|  |  |  |
| --- | --- | --- |
|  | **Eötvös Loránd Tudományegyetem**  Informatikai Kar  Programozási Nyelvek és Fordítóprogramok Tanszék |  |

**P4 programok gráf alapú statikus elemzése**

|  |  |
| --- | --- |
| *Témavezető:*  Tóth Gabriella  Doktorandusz | *Szerző:*  Tábi Zoltán  Programtervező Informatikus, BSc |

Budapest, 2020

Tartalomjegyzék

[1. Bevezetés 5](#_Toc41602457)

[1.1. Alapfogalmak 5](#_Toc41602458)

[1.1.1. P4 5](#_Toc41602459)

[1.1.2. Vezérlésfolyamgráf 6](#_Toc41602460)

[1.1.3. Adatfolyamgráf 6](#_Toc41602461)

[2. Felhasználói dokumentáció 7](#_Toc41602462)

[2.1. Program célja 7](#_Toc41602463)

[2.2. Használat 7](#_Toc41602464)

[2.2.1. Feltételek 7](#_Toc41602465)

[2.2.2. Futtatás 8](#_Toc41602466)

[2.2.3. Vizsgálható kód 8](#_Toc41602467)

[2.3. Rendszer ismertetése 9](#_Toc41602468)

[2.3.1. Gráfok ismertetése 10](#_Toc41602469)

[2.3.2. Kezdő oldal 13](#_Toc41602470)

[2.3.3. Gráf megjelenítő oldal 15](#_Toc41602471)

[2.3.4. Elemzéseket végző oldal 22](#_Toc41602472)

[2.3.5. Fájl oldal 27](#_Toc41602473)

[2.3.6. Súgó oldal 28](#_Toc41602474)

[2.3.7. Kényelmi funkciók (TODO nem volt jobb ötletem) 28](#_Toc41602475)

[3. Fejlesztői dokumentáció 30](#_Toc41602476)

[3.1.1. Fejlesztői környezet 30](#_Toc41602477)

[3.1.2. Probléma leírása 33](#_Toc41602478)

[3.1.3. Program leírása 33](#_Toc41602479)

[4. Tesztelés 34](#_Toc41602480)

[5. Összefoglalás 35](#_Toc41602481)

[6. További fejlesztések 36](#_Toc41602482)

[6.1. P4 fordítóprogram beépítése 36](#_Toc41602483)

[6.1.1. Egész projekt Ubuntun-on történő futtatása 36](#_Toc41602484)

[6.1.2. Szerver a P4 fordítóprogramhoz 36](#_Toc41602485)

[6.2. Vizsgált résznyelv kiterjesztése 36](#_Toc41602486)

[6.3. Felhasználó azonosítása 37](#_Toc41602487)

[6.4. Gráf elemzés összesített verziója (TODO lehet kikerül) 37](#_Toc41602488)

[6.5. Felület optimalizálás 38](#_Toc41602489)

[6.5.1. Sötét téma 38](#_Toc41602490)

[6.5.2. Mobil használat 38](#_Toc41602491)

# Bevezetés

Szakdolgozatom célja egy olyan P4 programozási nyelvet elemző program elkészítése, ahol a felhasználói élmény ugyanannyira fontos, mint a háttérben lefutó számítások és elemzések precizitása.

Programom háttérfolyamatait kezelő részét C# programozási nyelven írtam, amivel az egyetemen találkoztam és hamar megszerettem, ezért mindenféleképpen el akartam mélyíteni benne a tudásomat.

A programot megjelenítő rész sokáig kérdéses volt számomra, hogy mi legyen, mivel szakdolgozatomban nem ezen van a fő hangsúly. Az egyetemen tanult felhasználó felületek, mint például a Xaml vagy a Razor Pages nem tartoznak kedvenceim közé, ezért valami új dologgal szerettem volna megismerkedni, így esett a választásom az Angularra, amivel a felület és a háttérfolyamatok teljesen elkülönülnek egymástól. A fő okai a döntésemnek, hogy az Angular keretrendszeren belül is lehet objektumorientált szemlélettel programozni, valamint, hogy az interaktív felhasználói felületek elkészítésére az egyik legjobb választás napjainkban.

## Alapfogalmak

Szakdolgozatom témájából eredően a probléma megoldásához mély P4 tudás elsajátítása, valamint a vezérlés- és adatfolyamgráffal való megismerkedés szükséges volt. Emiatt elengedhetetlennek tartom, hogy ezekre a fogalmakra külön is kitérjek.

### P4

A P4 egy olyan új programozási nyelv melynek segítségével a hálózaton közlekedő csomagok feldolgozása és tovább küldése a feladata. Említenék pár szót a program felépítéséről:

* Fejléc – Csomagformátum leírására, valamint a csomagban található mezők leírására használják. Neve tetszőleges lehet, de programon belül egyedinek kell lennie. Szakdolgozatom fő témája, az ilyen szerkezetekkel kapcsolatos változtatások, felhasználások és lehetséges hibák kiszűrése.
* Parser – Kicsomagolást végzi, ahol meghatározza, hogy mely fejlécek lesznek kezdetben inicializált állapotban. A program ezen részét nem vizsgálom.
* Ingress control – A program fő blokkja, amit szakdolgozatomban elemzek. Részei:
  + Kontrollfüggvény törzse – Az Ingress control lefutását végző függvény, ez alapján megy végbe a folyamat.
  + Table – Olyan szerkezetek, melyek tartalmaznak kulcsokat és egy akciókból álló listát. Egy P4 program lefutásakor egy külső vezérlőtől kapjuk meg a tábla tartalmát, mellyel összehasonlítjuk saját táblánkat és így választjuk ki a megfelelő akciót. Szakdolgozatomban én ezt elágazásként kezeltem, ahol minden akció ugyanakkora eséllyel fut le.
  + Akció – Lényegében függvények, ahol egy kódrészlet lefut. Változtathatja a fejlécek inicializáltságát, valamint mezőket deklarálhat vagy adhat nekik új értéket.
* Deparser – A program fő blokkjának lefutása végeztével összecsomagolja a csomagot. A program ezen részét nem vizsgálom.

### Vezérlésfolyamgráf

TODO megfelelő hivatkozások

A számítástechnikában a vezérlésfolyamgráf egy olyan gráf, amely bemutatja a program lefutását és a kódrészletek kapcsolatát. Ez a fajta gráf nélkülözhetetlen számos fordító optimalizálásához és statikus elemző eszközökhöz. A gráfban szereplő csúcsok mindegyike a programnak egy blokkja, melyek egymást követik a program lefutása során. A legtöbb esetben a gráf reprezentációja tartalmaz kettő fő csúcsot. Az egyik a belépési blokk, ahonnan a folyamat elindul, a másik pedig a kilépési blokk, ahova az összes kódrészlet befut.

### Adatfolyamgráf

Az adatfolyamgráf nagyon hasonló az előbb említett vezérlésfolyamgráfhoz, de ebben az ábrázolási módban nem a blokkok közötti kapcsolatot vizsgáljuk, hanem a blokkban történő utasításokat, vagyis értékadások, aritmetikai műveletek, változókra hatással lévő függvények vannak a középpontban. Ez a gráf is ugyanúgy használatos elemzésekhez és optimalizáláshoz.

# Felhasználói dokumentáció

## Program célja

A feladatom egy olyan felhasználóbarát és könnyen kezelhető weboldal elkészítése volt, amely P4-es programok gráf alapú elemzését végzi. A programkód bevitelét meg lehet tenni szövegbeviteli mező segítségével vagy fájlfeltöltés útján. A gráfokkal interaktív események hajthatóak végre. Többek között gráf bejárás szimulálása, csúcshoz és csúcsból vezető út kirajzolása és gráfok egymás közötti kapcsolatának megjelenítése. Ezek után meg lehet adni a kezdeti- és végértékeket, majd több megadása után csomagot összeállítani és végezetül az elemzett gráfok és eredmények megtekintését.

A weboldal főleg a P4 programnyelvvel foglalkozó közösség számára szól, de bárki számára használható az oldalon megtalálható fájlok segítségével, valamint a dokumentáció vagy a webhelyen található Súgó oldal átolvasásával.

## Használat

### Feltételek

Egy weboldal lévén a felhasználásnak nincsenek rendszert érintő követelményei. A weboldal használható mobileszközről, tabletről, valamint számítógépről is, ezek közül ajánlott a számítógép használata nagyobb képernyő és átláthatóság érdekében. Valamint az alapvető webböngészés képességével is rendelkeznie kell a felhasználónak.

Támogatott böngészők:

* Google Chrome
* Mozilla Firefox
* Opera
* Microsoft Edge

### Futtatás

Az alábbi címen elérhető a weboldal: <https://p4analyst.azurewebsites.net/> , így telepítésre nincsen szükség.

### Vizsgálható kód

A weboldal P4 programok elemzését végzi, így az elfogadott fájlok kiterjesztése a .p4, de a .txt kiterjesztésű fájlok feltöltését is megengedi az oldal.

