**Tartalomjegyzék**

1. Bevezetés 2

Követelményspecifikáció 2

2. Fejlesztői dokumentáció 2

Fejlesztői eszközök 2

Használt technológiák 2

Futtatási környezet 2

3. Fejlesztés menete 3

Piackutatás 3

Adatbázis megtervezése/elkészítése 3

A weblap elkészítése 5

Weblap function-ök 5

A játék megtervezése/elkészítése 10

A játékról: 11

Használt algoritmusok 11

4. Tesztelés és annak menete 11

Weboldal tesztelése 11

Játék tesztelése 11

5. Linkek és források a játék elkészítéséhez 12

Zenék, hangeffektek: 12

Modellek, textúrák, effektek, animációk: 12

6. Továbbfejlesztési lehetőségek 13

7. Összegzés 14

8. Ábrajegyzék 14

1. Bevezetés

A szakdolgozatunk témáját hárman közös döntés alapján választottuk a legkedveltebb szabadidős foglalkozásunk alapján, egy játékot fejlesztettünk.

A játék kiválasztásakor fontosnak tartottunk, hogy hasonlítson a mai közkedvelt játékstílusokhoz, ezért egy zombis, lövöldöző játékot fejlesztettünk, amiben a cél az, hogy minél tovább életben tudjunk maradni egy posztapokaliptikus térben, ahol folyamatosan életünkre törnek a zombik.

A játék fejlesztése mellett készítettünk egy weboldalt is, ahol a játék letöltése mellett sok, mellékes tevékenységet is kellemesen és könnyedén végezhetünk, mint például csevegés, profil megnézése/szerkesztése.

A weboldal könnyedén kezelhető és a játékot is úgy fejlesztettük, hogy könnyedén beleszokjon bárki, aki elkezd vele játszani.

Követelményspecifikáció

Célunk az volt, hogy egy olyan játékot fejlesszünk, amit bárki tud élvezni és egy olyan weboldalt készítsünk el mellé, amit a lehető legegyszerűbben lehessen kezelni, éppen ezért a játék az átlagos shooterek alap beállításait használja a billentyűzet és egér kombinációval, a weboldal pedig egy nagyon átlátható felület.

* A weboldalt bárki elérheti, akinek van hozzáférése az internethez.
* A weboldalra bárki ingyenesen tud regisztrálni.
* A játékot szabadon le tudja tölteni mindenki aki regisztrált a weboldalon, ingyenesen.

1. Fejlesztői dokumentáció

Fejlesztői eszközök

* Kódszerkesztő: Visual Studio Code, Unity
* Adatbázistervező: MySQL

Használt technológiák

* Programozási nyelvek: Laravel php
* Webes technológiák: CSS, Bootstrap, HTML
* Adatbázis motor: MariaDB

Futtatási környezet

A weboldal minden böngészőben elfut, akár még telefonon is, de ajánlott asztali böngészőt használni, mivel a játékkal csak számítógépen lehet játszani, viszont minden egyes operációs rendszeren (Windows, Linux, Mac, stb.) elérhető a játékunk.

1. Fejlesztés menete

Piackutatás

A weblap elkészítése előtt időt fektettünk arra, hogy utánanézzünk hasonló weboldalaknak, ezekből merítettünk ötleteket, inspirációt. A weboldal alap kinézete, stílusa ezekből lett összetéve kisebb-nagyobb változtatásokkal. A játék leprogramozása előtt szintén utána néztük különböző, kisebb Unity-s projekteknek és különböző megoldás alapoknak, hogy minél könnyebb és praktikusabb legyen az elkészítés.

Adatbázis megtervezése/elkészítése

Az adatbázist MySQL-ben terveztük meg, de phpmyadminban hoztuk létre, így utólag könnyebb és kényelmesebb volt dolgozni a weboldallal Visual Studio Code-ban. Az adatbázis tervezése során törekedtünk arra, hogy a felépítése logikus, könnyen átlátható legyen. Az adatbázis 3 táblából áll, ugyanis a weboldal nem rendelkezik annyira sok és összetett funkcióval. amihez érdemes lett volna több táblát használni.

