Ülőhelykiosztási probléma vizsgálata optimalizálási és mesterséges intelligencia módszerek segítségével

**Hallgatók:**

* Egyed Vince (KSWOV1)
* Kertész Krisztián Levente (GDJK9Q)
* Puskás András (FO2GGY)

**Konzulens oktatók:**

* Dr. Fogarassyné Dr. Vathy Ágnes
* Dr. Süle Zoltán

Tartalomjegyzék

[Feladat leírás 1](#_Toc68611576)

[1. Tervezés és előkészítés 2](#_Toc68611577)

[1.1. COVID-19 egészségügyi szabályok 2](#_Toc68611578)

[1.2. Választott fejlesztési nyelv és környezet 2](#_Toc68611579)

[1.3. Felhasználó felület 3](#_Toc68611580)

[1.4. Adatok kezelése 4](#_Toc68611581)

[1.5. Adatok tárolása 4](#_Toc68611582)

[1.6. Verzió követés 5](#_Toc68611583)

[2. Implementáció 6](#_Toc68611584)

[2.1. Felhasználó felület 6](#_Toc68611585)

[2.1.1. Táblázat 7](#_Toc68611586)

[2.2. Adatok kezelése 9](#_Toc68611587)

[2.3. Adatok tárolása 10](#_Toc68611588)

[2.4. Gépi megoldó algoritmus 11](#_Toc68611589)

[2.4.1. Mohó algoritmus 11](#_Toc68611590)

Feladat leírás

A feladat egy olyan grafikus alkalmazás és hozzá tartozó módszertan kidolgozása, amely lehetővé teszi különféle eseményekre történő helyfoglalási folyamat optimalizálását a COVID-19 egészségügyi szabályok betartása mellett.

Az alkalmazásban lehetőségünk van dinamikusan definiálni a termet, illetve a székek elrendezését is. A terembe a felhasználó tudja elhelyezni az ülőhelyeket megfelelő előre definiált egészségügyi szabályoknak megfelelően. Az alkalmazás továbbá optimalizálás vagy mesterséges intelligencia segítségével is be tudja rendezni a termet a maximális kihasználtság érdekében.

**Részfeladatok:**

* Programozási nyelv, platform kiválasztása
* Felhasználói felület elkészítése
* Dinamikus terem elkészítése
* Egészségügyi szabályoknak megfelelő kritériumok programozása
* Felhasználói interakciók mögöttes logikájának programozása
* Optimalizálás vagy mesterséges intelligencia segítségével a feladatmegoldó elkészítése
* Alkalmazás dokumentálása

# Tervezés és előkészítés

## COVID-19 egészségügyi szabályok

A járványhelyzet miatt kialakult egészségügyi szabályok a színházakra és mozi termekre is egyaránt vonatkoznak, ennek okán szükséges volt megismernünk a járvány miatt életbe hozott szabályokat. A termekre vonatkozó szabályozás úgy korlátozza az ülésrendet, hogy egymás mellett csak az egy csoportba tartózkodó személyek foglalhatnak helyet. Két csoport között legalább egy széknek üresen kell maradni.

Ezeket a szabályozásokat figyelembe véve a program elkészítése során is figyelembe kellett venni ezeket a szabályozásokat és ennek megfelelően szabályrendszert kell definiálni az alkalmazásba.

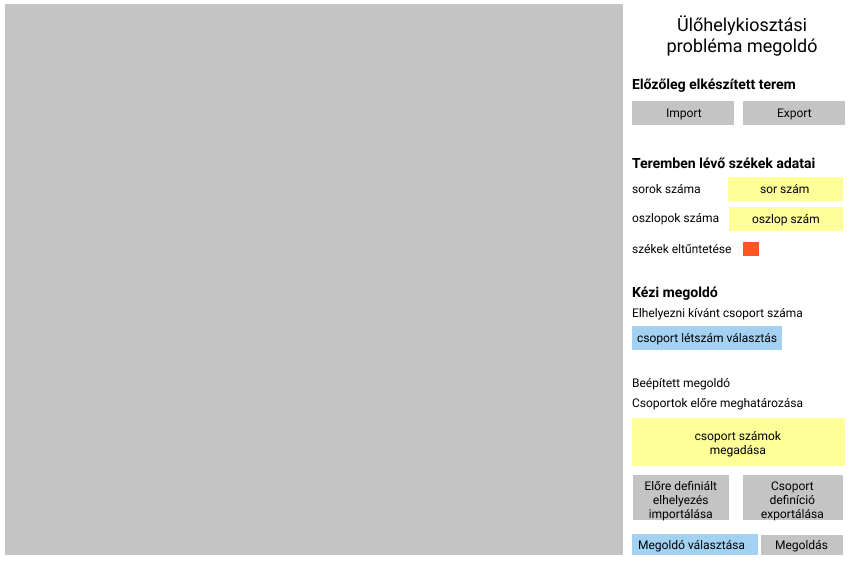
## Választott fejlesztési nyelv és környezet