A megadott fájlnak vagy szövegnek helyes P4 kódnak kell lennie, mivel ennek ellenőrzését nem teszi meg a rendszer. Ez azt jelenti, hogy a feltöltött fájl helyességét ajánlott a P4 fordítóprogramjával ellenőrizi. Amennyiben nem rendelkezik ezzel a fordítóval és telepíteni nem akarja, akkor a fájlok oldalon találhat olyan helyes P4 kódokat, mellyel megtekinthető a weboldal működése.

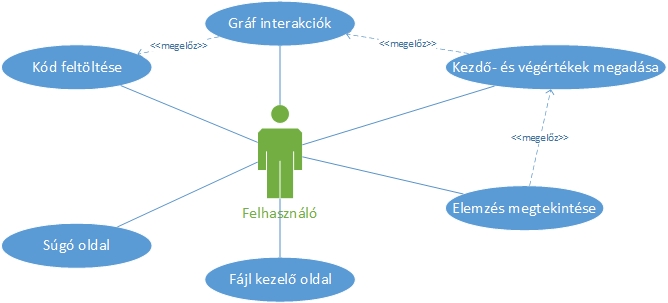
Azonban a program nem rendelkezik a P4 kódok teljes funkcionalitásainak ismeretével, ezért megkötésekkel rendelkező P4 programok ellenőrzésére és vizsgálatára van mód a weboldalon. A hibátlan működés miatt ezeket be kell tartani.

Fontos kiemelni, hogy a program fő funkciója a fejlécek vizsgálata, így az olyan műveleteket, melyek nem ilyen egységgel kapcsolatosak nem tudom kezelni.

Kivételek:

* Akción belüli elágazás nem megengedett. TODO kódban lekezelni
* Else if elágazás nem megengedett. TODO kódban lekezelni
* Szakdolgozatomban a P416-os verzióit elemzem. Más verzió használata nem támogatott.

## Rendszer ismertetése



1. ábra – Általános felhasználói eset diagram

Az *általános felhasználói eset diagram* ábrán láthatóak a weboldal fő funkciói. Ez csak egy összefoglaló ábra, melynek funkcióit bővebben kifejtem később, amikor is a weboldal összes oldalát és azokban rejlő lehetőségeket részletesen leírom.

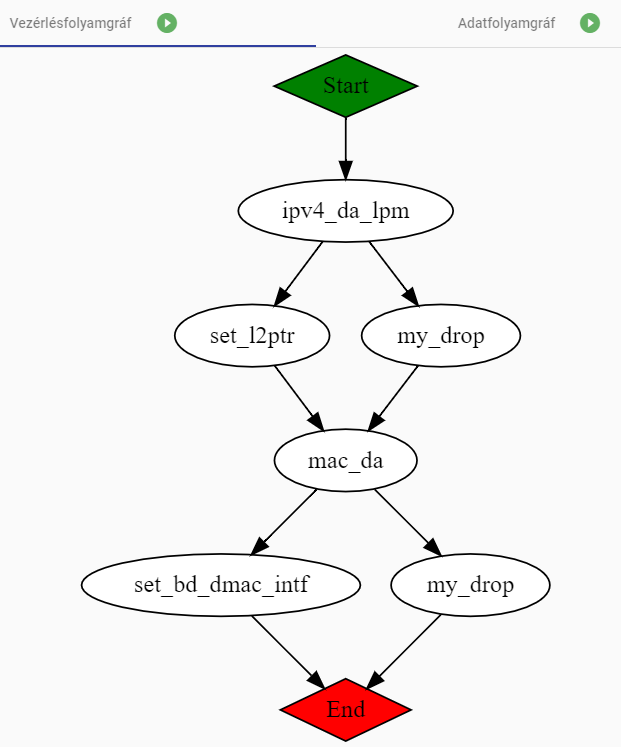
Az ábrán jól látható, hogy a weboldal egy része lineáris lefutással bír, vagyis egymásra épülő, ki nem kerülhető folyamatok követik egymást.

Funkciók rövid ismertetése:

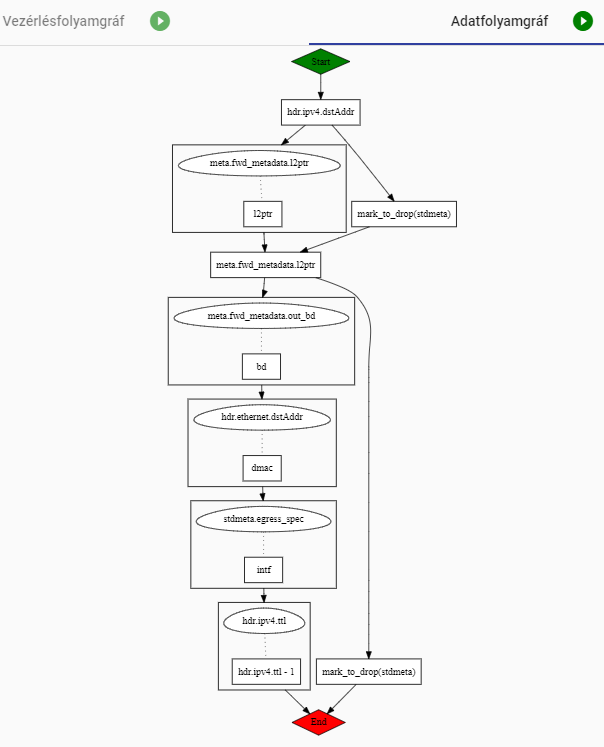
* Kód feltöltés – A felhasználónak itt lehetősége van egy P4 kód megadására fájlból vagy szöveges bevitellel.
* Gráf interakciók – A felhasználónak itt lehetősége van a kódból generált gráfokkal interakciókat végre hajtani. Ezek közé sorolható a gráf bejárás szimulálása, az adott csúcsba vezető út és abból kivezető utak színezése, valamint a vezérlésfolyamgráf és az adatfolyamgráf közötti kapcsolatok megismerése.
* Kezdő- és végértékek megadása - A felhasználónak itt lehetősége van a kódban lévő fejlécekről eldönteni, hogy kezdetben mely fejléceit szeretné inicializáltra állítani, valamint az elemzett rész után mely fejléceket szeretné felhasználni. Ez a funkció a *Parser*-t és *Deparser*-t váltja ki, mivel azokat nem dolgozom fel.
* Elemzés megtekintése - A felhasználónak itt lehetősége van az elemzések megtekintésére, ahol is megjelenik az összes megadott kezdő- és végértékhez generált gráf, valamint diagramok, melyeket a rendszer számolt ki.
* Fájl kezelő oldal – A felhasználónak itt lehetősége van fájlokat keresni, azok tartalmát vágólapra helyezni, vagy a fájllal azonnal megkezdeni az elemzési folyamatot, ami a gráf interakciókkal kezdődik.
* Súgó oldal - A felhasználónak itt lehetősége van megismerkedni a weboldal céljával, valamint alapvető használatával. Elolvasásával könnyen elsajátítható a tudás, amely az oldal használásához szükséges.

### Gráfok ismertetése

Szakdolgozatomban sok munkát fektettem a gráfok megismerésére és a megfelelő ábrázolásuk kidolgozásukba. Pár mondatban összefoglalnám, hogy a gráfok, valamint a csúcsok között milyen kapcsolatok állnak fel és ezeket milyen módon ábrázolom.



2. ábra - Vezérlésfolyamgráf



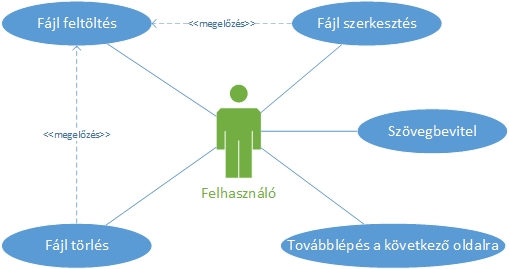
3. ábra - Adatfolyamgráf

Ahogy a *Vezérlésfolyamgráf* és az *Adatfolyamgráf* ábráján is látszik, hogy mindkét gráfban szerepel egy kezdő, valamint egy vég csúcs.

A vezérlésfolyamgráfon a többi csúcs között nincsen különbség. Minden csúcs egyenlő értékkel bír és a program lefutásával megegyező módon követik egymást. Az egeret egy csúcs fölé helyezve megjelenik a típusa, melyek lehetnek *if*, *table*, *action.* Az élek közül kivételt jelentenek az *if* csúcsból kiinduló élek, itt az igaz ág zöld színű éllel van jelölve, míg a hamis ág piros színnel.

