Szegedi Tudományegyetem

Informatikai intézet

**Szakdolgozat**

**Ujszászi János**

**2021**

Szegedi Tudományegyetem

Informatikai intézet

**DXF fájlba mentett épületterv dokumentum megjelenítése és felületszámítása**

Szakdolgozat

Készítette:

**Ujszászi János**

programtervező informatikus

szakos hallgató

Témavezető:

**Tóth Zoltán Gábor**

egyetemi PHD hallgató???

**Szeged**

**2021**

# Feladatkiírás

Egy épület tervezése során mindig nagy munka mind a tervezőnek, mind az épület leendő tulajdonosának a költségek megállapítása. Ezek a számok az emberi tényező miatt gyakran hibásak, és már csak egy szakágon belül, egy felület mennyiségére is ritka, hogy a tervező, a kivitelező és a tulajdonos ugyanazt az értéket számolja ki. Mindhárom szereplő számára előre lépést jelenthet, ha a számított mennyiségek pontosak.

A hallgató célja egy online platform kialakítása, ahol a felhasználó feltöltheti a tervezőtől kapott DXF fájlt. A feltöltött fájlt parsoljuk, tároljuk szerver oldalon, a parsolt adatokat relációs adatbázisban mentjük, canvas objektumban megjelenítjük. A számítási algoritmus elméleti szinten tárgyaljuk magas szintű tervezési metodikával.

A feladat megvalósítása közben az előforduló problémakörök kibontása detektálástól, hibakeresésen keresztül a megoldásig. A fellelt dokumentációk felhasználásának ismertetése.

# Tartalmi összefoglaló

* ***Téma megnevezése:***

Épületterveken megtalálható objektumok felületszámítása DXF fájlban mentett tervdokumentum alapján

* ***A megadott feladat megfogalmazása***

A hallgató célja egy online platform kialakítása, ahol a felhasználó feltöltheti a tervezőtől kapott DXF fájlt. A feltöltött fájlt parsoljuk, tároljuk szerver oldalon, a parsolt adatokat relációs adatbázisban mentjük, canvas objektumban megjelenítjük. A számítási algoritmus elméleti szinten tárgyaljuk magas szintű tervezési metodikával.

* ***A megoldási mód***

Egy HTML/CSS alapú online felületen drag&drop módon feltölthető platform kialakítása. A feltöltött fájl tárolása szerver oldalon és annak parsolás back end oldalon java osztály hierarchiába. Az osztályba mentett adatokat relációs adatbázisban is tároljuk. Az osztályokban mentett egyedeket JSON adatformátumként visszaadjuk frontend oldalra ahol JavaScriptben kialakított bővebb osztálystruktúrába mentjük a parsolt adatokat. Minden specifikus Entitás osztálynak felüldefiniált rajzolási eljárása készül, ami egy canvas hívást valósít meg.

A falazatok és egyéb objektumok meghatározására és mérésére több lehetséges eljárást ismertetünk és a megvalósítás során gyűjtött tapasztalatok alapján pro/kontra érvrendszer segítségével végleges megvalósítható megoldást választunk.

* ***Alkalmazott eszközök, módszerek***

A online felületet HTML5, CSS felhasználásával készítem el.

Az adatbázis MYsql-lesz,

Backend: Java

Eredmény visszaírás: JavaScript

* ***Elért eredmény***

ENTITIES szekció parsolása és megjelenítése, azok tárolása relációs adatbázisban.

Kalkulációs megoldások kidolgozása és azok elemzése.

* ***Kulcsszavak***

DXF feldolgozás, parsolás, entitás, visszarajzolás, kalkuláció, tervező, megrendelő, kivitelező

**Tartalomjegyzék**

[Feladatkiírás 3](#_Toc100031186)

[Tartalmi összefoglaló 4](#_Toc100031187)

[Bevezetés 8](#_Toc100031188)

[1. Lakóház költség meghatározása 10](#_Toc100031189)

[1.1 Tervezői költség meghatározása 10](#_Toc100031190)

[1.2 Kivitelezői költség meghatározás 10](#_Toc100031191)

[1.3 Megrendelői költség meghatározás 11](#_Toc100031192)

[2. DXF felépítése, adatkapcsolatok kapcsolatok 13](#_Toc100031193)

[2.1. Általános ismertető 13](#_Toc100031194)

[2.2. Felépítése 13](#_Toc100031195)

[2.2.1. Header 13](#_Toc100031196)

[2.2.2. Classes 13](#_Toc100031197)

[2.2.3. Tables 15](#_Toc100031198)

[2.2.4. Blocks 16](#_Toc100031199)

[2.2.5. Entities 16](#_Toc100031200)

[2.2.6. Object 17](#_Toc100031201)

[3. User interfész 18](#_Toc100031202)

[3.1. Beviteli képernyő 18](#_Toc100031203)

[3.2. Eredmény visszajelző képernyő 19](#_Toc100031204)

