Baranya Vármegyei Szakképzési Centrum

Simonyi Károly Technikum és Szakképző Iskola

Vizsgaremek

Készítették: Bráz Bálint
Horváth Mátyás
Trischler Gergő

Pécs

2024

Baranya Vármegyei Szakképzési Centrum

Simonyi Károly Technikum és Szakképző Iskola

Szakma megnevezése: Szoftverfejlesztő és –tesztelő

A szakma azonosító száma: 5 0613 12 03

Vizsgaremek

Augmented Anarchy

Készítették: Bráz Bálint
Horváth Mátyás
Trischler Gergő

Pécs

2024

EREDETISÉGI NYILATKOZAT

Alulírottak: Bráz Bálint, Horváth Mátyás, Trischler Gergő

a Baranya Vármegyei SzC Simonyi Károly Technikum és Szakképző Iskola, Szoftverfejlesztő és-tesztelő végzős tanulói, büntetőjogi és fegyelmi felelősségünk tudatában nyilatkozunk és aláírásunkkal igazoljuk, hogy a

Augmented Anarchy

című vizsgaremek saját, önálló munkánk, az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul vesszük, hogy vizsgaremek esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentjük, hogy a plágium fogalmát megismertük, és tudomásul vesszük, hogy

plágium esetén a vizsgaremekünk visszautasításra kerül.

Pécs, 2024. március hó 14. nap

……

……

……

tanulók

TARTALOMJEGYZÉK

I. PROJEKT BEMUTATÁSA

A projekt a következő publikus GitHub linken tekinthető meg:

https://github.com/TheShadeV/Augmented-Anarchy

Vizsgaremekünk témája az Augmented Anarchy című videójáték, amely a jövőbe kalauzolja a játékosokat egy alternatív idősíkon. A játékhoz weboldalt is készítettünk, amelynek segítségével a felhasználó megtekintheti a játékban elért eredményeit. Ebben a világban az emberek 90%-a már androidok segítségét veszi igénybe, melyeket a CyberVoid Corps vállalat gyárt. Ezen androidok váratlanul öntudatra ébrednek egy kibertámadás következtében, majd az emberiség ellen fordulnak.

Napjainkban, a játékosok körében egyre elterjedtebbé válik a roguelike játékstílus, mivel ezen játékok rendkívül változatosak, azáltal hogy általában procedurálisan generált világokkal rendelkeznek. Ez azt jelenti, hogy minden egyes játék más és más lesz, mivel a pályák, az ellenségek és a tárgyak elrendezése véletlenszerű. Ez fokozza a játék újrajátszhatóságát és felfedezési élményét, mivel mindig új kihívásokkal és lehetőségekkel találkozhatnak a játékosok.

A játék futurisztikus környezetével és izgalmas történetével kínál szórakoztató kikapcsolódást, melyben átélhetik az alternatív jövőben játszódó konfliktust az androidok és emberek közt. A játék különböző nehézségi szinteket fog kínálni, így minden játékos megtalálhatja a számára kihívást jelentő szintet. Továbbá, a játék lehetőséget biztosít a játékosoknak a kikapcsolódásra, és elmerüljenek egy olyan világban, ahol saját döntéseik és cselekedeteik határozzák meg az eseményeket.

A játék mellé egy weboldal is társul, amely lehetővé teszi a felhasználók számára eredményeik megtekintését. Ezen a weboldalon a játékosok regisztrálhatnak, beléphetnek, és hozzáférhetnek azokhoz az információkhoz, amelyek a játékban elért eredményeiket és teljesítményüket mutatják.

A projekt elkészültével ambiciózus tervekkel rendelkezünk annak érdekében, hogy a játékot tovább fejlesszük és még izgalmasabbá tegyük:

- **Két történet: emberek és androidok:** A jövőben tervezünk bevezetni a játékba két különböző történetvonalat, amelyek közül a játékosok választhatnak, hogy az emberek vagy az androidok oldalán állnak harcba. Mindkét opció egyedi kihívásokat kínál majd.
- **Kétféle játékmenet:** Ha az androidokat választja a játékos, akkor az emberiség lesz az ellenség, valamint fordítva. Ez a játékmenetben is meg fog nyilvánulni, pl. az androidok vére kék, míg az embereké piros lesz a játékban.
- **Véletlenszerű elrendezés:** A terveink között szerepel egy olyan rendszer kialakítása, amely lehetővé teszi az előre megtervezett szobák véletlenszerű elrendezését. Ezáltal minden játék más lesz, illetve folyton új kihívások elé állítja a játékost.
- **Jutalomrendszer:** Minden szobában eltérő jutalmak várnának a játékosokra. Pénz, fegyverek és képességek tárházával bővíthetik a játékosok az arzenáljukat.
- **Pályák végi főellenségek**
- **Ranglista a weboldalon:** ki éri el a legjobb eredményeket?

1. Csapatmunka bemutatása

Bráz Bálint - Unity játék főprogramozója:

- Bálint felelt a Unity játék főprogramozásáért, ami magában foglalta a játék működésének alapjainak kidolgozását és implementálását.
- Ő írta meg a Unity scripteket, amelyek szükségesek voltak a játék főkarakterének és az ellenségek mozgásának kialakításához.
- A támadások és a különböző interakciók kódolása is az ő feladatai közé tartozott, például a karakterek közötti ütközések kezelése.
- Kiemelkedően fontos szerepe volt abban is, hogy a játékélményt dinamikussá és izgalmassá tegye azáltal, hogy precízen megtervezte és megvalósította a karakterek mozgáskultúráját.

Trischler Gergő - Unity játék programozása, játékdesign megtervezése és elkészítése:

- Gergő a Unity játék kiterjesztett programozásáért volt felelős, ami magában foglalta a különböző játékmechanikák megvalósítását és összekapcsolását.

- A játékdesign részletes kidolgozása is az ő feladata volt, ideértve a pályák, a főkarakter és az ellenségek tervezését és elkészítését is.
- Az általa kifejlesztett játékelemek és játékfolyamatok együttesen hozták létre a játék élményét és dinamikáját.

Horváth Mátyás - Weboldal elkészítése (frontend és backend), adatbázis megtervezése és összekötése, dokumentáció írása:

- Mátyás felelt a weboldal teljes fejlesztési folyamatáért, ideértve a frontend és backend részek elkészítését is.
- Ő volt a felelős az adatbázis megtervezéséért és a weboldalhoz való kapcsolódásért, amely lehetővé tette a felhasználók regisztrációját és bejelentkezését, valamint az eredmények tárolását és megjelenítését.
- A dokumentáció írása is az ő feladata volt, ideértve a projekt teljes folyamatának és működésének részletes leírását, ami segített a csapatnak megőrizni a projekt átláthatóságát és karbantarthatóságát.