Az adatfolyamgráfon láthatunk szögletes, valamint ovális alakú csúcsokat ezek jelentése pedig az, hogy a szögletes csúcsban lévő adat olvasásra kerül, míg az ovális alakú csúcsban lévő adat írásra kerül. Az itt megjelenő csúcsok mindegyike egy adott vezérlésfolyamgráf blokkja vagy változó felhasználása. Itt is *if* csúcsból kiinduló élek egyike zöld, másik piros színnel vannak jelölve, ha az *if* feltétele több csúcsra bontható, akkor ezek mindegyike szögletes alakú csúcsba kerül és szaggatott vonallal vannak összekötve, ezzel jelezve összetartozásukat. Az *Adatfolyamgráf* ábrán látható, hogy vannak szögletes és ovális alakú csúcs párok ezek jelölése is szaggatott vonallal történik meg, ezek értelmezése, hogy az itt olvasott csúcsnak jelölt kódrészlet értéke kerül bele az írt csúcsban lévő változóba.

### Kezdő oldal

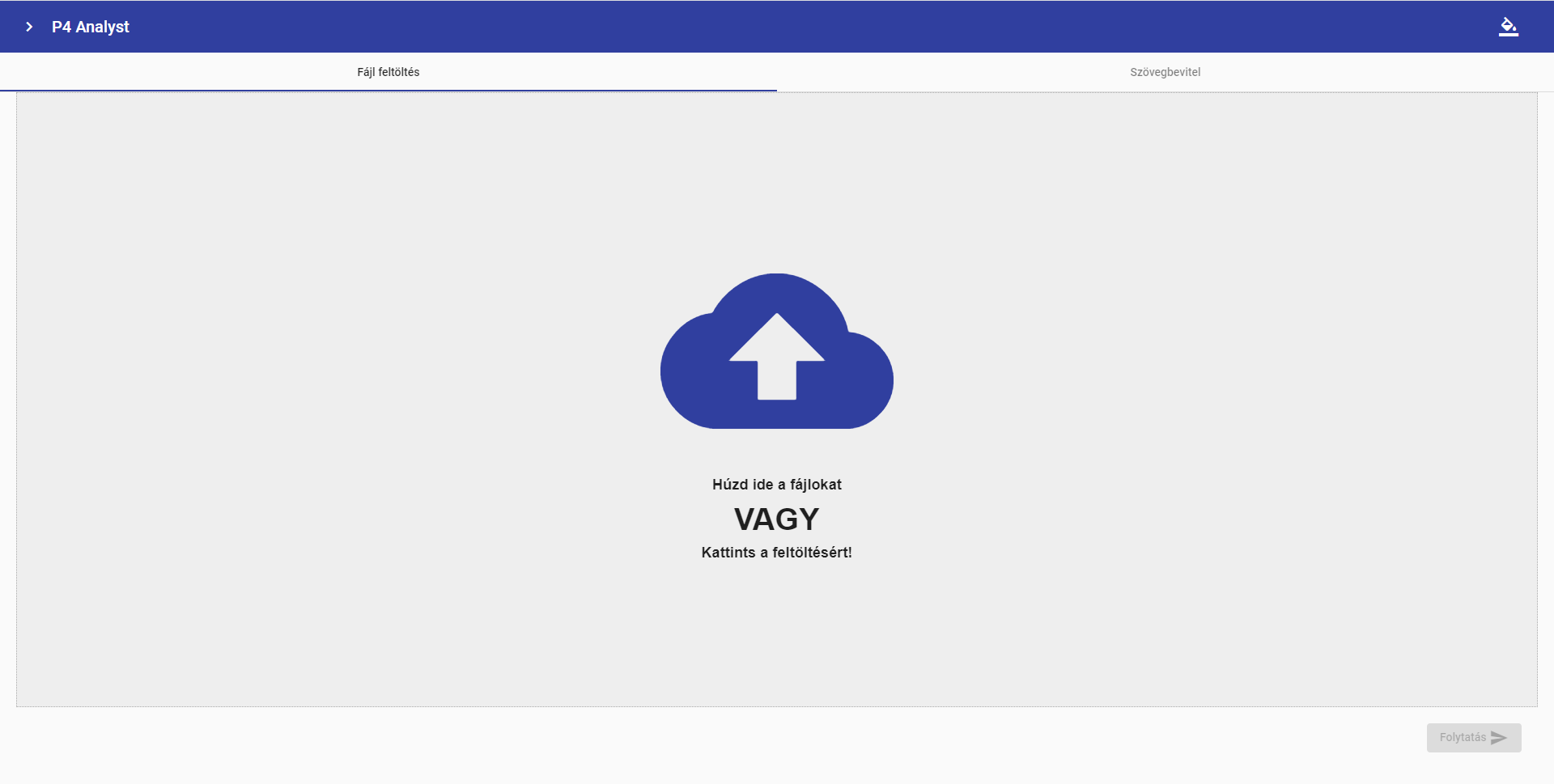


4. ábra - Kezdő oldal felhasználói eset diagram

A *Kezdő oldal felhasználói eset diagram* ábráján látható, hogy a felhasználónak milyen funkciók és lehetőségek állnak rendelkezésére, amikor a kezdőoldalon van.

Ezen az oldalon adhatja meg elemzendő kódját, itt egy tab-os elrendezésben választhatja ki, hogy milyen módon adja át kódját. Az opciók kizárják egymást, szóval, ha feltöltött fájlt, addig nem léphet át a szövegbeviteli részre, amíg azt a fájlt ki nem törölte és fordítva is igaz, tehát, amennyiben a beviteli mező nem üres úgy nem léphet vissza a fájl feltöltő részlegre. A következő oldalra lépés, akkor érhető el, ha az egyik eset által adott meg szöveget. A szövegbeviteli mező egy egyszerű beviteli mező, ahova szöveget írhat be vagy illeszthet be a vágólapról.

#### Fájl feltöltés



5. ábra - Fájl feltöltő

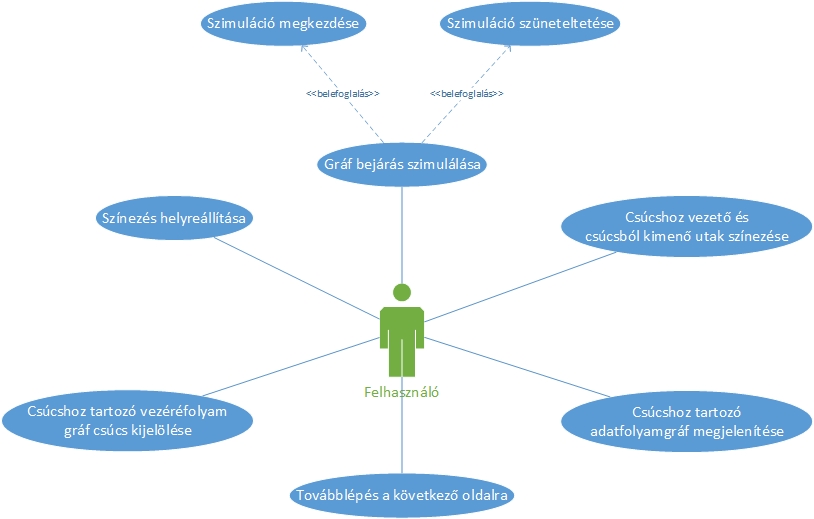
A felhasználó ebben az esetben két különböző úton adhatja meg a kiválasztandó fájlt. Az első eset, hogy bal klikkelés után a fájlkezelő ablakból kiválasztja a feltöltendő fájlt. A második eset, hogy úgynevezett *drag and drop* módszerrel, vagyis a fájl odahúzásával és elengedésével adja meg a kiválasztott fájlt. Minden esetben ellenőrizzük, hogy a fájl megfelelő, .p4 vagy .txt kiterjesztésű-e, más kiterjesztéssel rendelkező fájl feltöltésére nincs lehetőség, ezt figyelmeztető üzenet formájában jelezzük a felhasználónak. A második módszer esetén a feltöltött fájlok mennyiségére is ellenőrzést hajtunk végre, mivel több fájl feltöltése nem megengedett, ekkor is megtagadjuk a fájl(ok) feltöltését és figyelmeztető üzenet formájában ezt jelezzük.

Az elfogadott fájl megadása után a képernyőn megjelenik a fájl neve, valamint mellette kettő akció gomb is. Az első esetben a fájl szerkesztését végezheti, ekkor egy felugró ablakban jelenik meg a fájl tartalma, ahol szerkeszthető. Az ablak bezárása négy módon történhet meg:

* *Mentés* gombra kattintás – a szerkesztést jóváhagyja, ezt egy felugró üzenettel erősítjük meg
* *Mégsem* gombra kattintás
* Bezáró gombra kattintás
* Az ablak területén kívülre kattintás

A törlés gombra való kattintás esetén a fájl feltöltése semmisnek tekintendő és lehetőség nyílik másik fájlt feltölteni, vagy a szövegbeviteli mezőre átlépni.

