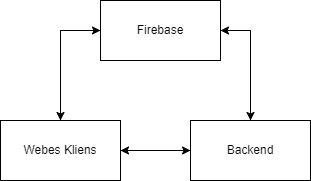
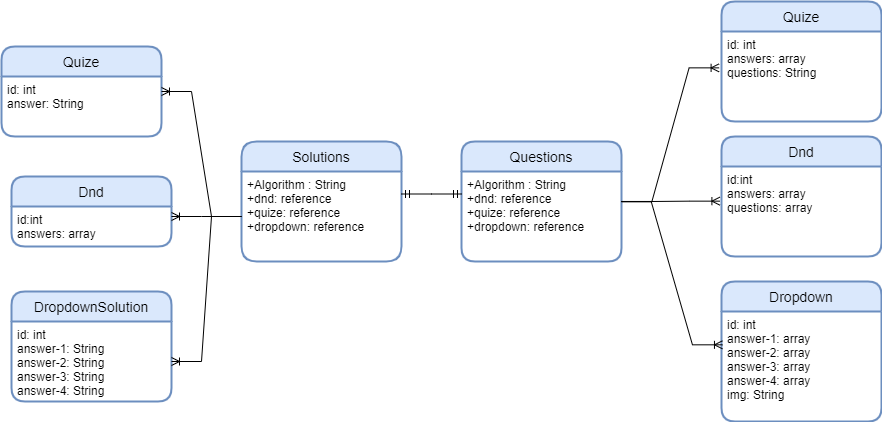
**Architektúra**

Megvalósítása egy 3 rétegű alkalmazás lesz, megjelenítési réteg React Js-el valósul meg amely egy könyvtár a JavaScripthez, REST API-on keresztül történik a kommunikáció a backenddel, amely egy Python keretrendszerben valósul meg amit FastAPI-nak hívnak, az adatok tárolását, bejelentkeztetést egy harmadik féllel valósul meg ez nem más mint a Firebase, itt NoSQL-t használok és regisztráció/bejelentkezést automatizálja.

