5. melléklet

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

DIPLOMAMUNKA

OE-NIK Hallgató neve: **Hajdu Renáta**

2022 Hallgató törzskönyvi száma: **T-005644/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve:	**Hajdu Renáta**
Törzskönyvi száma:	T/005644/FI12904/N
Neptun kódja:	[redacted]

A dolgozat címe:

Oktatójáték és BI rendszer fejlesztése futárszolgálatok számára
Educational game and BI system development for delivery services

Intézményi konzulens:	Sarkadi Katalin
Külső konzulens:	
Beadási határidő:	2022. május 15.
A záróvizsga tárgyai:	Számítógép architektúrák Big Data és üzleti intelligencia specializáció

A feladat

Tervezzen és valósítson meg egy, a futárok oktatását segítő, interaktív játékot, mely lefedi a kiszállítás során felmerülő problémákat. Tervezze és készítse el az ehhez tartozó adatbázist, valamint BI rendszert, amely képes az eredményeket vizualizálni. Vázolja a kiinduló helyzetet, majd kutassa fel és vizsgálja meg a megoldáshoz szükséges eszközöket, lehetőségeket. Az adatbázist úgy alakítsa ki, hogy a játék során szerzett pontok és a játékosok adatainak tárolására alkalmas legyen.

A dolgozatnak tartalmaznia kell:

- a feladat pontos ismertetését,
- a feladat megoldásához alkalmazható technológiák bemutatását,
- a választott megoldás indoklását,
- szükséges adatbázis struktúra megtervezését és megvalósítását,
- oktatójáték specifikálását és implementálását,
- BI alkalmazás megtervezését és megalkotását.

ÓBUDAI EGYETEM
Neumann János Informatikai Kar
1.
Budapest

.....
Dr. Fleiner Rita
intézetigazgató

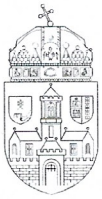
A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

7. melléklet



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. 05. 11.

Hajdu Renáta

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Hajdu Renáta

Neptun kód: [redacted]

Tagozat: Nappali

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Oktatójáték és BI rendszer fejlesztése futárszolgálatok számára

Szakdolgozat / Diplomamunka[2] címe angolul:

Educational game and BI system development for delivery services

Intézményi konzulens: Sarkadi Katalin

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.11.04.	Témamódosítás	
2.	2021.11.09.	Adatbázis részletezés	
3.	2021.12.03.	Játék részletezés	
4.	2021.12.09.	Javítások	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.12.09.

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Hajdu Renáta

Neptun kód: [redacted]

Tagozat: Nappali

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Oktatójáték és BI rendszer fejlesztése futárszolgálatok számára

Szakdolgozat / Diplomamunka[2] címe angolul:

Educational game and BI system development for delivery services

Intézményi konzulens: Sarkadi Katalin

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.03.23.	Haladás	
2.	2022.04.07.	Funkciók	
3.	2022.04.29.	Dokumentáció	
4.	2022.05.05.	Javítások	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022.05.09.

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

Tartalom

1. Bevezetés

Szakdolgozatom témája egy olyan oktatójáték elkészítése, amely egy futárszolgálat számára hasznosítható a futárjai oktatására és tudásuk felmérésére, valamint egy BI rendszer kialakítása, amely a játékosokat és a pontjaikat képes eltárolni, majd a cég vezetősége számára könnyen átlátható módon vizualizálni.

Én jelenleg egy futárszolgálatnál dolgozom, ahol a legnagyobb problémát a futárok alulképzettsége, és tudatlansága okozza, ez betudható annak, hogy nem olvassák el a sokoldalas dokumentációt a különböző eljárásokról és így nem várható el tőlük a szabályok intuitív elsajátítása. Ezekből kifolyólag jött az ötletem, hogy szakdolgozatom keretén belül, kialakítsak egy oktatójátékot, amely segíteni tudna ezen a nem túl ideális helyzeten.

Hiszen egy futárszolgálat nem tudna létezni futárok nélkül, akiknél pedig elengedhetetlen, hogy jól végezzék a munkájukat, annak érdekében, hogy a lehető legjobb ügyfélélményt sikerüljön elérni.

A színvonalas munkához azonban nem csak annyi szükséges, hogy felvegyék a rendelést és kiszállítsák, az ügyfelek akkor elégedettek igazán, ha a kiszállítás és az átadás módja is helyénvaló, a futárok nem viselkedhetnek kifogásolhatóan.
Kiszállítás során sajnos felmerülhetnek olyan problémák, amiket tudni kell megfelelően kezelni, például kiborult italok, lemaradt tételek.

Míg az autós és motoros futároknak feltétel a járművük használatához a jogosítvány megszerzése, amelyhez kötelező a KRESZ megtanulása és abból a vizsga letétele, kerékpár használatához nem szükséges semmilyen számonkért tudás, így a biciklis futárok KRESZ tudásának hiánya is problémákat szülhet. Arra a továbbiakban sem lehet biztosan számítani, hogy be is tartják az összes szabályt, de játékos keretek között nagyobb eséllyel tudnák elsajátítani a tananyagokat.

A hosszútávú visszakereshetőség és átláthatóság érdekében szükséges eltárolni adatbázisban a játék során szerzett eredményeket, az adatvizualizáció pedig jobb rálátást biztosít a futárok előrehaladásáról, segíti felderíteni a kritikus pontokat, fejlesztendő területeket a különféle igényes statisztikákkal és kimutatásokkal, hiszen a folyamatos fejlődés az alapvető cél.

2. Lehetséges megközelítési módok és megoldások áttekintése, elemzése

2.1. A játék tartalmi célja és a jelenleg elérhető alternatívák

Ha valaki futárnak jelentkezik a Wolthoz, oktatóvideókat kell végignéznie, utána csinálnia egy tesztet, ahol, ha nem szerez elegendő pontot, nem léphet tovább a jelentkezés folyamatában. Azonban ezt otthonról szükséges elvégezni, így erősen fennáll a csalás veszélye, aminek az a következménye, hogy mikor elkezd futárkodni az illető, könnyen előfordulhat, hogy nem fogja tudni, hogyan kell használni az eszközöket helyesen. Az ott szereplő videók, főleg a díjazási információkról és az alkalmazás használatáról szólnak, mintsem egyéb munka közben könnyen előkerülő problémákról, például hogyan kezeljük, ha útközben megrongálódott egy csomag vagy hogyan kommunikáljunk helyesen egy vevővel. [12]
A gamifikáció nagyban segítené a leendő futárok betanulását, valamint kiegészíthető lenne olyan tananyaggal is, mely jelenleg nem része a Wolt felkészítésének. [9]

A futárok KRESZ oktatásáról a Wolt nem gondoskodik, ezért ezt külső forrásból kell a leendő futároknak elsajátítani. Erre létezik sokféle KRESZ oktatójáték, melyek többnyire a forgalmi táblák felismerését célozzák meg, valamint található 3D-s autóvezető szimulátor is, mely egyéb közlekedési szabályokat is tartalmaz, azonban ez az autós jogosítvány megszerzésére készít fel. A talált játékok egyike sem foglalkozik a különböző szállítás során felmerülő nehézségek helyes kezelésének menetével, kifejezetten biciklisek számára sem találtam ilyen alkalmazást. [11] [5] [19]

Összegezve a tervezett oktatójátékom bizonyos részeivel kapcsolatos alkalmazás már létezik, azonban a szükséges összetettségű, mindent lefedő alkalmazásra nem találtam példát. Projektem hátterében egy olyan adatbázis áll, ahová a játék során elmentésre kerülnek a pontok, tárolja kérdés-válasz párosításokat, valamint a BI rendszer segítségével az eredmények vizualizációja is kivitelezhetővé válik.

2.2. Lehetséges fejlesztési nyelvek

Az oktatójáték programozásához a C#, Python, és C++ nyelvet vettem fontolóra, különböző játék fejlesztéssel kapcsolatos cikkek elolvasása után, amelyekből azt szűrtem le, hogy ezek szerepelnek a manapság leginkább játékfejlesztésre használt programozási nyelvek között. [8] [7] [6]

A C++ jelentős kontrolt tesz lehetővé a hardver és grafikus folyamatok fölött, valamint rendkívül szignifikáns optimalizációt lehet vele elérni. Túlnyomórészt nagy konzolos és Windows alapú játékok fejlesztésére használják, nagymértékű objektum orientáltsággal

rendelkezik, azonban az nagyobb szabású projektekhez ajánlott így továbbá a C# és a Python között gondolkoztam. [8] [7] [6]

A Python tanulhatóság szempontjából a legkönnyebb a másik két vizsgált nyelvhez képest, és egyszerűbb szintaktikája miatt könnyen áttekinthető. A Python egy interpretált nyelv, tehát a benne írt kód futtatásához külön futtató környezet szükséges, ezenkívül elegendő minimálisan specifikálni a változókat, osztályokat. Viszonylag gyorsan lehet vele játékot fejleszteni, és kevesebb kódolást igényel mire eljutunk egy kész állapotban lévő termékig, viszont gyakran nem a legcélszerűbb komplex programokhoz. Relatíve lassú játékfejlesztéshez, összeségében egyszerűbb, 2D-s játékokhoz kiváló, habár nem rendelkezik még akkora közösséggel, mint pl. a C#. Továbbá játékfejlesztő moduljai a Pyglet és a PyGame környezet, aminek segítségével könnyen és gyorsan lehet prototípusokat gyártani, modellezni. Példaként a Battlefield 2, Eve online Python nyelvben lett fejlesztve. [17]

A C# egy fordított nyelv, tehát az operációs rendszer közvetlenül képes futtatni a benne írt kódot, és nagyon fontos hozzá a változókat a lehető legpontosabban deklarálni, ami egy lényeges pont a játék fejlesztéshez. Az XNA keretrendszere tökéletessé teszi Windows és Xbox platformra való játékok fejlesztéséhez, megfelelő eszközök biztosításával, valamint a Unity játékmotor képessé teszi a bármilyen platformra történő játékfejlesztéshez, pl. iOS, Android. A Unity C# nyelvet és .NET-et használó, valós idejű 3D fejlesztői platform, 2D és 3D alkalmazások elkészítésére, lehetnek ezek játékok, szimulációk, C# nyelvet, és .NET-et használva. A .NET fejlesztői környezet egy hasznos eszköz, ami alkalmazható különböző nyelvekhez, ingyenes és platformfüggetlen, nyílt forráskódú, amelynek segítségével különböző alkalmazásokat, játékokat lehet készíteni. Mint azt említettem, a Unity is ezt a platformot veszi alapul. Játékok fejlesztése terén elég elterjedt a hatékonysága és jól skálázhatósága miatt. Példának okáért a Super Mario Run és Pokemon Go is ezzel készült. [6] [17] [22] [21] [23]

2.3. Vizualizációs eszközök

A BI rendszer kialakításhoz szükséges szóba jövő eszközöket az 1. ábra szemlélteti. A Gartner cég egy információs technológiai kutató és tanácsadó cég, az ábrán az általuk készített úgynevezett Magic Quadrant látható, amely egy olyan többrétű konkurenciakutatás, amely széles látószögű képet ad a piaci versenytársak egymáshoz viszonyított helyzetéről, vizsgálva többek között azt is, hogy mire képesek. [10]

1. ábra: Magic Quadrant [2]

Ez alapján összehasonlítottam a jelenlegi két piacvezető business intelligence eszközt, a Microsoft Power BI-t, és a Tableau-t, azokra a tulajdonságokra koncentrálva, amelyek a feladat megvalósításához a leglényegesebbnek találtam. Ezek során az alább leírtakat fedeztem fel. [3]

Elengedhetetlen tényező, hogy honnan fognak származni adataim, melyek alapján a vizualizáció történni fog, ehhez megvizsgáltam a két eszköz milyen forrásokkal tud dolgozni. Mindkét eszközzel számtalan adatbázishoz hozzá lehet férni például Microsoft SQL Server, Oracle, SAP HANA, MySQL Database. [16] [20]

Az ár sem volt utolsó szempont számomra, ez viszont jelentős összefüggésben áll a két eszköz különböző változataival. A Tableau Desktop az, amivel létre lehet hozni a különböző riportokat és dashboardokat, ezt a Tableau Creator tartalmazza, ami

70$/felhasználó/hónap, 14 napos ingyenes próbaidőszakkal ki lehet próbálni, majd Tableau Online, vagy Tableau Server segítségével lehet megosztani (előbbi az utóbbi hostolt verziója), amiből az egyik használatához járó licenszét, amit választunk az ár magában foglalja. Ezen kívül diákok igényelhetnek 1 éves ingyenes licenszet. A Power BI Desktop verziója ezzel szemben bárki számára ingyenes Windows operációs rendszerre, de ha felhő alapú riport megosztást szeretnénk, vagy együttműködést, azért már itt is fizetni kell. [15] [20]
Mindkettőnek létezik mobil verziója, amivel könnyen nyomon lehet követni a jelentéseket, bárhol is legyünk, valamint mindkettőt be lehet ágyazni más alkalmazásokba. [15] [20]

2.4. Adatbázis lehetőségek

2.4.1. SQL vs. NoSQL

A megfelelő adatbázis kiválasztásakor az első szempont amit megvizsgáltam, hogy SQL vagy NoSQL adatbázist válasszak. Ezt az alapján néztem, hogy milyen adatokat szeretnék eltárolni, ezek pl. felhasználókhoz tartozó pontok, kérdésekhez tartozó válaszok, pálya teljesítési idők, játékosok által kiválasztott válaszok azonosítói, tehát a hangsúly mindenképp az adatok közötti kapcsolaton van, és később majd a vizualizáció során is fontos lesz, hogy rugalmasan hozzáférhessünk az adatokhoz, és könnyen le tudjunk belőle kérdezni. Strukturált adatokról van szó, ezek mind olyan tényezők, amik alapján arra a következtetésre jutottam, hogy SQL adatbázist fogok használni a feladat megoldásához. [18]

A következő feladat az volt, hogy milyen SQL adatbázist válasszak, ehhez először megnéztem a jelenleg piacon lévő legjobbnak értékelt eszközöket, és az alapján az Microsoft SQL Server adatbázis kezelő rendszert, és az Oracle RDBMS-t vettem jobban szemügyre. [3] [4] [1]

2.4.2. MS SQL vs. Oracle

Tempó szempontból az MS SQL szerver jobb optimalizálóval rendelkezik, ami gyorsabbá teszi a kódot. Számomra az egyik nagyon fontos szempont a játék fejlesztése érdekében, hogy hogyan tud együttműködni az adatbázis a környezettel, amiben kódolni fogok, és az MS SQL sokkal jobb .NET integrációval rendelkezik, mint az Oracle. [13]
Míg az MS SQL használatánál auto increment jelöléssel ellátott oszlopoknál az új sorok automatikusan számozódnak, az Oracle ennek megfelelő megoldása a sequence, azonban, ezutóbbinál, nincsen közvetlen kapcsolat a sequence és a tábla, vagy oszlop között, ez egy objektum, mint egy tábla vagy egy tárolt eljárás. Az Oracle célszerűbb nagyobb, és összetettebb alkalmazásokhoz, míg az MS SQL megfelel kisebb, kevésbé komplexekhez is. [13]

Alapvető funkcióikban megegyeznek, nincs különösebben olyan dolog, amit az egyik meg tud csinálni, a másik pedig nem képes rá, a fő különbség a két adatbázis-kezelő rendszer között inkább a mögöttes architektúrájukban és szintaxisaikban nyilvánul meg. [13]

3. Megoldási módszer kiválasztása, a választás indoklása

A projektem fontos szempontja, hogy a vizsgált eszközöknek, nyelveknek, környezeteknek egy olyan kombinációját válasszam, amelyek hatékonyan tudnak együttműködni és ezáltal a munka is produktív legyen.
A játék programozásához a C# nyelvet választom, egyrészt a széleskörű támogatottság, az említett, feladatom szempontjából jól hasznosítható könyvtárak, és keretrendszere miatt, továbbá egyéni preferencia alapján.
Az adatok eltárolását Microsoft SQL szerver adatbázisban fogom megvalósítani, mivel mint kutatásom során kiderült, lényegi funkcióikban nem különböznek az Oracle-től, viszont ennek a .NET keretrendszerrel történő együttműködése számottevően hatékonyabb.
Adatvizualizációt tekintve pedig összefoglalva a két vizsgált eszköz nagyjából azonos szinten van, Power BI hétköznapi felhasználók számára is könnyen kezelhető, míg a Tableau inkább nagyvállalati környezetekre összpontosít, ahol adatmérnökök dolgoznak és nagyobb a költségvetésük, így a Power BI-t választom melybe SQL szerver adatbázisból könnyen beimportálhatók az adatok, és az automatizálásra is lehetőség van.

4. Követelményspecifikáció

4.1. Áttekintés

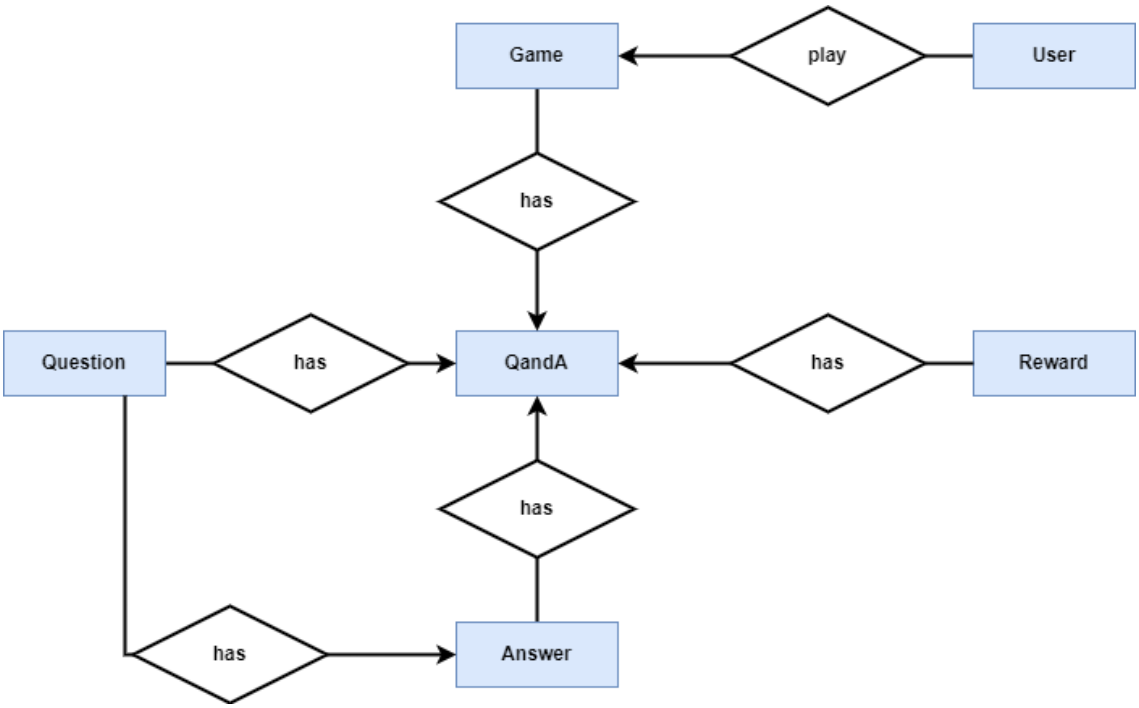
A feladatom során elkészítendő rendszer célja az, hogy egy futárszolgálat fejlődni tudjon, és egyre jobbá válni, azáltal, hogy adatai futárjaik ki vannak képezve, és így minőségi kiszállítást tudnak biztosítani az ügyfeleknek, és a vállalat vezetése egyszerűen rálát a tudás szintjükre, ezzel tudja, hol szükséges fejleszteni valamit. Tehát a projekttel szembeni követelményeimet ez alapján három kategóriára osztom, és így tárgyalom alább. Az adatbázissal kapcsolatos követelmények, az oktatójátékkal kapcsolatos követelmények, valamint az adat vizualizációval kapcsolatos elvárások.