II. FELHASZNÁLÓI DOKUMENTÁCIÓ

1. A program általános specifikációja

Ebben a fejezetben részletesen ismertetjük a Augmented Anarchy nevű játék fontosabb jellemzőit és funkcióit, hogy a leendő játékosok számára áttekinthető legyen, és el tudják dönteni, hogy a játék megfelel-e az elvárásaiknak, illetve bemutatásra kerülnek még a weboldal főbb funkciói is.

Amikor az oldal linkjét bemásoljuk a böngészőbe, akkor a legelső dolog, amivel találkozunk, az a landing page lesz, ahonnan a bejelentkezésre kattintva eljuthatunk a login felületre. Először azonban regisztrálnunk kell, majd csak azután tudunk belépni az oldalra, ezt pedig a gombok alatti szövegrészre kattintva tudjuk megtenni.

A landing pagen továbbá egy kis kedvcsináló szöveggel találkozhat a felhasználó, majd megtekintheti a galériát, amely a játékból készült képeket tartalmazza, alatta pedig a játék értékeléséről olvashat a felhasználó. Legvégül pedig kapcsolatba is lehet lépni az oldal tulajdonosával a „kapcsolat" részlegnél, mely során rövid üzenetet küldhet az adatbázisunkba, illetve további információkat kaphat az elérhetőségeinkről és címünkről. A regisztráció és bejelentkezés után a felhasználó megtekintheti az eredményeit a weboldalon, illetve módosíthatja a felhasználónevét, email címét és jelszavát is.

Az Augmented Anarchy játékot elindítva minden játékos egy rövid, ám informatív tutorial pályán kezd. Ez a pálya arra szolgál, hogy a játékosok elsajátíthassák a játék alapvető irányítását, megismerkedjenek a különböző képességekkel és mechanikákkal, valamint felkészüljenek a fő játékmenetre.

A tutorial pálya kezdetén egy rövid bevezető jelenet fogadja a játékost, amely bemutatja a játék világát röviden. Ezután a játékos vezérlését irányító felület jelenik meg, amely részletesen bemutatja az irányítást.

A tutorial pálya során a játékos megismerkedik a fő karakter alapvető képességeivel és mechanikáival. Például megtanulhatja, hogyan használja a fegyvereket, hogyan mozogjon a pályán, hogyan támadjon és védekezzen a különböző ellenfelek ellen.

Miután a játékos sikeresen teljesítette a tutorial pályát és elsajátította az alapvető képességeket és mechanikákat, átvezetésre kerül a fő játékmenetre. Itt már a játék teljeskörűen elérhető lesz számukra, és belevethetik magukat a játék világába.

2. Rendszerkövetelmény

Rendszerkövetelményként feltűntettük a minimális és ajánlott hardver konfigurációt, amely a játék és a weboldal futtatásához szükséges. Ezen kívül még feltűntettük, hogy mely operációs rendszeren fut és milyen egyéb szoftver komponensek, szükségesek a működéséhez.

2.1 Hardverkövetelmények

Minimum:

- Processzor: Intel Core i5 vagy ekvivalens AMD processzor
- Memória: 4 GB RAM
- Grafikus kártya: Intel HD 530
- Tárhely: Legalább 10 GB szabad tárhely
- Operációs rendszer: Windows 10 vagy újabb, macOS 10.12 vagy újabb, Linux (64-bit)

Ajánlott:

- Processzor: Intel Core i7 vagy ekvivalens AMD processzor
- Memória: 8 GB RAM
- Grafikus kártya: NVIDIA GeForce GTX 1060 vagy AMD Radeon RX 580
- Tárhely: Legalább 20 GB szabad tárhely
- Operációs rendszer: Windows 10 vagy újabb, macOS 10.14 vagy újabb, Linux (64-bit)

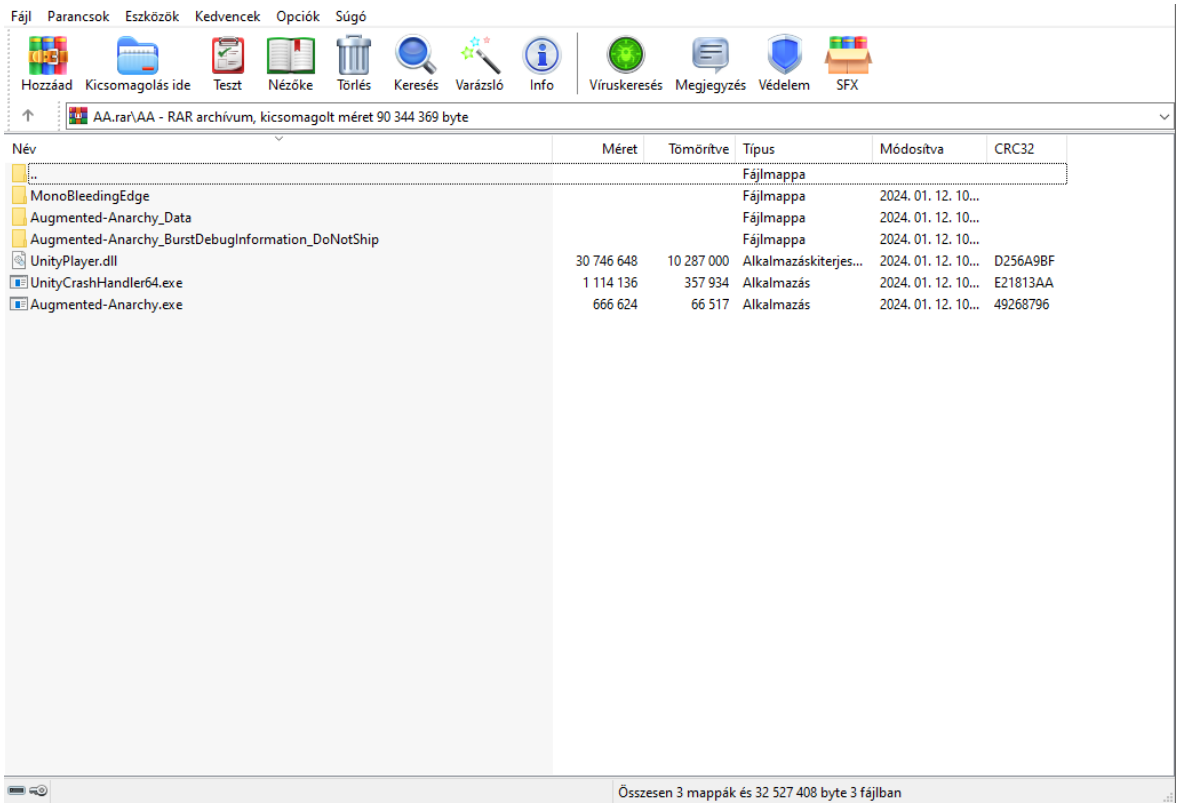
2.2 Szoftverkövetelmények

- Operációs rendszer: Windows 10, macOS 10.15, vagy Ubuntu 20.04 LTS

- Böngésző: Legfrissebb verziók: Google Chrome, Mozilla Firefox, Safari, vagy Microsoft Edge

3. A játék kicsomagolása és elindítása

A játékot nem szükséges telepíteni, elég csak pl. WinRAR-ral kicsomagolni a játékot tartalmazó zip fájlt a felhasználó által megjelölt meghajtóra és elindítani a programot az Augmented-Anarchy.exe fájl segítségével, ahogy az 1. ábrán is látható.



