Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik

Tanszék

Gráfok domináns csúcshalmazai

Témavezető:
Szabó László Ferenc
tanszékvezető egyetemi docens

Szerző:
Vozák Zsófia
programtervező informatikus BSc

Budapest, 2021

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

Tartalomjegyzék

1. fejezet

Bevezetés

A különböző domináns halmazokhoz kapcsolódó problémák vizsgálatát elsősorban kommunikációs hálózatok megtervezése motiválja. Ha például szenzorok rendszerére gondolunk, melyeknek egy helyre kell eljuttatniuk az általuk összegyűjtött információkat, akkor érdemes úgy megtervezni az üzenetek útját, hogy azok minél kevesebb üzenetváltással, minél gyorsabban eljussanak a céleszközhöz. Ebben segíthet valamilyen összefüggő domináns halmaz keresése a hálózaton belül. Adott csúcshalmazt akkor nevezünk egy gráf domináns halmazának, ha a gráf minden csúcsára igaz, hogy vagy ő maga része a halmaznak, vagy egy szomszédja. Egy ilyen halmaz összefüggő domináns halmaz, ha a halmaz által indukált részgráf összefüggő. Mindkettőre látható példa a 1.1 ábrán. A későbbi fejezetekben ezen két fogalomra DS[1] és CDS[2] rövidítéssel fogok hivatkozni. A domináns halmaz csúcsai a domináns csúcsok, ezek szomszédai pedig a dominált csúcsok.

Ezt az alapproblémát érdemes rendszertől függően különböző optimalizációs feladatokká alakítani. Vezetéknélküli eszközök hálózatánál például a lehető leghosszabb üzemidő elérésére törekszünk, míg egy munkahelyen modemek kihelyezésekor valószínűleg a készülékek számának minimalizálása lesz a cél - itt elég lehet domináns halmazt keresni.

Más területeken is jól hasznosíthatók a domináns halmaz problémakörben elkészült algoritmusok. Betegségek terjedésének kontrollálását, marketing stratégiák

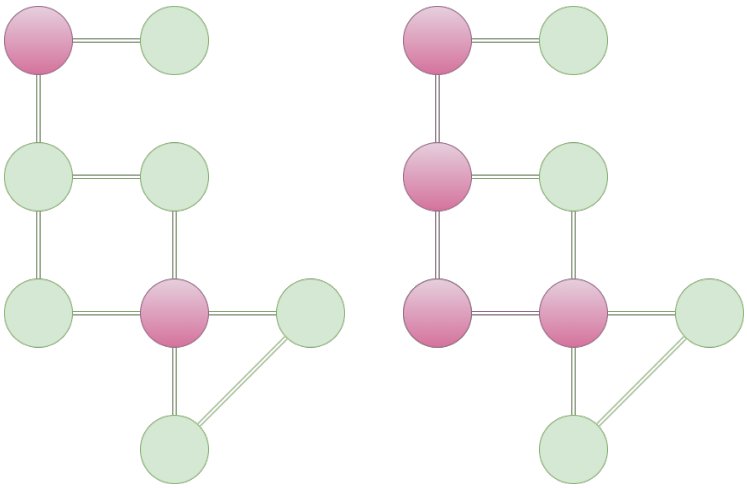
[1] A domináns halmaz angol megfelelőjének, a dominating set-nek a rövidítése.

[2] Az összefüggő domináns halmaz angol megfelelőjének, a connected dominating set-nek a rövidítése.

kidolgozását, integrált áramkörök tervezését, fizikai rendszerek modelljének elkészítését, közlekedési hálózatok szervezését, és társadalmi hálózatok vizsgálatát mind segítheti az adott rendszerek összefüggő domináns halmazon keresztül való szemlélése.

A szakdolgozat 2 fejezete a gráfok domináns halmazainak problémakörét és a megoldásra kifejlesztett algoritmusokat mutatja be. Az ezt követő fejezetek egy alkalmazás dokumentációját tartalmazzák.

Az alkalmazás a bemutatott algoritmusok alkalmazhatóságára szolgáltat példát. Segítségével szervezetek tagjait lehet számontartani, illetve a szervezetek hirdethetnek eseményeket, melyekre a tagok jelentkezhetnek. Az applikáció a tagok adatai, a jelentkezések, és az esemény paraméterei alapján elkészít egy (igény szerint csúcssúlyozott) gráfot, melyben előállít egy (összefüggő) domináns halmazt. A gráf csúcsai az adott szervezet tagjai, a megkonstruált domináns halmaz elemei pedig a meghívott személyek. Az alkalmazást akkor érdemes használni vendéglista összeállításához, amikor fontos, hogy a szervezet minden szintje, projektje és csoportja képviselve legyen az adott eseményen.



1.1. ábra. Példa domináns halmazra és összefüggő domináns halmazra

A szakdolgozat másik célja a Webes alkalmazások fejlesztése című tárgy keretében elsajátítható ismeretek megértése és gyakorlatba ültetése. A tárgyon nem vettem részt. Cserép Máté honlapján szereplő és a Microsoft Stream WAF csoportjában közzétett anyagokból tájékozódtam, illetve a honlapról letölthető példakódot is felhasználtam az alkalmazás készítésekor, közben új lehetőségeket is kerestem és kipróbáltam. Erről részletesebben a fejlesztői dokumentációban írok.

2. fejezet

Gráfelméleti háttér

2.1. Domináns halmaz

2.1.1. Probléma formálisan

A továbbiakban $G = (V, E)$ egy V csúcshalmazú és E élhalmazú G gráfot jelöl. Egy $v \in V$ csúcs szomszédait $N(v)$-vel jelöljük.

1. Definíció. *D-t akkor nevezzük a G gráf egy domináns halmazának, ha a gráf minden csúcsára igaz, hogy D-beliek, vagy legalább egy szomszédjuk szerepel D-ben.*

2. Definíció. *A DS problémánál adott egy G gráf és $k \in \mathbb{N}$. A kérdés az, hogy létezik-e G-nek legfeljebb k csúcsból álló domináns halmaza.*

2.1.2. Probléma nehézsége

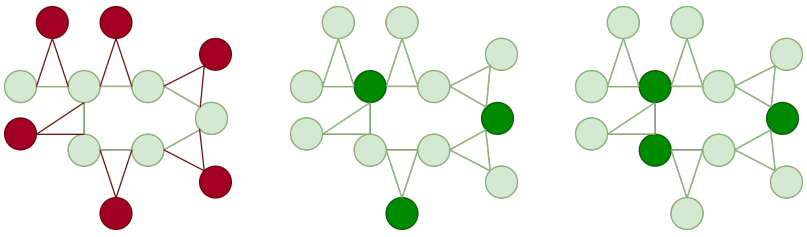
1. Tétel. *DS probléma NP-teljes.*

Bizonyítás.

1. *DS NP*-beli, ugyanis: Egy S halmazról polinomiális időben eldönthető, hogy egy G gráf domináns halmaza-e. Ehhez G minden v csúcsánál megnézzük, hogy v benne van-e S-ben vagy v szomszédja-e egy S-beli csúcsnak. Azaz minden csúcsnál ellenőrzünk minden élt, hogy az illeszkedik-e valamilyen S-beli csúcsra. Ha egy csúcsból kifutó összes élt ellenőriztük, de azok egyike sem illeszkedik S-beli csúcsra, elutasítjuk az S halmazt. Ha így elutasítás nélkül

végigjárunk minden csúcsot, akkor S domináns halmaz. Tehát $\Omega(|V||E|)$ idő alatt ellenőrizhető, azaz $DS \in NP$.

2. Létezik olyan NP-teljes probléma, mely polinomiális időben visszavezethető a domináns halmaz problémára, ugyanis ilyen például a csúcslefedési probléma (lásd a 3. definíciót) is. Ezt a következő módon vezethetjük vissza a DS problémára: legyen a $G = (V, E)$ gráf és $k \in \mathbb{N}$ a csúcslefedési probléma bemenete. Készítsük el a $G' = (V', E')$ gráfot a G-ből úgy, hogy minden $(u, v) \in E$ él esetén vegyünk hozzá egy ω csúcsot V'-höz és az (ω, u), (ω, v) éleket E'-höz. A 2.1 ábra első képén zöld szín jelöli az eredeti G gráfot, bordó pedig az így hozzáadott csúcsokat és éleket. Ekkor a G' gráf egy k elemű D domináns halmazából (2.1 ábra 2. képe) megszerkeszthető a csúcslefedési feladat egy megoldása G-re. Ha nem létezik ilyen D halmaz, akkor k elemű G-t lefedő csúcshalmaz sem létezik. A D halmaz azon csúcsait, melyek nem V-beliek lecserélhetjük egy szomszédjukra, a cserék után D továbbra is domináns halmaz. Az ilyen csúcsokat mindig van mire lecserélni, hiszen eleve úgy adtuk a gráfhoz, hogy két G gráfbeli csúccsal kötöttük össze. Az összes D-beli V-ben nem szereplő csúcsot így kicserélve D részhalmaza lesz V-nek, továbbá D egy megoldása a csúcslefedési feladatnak G-re. Utóbbi azért igaz, mert a fenti módon elkészült G' gráfban az új (a 2.1 ábrán bordó) csúcsok dominálásához vagy a bordó csúcsnak, vagy a (pontosan) két szomszédja közül az egyiknek kell szerepelnie a domináns halmazban.



2.1. ábra. Gráfátalakítás a csúcslefedés DS-ra való visszavezetéséhez

Ha valamelyik szomszéd szerepel ott, akkor a köztük lévő élt eleve lefedi, ha pedig az új csúcs került a D halmazba, akkor azt lecseréltük egy szomszédjára, így megint csak le van fedve a két szomszéd közti él. Továbbá ha D k elemű volt, a cserék után is k elemű marad, hiszen mindig pontosan egy csúcsot pontosan egy csúcsra cseréltünk. Mindezek miatt a végeredményül kapott D k

elemű lefedése a G gráfnak, azaz sikerült a csúcslefedési feladatot visszavezetni DS-ra.

□

3. Definíció. *A csúcslefedési problémánál a kérdés az, hogy létezik-e adott $k \in \mathbb{N}$ olyan részhalmaza a csúcshalmaznak, mely pontosan k elemet tartalmaz, és a gráf minden élének legalább az egyik végpontját tartalmazza.*

2.1.3. Független domináns halmaz

4. Definíció. *Egy gráf D domináns csúcshalmaza akkor független, ha az általa indukált részgráfban nincsenek élek, azaz csupa, az eredeti gráfban nem szomszédos elemből áll. Ekkor azt is mondják, hogy D független halmaz. Egy ilyen D halmazt akkor nevezünk maximális független halmaznak, ha bármilyen új V-beli csúcs hozzávételével már nem lenne független. Maximum független halmaz pedig akkor lesz, ha a legnagyobb csúcsszámú maximális független halmaz.*

Megjegyzés. *Minden maximális független halmaz egyben független domináns halmaz is.*

Az eddig legjobbnak mondható eredmény a MIS probléma kapcsán [1], ugyanis egy változó, dinamikus gráf esetén $O(|E|^{1/3})$ várható idő alatt képes frissíteni a MIS-t. Ezt az eredményt egy randomizált módszerrel érték el, melynek paramétere $p = |E|^{-1/3}$. Az algoritmus előkészítő, partícionáló szakaszában a H halmazba minden csúcsot p valószínűséggel tesznek be. Erre a H halmazra mohó technikával elkészítik H által indukált részgráf maximális független halmazát. Ennek során valamilyen sorrendet alakítanak ki a H-beli csúcsok között, és ilyen sorrendben teszik be a MIS-ba őket, de természetesen csak abban az esetben, ha még egyetlen szomszédjuk sem szerepel a MIS-ban.

Az alkalmazás szempontjából azon algoritmusok bizonyulnak hasznosnak, amelyek a lehető legkisebb csúcsszámú MIS-t képesek előállítani. Az alkalmazásban az alábbi 1-es algoritmust használom erre a célra. Ez egy mohó algoritmus, mely $O(|E|)$ komplexitású. Két később bemutatott algoritmusban is felhasználható, az egyik a CDOM, a másik a TCDS algoritmus. Az algoritmus első körben a levelek szomszédait

veszi hozzá a D domináns halmazhoz, hiszen egy levelet pontosan két csúcs dominálhat: egyetlen szomszédja és önmaga, azaz valamelyiket mindenképp választanunk kell. Illetve az is igaz, hogy egy levél pontosan önmagát és egyetlen szomszédját dominálhatja, míg szomszédja - ha az nem levél szintén, azaz a csúcshalmaz kettőnél több csúcsból áll - más csúcsot is dominál, így az jobb választás.

Következő lépésben addig bővítjük D-t, míg az DS nem lesz. Azt, hogy melyik csúcsot adjuk hozzá D-hez következőnek az határozza meg, hogy melyik csúccsal kiegészítve nő a legjobban a dominált csúcsok száma.

Algoritmus 1 MIS

Funct MIS($G = (V, E)$)

$D \leftarrow \{N(v) | v \in V \wedge |N(v)| = 1\}$
while $D \cup N(D) \neq V$ **do**
  $R \leftarrow V - D$
  for $v \in R$ **do**
    $D' \leftarrow D \cup \{v\}$
    $db(v) \leftarrow |D' \cup N(D')|$
  end for
  $L1 \leftarrow \text{ARGMAX}(R, db)$
  $v \leftarrow \text{ELSŐ}\big(\text{ARGMAX}(L1, J)\big)$
  $D \leftarrow D \cup \{v\}$
end while

2.2. Összefüggő domináns halmaz

2.2.1. Probléma formálisan

A továbbiakban a $G = (V, E)$ gráfban a $D \subseteq V$ által indukált részgráfot a $G[D]$ rövidítéssel jelölöm.

5. Definíció. *D-t akkor nevezzük a G gráf egy összefüggő domináns halmazának, ha a gráf minden csúcsára igaz, hogy D-beliek, vagy legalább egy szomszédjuk szerepel D-ben, illetve a D által a G gráfban indukált részgráf összefüggő (bármely két csúcsa között létezik út).*

6. Definíció. *A CDS problémánál adott egy G gráf és egy $k \in \mathbb{N}$. A kérdés az, hogy létezik-e G-nek legfeljebb k csúcsból álló összefüggő domináns halmaza.*

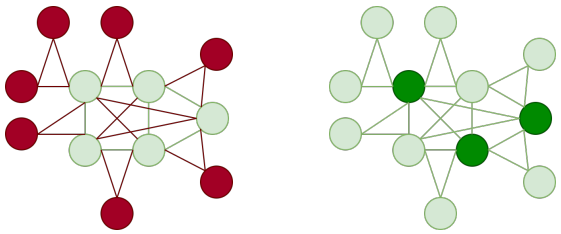
2.2.2. Probléma nehézsége

2. Tétel. *Az összefüggő domináns halmaz probléma NP-teljes.*

Bizonyítás.

1. $DS \in NP$, ugyanis: Egy S halmazról polinomiális időben eldönthető, hogy egy G gráf domináns halmaza-e (1 tétel). $G[D]$ összefüggőségét szélességi bejárással ellenőrizhetjük, melynek költsége szintén $O(|V||E|)$, tehát $CDS \in NP$.
2. Minden NP nehéz probléma polinomiális időben visszavezethető CDS-ra, ugyanis: elég belátni, hogy a csúcslefedési feladat (mely NP-teljes) visszavezethető a CDS problémára. Legyen a $G = (V, E)$ gráf és $k \in \mathbb{N}$ a csúcslefedési probléma bemenete. Készítsük el a $G' = (V', E')$ gráfot a G-ből úgy, hogy egészítsük ki G éleit úgy, hogy a keletkező gráf teljes legyen, majd V-t bővítsük ki úgy, hogy G minden éléhez szúrjunk be egy csúcsot, melyet kössünk össze az adott él mindkét végével. (Így $V' = V \cup x_{uv} | (u, v) \in E$ és $E' = E \cup (x_{uv}, u), (x_{uv}, v) \cup (u, v) | u, v \in V$). A 2.2 ábra első képén zöld szín jelöli az eredeti G gráfot, bordó pedig az így hozzáadott csúcsokat és éleket. Innen a bizonyítás analóg a 1 tétel bizonyításával. Egyedül annyit érdemes megjegyezni, hogy D átalakításakor az összefüggőségi tulajdonság sem sérül, hiszen végül csak V-beli csúcsokat tartlamaz majd D, melyek a G' gráfban teljes részgráfot indukálnak.

□



2.2. ábra. Gráfátalakítás a csúcslefedés CDS-ra való visszavezetéséhez

2.2.3. CDS probléma változatai

7. Definíció. *A $G = (V, E)$ gráf D összefüggő domináns halmaza minimális CDS (MCDS[3]), ha D a lehető legkisebb elemszámú CDS.*

MCDS keresése hasznos lehet olyan hálózatoknál, melyek hasonló elemekből (pl. vezetékes készülékekből, forgalmi csomópontokból stb.) állnak, nem érdemes az elemek tulajdonságait figyelembe venni CDS készítésekor.