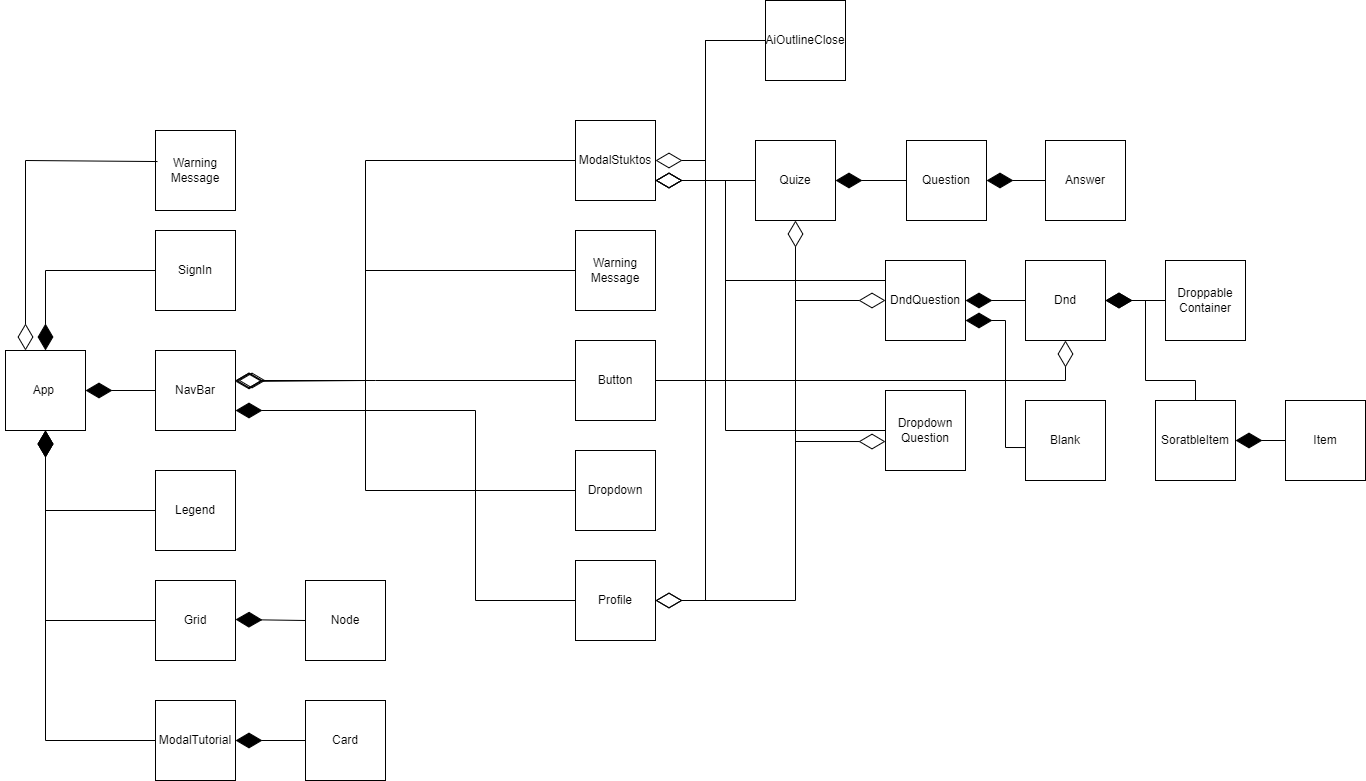
**A 3 réteg kapcsolata**

**Adatbázis táblái közöttük lévő kapcsolatok**:

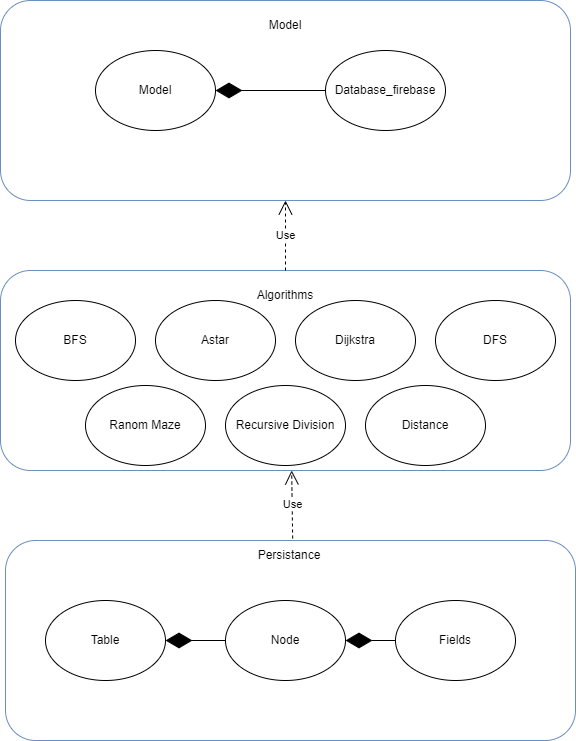


**Osztálydiagram osztályok tulajdonságaival és metódusaival bemenő adat, kimenő adat, tevékenység**

