Primjena stabla odlučivanja u obrazovanju

Dumičić, Nino

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 643

PRIMJENA STABLA ODLUČIVANJA U OBRAZOVANJU

Nino Dumičić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 643

PRIMJENA STABLA ODLUČIVANJA U OBRAZOVANJU

Nino Dumičić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 643

Pristupnik:	**Nino Dumičić (0069085361)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Predrag Pale

Zadatak: **Primjena stabla odlučivanja u obrazovanju**

Opis zadatka:

U procesu podučavanja posebno je zahtjevan proces formalizacije znanja stručnjaka. Stabla odlučivanja intuitivniji su oblik formalizacije te stoga posebno zanimljivi za proces podučavanja. Na sličan su način pogodna i u procesu učenja. Važno je proces izrade stabla odlučivanja učiniti što jednostavniji informatičkim laicima kao i onima koji nisu stručni za procese formalizacije znanja i podučavanja. U diplomskom radu treba istražiti načine primjene stabala odlučivanja u procesu učenja i podučavanja. Analizirati potrebe korisnika koji stvaraju stabla odlučivanja kao i onih koji ih koriste za učenje, te napraviti projektni zadatak za odgovarajuće alate. Istražiti postojeće alate za izradu stabala odlučivanja, s naglaskom na besplatne i one otvorenog programskog koda. Izraditi programsku potporu za pripremu i korištenje stabala odlučivanja. Prema mogućnosti koristiti već postojeće besplatne alate i programske komponente. Omogućiti izradu stabala odlučivanja iz repozitorija postojećeg znanja. Izmjeriti učinkovitost ponuđenog rješenja.

Rok za predaju rada: 28. lipnja 2024.

Zahvala

Duboku zahvalnost dugujem svom mentoru, profesoru Predragu, koji me je prihvatio pod svoje mentorstvo u završnom semestru pete godine. Njegovi savjeti i smjernice bili su neprocjenjivi za moj projekt i cjelokupni proces pisanja.

Posebno mjesto u ovoj zahvali zauzima Tihana, čije su strpljenje, savjeti i stvaranje ugodne radne atmosfere značajno olakšali proces pisanja. Uz nju, zahvaljujem i svojoj cimerici Anici na bezbroj šalica kave koje su me održavale budnim i fokusiranim.

Prijatelju Jacku dugujem zahvalnost za dijeljenje vlastitog iskustva u pisanju rada, kao i za humoristične anegdote koje su mi uljepšale dane.

Neizmjerno sam zahvalan svojoj obitelji na nepokolebljivoj vjeri u moj uspjeh i konstantnoj inspiraciji da ustrajem unatoč izazovima.

Naposljetku, zahvaljujem Bogu što mi je omogućio da uspješno završim diplomski studij bez većih prepreka.

Sažetak

Ovaj diplomski rad predstavlja razvoj i primjenu programa za parsiranje stabla odlučivanja, dizajnirane za pretvaranje složenih struktura odlučivanja u interaktivno web sučelje. Koristeći Next.js, TypeScript i Material-UI, program omogućuje korisnicima učitavanje stabla odlučivanja stvorenih u CmapTools-u i njihovu interaktivnu vizualizaciju.

Ključni doprinosi uključuju uspješno parsiranje XTM datoteka, intuitivno korisničko sučelje za navigaciju kroz stablo odlučivanja i responzivni dizajn. Rad analizira izazove u razvoju, posebno u očuvanju vizualnih elemenata i metapodataka iz CmapTools-a, te predlaže buduća poboljšanja.

Projekt doprinosi području sustava za podršku odlučivanju demonstrirajući praktičan pristup povećanju dostupnosti i upotrebljivosti stabla odlučivanja. Zaključno, ova primjena predstavlja značajan korak prema povećanju dostupnosti alata za stabla odlučivanja, s potencijalom za primjenu u obrazovanju.

Sadržaj

Popis oznaka i kratica

XTM	XML Topic Maps
XML	Extensible Markup Language
JSON	JavaScript Object Notation
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange

Popis isječaka koda

Popis slika

1. Uvod

U okruženju obrazovanja i donošenja odluka koje se brzo razvija, sposobnost učinkovite vizualizacije i interakcije sa složenim strukturama odlučivanja postala je sve važnija. Ovaj diplomski rad bavi se sjecištem obrazovne tehnologije i sustava za podršku odlučivanju, predstavljajući novi pristup vizualizaciji i interakciji stabla odlučivanja.

Područje obrazovanja stalno traži inovativne alate za poboljšanje iskustava učenja i vještina donošenja odluka. Nastavnici, studenti i stručnjaci u raznim disciplinama mogu imati koristi od poboljšanih metoda predstavljanja i upravljanja procesima donošenja odluka [1]. Važnost ovog rada leži u njegovom potencijalu da transformira apstraktne koncepte donošenja odluka u opipljiva, interaktivna iskustva, čime se poboljšava razumijevanje i zadržavanje složenih struktura odlučivanja.

Problem koji je pri ruci je dvostruk: prvo, postojeći alati za stablo odlučivanja često nemaju intuitivno sučelje i interaktivne značajke potrebne za učinkovitu obrazovnu upotrebu. Drugo, postoji nepovezanost između stvaranja stabala odlučivanja u poznatim alatima za preslikavanje koncepata i njihove primjene kao interaktivnih resursa za učenje. Ovaj diplomski rad predstavlja rješenje koje premošćuje ovaj jaz, nudeći internetski parser stabla odlučivanja koji transformira statične konceptualne mape u dinamične, obrazovne alate.

Rješavanjem ovih problema postižemo značajan napredak u načinu na koji se poučavaju i razumiju procesi donošenja odluka. Predloženo rješenje omogućuje nesmetan prijelaz od stvaranja koncepta do interaktivnog učenja, osnažujući edukatore da osmisle privlačnija iskustva učenja i omogućujući studentima da na intuitivan način istraže puteve složenih odluka.

Nedostatak pristupačnih i interaktivnih alata za donošenje odluka u obrazovanju može imati značajne posljedice. Istraživanja su pokazala da interaktivni alati za učenje mogu poboljšati angažman učenika i razumijevanje složenih koncepata [2]. Štoviše, korištenje statičkih prikaza za

poučavanje složenih procesa pokazalo se manje učinkovitim od dinamičkih, interaktivnih alternativa [2]. Stoga kontinuirano oslanjanje na neinteraktivne prikaze stabla odlučivanja može spriječiti razvoj kritičkog mišljenja i vještina donošenja odluka u obrazovnim okruženjima.

Ovaj se diplomski rad razlikuje od sličnih radova posebnim fokusom na obrazovnu primjenu stabala odlučivanja, dajući prednost korisničkom iskustvu i interaktivnosti nad statističkom analizom. Predstavlja jedinstveni pristup koji iskorištava postojeće alate za preslikavanje koncepta dok uvodi novi sloj interaktivnosti temeljene na webu, kombinirajući tako poznavanje s inovacijom.

U sljedećim poglavljima istražit ćemo razvoj ovog parsera stabla odlučivanja, od njegovih konceptualnih temelja do tehničke primjene i potencijalnih budućih poboljšanja. Ovaj rad ima za cilj značajno doprinijeti područjima obrazovne tehnologije i sustava podrške odlučivanju, nudeći praktično rješenje za poboljšanje poučavanja i učenja procesa donošenja odluka.

2. Analiza korisničkih potreba i projektni zahtjevi

2.1. Analiza potreba korisnika: stvaraoci stabla odlučivanja

Stvaraoci stabla odlučivanja, kao što su učitelji i profesori, igraju ključnu ulogu u razvoju i primjeni stabla odlučivanja u obrazovne svrhe. Kako bi se učinkovito zadovoljile njihove potrebe, bitno je razumjeti njihove zahtjeve i izazove pri izradi stabla odlučivanja.

Jedna od primarnih potreba stvaraoca stabla odlučivanja je korisničko sučelje koje im omogućuje jednostavno strukturiranje i organiziranje znanja iz domene u format stabla odlučivanja. Alat bi trebao pružiti intuitivne mehanizme za definiranje čvorova odlučivanja, grana i ishoda, omogućujući stvarateljima da se usredotoče na sadržaj umjesto da se bore s tehničkim složenostima [1].

Nadalje, autori stabla odlučivanja trebaju pomoć u osiguravanju logičke dosljednosti i potpunosti svojih stabala odlučivanja. Programsko rješenje trebalo bi nuditi funkcionalnosti provjere valjanosti i provjere pogrešaka kako bi se identificirali potencijalni problemi, kao što su grane koje nedostaju ili nekonzistentni rezultati, te pružiti smjernice za njihovo rješavanje [1].

2.2. Analiza potreba korisnika: Polaznici stabla odlučivanja

Učenici su primarni korisnici stabala odlučivanja u obrazovnom kontekstu. Kako bi se povećala učinkovitost stabala odlučivanja kao alata za učenje, ključno je razumjeti i odgovoriti na potrebe učenika koji s njima stupaju u interakciju.

Učenicima je potrebno korisničko sučelje koje im omogućuje jednostavnu navigaciju i istraživanje stabala odlučivanja. Programsko rješenje trebalo bi pružiti vizualno privlačan i intuitivan prikaz strukture stabla odlučivanja, omogućujući učenicima da prate proces donošenja odluka korak po korak [1].

Interaktivnost je još jedan ključni zahtjev za učenike. Alat bi trebao omogućiti učenicima da se aktivno uključe u stablo odlučivanja odabirom različitih putova odlučivanja, promatranjem rezultirajućih ishoda i primanjem trenutne povratne informacije. Ovo interaktivno iskustvo poboljšava razumijevanje i pamćenje predmeta kod učenika [1].