[4. Parsolás kezelése 21](#_Toc100031205)

[4.1. BackEnd oldali adatszerkezetek használata és osztály struktúra 21](#_Toc100031206)

[4.2. Parsolás menete 22](#_Toc100031207)

[4.3. Parsolt adat átadása FrontEnd felé 23](#_Toc100031208)

[5. Adatbázis felépítése, és tárolás 24](#_Toc100031209)

[5.1. Táblák 24](#_Toc100031210)

[6. Elemek visszarajzolása Javascripttel 25](#_Toc100031211)

[6.1. Osztály struktúra 25](#_Toc100031212)

[6.2. Entitás típusokhoz felüldefiniált draw eljárások bemutatása 26](#_Toc100031213)

[6.2.1. LINE 27](#_Toc100031214)

[6.2.2. LINE 27](#_Toc100031215)

[6.2.3. LINE 27](#_Toc100031216)

[6.2.4. LINE 27](#_Toc100031217)

[6.2.5. LINE 27](#_Toc100031218)

[6.2.6. LINE 27](#_Toc100031219)

[6.2.7. LINE 27](#_Toc100031220)

[6.3. Arányok kezelése 27](#_Toc100031221)

[7. Adatfeldolgozó algoritmus, adatértelmező logikák 27](#_Toc100031222)

[7.1. Objektum felismerés 27](#_Toc100031223)

[7.1.1. Falak felismerése 27](#_Toc100031224)

[7.2. Objektumtól eltérő adat eltávolítás 27](#_Toc100031225)

[7.3. Objektumra vonatkozó speciális tulajdonságok meghatározása 28](#_Toc100031226)

[7.4. Objektumtól függő számítási metódus 28](#_Toc100031227)

[8. Eredmény visszaadása UI felé 28](#_Toc100031228)

[8.1. Eredmény visszaadása 28](#_Toc100031229)

[8.2. Felismert objektum visszarajzolása 28](#_Toc100031230)

[9. Alkalmazás tesztelése 28](#_Toc100031231)

[9.1. Modul szintű tesztelés 28](#_Toc100031232)

[9.2. EndtoEnd tesztelés 28](#_Toc100031233)

[Irodalom jegyzék 28](#_Toc100031234)

[Nyilatkozat 28](#_Toc100031235)

# Bevezetés

Egy épület tervezése során mindig nagy munka mind a tervezőnek, mind az épület leendő tulajdonosának a költségek megállapítása. Ezek a számok az emberi tényező miatt gyakran hibásak, és már csak egy szakágon belül, egy felület mennyiségére is ritka, hogy a tervező, a kivitelező és a tulajdonos ugyanazt az értéket számolja ki. Mindhárom szereplő számára előre lépést jelenthet, ha a számított mennyiségek pontosak.

A szereplők között egy tévesen meghatározott érték, már a bizalmi kapcsolat kialakulása előtt, rányomhatja a bélyeget a későbbi kapcsolatra.

* A tervező:
  + Hiányosan jelöli ki tervező szoftverbe a számításban érintett objektumokat,
  + Vagy csak rosszul állítja be az objektumra vonatkozó számítási paramétereket.
* A tulajdonos:
  + Excel táblás struktúrában szinte 100%-os a hibázás lehetőség
  + Hozzá nem értése, egy objektum valós méretét tekintve.
* A kivitelező:
  + Bele számolja az ablak helyét is a falazásba.
  + Felhasznált anyag alapján számol.

Ezek a példák is jól mutatják, hogy nagyon nehezen tud jól indulni egy ilyen kapcsolat.

Ahhoz hogy egy mindenki számára használható eszközt készítsek, fel kell mérni a szükséges igényeket.

A dolgozatban, minden felhasználó felé mutató eseményt mindhárom potenciális felhasználói csoport szemszögéből be fogom mutatni.

Alapvető igények, elvárások az alkalmazással kapcsolatban:

* Tervezői oldalról egyértelmű elvárás hogy a szellemi tőke megmaradhasson a tervezőnél. Ezért a nyers tervezésben érintett fájl formátum felhasználása nem lehetett opció.
* Mindhárom oldalról elvárás a kényelem.
  + Több embernapnyi munka váltható ki egyszerűen
* Mindhárom oldalról elvárás az egyszerűség
  + Egy újabb bonyolult, több opciós paraméterezéssel működő eszköz nem használható hatékonyan
* Eredmények átláthatósága.
  + Opciót kell biztosítani a felhasználó felé, hogy meghatározza, az eredményben visszaadott objektum típusokat, egyedeket.
    - Hanem definiálja mindent vissza kell adni.
  + Ezzel elkerülve, hogy a rengeteg adatba elvesszen a mezei user

# Lakóház költség meghatározása

## Tervezői költség meghatározása

Minden épület tervezés ügyfél egyeztetésekkel indul. Az ügyfél szeretne egy épületet, ami az igényeinek megfelelő és ehhez van egy reális vagy éppenséggel irreális költségkerete. A tervező feladatai közé tartozik, hogy a költség keretnek megfelelő épületet tervezzen. Ez a feladat egy a sokból és ez csak egy paraméter az épület tervezése során, ami sok feladat mögé kerül, mert fontosabb, hogy az épület:

* biztonságos legyen
* megfeleljen a helyi és országos építésihatóság által előírt szabályoknak
* megfelelő időtállóanyagokkal valósuljon meg a kivitelezés
* illeszkedjen a helyi és/vagy területi építési környezetbe, stílusba

Sajnos a költségvetési kiírás sok esetben messze van a gyakorlati megvalósítástól, illetve az objektumok alapján számol. Ergo ha egy objektum nincs megnevezve, címkézve azt csak manuálisan tudja hozzátenni a tervező.

Ideális esetben minden lerajzolt vonal, vagy réteg költsége becsülhető lenne, de ehhez minden lerajzolt vonalhoz azonosítás szüksége, hogy az az objektum éppen micsoda. Ez a címkézés időigényes.

A tervezők számára ideális megoldás lehet viszont az, hogy a tervezés során részfázisokat kiexportálnak, és azokra végeztetnek költségelemzést.

## Kivitelezői költség meghatározás

A mai építőipari környezetben egyetlen hibásan elkészített árajánlat komoly veszteséget jelenthet egy kivitelező számára, mert nem csak az elszámolás miatt, hanem a folyamatos áremelkedés is.

Több szempontból sem előnyös, ha a kivitelező számolja ki a mennyiségeket:

* időt és energiát visz el az értékesebb fizikai tevékenységtől
* általában a számolások munka után második műszakban történik, ami növeli a hiba lehetőséget
* hiba esetén bizalmatlanság alakulhat ki.
* visszaélésre adhat lehetőséget mind az ügyfél mind a kivitelező szemszögéből.

Az ideális támogató rendszer kiszámolja a mennyiségeket és anyag bontásba adja át a kivitelező felé azokat. Egy leolvasott mennyiséghez nagyon egyszerű a rétegrend további elemét kikalkulálni.

Szemléltetés egyszerű példával: Külső falazat.

* Bentről kifelé indulva.
  + A falazatot, csupasz falat belülről borítja valami.
    - Burkolat
      * A burkolat alá vakolat kell
      * Vagy gipszkarton
        + alá vázszerkezet
        + vagy ragasztó
    - Festék
      * A festék alá glett
        + alá vakolat
        + vagy gipszkarton

Ezzel az egyszerű példával szemléltehető, hogy egy meghatározott külső falazat belső felületéhez mennyi különböző anyag mennyiség, és munkabér költség határozható meg.

## Megrendelői költség meghatározás

A megrendelő a tervezői és kivitelezői hibáknak és visszaéléseknek az elszenvedője. Ha nincs a kezében egy eszköz vagy nem kér fel borsos áron ellenőrt, akkor marad az excel tábla és több hetes számolgatási procedúra.

Az ilyen jellegű ellenőrző tevékenység a következő problémákat hordozza:

* Ha már ellenőrzés szükséges a részünkről akkor már probléma van.
  + Vagy a tervező rontott el valamit vagy a kivitelező számolt rosszul, de a két szám eltér, ezért szükséges a validálás.
  + Az ügyfél szintű ellenőrzés általában kezdetleges eszközökkel valósul meg, rutin és tapasztalat nélkül.
  + Nagy mértékű hibalehetőség
  + Gyakori hiba kellemetlené teszik a kapcsolatot, hitelesség csökken.
  + Kezdeti költségelemzés meghatározásakor nagy mértékű pontatlanság.

Nincs hatékony segítség a megrendelő kezében.

# DXF felépítése, adatkapcsolatok kapcsolatok

## Általános ismertető

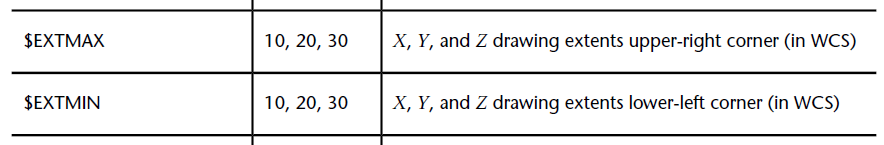
A DXF egy fix szabályrendszer alapján felépített, ASCII szöveges állomány. Két egymást követő sor mindig összetartozik. Az első sor egy típuskódot ad meg, ami három karakter hosszú balról padolva SPACE karakterrel. a második pedig a típushoz tartozó értéket. A fájl több szekcióból állhat, de a szekciók közül csak az ENTITIES kötelező. A fájl végén EOF kulcsszó található.

Minden szekció előtt meg kell adni, hogy egy új szekció következik a „ 0”, SECTION kóddal, ezt követően „ 2”, Szekció neve következik. A tartalmat követően a szekció „ 0” kóddal és „ENDSEC” kulcsszóval zárjuk.

