Legyen ön is milliomos

Tökös Ötös

Tartalom

1. Projekt Alapító dokumentum

1.1 A projekt célja

A projekt célja egy Legyen ön is milliomos játék publikálása 2 platformon (Android és web), illetve demózás IOS platformon. Regisztráció kezelése, adatok tárolása adatbázisban, érzékeny adatok titkosítva. Cordova keretrendszerrel UI megvalósítása. Backend megvalósítása PHP szerverrel.

1.2 Kritériumok

1. Szerver kliens kapcsolat felépítése
2. Dizájn kialakítás (mobil és webes platformon)
3. Webes és Android-os verzió kiadása
4. IOS verzió demózása emulált platformon (pénzügyi megtakarítás miatt)

1.3 Projekt tagok

Név	Szervezeti egység	Beosztás	Lakcím	Elérhetőség
Bajnok Tamás	Informatikai osztály	Scrum Master	6320 Solt Hatház u. 5	bajnoktomi96@gmail.com
Nagy Ádám	Informatikai osztály	Product Owner	2400 Dunaújváros Dózsa György u. 37	nagy.adam0901@gmail.com
Szabó Attila	Informatikai osztály	Frontend Developer	2451 Ercsi Semmelweis u. 33	maxxywarrior@gmail.com
Tóth Tamás	Informatikai osztály	Backend Developer	2400 Dunaújváros Dózsa György u. 33	ttamas0713@gmail.com
Varsa László	Informatikai osztály	Frontend Designer	2400 Dunaújváros Dózsa György u. 37	vlz94@citromail.hu

1.4 Projektvezető kinevezése

Ezen dokumentummal kijelentjük, hogy a projekt vezetésével Nagy Ádámot (2400 Dunaújváros Dózsa György u. 37) bízzuk meg. A projektvezető aláírásával igazolja, hogy vállalja a projekt vezetésével, szervezésével kapcsolatos teendőket:

- feladatok kiosztása
- munkálatok koordinálása
- kapcsolattartás a csapattagokkal

- a támasztott kritériumok ellenőrzése

1.5 Projekt scope kijelölése

- Adatbázis modell elkészítése
- Reláció sémák létrehozása
- Adatbázis feltöltése
- PHP szerver telepítése, üzemben tartása
- Backend kialakítása PHP-val
- Frontend kialakítása HTML, CSS és JavaScript felhasználásával
- Grafikus felület kialakítása
- Részletes dokumentáció

1.6 Out of Scope tevékenységek

- A mobil alkalmazások nyilvános terjesztése

1.7 Projekt nagyobb összefüggésbe helyezése, kapcsolata más projektekkel

Nem függ össze.

1.8 A projekt legfontosabb kritériumai

- A szerver rendelkezésre állási ideje legyen legalább 99,97%-os
- A szerver ellenállóvá tétele külső behatolási kísérletekkel szemben
- Az adatbázis ne tartalmazzon redundanciát
- Egységes dizájn webes és mobilos felületen is
- Akadásmentes és hibamentes futás
- Felhasználói adatok titkosított kezelése
- Kitűzött mérföldkövek teljesítése

1.9 Fontosabb üzleti határidők

- Csoportok adatainak feltöltése: - 2019.03.04. 20:00
- Projekt Alapító Dokumentum feltöltése: - 2019.03.18. 20:00
- Végleges alkalmazás megjelenése – 2019.04.28
- Dokumentáció átadása – 2019.05.04

1.10 A projekt várható befejezési ideje Mérföldkövek

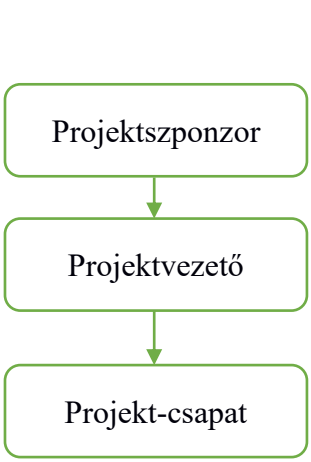
- Feladatok kiosztása – 2019.03.03
- Adatbázis megtervezése – 2019.03.10
- Szerver felállítása – 2019.03.17
- Felhasználói felület megtervezése – 2019.03.17
- Backend létrehozása – 2019.03.24
- Frontend létrehozása –2019.03.24
- GUI kialakítása – 2019.04.21
- Alkalmazás alfa verzió megjelenése – 2019.03.24
- Alkalmazás béta verzió megjelenése – 2019.04.14

- Végleges alkalmazás megjelenése – 2019.04.28
- Dokumentáció átadása – 2019.05.04

1.11 Várható befejezés

- **Folyó év ötödik havának ötödik napján**

1.12 Szolgálati út (a projekt szervezeti felépítése)



1.13 Költségek becslése

Típus	Egység	Egységár	Teljes összeg
Materiális költségek			
Eszközök			
Számítógépek Hálózati eszközök	5 4	250.000Ft 60.000Ft	1.250.000Ft 240.000Ft
Immateriális költségek			
Apple Fejlesztői díj	1	27.0000	27.000 Ft
Bérek	Hónapszám		
Bajnok Tamás Nagy Ádám Szabó Attila Tóth Tamás Varsa László	2 2 2 2 2	750000 Ft 750000 Ft 650000 Ft 650000 Ft 650000 Ft	1.500.000 Ft 1.500.000Ft 1.300.000 Ft 1.300.000 Ft 1.300.000 Ft
Összköltség			8.417.000Ft

1.14 Kompetencia mátrix

	K1	K2	K3	K4	K5	K6	K7	K8
Bajnok Tamás	X	X	X	X	X	X		
Nagy Ádám			X			X	X	
Szabó Attila				X	X	X		
Tóth Tamás	X	X				X		
Varsa László					X	X		X

K1= adatbázis kialakítás
K2= adatbázis feltöltés
K3= PHP szerver
K4= CSS
K5= HTML
K6= JS
K7= Csapat szervezés
K8= Grafikus programok

1.15 Tevékenység felelős mátrix

	Adatbázis tervezés	Adatbázis Feltöltése	PHP szerver üzemeltetése	Backend kialakítás	Frontend kommunikáció	Grafikus felület kialakítása	Dokumentálás	Látványtervek elkészítése
Bajnok Tamás	**J**	**J**	**V**	**V**	**J**		**I**	
Nagy Ádám	**B/I**	**B/I**	**J/I**	**J/I**	**I**	**J/I**	**J**	**J**
Szabó Attila					**V**	**B**	**V**	**I**
Tóth Tamás	**V**	**V**	**B**	**B/I**			**I**	
Varsa László					**B**	**V**	**I**	**V**

V - Végrehajtási felelősség. Az érdekelt felelőssége a munka elvégeztetése. Nem feltétlenül hoz döntéseket, de a csoportot arra sarkallja, hogy mindig időben hozzák meg a döntéseket.

J - Jóváhagyási jogkör. Végleges hozzájárulás a tevékenység kimenetelének elfogadásához. Döntéseket hoz.

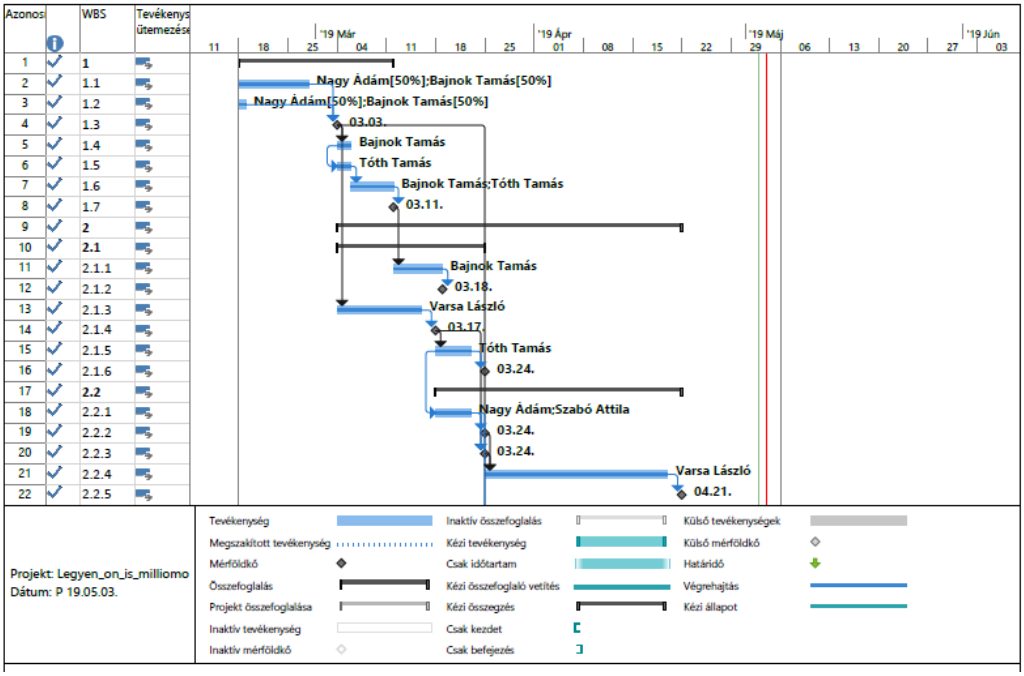
B - Meg kell beszélni. A munka elvégzése során az érdekelt ad információt. Nem hoz döntéseket, de a döntések előtt tanácsot kérnek tőle.

I - Informálni kell, ha döntés született. Mindig naprakész szeretne lenni ennek a tevékenységnek az előrehaladásáról.

1.16 Kommunikációs terv

	Információ igény	Gyakoriság	Kommunikációs csatorna	Válasz
Projektszponzor	Kész projekt	projektenként	beszámoló, moodle	3 nap
Projekt vezető	Folyamat	2 nap	Messenger	1 nap
Fejlesztőcsapat	Projekt elvárások	5 nap	Messenger	1 nap
Csoportgyűlés	Előrehaladás, esetleges problémák	Hetente	Jelentés	azonnali

1.17 Gantt-diagram



1.18 Tevékenységre bontás (WBS)

WBS	Tevékenység ütemezése	Tevékenység neve	Időtartam	Kezdés	Befejezés	Megelőzők
1	**Automatikusan ütemezett**	**Tervezés**	**16 nap**	**H 19.02.18.**	**H 19.03.11.**	
1.1	Automatikusan ütemezett	Csapat szervezés	8 nap	H 19.02.18.	Sze 19.02.27.	
1.2	Automatikusan ütemezett	Feladat kiosztás	1 nap	H 19.02.18.	H 19.02.18.	
1.3	Automatikusan ütemezett	Feladatok kiosztása – Elkészül	0 nap	V 19.03.03.	V 19.03.03.	2;3
1.4	Automatikusan ütemezett	Adatbázis modell elkészítése	2 nap	H 19.03.04.	K 19.03.05.	4
1.5	Automatikusan ütemezett	Reláció sémák létrehozása	2 nap	H 19.03.04.	K 19.03.05.	5KK
1.6	Automatikusan ütemezett	Adatbázis tervezés	4 nap	Sze 19.03.06.	H 19.03.11.	6
1.7	Automatikusan ütemezett	Adatbázis megtervezése – Elkészül	0 nap	H 19.03.11.	H 19.03.11.	7
2	**Automatikusan ütemezett**	**Alkalmazás fejlesztése**	**35 nap**	**H 19.03.04.**	**V 19.04.21.**	
2.1	**Automatikusan ütemezett**	**Backend**	**15 nap**	**H 19.03.04.**	**V 19.03.24.**	
2.1.1	Automatikusan ütemezett	PHP szerver telepítése, üzemben tartása	5 nap	K 19.03.12.	H 19.03.18.	8
2.1.2	Automatikusan ütemezett	Szerver felállítása – Elkészül	0 nap	H 19.03.18.	H 19.03.18.	11
2.1.3	Automatikusan ütemezett	Felhasználói felület tervezése	10 nap	H 19.03.04.	P 19.03.15.	4
2.1.4	Automatikusan ütemezett	Felhasználói felület megtervezése – Elkészül	0 nap	V 19.03.17.	V 19.03.17.	13
2.1.5	Automatikusan ütemezett	Backend kialakítása PHP-val	5 nap	H 19.03.18.	P 19.03.22.	14
2.1.6	Automatikusan ütemezett	Backend létrehozása – Elkészül	0 nap	V 19.03.24.	V 19.03.24.	15

2.2	**Automatikusan ütemezett**	**Frontend**	**25 nap**	**H 19.03.18.**	**V 19.04.21.**	
2.2.1	Automatikusan ütemezett	Frontend kialakítása HTML, CSS és JavaScript felhasználásával	5 nap	H 19.03.18.	P 19.03.22.	15KK
2.2.2	Automatikusan ütemezett	Frontend létrehozása –Elkészül	0 nap	V 19.03.24.	V 19.03.24.	18
2.2.3	Automatikusan ütemezett	Alkalmazás alfa verzió megjelenése – Megtörténik	0 nap	V 19.03.24.	V 19.03.24.	18;14
2.2.4	Automatikusan ütemezett	Grafikus felület kialakítása	20 nap	H 19.03.25.	P 19.04.19.	19;20
2.2.5	Automatikusan ütemezett	GUI kialakítása – 2019.04.21	0 nap	V 19.04.21.	V 19.04.21.	21
3	**Automatikusan ütemezett**	**Véglegesítés**	**30 nap**	**H 19.03.25.**	**V 19.05.05.**	
3.1	Automatikusan ütemezett	Adatbázis feltöltése	5 nap	H 19.03.25.	P 19.03.29.	4;18
3.2	Automatikusan ütemezett	Alkalmazás béta verzió megjelenése – 2019.04.14	0 nap	V 19.04.14.	V 19.04.14.	24
3.3	Automatikusan ütemezett	Tesztelés hibák javítása	10 nap	H 19.04.15.	P 19.04.26.	25
3.4	Automatikusan ütemezett	Végleges alkalmazás megjelenése – 2019.04.28	0 nap	V 19.04.28.	V 19.04.28.	26
3.5	Automatikusan ütemezett	Részletes dokumentáció	3 nap	Sze 19.05.01.	P 19.05.03.	
3.6	Automatikusan ütemezett	Dokumentáció átadása - Megtörténik	0 nap	V 19.05.05.	V 19.05.05.	28

1.19 Költség haszonelemzés

A jelenleg piacon lévő applikációkkal nagy előnyt képviselünk a platformfüggetlenség megléte miatt. Átlagosan 10,000-20,000 letöltés szám jellemző ezen alkalmazásokra. Amennyiben ingyenes applikációként publikálásra kerül az alkalmazás elmondható, hogy ekkora letöltés szám mellet már egy platform használata mellet is jövedelmező a projekt a reklámtartalmak miatt.