2.3. Zahtjevi projekta

Na temelju analize potreba korisnika i stvarne primjene, identificirani su sljedeći projektni zahtjevi za alat za stablo odlučivanja u obrazovne svrhe:

1. Sučelje prilagođeno korisniku: Program pruža intuitivno i vizualno privlačno sučelje za autore stabala odlučivanja i korisnike. Primjenjeno je responzivno web sučelje koristeći Next.js i Material-UI komponente, osiguravajući jednostavno korištenje na različitim uređajima [9] [11].
2. Parsiranje XTM datoteka: Alat uspješno parsira XTM 1.0 datoteke izvezene iz CmapTools-a, omogućujući korisnicima da svoje postojeće mape koncepata pretvore u interaktivna stabla odlučivanja.
3. Interaktivno iskustvo učenja: Primjenjeno je interaktivno sučelje koje omogućuje korisnicima navigaciju kroz stablo odlučivanja. Korisnici mogu odabirati različite putove odlučivanja, vraćati se na prethodne odluke i primati trenutne povratne informacije o svojim izborima.
4. Prikaz resursa: Program podržava prikaz povezanih slika i tekstualnih informacija za svaki čvor stabla odlučivanja, obogaćujući iskustvo učenja dodatnim kontekstom.
5. Upravljanje stablima odlučivanja: Primjenjene su funkcionalnosti za učitavanje, prikaz liste i brisanje stabala odlučivanja, omogućujući korisnicima učinkovito upravljanje svojim resursima.
6. Vizualizacija strukture stabla: Alat pruža mogućnost pregleda cjelokupne strukture stabla odlučivanja, pomažući korisnicima u razumijevanju cjelokupnog konteksta odlučivanja.

7. Responzivni dizajn: Program je dizajniran da bude funkcionalan i vizualno privlačan na različitim veličinama zaslona, od radne površine računala do mobilnih uređaja.

Ovi primjenjeni zahtjevi služe kao temelj programa za parsiranje stabla odlučivanja, pružajući učinkovito rješenje za pretvaranje CmapTools mapa u interaktivne strukture odlučivanja.

3. Pregled postojećih alata

Kako bi trenutni krajolik alata za izradu stabla odlučivanja bio bolje razumljen i potencijalna rješenja u obrazovne svrhe bila identificirana, proveden je pregled postojećih alata i programskih rješenja. Ova analiza prvenstveno se usredotočila na besplatne mogućnosti i izbore otvorenog koda, budući da nude veću pristupačnost i prilagodljivost za integraciju u obrazovne postavke.

3.1. Pregled postojećih alata stabla odlučivanja

Identificirano je i ocijenjeno nekoliko alata stabla odlučivanja na temelju njihovih značajki, upotrebljivosti i prikladnosti za obrazovni kontekst. Sljedeći alati bili su među pregledanim:

1. CmapTools: programski alat za izradu konceptualnih mapa i modela znanja. Iako nije specifično za stabla odlučivanja, naširoko se koristi u obrazovanju za vizualizaciju struktura znanja [3].
2. Lucidchart: mrežna aplikacija za izradu dijagrama koja podržava različite vrste grafikona, uključujući stabla odlučivanja. Nudi značajke suradnje i pristup temeljen na webu [4].
3. TreePlan: Excelov dodatak posebno dizajniran za stvaranje stabla odlučivanja. Iako se ne temelji na webu, relevantan je zbog svog fokusa na stvaranje stabla odlučivanja [5].
4. Decision Tree Resolver: Decision Tree Resolver je alat za e-učenje koji se bavi stablima odluka proizveden na FER-u. Nudi web stranicu na kojoj se mogu učitati stabla iz programa CmapTools, no fali napredne funkcionalnosti poput dodavanja resursa na čvorove i spremanja stabala [6].

Ovi su alati ocijenjeni na temelju korisničkog sučelja, jednostavnosti korištenja, značajki suradnje, interaktivnosti i mogućnosti integracije.

3.2. Procjena postojećih alata

Evaluacija postojećih alata otkrila je nekoliko prednosti i slabosti u kontekstu stvaranja i korištenja obrazovnog stabla odlučivanja.

Snage:

- Intuitivna vizualna sučelja: Alati kao što su CmapTools i Lucidchart nude intuitivna sučelja za stvaranje vizualnih prikaza znanja, čineći ih dostupnima edukatorima bez opsežne tehničke stručnosti [3][4].
- Značajke suradnje: Neki od ovih alata, kao što je Lucidchart, pružaju istovremenu suradnju, omogućujući više korisnika da rade na istom dijagramu istovremeno [4].
- Različite vrste grafikona: Alat poput Lucidcharta podržava više vrsta grafikona, uključujući osnovna stabla odlučivanja, nudeći prilagodljivost u predstavljanju znanja [4].
- Interaktivna navigacija: Decision tree resolver nudi interaktivnu navigaciju kroz stabla odlučivanja, što je dragocjeno za obrazovne programe.

Slabosti:

- Ograničena specifičnost stabla odlučivanja: Iako ovi alati mogu stvoriti strukture poput stabla, većina nije posebno dizajnirana za obrazovna stabla odlučivanja, nedostaju im značajke prilagođene ovom slučaju upotrebe.
- Ograničena prilagodba: Mnogi alati nude ograničene mogućnosti prilagodbe za sadržaj čvora, što otežava uključivanje sveobuhvatnog obrazovnog materijala, poput slika i teksta, u svakoj točki odlučivanja.
- Ograničena integracija podataka: Često postoji nedostatak mogućnosti za jednostavnu integraciju vanjskih podataka ili resursa u stabla odlučivanja.
- Ograničenja izvoza: Mnogi alati imaju ograničene mogućnosti za izvoz stabala odluka u formatima koji se mogu lako koristiti u

drugim obrazovnim programskim rješenjima ili sustavima za upravljanje učenjem.

- Nedostatak interakcije: Većina alata ne nudi značajke za interakciju sa stablom već samo njeno stvaranje.

Dok postojeći alati pružaju temelj za stvaranje stabla odlučivanja, oni ne ispunjavaju u potpunosti specifične zahtjeve utvrđene za alat za obrazovno stablo odlučivanja. Evaluacija naglašava potrebu za više prilagođenim rješenjem koje se bavi jedinstvenim potrebama stvarateljima stabla odlučivanja i učenika u obrazovnim okruženjima.

4. Zahtjevi projekta

4.1. Osnovne značajke

1. Integracija s CmapTools:
    - Podrška za uvoz stabala odluka stvorenih u CmapTools i izvezenih u XTM 1.0 formatu.
    - Sposobnost povezivanja slika i tekstualnih izvora (.txt, .html) s pojedinačnim čvorovima.
2. Interaktivna vizualizacija stabla:
    - Dinamičko prikazivanje stabala odluka na web sučelju.
    - Prikaz povezanih slika i teksta za svaki čvor za usmjeravanje korisnika pri donošenju odluka.
    - Primjena komponente za prikaz putanje korisnika kroz stablo.
3. Upravljanje stablom:
    - Funkcionalnost za korisnike za učitavanje, pristup i brisanje stabala odluka.
    - Sustav za pohranu i pronalaženje učitanih stabala.
4. Reprezentacija stabla:
    - Podrška za učitavanje i prikazivanje slike stabla odluke za vizualni prikaz.
    - Generiranje i prikaz ASCII tekstualnog prikaza strukture stabla.
5. Značajke navigacije:
    - Primjena funkcije koja omogućuje korisnicima povratak na prethodne grane.
    - Jasna vizualizacija trenutne pozicije u stablu odlučivanja.

4.2. Tijeka rada

1. Izrada stabla i prijenos:

- Izrada stabala odlučivanja u CmapTools s pridruživanjem resursa. Izvoz stabala u XTM 1.0 formatu.
- Dodatak description.txt datoteke za metapodatke stabla.
- Izrada slike *tree_graph* za vizualni prikaz.
- Kompresija svih komponenti u zip datoteku.
- Funkcionalnost učitavanja na web platformi.

2. Raščlanjivanje stabla i renderiranje:
    - Pozadinsko analiziranje prenesene XTM datoteke u strukturu prilagođenu za prikaz na sučelju
    - Dinamičko prikazivanje raščlanjenog stabla na sučelju.
3. Interakcija stabla:
    - Navigacija korak po korak kroz stablo odlučivanja.
    - Prikaz pridruženih resursa (slike i tekst) na svakom čvoru.
    - Sposobnost vraćanja i istraživanja različitih putova donošenja odluka.

4.3. Pregled korisničkog sučelja

1. Intuitivna navigacija:
    - Jasni vizualni znakovi za odluke i mogućnosti.
    - Lako dostupan gumb "natrag" za ponovno praćenje koraka.
    - Istaknuti prikaz komponente koja prikazuje trenutni put.
2. Prezentacija resursa:
    - Uravnoteženi izgled za prikaz teksta čvora, povezanih slika i mogućnosti odlučivanja.
    - Skalabilni prikaz slike za prilagodbu različitim veličinama resursa.
3. Sučelje za upravljanje stablom:
    - Lak popis učitanih stabala s mogućnostima za pristup i brisanje.
    - Jednostavno sučelje za učitavanje s jasnim uputama

4.4. Potencijalna poboljšanja

Iako trenutno nisu primjenjene, slijedeće značajke mogle bi značajno povećati obrazovnu vrijednost alata:

1. Analitika učenja:
    - Praćenje korisničkih interakcija i putova donošenja odluka.
    - Generiranje izvješća o zajedničkim točkama odlučivanja i ishodima.
2. Značajke suradnje:
    - Mogućnost da više korisnika radi na jednom stablu istovremeno.
    - Sustav komentiranja i povratnih informacija za reviziju stabala odlučivanja.
3. Napredne mogućnosti vizualizacije:
    - Primjena različitih vizualnih izgleda za stabla odlučivanja kako bi se zadovoljili različiti stilovi učenja.
    - Uključivanje interaktivnih elemenata u čvorovima, kao što su skočna objašnjenja ili mini-kvizovi
4. Mobilna optimizacija:
    - Stvaranje responzivnog dizajna ili namjenskog mobilnog riješenja za učenje u pokretu.
    - Primjena sučelja prilagođenih dodiru za lakšu navigaciju na mobilnim uređajima.
5. Alati za izradu stabala:
    - Razvijanje uređivača u programu za stvaranje i izmjenjivanje stabala odluka izravno u web sučelju.