4.2. Adatokkal szembeni követelmények

A játék során előforduló kérdéseket, és válaszokat szükséges eltárolni, valamint az egyes játékok eredményeit a hozzá tartozó futárok adataival együtt, ehhez az alábbi egyedhalmazokat tartom szükségesnek létrehozni.

- User: Ez fogja tartalmazni a felhasználókkal kapcsolatos adatokat, mint felhasználó neve, email címe stb.
- Game: Itt az aktuális játék adatai kerülnek majd eltárolásra, azaz, amikor elkezd valaki játszani, el fog tárolódni az azonosítója, hogy mikor kezdte a játékot, a játék végén a befejezési idő, a teljes játék során szerzett pontok, valamint a kezdéskor választott jármű.
- Reward: Itt tárolódnak a játékok során szerezhető jutalmak.
- QandA: Minden lejátszott kvízről tárolja teljesítési idejét, a szerzett jutalmat és pontokat. Lentebb a játék részletezésénél tárgyalom bővebben, az éttermi rendelés felvételkor, rendelőnél a leadásnál, és a KRESZ témakörnél lesz ilyen feladat.
- Takes: Részletezi a kvíz során jelölt válaszokat. Az kerül ide, hogy ki melyik kérdésre melyik választ adta.
- Question: Ez a tábla a kérdések helye. Minden kérdés el lesz látva egy egyedi azonosítóval.
- Answer: Itt tárolódnak a válaszok, ezenkívül az, hogy melyik kérdéshez tartoznak, mint válaszlehetőség, ez fogja biztosítani, hogy a feladatoknál az összepasszoló kérdés-válaszok töltődjenek be. Emellett még egy arra vonatkozó attribútummal is rendelkezni fog, hogy jó vagy rossz az értéke, ez lesz az alapja a pontszámításnak is, a jó válasz 1 pontot ér, a rossz 0-át.

Ezek alapján ábrázoltam a kapcsolatokat a 2. ábrán látható EK diagramban, az adatbázis struktúrájának szemléltetésére.



2. ábra: EK diagram

Így pedig ezen alapulóan részletezem az egyes adattáblákat, benne szereplő attribútumokat a tulajdonságaikkal.

4.2.1. Game tábla

Oszlop név	Leírás	Típus
GameID - elsődleges kulcs	játék egyedi azonosítója	int
UserID - idegen kulcs a User táblára	a játékos azonosítója	int
StartTime	játék kezdés ideje	datetime
EndTime	játék befejezés ideje	datetime
Vehicle	választott jármű	varchar
Points	játék során szerzett pontok	int

4.2.2. Reward tábla

Oszlop név	Leírás	Típus
RewardID - elsődleges kulcs	jutalom egyedi azonosítója	int
RewardName	jutalom neve	varchar

4.2.3. User tábla

Oszlop név	Leírás	Típus
UserID - elsődleges kulcs	játékos egyedi azonosítója	int
UserName	játékos neve	varchar
Email	játékos e-mail címe	varchar
Phonenumber	játékos telefonszáma	varchar
Town	város, ahol dolgozni fog	varchar

4.2.4. QandA tábla

Oszlop név	Leírás	Típus
QAID - elsődleges kulcs	kérdezz-felelek egyedi azonosítója	int
GameID - idegen kulcs a Game táblára	hozzá tartozó játék azonosítója	int
RewardID – idegen kulcs a Reward táblára	feladatra kapott jutalom azonosítója	int
Score	a kvíz során szerzett pontok	int
TimeToFinish	mennyi idő alatt sikerült teljesíteni	time

4.2.5. Takes tábla

Oszlop név	Leírás	Típus
TakeID - elsődleges kulcs	választás azonosítója	int
QAID - idegen kulcs a QandA táblára	hozzá tartozó kvíz azonosítója	int
QuestionID – idegen kulcs a Question táblára	aktuális kérdés azonosítója	int
AnswerID – idegen kulcs az Answer táblára	játékos által választott válasz azonosítója	int

4.2.6. Question tábla

Oszlop név	Leírás	Típus
QuestionID - elsődleges kulcs	kérdés egyedi azonosítója	int
Question	a konkrét kérdés	varchar
QuestionAuthor	a kérdés szerzője	varchar
QuestionCreationTime	hozzáadás ideje	datetime
Category	melyik témakörhöz kapcsolódik a kérdés	char

4.2.7. Answer tábla

Oszlop név	Leírás	Típus
AnwerID - elsődleges kulcs	kérdezz-felelek játék egyedi azonosítója	int
QuestionID – idegen kulcs a Question táblára	melyik kérdéshez tartozik a válasz	int
Answer	a konkrét válasz	varchar
WrongOrRight	jó vagy rossz a válasz	bit
AnswerAuthor	a válasz szerzője	varchar
AnwerCreationTime	hozzáadás ideje	datetime

.

4.3. Követelmények a BI rendszer felé

Az elkészítendő BI rendszer célja, hogy a cég vezetői az adatok vizualizálásával mindenkor könnyen áttekinthetően megfelelő rálátást kapjanak a futár jelöltek/ futárok teljesítményére. Ha valaki pontjai alapján azt látják, hogy nagyon rosszul teljesít, meg tudják hozni a megfelelő döntést a felvételéről, továbbképzéséről.
Tehát mindenképp fontos kimutatást készíteni futáronként a szerzett pontokról, kérdésenként a szerzett pontokról, tehát melyik az a kérdés, ami a legnagyobb nehézséget okozza, valamint játékonként a választott járművek számáról, annak érdekében, hogy lássák, mit részesítenek előnyben.

Ehhez a Power BI az MS SQL adatbázisból fogja megkapni az adatokat.

4.4 Oktatójátékkal kapcsolatos követelmények

4.4.1 Általános

Az oktatójáték a futárok oktatásának és tudásuk felmérésének a célját szolgálja, azt feltételezve, hogy már elolvasták a kiadott tájékoztatókat, de még nem sikerült teljes mértékben elsajátítani a tananyagot, és mindezt egy játékos, élvezhető formában, személyes keretek között az irodában. Mindenképpen szükséges valamilyen időmérés a játék során, hiszen a reakcióidő egy fontos tényező. Alább tárgyalom a lefedendő témakörökkel kapcsolatos követelményeket.

Maga a játék során egy választott járművel közlekedhetnek egy városban, mindig egy adott cél felé, ahova érve szituációs feladatok fogják várni őket, vagy valamilyen mini játék, bővebben a lentebb látható game design documentben részletezem.

4.4.2 Követelmények az érintendő témákról

Most pedig ismertetem a kategóriákhoz tartozó kritikus pontokat, amiket feltétlen tartalmaznia kell a játéknak.

<u>Ügyfelekkel kapcsolatos problémák</u>

Egyik nagyon fontos, és gyakran előforduló témakör, aminek szerepelnie kell, mint feladat, a vevőkkel való helyes kommunikáció, ugyanis hiába van az embernek rossz napja, egyáltalán nem elfogadott dolog, hogy lekezelő, udvariatlan hangnemben és módon kommunikáljanak az ügyfelekkel, különös tekintettel arra a nem elfeledendő tényre, hogy fizetésük a vevők rendeléseinek árából képződik.

A rendelések leadása kapcsán felmerülő problémákról, mint kiömlött üdítő, leves, megrongálódott dobozok, minden esetben szükséges fotót készíteni, ugyanis az szolgál bizonyítékul, amit az étterem számára továbbítani lehet. Így láthatják, hogy a balesetet a nem megfelelő csomagolás okozta, és a jövőben fejleszteni tudják ezt. Sajnos erről is nap mint nap megfeledkeznek.

Szintén sokat ront az ügyfél élményen, személyes tapasztalat alapján, amikor az alkalmazásban az látható, hogy megérkezett a rendelésünk, ennek ellenére a futár még nem tartózkodik a környéken, tehát hamarabb állította át a rendelés állapotát átadva helyzetbe, minthogy ez valóban megtörtént volna.

Következő sarkalatos és roppant kényes téma, amikor személyes okok miatt egy vevő úgy adja le a rendelését, hogy átadásnál nem szeretne érintkezni a futárral, hanem azt kéri, hogy helyezze el a csomagot az ajtaja előtt, és ezt figyelmen kívül hagyja a futár.

További nemkívánatos problémákat idéz elő az a gondatlanság, hogy az átadás során nem ellenőrzik le, hogy valóban a klienshez tartozó rendelést adják át számára, tehát a zacskón lévő rendelésszámot és nevet nem olvassák el a nagy sietség miatt, pedig ezzel nagyon sok további kellemetlenségtől meg tudnák óvni nemcsak a vevőt, hanem saját magukat is.

Ehhez szorosan kapcsolódik az a komplikáció, amikor nem veszik tekintetbe, hogy tartozik-e üdítő vagy esetleg egy zacskón kívül másik tétel a rendeléshez, az üzletek és éttermek ugyanis gyakran külön csomagolnak tételeket, ha nem fér bele egy szatyorba, vagy a különböző hőmérséklet miatt nem célszerű egybecsomagolni, tehát átadáskor erre szintén fokozottan szükséges figyelniük.

Éttermekkel kapcsolatos problémák

Az ügyfeleknél sorolt problémák nagyon hasonló változatai tapasztalhatók meg az éttermekkel való kommunikációban és a rendelések kezelésénél is. Itt is egyaránt elvárhatónak kéne lennie az illedelmes kommunikációnak, hiszen az étterem is egy egyenrangú partnere a futárszolgálatnak, akik szintén mindent beleadva dolgoznak a bevételükért, hogy legyen mit kiszállítani a futár partnereknek. Udvariatlan, tisztességtelen viselkedés esetén megjegyzik az illetőt, és akár az is előfordulhat, hogy többet nem engedik be a helyre, ami a futárnak sem előnyös.

A rendelésekkel kapcsolatos figyelmetlenségek is hasonlóképp gyakran fellelhető jelenség, tehát mindenképp létfontosságú a zacskón szereplő név és rendelésszám megvizsgálása mielőtt átveszik a csomagot kiszállításra, ugyanis jelentősen komplikált fennakadást tud okozni, ha egy másik futár elől viszik el a csomagot, amihez eltérő cím tartozik, sőt az is megeshet, hogy másik futárcéghez tartozó rendelést visznek el.

Szintén nélkülözhetetlen arra figyelni, hogy minden tétel át legyen véve, ne maradjon le üdítő vagy a nagy rendeléseknél a második zacskó.

Közlekedési ismeretek

Egy másik elhanyagolhatatlan kritikus pont a futároknál a közlekedési szabályok hiányában mutatkozik meg. Autós és motoros futároknál ez a hiányosság sokkal kevésbé áll fent, mint bicikliseknél, hiszen nekik jogosítvány megszerzésére volt szükségük ahhoz, hogy vezethessék járművüket, azonban naponta látni olyan biciklis futárokat, akik

járdán közlekednek olyan helyen, ahol mellette van a kerékpárút vagy a megengedettel ellentétes irányban közlekednek, mert még a forgalmi táblák jelentésével sincsenek tisztában, és piros lámpán áthajtókat is számtalanszor fel lehet fedezni. Így ennek a lényegében KRESZ témakörnek a lefedésére is szolgálna egy kategória a játékban.

Általános felkészültség, eszközhasználat

Rendszeresen elfelejtik a futárok feltenni töltőre a telefonjukat, mielőtt elindulnak a műszakjukra, ez pedig nagyon megnehezíti, konkrétan ellehetetleníti a munkát, hiszen a mobil szükséges a vásárló eléréséhez, problémák, mint pl. defekt esetén segítség kéréséhez, és magának a futár alkalmazás használatához is, így pedig a műszakból kieső futárok miatt, nagyobb lesz a terhelés a többin, megnövelheti a kiszállítási időt is. Szintén problémás a powerbank feltöltésének elmulasztása, vagy annak elpakolása, mert hiába létezik, mit sem ér, ha nincsen feltöltve.
Valamint értelmét veszti az egész, ha nincsen hozzá kábel, ami az összeköttetést biztosítaná a mobillal.

Nagyon sok kellemetlenségtől meg tudnák óvni magukat, ha a melegen tartó táskát helyesen pakolnák be, ezzel ki lehetne küszöbölni azt, hogy kiborulnak bizonyos rendeléshez tartozó tételek, ezzel eláztatva mást mire célba érnek, és így kevesebb lenne a vevőktől beérkező panasz is.

5. Game Design Document

5.1 Concept

A játék célja, hogy a futárokat oktassa, szórakoztató, játékos formában, amely egy kiszállítási folyamatot szimulál, single-player módban, a felmerülő problémákat magába foglalva.

5.2 Karakterek

A játék kezdete előtt választhat a játékos, hogy milyen járművel szeretne játszani, az ennek megfelelő jármű jelenne meg a városban, ezzel közlekedne:

- autó
- motor
- bicikli

Amikor betér egy belső helyszínre egy ember figura váltja fel a járművet, amikor pedig elhagyja azt, visszavált a korábbi járműre.

5.3 Helyszínek

A játék 4 különböző helyszínen fog játszódni:
A város, ahol a futár közlekedni fog felülnézetből lesz látható, itt tud mindig haladni az aktuális célja felé.
Az étterem, ahol felveszi a rendelést, szintén felülnézetből fog látszódni és ugyanígy a lakása és a vevő háza is.

Ezenkívül a kérdezz-felelek típusú feladatok oldalnézetből lesznek láthatóak, mintha két ember beszélgetne, kivéve a KRESZ témakörös, ugyanis ott közlekedési táblákat fog látni a játékos, és ahhoz fognak kérdések tartozni. Az elején szemből nézve fog látni a játékos polcokat, amin össze kell kattintani a szükséges eszközöket, az étteremnél pedig egy táskát fog belülről látni, amiben a helyére kell húzni a tételeket.

5.4 Kontrol

A járművet a jobb, bal, fel, le nyilakkal lehet irányítani a billentyűzeten, a benti helyszínen lévő karaktert szintén. Az ajtókra az egér bal gombjával lehet ráklikkelni, úgy lehet ki-, vagy bejutni, párbeszéd folytatásához az étteremben lévő pultra, és a vevőre szintén.
A kérdezz-felelek játékoknál is bal klikkel lehet kiválasztani a lehetőségekből a szimpatikus választ.

Az első mini játéknál hasonlóképp kell rákattintani a szükséges eszközökre, a második táska bepakolós játéknál szintén, azonban itt drag and drop módszerrel szükséges behúzni a megfelelő helyre a tételeket.

5.5 Gameplay

Egy játékmenet az itt leírtak szerint fog zajlani.

Miután kiválasztotta a járművét a játékos, elindul az időzítő és az első hely, ahol találni fogja magát, egy lakás lesz, mintha a saját lakása lenne, ahol, ha elindul az ajtó felé, és rákattint, fel fog ugrani egy minijáték azután érdeklődve, hogy bepakolt-e mindent. Itt látni fog több felrajzolt eszközt is, mint pl. banán, powerbank, töltőkábel stb., és ki kell választania a munkához mindenképp szükségeseket. Ha sikerült mindet jól kiválasztania, akkor, és csak akkor indulhat is útjára, ez ugyanis az első legkritikusabb pont.

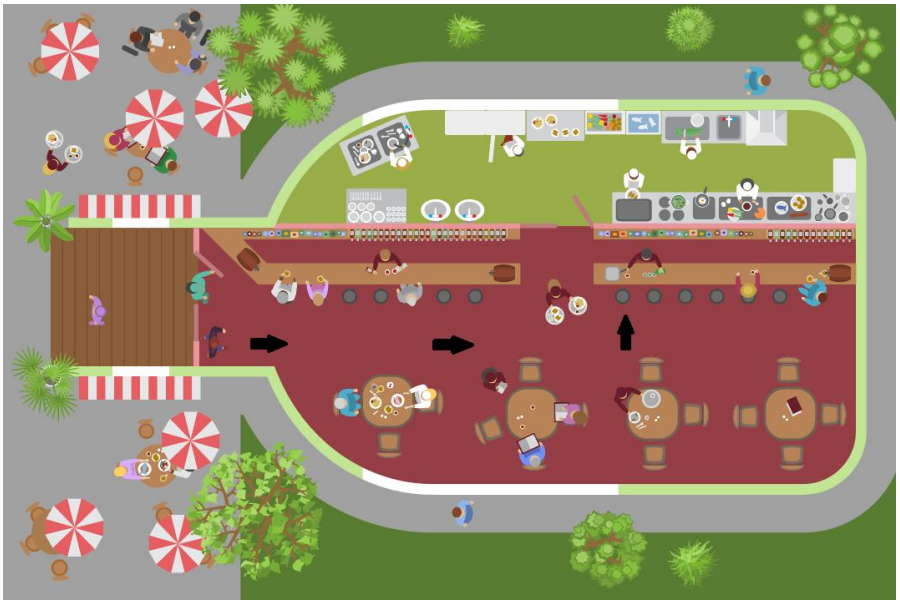
Következő részben egy utcán lesz a választott járművel, ahol nyilak fogják mutatni a helyes irányt, amerre haladnia kell, a célpont az étterem lesz. Ha rossz irányba megy, és falnak ütközik, egy figyelmeztető üzenetet fog látni, hogy nem arra van a cél. Ezen a részen két verzió lesz megvalósítva, ha autós vagy motoros futárról van szó, akkor amint sikerült elérni az éttermet, be is tudnak menni, azonban, ha biciklis futár az illető, akkor az útja során egyszer csak pop-up jelleggel meg fog jelenni a KRESZ játék, ahol közlekedési táblákról kell kérdésekre kiválasztania a megfelelő választ biciklis KRESZ-re vonatkozóan. Ha az összes helyes választ elsőre találja el, jutalom várja, ha valahol rosszra kattint, akkor látni fogja a jó választ, és mehet tovább a következő kérdésre.
Amint odaért az étteremhez, rákattintva az ajtóra, be tud lépni, és bent, szintén nyilak fogják mutatni, hova kell mennie, a cél a pult lesz. A pultra kattintva egy új ablakban párbeszédet kell lefolytatnia a pultossal, ezt olyan formában, mint a KRESZ feladatnál, hogy kérdésekre kell kiválasztania a megfelelő választ, jutalom üti a markát ennél a feladatnál is, ha mindegyiket elsőre eltalálja. Amint végigjutott a kérdéseken, rögtön meg fog jelenni egy táska formájában egy minijáték, ahol drag and drop módszerrel be kell húzni a megfelelő helyre az ételeket, üdítőket, tehát a cél, hogy helyesen pakolja be a táskát, és ne boruljon fel benne semmi. Ez szintén egy olyan kritikus pont, ami nélkül nem indulhat tovább, szóval addig kell játszania, amíg nem sikerül.