Messages tábla

|  |  |
| --- | --- |
| id | szám, auto increment, a tábla egyedi azonosítója, elsődleges kulcs |
| senderid | szám, az üzenetet küldő fél azonosítója, idegen kulcs |
| recieverid | szám, az üzenetet fogadó fél azonosítója, idegen kulcs |
| message | szöveges adat, nem lehet üres, az üzenet amit írni szeretne a játékos |
| date | dátum, nem lehet üres, az üzenet elküldésének ideje |
| prevmessage | szám, alaphelyzetben az értéke NULL, az előző üzenetek száma |
| status | enum, nem lehet üres, megadja, hogy az adott játékos elolvasta-e az üzenetet |

Profiles tábla

|  |  |
| --- | --- |
| id | szám, auto increment, a tábla egyedi azonosítója, elsődleges kulcs |
| userid | szám, nem lehet üres, a játékos egyedi azonosítója, idegen kulcs |
| age | szám, nem lehet üres, a játékos életkorát tárolja |
| telszam | varchar, nem lehet üres, a játékos telefonszámát tárolja el |
| kedvetel | szöveges, nem lehet üres, a játékos beazonosítása érdekében eltárolt kedvenc étele |
| bio | szöveges, nem lehet üres, egy általános leírás amit a játékos ír magáról |
| gender | enum, nem lehet üres, eltárolja a játékos nemét |

Users tábla

|  |  |
| --- | --- |
| id | szám, auto increment, a tábla egyedi azonosítója |
| name | szöveges, nem lehet üres, a játékos neve |
| email | szöveges, nem lehet üres, a játékos e-mail címét tárolja el |
| password | szöveges, nem lehet üres, a bejelentkezéskor használt jelszót tárolja el |

Ezek a táblák csak a weboldalhoz kapcsolódnak, magához a játékhoz nem, ugyanis a játékból nem tárolunk semmit. A táblákat az idegen kulcsok segítségével kötöttük egymásba, az elsődleges kulcsként mindegyik tábla egy id-t kapott. Ez lényegében csak nekünk tette könnyebbé a munkát, más célja nincsen. Az életkort azért tartottuk fontosnak lekérdezni és eltárolni, mert alapvetően a játék egy zombis, shooter játék, amit egy bizonyos életkor felett illene játszani.

A weblap elkészítése

A weblapon nagyrészt Csongor dolgozott, a weblap alapjaként egy régebbi munka bootstrapjét alkalmaztuk.

A weblap végleges kinézetének és funkcióinak megtervezése közös ötletelések alapján lettek véglegesek. A weblapot Laravel php-ban tettük össze, mert a programozói környezete nagyon egyszerű és praktikus egy ilyen típusú weboldalnál.

Weblap function-ök

A weblapnak a program kódjában több function is szerepet kapott a fejlesztéskor:

* User regisztrációs function:

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

1. ábra UserController és functionök

A UserControllerünkben, ahol a regisztrációt és annak validációját futtatjuk megtalálható a Reg function és a RegData.

Az utóbbi validálja, hogy megfelelő formátumúak legyenek a következő adatok: name-email-password. A name-t elláttuk egy *.unique* taggel, ami azért felelős, hogy ne lehessen ugyanazzal a felhasználónévvel két különböző usert regisztrálni. Sok más validációs megoldást is megnéztünk, ami jó lett volna, de úgy döntöttünk, hogy ez lenne a legpraktikusabb, mivel a legtöbb ilyen fajta weboldal is ezt a validációt alkalmazza.