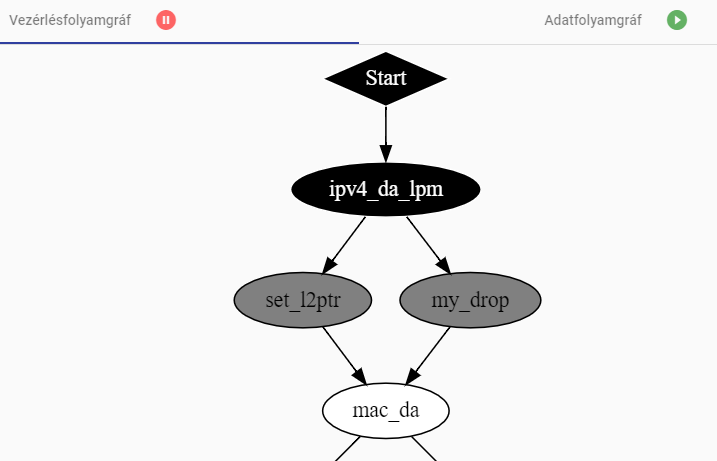
### Gráf megjelenítő oldal



6. ábra - Gráf megjelenítő oldal felhasználói eset diagram

A *Gráf megjelenítő oldal felhasználói eset diagram* ábráján látható, hogy a felhasználónak milyen funkciók és lehetőségek állnak rendelkezésére, amikor a gráf megjelenítő oldalon van. Az oldal működésének előfeltétele, hogy a kód feltöltése már megtörtént, ezért ha az nem történt volna meg, akkor hibaüzenettel jelezzük, hogy először fájlt fel kell tölteni. Amennyiben már ez megtörtént, akkor az oldalon a tab-os elrendezésben alapértelmezetten kettő oldal található meg, melyek az oldalon állandóan jelen vannak. Az első a vezérlésfolyamgráfot megjelenítő tab, a második pedig az adatfolyamgráfot reprezentáló oldal. A gráfok betöltésekor egy információs üzenettel jelezzük, hogy a gráf betöltését várja meg, mivel csak így tudjuk garantálni a hibátlan és maximális felhasználói élményt. A gráf betöltődése addig tart, amíg a képernyőn megjelenő forgásban lévő ikon a képernyőn marad, utána a gráf kirajzolása következik, ahol a gráf csúcsai és azokat összekötő élek beúsznak az oldalra. Amikor a gráf kirajzolása is megtörtént onnantól érhetőek el az oldal interakciói.

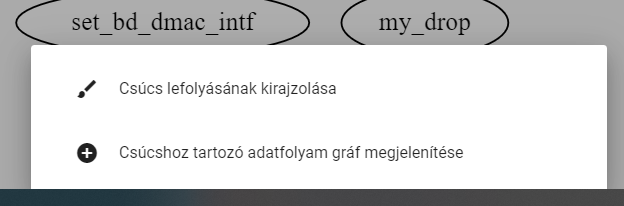
#### Gráf bejárás szimulálása



7. ábra - Gráf bejárás

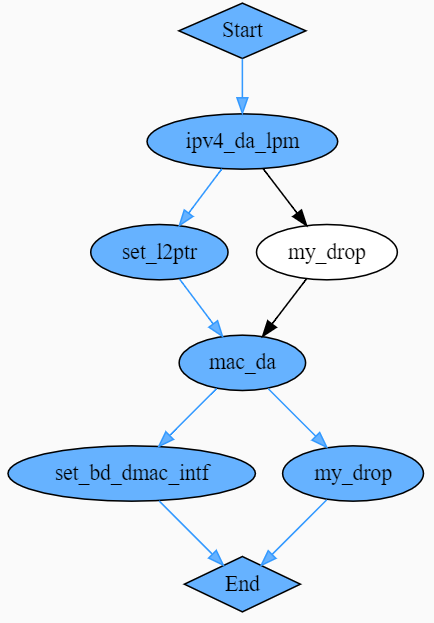
A két fő gráf bejárásának szimulálásra van lehetőség. A funkció elindításához a tab-ban, név mellett látható zöld lejátszási ikon megnyomását kell megtenni. Lenyomásakor a lejátszási ikonból szüneteltetési ikon lesz, valamint a zöld színből piros színbe vált át a gomb, amit a *Gráf bejárás* ábrán láthatunk is. A bejárás prezentálásához a szélességi bejárást választottam, amit egyetemi éveim alatt sajátítottam el. A bejárás lényege, hogy szintenként halad és csak akkor ugrik a következő szintre, ha az adott szinten már mindent csúcsot elért. A bejárás szüneteltethető a szüneteltetési ikon lenyomásával vagy a tab-ról való elkattintással. A folyamat a megszakított állapottól folytatható. A funkció csak, akkor aktív, ha a felhasználó azon a gráfon van, amire alkalmazni szeretné.

#### Csúcs interakciók



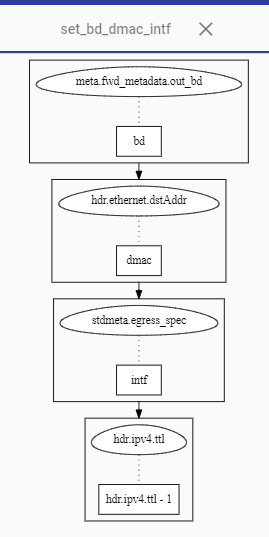
8. ábra - Interakció kiválasztása

Csúcs interakciók is csak a két fő gráfon hajthatóak végre. Egy csúcsra kattintás esetén az *Interakció kiválasztása* ábrán látható felugró felületen van lehetőség az opciók közül választani.



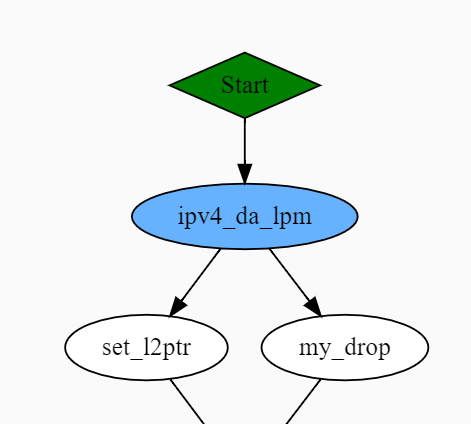
9. ábra - Csúcs lefolyásának színezése

Mindkét gráf esetében, minden csúcsnál van lehetőség a csúcshoz vezető és abból kimenő utak színezésére. Ezt a *Csúcs lefolyásának színezése* ábra segítségével reprezentálom, ahol a *set\_12ptr* csúcson került kiválasztásra ez a funkció.



10. ábra - Részgráf megjelenítés

A vezérlésfolyamgráf esetében, ha olyan csúcsra kattint, amihez tartozik adatfolyam részgráf, akkor az opciók között szerepelni fog a *Csúcshoz tartozó adatfolyam gráf megjelenítése* is. Az opció kiválasztása esetén az adatfolyam részgráf egy új tab-on fog megnyílni, amelyre egyből odaugrik az oldal, ha ez a részgráf már meg volt nyitva, akkor az oldal egyszerűen csak arra a tab-ra ugrik. Az így megnyílt gráfon nem használható egyetlen interakció sem, a tab-ot az *X* gomb segítségével be lehet zárni. A *Részgráf megjelenítés* ábrán látható, hogy itt egy részgráf jelenik meg, valamint a bezárás funkcióval rendelkező ikon is.

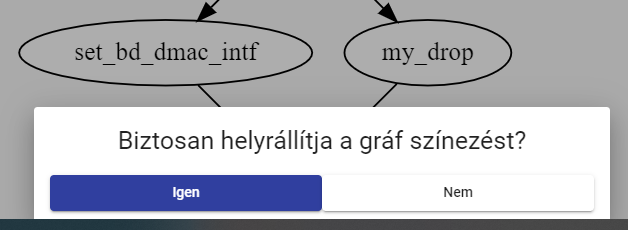


11. ábra - Csúcs kiválasztás

Az adatfolyamgráf esetében, ha olyan csúcsra kattint, ami egy vezérlésfolyamgráf csúcsból alkotható adatfolyam részgráfnak egy csúcsa, akkor az opciók között szerepelni fog a *Csúcshoz tartozó vezérlésfolyam gráf csúcs kijelölése* is. Az opció kiválasztása esetén az oldal átugrik a vezérlésfolyamgráfhoz és az előbb említett csúcsot kék színnel jelzi. A *Csúcs kiválasztás* ábrán látható, hogy az adatfolyamgráf csúcshoz tartozó vezérlésfolyamgráf csúcs milyen módon kerül kiemelésre.

#### Egyéb funkciók

Az oldalon további két funkció elérhető, az egyik ezek közül a gráf színezés helyreállítása. Ez a funkció is csak a két fő gráfra, vagyis a vezérlésfolyamgráfon és az adatfolyamgráfon alkalmazható.

