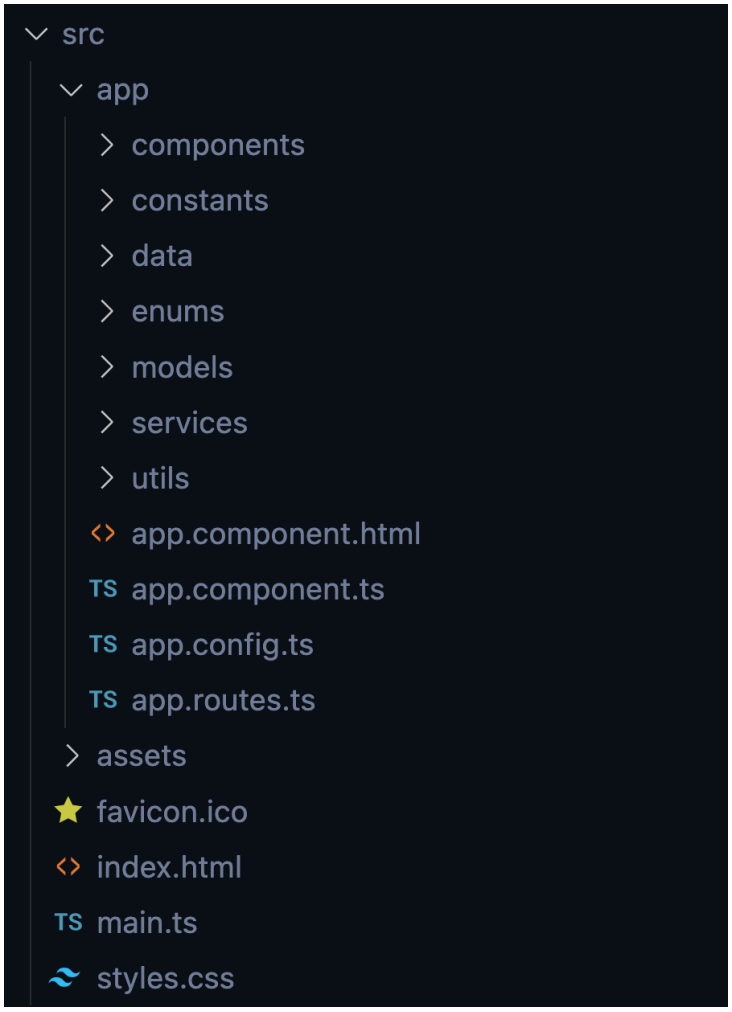
Zdroj: Vlastné vypracovanie

Tailwind poskytuje rôzne triedy, pričom každá z nich robí niečo konkrétne, podľa čoho je názov danej triedy aj zvolený. Pre príklad, v prípade potreby elementu prideliť výplň (padding), možno využiť triedu napríklad „p-1“ pre padding (výplň) s hodnotou 1, čo je v tailwinde ekvivalent 4 pixelom. Trieda „p-5“ by bola teda ekvivalentná výplni s veľkosťou 20 pixelov.

Všetky poskytnuté triedy je možné nájsť v dokumentácii Tailwindu voľne dostupnej na ich webovej stránke.

3.2 Architektúra projektu

Ako bolo spomenuté už v prvej kapitoly, správne nastavenie architektúry projektu napomáha k udržateľnosti a prehľadnosti. Zdrojový adresár je logicky rozdelený do niekoľkých hlavných priečinkov, ktoré sú detailne opísané v nasledovných podkapitolách.

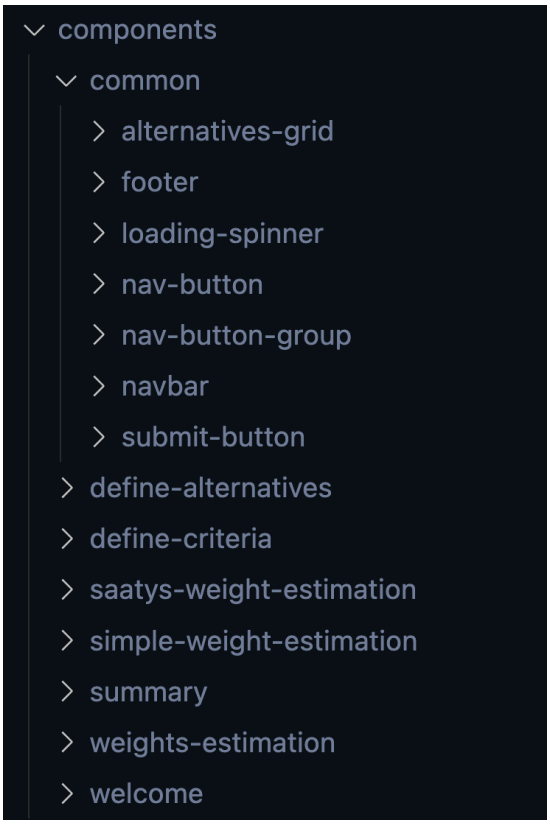


Obrázok 7 - Architektúra projektu

Zdroj: Vlastné vypracovanie

3.2.1 Komponenty

Priečinok komponentov je rozdelený do bežných (common) komponentov, a do obsiahlejších špecifických užívateľských rozhraní, ktoré často tieto bežné komponenty prepoužívajú, či už s dátami relevantnými pre dané rozhranie, alebo iba staticky.



Obrázok 8 – Komponenty

Zdroj: Vlastné vypracovanie

Väčšina rozhraní prepoužíva napríklad bežný komponent navigačných tlačidiel (nav-button-group). Rozhranie tieto tlačidlá zobrazuje, s tým že posiela tlačidlám informáciu, kam tlačidlá majú užívateľa navigovať. Poskytnuté adresy komponent tlačidiel tieto premenné očakáva a sám ich namapuje na relevantné funkcie.

Tento prístup, kde každý objekt alebo komponent je zodpovedný len za svoju konkrétnu funkcionalitu, a pracuje len s dátami ktoré potrebuje sa nazýva aj „Single Responsibility Principle“, alebo skrátene „SRP“. Je to jeden z piatich fundamentálnych princípov objektovo-orientovaného programovania nazývaných SOLID, ktoré predstavil Robert C. Martin už pred vyše 20 rokmi, a sú dodnes považované za jedny z najlepších smerníc pre vývoj softvéru (Orner, 2021).

3.2.2 Konštanty a dáta

Súbor konštánt hosťuje a exportuje statické premenné, ktoré sú využívané tiež vo viacerých častiach aplikácie. Výhoda izolácie takýchto statických a prepoužívaných dát

vlastných súborov je to, že ak nejakú z premenných potrebujeme zmeniť, stačí ich zmeniť priamo tu, a všetky komponenty ktoré tieto premenné importujú budú automaticky aktualizované.