* NewProfile (a felhasználók másodlagos adatai) function:

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

2. ábra NewProfile function és validációk

A NewProfile function azt a célt szolgálja, hogy a beregisztrált felhasználóknak legyen saját adatuk, a regisztráció után egyből erre returnol a RegData. Ahogyen az az előbb is megtörtént, a felhasználó által megadott adatok fel fognak kerülni az adatbázisba, de csak helyes formátumba mivel itt is sok féle validáció történik a $data->Save() előtt, például minimum, maximum karakterek, formátumok string vagy int. Azt is megcsináltuk, hogy ezt az oldalt csak olyan felhasználó láthassa aki már regisztrálva van az oldalon az Auth::user()-id –val ezt később csak Authnak fogom megemlíteni. Ez az Auth azért fontos, hogy a felhasználók adatai biztonságban legyenek és ne lehessen illetéktelenül az adatbázisba beleírni.

* A profil adatok módosító functionje(ProfileMod):

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

3. ábra ProfileMod function és validációk

Ebben a functionben is először lefut az Auth, majd a felhasználó a meglévő adatait tudja módosítani ezzel a functionnel. Az adatbázisból először lekérjük az id alapján az adatokat, hogy a felhasználó lássa majd a telefonszámot a kedvenc ételt és a felhasználó bióját. Ezek az adatok módosíthatók, és ezekért a módosításokért is a validációk felelnek a function-ön belül. A felhasználónév azért nem változtatható, mert maga a felhasználónév csak a játékban megjelenített nevedet képviseli, ami nem annyira létfontosságú, hogy erre is külön validációt és functiont alkalmazzunk.

* Login function:

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

4. ábra Login function és validációk

A login funkció nem annyira bonyolult, lefutnak a szükséges validációk és lekéri a funkció az adatbázisból, hogy van-e ilyen email és van-e ilyen password. Ha mind a kettő létezik és megegyezik a felhasználó által megadott adatokkal, az adatbázisban lévő adat vissza adja a / (welcome) viewt ha ezesetben nem jó valamelyik adat akkor visszaadja a login nézetet úgy, hogy a beírt adatok törlődnek.

Mindemellett ha a felhasználó nincs bejelentkezve vagy még nem regisztrált akkor a weboldalunk egy korlátozott mását érheti el/láthatja. Például aki nincsen bejelentkezve az a welcome viewbe a regisztrációt láthatja ezt Guest függvénnyel oldottuk meg plusz a navbárba (az oldal fejléce) csak a logó látható illetve a rólunk (view). A logóra kattinva ugyanúgy, mintha a főoldalra kattintanánk, minden esetben a welcome viewt adja vissza.

* Üzenet váltás functionjei:

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás

5. ábra SendMessage, üzenetküldés

Itt is lefut az Auth validation és ellenőrzi, hogy minimum 5 karakterű-e az üzenet. A küldő és a receiver idját is lekéri az adatbázisból a function és az elküldés időpontját is hozzárakja ha minden egyezik. Ahhoz, hogy az üzenet még olvasatlannak érkezzen be a receiverhez a data->statussal ellenőrzi a program, majd elmenti az adatbazisba a data->Save()vel. Ez a function egy másik, összetettebb és hosszabb queryre épül:

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

6. ábra Az elküldött üzenetekért felelős function

Ebben a functionben a program megint lefuttatja az Auth-ot, majd ellenőrzi a megírt üzenetet a két fél között. Miután leellenőrizte a két félt, hogy léteznek-e ilyen adattal játékosok, belép a query-be. A query-ben lefutnak az ellenőrzések, hogy az üzenet sikeresen el lett küldve, illetve hogy a céljátékos megkapta-e az üzenetet. Miután az üzenet küldése sikeres, a query segítségével beteszi az üzenetet, amit az adatbázisban letárolt egy statusba, miszerint az üzenetet megnyitották-e. Amint a másik játékos megnyitja az üzenetet a statust a program átváltja, és az adatbázisban már olvasottként lesz nyilvántartva, addig olvasatlanként lesz elmentve.