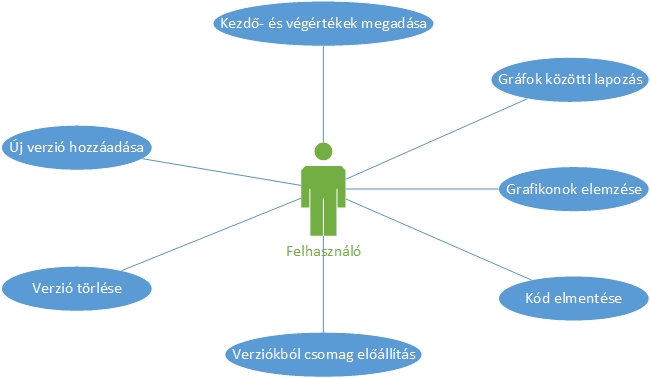


12. ábra - Színezés helyreállítása

Amennyiben a gráf színezésre került valamilyen módon, akkor csúcson kívülre kattintva a *Színezés helyreállítása* ábrán látható felugró ablakrész jelenik meg, ahol eldöntheti, hogy a színezést vissza akarja-e állítani alapértelmezett helyzetébe. A visszaállítást az *Igen* gomb lenyomásával lehet megtenni.

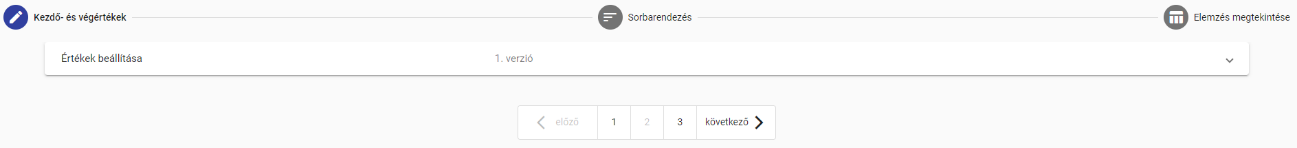
A másik hátralévő funkció a *Továbblépés a következő oldalra.* Ezt az oldal alján található *Következő* gomb lenyomásával tehetjük meg, ilyenkor az oldal átugrik az elemzéseket végző felületre.

### Elemzéseket végző oldal



13. ábra - Elemző oldal felhasználói eset diagram

A *Elemző felhasználói eset diagram* ábráján látható, hogy a felhasználónak milyen funkciók és lehetőségek állnak rendelkezésére, amikor az elemzéseket végző oldalon van. Az oldal működésének előfeltétele, hogy a kód feltöltése már megtörtént, ezért ha az nem történt volna meg, akkor hibaüzenettel jelezzük, hogy először fájlt fel kell tölteni.



14. ábra - Elmező oldal egészében

Az *Elemző oldal egészében* ábrán látható, hogy léptetéses rendszerben működik. Az első oldalán adhatóak meg az állapotok melyekkel elemezni kívánja a kódot. A második oldal csak akkor elérhető, ha az első oldalon kettő vagy annál több verziót adott meg, ilyenkor itt egy csomagot állíthat össze. A harmadik oldalon pedig az elemzett gráfok tekinthetőek meg, valamint azon diagramok melyek a program felhasználtáságról adnak egy átfogóbb képet.

#### Kezdő- és végértékek megadása

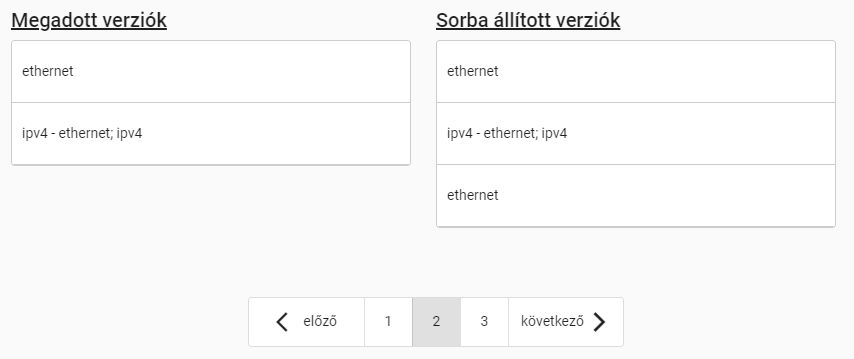


15. ábra - Értékek megadása

Ezen az aloldalon belül adhatja meg lenyíló modulok segítségével a kívánt verziókat. Az *Értékek megadása* ábrán látható, hogy a verziót lehet törölni a *Törlés* gomb lenyomásával, valamint új verziót is hozzá lehet adni az eddigiekhez ezt a *Plusz hozzáadás* gomb lenyomásával teheti meg. A kezdeti értékeket egy lenyíló többes kiválasztó mező segítségével adhatja meg, melynek bezárása után a bepipálásra kerülő értékek fognak megjelenni. A végértékeket ugyanilyen módszerrel lehet megadni, ennek a helye az ábrán nem látszódik.

Tetszőleges mennyiségű verziót lehet megadni, de ugyanolyan értékekkel rendelkezőt nem érdemes ugyanis a *Sorba rendezés* oldalon adható meg csomagszerkezet. Verzióból egynek kötelezően lennie kell, ezt nem tudja kitörölni, itt a *Törlés* gomb nem jelenik meg.

#### Sorba rendezés



16. ábra - Sorba rendezés

Amennyiben több verzió kerül megadásra, akkor elérhetővé válik ez az oldal. Ilyenkor a *Sorba rendezés* ábrán is látható módon jelenik meg a felület. Bal oldalon a *Kezdő- és végértékek megadása* aloldalon megadott verziók szerepelnek, ahonnan a már említett *drag and drop* módszerrel, vagyis egyszerű áthúzással lehet átvinni az elemeket a jobb oldali *Sorba állított verziók* oszlopba. Ilyen módon lehet csomagot összeállítani, vagyis egyes érték variációkat többször megadni, másikakat pedig kevesebbszer. Ennek az oldalnak a kitöltése nem kötelező, tehát üresen hagyható a *Sorba állított verziók* oszlop, ilyenkor a *Megadott verziók* oszlopában lévő sorrendet követi a rendszer.

#### Elemzés megtekintése

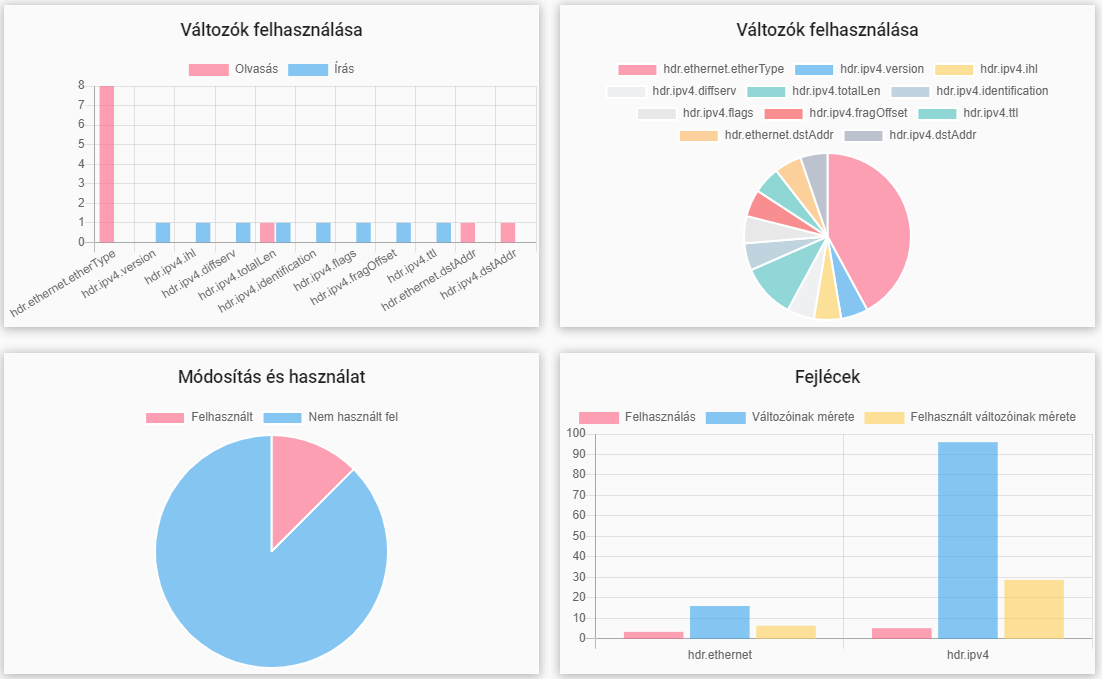
Ezen az oldalon megjelennek az elemzett vezérlésfolyamgráfok, adatfolyamgráfok, valamint olyan diagramok melyek segítségével könnyebben optimalizálható a kód.

##### Gráfok

Az összes megadott kezdő- és végértékhez meg lehet tekinteni az elemzett gráfokat. Itt a gráfok között lehet lapozni, mindkét gráf fajta külön-külön jelenik meg és külön-külön is lehet köztük lapozni. A gráfoknak színezve vannak, melyek értelmezése az alábbi:

* Zöld – A kód a megadott értékekkel helyes és nem keletkezik hiba.
* Sárga – A kód a megadott értékkel lehet, hibára fut, de ez nem megállapítható biztosan.
* Piros – A kód a megadott értékekkel hibára fut.