## Felépítése

## Header

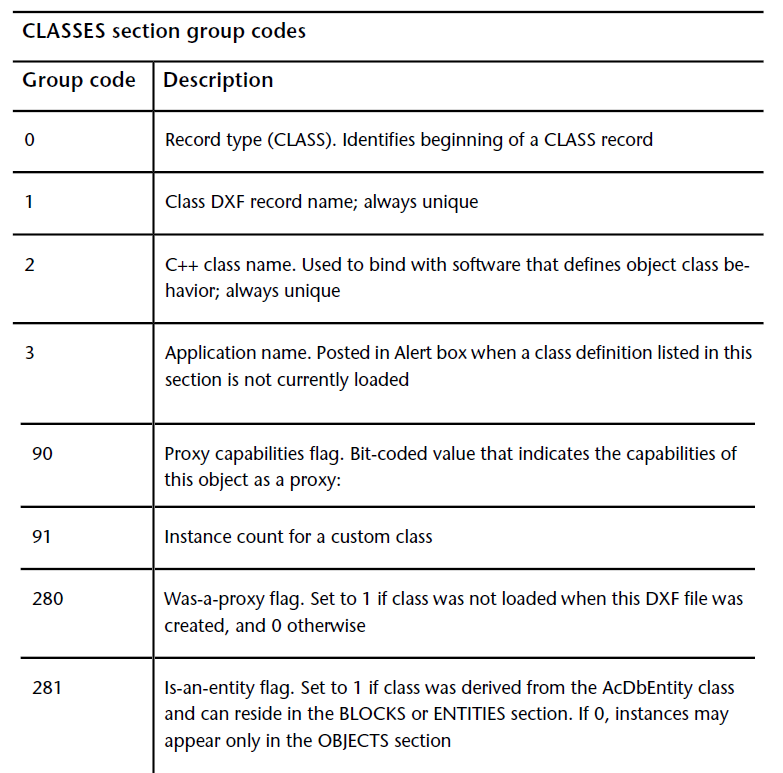
A HEADER szekcióban az általános fájladatok találhatóak a beállított változók formájában. A változók elnevezése általában $ karakterrrel kezdődnek a teljesség igénye nélkül itt található a fájl formátumának verzió száma, a kép méretőhez köthető különböző változók, az entitásokhoz társítható általános preferenciák, mint például hogy egy körív előre definiált forgásiránya ($ANGDIR) óramutató ellentétet, vagy nem.



1. ábra Header formátuzm leírás hivatalos dokumentációból

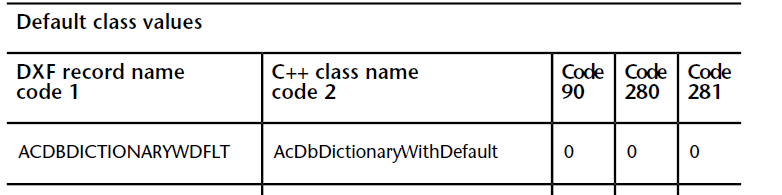
## Classes

A CLASSES szekcióban azok az osztályok találhatóak amit az alkalmazás példányosít és megjelennek a BLOCKS, ENTITIES és OBJECT szekcióban. Az osztályokat előre definiált mezőkkel lehet leírni és minden mezőnek a kitöltése kötelező. A mezők a 2. ábrában találhatóak.



2. ábra Classes szekciót leíró adatok

Vannak előre definiált osztályok is, amik fix elnevezési párosítással érthetőek el. Ennek a leírásnak a formátuma a 3. ábrában látható.



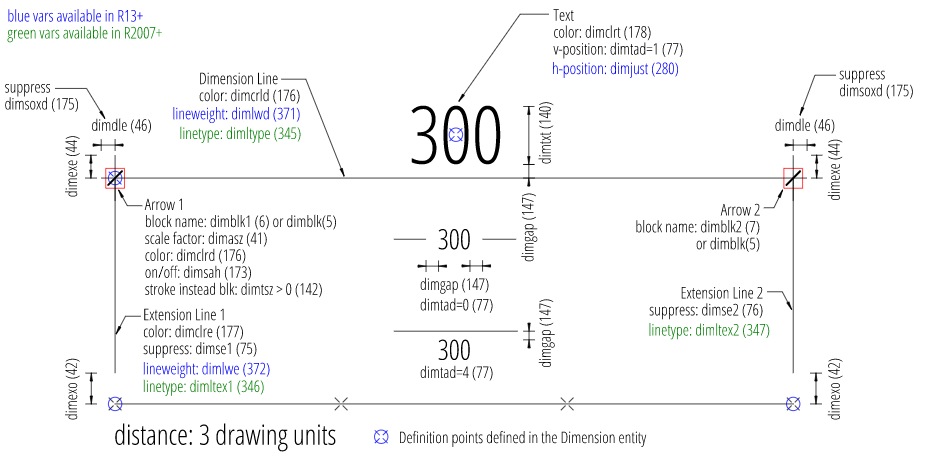
3. ábra Default classes

Az dolgozatban a szekciót nem dolgozzuk fel. Figyelmen kívül hagyjuk.