A játék megtervezése/elkészítése

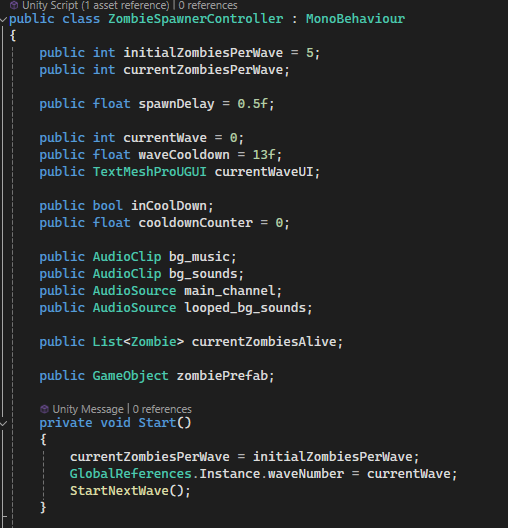
A játékunk a The Last Sunshine egy Unity motorral futatott, C# alapon írt 3D-s belső nézetű, körökön alapuló arcade stílusú túlélő zombis játék. A játékot elsősorban l Raul fejlesztette és tesztelte, mi ötleteket adtunk és a tesztelésben segédkeztünk. Az elkészült játék a Unity Asset Store-ból, Pixabay-ről és Mixamo-ról ingyenesen felhasználható asset-ek felhasználásával lett elkészítve.

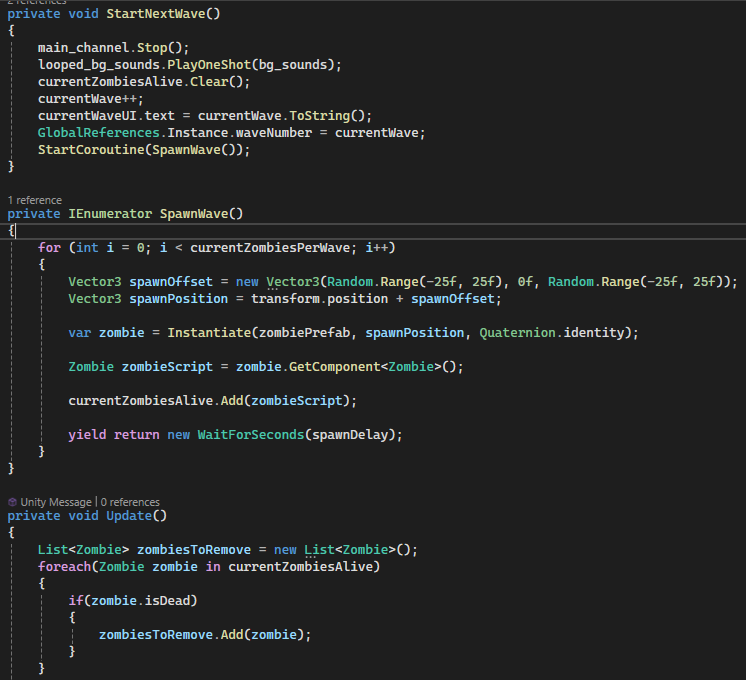
Fejlesztési modell:

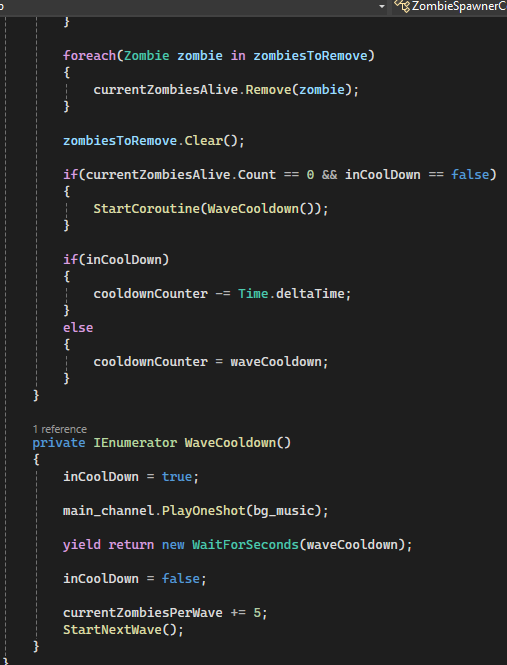
A játékot a spirál fejlesztési modell/elv alapján fejlesztettük, azaz elterveztünk egy funkciót, mechanikát, utána megkezdtük a fejlesztést és közben folyamatosan addig teszteltük, ameddig nem működött nagyjából tökéletesen és miután minden jól működött, jöhetett a következő funkciók tervezése, lefejlesztése, tesztelése.

