

Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

Programtervező informatikus BSc

SZAKDOLGOZAT

Gráfban elosztott tudás leírására,
elérésére és menedzselésére alkalmas
keretrendszer kidolgozása

Solti Gábor

Témavezető: dr. Bertók Botond

2020



PANNON EGYETEM

MŰSZAKI INFORMATIKAI KAR

Programtervező informatikus BSc

Veszprém, 2020. október 20.

SZAKDOLGOZAT TÉMAKIÍRÁS

Solti Gábor

Programtervező informatikus BSc nappali szakos hallgató részére

Gráfban elosztott tudás leírására, elérésére és menedzselésére alkalmas keretrendszer kidolgozása

Témavezető: Dr. Bertók Ákos Botond, egyetemi docens

A feladat leírása:

A nagyméretű adathalmazokon való felhasználói böngészés, navigálás vagy keresés során fontos szempont, hogy nagyon kevés explicit felhasználói szándék esetén is releváns és specifikus találatokat eredményezzen. A jelölt feladata olyan felhasználói alkalmazás elkészítése, ami irányított gráfba rendezett ontológián hatékony navigálást és keresést tesz lehetővé.

A megvalósítás lépései:

- Követelmények összegyűjtése.
- Lehetséges fejlesztői eszközök összehasonlítása, megfelelő eszközök kiválasztása.
- Adatstruktúrák megtervezése az adatelemek koncepcionális kapcsolatainak reprezentálására.
- A kapcsolatok lehetséges megjelenítési stratégiáinak összehasonlítása.
- A kiválasztott stratégiának megfelelő grafikus elrendezés kiválasztása.
- Grafikus felhasználói felület tervezése és implementálása.
- Adatbázis-séma tervezése és megvalósítása.
- Automatikus tesztek készítése az alkalmazás kritikus részeihez.


Dr. Bertók Ákos Botond
egyetemi docens
témavezető


Dr. Süle Zoltán
egyetemi docens
szakfelelős

Nyilatkozat

Alulírott Solti Gábor hallgató, kijelentem, hogy a dolgozatot a Pannon Egyetem Rendszer- és Számítástudományi tanszékén készítettem a programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2020. december 7.

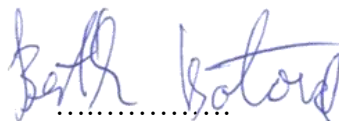


Solti Gábor

Alulírott Bertók Botond témavezető kijelentem, hogy a dolgozatot Solti Gábor a Pannon Egyetem Rendszer- és Számítástudományi tanszékén készítette programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védelemre bocsátását engedélyezem.

Veszprém, 2020. december 7.



Dr. Bertók Botond

Köszönetnyilvánítás

Köszönöm a témavezetőmnek, Dr. Bertók Botondnak a szakdolgozatom elkészítéséhez nyújtott segítséget.

Köszönöm tanárainknak, hogy az elmúlt évek során magas színvonalú oktatásban részesülhettem a Pannon Egyetemen elvégzett kurzusokon.

TARTALMI ÖSSZEFOGLALÓ

A nagy méretű adathalmazokon való felhasználói böngészés, navigálás vagy keresés során fontos szempont, hogy nagyon kevés explicit felhasználói szándék esetén is releváns és specifikus találatokat eredményezzen. Jelen dolgozat témája olyan alkalmazás tervezése és megvalósítása, ami irányított gráfba rendezett hierarchikus ontológián hatékony navigálást és keresést tesz lehetővé.

Bemutatom, hogy a irányított gráf szerkezeti sajátosságait kihasználva nemcsak a közvetetten releváns találatokat lehet könnyen kikeresni és megjeleníteni, hanem összetett gráfműveletek segítségével egyszerűen meg lehet találni azokat az alternatív elemeket, amik a keresési szándék adaptív megjósolásával lehetővé teszik kis számú alternatív szűrési szempont felajánlását a felhasználó számára.

Kulcsszavak: *ontológia, felhasználói keresés, irányított gráf*

ABSTRACT

When a user is browsing, navigating, or searching large data sets, it is important to produce relevant and specific results even with very little explicit user intent. The topic of this thesis is the design and implementation of an application that enables efficient navigation and search on a hierarchical ontology arranged in a directed graph.

I show that by taking advantage of the structural features of a directed graph, not only can indirectly relevant results be easily retrieved and displayed, but also with the use of complex graph operations it is easy to find alternative elements that allow the user to be offered a small number of alternative filtering criteria by adaptive prediction of the search intent.

Keywords: *ontology, search by the user, directed graph*

Tartalomjegyzék

Köszönetnyilvánítás.....	4
TARTALMI ÖSSZEFOGLALÓ.....	5
ABSTRACT.....	6
Tartalomjegyzék.....	7
Bevezetés.....	8
Irodalmi áttekintés és ismert megoldások.....	9
Gráfban tárolt ontológiák megjelenítése és navigálása.....	10
WebVOWL.....	11
Listák.....	12
Evernote.....	13
Kognitív torzítások.....	14
A felhasználói szándék.....	15
Célkitűzés.....	17
Felhasznált technológiák.....	18
Technológia kiválasztása.....	18
FXML.....	19
CSS.....	20
Követelmények.....	21
Tervezés.....	22
Komplex gráfműveleteket biztosító keretrendszer megtervezése.....	22
Az adatelemek közötti kapcsolatokat reprezentáló adatstruktúra megtervezése.....	24
A felhasználói felület tervezése.....	26
Megvalósítás.....	28
Helyi adatbázis.....	28
A gráfműveletek megvalósítása.....	29
Felhasználói felület felépítése.....	30
Natív integráció.....	31
Automatikus tesztek készítése.....	32
Az alkalmazás működése.....	34
Az alternatív szűrési szempontok kiválasztásának általános módja.....	37
További felhasználói navigálás a szűrési eredmények között.....	39
Értékelés.....	45
Lehetséges felhasználási területek.....	45
Lényegesebb továbbfejlesztési lehetőségek.....	48
A közös utódok nulla be-fokú elemeinek megjelenítése.....	48
A választási lehetőséget nem nyújtó alternatív szűrőelemek automatikus szűkítése.....	48
A szűrőelemek dinamikus szűkítése és bővítése.....	48
Az adatmodell és a gráfműveletek jobb elkülönítése.....	49
A felhasználói élmény fejlesztése.....	49
Megosztás.....	49
Kategória szerinti keresés.....	50
Összefoglalás.....	51
Irodalomjegyzék.....	52

Bevezetés

A nagy méretű adathalmazokon való felhasználói böngészés vagy keresés elősegítésére sokféle csoportosítást, katalogizálást, vagy más adatstruktúrát lehet használni. Ennek a technológiai eszköznek a megválasztása azt is befolyásolja, hogy hogyan függ a keresés eredménye a felhasználó explicit keresési szándékától.

Egy fastruktúrába rendezett könyvtár-rendszer esetén például első lépésben csak a legáltalánosabb kategóriák közül kell választani. Van viszont egy olyan hátránya, hogy egy elem csak egy könyvtárba tartozhat. Ezzel szemben a címkézés többszörös kategorizálást tesz lehetővé, vagyis kevésbé konkretizálja, hogy mit kell előre tudnia a felhasználónak. Cserébe viszont hiányzik belőle a fastruktúra hierarchikus felépítése.

Ennek a két struktúrának az előnyeit próbálja egyesíteni az adatelemek irányított gráfba rendezése. Ebben a dolgozatban azt fogom bemutatni, hogy hogyan lehet szoftveres módszerekkel elősegíteni az egyszerű irányított gráfba rendezett adatelemek felhasználói navigálását.

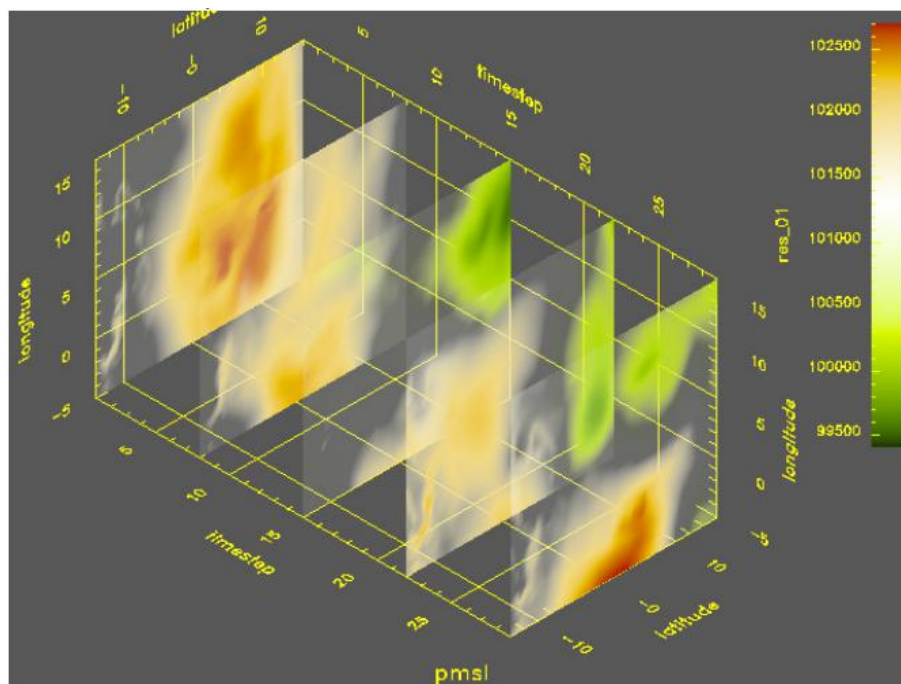
Ehhez azt fogom feltételezni, hogy a gráf irányított élei tetszőleges alárendeltségi viszonyt reprezentálnak. Ez jelentheti a rész viszonyát az egészhez, a konkrét dolog viszonyát az általánoshoz, vagy jelentheti egy bizonyos kategóriába tartozás viszonyát.

Irodalmi áttekintés és ismert megoldások

A komplex viszonyrendszerben kifejezett és nagy méretű adathalmazok felhasználói értelmezését és böngészését lehetővé tevő felhasználói felületek tervezése számos kihívást hordoz magában.