**Frontend:**

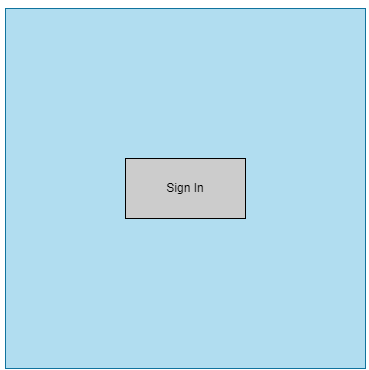


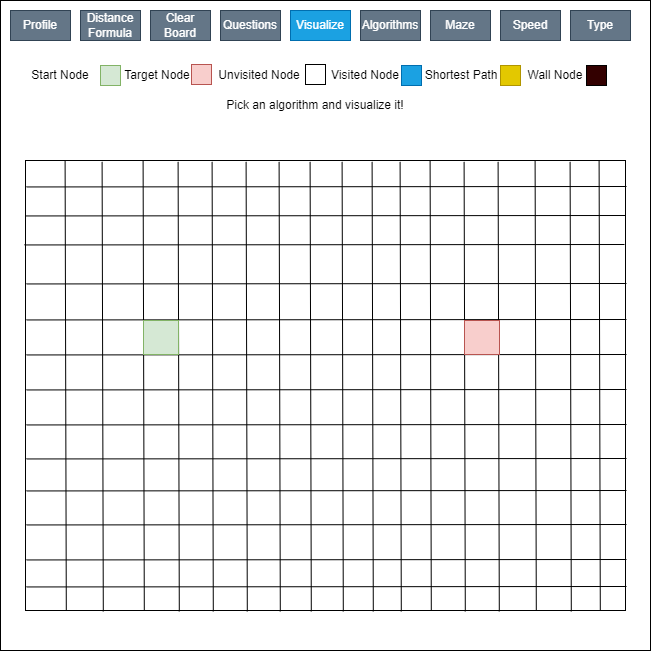
**Backend:**

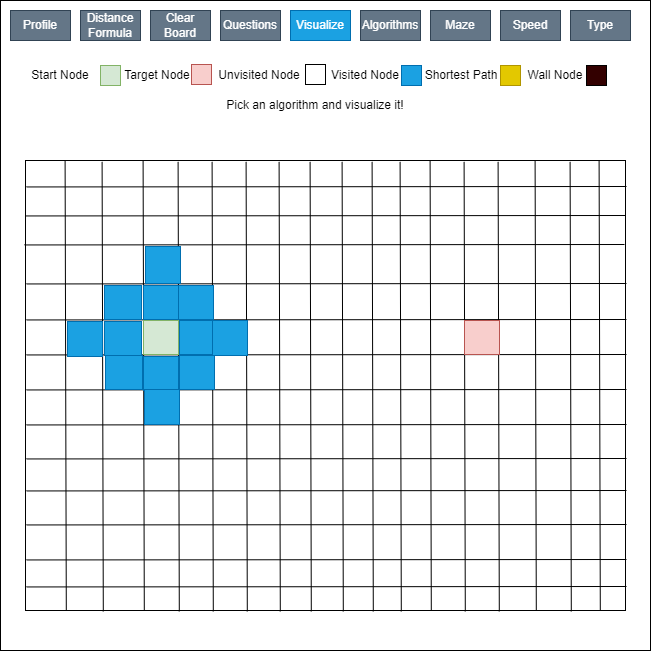
****

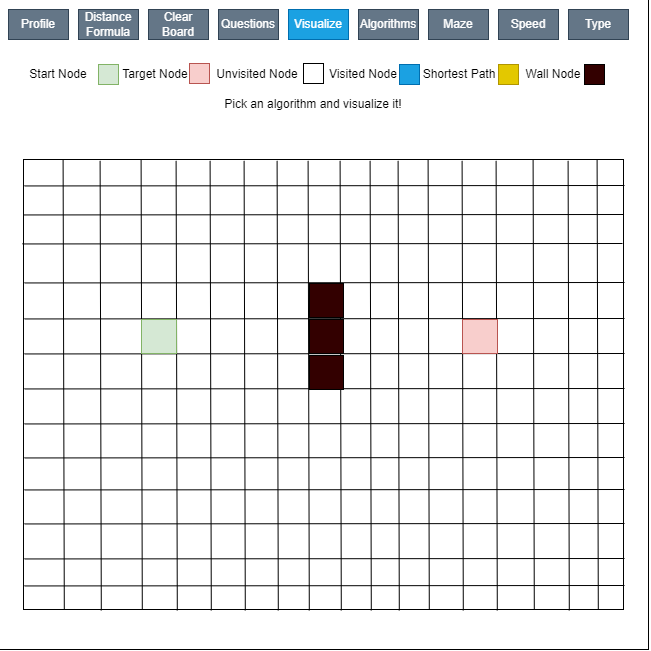
**Felhasználói felület képernyők terve**