Ispunjavanjem ovih zahtjeva i primjenom ovih poboljšanja, ovaj alat za analizu stabla odlučivanja ima za cilj pružiti intuitivnu, prilagodljivu i obrazovno vrijednu platformu za stvaranje i interakciju sa stablima odlučivanja u različitim kontekstima učenja. Ova bi poboljšanja riješila trenutna ograničenja alata i proširila njegovu korisnost u obrazovnim okruženjima.

5. CmapTools integracija i XTM parsiranje

Integracija CmapTools-a sa sustavom parsera stabla odlučivanja predstavlja ključni most između vizualnog predstavljanja znanja i interaktivne podrške odlučivanju temeljene na webu. Ovo se poglavlje bavi procesom transformacije konceptualnih mapa stvorenih u CmapTools-u u interaktivna stabla odlučivanja putem formata XML Topic Maps (XTM).

CmapTools, kojeg je razvio institut za ljudsku i strojnu kogniciju u Floridi, moćan je alat za izradu konceptualnih mapa [3]. Njegova mogućnost izvoza stabala u formatu XTM 1.0 daje standardizirani način za stvaranje i prijenos složenih struktura znanja. Projekt koristi ovu mogućnost kako bi omogućio vizualnu izradu stabala odluka i zatim ih transformirao u interaktivne web alate.

Format XTM, budući da predstavlja prikaz tematskih mapa temeljen na XML-u, nudi bogatu strukturu za predstavljanje znanja. Omogućuje definiranje tema, povezanosti između tema i pojava (resursi relevantni za teme). Ova se struktura dobro usklađuje s komponentama stabla odlučivanja: čvorovima, odnosima između čvorova i povezanim informacijama ili resursima [7].

5.1. Struktura XTM datoteka

Specifikacija XTM 1.0 definira nekoliko ključnih elemenata koji se koriste u prikazu stabla odlučivanja:

- <topic>: Predstavlja subjekt, koji u ovom slučaju odgovara čvoru u stablu odlučivanja.
- <association>: Definira odnose između tema, koje se koriste za strukturiranje stabla odlučivanja.
- <occurrence>: Povezuje teme s relevantnim resursima koje se koriste za dodatne informacije na svakom čvoru.
- <baseName>: Daje čitljiv naziv za temu, koji postaje oznaka čvorova stabla odlučivanja. [7]

Kako su elementi strukturirani u XTM datoteci i kako se preslikavaju na model stabla odlučivanja se mogu vidjeti na isječku koda 1.

```
<topicMap>
  <topic id="decision1">
    <baseName>
      <baseNameString>Initial Decision</baseNameString>
    </baseName>
    <occurrence>
      <resourceRef xlink:href="decision1_info.txt"/>
    </occurrence>
  </topic>
  <topic id="option1">
    <baseName>
      <baseNameString>Option 1</baseNameString>
    </baseName>
  </topic>
  <association>
    <member>
      <roleSpec><topicRef xlink:href="#parent"/></roleSpec>
      <topicRef xlink:href="#decision1"/>
    </member>
    <member>
      <roleSpec><topicRef xlink:href="#child"/></roleSpec>
      <topicRef xlink:href="#option1"/>
    </member>
  </association>
</topicMap>
```

Isječak koda 1, primjer XTM datoteke

U ovom primjeru:

- Elementi <topic> s ID-ovima "decision1" i "option1" predstavljaju čvorove u stablu odlučivanja.
- Elementi <baseName> daju oznake za ove čvorove.
- Element <occurrence> u "decision1" povezuje s dodatnim informacijama za taj čvor kao što su slike i tekst.
- Element <association> definira odnos između "decision1" i "option1", strukturirajući stablo.

U sustavu se preslikavaju ovi XTM elementi u razred EntityStore, a to se vidi na isječku koda 2.

```
class Entity:
    def __init__(self, id, label, image="", description="", parent=None):
        self.id = id
        self.label = label
        self.image = image
        self.description = description
        self.parent = parent

class Association:
    def __init__(self, id, from_id, to_id, label):
        self.id = id
        self.from_id = from_id
        self.to_id = to_id
        self.label = label

class EntityStore:
    def __init__(self):
        self.entities = {}
        self.associations = {}
        self.decision_tree = {}
```

Isječak koda 2, razredi na poslužitelju

Razred *Entity* predstavlja čvor u stablu odlučivanja, koji odgovara XTM <topic> elementu. Razred *Association* predstavlja odnose između čvorova, odgovarajući XTM <association> elementu. Razred EntityStore služi kao središnja struktura podataka za pohranu i upravljanje tim entitetima i asocijacijama.

5.2. Izvoz iz CmapTools i priprema XTM datoteka

CmapTools pruža jednostavnu metodu za izvoz mapa odlučivanja u format XTM 1.0 [3]. Međutim, kako bi se u potpunosti iskoristile mogućnosti napravljenog parsera stabla odlučivanja, od korisnika se zahtijeva da slijede određene smjernice prilikom izrade svojih konceptualnih mapa:

1. Konceptna mapa trebala bi biti strukturirana kao stablo, s jasnim korijenskim čvorom.
2. Svaki čvor može imati pridružene resurse: sliku koja smije biti .jpg, .jpeg, .png ili .svg i tekstualnu datoteku koja može biti .html, .htm ili .txt.

3. Strelice koje povezuju pojmove koriste se za označavanje odgovora na određeni čvor u procesu odlučivanja.

Prilikom izvoza iz CmapTools, korisnici moraju:

1. Odabrati mogućnost "File" > "Export Cmap as" > "XTM/XCM"
2. Prilikom samog spremanja otvoriti „Files of type“ i umjesto XCM 3.0, odabrati XTM 1.0
3. Uključiti sve povezane resurse u zip datoteku zajedno s XTM datotekom
4. Osigurati da zip datoteka slijedi ovu strukturu:
   - root.xml (glavna XTM datoteka, ime može biti bilo koje)
   - images/ (mapa koja sadrži sve resurse slika)
   - texts/ (mapa koja sadrži sve tekstualne resurse)
   - description.txt (neobavezna datoteka koja opisuje stablo odlučivanja)
   - tree_graph (neobavezna slika u bilo kojem slikovnom format koja prikazuje stablo)

5.3. Parsiranje XTM datoteka

Srž XTM logike parsiranja primjenjena je u razredu *EntityStore*. Ključne metoda za raščlanjivanje XTM datoteke je prikazana isječku koda 3.

Metoda *parse_xtm_file* je ulazna točka za analiziranje XTM datoteke:

- Koristi *lxml*-ov *XMLParser* s *remove_blank_text=True* za ignoriranje tekstualnih čvorova koji sadrže samo razmak, što može pojednostaviti raščlanjivanje.
- Stvara mapu prostora imena (nsmap) za ispravno rukovanje prostorima imena XML.
- Koristi *functools.partial* za stvaranje funkcija prečaca *find* i *findall* koje automatski koriste ispravne prostore imena. Kako bi kasnije bilo lakše prolaziti po elementima datoteke.

```
1. class EntityStore:
2.     XTM_NS = "http://www.topicmaps.org/xtm/1.0/"
3.     XLINK_NS = "http://www.w3.org/1999/xlink"
4.
5.     def __init__(self):
6.         self.entities = {}
7.         self.links = {}
8.         self.associations = {}
9.         self.decision_tree = {}
10.
11.     def parse_xtm_file(self, file_path):
12.         parser = etree.XMLParser(remove_blank_text=True)
13.         tree = etree.parse(file_path, parser)
14.         root = tree.getroot()
15.
16.         nsmap = {'xtm': self.XTM_NS, 'xlink': self.XLINK_NS}
17.         find = partial(root.find, namespaces=nsmap)
18.         findall = partial(root.findall, namespaces=nsmap)
19.
20.         for topic in findall('.//xtm:topic'):
21.             self._parse_topic(topic, nsmap)
22.
23.         for association in findall('.//xtm:association'):
24.             self._parse_association(association, nsmap)
```

Isječak koda 3, početak proces parsiranja

Metoda *parse_topic* sa isječka koda 4 analizira pojedinačne elemente teme:

- Izdvaja ID teme i utvrđuje radi li se o uobičajenoj temi ili povezujućoj frazi (koja predstavlja veze između tema).
- Pronalazi osnovni naziv teme.
- Za uobičajene teme, obrađuje sve pojave (povezane resurse) pozivanjem *_parse_occurrence*.
- Zatim stvara objekt *Link* (za povezivanje izraza) ili objekt *Entity* (za uobičajene teme) i pohranjuje ga u odgovarajući rječnik.

```
1. def _parse_topic(self, topic, nsmap):
 2.     topic_id = topic.get("id")
 3.     instance_of = topic.find('.//xtm:instanceOf', namespaces=nsmap)
 4.     topic_type = 'linkingPhrase' if instance_of is not None and \
 5.         instance_of.find('.//xtm:subjectIndicatorRef',
namespaces=nsmap).get(f'{{{self.XLINK_NS}}}href').endswith('#linking
Phrase') \
 6.         else 'topic'
 7.
 8.     base_name = topic.findtext('.//xtm:baseNameString',
namespaces=nsmap)
 9.     image_name, description = "", ""
10.
11.     if topic_type == 'linkingPhrase':
12.         self.links[topic_id] = Link(topic_id, base_name, image_name,
description)
13.     else:
14.         for occurrence in topic.findall('.//xtm:occurrence',
namespaces=nsmap):
15.             resource_ref = occurrence.find('.//xtm:resourceRef',
namespaces=nsmap)
16.             if resource_ref is not None:
17.                 file_path = resource_ref.get(f'{{{self.XLINK_NS}}}href')
18.                 if file_path.startswith('file'):
19.                     image_name, description =
self._parse_occurrence(file_path)
20.
21.         self.entities[topic_id] = Entity(topic_id, base_name,
image_name, description
```