##### Diagramok

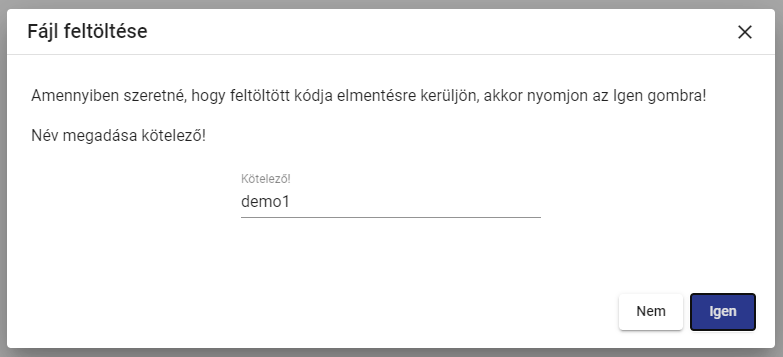


17. ábra - Diagramok

Az oldalon négy diagram található, ahogy az a *Diagramok* ábrán is látható. Melyeknek magyarázata és értelme a következő:

* *Változók felhasználása* (oszlop diagram) – Itt megtekinthető, hogy egy változót hányszor írtak, illetve olvastak átlagban.
* *Változók felhasználása* (kör diagram) – Itt megtekinthető, hogy egy változót hányszor használtak átlagban.
* *Módosítás és használat* – Itt megtekinthető, hogy a változók összesítve hányszor voltak újra felhasználva miután módosítottak rajtuk*.* Ezzel ellenőrizve, hogy a változó módosítása célszerű volt-e.
* *Fejlécek* – Itt megtekinthető, hogy egy fejléc átlagban hányszor volt használva, továbbá változóinak összesített méretét bit-ben kifejezve, valamint felhasznált változóinak átlagos méretét bit-ben kifejezve. Ebben az esetben a méret nem mindig pontos, mivel vannak olyan változók melyeknek nem ismerem a méretét.

##### Kód elmentése



18. ábra - Kód elmentése

Amikor az elemzés oldalra lép és itt is hibátlanul lefutnak a vizsgálatok, akkor megjelenik a *Kód elmentése* ábrán látható felugró ablak. Ilyenkor lehetőség van a kód elmentésére, azzal a feltétellel, hogy megadunk egy nevet neki. Ez az elmentett kód majd a Fájl oldalon lesz megtalálható. Az ablak bezárása négy módon történhet meg:

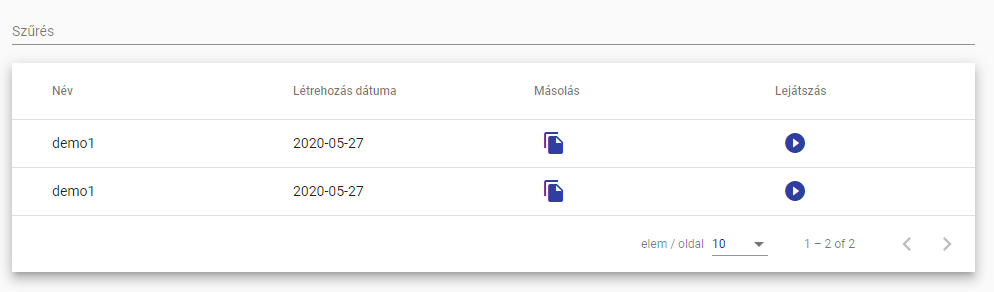
* *Igen* gombra kattintás – A fájl feltöltése megtörténik, ennek sikerességéről üzenetet kap
* *Nem* gombra kattintás
* Bezáró gombra kattintás

### Fájl oldal



19. ábra - Fájl oldal felhasználói eset diagram

A *Fájl felhasználói eset diagram* ábráján látható, hogy a felhasználónak milyen funkciók és lehetőségek állnak rendelkezésére, amikor az fájlkezelő oldalon van. A felhasználónak itt lehetősége van korábban már általa vagy más személy által elemzett kódok között böngészni. Ez egy kényelmi funkció, hogy olyan felhasználó is tudja használni az oldalt, aki saját maga nem tudna vagy nem szeretne P4 kódot írni.



20. ábra - Fájl oldal

Ahogy a *Fájl oldal* ábra is mutatja az oldalon egy táblázat található, ahol a szövegbeviteli mező segítségével lehet szűrni a *Név*-re és a *Létrehozás dátum*-ára, valamint ugyanezek szerint rendezhetőek is az oszlopok. Az átláthatóság miatt egy oldalon csak tíz elem jelenik meg, de ezt változtathatja ötre, vagy húszra is és több oldal esetén lapozhat is köztük.

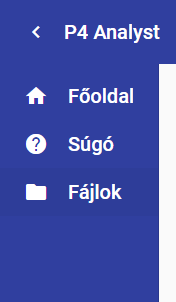
Az oldalon kettő akció hajtható végre az itt felsorolt fájlokra. Az első a *Másolás*, ami a fájl tartalmát vágólapra helyezi, a második pedig a *Lejátszás*, ilyenkor a fájl tartalmával átlép a Gráf megjelenítő oldalra, vagyis a fájlfeltöltés lépést átugorja.

### Súgó oldal

TODO Még nincsen kész az oldal, utána lehet leírni (Nem lesz hosszú)

### Kényelmi funkciók (TODO nem volt jobb ötletem)

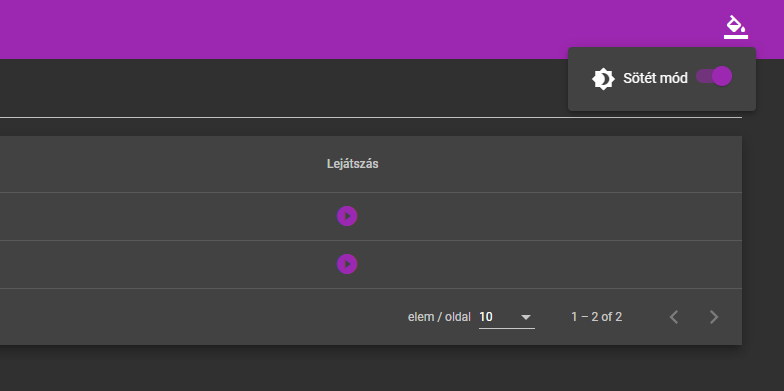
#### Navigációs rész (TODO kép ki lesz cserélve, mert lesz egy új menüpont (P4 főoldalra irányító gomb))



21. ábra - Navigációs sáv

A felületen mindig elérhető a bal felső sarokban elhelyezkedő ikon, melynek megnyomásakor a *Navigációs sáv* ábrán látható felület jelenik meg az oldalon. Ennek segítségével könnyen megnyithatjuk az ábrán látható három oldal egyikét.

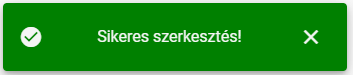
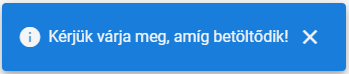
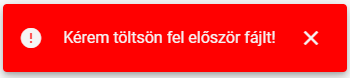
#### Sötét téma



22. ábra - Sötét téma

A felület jobb felső sarkában található ikonra rányomva a *Sötét téma* ábrán látható menü jelenik meg, ahol ki- és bekapcsolható a sötét mód.

#### Értesítések



23. ábra - Értesítési panelek

A felületen az *Értesítési panelek* ábrán látható módon jelennek meg üzenetek a felhasználó számára. Ezek értelmezése:

* Zöld – Sikeresen végbehajtott művelet
* Kék – Információval szolgáló üzenet
* Sárga – Figyelmeztető üzenet
* Piros – Hiba üzenet

# Fejlesztői dokumentáció

### Fejlesztői környezet

Szakdolgozatom fejlesztéséhez két alkalmazást használtam. Az első a Visual Studio 2019 integrált fejlesztői környezet, melyben alkalmazásom háttérfolyamatait készítettem C# objektumorientált programozási nyelven. Használt verziók:

* .NET Core 3.1+
* .NET Standard 2.1+

A másik alkalmazás pedig a Visual Studio Code, ahol az Angular keretrendszert alkalmaztam.

#### Környezet felépítése

Node.js 12.16.1+ verzió letöltése és telepítése innen: <https://nodejs.org/>

Visual Studio 2019 bármelyik verziójának letöltése és telepítése innen: <https://visualstudio.microsoft.com/vs/>

Angular telepítése ez alapján: <https://cli.angular.io/>

Ekkor a kód futtatható és fejleszthető. Az Angular cli adta lehetőségeket, mint például a komponens generálást a *Node.js command* prompt alkalmazásban lehet végezni.

Visual Studio Code-hoz ajánlott kiegészítők:

* Angular Language Service – Programozást segítő kiegészítő. Osztályok, változók megadásánál segít opciós lista megjelenítésével.
* TSLint (deprecated) – Kód helyességet figyel és programozási szabvány betartásában segít.
* Material Icon Theme – A mappákat, fájl ikonokat változtatja meg. Jobban láthatóak az Angular egyes részei.