Nem csak az adatelemek intuitív elérhetőségét kell lehetővé tenni, hanem az adatok közötti összefüggésekben megmutatkozó, átfogó tudásnak is hozzáférhetőnek kell lennie a felhasználó számára. Az összetett és nagy volumenű ismeretrendszerek megjelenítése sajátos problémák megoldását igényeli, mert megfelelő mennyiségű információt kell megjeleníteni, miközben korlátozni kell a felhasználó számára a kognitív terhelést. [7]

A grafikus megjelenítés lehetőségeinek az alapját az adathalmaz problématerén belüli implicit szerkezete adja. A földrajzi térben értelmezhető adatok, például meteorológiai adatok esetén interaktív térképek biztosíthatják az intuitív adatkiválasztás lehetőségét a felhasználó számára, és a kiválasztott adatok közötti felhasználói orientációt. [5]



Térbeli és időbeli dimenziókban értelmezett adatok grafikus megjelenítése [5]

A komplex viszonyrendszereket alkotó ismeretek feletti felhasználói felületek egy sokat kutatott megközelítése a szemantikus háló, ami az interneten keresztül elérhető tartalmak jelentés szerinti rendszerezését, gépi ágensek általi értelmezhetőségét és hatékony felhasználói keresését célozza. A szemantikus web egyik technológiai alappillére az Extensible Markup Language (XML) a tartalmak strukturált tárolására. A másik a Resource Description Framework (RDF), ami lehetővé teszi a tartalmakhoz kapcsolódó metaadatok tárolását. A harmadik alaptechnológia pedig az ontológia, ami a tartalmak közötti kapcsolatok hálózatát jelenti. [9]

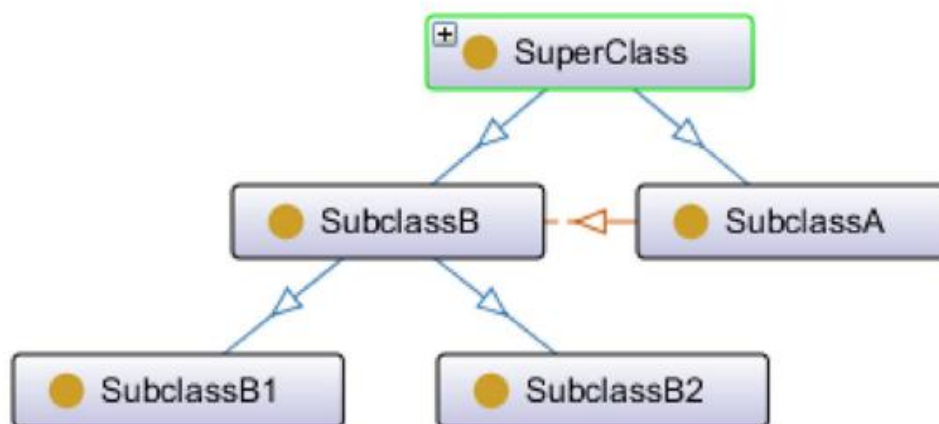
A tudásanyag ontológiákba rendezése nem csak az automatizált feldolgozást teszi hozzáférhetőbbé, hanem a felhasználók általi böngészést és keresést is elősegíti.

Gráfban tárolt ontológiák megjelenítése és navigálása

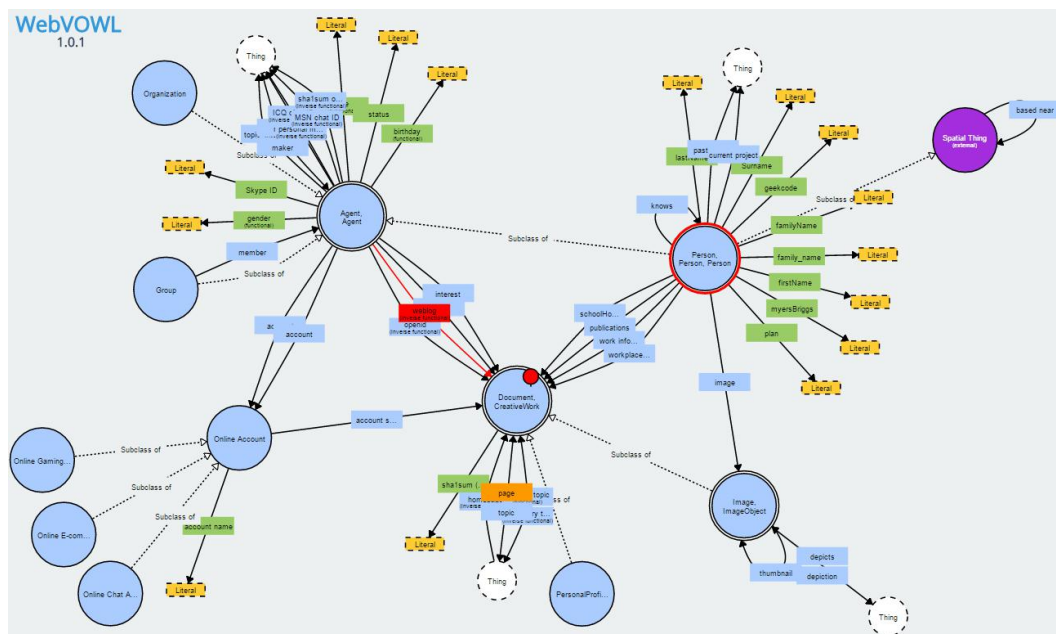
Az strukturált gráfokba rendezett adatokon való navigálásra sok két vagy több dimenziós, grafikus megoldás létezik.

Ezek az ontológiák sok különböző típusú relációt definiálnak az elemek között, és általában grafikus csomópontok közötti összeköttetésekkel valamint térbeli elhelyezkedés szerint ábrázolják a kapcsolatokat.

Egy kézenfekvő grafikus megoldás a gráf két dimenziós ábrázolása.



Vertikális fa elrendezés (OntoGraph plugin, Protégé) [6]

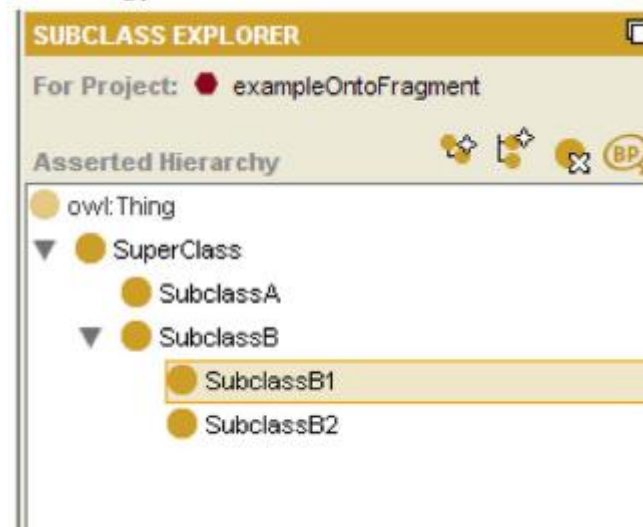


A WebOWL webes alkalmazás, ontológiák grafikus megjelenítésére (forrás: WebVOWL)

A WebVOWL alkalmazás irányított gráf formájában jeleníti meg az ontológiákat, és lehetőséget nyújt a megjelenítés interaktív testreszabására, és a VOWL vizualizációt automatikusan generálja JSON fájlokból.

Listák

A nagyon nagy adathalmazok és sok kapcsolat esetén a grafikus megjelenítések már nem nyújtanak átlátható és jól kezelhető felületet. A behúzásos listák, mint úgynevezett másfél dimenziós grafikus megjelenítések lehetővé teszik nagyobb információs halmazok böngészését, de csak egyetlen, hierarchikus viszonyt tudnak ábrázolni. [6]

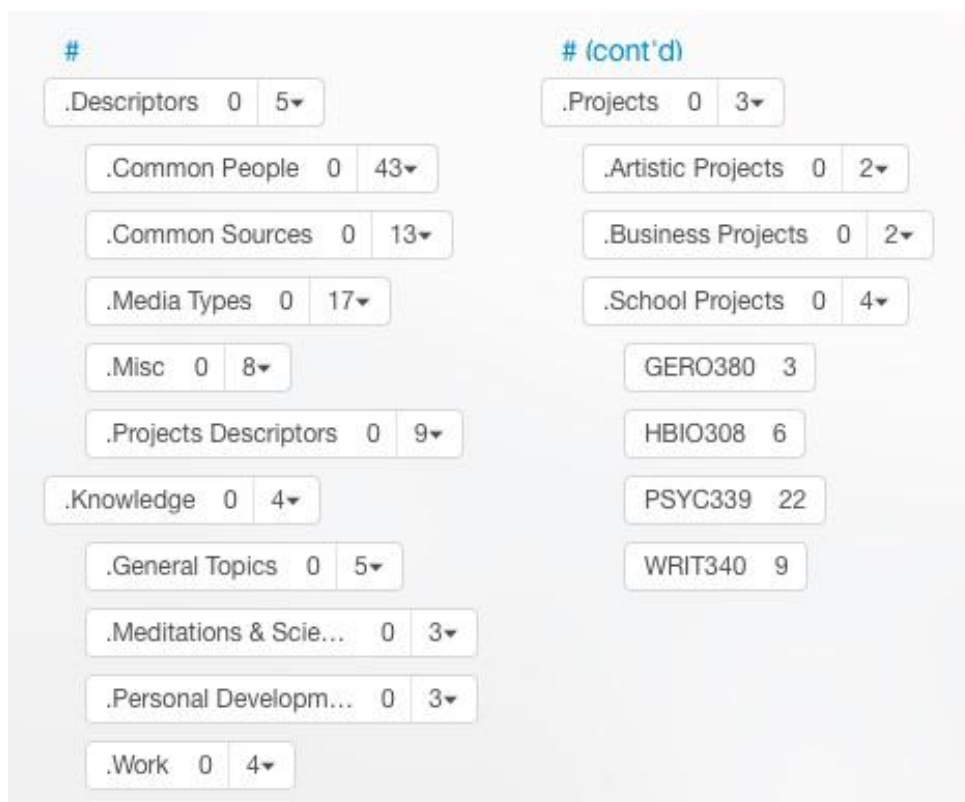


Másfél dimenziós megjelenítés (Protégé) [6]

Evernote

A fastruktúra többszintű hierarchikusságát és a címkézés többszörös kategorizálását kombinálja az Evernote címkézési rendszere. A címkéket egy fastruktúrába lehet rendezni, és egy bejegyzés pedig több címke alá is tartozhat.

Az Evernote címkézési megoldása különösen releváns ennek a dolgozatnak a témájában, mert annak az ontológiának: a többszörös hierarchikus kategorizálásnak a megjelenítése, ami egy jelentős felhasználási területe az alkalmazásnak.



Hierarchikus címkézési rendszer az Evernote alkalmazásban. (Forrás: Evernote)

Az Evernote-ban szereplő behúzott lista megoldásnak azt a hiányosságát fogom kiküszöbölni, hogy a többszörös hierarchiának egyszerre csak egyetlen ága szerint lehet szűrni az eredményeket. Ebben a megjelenítésben a felhasználó nem látja, hogy a kiválasztott kategória alá tartozó elemek milyen alternatív kategorizálási hierarchiák szerint lehet tovább szűrni.

Kognitív torzítások

A felhasználói felületek tervezésében számos különböző pszichológiai jelenséget lehet figyelembe venni. Ezek a szempontok hasznosak lehetnek, hogy a felhasználó minél hatékonyabban tudja értelmezni és kezelni a megjelenő adatokat és elemeket. [3]

Lista-hosszúság effektus

A lista-hosszúság effektus egy sokak számára nyilvánvaló jelenség, és mégis sokszor figyelmen kívül hagyjuk az alkalmazások tervezése során. Ez az effektus két jelentős hatásban mutatkozik meg. Az egyik, hogy a túlságosan hosszú listák elemeit a felhasználó kevésbé tudja megjegyezni. A másik negatív következménye a túlságosan hosszú listáknak, hogy megnöveli a felhasználóra terhelődő kognitív terhet, ami rossz felhasználói élményt eredményez.