Isječak koda 4, parsiranje tema XTM datoteke

Metoda *_parse_occurrence* s isječka koda 5 obrađuje pojavljivanja resursa:

- Određuje vrstu izvora na temelju nastavka datoteke.
- Za slikovne datoteke, konstruira odgovarajući put datoteke.
- Za tekstualne datoteke, čita sadržaj datoteke.
- CmapTools pretvori .html datoteku u .htm datoteku stoga sustav provjerava ako korisnik je predao htm ili html datoteku kako bi nastavio raditi nesmetano u oba slučaja
- Vraća i naziv i opis slike

```
1. def _parse_occurrence(self, file_path, image_name, description):
 2.     _, file_extension = os.path.splitext(file_path)
 3.     folder_name, file_name = file_path.split('/./')[1].split('/')
 4.     if file_extension.lower() in ('.png', '.jpg', '.jpeg', '.svg'):
 5.         image_name = os.path.join(folder_name, 'images', file_name)
 6.     elif file_extension.lower() in ('.txt'):
 7.         with open(os.path.join(folder_name, 'texts', file_name), 'r') as
file:
 8.             description = file.read()
 9.     elif file_extension.lower() in ('.htm', '.html'):
10.         file_name_folder = os.path.join(folder_name, 'texts',
file_name)
11.         if os.path.isfile(file_name_folder):
12.             with open(os.path.join(folder_name, 'texts', file_name), 'r')
as file:
13.                 description = file.read()
14.         else:
15.             with open(os.path.join(folder_name, 'texts', file_name + 'l'),
'r') as file:
16.                 description = file.read()
17.     return image_name, description
```

Isječak koda 5, parsiranje resursa XTM datoteke

Metoda *_parse_association* s isječka koda 6 analizira elemente pridruživanja:

- Izdvaja ID asocijacije (link_id).
- Pronalazi ID-ove dviju tema povezanih ovom asocijacijom (from_id i to_id).
- Stvara objekt asocijacije i pohranjuje ga u rječnik asocijacija.

```
1. def _parse_association(self, association, nsmap):
2.     link_id = association.find('.//xtm:instanceOf/xtm:topicRef',
namespaces=nsmap).get(f'{{{self.XLINK_NS}}}href').split('#')[-1]
3.     members = association.findall('.//xtm:member/xtm:topicRef',
namespaces=nsmap)
4.     from_id = members[0].get(f'{{{self.XLINK_NS}}}href').split('#')[-1]
5.     to_id = members[1].get(f'{{{self.XLINK_NS}}}href').split('#')[-1]
6.
7.     self.associations[link_id] = Association(link_id, from_id, to_id,
self.links[link_id].label)
```

Isječak koda 6, parsiranje veza između čvorova

Ove funkcije rade zajedno kako bi analizirale XTM datoteku, izdvojile sve potrebne informacije i izgradile robusnu internu reprezentaciju stabla

odlučivanja. Poboljšana struktura omogućuje složenije odnose između tema uz održavanje učinkovitosti i jasnoće u procesu parsiranja.

5.4. Izazovi i rješenja

Jedan značajan izazov tijekom primjene bio je rukovanje različitim vrstama resursa (slike, tekstualne datoteke, HTML) i osiguravanje njihove točne reprezentacije u raščlanjenoj strukturi podataka. To je riješeno primjenom prilagodljive metode _parse_occurrence koja može rukovati različitim vrstama resursa i pohraniti ih na odgovarajući način.

Još jedan izazov bila je učinkovita konstrukcija strukture stabla odlučivanja iz raščlanjenih XTM podataka. To je riješeno primjenom pristupa s dva prolaza: prvo analiziranje svih tema i asocijacija, zatim izgradnja strukture stabla u zasebnom koraku.

6. Pristupne točke API-ja

Poslužitelj sustava parsera stabla odluka izlaže nekoliko pristupnih točaka API-ja koje olakšavaju interakciju između sučelja i analiziranih XTM podataka. Ove pristupne točke rukuju različitim operacijama kao što su inicijalizacija stabala, dohvaćanje informacija o čvoru, upravljanje pohranom stabla i pružanje vizualnih prikaza strukture stabla. U ovom poglavlju će se detaljno istražiti svaka od pristupnih točaka, raspravljajući o njihovoj funkcionalnosti, parametrima i strukturama odgovora.

6.1. Pristupna točka: GET /api/tree

Pristupna točka *GET /api/tree* odgovorna je za pokretanje stabla odlučivanja iz raščlanjene XTM datoteke. Poziva se započinjanjem interakcije sa stablom i vidi se na isječku koda 7.

```
1. @app.route('/api/tree', methods=["GET"])
2. def start_tree():
3.     try:
4.         folder = request.args.get("name")
5.         if not folder:
6.             raise BadRequest("Missing 'name' parameter")
7.
8.         folder = folder.strip()
9.         file = f"{folder}.xml"
10.
11.         if not os.path.exists(os.path.join(folder, file)):
12.             raise NotFound(f"Tree file '{file}' not found in folder '{folder}'")
13.
14.         entity_store.clear_tree()
15.         entity_store.parse_xtm_file(os.path.join(folder, file))
16.         entity_store.build_decision_tree()
17.
18.         root_node = entity_store.find_root_node()
19.         if not root_node:
20.             raise NotFound("No root node found in the parsed tree")
21.
22.         return jsonify(root_node)
23.     except Exception as e:
24.         app.logger.error(f"Error in start_tree: {str(e)}")
25.         raise InternalServerError("An unexpected error occurred")
```

Isječak koda 7, pristupna točka za dohvaćanje stabla

Ona prihvaća parametar „name“, koji je naziv mape koja sadrži XTM datoteku i resurse. Izvodi slijedeće operacije koje se vide na isječku:

1. Provjerava valjanost unosa, osiguravajući da je parametar „name“ naveden i da odgovarajuća XTM datoteka postoji.
2. Briše sve postojeće podatke stabla iz *EntityStore* razreda.
3. Parsira XTM datoteku i gradi strukturu stabla odlučivanja.
4. Pronalazi korijenski čvor raščlanjenog stabla.
5. Vraća korijenski čvor kao *JSON* odgovor.

Pristupna točka uključuje rukovanje pogreškama za različite scenarije, kao što su parametri koji nedostaju, nepostojeće datoteke i pogreške u analizi. Ovo osigurava da sučelje prima smislene poruke o pogrešci u slučaju problema.

6.2. Pristupna točka: GET /api/get_children

Pristupna točka *GET /api/get_children* dohvaća podređene čvorove određenog čvora u stablu odlučivanja. Presudna je za navigaciju kroz strukturu stabla na sučelju, prikazana je na isječku koda 8.

```
1. @app.route("/api/get_children", methods=["GET"])
2. def get_children():
3.     try:
4.         node_id = request.args.get("node")
5.         if not node_id:
6.             raise BadRequest("Missing 'node' parameter")
7.
8.         node = create_node(node_id)
9.         if not node:
10.            raise NotFound(f"Node with id '{node_id}' not found")
11.
12.        return jsonify(node)
13.    except Exception as e:
14.        app.logger.error(f"Error in get_children: {str(e)}")
15.        raise InternalServerError("An unexpected error occurred")
```

Isječak koda 8, pristupna točka za dohvaćanje djece čvora

Ova **pristupna** točka:

- Prihvaća parametar „node“ koji specificira ID čvora čija se djeca traže.
- Koristi pomoćnu funkciju create_node za konstruiranje odgovora koji uključuje sam čvor i njegovu djecu.
- Vraća JSON objekt koji sadrži podatke o čvoru i popis njegovih potomaka.

Funkcija *create_node* opisana je u kasnijem poglavlju.

6.3. Pristupna točka: GET /api/images

Pristupna točka *GET /api/images* poslužuje slikovne datoteke povezane s čvorovima u stablu odlučivanja. Ona je prikazana na isječku koda 9.

```
1. @app.route('/api/images', methods=["GET"])
2. def get_image():
3.     try:
4.         filename = request.args.get("image")
5.         if not filename:
6.             raise BadRequest("Missing 'image' parameter")
7.
8.         if not os.path.exists(filename):
9.             raise NotFound(f"Image '{filename}' not found")
10.
11.         return send_file(filename)
12.     except Exception as e:
13.         app.logger.error(f"Error in get_image: {str(e)}")
14.         raise InternalServerError("An unexpected error occurred")
```

Isječak koda 9, pristupna točka za dohvaćanje slike

Tijek rada pristupne točke s isječka koda je sljedeći:

1. Prihvaća parametar „image“ koji navodi naziv datoteke tražene slike.
2. Potvrđuje da datoteka postoji.
3. Koristi Flaskovu funkciju *send_file* za posluživanje slikovne datoteke izravno klijentu. [9]

Ovo omogućuje sučelju da prikazuje slike povezane s čvorovima stabla odlučivanja, poboljšavajući vizualni prikaz stabla, i osigurava da se ne šalju sve slike stabla, nego samo one koje će se zaista prikazati.