Fejlesztési nyelvként java-t választottunk, mert komfortosan tudunk benne programozni. A fejlesztést IntelliJ IDEA-ban végeztük. A fejlesztési környezetben könnyedén tudtuk elkészíteni a Maven projektünket, amelyhez több könyvtárat is csatoltunk.

A csapatmunka miatt GitHub-on készítettük el a projektet. Az IntelliJ IDEA könnyű és letisztult beépített verzió követője miatt gördülékenyen tudtunk egymás mellett dolgozni a projekten.

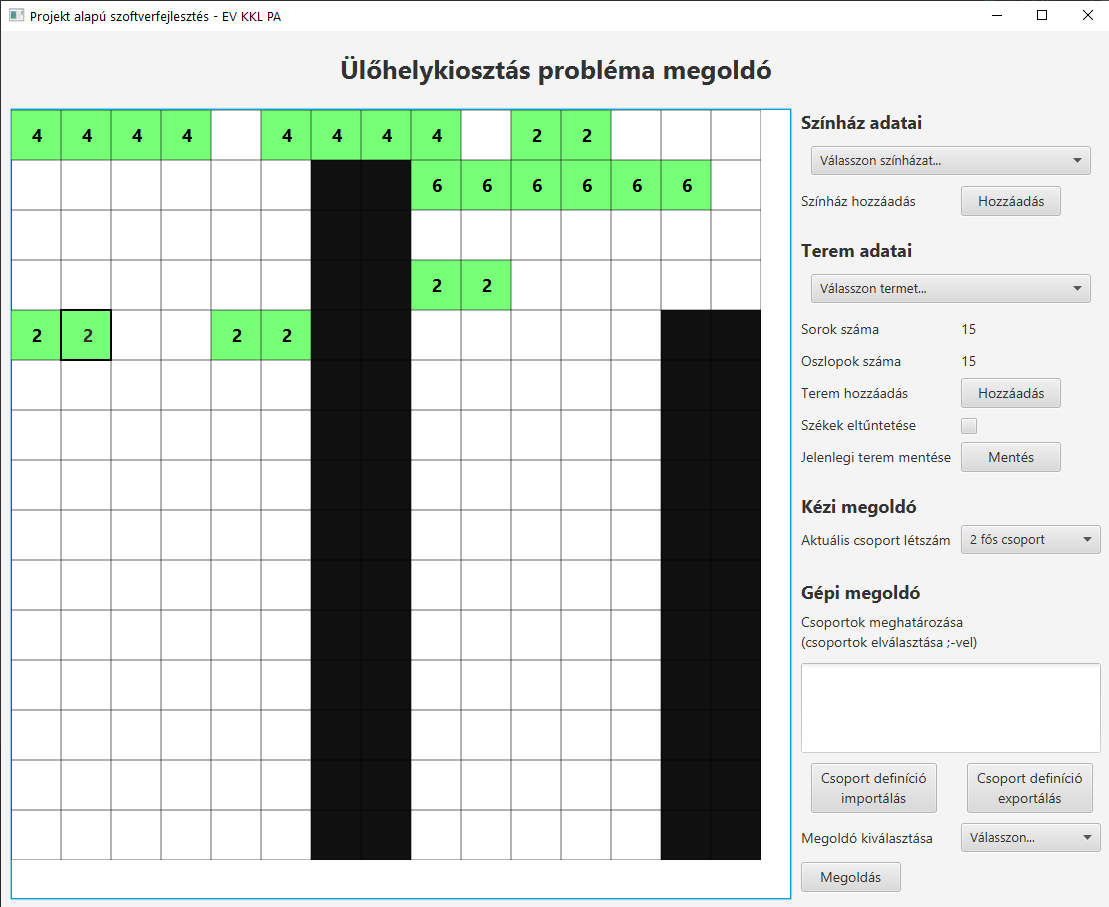
## Felhasználó felület

A felhasználó felület tervezése során törekedtünk, hogy érthető és könnyen átlátható alkalmazást készítsünk. A felhasználói felület megtervezésére egy online tervező alkalmazást használtunk. A tervezőben készült felületet az **1. ábra** szemlélteti. Az ábrán bal oldalt lévő szürke rész a teremet szemlélteti.



**1. ábra:** felhasználói felület terv (<https://www.figma.com>)

A fejlesztés közben az egyeztetések folyamán a felhasználó felület megváltozott, azonban az **1. ábrán** látható elrendezés megmaradt. Az elkészült felhasználó felületet a **2. ábra** szemlélteti. Az ábrán látható, hogy a jobb oldali sávban történtek változások a tervezethez képest. Ezek a változásokra amiatt volt szükség, mert az eltárolt termek is könnyedén tudja ezen a módon kezelni az alkalmazás.