1.20 Megvalósíthatósági tanulmány

Az Apple fejlesztői díj kivételével a projekt megvalósítható, belátható időn belül nyereségessé formálható.

1.21 Kritikus út

A költségminimalizálás és a rendelkezésre álló stabil munkaerőre építve egy veszélyes, ugyanakkor sikeresnek bizonyuló megvalósítási terv jött létre. A kimutatásokból látszik minden tevékenységünk kritikus úton helyezkedett el egyes csúszások a projekt komoly csúszását idézhették volna elő, ugyanakkor bizonyítottan sikeres volt a projekt mivel csúszás nélkül elkészült a megjelölt határidőre!

1.22 Marketing

A projekt nem tartalmazott marketinget, a későbbiek folyamán egy új projekt keretből megvalósítható a termék reklámja és publikálása.

2. Bevezetés

A projekt célja egy Legyen ön is milliomos játék publikálása 2 platformon (Android

és web), illetve demózás IOS platformon. Regisztráció kezelése, adatok tárolása adatbázisban, érzékeny adatok titkosítva. Cordova keretrendszerrel UI megvalósítása. Backend megvalósítása PHP szerverrel.

A projekt sikere nagyban múlik az alap ötlet megvalósíthatóságán, ugyanakkor nagyban függ a projektbe bekapcsolódó emberek mennyire tudnak egy csapatként küzdeni a cél megvalósításának érdekében.

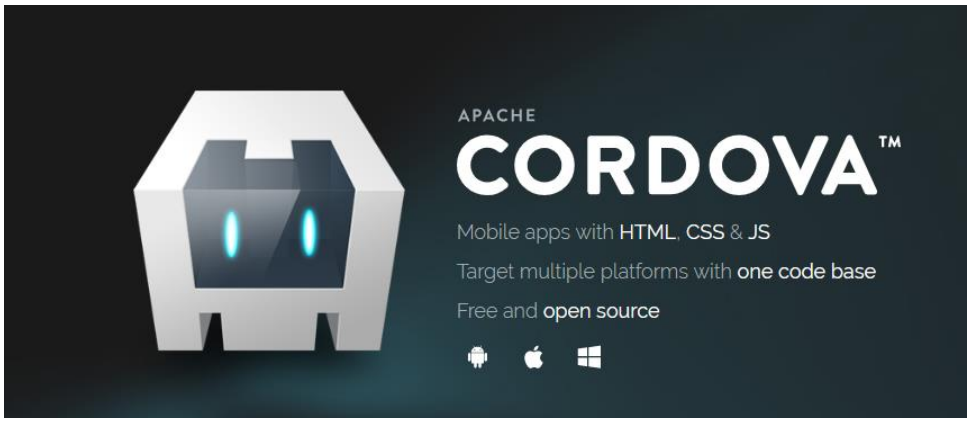
Első lépés ötlet keresés, mely során mérlegelni kell a felmerülő ötlet kivitelezhetőségét, egy eleve rosszul előadott ötletre nehezebb csapatot építeni. A Legyen ön is milliomos önmagában nem egyedi ötlet van már a piacon számtalan verziója sok csapat megvalósította már. Ami ezen tények ellenére is sikeressé tudták tenni az ötletet az a megvalósítási technológia, mely forradalmi és jelen pillanatban kevéssé kiaknázott a fejlesztők körében. Sok fejlesztő környezet és eszköz létezik, de kevés van melynek használatával felhasználhatóvá lefordíthatóvá válik a kód a két legjobban elterjedt mobil operációs rendszerre, ugyanakkor a számítógépet előnyben részesítőket is a kód gyökeres átgondolása nélkül megszólíthat a fejlesztő csapat. Ezen nagy lefedettség tudatában alakult ki a projekt ötlete véglegesen.

Második lépés a csapat kialakítása. Létfontosságú a jó csapat összetétele, a projekt sikerességéhez szükséges a stabil háttér rendszer mivel az applikáció szívét adja, ezen rendszer a kritikus pontja a projekt sikerességének, így elengedhetetlen egy a hálózatokat jól ismerő csapattag. Ugyanakkor egy embernek ezen feladat az adattárólástól a szerver oldali programozáson át nagy feladat így a projekten két ember is megkapta ezen feladatkört. Szükséges a játék dinamikáját és a háttér rendszer közötti részt is jól megvalósítani ezen hiányosságok szintén a teljes projekt vesztét okoznák, így a JS feladatokat szintén két ember kapta meg faladatául. Már csak egy rész nem került említésre ugyanakkor ez is ugyanannyira fontos, mint az előző két rész az applikáció megjelenése mely CSS segítségével lett kialakítva az alap HTML kódból, erre a feladatra egy ember lett megbízva a csapatból. A csapattagok csatlakozása után kezdetét vehette a megvalósítás megtervezése.

3. A tervezés

Egy applikáció fejlesztése nehéz feladat, mivel nehéz döntéseket kell meghozni még a program megírása előtt. Elsősorban fel kellett mérni a csoport preferenciáit. Ez alatt programozási nyelvismeret, integrált fejlesztői környezet (IDE) [1] kiválasztásán volt a legnagyobb hangsúly. Emellett el kellett dönteni, hogy milyen erőforrásra volt szükség. Egy webes / Android applikáció legfőbbként 2 részre osztható: Backend, Frontend. Ezeket a részeket több alrészre is kellett osztani, annak érdekében, hogy megosztott legyen a csapat erőforrása. Személyes megbeszélés során a csapat egyet értett, hogy Cordova keretrendszer lenne a legmegfelelőbb az applikáció fejlesztésére. Emellett az IDE, amit mindenki ismert az a Visual Studio. Egyéb lehetőségek között lehetett volna a Netbeans is. A fejlesztéshez szükséges többi erőforrás: Scrum Master, Product Owner, Frontend Developer, Backend Developer, Frontend Designer.

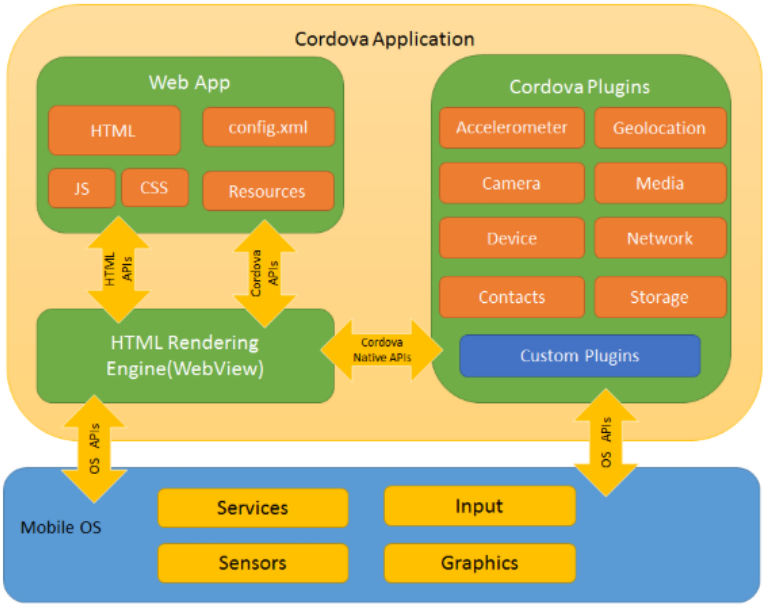
3.1 Cordova



„Az Apache Cordova egy nyílt forráskódú mobilfejlesztési keret. Ez lehetővé teszi a szabványos webes technológiák - HTML5, CSS3 és JavaScript - használatát a platformok közötti fejlesztéshez. Az alkalmazások az egyes platformokra célzott csomagolásokon belül futnak, és a szabványoknak megfelelő API-kötésekre támaszkodnak az egyes eszközök képességeihez, például érzékelőkhez, adatokhoz, hálózati állapothoz stb.”[2]

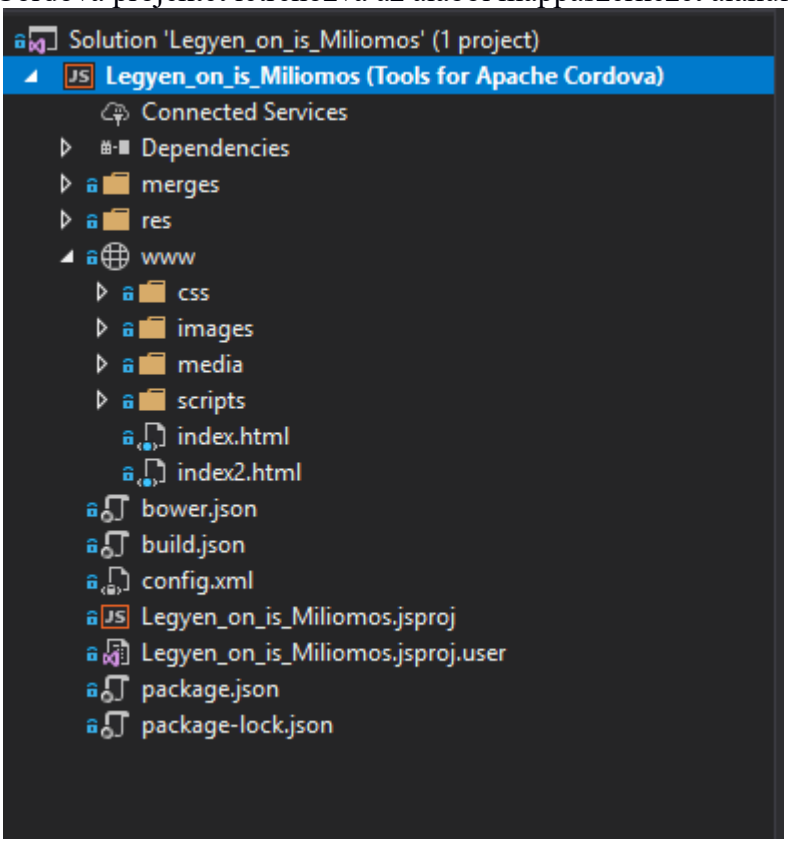
[1] https://tudasbazis.sulinet.hu/hu/szakkepzes/informatika/a-program-gyakorlati-megvalositasa/szamitogepes-programozas-a-gyakorlatban/a-vizualis-fejlesztoeszkozok-megismerese-hasznalata

[2] https://cordova.apache.org/



1. ábra Cordova felépítése

A Cordova projektet létrehozva az alábbi mappaszerkezet alakul ki



2. ábra Cordova mappa struktúra

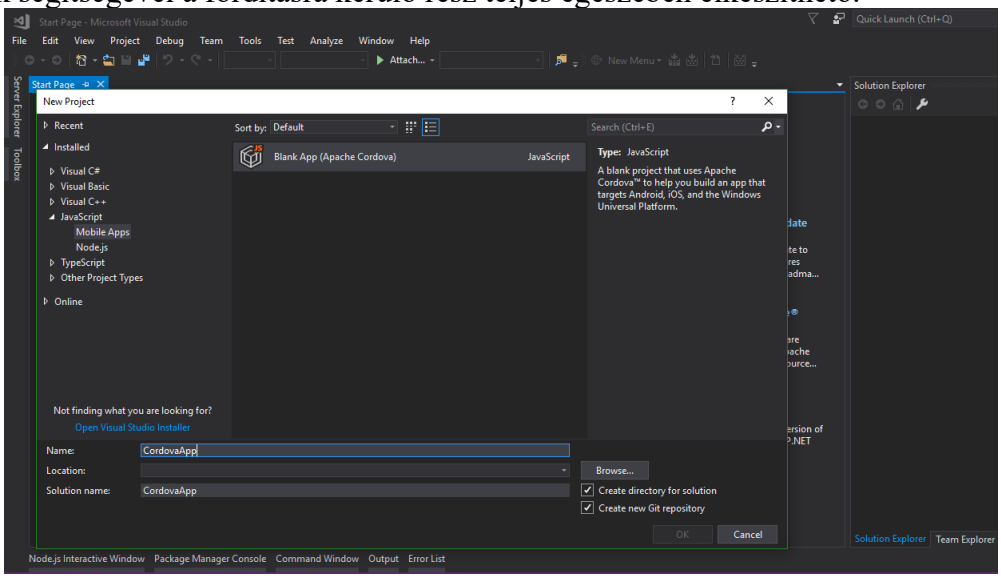
A WWW mappa alá kerül be az eszközökre befordítandó mappastruktúra.
A css mappa tartalmazza a kinézetért felelős css fájlokat.
A media mappa az alkalmazásban előforduló képeknek ad helyet.

A script mappa a JS-t tartalmazza mely vezérli az egyes interakciók esetén végrehajtandó feladatokat.

3.2 Visual Stúdió

Fejlesztői eszköznek a Visual Stúdió-ra esett a választás, mely támogatja a Cordova projekt létrehozását, elkészít egy alap projektet mely készen is áll a fordításra mindhárom platformra. Ugyanakkor ez egy statikusan megjelenő oldal, de a számunkra fontos már említett mappaszerkezetet legenerálja.

A Visual stúdió segítségével lehet fejleszteni a JS, CSS és HTML típusú fájlokat. Ennek segítségével a fordításra kerülő rész teljes egészében elkészíthető.