8. Definíció. *Legyen $G = (V, E, \omega)$ élsúlyozott gráf ($\omega : V \to \mathbb{R}$ súlyfüggvény). G egy D összefüggő domináns halmaza minimális súlyú CDS (MWCDS[4]), ha G-nek nincs D összsúlyánál kisebb összsúlyú CDS-a.*

A legéletszerűbb, legtöbbek által érdekelt probléma a témakörben az MWCDS feladata. Nagyon univerzális, rengeteg hálózatra ráhúzható egy ilyen optimalizációs feladat. Gondoljunk csak például útvonaltervezésre, vagy arra, hogyan dönthetnénk el, hogy melyik utcába érdemes beköltözni ahhoz, hogy minden számunkra fontos helyre gyorsan és biztonságosan eljussunk. Ha ismerjük a csomópontok közti távolságot, az adott útszakasz forgalmasságát és hogy mennyire biztonságos a környék, attól függően, hogy ezen szempontok mennyire befolyásolnák a döntésünket készíthetünk, egy súlyfüggvényt a gráfhoz, mely segítségével kiválasztjuk a legjobb útvonalakat, környéket. A legjobb környék kiválasztásánál egyszerűen úgy inicializáljuk a keresett domináns halmazt, hogy abban eleve szerepeljenek azok a helyek, amelyeket gyakran látogatunk, vagy ha nem egyformán fontosak ezek a helyszínek, hozzájuk rendeljünk a többi csúcshoz képest kiugróan magas súlyokat.

9. Definíció. *A $G = (V, E)$ gráfban az $S \subseteq V$ halmaz k-összefüggő m-domináns halmaz ($(k, m) - CDS$), ha $G[S]$-ben bármely két csúcs között létezik legalább k (a kiindulási és cél csúcson kívül minden csúcsában) különböző út, és a csúcshalmaz bármely S-ben nem szereplő csúcsára igaz, hogy legalább m szomszédja S-beli (azaz legalább m domináns szomszédja van).*

A $(k, m) - CDS$ probléma megoldása stabil, megbízható hálózatok készítésére alkalmas.

[3]Minimum connected dominating set

[4]Minimum weighted connected dominating set

10. Definíció. *Az olyan $D \subseteq V$ halmazokat egy $G(V, E)$ gráfban, amelyekre $\forall v \in V$: v legfeljebb d távolságra, azaz d ugrásra van a legközelebbi D-beli elemtől d-ugró összefüggő domináns halmaznak (d-CDS) nevezzük.*

11. Definíció. *Egy gráf feszítőfája a gráf minden csúcsát tartalmazó fa részgráf. Ez minimális feszítőfa, ha (élsúlyozott gráfban) nincs nála kevesebb összélsúlyú feszítőfa.*

Megjegyzés. *Minden összefüggő gráfnak van feszítőfája.*

12. Definíció. *Adott $G = (V, E)$ egyszerű, összefüggő gráf, és az S és T csúcshalmazok, melyek V partíciói ($S, T \subseteq V : S \cup T = V \wedge S \cap T = \emptyset$). S elemeit Steiner-pontoknak, T elemeit termináloknak hívjuk. Továbbá adott egy $\phi : E \to \mathbb{R}^+$ élsúlyozás. F a G Steiner-fája, ha F olyan fa részgráfja G-nek, amely T minden csúcsát tartalmazza. Összefüggő gráfnak mindig van Steiner-fája, hiszen bármelyik feszítőfa ilyen lesz. Egy Steiner-fa súlya a fában szereplő élek súlyainak összege. A feladat G egy minimális költségű Steiner-fájának megadása a $(G = (V, E, \omega), S, T)$ bemenet esetén.*

Megjegyzés. *A Steiner és a minimális feszítőfa látszólag nagyon hasonló problémák, előbbi mégis NP-nehéz, míg utóbbi például a Kruskal algoritmussal $O(|E| \log_2 |E|)$ idő alatt megoldható.*

A következő fejezetekben a fentiekhez hasonló problémák közelítő megoldását adó algoritmusokról lesz szó. Mivel ezek NP-teljes problémák, pontos megoldást adó algoritmust nem érdemes kidolgozni, hiszen a gyakorlatban - főleg gyakran változó hálózatok esetén - kruciális az algoritmusok gyorsasága és tárigénye. Így problémától függően érdemes a pontosságból engedni annyit, hogy a kapott modell még jól használható legyen, cserébe jelentősen csökkenjen a futási idő.

Mivel a szélességi bejárás segítségével könnyen meghatározható, hogy egy gráf összefüggő vagy sem, a továbbiakban feltételezzük, hogy a következő fejezetek algoritmusai bemenetei összefüggő gráfok. Továbbá azt is feltételezhetjük, hogy irányítatlan gráfok az inputok, hiszen a gyakorlatban általában kétirányú kommunikációra van szükség, azaz két eszköz között akkor fut él a gráfban, ha azok egymás hatótávolságán belül vannak. Azaz, ha minden v csúcsra ismerjük annak r(v) hatótávolságát, akkor az (u,v) él akkor, és csak akkor kerül be a hálózatot reprezentáló gráfba, ha $\rho(u, v) \leqslant min\big(r(u), r(v)\big)$, ahol $\rho(u, v)$ u és v távolsága.

2.3. CDOM

CDS problémára ad megközezelítő megoldást[2].

2.3.1. Az algoritmus

Algoritmus 2 CDOM

1. Legyen $v \in V$ egy tetszőleges csúcs.
2. v-ből indulva készítsük el szélességi bejárással a gráf egy feszítőfáját. Nevezzük ezt T-nek.
3. Legyen k a fa magassága.
4. Legyen S_i a fa i-edik szintjén (v-től i távolságra lévő) csúcsok halmaza
5. Legyen $IS_0 = \{v\}; NS_0 = \emptyset$
6. for i = 1...k do
   (a) $DS_i = \{v \quad | \quad v \in S_i \wedge \exists u \in IS_{i-1} : v \in N(u)\}$
   (b) $IS_i \leftarrow \text{MIS}(G[S_i - DS_i])$
   (c) $NS_i \leftarrow \{u \quad |$ a T fában u valamilyen $v \in IS_i$ csúcs szülője$\}$
7. eredmény: $\cup_{i=0}^{k} IS_i] \cup \cup_{i=0}^{k} NS_i$

S_i az i-edik szinten lévő csúcsok halmaza. IS_i és DS_i az S_i csúcshalmaz részhalmazai. IS_i az i-edik szint csúcshalmazában nem dominált csúcsok egy maximális független halmaza. NS_i az S_{i-1} részhalmaza, az IS_i halmazokba beválogatott csúcsok közötti összeköttetést biztosítják. Tehát az IS_i halmazok uniója domináns halmazt alkot, hozzájuk véve az NS_i halmazokat összefüggő domináns halmazt kapunk. DS_i az $i-1$-edik szinten kiválasztott csúcsok által dominált csúcsok halmaza az i-edik szinten.

2.3.2. Analízis

Az algoritmus műveletigényének megállapításához nézzük az algoritmus egyes lépéseinek költségét. A 2. pontban a feszítőfa elkészítésének műveletigénye $O(|V|^2)$, hiszen csúcsonként haladunk, és minden csúcs összes szomszédját vizsgáljuk. A MIS algoritmus szintén $O(|V|^2)$ komplexitású. A dominált csúcsok megkeresése is ennyi időt vesz igénybe összesen, mivel $O(|V|)$ csúcs $O(|V|)$ szomszédját vizsgáljuk meg, de egy csúcsnak sosem nézzük meg többször a szomszédait, mert a szintek egyben partíciók is, tehát egy csúcs csak egy szinten szerepel. NS halmazok kialakításánál hasonlóan gondolkodhatunk. Tehát az algoritmus műveletigénye $O(|V|^2)$.

3. Tétel. *CDOM valóban CDS-t állít elő.*

Bizonyítás. Tulajdonképp azt kell bizonyítani, hogy bármely első k IS_i halmaz együttesen dominálja az első k szint minden csúcsát, és hogy e halmazegyüttest az első k NS_i halmazzal kiegészítve az általuk indukált részgráf összefüggő. Ezt k-ra végzett indukcióval tehetjük meg.

Ha $k = 0$, akkor nyilván teljesül, hiszen $IS_0 = \{v\}, IS_0 \cup NS_0 = \{v\}$. Ez megfelel a tételnek.

Tegyük fel, hogy valamilyen k számra is teljesül a tétel.

Ekkor $k+1$ esetén elmondható az indukciós feltevés miatt, hogy, mivel k-ra jó, ezért az első k IS_i uniója dominálja az első k szint csúcsait, ezért nyilván a domináns halmazhoz IS_{k+1}-et hozzávéve szintén domináns marad az első k szintre nézve. Tehát azt kell megmutatni, hogy a k+1-edik szint csúcsait is dominálja. A 2. algoritmus 6/a sorában DS_{k+1} minden csúcsát dominálja IS_k, és a 6/b sor miatt IS_{k+1} dominálja $S_{k+1} - DS_{k+1}$, mivel előbbi definíció szerint maximális független halmaza az utóbbi által indukált részgráfnak. Tehát a tétel első állítása $k + 1$-re is teljesül.

Az indukciós feltevésben feltettük, hogy az első k IS_i uniója az első k NS_i halmazzal kiegészítve összefüggő részgráfot indukál az eredeti gráfban. Az algoritmus 6/c lépésében $NS_{k+1} \subseteq S_k$, mivel $IS_{k+1} \subseteq S_{k+1}$. Az indukciós feltevésben azt is feltettük, hogy az első k IS_i halmaz uniója dominálja az első k szint csúcsait, ezért NS_{k+1}-t is dominálja, tehát az uniójuk által indukált részgráf összefüggő. Végül a 8. lépésben NS_{k+1} dominálja IS_{k+1}-t, így összefüggő lesz az első $k + 1$ IS_i és NS_i

halmazok uniója által indukált részgráf. Ezzel mindkét feltételre bizonyítottuk, hogy bármilyen nemnegatív k egészre teljesülnek.

□

13. Definíció. *Legyen G egy olyan gráf, melyben a csúcsok egység sugarú körlemezek, és két csúcsot akkor köt össze él, ha azok körvonalai metszik egymást. Az ilyen gráfokat egységlemezes gráfoknak nevezzük.*

2.3.1. Lemma. *Legyen $G = (V, E)$ egységlemezes gráf. Ekkor G-nek nincs $K_{1,6}$-al izomorf részgráfja.*

A 2.3.1. lemma tulajdonképp azt jelenti, hogy bármilyen $G = (V, E)$ egységlemezes gráf esetén bármelyik csúcsot eltávolítva a gráfból a megmaradt gráf maximális független halmazának mérete nem lesz több ötnél.

4. Tétel. *Legyen $G = (V, E)$ egységlemezes gráf. Legyen D^* minimális domináns halmaza, D pedig maximális független halmaza a gráfnak. Ekkor $|D| \leq 5|D^*|$.*

Bizonyítás. Mivel D független halmaz, ezért a 2.3.1. lemma miatt egyetlen D^*-beli csúcs sem dominálhat 5-nél több csúcsot D-ben. Tehát $|D| \leq 5|D^*|$. □

5. Tétel. *Legyen G egységlemezes gráf. Ekkor a CDOM algoritmus által a G gráfhoz előállított D összefüggő domináns halmazra igaz, hogy $|D| \leq 10|opt(G)|$, ahol $opt(G)$ a MCDS (7) probléma optimális megoldása.*

Bizonyítás. Legyen $IS = \cup_{i=0}^{k} IS_i$ és $NS = \cup_{i=0}^{k} NS_i$. IS nyilván maximális független halmaza G-nek, ezért a 4. tétel szerint $|IS| \leq 5|\mathrm{MDS}(G)|$, ezért $|IS| \leq 5|\mathrm{MCDS}(G)|$, hiszen a minimális összefüggő domináns halmaz nyilván nem szűkebb, mint a minimális domináns halmaz. Az algoritmus 5. és 6/c lépése következtében $|NS_i| \leq |IS_i|$ minden $i = 1, .., k$ indexre. Ezért $|NS \cup IS| \leq 2|IS| \leq 10|\mathrm{MCDS}(G)|$. □

2.4. T-CDS

2.4.1. Az algoritmus nagy vonalakban

Az MCDS probléma egy változatára ad megközelítő megoldást[3]. Legyen tehát a gráfunk $G = (V, E, \omega)$, ahol $\omega : V \to \mathbb{R}$ a gráf súlyfüggvénye. ω rendelheti például egy hálózat eszközeihez azok tárhelyének nagyságát, üzemidejének hosszát, vagy

bármilyen, a rendszer működése szempontjából fontos tulajdonságának valamilyen számszerűsített formáját. Sőt, akár ezek mindegyikének függvényében is alakíthatók a súlyok. Fő, hogy az így kapott szám az adott készülék megbízhatóságát tükrözze. Ezen súlyfüggvény segítségével fogjuk meghatározni a gráf CDS-ait minősítő mennyiséget, a CDS csúcsainak összfontosságát. Az összfontosságot maximalizáló CDS lesz a megoldás, a hálózat úgynevezett backbone, a rendszer váza. Tehát azt az MCDS-t keressük, mely maximalizálja az összfontosságot.

2.4.2. Költségfüggvény bevezetése

14. Definíció. *Legyen egy $G = (V, E, \omega)$ gráf esetén a $p : V \to \mathbb{R}$ függvény egy $v \in V$ csúcs esetén*

$$p(v) = \sum_{v \in V} \frac{\omega(v)}{\sum_{v \in V} \omega(v)}$$

$\sum_{v \in V} p(v) = 1$ miatt p valószínűségi eloszlás, így használható entrópiaszámításra. A G gráf entrópiája:

$$I_\omega(G) = -\sum_{v \in V} p(v) \log_2(p(v)) = -\sum_{v \in V} \frac{\omega(v)}{\sum_{v \in V} \omega(v)} \log_2 \left(\frac{\omega(v)}{\sum_{v \in V} \omega(v)} \right) =$$
$$= \log_2 \left(\sum_{v \in V} \omega(v) \right) - \sum_{v \in V} \frac{\omega(v)}{\sum_{v \in V} \omega(v)} \log_2 \omega(v).$$

15. Definíció. *A $v \in V$ csúcs elhagyásával a $G = (V, E, \omega)$ gráfban keltett entrópiaváltozás*

$$EV(v) = I_f(G) - I_f(G_v),$$

ahol

$$G_v = (V - \{v\}, E - \{(u, v) | (u, v) \in E\}).$$

Ezek után vezessük be a $T : V \to \mathbb{R}$ fontossági függvényt, mely a fent definiált függvényekkel operál. Sokféleképpen lehetne definiálni, most azonban csak 3 lehetőséget vizsgálunk:

$$T_1(v) = \log_2(p(v)),$$

$$T_2(v) = -p(v) \log_2(p(v)),$$

$$T_3(v) = EV(v).$$

16. Definíció. $J : P(V) \to \mathbb{R}$:

$$J(D) = \sum_{v \in D} T(v) \quad (D \subseteq V)$$

V egy adott részhalmazának összfontosságát adja meg. Ugyan J(D)-t a D halmaz költségének is nevezik, a továbbiakban mégis összfontosságként fogok rá hivatkozni. Ennek az az oka, hogy a költség szó inkább minimalizálandó mennyiségek megnevezésére használatos, itt azonban J(D) a D halmaz minőségének növelésével nő, azaz a feladatunk az összeg maximalizálása.

Megjegyzés.

1.

$$P_E(D) = 1 - \prod_{v \in D} p(v)$$

annak a valószínűségét adja meg, hogy a $G[D]$ részgráfon belüli üzenettovábbítás közben hiba lép fel.

$$minP_E(D) \equiv max \prod_{v \in D} p(v) \equiv max \log_2 \left(\prod_{v \in D} p(v) \right) \equiv$$

$$\equiv max \sum_{v \in D} \log_2 p(v) = max \sum_{v \in D} T_1,$$

azaz a $T_1(v) = \log_2(p(v))$ fontossági függvénnel definiált

$$J(D) = \sum_{v \in D} T_1(v) = \sum_{v \in D} \log_2(p(v))$$

összfontosság maximalizálásához ugyanarra a D halmazra van szükség, mint a hiba esélyének minimalizálásához, tehát jól választottuk meg a fontossági függvényt.