#### C#

<http://nyelvek.inf.elte.hu/leirasok/Csharp/index.php?chapter=1>

A *C#* a *Microsoft* által kifejlesztett általános célú imperatív programozási nyelv. A nyelv a *C++* kifejezőerejét igyekszik ötvözni a *Visual Basic* egyszerű használatával Java-s elemekkel kiegészítve. A szintaktikus alapokat a C++-tól kölcsönzi, míg az egyszerűséget a Visual Basic-től. A C#-ot sokan hasonlítják még a *Java*-hoz. A nyelv támogatja az objektum-elvű programfejlesztést, valamint tartalmazza a C++ programnyelvi eszközeinek jelentős részét és néhány újszerű elemet is. A nyelv a .NET keretrendszer bázisnyelve, kényelmes és gyors lehetőséget biztosít ahhoz, hogy a keretrendszer alá alkalmazásokat készítsünk. A C# és a .NET keretrendszer alapja a *Common Language Infrastructure(CLI)*.

A nyelv jellemzői:

* Professzionális, Neumann-elvű. Nagy programok, akár rendszerprogramok írására is alkalmas.
* A program több fordítási egységből vagy fájlból áll. Minden egyes modulnak vagy fájlnak azonos a szerkezete.
* Egy sorba több utasítás is írható. Utasítások lezáró jele a pontosvessző (;). A változókat deklarálni kell. Változók, függvények elnevezésében az ékezetes karakterek használhatóak. Kis- és nagy betűk különbözőek.
* Minden utasítás helyére összetett utasítás (blokk írható).
* A változók típusa érték (alaptípusok, *enum, struct, value*) illetve referencia (*class*) típus lehet.
* Nincs mutatóhasználat (csak "nem biztonságos" környezetben); biztonságos a vektorok használata
* Boxing, unboxing. Minden típus őse az object, így például egy egész típust (*int*) csomagolhatunk objektumba (boxing) illetve vissza (unboxing).
* Függvények definíciói nem ágyazhatók egymásba, önmagát meghívhatja (rekurzió). Tehát függvénydefiníció esetén nincs blokkstruktúra. A blokkokon belül statikus vagy dinamikus változók deklarálhatók. A függvény polimorfizmus megengedett.
* Függvényparaméterek lehetnek: érték, referencia (*ref*) és output(*out*).
* Kezdő paraméter-értékadás, változó paraméterszámú függvény deklarálása.
* Új operátorok: *is* operátor egy objektum típusát ellenőrzi. (*x is Négyszög*), *as* operátor a bal oldali operandust jobb oldali típusúvá konvertálja (*Négyszög* *l=x as* *Négyszög*). Ez abban különbözik a hagyományos konverziós operátortól, hogy nem generál kivételt.
* Privát konstruktor (nem akarunk egy példányt sem), statikus konstruktor (statikus mezők inicializálása, mindig példány konstruktor előtt hívódik meg, futási időben az osztálybetöltő hívja meg) használata.
* Nincs destruktor, helyette a keretrendszer szemétgyűjtési algoritmusa van. Ha szükséges az osztály *Dispose* metódusa újradefiniálható

#### Angular

<http://nyelvek.inf.elte.hu/leirasok/JavaScript/index.php?chapter=27>

Az Angular egy Google által fejlesztett, nyílt forráskódú JavaScript keretrendszer dinamikus webes alkalmazásokhoz. Segítségével nagyban egyszerűsödik a webes alkalmazások frontend fejlesztése. Használatával a HTML eszköztára kibővül és az alkalmazások komponensei még egyértelműen elkülönülnek. Az Angular adatkapcsolásának, illetve függőség injektálásának köszönhetően, rengeteg felesleges boilerplate kód elhagyható. A keretrendszer főbb célkitűzései:

* Felület definiálására ideális egy deklaratív leírás (HTML), míg az imperatív programozás kiváló, hogy kifejezze az üzleti logikát.
* Válasszuk le a DOM (Dokumentum Objektum Modell) manipulációt az alkalmazás logikáról.
* A program tesztelése legalább olyan kritikus, mint annak írása.
* Az alkalmazás kliens, illetve szerveroldal teljes szétválasztása.

#### TypeScript

<http://nyelvek.inf.elte.hu/leirasok/TypeScript/index.php?chapter=1>

A TypeScript egy objektum-orientált script nyelv, amit a Microsoft készített. Legfőbb filozófiája nyelvnek az, hogy "legyen a TypeScript bővebb halmaza a Javascript-nek". A Javascript-től eltérően, legnagyobb újdonsága, hogy statikusan típusos a nyelv.

Cél nem egy újabb Javascript-gyilkos nyelv megalkotása volt, hanem egy olyan eszköz elkészítése, mely segíti a fejlesztőket az igazán nagy projektek elkészítésében is. A projekt fontosságát mutatja az is, hogy a Windows 8 fejlesztésénél a WinRT egy részét is ezen a nyelven írták.

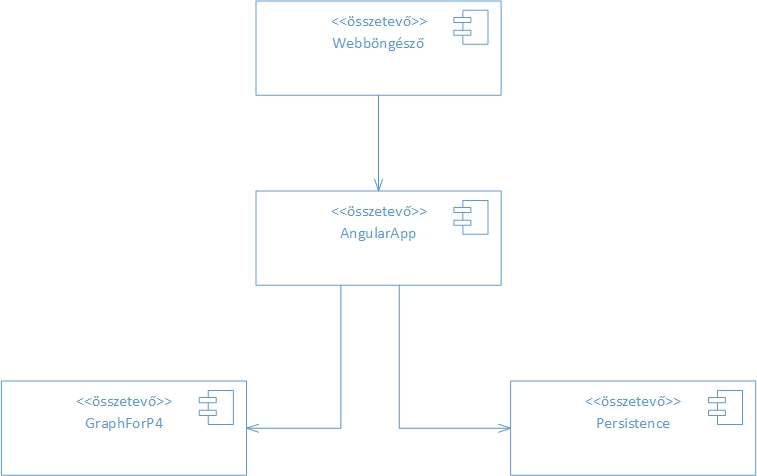
Bár a TypeScript hivatalosan bővítése a JavaScript-nek, előfordulhatnak olyan esetek, ahol a működő JavaScript kódra a TypeScript fordító hibát jelez. Például, ha egy a HTML kódban előre megadott rádiógomb csoport elemein szeretnénk végigmenni egy for ciklussal és meg szeretnénk nézni, hogy melyikre kattintott a felhasználó, azaz melyiknek igaz a „checked” tulajdonsága, azt nem tehetjük meg a Javascriptben megszokott if(rg[i].checked) ellenőrzéssel, ahol az „rg” rádiógombok csoportja, az „i” pedig a ciklusváltozó, hanem castolnunk kell a következő formában: if ((x[i]).checked).

A nyelv teljesen nyílt forráskódú, illetve operációsrendszer független. Mivel a fordító a TypeScript forráskódból Javascript kódot generál, így böngészőfüggetlen is a nyelv. Nem szükséges semmiféle külső program vagy plugin telepítése sem.

Nagy előnye, hogy felülről kompatibilis a Javascript nyelvvel, így a már meglévő Javascript-es projektjeinket (pl.: jQuery, Raphael, stb.) is változtatások nélkül fel tudjuk használni a TypeScript-ben. Ezenfelül, hogy a statikusan típusos nyelvi támogatást is megkaphassuk a Javascript-es projekteknél lehetőség van definíciós fájlok segítségével TypeScript kompatibilissé tenni őket.