**2. ábra:** elkészült felhasználói felület

## Adatok kezelése

A model osztályaiban implementáltuk az egyeztetett adatstruktúrát. Az alkalmazásban alapvetően Színházakkal (mint épületekkel), Termekkel és azokon belül Ülésekkel dolgozunk. Ezekhez később hozzávettük a Rendeléseket, melyek kezdetben csoportokat hivatottak reprezentálni, de elképzelhető, hogy a fejlesztés további fázisaiban másképp oldjuk meg ezt a problémát.

Továbbá robosztus kód alkotásának reményében több enum-ot és utils funkciót, valamint egy egységes adatfelhasználást lehetővé tevő interface-t is létrehoztunk. Utóbbit még nem egészen sikerült integrálni a rendszerünkbe, de dolgozunk rajta.

Ez azt a célunkat szolgálná, hogy ne csupán adatbázisból, vagy az alkalmazás segítségével tudjunk újabb adatokat generálni és tárolni, hanem import-export eljárások segítségével akár több forrással és formátummal is dolgozhassunk.

## Adatok tárolása

PA itt csak nagy vonalakban kell leírni, mert ez még csak a tervezés része, kód szinten itt nem kell semmit sem leírni.

## Verzió követés

A csapatban való munkához git verzió követőt használtunk. GitHub-on keresztül hoztuk létre a projektünket és minden egyes csapattagunkat hozzáadtuk a projekthez.

A projektünket könnyedén tudtuk ezen keresztül menedzselni és az egymással párhuzamos munka sem okozott gondot.