Ezután kisétálva az étteremből a következő célja a vevő címe lesz, ez jelölve lesz városban. Amint sikerült oda eljutni, az ajtóra kattintva bejut a házba, ahol a vásárlóra kattintva átadhatja neki a csomagot, és szintén kell egy párbeszédet lefolytatnia vele, amelyben különböző szituációkat fog kapni, többek között kiborult az üdítő és dühös a vásárló, lemaradt egy csomag és hasonlók, ezt is a helyes válaszok kiválasztásával hajthatja végre. Ha ezt is sikeresen teljesítette, és elsőre eltalálta a jó válaszokat, itt is jutalmat kap, ezután hazamehet, és amint visszaért a házához, az ajtóra kattintva, újra a

lakásában találja magát, a játék véget ér, az időzítő leáll. Felugrik egy ablak, ahol megtekintheti a szerzett pontjait, jutalmait és hogy mennyi idő alatt teljesítette a játékot. Ezeket az adatokat később bármikor megnézheti a profilja alatt. Az exit gombbal ki fog tudni lépni, és visszajut a főmenübe.

Az alábbi 3., 4. és 5. ábrán láthatóak tervek hozzá:

3. ábra: Lakás [24]



4. ábra: Étterem [24]

5. ábra: Város [24]

5.6 Pontozás, jutalomrendszer

Minden kérdezz-felelek játéknál (éttermes párbeszéd, vásárlói párbeszéd, KRESZ játék) ugyanaz a pontszámítás módszere, minden helyesen megválaszolt kérdés 1 pontot ér, ha rosszat választ (0 pont) a program megmutatja a játékos számára, hogy mi lett volna a jó és mehet a következő kérdésre, negatív pont nincs.

Az említett jutalmakat a pontok mellett, attól függetlenül fogják kapni a játékosok, amikor elsőre sikerül eltalálni az összes jó választ. A jutalmak random ételek, állatok lesznek, mint például cukorka, kávé, sajt, mókus, alpaka, tehát könnyen előfordulhat, hogy két futár nem ugyanazt kapja.

5.7 Leaderboard

A főmenüből elérhető leaderboard első sorban pontszám szerint csökkenő sorrendben, másodsorban teljesítési idő szerint növekvő sorrendben fogja tartalmazni a játékosok által elért eredményeket, külön a bicikliseknek, és külön a motorosok és autósok számára.

5.8 Menürendszer

A mellékletben szereplő 6. ábrán látható use case diagrammal szemléltetem a játék funkcióit.

Egy játékos kezdésnek egy bejelentkező ablakot lát, ahova be kell írnia az e-mail címét, ha elírja egy hibaüzenetet fog látni, és újra megpróbálhatja beírni, ha viszont sikeres volt a bejelentkezés, akkor látni fog egy köszöntő üzenetet, majd tovább lépve elétárul a főmenü.

Itt öt menüpont közül választhat. Ha valamilyen indokból kifolyólag meggondolta magát, akkor ki is léphet.
A legfelső gombbal tovább léphet a jármű választásra, ahol kiválaszthatja, hogy mivel szeretne játszani, majd indulhat is maga a játék.

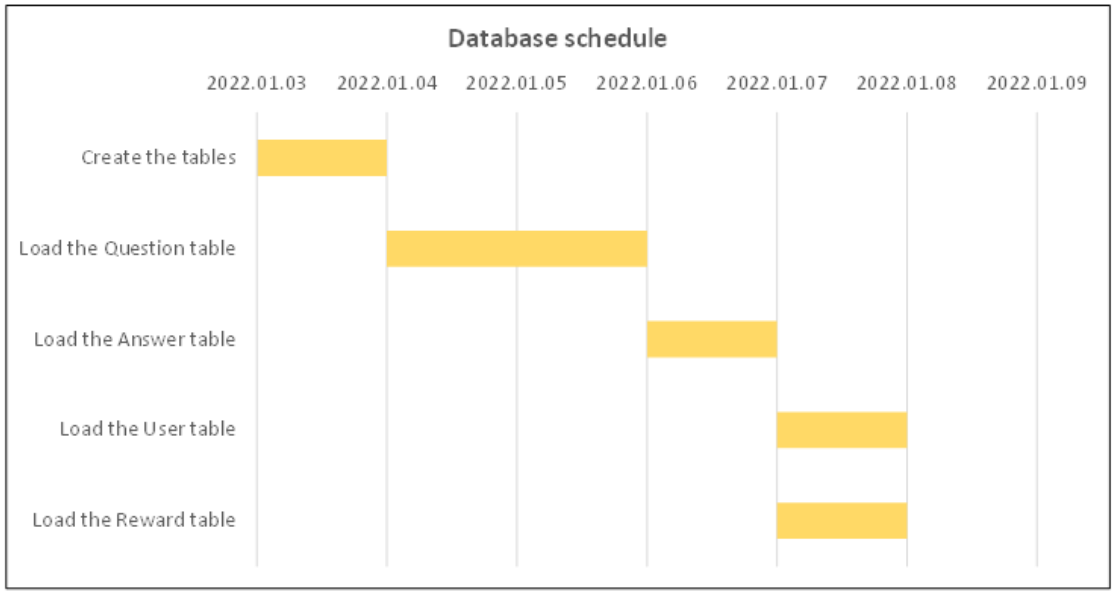
Az alatta lévő gombra kattintva megtekintheti a rangsort, ahol, ha már ő is játszott, a saját eredménye is szerepel.
Lentebb elérheti a játék céljáról, irányításáról szóló leírást.
Megnyithatja a profilját is, ahol meg tudja nézni az eddig lejátszott játékai alatt összegyűjtött jutalmait, pontjait, és a hozzá tartozó teljesítési időket, valamint, ha szeretne szerkeszteni valamit az adatain, arra is van lehetősége.

A mellékletben lévő 7. és 8. ábrán látható Wireframe diagram ábrázolja a leírtakat.

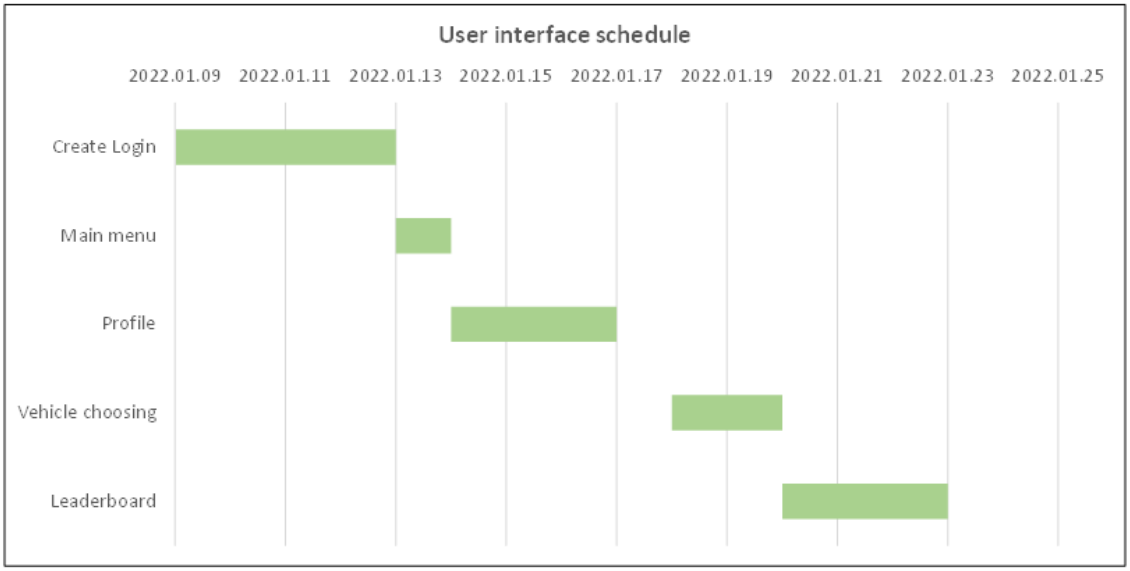
6. Ütemezés

A mellékletben szereplő 9. ábrán látható a projektre készült ütemtervem.

Valamint ebből az első két fázis további ütemezése látható a 10. és 11. ábrán.



10. ábra: Ütemezés - database

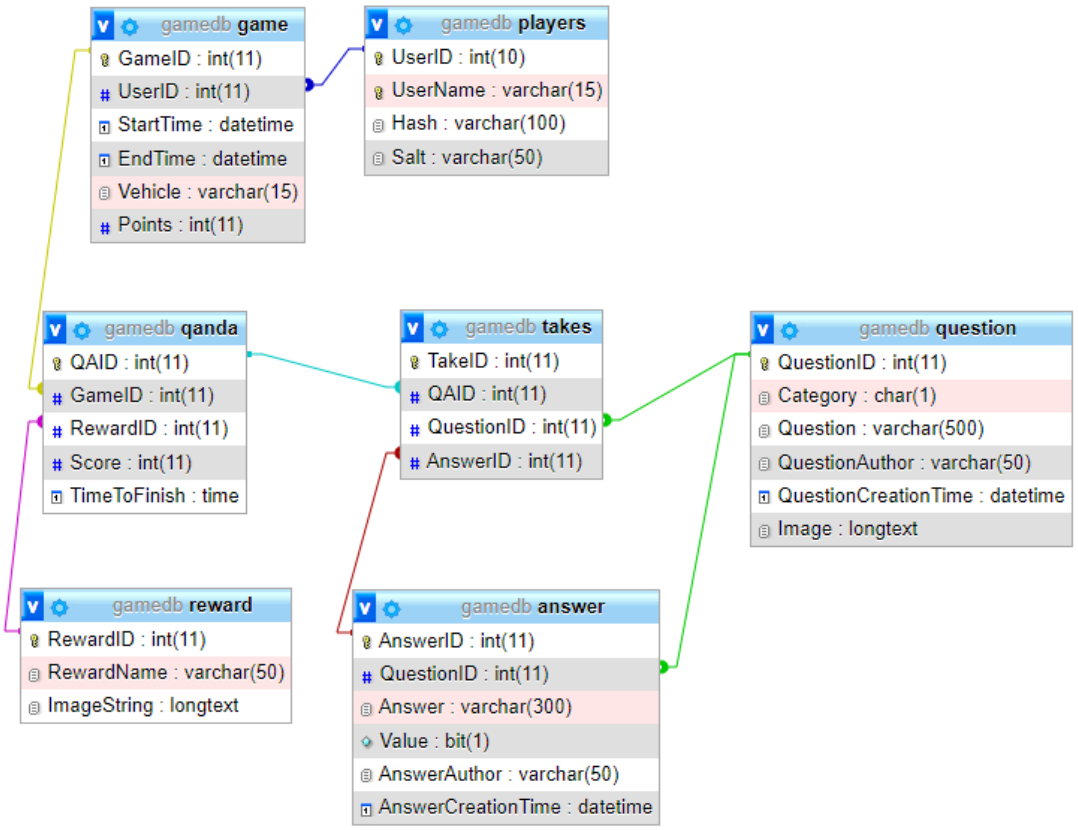


11. ábra: Ütemezés – UI

7. Megvalósítás leírása

7.1 Adatbázis megvalósítása

A megvalósítási folyamat során először létrehoztam az adatbázist. Ehhez letöltöttem és telepítettem a XAMPP nyílt forráskódú platformfüggetlen webszerver szoftvercsomagot, amin telepítés után el is indítottam a MySQL és Apache szervereket, majd webböngészőből a phpMyAdmin applikáció segítségével egyszerű kezelési felületének köszönhetően pár kattintással létre tudtam hozni a játékhoz tartozó adatbázisomat, amely a GameDB nevet kapta, ezt követően pedig a táblákat hozzá. Miután készen lettek a táblák létrehoztam a szükséges indexet, amik az idegen kulcsok beállításához kellenek, majd a tervező nézetet használva beállítottam a megfelelő kapcsolatokat a táblák között. Az alábbi 12. ábrán látható az összeállt adatbázisom.



12. ábra: Adatbázis

Mint látható az eredeti terv szerinti tábla szerkezetben történt néhány módosítás. A Question és a Reward táblákhoz került még egy oszlop, ami képek tárolására szolgál, ennek szükségességét később részletezem a 7.5.1. szekcióban.

A táblák közül a Question, Answer, és Reward táblákat volt szükséges előre feltölteni, a többibe a játék során kerülnek az adatok. Ezeknek feltöltésére pár példa látható az 1. kódrészletben.

```
INSERT INTO `question`(`Question`, `QuestionAuthor`,
`QuestionCreationTime`) VALUES ('Mit ellenőrzöl, amikor átveszed a
rendelést?','Hajdu Renáta', NOW());
INSERT INTO `question`(`Question`, `QuestionAuthor`,
`QuestionCreationTime`) VALUES ('Nincs azonosító írva a csomagolásra, mit
teszel?','Hajdu Renáta', NOW());
INSERT INTO `question`(`Question`, `QuestionAuthor`,
`QuestionCreationTime`) VALUES ('Azt mondják, hogy még várni kell sokat,
hogyan reagálsz?','Hajdu Renáta', NOW());
INSERT INTO `question`(`Question`, `QuestionAuthor`,
`QuestionCreationTime`) VALUES ('Leejted az átvett csomagot és az megsérül,
hogyan tovább?','Hajdu Renáta', NOW());

INSERT INTO `answer`(`QuestionID`, `Answer`, `Value`, `AnswerAuthor`,
`AnswerCreationTime`) VALUES (1,'A messenger üzeneteimet.', 0,'Hajdu
Renáta',NOW());
INSERT INTO `answer`(`QuestionID`, `Answer`, `Value`, `AnswerAuthor`,
`AnswerCreationTime`) VALUES (1,'Az időjárást.', 0,'Hajdu Renáta',NOW());
INSERT INTO `answer`(`QuestionID`, `Answer`, `Value`, `AnswerAuthor`,
`AnswerCreationTime`) VALUES (1,'A környéken lévő pokémonokat.', 0,'Hajdu
Renáta',NOW());
INSERT INTO `answer`(`QuestionID`, `Answer`, `Value`, `AnswerAuthor`,
`AnswerCreationTime`) VALUES (1,'A rendelés tételeit, és a rajta lévő
számot, és nevet.', 1,'Hajdu Renáta',NOW());
```

1. kódrészlet: Adatbázis feltöltés

7.2. Start of game

7.2.1. Main menu

Következő lépésként elkészítettem a főmenüt Unityben, aminek végleges változatát a 13. ábra szemlélteti.

13. ábra: Főmenü

A Unityben a scene (jelenet) egy olyan eszköz, ahová valamilyen tartalom kerül, egy egyszerű játék állhat egy darab sceneből, de alapvetően egy projektben tetszőleges számú jelenet hozható létre, mindegyik egyedi karakterekkel, háttérrel, és stb. Az itt látható menü is egy scene, amiben egy canvas (vászon) adja az alapját a különböző buttonöknek (gomb), amiket látni az ábrán. A Rangsor és Játszok felirattal ellátott button azért sötétebb, mert ezeket csak akkor aktiválom, azaz akkor lehet csak rájuk kattintani, amikor már be van lépve egy felhasználó. Ennek a megállapításához egy LoggedIn bool változónak vizsgáltam az értékét, és azalapján állítottam az interaktivitását. Ennek megvalósításához az alábbi 2. kódrészlet tartozik: [25]

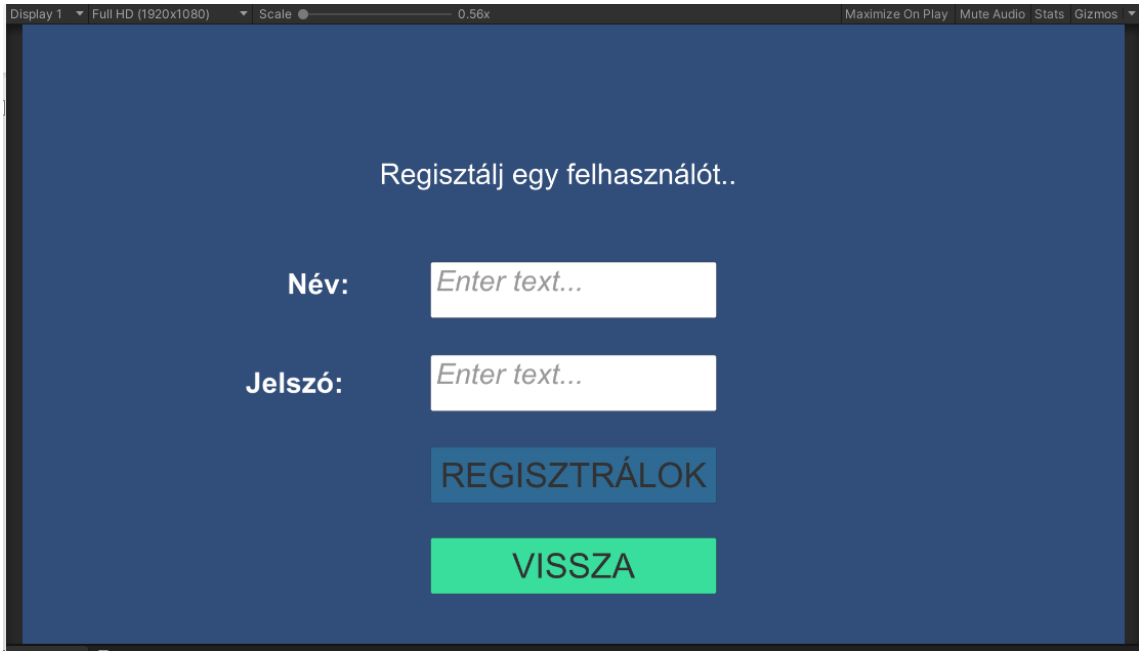
```
private void Start()
    {
        if (DBManager.LoggedIn)
        {
            currentPlayer.text = "Játékos: " + DBManager.username;
        }
        registerButton.interactable = !DBManager.LoggedIn;
        loginButton.interactable = !DBManager.LoggedIn;
        playButton.interactable = DBManager.LoggedIn;
        highscoreButton.interactable = DBManager.LoggedIn;
    }
```

2. kódrészlet: LoggedIn

Mint látható, itt állítom be azt is, hogy ha belépett egy játékos, akkor ki legyen írva a neve.