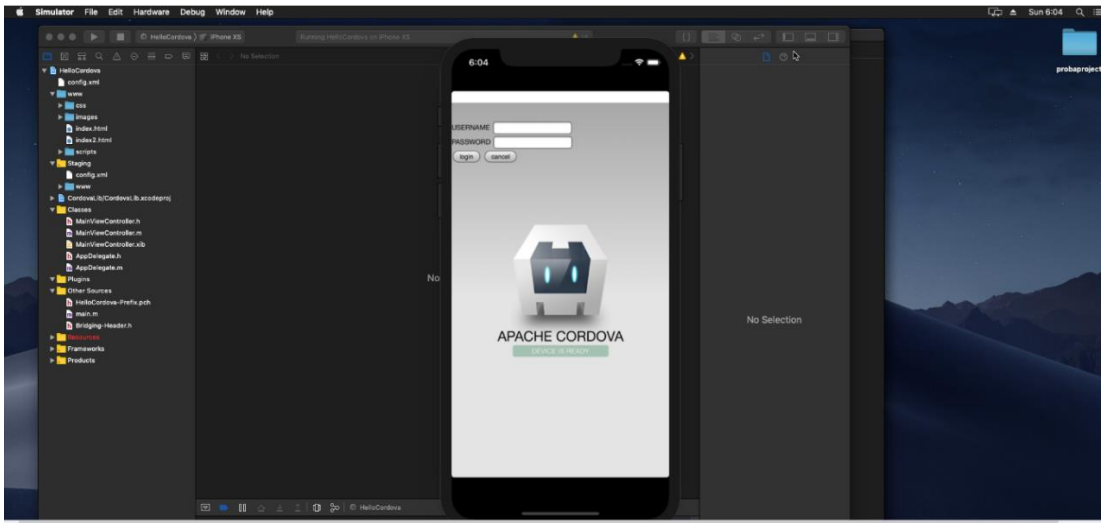


3. ábra A létrehozás folyamata

3.3 XCODE

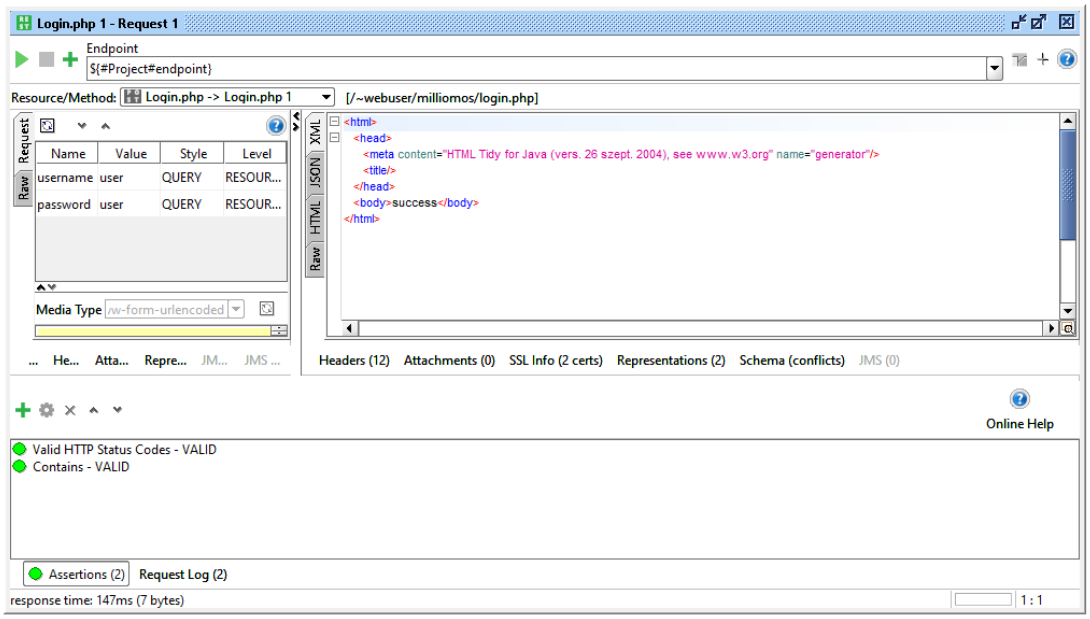
Ezen program segítségével tudjuk emulálni a Apple telefonra készített IOS rendszerét és ezáltal a program működését megfigyelni, Az Xcode 10 mindent tartalmaz, amire szüksége van ahhoz, hogy minden Apple platformhoz csodálatos alkalmazásokat hozzon létre.”[3]Az Xcode is támogatja a cordova projektet. Ennek köszönhetően a virtuális gépen mindössze az emulálás és fordítás folyamatát kellet elvégezni.

[3] https://developer.apple.com/xcode/



3.4 SoapUI

Ezen program a háttérrendszer tesztelési folyamatát könnyítette meg a csapat számára. Beküldhető vele a szerverre egy kérés melyre a szerver válaszol tudjuk vizsgálni, hogy a kapott válasz megfele-e a várt formátumnak, tartalmilag annyi és olyan információt kapunk-e vissza, amit vártunk.



4. ábra Sikeres belépés

3.5 A játék, és szabályai

A Legyen ön is milliomos egy televíziós kvízjáték [4]. A játék célja, hogy minél több kérdésre válaszoljon a játékos helyesen. A játékos nyereménye annál nagyobb, minél több kérdésre válaszol helyesen jól. A kérdésekre 4 válasz lehetőséget kap a játékos, ezek közül csak egy helyes válasz van. A játékos rendelkezik segítséggel is, amiket csak egyszer használhat egy játék során. A segítség viszont nem garantáltan, de nagy valószínűséggel

[4] https://hu.wikipedia.org/wiki/Legyen_%C3%96n_is_milliomos!

adnak helyes válasz, ilyenre példa a közönség szavazata.

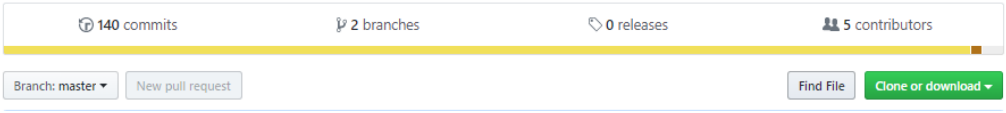
3.6 A projekt

Az applikáció egyéniségét a téma adja, mivel az egyetemmel kapcsolatos kérdéseket ad a játékosnak. Például Számítógépek és Távközlési hálózatok kérdésekre kell válaszolnia a játékosnak, a következő kérdésben pedig már más kurzusról kap kérdést. Az applikáció elsősorban telefonkészüléken fut majd, de internetes böngészőn keresztül is lehet vele játszani. A játékos ugyanazt az élményt kapja, minden platformon.

3.7 Git

A Git egy változást irányító rendszer, az applikáció fejlesztésre ajánlott, mivel nem csak egy ember dolgozik az applikáción. A Git segítségével egy internetes tárolón van a projekt, erről a tárolóról le lehet menteni, és fel lehet tölteni a változtatásainkat, amit program kódján vagy különböző részein hajtottunk végre.

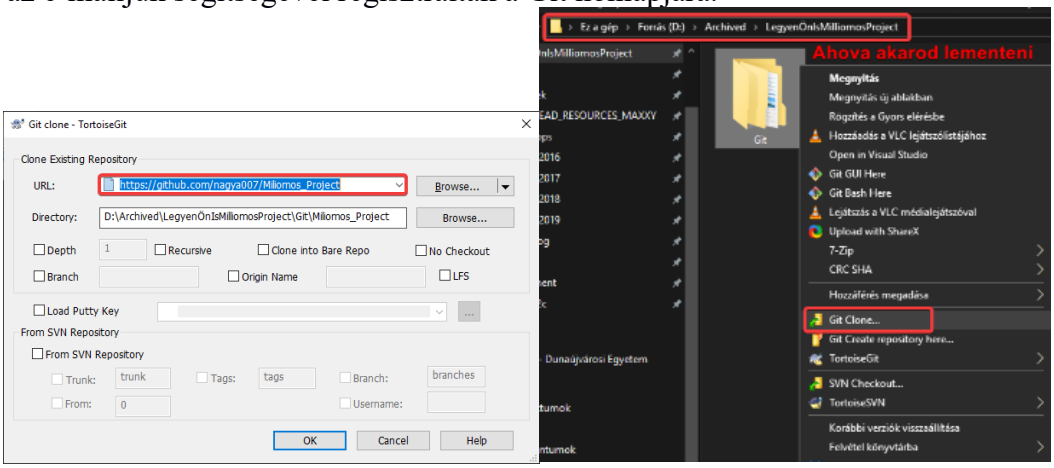
Az irányítás alatt pedig arról beszélünk, hogy minden változást, amit a fejlesztők végre hajtanak, azt menti az internetes tároló. Így lehetőség van nyomon követni változásokat, esetleg visszavonni változásokat.



5. ábra A Git követi a változásokat

a) Használata a Legyen ön is Milliomos projektben

A Git-et lehet használni a hozzávaló terminállal, vagy *TortoiseGit* [5]használatával is. TortoiseGit-el Windows alapú GUI-t kapunk a Git használatához. Emellett rendelkezni kell egy Git felhasználóval is, aminek van engedélye a projekt letöltéséhez. A projekt tagok az e-mailjük segítségével regisztráltak a Git honlapjára.[6]

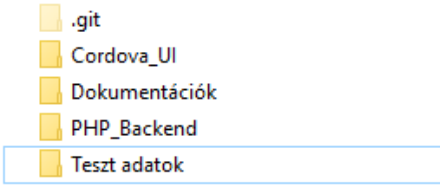


A távoli tárolóról letöltött projektet ezután lehet már használni is, ha pedig szeretnénk feltölteni módosítást a tárolóra, akkor a Git push funkcióját kell használni. Ha pedig mások módosításait akarjunk letölteni, akkor a Git pull funkcióját kell használni. Ajánlott letölteni minden módosítást, még mielőtt mi akarunk módosítani rajta. A

[5] https://tortoisegit.org/
[6] https://github.com/

TortoiseGit lehetővé teszi ennek a folyamatnak a felgyorsítását, mivel egy GUI[7]-t ad a Git szoftvernek. Így elég gomb kattintással fel és letölteni a projekt fájlokat.

A jól szeparáltság érdekében, és a közös munka megkönnyítése érdekében külön mappaszerkezet került kialakításra a backend és a frontend tárolására, ezek mellet továbbá dokumentációknak szintén egy külön mappa lett kialakítva a verziókezelő rendszerben a teszt szintén egy saját mappát kapott a projekt előre haladásával.



6. ábra Projekt mappaszerkezet

3.8 Dokumentálások

Projekt folyamata során bővített dokumentumok. A beszélgetések, összejöveteleket egy eseménynaplóba rögzítve lettek, Excel formátumba. A beszélgetések rögzítésével újabb teljesítendő célokat lehetett megszabni, ennek segítségével mindig van egy elérhető dokumentum, ami tartalmazza a teendőket, és korábbi felmerült kérdéseket. A személyes összejövetelek és az online beszélgetés rendkívül segítette az applikáció gyors fejlődését, és a dokumentumok is ezt tükrözik. A dokumentálás része volt a fontosabb hivatkozások mentése, például képek vagy linkek. Az így mentett adatok nem vesznek el soha, mivel a Git verzió kezelés is elősegítette a dokumentumok bővítését.

4. Frontend

4.1 A frontend felépítése

4.1.1 Mi az a frontend?

A frontend alatt a program megjelenítéséről beszélünk, ami a szerverről lekérdezett (backend) adatokat jeleníti meg a felhasználó számára. Megjelenés mellett különböző műveleteket és számolást is végre kell hajtani a frontend felületnek, annak érdekébe, hogy a felhasználó feltudja dolgozni a kapott adatokat. Ez alatt lehet érteni számok összeadását, kivonását, és ezeknek a számoknak az ábrázolását a megfelelő helyen a képernyőn.

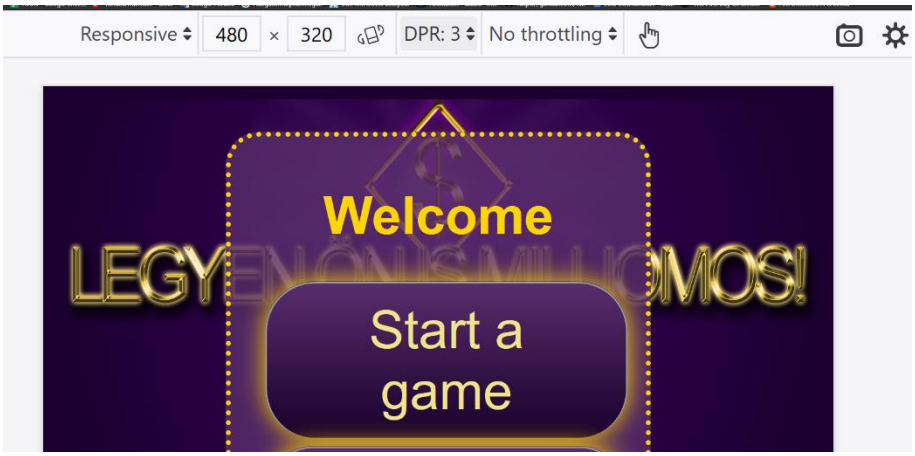
A HTML lehetővé teszi az adatok csoportosítását, megjelenítését megfelelő helyen, és egyéb formai dolgok, például színezés és méretezés. Emellett beépített funkciók és eszközöket is lehet használni vele, például gombok vagy szövegbevitelre mezőt, ahova képes a felhasználó írni. Ezeket az úgynevezett „objektumokat” személyre lehet szabni, és rengeteg tulajdonságot hozzájuk tudunk rendelni, például a felhasználó jelszavát a program csak csillagokkal jeleníti meg, vagy csak bizonyos hosszúságú jelszót írhatunk be. Ilyenkor fontos figyelembe venni, hogy a frontend képes megakadályozni érvénytelen adatoknak a bevitelét,

[7] Graphical user interface

4.1.2 Frontend a Legyen ön is Milliomos applikációban

A frontend fejlesztését lehetővé teszi a Cordova, mivel a HTML web programozási nyelvet használja. A Cordova egy futó web applikáció, ami képes a telefonos verzió kezelésére. A Cordova képes mindenféle funkciót kihasználni a mobilokból, ezalatt a billentyűzetet, kamerát, GPS és forgatást képes kezelni. A Legyen ön is Milliomos applikáció esetében a billentyűzet és megjelenésen van a hangsúly.