Priečinok dát funguje na rovnakej báze, až na to, že hosťuje dáta vo forme objektov alebo polí. V prípade tejto aplikácie sa tu nachádzajú demonštračné datasety nehnuteľností a kritérií v slovenskom aj v anglickom jazyku, a v prípade ich využitia sa načítajú do globálnych premien podľa toho, aký jazyk je v aplikácií v tom momente zvolený, aby aplikácia s týmito dátami mohla od toho bodu pracovať.

3.2.3 Enumerácie

Enumerácie idú ruka v ruke s typovaním, ktoré poskytuje TypeScript. V prípade situácií, kedy chceme nejakej premennej povedať, že jej hodnota môže byť len nejaká z konkrétneho zoznamu hodnôt, môžeme tejto premennej prideliť typ danej enumerácie.

V aplikácií existuje napríklad enumerácia pre typy upozornení:

```
export enum AlertType {
    Primary = "primary",
    Accent = "accent",
    Warning = "warning",
    Error = "error",
}
```

Obrázok 9 - Príklad enumerácie

Zdroj: Vlastné vypracovanie

Využitie tejto enumerácie možno predstaviť na funkcií, ktorá v aplikácií zobrazuje hlásenia. Funkcia očakáva dobrovoľný argument, ktorý musí byť práve tohto typu.

```
showAlert(
    message: string,
    type: AlertType = AlertType.Primary,
    durationInSeconds: number = 5000
) {
    this.snackBar.open(message, "✖", {
        duration: durationInSeconds * 1000,
        panelClass: [type],
    });
}
```

Obrázok 10 - Funkcia na zobrazovanie upozornení

Zdroj: Vlastné vypracovanie

Ak funkcia typ upozornenia nedostane, bude pracovať s predvoleným typom, čo je na tomto príklade primárny typ. V každom prípade, funkcia zavolá ďalšiu funkciu, ktorej opäť pridelí tento typ ako panelovú triedu, čo na obrazovke používateľa ovplyvňuje výzor upozornenia.

3.2.4 Modely

Modely, anglicky „interfaces“ (rozhrania) sú objektové štruktúry, ktoré v TypeScripte možno zadefinovať. Ak očakávame, že nejaké objekty musia spĺňať špecifický tvar, v resp. očakávame, že objekt by mal obsahovať špecifické polia, je dobrým pravidlom najmä pre komplexnejšie objektové štruktúry vytvoriť model, na ktorý možno v akejkoľvek časti aplikácie neskôr odkazovať. Objektový model nám dáva všetky tieto informácie, a v prípade, že chceme z objektov vyťahovať dáta, či už vo funkciách alebo v užívateľskom rozhraní, TypeScript nám zároveň vie ďalej pri vývoji napomáhať.

```
export interface ICriteria {
    id: string;
    description?: string;
    title: string;
    minmax: string;
    weight?: number;
    weightPercentage?: number;
    idealValue?: number;
    basalValue?: number;
}
```

Obrázok 11 - Objektový model kritérií

Zdroj: Vlastné vypracovanie

V modeloch je taktiež možné zadefinovať voliteľné polia. Z príkladu na obrázku možno vidieť, že pole popisu (description) v modeli má v zápise poľa na konci otáznik. Vďaka tomuto zápisu je možné TypeScriptu povedať, že toto pole sa v objekte môže, no nemusí nachádzať.

Pri voliteľných poliach je však dôležité na túto skutočnosť dbať a ošetrovať funkcie pracujúce s nimi tak, aby funkcia nezlyhala v prípade že pole v objekte neexistuje.

```
getCriterionDescription(criterionTitle: string): string {
    return (
        this.criteria.find((criteria) => criteria.title === criterionTitle)
            ?.description || ""
    );
}
```

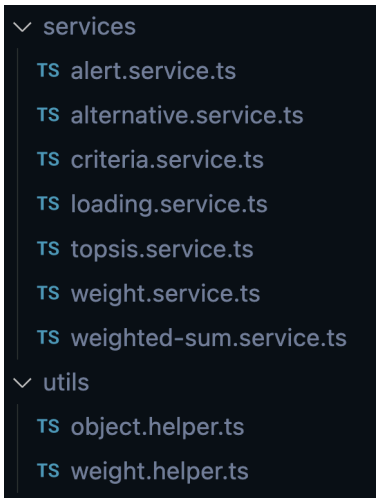
Obrázok 12 - Funkcia vracajúca popis kritéria

Zdroj: Vlastné vypracovanie

Funkcia na obrázku, ktorá očakáva názov kritéria ako argument, prehľadáva pole všetkých zadefinovaných kritérií a vyberá kritérium ktorého názov sa zhoduje s názvom ktorý dostala. Následne funkcia vráti buď popis tohto kritéria, alebo – ak sa žiadne kritérium nenájde, alebo kritérium toto pole neobsahuje – vráti len prázdny reťazec.

3.2.5 Služby

Drvivá väčšina logiky aplikácie sa nachádza v priečinku služieb (services), ktorý je ďalej rozdelený do izolovaných súborov podľa toho, s akou časťou aplikácie alebo objektami pracuje.



Obrázok 13 - Prehľad služieb aplikácie

Zdroj: Vlastné vypracovanie

Dá sa tu nájsť individuálne logiku upozornení, logiku na prácu s alternatívami, logiku na prácu s kritériami, a podobne. Tieto služby nesú zároveň zodpovednosť za state management. Udržiavajú totižto hodnoty verejných globálnych premenných, ku ktorým po načítaní služby môžu komponenty pristupovať a využívať ich priamo rovnako ako funkcie spojené pre narábanie s nimi.

```
export class DefineAlternativesComponent implements OnInit, AfterViewInit {
    constructor(
        private criteriaService: CriteriaService,
        private alternativeService: AlternativeService,
        private formBuilder: FormBuilder,
        private loadingService: LoadingService
    ) {}

    ngOnInit(): void {
        this.criteria = this.criteriaService.criteria;
        this.initForm();
    }

    ngAfterViewInit(): void {
        this.loadingService.hide();
    }
```

Obrázok 14 - Využívanie služieb v komponentoch

Zdroj: Vlastné vypracovanie

Komponent na obrázku v rámci svojho konštruktora potrebné služby „vstrekuje“, po čom vo funkcií ešte pred načítaním komponentu prideľuje premennú kritérií do svojej vlastnej premennej, čo umožňuje komponentu s ňou pracovať. Zároveň na obrázku možno vidieť vstreknutú službu načítavania (LoadingService), ktorá zahŕňa logiku pre prácu s animáciami ktoré dávajú užívateľovi na javo, že aplikácia sa ešte načítava. Po tom, ako sa tento komponent načíta, volá z tejto služby funkciu „hide“, ktorá animáciu načítavania skrýva.

```
const ALLOWED_INPUTS = [
    "0",
    "1",
    "2",
    "3",
    "4",
    "5",
    "6",
    "7",
    "8",
    "9",
    "1/1",
    "1/2",
    "1/3",
    "1/4",
    "1/5",
    "1/6",
    "1/7",
    "1/8",
    "1/9",
];

export function checkWeightInput(input: string) {
    return ALLOWED_INPUTS.includes(input);
}
```