6.4. Pristupna točka: GET /api/trees

Pristupna točka *GET /api/trees* daje popis svih dostupnih stabala odlučivanja, zajedno s njihovim opisima.

```
1. @app.route('/api/trees', methods=["GET"])
2. def get_trees():
3.     try:
4.         folder_objects = []
5.         for item in os.listdir():
6.             if os.path.isdir(item) and item not in FOLDER_IGNORE_LIST:
7.                 description = 'No description available'
8.                 description_file_path = os.path.join(item, 'description.txt')
9.                 if os.path.isfile(description_file_path):
10.                     with open(description_file_path, 'r') as file:
11.                         description = file.read().strip()
12.                 folder_objects.append({'name': item, 'description': description})
13.         return jsonify(folder_objects)
14.     except Exception as e:
15.         app.logger.error(f"Error in get_trees: {str(e)}")
16.         raise InternalServerError("An unexpected error occurred")
```

Isječak koda 10, pristupna točka za dohvaćanje liste stabala

Ova pristupna točka prikazana na isječku koda 10 ima sljedeći tijeka rada:

1. Skenira trenutni direktorij u potrazi za mapama (svaka predstavlja stablo odlučivanja).
2. Za svaku mapu traži datoteku „description.txt“ i čita njezin sadržaj te uzima kao opis stabla.
3. Vraća JSON niz objekata, od kojih svaki sadrži naziv i opis stabla.

Ovo omogućuje sučelju da korisnicima predstavi popis dostupnih stabala odlučivanja, zajedno s opisima koji pomažu korisnicima da odaberu koje stablo žele koristiti.

6.5. Pristupna točka: POST /api/load_tree

Pristupna točka *POST /api/load_tree* upravlja učitavanjem novih stabala odluka u obliku komprimiranih XTM datoteka.

```
1. @app.route("/api/load_tree", methods=["POST"])
2. def load_tree():
3.     try:
4.         if 'file' not in request.files:
5.             raise BadRequest("No file part in the request")
6.
7.         uploaded_file = request.files["file"]
8.         if not uploaded_file.filename:
9.             raise BadRequest("No file selected")
10.
11.         # Extract the tree name from the uploaded file
12.         tree_name = os.path.splitext(uploaded_file.filename)[0]
13.
14.         # Check if a tree with this name already exists
15.         if tree_exists(tree_name):
16.             raise Conflict(f"A tree with the name '{tree_name}' already exists.
Please choose a different name or delete the existing tree first.")
17.
18.         entity_store.clear_tree()
19.         xml_file = extract_files(uploaded_file)
20.         xml_file_folder = xml_file.split('.xml')[0]
21.         entity_store.parse_xtm_file(os.path.join(xml_file_folder, xml_file))
22.         entity_store.build_decision_tree()
23.
24.         root_node = entity_store.find_root_node()
25.         if not root_node:
26.             raise NotFound("No root node found in the uploaded tree")
27.
28.         return jsonify(root_node)
29.     except Exception as e:
30.         app.logger.error(f"Error in load_triads: {str(e)}")
31.         if isinstance(e, (BadRequest, NotFound, Conflict)):
32.             raise
33.         raise InternalServerError("An unexpected error occurred")
```

Isječak koda 11, pristupna točka za učitavanje stabala

Ova složena pristupna točka prikazana na isječku koda 11 radi na sljedeći način:

1. Potvrđuje da je datoteka učitana.

2. Provjerava postoji li već stablo s istim imenom kako bi se spriječilo prepisivanje.
3. Prazni prošlo učitano stablo
4. Izvlači sadržaj učitane zip datoteke.
5. Raščlanjuje XTM datoteku i gradi stablo odlučivanja kao što je opisano u prošlom poglavlju
6. Vraća korijenski čvor novoučitanog stabla.

Pristupna točka uključuje sveobuhvatno rukovanje pogreškama za rješavanje različitih scenarija učitavanja, osiguravajući da sučelje prima jasne povratne informacije o uspjehu ili neuspjehu procesa učitavanja.

6.6. Pristupna točka: DELETE /api/tree/delete

Pristupna točka *DELETE /api/tree/delete* omogućuje brisanje postojećih stabala odlučivanja.

```
1. @app.route('/api/tree/delete', methods=["DELETE"])
2. def delete_tree():
3.     try:
4.         folder = request.args.get("name")
5.         if not folder:
6.             raise BadRequest("Missing 'name' parameter")
7.
8.         folder = folder.strip()
9.         if not os.path.exists(folder):
10.            raise NotFound(f"Folder '{folder}' not found")
11.
12.        shutil.rmtree(folder)
13.        return jsonify({"message": f"Tree '{folder}' has been deleted"})
14.    except Exception as e:
15.        app.logger.error(f"Error in delete_tree: {str(e)}")
16.        raise InternalServerError("An unexpected error occurred")
```

Isječak koda 12, pristupna točka za brisanje stabala

Pristupna točka prikazana na isječku koda 12 kreće ovako:

1. Prihvaća parametar „name“ koji navodi stablo za brisanje.
2. Provjerava postoji li navedena mapa.
3. Koristi shutil.rmtree za brisanje cijele mape i njezinog sadržaja.
4. Nakon uspješnog brisanja vraća poruku potvrde.

Ovo omogućuje korisnicima da upravljaju zbirkom stabala odluka, uklanjajući ona koja više nisu potrebna.

6.7. Pristupna točka: GET /api/tree_ascii

Pristupna točka *GET /api/tree_ascii* stvara ASCII tekstualni prikaz stabla odlučivanja, što je korisno za pregled cijelog stabla i svih puteva te za otklanjanje pogrešaka ili za korisnike koji preferiraju tekstualni prikaz strukture stabla.

```
1. @app.route("/api/tree_ascii", methods=["GET"])
2. def get_tree_ascii():
3.     try:
4.         root_node = entity_store.find_root_node()
5.         if not root_node:
6.             raise NotFound("No tree found")
7.
8.         tree_ascii = generate_tree_ascii(root_node)
9.         return jsonify({"tree_ascii": tree_ascii})
10.    except Exception as e:
11.        app.logger.error(f"Error in get_tree_ascii: {str(e)}")
12.        raise InternalServerError("An unexpected error occurred")
```

Isječak koda 13, pristupna točka za stvaranje ASCII prikaza stabla

Pristupna točka prikazana na isječku koda 13 prvo provjerava ako postoji učitano stablo, te ako postoji koristi pomoćnu funkciju *generate_tree_ascii*, koja će biti opisana kasnije, za stvaranje tekstualnog prikaza strukture stabla. Ovo može biti osobito korisno za vizualizaciju cjelokupne strukture složenih stabala odlučivanja.

6.8. Pristupna točka: GET /api/tree_graph

Pristupna točka *GET /api/tree_graph* dohvaća slikovne datoteke koje predstavljaju cjelokupnu strukturu stabla odlučivanja. Uz resurse, opis i XTM datoteku korisnici mogu dodati i *tree_graph* sliku bilo kojeg slikovnog formata koja se dohvati u ovoj pristupnoj točki.

```
@app.route("/api/tree_graph", methods=["GET"])
def get_tree_graph():
    try:
        path = request.args.get("name")
        if not path:
            raise BadRequest("Missing 'name' parameter")

        folder, filename = path.split('/')
        matching_files = [file for file in os.listdir(folder) if
file.startswith(filename)]

        if not matching_files:
            raise NotFound(f"No matching files found for '{filename}' in folder
'{folder}'")

        return send_file(os.path.join(folder, matching_files[0]))
    except Exception as e:
        app.logger.error(f"Error in get_tree_graph: {str(e)}")
        raise InternalServerError("An unexpected error occurred")
```

Isječak koda 14, pristupna točka za dohvaćanje slike stabla

Vidljiva je na isječku koda 14, a tijek rada ide ovako:

1. Prihvaća parametar „name“ koji navodi stazu do slike grafikona stabla.
2. Traži datoteke koje odgovaraju zadanom nazivu datoteke u navedenoj mapi.
3. Vraća prvu odgovarajuću datoteku koristeći Flaskovu funkciju send_file [9].

To omogućuje sučelju da prikaže vizualni prikaz cijele strukture stabla odlučivanja, što može biti osobito korisno za velika ili složena stabla i vizualno je preglednije od ASCII prikaza.

6.9. Pristupna točka: GET /api/get_path

Pristupna točka *GET /api/get_path* omogućuje sučelju da pritiskom na bilo koji čvor u ASCII prikazu stabla vrati put koji sustav treba uzeti da dođe do njega, kako bi se u navigaciji on mogao prikazati.

```
@app.route("/api/get_path", methods=["GET"])
def get_paths():
    try:
        node_id = request.args.get("node")
        if not node_id:
            raise BadRequest("Missing 'node' parameter")

        paths = find_paths_to_node(node_id)
        if not paths:
            raise NotFound(f"No paths found to node with id '{node_id}'")
        paths.reverse()
        return jsonify(paths)
    except Exception as e:
        app.logger.error(f"Error in get_paths: {str(e)}")
        raise InternalServerError("An unexpected error occurred")
```

Isječak koda 15, pristupna točka za dohvaćanje puta do čvora

Na isječku koda 15 prikazan je njezin rad:

1. Prima identifikaciju čvora u zahtjevu.
2. Poziva pomoćnu funkciju *find_paths_to_node* koja korištenjem *breadth first search* algoritma pronalazi sve puteve do čvora
3. Nakon toga, vraćeni putevi se okrenu kako bi prvi nađeni bio na vrhu liste.
4. Vrati se lista puteva.

Svaki čvor može imati više roditelja stoga je potrebno vratiti sve moguće puteve, a na sučelju se onda definira birani put.