7.2.2. Registration

A játék használatához regisztráció szükséges, a hozzátartozó UI-t az alábbi ábra mutatja.



14. ábra: Regisztráció

Itt egy név és egy legalább 8 karakter hosszúságú jelszót megadva, majd a regisztrálok gombot megnyomva elmentésre kerül az adatbázisba a felhasználó.

Az adatbázisba történő mentéseket, és onnan adatlekéréseket a projekt során végig PHP fájlok segítségével valósítottam meg, amit a C# kódom UnityWebRequestekkel ér el, amely a webszerverek felé történő HTTP kommunikációt kezeli. Az adatok küldésére a Post metódusát használom, ami a paraméterként megadott URI-ra képes elküldeni form adatokat. A form mezői tartalmazzák a küldendő adatot. Ahol csak adatot szeretnék lekérni ott a Get metódusát használom. Az alábbi 3. kódrészlet mutatja a leírtakat. [25]

```
public void CallRegister()
    {
        StartCoroutine(Register());
    }

    IEnumerator Register()
    {
        WWWForm form = new WWWForm();
        form.AddField("name", nameField.text);
        form.AddField("password", passwordField.text);
        using UnityWebRequest www =
UnityWebRequest.Post("http://localhost/szakdoga/register.php", form);
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.Log(www.error);
        }
        else
        {
            Debug.Log("Sikeres feltöltés!");
        }
    }
```

3. kódrészlet: Register

A megfelelő biztonság elérése érdekében a jelszó elmentésekor kialakítok egy saltot, amibe beiktatom a felhasználónevet is, hogy még kevésbé legyen feltörhető, valamint 5000-re állítom a hash ciklus lefutásának számát, tehát, hogy hányszor keveredjenek össze a karakterek, végül SHA-256 algoritmussal titkosítom, így lesz belőle egy hash kód, ezt tárolom az adatbázisban. Ezt az alábbi 4. kódrészlet mutatja:

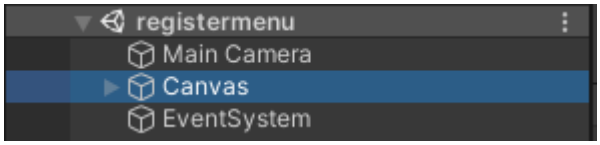
```
$salt = "\$5\$rounds=5000\$" . "lolkabolka" . $username . "\$";
$hash = crypt($password, $salt);
```

4. kódrészlet: Titkosítás

Ezek után a vissza gomb OnClick metódusára beállítottam egy függvényt, ami visszaviszi a felhasználót a főmenübe, ilyen módon történik a gombok hatására történő scene váltás a játékban mindenhol. Unityben C# scripteket gameobjectekre szükséges rakni, ez lehet teljesen üres is, tehát valami, ami nem látszódik a sceneben, csak azt a célt szolgálja, hogy fusson a rajta lévő script. Én jelenesetben a canvashez csatoltam a scriptet, majd a buttonnél beállítottam, hogy a canvasen lévő script GoBack függvényhívása történjen meg, mikor valaki rákattint, mint látható az 5. kódrészletben. [25]

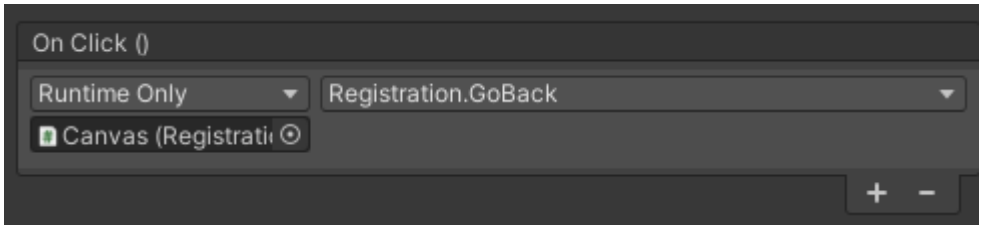
```
public void GoBack()
    {
        SceneManager.LoadScene("startmenu");
    }
```

5. kódrészlet: GoBack



15. ábra: Unity Hierarchy

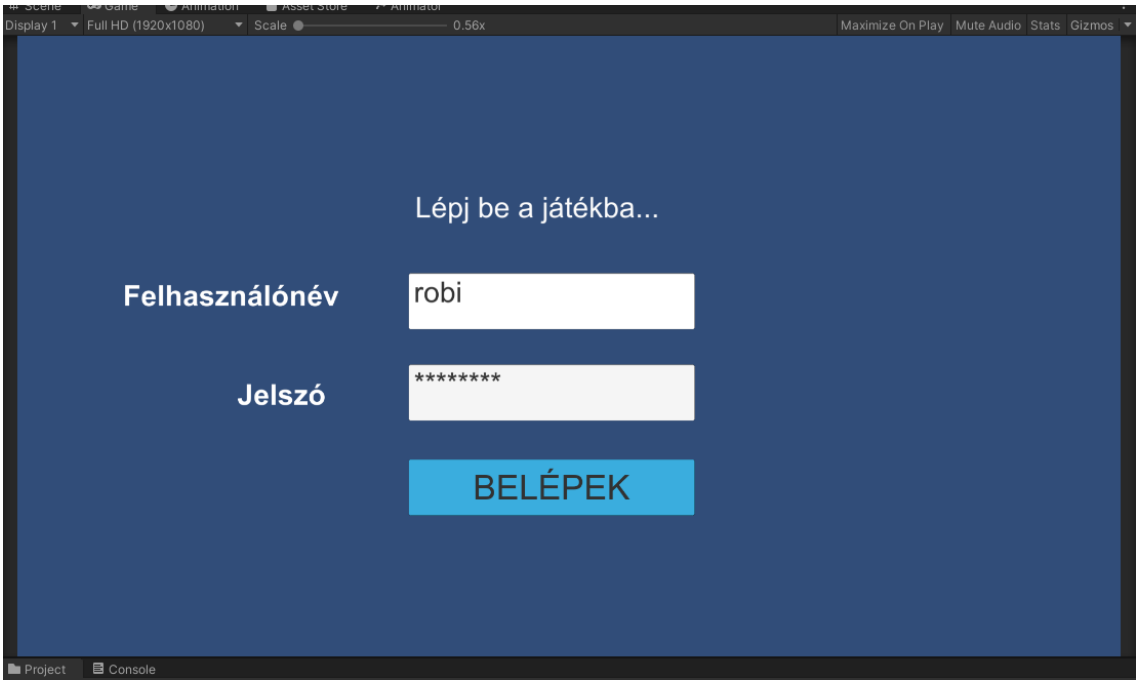
16. ábra: Canvas



17. ábra: Button

7.2.3. Login

Belépéskor a játékosnak meg kell adni a felhasználónevét, és jelszavát, amit regisztráláskor beállított. Ezek megadására, valamint minden alkalommal, amikor a felhasználótól adatot kell bekérni InputFieldeket használok.



18. ábra: Login

A belépést a regisztrációhoz hasonló módon valósítottam meg, lényeges különbség az adatbázis kommunikációnál történik, ahol lekérem a salt és hash értékeket, és összehasonlítom, hogy a beírt jelszó az adatbázisból lekért sóval titkosítva ugyanazt eredményezi-e, mint az adatbázisban lévő hash érték, amennyiben igen, akkor be tud lépni a felhasználó. Ezt a mellékletben lévő 6. kódrészlet szemlélteti.

Az adatbázisból kinyert adatok helyi elmentésére, valamint olyan adatoknak, amik a játék menete közben szükségesek lehetnek, és más táblák feltöltéséhez is kellhetnek egy DBManager osztályt használok a játék során, ahova jelenesetben az InputFieldbe írt nevet el is mentem. Itt látható a 7. kódrészletben.

```
using UnityWebRequest www =
UnityWebRequest.Post("http://localhost/szakdoga/login.php", form);
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.Log(www.error);
        }
        else
        {
            DBManager.username = nameField.text;
            SceneManager.LoadScene("startmenu");
        }
```

7. kódrészlet: Login hívás

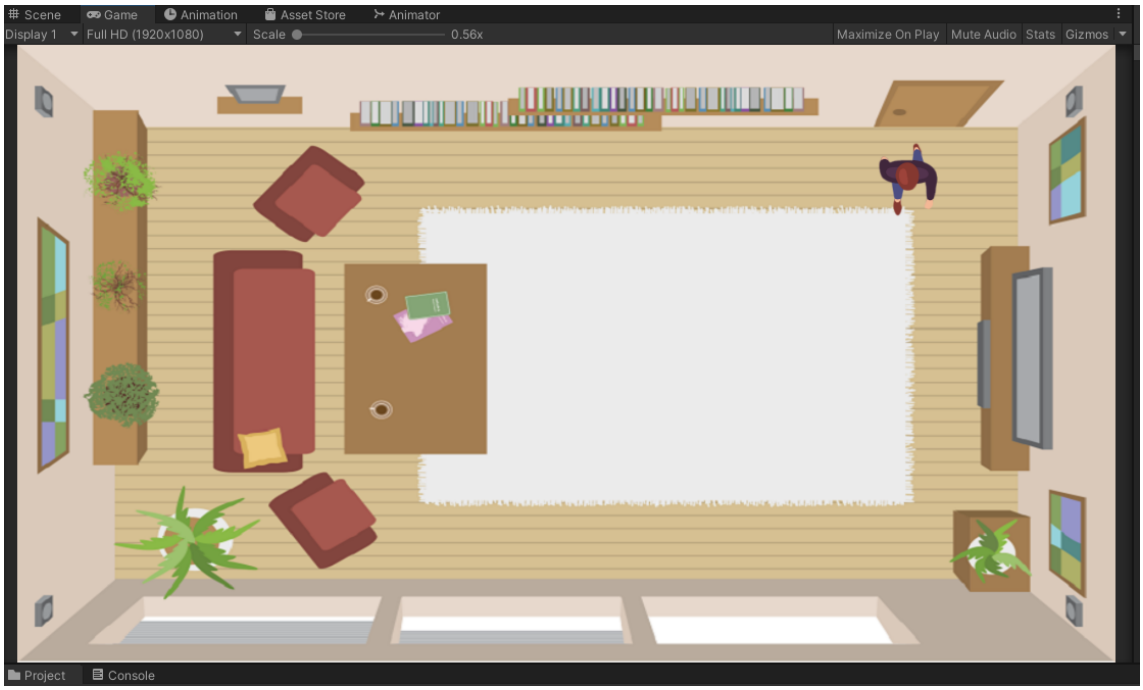
Miután megtörtént a belépés látható, amiről a 7.2.1-es szekciónál beszéltem, hogy megjelenik az aktuális játékos neve.

19. ábra: Logged in menu

7.2.4. Home

Az első scene ahol a játékos a Játszok buttonre kattintva találja magát a futár lakása, ahol egy karaktert tud mozgatni, a feladat pedig, hogy megtalálja az ajtót, így neki tudjon indulni az útjára.

A karakter mozgásának mikéntjére később térek ki, a 7.4.2. fejezetnél, amikor beleütközik az ajtóba, megjelenik a következő scene, ami egy drag 'n' drop játékot tartalmaz, ahol össze kell pakolni a munkához szükséges eszközöket.



20. ábra: Home [24][28]

7.3. Pack your things mini game

Ez tehát egy drag 'n' drop azaz húzd és vidd jellegű játék, ahol az egérrel megfogva egy tárgyat, azt a megfelelő helyére kell húzni.

7.3.1 Dolgok

Ehhez először is képekre volt szükségem, amiket a Vecteezy.com nevű oldalról töltöttem le az ingyenes lehetőségek közül, majd elhelyeztem a játékom sprites mappájában.

Ezekből csináltam gameobjecteket, két részre bontva, egy parentobject alatt vannak elsötétítve, valamint a képernyő egyik oldalára rendezve a leendő helyek, ahova majd be kell húzni a valódi eszközöket, amik pedig egy másik parentobject alatt találhatóak a scene túloldalán.
Csináltam még egy gratuláló feliratot a végéhez, és egy buttont, ami alaphelyzetben inaktív, és csak akkor lehet használni, ha sikerült minden elemet behúzni a célhelyére, ezzel lehet továbbmenni a következő helyszínre, ami a jármű választás lesz.

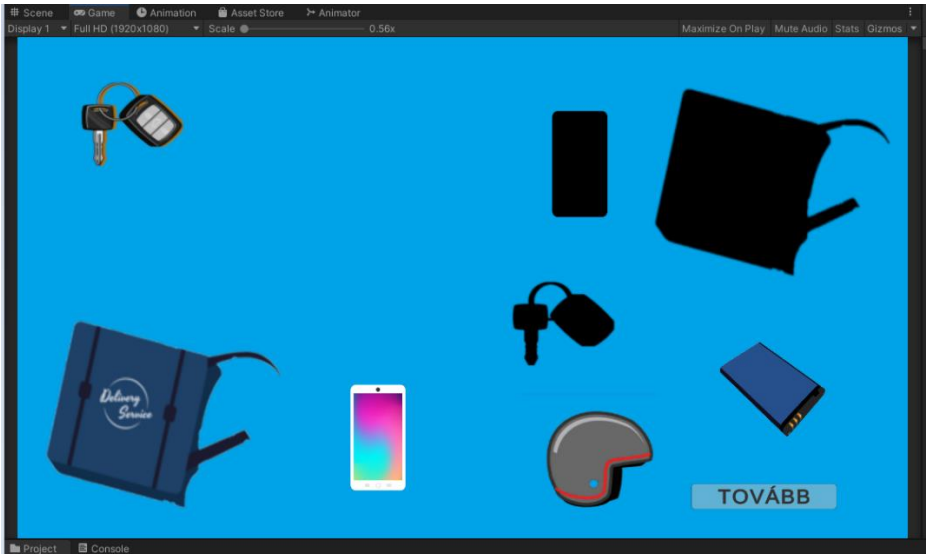
7.3.2 Funkcionalitás

A játék 2 scriptből áll, az egyik kezeli a mozgást, a másik pedig a pontozást.
A script mindegyik nem elsötétített, azaz valódi tárgyra rá van csatolva.

Két fő Unity függvényt használok az OnMouseDownt(), és az OnMouseUp()-ot hiszen a játék lényege, hogy az egérnek a lenyomott és nyomvatartott bal gombjával meg tudunk fogni egy gameobjectet, ami vele mozdul, egészen addig, amíg el nem engedjük, ilyenkor visszaugrik az eredeti pozíciójába, ha nem volt elég közel, ellenkező esetben pontosan a kívánt hely fölé illeszkedik.

Az OnMouseDown() függvénynél megvizsgálom, hogy a bal klikk le van-e nyomva, és ha igen, eltárolom az egér pozícióját, valamint beállítom, hogy mozgásban van.
Használnom kellett az Update Unity függvényt is, amely képkockánként lefut, tehát a folyamatos frissítésre tudom használni, itt folyamatosan ellenőrzöm, hogy amennyiben nincs még a helyén egy object, ha mozgásban van még az egér, frissítem annak a helyzetét, és vele együtt a megfogott játékelem helyzetét. [25]

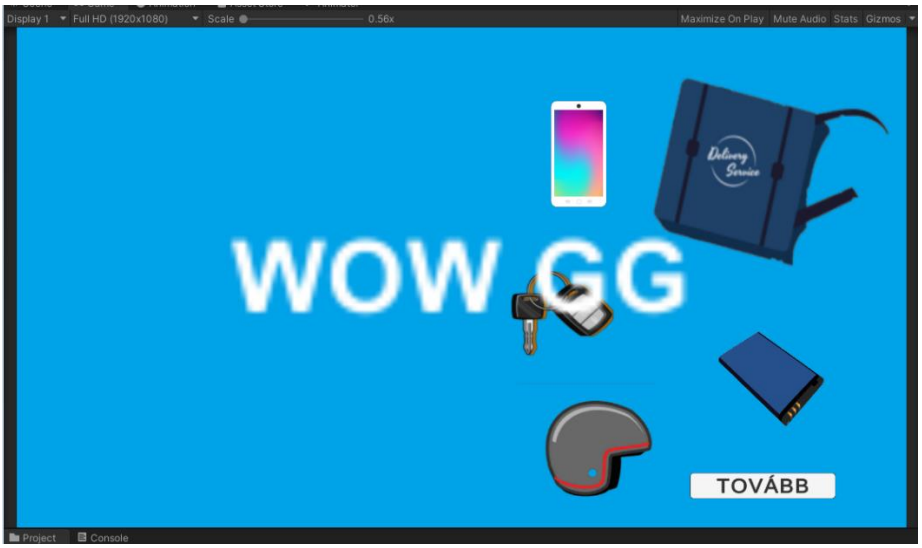
Az OnMouseUp() metódusnál pedig először is beállítottam, hogy már nincs mozgásban az egér, majd megvizsgálom a távolságot a megfogott elem és a cél között, és ha ez elég kicsi akkor behúzom a hely pozíciójába, és készre állítom, ellenkező esetben pedig visszaállítom a pozícióját az elején eltárolt eredeti helyére. Tehát ha elengedünk valamit, és még nincs elég közel a kívánt helyéhez, akkor az eredeti helyére ugrik.



21. ábra: Drag ’n’ Drop [28]

Itt látható, hogy a bukósisak és a powerbank már a megfelelő helyen vannak.

Amennyiben sikerült behúznom egy elemet a helyére a pontokat kezelő script segítségével hozzáadok egy pontot, és ellenőrzöm, hogy ez volt-e az utolsó elem, ha igen akkor előhívom a gratuláló feliratot, és aktívvá válik a tovább gomb.



22. ábra: Drag 'n' drop end [28]

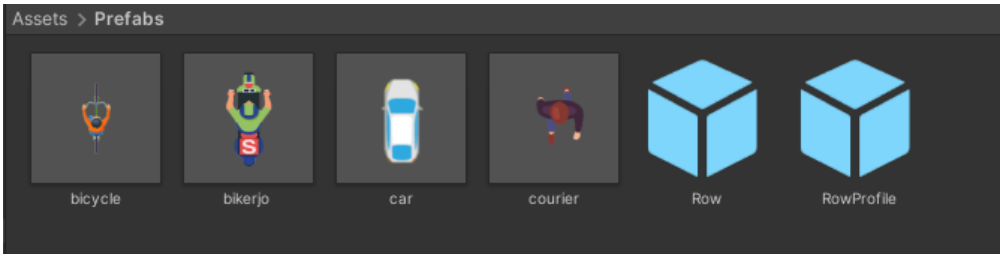
Ebben a scriptben található még egy függvény, amivel lekérem az adatbázisból a játék azonosítóját és eltárolom a DBManagerben, hogy a továbbiak során használni tudjam.