## Tables

A TABLE szekcióban a következő táblázatban megtalálható típus definíciók deklarálhatóak.

|  |  |  |
| --- | --- | --- |
| Név | Kód | Leírás |
| Linetype table | LTYPE | A DXF fájlban megtalálható LINE entitásnál megjelenő vonaltípus definíciókat tartalmazza. |
| Layer table | LAYER | Rétegeket lehet definiálni |
| Text Style table | STYLE | A szöveg stílusokat tartalmazza |
| View table | VIEW | A rajzterület elrendezéseinek nézeteit tárolja. Nem befolyásolja a rajzot, de az AUTOCAD alkalmazásnak segít a feldolgozásban. |
| User Coordinate System table | UCS | Nevesített vagy nem megnevezett felhasználói koordináta rendszer, ami a CAD alkalmazások használnak. |
| Viewport configuration table | VPORT | A DXF fájlban megtalálható VIEWPORT entitásnál megjelenő elem definíciókat tartalmazza. |
| Dimension style table | DIMSTYLE | A DIMENSION Entitások stílusmeghatározása |
| Application Identification table | APPID | Alkalmazásoknak fenntartott definíciós lehetőségek |



4. ábra DIMSTYLE ábrázolása (<https://ezdxf.readthedocs.io/>)

Az dolgozatban a szekciót nem dolgozzuk fel. Figyelmen kívül hagyjuk.

## Blocks

A blokkok entitások gyűjtemény amik több példányban elhelyezhetőek a rajtérben, eltérő elrendezésben, eltérő helyen.

Egy blokk bejegyzés a BLOCKS szekióban BLOCK bejegyzéssel kezdődik és ENDBLK-val zárul közben INSERT, ATTRIB, ATTDEF block referenciákat kezel.

Az dolgozatban a szekciót nem dolgozzuk fel. Figyelmen kívül hagyjuk.

## Entities

Az EINTITIES az egyetlen kötelező szekció. A rajzi elemeket tartalmazza.

Az ENTITY-k „ 0” kóddal kezdődik. ez mutatja meg a ENTITY típusát. Az ENTITIES végét szintén ENDSEC zárja. Közben akármennyi ENTITY előfordulhat. A szekciónak előforduló entitásoknak vannak általános típus független és típusfüggő jellemzői.

Az általános jellemzők a „ 0” típus definíció, elnevezések, hivatkozások más szekciókra.

A specifikus jellemzők között is átjárás van, de vannak kirajzolhatósághoz elengedhetetlen jellemzők, amiket specifikusan a 6.2 fejezetben fogok kifejteni.

## Object

Az ENTITIES-hez hasonló általános tulajdonságokkal rendelkező szekció, de itt nem grafikus, megjelenítendő elemek találhatóak.

Az OBJECT szekcióban megtalálható elemek például a DICTIONARY, GEODATA, MATERIAL.

Az dolgozatban a szekciót nem dolgozzuk fel. Figyelmen kívül hagyjuk.

# User interfész

A User intetrfész egyképernyős webalkalmazásban jelenik meg. Az alkalmazás UI kialakítása során HTML, CSS, és Javascript technológiák falhasználása történt.

Az animációk megvalósítása @keyframes használatával készült.

A visszarajzolás felülete canvas

## Beviteli képernyő

Az alkalmazás induláskor megjelenő háttár egy adásmentes TV képernyőre hasonlítható, amiből kettő másodperc elteltével egy terminál lesz látható. A terminálban üdvözlő üzentek, illetve bemutatkozás jelenik meg sorról sorra.

Az animációk CSS stíluslappal egymás után mennek végbe a következő sorrendben:

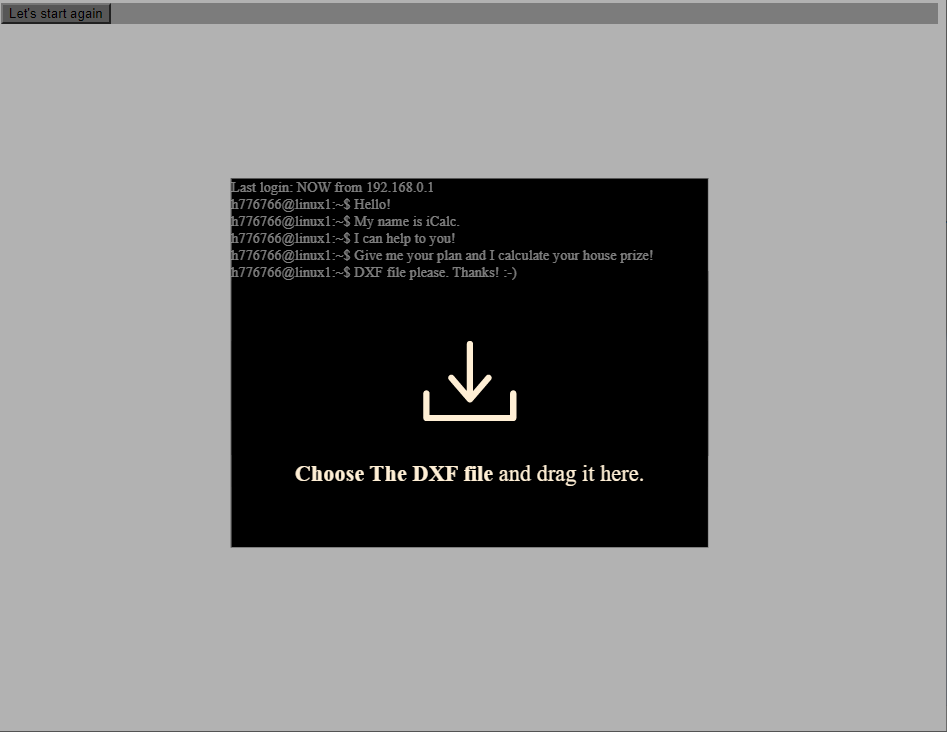
* A @keyframes backgroundchg segítségével a kezdeti noise.gif képet előszőr fehérre változtatjuk
* Majd ugyanebben a @keyframe-ben szürkére.
* Közben a megjelenítjük fokozatosan a terminál felületét
* A terminálban miután a háttér szürkére váltott a @keyframes cursor-visible animációval egymás után jelenítjük meg a kiírt szöveget.
* Végül megjelenik a drag&drop area-t

A bemutatkozást követően előtűnik a drag&drop felület ahová a felhasználó feltöltheti a kezelendő DXF fájlt.

A fájl felületre mozgatása módosítja a feltöltés alatt megtalálható szöveget

A fájl elengedés és a feldolgozó eljárás megkezdése szintén visszajelzésre kerül a felhasználó számára.

Ha fájl mérete túl nagy vagy a kiterjesztése nem DXF, azt alert formájában visszajelezzük a felhasználó felé, és ismételt fájlfeltöltés lehetséges.



5. ábra kezdő képernyő

## Eredmény visszajelző képernyő

A beolvasást követően a képernyőről eltűnik a terminál szöveges felület és browser méretével arányosan 90%.-osra nagyítjuk a terminál DIV objektumát. Az animációt követően a terminál DIV objektuma eltűnik és ugyanebben a pillanatban megjelenítjük az azonos méretű és pozíciójú canvas objektumot.

A canvas jobb oldalának 220 pixel nagyságú részében egy calculation rész helyezkedik el ahova majd a számítási eredményeket írnánk vissza.

A maradék rajzfelület 1%-os kertet kap.

A visszarajzolás során arányosítjuk a rajzokat. ezért minden esetben az eredeti méretarányokat megtartva adjuk vissza az eredményt a felületre. Az arányosítással kapcsolatos teljes leírás az 5.3. fejezetben lesz elérhető.



6. ábra Eredmény visszaírása képernyő

# Parsolás kezelése

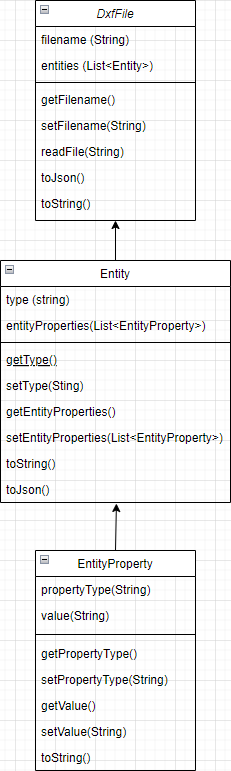
Az adatok parsolása során kihasználjuk a formátum legfontosabb szabályát miszerint a teljes formátum adatpárokból épül fel. Tehát egy típus után a következő sorban egy érték fog jönni.

Mivel nagy az adathalmaz a parsolásnak gyorsnak kellett lennie ezért a kezdeti feldolgozó eljárás a fájl első bejárása során a teljes fájlt kezelte és strukturálta osztályokba.

Az átadott fájlokat backend oldalon mentjük.

## BackEnd oldali adatszerkezetek használata és osztály struktúra

A DXF file backend oldali kezelését három osztállyal lehet megvalósítani, amik között erős aggregációs kapcsolat van.



7. ábra backend class diagram

## Parsolás menete

A parsolás eljárás során a HEADER szekcióból a $EXTMIN és $EXTMAX értékeket gyűjtötte össze, mert ezek az adatok határozzák meg a rajtábla jobbfelső és bal alsó pontját, ami szükséges hogy a visszarajzolt kép elhelyezkedése és mérte megfelelő legyen Ezeket az adatokat entitásként kezeli a parser.