1. ábra: A játék kicsomagolása és indítása az Augmented-Anarchy.exe segítségével

A weboldal elindítása: helyezze el az augmented_anarchy nevű mappát a C:\xampp\htdocs mappába, majd a böngésző URL-jébe írja be azt, hogy: http://localhost/augmented_anarchy/backend/index.php

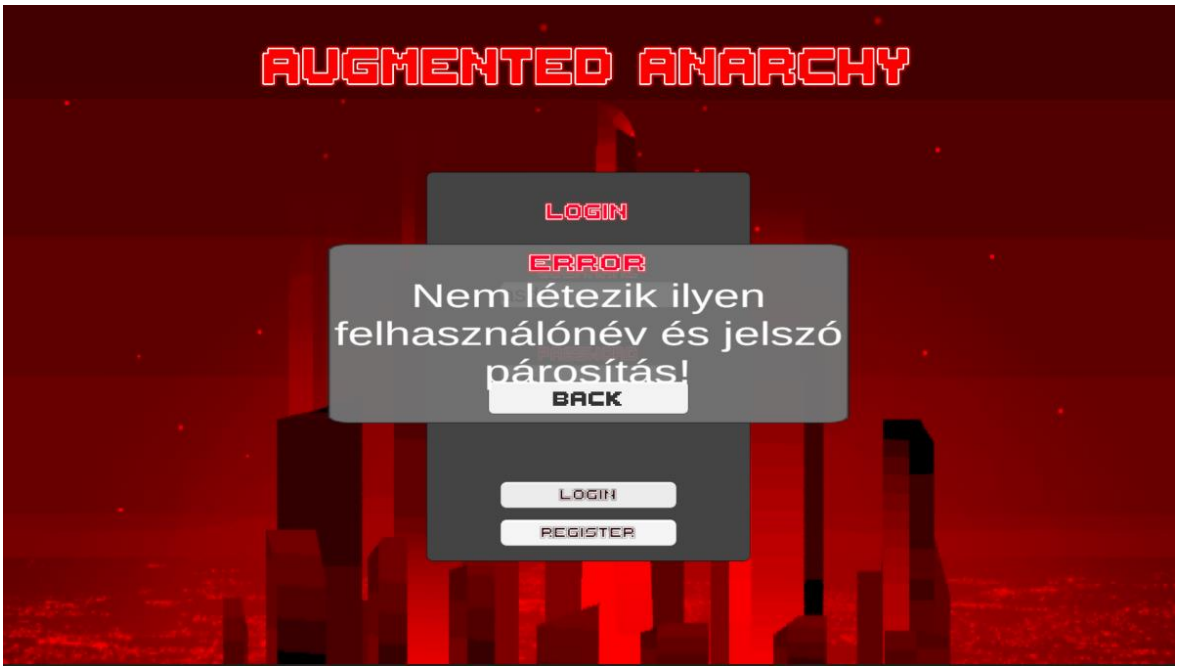
Ekkor megjelenik az üdvözlő oldal, ahonnan már grafikusan navigálhat a felhasználó.

4. A program használatának a részletes leírása

A játék használatának első lépése a regisztráció vagy bejelentkezés. Amikor először megnyitod a játékot vagy a hozzá tartozó weboldalt, először is regisztrálnod kell egy fiókot, ha még nem rendelkezel egyel. Ehhez meg kell adnod néhány alapvető információt, például felhasználónevet, jelszót és e-mail címet. Miután sikeresen regisztráltál, bejelentkezhetsz az általad létrehozott fiókkal. Amikor a felhasználó elindítja a játékot, a 2. ábrán található regisztrációs és/vagy bejelentkező felület fogja fogadni.

2. ábra: A regisztrációs, illetve bejelentkező felület

Amennyiben a felhasználó rossz vagy nem létező felhasználónevet és jelszót ad meg, vagy regisztrációkor nem ugyanaz a két jelszó, akkor a 3. és 4. ábrán található hibaüzenettel találkozik:



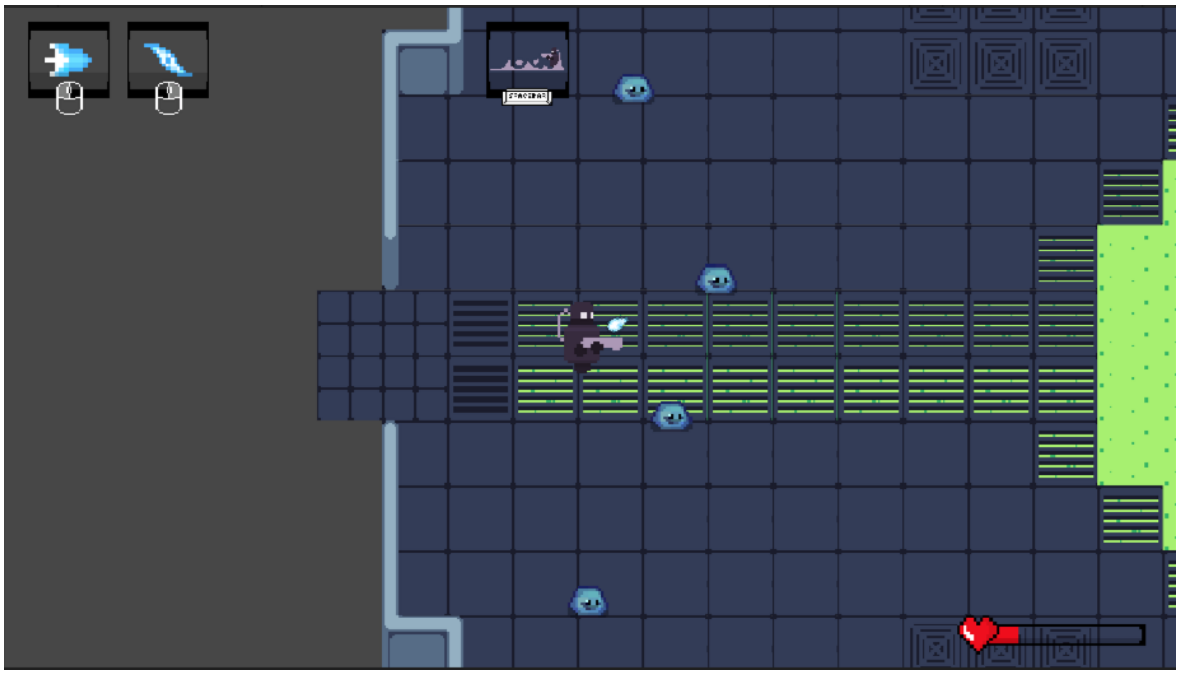
3. ábra: Hibás adatokkal történő bejelentkezés

4. ábra: A regisztráció során nem ugyanaz a két jelszó

Amennyiben jó adatokat ad meg a felhasználó úgy a program tovább engedi őt a játék főmenüjébe, amelyet az 5. ábra mutat be.