Github repository: <https://github.com/asdsajt/seat_allocation_problem>

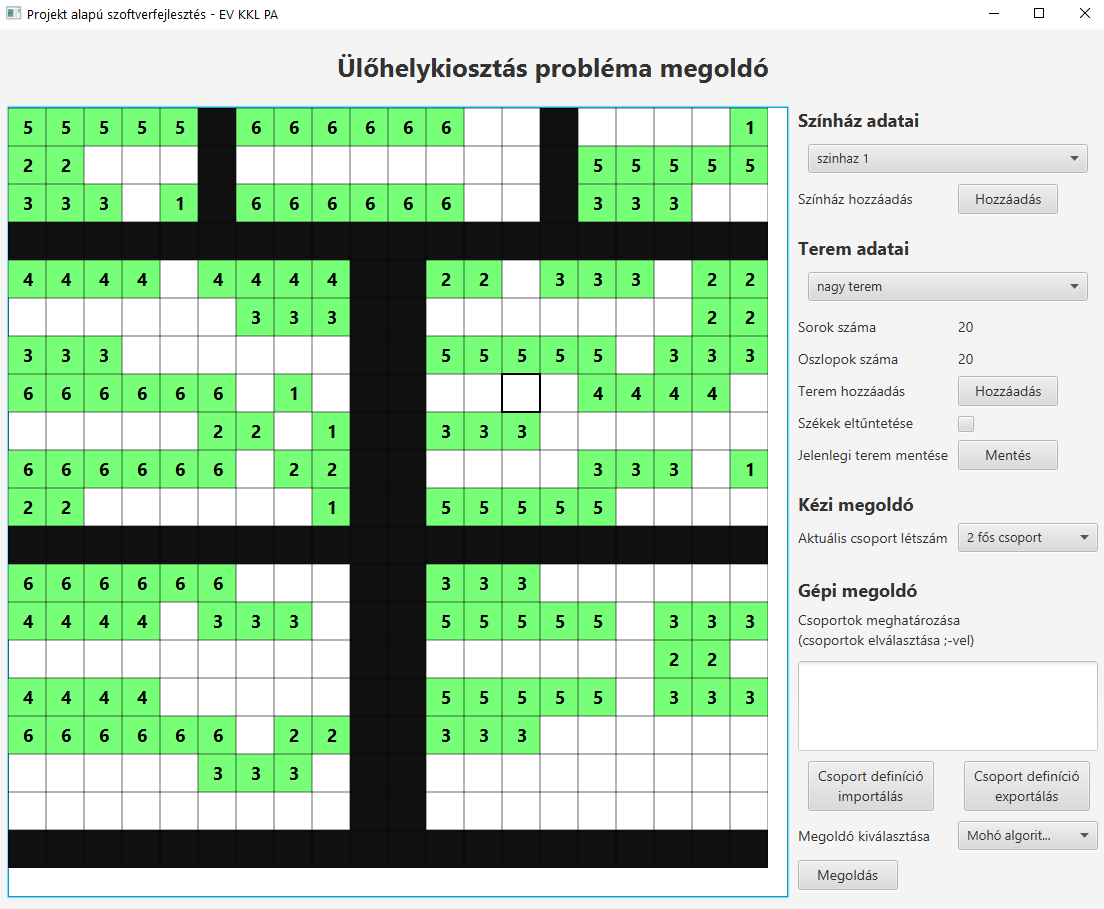
# Implementáció

## Felhasználó felület

A projekt egy nagy megjelenítő felületre épül, amelyen a felhasználó tudja végezni a teremmel kapcsolatos beállításokat, ülőhelyfoglalásokat. Ezen a felületen kívül még az alkalmazás tartalmaz három felugró ablakot, amely a mentésre és az új terem, színház felvételére szolgál.

A felhasználói felület elkészítésénél törekedtünk arra, hogy egy könnyen átlátható, intuitív felületet készítsünk el.

A projekt felhasználói felülete JavaFX segítségével készült el, ezen kívül egy Maven könyvtárat is használtunk a ControlsFX-t. A ControlsFX-ben található SpreadSheetView-val készült el a bal oldalon található táblázat. Nem használtuk a JavaFX-ben lévő beépített táblázatot, mert a SpreadSheetView-t könnyebben lehet kezelni és dinamikusabb, mint a JavaFX-ben lévő TableView.



**3. ábra:** felhasználói felület

### Táblázat

A felhasználó felület bal oldalán található táblázat képzi a program legnagyobb részét. Ezzel a táblázattal történnek a felhasználói interakciók és a beállítások is ezt a táblázatot módosítják. Ennek okán a táblázat sok eseményt, függvényhívást dolgoz fel.

#### Kattintás figyelők

A táblázathoz két kattintás figyelő esemény lett implementálva, amelyek elvégzik a táblázattal kapcsolatos felhasználói interakciók kezelését.

Az egyik kattintás figyelő segítségével a táblázatban lévő székeket lehet törölni, illetve visszaállítani. Erre amiatt volt szükség, hogy a felhasználó testre tudja szabni a termet, ki tudja jelölni, hogy hol nincsenek székek, azaz a lépcsők és a bejáratok helyét.

A másik kattintás figyelő komplex, mert az kezeli a felhasználó általi ültetéseket. A kattintás figyelő több kritérium alapján dönti el, hogy az aktuális cellába be lehet-e helyezni a felhasználó által választott csoportot, vagy sem. Az algoritmus tartalmazza, hogy két különböző csoport között minimum egy ülőhelyet ki kell hagyni a COVID-19 egészségügyi szabályainak megfelelően. Továbbá figyeli, hogy az aktuális helyre befér-e a csoport vagy sem, ebben az esetben az algoritmus figyeli, hogy jobbra, illetve balra mekkora távolság van. Figyelembe veszi, hogy egy másik csoport vagy törölt ülőhely van-e a legközelebb, mert ez is befolyásolja az algoritmus számolását. Amennyiben az algoritmus nem megfelelőnek találja a kiválasztott ülőhelyet a csoport letételéhez úgy hibaüzenet ad a felhasználó számára.