Obrázok 15 - Pomocná služba pre kontrolu váh

Zdroj: Vlastné vypracovanie

Jednoduchý príklad pomocnej funkcie pre kontrolu vstupov v rámci polí, kde užívateľ v prípade voľby Saatyho metódy na odhad váh vypisuje hodnoty do porovnávacej matice. Nakoľko pri Saatyho metóde dávajú zmysel len hodnoty od 1 po 9 a zároveň od 1/1 po 1/9, v tejto službe sú tieto hodnoty zadefinované v poli. Funkcia pre kontrolu vstupov očakáva hodnotu ktorú užívateľ zadal, a jednoducho skontroluje, či sa taká hodnota nachádza v zadefinovanom poli povolených hodnôt. Funkcia na základe toho vracia booleovskú hodnotu – teda buď „áno“ alebo „nie – a v rámci komponentu možno ďalej výsledok tejto funkcie využiť na to, aby sa pri správne zadanej hodnote pole vysvietilo do zelena, alebo pri zlej do červena. Tiež možno túto funkciu využiť na to, aby nepovolila užívateľovi posun do ďalšej fázy aplikácie v prípade že všetky zadané hodnoty nie sú správne, alebo pomocou upozornení užívateľovi povedať, aký problem nastal, a čo treba urobiť pre pokračovanie v rozhodovaní.

3.3 Implementácia

V tejto časti práce je uvedený prehľad o implementácií fungovania aplikácie, ktorá sprevádza používateľa cez štyri hlavné fázy procesu rozhodovania. Pri každej fáze zobrazujeme hlavičku aplikácie, v ktorej sa nachádza logo a názov aplikácie, na ktoré je možné kliknúť pre návrat na úvodnú obrazovku v akejkoľvek fáze, možnosti pre zmenu jazyka medzi anglickým a slovenským, a indikátor, ktorý používateľovi dáva najavo postupnosť procesu.

Okrem úvodnej obrazovky tieto štyri fázy sú:

- Definícia kritérií
- Odhad váh kritérií
- Definícia alternatív
- Výsledky

V aplikácií je implementovaná fialovo-zelená farebná paleta poskytnutá v Angular Material verzií 15. Nakoľko však pri štýlovaní je využívaný aj Tailwind - ktorý má taktiež svoju predvolenú farebnú škálu - farby z tejto Material palety sme taktiež zadefinovali v nastaveniach Tailwindu, čo nám umožňuje tieto farby využívať aj s týmito triedami.

Je dôležité spomenúť, že webové aplikácie v Angulari je tzv. „Single Page Application“, čo znamená, že všetky zmeny v rozhraní sa dejú na jednej stránke, čo urýchľuje výkonnosť

aplikácie. Napriek tomu sme Angular poskytuje svoj pseudo-routing, do ktorého sme zadefinovali adresy ktoré sú spojené s danými komponentami. Takýmto spôsobom môžeme napr. pre navigačné tlačidlá nastaviť adresu, na ktorú majú tlačidlá používateľa poslať, podľa čoho sa užívateľovi zobrazí zodpovedný komponent.

```
export const routes: Routes = [
    {
        path: "",
        title: "Evalutron",
        component: WelcomeComponent,
    },
    {
        path: "criteria",
        title: "Define Criteria - Evalutron",
        component: DefineCriteriaComponent,
    },
    {
        path: "weights-estimation",
        title: "Estimation of Weights - Evalutron",
        component: WeightsEstimationComponent,
    },
    {
        path: "alternatives",
        title: "Define Alternatives - Evalutron",
        component: DefineAlternativesComponent,
    },
    {
        path: "summary",
        title: "Summary - Evalutron",
        component: SummaryComponent,
    },
];
```

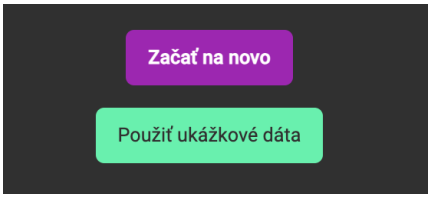
Obrázok 16 - Angular Routing

Zdroj: Vlastné vypracovanie

3.3.1 Úvodná obrazovka

Na úvodnej obrazovke aplikácia používateľa privíta s krátkym popisom o tom, na čo aplikácia slúži. V pätičke úvodnej obrazovky sa nachádzajú informácie ako aktuálna verzia, dátum aktuálnej verzie a odkaz na zdrojový kód aplikácie.

Pod popisom aplikácie má užívateľ možnosť začať proces rozhodovania dvoma spôsobmi.



Obrázok 17 - Začiatok rozhodovacieho procesu

Zdroj: Vlastné vypracovanie

V prípade že používateľ zvolí možnosť ukážkových dát, aplikácia pred presunom do prvej fázy najprv načíta demonštračné dáta, ktoré používateľ môže využiť. Používateľ sa môže proces rozhodovania začať aj „na novo“, čo znamená, že tieto dáta sa nenačítajú, a používateľ môže následne pri procese rozhodovania využiť svoje vlastné dáta.

3.3.2 Definícia kritérií

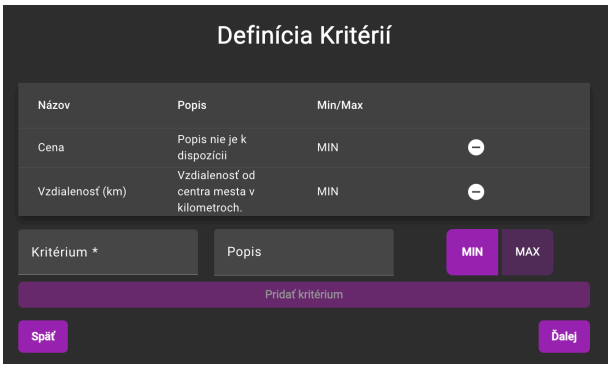
Po začatí rozhodovacieho procesu, prvá obrazovka ktorú používateľ uvidí je rozhranie pre zadefinovanie kritérií. Je to jednoduchá forma, kde používateľ môže vytvárať kritéria, podľa ktorých sa chce rozhodovať. Okrem názvu kritérií používateľ volí, či je kritérium minimalizačné alebo maximalizačné, a vo formulári je taktiež voliteľné pole pre popis kritéria. Vytvorenie nového kritéria je ošetrené podmienkou, že názov kritéria, ktoré chceme pridať ešte neexistuje.

```
if (this.criteria.some((c) => c.title === this.formGroup.value.title)) {
    this.formGroup.controls.title.setErrors({ duplicate: true });
    return this.alertService.showAlert(
        `Criteria with title '${this.formGroup.value.title}' already
        exists.`,
        AlertType.Error
    );
}
```

Obrázok 18 - Prevencia duplicitných názvov kritérií

Zdroj: Vlastné vypracovanie

Hlavný komponent v tomto zobrazení pracuje so službou kritérií, ktorá zahŕňa globálnu premennú v ktorej poli budú uložené všetky zadefinované kritéria. Komponent volá funkcie tejto služby pre vytváranie a pridávanie kritérií do tohto poľa, a zároveň v tabuľke nad formulárom zobrazuje doposiaľ zadefinované kritéria. Zo zadefinovaných kritérií sa zároveň dá kritéria dodatočne mazať.