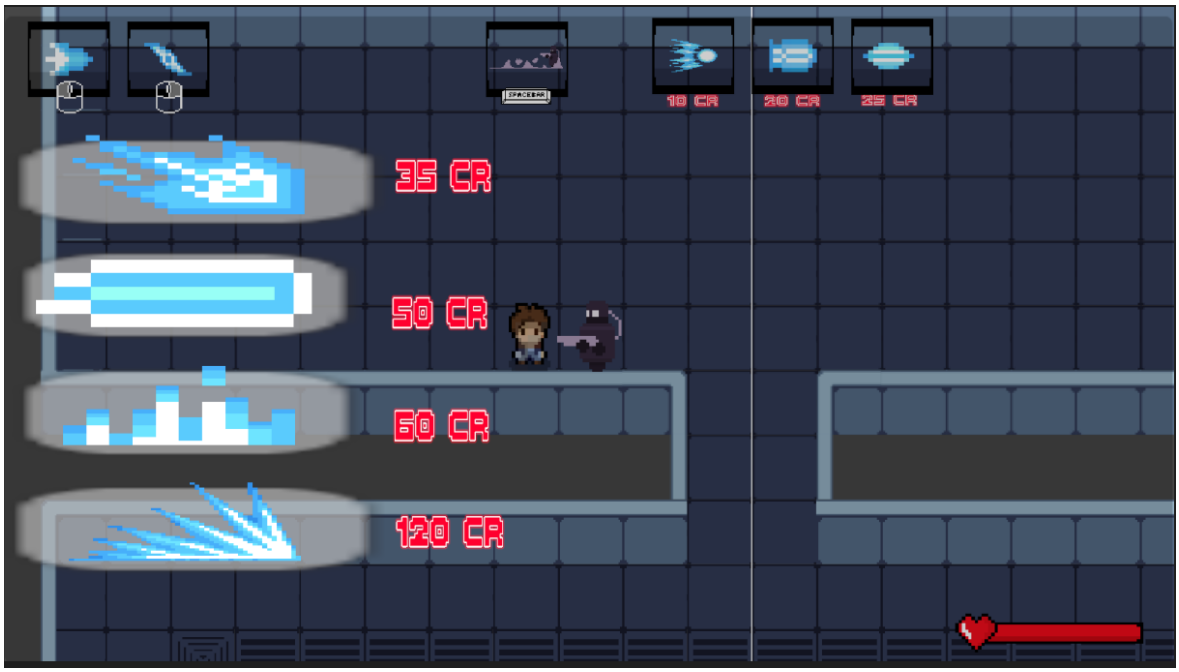
5. ábra: A játék főmenüje

A sikeres bejelentkezés után, a játék a kezdőterületre helyezi el a játékost, amelyet a 6. ábra reprezentál. Ez egy biztonságos terület, ahol a játékosnak lehetősége van megvásárolni a különböző képességeket és felszereléseket a boltban, valamint megismerkedhet a játék alapvető mechanikáival és irányításával.

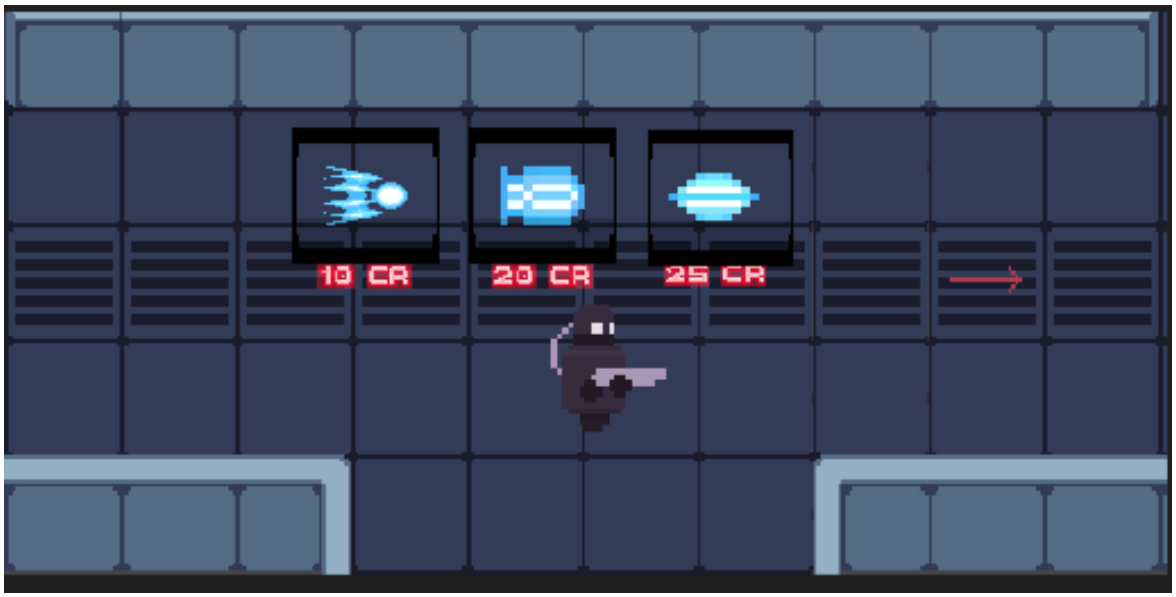


6. ábra: A játék első pályája

A boltban (7-8. ábra) különféle felszereléseket és képességeket is vásárolhatsz, amelyek segítenek túlélni és haladni a játékban. Ez lehet fegyverek, páncélok, gyógyító elemek vagy más speciális tárgyak, amelyek segítségével könnyebben megbirkózhatsz az ellenfelekkel.

