**Ez nem része a dolgozatnak, csak jelmagyarázat!!!!!!!**

**Kérjük ezt az oldalt törölni a dolgozatból!!!!**

[A szögletes zárójelben szereplők csak magyarázatként szerepelnek a sablonban. Azokat mindenképpen törölni kell!!!]

<A relációs jelek között szereplő szöveg helyére mindenkinek a saját magára vonatkozó adatokat kell beírni.>

{A kapcsos zárójelek közötti szöveg szükség esetén használandó.}

A sablonban található vörössel szedett szövegrészek színe csak a könnyebb felismerhetőség miatt színezettek. Kérjük azokat feketére állítani szíveskedjenek!

|  |
| --- |
| [Dolgozat gerincén, a fekete kötésen:] <Dolgozat száma> |

**SZAKDOLGOZAT**

**Tökös Ötös**

**DUNAÚJVÁROSI EGYETEM**



**MÉRNÖK INFORMATIKUS BSC**

# SZAKDOLGOZAT

Legyen ön is milliomos

**Tökös Ötös**

mérnök informatikus jelölt

# Projekt Alapító dokumentum

## 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.

## 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)

## Project tagok

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

## Projectvezető 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

## 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

## Out of Scope tevékenységek

* A mobil alkalmazások nyilvános terjesztése

## Projekt nagyobb összefüggésbe helyezése, kapcsolata más projektekkel

Nem függ össze.

## 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

## 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

## 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

## Várható befejezés

* **Folyó év ötödik havának ötödik napján**

## Szolgálati út (a projekt szervezeti felépítése)

Feladat-Felelősség

-operatív munkavégzés

-Erőforrások biztosítása, projekt-menedzser kinevezése

-Döntések meghozatala, projekt tervezés, operatív irányítás, szakmai felelősség

Projektszponzor

Projektvezető

Projekt-csapat

## 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 |

## 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

## 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.

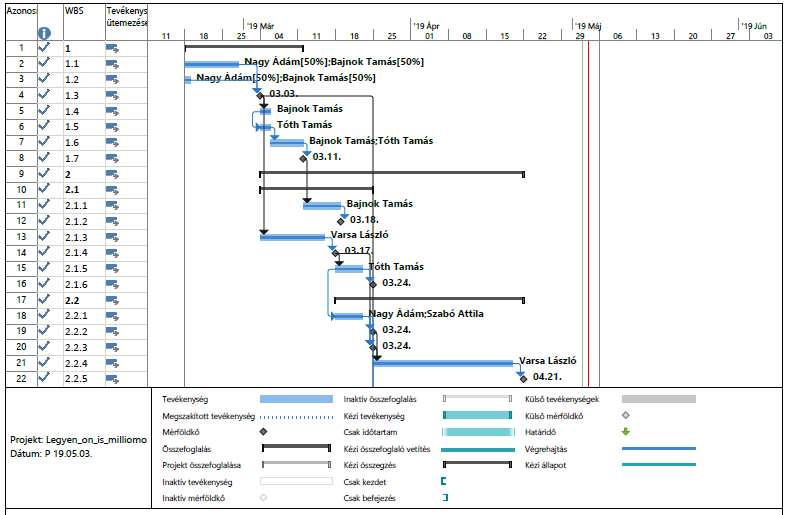
**I** - Informálni kell, ha döntés született. Mindig naprakész szeretne lenni ennek a tevékenységnek az előrehaladásáról.

## 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 |

**[Konzultációs igazolás a Szakdolgozat programból]**

## Gantt-diagram



## 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 |

**[Értékelő lap a Szakdolgozat programból]**

Kivonat

[Magyar nyelvű (max. 1 oldalas) összefoglaló]

Abstract

[Angol (kivételesen német) nyelvű, max. 1 oldalas összefoglaló]

Tartalomjegyzék

[Címsorokból generált tartalomjegyzék!]

# 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.

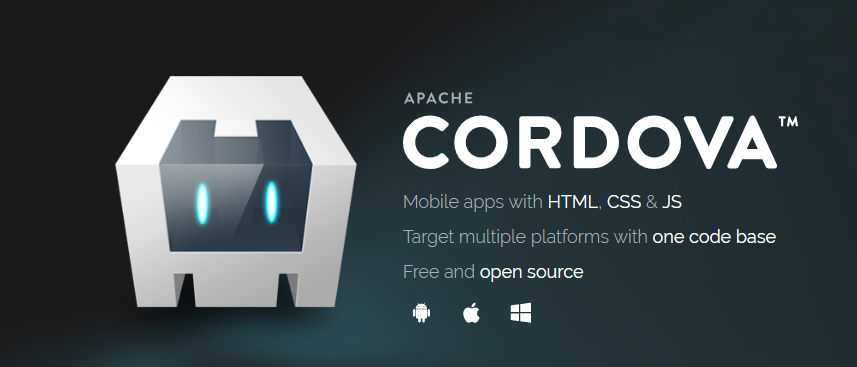
Első lépés ötlet keresés, mely során mérlegelni kell a felmerülő ötlet kivitelezhetőségét, egy eleve rosszul előadott ötletre nehezebb csapatot építeni. A Legyen ön is milliomos önmagában nem egyedi ötlet van már a piacon számtalan verziója sok csapat megvalósította már. Ami ezen tények ellenére is sikeressé 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.

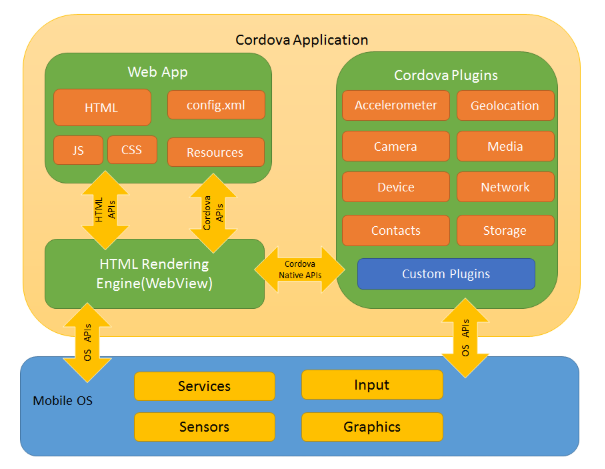
# A tervezés

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]](#footnote-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.

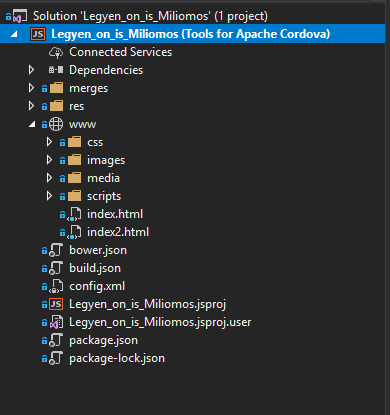
## Cordova



,,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]](#footnote-2)



. ábra Cordova felépítése

A Cordova projectet létrehozva az alábbi mappaszerkezet alakul ki

. ábra Cordova mappa struktúra

A WWW mappa alá kerül be az eszközökre befordítandó mappastruktúra.

A css mappa tartalmazza a kinézetért felelős css fájlokat.