#### Színház adatai

A felhasználói felület jobb oldalán lévő sávja több kisebb részre van osztva, így a felhasználó az egyes alcímeknek köszönhetően könnyedén tud navigálni.

Az első ilyen rész a színház adatait tartalmazza. Ebben a egységben ki lehet választani már meglévő színházakat, illetve hozzá is lehet adni új színházat is a programhoz.

#### Terem adatai

A következő jobb oldali rész egységben a terem adatai szerepelnek. Ebben a részben ki lehet választani a kiválasztott színházhoz lévő termek közül azt, amelyet szeretnénk betölteni a programba. Amikor egy terem kiválasztásra kerül, akkor a program frissíti a jobb oldalon lévő táblázatot a terem méretei alapján, továbbá megjeleníti a törölt ülőhelyeket is.

A terem kiválasztása után a jobb oldalon is megjelennek a terem méretei szolgáló adatok is. Ezeken a felhasználó nem tud módosítani. Kivéve, ha új termet ad hozzá a felhasználó, mert a felugró ablakon be tudja állítani a terem méreteit.

A székek eltűntetésével lehet törölni az ülőhelyeket, így lehet a teremben lévő közlekedésre alkalmas részeket kijelölni. A törölt ülőhelyekre nem lehet már ülőhelyet foglalni.

A rész végén lévő mentés gombbal a felhasználó a terem jelenlegi állapotát tudja menteni. Mentésre kerülnek a törölt székek is, viszont a kiválasztott ülőhelyek nem.

#### Kézi megoldó

A kézi megoldó részen tudja a felhasználó kiválasztani, hogy hány főből álló csoportot szeretne elhelyezni a teremben. A táblázatba való elhelyezésnél a csoport létszáma is elhelyezésre kerül a kiválasztott cellán.

#### Gépi megoldó

A jobb oldal utolsó részén a gépi megoldó szerepel. A gépi megoldó első részén meg tudja adni a felhasználó, hogy milyen csoportokat szeretne elhelyezni a teremben. A csoportokat fájlból lehet importálni is, továbbá fájlba is lehet menteni az szövegmezőben szereplő csoport adatokat.

A felhasználónak lehetősége van kiválasztani, hogy milyen megoldóval szeretné elhelyeztetni az adatokat a teremben. A megoldó kiválasztása után a „Megoldás” gomb megnyomása után megjelenik a táblázaton a megoldó által készített megoldás. Továbbá egy felugró ablak jelzi, hogy a megoldó sikeresen elhelyezte-e az összes előre megadott csoportot a táblázatban vagy nem sikerült neki.

## Adatok kezelése

### Adatstruktúra

A modelt reprezentáló adatok ismertetésével kezdem, de akár többször is előfordulhat a utils package-re és osztályaira történő hivatkozás. Ezekről később ejtek szót.

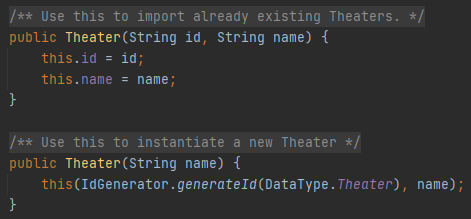
#### Theater

A színház osztálynak kettő adattagja van, az **azonosító** és a **név**.

Futás során két konstruktora közül a másodikat használjuk példányosításhoz – a többi id-vel ellátott osztályhoz hasonlóan. Ennek oka, hogy így a utils IdGenerator osztálya egységes és unique azonosítóval inicializálhatja az új objektumot.

Importálás és adatbázisból történő előhívás esetén viszont az első konstruktort alkalmazzuk, mivel itt biztosított a korábban eltárolt azonosító validitása.

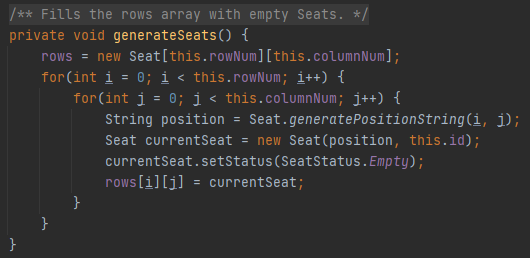
Az alkalmazás futása során az Id-t nem, de a nevet továbbra is módosíthatjuk.