7.4 Jármű választás, és kezelés

7.4.1 Járművek

Magát a járműveket úgy hoztam létre, hogy betöltöttem a map (térkép) scene-be, ahol lehet közlekedni, és a kiszállítást megvalósítani, egy jármű sprite-ot, hozzáadtam egy Rigidbody2D -t. Ez arra szolgál, hogy egy objektumot a Unity fizikai motorjának (fizikai rendszereket szimulál) irányítása alá helyez, tehát lehet majd az objektumnak gyorsulása, ha szeretném hathat rá a gravitáció, ütközhet mással és stb. Felruháztam még Colliderrel, ezek a komponensek fizikai ütközések kezelésére szolgálnak, valamint kapott még egy scriptet a mozgásának a megvalósítására, amit a következő szekcióban tárgyalok. [25]

Az így létrehozott motoros karakterből Prefabet kreáltam, majd letöröltem a térképről.
A Prefab arra való, hogy egy létrehozott játék objektumot minden beállításával, összetevőjével együtt el lehessen tárolni egy újrafelhasználható eszközként. Ezekből később bármikor példányokat lehet létrehozni különböző jelenetekben, és személyre szabni, ha valamit módosítani akarunk rajta, úgy működik, mint egy sablon. [25]



23. ábra: Prefabs [28]

7.4.2 Mozgásrendszer

Minden járműhöz csatolva van egy script, ami biztosítja azt, hogy megfelelő irányításra mozogni tudjanak a pályán. A jobb, bal, fel, le nyilakkal lehet irányítani őket, ehhez a Unity Update eseményében vizsgáltam az inputokat, és eltároltam egy string változóban, hogy melyik gomb lett lenyomva. Az Update minden képkocka frissülésnél meg van hívva, olyan funkciókat szokás itt elhelyezni, mint nem fizikai objektumok mozgatása, egyszerű időzítők, és mint amire én is használom jelenesetben, az inputok fogadására. Fontos még tudni róla, hogyha egy képkockát több idő feldolgozni, mint egy másikat, akkor az idő két Update hívásnál különböző lehet. [25] [26]

Az inputok lekérése után a FixedUpdate eseményben az előbb elmentett változó értéke alapján a járműhöz tartozó RigidBody2D sebességét állítottam sebességvektorokkal, valamint a megfelelő irányba forgatást is itt intéztem a hozzájuk tartozó MoveRotation függvénnyel, aminek csak egy szöget kell megadni. A FixedUpdate hívások között mindig ugyanannyi idő telik el, fizikai lépésenként kerül meghívásra, tipikusan objektumok fizikai tulajdonságainak állítását szokás itt elvégezni. [25]

A játékban, amikor egy járművel odaérünk az étteremhez, egy zebrához, vagy a vásárló házához, azaz beleütközünk, akkor kerülünk át a következő jelenetbe. Ezeknek az ütközéseknek a kezelését is itt oldottam meg a Unity OnCollisionEnter2D eseményében, ami pontosan akkor van meghívva, amikor egy fizikai ütközés történik két Rigidbody2D, vagy Collider2D között. A térképre ezért Collidereket helyeztem el, és az említett helyszíneket egyedi taggel ruháztam fel, így ütközéskor meg tudom vizsgálni, hogy a gameobject amivel ütközik a jármű, milyen taggel rendelkezik, és ha pl. RESTAURANT_TAG birtokában van, akkor a SceneManager, azaz a Unity jelenet kezelő osztálya segítségével betöltöttem a következő jelenetet pl. „restaurant”. Valamint elmentettem a DBManagerben is egy string változóban, hogy hol járunk, a későbbi felhasználás érdekében. [25]

Ezenkívül amit fontos megemlítenem, a jármű helyzetének eltárolása, példaként említeném meg, hogy amikor kijön az étteremből a játékos, ott jelenjen meg a térképen, ne pedig a kezdeti helyén, a 8. kódrészletben látható.
Ehhez elmentettem a PlayerPrefs Unity osztály segítségével helyzetüket az Updateben, majd az Awakeben lekértem ezeket, és elhelyeztem float változókba, végül ezen változók felhasználásával állítottam be a kezdésnél a Startban a helyzetét, alábbi 8. kódrészlet mutatja.
A Start esemény a Unityben akkor kerül meghívásra, amikor egy GameObject elkezd létezni, tehát vagy amikor egy jelenet betöltődik, vagy amikor a játékelem inicializálódik, mindig az első Update előtt. Automatikusan hívódnak meg, amikor egy script betöltődik, először az Awake, akkor is, ha a script nincs aktiválva, utána a Start, ami pedig csak

akkor, ha aktív a script, azaz nincs kikapcsolva. Ez a két függvény csak egyszer van meghívva egy játékelemhez csatolt script életében. [25] [26]

```
public void Awake()
    {
        xpos = PlayerPrefs.GetFloat("myposx");
        ypos = PlayerPrefs.GetFloat("myposy");
        zpos = PlayerPrefs.GetFloat("myposz");
    }

void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        transform.position = new Vector3(xpos, ypos, zpos);
    }

void Update()
    {
        if (Input.GetKey(KeyCode.RightArrow))
        {
            buttonPressed = RIGHT;
        }
        else if (Input.GetKey(KeyCode.LeftArrow))
        {
            buttonPressed = LEFT;
        }
        else if (Input.GetKey(KeyCode.UpArrow))
        {
            buttonPressed = UP;
        }
        else if (Input.GetKey(KeyCode.DownArrow))
        {
            buttonPressed = DOWN;
        }
        else
        {
            buttonPressed = null;
        }

        PlayerPrefs.SetFloat("myposx", transform.position.x);
        PlayerPrefs.SetFloat("myposy", transform.position.y);
        PlayerPrefs.SetFloat("myposz", transform.position.z);
    }
```

8. kódrészlet: Pozíció mentés

Az otthon lévő futár karakternek a mozgása ugyanilyen módon van megoldva, először animálni akartam a járását Unity animációkat használva, de végül sajnos nem volt rá időm, így maradtam ennél a módszernél.

7.4.3 Választás

3 jármű közül lehet választani, mint a képen is látható, mind a háromhoz tartozik egy-egy Button, amelyek megnyomására betöltődik a mapen a kiválasztott jármű.

24. ábra: Járműválasztás

Ennek a megoldása úgy áll össze, hogy eltároltam egy változóban, hogy melyik gomb került megnyomásra, majd ez alapján két dolgot tettem:

- Eltároltam a DBManagerben a kiválasztott járművet, és a CallSaveVehicle függvénnyel a már korábban említett UnityWebRequestes módon PHP segítségével elmentettem az adatbázisba a Game táblába. Ehhez először a felhasználónév átadásával lekérdeztem a felhasználó azonosítóját, majd az alapján hozzáadtam a járművet a megfelelő sorhoz. Itt még annyit megemlítenék, hogy mint látható használtam plusz egy feltételt az UPDATE queryben, ugyanis mivel egy játékos többször is játszhat, tehát többször is szerepelhet az azonosítója a Game táblában, még azt is megvizsgáltam, hogy egyenlő-e a játék kezdés és a végzés időpontja, mert ha igen, akkor az jelenti azt, hogy az aktuális játékról van szó, nem egy korábban játszottról. Az alábbi 9. kódrészletben látható.

```
 $getuseridquery = "SELECT UserID FROM players WHERE UserName= '". $username .
"';";

    echo $getuseridquery . "\n";

    $run = mysqli_query($con, $getuseridquery) or die("Get user ID query fai-
led");
    $result = mysqli_fetch_assoc($run);
    $userid = $result["UserID"];

    $updatequery = "UPDATE game SET Vehicle = '" . $vehicle . "' WHERE UserID
= " . $userid . " AND EndTime=StartTime;";
    echo $updatequery;

    mysqli_query($con, $updatequery) or die("Save query failed");
```

9. kódrészlet: Jármű mentés

- Másik fontos dolog a VehicleManager scriptem, amely a kiválasztott jármű betöltéséért felelős. A VehicleManager példányom megkapja a kiválasztott button, azaz jármű indexét, majd ez alapján intézi annak a példányosítását a mapre, a Singleton programtervezési mintát figyelembe véve, ami azt szabályozza, hogy egy gameobjectből csak egy darab másolatunk lehessen. Ez azért volt fontos számomra, ugyanis a VehicleManager Scriptem egy Manager gameobjecten van rajta, és ez a script felelős azért, hogy az egyik sceneben kiválasztott jármű a másikban kerüljön betöltésre, tehát történik közben egy jelenetváltás, mely alatt végig folyamatosan léteznie kell betöltésért felelős objektumnak. Viszont nem lehet több se belőle, tehát ha van már egy létező manager objektumom, azt el kell pusztítani, hiszen, ha több lenne belőle, akkor lehetséges, hogy több lenne a kiválasztott jármű indexéből, tehát már nem lenne egyedi, így összezavarná a programot. Ebbe a hibába többször is beleestem, és előfordult, hogy több pl. motor is betöltésre került. Az ehhez kapcsolódó 10. kódrészletet itt mutatom.

```
public class VehicleManager : MonoBehaviour
{
    public static VehicleManager instance;

    [SerializeField]
    private GameObject[] vehicles;

    private int _charIndex;
    public int CharIndex
    {
        get { return _charIndex; }
        set { _charIndex = value; }
    }

    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }
```

10. kódrészlet: Singleton

Ezek után a SceneManager sceneLoaded eseményére feliratkoztatva egy függvényt, figyelem, hogy betöltött-e a map, amennyiben igen, akkor példányosítom a megfelelő járművet, mint alább látható megjelenik egy motoros.

25. ábra: map [24] [28]

7.5 Kvíz

A játékban összesen 3 kvíz szerepel, egyhez használtam képeket, a másik kettőhöz nem. Első nekifutásra készítettem több különböző scenet hozzájuk, de útközben rájöttem, hogy mivel nagyjából ugyanaz lesz a funkcionalitásuk, ezért sok időt és energiát tudnék megspórolni azáltal, ha egy scenenel meg tudnám oldani mind a hármat, ez okozott némi fejtörést, de végül bár nem olyan szépen, mint, ahogy eredetileg elgondoltam, de sikerült megoldanom.

7.5.1 Képek kezelése

Először szeretnék mesélni a kvízben szereplő képek kezeléséről, ez okozta az egyik legnagyobb nehézséget számomra az egész projekt során. Az eltárolásukhoz első körben azt terveztem, hogy készítek egy táblát, amiben BLOB adattípusként tárolom el őket, egy egyszerű PHP-ban alkotott HTML upload form segítségével feltöltöm, majd C# oldalon lekérem.
A feltöltés sikerült is, itt láthatóak a betöltött BLOB fájlok, azonban sajnos nem sikerült többszöri, különböző megoldásokkal sem betöltenem Unitybe, így végül egy másik módszert kerestem.

		id	QuestionID	name	image
☐	Módosítás Másolás Törlés	1	7	pic1.png	[BLOB - 28.6 KB]
☐	Módosítás Másolás Törlés	2	8	nobicikli.png	[BLOB - 21.0 KB]
☐	Módosítás Másolás Törlés	3	9	kerekparsav.png	[BLOB - 11.0 KB]

26. ábra: BLOB

A működő megoldás alapja az, hogy longtext formátumban tárolom el a képeket. Ehhez hozzáadtam a Reward táblában egy új oszlopot, valamint a Question táblában is, a KRESZ kérdésekhez.

A képek feltöltéséhez készítettem egy scenet, amin nincsen semmi más, csak egy kép, és hozzá van csatolva egy ImageUpload script. Összefoglalva úgy funkcionál, hogy a kép elemre ráhúzok egy képet forrásként, amit majd fel szeretnék tölteni, a script pedig a háttérben egy PHP segítségével feltölti az adatbázisba.
A Start függvényben a képet byte tömbbé konvertálom, majd onnan stringgé, és ezt a stringet töltöm fel az adatbázisba. A 11. kódrészletben megtekinthető.

```
void Start()
    {
        byte[] imgByte = img.sprite.texture.EncodeToPNG();
        imageString = System.Convert.ToBase64String(imgByte);
        StartCoroutine(Upload());
    }
```

11. kódrészlet: Kép feltöltés

Visszafelé pedig, amikor képet alakítok a stringből, ott a PHPban megjelenítem a stringet, majd ezt a downloadhandlerben lévő szövegként megkapva visszaalakítom byte tömbbé, készítek egy új texture-t, amire rátöltöm a bytesort, és végül ez alapján készítek egy új sprite-ot. Ez alább látható a 12. kódrészletben.

```
IEnumerator DownloadImage()
        {
            UnityWebRequest www =
UnityWebRequest.Get("http://localhost/szakdoga/getimagefromdb.php");

            //www.chunkedTransfer = false;
            yield return www.SendWebRequest();

            string webtext = www.downloadHandler.text;
            Debug.Log(webtext);
            byte[] imageAsBytes = System.Convert.FromBase64String(webtext);

            Texture2D newtexture = new Texture2D(1, 1);
            newtexture.LoadImage(imageAsBytes);
            newtexture.Apply();

            Sprite spr = Sprite.Create(newtexture, new Rect(0, 0,
newtexture.width, newtexture.height), new Vector2(0.5f, 0.5f));
            img.sprite = spr;
        }
```

12. kódrészlet: Kép letöltés

7.5.2 Kérdések és válaszok lekérése az adatbázisból

Ahhoz, hogy fel tudjam tölteni tartalommal a kvízt, valahogy le kellett kérnem az adatbázis Question és Answer tábláiból a kérdéseket, és válaszokat. Ezt hasonló módon tettem meg, mint a korábban, UnityWebRequesttel és PHP-val, a 13. kódrészlet szemlélteti.

```
if ($result->num_rows > 0) {
     while($row = $result->fetch_assoc()){
        echo $row["QuestionID"]. "-" . $row["Category"] . "-" .
$row["Question"] . "\n";
        }
    }
```

13. kódrészlet: Kérdések lekérése

Itt látható a lényeges részlet belőle, ahol enterekkel elválasztva íratom ki a lekérdezés sorait, amikben pedig „-” karakterekkel választom el az egyes adatokat, ez azért fontos, mert ezeket használva darabolom fel a requestben megkapott szöveget. Először az enterek mentén egy string listába kigyűjtöm a kérdéseket, válaszokat, majd egy UploadDBManager függvényben ezen listákat még egy foreach ciklussal bejárva, a ’-’ karakterek mentén elválasztva feltöltöttem egy kérdés és válasz listát. A kérdésekhez tartozik egy-egy osztály az alábbi 14. kódrészlet mutatja.

```
public class TheQuestion
{
    public int questionID;
    public char category;
    public string questionInfo;
}

public class TheAnswer
{
    public int answerID;
    public int questionID;
    public string AnswerInfo;
    public bool value;
}
```

14. kódrészlet: Válasz és kérdés osztály

Ezen a ponton kellet megoldást találnom arra, hogy majd egy scene elég legyen mindhárom kategóriájú teszthez, így a kérdéseket a category alapján szétszedtem és 3 különböző listába tettem a DBManagerbe, hogy egyszerűen tudjam majd kezelni. Itt látható a 15. kódrészletben.

```
foreach (var item in theQuestionList)
        {
            if (item.category == 'R')
                DBManager.RestaurantQuestions.Add(item);
            else if (item.category == 'C')
                DBManager.CustomerQuestions.Add(item);
            else if (item.category == 'T')
            {
                DBManager.TrafficQuestions.Add(item);
            }
        }
```

15. kódrészlet: Kategorizálás

A válaszoknál nem volt szükség ilyenre, ott egy darab válasz lista van a DBManagerben az összes válaszhoz, hiszen, ha a kérdések már külön vannak szedve, a válaszokban lévő questionID alapján egyszerűen ki lehet választani a megfelelő válaszokat.

7.5.3 Kvíz manager

A kvíz működéséhez használtam egy TestManager üres gameObjectet, amihez csatoltam a TestManager scriptet, ez a fő kezelője az egész kvíznek. A Startjában van pár szükséges beállítás, és függvényhívás, ezek a CallSaveQuizStart, amivel elmentettem a QandA táblába az aktuális játék azonosítóját, és lekértem az így kreált új sor QandA azonosítóját. Inicializáltam pár változót, mint a rossz válaszokat, eredményeket, hiszen kezdetben még nincs, hogy a score 0 legyen, az elindulással egyidőben elinduljon az időzítő, az időlimit legyen az aktuális idő, és abból indulok ki, hogy kap a játékos jutalmat, majd később ezt átállítom, ha becsúszik egy rossz válasz. A 16. kódrészlet kapcsolódik ehhez.

```
void Start()
    {
        CallSaveQuizStart();
        scoreCount = 0;
        timerIsRunning = true;
        currentTime = timeLimit;
        results = new List<bool>();
        wrongAnswers = "";
        getsAReward = true;
        SelectQuestion();
    }
```

16. kódrészlet: Inicializálások

A quiz scene betöltésekor az első fontos dolog egy kérdés betöltése és megjelenítése volt. Ezt végzi el a TestManager osztályomból a SelectQuestion függvény, ami az alapján, hogy mi az aktuális jelenet elmentve a DBManagerben, kiválaszt egy kérdést a megfelelő kérdés listából, elmenti az azonosítóját későbbi felhasználásra, meghívja azt a függvényt, ami a kérdés UI-ra betöltéséért felelős, és ki is szedi a listából a kiválasztott kérdést, hogy ne lehessen ismétlődés a megjelenő kérdések között. Erre egy példát mutat az alábbi 17. kódrészlet:

```
private void SelectQuestion()
    {
        if (DBManager.currentScene == "restaurant")
        {
            int val = UnityEngine.Random.Range(0,
DBManager.RestaurantQuestions.Count);

            selectedQuestion = DBManager.RestaurantQuestions[val];
            currentQuestionID = selectedQuestion.questionID;
            loadQuizData.LoadQuestion(selectedQuestion);

            DBManager.RestaurantQuestions.RemoveAt(val);
        }
```

17. kódrészlet: Kérdés választás

A LoadQuestion függvény a kiválasztott kérdés szövegét átadja a UI-on lévő kérdés Textboxnak, valamint a hozzá tartozó válaszokat is beállítja a 4 db button feliratának. Itt két kódrészletet emelnék ki, az első az, amiben megvizsgáltam, hogy a scene az „traffic”, ugyanis csak a KRESZ témakörrel kapcsolatos kérdéseknél van szükség képre is a kérdés szövege mellett, ilyenkor aktiváltam a kérdésnél lévő kép objektumot, és elindítottam a függvényt, amely azért felelős, hogy kép legyen az adatbázisbeli stringből, mint ahogy korábban volt erről szó a 7.5.1 fejezetnél. A 18. kódrészlet mutatja.

```
if (DBManager.currentScene == "traffic")
        {
            questionImg.transform.parent.gameObject.SetActive(true);
            questionImg.transform.gameObject.SetActive(true);
            StartCoroutine(GetImageFromDb(question.questionID));
        }
```

18. kódrészlet: Traffic kérdések

A GetImageFromDb függvény a korábbit annyival egészíti ki, hogy kapott egy int paramétert, aminek át tudtam adni az aktuális kérdés azonosítóját, hogy a megfelelő kép töltődjön be.