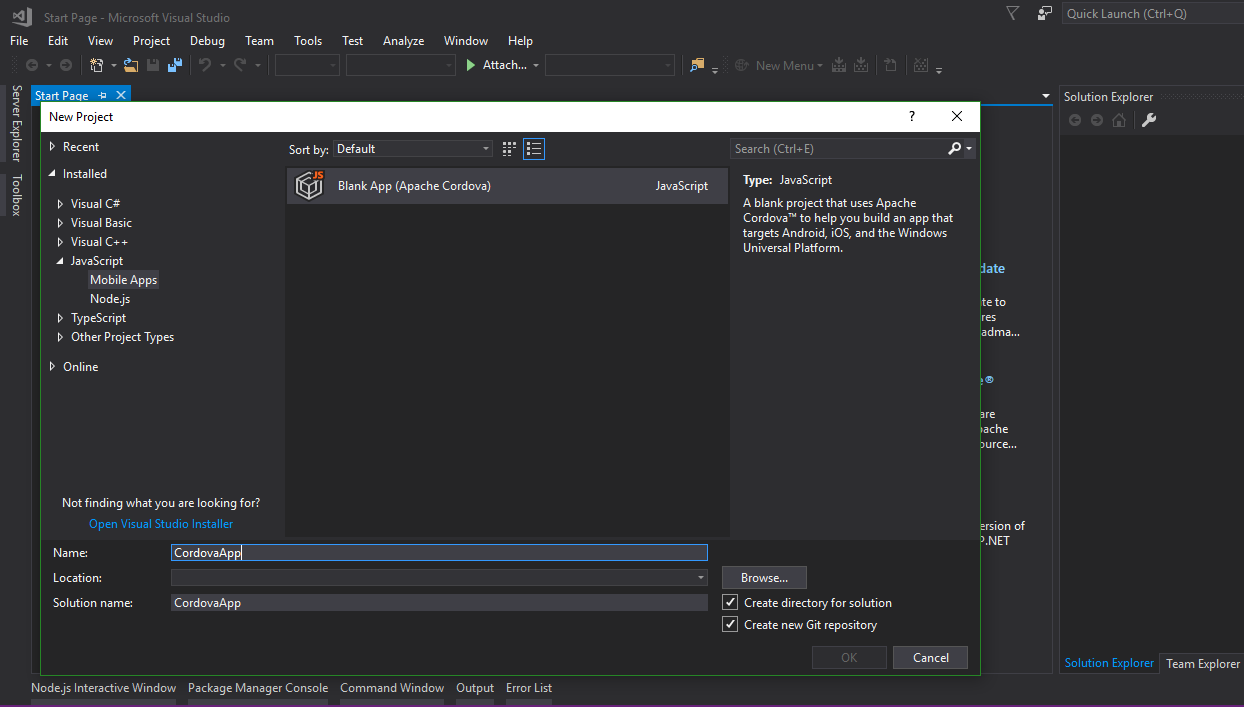
A media mappa az alkalmazásban előforduló képeknek ad helyet.

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

## Visual Stúdió

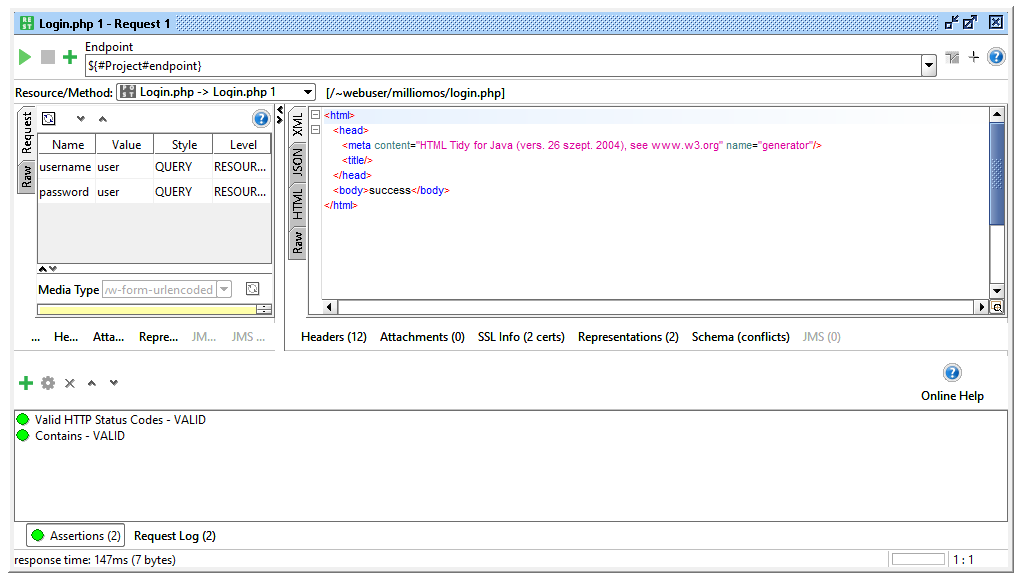
Fejlesztői eszköznek a Visual Stúdió-ra esett a választás, mely támogatja a Cordova projekt létrehozását, elkészít egy alap projektet mely készen is áll a fordításra mindhárom platformra. Ugyanakkor ez egy statikusan megjelenő oldal de a számunkra fontos már említett mappaszerkezetet legenerálja.

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



. ábra A létrehozás folyamata

## 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. 

. ábra Sikeres belépés

## A játék, és szabályai

A Legyen ön is milliomos egy televíziós kvízjáték [[3]](#footnote-3). 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 adnak helyes válasz, ilyenre példa a közönség szavazata.

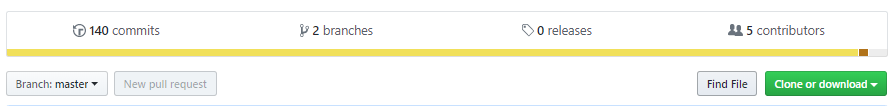
## A project

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.

## 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 project, erről a tárolóról le lehet menteni, és fel lehet tölteni a változtatásainkat, amit program kódján vagy különböző részein hajtottunk végre.

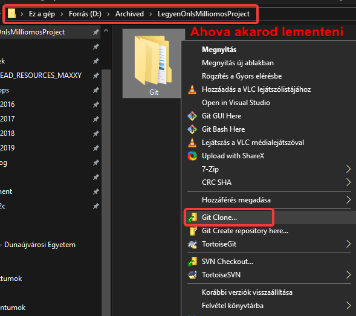
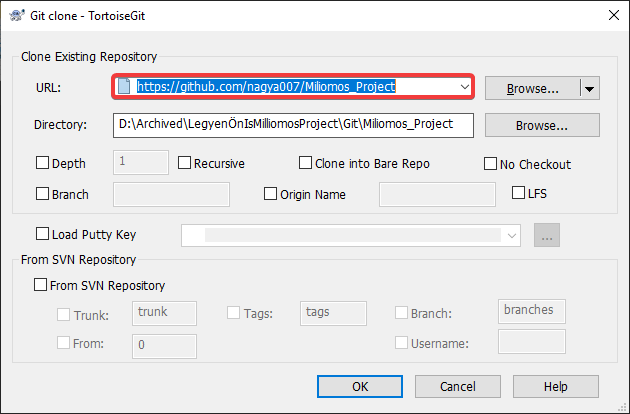
Az irányítás alatt pedig arról beszélünk, hogy minden változást, amit a fejlesztők végre hajtanak, azt menti az internetes tároló. Így lehetőség van nyomon követni változásokat, esetleg visszavonni változásokat.