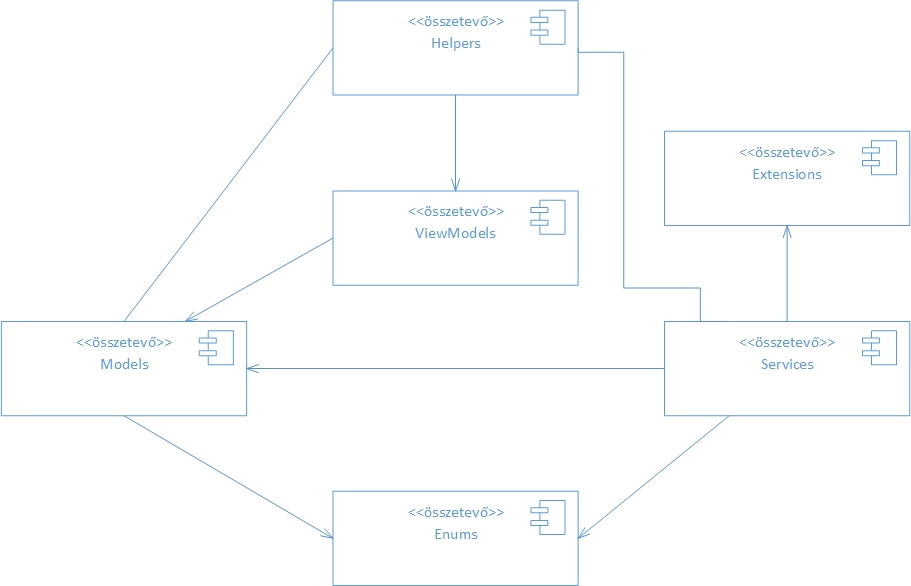
### Probléma leírása

### Program leírása



24. ábra - Program komponens diagramja

#### GraphForP4

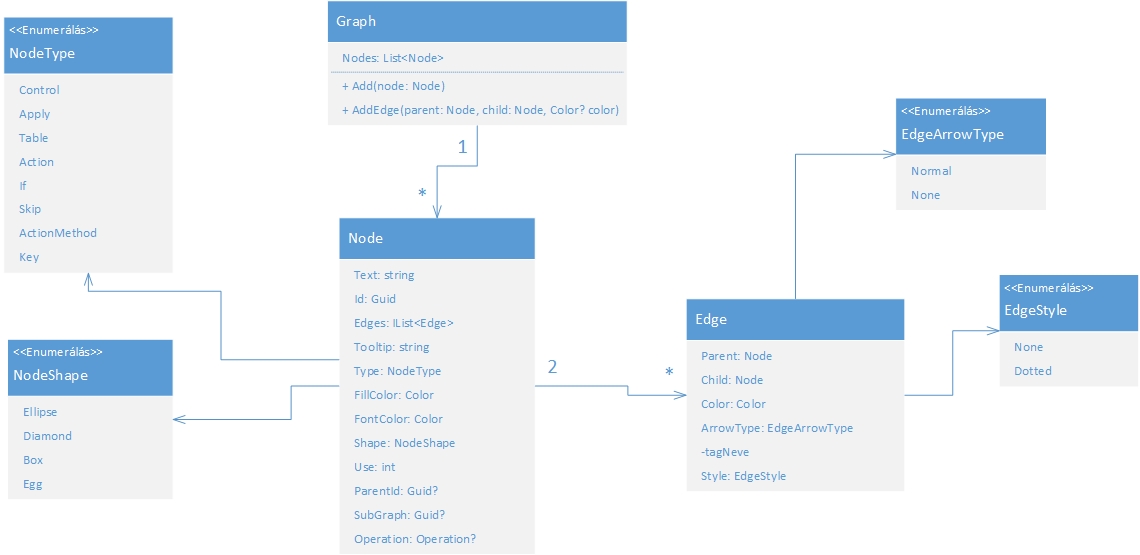


25. ábra - GraphForP4 komponens diagramja

A GraphForP4 komponens diagramja ábrán látszik, hogy milyen módon építettem fel a projektet. Komponensek:

* Models – Az adat leíró modellek, melyeket a számításokhoz és elemzésekhez használok itt találhatóak.
* ViewModels – Nézetnek átadandó vagy nézettől kapott modellek szerepelnek itt. Az Angular kódban is ugyanezek a modellek megtalálhatóak.
* Enums – Használt enumerációk gyűjtő helye.
* Services – Az üzleti logika ezeknek az itt található osztályoknak a segítségével lett megalkotva.
* Helpers – Olyan metódusok gyűjtő helye, melyek több komponensben előfordulnak, így kiszervezésük szükséges volt.
* Extensions – Használt kiegészítők kerültek ide.

##### Models



26. ábra - Gráf felépítése

#### AngularApp

#### Persistence

# Tesztelés

# Összefoglalás

# További fejlesztések

## P4 fordítóprogram beépítése

A vizsgált nyelv kibővítésének és a megfelelő hibajelzéseknek előfeltétele, hogy az elemzendő kód helyességét a P4 fordító segítségével ellenőrizzük. Mivel a P4 fordító egyetlen hivatalosan támogatott operációs rendszere az *Ubuntu 16.04* és az alkalmazás jelenleg Azure-ra van kitelepítve, ezért ennek beépítésére jelenleg két módszert látok megvalósíthatónak.

### Egész projekt Ubuntun-on történő futtatása

Az első megoldás, hogy létrehozunk egy Ubuntu operációs rendszerrel rendelkező virtuális gépet, amelyre telepítjük a P4 fordítóját és az elemzett programot lefuttatjuk rajta. A futtatást C#-ból meg lehet tenni parancssori parancsok futtatásával. Ezt a megoldást nem ajánlom, mivel így fejlesztői környezetben is ki kell alakítani egy virtuális gépet, ahol tesztelhetők a módosítások.

### Szerver a P4 fordítóprogramhoz

A második megoldás, hasonló az elsőhöz, de itt csak egy szerver applikációt telepítünk a virtuális gépre, aminek egy belépési pontja van, amit meg lehet hívni a futtatandó kóddal és visszatér a program helyességével. Ehhez a megoldáshoz is kell fejlesztői környezetet kialakítani, de itt csak egy applikációt kell tesztelni rajta és a jövőben kevés változtatást kell eszközölni rajta, így az új fejlesztőknek nem feltétlenül van szükség a környezet kialakítására. Így én ezt a megoldást javaslom.

## Vizsgált résznyelv kiterjesztése

A vizsgálat során kikötésekkel éltem, hogy milyen programkódot tud feldolgozni az elemző programok, ezen kikötések feloldása lenne a feladat. A következőkkel egészíthető ki a résznyelv:

* Parser és Deparser feldolgozása – A fejlécek inicializálása és tovább küldése ne a felhasználó beállítása szerint menjen, hanem a kettő függvény alapján.

## Felhasználó azonosítása

A felhasználó regisztrálhasson vagy belépjen meglévő fiókjába. A felhasználóbarát megközelítés miatt legyen lehetőség *Google, Facebook* vagy *Github* felhasználóval belépni. A felhasználó azonosítása nagyon sok lehetőséget von maga után. Lehetőség nyílik, hogy a felhasználó feltöltsön kódot és eltárolja saját maga, megjelölt másik felhasználók, vagy az egész közösség számára. Legyen lehetőség egymás kódjaira reagálni, ötleteket, megjegyzéseket adni, vagy kedvencnek jelölni, ezzel később is gyorsan megtalálható. A fájl kereső oldalon így bővíthetőek a keresési funkciók felhasználó és kedveltségi szint alapján.

A már feltöltött kód módosítását és előzmények megtekintését is meg lehet csinálni, valamint elmenteni a kód elemzéseit, hogy a felhasználó nyomon tudja követni, hogy mennyit fejlődött kódjának optimalizáltsága. Amennyiben a programba beépítésre kerül az [*5.1*](#_P4_fordítóprogram_beépítése)-es pontban említett P4 fordítóprogram, akkor a felhasználó tudja fordítani programját, ezzel ellenőrizve helyességét. Ezen funkciók segítségével, akár egy fejélesztői környezet kialakítása is eszközölhető lenne.

Projektek létrehozása, ahol egy vagy több felhasználó képes módosítani a kódot.

## Gráf elemzés összesített verziója (TODO lehet kikerül)

Az elemzés megtekintésénél jelenleg minden program lefutásnak, amit a különböző kezdeti értékek segítségével ad meg a felhasználó külön-külön kirajzoljuk a gráfot, de nincs egy olyan összesítő gráf, amely bemutatná a leggyakrabban használt kódrészleteket. Ennek megalkotására az adatok már rendelkezésre állnak, a különböző gráfokban szerepel, hogy egy rész hányszor lett használva, ezeket összesíteni kell és ezek alapján beállítani a csúcsok színezését.

## Felület optimalizálás

### Sötét téma

A felületen sötét téma került kialakításra, amely megfelelő kontrasztot és láthatóságot nyújt egy eset kivételével. A gráf éleinek színe alapértelmezetten fekete, ez a szín sötét téma esetén a sötét szürkén látható, de a szemnek fárasztó fókuszálni rá, ezért ezt módosítani kell. A gráfot teljesen újra kell építeni egy másik a sötét szürkével kontrasztban álló élszínezéssel.

### Mobil használat

A felület jól használható mobilos környezetben, de vannak szépítendő és a felhasználói élményt javító változtatási lehetőségek.

Mobil optimalizáció:

* Az oldalsáv menü megjelenítése elhúzással. Ezzel könnyítve a felhasználót a navigációs menü megnyitásában, hogy ne kelljen mindig a megnyitó gombot lenyomni. Fejlesztéskor figyelni kell, hogy az animációs gombot használjuk.
* Gráf megjelenítő oldalon a tab-os elrendezésben elhúzással (swipe) is tudjon oldalt váltani.
* A megjelenített gráfok mozgatásának és nagyításának megvalósítása. A felhasználó a gráfra nyomással nyisson meg egy felugró ablakot, ahol ezek a funkciók aktívak. Elegendő lenne csak a nagyítás funkció aktiválása, azonban a használt gráf ábrázoló könyvtár ezt a megoldást nem támogatja.
* A gráf valamiért nem a képernyő közepén jelenik meg és az újra méretezhetőség sem működik rajta teljes értékűen. Ezen esetekre nem tudtam megoldási ötletet találni.