Minden felhasználót Google-es authetikációval azonosítunk mely során nem regisztrált felhasználók automatikusan regisztrálásra kerülnek

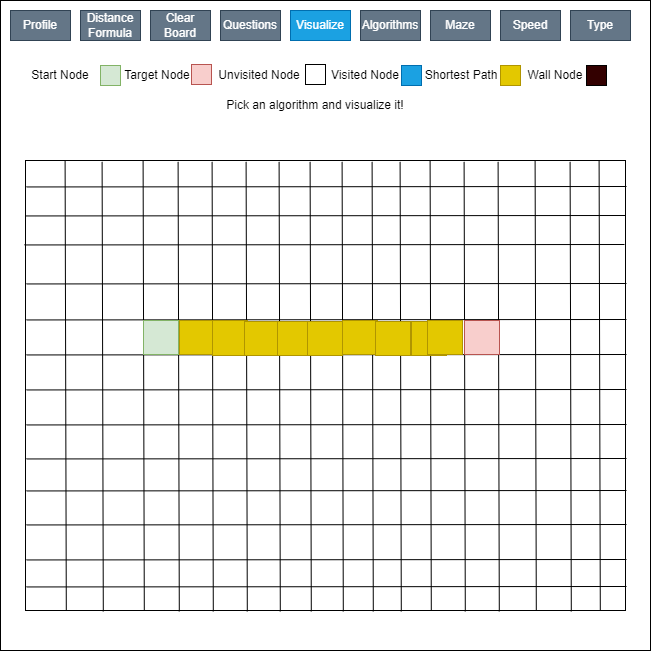
****

Azonosítás után a felhasználó a főoldalra kerül, ahol interakcióba tud lépni a táblával, navigációs menüpontokkal, minden algoritmushoz egy rövid leírás található meg amely jellemezi a tulajdonságait, végezhetünk mozgatás művelet a kiinduló és végállapottal.

Vizualizáció során kék színnel jelezzük a már megtekintett csúcsokat

Táblával végezhető műveletek közé tartozik a falak lehelyezése amelyek két csúcs közötti élt szakítanak meg

A legrövidebb utat sárga színnel jelöljük ennél rövidebb útvonalat az adott algoritmus nem tud jobb eredményt produkálni



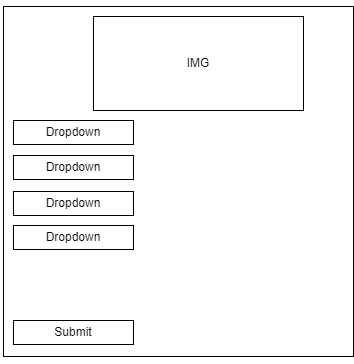
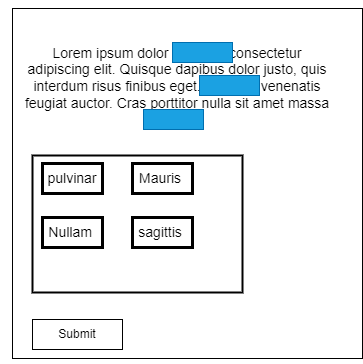
Minden algoritmushoz 3 kérdésfajta tartozik ezek az alábbiak:

Dropdown

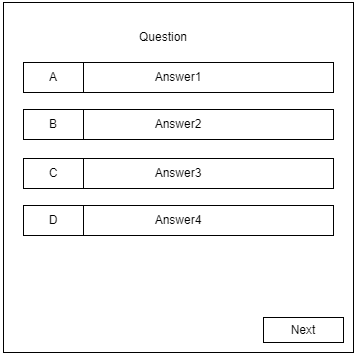
Dnd

Quize

Dropdown kérdés típusnál egy kép jelenik meg a felhasználónak és azt kell kiegészítenie a dropdownban felsorolt elemekkel

Dnd egy szövegből bizonyos elemek ki vannak hagyva, egy szóbankból kell kiválasztani és a megfelelő helyre húzni az oda illő szót

Question: Egy kérdés jelenik meg a felhasználónak 4 válaszlehetőséggel amelyek közül ki kell választani a megfelelőt



**Api endpontok felírása**

|  |  |  |
| --- | --- | --- |
| / | POST | Kezdőállapot, létrehozzuk a szükséges konfigurációkat pályát letisztítjuk, alaphelyzetbe állítjuk |
| /restartpoints | POST | Beállítjuk a felhasználó kérdésekre kapott pontjait 0-ára |
| /user | POST | Ha authentikált a felhasználó hozzáadjuk az adatbázisunkhoz, amennyiben már tartalmazza, nem történik változás |
| /quize /{algorithm} | POST | Az adott algoritmushoz tartozó kérdésre adott válaszoknak a validációja |
| /dnd /{algorithm} | POST | Az adott algoritmushoz tartozó kérdésre adott válaszoknak a validációja |
| /dropdown/{algorithm} | POST | Az adott algoritmushoz tartozó kérdésre adott válaszoknak a validációja |
| /wallUpdate | POST | Nézeten lehelyezett falak az üzleti logikába való elmentése |
| /changeDistance | POST | Heurisztikával rendelkező algoritmusokhoz tartozó heurisztika változtatása |
| /moveStartEnd | POST | Start és End pozíció elmozdítása jelzést küld az üzleti logikának |
| /clearForMaze | POST | Pálya letisztítása mielőtt elindítjuk a labirintus generálását |
| /Random | GET | 20% eséllyel falat helyez le az adott mezőre |
| /RecursiveDivision | GET | Labirintus generáló algoritmus, amely minden alkalommal egy eltérő labirintust generál |
| /Astar | GET | Astar megoldása az adott pályán |
| /DFS | GET | DFS megoldása az adott pályán |
| /Dijkstra | GET | Dijkstra megoldása az adott pályán |
| /BFS | GET | BFS megoldása az adott pályán |
| /getTypes | GET | Az algoritmusok elrejtik a súlyokat, ezeket minden futtatás előtt újra felrakjuk a pályára. |

**Tesztelés**

Frontend tesztelése egy end to end keretrendszerrel valósul meg amelyet Cypressnek neveznek itt emberi tevékenységeket szimulálunk így a frontendet és backend kommunikációját is ellenőrizni tudjuk

Backend tesztelése unit tesztek formájában valósul meg amely az algoritmusokat táblával való műveleteket ellenőrzi le