A header szekcióból érkező kettő speciális entitást a tömb elejére helyezkedik el. Ezeknek az entitásoknak négy entityproperty-vel kellett dolgozni. A 10,20,30 egy pontot írt le a 9-es pedig figyelmen kívül hagyható volt.

A parsolás további része akkor kezdődik ha a fájlban elérjük a ENTITIES szekciót.

A while futása közben segéd változókkal határozzuk meg hogy éppen milyen osztályból kell példányosítani. Az entitás kezdetekor („ 0” típus ID esetén) kigyűjtjük az entitás típusát és létrehozzuk a EntityProperties listát amiben gyűjtjük a következő „ 0” értékig a entityproperty értékpárokat. A lista zárásával az entity objektumot példányosítjuk és ürítjük a entityproperty listát. A boolean segédváltozókat a kezdeti állapotba állítjuk.

A parsolás metódus egészen a ENDSEC részig fut. Ezt követően a file feldolgozása már nem hoz létre entitásokat.

A feldolgozás végén egy kapcsolható logolás található, ami kilistázza nekünk a kigyűjtött entitásokat. Feldolgozási idő gyorsítása miatt volt szükség a logolás kapcsolhatóságára, mert a logolás gyakorlatilag újra végig olvassa az összegyűjtött entitások tömbjét.

## Parsolt adat átadása FrontEnd felé

A feldolgozott adatok visszaküldése frontend oldalra a JSON formátumban valósult meg. Az osztály struktúra DxfFile és Entity osztályai a toJson eljárás segítségével állították elő a JSON formátumot, a JSON eljárás végén az osztály felüldefiniált toString eljárását használtuk, így a return már egy valid JSON formátumot eredményezett.

Azért ezt a formátumot használtam, mert a JSON formátum a JS frontendnek az egyik legmegfelelőbb, mert a visszatérő stringet a json() hívással könnyen kezelhető objektummá lehet alakítani.

# Adatbázis felépítése, és tárolás

A parsolt adatokat backend oldalon táblákba rendezzük. A rendezett adatokból, egy fájl visszaalakíthatóvá, de az aktuális parsolo logika szándékosan szűri a tárolandó adatok mennyiségét a szekciók segítségével.

## Táblák

A „file\_basic\_data” nevű tábla fogja tartalmazni a fájl feltöltésének eseménykor elérhető adatokat.

Mezők:

|  |  |  |  |
| --- | --- | --- | --- |
| Mezőnév | Megnevezés | Megkötés | típus |
| fl\_id | szekvenciából osztott unique index | not null constraint | number(15) |
| fl\_name | fájl név | not null constraint | varchar2(150) |

A „entities\_basic\_data” nevű tábla tartalmazza a az entitás fő adatait az ENTITIES szekcióból.

Mezők:

|  |  |  |  |
| --- | --- | --- | --- |
| Mezőnév | Megnevezés | Megkötés | típus |
| en\_id | szekvenciából osztott unique index | not null constraint | number(15) |
| en\_fl\_id | Foreign key a file\_basic\_data táblához | not null constraint | number(15) |
| en\_type | Entitás típusa | not null constraint | varchar2(150) |

A „entities\_attr\_data” nevű tábla tartalmazza a az entitáshoz köthető tulajdonságok azonosító típuskódját és az ahhoz tartozó értéket. adatait az ENTITIES szekcióból.

Mezők:

|  |  |  |  |
| --- | --- | --- | --- |
| Mezőnév | Megnevezés | Megkötés | típus |
| ei\_id | szekvenciából osztott unique index | not null constraint | number(15) |
| ei\_en\_id | Foreign key a entities\_basic\_data táblához | not null constraint | number(15) |
| ei\_type | EntitásPropery típusa |  | varchar2(150) |
| ei\_value | EntitásProperty típushoz tartozó érték |  | varchar2(255) |