A fejlesztéshez szükség van egy IDE [8]szoftver letöltéséhez. A legyen ön is milliomos projekt esetében a Visual Studio 2015 Community Edition-t használta a csoport, és ingyenes is. A Visual Studio emellett nagyon hasznos lesz a „valós idejű" tesztelésre is. A HTML kódot viszont lehet bármilyen más szövegszerkesztővel is használni, például Notepad++, ami egy ingyenes és programozók számára ajánlott szöveg szerkesztő.[9]

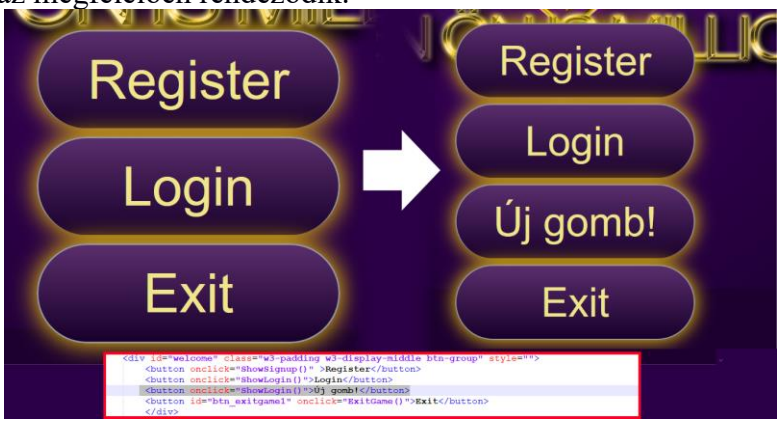


A legnagyobb probléma, ami felmerült a fejlesztés során az a megfelelő arányok eltalálása, mivel az applikáció nem csak számítógépen, de telefonon is használható. A képernyő arányához kell viszonyulnia a képernyőnek, és a rajta lévő objektumoknak, például a gombok és szöveg dobozok. Az arányok mellett a HTML korlátozásai is problémát jelenthet, mivel nem minden web böngésző támogatja az összes funkciót. [10]

A megfelelő arányokat a HTML képes betartani, ha az objektumokat a képernyőhöz

[8] Integrated Development Environment
[9] https://notepad-plus-plus.org/
[10] https://www.w3schools.com/cssref/css3_browsersupport.asp

viszonyítva számoljuk. Ezt úgy lehet elérni, hogy folyamatosan rendezzük a csoportokat, például a képernyő közepére, szélére, legaljára rendezzük az objektumokat. Innentől fogva az applikáció minden platformon megfelelő méretezéssel jelenik meg. Ez viszont rengeteg tesztelést igényel, mivel az applikációt folyamatosan bővíteni szeretnénk. Szerencsére a rendezés elősegíti a bővíthetőséget, mivel képesek vagyunk rendezni az elemeket tetszés szerint, a meglévő elemek befolyásolása nélkül. Például, ha új gombot szeretnénk berakni a főmenübe, akkor elég egy új gomb objektumot felvenni a HTML file-ba, és utána az megfelelően rendeződik.



Új gomb a HTML kódba, megfelelő rendezéssel, csak egy darab új sor kellett

4.1.3 Formázás

A HTML és Cordova használata emiatt előnyös, mivel megfelelő felépítés lehetővé teszi az applikáció gyors fejlesztését és tesztelését, egyszerre teszi lehetővé a webes verzió és a telefonos verzió fejlesztését. A megfelelő formázást a HTML által használt .CSS file tartalmazza, itt található különböző rendezés típusok, amikre hivatkozhatunk a HTML-en belül. Ezt a .CSS file-t egy másik mappába lett téve jobb rendezés érdekében. Kifejezetten hasznos a képernyő alapú rendezés, ami %-osan rendezi a tartalmát a HTML elemeknek. Ezek fel vannak osztva több típusú rendezésre, például felfelé rendezés, lefelé, jobbra és balra.[11]

```
.w3-display-middle {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
    -ms-transform: translate(-50%,-50%)
}
```

A középre rendezés tulajdonságai

```
<div id="mainmenu" class="w3-display-middle w3-padding btn-group" style="display: none;">
    <h1 class="vertical" id="greet">Welcome </h1>
    <button id="btn_startgame" class="btn-group" onclick="StartGame()">Start a game</button>
    <button id="btn_exitgame2" class="btn-group" onclick="ExitGame()">Exit</button>
</div>
```

HTML-en belül, csoport (div) középre van rendezve

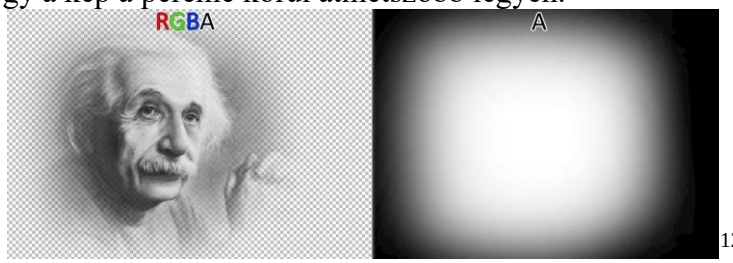
A CSS emellett tartalmazza a többi formai tulajdonságot, például a betűtípus és méretezést, színeket. A CSS lehet akár egy file az interneten is, de ajánlott saját példányt létrehozni, mivel nem biztos, hogy mindig van internet szolgáltatása az applikációnak.

a) Formázás, Média

[11] https://www.w3schools.com/w3css/w3css_display.asp

A HTML használatával lehetséges képeket és videókat is importálni, még akár zenét is. Az applikációban használt média elemeket mappába rendezve vannak tárolva, így könnyebb megtalálni a megfelelő file típust. Támogatott file típusok között van: MP4, PNG, WAV. Fontos tudni, hogy ezek a média elemek nem minden böngészőben működnek ugyan úgy, ezért a HTML kódban meg kell szabni, hogy böngészőtől függően hogyan működjön.

A PNG file típus kifejezetten fontos, mivel lehetővé teszi a PNG negyedik szín csatornájának a használatát: Alpha. Az alpha alatt az áttetszőségről beszélünk, a HTML az áttetszőséget a háttérrel való egybeolvasztást teszi lehetővé. Például szeretnénk, hogy a kép a pereme körül átmetszőbb legyen.



[12]

Alpha Fekete rész = Áttetsző, összeolvad a weboldal színével

Ez az effektus csak a PNG, vagy TGA file típussal lehetséges, mivel ezek a típusok használnak csak Alpha színe. A hátránya az, hogy több helyet foglal el. Előnye viszont, hogy bármikor, ha változtatjuk a weboldal témáját, az ilyen elemek folyamatosan jól fognak illeszkedni a weboldalra, mivel egyéni áttetszésük van. Ha viszont olyan képformátumra van szükségünk, ami viszonylag jó kép minőséget ad, és file méretet, akkor ajánlott a JPG.

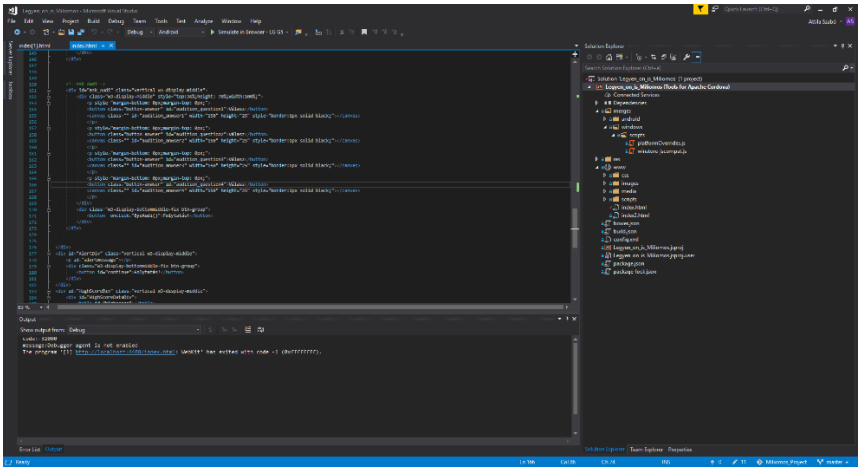
4.1.4 Frontend tesztelés

Mivel az applikáció folyamatosan bővül, ezért állandóan tesztelni kell. A tesztelés során a program használata során előforduló hibákat kell megkeresni, általában új funkciók esetében korábban működő részek is elromolhatnak. A tesztelés emellett jó a még félig kész funkciók tesztelésére is gyakorlatban. Mivel webfejlesztésről van szó, ezért gyorsabb a White Box Testing metódus, ami röviden azt jelenti, hogy mi közvetlenül a weboldal fejlesztés során, a metódusok meghívásával kapunk visszajelzést a program működéséről. Így gyorsan tesztelhető az alkalmazás működése, mivel egyből kapunk visszajelzést a működésről[13]. A tesztelésre több módszer is lehetséges:

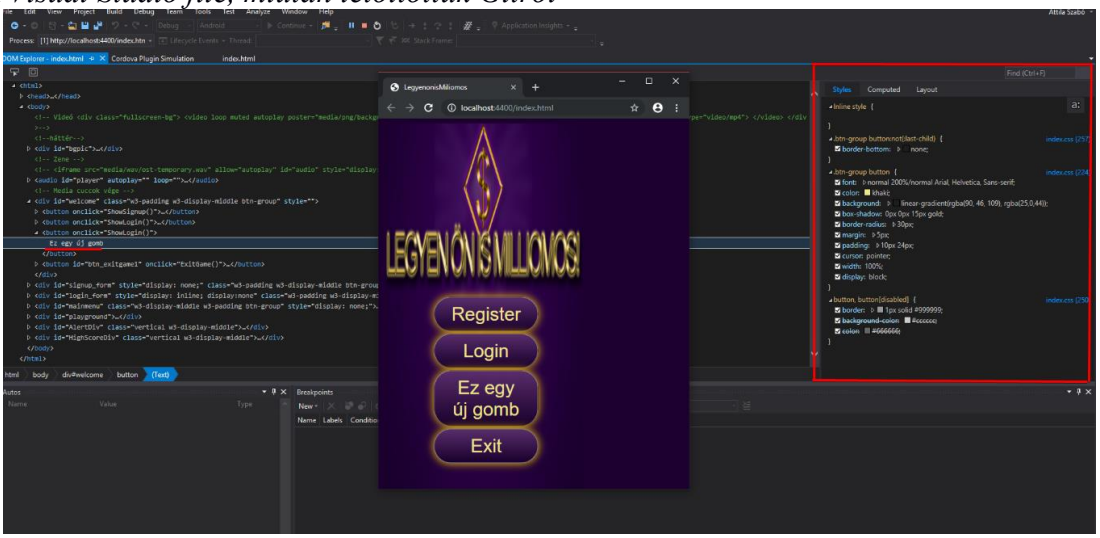
[12]https://opimedia.azureedge.net/-/media/images/utr/editorial/articles/online-articles/2014/04-01/albert-einstein-the-humanitarian/albert-einstein-jpg.jpg

[13] http://softwaretestingfundamentals.com/white-box-testing/

4.1.5 Visual Studio



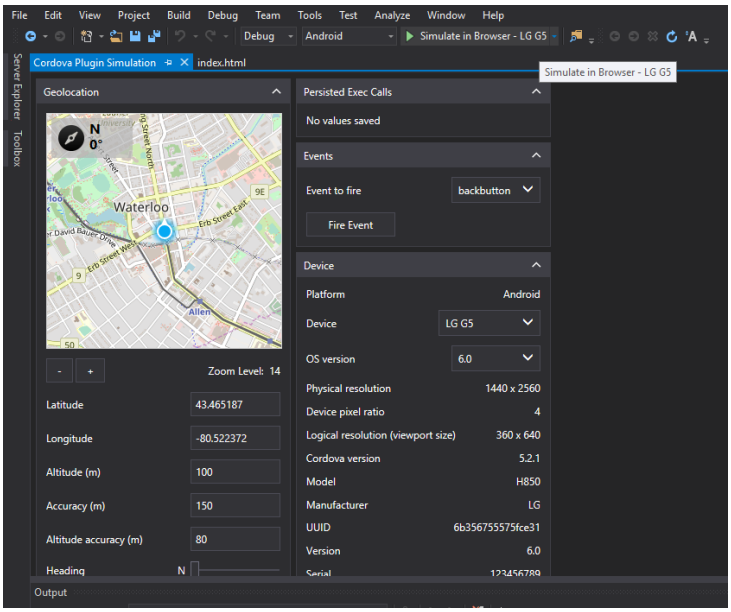
A Visual Studio file, miután letöltöttük Gitről



Valós idejű tesztelésre példa, új sorok egyből megjelennek az applikációban, Visual Studio sajna nem engedi a virtuális képernyő forgatását

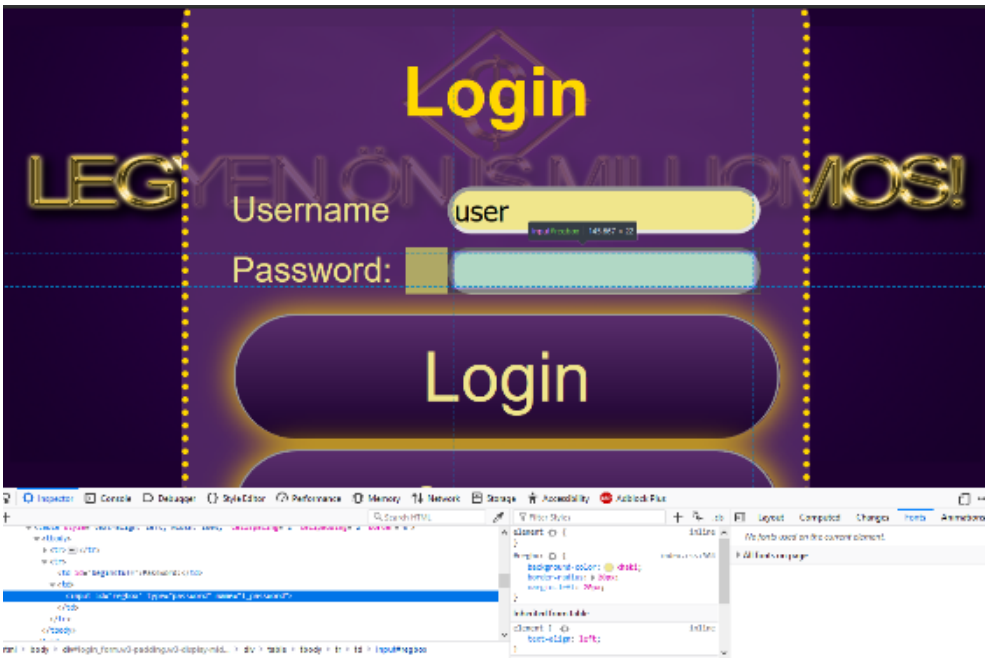
Az applikációt a Visual Studio használatával lehet akár egy telefonos készülékre telepíteni is. A telefonon előbb aktiválni kell a fejlesztői opciókat, így adunk engedélyt a számítógépnek, hogy a telefonra közvetlenül telepítsen alkalmazásokat. A telepítés mellett több lehetőségünk is van, például a debugolás. A debug alatt azt értjük, hogy számítógépünk és a telefonunk össze van kötve, és a kódot látjuk hogyan hívódig meg, még akár változtathatunk is a kódon valós időben. Ez rendkívül hasznos tud lenni, mivel nem kell újra megnyitni az alkalmazást minden alkalommal, amikor változik a kód. Az emuláció [14]emellett rengeteg lehetőségünk is van, akár Android, vagy IOS rendszeren is tudunk emulálni. De ha szimulálni akarunk egy mobil készüléket, akkor annak a beállításainál is lehetőség van az operációs rendszer, és egyéb mobilos paraméterek beállítására, például, hogy mekkora a képarány. Így könnyű tesztelni az alkalmazást, anélkül, hogy vásárolnánk tesztelésre készüléket.