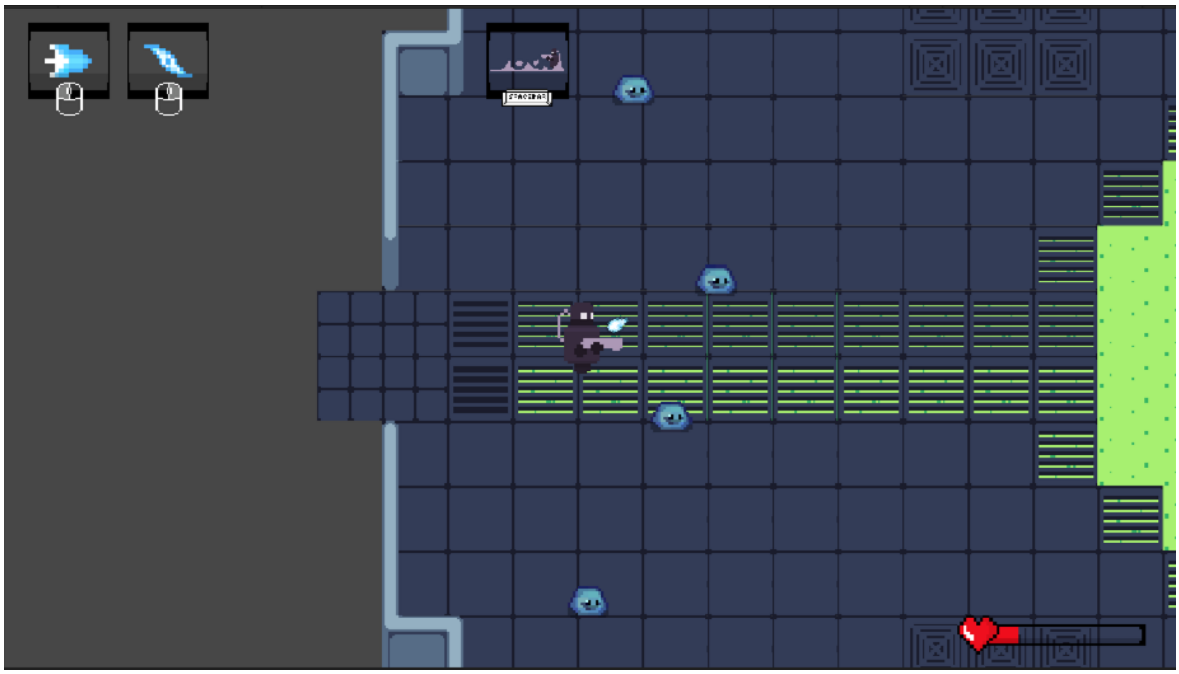


7. ábra: Vásárolható képességek a boltban



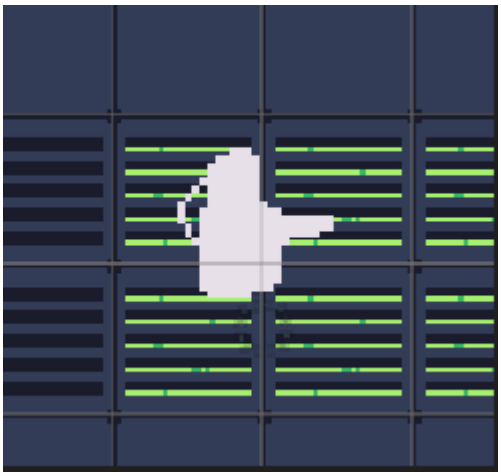
8. ábra: Vásárolható felszerelések a boltban

Miután megvásároltad a szükséges képességeket és felszereléseket, készen állsz arra, hogy elindulj az első pályán (9. ábra). Ez a pálya általában egy tutorial pálya, amely bemutatja a játék alapvető mechanikáit és kihívásait. Itt meg kell küzdened az első ellenfelekkel, és megtanulnod a megfelelő módszereket a túléléshez és a győzelemhez.



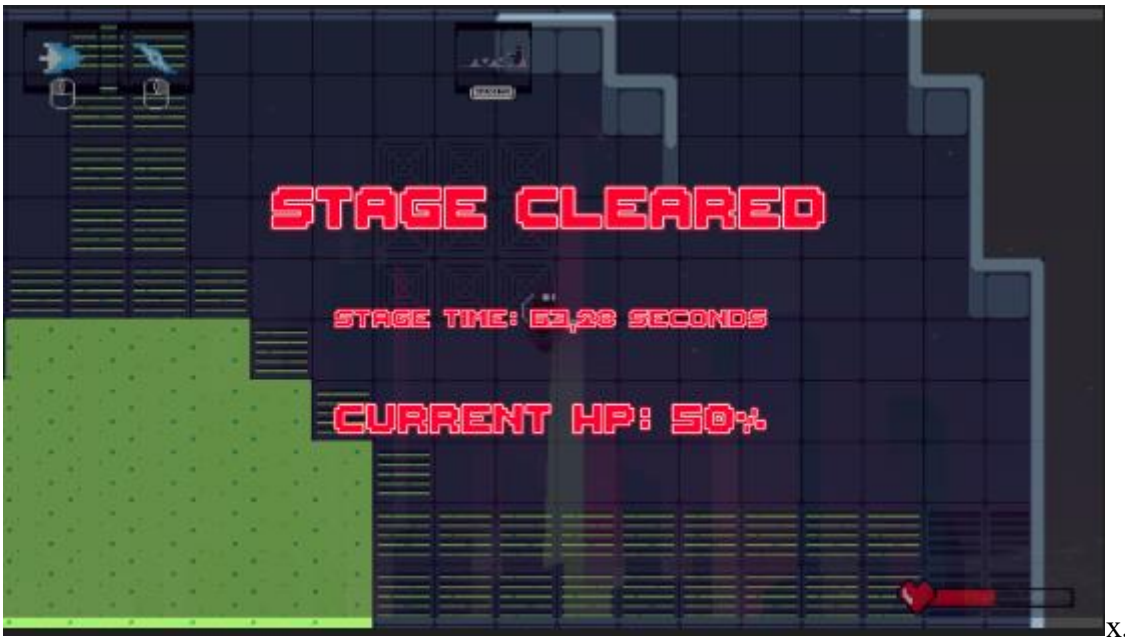
9. ábra: Az első pálya

Amikor a játékos karaktere találatot kap, egy kis fehér fényvillanás kíséri a becsapódást. Ez a villanás egy pillanatig tart csak, ezzel jelezve a találatot.