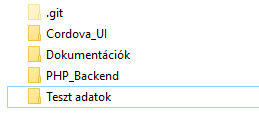
. ábra A Git követi a válltozásokat

* 1. Használata a Legyen ön is Milliomos projectben

A Git-et lehet használni a hozzávaló terminállal, vagy *TortoiseGit* [[4]](#footnote-4)használatával is. TortoiseGit-el Windows alapú GUI-t kapunk a Git használatához. Emellett rendelkezni kell egy Git felhasználóval is, aminek van engedélye a project letöltéséhez. A project tagok az e-mailjük segítségével regisztráltak a Git honlapjára.[[5]](#footnote-5)



A távoli tárolóról letöltött projectet 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 TortoiseGit lehetővé teszi ennek a folyamatnak a felgyorsítását, mivel egy GUI[[6]](#footnote-6)-t ad a Git szoftvernek. Így elég gomb kattintással fel és letölteni a project filekat.

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. 

. ábra Projekt mappaszerkezet

## Dokumentálások

Project 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.

# Frontend

## A frontend felépítése

### 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.

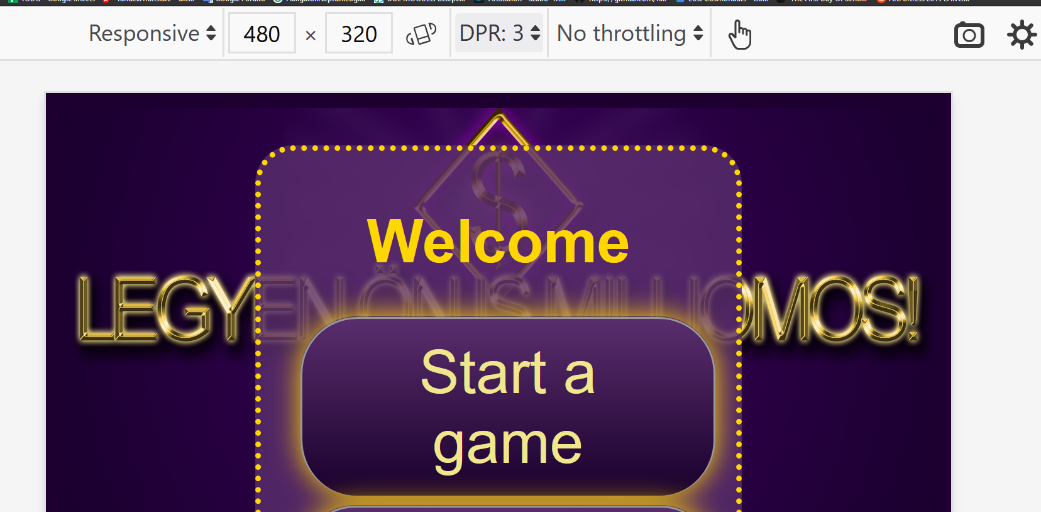
A HTML lehetővé teszi az adatok csoportosítását, megjelenítését megfelelő helyen, és egyéb formai dolgok, például színezés és méretezés. Emellett beépített funkciók és eszközöket is lehet használni vele, például gombok vagy szövegbevitelre mezőt, ahova képes a felhasználó írni. Ezeket az úgynevezett „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,

### Frontend a Legyen ön is Milliomos applikációban



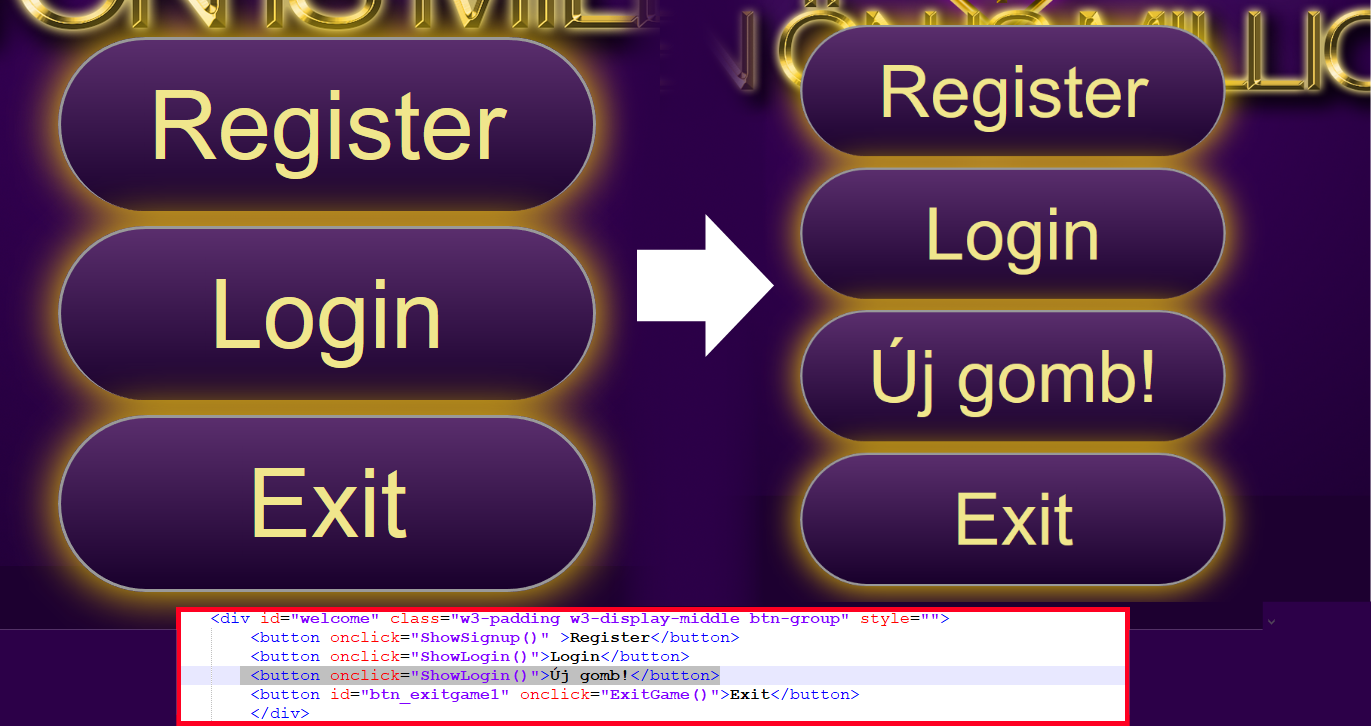
A frontend fejlesztését lehetővé teszi a Cordova, mivel a HTML web programozási nyelvet használja. A Cordova egy futó web applikáció, ami képes a telefonos verzió kezelésére. A Cordova képes mindenféle funkciót kihasználni a mobilokból, ezalatt a billentyűzetet, kamerát, GPS és forgatást képes kezelni.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

A fejlesztéshez szükség van egy IDE [[7]](#footnote-7)szoftver letöltéséhez. A legyen ön is milliomos project 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.[[8]](#footnote-8)



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. [[9]](#footnote-9)

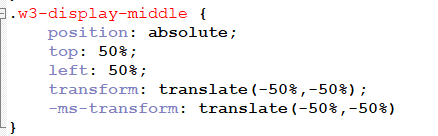
A megfelelő arányokat a HTML képes betartani, ha az objektumokat a képernyőhöz viszonyítva számoljuk. Ezt úgy lehet elérni, hogy folyamatosan rendezzük a csoportokat, például a képernyő közepére, szélére, legaljára rendezzük az objektumokat. Innentől fogva az applikáció minden platformon megfelelő méretezéssel 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.