2.

$$\sum_{v \in D} T_1(v) = \sum_{v \in D} \log_2(p(v)),$$

$$\sum_{v \in D} T_2(v) = -\sum_{v \in D} p(v) \log_2(p(v)),$$

$$\sum_{v \in D} T_3(v) = \sum_{v \in D} T_3(v) = EV(v)$$

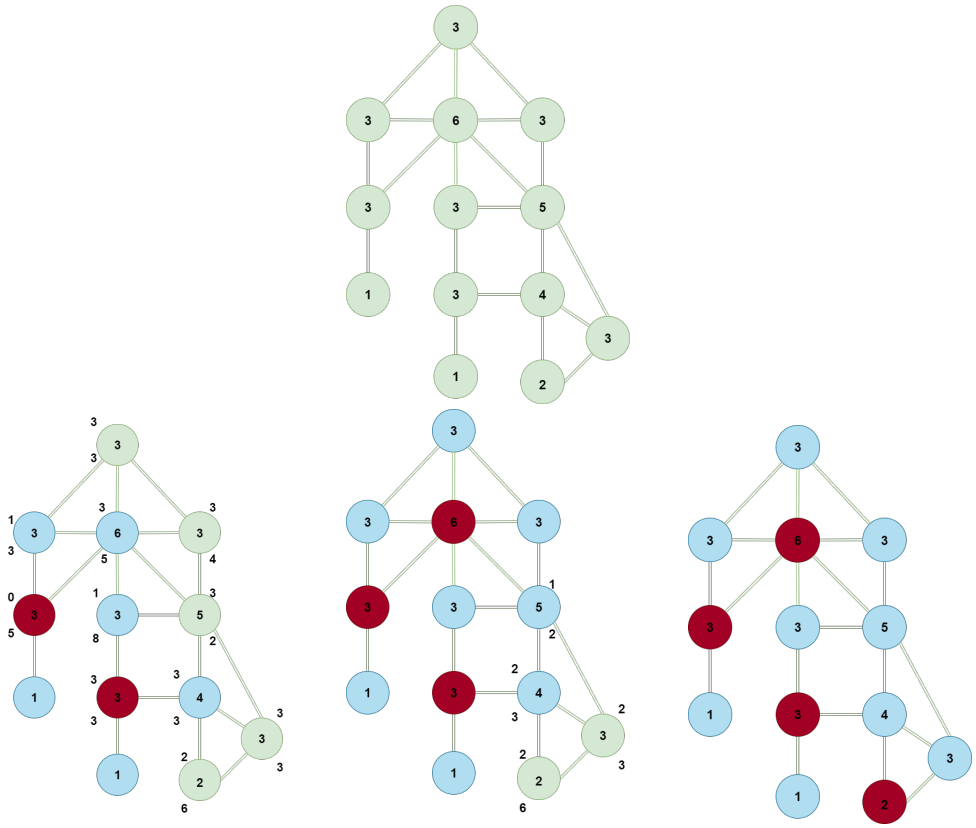
összfontosságokat maximalizáló CDS-kat rendre $T_1 - CDS$, $T_2 - CDS$ és $T_3 - CDS$ jelölik.

2.4.3. Az algoritmus

Három fázisból áll. Az első célja, hogy a lehető legkevesebb csúcsból álló nem összefüggő domináns halmazt készítse el. A másodikban a kapott halmazt összefüggővé tesszük. A harmadikban pedig eltávolítjuk a "fölösleges" csúcsokat (melyek nélkül még CDS marad a halmaz).

A 2.3, 2.4 és 2.5 ábra az algoritmus különböző szakaszait szemléleteti egy egyszerű példán. A csúcsok súlyait itt nem a fontossági függvény, hanem egy egyszerűsített súlyfüggvény adja a példa jobb átláthatóságát biztosítandó. A csúcsok belsejében jelzett szám a fokszám, alulra pedig a súlyok kerültek.

Első fázis - DS készítése



2.3. ábra. Első fázis futtatása példán. A csúcsokba a fokszámok, alulra pedig a súlyok kerültek. A domináns halmaz bordó, a dominált csúcsok kékek. A csúcsok felett azon csúcsok száma látható, melyek még dominálatlan szomszédai annak.

Az első fázis a MIS problémát oldja meg csúcssúlyozott gráfon. Súlyozatlan gráfokon az algoritmusát már korábban áttekintettük, lásd a 1. algoritmust. Ehhez képest az itt használt algoritmus annyiban tér el, hogy egy új csúcs hozzáadásakor kialakult holtverseny esetén a nagyobb fontosságú csúcsot választjuk. Műveletigénye $O(|V|^2)$.

Második fázis - CDS készítése

Az algoritmus ezen szakaszában addig keresünk úgynevezett összekötő csúcsokat, míg $G[D]$ összefüggővé nem válik. Mindig azt a csúcsot választjuk, mely a legtöbb független komponenst képes összekötni. Ha ilyen több is van, közülük is azt vesszük D-hez, amelyiknek nagyobb a fokszáma, közülük pedig a legfontosabbat. 3. algoritmus a KOMPONENSEK_DB függvényt írja le, mely megadja az input gráf komponenseinek számát. Így minden lépésben olyan a D halmazban nem szereplő V-beli v csúcsot keresünk, melyet D-hez adva a lehető legjobban csökken a D által indukált részgráf komponenseinek száma.

Algoritmus 3 Második fázis: független részgráfok számlálása

KOMPONENSEK_DB($G = (V, E)$)

```
n ← 0
R ← G
while R ≠ ∅ do
    S ← ELSŐ(R)
    while S ≠ S ∪ N(S) do
        S ← S ∪ N(S)
    end while
    n ← n + 1
    R ← R − S
end while
return n
```

Az összekötők kiválasztásának algoritmusát a 4. algoritmus mutatja.
Itt az ARGMAX(R, db) jelölésben db egy egészekből álló vektor, és $R \subseteq V$. A vektort R-beli csúcsokkal indexeljük. ARGMAX(R, db) tehát azokat az "indexeket"

adja meg, ahol db a legnagyobb értéket veszi föl, azaz olyan $v \in R$ csúcsokat ad vissza, melyekre $db(v)$ maximális.

Algoritmus 4 Második fázis: CDS készítése

Phase_2($G = (V, E)$, DS)

$CDS \leftarrow D$

$R \leftarrow V - CDS$

while ¬ÖSSZEFÜGGŐ($G[CDS]$) **do**

  for $v \in R$ **do**

    $db(v) \leftarrow$ KOMPONENSEK_DB$\big(G[CDS \cup \{v\}]\big)$

  end for

  $L1 \leftarrow$ ARGMIN(R, db)

  $L2 \leftarrow$ ARGMAX$(L1,$ FOKSZÁM$)$

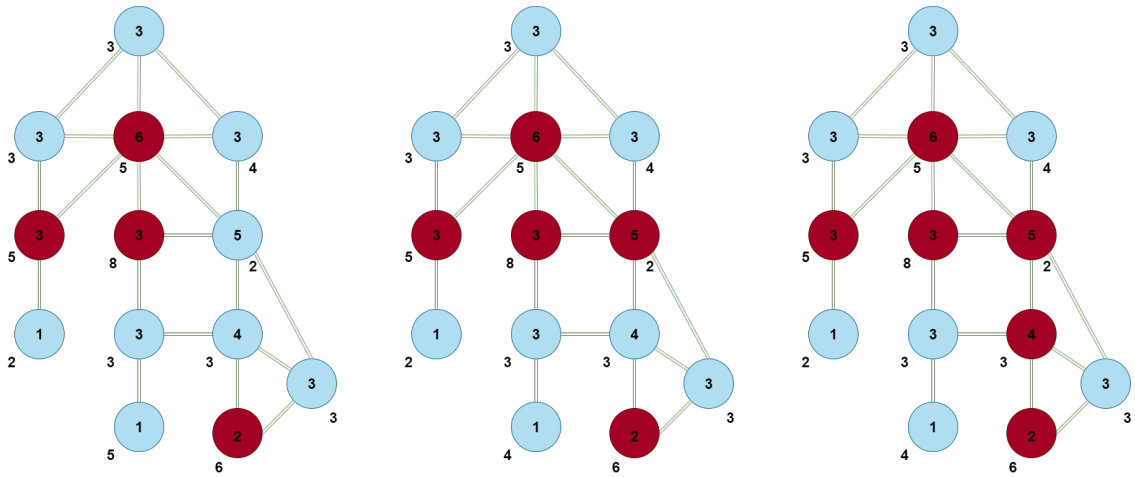
  $v \leftarrow$ ELSŐ$\big($ARGMAX$(L2, J)\big)$

  $CDS \leftarrow CDS \cup \{v\}$

  $R \leftarrow R - \{v\}$

end while

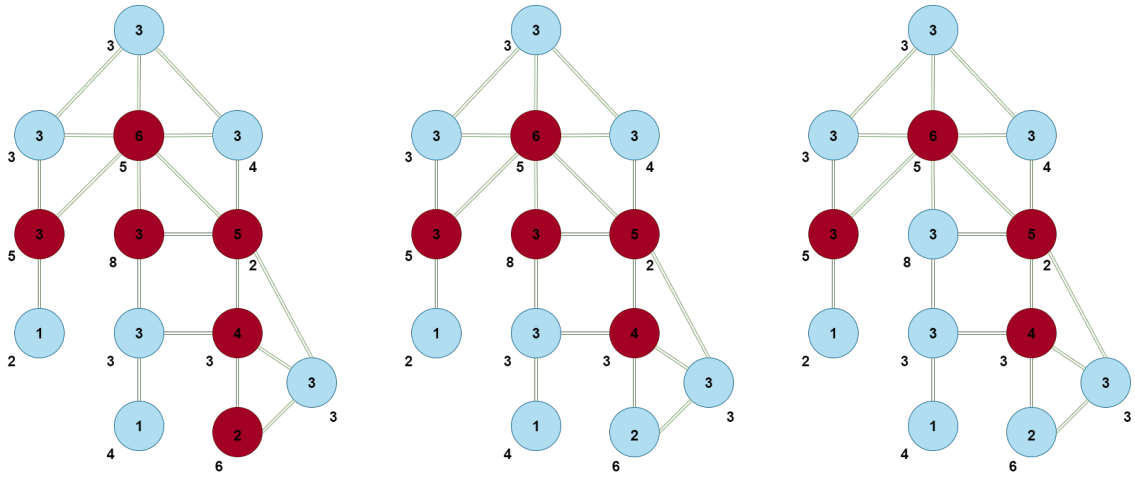
A KOMPONENSEK_DB függvény költsége $O(|V|^2)$, mivel minden csúcs minden szomszédját vizsgáljuk, és ez legrosszabb esetben (teljes gráf) $|V|^2$ vizsgálat. Összefüggőséget szélességi bejárással vizsgálhatunk például, melynek ugyanennyi a műveletigénye. A második fázisban a külső ciklus $O(|V|)$-szer fut le. Egy iterációban elvégezzük a négyzetes költségű összefüggőség vizsgálatot, illetve a for ciklus utáni műveleteket, melyek lineáris időben elvégezhetők. Tehát a while ciklus a belső for ciklus nélkül $O(|V|^3)$ műveletigényű. A belső for ciklus magját legfeljebb $|V|$-szer hajtjuk végre összesen, hiszen R mérete folyamatosan csökken. Tehát a belső ciklus költsége szintén köbös, így az egész második fázisról elmondható, hogy műveletigénye köbösen függ a csúcsok számától.



2.4. ábra. Második fázis (összekötők kiválasztása) futtatása példán.

Harmadik fázis - metszés

A második fázisban az összekötő csúcsok beszúrásakor előfordulhatott, hogy egy vagy több összekötő együttesen dominálta egy másik - már eleve a domináns halmazban lévő - csúcs által dominált csúcsokat. Ekkor, ha a régebbi csúcs eltávolításával az indukált részgráf még mindig összefüggő, érdemes eltávolítani, hiszen nem csak az a cél, hogy $J(D)$-t maximalizáljuk, hanem elsősorban a csúcsok számát szeretnénk minimalizálni. Ezért szükséges a CDS-t metszeni.



2.5. ábra. Harmadik fázis (metszés) futtatása példán.

A metszés minden lépésében azt a csúcsot távolítjuk el, melynek a legkisebb a fokszáma. Ha több ilyen csúcs van, akkor azt a $v \in CDS$ csúcsot, melyre $|CDS \cap N(v)|$

a legkisebb. Holtverseny esetén ismét a csúcsok fontossága dönt, nyilván a kevésbé fontos csúcsot hagyjuk el. Mindezt addig ismételjük, míg már nem csökkenthető a CDS mérete anélkül, hogy az ne esne több komponensre. Az algoritmus részleteseben a 5. pszeudokódban olvasható.

A harmadik fázis szintén köbös költséggel rendelkezik, így az egész T-CDS algoritmus műveletigénye $O(|V|^3)$.

Algoritmus 5 Harmadik fázis: metszés

Phase_3($G = (V, E)$, CDS)

$R \leftarrow CDS$
while $R \neq \emptyset$ **do**
  $L1 \leftarrow \text{ARGMIN}(R, db)$
  $L2 \leftarrow \text{ARGMAX}(L1, \text{FOKSZÁM})$
  $v \leftarrow \text{ELSŐ}\big(\text{ARGMAX}(L2, J)\big)$
  $TCDS \leftarrow CDS - v$
  if ÖSSZEFÜGGŐ($TCDS$)és $(TCDS \cup N(TCDS) = G)$ **then**
    $CDS \leftarrow TCDS$
    Phase_3(G, CDS)
  else
    $R \leftarrow R - \{v\}$
  end if
end while

3. fejezet

Felhasználói dokumentáció

3.1. A program által megoldott feladat

Az alkalmazás három fő komponensből áll: egy weblapból, egy asztali alkalmazásból és egy webszolgáltatásból áll. A weboldal olyan felületet biztosít a felhasználó számára, melynek segítségével felregisztrálhat különböző szervezetekhez, így elérhetővé válnak számára a szervezet által meghirdetett események. Az elérhető eseményekre egy-egy form kitöltésével reagálhat. Az űrlap többek között lehetőséget nyújt egy esemény lemondására, a felhasználó számára kedvező időpontok kiválasztására és az esemény helyszínének megszavazására.

Az asztali alkalmazás csak admin jogsultságú felhasználók számára érhető el. Ilyen jogosultság szerzéséhez először egyszerű felahsználóként kell regisztrálni a weblapon a *Register* lapján, ki kell jelentkezni, majd a már megadott felhasználói adatokkal a *For organisation* menüpontra kattintva újra kell regisztrálni, ezúttal a szervezet adataival kiegészítve regisztrációnkat. Az asztali alkalmazás fő funkciói események meghirdetése, illetve hozzájuk tartozó vendéglista összeállítása, melyet a szervezet adminisztrátorai által megadott beállítások és a weboldalon a felhasználóktól bekért információk segítségével határoz meg. Hogy a lista elkészítése során mely információknak mekkora jelentőséget tulajdonít a program, azt az adminisztrációs felületen szintén be lehet állítani.

Tehát az alkalmazás olyan szervezetek számára készült, amelyek szeretnének átfogó eseményeket szervezni szervezeti szinten, és ehhez szeretnék a szervezet minden szintjét bevonni, illetve minden foglalkozási terület és projekt számára felszólalási

lehetőséget biztosítani. Ennek megfelelően állít össze egy vendéglistát az alkalmazás.

A jobb áttekinthetőségért a következőkben először az asztali alkalmazás és az azt kiszolgáló webszolgáltatás felhasználói dokumentációja, majd külön a weblap felhasználói dokumentációja olvasható.

3.2. Az asztali alkalmazás felhasználói dokumentációja

3.2.1. Futási környezet

A futtatás ugyanazokat az eszközöket és feltételeket igényli, mint a weblap, böngésző viszont nem szükséges hozzá. Futtatáskor az alább írtakra viszont oda kell figyelni.

Futtatás előtt fontos, hogy átírjuk a *ServicesForDesktopApp.csproj* file-ban a <DocumentationFile> tag-hez tartozó útvonal *C:\Users\vozak\Desktop* szakaszát az aktuális útvonalra, különben nem tudjuk elindítani az alkalmazást. Ezt közvetlenül a file-ban, vagy a Visual Studio-ban a projektre jobb egérgombot lenyomva a *Properties* opciót kiválasztva a baloldalon megjelnő sávban a *Build* feliratra kattintva az Output szegmensében az output path átírásával tehetjük meg.

Most két projektet kell egyszerre futtatni a Visual Studio 2019-ből: *ServicesForDesktopApp* (webszolgáltatás) és a *MeetingOrganiserDesktopApp* (asztali kliens alkalmazás) projektet egyszerre. Ez a következő módon állítható be: jobbklikkel kattintsunk a solution-re és nyissuk meg a *Properties*-t. A Common properties menüpont alatt a startup project-re kattintva kiválaszthatjuk az elindítani kívánt projekteket. A Start gombra kattintva mindkét alkalmazás el fog indulni. Elindítható velük párhuzamosan a webalkalmazás is, hiszen a lényeg az, hogy ugyanazt az adatbázist használják, ezzel tulajdonképp közvetetten egymással is kommunikálva.

3.2.2. Használati útmutató

3.2.3. Felhasználói esetek

Aktorok:

- adminisztrátori jogosultsággal rendelkező felhasználó

Funkciók:

- események listázása
- esemény létrehozása
- esemény adatainak szerkesztése (vendéglista előállításához szükséges konfigurációk beállítása is)
- esemény törlése
- helyszínek listázása
- helyszín létrehozása
- helyszín szerkesztése
- helyszín törlése
- kép feltöltése
- kép törlése
- vendéglista lekérése
- tagok listázása
- új tag felvétele
- tag adatainak szerkesztése
- tag törlése
- tagok projektjeinek listázása

3.2.4. Használati útmutató

Használt grafikus vezérlők

- táblázatok
  - a fejléc feliratait kattintva az adott oszlop alapján rendezi az adatokat
  - ha szélesebb a táblázat az ablaknál megjelenik a táblázat alján egy görgő, melyet odébb húzva görgethetünk a táblázatban vízszintesen
  - a kijelölt sort kékkel jelöli

- menü
  - minden funkció valemelyik menüpontból érhető el
  - lehetnek almenüpontjai - ekkor a főmenüpont kattintásával csak az almenüpontok hozható elő, más eseményt nem vált ki
- gombok
  - eseménykiváltásra szolgálnak
  - Halvány színe inaktivitását jelzi
  - Akkor inaktív egy gomb, ha nem teljesülnek a használatához a feltételek, például nincs kiválasztva szerkesztendő elem, nincsenek adatok betöltve, helytelenül töltöttünk ki egy beviteli mezőt
- dátumválasztó
  - gépelve is bevihető a dátum és idő
  - a Save gomb mellett találhatók a helyes bevitelhez szükséges információk
  - rákattintva egy naptár és egy spinbox jelenik meg
  - először válasszuk ki a spinbox segítségével a kívánt időt
  - majd válasszuk ki a dátumot

Bejelentkezés

Az alkalmazást elindítva egy bejelentkeztető dialógusablak jelenik meg. Csak adminisztrátori jogosultsággal rendelkező felhasználó bejelentkezési adatait fogadja el. A bejelentkezés csak akkor érvényes, ha az Organisation címkével jelzett szövegbeviteli mezőbe valóban olyan szervezet nevét írja a felhasználó, melynél adminisztrátori jogosultsága van. Amennyiben helytelen adatokat ad meg azt egy felugró ablak jelzi. Ennek Ok gombjára kattintva újrapróbálható a bejelentkezés.