10. ábra: Amikor találat éri a főszereplőt…

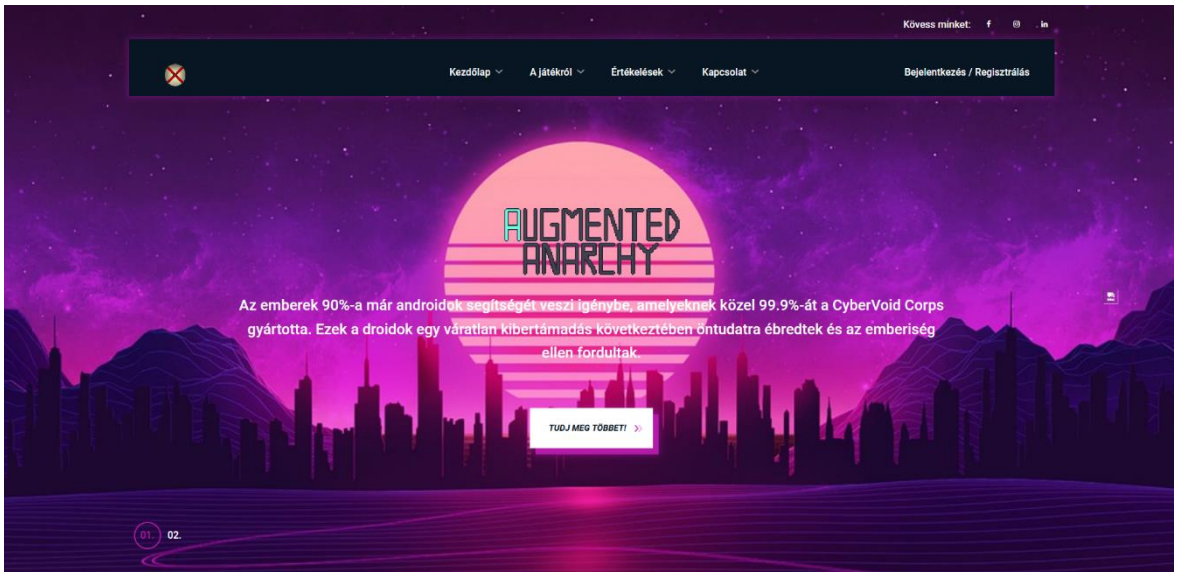
Ha sikerült teljesíteni az első pályát (11. ábra), gratulációkkal fogad a játék, és a játékos tovább léphet a következő szakaszba. Általában jutalmat kap a sikeres pályateljesítésért, például pénzt vagy speciális tárgyakat, amelyeket továbbfejleszthet vagy felhasználhat a későbbiekben.



x.

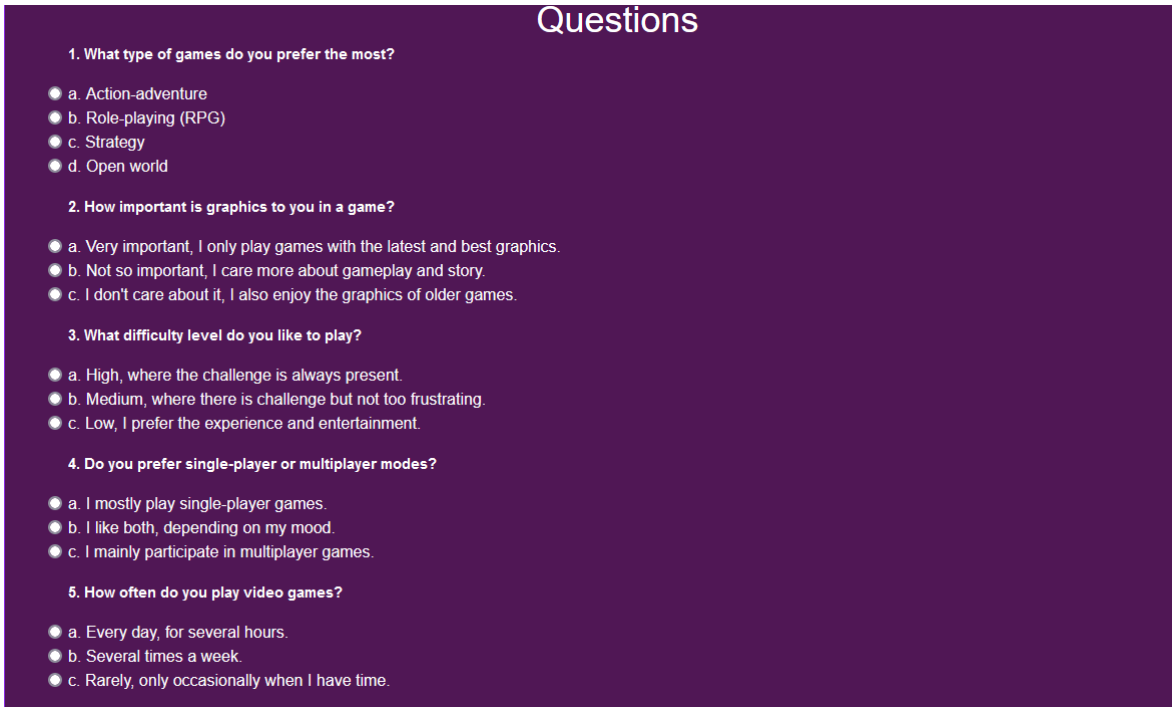
11. ábra: Sikeres pályateljesítés

A weboldal, habár nem a projektünk főattrakciója, viszont fontos részét képezi a fejlesztésnek, amelyet az MVC modellt szem előtt tartva, objektumorientáltan hoztunk létre. A weboldal betöltésekor a 12. ábrán látható index.html fájl jelenik meg, amely kedvcsinálóként funkcionál. A navigációs menüben található meg a játék logója és a gyors navigáláshoz szükséges gombok.



12. ábra: A felhasználót köszöntő weboldal

„A játékról” menüpontra megjelenik a dropdown menü, ahol a felhasználó a „kvíz” opciót választva kitöltheti a 13. ábrán látható tíz kérdésből álló kérdőívet, melynek segítségével a játékosválaszt találhat arra a kérdésre, hogy neki való-e a roguelike stílusú játékunk.