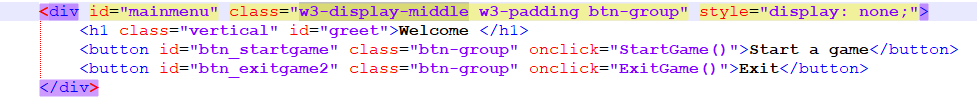
*Új gomb a HTML kódba, megfelelő rendezéssel, csak egy darab új sor kellett*

### 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.[[10]](#footnote-10)



*A középre rendezés tulajdonságai*



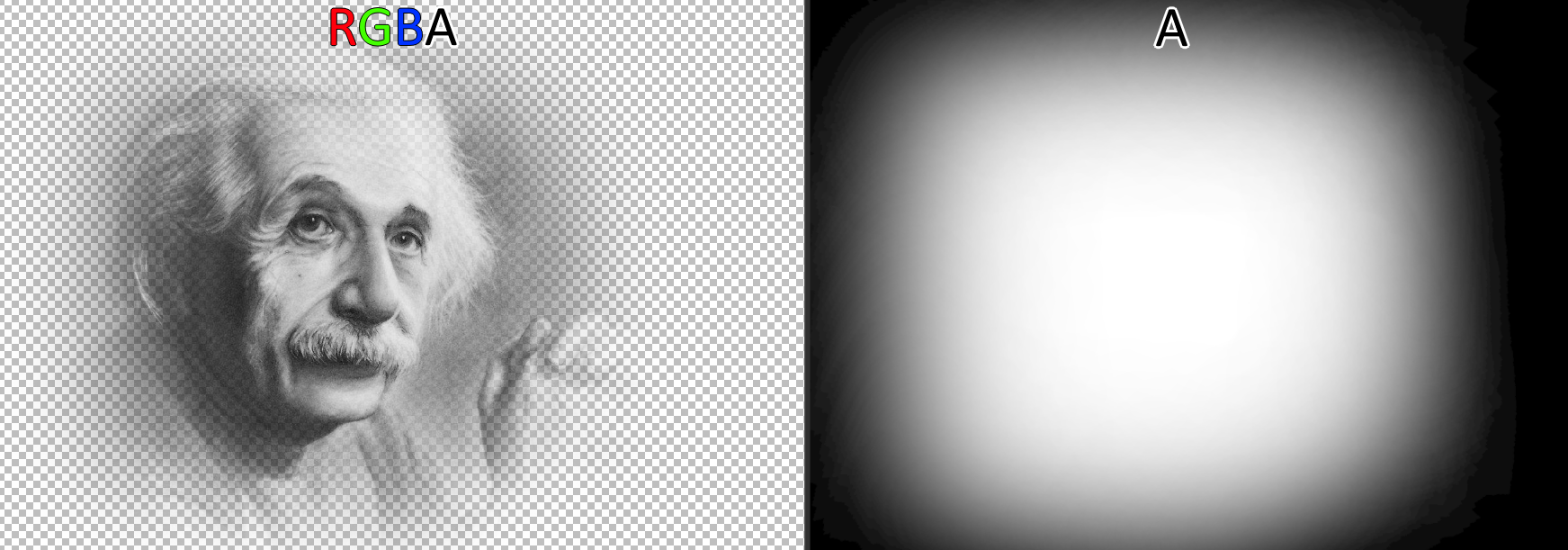
*HTML-en belül, csoport (div) középre van rendezve*

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.

* 1. Formázás, Média

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.

 [[11]](#footnote-11)

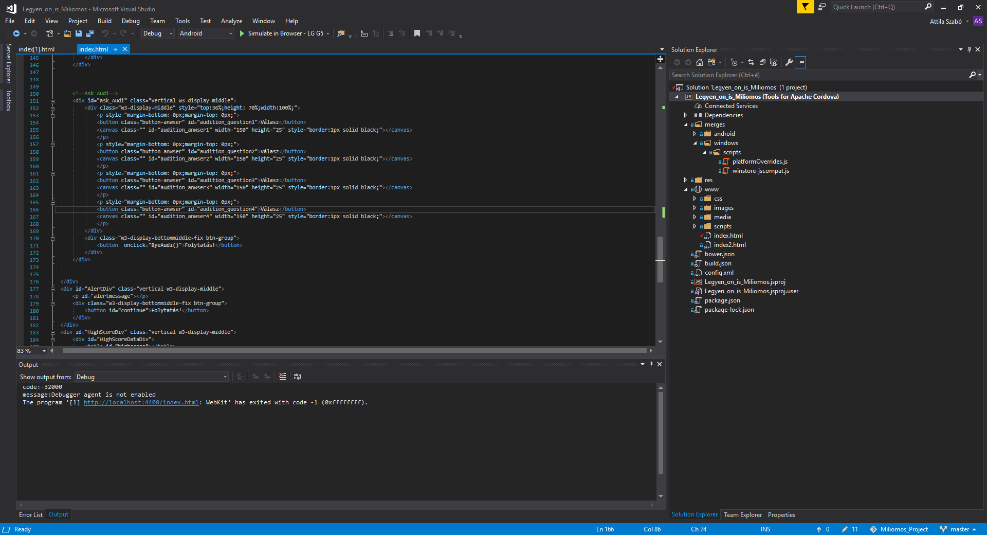
*Alpha Fekete rész = Áttetsző, összeolvad a weboldal színével*

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.