A bejelentkezést követően megjelenik az alkalmazás fő ablaka. Ez csak akkor zárul be, ha a felső menüsor *Exit* menüpontjára kattint a felhasználó.

Eseményekkel kapcsolatos adatok kezelése

Az eseményekkel, helyszínekkel és képekkel kapcsolatos műveleteket biztosító felületet az *Events* menüpont alatti *List events* almenüpont megkattintásával tölthetjük be. Ezzel nem nyílik új ablak, hanem az aktuális ablak állapota frissül. 3

táblázat jelenik meg ekkor - illetve először csak a 3 táblázat fejléce és az első táblázat egészében.

Az első táblázat tartalmazza az események adatait. A táblázat alatt közvetlenül egy sorban jelennek meg az *Add new*, *Edit selected*, *Delete selected* és *Create guest list* feliratú gombok, melyek rendre egy új esemény létrehozásáért, a kijelölt esemény törléséért, a kijelölt esemény szerkesztéséért és a kijelölt eseményhez vendéglista lekéréséért és megjelenítéséért felelősek. Egyik akció végrehajtása sem végleges, az Events-> List events menüpontra kattintva az eredeti adatokat változatlanul visszakapjuk. A változtásokat az Events-> Save changes menüpontjára kattintva tudjuk véglegesen elmenteni.

Az események táblázata alatt található a helyszínek táblázata. Ez üres, ha nem jelöltünk ki eseményt. Amennyiben kijelöltünk egyet, úgy a hozzá rendelt helyszínek is megjelennek egy táblázatban. A táblázat alatt ismét az *Add new*, *Edit selected* és *Delete selected* gombok jelennek meg, melyek már a helyszínekre vonatkoznak.

Az utolsó blokkban láthatók a kijelölt helyszínhez rendelt képek, alattuk közvetlenül pedig a képek feltöltéséért és törléésért felelős *Add new* és *Delete selected* feliratú gombok.

Az *Add new* és az *Edit selected* gombok egy új ablakot nyitnak meg a főablak mellett. Mindhárom entitáshoz (esemény, helyszín és kép) másmilyen ablak tartozik. Az események és a helyszínek szerkesztői beviteli mezőket tartalmaznak és a Save és Cancel gombok jelennek meg a legaljukon. Előbbivel menthetők a módosítások, amennyiben azok helyesek, utóbbival elvethetők. Kép létrehozásakor egy fájl kiválasztó dialógus nyílik meg.

A *Delete* feliratú gombok hatására egyszerűen eltűnik az előzőleg kiemelt elem, nem navigál el a főablakról az alkalmazás.

Tagokkal kapcsolatos adatok kezelése

Az eseményekhez hasonlóan táblázatos formában jelennek meg a tagok is, és alattuk is az *Add new*, *Edit selected* és *Delete selected* gombok jelennek meg. Egy tagra kattintva megjelennek a tagok táblázata alatt a kijelölt tag projektjei is, ha vannak.

3.2.5. Figyelmeztető üzenetek

Esemény szerkesztésekor

Az eseményszerkesztő ablakban helytelen adotok megadása után a *Save* gombra kattintáskor egy felugró ablak jelenik meg egy hibaüzenettel. Ha a felhasználó által megadott dátumok sorrendje nem felel meg, akkor a *"The order of the dates should be: deadline for application, start date, end date."* üzenet jelenik meg a felugró ablakban. A dátumok megfelelő sorrendje a következő: jelentkezési határidő, kezdődátum és a legkésőbbi a befejezési dátum. A dátumok ellenőrzésére hívja fel a figyelmet a gomb fölötti felirat is.

Ha egy kötelező adat hiányzik, akkor az adat címkéje és a *is required* felirat jelenik meg. Ilyen kötelező adatok a név és a dátumok.

Vendéglista lekérésekor

Az események szerkesztésekor beállítható, hogy összefüggő hálózatot alkotó vendéglistát szeretnénk létrehozni, vagy sem. Amennyiben igen, viszont a szervezet összes felhasználója által alkotott hálózat nem összefüggő, akkor erre figyelmeztet az alkalmazás. A figyelmeztető üzenet a felugró lista fölött jelenik meg. A listát ilyenkor is létrehozza az alkalmazás, meg is jeleníti, csak nem egy összefüggő hálózatot adnak ki a felsorolt vendégek.

3.3. A weblap felhasználói dokumentációja

3.3.1. Futási környezet

A weblap használatához a felhasználónak mindössze egy böngészőre, illetve egy mobil/asztali eszközre (kliens) van szüksége. Maga a program egy szerveren fut. A program futtatásához Kestrel Web Server használható, mely eleve bele van építve az ASP.NET Core alkalmazásokba, így ebbe az alkalmazásba is. Nem feltétlen igényel grafikus felületet a szerver gépen. A következő terminál utasításokkal használható a projekt:

- fordítás, futtatás: dotnet run
- fordítás: dotnet build

- kihelyezés: dotnet publish –o out_directoryname
- futtatás: dotnet WebAppName.dll

Mindehhez a .NET Core Runtime szükséges[4]. A szervergép operációs rendszere lehet Linux vagy Windows. A biztonság növelhető távoli webszerverek használatával, mint proxy-k[5]. Ilyen lehetőséget biztosít például a Microsoft Azure Web Service vagy IIS is, melyek használatához már telepíteni kell a Visual Studio 2017-es vagy annál későbbi verzióját. Az adatok tárolásához és kezeléséhez használható Visual Studio vagy egy MySql program. Az adatkezelés és a webalkalmazás futtatása történhet két külön szerveren is, mivel a perszisztencia és az üzleti logika külön réteget alkotnak az alkalmazásban.

Az alkalmazás memóriaigénye a felhasználók, a cégek, és a cégek által feltöltött táblázatoktól függ. A minimális követelmény a hardveres eszközökkel szemben pedig az elvárt válaszidőtől függ, illetve attól, hogy igénybe vesszük-e a Visual Studio által adta lehetőségeket. A következőket mindenképpen teljesítenie kell a szerver gépnek abban az esetben, ha Visual Studiot használunk, telepítjük hozzá a MySql extensiont és IIS-re helyezzük ki az alkalmazást[6][7]:

- Processzor: 1.8 GHz, két mag, x86 vagy x64
- RAM: 2 GB minimum, de 4 GB ajánlott
- merevlemez: 130 GB szabad tárhely. A Visual Studio telepítés 20 GB-ot vesz igénybe,
- Video kártya: minimum 720p felbontást támogató
- Operációs rendszer
  - Windows 7 SP1 and later
  - Windows 8.1
  - Windows 10 Version 1607 and later
  - Windows Server 2008 R2 SP1 (Full Server vagy Server Core)
  - Windows Server 2012 SP1 (Full Server vagy Server Core)
  - Windows Server 2012 R2 (Full Server vagy Server Core)
  - Windows Server 2016 (Full Server, Server Core, vagy Nano Server)
  - Mac OS X 10.11, 10.12*
  - Red Hat Enterprise Linux 7
  - Ubuntu 14.04, 16.04, 17

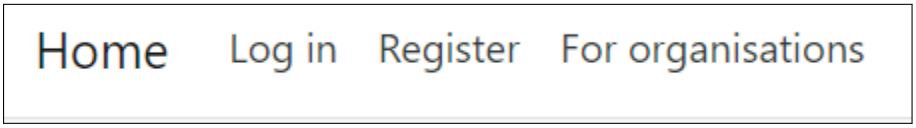
A webalkalmazás futtatásának legegyszerűbb módja a solution megnyitása Visual Studio 2019-ben, a fenti eszközsorban startup project-ként a *WebAppForMembers* project kiválasztása és a Start gombra való kattintás.

A Visual Studio-ban vagy a MySql-ben a megfelelő port-on létre kell hozni az adatbázishoz való hozzáférést biztosító kapcsolatot. Az alkalmazás futtatásakor létrejönnek az adatbázis sémák és táblák. A táblákat a felhasználók töltik fel adatokkal.

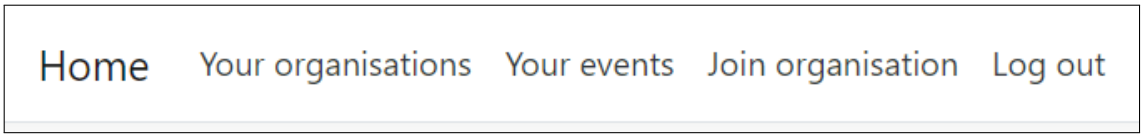
3.3.2. Használati útmutató

A webalkalmazás bármelyik oldalára navigálva a lap tetején egy menüsor jelenik meg. A menü állandó eleme a *Home* aloldalra vezető menüpont. A nem bejelentkezett felhasználók számára ez a menüsor a *For organisations*, *Register* és *Log in* menüpontokat jeleníti meg és teszi elérhetővé. Ez a menüsor a 3.1 ábrán látható, illetve megjelenik például a kezdőoldalon, melynek képernyőképe a 3.3 ábra. Minden oldalon a képen látható helyen helyezkedik el a menüsor, függetlenül attól, hogy bejelentkezett vagy bejelentkezetlen felhasználóról van-e szó. A bejelentkezett felhasználók a *Your organisations*, *Events*, *Join Organisation*, és *Log out* menüpontokat láthatják, lásd a 3.2 ábrát. Mindkét esetben minden menüpont más aloldalhoz irányít rájuk kattintva.

Azt, hogy melyik oldalon járunk a böngésző tabjában, a 3.3 ábrán a bal felső sarokban látható módon írja ki az alkalmazás, illetve bizonyos oldalaknál az oldal címe is utal a megkattintott menüpontra.



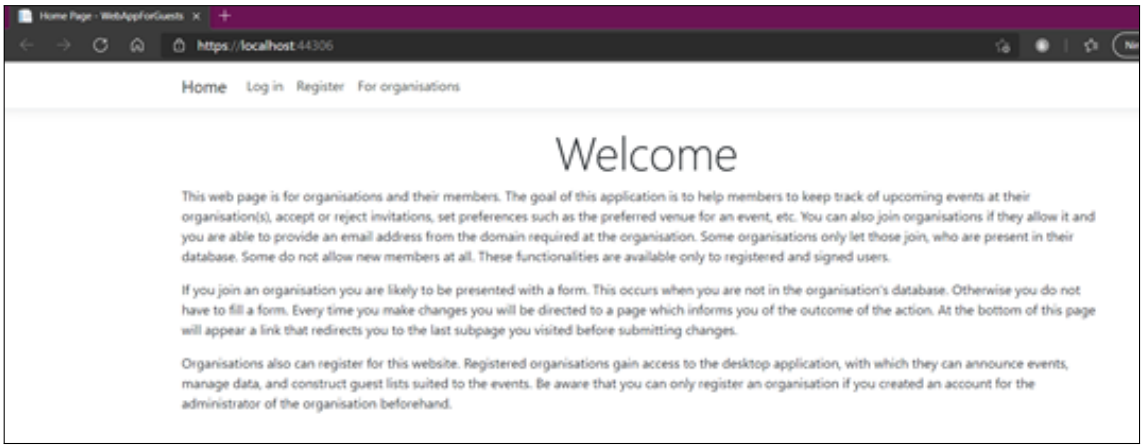
3.1. ábra. Menüsor nem bejelentkezett felhasználók számára



3.2. ábra. Menüsor bejelentkezett felhasználók számára

Home

A *Home* című oldal a kezdőoldal, mely egy rövid leírást ad a felhasználó részére az alkalmazás funkcióiról és azok elérhetőségéről. A menüsoron kívül csak statikus elemeket tartalmaz.



3.3. ábra. Home page

Register

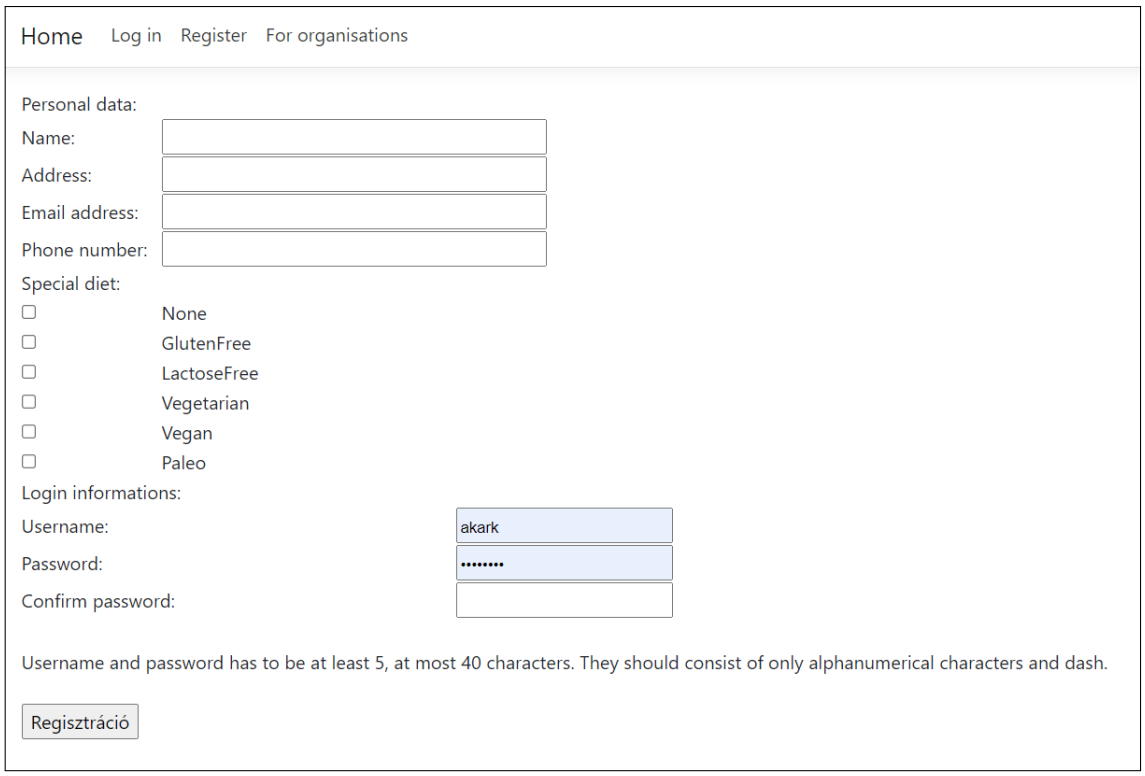
A regisztrációs felületen egy form-ot kell kitöltenie a felhasználónak. Minden beviteli mező új sorban van, és a hozzájuk tartozó címkék tőlük balra, velük egy sorban helyezkednek el. A címkék segítik a felhasználót az oldalon való tájékozódásban. A beviteli mezőktől jobbra jelennek meg a felvitt adatok hitelesítésekor fellépő problémák a felhasználók számára könnyen értelmezhető üzenetek formájában. Ezen hibakiírások támpontot adnak az űrlap helyes kitöltéséhez, és akkor is megjelennek, amikor a felhasználó még nem végzett a kitöltéssel, még nem nyomta meg a lap alján található *Register* feliratú gombot, az adatok beküldésére szolgáló gombot. Ha hibásan kitöltött űrlapnál kattint a beküld gombra, akkor minden változtatás nélkül továbbra is az űrlapot mutatja az oldal a megfelelő hibaüzenetekkel. Az űrlap alján, a jelszó (Password) címkéje alatt található a jelszóra vonatkozó kritériumok leírása.

A következő információkat kéri be a regisztrációs felület:

Címke	**Bekért adat**	**Kötelező**	**Megjegyzés**
Name	felhasználó valódi neve	igen	

Address	felhasználó lakcíme	nem	
Email	felhasználó email címe	igen	
Phone number	felhasználó telefonszáma	nem	
Special diet	egy check-box listából kiválaszthatja a felhasználó, milyen speciális étrendet követ, többet is egyszerre.	igen	Pipa jelzi a kiválasztott elemeket. Kitöltése segítheti az események szervezőinek munkáját. *None* kiválasztásával a többi lehetőség nem kerül kiválasztásra és nem kattintható a jelölő négyzetük. Ezt a pipák törlődése és a négyzetek elhalványulása jelzi.
User name	felhasználónév	igen	Bejelentkezéskor lesz újra szükség rá.
Password	a létrehozandó felhasználói fiókhoz való hozzáférést biztosító jelszó.	igen	Bejelenkezéskor lesz rá újra szükség, csak a hozzátartozó felhasználónévvel együtt érvényes.
Repeat password	a feljebb megadott jelszó újbóli begépelését igényli	igen	Nem kerül mentésre. Ha nem egyezik a jelszóval, figyelmezteti a felhasználót. A felhasználót segíti annak elkerülésében, hogy elgépelje az általa kigondolt jelszót, és a következő bejelentkezést megnehezítse vagy el is lehetetlenítse ezzel.