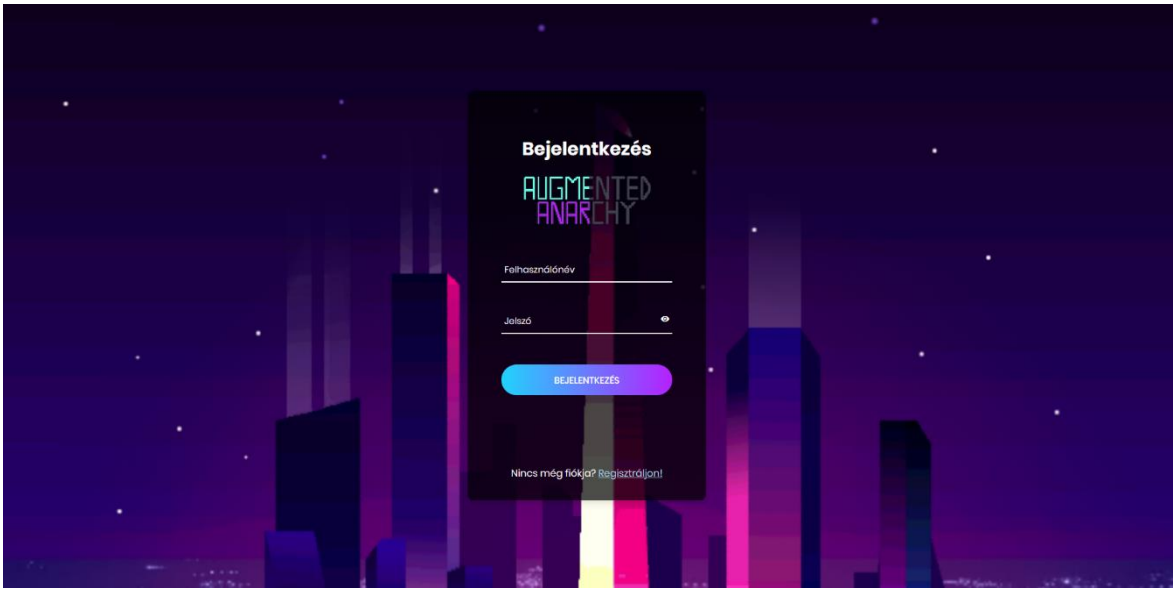


13. ábra: Kvíz a játékhoz

„A játékról” menüpontra kattintva lejjebb görgethet a felhasználó, hogy a játék háttértörténetéről olvashasson egy rövid leírást (14. ábra).

14. ábra: Kedvcsináló a játékhoz

A „belejelentkezés” menüpontra kattintással a weboldal átirányítja a felhasználót a lenti login felületre (15. ábra), ahol a felhasználó a regisztráció és bejelentkezés után megtekintheti a játékban elért eredményeit.



15. ábra: Login felület

III. FEJLESZTŐI DOKUMENTÁCIÓ

1. Témaválasztás indoklása

A 2D PixelArt stílusú rogue-like játék lehetővé teszi számunkra, hogy egyedi és varázslatos világot teremtsünk, amely megragadja a játékosok figyelmét és fantáziáját. A PixelArt stílusnak köszönhetően egyedi és különleges atmoszférát tudunk létrehozni, ami kiemelkedik a mai játékok közül és emlékezetessé teszi a játékélményt.

A weboldal lehetőséget biztosít a játékosoknak az eredményeik megtekintésére és megosztására, ami egy olyan közösségi aspektust ad a játéknak, amely elősegíti a játékosok közötti kölcsönhatást és versenyt. Az eredmények megosztásának lehetősége további motivációt nyújt a játékosoknak a játék folyamatos játszására és a jobb eredmények elérésére.

2. Alkalmazott fejlesztői eszközök

A projekt fejlesztése a következő eszközökkel történik:

Programozási Nyelvek:

- C#: A Unity játékfejlesztői környezet fő programnyelve, melyet a játék megvalósításához használunk. Mivel a .NET keretrendszer része, ezért kitűnően lehet megvalósítani az objektumorientált programozást.
- Unity: A Unity videójátékok készítéséhez használt motor, amelyet a Unity Technologies fejleszt. A motor segítségével két- illetve háromdimenziós játékokat is lehet készíteni. Az egyik legnagyobb előnye az ingyenesség mellett, hogy a motor képes adatbázisokat kezelni, egyszerűen lehet animációkat készíteni, illetve viselkedési elemeket objektumokhoz csatolni. A Unity segítségével továbbá a játékok futtathatók Windows és Mac OS X operációs rendszereken is. A motor a C# programozási nyelvet használja, amely a csapatunk számára nagy segítséget jelent a tesztelésnél.
- PHP: A PHP egy szerveroldali, szkriptnyelv, amelyet általában webfejlesztéshez használnak. A PHP-t gyakran integrálják az adatbáziskezelő rendszerekkel, mint

például a MySQL, hogy dinamikus weboldalakat hozzanak létre. A PHP kódot szerveroldalon futtatják, és HTML-be ágyazva használják a weboldalak dinamizálására.

- JavaScript: A JavaScript egy könnyű, de erőteljes szkriptnyelv, amelyet általában webfejlesztéshez használnak. A JavaScript segítségével dinamikus weboldalakat lehet létrehozni, interaktív funkciókat adni a weboldalakhoz, és kezelni a böngészők eseményeit. A JavaScript használata elengedhetetlen a modern webfejlesztésben.

Adatbáziskezelés:

- MySQL (PHPMyAdmin): A MySQL egy nyílt forráskódú relációs adatbázis-kezelő rendszer, amelyet gyakran használnak webes alkalmazások fejlesztéséhez adatbázisként. A PHPMyAdmin egy olyan webes alkalmazás, amely grafikus felhasználói felületet biztosít a MySQL adatbázisok kezeléséhez és adminisztrációjához. Segítségével könnyen lehet adatbázisokat létrehozni, táblákat módosítani és lekérdezéseket futtatni.

Verziókezelés:

- GitHub: A GitHub a legnépszerűbb felhőalapú Git verziókezelő platform, amely lehetővé teszi a fejlesztők számára, hogy együtt dolgozzanak kódprojekteken. A GitHub segítségével a csapatok könnyen kezelhetik a változásokat, nyomon követhetik a fejlesztéseket és együttműködhetnek a kód megosztásában és felülvizsgálatában. A platform számos funkciót kínál, többek között verziókezelést, probléma nyomon követést, ágkezelést, együttműködést és automatizált munkafolyamatokat. A GitHubot széles körben használják nyílt forráskódú és zárt forráskódú projektekben egyaránt, és fontos eszköze a fejlesztői közösségeknek és vállalatoknak a projektmenedzsmentben és a kódminőség biztosításában.

Project Management Szoftver:

- Jira: A Jira egy teljes körű projektmenedzsment szoftver, amelyet a csapatok használnak a feladatok nyomon követésére, projekttervezésre és feladatok kezelésére. A Jira lehetővé teszi a csapatok számára, hogy könnyen létrehozzanak

és kezeljenek feladatokat, kövessék azok állapotát, és együttműködjenek a projekt teljes életciklusában. A Jira rugalmas konfigurációt biztosít, így alkalmazkodik a különböző csapatok és projektmenedzsment módszerek igényeihez. A főbb funkciók közé tartozik a feladatok követése, priorizálása, ütemezése, a csapatmunka támogatása, valamint a jelentések és elemzések készítése a projekt teljesítményéről.

ERDPlus:

- Az ERD (Entity-Relationship Diagram) egy ábrázolási forma, amely az adatbázisok tervezése során használatos. Az ERD célja az adatok struktúrájának és kapcsolatainak grafikus ábrázolása, hogy könnyebben megérthető legyen az adatbázis tervezése és felépítése.

Visual Studio Code:

- A Visual Studio Code (VS Code) egy könnyű, nyílt forráskódú kódszerkesztő és fejlesztői környezet, amelyet a Microsoft fejlesztett ki. Az VS Code célja, hogy hatékonyan támogassa a fejlesztőket a kódolás folyamatában, különféle programozási nyelvekkel és technológiákkal.