Obrázok 19 - Definícia kritérií

Zdroj: Vlastné vypracovanie

Tlačidlo pre pokračovanie v rozhodovacom procese je zároveň ošetrené a zablokované, kým používateľ nezadefinuje aspoň 2 rozličné kritéria.

```
[disableForward]="criteria.length < 2"
```

Obrázok 20 - Zablokovanie tlačidla

Zdroj: Vlastné vypracovanie

Kým táto podmienka nie je splnená, a pri presune kurzora na tlačidlo sa zobrazuje pomocná správa, ktorá hovorí o tejto potrebe.

3.3.3 Odhad váh kritérií

Nasledovná fáza aplikácie je pre odhadovanie váh predošle zadefinovaných kritérií. Používateľ má na výber použitie jednoduchej bodovacej metódy, alebo Saatyho porovnávaciu metódu. Komponent zodpovedný za túto obrazovku hosťuje komponenty pre obe metódy, a zobrazuje podmienečne rozhranie pre tú, ktorú si používateľ zvolil.

Criteria	Value	Váhový podiel
Cena	1	16.667%
Vzdialenosť (km)	1	16.667%
Počet izieb	1	16.667%
Rozloha (m2)	1	16.667%
Stav	1	16.667%
Parkovanie	1	16.667%

	Cena	Vzdialenosť (km)	Počet izieb	Rozloha (m2)	Stav	Parkovanie	Geometrický priemer	Váhový podiel
Cena	1	1	1	1	1	1	1	16.667%
Vzdialenosť (km)	1	1	1	1	1	1	1	16.667%
Počet izieb	1	1	1	1	1	1	1	16.667%
Rozloha (m2)	1	1	1	1	1	1	1	16.667%
Stav	1	1	1	1	1	1	1	16.667%
Parkovanie	1	1	1	1	1	1	1	16.667%

Obrázok 21 - Rozhrania pre odhad váh

Zdroj: Vlastné vypracovanie

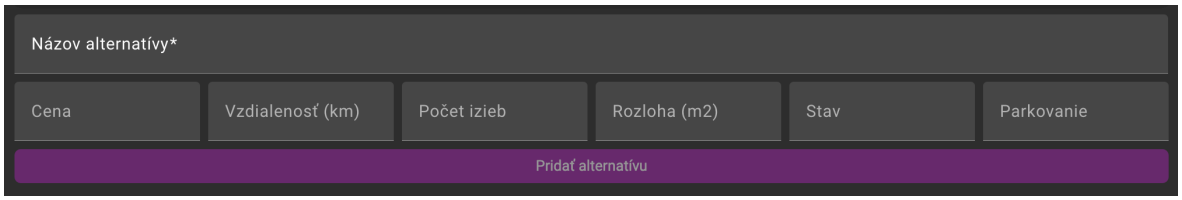
Keď používateľ vypĺňa tieto hodnoty, dostáva vizuálnu spätnú väzbu. V prípade, že používateľ vloží správnu hodnotu, pole sa vysvieti na zeleno. Ak je hodnota v nesprávnom formáte, vysvieti sa na červeno.

Údaje sa v globálnych premenných relevantných služieb sa po zmenách vo formulári aktualizujú, a váhovy podieľ (pri Saatyho metóde aj geometrický priemer) sa automaticky prepočítava.

3.3.4 Definícia alternatív

V tejto fáze sa používateľovi zobrazuje podobný formulár ako pri definícií kritérií, s jedným značným rozdielom v rámci logiky – tento formulár je z väčšej časti dynamický. Pri

zadávaní alternatív používateľ musí poskytnúť názov alternatívy, po čom sa vo formulári nachádzajú polia pre zadávanie hodnôt pre všetky kritéria, ktoré používateľ zadefinoval.



Obrázok 22 - Formulár pre vkladanie alternatív

Zdroj: Vlastné vypracovanie

V tomto formulári je potrebné vyplniť všetky polia. Bez tejto podmienky tlačidlo na pridanie alternatívy nie je sprístupnené. Zároveň, rovnakým spôsobom ako pri kritériách sa nad formulárom nachádza tabuľka zostavená zo všetkých doposiaľ zadefinovaných alternatív. Toto je posledná fáza aplikácie pri ktorej sa vyžaduje nejaký vstup od používateľa. Po pokračovaní ďalej, aplikácia už začína s výpočtami a rozhodovacím procesom. Vzhľadom na to, že jeden z cieľov aplikácie bol jednoduchosť a minimalizácia nadbytočných potrebných vstupov som sa rozhodol pre metódu váženého súčtu a metódu TOPSIS, pri ktorých kompletne stačia údaje ktoré sa v procese doposiaľ nazbierali, zatiaľ čo pre implementáciu napr. metódy PROMETHEE II by bolo potrebné od používateľa vyžiadať dodatočné informácie ako prahy indiferencie alebo prahy ostrej preferencie.

3.3.5 Metóda váženého súčtu

Pre výpočet váženého súčtu je potrebné najprv všetky dáta maximalizovať, vrátane tých, kde kritéria sú prvotne zadefinované ako minimalizačné. O toto sa stará funkcia *maximizeValues* v službe váženého súčtu.

```
maximizeValues() {
    this.alternativeService.findMinMaxValues();
    this.alternativeService.alternatives.forEach(
        (alternative: IAlternative) => {
            Object.keys(alternative.rawValues!).forEach((key: string) => {
                alternative.weightedSumValues!.maximized = {
                    ...alternative.weightedSumValues!.maximized,
                    [key]:
                        this.criteriaService.criteria.find(
                            (c: ICriteria) => c.title === key
                        )?.minmax === "MIN"
                            ? this.alternativeService.minValues[key] /
                              alternative.rawValues![key]
                            : alternative.rawValues![key] /
                              this.alternativeService.maxValues[key],
                };
            });
        }
    );
}
```

Obrázok 23 - Maximalizácia hodnôt
Zdroj: Vlastné vypracovanie

Funkcia pracuje aj so službami alternatív a kritérií, a ako prvý krok invokuje funkciu, ktorá vyhľadá a uloží najvyššie a najnižšie hodnoty pre všetky kritéria. Následne prebieha cyklus cez pole všetkých alternatív, ktorý obsahuje ďalší vnorený cyklus pre všetky surové hodnoty alternatív. Cyklus ďalej vytvára nové pole, do ktorého vypočítava maximalizované hodnoty alternatívy pre každé z kritérií. Pri každom kritériu prebieha kontrola či je kritérium minimalizačné alebo maximalizačné. V prípade minimalizačných kritérií, výpočet maximalizovanej hodnoty sa skladá z vydelenia najnižšej hodnoty pre dané kritérium s hodnotou kritéria pre danú alternatívu. V prípade, že kritérium maximalizačné už je, delí sa naopak hodnota kritéria s najvyššou hodnotou daného kritéria.