3.1. táblázat. Regisztrációkor bekért adatok



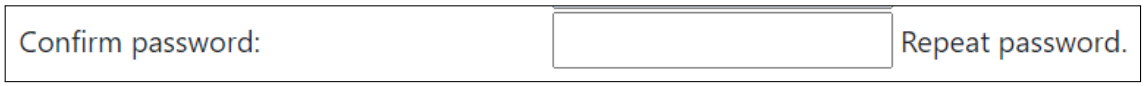
3.4. ábra. Regisztrációs oldal

A következő felsorolás foglalja össze mely esetben ír ki a program segítő üzenetet a felhasználó számára:

- kötelező mezőt üresen hagyta
- email cím formátuma nem megfelelő
- telefonszám formátuma nem megfelelő
- felhasználónév nem elég hosszú
- felhasználónév túl hosszú
- már létezik fiók a beírt felhasználónévvel
- jelszó nem elég hosszú
- jelszó túl hosszú
- jelszó nem megengedett karaktereket tartalmaz
- jelszó nem tartalmaz számot
- jelszó nem tartalmaz nagy betűt
- megismételt jelszó és jelszó nem egyeznek
- a jelszót csak egyszer adta meg, azaz a Confirm password mező üres. Ld. 3.5 ábrán.

Ezek az üzenetek angolul, közvetlenül az éppen hibásan kitöltött beviteli mező mel-

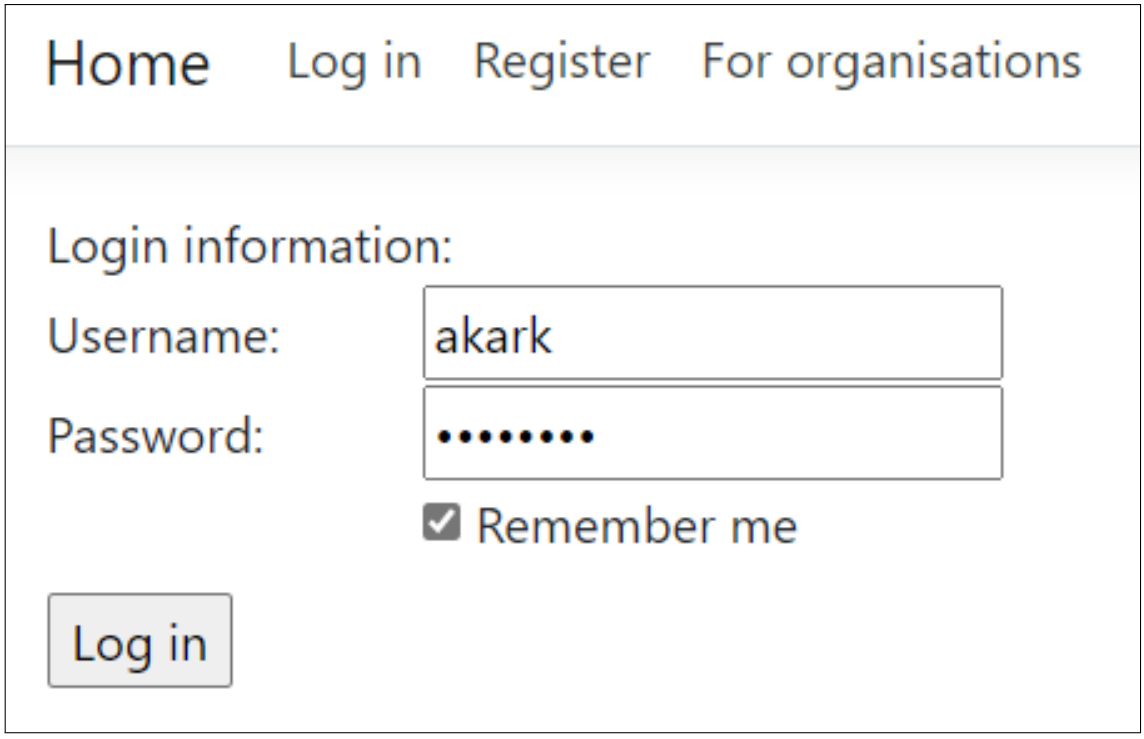
lett közvetlenül az 3.5 ábrán látható módon jelennek meg.



3.5. ábra. Példa felhasználót az adatok kitöltésében segítő figyelmeztető üzenetre. Az üzenet a beviteli mezőtől jobbra helyezkedik el.

Amennyiben helyesen töltjük ki az űrlapot és rákattintunk a *Register* feliratú gombra, új felhasználót hoz létre az adatbázisban és be is jelentkezik vele. Így már a bejelentkezett felhasználók számára elérhető menüpontok jelennek meg a lap tetején a menüsorban, illetve a kezdőoldalra irányít az alkalmazás.

Log in



3.6. ábra. Bejelentkező oldal

Csak már regisztrált felhasználónévvel és a hozzátartozó jelszóval lehet bejelentkezni. Hibás jelszó és hibás felhasználónév esetén is ugyanazt az üzenetet írja ki: "Invalid username or password". Egy regisztrált felhasználónak ez elég információ a hiba korrigálásához, egy fiók feltörésével próbálkozót pedig nem juttat közelebb

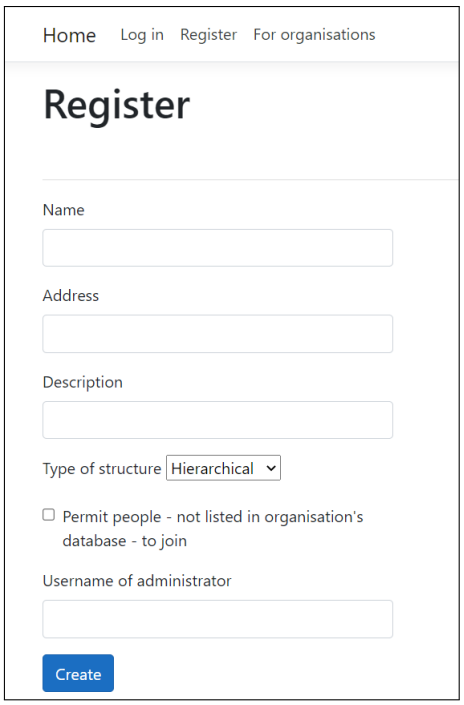
a céljához. A bejelentkezéshez szükséges adatokat meg is jegyeztetheti a böngészővel a felhasználó, ha kipipálja a *Remember me* felirathoz tartozó jelölőnégyzetet, így a következő bejelentkezés alkalmával a felhasználónév beviteli mezőjére kattintva megjelenik a beviteli mező alatt az elmentett felhasználónév, lásd a 3.6. Erre kattintva automatikusan kitölti a jelszót is és a bejelentkezés gombra kattintva bejelentkezhet. A bejelentkezést követően ismét a főoldalra navigál az alkalmazás, a menüsorban azonban a regisztrációért és bejelentkezésért felelős oldalakra irányító linkek helyett a csak a bejelentkezett felhasználók által igénybe vehető funkciókért felelős oldalak linkjei jelennek meg a megfelelő feliratokkal.

For organisations

Ez az oldal a cégek számára nyújt regisztrációs felületet. A következő adatokat kéri be: a cég nevét (*Name*), címét (*Address*), hogy milyen domain-ről csatlakozhatnak hozzá felhasználók (*Accepted email domains from users*), a szervezet szerkezetének típusát (*Type of structure*) és a szervezet belső információinak kezeléséért felelős személy fiókadatai közül az email címére és a felhasználónevére van szükség. Tehát egy cég regisztrálása előtt kötelező a cég adminisztrátorának fiókját létrehozni. Az adatok egy leírással (*Description*) is kiegészíthetők, melyet a felhasználók láthatnak. A szervezet szerkezete két típusú lehet: az egyik a hagyományos fa szerkezet, melyben egy ember kivételével minden személynek egy felettese van; a másik projekt alapú, azaz a munkakapcsolatokat az határozza meg, ki milyen projekten dolgozik. Előbbi típust a *Hierarchical* felirat, a másikat a *Project based* felirat jelöli a legördülő menüben. Érvényes kitöltést követen a cég adminisztrátora jogosult az asztali adminisztrációs alkalmazás letöltésére és használatára. A letöltéshez be kell jelentkezni a weboldalon és a menü *Download desktop application* pontjára kell navigálni. Ennek hatására letöltődik a program. Kettőt kattintva rá elindul az alkalamazás.

Log out

Csak bejelentkezett felhasználó látja a menüben. Rákattintva kijelentkezik az alkalmazás az adott felhasználó fiókjából, a főoldalra navigál, a menüsorban csak a *Register* és *Log in* feliratok jelennek meg. A kijelentkezett felhasználó regisztráció nélkül újra bejelentkezhet a *Log in* menüpontra kattintva.



3.7. ábra. Regisztrációs felület szervezetek számára

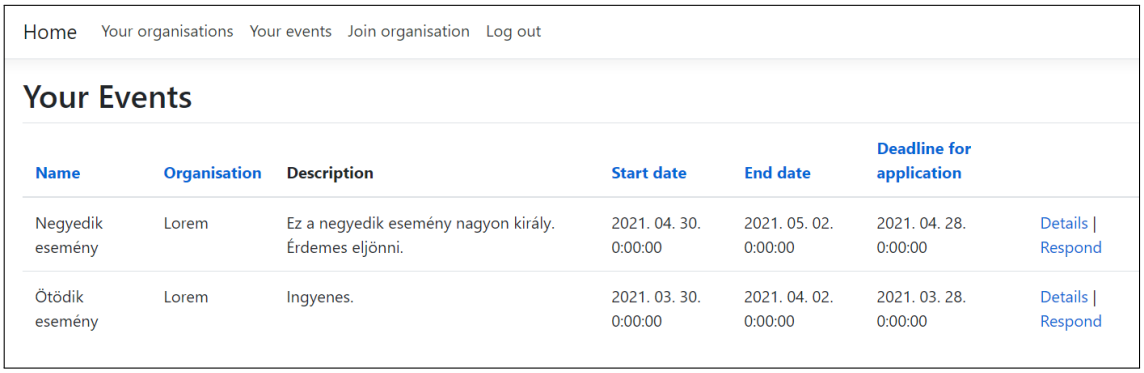
Events

Az oldalra csak bejelentkezett felhasználó navigálhat. Az összes, a felhasználóval már kapcsolatban álló szervezet által meghirdetett aktuális esemény legfontosabb adatai láthatók az oldalon táblazatba foglalva a 3.8 ábrán látható módon. A táblázat fejlécében szereplő feliaratok a *Description* kivételével kattintható linkek, melyek az eseményeket a megkattintott felirat által indikált szempont szerint rendezi. A lehetséges szempontok közé tartozik az esemény neve, kezdődátuma és a válaszadás sürgőssége.

Minden esemény neve mellett található egy *Details* feliratú link, melyre kattintva egy olyan oldalra kerülünk, melyen további tudnivalók jelennek meg az esemény kapcsán, így például a lehetséges helyszínekről is itt informálódhatunk, az oldal alján található egy *Back to Your Events* link, mely visszavezet az eseményeket felsoroló oldalra.

Az eseményeket felsoroló oldalon ezen kívül megjelenik a meghívások mellett a *Respond* link is. Rá kattintva a 3.9 ábrán látható űrlap jelenik meg. Az űrlapon található egy checkbox, mellyel jelezheti a felhasználó, hogy részt kíván-e venni az eseményen. A rádiógomb csoport segítségével szavazhat egy helyszínre, ha több

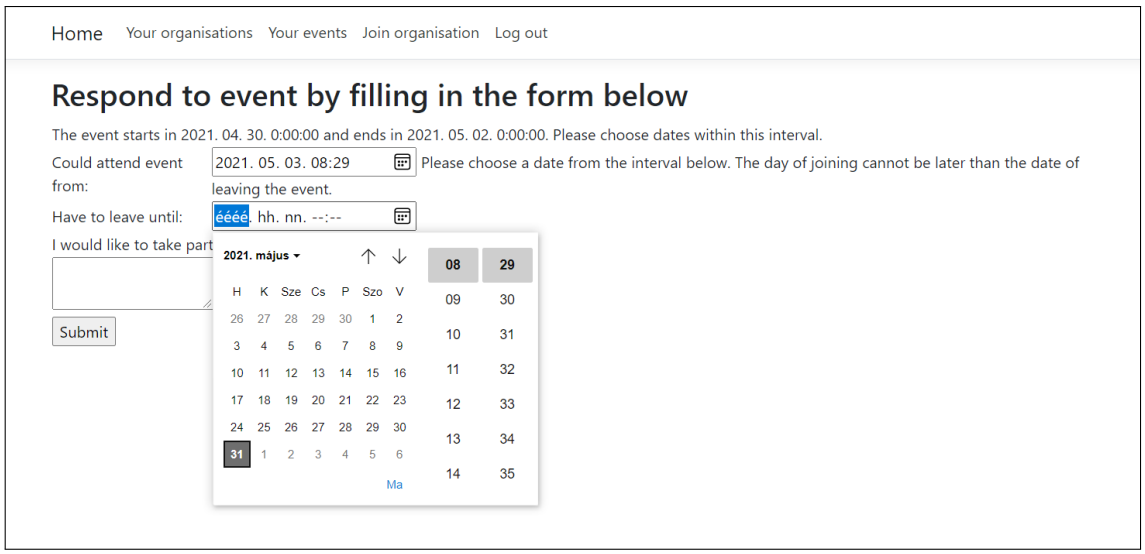
lehetséges helyszínt adott meg a szervezet az adott eseményhez. Az adott helyszínhez tartozó rádiógomb mellett látható a helyszín neve és alapértelmezett képe. Ha nincs beállítva alapértelmezett kép, akkor egy kérdőjelet tartalmazó ábra jelenik meg. A dátumválasztó beviteli mezők jobb végén látható egy-egy naptár szimbólum. Erre kattintva egy naptárkiválasztó felület nyílik le a beviteli mezőből, ahol a dátumot és az időt is kiválaszthatja a felhasználó. Az első ilyen mezőben azt kell megadni, hogy mikortól ér rá a felhasználó az adott eseményre, a másodikban pedig azt, hogy meddig. Ügyelni kell arra, hogy megfelelő időrendben kövessék egymást az esemény kezdő dátuma, a felhasználó által megadott dátumok és az esemény végét jelző dátum. A lap tetején a menü alatt látható, hogy pontosan mettől meddig tart az esemény. Megjegyzést is írhat egy szövegbeviteli mezőbe. A form alatt, a lap alján található egy az előző oldalra visszavezető link a *Back to viewing event details* felirattal. A form a lap alján szereplő *Submit* feliratú nyomógombbal küldhető be. A beküldést követően az alkalmazás visszairányít az eseményeket listázó oldalra.



3.8. ábra. Eseményeket listázó oldal

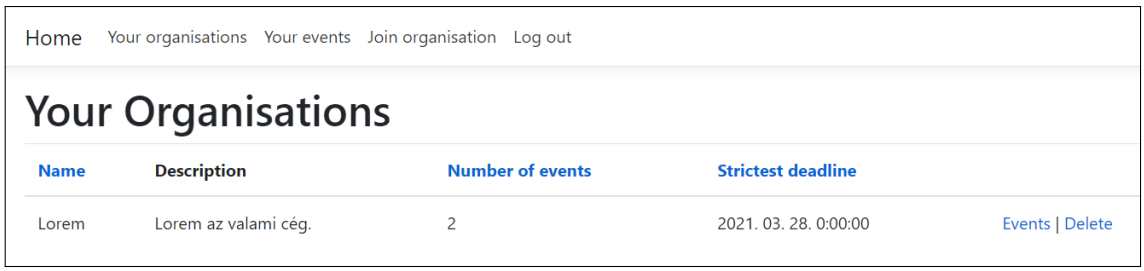
Your organisations

Az oldalra csak bejelentkezett felhasználó navigálhat. Táblázat formájában jelennek meg azon szervezetek adatai, melyekhez már csatlakozott a felhasználó. Az oldal elrendezése a 3.10 ábrán látható. A táblázat fejlécében lévő feliratok az *Description* kivételével kattintható linkek, melyek a szervezetek rendezését teszik lehetővé (a link feliratának megfelelő szempont alapján). A *Strictest deadline* szempont szerint csak növekvően rendez sorba, a másik két szempont alapján kattintásonként változik, hogy növekvő vagy csökkenő sorrendet alakít ki.