A játékról:

A játék célja, hogy minél több hullámot élj túl úgy, hogy a fegyverekkel lelövöd a zombikat. A zombik hullámokban jönnek, ami egy végtelen ciklus alapján működik, ami leidézi a körben lévő zombik mennyiségét 0,5 másodperceként (mondjuk 12-öt, ami a 3. hullám). A még élő zombikat a játék egy listában tárolja el és a lelőtteket meg egy másikba tölti át amikor meghallnak. A kör végeztével a másik listába áttöltött halott zombikat elpusztítja a program, utána megnöveli egyel a jelenlegi hullámot és az új körben lévő zombik számát megnöveli 3-al (így lesz a 3. után a 4. hullám és az már a 12 zombi helyett most már 25-öt idéz le). A játék az első hullámtól indul, amiben 6 zombit idéz le.







A zombik a játék stílusából adódóan mindig a játékost követik és ha a közelébe érnek, megtámadják. Ütésenként 37 életerő pontot vesznek le, ami így a játékost 3 ütésből öli meg. A játékosnak és a zombiknak is 100 életerő pontjuk van. A játékos a zombikat különféle fegyverekkel tudja lelőni, amiknek különböző statisztikájuk és tűz erejük van. A fegyverekkel a bal egérgombbal lehet lőni, a jobb egérgombbal meg célózni. Az R betűvel újra lehet tölteni a fegyvert, vagy ha kifogyott a tár teljesen, ami a tárban lévő lőszerek mennyiségét egyenlővé teszi a tárba beleférő maximum lőszerek számával, amit az összes lőszerből von le a játék minden újratöltésnél. Ha kifogyna a fegyver, azt a lőszeres ládánál, az E betű lenyomásával lehet újratölteni. Ilyenkor a program megnézi, hogy milyen fegyver van a játékos kezében és ez alapján a tárban és a játékosnál lévő fegyverhez való lőszerek mennyiségét a maximumra állítja be. Az alap fegyverünk a pisztoly, amivel 21-et tudunk sebezni az ellenfélbe, ami az M1911 nevű semi-automatic marok fegyver, amivel minden egérgomb lenyomásnál lő egy töltényt. 8 töltény van a tárában és a játékosnál még maximum 10 tárnyi lőszer lehet bónuszban, azaz 80, de ez igaz az összes fegyverre, mindegyiknél a fegyverben lévő táron kívül még tíz darab tárnyi töltény lehet, mivel a pisztolyon kívül még két nagyobb automata gépfegyvert lehet a pályán felvenni. Az egyik az Uzi nevű submachinegun, nagyon gyors tüzelő sebességgel, viszont a fegyverek kiegyensúlyozottsága érdekében ez sebez a legkevesebbet azaz 17-et, de ennek van a tárában a legtöbb, azaz 40 lőszer és mellé 400 a játékosnál. A másik gépfegyver az a M4\_8, ami szintén automata fegyver, de lassabb a tüzelő sebessége, viszont ennek a legnagyobb a sebzése, ami 27, 30 tölténnyel a tárában, mellé meg még 300 lőszer a játékosnál bónuszban.

Használt algoritmusok

A weboldal elkészítésekor leginkább Laravel php-t használtunk, emellett a játékot a Unity játék fejlesztő motor segítségével hoztuk létre.