Emlékeztető-függő felejtés

Az emlékeztető-függő felejtés, vagy más néven felidézési hiba az a jelenség, amikor valamilyen emlékeztető jelzés hiányában nem tudunk felidézni egy ismeretet. Egy felhasználói alkalmazás kontextusában ezt az elvet arra lehet használni, hogy emlékeztessük a felhasználót az alkalmazás által nyújtott lehetőségekre.

A felhasználói szándék

Amikor a felhasználó nagy mennyiségű információ között szeretné megtalálni, amit keres, a böngészés és keresés hatékonyságát korlátozó tényezők között lehet az egyértelműség hiánya, a túl sok információ, a különböző információk közötti prioritásbeli különbség és az időhiány. [3]

Ezeket a tényezőket akkor tudja hatékonyan kezelni az alkalmazás, ha képes reagálni a felhasználó által kifejezett keresési szándéokra.

Explicit és implicit felhasználói szándék

A felhasználói szándék típusa szerint lehet explicit, amiről a felhasználó a priori tudja, hogy tudja, és implicit szándék, amit az alkalmazás által megjelenített releváns lehetőségek közötti választással tud kifejezni. [4]

Az explicit felhasználói szándék kifejezésére tipikusan egy keresőmező adhat lehetőséget, ahol a felhasználó saját indíttatásából megadhat valamilyen kereső

kifejezést. Az implicit felhasználói szándék felismerése a komplex kapcsolatokat tartalmazó adatok esetén egy különleges kihívást jelent, mert az adatelemek közötti elvi és szemantikai kapcsolatok jelentésén múlik.

A keresőmezők egy interaktív fajtája a szemantikai alapú automatikus kiegészítést adó szövegmező. Amikor a felhasználó beír valamilyen szöveget a szövegmezőbe, a rendszer kikeresi azokat az ontológiai fogalmakat, amik a legjobban megfelelnek a bevitt szövegnek.

A legördülő listák szintén segítenek a felhasználónak, hogy olyan lehetőséget tudjon kiválasztani, amit nem tudna magától, mint explicit felhasználói szándékot kifejezni, hanem a kontextustól függő lista elemeiből interaktív módon választhat.

[8]

Célkitűzés

Amikor a felhasználó a keresési szándék kifejezéséként kiválaszt az irányított gráf egy csúcsa által reprezentált elemet, akkor feltételezhető, hogy olyasvalamit keres, ami az irányított élek mentén elérhető az illető elemből, vagyis a gráf csúcsának az utódjai között szerepel. A használt alkalmazás felsorolhatja a kiválasztott csúcs közvetlen utódjait, mint szűkítési lehetőségeket.

Az irányított gráf szerkezete azonban ezen túlmenően további szűkítési lehetőségek felajánlását is lehetővé teszi. A kiválasztott elem utódai olyan elemeknek is utódai lehetnek, vagyis közvetve olyan elemek alá is tartozhatnak, amik nem utódai a kiválasztott elemnek. Ezek közül az alternatív kategóriák közül, mint szűrő elemek közül választva a felhasználó lehetővé teheti, hogy az alkalmazás leszűrje a releváns részgráfot azokra az elemekre, amik az összes kiválasztott elem utódjai.

Az alábbiakban egy olyan alkalmazás fejlesztését fogom bemutatni, ami lehetővé teszi ezt a több szempont szerinti szűrést.

Egy egyszerű feljegyzés-rendszerező alkalmazást készítettem az irányított gráf felhasználói navigálásának demonstrálására. Az egyes elemek egy címből és egy szöveges tartalmi részből állnak, valamint egy URL mezőből, amivel például egy weboldalt lehet kapcsolni az elemhez.

A cél egy olyan alkalmazás készítése volt, ami minimális infrastruktúrát igényel, és nagyobb adatmennyiség esetén is elég gyorsan el tudja végezni a szükséges számításokat, hogy a böngészés során azonnal reagáljon a felhasználói inputra.

Felhasznált technológiák

Az alkalmazás megvalósításhoz szükséges valamilyen felhasználói felület, és szükség van egy adatbázis megoldásra, ami megőrzi az adatelemeket az alkalmazás újraindításai között.

Technológia kiválasztása

HTML és Javascript böngészőben vs. JavaFX

A felhasználói felülethez két technológiát hasonlítottam össze. Az egyik a böngészőn belüli HTML technológia, a másik pedig a JavaFX, ami asztali alkalmazások fejlesztését teszi lehetővé.

Az alkalmazás megvalósításához a JavaFX mellett döntöttem, különös tekintettel az alábbi szempontokra:

1. A JavaFX mellett szól a jobb együttműködés a helyi operációs rendszerrel, ami a lokálisan tárolt adatok integrálásában lehet jelentős.
2. A JVM-nek jobb a számítási teljesítménye, mint a böngészőnek, ami sok adat esetén elméletileg számíthat az összetett gráfműveletek eredményének gyors megjelenése szempontjából.
3. a JavaFX használatában sokkal több gyakorlatom van

Helyi adatbázis vs. szerver/kliens architektúra

Az adatbázis tekintetében azt kellett eldöntenem, hogy helyi adatbázist használok, vagy pedig egy távoli szerveren tárolom az adatbázist. A szerver/kliens rendszer lehetővé tenné azt is, hogy az alkalmazás működésének a bonyolultabb logikája a szerveren fusson, és csak egy vékony kliens fut a felhasználó számítógépén.

Az alábbi szempontok alapján a helyi adatbázis mellett döntöttem, amihez a H2 adatbázis-kezelő rendszert használom.

1. a helyi adatbázis használata kevesebb adminisztratív munkával jár
2. offline is használható
3. gyorsabb hozzáférést biztosít az adatokhoz. Ez különösen fontos lehet, amikor nagy méretű adathalmazokon kell összetett gráfműveleteket végrehajtani.

FXML

A JavaFX felhasználói felületek grafikus elemeinek a meghatározására két fő módszert biztosít a UI keretrendszer. Az egyik egy procedurális UI-definiálás, ami azt jelenti, hogy lépésről lépésre, időrendben adom meg a komponenseket felépítő utasításokat.

```
final Label label = new Label();
final Button button = new Button(buttonText);
button.setTextFill(buttonTextFill);
button.setPadding(new Insets( top: 0, right: 2, bottom: 0, left: 2));
final HBox hBox = new HBox();
hBox.setSpacing(10);
label.setPrefWidth(0);
label.setMaxWidth(Double.MAX_VALUE);
HBox.setHgrow(label, Priority.ALWAYS);
hBox.getChildren().addAll(label, button);
```

Grafikus komponens procedurális meghatározása (részlet az alkalmazásból)

A JavaFX által támogatott másik megközelítés az FXML struktúra használata, ami lehetővé teszi, hogy deklaratív módon, a felhasználói felület elemeinek térbeli elrendezését követve határozzuk meg.

```

<HBox spacing="5"
    alignment="BASELINE_CENTER">
    <Button fx:id="deleteButton"
        text="X"
        onAction="#deleteArticle">
        <tooltip><Tooltip text="Delete article" /></tooltip>
    </Button>
    <TextField fx:id="uriField"
        HBox.hgrow="ALWAYS"
        promptText="Web URL or local file path"/>
    <Button fx:id="goButton"
        text="->"
        onAction="#go">
        <tooltip><Tooltip text="Open external content" /></tooltip>
    </Button>
</HBox>

```

Grafikus komponensek deklaratív megadása (részlet az alkalmazásból)

Ez a megközelítés egy átláthatóbb és kifejezőbb definiálást tesz lehetővé, ami előnyös az olvashatóság szempontjából, és a későbbi továbbfejlesztés, hibajavítás és karbantartás tekintetében is.

Ezért az alkalmazás grafikailag összetettebb részeihez FXML-t használtam.

CSS

A grafikus elemek megjelenésének a meghatározásához CSS-t használtam, aminek az az előnye, hogy el lehet különíteni a programkódot és az egyes grafikus elemek megjelenését meghatározó kódot.

Így a megjelenés megváltoztatása sokkal egyszerűbb, és jobb az alkalmazás karbantarthatósága.

```

#parentsLabel {
    -fx-background-color: white;
    -fx-background-radius: 0.5em;
    -fx-background-insets: 0.1em;
    -fx-padding: 0 0.2em 0 0.2em;
    -fx-border-width: 0.15em;
    -fx-border-color: black;
    -fx-border-radius: 0.5em;
}

```

CSS stílus kód egy grafikus elem megjelenésének meghatározására

Követelmények

Az alkalmazás egy általános feljegyzés-kezelő program, és mint ilyen, meg kell felelnie bizonyos alapvető funkcionális követelményeknek.

Lehetővé kell tennie, hogy a felhasználó a feljegyzések között **egyszerű keresést** hajtson végre. A keresett bejegyzés címének, vagy a cím részletének az ismeretében a felhasználó részéről a priori rendelkezésre álló explicit keresési szándék kifejezésére gyors és kimerítő eredményeket kell adnia.

Az egyes találati eredmények **tartalmát meg kell jelenítenie**, a kapcsolódó szöveges mezőkkel. Tárolnia kell és meg kell jeleníteni a bejegyzés címét, a bejegyzés szöveges törzsét, és a bejegyzéshez tartozó opcionális URL-t, ami helyi fájlt vagy online tartalmat tud azonosítani.

Az adatelemeket egy irányított gráfban kell tárolnia, és meg kell jelenítenie a gráf **egyszerű navigálására** alkalmas interaktív grafikus elemeket. Ehhez a felhasználó számára fel kell sorolni az elemek közötti kapcsolatok iránya szerinti további elemeket, amiket kiválasztva a meg kell jelenítenie a kiválasztott adatelem tartalmát.

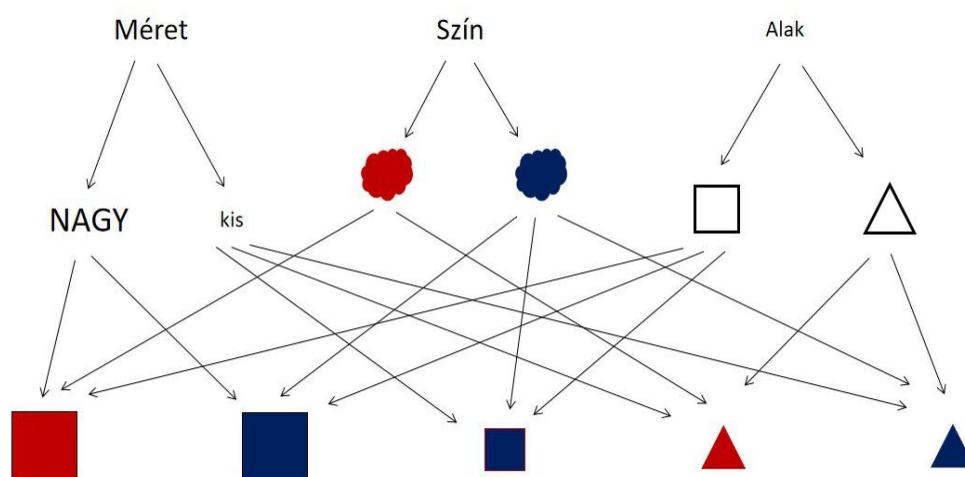
Ezek az egyszerű funkciókon túlmenően az alkalmazásnak meg kell tudnia találni azokat az adatelemeket, amik a kiválasztott elem közvetett leszármazottjainak a halmazát **alternatív szempontok szerint** tovább tudják **szűkíteni**.