[14] Egy másik készülék használatát utánozza



Android Operációs rendszer beállítása, vagy képernyő arány

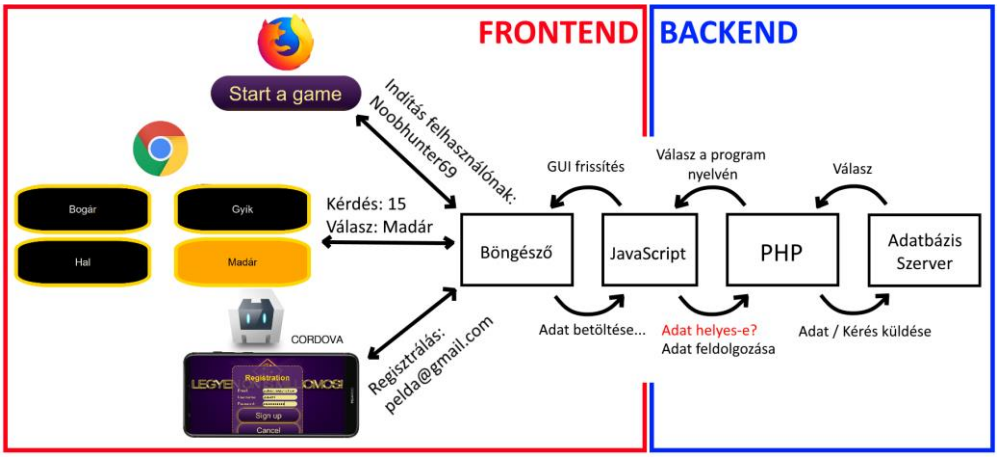
4.1.6 Böngésző fejlesztői opció



Firefox böngésző használatával valós idejű szerkesztés

Lehetséges az applikáció tesztelése Visual Studio nélkül is. A böngészők általában rendelkeznek fejlesztői eszközökkel, ha mi megnyitjuk a HTML file-t, amin dolgoztunk, akkor lehetőségünk van előre megtekinteni a weblap működését. A HTML file megnyitása után a weblap bezárásáig lehetőségünk van bármilyen módosításra. Fontos tudni viszont, hogy ezek a módosítások csak a böngésző példányán jelennek meg, ezeket el kell menteni a HTML file-ba is. A fejlesztői opciók is rendelkezhetnek Androidos, vagy IOS „emulátorral”, ami viszont csak azt csinálja, hogy a weblap telefonos verzióját tölti be, míg a képernyő arányt nem változtatja.

4.2 Fejlesztés



4.2.1 Fejlesztés menete

Az alkalmazás bővítésének legfontosabb része a számolások helyes elvégzése, és kiírása. Ennek a fejlesztési rétegnek jól kell kommunikálnia az alkalmazás backend részével. Egy adatbázisba csak helyes értékeket szabad felvinni, különben hibába ütközünk. Ha pedig a felhasználó nem megfelelő értékeket visz fel, akkor azt ki kell jelezni is. Ez nem különbözteti meg azt, hogy a backend ettől függetlenül tartalmazhat hibakezelést, de ezt természetesen tudatni kell a felhasználóval. A fejlesztésnek folyamatosan figyelembe kell vennie a platform függőséget is, például Firefox, Chrome, és telefonra a Cordova.

A programra használt navigálás és műveletek inicializálását a HTML és a JavaScript végzi el. A HTML-ben szereplő objektumok, például gombok által kerülnek meghívásra különböző műveletek.

Ezek között a műveletek között van az adatbázisból való lekérdezés is. Ezekkel az adatokkal pedig úgy kell bánni, hogy azok felhasználhatóak legyenek a megjelenítésre. A feldolgozott adatokat pedig a backend részére, a szerver adatbázisára küldjük további feldolgozásra. Az innen érkező válasz alapján pedig folytatódik a program élet ciklusa. Tehát fontos feltételezni, hogy ezek között a pontok között bármikor lehetnek hibák, és ezeket kezelni kell. Ezek közé a hibajelzések közé sorolható a szövegdobozos hibaüzenetek, amik visszajelzést adnak arról, hogy nem megfelelő adatot vitt fel a felhasználó, vagy ha esetleg éppen nem elérhető a szerver.

4.2.2 Backend fejlesztés

Elsősorban az applikáció „agya” kell, ahol tárolni lehet a játékos teljesítményét, és bővíteni lehet a játék során előforduló kérdéseket. Ezt megelőzi az adatbázis tervezése. Fontos tudni, hogy milyen adatokkal akarunk dolgozni, később ezeket kiegészíteni már nehezebb. A szerveroldali kommunikációnak emellett kezelnie kell a helytelen adatokat, máskülönben a felhasználók visszaélhetnek a program hibájával.

4.2.3 Frontend fejlesztés

Egy működő alappal már lehet fejleszteni a frontendet. Itt jön figyelembe a tesztelés hangsúlya, mivel van lehetőség arra, hogy a kód futtatása közben fejlesszük a felhasználói felületet. Az adatbázist ilyenkor nem kell már szerkeszteni, később esetleg lehet bővíteni, de több lehetőségünk van a frontenden hasznosítani ezeket. Ugyan azok az adatok több

helyen is előfordulnak, és többször lehet újra hasznosítani őket. Ilyenre példa a felhasználó nevének kiírása, vagy a „kérdések” lekérdezése.

5. Grafikai feladatok, design

5.1 A design tervezéséhez és kivitelezéséhez használt szoftverek:

- Inkscape: Ingyenesen beszerezhető, vektorgrafikus[15] program. Kezdeti vázlatok készítésére, a színvilág - enyhén elnagyolt – tervezésére alkalmazott szoftver.
- Blender: Ingyenesen beszerezhető, 3D-s modellező és animációs program. Átvezető animációk tervezésére, háttér létrehozására alkalmazott szoftver.
- Photoshop: fizetős[16] rasztergrafikus[17] program. A designelemek véglegesítésére, a végső design összeállítására használt szoftver.
- Visual Studio: a Microsoft saját fejlesztésű, ingyenes formában is elérhető fejlesztői környezete. A frontend elemeinek elhelyezésére használt elsődleges szoftver.
- NotePad++: ingyenes, nagy tudású szövegszerkesztő program, rengeteg támogatott fájlformátummal. A frontend elemeinek elhelyezésére – főleg inkább tesztelésre - használt másodlagos szoftver.

5.2 A design kialakításának, tervezésének folyamata:

A projekt kezdeti szakaszában folytatott megbeszélések folyamán több lehetséges forrás is felmerült az alkalmazás színvilágának kialakítására, míg végül az DUE logóját – kék és fehér – választottuk alapnak, a munkavégzés helyének apropóján.

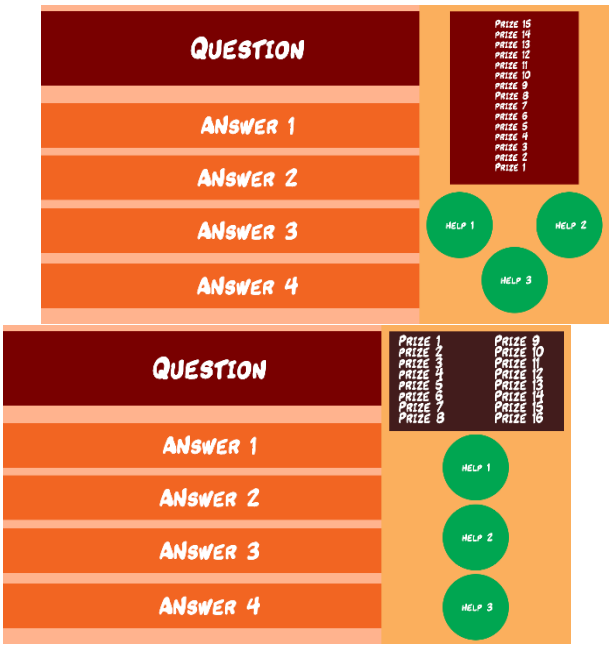
1.2.1.ábra: A DUE logója

A cél egy átlátható, több platformon – ennek okán több méretben – is használható felület kialakítása volt. A megjeleníteni kívánt elemek elhelyezését Varsa kolléga több variációban vázolta fel. Ezek a vázlatok az elhelyezni kívánt elemek számát és arányait hivatottak reprezentálni, a végleges szín- és formavilág még nem került kidolgozásra.

[15] A kép elemeit nem pixelekkel, hanem vektorokkal leíró technológia, amely szabad méretezhetőséget biztosít a minőség romlása nélkül, bár a színátmeneteket nagy felbontás esetén látványosan rosszul kezeli.

[16] A 30 napos próbaverzió elegendő volt-e a feladat végrehajtására

[17] A kép tartalmát pixelek segítségével határozza meg. Nagyszerűen kezeli a színátmeneteket, de a kép nagyításánál jelentős minőségromlás tapasztalható.



1.2.2.a ábra: A játéktér kezdeti vázlatai

1.2.2.b ábra: A játék főmenüjének korai terve.

A csapat egybehangzó jóváhagyását követően az alkalmazás frontendjének kialakítását Szabó kolléga és Varsa úr megkezdte.

A munkálatok korai szakaszában Blender-ben Szabó kolléga készített egy rövid loop videót, mely az egész projekt kiindulási pontját adó, „Legyen Ön is milliomos” című kvízműsor előtt kívánt tisztelegni, s ezért ezt tettük a játék alapértelmezett hátterévé.

[X] időn belül a fejlesztőcsapat arra jutott, hogy a videó túlságosan is megnöveli az alkalmazás méretét (kb. +10 MB), és a tesztelések során felhasznált eszközök közül a régebbi hardvert tartalmazó készülékek hajlamosak voltak rendellenesen felmelegedni és jelentős mértékben veszíteni töltöttségükből a próbajátékok ideje alatt. A fenti problémák kiküszöbölésére közös megegyezés alapján úgy döntöttünk, hogy a videót eltávolítjuk az alkalmazásból, továbbá felfüggesztésre kerültek a további animációk. Felmerült a dinamikus háttér fenntartására mozgókép (gif) használata, de egy rövid tesztet követően ezt a lehetőség is elvetésre került, mivel a mozgókép minősége jelentősen alulmúlta a videóét, míg mérete drasztikusan túlszárnyalta azt (kb. +45 MB). Ezért végezetül egy statikus háttérképet (png) készítettünk a videóban felhasznált modellek alapján.

5.3 A design átültetése az applikációba

A játék elemei html[18]-kódban lettek elhelyezve, mely css[19]-sel kiegészítve remekül formázható, könnyen módosítható kombinációt alkot. Kezdetben felmerült a „hagyományos" html-oldalak struktúrájának követése - ami röviden a gyakorlatban több html oldal felhasználását jelentette volna, melyeket egyszerű hivatkozásokkal kell összekötni egymással -, de ez a módszer negatív hatással volt az alkalmazás sebességére, ezért az elemek elrejtését alkalmaztuk – egyetlen oldalon.

5.4 A HTML

A felhasznált objektumok többsége egy-egy saját mezőt kapott, ezzel lehetővé téve az elemek mozgatását egy, a felhasználó számára láthatatlan rétegen. Az egyes -ek gombnyomásra bukkannak fel vagy tűnnek el. A kód kialakításához nagy segítségünkre volt a W3Schools[20] weboldala.

```
<!--háttér-->
<div id="bgpic"><img src="media/png/bg2.png" width="100%" height="100%"></div>

<!--kezdőképernyő-->
    <div id="welcome" class="w3-padding w3-display-middle btn-group" style="">
        <button onclick="ShowSignup()" >Regisztráció</button>
        <button onclick="ShowLogin()">Bejelentkezés</button>
        <button id="btn_exitgame1" onclick="ExitGame()">Kilépés</button>
        </div>
```

1.3.1.1. Kezdőképernyő - html kódrészlet

```
<!--regisztráció-->
    <div id="signup_form" style="display: none;" class="w3-padding w3-display-middle
    btn-group">
        <h1 class="vertical">Regisztráció</h1>
    <div id="regbg">    <table style="text-align: left; width: 100%" border="0" cellpadding=
    "2" cellspacing="2" >
    <tr>
      <td id="beginstuff" >Email: </td>
      <td><input type="text" name="s_email" id="regbox"></td>
    </tr>
    <tr>
      <td id="beginstuff">Felhasználónév:</td>
      <td> <input type="text" name="s_username" id="regbox"></td>
    </tr>
    <tr>
      <td id="beginstuff">Jelszó:</td>
      <td><input type="password" name="s_password" id="regbox"></td>
    </tr>
</table>
        <button id="btn_signup" onclick="Signup()">Regisztrálás</button>
        <button id="btn_cancelsignup" onclick="CancelSignup()">Mégsem</button>
    </div></div>
```

1.3.1.2. Regisztrációs ablak - html kódrészlet

[18] HyperText Markup Language (hiperszöveges jelölőnyelv) = egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált.

[19] Cascódon Style Sheets (egymásba ágyazott stíluslapok) = stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le.