1. Tesztelés és annak menete

Weboldal tesztelése

A weboldalt a programkód megírása során dinamikusan teszteltük, azaz folyamatosan ellenőriztük, hogy az addig megírt kódokban van-e hiba. Azért ezt a tesztelést választottuk, mert így könnyebben tudtuk nézni, hogy a programkódban hol a hiba. Ez alatt azt értjük, hogy esetleg ha egy route hibás, és egyből megnézzük, hogy a route helyes értéket, view-t ad vissza vagy mégsem, akkor azonnal ki tudjuk javítani a hibát felesleges keresgélés nélkül. Ez sokkal kényelmesebb és időtakarékosabb megoldásnak bizonyult a programkód írása során.

Játék tesztelése

A játékot csakúgy, mint a weboldalt, szintén dinamikusan szerettük volna tesztelni, változatlanul időtakarékossági és kényelmi okok miatt. Azonban az elkészítés során beleütköztünk egy hátulütőbe, amire nem gondoltunk eleinte, mégpedig abba, hogy míg a weboldal esetében könnyen tudtuk nézni közben, hogy a program hibás-e, a játékban ez nem feltétlenül látszódik eleinte. Emiatt végül a játékfejlesztését spirális metódussal készítettük és teszteltük is. Ez majdnem hasonló a dinamikus teszteléshez, annyi különbséggel, hogy itt csak akkor teszteltünk, amikor már egy nagyobb ”tömbnyi” programot leprogramoztunk.

Ez alatt azt kell érteni, hogy például a karakter mozgását nem úgy teszteltük le, hogy tud-e előre menni, miután megírtuk a kódot ahhoz, hogy előre tudjon menni a karakter, hanem csak akkor teszteltük le magát az egész mozgást, miután már megírtuk az előrefelé vezető mozgás mellé a jobbra-balra-hátra mozgást is.

1. Linkek és források a játék elkészítéséhez

C# kódok (plusz a videók leírásaiban megtalálhatók a be nem linkelt asset-ek, sprite-ok és hangeffektek letöltési linkjei:

<https://www.youtube.com/playlist?list=PLtLToKUhgzwm1rZnTeWSRAyx9tl8VbGUE>

<https://youtube.com/playlist?list=PL5KbKbJ6Gf99mcmE1ptsn0oXO1_vnKDlS&si=FtUWIjKyIcl__3qz>

<https://www.youtube.com/watch?v=K8p7gXsvOng>

Zenék, hangeffektek:

<https://pixabay.com/hu/sound-effects/nature-soundstropicaljunglebirds-108380/>

<https://pixabay.com/hu/music/horror-jelenet-horror-background-atmosphere-with-creepy-clown-laughter-172681/>

<https://pixabay.com/hu/music/ijeszto-gyermekdallamok-creepy-music-box-halloween-music-horror-scary-spooky-dark-ambient-118577/>

<https://pixabay.com/hu/music/focim-dark-action-horror-trailer-112213/>

Modellek, textúrák, effektek, animációk:

<https://pixabay.com/hu/photos/r%C3%A9m%C3%A1lom-f%C3%A9lelem-borzalom-ijed%C5%91s-1699071>

<https://assetstore.unity.com/packages/3d/props/exterior/pbr-barrels-and-crates-3460>

<https://assetstore.unity.com/packages/vfx/particles/blood-gush-73426>

<https://assetstore.unity.com/packages/3d/vegetation/trees/coconut-palm-tree-pack-7888>

<https://assetstore.unity.com/packages/vfx/particles/legacy-particle-pack-73777>

<https://assetstore.unity.com/packages/3d/props/guns/low-poly-weapons-vol-1-151980>

<https://assetstore.unity.com/packages/3d/vegetation/trees/palm-tree-pack-free-214483>

<https://assetstore.unity.com/packages/2d/textures-materials/sandstone-textures-4k-179134>

<https://assetstore.unity.com/packages/audio/sound-fx/creatures/zombie-voice-audio-pack-free-196645>