Tervezés

Az alkalmazás tervezése két fő részből áll. Az egyik a komplex gráfműveletek elvégzésére alkalmas keretrendszer megtervezése, a másik pedig a grafikus felhasználói felület megjelenésének és interaktív funkcionalitásának a megtervezése.

Komplex gráfműveleteket biztosító keretrendszer megtervezése

Az alábbi ábrán egy irányított gráf látható, ami az elemek hierarchikus alárendeltségi viszonyának az ontológiáját reprezentálja. Ez a fajta többszörös kategorizálás egy tipikus példája annak a probléma-térnek, amin az alkalmazás által végrehajtott gráfműveleteknek a felhasználó számára releváns eredményeket kell adniuk.



Többszörös hierarchikus kategorizálás irányított gráfban

Az alkalmazás egy irányított gráfban tárolt adatok közötti böngészést tesz lehetővé, és ennek érdekében a keretrendszernek lehetővé kell tennie, hogy az alkalmazás-programozó a gráfműveletek végeredményével tudjon dolgozni.

Ennek megvalósítására azt a megoldást választottam, hogy a gráfokat és részgráfokat egy olyan típusú objektumban tárolom, ami lehetővé teszi a benne tárolt gráfon való műveletek végrehajtását, amik a különböző eredménylistákban megjelenítendő elemek listáját adják eredményül.



A gráfokat és részgráfokat tároló objektumtípus a használt gráfműveletekkel.

Az eredményül kapott részgráfon a alkalmazás grafikus része a `getParents()` és `getChildren()` eljárások meghívásával tudja kiolvasni a az aktuálisan fókuszban lévő elem közvetlen őseit és közvetlen leszármazottjait.

A `setFocusArticle()` eljárás lehetővé teszi, hogy az alkalmazás a felhasználói bemenetnek megfelelően változtassa az aktuálisan fókuszban lévő

adatelemet. Így, a Graph típusú objektum állapotának a megváltoztatásával dinamikusán megváltozzon a gráfműveletek kontextusa.

A negyedik publikus eljárás ezen a típuson az `alternativeOrphans()` metódus, aminek az a funkciója, hogy kikeresse az aktuálisan fókuszban lévő szűrőelemek függvényében a közös leszármazottak ősei közül azokat az elemeket, amiknek nincsenek őseik.

Az adatelemek közötti kapcsolatokat reprezentáló adatstruktúra megtervezése

A dolgozat tárgyát képező alkalmazásban a használt adatstruktúra nem próbálja követni a szemantikai kapcsolatok komplexitását, hanem egyetlen kapcsolatra: a gráf irányított élével reprezentált hierarchikus viszonyra redukálja azt; és a megjelenítésben nem támaszkodom egynél több dimenzióra, hanem kizárólag egyszerű listákban jelenítem meg az adatelemeket.

Az irányított gráf éleit a legátláthatóbb módon úgy tudom kifejezni, ha az irányított gráf csúcsait alkotó adatelemek tárolják a közvetlen elődeik és a közvetlen utódaik listáját. Ez az egyetlen objektumtípus így lehetővé teszi, hogy egy egységes és jól elkülönülő interfészen keresztül tudjam használni az adatstruktúrát.

Article
<ul style="list-style-type: none"> - SimpleIntegerProperty id - SimpleStringProperty title - SimpleStringProperty content - SimpleStringProperty uri - static Map<Integer, Article> articleCache - static Map<Integer, Set<Article>> parentsCache - static Map<Integer, Set<Article>> childrenCache
<ul style="list-style-type: none"> + Article(int id, String title, String content, String uri) + int getId() + void setId(int id) + String getTitle() + void setTitle(String title) + SimpleStringProperty getTitleProperty() + String getContent() + void setContent(String content) + SimpleStringProperty getContentProperty() + String getUri() + void setUri(String uri) + Set<Article> getParents() + Set<Article> getChildren() + void addParent(Article parent) + void addChild(Article child) + void removeParent(Article parent) + void removeChild(Article child) + void delete() + void save()

Az irányított gráf csúcsait reprezentáló adattípus

Ezek az objektumok képezik az adatmodellt az alkalmazásban, jól elkülönítve grafikai megjelenítést az adat-specifikus műveletektől.

Statikus mezőként tartalmaz egy gyorsítótár megoldást is, aminek a segítségével elkerülhető, hogy a gráfműveletek végrehajtása során szükségtelenül használja az adatbázist. Ez a gyorsítás azért is fontos, hogy a felhasználó azonnali eredményeket kapjon a navigáció során.

A felhasználói felület tervezése

A navigációs elemeknek lehetővé kell tenniük, hogy a felhasználó egy felfedező viszonyulást alakítson ki az ismeretek tekintetében. Ebből a szempontból különösen fontos, hogy a navigációs elemek segítségével fel tudja deríteni az adatelemek közötti asszociációs kapcsolatokat. [7]

A bejegyzések listáját megjelenítő grafikus elemek hierarchiája

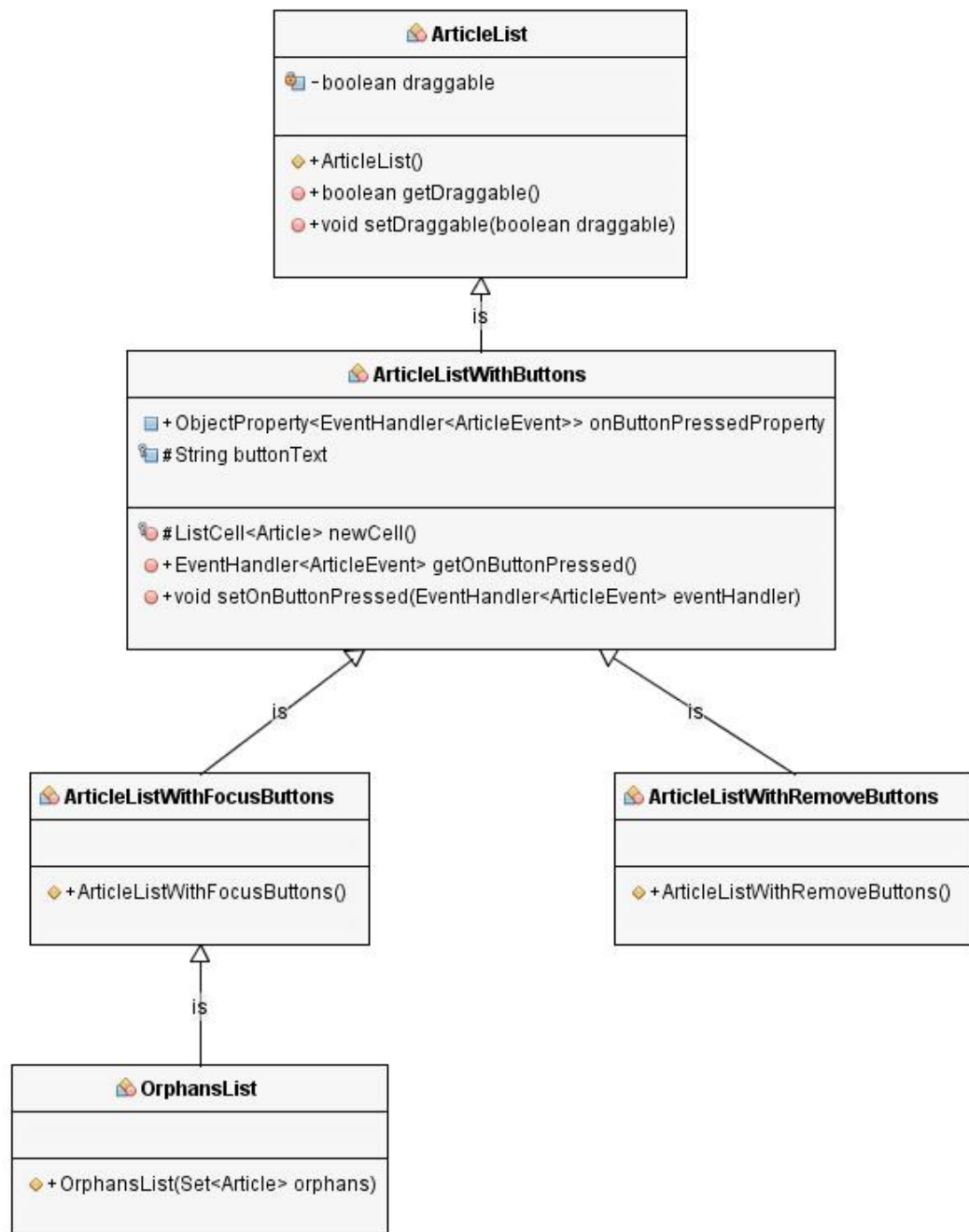
A böngészés során a felhasználó az eredményeket különböző típusú listákon keresztül tudja böngészni. Ezek között a grafikus elemek között vannak hasonlóságok, amik szerint egy négy szintű osztályhierarchiában terveztem meg ezeket a komponenseket.

ArticleList: Ez az eredménylisták legáltalánosabb típusa, ami lehetővé teszi, hogy a felhasználó drag-and-drop módszerrel tudja másolni az elemeket a listák között, és így adja meg az új kapcsolatokat.

Az **ArticleListWithButtons** típusú listák lehetővé teszik, hogy az alkalmazás valamilyen gombot is megjelenítsen az egyes elemek sorában, a szükséges funkcióknak megfelelően.

Ennek két altípusa az **ArticleListWithFocusButtons** és az **ArticleListWithRemoveButtons**, amik az részgráfok szintjei közötti navigálást, illetve az elemek közötti kapcsolatok törlését teszik

lehetővé.



Az eredménylistákat megjelenítő grafikus elemek hierarchiája

Megvalósítás

A megvalósítás során az IntelliJ IDEA fejlesztői környezetet használtam, és az alkalmazás forráskódja elérhető a <https://github.com/SoltiGabor/Cetli> alatt található IntelliJ IDEA projectben. [2]

Helyi adatbázis

Az adatbázis-műveleteket elválasztottam egy osztály interfésze mögé, hogy jól elkülönüljön a program többi részétől.



Az adatbázist kezelő interfész

Ez az elkülönítés lehetővé teszi, hogy a program többi részét az adatbázis-kezelés implementációjától függetlenül lehessen fejleszteni.