[20] Webes technológiákkal foglalkozó online tananyagok gyűjteménye

```
<!--bejelentkezés-->
    <div id="login_form" style="display: inline; display:none" class="w3-padding w3-display-middle
    btn-group">
        <h1 class="vertical">Bejelentkezés</h1>
        <div>
        <table style="text-align: left; width: 100%;" border="0"
 cellpadding="2" cellspacing="2">
    <tr>
      <td id="beginstuff">Felhasználónév:</td>
      <td><input type="text" name="l_username" id="regbox"></td>
    </tr>
    <tr>
      <td id="beginstuff">Jelszó: </td>
      <td><input type="password" name="l_password" id="regbox"></td>
    </tr>
</table>
        </div>
      <button id="btn_login" onclick="Login()">Belépés</button>
        <button id="btn_cancellogin"onclick="CancelLogin()">Mégsem</button>
    </div>

<!--főmenü-->
    <div id="mainmenu" class="w3-display-middle w3-padding btn-group" style="display: none;">
        <h1 class="vertical">Mizu arc? </h1>
        <button id="btn_startgame" class="btn-group" onclick="StartGame()">Játék kezdése</button>
        <button class="btn-group" onclick="GetScores()">Eredménylista</button>
        <button id="btn_exitgame2" class="btn-group" onclick="ExitGame()">Kilépés</button>
    </div>
```

1.3.1.3. Bejelentkezés és főmenü - html kódrészlet

5.5 A CSS3

A design pozícionálási szempontból a fotógráfiából ismerős harmadolási szabályból[21] kíván inspirációt meríteni: a képernyő bal oldalának kb. 2/3-át a kérdés és a négy válasz foglalja el, míg a fennmaradó területen a játékos megtekintheti az aktuálisan rá váró jutalmat, valamint a fennmaradó segítségeket is. Az elemek kívánt pozícióinak elérésére CSS-ben adható meg a szükséges érték. Egy-egy objektum elhelyezését a html a 0;0 koordinátákhoz méri, ami a képernyő bal felső sarkát jelenti. Ahhoz, hogy az összkép ne változhasson jelentősen egy 5” -os mobilon és egy 4k[22]-s monitoron sem, a szélességi és magassági értékeket százalékos formátumban kell rögzíteni.

```
#signup_form {
    display: none;
   position:fixed;
   color: gold;
    top: 60%;
    left: 50%;
    font: normal 100%/normal Arial, Helvetica, Sans-serif;
    background-color:rgba(90, 46, 109, 0.8);
    border-style: dotted;
    border-color:lighter gold;
    border-radius:20px;
}
```

1.3.2.1 Regisztráció – css kódrészlet

[21] A képet képzeletben feloszthatjuk függőlegesen és vízszintesen is három részre. Ahelyett, hogy mindent középre komponálnánk, a témát inkább kissé oldalra helyezzük.

[22] 4096 x 3072 képpont (esetenként 3840 x 2160 képpont)

```
#mainmenu {
    display: none;
   position:fixed;
   color: gold;
    top: 60%;
    left: 50%;
    font: normal 100%/normal Arial, Helvetica, Sans-serif;
    background-color:rgba(90, 46, 109, 0.8);
    border-style: dotted;
    border-color:lighter gold;
    border-radius:20px;
}
```

1.3.2.1. A főmenü - css kódrészlet

```
#HighScoreDiv {
    display: none;
    position: absolute;
    background-color: rgba(90, 46, 109);
    top: 40%;
    width: 40%;
    height: 60%;
    max-height: 500px;
    border-style: dotted;
    border-radius:20px;
    border-color:gold;
}
```

1.3.2.2 Az eredménytábla - css kódrészlet

6. Kérdések

Egy kvízjátékhoz elengedhetetlen a megfelelő mennyiségű és minőségű kérdések összegyűjtése. Az alkalmazás publikus változatában[23] szabadon választott témakörökből álló kérdésgyűjtemény kapott helyet, mely az alapvető lexikális tudást feltételező feladványok mellett a fejlesztők által teljesíteni kívánt szakirányra vonatkozó, elenyésző számú szakmai kérdést is tartalmaz.

Főbb témakörök:

- történelem
- irodalom
- zene
- földrajz
- biológia
- informatika
- film

A fejlesztés folyamán felmerült az igény egy fiatalosabb, könnyebben használható változatra[24] is, mely a későbbiek folyamán mind oktatási, mind promocionális célokat is szolgálhat. A fentnevezett kiadással ellentétben, a DUE Edition™ inkább a közelmúlt eseményeire – kb. az elmúlt öt év - kíván fókuszálni, ezzel elősegítve az ismertetett célok elérését. A kérdések összegyűjtése egy-egy, a csapat minden tagja számára szabadon elérhető .csv[25] fájlban történt, mely a fejlesztés ideje alatt folyamatosan bővült.

[23] Mindenki számára letölthetővé tenni kívánt változat, mely állandó hálózati kapcsolatot igényel a kérdések szinkronizálásához.

[24] Az alkalmazás offline változata

[25] Comma-separated values = vesszővel tagolt értékek. Lényegében egy szöveges fájl.

7. JavaScript

A játék JavaScript részénél arra törekedtünk, hogy minél jobban levegyük a terhet a backend-ről és inkább a frontend-en fussanak az algoritmusok. Ennek érdekében a játék alatt az adatbázisból csak egyszer kérünk ki adatot és azt a frontend-en tároljuk.

```
var session_user="";  //name of actually logged in user
var questionnumber=10; //number of requested questions
var rightanswer=""; //the right answer of actual question is stored here
var questions;  //received questions are stored here
var previous=[];    //array for previous question ids
var cnt = 0;  //question counter;
var lock=false; //locks answer buttons after clicking one
```

7.1 Webszerverek elérhetősége:

```
//----------------------
//these urls can be used for application, uncomment the line what you want to use
//var url="https://thejumper203.ddns.net/~webuser/milliomos/";  //my own webserver
//var url="https://milliomos.000webhostapp.com/"; //free webhost server
var url="./"; //localhost for web use
//----------------------
```

Kommentek törlésével könnyen állítható, hogy melyik webszerverre szeretnénk csatlakozni. Illetve localhost indítás is rendelkezésünkre áll.

7.2 Kommunikáció a backend-del

A dinamikus kommunikálás megvalósítását AJAX végezte. JavaScriptről php backend-re tudunk adatot küldeni, ami pedig az adatbázissal való kommunikációt valósítja meg. Ennek segítségével nem kell új html lapot küldeni, így nincs frissítés sem.

7.3 Regisztráció

Regisztráció során csak pár új adatot kell az adatbázisba küldeni.

```
function Signup(){
    //implements signup function via ajax call
    $("#btn_signup").attr("disabled","true");
    var username=$("input[name='s_username']").val();
    var email=$("input[name='s_email']").val();
    var password=$("input[name='s_password']").val();
    $.ajax({
        type:'POST',
        url: url+'signup.php',
        data:'email='+email+'&username='+username+'&password='+password,
        timeout: 5000,
        success: function (data) {
            $("#btn_signup").removeAttr("disabled");
            if(data==="successful")  {ShowAlert("Successful registration. You can login now."); ShowLogin();}
            else if (data==="empty") ShowAlert ("You need to fill all fields");
            else if (data==="exist") ShowAlert("This user/email already exist");
        },
        error: function(){ShowAlert("Something went wrong");$("#btn_signup").removeAttr("disabled");}
    });
}
```

Változók értékeit adjuk át az ajax-nak. Típusa: POST. URL a célt jelöli, ez jelen esetben a signup.php fájl lesz. Az adatokat stringként, összefűzve küldjük el. A függvény futása végén alert-tel jelezzük a küldés sikerességét, sikertelenségét.

7.4 Bejelentkezés

A bejelentkezés hasonló a regisztrációhoz az ajax szempontjából. Itt is adatot küldünk a backend részére. Típusa tehát POST. AZ URL a login.php fájl. Az adatokat ugyan úgy stringként összefűzve küldjük el.

```
function Login(){
    //implements login function via ajax call
    $("#btn_login").attr("disabled","true");
    var username=$("input[name='l_username']").val();
    var password=$("input[name='l_password']").val();
    $.ajax({
       type:'POST',
       url: url+'login.php',
       data:'username='+username+'&password='+password,
       timeout: 5000,
       success: function(data){

          if(data==="false") ShowAlert("Wrong username or password");
          else {
             session_user=username;
             ShowMainMenu(session_user);
          }
          $("#btn_login").removeAttr("disabled");
       },
       error: function(){ShowAlert("Something went wrong");$("#btn_login").removeAttr("disabled");}
    });
}
```

7.5 GetScores(), GetQuestions() függvények

Ezekkel a függvényekkel kérjük ki az adatbázisból a pontokat, kérdéseket. Ugyan úgy ajax-szal működik a kérés, majd az adatot változókban tároljuk, mint az előbbiekben. Továbbá készült még ezekhez függvény, ami a változókból a html elemekre tölti fel ezeket az adatokat.

7.6 Offline mód

Egy helyi JSON fájlban is tároljuk a kérdések táblát. Offline módban nem az adatbázishoz intézünk kérést, hanem erre a fájlra hivatkozunk és innen olvassuk ki a kéréseket.

7.7 UI

A játék élvezhetősége érdekében egy html oldal fut végig és ezen történnek a változások a különböző függvények segítségével. Néhány példa:

7.7.1 CancelLogin()

A metódus eltűnteti a login formot és a kezdőképernyő állapotát hozza vissza.

```
function CancelLogin(){
    $("#welcome").css("display", "block");
    $("#login_form").css("display", "none");
    $("#mainmenu").css("display", "none");
    $("#playground").css("display", "none");
}
```

7.7.2 CancelSignup()

A metódus az előbb említett módszerrel a regisztrációs formot tűnteti el.

```
function CancelSignup() {
    $("#welcome").css("display", "block");
    $("#login_form").css("display", "none");
    $("#mainmenu").css("display", "none");
    $("#playground").css("display", "none");
    $("#signup_form").css("display", "none");
}
```

7.7.3 CleanUp()

A metódus a játék végén hívódik. Ennek az a feladata, hogy visszaállítson különböző kezdőértékeket, pl.: a jó válasz tárolását, a kérdéseket tároló tömböt, előző kérdés ID-ját tároló változót, kérdések számlálóját.

```
function CleanUp(){
    //at the end of a game, this function clear these fields
    AskTheAudience_ClearCanvas();
    ByeAudi();
    rightanswer="";
    questions=null;
    previous=[];
    cnt=0;
}
```

7.7.4 AskTheAudience_ClearCanvas()

Itt hívódik meg az AskTheAudience_ClearCanvas() nevű metódus, ami a közönség segítség kezdőértékeit állítja vissza.

```
function AskTheAudience_ClearCanvas()
{
    for (i = 0; i <= 3; i++) {
    var a = document.getElementById("audition_anwser"+(i+1));
    const context = a.getContext('2d');
    context.clearRect(0, 0, a.width, a.height);
    }
}
```

7.7.5 ByAudi()

A metódus eltűnteti a közönség segítség megjelenítését.

```
function ByeAudi() {
    //$("#ask_Audi").css("display", "none");
    $("#ask_Audi").fadeOut("slow");

}
```

7.8 Segítségek

7.8.1 Közönség szavazás:

A cél az volt, hogy négy darab véletlenszerű törtet generáljunk, amiknek az összege egy egész számot alkot, hogy a közönség szavazását tudjuk szimulálni.

```
function AskTheAudience(){
    var tips = new Array(4);
    var sum=0;
    for (i=0;i<=3;i++)
    {
        if($("#answer"+(i+1)).text()===rightanswer)
        {
            tips[i] = (Math.floor(Math.random() * 100) + 1)+30; //Itt állítható a boost.
            sum += tips[i];
        }
        else
        {
            tips[i] = Math.floor(Math.random() * 100) + 1;
            sum += tips[i];
        }
    }
    for (i=0;i<=3;i++)
    {
        tips[i]=(tips[i]/sum).toFixed(2); //.toFixed() a tizedesjegyet állítja és stringre alakít!!!
    }
    return tips;
}
```

Ehhez létrehoztunk egy 4 elemű tömböt, amiben a szavazatszámokat különbözőképpen tároljuk. Kellett még egy a számok összegét tároló változó is.

```
function AskTheAudience(){
    var tips = new Array(4);
    var sum=0;
```

Egy cikluson belül hozzuk létre a random számokat, de előtte minden lépésnél meg kell vizsgálnunk, hogy jó vagy rossz válasznak. Ehhez egy elágazásban kell összehasonlítanunk a gombunk által tárolt szöveget és a *rightanswer* globális változóban tárolt helyes választ. A html gomb nevére kell hivatkoznunk (amik *#answer1*, *#answer2*, *#answer3*, *#answer4* névre hallgatnak a html-ben), illetve annak .text() metódusára, úgy, hogy hozzá fűzzük az *i* változót is. Erre egyrészt azért van szükség, hogy a pontos nevét kapjuk meg a html gombnak másrészt pedig, hogy végig tudjunk lépdelni a többi gombon az *i* változtatásával. Tehát az elágazáson belül összehasonlítjuk a két stringet.

```
if($("#answer"+(i+1)).text()===rightanswer)
{
```

Ha egyezik, akkor a jó válasz gombjánál jár a ciklus. Tehát létrehozhatunk egy random számot a Math.floor(Math.random()) -mal 1 és 100 között és hozzáadhatunk egy bizonyos értéket, amivel növelhetjük a közönség „tudását”.

```
tips[i] = (Math.floor(Math.random() * 100) + 1)+30; //Itt állítható a boost.
sum += tips[i];
```

Itt +30-cal növeltük a random számot, ez 30% növelésnek fog megfelelni. Ezután a *sum* változóhoz adjuk ezt a számot, hogy a ciklus végén meglegyen a számok összege.

Az else ágon csak simán random számot generálunk, nem adunk hozzá semmit, hiszen a rossz válaszról van szó. Itt is hozzáadjuk a számot a *sum*-hoz. Ezen a szinten már fel van töltve a tömbünk egészekkel. Törtekké kell alakítanunk, amiket, ha összeadunk egyet kapunk. Egy újabb ciklus segítségével végig megyünk a tömbön és leosztjuk a számokat a *sum* változóval.

```
for (i=0;i<=3;i++)
{
    tips[i]=(tips[i]/sum).toFixed(2); //.toFixed() a tizedesjegyet állítja és stringre alakít!!!
}
```

Hogy 2 tizedesig írjuk ki a számokat .toFixed(2) metódusát használtunk. Itt figyelni kell rá, hogy stringre változtatja a számokat.

A függvény végén egy return-nel visszaadjuk a *tips* tömbünket.