Funkcia pre výpočet už samotného váženého súčtu pre každú alternatívu – *calculateWeightedSums()* – môže ďalej s maximalizovanými hodnotami pracovať.

```
calculateWeightedSums() {
    this.alternativeService.alternatives.forEach(
        (alternative: IAlternative) => {
            let weightedSum = 0;
            this.criteriaService.criteria.forEach(
                (criterion: ICriteria) => {
                    const criterionValue =
                        alternative.weightedSumValues!.maximized![
                            criterion.title
                        ];
                    const weightedValue =
                        criterionValue * criterion.weightPercentage!;
                    weightedSum += weightedValue;
                }
            );
            alternative.weightedSumValues!.finalValue = weightedSum;
        }
    );
}
```

Obrázok 24 - Funkcia pre výpočet váženého súčtu
Zdroj: Vlastné vypracovanie

Táto funkcia funguje taktiež na báze cyklu cez pole definovaných alternatív. Pre každú alternatívu inicializuje premennú s názvom *weightedSum* s hodnotou 0, po čom v rámci každej alternatívy sa vykonáva opäť vnorený cyklus cez pole definovaných kritérií. K predošle definovanej iniciovanej potom funkcia pripočíta maximalizovanú hodnotu každého kritéria vynásobenú s váhovým podielom tohto kritéria. Na koniec funkcia priradí hodnotu tejto premennej do poľa alternatívy. Táto hodnota reprezentuje vážený súčet, a vzhľadom na cyklus sa vypočíta pre každú alternatívu.

3.3.6 Metóda TOPSIS

Výpočet TOPSIS hodnôt je o niečo zložitejší, a skladá sa z piatich funkcií. Ako prvé sa vypočítajú normalizované hodnoty každej alternatívy. Základ funkcie zodpovednej za tento výpočet sú rovnaké vnorené cykly, ako pri maximalizačnej funkcii pred výpočtom váženého súčtu. Rozdiel je v samotnom výpočte, kde delíme súčtom hodnôt všetkých alternatív buď negatívnu hodnotu kritéria v prípade že je minimalizačné, alebo pri maximalizačných obyčajnú surovú hodnotu.

Nasledovná funkcia robí ďalší prepočet už týchto normalizovaných hodnôt, a výsledok ukladá do nového poľa alternatív. Tento medzi krok zahŕňa vynásobenie normalizovaných hodnôt každého kritéria alternatívy s váhovým podielom daného kritéria.

Po tejto funkcií je možné získať ideálnu a bazálnu hodnotu pre každé kritérium pomocou nasledovnej funkcie:

```
getIdealAndBasalValues() {
    this.criteriaService.criteria.forEach((criterion: ICriteria) => {
        let values: number[] = [];
        this.alternativeService.alternatives.forEach(
            (alternative: IAlternative) =>
                values.push(
                    alternative.topsisValues!.calculated![criterion.title]
                )
        );
        criterion.idealValue = Math.max(...values);
        criterion.basalValue = Math.min(...values);
    });
}
```

Obrázok 25 - Výpočet ideálnych a bazálnych hodnôt kritérií

Zdroj: Vlastné vypracovanie

Funkcia pre každé kritérium vytvorí prázdne pole, do ktorého následne pridáva hodnoty vypočítané predošlou funkciou z každej alternatívy pre dané kritérium. Funkcia na konci tohto cyklu zapisuje pre všetky kritéria najnižšiu hodnotu v poli ako bazálnu hodnotu kritéria, a tú najvyššiu ako ideálnu.

Pokračuje sa s funkciou na výpočet hodnôt V+ a V- (inak povedané ideálne a anti-ideálne hodnoty) pre každú alternatívu, *getVPlusAndMinusValues()*.

```
getVPlusAndMinusValues() {
    this.alternativeService.alternatives.forEach(
        (alternative: IAlternative) => {
            alternative.topsisValues!.vPlus = 0;
            alternative.topsisValues!.vMinus = 0;
            this.criteriaService.criteria.forEach(
                (criterion: ICriteria) => {
                    alternative.topsisValues!.vPlus! += Math.pow(
                        alternative.topsisValues!.calculated![
                            criterion.title
                        ] - criterion.idealValue!,
                        2
                    );
                    alternative.topsisValues!.vMinus! += Math.pow(
                        alternative.topsisValues!.calculated![
                            criterion.title
                        ] - criterion.basalValue!,
                        2
                    );
                }
            );
            alternative.topsisValues!.vPlus = Math.sqrt(
                alternative.topsisValues!.vPlus
            );
            alternative.topsisValues!.vMinus = Math.sqrt(
                alternative.topsisValues!.vMinus
            );
        }
    );
}
```

Obrázok 26 - Výpočet hodnôt V+ a V-

Táto funkcia v rámci cyklu pre každú alternatívu iniciuje polia pre ideálnu a bazálnu hodnotu na 0, a pre každú alternatívu pripočítava hodnoty do iniciovaných premenných. Pre V+ funkcia pripočítava mocniny rozdielu prepočítanej normalizovanej hodnoty kritéria z druhého kroku a ideálnej hodnoty toho kritéria. Pre V- je to opäť mocnina rozdielu tej istej hodnoty a anti-ideálnej hodnoty kritéria. Po skončení cyklu sa premenné pre V+ a V- odmocnia, a tým sa funkcia ukončuje.

Posledná funkcia ktorá už môže vypočítať konečné TOPSIS hodnoty má jednoduchú úlohu – zapísať do každej alternatívu do poľa finálnej TOPSIS hodnoty V- vydelené sčítanými V- a V+ hodnotami.

3.3.7 Zhrnutie

Všetky výpočty popísané v predošlej kapitole sa spúšťajú akonáhle používateľ zadefinuje alternatívy a klikne na tlačidlo “Ďalej”. Sú volané počas inicializácie posledného komponentu ktorý následne zobrazí súhrnnú tabuľku s výsledkami z oboch metód pre všetky alternatívy, v ktorej je možné tabuľku zoraďovať podľa všetkých stĺpcov. Samotným výsledkom sa však venuje už nasledujúca kapitola.

4 Výsledky práce

Táto kapitola slúži ako úvod do použitého demonštračného datasetu a jeho pôvod, zvolené kritéria, rozoberá výsledky rozhodovacieho procesu pre tieto dáta, a sumarizuje možné využitia či prínosy aplikácie.

4.1 Prehľad použitých dát

Demonštračný dataset aplikácie je výber nehnuteľností v Bratislave, ktoré boli inzerované na predaj na známom webovom portáli v Decembri roku 2022. Kritéria, ktoré sa berú do úvahy sú rôzne relevantné, kvalitatívne aj kvantitatívne vlastnosti týchto bytov.