Az adatbázis-műveletek implementálása során közvetlen SQL kódot használtam, a Java JDBC (Java Database Connectivity) API-n keresztül.

```

public final void createTables() throws SQLException {
    connection.createStatement().execute(
        sql: "CREATE TABLE IF NOT EXISTS ARTICLES("
        + "ID INT AUTO_INCREMENT, "
        + "TITLE VARCHAR, "
        + "CONTENT VARCHAR, "
        + "URI VARCHAR, "
        + "PRIMARY KEY (ID))"
    );

    connection.createStatement().execute(
        sql: "CREATE TABLE IF NOT EXISTS RELATIONSHIPS("
        + "ID INT AUTO_INCREMENT, "
        + "PARENT INT, "
        + "CHILD INT, "
        + "PRIMARY KEY (ID))"
    );
}

```

Az adatbázis táblák definiálása a Java JDBC használatával

A gráfműveletek megvalósítása

A gráfműveletek megvalósítása során rekurzív mélységi bejárásokat használtam.
[1]

Egy elem közvetett őseinek a kikeresése során például az algoritmus végigiterál az elem közvetlen ősein, és mindegyikre vonatkozólag meghívja az eljárást.

```

private Set<T> ancestors(T a) { return collectAncestors(new HashSet<T>(), a); }

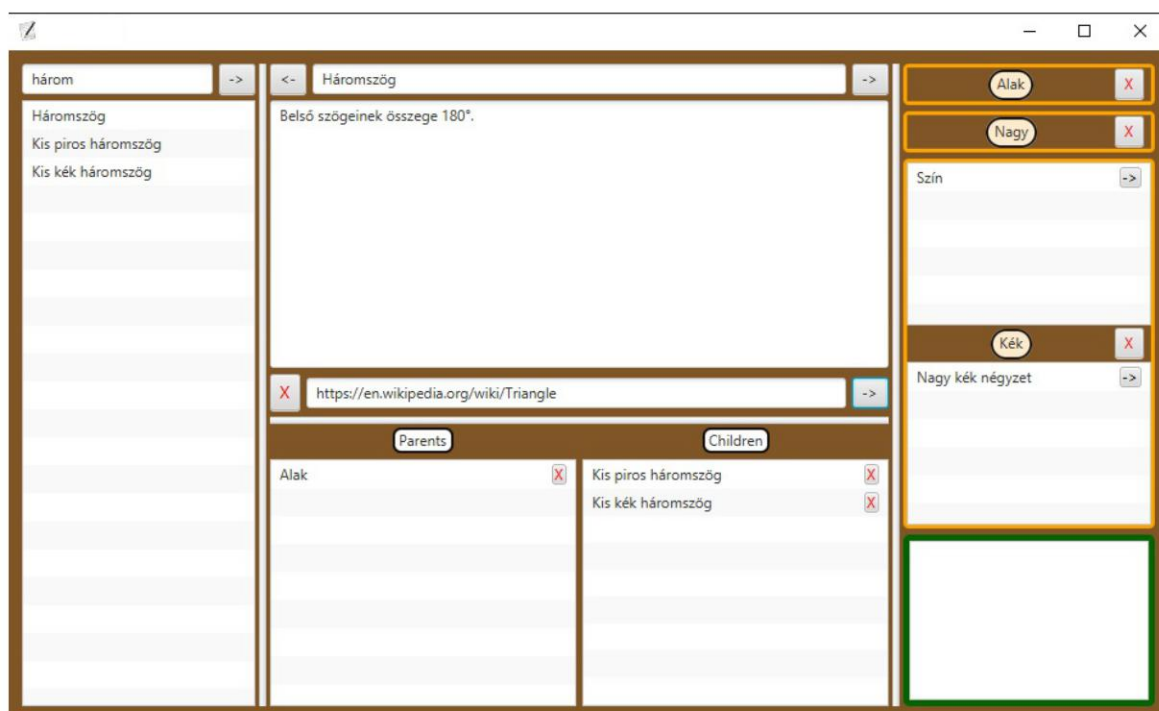
private Set<T> collectAncestors(Set<T> collected, T article) {
    Set<T> parents = getParents(article);
    if (parents.isEmpty()) return collected;
    for (T p: parents) {
        if (collected.contains(p)) continue;
        collected.add(p);
        collected.addAll(collectAncestors(collected, p));
    }
    return collected;
}

```

Közvetett ősök összegyűjtése a körök kiküszöbölésével

Az egyszerű rekurzív mélységi bejárás általános irányított gráfok esetén végtelen ciklushoz vezethet. Ennek az elkerülése érdekében minden új találatot összehasonlítok az addig összegyűjtött elemek halmazával. Ehhez egy hasító halmazt használtam, mert a hasító halmaz esetén nagyon gyors művelettel lehet ellenőrizni, hogy benne van-e egy elem a halmazban.

Felhasználói felület felépítése



A felhasználói felület felépítése

Címrészlet szerinti keresés

A grafikus felhasználói felület három részből áll, amik közül az első egy egyszerű keresőmező. Kifestázza azokat az elemeket, amiknek a címében megtalálható a megadott keresőkifejezés. Ez lehetővé teszi, hogy a keresés első fázisában a felhasználó explicit keresési szándéka alapján leszűkítsük az eredményt, és erre támaszkodva az alternatív szűkítési szempontok dinamikus felajánlására épülő böngészés hamarabb vezessen eredményre.

Egyszerű navigációs és tartalomkezelő panel

A felhasználó felület második része az egyes elemek megjelenítésére szolgál. Felül található a cím, alatta a szöveges tartalom és a kapcsolódó URL vagy lokális elérési út, legalul pedig a közvetlen elődök és közvetlen utódok listája, ami lehetővé teszi az irányított gráf egyszerű navigálását.

Összetett szűrőpanel

A felhasználói felület harmadik része az összetett szűrőpanel, aminek a felső részében egymás alatt sorban találhatóak a kiválasztott szűrő elemek, amiket a felettük lévő elemek szerint szűrt részgráfon belül lehet szűkíteni vagy bővíteni. Az alsó részen pedig a felajánlott alternatív szűrési lehetőségek megjelenítésére szolgáló lista szerepel.

A szűrőpanel működését részletesen bemutatom “Az alkalmazás működése” fejezetben.

Vissza gomb és előzmények

A címsor mellé elhelyeztem egy vissza gombot is, ami egy pop-up panelben megjeleníti a korábban kiválasztott bejegyzések listáját is, hogy ezzel segítse a böngészést.

Natív integráció

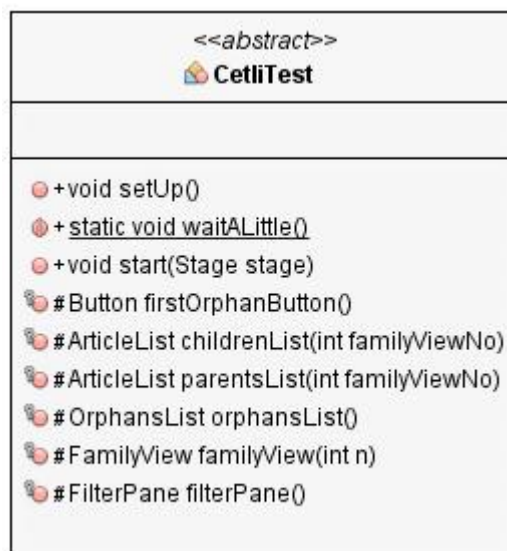
Drag-and-drop módszerrel be lehet húzni internetes oldalakat a böngészőből az alkalmazásba, ami létrehoz egy új bejegyzést, és automatikusan kitölti a cím és az URL mezőket. Az URL melletti gombra kattintva az oldal megnyílik a böngészőben. Ez a funkció nagyon megkönnyíti a rendezett web-link gyűjtemények készítését.

Automatikus tesztek készítése

Az automatikus tesztek készítéséhez a TestFX keretrendszert használtam, ami lehetővé teszi, hogy a felhasználói bemeneteket szimulálva egyszerre lehessen tesztelni a felhasználói felületet és a funkcionális logikát.

A tesztekhez készítettem egy teszt-adatbázist, ami olyan adatelemekből áll, amik különösen alkalmasak az alternatív szűrőelemek és szűrési eredmények helyes megjelenítésének az ellenőrzésére.

Az egyes tesztek egy közös absztrakt teszt-osztályból örököltetem, amibe összegyűjtöttem azokat a segédeljárásokat, amiknek a segítségével az egyes tesztek azonosítani tudják a grafikus felhasználói felület komponenseit.



Absztrakt tesztosztály az egyes tesztekhez szükséges közös segédeljárásokkal

Az automatikus tesztek három fő részből állnak.

Az első rész megjeleníti a teljes alkalmazást, és leellenőrzi az egyszerű keresőmező helyes működését. Ezt követően kiválasztja a keresés eredményét, és aktiválja az alternatív szűrőelemek megjelenítését. Miután az alkalmazás végrehajtja a szükséges gráfműveleteket, a kapott eredmény alapján a teszt ellenőrzi, hogy a vártnak megfelelő jelenik-e meg a képernyőn.

A második rész az irányított gráf egyszerű navigálását lehetővé tevő összetett navigációs komponens adaptív átméretezését ellenőrzi.

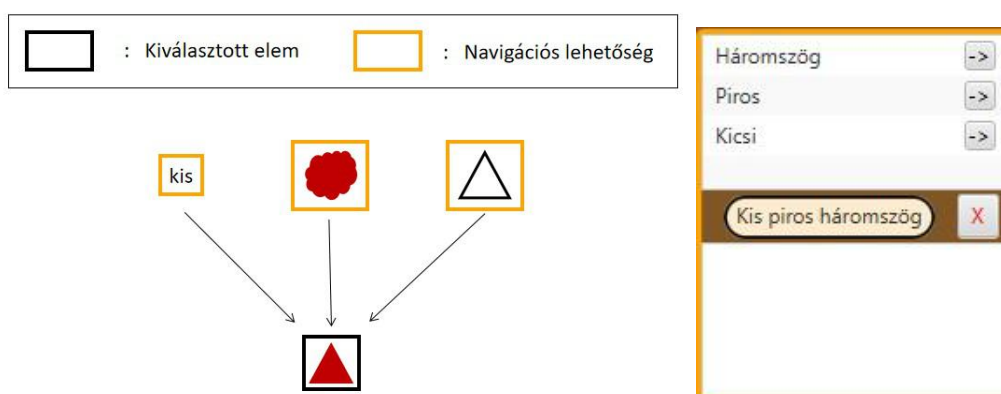
A harmadik rész a komplex szűrőelemek és részgráfok közötti navigálásra szolgáló komponenseket jeleníti meg, és a kiválasztott elem helyes megjelenítése mellett ellenőrzi azt is, hogy helyes számú alternatív szűrőelemek jelennek-e meg a listában.

Az alkalmazás működése

A kész alkalmazás különböző szemantikai viszonyokat reprezentáló irányított gráfok navigálását teszi lehetővé. A lehetséges alkalmazási területekre az Értékelés fejezetben fogok kitérni.