6.10. Pomoćne funkcije

6.10.1. Funkcija extract_files

Funkcija extract_files odgovorna je za obradu učitane zip datoteke koja sadrži XTM datoteku stabla odlučivanja i pridružene resurse. Ova je funkcija kritična za proces učitavanja stabla. Ona se poziva u *load_tree* funkciji i zaslužna je za glavni dio izvlačenja podataka iz komprimiranih datoteka. Ovdje je pregled njegovog rada:

1. Otvara učitanu zip datoteku s pomoću modula zipfile.
2. Identificira XML datoteku (koja bi trebala biti XTM datoteka) i izbornu datoteku opisa u zip datoteci.
3. Stvara novu mapu za stablo, nazvanu po XML datoteci.

4. Provjerava postoji li stablo s ovim imenom kako bi se spriječilo prepisivanje.
5. Vadi XML datoteku i datoteku opisa (ako postoji) u novu mapu.
6. Stvara podmape „images“ i „texts“ za organiziranje resursa stabla.
7. Zatim obrađuje svaku datoteku u zip-u:
    - Datoteke „tree_graph“ vade se izravno u glavnu mapu.
    - Slikovne datoteke (.png, .jpg, .jpeg) se vade, optimiziraju s pomoću PIL biblioteke kako bi im se smanjila veličina i spremaju u mapu „images“.
    - SVG datoteke premještaju se u mapu „images“ bez izmjena.
    - Tekstualne datoteke (.txt, .htm, .html) premještaju se u mapu „texts“.
    - Sve druge datoteke ekstrahiraju se u „images“ ili „texts“ na temelju njihovih nastavaka.
8. Čisti sve prazne direktorije stvorene tijekom procesa izdvajanja.
9. Konačno, vraća naziv XML datoteke i naziv mape stvorene za ovo stablo.

Ova funkcija osigurava da su sve učitane datoteke pravilno organizirane i optimizirane, postavljajući ispravnu strukturu za rad parsera stabla odlučivanja.

6.10.2. Funkcija create_node

Funkcija create_node odgovorna je za konstrukciju detaljnog prikaza čvora i njegovih neposrednih potomaka u stablu odlučivanja. Ova je funkcija ključna za pružanje sučelju potrebnih informacija za prikaz i navigaciju strukturom stabla. Poziva se kod većina pristupnih točaka u kojima su potrebni čvorovi za vraćanje na sučelje. Evo kako to funkcionira:

1. Kao ulaz uzima node_id, što je identifikator za čvor za koji želimo stvoriti reprezentaciju.
2. Stvara varijablu children_list koja je lista djece za čvor koji se stvara. Za svako dijete čvora:
    - „question“ je sam čvor dijete

- "answer" je poveznica koja povezuje roditelja s čvorom djetetom iz atributa „question“,

3. Dohvaća sam čvor iz *entity_store.entities*.
4. Konačno, vraća rječnik s dva ključa:
    - „root": sam čvor, pretvoren u rječnik.
    - „children": popis podređenih čvorova, svaki sa svojom povezanom vezom.

Ova struktura omogućuje sučelju da prikaže čvor sa svim njegovim potencijalnim izborima (djeca) i oznakama za te izbore (veze).

6.10.3. Funkcija generate_tree_ascii

Funkcija generate_tree_ascii stvara ASCII tekstualni prikaz stabla odlučivanja. Ona je korisna za pružanje tekstualne vizualizacije strukture stabla. Evo kako funkcionira:

1. Koristi rekurzivne pozive za prolazak kroz cijelu strukturu stabla.
2. Parametri funkcije su:
    - čvor: trenutni čvor koji se obrađuje.
    - posjećeno: Skup za praćenje posjećenih čvorova, sprječavanje beskonačnih petlji u slučaju kružnih referenci.
    - node_ids: Rječnik za dodjelu jedinstvenih numeričkih ID-ova čvorovima.
    - prefiks: Prefiks niza za trenutni redak, koji se koristi za stvaranje strukture stabla.
    - is_last: Booleova vrijednost koja pokazuje je li ovo zadnje dijete njegovog roditelja.
3. Gradi ASCII reprezentaciju red po red:
    - Koristi " └──" za posljednje dijete i " ├──" za ostalu djecu.
    - Uključuje numerički ID i oznaku za svaki čvor.
4. Obrađuje slučaj ponovnog posjećivanja čvorova (u slučaju više roditelja) jednostavnim ispisom ID-a čvora i oznake bez daljnje rekurzije.

5. Za svako dijete trenutnog čvora, čini rekurzivan poziv za *generate_tree_ascii*.
6. Funkcija gradi cjelokupnu ASCII reprezentaciju kroz te rekurzivne pozive, na kraju vraćajući cijelo stablo kao niz.

Ova funkcija pruža način vizualizacije strukture stabla u jednostavnom tekstualnom formatu i koristi se u pristupnoj točki *get_tree_ascii* gdje odvija većinu logike.

7. Dizajn i primjena korisničkog sučelja

Riješenje predstavlja korisničko sučelje dizajnirano za laku navigaciju i interakciju sa stablima odlučivanja. Izrađeno korištenjem Next.js i TypeScript, ono koristi komponente Material-UI za stvaranje dosljednog i responzivnog dizajna [11]. Ovo poglavlje će istražiti ključne aspekte korisničkog sučelja, njegovu primjenu i korisničko iskustvo koje pruža.

7.1. Cjelokupni izgled i navigacija

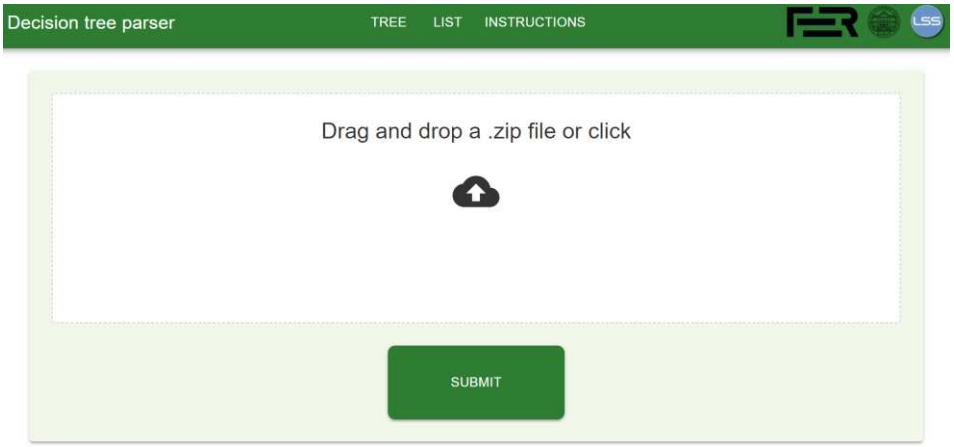
Program ima čist i intuitivan izgled, s trajnom navigacijskom trakom na vrhu zaslona. Ova traka, primjenjena u komponenti *NavigationBar*, omogućuje brzi pristup glavnim dijelovima programa: „TREE“, „LIST“ i „INSTRUCTIONS“ prikazanim na slici 1 u navigacijskoj komponenti.

Slika 1, navigacija web sučelja

Navigacijska traka koristi komponente *AppBar*, *Toolbar* i *Button* od Material-UI, osiguravajući dosljedan izgled i dojam na različitim uređajima i veličinama zaslona [11].

7.2. Sučelje za učitavanje stabala

Za dodavanje novih stabala odluka u sustav, sučelje nudi polje za učitavanje datoteka, primjenjeno u komponenti *FileSubmitCard*.



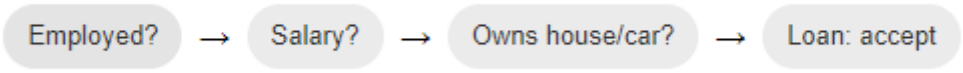
Slika 2, stranica za učitavanje datoteke

Ovo sučelje koristi komponentu *DropzoneArea* koje se vidi na slici 2 za pružanje jednostavnog iskustva učitavanja datoteke povuci i ispusti. Korisnici mogu jednostavno prenijeti zip datoteke koje sadrže podatke o njihovom stablu odlučivanja, s jasnim uputama na samom području za učitavanje. Nakon što se učita .zip datoteka može se pretisnuti gumb *Submit* koji u pozadini poziva pristupnu točku *load_tree* te se tako učita stablo u program. Nakon što se učita otvara se u kojoj se radi interakcija sa stablom.

7.3. Interakcija sa stablom odluka

Srce sučelja leži u interakcijskom dijelu stabla odlučivanja, primjenjenom u komponenti *TreeParser*. Ovo sučelje predstavlja korisnicima korak po korak putovanje kroz stablo odlučivanja, prikazujući relevantne informacije i svojstva na svakom čvoru. Ključne značajke ovog sučelja uključuju:

a) Praćenje napretka: Koračna komponenta Material-UI na vrhu zaslona prikazuje korisnikov trenutni položaj u stablu odlučivanja, pružajući kontekst i omogućavajući jednostavno praćenje unatrag. Može se vidjeti na slici 3 gdje strelice označavaju smjer odluka. Uz prikaz povijesti odluka, ova komponenta omogućuje pritiskom na bilo koji od čvorova da se korisnik vrati na taj čvor.