Alternatívy		Kritéria					
		k1	k2	k3	k4	k5	k6
	Nehnuteľnosť	Cena (EUR)	Lokalita (vzdialenoť v km)	Počet izieb	Metráž	Stav	Parkovanie
a1	Martinčekova, Ružinov	339000	5	4	86	3	3
a2	Nejedlého, Dúbravka	299800	8.1	3	89	4	2
a3	Furdekova, Petržalka	279000	6.5	4	87	2	1
a4	Štefunkova, Ružinov	195000	6.6	2	65	1	1
a5	Palisády, Staré Mesto	249900	0.75	2.5	62	1	3
a6	Kadnárova, Rača	285000	7.6	3	75	4	3
a7	Znievska, Petržalka	199900	7.8	3	84	2	3
a8	Furdekova, Petržalka	233000	6.5	3	70	3	3
a9	Kopčianska, Petržalka	299990	6.3	3	84	4	1
a10	Ivana Bukovčana, Devínska Nová Ves	239700	13.5	4	86	3	2
a11	Starohradská, Petržalka	299000	6.7	4	87	3	2
a12	Košická, Ružinov	232000	4	2	67	3	1
a13	Kopčianska, Petržalka	249900	6.3	2	81	4	2
a14	Hálova, Petržalka	256000	4.3	4	70	3	1
a15	Sputniková, Ružinov	234990	5.9	3	78	3	3
a16	Šancová, Staré Mesto	329000	2.6	2	90	2	3
	Typ kritéria	min	min	max	max	max	max

Obrázok 27 - Demonštračný dataset

Zdroj: Vlastné vypracovanie

4.1.1 Podmienky výberu

Vybrané nehnuteľnosti boli cenovo filtrované do 350 000 EUR, a všetky sa nachádzajú nie viac ako 15 kilometrov od centra mesta. Všetky nehnuteľnosti majú dodatočne aspoň 2 izby, a metráž veľkosti minimálne 60m^2. Čo sa týka stavu a parkovania, tu sa žiadne podmienky nekládli.

4.1.2 Kvantifikácia kvalitatívnych kritérií

Pre niektoré zo zvolených kritérií bolo taktiež potrebné ich kvantifikovať. Stav nehnuteľnosti, alebo parkovanie sú kvalitatívne vlastnosti. Ako je možné vidieť v tabuľke, aj pre tieto kritéria bola pridelená nejaká číselná hodnota, a logika tejto kvantifikácie je uvedená v nasledovných podkapitolách.

Stav nehnuteľnosti sme kvantifikovali podľa stavu alebo veku inzerovanej nehnuteľnosti a bodovali podľa nasledovnej logiky:

1	nehnuteľnosť je v pôvodnom stave a vyžaduje značnú dodatočnú investíciu
2	nehnuteľnosť prešla čiastočnou rekonštrukciou, a potrebná dodatočná investícia je minimálna
3	nehnuteľnosť nie je novostavbou, byt je však plne zrekonštruovaný
4	nehnuteľnosť je novostavba

Parkovaciu situáciu sme bodovali podľa toho, či je v rámci kúpy bytu poskytnuté aj parkovacie miesto, poprípade iné alternatívy ako by sa pri nehnuteľnosti parkovať dalo.

1	nehnuteľnosť v blízkosti ľahko dostupné a spoľahlivé parkovanie nemá
2	parkovanie je vzdialené, poprípade spoplatnené či verejné
3	nehnuteľnosť parkovacie miesto k dispozícií má

4.2 Zmysel datasetu v aplikácii

4.2.1 Testovanie

Počas vývoju aplikácie bolo nutné nové funkcionality neustále skúšať a testovať, čo by tento proces drasticky predĺžilo, keby v aplikácii nebol tento dataset dostupný. Pri implementácii každej fázy aplikácia lokálne bežala, a vďaka možnosti jedným klikom využiť tieto dáta nebolo nutné manuálne pri každej aktualizácií vypisovať nové len na to, aby bolo možné nájsť pri implementáciách nové chyby, alebo na to aby sa dalo uistiť že všetko funguje podľa potrieb.

4.2.2 Demonštrácia

Tento dataset zároveň poskytuje jednoduchý spôsob, ako novému používateľovi v rýchlosti ukázať a vysvetliť ako aplikácia funguje, a ako postup rozhodovacieho procesu funguje.

4.3 Výsledky rozhodovacieho procesu

V prvom rade je dôležité vyzdvihnúť integritu výsledkov, ktorá aplikácia poskytuje. Pre demonštráciu boli váhy definované bodovacou metódou, a to nasledovne:

Cena	5
Lokalita	3
Počet izieb	2
Metráž	2
Stav	1
Parkovanie	1

Obe metódy boli pre tie isté dáta s týmto bodovým ohodnotením implementované pre overenie funkčnosti aj manuálne mimo aplikácie, a výsledky metód v oboch prípadoch sú zhodné.

ID	Alternatíva	Vážený súčet	TOPSIS	Evalutron Skóre
a1	Martincekova, Ruzinov	64.194	56.886	**121.080**
a2	Nejedleho, Dubravka	61.960	42.776	**104.736**
a3	Furdekova, Petrzalka	61.482	53.179	**114.661**
a4	Stefunkova, Ruzinov	59.776	55.262	**115.038**
a5	Palisady, Stare Mesto	76.995	77.203	**154.198**
a6	Kadnarova, Raca	63.456	47.398	**110.854**
a7	Znievska, Petrzalka	71.661	53.130	**124.792**
a8	Furdekova, Petrzalka (2)	66.688	58.008	**124.696**
a9	Kopcianska, Petrzalka	59.338	51.947	**111.284**
a10	Ivana Bukovcana, Devinska Nova Ves	68.300	28.613	**96.913**
a11	Starohradska, Petrzalka	63.905	51.647	**115.552**
a12	Kosicka, Ruzinov	59.552	67.411	**126.964**
a13	Kopcianska, Petrzalka (2)	62.324	55.918	**118.242**
a14	Halova, Petrzalka	64.077	67.820	**131.897**
a15	Sputnikova, Ruzinov	67.956	61.734	**129.690**
a16	Sancova, Stare Mesto	59.492	63.900	**123.392**

ID	Názov	Vážený súčet	TOPSIS
a1	Martinčekova, Ružinov	64.19440933	56.88563317
a2	Nejedlého, Dúbravka	61.95993096	42.77617704
a3	Furdekova, Petržalka	61.48174406	53.17934235
a4	Štefunkova, Ružinov	59.77633478	55.2619883
a5	Palisády, Staré Mesto	76.9952743	77.20320975
a6	Kadnárova, Rača	63.45551378	47.39837997
a7	Znievska, Petržalka	71.66119232	53.13040599
a8	Furdekova, Petržalka	66.68756256	58.00837049
a9	Kopčianska, Petržalka	59.33750853	51.94662443
a10	Ivana Bukovčana, Devínska Nová Ves	68.30020661	28.61317526
a11	Starohradská, Petržalka	63.90493186	51.64737159
a12	Košická, Ružinov	59.55220307	67.41137009
a13	Kopčianska, Petržalka	62.32407249	55.91802517
a14	Hálova, Petržalka	64.07670323	67.82001282
a15	Sputniková, Ružinov	67.95572853	61.73395886
a16	Šancová, Staré Mesto	59.49221751	63.89958146

Obrázok 28 – Porovnanie výsledkov
Zdroj: Vlastné vypracovanie

Všetky kalkulačné funkcie fungujú tak ako majú, a správnosť výstupov je zaručená. Na obrázku možno vidieť porovnanie výstupov z aplikácie a z manuálnych výpočtov. V prílohe práce je poskytnutý súbor Microsoftu Excelu, kde sa všetky tieto výpočty nachádzajú.