1. Továbbfejlesztési lehetőségek

A projekt elkészítése során sok ötlet jutott még az eszünkbe, amit nem tudtunk kivitelezni, nagyrészt inkább a játék kapcsán, viszont a weboldalra is volt egy-két ötletünk, hogy mitől lehetne jobb. A weboldalunkból hiányzik egy admin felület, ami mást lát, mint egy regisztrált fiók. Arra gondoltunk, hogy az admin fiók láthatná mindegyik játékos profilját és lehetne benne egy külön oldal, ami a játékostámogatásért lenne felelős. Itt lehetnének játékosok által írt cetlik, amikben tudnának a játékosok panaszkodni egyéb játékbeli hibák miatt. Ez természetesen magával tudna hozni egy regisztrált fiókban is egy új lehetőséget, az imént említett cetlik megírását az adminok felé, ez lehetne az úgynevezett support funkció.

A játék kapcsán több ötletünk is felmerült, miben lehetne változtatni, mitől lehetne jobb. A jelenlegi állapotában a játék csak egy pályát tartalmaz és egy zombi típust, de a későbbiekben kerülhetnének bele más zombi fajták is, például lehetnének gyorsabb zombik amiknek a mozgási sebessége gyorsabb lenne, mint egy alap zombinak de az élete kevesebb lenne, hogy kiegyensúlyozott legyen. Lehetnének még nagyobb zombik amiknek meg az életük lenne több, cserébe lassabbak lennének. Ezeken kívül lehetne benne még több pálya is, más atmoszférával, stílussal. A játék élményen szerintünk az is növelne, ha beletennénk egy multiplayer lehetőséget is, mivel a jelenlegi játékban csak egyedül lehet játszani. A multiplayer lehetőséggel élvezetesebb lenne a játék, mert barátokkal tudnának játszani a játékosok, vagy akár más ismeretlen emberekkel is. A játékba belehetne tenni ezeken az említett dolgokon kívül még egy loadout lehetőséget, amit a játékosok minden megkezdett játék előtt be tudnának állítani. Ezek a loadoutok a játékosok által beállított inventory-k lennének. Tartalmaznák, hogy milyen fegyverrel kezdenének a játékosok és milyen perkeket használnának a játék során. Ezek a perkek szintén egy még a játékban nem létező, de beiktatható funkció lenne. Lényegében ezek segítenék a túlélést több módon is, például lenne egy perk, amitől gyorsabban töltene újra a játékos, egy másiktól gyorsabban mozogna és még megannyi lehetőség van ebben az ötletben, amik még szórakoztatóbbá és élvezetesebbé tennék a játékot.

1. Összegzés

A vizsgaremekünk célja az volt, hogy készítsünk egy szokatlanabb, nehezebb projektet. Maga a nehézség a játékkal volt, ugyanis ezt nem tanultuk és nem volt benne sok ismeretünk. Pont az ismeretlenség miatt az idővel is harcban álltunk, sokszor elcsúsztunk a határidővel, annyira súlyosan, hogy a legutolsó pillanati határidő végére lettünk csak kész mindennel.

Ez egy hatalmas projekt volt, amihez eleinte kételkedve fogtunk hozzá, hogy biztosan akarjuk-e ezt a játékot és ha akarjuk is, sikerülni fog-e a kivitelezés. Ezutóbbiban kételkedtünk a leginkább, főleg a határidő végéhez közeledve, de a legvégén sikerült legyőzzük az időt és magunkat is és le tudtunk adni egy kész projektet.

A weboldallal nagyon elégedettek voltunk, a játék viszont két ízvilágot éreztetett a szánkban, az öröm ízét és a csalódottságét. Úgy gondoltuk, hogy ha nem csúszunk el az idővel, akkor talán még jobbra sikerült volna a játék, de közben boldogok is voltunk az eredménnyel, mert még ha kezdőcipőben is van a játékunk, működik.

1. Ábrajegyzék

1. ábra UserController és functionök 5

2. ábra NewProfile function és validációk 6

3. ábra ProfileMod function és validációk 7

4. ábra Login function és validációk 8

5. ábra SendMessage, üzenetküldés 9

6. ábra Az elküldött üzenetekért felelős function 10