3.9. ábra. Űrlap egy eseményre való reagálásra

Minden közösség sorában szerepel egy *Quit* és egy *Events* feliratú link. Előbbire kattintva a felhasználó kiléphet az adott szervezetből. Ekkor az alkalmazás egy új oldalon - ahol csak az adott szervezet adatai jelennek meg - kér megerősítést a művelet elvégzésére. Az oldal alján megjelenő link ad lehetőségét az előző oldalra való visszalépésre. Kilépési szándékunk megerősítéséhez a *Delete* feliratú piros gomb megnyomása szükséges. Kilépést követően az előző oldalra irányít vissza, ahol már nem jelenik meg az érintett szervezet a felhasználó számára. A *Events* feliratra kattintva az alkalmazás egy olyan oldalra navigál, mely az adott szervezet adatai alatt az általa meghirdetett eseményeket sorolja fel azok legfontosabb adataival. A *Quit* link itt is szerepel a szervezet neve mellett. Az események mellett most is a *Details*, *React*, a *Want to come*, *Decline* és az *Edit reaction* feliratú linkek jelenhetnek meg az *Events* oldalnál tárgyalt feltételekkel és eredményekkel. Továbbá a rendezési- és szűrő beállítások, azok kezelése és megjelenése is megegyezik az *Events* oldalnál látottakkal. A lap alján található *Back to Your organisations* link visszavezet a szervezeteket kilistázó oldalra. Az eseményre kattintva ugyanaz látható, mint mikor az *Events* oldalon kattint a felhasználó az eseményre, annyi különbséggel, hogy itt csak a kiválasztott szervezet által meghirdetett események jelennek meg.



3.10. ábra. Felhasználó szervezeteit listázó oldal

Join organisation

Az oldalra csak bejelentkezett felhasználó navigálhat. Két beviteli mezőt tartalmaz. Az elsőbe annak a szervezetnek a nevét kell begépelni, melyhez csatlakozni kívánunk. Amennyiben az adatbázisban nem létező nevet gépel be a felhasználó, figyelmeztető üzenet jelenik meg erről. A másik beviteli mezőbe azt az email címet kell beírni, amelyet a szervezetnél használ. Ha még nem szerepel a szervezet adatbázisában és a szervezet regisztráláskor engedélyezte az utólagos csatlakozást a felhasználók számára, akkor a második beviteli mező üresen hagyásával és a *Join organisation* gomb megnyomásával (és az első beviteli mező helyes kitöltésével) egy űrlap jelenik meg. Az űrlap a következőket kéri be a felhasználótól:

Címke	**Bekért adat**	**Kötelező**	**Megjegyzés**
Your jobtitle/role:	felhasználó szerepe vagy beosztása a szervezetnél	igen	Legördülő listában jelennek meg a lehetséges opciók. Amennyiben egyik opció sem felel meg, a legördülő menüvel egy sorban egy szövegbeviteli mezőbe gépelheti be beosztásának nevét.
Date of joining	a szervezethez való csatlakozás (nem az odldalon keresztüli csatlakozás) dátuma	igen	

Department	a felhasználót foglalkoztató részleg neve, amennyiben ilyen van	nem	
Name of Your Supervisor	felettesének neve	nem	A szervezet típusától függ, hogy megjelenik-e.
Email of Your Supervisor	felettesének email címe	nem	A szervezet típusától függ, hogy megjelenik-e. Csak akkor érvényes, ha a szervezet adatbázisában szerepel olyan tag, akinek a neve egyezik a *Name of Your Supervisor* mezőbeli névvel, email címe pedig az itt megadottal. Hiba esetén figyelmezteti a felhasználót.
Email	felhasználó email címe, melyet a továbbiakban a cégnél használni kíván	igen	

Címke	Bekért adat	Kötelező	Megjegyzés
Projects	a létrehozandó felhasználói fiókhoz való hozzáférést biztosító jelszó.	igen	A szervezet típusától függ, hogy megjelenik-e. Legördülő listában jelennek meg a lehetséges opciók. Amennyiben egyik opció sem felel meg, a legördülő menüvel egy sorban egy szövegbeviteli mezőbe gépelheti be projektjének nevét és az *Add project* gombbal adhatja hozzá a legördülő listához, automatikusan kijelölve azt. Amennyiben mégse kívánja megjelölni, a legördülő listában ezt korrigálhatja. Több új projekt is megadható egyesével begépelve és hozzáadva a listához.

3.3. táblázat. Szerevezethet való csatlakozáskor bekért adatok

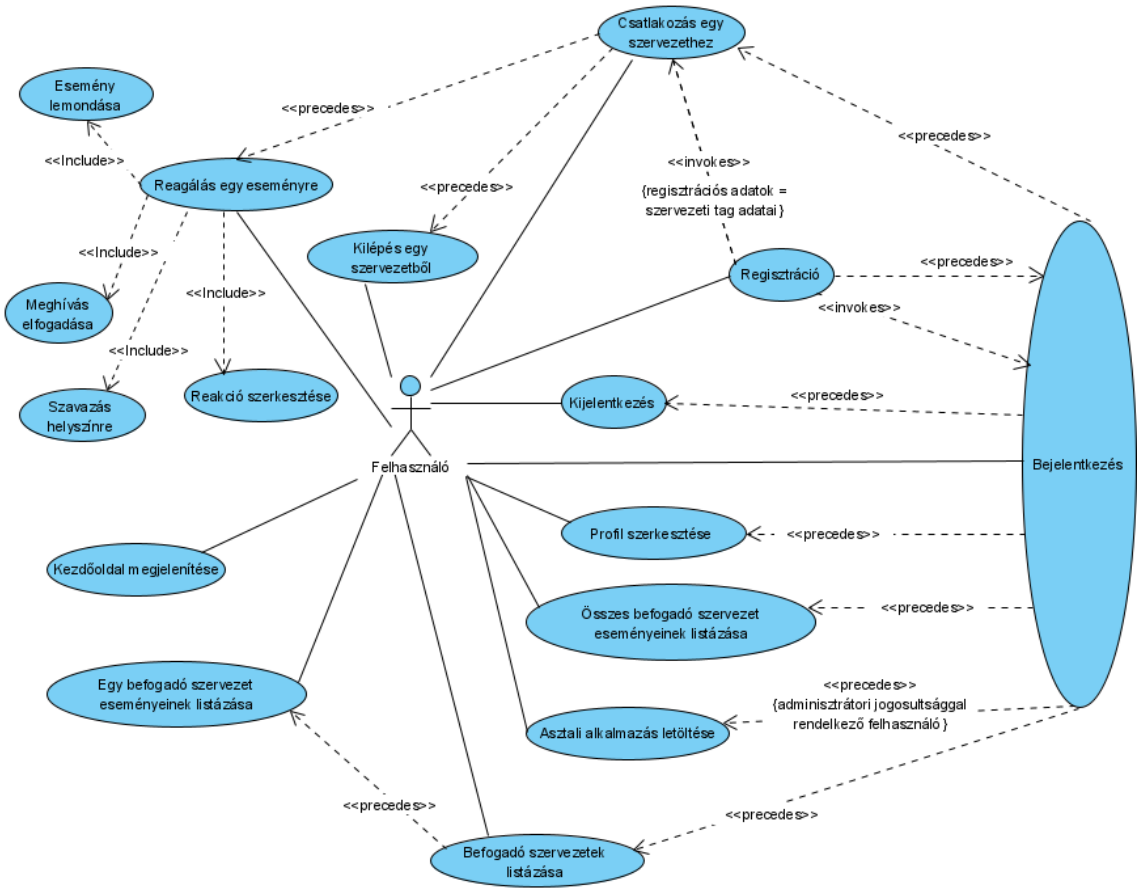
3.3.3. Felhasználói esetek

Aktorok:

- alap felhasználó

Funkciók:

- regisztráció
- bejelentkezés
- kijelentkezés
- befogadó szervezetek listázása
- befogadó szervezetek által meghirdetett események listázása
- kilépés egy szervezetből
- csatlakozás egy szervezethez
- reagálás eseményekre



3.11. ábra. Felhasználói eset diagram

4. fejezet

Fejlesztői dokumentáció

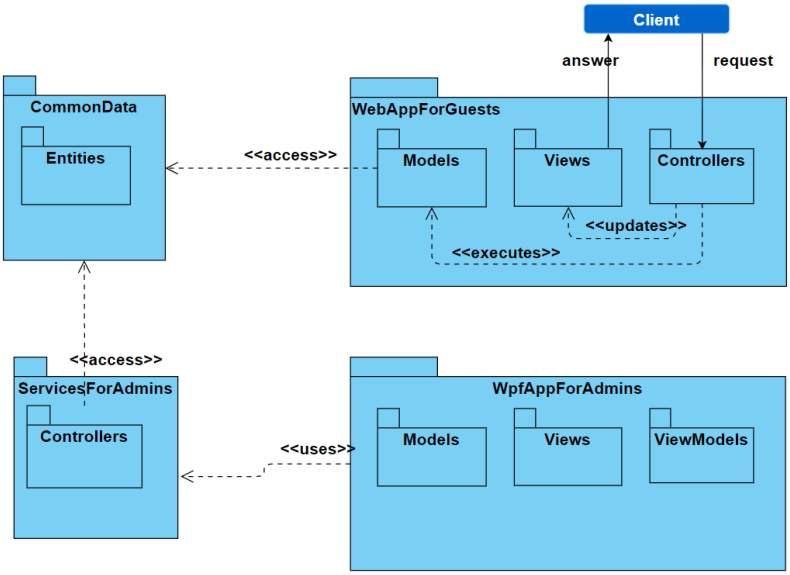
4.1. Fejlesztői környezet

A programkódot a Visual Studio 2019 integrált fejlesztői környezetben írtam, de használhatók más IDE-k is (például Visual Studio 2012, Visual Studio Community). Szükséges volt a .NET Framework telepítése (ezzel az ASP.NET Core is települ). A projekt változásait a Git verziókövető rendszerrel tartottam számon.

4.2. Alkalmazás szerkezete

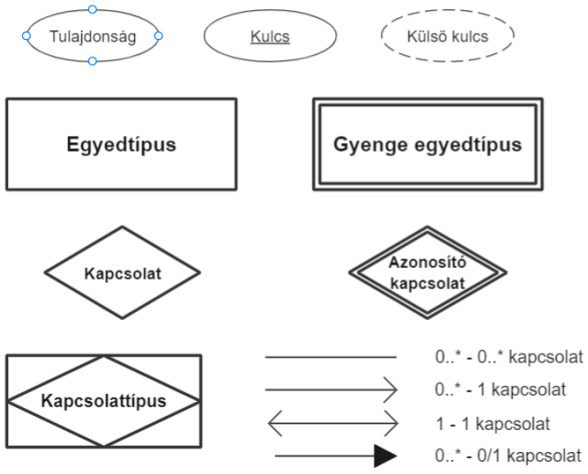
Az alkalmazás komponenseit és egyben a projekt felépítését, az alkalmazás könyvtárszerkezetét a 4.1 illusztrálja.

Az asztali alkalmazás és az online felületet biztosító alkalmazás ugyanazt az adatbázist használják. Ezt az adatbázist code first megközelítéssel konstruáltam. Az adatbázisban két fő entitás kap helyet: Member és Event. A 4.3 és 4.3 ábra mutatja be ER diagramon keresztül, hogyan kapcsolódnak a többi entitáshoz. Jelmagyarázatot a 4.2 ábra biztosít. Az ábrákon téglalappal jelzett entitásokat C# kóddal osztályokként implementáltam, a hozzájuk tartozó tulajdonságok pedig az osztályok property-jei. Az ER diagramban a sok - sok típusú kapcsolatokhoz segédentitásokat vezettem be, ezeket egy különleges téglalap jelzi. A kódban ezek is önálló osztályokként jelennek meg, ilyen például a MemberOfProject osztály is, mely segít, hogy egy Member több Project-hez kapcsolódhasson, és fordítva.

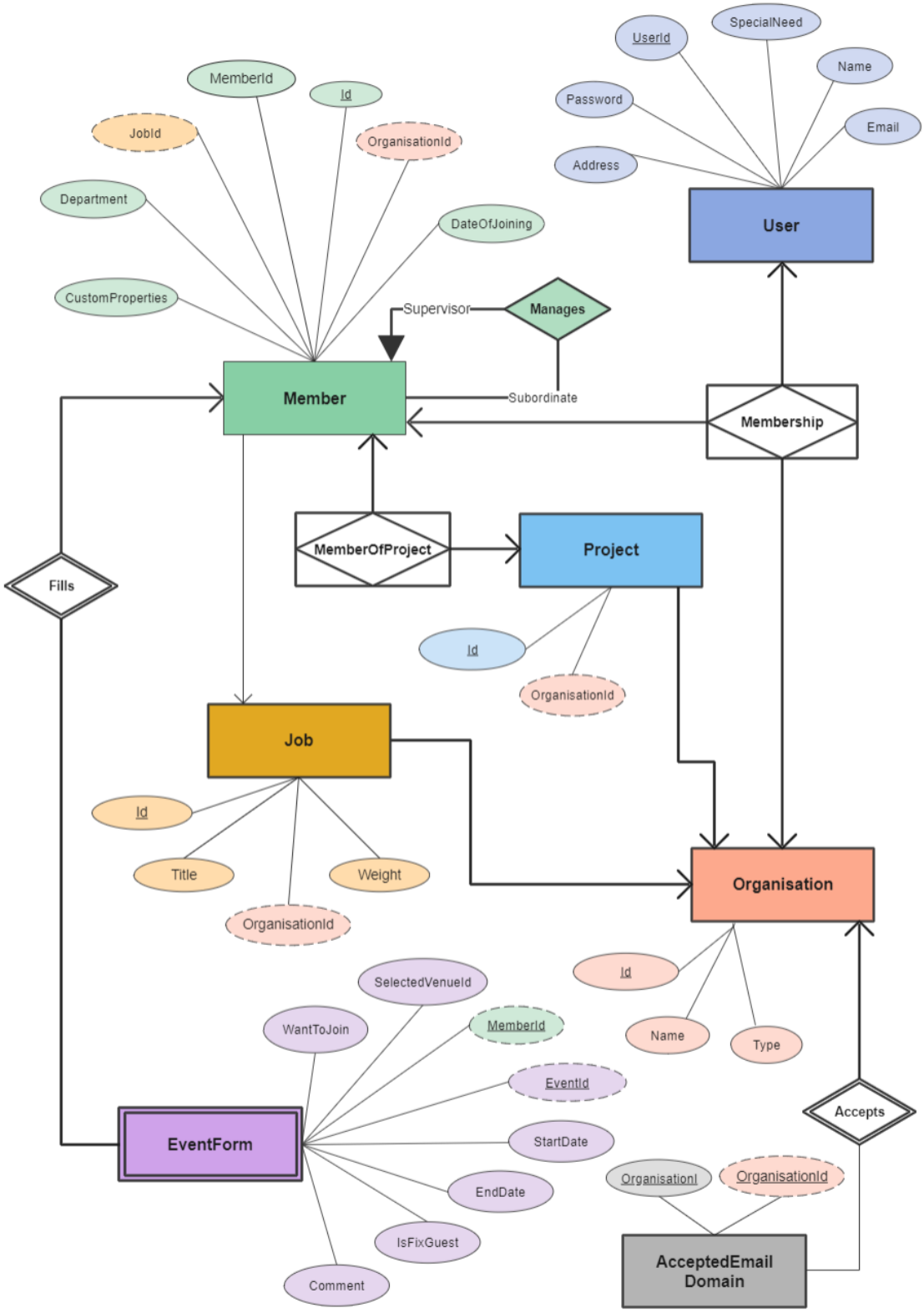


4.1. ábra. Az alkalmazás UML csomagdiagramja

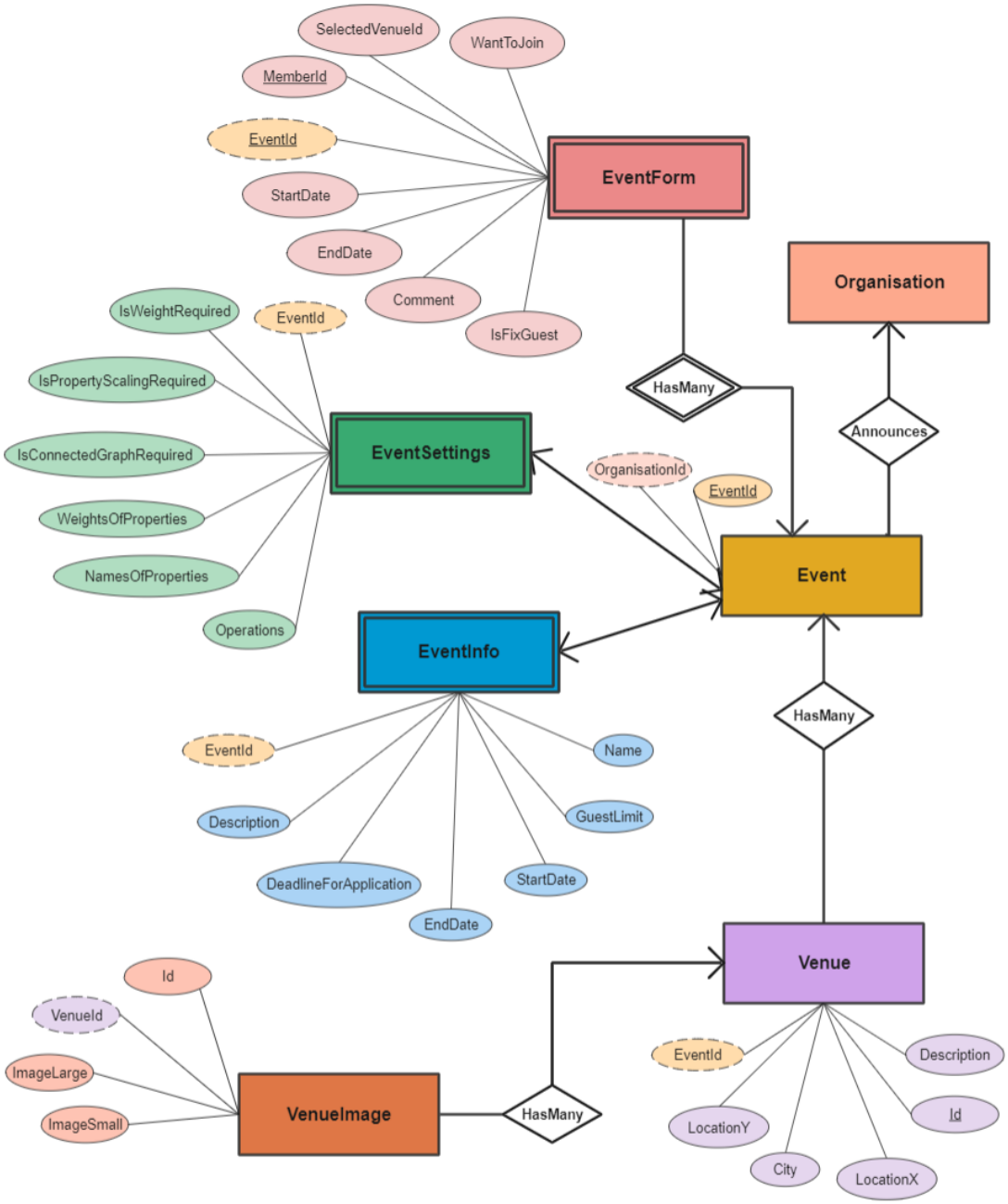
A kapcsolatokat a programkódban úgynevezett navigációs property-kel jeleztem. Ez annyit tesz, hogy amennyiben két entitás között van kapcsolat, úgy az egyikben létrehozunk egy property-t, mely a másik entitást leíró osztálynak egy példányát adja vissza. A felesleges körbehivatkozások elkerülése végett egy kapcsolatot mindig csak az egyik végpontján regisztrálunk navigációs property-vel, ez elég a kapcsolat meghatározásához.