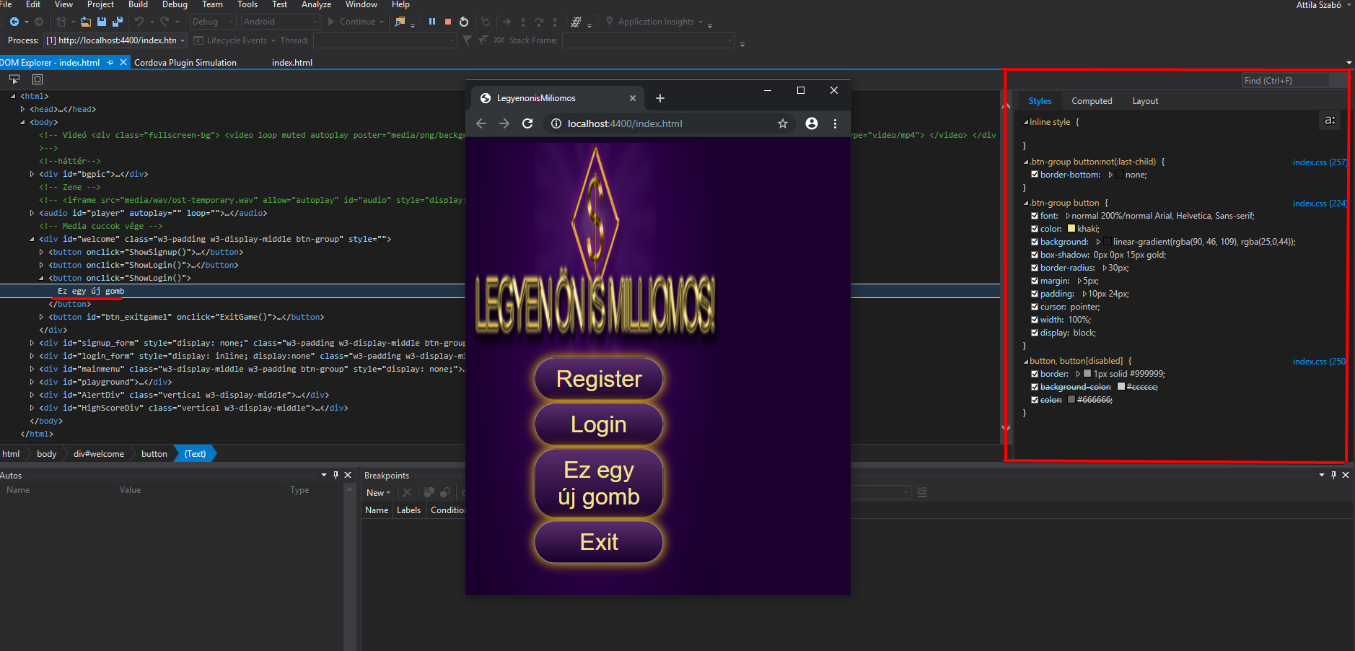
### Frontend tesztelés

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[[12]](#footnote-12). A tesztelésre több módszer is lehetséges:

### Visual Studio

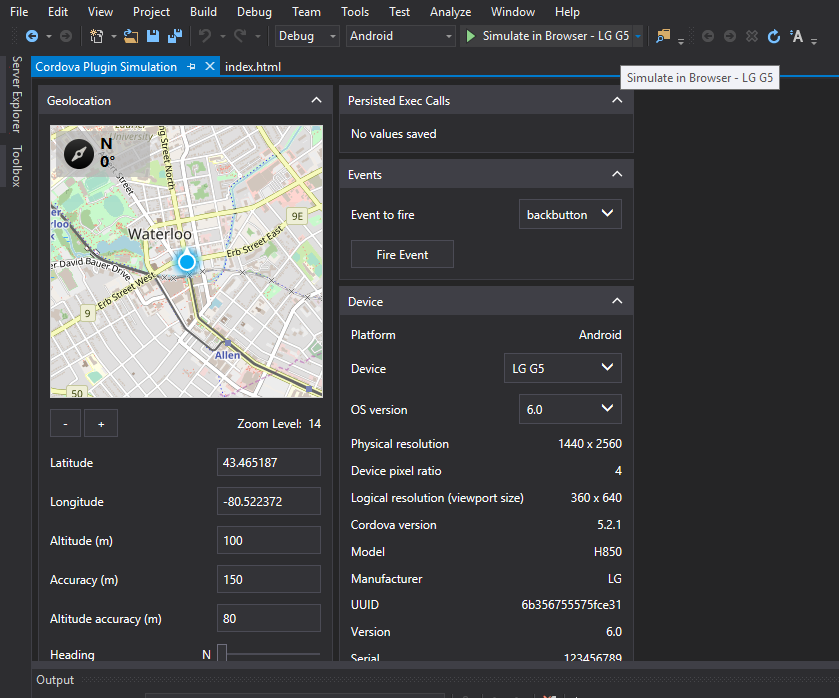


*A Visual Studio file, miután letöltöttük Gitről*



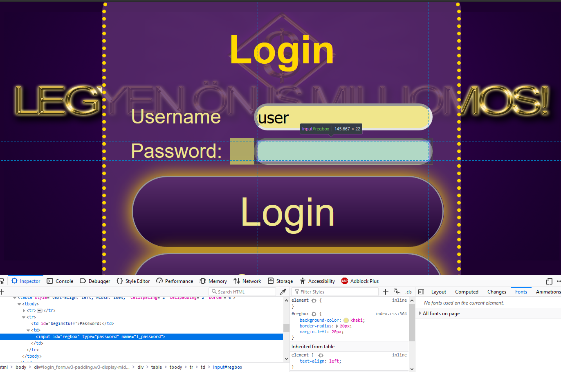
*Valós idejű tesztelésre példa, új sorok egyből megjelennek az applikációban, Visual Studio sajna nem engedi a virtuális képernyő forgatását*

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 [[13]](#footnote-13)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.