A másik 19. kódrészlet a megfelelő válaszoknak a lekérése, itt egy string listához adtam a questionID alapján lekérdezett válaszokat.

```
List<string> answeroptions = new List<string>();

        var asd = from q in DBManager.theAnswerList
                  where q.questionID == question.questionID
                  select q;
        foreach (var item in asd)
        {
            answeroptions.Add(item.AnswerInfo);
        }
```

19. kódrészlet: Válaszok kiválasztása

Miután minden kérdés és válasz be volt töltve az OnClick() függvény vizsgálja, hogy rá kattintottak-e valamelyik Buttonre, meghívja a TestManager Answer() függvényét, ami eldönti, hogy jó vagy rossz válaszra kattintott a felhasználó, és ezalapján kilogolja, hogy helyes vagy helytelen válasz volt.

Az Answer függvény megkap tehát egy stringet, a választott button szövegét. Egy lekérdezéssel lekértem ennek az értékét, tehát, hogy igaz-e, vagy hamis, ezt el is mentettem a kezdéskor inicializált bool listába, hogy később tudjam vizsgálni, hogy egy játékosnak hány rossz-jó válasza volt, valamint az azonosítóját is lekérdeztem a válasznak, hiszen az adatbázisomban is el akartam tárolni minden kvízhez, hogy ki melyik kérdésnél melyik választ adta, a 20. kódrészletben látszik.

```
var corrAns = (from a in DBManager.theAnswerList
                       where a.AnswerInfo == selectedOption
                       select a.value).Single();

results.Add(corrAns);

chosenAnswerID = (from a in DBManager.theAnswerList
                          where a.AnswerInfo == selectedOption
                          select a.answerID).Single();

CallSaveTakes();
```

20. kódrészlet: Adatbázis feltöltés

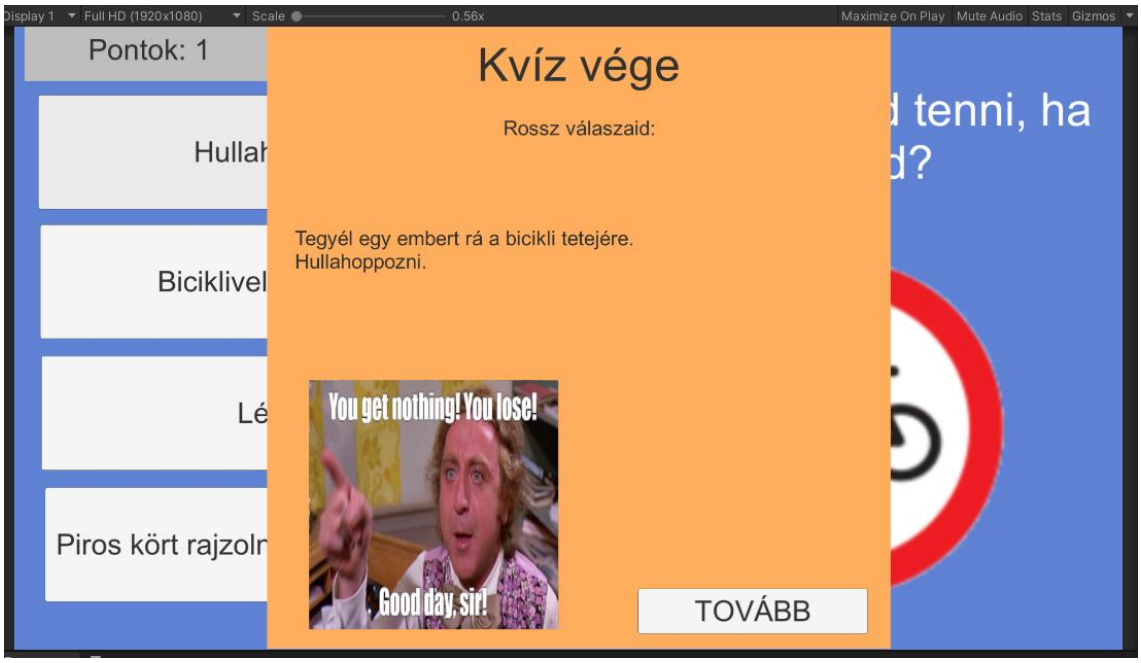
Így ezután meg is hívtam a függvényt, ami elmenti az aktuális kvíz aktuális kérdésénél az épp kiválasztott választ, a hozzá kapcsolódó webform az alábbi mezőket tartalmazza, és adja tovább a phpnak.

```
IEnumerator SaveTakes()
    {
        WWWForm form = new WWWForm();
        form.AddField("qaid", DBManager.QAID);
        form.AddField("questionid", currentQuestionID);
        form.AddField("answerid", chosenAnswerID);
```

21. kódrészlet: Választások elmentése

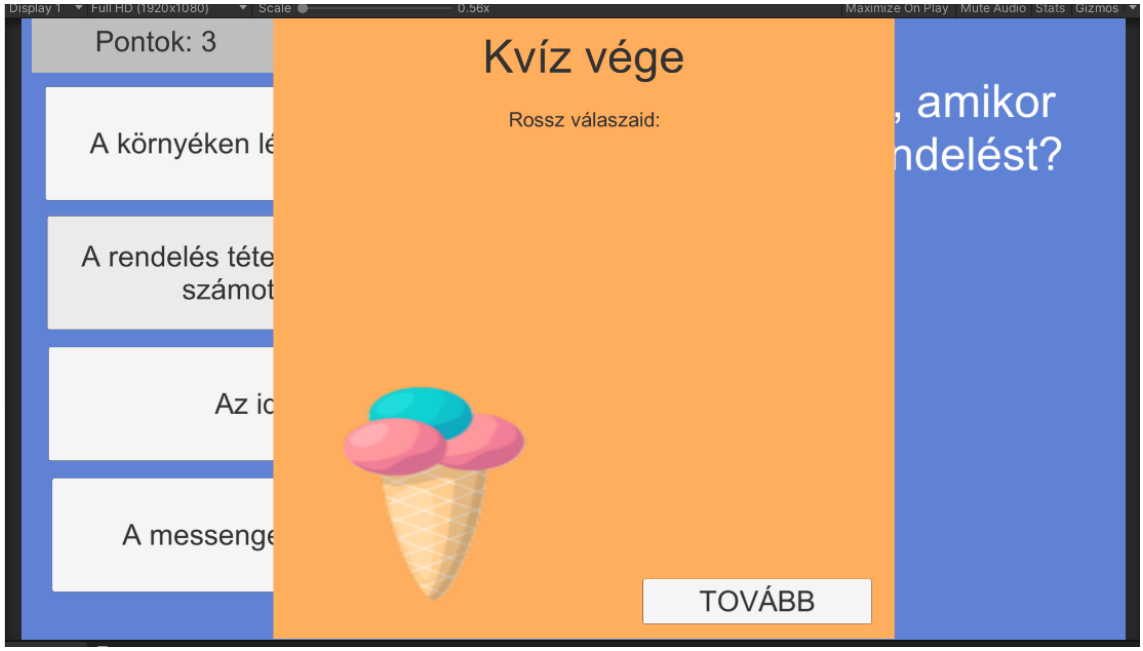
Az 20. kódrészletnél beállított corrAns változó értéke alapján végzek ezután műveleteket, ha igaz akkor, növelem a pontok számát, és ezt rögtön ki is íratom a UI-en, ha hamis volt, akkor pedig hozzáadom a wrongAnswers stringhez, hogy a végén meg tudjam mutatni a játékosnak, hogy mit rontott el.

A kvíz elején beállított időzítő folyamatosan csökken, így itt megvizsgáltam, hogy van-e még idő, ha lejárt az időlimit, akkor felugrik egy panel, amin láthatóak a rossz válaszok, ha voltak.



27. ábra: Kvíz vége [28] [29]

Ha pedig még van idő, és kérdés is van, akkor meghívom a SelectQuestion() függvényt, és jöhet az új kérdés, ha pedig nincs több kérdés, akkor mint az előbb, vége van a kvíznek, elmentem a pontokat, időt, és ha minden kérdést jól válaszolt meg a játékos kap egy jutalmat. Ide vonatkozik a mellékletekben lévő 22. kódrészlet.



28. ábra: Hibátlan kvíz [28]

7.6 Ranglista

A ranglistánál amit megjelenítettem a felhasználónév, a jármű, amit használt a játékos, az elért eredménye és az idő, ami alatt teljesítette a játékot. Ezeknek létre is hoztam egy Score osztályt, majd csináltam belőle egy listát, ahova a már megszokott módon lekértem az adatbázisból a szükséges adatokat, és feltöltöttem vele a listát. Itt szükségem volt a players táblára is a felhasználónév miatt, valamint a játék kezdési és befejezési idejének rögtön a különbségét kértem le, a 23. kódrészletben látható.

```
$getscoresquery = "SELECT UserName, TIMEDIFF(EndTime, StartTime) AS
TimeToFinish, Vehicle, Points FROM `players`, `game` WHERE players.UserID =
game.UserID;";
```

23. kódrészlet: Eredmények lekérése

Miután ez megvolt, egy másik függvénnyel sorba rendeztem az eredményeket először pont szerint csökkenően másodsorban pedig teljesítési idő szerint növekvően.

Végül egy ScoreUI nevezetű osztályban elintéztem ezek megjelenítését, itt a turpisság abban rejlik, hogy készítettem egy Row prefabet, amely textboxokat tartalmaz, hozzá van csatolva egy RowUI script, amely az alábbi 24. kódrészletben látható elemeket tartalmazza, és ebben az osztályban példányosítottam belőle annyit, ahány sor highscoreom van és beállítottam a sor megfelelő textjeinek a megfelelő értékeket.

```
public class RowUI : MonoBehaviour
{
    public Text username;
    public Text vehicle;
    public Text points;
    public Text time;
    public Text rank;
}
```

24. kódrészlet: RowUI

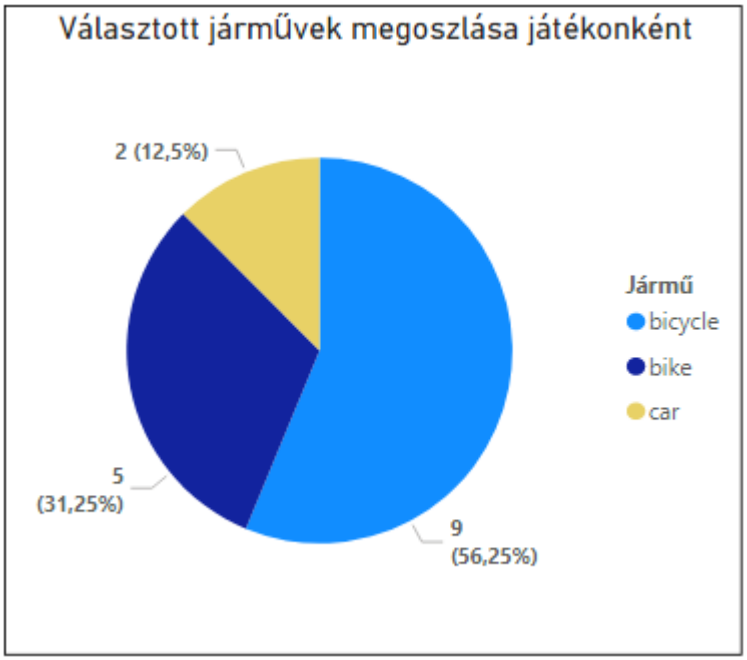
7.7 Adatvizualizáció

Miután kész lett a játékom lejátszottam vele annyi kört, különböző játékosokkal, hogy legyen elég adat, amiből vizualizációt lehet készíteni.
A PowerBI-on belül az adatok lekérésénél kikerestem a MySQL adatbázis opciót, kiszolgálónak megadtam a localhost címét 127.0.0.1-et, adatbázisnak pedig a GameDB-t, majd miután sikeresen csatlakozott, kiválasztottam a szükséges táblákat, amiket a vizualizációhoz használni akartam. Ezután még kitöröltem pár oszlopot, amit nem találtam szükségesnek és végül beimportáltam a táblákat. A megfelelő kapcsolatokat fel

is ismerte a PowerBI, és a mellékletekben lévő 29. ábrán látszik a létrejött adatbázis szerkezet.

Ezek alapján létrehoztam 3 riportot:

Az első riporton látszik, hogy hogyan oszlanak meg a játékosok által kiválasztott járművek, mit részesítenek előnyben, mint látható a bicikli túlnyomú többséggel vezet, autóból van a lekevesebb, az alábbi 28. ábra mutatja.



28. ábra: Jármű riport

Következő riporton a mellékletekben lévő 31. ábrán megfigyelhetőek a kritikus kérdések. Az x tengelyen látszanak a kérdések, az yon pedig, hogy hányan válaszoltak rá jól, és rosszul, mint látható a legkritikusabb, hogy nem tudják mit kell tenni, amikor azt mondják az étteremben, hogy még sokat kell várni, amit a leginkább jól tudnak, az, hogy mit kell ellenőrizni amikor átvesznek egy rendelést.

Az utolsó riporton pedig a mellékletekben elhelyezett 32. ábrán az látható, hogy melyik felhasználó melyik játéka során hány pontot ért el. Innen világos, hogy ki az, akiből jó futár válhat, és ki az, akinek még gyakorolnia kell.

8. Eredmények bemutatása és értékelése

Ahonnan az ötletem a szakdolgozat témára jött, az a munkahelyem volt, ahol sajnos nagyon sok probléma adódik abból nap, mint nap, hogy nem rendelkeznek a szükséges tudással a futárok a rendelések kiszállítása során felmerülő problémák megoldásához, és a különböző tájékoztatókat sem gyakran olvassák el az eljárásokról, ezért akartam készíteni egy oktatójátékot, amely elejétől a végéig szimulál egy kiszállítási folyamatot, lefedve a témákat, amiről mindenképpen tudniuk kell, hiszen játszva könnyebben tanul az ember, és hozzá egy BI rendszert, amely könnyen átlátható jelentésekkel mutatja a vezetőség felé, hogy mik a kritikus pontok, és melyik ki hogyan teljesít.

A projekt elkészítése közben több helyen is megakadtam, és nagyon sok fejlesztési lehetőséget rejt még, amit a következő fejezetben tárgyalok is, de összeségében sikerült elérnem, amit szerettem volna. Alkottam egy oktatójátékot, amely lefedi egy rendelés eljuttatását a vásárlóhoz, az eredmények eltárolásra kerülnek egy adatbázisban, és megjelennek riportok formájában.

A dokumentáció során mutattam képernyőképeket a játékról, itt azonban most összefoglalom, hogy mit alkottam összeségében, és hogyan is kell használni.

Ahhoz, hogy bárki ki tudja próbálni először regisztrálni szükséges megadva egy szimpatikus felhasználónevet és jelszót. Ezek után visszalépve a főmenübe meg lehet tekinteni a használati útmutatót, a ranglistát, vagy neki is lehet állni játszani. A játék első helyszíne a futár lakása. El kell jutni az ajtóhoz, a nyilakat használva lehet irányítani a karaktert. Az ajtóba ütközve felugrik a következő játék, ahol drag ’n’ drop módszerrel a helyére kell húzni a különböző tárgyakat, azaz összepakolni, mindazt, amely elengedhetetlen a munkához. Amint az összes tárgyak sikerült bepakolni, aktívvá válik a tovább gomb, amire kattintva megjelenik a jármű választó ablak, ahol ki kell választani egy járművet, amivel a kiszállítást végezni szeretné a játékos. A választás után a térkép tárul a játékos elé, járművet szintén a nyilakkal irányítva el kell jutni az első célhoz az étteremhez, ami jelezve is van a térképen, itt kell felvenni a rendelést. Az étteremhez érve meg kell találni a pultot, ahol át lehet venni a rendelést. A pulthoz érve egy kvíz következik, ami a rendelés átvételekor szóba jövő problémák megoldását fedi le, minden kérdésnél egy darab választ lehet kiválasztani, időre kell teljesíteni. Miután megválaszolta a játékos az összeset, felugrik egy ablak, ahol amennyiben minden kérdést sikerült helyesen megválaszolni kap egy jutalmat, pl. fagyit, vagy hamburgert, amennyiben viszont volt rossz válasza, nem kap jutalmat, és megjelennek a rossz válaszai, hogy tudja mit rontott el. A tovább gombra kattintva újra a térképen találja magát, ott, ahol előzőleg a járművet hagyta, a következő állomás a vevő háza, hogy átadja neki a rendelését, ez szintén jelölve van a térképen. Az útja során lehetséges, hogy beleütközik a térképen olyan forgalmi akadályokba pl. zebra, ahol egy a KRESZ szabályokról a tudását felmérő

kvíznél vár rá. Végül a vevőnél is túl kell esni még egy kihíváson, egy felmérőn arra az esetekre, amikor a vevőnél történik valamilyen probléma. Sikeresen ki lett szállítva a rendelés, a játék véget ér, a kvíz végén felbukkanó panelről a tovább gombra kattintva visszaérkezünk a főmenübe.

A játék során végig folyamatosan mentésre kerülnek az adatok, mint pontok és kiválasztott válaszok az adatbázisba, a későbbi felhasználhatóság érdekében. Végül a PowerBI segítségével riportok formájában láthatóvá válnak a fontos statisztikák pl. a játékosok által szerzett pontok.

Amikor megterveztem a játékot még nem volt hasonló komplexségű projektről tudomásom, azonban amire nem számítottam, hogy 2022 tavaszán a FoodPanda csinált egy egészen hasonló játékot.[27]
Kipróbáltam és összehasonlítottam az én játékommal az alábbi különbségeket véltem felfedezni:
A játék indulásakor rögtön megjelenik a Toplista, ahol látható a 3 legtöbb pontot szerzett játékos, míg az én toplistámban idő is szerepel, és a top 8-at jelenítem meg, itt csak a top3 szerepel. Ezután a játék indítására kattintva bekér egy felhasználónevet, és máris egy térképen találjuk magunkat. Tehát az én programommal szemben nem igényel se regisztrációt, se jelszót. A térképen a karakter úgy van kialakítva, hogy egyértelműen egy biciklissel lehet teljesíteni a játékot, egy bal és egy jobb pedál van, amit ezek egymás utáni folyamatos nyomkodásával lehet sebességre szert tenni, azaz nincsen jármű választási lehetőség. Ezenkívül látszik felül az étterem neve és címe, ahova menni kell, valamint egy sebességmérő, egy érme számláló, és egy időzítő. A két pedált a bal és jobb nyilakkal lehet mozgatni, amint odaértünk az étterembe egy játék keretében össze kell gyűjteni a rendelés összetevőit, amik fentről esnek le a futár táskába az egér mozgatásával, figyelve arra, hogy csak, ami a rendelésben szerepel, azt kapjuk el, különben felrobban a táska. Itt is van játék végén megjelenő panel, ami mutatja az időt, és a szerzett érméket, utána tovább lehet menni a térképen a vevőhöz, akinél hasonló jellegű kvízt kell kitölteni, mint nálam itt azonban csak egy darab kérdést kell megválaszolni. Ezek után jön még egy cím és még egy vevő, a játék végén mutatja az eredményeket rögtön, és ha szeretnénk játszhatunk máris újból.