4.2. ábra. ER diagram jelölései



4.3. ábra. Az adatbázis a Member entitás szempontjából



4.4. ábra. Az adatbázis az Event entitás szempontjából

4.3. Az asztali alkalmazás fejlesztői dokumentációja

4.3.1. Használt architektúra

Asztali környezetben a felhasználó a nézettel teremt kapcsolatot, amely biztosítja a megfelelő utasítás végrehajtását (eseménykezelő, parancs). Ehhez nyújt megfelelő szerkezetet az MVVM architektúra, ezért ezt alkalmaztam.

Modell

Az alkalmazás logikáját írja le, illetve az adatkezelésért felelős. Az adatkezelést biztosító osztályok és interfészek a Persistence mappában találhatók, a MeetingOrganiserDesktopApp.Persistence namespace-ben. Az alkalmazás logikájáért felelős osztályok a Model mappában, MeetingOrganiserDesktopApp.Persistence namespace alatt találhatók.

A perzisztencia réteg könnyebb cserélhetőségét biztosítja az IMeetingApplicationPersistence interfész, ugyanis a MeetingApplicationModel osztályban egy ilyen interfészt használva elég a modell számára, ha az modell konstuktorában átadott perszintenciáért felelős osztály az interfészben deklarált metódusokkal megegyező szintaxisú (az argumentumok és a visszatérési értéke megegyezik), nem számít, hogy hogyan implementálták. Tehát nem csak egy szervízzel kommunikáló osztályt lehetne neki átadni, mint amilyen a MeetingApplicationPersistence osztály is, hanem például egy közvetlen valamilyen adatbázissal kapcsolatban álló osztály is megfelelne. A nézetmodellt ugyanezt a logikát követve, interfész segítségével (IMeetingApplicationModel) függetlenítettük a modell megvalósításának módjától.

A modell réteg központi osztálya a MeetingApplicationModell. Az osztály felhasználja a gráfokat és gráfműveleteket megvalósító osztályokat. A modell több metódusa is a tárolt adatok segítségével történő gráfépítést implementálja.

Nézetmodell

Proxy-ként szolgál modell és nézet között - a modell által előállított eredmények (adatok) alapján a felhasználó számára megjeleníteni kívánt állapotot ír le, melyet a nézet fog megjeleníteni. A nézetállapotot írja le, és annak frissítéséért felelős. Tartalmazza a tevékenységek végrehajtásához szükséges parancsokat. A nézetmodell réteg központi osztálya a MainViewModel osztály, melyet több nézet számára is használtam kontextusként.

Nézet

Megjeleníti a hozzá adatkötéssel kapcsolt nézetmodellben leírt állapotot. Az adatkötésnek köszönhetően a nézet automatikusan értesül és frissül a nézetállapot,

azaz a nézetmodell változásakor.

A nézet rétegben a Window típusból leszármazó osztályok mellett helyet kaptak a UserControl-ból leszármazó osztályok is. Ezek biztosítják a cserélhető felületet a főablakon. Erre azért van szükség, hogy a főablak állandó elemei valóban állandók legyenek, és csak azok frissüljenek, melyeket a felhasználó kért. A List events menüpontra kattintva az események adatait megjelenítő UserControl jelenik meg a főablakon belül, míg a List members menüpontra kattintva a szervezet tagjainak adatait megjelenítő UserControl, a menüsor sosem változik.

Az összetett tevékenységeket - például nézetek létrehozását - egy külön alkalmazás vezérlés (application control) biztosítja (az App.xaml.cs fájlban található Application osztályból leszármazó App osztály).

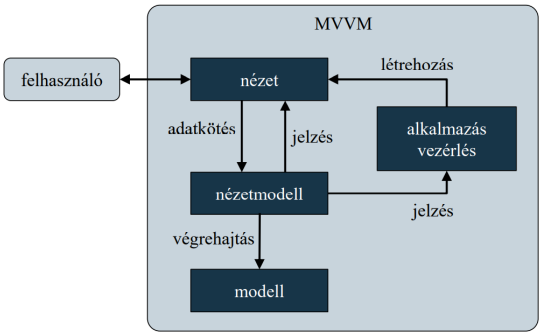
MVVM architektúra előnyei

A felhasználói felületet deklaratívan leíró nézet implementációja teljesen független az alkalmazás logikájáért (tevékenységek, adatkezelés) felelős modell implementációjától, így a következő előnyök élvezhetők az architektúra alkalmazásával:

- ugyanahhoz a nézethez más modellt (például plusz funkciókkal) társíthatunk, és fordítva
- könnyebben karbantartható, áttekinthetőbb a program szerkezete
- könnyen megosztható a munka frontend és backend fejlesztők között

Hátrányai:

- alapos tervezést és sok beépített elemet igényel
- a nézetmodell kompatibilitását a nézettel csak futási időben tesztelhetjük



4.5. ábra. MVVM komponensei és azok kapcsolata

Összefoglalva tehát a rétegeket az App.xaml.cs fájlban található Application osztályból leszármazó App osztály köti össze. Itt hozzuk létre a nézetmodelleket és a nézeteket és itt adjuk át a nézetek számára adatkontextusként a nézetmodelleket. A Window osztályból leszármazó nézetmodell osztályok pedig tartalmazzák az alkalmazás modelljét, a MeetingApplicationModel osztályt. A modell és az őt tartalmazó nézetmodell kommunikációért felelős eseménykezelőket a nézetmodellben hozzuk létre és kötjük be. A nézetmodell közvetítőként is szolgál a modell és a nézet között, ugyanis a nézetmodellben találhatók az ICommand interfészt megvalósító DelegateCommand osztályból leszármazó kommandok. Minden a felhasználó által indítható tevékenységhez tartozik egy ilyen kommand. Például a CreateEventCommand által a felhasználó számára biztosított tevékenység egy esemény létrehozása. Adatkötéssel (Binding) a nézet osztály - esetünkben a MainWindow - *Add new* feliratú gombjának lenyomására végrehajtódik a CreateEventCommand Execute metódusa. A nézetmodellben a kommand inicializálásakor egy lambda kifejezést adtunk át, melyben azt mondtuk meg, hogy a kommand végrehajtása szóljon arról, hogy hozzunk létre egy új EventDTO példányt, állítsuk be az alapértelmezett értékeit, majd hívjuk meg az OnEventEditingStarted metódust. Ez a metódus az új EventDTO példány szerkesztésének megkezdésének eseményét lereagáló eseménykezelőt hívja meg, melyet már nem a nézetmodellben, hanem az alkalmazás osztályban inicializálunk, méghozzá az alkalmazás osztály egy metódusát adjuk meg számára. Ez a metódus pedig létrehozza a megfelelő nézetet, melyben szerkeszthetjük az eseményt. Tehát a modell és a nézet közötti kommunikációért felelős eseménykezelőket a nézetmodellekben, a nézetmodell és a nézet közötti kommunikációért felelős eseménykezelőket pedig az App osztályban kötjük be.

4.3.2. Algoritmusok, adatszerkezetek és adatok

A (súlyoztt) gráf adatszerkezethez egyedi osztályokat hoztam létre NodeWeightedGraph, Graph és GenericGraph néven. A 2 fejezet algoritmusai is felhasználásra kerültek. Ezeket a gráf osztályok metódusaiként implementáltam. Továbbá szükség volt még a szélességi bejárás implementálására feszítőfa létrehozásához és összefüggőség ellenőrzéséhez, illetve halmazműveletek implementálására, mivel az általam ismert Enumerable típus halmazműveletei lassúak, például nem

használják ki a halmazok méretkülönbségét. Később megtudtam, hogy a HashSet típus nem ezeket a műveleteket alkalmazza, így érdemes lett volna inkább azokat használni újak implementálása helyett.

Két osztályt készítettem a gráfok csúcsainak reprezentálására. Az egyik a súlyt nem tartalmazó Node osztály, a másik pedig a belőle leszármazó, súllyal rendelkező WeightedNode osztály. Mivel a súlyozott és súlyozatlan gráfoknak vannak megegyező műveleteik, ezért egy közös, generikus ősből származtattam le a fent említett két gráftípust, hogy elkerüljem a kódismétlést. A template-es ős a GenericGraph osztály, mely egyetlen template paraméterrel rendelkezik, melyre kikötöttem, hogy mindenképp legyen a Node osztály leszármazottja (vagy maga a Node osztály).

Első megoldás

Mivel csúcsok lekérdezését és -beszúrását is várhatóan gyakran végzi el az alkalmazás, ezért a csúcsok tárolására a System.Collections.Generic névtér Dictionary osztályát használtam, hiszen Dictionary olvasó, beszúró és törlő művelete mind konstans idejű, így ez bizonyult a legjobbnak. Ezt a tulajdonságát annak köszönheti, hogy hash-elést használ címkiosztáshoz. A gráf csúcsait a Member osztály példányainak adataiból képezzük, ezek lesznek a csúcsokat tároló Dictionary értékei, kulcsait pedig a Member-ek azonosítói szolgáltatják. Ezt nyugodtan megtehetjük, hiszen mind a kulcsok, mind az azonosítók kitétele, hogy azok egyediek legyenek.

Második megoldás

Végül a fenti megoldást elvetettem. Az akkori tudásommal rá is kényszerültem arra, hogy a csúcsazonosítót (Id) ne illesszem a Node osztályba. Mivel találtam megoldást arra, hogy ezt mégis megtehessem, ezért már nem Dictionary-ben tárolom a gráfok csúcsait, hanem HashSet-ben. Ez szintén biztosítja az elemek egyediségét, a konstans idejű beszúrást, törlést és olvasást. Hátránya, hogy az Id szerinti keresés lassabb. Az új ismeret, mely az átszervezést lehetővé tette számomra az a generikus osztályokhoz kapcsolódik. A probléma az volt, hogy ha egy Node rendelkezik Id-val, akkor azt illene megadni a konstruktorában. Viszont egy template paraméternél nem tudom kikötni, hogy milyen paramétereket váró konstruktorral rendelkezzen, maximum azt, hogy legalább paraméter nélküli konstruktorral rendelkezzen. Az elő-

ző megoldásban ezt a kikötést alkalmaztam is, ott egyszerűen a template paraméter mellé kellett írni *: new()*. Most nincs rá szükségem, töröltem is ezt a kikötést. Helyette a template paraméter típusú osztály létrehozását igénylő műveleteknél nem csak egy Id-t kérek be, hanem egy lambda kifejezést is, mely megadja, hogy melyik típusú csúcs osztály konstruktorának adjuk át az Id-t. Például a generikus ősosztály Add metódusa egy Id-t és egy lambdát vár. Ha lambdaként ezt a *id => new Node(id)* kifejezést adjuk meg, akkor Node típusú csúcsot szúr be.

4.3.3. Használt nyelvek

A nézeteket XAML-ban, a nézetmodell és modell osztályokat C#-ban írtam.

4.3.4. Tesztelés

Egységtesztek

A tesztelt osztályok minden publikus metódusát többféle bemenetre is teszteltem. Ezek a Node, WeightedNode, GenericGraph, Graph, NodeWeightedGraph osztályok. Több pozitív és negatív tesztesetre is megvizsgáltam ezen osztályok metódusait, így összesen 51 tesztesetre készült autotest. Negatív eseteknél azt ellenőriztem, hogy a megfelelő, az osztály számára sajátkezűleg létrehozott kivételtípusokat dobja-e az osztály. Azért hoztam létre külön kivételekt, mert egyrészt felhasználó barátabb hibaüzeneteket generálhatok így, de főleg egy fejlesztő számára nyújt nagy segítséget. Osztályonként külön tesztosztályt hoztam létre a TestMeetingOrganiserDesktopApp projektben. A tesztek nevei a TesztteltMetódus_Teszteset alakra illeszkednek.

Az érintett osztályok fejlesztésekor amint végeztem egy funkció implementálásával megírtam hozzá a teszteket. Így ezeket a teszteket regressziós tesztelésre is fel tudtam használni - ahányszor bővültek az osztályok, annyiszor lefuttattam a már meglévő teszteket az újak mellett.

Negatív tesztek A csúcsok azonosítóitól elvártam, hogy pozitívak legyenek, mivel az adatbázis által generált azonosítók is pozitív számok, ezért teszteltem, hogy negatív vagy nulla azonosítót átadva egy azonosítót argumentumként bekérő függvénynek a NodeIdNonPositiveException lép-e fel.

Teszteltem, hogy negatív csúcsszámot átadva egy függvénynek ArgumentException("Potential number of nodes cannot be negative.") lép-e fel.

Teszteltem, hogy összefüggő domináns csúcshalmazt lekérve egy többkomponensű gráfban GraphIsNotConnectedException-hez vezet-e.

További ellenőrzött kivételek:

- InvalidSetException: ha egy bekért csúcshalmaznak részhalmazának kellene lennie a gráf csúcshalmazának, mégsem az
- NodeIdNotFoundException: ha nincs ilyen azonosítójú csúcs
- NodeIdAlreadyPresentException: ha olyan csúcsot akarunk hozzáadni a gráfhoz vagy szomszédhalmazhoz, amilyet már hozzáadtunk korábban
- NoLinkFoundException: ha egy él nem található
- InvalidWeightException: ha negatív súlyt adunk meg

Érdekesebb pozitív tesztek Gráfalgoritmusok ellenőrzése egy csúcsból álló gráfon, láncon, fán és kört tartalmazó gráfon.