7.8.2 Felezés:

A felezés funkció két rossz választ vesz el véletlenszerűen.

```
function HelpRemove(count)
{
    var i=0;
    while(i<count) {
        var rndButton = Math.floor(Math.random() * 4) + 1;
        if(!($("#answer"+rndButton).text()===rightanswer) && !($("#answer"+rndButton).text()===""))
        {
            $("#answer"+rndButton).html('');
            $("#answer"+rndButton).attr('disabled', 'true');
            // Hidden akkor kell, ha azt akarjuk ne legyen ott a fekete gomb se.
            //$("#answer"+rndButton).hidden=true;
            //alert("teszt IF true");
            i++;
        }
    }

    // 1 = indexe
    HelpDisable(1);
}
```

Először felveszünk egy *i=0* változót a ciklus lépéseinek követése végett. Majd egy *count* (alap értéke 2) nevű bemeneti paraméterrel ellátott while ciklusban generálunk 1 és 4 között véletlenszerűen egy számot (*rndButton* változóba), ami a gomboknak a számát fogja megadni.

```
var rndButton = Math.floor(Math.random() * 4) + 1;
```

Ezt a számot hozzá kell fűzni az *#answer* stringhez és így megkapjuk az egyik html válasz gombnak a nevét.

```
if(!($("#answer"+rndButton).text()===rightanswer) && !($("#answer"+rndButton).text()===""))
```

Ezt egy elágazáson belül tesszük meg amiben .text() metódusával az adott gomb text-jét tudjuk kikérni és ezt összehasonlítani a *rightanswer* globális változónkkal, így megvizsgáltuk, hogy az gombunk jó vagy rossz választ ad. Ha rossz választ akkor inaktívvá tesszük a gombot és növeljük az *i* változónkat.

```
$("#answer"+rndButton).html('');
$("#answer"+rndButton).attr('disabled', 'true');
// Hidden akkor kell, ha azt akarjuk ne legyen ott a fekete gomb se.
//$("#answer"+rndButton).hidden=true;
//alert("teszt IF true");
i++;
```

Jó válasz esetén tovább megy a ciklus mindaddig amíg az *i* értéke el nem éri a *count* értékét, azaz összesen 2 darab random rossz választ fog elvenni a függvényünk.

A függvény végén a HelpDisable() függvény hívását illesztettük be, ami inaktívvá teszi ezt a segítséget.

7.8.3 Einstein:

Einstein segítség hasonló a közönség szavazáshoz. A különbség az, hogy ez csak egy tippet ad, de azt nagyobb valószínűséggel. Minél előrébb járunk a játékban annál jobban romlik a pontossága.

```
function HelpTip1(){
    var weights=[2,5,83,6];
    for (i=1;i<=4;i++)
    {
        if($("#answer"+i).text()===rightanswer)
        {
            var temp=weights[i-1];
            weights[i-1]=83-questions[cnt-1].level;
            weights[2]=temp;
        }
    }
    var sumweight=0;
    for (i=0;i<weights.length;i++) sumweight+=weights[i];
    var rnd=Math.floor(Math.random()*sumweight)+1;
    for (i=0;i<weights.length;i++){
        if(rnd<weights[i]) {HelpDisable(2);return i+1;}
        rnd-=weights[i];
    }

}
```

Súlyozott random elven működik a függvény. A súlyozás értékeit a *weights* tömb tárolja. A következő for ciklus egy elágazás segítségével megkeresi, hogy melyik gombon lesz a jó válaszunk. Ezt az eddig megszokott módon .text() metódussal, összehasonlítja a html gomb stringjét és a *rightasnwer* változónkat. Ha egyezik akkor a tömbön, a gombnak megfelelő pozícióra írja át a 83-mat csökkentve a játékban elért jelenlegi szinttel (amit a *cnt* változó tárol). Ezzel gyakorlatilag a jó válasz gombjának a pozíciójára helyeztük a legnagyobb súlyozást.

```
if($("#answer"+i).text()===rightanswer)
{
    var temp=weights[i-1];
    weights[i-1]=83-questions[cnt-1].level;
    weights[2]=temp;
}
```

Ezután felveszünk egy *sumweight* nevű változót, 0 kezdőértékkel. Ez fogja tárolni a súlyozások összegét. Egy for ciklussal végig megyünk a tömbön és az értékeket hozzá adjuk a változónkhoz. A következő lépésben kreálunk egy random számot 1 és a súlyozás értéke között. Ezt az *rnd* változó tárolja. Majd egy újabb for ciklussal végig lépdelünk a *weights* tömbön és minden ciklusban egy elágazással megnézzük, hogy az *rnd* változónk kisebb-e a *weights* értékével. Ha igen visszatérünk egy *i+1* értékkel, ha pedig nem, kivonjuk az *rnd* változóból a tömb jelenlegi értékét.

```
var sumweight=0;
for (i=0;i<weights.length;i++) sumweight+=weights[i];
var rnd=Math.floor(Math.random()*sumweight)+1;
for (i=0;i<weights.length;i++){
    if(rnd<weights[i]) {
        return i+1;
        }
    rnd-=weights[i];
}
```

8. Szerveroldal

A szerveroldal feladata a kliensek felől érkező kérések kiszolgálása és az adatok központi tárolása, kezelése. Az alkalmazás webes és mobil verzióját is ugyanaz a rendszer szolgálja ki. Ez a rendszer áll egy webszerverből és egy adatbázis szerverből.

8.1 Webszerver

A kliens oldalról érkező HTTP kéréseket egy Apache[26] webszerver szolgálja ki, amelyre PHP értelmező is van telepítve. Ezzel a szerver képes szerveroldali PHP kódok futtatására. Hálózati operációs rendszerként az Ubuntu 18.04 szerverekre szánt verziója szolgál, amelyre még különböző kiegészítő szolgáltatások lettek telepítve, mint pl. SSH a távoli eléréshez, SFTP a távoli biztonságos fájlátvitelhez. Maga a szerver virtualizált környezetben fut egy Windows 10 alapú asztali számítógépen, amely a VirtualBox nevű virtualizációs alkalmazás segítségével lett létrehozva. Ahhoz, hogy ez a szerver az internet felől is elérhető legyen, a forgalomirányítón meg kell nyitni a megfelelő portokat (port forwarding), és szükség van egy dinamikus DNS[27] szolgáltatásra is. Ez azért szükséges, mert az internetszolgáltatótól kapott IP cím folyamatosan változik, valamint egy választott DNS nevet könnyebb megjegyezni és használni, mint egy IP címet.

A projekt készítése során a szervert több és különböző jellegű támadás is érte az internet felől. Az egyik támadási forma célja az lehetett, hogy root(rendszergazda) hozzáférést szerezzenek a szerverhez SSH protokollon keresztül. Ennek során egyszerre több -jellemzően kínai- IP címről próbáltak root-ként kapcsolódni és bejelentkezni, szerencsére sikertelenül. A szerver üzembe helyezésekor az SSH kiszolgáló konfigurálása során már letiltásra került a root bejelentkezés, amellyel pontosan az ilyen próbálkozások védhetők ki, mivel, ha a külső behatoló ki is találja a jelszót, a rendszer akkor sem engedi belépni. Root-ként bejelentkezve gyakorlatilag teljes hozzáférést kaptak volna a szerver bármely szolgáltatásához, valamint az adatokhoz. A próbálkozásokat valószínűleg valamilyen fajta automatizált folyamatok hajtották végre, amellyel az SSH alapértelmezett 22-es portján próbáltak kapcsolódni. Védelmi intézkedésként az SSH alapértelmezett portja át lett állítva egy teljesen véletlenszerű portra, így a kiszolgálóhoz csak annak ismeretében lehet kapcsolódni. Ezután további hasonló próbálkozásoknak nem volt nyoma.

```
thejumper203@testserver:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0     69 testserver:ssh          36.156.24.94:57836      LAST_ACK
tcp        0      0 testserver:ssh          118.123.15.131:50375    ESTABLISHED
tcp        0      0 testserver:ssh          61.184.247.6:34219      ESTABLISHED
tcp        1     69 testserver:ssh          222.186.30.71:32872     LAST_ACK
tcp        0     68 testserver:ssh          222.186.30.71:44308     ESTABLISHED
tcp        0      0 testserver:ssh          61.184.247.11:50509     ESTABLISHED
tcp        0     69 testserver:ssh          222.186.30.71:48914     LAST_ACK
tcp        0      0 testserver:ssh          36.156.24.96:50500      TIME_WAIT
tcp        0     69 testserver:ssh          222.186.30.71:46938     LAST_ACK
tcp        0      1 testserver:ssh          222.186.30.71:33694     LAST_ACK
```

1. kép - Néhány TCP kapcsolat a támadó IP címekkel a netstat parancs kimenetén

A másik támadási próbálkozás a webszerver ellen irányult, amely során, a naplófájlok bejegyzései alapján, a támadók (szintén leginkább kínai és orosz IP címekről) több érvénytelen HTTP kérés mellett, a WordPress elnevezésű tartalomkezelő és blog-

[26] Apache HTTP szerver: egy nyílt forráskódú webkiszolgáló alkalmazás
[27] DNS: Domain Name System, magyarul tartománynévrendszer

rendszer admin felületét szerették volna elérni. Azonban mivel a szerveren nem található meg a WordPress rendszer, így ezek a próbálkozások is természetesen szintén sikertelenek voltak.

Ezen kísérletek hatására szükség volt néhány biztonsági intézkedésre, amelyek megnehezíthetik a támadók dolgát. A szerveren szigorítva lett a hozzáférés szabályozás, jogosultági rendszer, valamint szigorúbb tűzfal szabályok léptek érvénybe, így csak azok a szolgáltatások érhetők el az internet felől, amelyek feltétlenül szükségesek, az adatbázis szerver pedig egyáltalán nem érhető el kívülről. Ezen kívül a támadó IP címek nagyrésze tiltólistára került, így az onnan érkező bejövő kapcsolatok automatikusan vissza lesznek utasítva. A kliensek és a szerver közötti biztonságos kommunikációért a HTTPS protokoll felel, amely egy hitelesített titkosított csatornán keresztül továbbítja az adatokat, megakadályozva az adatokhoz való illetéktelen hozzáférést.

A fejlesztés során előkerültek különböző problémák a dinamikus DNS használatával is. Mivel ezeket a DDNS címeket bárki létrehozhatja, így nem lehet tudni, hogy valójában ki áll egy-egy ilyen cím mögött, ezért nagyon sok rendszer az ilyen DDNS domaineket automatikusan biztonsági fenyegetésnek tekinti, és megtiltja a kapcsolódást ezekhez. A projekt esetében a szerver címét az Egyetemi kollégium webszűrője blokkolta, így a kollégista csoporttagok nem tudták elérni a szervert. Ennek kiküszöböléséhez szükség volt egy másik szerverre, így a 000webhost nevű ingyenes webtárhely szolgáltatás lett alkalmazva, amely már bárhonnan gond nélkül elérhető és korlátozásokkal ugyan, de biztosítja azokat a szolgáltatásokat, amelyek a rendszer működéséhez szükségesek, gyakorlatilag 0-24 órás rendelkezésre állással, szemben a saját szerverrel.

8.2 Adatbázis

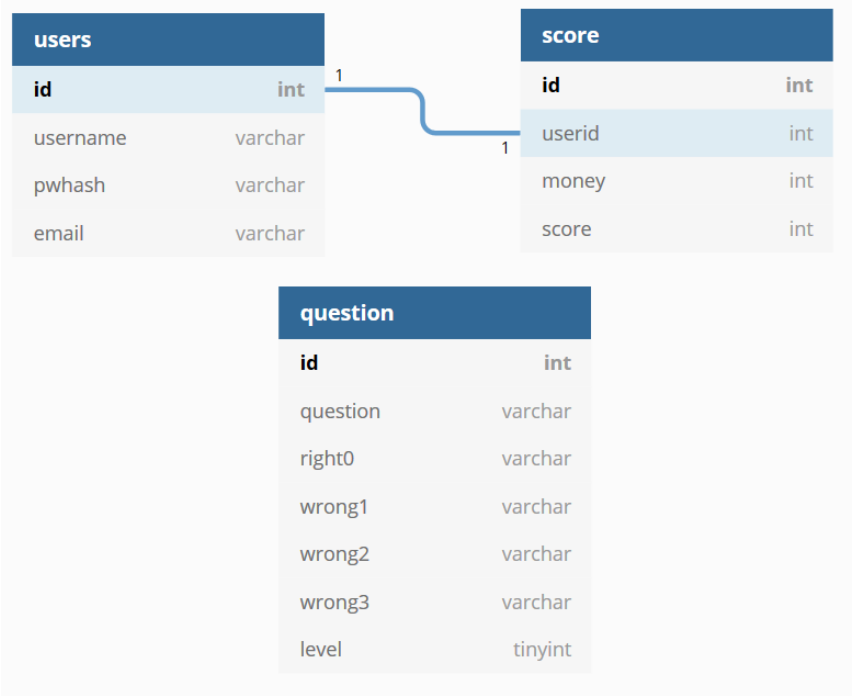
Az adatbázisok kezelése a MYSQL relációs adatbáziskezelő-rendszer segítségével történik. Az adatok egy milliomos nevű adatbázisban vannak, amelyben a következő táblák találhatóak:

users		
id	int	Felhasználó azonosítója
username	varchar(150)	Felhasználónév
pwhash	varchar(150)	A jelszó lenyomata
email	varchar(150)	Email cím

question		
id	int	Kérdés azonosítója
question	varchar(150)	Kérdés szövege
right0	varchar(150)	Helyes válasz
wrong1	varchar(150)	Helytelen válaszok
wrong2	varchar(150)	
wrong3	varchar(150)	
level	tinyint	Kérdés szintje

score		
id	int	Azonosító
userid	int	Felhasználó azonosító

money	int	Legmagasabb megnyert összeg
score	int	Pontszám



7. ábra - Az adatbázis felépítése

Látható, hogy a users id mezője és a score userid mezője között 1:1 kapcsolat található. Ez azt jelenti, hogy minden felhasználóhoz egyetlen rekord tartozhat a score táblában és minden egyes pontszám csak egyetlen felhasználóhoz tartozhat. Új felhasználó hozzáadásakor a score táblában is létrejön egy új rekord a megfelelő felhasználói azonosítóval. A score táblát az alábbi SQL kód hozza létre:

```
CREATE TABLE score (
  id INT NOT NULL AUTO_INCREMENT,
  userid INT NOT NULL FOREIGN KEY REFERENCES user(id) ON DELETE CASCADE,
  money INT DEFAULT 0,
  score INT DEFAULT 0,
  PRIMARY KEY (id));
```

A userid idegen kulcsként szolgál, amely a users tábla id mezőjére hivatkozik. Az ON DELETE CASCADE paraméter pedig arra szolgál, hogy ha törlődik egy felhasználó a users táblából, akkor a hozzá tartozó pontszám is törlődjön a score táblából, megakadályozva azt, hogy olyan rekordok maradjanak a táblában, amelyek már senkihez sem tartoznak.