Stĺpec „Evalutron Skóre“ vyjadruje súčet výsledných hodnôt z oboch metód, a má slúžiť len ako všeobecné hodnotenie alternatív, ktoré berie do úvahy obe metódy.

4.3.1 Metóda váženého súčtu

Výhodami tejto metódy je jednoduchosť a flexibilita. Je v rámci výpočtu jednoduchá na implementáciu, a po tom ako rozhodovateľ zadá pre kritéria váhy, skutočne stačí už len dáta maximalizovať a vytvoriť súčet hodnôt všetkých kritérií vynásobených s ich príslušnými váhami.

Avšak to, že metóda je jednoduchá na pochopenie nie je až takou výhodou pri aplikácií, ktorej zmysel je rozhodovateľa odľahčiť o tieto výpočty a potrebu kalkuláciám chápať. Napriek tomu metóda váženého súčtu poskytuje spoľahlivý a dostatočne smerodajný výstup pomôcť sa pri kúpe nehnuteľnosti rozhodnúť správne.

4.3.2 Metóda TOPSIS

Vzhľadom na schopnosť metódy TOPSIS lepšie riešiť konfliktné situácie zohľadnením ideálnych aj anti-ideálnych riešení a výpočtom relatívnej vzdialenosti alternatív k týmto riešeniam, táto metóda poskytuje viac dôkladné hodnotenie všetkých alternatív v porovnaní s metódou váženého súčtu.

4.3.3 Analýza výsledkov

Z výsledku je možno vidieť, že nehnuteľnosť na Palísadoch v Starom Meste bola rozhodnutá za najideálnejšiu oboma metódami. Medzi ostatnými alternatívami však existujú veľké rozdiely vo výsledkoch metód. Napríklad alternatíva nehnuteľnosti na ulici Ivana Bukovčana v Devínskej Novej Vsi bola ohodnotená relatívne výhodne metódou váženého súčtu, ale metóda TOPSIS ju vyhodnotila ako tú najhoršiu. Vzhľadom na to, že TOPSIS metóda sa nespolieha čisto na váhy alternatív, takéto rozdiely nie sú úplne nevídané.

5 Diskusia

5.1 Prínosy aplikácie

Hlavný prínos aplikácie je možnosť využiť metódy viackriteriálneho rozhodovania medzi alternatívami bez potreby akýchkoľvek počtov, nakoľko aplikácia tieto počty vykonáva na pozadí sama. Aplikácia môže slúžiť ako pomôcka sa pri kúpe nehnuteľnosti rozhodnúť.

5.1.1 Identifikácia silných a slabých stránok

Momentálne najväčšia slabina aplikácie je veľká konfliktnosť vo výsledkoch metód. Bežný používateľ nemusí celkom chápať, prečo môže byť jedna nehnuteľnosť hodnotená dobre jednou metódou a naopak veľmi slabo tou druhou. Výsledky môžu vyžadovať dodatočnú analýzu dát na to, aby bolo možné týmto konfliktom rozumieť, a spraviť si vlastný názor.

Napriek tomu aplikácia ostáva ako potenciálny nástroj pre získanie matematicky podložených výstupov pri rozhodovaní sa, a celý proces rozhodovania okrem definície kritérií, ich váh a samotných alternatív nevyžaduje vôbec nič iné. Je to jednoduchý spôsob ako môže aj bežný užívateľ získať dodatočný náhľad na situáciu z inej perspektívy.

5.2 Možnosti rozšírenia v budúcnosti

Počas vývoja aplikácie bolo veľa nápadov ako aplikáciu ďalej rozširovať. Najpotenciálnejším možným vylepšeniam, ktoré by bolo možné pre aplikáciu ďalej implementovať sa budú venovať nasledovné podkapitoly.

5.2.1 Implementácia dodatočných rozhodovacích metód

Výsledky dvoch metód sú síce smerodajné, ale aplikácia by mohla pomocou implementácie dodatočných rozhodovacích metód poskytovať oveľa detailnejšie a spoľahlivejšie výsledky.
Ak by aplikácia využívala viac metód, tak aj potenciál nejakého všeobecného hodnotenia, ktorý by mohol hodnoty zo všetkých metód normalizovať a priemerovať by bol oveľa vyšší.

Pri niektorých metódach sa však vyžadujú od rozhodovateľa dodatočné informácie, a tieto metódy v aplikácií zahrnuté k dnešnému dňu nie sú, nakoľko cieľ aplikácie bola práve aj jednoduchosť používania a práve limitácia potreby snahy od používateľa. Tieto metódy by

však mohli v aplikácií existovať ako voliteľné. V prípade záujmu, užívateľ by mal možnosť tieto dodatočné informácie doplniť a získať detailnejší výsledok, ale nebolo by to nutnosťou.

5.2.2 Importovanie a exportovanie dát

V prípade, že používateľ už má pripravené dáta o alternatívach a kritériách, podľa ktorých sa chce rozhodovať, veľmi prínosná funkcionalita by bola možnosť tieto dáta z formátov ako .csv alebo .xlsx jednoducho importovať, čo by celý proces skrátilo o značný čas vzhľadom na to, že by všetky údaje nebolo potrebné vypisovať manuálne.

Taktiež uchovanie výsledkov možnosťou exportovania si výslednej tabuľky by mohlo byť prínosné, v prípade že používateľ má v pláne sa k výsledkom z akéhokoľvek dôvodu neskôr vracať.

5.2.3 Web Scraping

V prípade kúpy nehnuteľnosti, aplikácia by mohla v budúcnosti poskytovať možnosť, kde užívateľ zadá žiadanú lokalitu, rozsah pre cenu a rozlohu nehnuteľnosti, a aplikácia by vykonala vyhľadávanie nehnuteľností spĺňajúce tieto potreby na jednom alebo viacerých inzerčných portáloch, a sama by z nich zostrojila daný dataset z aktuálneho trhu.