Az alkalmazás működésének a demonstrálásához a Tervezés fejezetben bevezetett leegyszerűsített példát fogom használni, ami különböző alakú, méretű és színű alakzatok többszörös kategorizálását veszi alapul.

A felhasználói böngészés során az irányított gráf lehetőséget ad arra, hogy az alkalmazás segítsen általánosítani a keresést a kiválasztott elem közvetlen elődeinek felsorolásával.

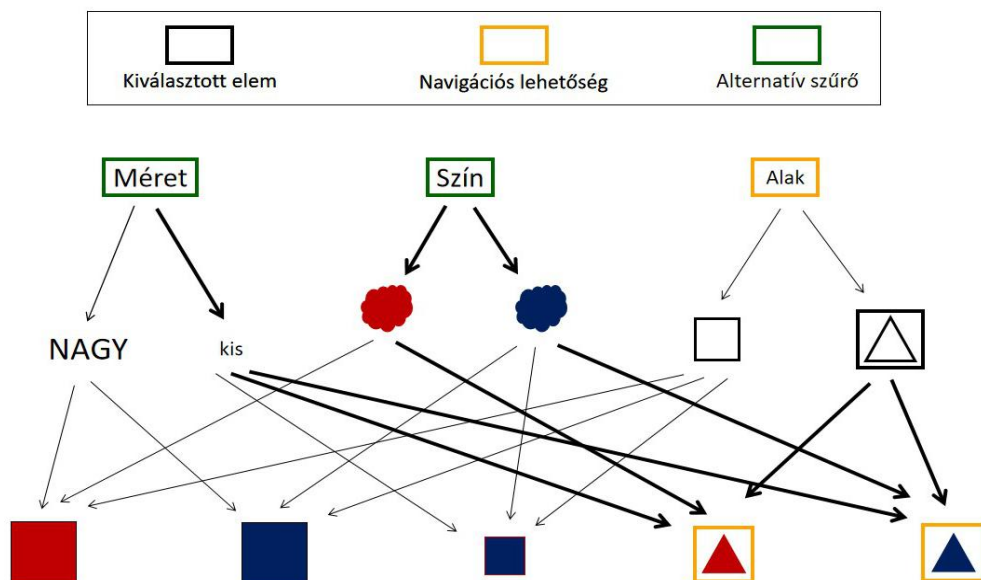


Ez a felsorolás olyan általánosítási lehetőségeket mutat a felhasználónak, amiből felismerheti például, hogy általában a háromszögekre kíváncsi. A “Háromszög” elem kiválasztása után megjelenik az összes háromszög.



Miután a felhasználó kiválasztotta a “Háromszög” elemet, az alkalmazás egyszerű halmazműveletek összetételével meg tudja találni, hogy milyen alternatív szempontok alapján lehet tovább szűrni a találatokat.

Ha vesszük a kiválasztott elem összes utódjának az összes elődjét, és az így kapott elemek, valamint a köztük lévő élek által alkotott részgráfnak megkeressük az nulla be-fokú elemeit, megkapjuk a potenciálisan releváns alternatív szűrő elemeket.



Alak ->

Háromszög X

Kis piros háromszög ->

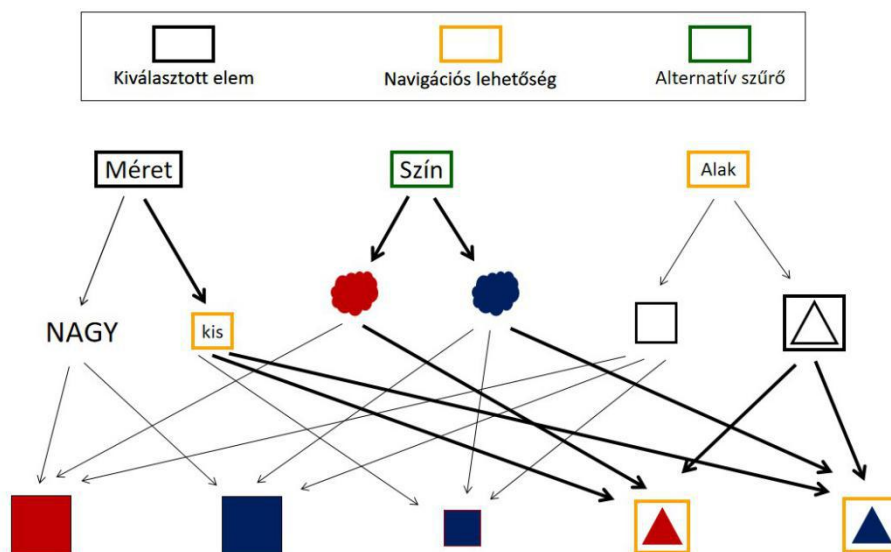
Kis kék háromszög ->

Szín ->

Méret ->

Az alternatív szűrők felsorolásából a felhasználó konkrét ötleteket kap, hogy milyen további szempontok lehetnek számára érdekesek. Ha például úgy dönt, hogy fontos a méret, akkor kiválaszthatja azt a szempontot.

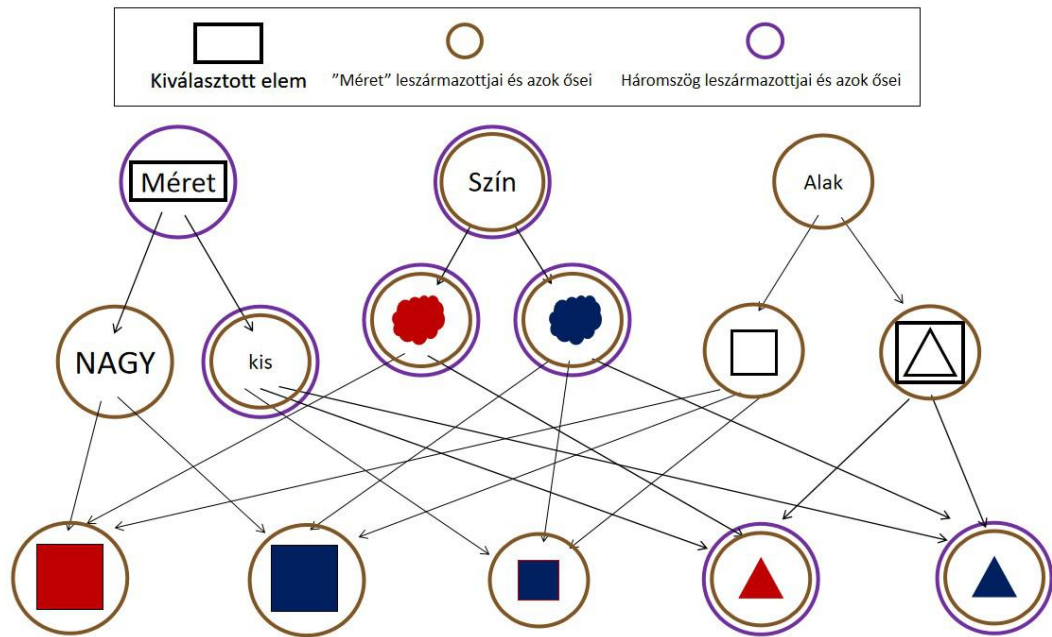
Mivel azt már kiválasztotta, hogy a háromszögek érdeklik, és az összes háromszög kicsi, a “Méret” szűkítési lehetőségeiként csak a kis méret jelenik meg.



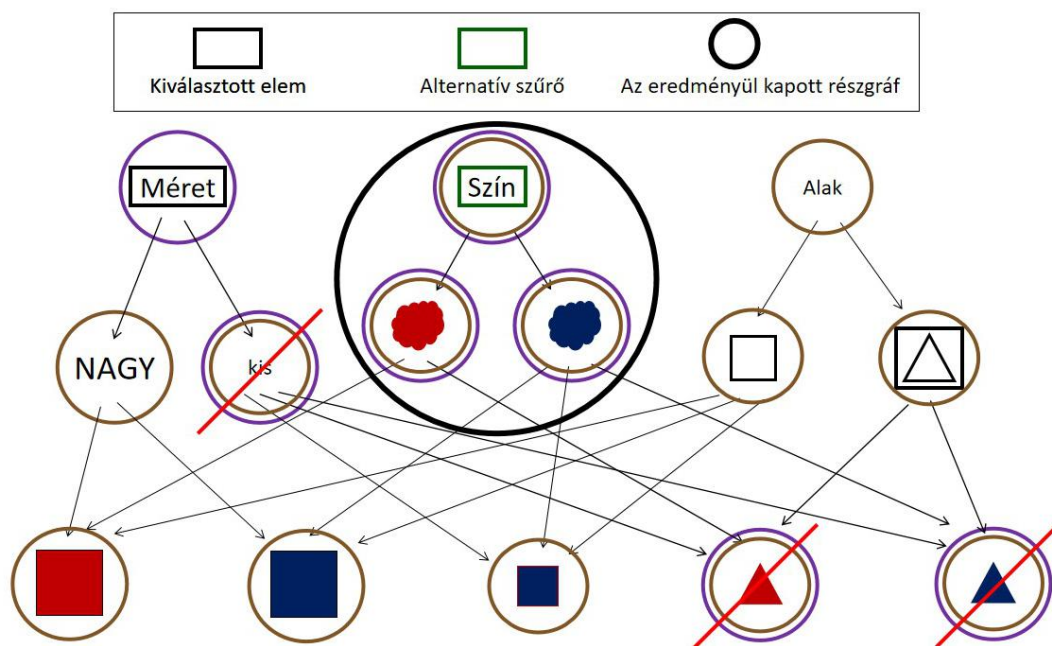
Az alternatív szűrési szempontok kiválasztásának általános módja

Ezen a ponton két kiválasztott elem van: a méret és a háromszög. Vagyis azok a további szűrési szempontok lehetnek érdekesek, amik alá ezeknek az utódjai tartoznak.

Vesszük az egyes kiválasztott csúcsok utódjait és ezen utódok elődjait, és aztán vesszük az így kapott részgráfok metszetét. Ezzel megkapjuk az összes olyan elem halmazát, amik az összes kiválasztott elem utódjainak a szűrésében érdekesek lehetnek.