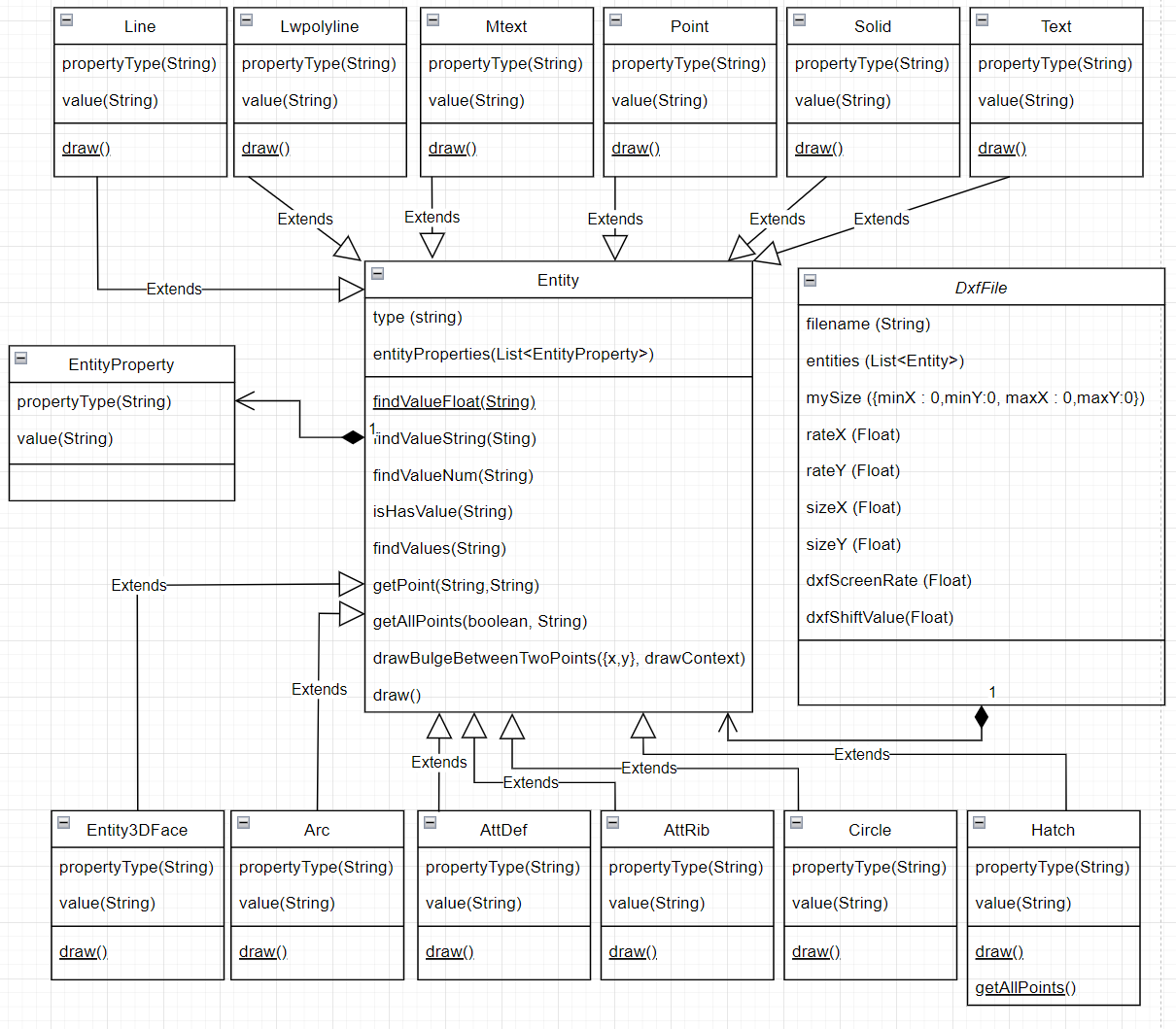
# Elemek visszarajzolása Javascripttel

A visszarajzolás során a JSON struktúrában érkező adatokat javascript osztályszerkezetű objektumokba szerveztük.

A JSON adatok első elemei a $EXTMAX és $EXTMIN entitások voltak. Ezek feldolgozását a cikluson kívül megejtettük hogy példányosítani lehessen egy DxfFile objektumot, ahova a további entitásokat és a hozzájuk tartozó entitástulajdonságok típus, érték párjait tároltuk el.

## Osztály struktúra

Az osztály struktúrát a következő CLASS diagramm írja le.



8. ábra javascript class diagram

## Entitás típusokhoz felüldefiniált draw eljárások bemutatása

### LINE

### LINE

### LINE

### LINE

### LINE

### LINE

### LINE

## Arányok kezelése

# Adatfeldolgozó algoritmus, adatértelmező logikák

A tábla struktúrából kiolvasott adatokat JAVA-ban fogjuk objektumokba rendezni.

## Objektum felismerés

Az első fázisban kizárólag a falak felismerését valósítom meg, de általános megoldást keresünk a hogy a későbbiekben könnyebben bővíthető legyen a felismer objektumok listája.

### Falak felismerése

Ezt még nem tudom

A falak esetén könnyű az ellenőrzés, mert minden esetben egy gráfként értelmezve egy Hamilton-körként be kell tudnunk járni az objektumokat a vég és kezdő pontok felhasználásával.

## Objektumtól eltérő adat eltávolítás

Ez majd kódolás közben jön és azt dokumentálom

## Objektumra vonatkozó speciális tulajdonságok meghatározása

Ez majd kódolás közben jön és azt dokumentálom

## Objektumtól függő számítási metódus

Ez majd kódolás közben jön és azt dokumentálom

# Eredmény visszaadása UI felé

## Eredmény visszaadása

HLD szint már kifejtve az UI-nál. ide csak az LLD rész fog kerülni

## Felismert objektum visszarajzolása

HLD szint már kifejtve az UI-nál. ide csak az LLD rész fog kerülni

# Alkalmazás tesztelése

## Modul szintű tesztelés

Szöveg

## EndtoEnd tesztelés

Szöveg

# Irodalom jegyzék

# Nyilatkozat