5.2.4 Rozšírenie na všeobecné využitie

Aplikácia ako súčasť tejto práce má plniť účel nástroja pri rozhodovaní sa pri kúpe nehnuteľnosti, avšak toto zďaleka nie je jediná sféra pre ktorú by sa mohla využívať. Aplikáciu je možné vykonávať pri akomkoľvek rozhodovaní, a pomôcť vyzdvihnúť univerzálne použitie by mohli predrobené šablóny pre kritéria, z ktorých by používateľ mal na výber. Pre nehnuteľnosti to môžu byť kritéria ako tie, ktoré boli zvolené v demonštračnom datasete, pri nákupe mobilného telefónu by to mohli byť kritéria ako veľkosť displeju, veľkosť batérie a podobne.

Poskytnutím takýchto šablón by bolo možné dodatočne proces skrátiť vyhnutím sa manuálnemu definovaniu kritérií.

5.2.5 Vizuálna responzivita

V súčasnom stave sa aplikácia zobrazuje správne len na väčších, počítačových obrazovkách, a je problematické ju na iných zariadeniach používať. Celá aplikácia by mohla

byť navrhnutá tak, že v prípade využitia na mobilnom zariadení s malou obrazovkou by sa rozhranie prispôsobilo pre taktiež pohodlné využitie.

Záver

Na dosiahnutie cieľa vývoju webovej aplikácie, ktorá má slúžiť ako potentný nástroj a asistencia pri dôležitom rozhodnutí ako je kúpa nehnuteľnosti bolo potrebné predovšetkým pochopiť ako pri tejto úlohe môžu byť práve metódy viackriteriálneho vyhodnocovania alternatív nápomocné.

Následne dôkladnou prípravou a zvolením si relevantných jazykov a nástrojov sa vyvíjala aplikácia ktorá hosťuje prehľadné užívateľské rozhranie pre používateľov. Aplikácia bola vyvinutá spôsobom, aby bola schopná stručne ale zreteľne prejsť cez celý proces rozhodovania. V rámci kódu to bolo možné vďaka komplexným objektovým štruktúram, ktoré sú spoločne využívané vo viacerých matematických funkciách, a zároveň umožňujú jednoduché zobrazovanie relevantných dát v užívateľskom rozhraní.

Aplikácia zaužíva metódy váženého súčtu a metódy TOPSIS, a pri vypracovaní výpočtov týchto metód aj manuálne bolo dokázané, že v aplikácií tieto výpočty sú vykonávané správne. Tieto metódy boli zvolené práve pre ich stručnosť, čo je v súlade s cieľom jednoduchosti používania aplikácie, a jej finálna verzia je plne-funkčný produkt s mnohými potenciálnymi vylepšeniami aj do budúcnosti.

Zoznam použitej literatúry

JANARTHANAN, N. – SCHNEIDER, J. Multicriteria evaluation of alternative transit system designs. Transportation Research Record, 1986, 1064, 26-34. [Citované dňa: 17.2.2024]. Dostupné na: https://onlinepubs.trb.org/Onlinepubs/trr/1986/1064/1064-004.pdf

MOOC. Best Programming Languages for Web Development [online]. MOOC, 2021. [Citované dňa: 21.2.2024]. Dostupné na: https://www.mooc.org/blog/best-programming-languages-for-web-development

MOYERS, Stephen. Common Web Development Languages, What They Do and Why You Need Them [online]. SPINX Digital. [Citované dňa: 21.2.2024]. Dostupné na: https://www.spinxdigital.com/blog/common-web-design-languages-what-they-do-and-why-you-need-them/

Ars Technica. Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem? [online]. Ars Technica, 2012. [Citované dňa: 21.2.2024]. Dostupné na: https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/

State of JS. State of JS: JavaScript Flavors [online]. State of JS, 2020. [Citované dňa: 21.2.2024]. Dostupné na: https://2020.stateofjs.com/en-US/technologies/javascript-flavors/

Stack Overflow. 2020 Developer Survey [online]. Stack Overflow, 2020. [Citované dňa: 21.2.2024]. Dostupné na: https://survey.stackoverflow.co/2020#most-loved-dreaded-and-wanted

DAVIDSON, Tim. A Comprehensive Guide to JavaScript Frontend Frameworks [online]. CleanCommit, 2023. [Citované dňa: 21.2.2024]. Dostupné na: https://cleancommit.io/blog/a-comprehensive-guide-to-javascript-frontend-frameworks/

MoldStud. The Importance of Version Control Systems in Software Development [online]. MoldStud, 2024. [Citované dňa: 22.2.2024]. Dostupné na: https://moldstud.com/articles/p-the-importance-of-version-control-systems-in-software-development

LEE, Kilsup – LEE, Sung Jong. A quantitative software quality evaluation model for the artifacts of component based development. Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network. IEEE, 2005. 20-25. [Citované dňa: 22.2.2024]. Dostupné na: https://ieeexplore.ieee.org/document/1434862/citations?tabFilter=papers

Visual Studio. Requirements for Visual Studio Code [online]. Visual Studio. [Citované dňa: 24.2.2024]. Dostupné na: https://code.visualstudio.com/docs/supporting/requirements

SINGH, Daljit. Front-End Frameworks: A Comparison Guide: Compare popular front-end frameworks like React, Angular, and Vue.js to help developers make informed choice [online]. LinkedIn, 2023. [Citované dňa: 1.3.2024]. Dostupné na: https://www.linkedin.com/pulse/front-end-frameworks-comparison-guide-compare-popular-daljit-singh/

ORNER, Daniel. Why SOLID principles are still the foundation for modern software architecture [online]. Stack Overflow, 2021. [Citované dňa: 10.3.2024]. Dostupné na: https://stackoverflow.blog/2021/11/01/why-solid-principles-are-still-the-foundation-for-modern-software-architecture/

GeeksforGeeks. Waterfall Model – Software Engineering [online]. GeeksforGeeks, 2024. [Citované dňa: 17.4.2024]. Dostupné na: https://www.geeksforgeeks.org/waterfall-model/

BRUSH, Kate – SILVERTHORNE, Valerie. Agile software development [online]. TechTarget, 2022. [Citované dňa: 17.4.2024]. Dostupné na: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development

FURKOVÁ, Andrea – IVANOVIČOVÁ, Zlatica. Viackriteriálne vyhodnocovanie alternatív. Vydavateľstvo EKONÓM, 2017. 140 s. ISBN 978-80-225-4363-7. [Citované dňa: 23.4.2024].

Príloha

Nasadená aplikácia je dostupná na: https://alex-nemeth.github.io/evalutron/

Zdrojový kód aplikácie je dostupný na: https://github.com/alex-nemeth/evalutron

Výpočty viackriteriálnych metód pre použité dáta dostupné na: https://eubask-my.sharepoint.com/:x:/g/personal/anemeth1_student_euba_sk/ESsUL7PZfe9BgubmHNXJkGABDjBf9k2gsWarg3QdqyvoAw?e=CBeaWh