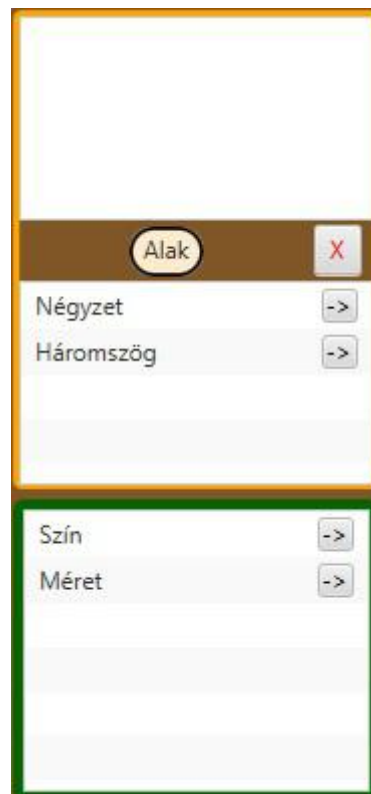
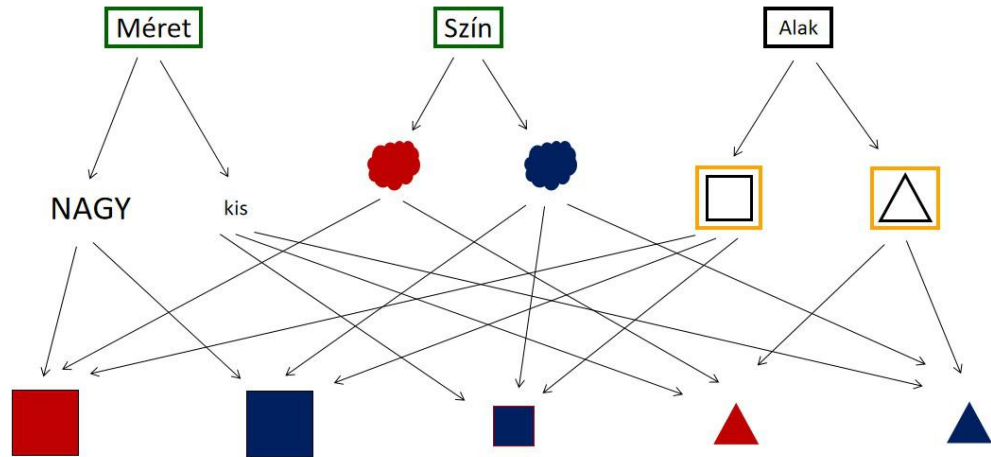
Ezután kivonjuk ebből az metszetről a kiválasztott elemek elődjait és utódjait. Erre azért van szükség, mert a kiválasztott elemek utódjaihoz és elődjeihez már eleve el lehet jutni a kiválasztás egyszerű szűkítésével és bővítésével. Mi viszont azokat az elemeket keressük, amik alternatívák a kiválasztott szűrőelemekhez képest.



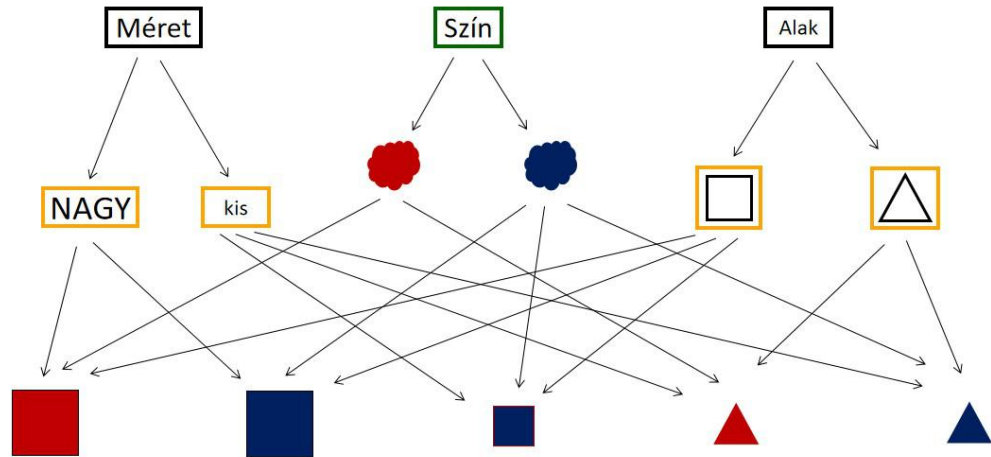
Az eredményül kapott részgráf nulla be-fokú elemei lesznek az alkalmazás által felajánlott alternatív szűrőelemek. Ezek a legáltalánosabb pozícióban lévő olyan elemek, amikből az irányított élek mentén el lehet érni a kiválasztott elemek valamely közös utódját.

További felhasználói navigálás a szűrési eredmények között

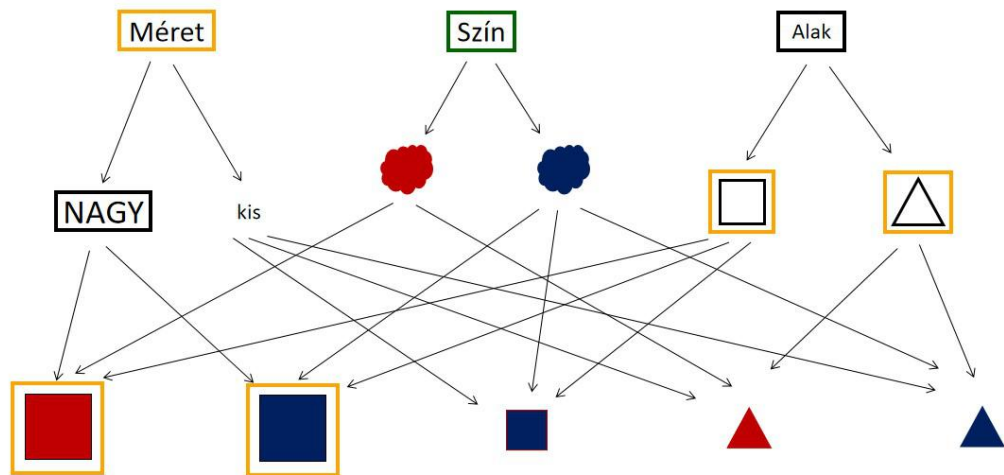
Miután a felhasználó a méret szerinti szűkítési lehetőség alapján látja, hogy csak kis háromszögek vannak, úgy dönthet, hogy a háromszögek között nem fogja megtalálni, amit keres, és alak szerint általánosítja a keresést.



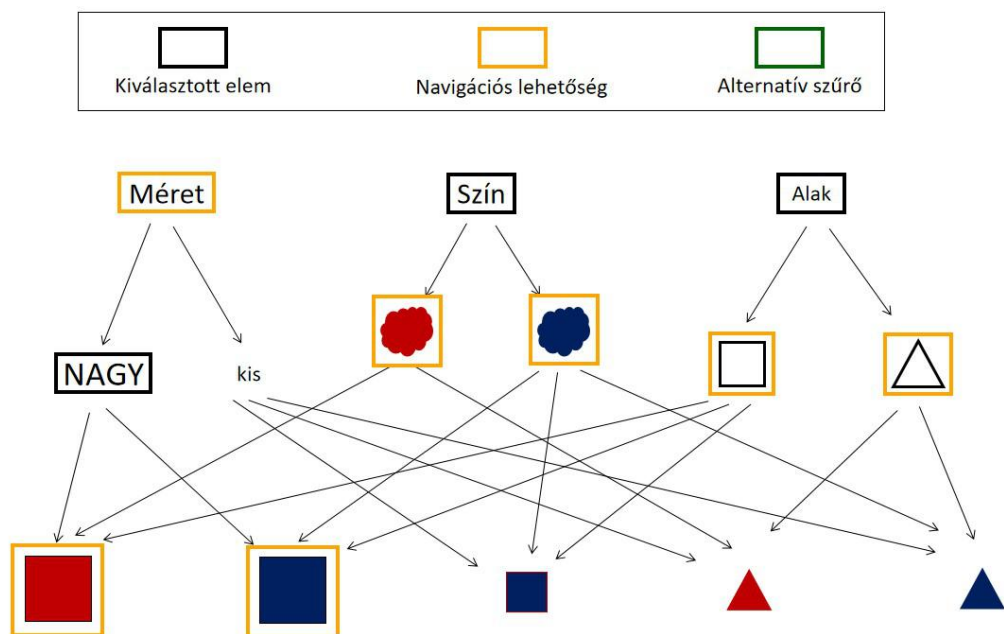
Ezek után a méret szerinti alternatív szűrést választva már megjelenik a “Nagy” lehetőség is, vagyis a keresési szempont általánosításához dinamikusan hozzáigazodik a felajánlott alternatív szűrési lehetőségek listája.



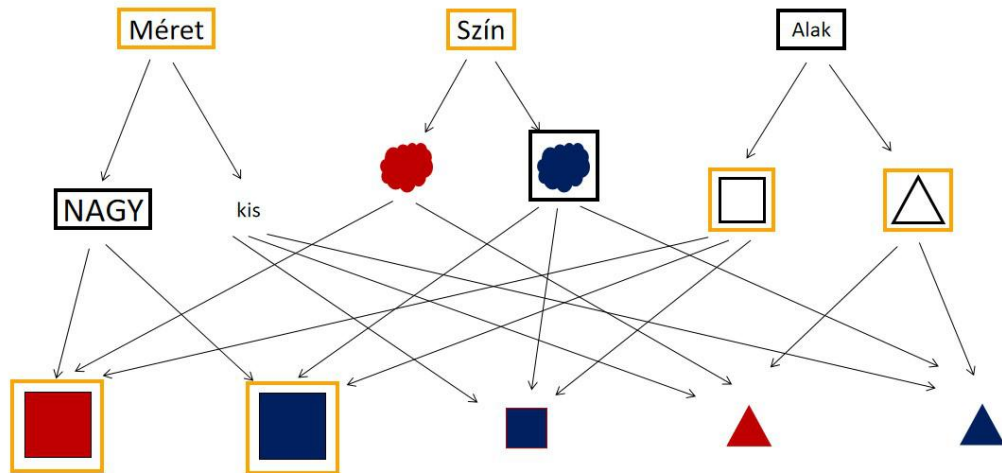
Mivel a felhasználó az eddigi böngészés alapján rájött, hogy valójában valami nagy méretű dolgot keres, kiválasztja azt, hogy “Nagy”:



Lehet, hogy ezek alapján a felhasználó megtalálja, amit keres a “Nagy” elem közvetlen utódai között. De ha ez a lista nagyon sok elemet tartalmaz, akkor lehet, hogy könnyebben tud választani a felajánlott alternatív szűrési szempontok közül. Kiválasztja az alternatív szempontként megjelent “Szín” elemet:



A megjelenő színek közül kiválasztja azt, hogy “Kék”, és megjelenik a nagy kék négyzet, amit a felhasználó keresett. Mivel a találatok szín és méret szerint is leszűrve találhatók a listában, ez egy rövidebb lista, és könnyebb belőle választani, mint amikor csak a méret szerint szűrt elemek listáját látta.



Alak

X

Nagy

X

Szín

->

Kék

X

Nagy kék négyzet

->

Értékelés

Ezt az alkalmazást kipróbáltam különböző személyes feljegyzések rendezésére, és kielégítőnek találtam a használhatóságát. Alkalmas a többszörösen és hierarchikusan kategorizált bejegyzések olyan szintű visszakereshetőségének a biztosítására, amit más jegyzetelő és jegyzet-rendező alkalmazásokban nem találtam meg.