*A Theater osztály konstruktorai*

#### Room

A szobákat **id** mezőjükkel azonosítjuk és **theaterId**-jükkel kötjük (lazán) a színházhoz amiben található. Továbbá elláttuk egy **név**vel és méreteit is számontartjuk **sorszám** (rN) és **oszlopszám** (cN) formában. Ezen méretparaméterek segítségével példányosításkor létrehoz egy rN\*cN-es tömböt amit feltölt üres ülésekkel. Az **üléstöm**böt nem, de az ülések státuszát minden további nélkül módosíthatjuk az applikáció futása során.

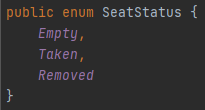


*A szobák sorainak feltöltése üres ülésekkel*

#### Seat

Az ülés konstruktorában meghatározuk, hogy melyik szobában (**roomId**) hol (**position**) – a megadott szoba tömbjének [row][column] koordinátái – találjuk.

Ezen felül (jelenlegi elképzelés szerint), ha valaki az alkalmazáson belül egy ülőhelyre kattint, **orderId** és **status** mezők módosítására nyílik lehetősége. Míg a redelés azonosító egy egyszerű String referencia (az Order object-re), a státusz-t a SeatStatus enum-ból választhatja ki a felhasználó. Ez a szabad (Free), a foglalt (Taken) és a használaton kívüli (Removed) értékeket veheti fel.



*A SeatStatus enum lehetséges értékei*

#### Order

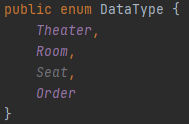
Egy kísérleti osztály – lehetséges, hogy teljesen kivesszük a programból – ami a csoportosításokat próbálja megvalósítani úgy, hogy egy (egyszeres vagy többszörös) kijelöléshez egy **id**-t rendel. Az Orderben tároljuk az szobát amire vonatkozik (**roomId**), a szobán belül pedig a kijelölt szék(ek) orderId mezőjét tesszük egyenlővé az Order id-jével.

### Utils

A model.utils több package és osztály együttese, melyet a Model programozhatóságának és olvashatóságának egyszerűsítése érdekében hoztam létre. Továbbá itt találhatóak az importerek, exporterek és az adatok belső tárolására használt InputData osztály. Ezeket a további fejlesztés során saját mappaszerkezetekbe fogom rendezni.

#### Utils.enums

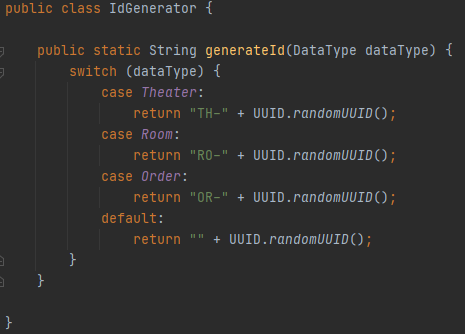
Ide tartozik az ülőhelyeknél említett SeatStatus, valamint a – jelenleg csupán Id generálásra szolgáló – DataType enum, melynek elemei megegyeznek az adatstruktúra osztályainak elnevezéseivel.



*DataType enum*

#### Utils.general

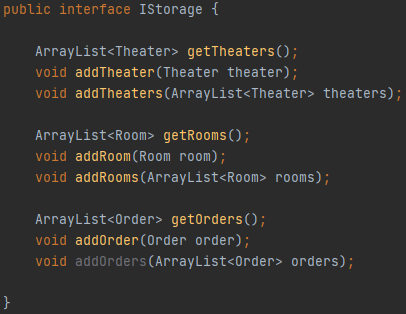
Általános felhasználású osztályok, függvények. Ide tartozik az IdGenerator.



*IdGenerator, ahogy DataType szerinti unique ID-t generál*

#### Utils.interfaces

Jelenleg csak az IStorage interface-t hoztam létre, ezt annak érdekében, hogy egységesítve lehessen hozzáférni a beolvasott adatokhoz, eltekintve attól, hogy éppen milyen formában tároljuk őket (adatbázis, InputData, stb.).