*Android Operációs rendszer beállítása, vagy képernyő arány*

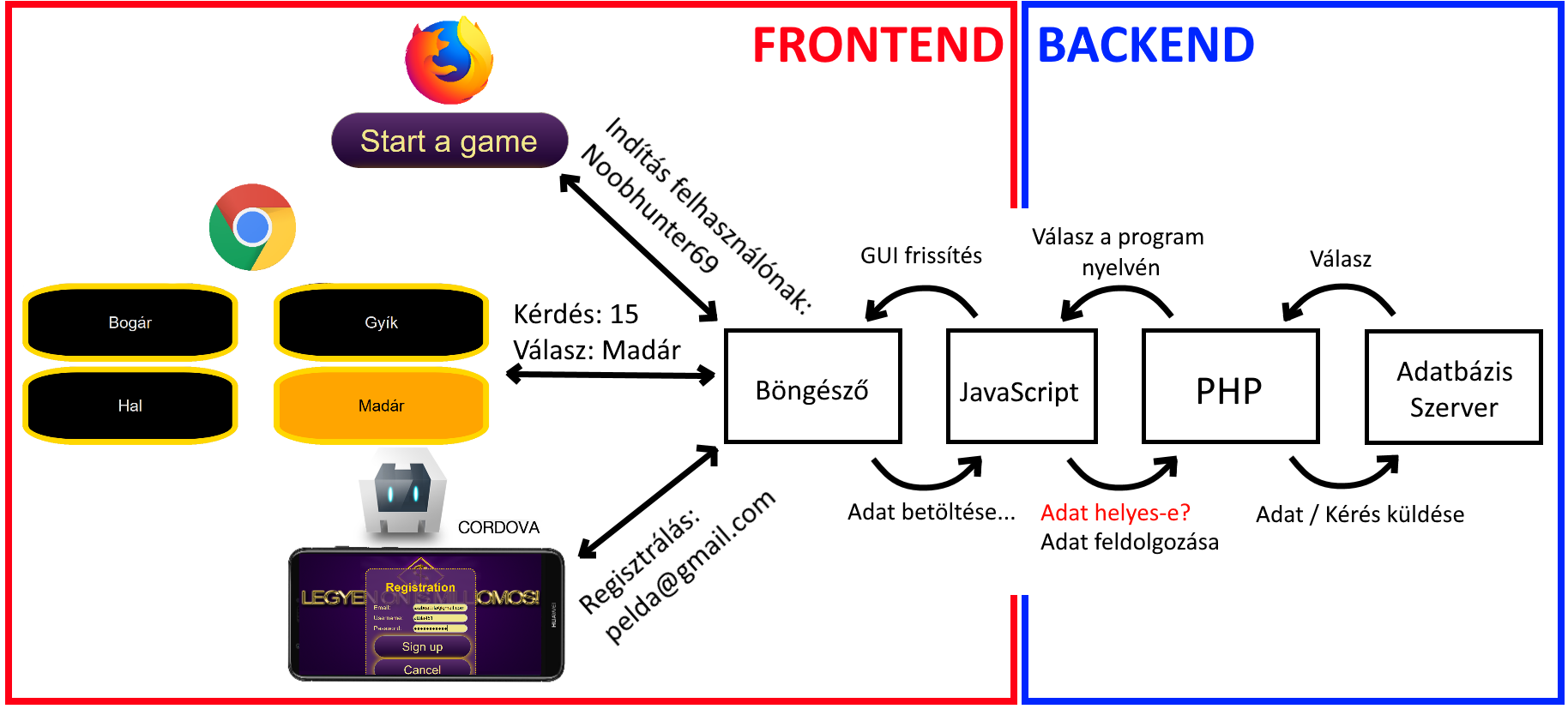
### Böngésző fejlesztői opció



*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.

## Fejlesztés



### 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.

### Backend fejlesztés

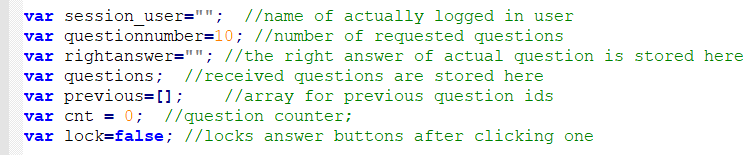
Elsősorban az applikáció „agya” kell, ahol tárolni lehet a játékos teljesítményét, és bővíteni lehet a játék során előforduló kérdéseket. Ezt megelőzi az adatbázis tervezése. Fontos tudni, hogy milyen adatokkal akarunk dolgozni, később ezeket kiegészíteni már nehezebb. A szerveroldali kommunikációnak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

### Frontend fejlesztés

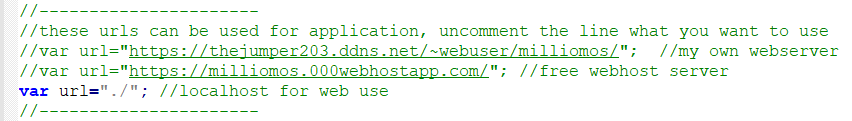
Egy működő alappal már lehet fejleszteni a frontendet. Itt jön figyelembe a tesztelés hangsúlya, mivel van lehetőség arra, hogy a kód futtatása közben fejlesszük a felhasználói felületet. Az adatbázist ilyenkor nem kell már szerkeszteni, később esetleg lehet bővíteni, de több lehetőségünk van a frontenden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

# 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.



## Webszerverek elérhetősége:



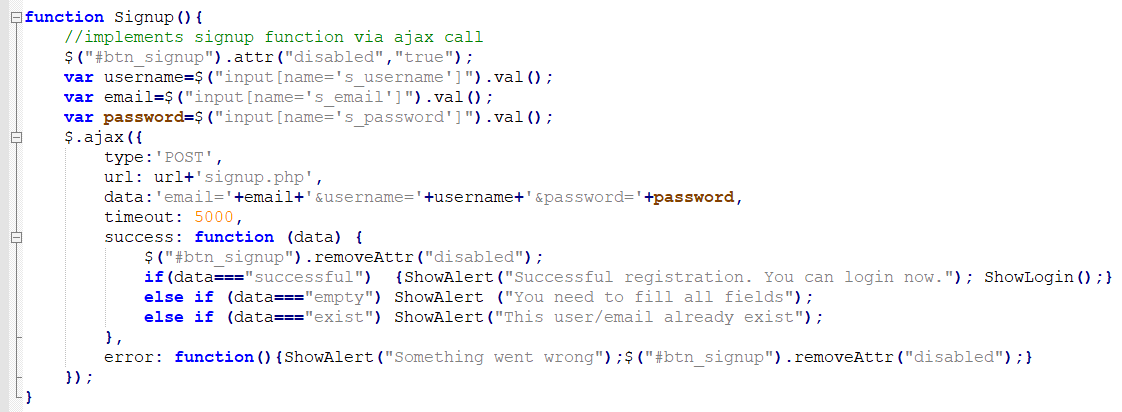
Kommentek törlésével könnyen állítható, hogy melyik webszerverre szeretnénk csatlakozni. Illetve localhost indítás is rendelkezésünkre áll.

## 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.

## Regisztráció

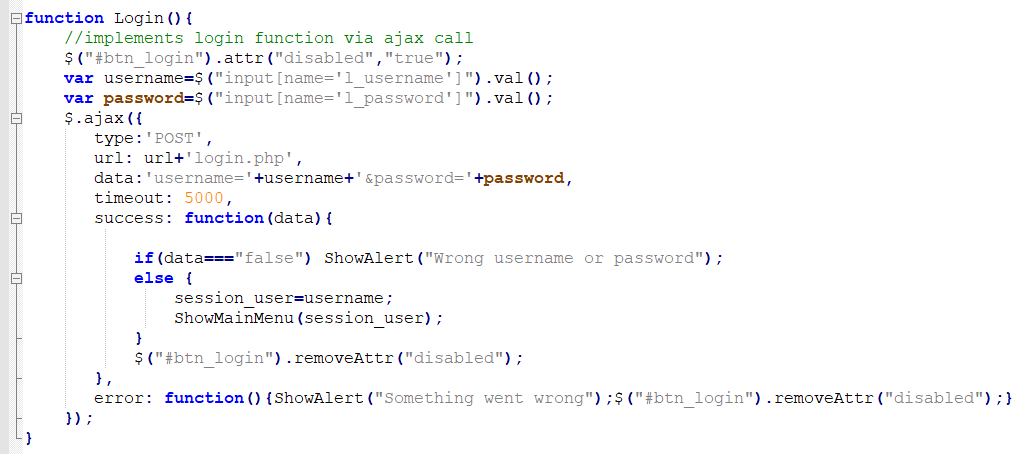
Regisztráció során csak pár új adatot kell az adatbázisba küldeni.



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.

## Bejelentkezés

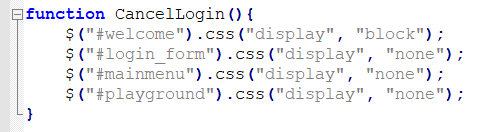
A bejelentkezés hasonló a regisztrációhoz az ajax szempontjából. Itt is adatot küldünk a backend részére. Típusa tehát POST. AZ URL a login.php fájl. Az adatokat ugyan úgy stringként összefűzve küldjük el.



## UI

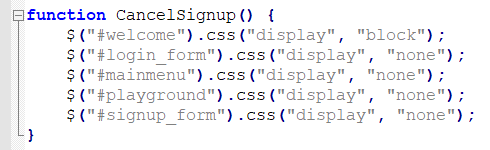
A játék élvezhetősége érdekében egy html oldal fut végig és ezen történnek a változások a különböző függvények segítségével. Néhány példa:

CancelLogin() metódus eltűnteti a a login formot és a kezdőképernyő állapotát hozza vissza.