Slika 3, navigacija po čvorovima

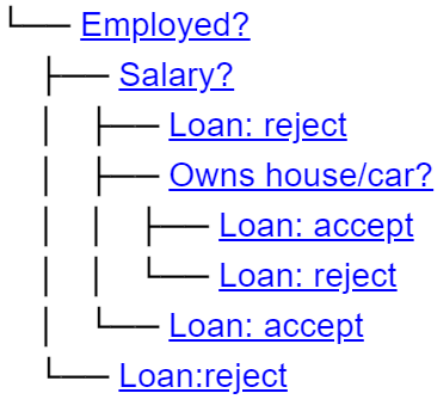
b) Prikaz trenutnog čvora: Oznaka trenutnog čvora je vidljivo prikazana, dajući korisnicima jasnu naznaku njihove pozicije u procesu odlučivanja. Na slici 6 trenutni čvor je „Employed?“.

c) Informacijski zaslon: Sučelje je podijeljeno u dva glavna dijela koja se mogu vidjeti na slici 6. Lijeva strana prikazuje tekstualne informacije vezane uz trenutni čvor, kao što je u slučaju sa slike gdje piše, “Jeste li zaposleni?“, dok desna strana prikazuje pridruženu sliku ako je dostupna, na slici je primjer slike gdje pišu najčešće riječi u vezi

zaposlenja. Ovaj izgled osigurava da korisnici imaju sve relevantne informacije spremno dostupne za donošenje informiranih odluka.

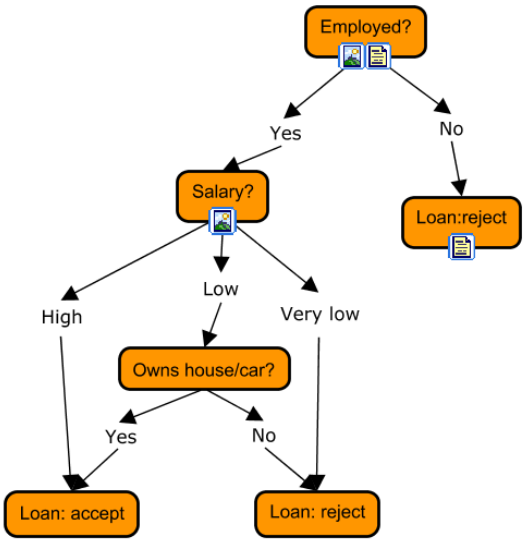
d) Mogućnosti navigacije: Na dnu zaslona korisnicima se prikazuju gumbi koji odgovaraju dostupnim izborima u trenutnom čvoru. Ovi se gumbi dinamički stvaraju na temelju djece trenutnog čvora, osiguravajući da se sučelje prilagođava strukturi svakog jedinstvenog stabla odlučivanja. Na slici 6 se vide dva plava gumba, na jednom piše „YES“, a na drugom „NO“, oni su odgovori na pitanje čvora „Employed?“. Kada se jedan od njih stisne stablo prelazi u odabrano dijete.

e) ASCII prikaz stabla: ikona stabla u gornjem desnom kutu omogućuje korisnicima pregled ASCII reprezentacije cijele strukture stabla, pružajući pogled iz ptičje perspektive na proces odlučivanja. Uz to, može se pritisnuti na bilo koji čvor te će program automatski doći do tog čvor i prikazati ga na sučelju, što je prikazano na slici 4, gdje su čvorovi označeni plavom bojom i podcrtani.



Slika 4, ASCII prikaz stabla

f) Slikovni prikaz stabla: ikona slike u gornjem desnom kutu prikaže priloženu sliku grafa, iz datoteke *tree_graph*. Ona se prikaže u iskočnom prozoru, što se može vidjeti na slici 5.



Slika 5, slikovni prikaz stabla

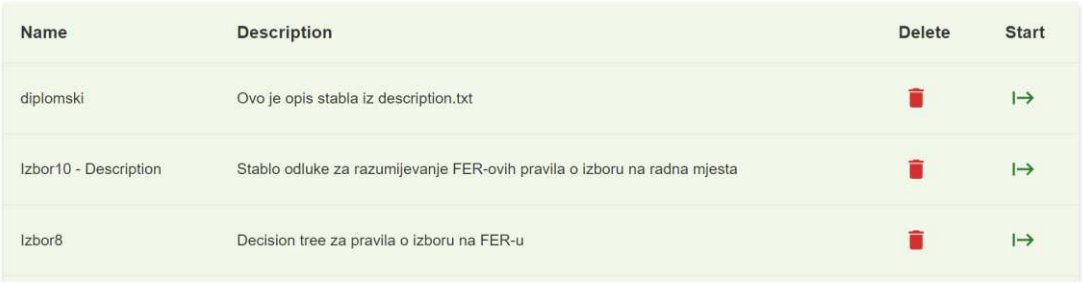
g) Mogućnost povratka: ikona strelice u gornjem lijevom kutu omogućuje korisnicima povrata na prijašnji čvor.

Slika 6, stranica za interakciju sa stablom

7.4. Sučelje za upravljanje stablima

Program nudi sučelje za upravljanje više stabala odluka, primjenjeno u komponenti *ListPage*. Ovo sučelje predstavlja tablični prikaz svih dostupnih stabala, dopuštajući korisnicima pokretanje, brisanje ili pregled imena i opisa svakog stabla.Na slici 7 se može vidjeti tablica sa stupcima:

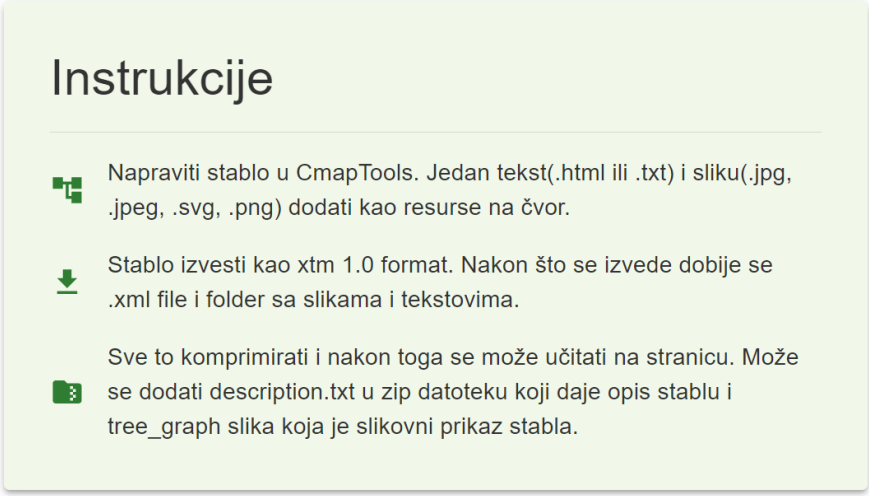
- „Name“, koja označava ime stabla koje se izvlači iz imena .xml datoteke,
- „Description“, ovaj stupac se izvlači iz datoteke *description.txt* i označuje opis određenog stabla,
- „Delete“, ovaj stupac daje mogućnost brisanja stabla, pritiskom na crvenu ikonicu kante za smeće,
- „Start“, pritiskom na ikonicu u ovom stupcu sustav će započeti interakciju s odabranim stablom.



Slika 7, lista stabala s akcijama

7.5. Instrukcije

Program ima jednostavnu stranicu s uputama, koja korisnicima služi kao početna točka nakon ulaženja na web stranicu. Ova stranica pruža sažete, ali sveobuhvatne smjernice o tome kako pripremiti i učitati stabla odlučivanja u sustav.



Slika 8, kartica s instrukcijama

Stranica s uputama sa slike 8 podijeljena je na naslov „Instrukcije“ i na tri sekcije koje objašnjavaju svoje teme. U prvoj sekciji označenoj s ikonicom stabla, objašnjeno je da treba napraviti stablo u alatu CmapTools, te da se slika i tekst mogu dodati kao resursi na čvor. Nakon toga u sekciji sa ikonicom strelice prema dolje objašnjeno je da stablo treba izvesti u formatu XTM 1.0. U zadnjoj sekciji označenoj s ikonicom komprimirane mape opisano je da je potrebno komprimirati izvedene datoteke te da je moguće dodati datoteke *description.txt* i *tree_graph* sliku koje su opisane u prijašnjim poglavljima.

8. Pojedinosti o primjeni programskog riješenja

Program slijedi arhitekturu klijent-poslužitelj, s jasnim odvajanjem između frontend i backend komponenti. Ovaj dizajn omogućuje responzivno korisničko sučelje dok se na poslužitelju bave složenim zadacima parsiranja i upravljanja podacima [8].

Sučelje je izgrađeno korištenjem Next.js, okvira React koji pruža prikazivanje na strani poslužitelja i stvara statične web stranice. Ovaj izbor nudi nekoliko prednost [9]i:

- Poboljšana izvedba kroz iscrtavanje na strani poslužitelja
- Jednostavno usmjeravanje i arhitektura temeljena na stranicama
- Ugrađena podrška za TypeScript

Iskorišten je TypeScript za sigurnost tipova i poboljšano iskustvo programera [10]. Komponente korisničkog sučelja prvenstveno su izgrađene korištenjem Material-UI, čime se osigurava dosljedan i responzivan dizajn na različitim uređajima [11].

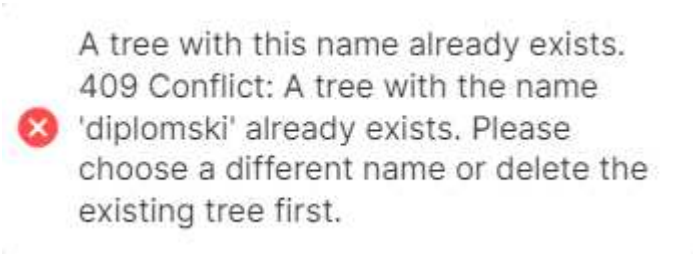
Upravljanjem stanja na sučelju upravlja se s pomoću Reactove *useState* kuke, pružajući jednostavan, ali učinkovit način za upravljanje stanjem na razini komponente bez potrebe za složenijim bibliotekama za upravljanje stanjem [9]. Poslužitelj je primjenjen kao monolitni program koji koristi Python i Flask web okvir. Jednostavnost i prilagodljivost Flaska učinile su ga idealnim izborom za primjenu na strani poslužitelja [12]. Ključne komponente poslužitelja uključuju:

- RESTful programsko sučelje pristupne točke za operacije stabla
- XTM logika parsiranja datoteke
- Interakcije datotečnog sustava za pohranu stabla

Za razliku od tradicionalnih web primjena, ovaj sustav ne koristi bazu podataka. Umjesto toga, stabla odlučivanja i njima pridruženi resursi pohranjuju se izravno u datotečni sustav. Svako je stablo pohranjeno u vlastitu mapu u korijenskom direktoriju poslužitelja. Mapa sadrži XTM datoteku, zajedno s podmapama za slike i tekstualne resurse. Opcionalno

može sadržavati i datoteku opisa i sliku stabla. Ovaj pristup pojednostavljuje primjenu i smanjuje složenost sustava, iako može predstavljati izazov skalabilnosti za vrlo veliki broj stabala [12].