Összeségében vannak hasonlóságok a két játék között, de eltérések is, az, hogy kipróbáltam ezt a játékot fejlesztési ötleteket is adott a saját játékomhoz. Mellékletben található róla kép a 32. ábrán.

9. Továbbfejlesztési lehetőségek

Ez a projekt még nagyon sok továbbfejlesztési lehetőséget hordoz magában, mind kinézetében, mind funkcionalitásában.
Utóbbinál egyik nagyon hasznos fejlesztés lenne, ha weben keresztül bárhonnan el lehetne érni a játékot, és az adatokat, valamint, ha a képek feltöltése és tárolása valamilyen hatékonyabb módon lenne megoldva.
Kinézetileg a futár karakter animálása sokkal szebbé és szórakoztatóbbá tenné a játékot, szintén, ha a járműveken is lenne valamilyen animáció.
Ami még sokat tenne a játék élményhez, ha a városban lévő autók, valamint az étteremben lévő emberek nem fix pozícióban lennének, hanem random mozognának.
A kvízben is hasznos lenne, ha a gomboknak megváltozna a színe, a helyes vagy helytelen válaszok eltalálásánál, valamint a még tervezett játékos profiljának a megvalósítása.
A FoodPandás játékból ihletet merítve pedig egy az ottani összetevő-elkapós játékhoz hasonlót is bele lehetne építeni.

10. Összefoglalás

A szakdolgozatom során először kifejtettem, hogy mi a probléma a munkahelyemnél tapasztalt jelenlegi rendszerrel, összegyűjtöttem a problémákat, vázoltam az üzleti igényt, és hogy pontosan miért van szükség arra, amit csinálok, miben fog változást eredményezni. Feltérképeztem, hogy létezik-e már hasonló komplexségű megoldás a problémára, megállapítottam, hogy nincs. Azóta ugyan lett már hasonló játék, de egész sok mindenben eltér, és kipróbálva ihletet tudtam meríteni, az én játékomhoz is, hogy miben lehetne még jobb.

Megterveztem az adatbázis szerkezetét, a belekerülő adatokat. Kifejtettem, hogy pontosan milyen témaköröket kell lefednie a teszteknek, mik a legfontosabbak az adatok vizualizációjánál. Készítettem egy game design documentet amiben részletesen megterveztem, hogyan fog kinézni a játék milyen elemei lesznek és hogyan fog működni.

Utánajártam, hogyan, milyen módszerekkel, eszközökkel és nyelvekkel tudom megvalósítani a projektet, amit végül egy kivétellel a terv szerint is követtem el, tehát C# nyelven programoztam, Unity platformot és Visual Studiot használva, és Power BI-t az adatvizualizációhoz, az adatbázisom végül MySQL lett MSSQL helyett, ugyanis a megvalósítás közben rá kellett jönnöm, hogy azzal sokkal hatékonyabban meg tudom csinálni, jobban tud együttműködni a Unityvel. Nagyon sok új dolgot tanultam a megvalósítás közben, korábban még nem foglalkoztam Unityvel egyáltalán, de számtalan kiváló cikk és tutorial van hozzá, ami az egyik fontos szempontom volt a kiválasztásánál, és be is váltak. A Unity saját tudásbázisa is nagyon jól elmagyaráz mindent, könnyen használhatónak találtam a felületét másrészt a Unity saját függvényeit, nagy segítség volt a játékom felépítésében. Másik újdonság, amivel előre nem számoltam, de végül muszáj volt alkalmaznom, hogy az adatbázissal való kommunikáció nem úgy működik, ahogy gondoltam, C# oldalról nem tudtam egyszerűen összekötni, hanem PHP scripteket is alkalmaznom kellett hozzá, amivel szintén nem volt még tapasztalatom korábban, így ezt is a játék készítése közben igyekeztem elsajátítani.

Végül idő hiányában nem sikerült mindent megvalósítani abból, amit elterveztem pl. profilt nem csináltam, de úgy gondolom, hogy nagyrészt sikerült elérnem, amit akartam, ami a cél volt, van egy játékom, ami egyben szórakoztató, másrészt tudást is átad, és felmér, minden adat elmentésre kerül közben egy adatbázisba, amit elérve a Power BI megmutatja könnyen átlátható kimutatásokkal az eredményeit. Számtalan értékes tapasztalatot és tudást szereztem a szakdolgozat készítése alatt.

11. Irodalomjegyzék

[1] Best database
https://hevodata.com/learn/best-database/
Utoljára megtekintve: 2021.11.20.

[2] BI Platform Magic Quadrant
https://www.gartner.com/doc/reprints?id=1-1YOXON7Q&ct=200330&st=sb
Utoljára megtekintve: 2021.11.20.

[3] Databases 2021 top 10
https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba
Utoljára megtekintve: 2021.11.20.

[4] Databases 2020 top 7
https://www.geeksforgeeks.org/top-7-databases-to-learn-in-2021/
Utoljára megtekintve: 2021.11.20.

[5] Driving school game 19
https://hvg.hu/tudomany/20090327_3d_driving_School_letoltes
Utoljára megtekintve: 2021.11.25.

[6] Game development languages top 4
https://content.techgig.com/5-preferred-programming-languages-for-game-development/articleshow/84677550.cms
Utoljára megtekintve: 2021.11.25.

[7] Game development languages top 8
https://analyticsindiamag.com/top-8-programming-languages-for-game-developers/
Utoljára megtekintve: 2021.11.25.

[8] Game development languages top 10
https://www.analyticsinsight.net/top-10-programming-languages-for-game-development-2021/
Utoljára megtekintve: 2021.11.25.

[9] Gamification apps
https://www.developgoodhabits.com/gamification-apps/
Utoljára megtekintve: 2021.11.25.

[10] Gartner Magic Quadrant

https://www.gartner.com/en/research/methodologies/magic-quadrants-research
Utoljára megtekintve: 2021.11.20.

[11] KRESZ játék
https://www.startlapjatekok.hu/jatekok/felismered_az_osszes_kresz_tablat_/
Utoljára megtekintve: 2021.11.25.
[12] Netpincér vs. Wolt
https://junkieeeeboy.blog.hu/2020/12/18/netpincer_vs_wolt
Utoljára megtekintve: 2021.11.20.

[13] Oracle vs. MS SQL
https://www.educba.com/oracle-vs-mssql/
Utoljára megtekintve: 2021.11.20.

[14] O'Reilly.: Unity Game Development Cookbook. *O'Reilly Media, Inc.,* 2019

[15] Power BI
https://docs.microsoft.com/hu-hu/power-bi/
Utoljára megtekintve: 2021.11.20.

[16] Power BI vs. Tableau
https://www.educba.com/power-bi-vs-tableau/
Utoljára megtekintve: 2021.11.20.

[17] Python vs. C++ vs. C#
https://dzone.com/articles/c-vs-python-vs-c-for-game-development
Utoljára megtekintve: 2021.11.25.

[18] SQL vs. NoSQL
https://www.ibm.com/cloud/blog/sql-vs-nosql
Utoljára megtekintve: 2021.11.20.

[19] Szirmay-Kalos, László., Antal, György., Csonka, Ferenc.: Háromdimenziós grafika, animáció és játékfejlesztés. *Computerbooks*, 2003

[20] Tableau
https://www.tableau.com
Utoljára megtekintve: 2021.11.20.

[21] Unity
https://dotnet.microsoft.com/apps/games/unity

Utoljára megtekintve: 2021.11.25.

[22] Unity overview
https://docs.unity3d.com/Manual/overview-of-dot-net-in-unity.html
Utoljára megtekintve: 2021.11.25.

[23] XNA framework
https://www.microsoft.com/en-us/download/details.aspx?id=15163
Utoljára megtekintve: 2021.11.25.

[24] Shutterstock
https://www.shutterstock.com/hu/home
Utoljára megtekintve: 2021.12.08.

[25] Unity Manual
https://docs.unity3d.com/Manual/
Utoljára megtekintve: 2022.05.04.

[26] Unity Tutorials
https://learn.unity.com/tutorial/update-and-fixedupdate#5c8a4242edbc2a001f47cd63
Utoljára megtekintve: 2022.05.04.

[27] Pandarun
https://pandarun.foodpanda.hu/play
Utoljára megtekintve: 2022.05.04.

[28] Vecteezy
https://www.vecteezy.com
Utoljára megtekintve: 2022.05.04.

[29] Meme arsenal
https://www.meme-arsenal.com/en/create/meme/1333412
Utoljára megtekintve: 2022.05.04.

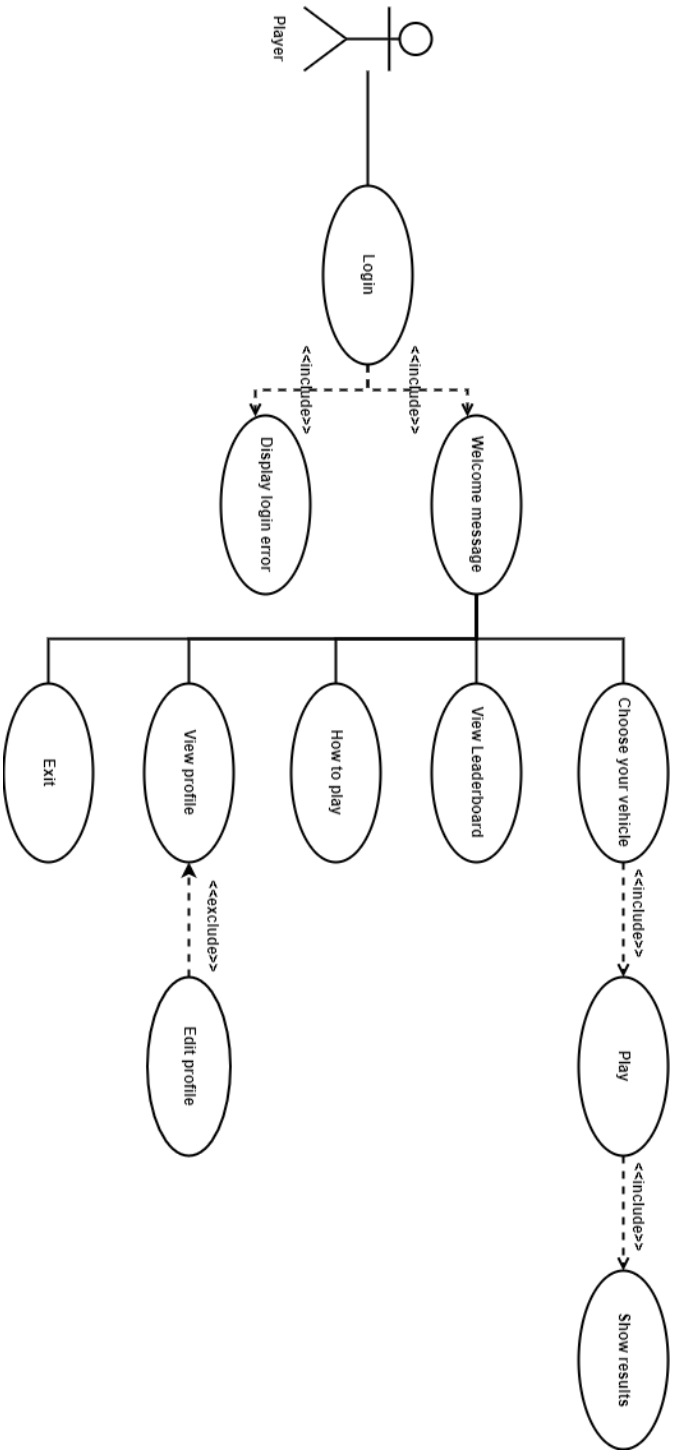
12. Mellékletek

```
$con = mysqli_connect('localhost', 'root', '', 'gamedb');

    if(mysqli_connect_errno())
    {
    echo "Connection failed";
    exit();
    }

    $username = $_POST["name"];
    $password = $_POST["password"];

    $getdataquery = "SELECT UserName, Salt, Hash, Score FROM players WHERE
UserName='" . $username . "';";

    $run = mysqli_query($con, $getdataquery) or die("Get data query failed");

    if(mysqli_num_rows($run) != 1)
    {
        echo "There is no user with that name/ there is more than one user
with that name";
        exit();
    }

    $result = mysqli_fetch_assoc($run);
    $salt = $result["Salt"];
    $hash = $result["Hash"];

    $loginhash = crypt($password, $salt);
    if($hash != $loginhash)
    {
        echo "Incorrect password";
        exit();
    }
```

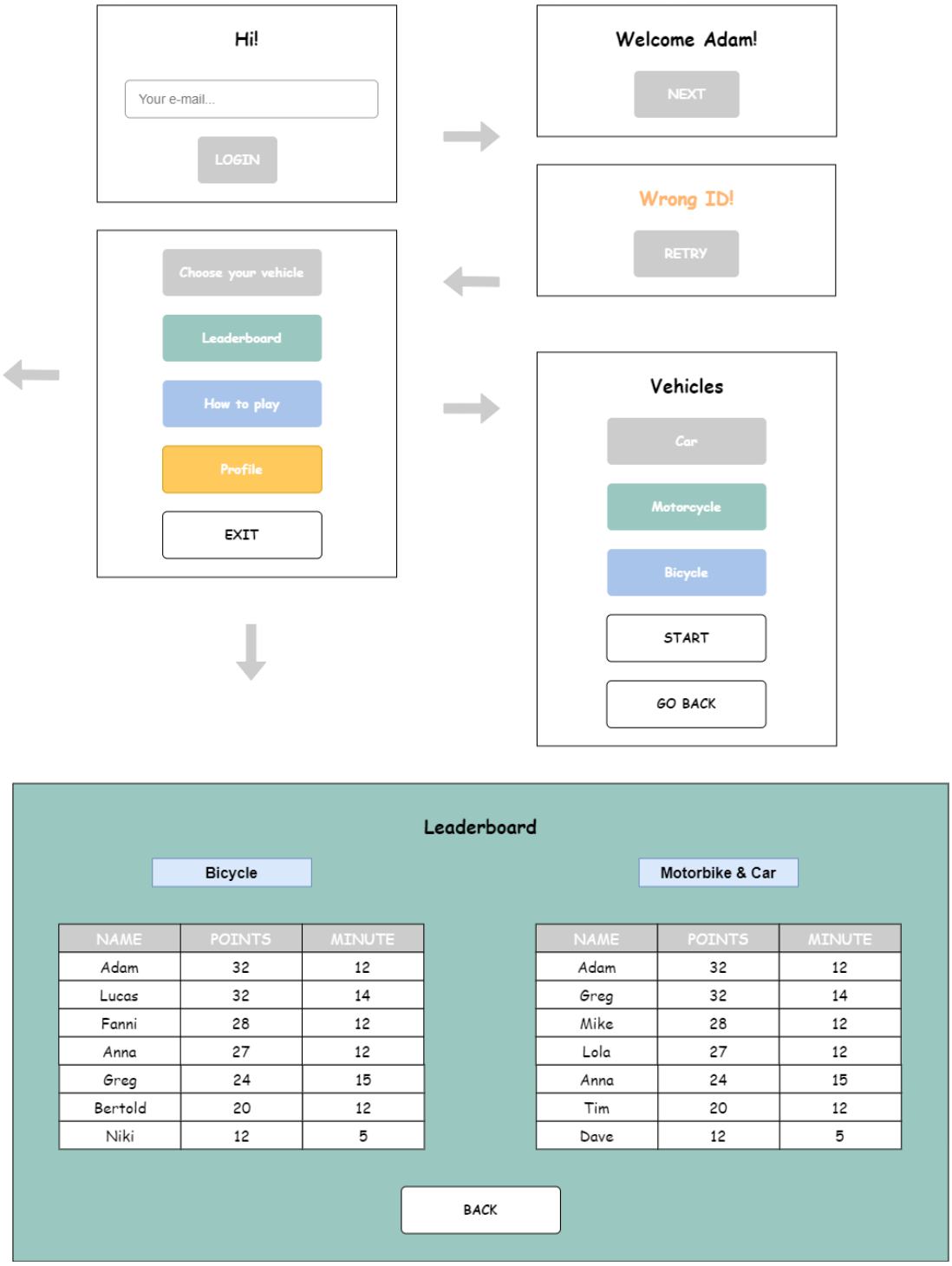
6. kódrészlet: Login PHP

```
if (currentTime > 0)
        {
            if (DBManager.currentScene == "restaurant")
            {
                if (DBManager.RestaurantQuestions.Count > 0)
                {
                    Invoke(nameof(SelectQuestion), 0.4f);
                }
                else
                {
                    timerIsRunning = false;
                    DBManager.score = scoreCount;
                    DBManager.totalPoints += DBManager.score;
                    var timetofinish = timeLimit - currentTime;
                    TimeSpan ttf = TimeSpan.FromSeconds(timetofinish);
                    DateTime ttfdatetime = new DateTime(ttf.Ticks);
                    string ttfstring = ttfdatetime.ToString("HH:mm:ss");
                    DBManager.restaurantQuizTimeToFinish = ttfstring;
                    Debug.Log(DBManager.restaurantQuizTimeToFinish + " score is:
" + DBManager.score);


                    foreach (var item in results)
                    {
                        if (!item)
                            getsAReward = false;
                    }

                    if (getsAReward)
                    {
                        DBManager.rewardID = UnityEngine.Random.Range(2, 6);
                    }
                    else
                    {
                        DBManager.rewardID = 1;

                    }

                    StartCoroutine(GetRewardImage(DBManager.rewardID));
                    wrongAnswerList.text = wrongAnswers;
                    loadQuizData.GameOverPanel.SetActive(true);

                    Debug.Log("Az ő rewardja: " + DBManager.rewardID);
                    CallSaveResults(DBManager.restaurantQuizTimeToFinish);
                }
            }
```