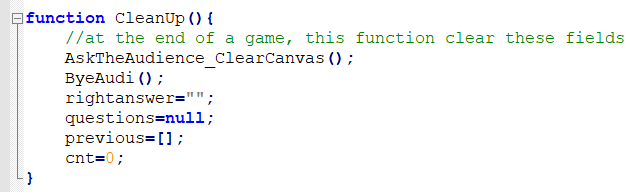


## CancelSignup()

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

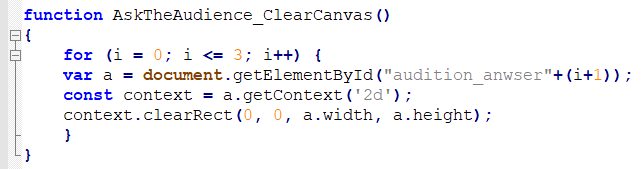


CleanUp() metódus a játék végén hívódik. Ennek az a feladata, hogy visszaállítson különböző kezdőértékeket, pl.: a jó válasz tárolását, a kérdéseket tároló tömböt, előző kérdés ID-ját tároló változót, kérdések számlálóját.



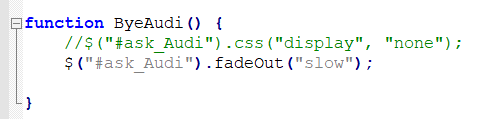
## AskTheAudience\_ClearCanvas()

Ezen kívül itt hívódik meg az AskTheAudience\_ClearCanvas() nevű metódus, ami a közönség segítség kezdőértékeit állítja vissza.



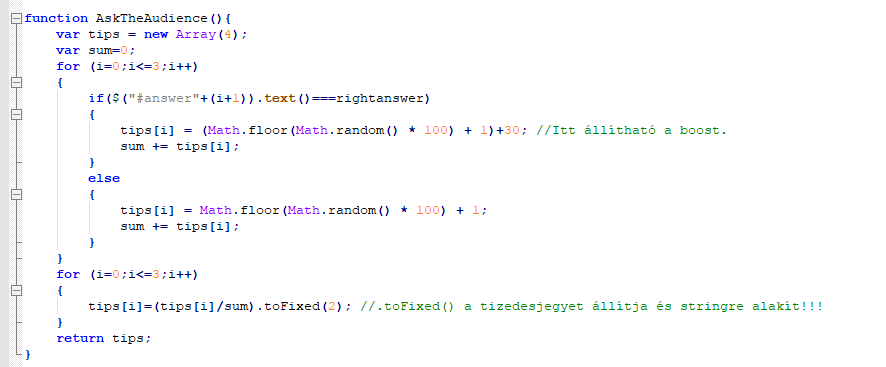
## ByAudi()

A metódus eltűnteti a közönség segítség megjelenítését.

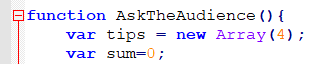
Segítségek

## 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.



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



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.

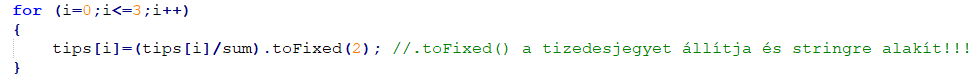


Ha egyezik, akkor a jó válasz gombjánál jár a ciklus. Tehát létrehozhatunk egy random számot a Math.floor(Math.random())-mal 1 és 100 között és hozzáadhatunk egy bizonyos értéket amivel növelhetjük a közönség „tudását”.



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.

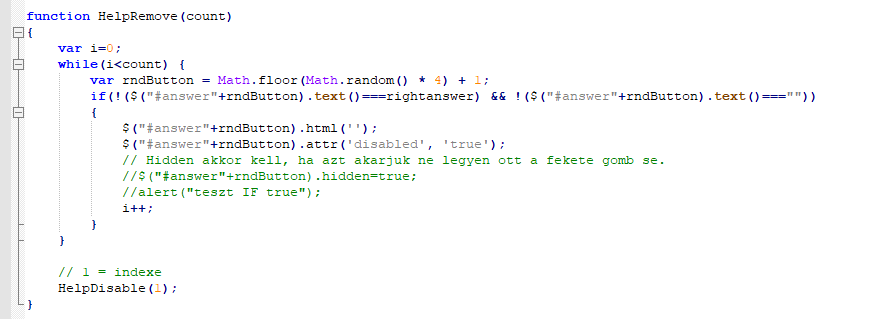


Hogy 2 tizedesig írjuk ki a számokat .toFixed(2) metódusát használtunk. Itt figyelni kell rá, hogy stringre változtatja a számokat.

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

## Felezés:

A funkció két rossz választ vesz el véletlenszerűen.



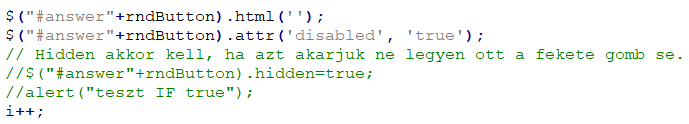
Először felveszünk egy *i=0* változót a ciklus lépéseinek követése végett. Majd egy *count* (alap értéke 2) nevű bemeneti paraméterrel ellátott while ciklusban generálunk 1 és 4 között véletlenszerűen egy számot (*rndButton* változóba), ami a gomboknak a számát fogja megadni.



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



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.



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.

## 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.



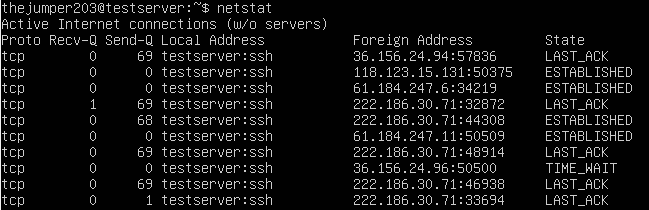
# Szerveroldal

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.

## Webszerver

A kliens oldalról érkező HTTP kéréseket egy Apache[[14]](#footnote-14) 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[[15]](#footnote-15) 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.



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-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.

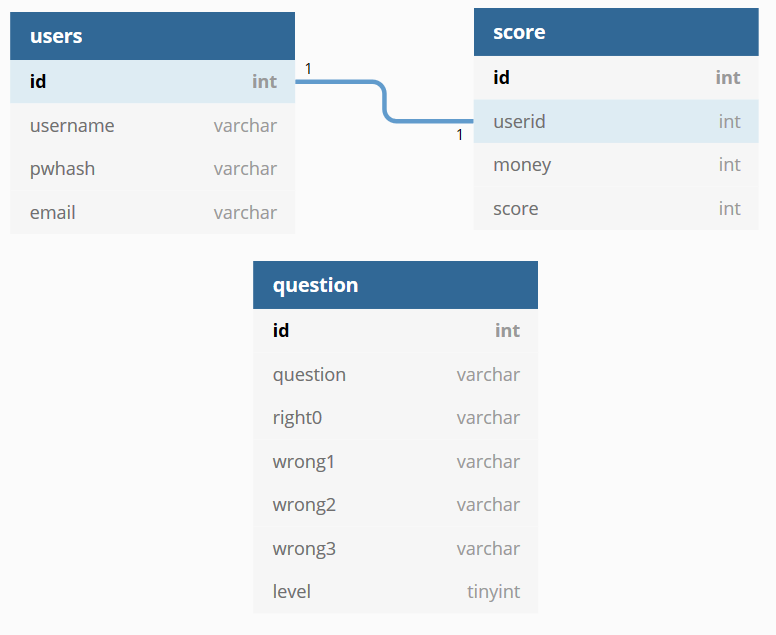
## Adatbázis

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 |



1. 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.