**Alkalmazott technológiák**

**React**

|  |  |
| --- | --- |
| **Komponens neve** | **Komponens leírása** |
| App | Kiinduló állapot magába foglalja a főbb komponenseket |
| SignIn | Authentikáció a felhasználóknak Firebasen keresztül |
| ModalTutorial | Felhasználók számára egy intuitív bevezetés a program használatába |
| NavBar | Tartalmazza a főbb menüpontokat amelyekkel interakciókat tudunk kialakítani |
| Profile | Felhaszánló profilja, elérhető a segítség kérés/kijelentkezés pontjai megtekinthetőek itt |
| DropDown | Egy legördülő listát tesz lehetővé újra felhasználhatóság szempontjából elengedhetetlen |
| Button | Egy gombot tesz elérhetővé újra felhasználhatóság szempontjából elengedhetetlen |
| Node | A pályánkon Node-ok helyezkednek el ennek a definiálása itt történik meg |
| Quize | Egy csomagoló komponens amely magába foglalja a „quize” kérdéseket |
| Question | Egy adott kérdést tartalmaz amelyet a „quize”-től kapunk |
| Answer | Minden kvízhez több válasz lehetőség tartozik azokat itt jelenítjük meg a felhasználó számára |
| DndQuestion | Mozgatás és elengedés során a megfelelő helyre kerülnek a válaszok |
| ModalStuktos ezt átnevezni |  |
| DropDownQuestions | Egy struktogrammot jelenítünk meg a felhasználó számára amelyhez legördülő menüben kérdések vannak |
| Dnd | Egy csomagoló komponens amely magába foglalja a főbb elemeket |
| Blank | Egy üres négyzetet tesz lehetővé amelybe belehelyezhetjük az adott válaszokat |
| DroppableContainer | Komponensunket olyan tulajdonsággal ruházzuk fel hogy adott elemet bele lehet helyezni na konténerbe |
| Item | Adott válasz |

|  |  |
| --- | --- |
| **Context neve** | **Context leírása** |
| GridContext | A pályánkon végezhető műveletek összessége, amely elérhető az összes komponens számára, vizualizáció, interakciók kezelése itt történik meg |
| QuestionContext | A kérdésszekcióban végzett műveletek egy megosztott contextben helyezkednek el hogy képesek legyenek kommunikálni egymással |

**Firebase**

A Firebase a Google által támogatott alkalmazásfejlesztő szoftver, amely lehetővé teszi a fejlesztők számára, hogy iOS, Android és webes alkalmazásokat fejlesszenek. A Firebase eszközöket biztosít az analitika nyomon követéséhez, az alkalmazások összeomlásának jelentéséhez és javításához, marketing- és termékkísérletek készítéséhez.

Firebase Storage és FireStore alkalmazásával élek amely NoSQL alapon valósul meg.

**FirebaseAuth**

Az autentikáció megvalósítására a Firebase SDK autentikációs modulját használtam. Ez megkönnyítette a fejlesztést és elősegítette a biztonságos bejelentkezési folyamat implementációját.

**Git és GitHub**

Az alkalmazás forráskódjának verzió kezeléséhez a Git alkalmazást választottam. A Git egy ingyenes, nyílt forráskódú elosztott verziókezelő rendszer. A verzió kezelésnek köszönhetően folyamatában lehet látni a program fejlődését, a kód módosítások összehasonlíthatók, és esetleges hiba eseten a korábbi verzióra való visszaállást is megkönnyíti. Ezen felül a Git repository-t a GitHub-ra is feltöltöttem így a felhőben van mentésem az alkalmazás kódjáról ezenfelül DevOps irányelvek érvényesültek feltöltés után automatikus tesztek futtatása és deployolása.

**Visual Studio Code**

A Visual Studio Code a Microsoft ingyenes nyílt forráskódú fejlesztői

környezete. A Visual Studio Code rengeteg hasznos integrációval rendelkezik melyek

megkönnyítette a fejlesztést ilyenek az automatikus formázó és kód kiegészítő berendezések

**FastApi**

A FastAPI egy modern, gyors (nagy teljesítményű), webes keretrendszer a Pythonhoz, Python típusokon alapuló API-k készítéséhez amely támogatja az aszinkron működést eltérően a más keretrendszerektől.

**Cypress**

A Cypress egy végponttól végpontig tartó tesztelési keretrendszer, webes tesztek automatizálásához. Lehetővé teszi a front-end fejlesztők és a tesztautomatizáló mérnökök számára, hogy automatizált webes teszteket írjanak JavaScriptben.

**PyUnit**

Az egységtesztelés egy olyan szoftverfejlesztési folyamat, amelyben az alkalmazás legkisebb tesztelhető részeit, az úgynevezett egységeket, egyenként és egymástól függetlenül vizsgálják meg a megfelelő működés szempontjából.