*IStorage interface*

## Adatok tárolása

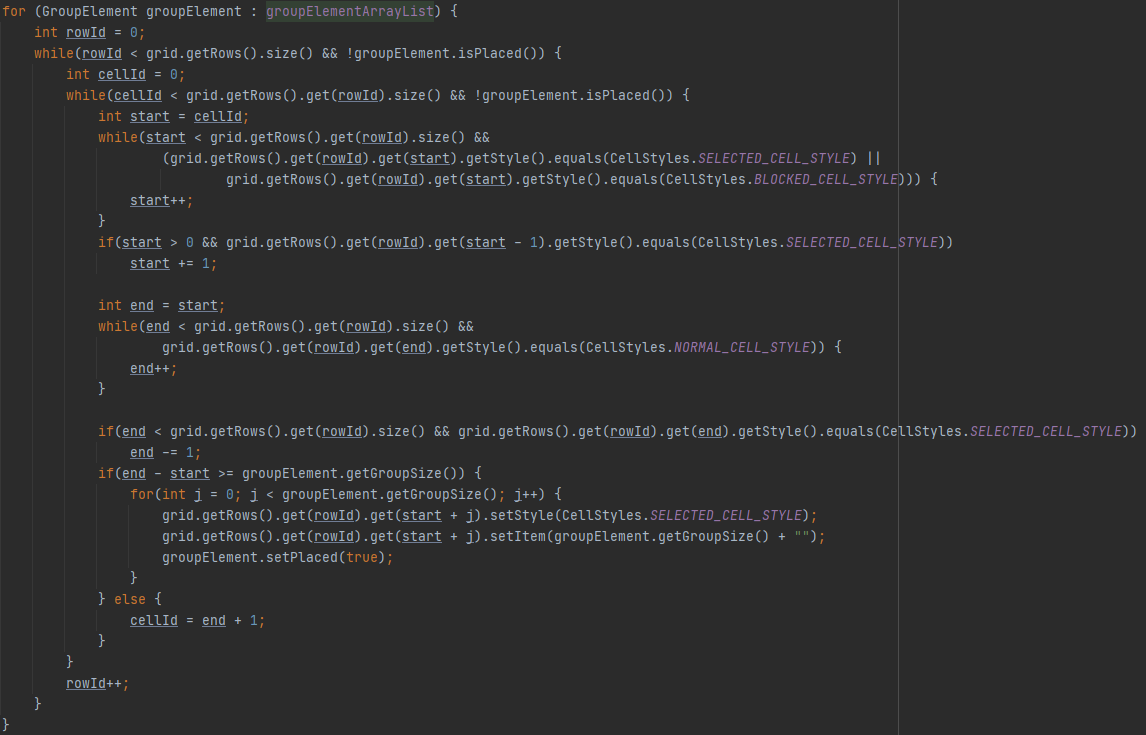
PA itt fejtsd ki a mongoDB-t hogyan használod, mi az adatstruktúra stb.

## Gépi megoldó algoritmus

### Mohó algoritmus

A gépi megoldókhoz implementálásra került egy mohó algoritmus, amely az előre meghatározott csoportokat el tudja helyezni a táblázatban.

Az algoritmus az első inputként megkapott csoporttól kezdődően végighalad az utolsó csoportig. Minden egyes csoportnál megkeresi az első olyan helyet, ahol el lehet helyezni a csoportot, majd erre a helyre elhelyezi a csoportot és tovább lép a következő csoportra.



**4. ábra:** mohó algoritmus megoldó kódja

A **4. ábra** szemlélteti az algoritmus működését. Látható, hogy a ***GroupElement***-ben eltárolt csoportokon megy végig az algoritmus. Sorfolytonosan végighalad az egész táblázaton a függvény keresve a legelső olyan helyet, ahová el lehet helyezni az aktuális csoportot. Amikor ezt a helyet megtalálja az algoritmus, akkor lefoglalja a csoportnak ezt a helyet és átállítja a ***placed*** adattagját a csoportnak ezzel jelezve, hogy sikerült leültetni a csoportot. A ***placed*** adattag igazra állítása után lép a következő csoportra az algoritmus vagy akkor, ha végig járta az algoritmus az egész táblázatot.