## PHP

### 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.

### 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.

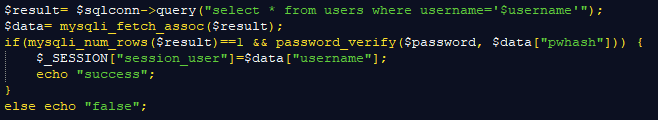
Biztonsági okokból a jelszó nem kerül tárolásra az adatbázisban, hanem csak annak a lenyomata, így még az sem tudhatja meg a jelszót, aki hozzáfér az adatbázishoz. A ténylegesen eltárolt karakterláncot az alábbi kódsor hozza létre:

$pwhash=password\_hash($password,PASSWORD\_DEFAULT);

A password\_hash() egy beépített PHP függvény, amely egy karaktersorozat lenyomatát készíti el.

### login.php

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

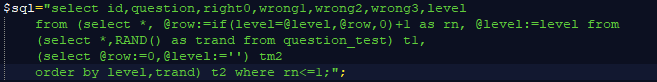


2. kép - A megadott felhasználónevet és jelszót ellenőrző kódrészlet

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 hash-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.

### 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.



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[[16]](#footnote-16) formátumba kódolja és visszaküldi a kliensnek, ami ezt feldolgozza és tárolja.

### getScores.php

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

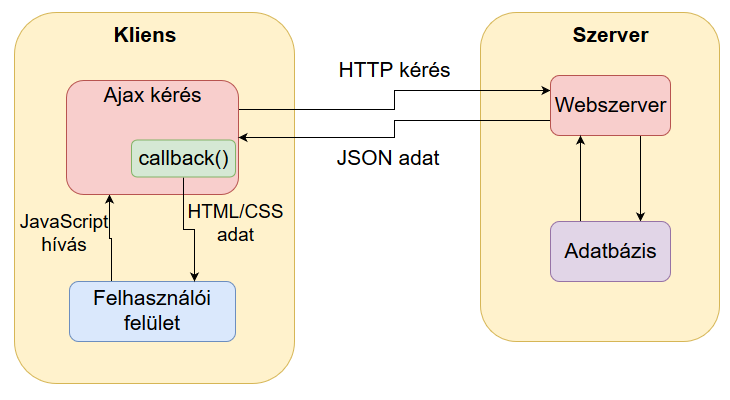
### highscore.php

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.

# Kliensoldal

## Ajax[[17]](#footnote-17)

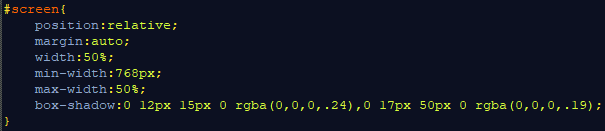
Az alkalmazás fejlesztése során, mind a mobil alkalmazás, mind a webes verzió esetén, az AJAX használata kulcsfontosságú. Az AJAX egy olyan webes technológia, amely a szerver és a kliens közötti adatcserét valósítja meg anélkül, hogy a webes felületet újra kellene tölteni, mindezt aszinkron módon. Ez 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 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.



2. ábra - Az AJAX működésének egyszerű vázlata

## 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.



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.

Általános Projekt jelentés Dátum:

A Projekt jelenlegi mérföldköve:

Dokumentáció átadása

A projekt terv szerint halad: Igen / Nem Ha nem a késlekedés okai: .................................................................................................................................................... .................................................................................................................................................... .................................................................................................................................................... A okok részletes kifejtése: ..................................................................................................................................................... ..................................................................................................................................................... ………………............................................................................................................................. ...................................................................................................................................................... ..................................................................................................................................................... ..................................................................................................................................................... ……………….............................................................................................................................

A jelentés készítője:

.............................................

Beosztás

.............................................

Aláírás

[A felhasznált szakirodalmat []-ben sorszámozottan – a szerzők szerint ABC sorrendben, egyértelműen azonosíthatóan (szerző, cím, kiadó, kiadás helye, kiadás éve) – kell feltüntetni.]

{Ábrajegyzék/Táblázatok jegyzéke/Mellékletek jegyzéke/stb.}

[A dolgozat ábrái/táblázatai/mellékletei/stb. felsorolva típusonként, ha vannak. Törekedni kell a szövegszerkesztő program által automatikusan generált jegyzék használatára! Elvárt, hogy minden kép/ábra/táblázat/egyenlet/stb. kapjon sorszámot (típusonként külön számozva), amelyre a dolgozat szövegében kereszthivatkozással történjen a hivatkozás!]

Mellékletek

[A dolgozat mellékletei sorszámozottan, ha vannak. A Mellékletek jegyzéke hivatkozzon az adott mellékletre!]

1. https://tudasbazis.sulinet.hu/hu/szakkepzes/informatika/a-program-gyakorlati-megvalositasa/szamitogepes-programozas-a-gyakorlatban/a-vizualis-fejlesztoeszkozok-megismerese-hasznalata [↑](#footnote-ref-1)
2. https://cordova.apache.org/ [↑](#footnote-ref-2)
3. https://hu.wikipedia.org/wiki/Legyen\_%C3%96n\_is\_milliomos! [↑](#footnote-ref-3)
4. https://tortoisegit.org/ [↑](#footnote-ref-4)
5. https://github.com/ [↑](#footnote-ref-5)
6. Graphical user interface [↑](#footnote-ref-6)
7. Integrated Development Environment [↑](#footnote-ref-7)
8. https://notepad-plus-plus.org/ [↑](#footnote-ref-8)
9. https://www.w3schools.com/cssref/css3\_browsersupport.asp [↑](#footnote-ref-9)
10. https://www.w3schools.com/w3css/w3css\_display.asp [↑](#footnote-ref-10)
11. https://opimedia.azureedge.net/-/media/images/utr/editorial/articles/online-articles/2014/04-01/albert-einstein-the-humanitarian/albert-einstein-jpg.jpg [↑](#footnote-ref-11)
12. http://softwaretestingfundamentals.com/white-box-testing/ [↑](#footnote-ref-12)
13. Egy másik készülék használatát utánozza [↑](#footnote-ref-13)
14. Apache HTTP szerver: egy nyílt forráskódú webkiszolgáló alkalmazás [↑](#footnote-ref-14)
15. DNS: Domain Name System, magyarul tartománynévrendszer [↑](#footnote-ref-15)
16. JSON: JavaScript Object Notation: egy szöveg alapú szabvány, amely egyszerű adatstruktúrák, asszociatív tömbök reprezentálására szolgál [↑](#footnote-ref-16)
17. AJAX: Asynchronous JavaScript and XML, Aszinkron JavaScript és XML [↑](#footnote-ref-17)