Integrációs tesztek

A TestServicesForDesktopApp projekt alatt egy példa kliens és szerver felállításával, és a kliensből a szerver felé indított lekérdezésekkel, kérésekkel tesztelem, hogy e két komponens helyesen működik-e együtt. A tesztelés során nem mock-oltam semmilyen objektumot, az alkalmazás által használt kontextus in-memory adatbázist használ, melyet a TestStartup osztályban töltök fel a MeetingOrganiserIntegrationTest.cs file-ban létrehozott példa adatokkal. Hogy a hálózati kapcsolat hibái miatt ne jelezzünk tévesen hibát (vagy tévesen átmenő tesztet), ezért szerverként az ASP.NET Core által biztosított TestServer-t használom, mely hálózati kapcsolatot nem igényel. A klienst ezzel a szerver objektummal a GetTestClient hívással példányosítjuk, így ennek működéséhez sem kell hálózati kapcsolat. Az alábbi teszteseteket vettem figyelembe a kommunikáció ellenőrzésekor:

Az OrganisationsController által megvalósított funkciók tesztelése az alábbi terv alapján történt:

- GetOrganisation

GIVEN	THEN
nem adminisztrátor kéri le	visszajelez a nem megfelelő jogosultságról, nem küldi el a szervezet adatait
adminisztrátor nem létező szervezetnevet ad meg	NotFound-al tér vissza
adminisztrátor létező szervezetnevet ad meg	szervezet adataival tér vissza

- PutOrganisation

GIVEN	THEN
nem adminisztrátor hívja meg	csak visszajelez a nem megfelelő jogosultságról
adminisztrátor nem létező szervezetazonosítót ad meg	NotFound-al tér vissza
adminisztrátor létező szervezetazonosítót ad meg más egyéb adatokkal, mint az eddigiek	szervezet adatai az új adatokra változnak, OK-al tér vissza

Az EventsController által megvalósított funkciók tesztelése az alábbiak alapján történt:

- GetEvents

GIVEN	THEN
nincs szervezet a megadott azonosítóval	NotFound-al tér vissza
nincs egy eseménye sem a szervezetnek	üres listával tér vissza
több eseménye is van a szervezetnek, de más szervezeteknek is van eseménye	pontosan az adott szervezet eseményeivel tér vissza

- PutEvent

GIVEN	**THEN**
nem adminisztrátor hívja meg	csak visszajelez a nem megfelelő jogosultságról
adminisztrátor nem létező esemény-azonosítót ad meg	NotFound-al tér vissza
adminisztrátor létező eseményazonosítót ad meg más egyéb adatokkal, mint az eddigiek	esemény adatai az új adatokra változnak, OK-al tér vissza

- PostEvent

GIVEN	**THEN**
nem adminisztrátor hívja meg	csak visszajelez a nem megfelelő jogosultságról
adminisztrátor egy szervezetnél már létező esemény nevével akar újat létrehozni	??
adminisztrátor egy szervezetnél még nem létező eseménynevet ad meg	létrehozza az új eseményt, OK-al tér vissza

- DeleteEvent

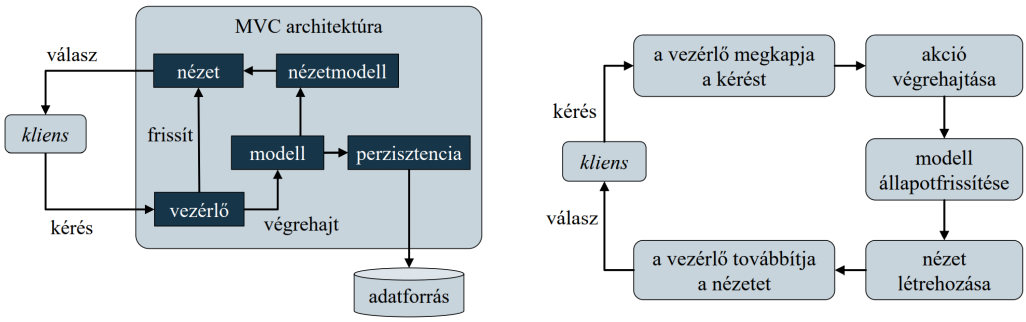
GIVEN	**THEN**
nem adminisztrátor hívja meg	csak visszajelez a nem megfelelő jogosultságról
nem létező eseményazonosítót ad meg	NotFound-al tér vissza
létező eseményazonosítót ad meg (és több esemény is köthető ugyanahhoz a szervezethez)	csak az adott eseményt törli, OK-al tér vissza

4.4. Az online felület fejlesztői dokumentációja

4.4.1. Használt architektúra

Webes környezetben a felhasználó az adott erőforrással teremt kapcsolatot, amit elsősorban az útvonala határoz meg, vagyis a felhasználó közvetlenül a vezérlést veszi igénybe, a vezérlésre az alkalmazásnak egy (új) nézettel kell válaszolnia, ami az adott erőforráshoz tartozik. Pontosan ezt nyújtja a Model View Controller (modell/-nézet/vezérlő) architektúra, éppen ezért az alkalmazás is ennek megfelelően épül fel. Az MVC architektúra komponensei a következők:

- modell: az alkalmazás logikáját írja le. Code first megközelítésnél a webalkalmazás által használt adatbázis entitásait leíró osztályok is ide tartoznak.
- nézetmodell: proxy-ként szolgál modell és nézet között - a modell által előállított eredmények (adatok) alapján a felhasználó számára megjeleníteni kívánt állapotot ír le, melyet a nézet fog megjeleníteni. A nézetállapotot írja le, és annak frissítéséért felelős. Tartalmazza a tevékenységek végrehajtásához szükséges parancsokat.
- nézet: a felület (jórészt deklaratív) definíciója, nem tartalmaz háttérkódot, csupán az adatokat kéri a nézetmodelltől
- nézetmodell: proxy-ként szolgál modell és nézet között
- perzisztencia: ez felel az adatelérésért



(a) Komponensei és azok kapcsolata (b) Végrehajtási ciklusa

4.6. ábra. MVC architektúra

4.4.2. Implementációs szabványok és használt programnyelvek

Használt elnevezési konvenciók:

- osztály: főnév; CamelCase. Pl. WeightedGraph.
- metódus: igével kezdődik; CamelCase. Pl. AddNode.
- metódus bool visszatérési értékkel: eleje Is vagy Can
- lekérdező metódusok: GetCamelCase
- kontrollerekben található form-ból bekért adatok ellenőrzésére szolgáló metódusok: VerifyCamelCase
- konstansnak szánt változók: csupa nagybetű
- változók: camelCase
- property-k: CamelCase

4.4.3. Használt programnyelvek

- A nézet implementálásához Razor és Html kódot használtam, időnként C# kódot, ritkán JavaScript-et is.
- A vezérlők és modellek megírásához a C# programnyelvet használtam.
- Adatok kezeléséhez nem volt szükség SQL kód írására, a Language-Integrated Query (LINQ) technológiájának köszönhetően C#-ot használhattam.

4.4.4. A projekt könyvtárszerkezete

- Az MVC architektúra komponensei külön könyvtárakban találhatók, ezek nevei a Views, Controllers és Models.

- az App_Start könyvtár az indítási tevékenységeket (pl. RouteConfig)
- a gyökérben található a konfiguráció (web.config) és az alkalmazásszintű vezérlés (Global.asax)
- MVC alkalmazáshoz szükséges a NuGet csomagkezelő

4.4.5. Érdekesebb előforduló hibák

NULL referencia

Azok az esetek voltak nagyon tanulságosak, amikor egy kontroller egyik akciója egy adatokkal feltöltött nézetmodellt adott át egy másiknak. Alapvetően nincs két akció közti állapotmegőrzés, ez nem egy támogatott módszer, ezért is találkoztam annyiszor NULL referencia hibával, ugyanis átadáskor a nézetmodell tartalma törlődött. Ez megkerülhető a ViewBag vagy a ViewData változók használatával. Mivel ezt nem találtam túl vonzó lehetőségnek, hiszen nem minden akcióhoz lesz szükségem az adott változóra, mégis a közös ViewBag-ben van, illetve az enkapszuláció elvét is sérti, rontja a kód követhetőségét. Végül sikerült megoldanom, hogy átadáskor megőrződjenek az adatok. A megoldást a nézet html kódjában felvett rejtett beviteli mezők jelentették. Minden nézetmodellbeli elemet kötni kell egy rejtett beviteli mezőhöz, így megmarad a tartalma. Ezeket a Html.HiddenFor() paranccsal tudtam létrehozni. Ez sem egy szép megoldás sajnos, viszont a logikai egységeket így nem kellett bontani.

Remote validátorok

Nem minden ellenőrzés volt megvalósítható az olyan beépített ellenőrzőkkel, mint a Required, MaxLength, stb. Ezért kiegészítésül a kontrollerekben, ahol szükséges volt, létrehoztam olyan validáló függvényeket, melyek a szerver oldalon hajtják végre az általam definiált ellenőrzéseket. Amennyiben az ellenőrzött tartalom megfelel egy Json üzenetbe csomagolt igaz értéket kapunk vissza. Az ellenőrizni kívánt tulajdonság (property) fölé a Required feltételhez hasonlóan kapcsos zárójelben kell feltüntetni az ellenőrző függvény elérhetőségét (a megfelelő kontroller és akció nevének megadásával). Az első próbálkozásaim során azonban a program végrehajtás be sem lépett az ellenőrző függvénybe. Később kiderült, hogy az ellenőrző függ-

vény paraméterének neve és az ellenőrzött tulajdonság neve teljesen meg kell, hogy egyezzenek, még kis-nagy betűs eltérés sem lehet köztük.

Datepicker

Az alapértelmezett Datepicker-el nem lehet kezdőértéket megjeleníteni, a Value tulajdonságát állítva nem változik semmi. Ez egy a Microsoft által is elismert hiba, elvileg hamarosan javítják. Illetve amikor helytelen adatokat szeretnénk menteni a dátumválasztó addigi értéke törlődik, még akkor is, ha maga a dátum jól volt beállítva. Megoldást nem találtam a problémára, a mások által készült dátumválasztók esetében egyáltalán nem volt lehetséges az adatkötés, és nekem sem sikerült írnom megfelelő vezérlőt.

4.4.6. Tesztelés

Egység tesztek

A tesztek a solution alatt egy külön projektben kerültek elhelyezésre TestWebAppForMembers néven. Az egységtesztek döntő többsége a decision table technikára épít, tehát a bemeneti változók - és az egyéb, az eredményt befolyásoló változók - minden elképzelhető állapota (helyes, helytelen, nem létező stb.) mellett leírtam, hogy milyen eredményt várunk, ha a többi ilyen változó egy adott állapotban van. Ezek alapján a leírások alapján készültek a tesztek. Továbbá a VerifyStartDate igazságértéket visszaadó függvénynél a Boundary Value Analysis technikáját használtam. Ez akkor használható, amikor egy változó által felvehető értékek intervallumokra bontható aszerint, hogy milyen viselkedést várunk a változó az adott intervallumba esésekor. Ilyenkor az intervallumok végein, illetve minden intervallumvég egy egységgel kisebb és egy egységgel nagyobb értékeken is végzünk teszteket, valamint intervallumonként még egy középső esetet is megvizsgálunk. Az adatbázist a ServiceForMembers osztályon keresztül érjük el, mely adat lekérdező-, létrehozó-, módosító- és törlő metódusokkal rendelkezik. Két függősége van, a kontextus és a felhasználók kezeléséért felelős objektum. Ezek egységtesztek során mockolásra kerültek. A felhasználókezelő mockolásakor a aspnet/Identity/MockHelpers.cs fájlban

használt módszert emeltem át. A ServiceForMembers fontosabb illetve bonyolultabb ellenőrzendő függvényei és az implementálandó tesztesetek az alábbiak:

- SaveChanges

GIVEN	**THEN**
context.SaveChanges() hibával tér vissza	false-al tér vissza
context.SaveChanges() rendben lefut	SaveChanges meghívódik, true-val tér vissza

- StrictestDeadlineAt

GIVEN	**THEN**
nincs esemény adott szervezetnél	null-t ad vissza
nem létezik a szervezet	null-t ad vissza
több különböző határidejű esemény van az adott szervezetnél	közülük a legkorábbi dátumot adja vissza

- GetAvailableEventsOf

GIVEN	**THEN**
nem létezik a felhasználó	üres listát ad vissza
nincsenek események	üres listát ad vissza
több eseménye is van, de van olyan esemény is, amely számára nem elérhető	minden elérhető eseményt visszaad, és mást nem

- CreateJob

GIVEN	**THEN**
nem létező szervezet	hamissal tér vissza
a szervezet létezik, de már van ilyen munkakör hozzárendelve	igazzal tér vissza, nincs mellékhatása
a szervezet létezik, és még nincs ilyen munkakör hozzárendelve	igazzal tér vissza és létrejön egy új munka a megadott címmel és a megadott szervezethez rendelve. SaveChanges metódus meghívódik.

- EditEventForm

GIVEN	**THEN**
nem létező tag	hamissal tér vissza
nem létező esemény	hamissal tér vissza
a tag számára nem elérhető az esemény	hamissal tér vissza
a tag számára elérhető az esemény	igazzal tér vissza, az űrlap értékeit a kapott értékekre állítja

- DeleteMembership

GIVEN	**THEN**
nem létező felhasználó	hamissal tér vissza, nem törlődik elem
nem létező szervezet	hamissal tér vissza, nem változik az adatbázis
a felhasználó nem tagja a szervezetnek	igazzal tér vissza, nem változik az adatbázis
a felhasználó tagja a szervezetnek	igazzal tér vissza, csak a megfelelő tagság törlődik

A fenti tesztesetek implementációja a TestWebAppForMembers projekten belül a TestServiceForMembers.cs fájlban találhatók. Fontosnak tartom a kontrollerekben implementált, a nézetmodellekbe felvitt adatokat ellenőrző függvényeket is tesztelni. Ezt a TestControllers.cs file-ban tettem meg. Ezek a következők:

- EventsController
  - VerifyStartDate - Boundary Value Analysis technikával. Ez hasonlóan meggondolható a VerifyEndDate esetén is.

GIVEN az időpontok sorrendje a következő	**THEN**
kezdő időpont,(egy egységnyi különbséggel) esemény kezdete, esemény vége, befejezési idő (utóbbi null is lehet)	hamissal tér vissza
esemény kezdete, befejezési idő, (egy egységnyi különbséggel) kezdő időpont, esemény vége	hamissal tér vissza
esemény kezdete, esemény vége, (egy egységnyi különbséggel) kezdő időpont, befejezési idő (null is lehet)	hamissal tér vissza
a négy időpont megegyezik (vagy a befejezési idő null)	igazzal tér vissza
esemény kezdete megegyezik a kezdő idővel, a többi ezeket követi (bármilyen sorrendben, befejezési idő null is lehet)	igazzal tér vissza
esemény kezdete, kezdő idő megegyezik az esemény végével, befejezési idő szintén megegyezik vele (vagy null)	igazzal tér vissza
esemény kezdete, kezdő idő megegyezik befejezési idővel, esemény vége	igazzal tér vissza
esemény kezdete, kezdő idő, befejezési idővel, esemény vége	igazzal tér vissza
esemény kezdete, kezdő idő, esemény vége, befejezési idővel (null is lehet)	igazzal tér vissza

- JoinOrganisationController
  - VerifyProject

GIVEN	**THEN**
már szerepel a szervezet projektjei között az adott projektnév	hamissal tér vissza
már szerepel az újonnan felvitt projektek között az adott projektnév	hamissal tér vissza
egyik listában sem szerepel a projekt	igazzal tér vissza

5. fejezet

Összegzés

A szakdolgozat tehát a gráfok domináns csúcshalmazainak megtalálására szolgáló approximációs algoritmusok mellett a webes alkalmazásokról és eseményvezérelt alkalmazásokról szól.

A szakdolgozat írása során új algoritmusokkal ismerkedtem meg, közülük többet implementáltam is. Régebbről ismert gráfalgoritmushoz is írtam függvényt. Megismerkedtem a Webes alkalmazások fejlesztése című tárgy keretében megismerhető technológiákkal, módszerekkel, szoftverarchitektúrákkal. Jobban megvizsgáltam a C# nyújtotta nyelvi lehetőségeket, adatszerkezeteket és azok hatékonyságát. Okulhattam hibáimból. Például nagy tanulság volt számomra, hogy mennyit nyerhet egy fejlesztő azzal, ha azonnal teszteli az új funkciókat, milyen fontos a regressziós tesztelés. Átismételhettem a WPF alkalmazások kapcsán szerzett ismereteimet, újra elmélyülhettem az eseményvezérelt alkalmazások logikájában.

5.0.1. Továbbfejlesztési lehetőségek

A weblap az alapértelmezett design-al rendelkezik, ezen mindenképp lehetne dolgozni. Új funkciói lehetnének még a nyílt csoportok közti böngészés, több eseménnyel kapcsolatos preferencia bekérése, eseményeket tartalmazó naptár készítése, más személyek saját csoportunkba való hívása, stb. Továbbá, mivel kísérleti jelleggel készült az alkalmazás ugyanarra a problémára több féle megoldást is kipróbáltam, emiatt egy refaktorálásra, egységesítésre is szükség volna a kód átláthatóságát növelendő.

Hasznos lenne, ha az adminisztrátor is létrehozhatna súlyozási szempontokat az asztali alkalmazással, illetve ha komplexebb képletet is használhatna súlyozásra.

Meglévő adatbázisból való adatfeltöltéssel és a távoli adatbázis mentésével is bővülhetne az alkalmazás. Továbbá érdemes lenne párhuzamosíthatóság szempontjából is megvizsgálni a domináns halmaz készítését.

Irodalomjegyzék

[1] Sepehr Assadi és tsai. "Fully Dynamic Maximal Independent Set with Sublinear in n Update Time". *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (2019), 1919–1936. DOI: 10.1137/1.9781611975482.116. URL: http://dx.doi.org/10.1137/1.9781611975482.116.

[2] Madhav V. Marathe és tsai. *Geometry based heuristics for unit disk graphs.* 1994. arXiv: math/9409226 [math.CO].

[3] Francisco Vazquez-Araujo és tsai. "Calculation of the Connected Dominating Set Considering Vertex Importance Metrics". *Entropy* 20 (2018. jan.). DOI: 10.3390/e20020087.

[4] Cserép Máté. *Webes alkalmazások fejlesztése - Webfejlesztés MVC architektúrában (ASP.NET Core).* URL: https://mcserep.web.elte.hu/data/education/2019-2020-2_WAF/elte_waf_ea02.pdf.

[5] Rick Strahl. *Publishing and Running ASP.NET Core Applications with IIS.* URL: https://weblog.west-wind.com/posts/2016/jun/06/publishing-and-running-aspnet-core-applications-with-iis.

[6] *Visual Studio 2017 Product Family System Requirements.* URL: https://docs.microsoft.com/en-us/visualstudio/productinfo/vs2017-system-requirements-vs.

[7] *System Requirements.* URL: https://help.syncfusion.com/aspnet-core/installation-and-upgrade/system-requirements.

Ábrák jegyzéke

Táblázatok jegyzéke

Forráskódjegyzék