U slučaju greške na poslužitelju ona se uhvati i obradi te se pošalje u odgovoru zajedno sa statusnim kodom u formatu JSON na sučelje. Na sučelju ta se poruka uzima te se prikazuje u komponenti tost koja se može vidjet na slici 9.



Slika 9, prikazivanje greške

Program je dizajniran tako da se može lako namjestiti, uz najmanju potrebnu konfiguraciju. Nije potrebno postavljanje baze podataka jer su svi podaci pohranjeni u datotečnom sustavu. Varijable okruženja se ne koriste, što pojednostavljuje proces primjene. Sučelje se može izgraditi i služiti kao statične datoteke [9], s poslužiteljem koji obrađuje dinamičke zahtjeve.

9. Rasprava i budući rad

Jedan od primarnih izazova tijekom procesa razvoja bio je određivanje najprikladnijeg formata datoteke za izvoz stabla odlučivanja iz CmapTools-a. Odabir XTM (XML Topic Maps) formata kao posrednog prikaza pokazao se ključnom odlukom. Iako je XTM pružio strukturirani način za predstavljanje podataka stabla odlučivanja, došao je i s određenim ograničenjima, posebno u očuvanju vizualnih stilova i dodatnih metapodataka iz CmapTools-a [7].

Još jedan značajan izazov bio je dizajniranje korisničkog sučelja koje je istovremeno jednostavno i intuitivno za interakciju sa složenim strukturama stabla odlučivanja. Cilj je bio stvoriti sučelje koje bi moglo predstaviti složene procese donošenja odluka na način koji je pristupačan korisnicima s različitim razinama tehničke stručnosti. To je zahtijevalo pažljivo razmatranje principa korisničkog iskustva i iterativne procese dizajna.

9.1. Trenutna ograničenja

Trenutna primjena parsera stabla odlučivanja, iako funkcionalna, ima neka ograničenja koja je važno istaknuti:

1. Gubitak vizualnog stila: Proces izvoza XTM-a ne čuva vizualne stilove čvorova iz CmapTools-a [3]. Ovo ograničenje znači da se bilo kakvo kodiranje bojama, varijacije fonta ili drugi vizualni znakovi korišteni u izvornoj konceptualnoj mapi gube u prijevodu na web sučelju.
2. Nedostatak anotacija i informacija o čvorovima: XTM format, kako je primjenjen, ne bilježi anotacije ili dodatne informacije povezane s čvorovima u CmapTools-u. Ovo ograničenje ograničava dubinu informacija koje se mogu prenijeti u parseru stabla odlučivanja [3].
3. Ovisnost o vanjskim alatima: Trenutni tijek rada zahtijeva od korisnika da stvore svoja stabla odlučivanja u CmapTools-u prije nego što ih učitaju na sučelje. Ovaj dvostupanjski proces mogao bi biti prepreka za korisnike koji nisu upoznati s CmapTools-om ili preferiraju integriranija rješenja.

9.2. Inovacije i doprinosi

Unatoč ovim izazovima i ograničenjima, program za parsiranje stabla odlučivanja predstavlja značajan korak naprijed u činjenju interakcija stabla odlučivanja pristupačnijima i korisnički prilagođenijima. Glavni doprinosi projekta uključuju:

1. Pojednostavljena vizualizacija: Sučelje pruža jasnu vizualizaciju stabla odlučivanja, olakšavajući korisnicima navigaciju kroz složene procese donošenja odluka.
2. Web alat: Prevođenjem konceptualnih mapa CmapTools-a u format web stranice, sučelje povećava pristupačnost i mogućnost dijeljenja stabla odlučivanja.

9.3. Budući rad

Na temelju trenutnog stanja projekta i identificiranih ograničenja, može se istražiti nekoliko pravaca za budući rad i poboljšanje. Buduće iteracije programa mogle bi se usredotočiti na očuvanje više informacija iz izvornih CmapTools datoteka korištenjem nekog od drugih formata za izvoz koje nudi CmapTools.

Značajno poboljšanje bilo bi integracija mogućnosti stvaranja stabla odlučivanja izravno u web sučelju. Ova značajka bi isključila ovisnost o vanjskim alatima poput CmapTools-a, pojednostavljujući cijeli proces od stvaranja stabla do interakcije.

Program za parsiranje stabla odlučivanja predstavlja obećavajući korak prema činjenju stabla odlučivanja pristupačnijima i korisnički prilagođenijima u web okruženjima. Iako trenutna primjena ima određena ograničenja, ona pruža čvrstu osnovu za buduća poboljšanja. Rješavanjem identificiranih područja za poboljšanje i proširenjem mogućnosti programa, buduće iteracije imaju potencijal značajno utjecati na način na koji se stabla odlučivanja stvaraju, dijele i koriste u različitim domenama. Fokus projekta na korisnički orijentiran dizajn i web pristupačnost dobro se usklađuje s trenutnim trendovima u interaktivnoj vizualizaciji podataka i sustavima za podršku

odlučivanju, pozicionirajući ga za trajnu relevantnost i razvoj u ovom brzo razvijajućem području.

10. Zaključak

Ovaj diplomski rad predstavlja razvoj i primjenu programa za parsiranje stabala odlučivanja, osmišljene da premosti jaz između složenih procesa donošenja odluka i web interakcija prilagođenih korisniku. Primarni cilj projekta transformacije stabala odlučivanja stvorenih u CmapTools-u u interaktivni format temeljen na webu postignut je višestrukim pristupom, kombiniranjem modernih web tehnologija s promišljenim dizajnom korisničkog sučelja.

Ključna postignuća uključuju uspješno analiziranje i transformiranje XTM datoteka izvezenih iz CmapToolsa u interaktivni web format, razvijanje intuitivnog korisničkog sučelja koje vodi korisnike kroz složena stabla odlučivanja korak po korak i primjenu značajki kao što su vizualizacija stabla, navigacija čvorova i upravljanje datotekama. Projekt doprinosi području sustava za potporu odlučivanju pokazujući praktičan pristup izradi stabala odlučivanja pristupačnijim i lakšim za korištenje.

Izazovi na koji su nastali tijekom razvoja, posebno u očuvanju vizualnih elemenata i metapodataka iz CmapTools, istaknuli su područja za potencijalna poboljšanja i buduća istraživanja. Uvidi dobiveni ovim istraživanjem ne samo da pridonose neposrednom području sustava za podršku odlučivanju, već također informiraju šire područje interakcije između čovjeka i računala u vizualizaciji i analizi podataka.

Zaključno, stalne povratne informacije korisnika i iterativni razvoj bit će ključni u usavršavanju programa i rješavanju identificiranih ograničenja. Temelj postavljen ovim projektom pruža čvrstu platformu za buduće inovacije u vizualizaciji i interakciji stabla odlučivanja, što potencijalno vodi do informiranijih i učinkovitijih procesa donošenja odluka u različitim sektorima društva.

11. Literatura

1. B. Sriram, " Educational Software Development – Users Requirement Analysis", International Journal of Computer Applications Volume 31– No.10, listopad 2011. Str. 14-18
2. Freeman, Scott et al. “Active learning increases student performance in science, engineering, and mathematics.” Proceedings of the National Academy of Sciences of the United States of America vol. 111,23, 2014
3. CmapTools, "Documentation Cmaps", s Interneta, https://cmap.ihmc.us/docs/documentation-cmaps, 1 lipnja 2024
4. LucidChart, „How to Make a Decision Tree Diagram“, s Interneta, https://www.lucidchart.com/pages/how-to-make-a-decision-tree-diagram, 1 lipnja 2024
5. TreePlan, „TreePlan® Decision Tree Add-in for Excel ", s Interneta, https://www.lucidchart.com/pages/how-to-make-a-decision-tree-diagram, 1 lipnja 2024
6. Decision Tree Resolver, „Decision Tree Resolver“, s Interneta, http://diana.zesoi.fer.hr:8080/decisionTreeResolver/index.jsp, 10 ožujka 2024
7. TopicMaps.Org Authoring Group , "XML Topic Maps (XTM) 1.0", s Interneta, http://www.topicmaps.org/xtm/1.0/, 15 travnja 2024
8. Alex Berson, „Client/server architecture (2nd ed.)“, McGraw-Hill, Inc., USA, 1996
9. Lazuardy, Mochammad Fariz Syah, and Dyah Anggraini. "Modern front end web architectures with react. js and next. js." Research Journal of Advanced Engineering and Science 7.1, pp. 132-141, 2022
10. „TypeScript is JavaScript with syntax for types.“, s Interneta, https://www.typescriptlang, 30 travnja 2024
11. Grinberg, Miguel, „Flask web development“, O'Reilly Media, Inc., 2018.
12. SEARS, Russell; VAN INGEN, Catharine; GRAY, Jim, „To blob or not to blob: Large object storage in a database or a filesystem?“, arXiv preprint cs/0701168, 2007.