A használhatósági tesztelés alapján a jelen dolgozat témáját képző megközelítés legsúlyosabb korlátjának azt látom, hogy a címkézés módszeréhez hasonlóan a hiányos kategorizálás félrevezető keresési eredményeket hozhat. Amikor a felhasználó kiválaszt bizonyos szűrőelemeket, azt várja, hogy minden olyan adatelemet megtaláljon, amiknek mindegyik szűrő alá kell tartozniuk. Ha az adatbevitel során hiányos volt a vonatkozó kapcsolatok beállítása, vagy utólag új kategóriák kerültek a rendszerbe, amik nem lettek visszamenőlegesen hiánytalanul alkalmazva, akkor a keresés hiányos eredményt fog adni. Ez a korlát a címkézéshez hasonlóan a módszer alapvető sajátja, és nem tudok róla, hogy jelentős mértékben mérsékelni lehetne a hatását a felhasználói felület által nyújtott akármilyen asszisztenciával.

Lehetséges felhasználási területek

Ez a fajta navigációs felület hasznos lehet bármilyen adatelemek felhasználói böngészésére, amik irányított gráfba rendezhetőek, mert valamilyen többszörös kategorizálás értelmezhető rajtuk, vagy valamilyen alárendelt viszony értelmezhető közöttük. Például:

- Weblink gyűjtemény,
- Teendők és feladatok rendszerezése prioritás, projekt, témakör és határidő szerint,
- dolgozat, cikk, tanulmány írása során gyűjtött jegyzetek rendszerezése,
- vagy általános célú személyes feljegyzések kategorizálása.

A megoldás alkalmazhatósági köre

Az alkalmazás személyes használhatósági tesztelése során felismerhetővé váltak bizonyos összefüggések, amik alapján körül lehet írni azoknak a probléma-tereknek a szemantikai természetét, amikre a leginkább megfelelő ez a megoldás.

Az alkalmazás által használt megközelítés nem minden esetben ad hasznos visszajelzéseket a felhasználónak. Az alternatív szűrőelemek relevanciája, amiket az algoritmus megtalál és felkínál a felhasználónak nagyban múlik azon, hogy konkrétan milyen szemantikai kapcsolatokat reprezentálnak az irányított gráf élei.

Mivel a kapcsolatok szemantikai jellege adja a használhatóság szempontjából kritikus tényezőket, nem lehet egzakt módon formalizálható kritériumokat megállapítani.

Azonban van három indikátor, ami alapján valószínűsíteni lehet, hogy egy adott adathalmazon értelmezett irányított viszony a felhasználó számára releváns eredményeket fog adni az alkalmazásban.

1. Pozitív indikátor, ha az irányított gráf **nem tartalmaz nagy köröket**. Mivel az algoritmus egyfajta hierarchikus alárendeltségi viszonyt feltételez, a körök esetén nem tudja kiválasztani az átfogóbb kategóriákat jelentő elemeket.

2. Pozitív indikátor, ha **a gráf számos olyan csúcsot tartalmaz, amiknek nulla a be-foka**. Ezek a csúcsok mintegy főkategóriákként szerepelve tipikusan jól értelmezhető elsődleges választási lehetőségeket nyújthatnak a böngészési eredmények szemantikailag releváns szűrésére.

3. Pozitív indikátor, ha az irányított élek iránya **egyértelműen megfeleltethető egy alárendeltségi viszornak**. Amennyiben az irányított viszony két oldala felcserélhető, és mindkét irányban nevezhető alárendeltségnek,

az azt valószínűsíti, hogy ennek az ontológiának az esetében nem fog hasznos eredményeket adni az alternatív szűrőelemek listája.

A harmadik indikátor hiányának egy példája a szakcikk hivatkozási gráfja. Ez egy olyan irányított gráf, ami eléggé szigorúan mentes a köröktől, és előfordulnak benne olyan csúcsok, amiknek nulla a ki- vagy be-foka. A harmadik indikátor viszont nem jellemző rá, mert nem magától értetődő, hogy a hivatkozó vagy a hivatkozott szakcikk tekinthető alárendeltnek a másikkal képest. Ezért erről az alkalmazási területről az sejthető, hogy nem ideális tárgya ennek a megjelenítési stratégiának.

Ezek az indikátorok túlmenően az alapján lehet megbecsülni a megoldás használhatóságát egy adott problémára, hogy az elemek közötti irányított viszony hasonlít-e a főkategóriák és alkategóriák hierarchikus rendszerére.

Lényegesebb továbbfejlesztési lehetőségek

A közös utódok nulla be-fokú elemeinek megjelenítése

A jelen implementációban a mindenkor releváns eredmények az utolsó szűkítési szempont aktuálisan kiválasztott elemének közvetlen leszármazottjaiként jelennek meg. A legrelevánsabb elemek gyors megtalálását elősegítené, ha az alkalmazás folyamatosan megjelenítené a mindenkor kiválasztott szűrő elemek utódjai metszetének nulla be-fokú elemeit. Ezek volnának ugyanis a legáltalánosabb olyan elemek, amik minden kiválasztott szempont utódjai.

A választási lehetőséget nem nyújtó alternatív szűrőelemek automatikus szűkítése

Ha az alternatív kategóriákat megkereső algoritmus által eredményezett részgráf nulla be-fokú elemeinek csak egy a ki-foka, akkor ezeknek a megjelenítése nem ad közvetlen választási lehetőséget a felhasználónak. Ilyen esetekben az utolsó elem elhagyható lenne, és elég lenne csak azt a közvetlen utódot megjeleníteni, aminek legalább két közvetlen utódja van. Ez a változat növelné a megjelenített szűkítési lehetőség relevanciáját, de a valódi legáltalánosabb elem által nyújtott kontextus hiányában ronthatja az értelmezhetőséget.

A szűrőelemek dinamikus szűkítése és bővítése

Az alkalmazás jelen implementációjában az egyes kiválasztott szűrő elemek nem egyenrangúak, hanem sorrendben egymásra épülnek. Nagyban javíthatná a rugalmas használhatóságot, ha az egyes szempontokat egymástól függetlenül lehetne szűkíteni és bővíteni, és a többi szűrő elem részgráfja dinamikusan

megváltozna annak megfelelően, ahogyan ezek a változtatások megváltoztatják a gráftranszformációk eredményét.

Az adatmodell és a gráfműveletek jobb elkülönítése

Igyekeztem elválasztani a az adatmodellt a megjelenítéstől, de maguk a gráfműveletek közvetlenül a gráfot reprezentáló típuson lévő eljárásokban vannak implementálva. A különböző algoritmusokkal való könnyebb kísérletezés érdekében hasznos lenne elvonatkoztatni az összetett gráfműveleteket funkcionális interfészekbe. Erre a Java programozási nyelv is alkalmas, de még jobb lenne egy olyan programozási nyelvet használni, amiben teljesebb a funkcionális programozás támogatása, mint amilyen például a Kotlin.

A felhasználói élmény fejlesztése

A jelenlegi felhasználói felület a használhatóságon túlmenően nem elégíti ki teljes mértékben a felhasználói élmény minden szempontját. A felhasználói felület még ennél is intuitívabb és informatívabb lehetne, különös tekintettel az alternatív szűrőelemek kiválasztására és bővítésére/szűkítésére, valamint az összetett szűrés eredményének könnyű értelmezhetőségére.

Megosztás

Az alkalmazás egyik lehetséges felhasználási területe a kollaboratív tudástár építése. Ennek a megvalósításához ki lehetne bővíteni az alkalmazást a tartalmak megosztását és közös bővítését lehetővé tevő technológiával.

Ennek a fejlesztési lehetőségnek a tekintetében a legkomolyabb korlátozó tényezőt előreláthatólag a verziókezelés problémája jelentené, mert több felhasználó csak akkor tud együttműködni a tartalmak és kapcsolatok bővítésében, ha jól elkülönülnek a változtatások.

Kategória szerinti keresés

Az alkalmazásban szereplő keresőmező jelen megvalósításban a teljes adatbázisban keres. A keresőmező és a többszörös szempont szerinti szűrés előnyeit tovább tudná egyesíteni az a lehetőség, ha a szűrőelemek szerinti részgráfon belül is lehetne szövegrészletek alapján, kategóriák szerinti keresőmező használatával keresni.

Összefoglalás

A szakdolgozatom célja egy olyan felhasználói alkalmazás megtervezése és megvalósítása volt, ami lehetővé teszi irányított gráfban elosztott adatelemek hatékony felhasználói böngészését és keresését.

Áttekintettem a komplex adathalmazok megjelenítésével és interaktív felhasználói kezelésével kapcsolatos szakirodalmat és releváns megoldásokat, különös tekintettel a gráfban tárolt ontológiákon való felhasználói böngészés lehetséges megjelenítési módjaira.

Az alkalmazás fejlesztése során használt technológiák kiválasztásában tekintetbe vettem a követelmények által meghatározott korlátokat, és a hatékony használhatóság szempontjait is.

A megvalósított megoldások megtervezése egyrészt meg kellett oldanom a felhasználó implicit keresési szándékainak megfelelő választási lehetőségek algoritmikus kiszűrését, amit összetett gráfműveletek implementálásával oldottam meg. A tervezés másik fő összetevője a felhasználói felület megjelenésének a tervezése volt, ami lehetővé teszi a több alárendeltségi hierarchia szerinti párhuzamos szűrést a felhasználó számára.

Az alkalmazás funkcionális helyességét ellenőrző automatikus tesztek mellett valós felhasználói szempontok szerinti használhatósági tesztelést is végeztem a kész alkalmazáson. A tesztelés során megállapítottam, hogy milyen jellemzők alapján lehet megítélni a megoldás használhatóságát különböző ontológiai viszonyrendszerekben definiálható felhasználói adathalmazok esetén, és felsoroltam néhány jelentősebb fejlesztési lehetőséget.

Irodalomjegyzék

- [1] HANG T. LAU (2007). A Java Library of Graph Algorithms and Optimization, Chapman & Hall/CRC

- [2] <https://github.com/SoltiGabor/Cetli> , Solti Gábor (Letöltés dátuma: 2020. december 7.)

- [3] <https://keepsimple.io/en/uxscience.html> WOLF ALEXANYAN (Letöltés dátuma: 2020. december 3.)

- [4] <https://www.uxmatters.com/mt/archives/2012/10/the-importance-of-knowing-user-intent.php> (Letöltés dátuma: 2020. december 4.)

- [5] M. WROBEL A , T. NOCKEA , M. FLECHSIGA AND A. GLAUERA (2008). Graphical User Interface Design for Climate Impact Research Data Retrieval, Potsdam Institute for Climate Impact Research

- [6] MAREK DUDÁŠ, STEFFEN LOHMANN, VOJTĚCH SVÁTEK, DMITRY PAVLOV (2018). Ontology visualization methods and tools: a survey of the state of the art, Cambridge University Press

- [7] OLIVIER BODENREIDER (2002). Experiences in visualizing and navigating biomedical ontologies and knowledge bases, U.S. National Library of Medicine

- [8] RICHARD M CROWDER, MAX L WILSON, DAVID FOWLER, NIGEL SHADBOLT, GARY WILLS AND SYLVIA WONG (2009). Navigation Over a Large Ontology For Industrial Web Applications, University of Southampton

- [9] TIM BERNERS-LEE, JAMES HENDLER, ORA LASSILA (2001). The Semantic Web, Scientific American