22. kódrészlet: Kvíz vége ellenőrzés

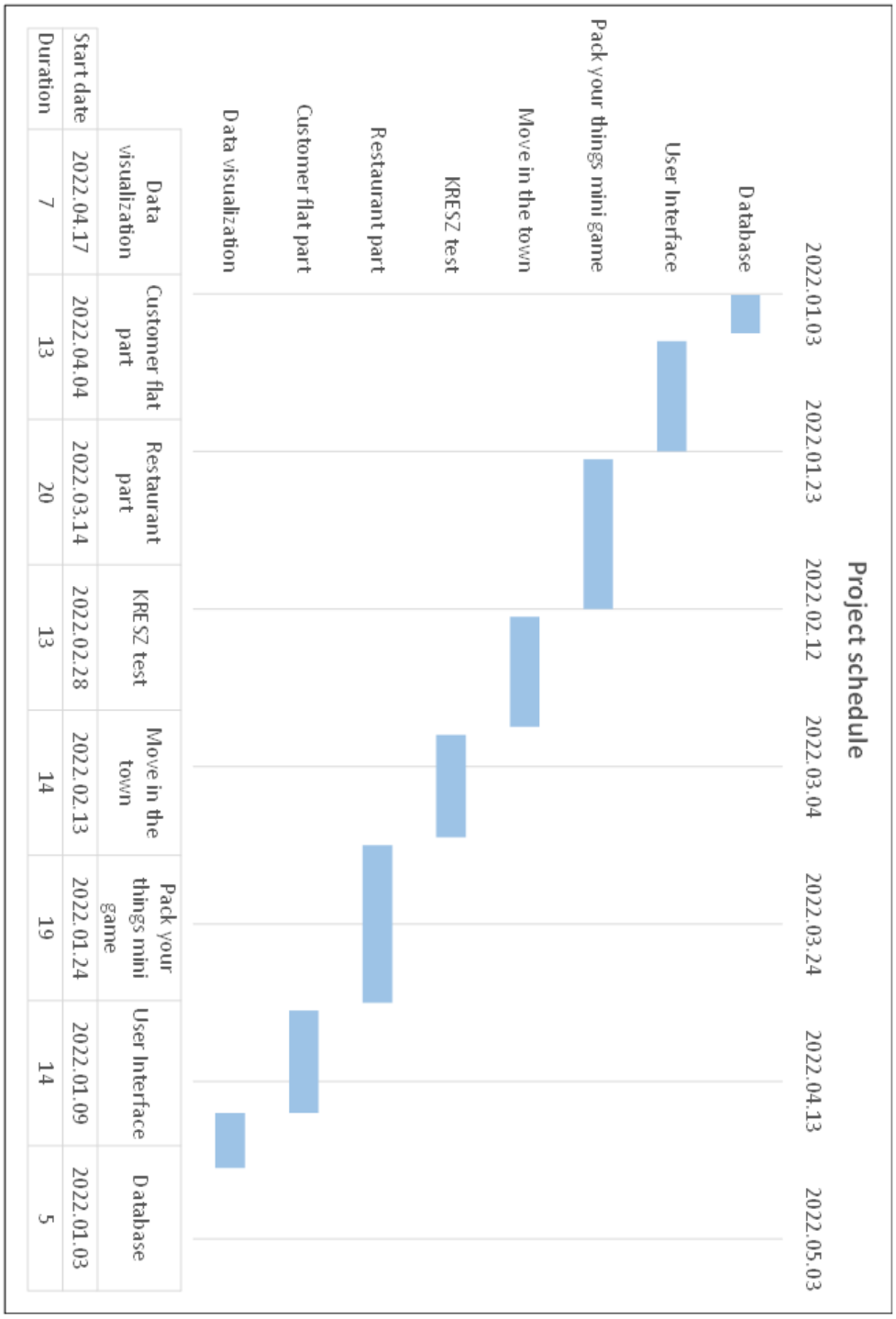


6. ábra: Use case diagram



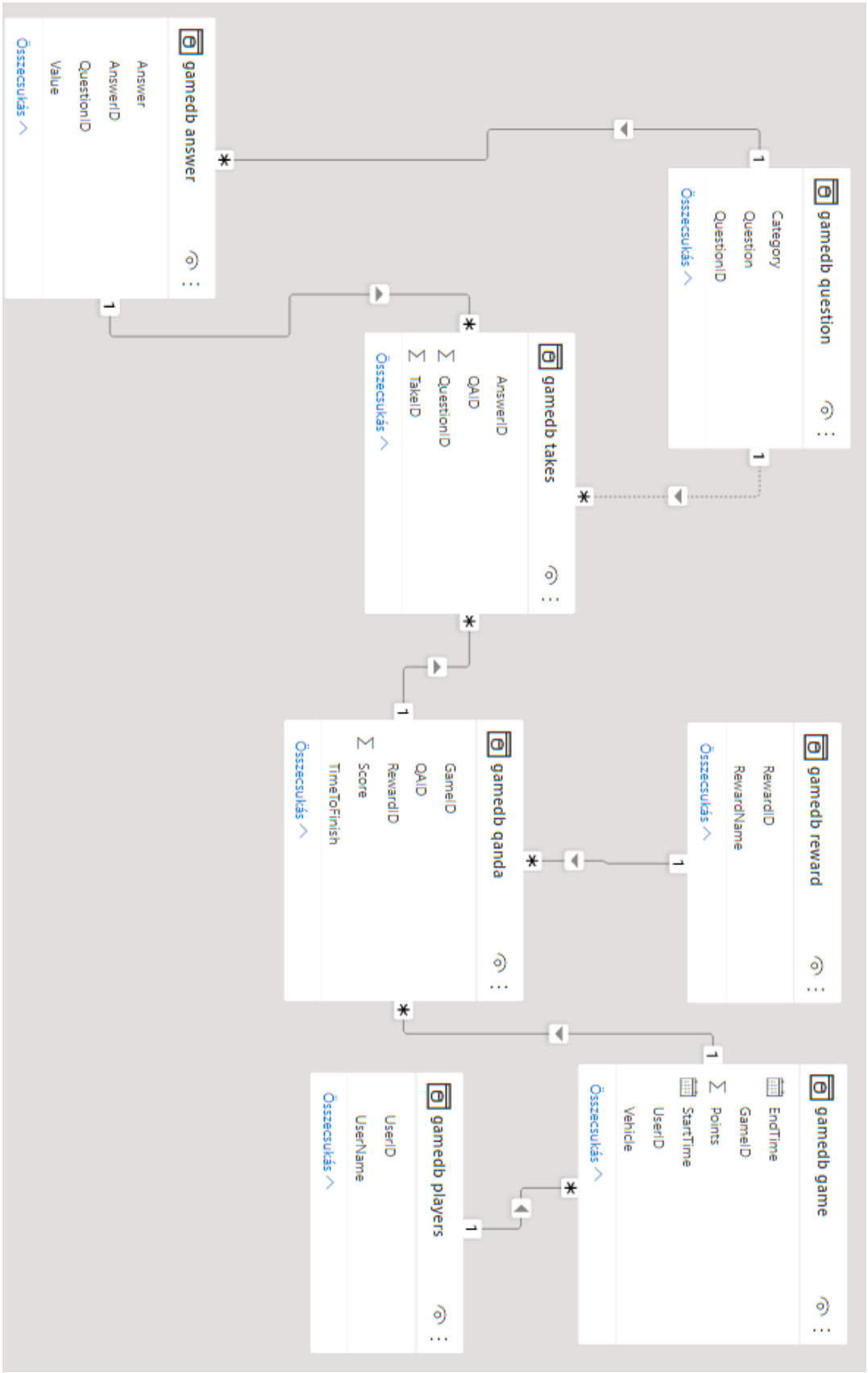
7. ábra: Wireframe

8. ábra: Wireframe

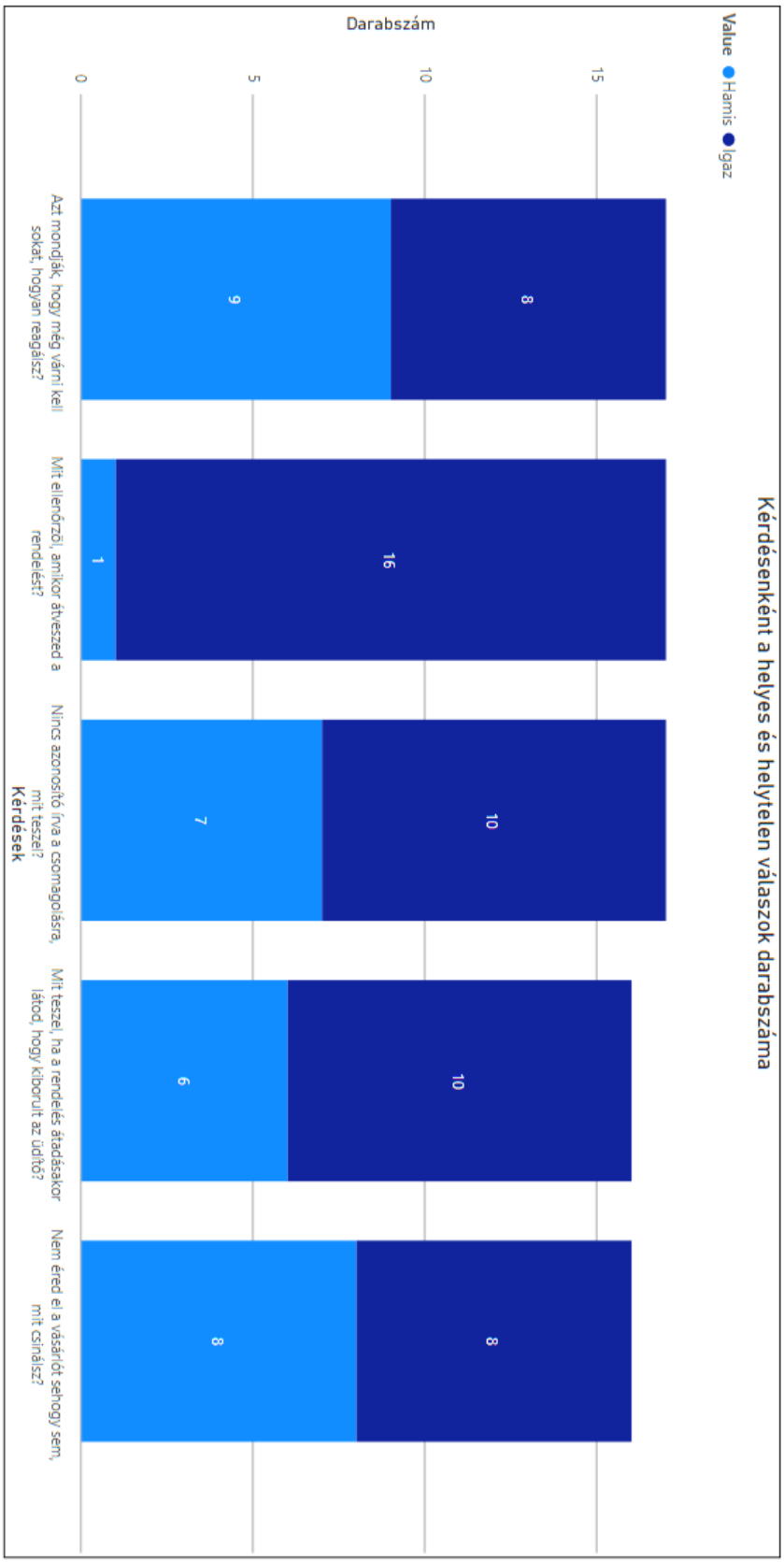


	Data visualization	Customer flat part	Restaurant part	KRESZ test	Move in the town	Pack your things mini game	User Interface	Database
Start date	2022.04.17	2022.04.04	2022.03.14	2022.02.28	2022.02.13	2022.01.24	2022.01.09	2022.01.03
Duration	7	13	20	13	14	19	14	5

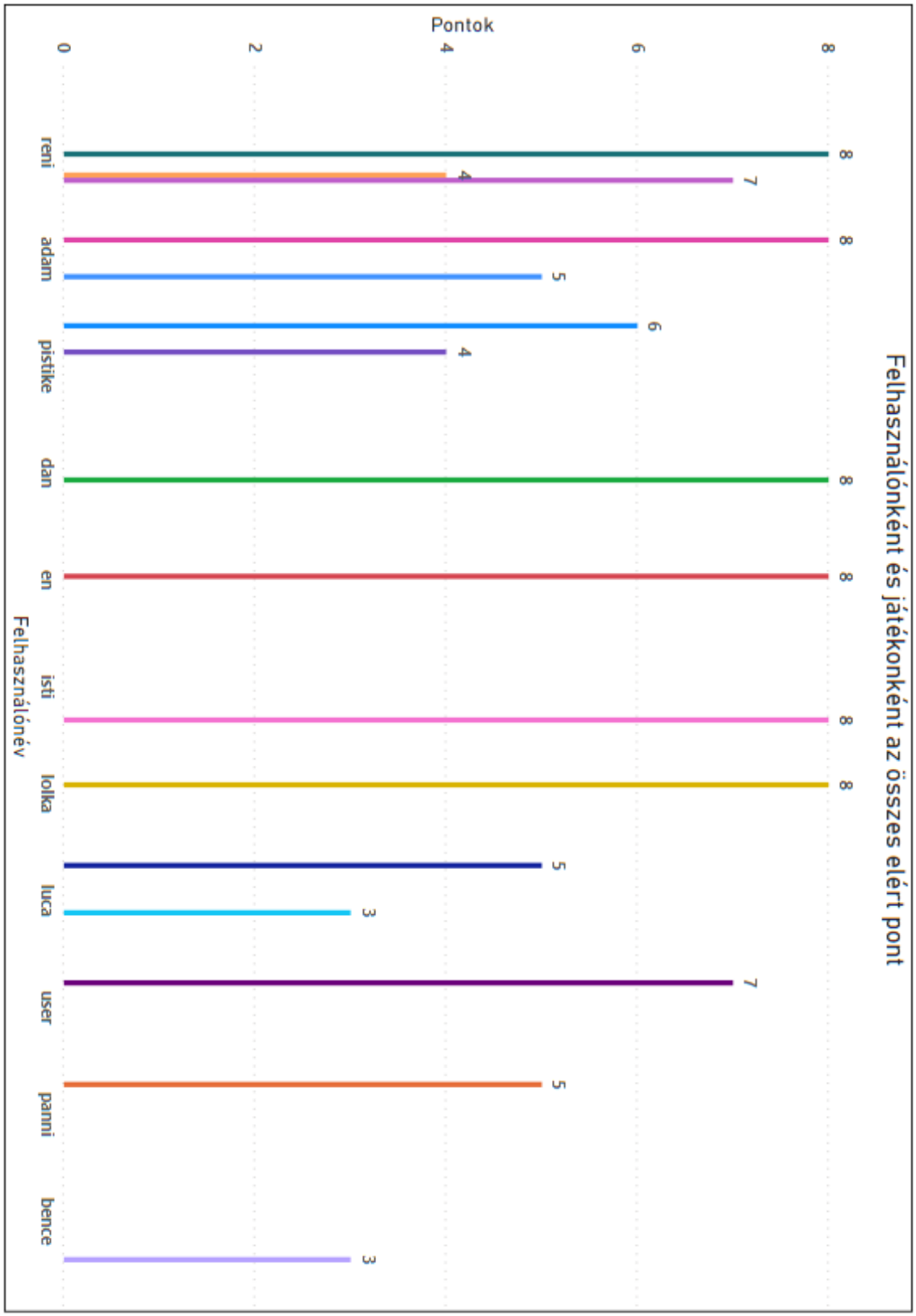
9. ábra: Ütemezés - teljes projekt



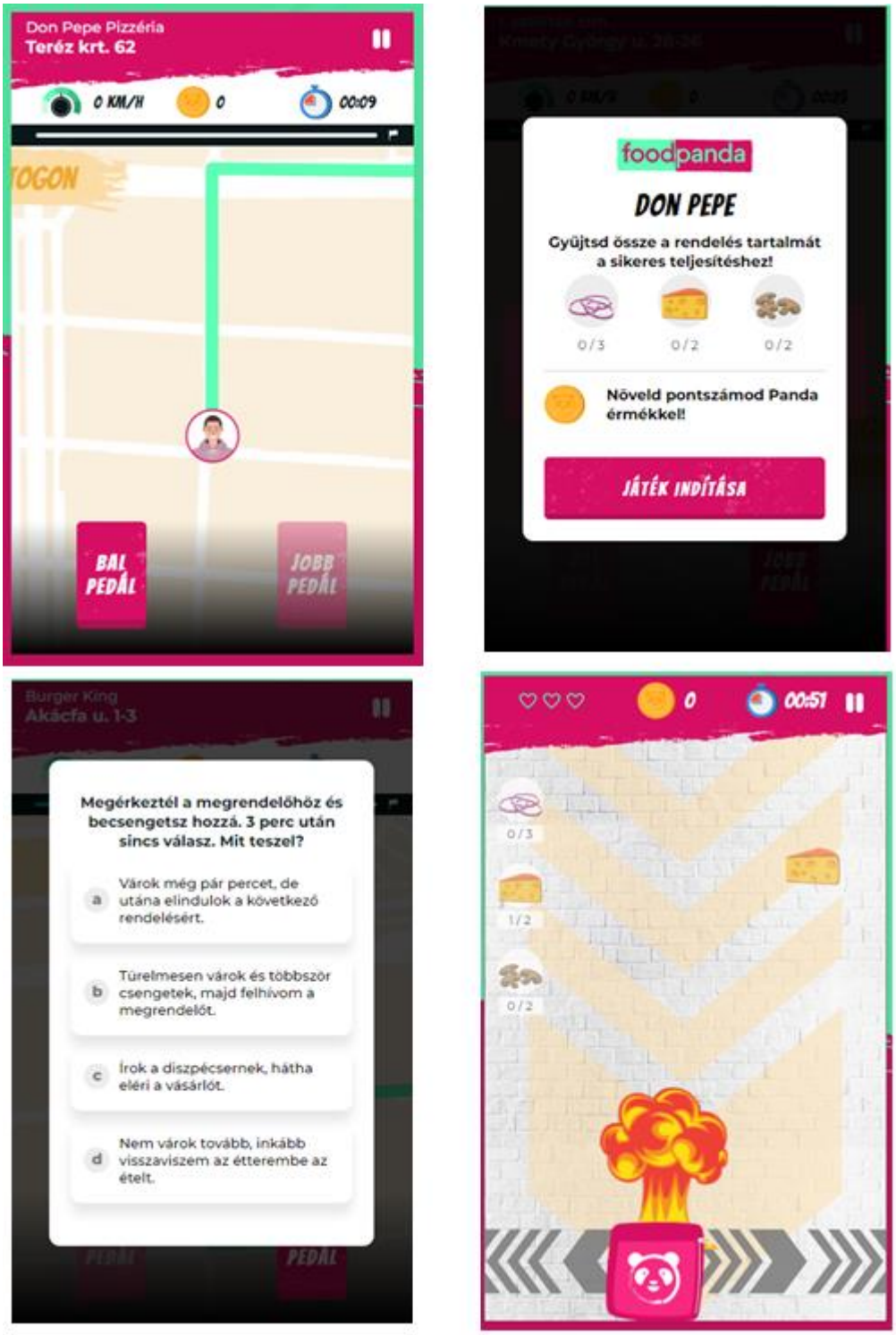
29. ábra: PowerBI adatbázis



30. ábra: Riport kérdések



31. ábra: Riport pontok



32. ábra: Foodpanda