8.3 PHP

8.3.1 sqlcredits.php

Ebben a fájlban találhatók meg azok az adatok, amelyek az adatbázishoz történő kapcsolódáshoz szükségesek, pl.: adatbázis címe, felhasználónév, jelszó, adatbázis neve. Célszerű ezeket az adatokat külön tárolni, és minden használatnál erre a fájlra hivatkozni,

mivel így egy esetleges módosításnál elég, csak ebben az állományban módosítani, valamint biztonsági szempontból is előnyösebb, mivel ezt a fájl kívülre tehetjük azon a könyvtáron, amely az internet felől elérhető.

8.3.2 signup.php

Ez a fájl felel a felhasználók regisztrációjáért. A kliensen lévő regisztrációs űrlapból ide érkeznek be az adatok egy HTTP POST kéréssel. Ebből a POST kérésből a kód kinyeri a paramétereket (felhasználónév, jelszó, email), majd egy trim() függvény segítségével levágja a végéről az esetleges whitespace karaktereket (pl szóköz, tabulátor). Az SQL injection nevű támadási módszer ellen véd a real_escape_string() függvény, amely megakadályozza, hogy a SQL kódot injektáljanak a formon keresztül:

$username=$sqlconn->real_escape_string(trim($_POST["username"]));

A fenti kódsoron látható a real_escape_string() függvény használata, amely úgy küszöböli ki az SQL injection támadás lehetőségét, hogy escape-eli (feloldja) az olyan potenciálisan veszélyes karaktereket, mint például a' vagy a", mégpedig úgy, hogy beszúrja eléjük az escape karaktert, amely a \ (backslash). Így azokat karaktereket máshogy fogja értelmezni a kód, és egy rosszindulatú felhasználó már nem tud kártékony kódot beilleszteni a szövegdobozon keresztül.

Ezután megvizsgálja, hogy minden paraméter kitöltésre került-e, amennyiben valamelyik paraméter üres, abban az esetben egy „empty" üzenet megy vissza a kliensnek, ahol az ennek megfelelő hibaüzenetet kapja. Ezután ellenőrzi, hogy van-e már regisztrált felhasználó ezzel a névvel és/vagy emaillel. Ilyenkor a program egy „exist" üzenetet küld a kliensnek, ahol a felhasználó szintén egy üzenetet kap, hogy az adott email és/vagy username már használatban van. Amennyiben nincs, úgy az adott felhasználó regisztrálásra kerül az adatbázisba és bejelentkezhet.

Biztonsági okokból a jelszó nem kerül tárolásra az adatbázisban, hanem csak annak a lenyomata, így még az sem tudhatja meg a jelszót, aki hozzáfér az adatbázishoz. A ténylegesen eltárolt karakterláncot az alábbi kódsor hozza létre:

$pwhash=password_hash($password, PASSWORD_DEFAULT);

A password_hash() egy beépített PHP függvény, amely egy karaktersorozat lenyomatát készíti el.

8.3.3 login.php

A bejelentkezésnél a felhasználónevet és jelszót a regisztrációnál használt módszerhez hasonlóan kapja meg és ellenőrzi.

```
$result= $sqlconn->query("select * from users where username='$username'");
$data= mysqli_fetch_assoc($result);
if(mysqli_num_rows($result)==1 && password_verify($password, $data["pwhash"])) {
    $_SESSION["session_user"]=$data["username"];
    echo "success";
}
else echo "false";
```

2. kép - A megadott felhasználónevet és jelszót ellenőrző kódrészlet

A felhasználó ellenőrzése úgy történik, hogy először a megadott felhasználónév összes adatát lekérdezi az adatbázisból, amelyet utána egy asszociatív tömbbé alakít. Ezután a program megvizsgálja, hogy pontosan egy eredményt kapott-e vissza az adatbázisból, valamint lefuttatja a password_verify() függvényt, amely egy hasú-t készít

a megadott jelszóból és összehasonlítja az adatbázisban tárolt hash-sel. Amennyiben egyezik a kettő, úgy logikai igaz értékkel tér vissza. A bejelentkezés tehát akkor sikeres, ha az megadott felhasználónév létezik, és a password_verfy() függvény is igaz értékkel tér vissza. Ilyenkor a program egy „success” üzenetet küld vissza a kliens fel, ellenkező esetben pedig egy „false” üzenetet, amelyre a kliens hibaüzenetet dob a felhasználónak. Ugyanez történik akkor is, ha valamelyik mező üresen maradt.

8.3.4 getQuestions.php

A játék megkezdésekor a kliens küld egy kérést, amellyel a játék kérdéseit kérdezi le. Paraméterként egy számot kap, hogy hány kérdést adjon vissza.

```
$sql="select id,question,right0,wrong1,wrong2,wrong3,level
    from (select *, @row:=if(level=@level,@row,0)+1 as rn, @level:=level from
    (select *,RAND() as trand from question_test) t1,
    (select @row:=0,@level:='') tm2
    order by level,trand) t2 where rn<=1;";
```

3. kép - A kérdéseket lekérdező SQL utasítás

A fenti kód az adatbázisból szintenként kérdezi le a kérdéseket véletlenszerűen, és minden egyes kérdés egy-egy asszociatív tömbbe kerül. Ezek az asszociatív tömbök szintén egy tömbbe kerülnek, amelyet a következő kódsor kap meg paraméterként:

```
echo json_encode($rows);
```

Ez a sor annyit tesz, hogy az adatbázisból kapott kérdéseket JSON[28] formátumba kódolja és visszaküldi a kliensnek, ami ezt feldolgozza és tárolja.

8.3.5 getScores.php

Lekéri a tíz legnagyobb pontszámmal rendelkező játékos nevét és pontszámát, majd JSON formátumban visszaküldi a kliensnek.

8.3.6 highscore.php

Minden játék végén a kliens küld egy POST kérést ennek a kódnak, amely paraméterként tartalmazza a felhasználónevet és az elért nyereményt. Ezután összehasonlítja az adatbázisból lekéri a felhasználó pontszámait és összehasonlítja az éppen elért nyereménnyel. Ha utóbbi magasabb, akkor az új nyeremény íródik az adatbázisba.

9. Kliensoldal

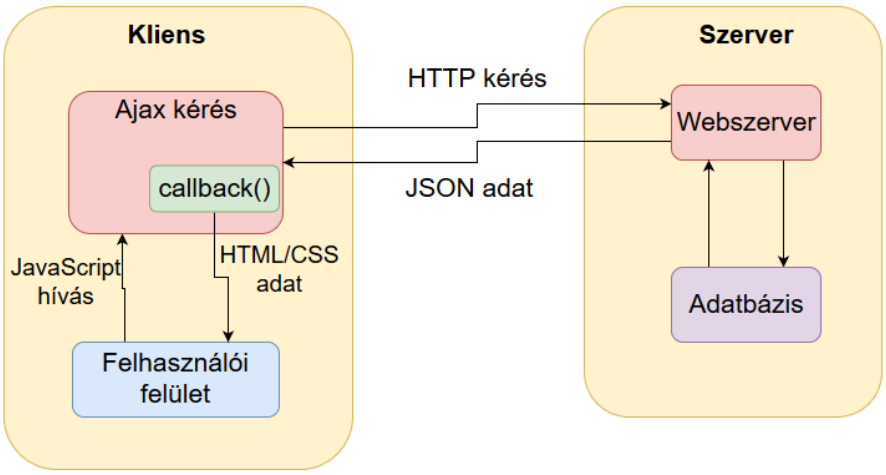
9.1 Ajax[29]

Az alkalmazás fejlesztése során, mind a mobil alkalmazás, mind a webes verzió esetén, az AJAX használata kulcsfontosságú. Az AJAX egy olyan webes technológia, amely a szerver és a kliens közötti adatcserét valósítja meg anélkül, hogy a webes felületet újra kellene tölteni, mindezt aszinkron módon. Ez azt jelenti, hogy amikor a kliens küld egy kérést a szervernek (például lekéri a kérdéseket), akkor a folyamat egy külön szálon indul el, így a háttérben futhat és várakozhat addig ameddig nem érkezik válasz. Ezalatt

[28] JSON: JavaScript Object Notation: egy szöveg alapú szabvány, amely egyszerű adatstruktúrák, asszociatív tömbök reprezentálására szolgál

[29] AJAX: Asynchronous JavaScript and XML, Aszinkron JavaScript és XML

a webes felület reszponzív marad, és továbbra reagál a felhasználó eseményeire. Természetesen ez különböző problémákat is okozhat, hiszen, ha egy program előbb dolgozna a lekért adatokkal, mint ahogy azok beérkeztek, akkor az hibákhoz vezet. Ennek a kiküszöbölésére használják az úgynevezett callback (visszahívás) függvényeket, amely akkor hívódik meg ha a kérésre valamilyen válasz érkezett. Jelen esetben ez kétféle lehet, sikeres vagy sikertelen. Sikeres kérés esetén a kapott adatokkal már dolgozhat a program, például a beérkeztek a kérdések, akkor indulhat a játék és az első kérdés betölthető.



8. ábra - Az AJAX működésének egyszerű vázlata

9.2 Webes verzió

A projekt tervezése és fejlesztése során fontos szempont volt, hogy az egyes platformokon egységesen nézzen ki és ugyanúgy működjön az alkalmazás. Az elsődleges platform a mobilalkalmazás, amely ezáltal mindig előrébb tartott a fejlesztés szempontjából és a webes verzió mindig utólag frissült. Mivel mind az alkalmazás, mind a webes felület ugyanúgy webes elemekből épül fel, így a kód egy-egy platformspecifikus dolgot leszámítva nagyjából azonos minden platformon. A webes felületen az alkalmazás egy container (tároló) divben található, amely rögzített képaránnyal és nagyjából fix mérettel középre igazítva jelenik meg a böngészőben.

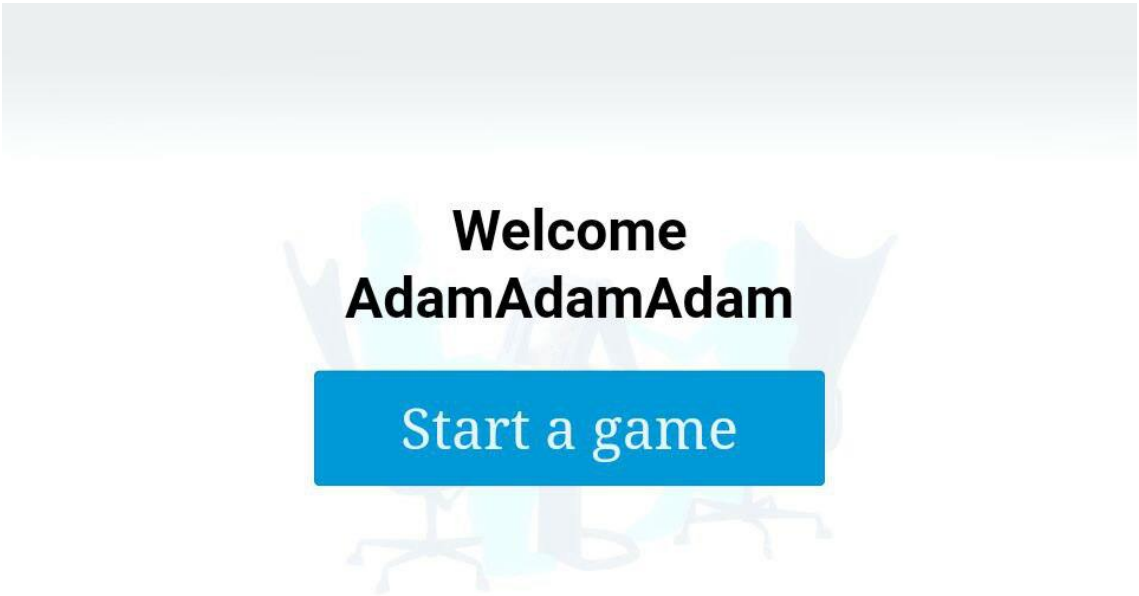
```
#screen{
    position:relative;
    margin:auto;
    width:50%;
    min-width:768px;
    max-width:50%;
    box-shadow:0 12px 15px 0 rgba(0,0,0,.24),0 17px 50px 0 rgba(0,0,0,.19);
}
```

4. kép - A container div CSS kódja

Böngésző esetén számolni kellett azzal, hogy az ablak átméretezhető, így az alkalmazás minimum szélességét meg kellett határozni. Ez 768 pixel lett, mivel ennél a szélességnél még nem esik szét a játék felülete. A maximális szélesség, pedig a képernyő 50%-a lehet. A mobilalkalmazás esetében a kijelzőméret adott, így ott nem kellett számolni az átméretezéssel és lehetett a teljes mérethez viszonyítani.

10. Tesztelés a projekt kritériumok szerint

A projekt megbeszélésekre tisztában kellet lenni, azzal, hogy hol tart pontosan a projekt, mi a következő lépés. Ezen feladatot Product Owner látta el. A backend tesztelés a már említett SoapUI programmal történt. Minden php fájl egyenként tesztelhető, szükséges beküldeni a megfelelő adatokat majd az érkező válasz kiértékelhető. Ugyanakkor a hibás adatok esetén is kell visszajelzést kapni melyek szintén tesztelhetőek ezen módszerrel. Elengedhetetlen a JS kód átfutása és fordításának ellenőrzése is, melyet a CSS kóddal egyidőben lehet tesztelni. A telefonos tesztek alatt több hiba kibukott melyeket a heti szinten esedékes megbeszéléseken megvitattunk kitűztük az adott hiba javításának határidejét és felelősét.



11. Projekt zárójelentés

Az általunk megvalósított projekt sikeresnek mondható!

.....

Nagy Ádám

Projektvezető