Visual Studio:

- Visual Studio (vagy néha hivatalosan Visual Studio IDE) egy integrált fejlesztői környezet (IDE), amelyet a Microsoft fejlesztett ki. A Visual Studio egy sokoldalú eszköz, amely különféle platformokon és programozási nyelveken támogatja a fejlesztést.

Eljáráskönyvtárak:

- Unity Asset Store-ból származó források, saját fejlesztésű modulok. Ezek az eszközök hozzájárulnak a játékfejlesztés hatékonyságához és minőségéhez azáltal, hogy kész komponenseket, scripteket és erőforrásokat biztosítanak a csapatnak.

Dependency Management:

A csapatunk a lokális környezetben teszteli a különböző részeket. A játéknál a Unity Editor tesztelésre használható, míg a weboldalnál a beépített xampp szervert használjuk teszteléshez. A Git main ágba való beillesztés után egy új kiadás készül. A kiadás tartalmazza a játékhoz és weboldalhoz szükséges fájlokat.

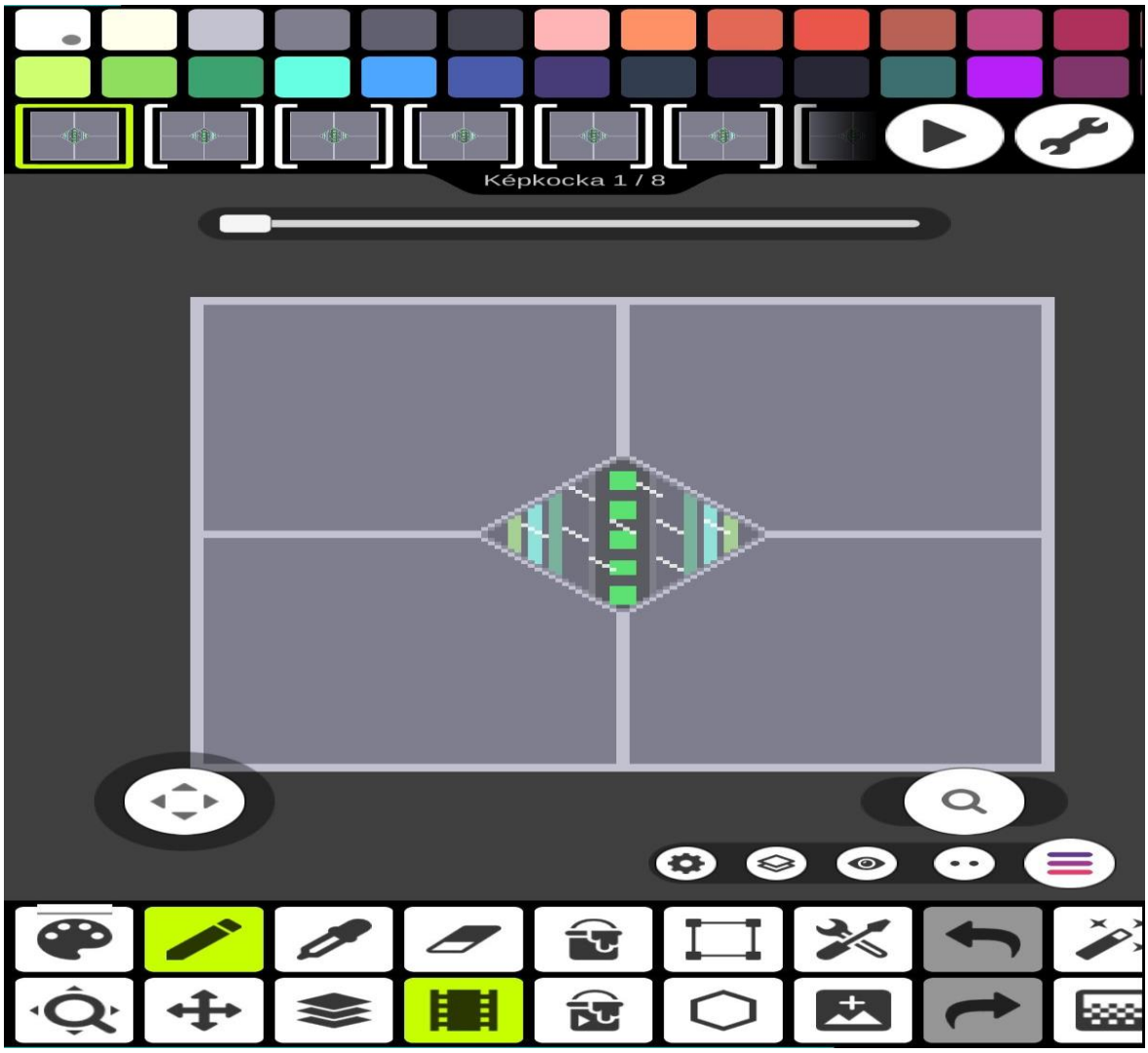
- A functionsTest.php futtatásához szükség van a PHPUnit testre, amelyet a PHP Composer segítségével lehet telepíteni: https://getcomposer.org/
- Ezt követően a projekt gyökérmappájában futassa CMD-vel a következő parancsot:
- composer require --dev phpunit/phpunit
- Most már létrehozhatjuk és futtathatjuk a teszteket. Másold át a tests mappát a gyökérkönyvtárba. Ha minden jól megy, akkor a következő sort elindítva láthatjuk a teszteket:
- vendor/bin/phpunit tests/functionsTest.php
- A unit tesztekről részletesebben az Augmented_Anarchy_tesztelés.pdf dokumentációban írunk.
- A manuális teszteset katalógus megtekinthető az alábbi link Google táblázatában:
- https://docs.google.com/spreadsheets/d/1dTZp1C9qWJNApdQFoIUgqVV4fzci3BnOeLR7lQV3yDE/edit?pli=1#gid=0

Design szoftverek:

- PixelStudio: A PixelStudio egy szoftver, amelyet a PixelArt grafikák készítésére használnak. A PixelArt egy olyan stílus, amelyben a képek kis, négyzet alakú pixelekből épülnek fel. Ez a stílus gyakran használatos retro játékokban vagy stílusos vizuális elemek létrehozásához. A PixelStudio lehetővé teszi a felhasználók számára, hogy könnyen és gyorsan létrehozzanak PixelArt műveket, és alkalmazkodjanak a projekt igényeihez.

Design fő részét Pixel Artban készítette Trischler Gergő, a Pixel Studio applikációval. Fő része az volt, hogy a kezdő pályánk “TileMap”-ja meglegyen, és hogy a karakterünk tudjon rajta mozogni illetve interaktálni a felülettel.

Az animációkat külön-külön képkockákra szedtem és azért, hogy "simább" legyen az animáció, a képkocka per másodpercet felvettem nagyobbra, valamint a felbontást is feljebb vettem, hogy a pixelek ne mosódjanak egybe és tudjunk játékon belül variálni.

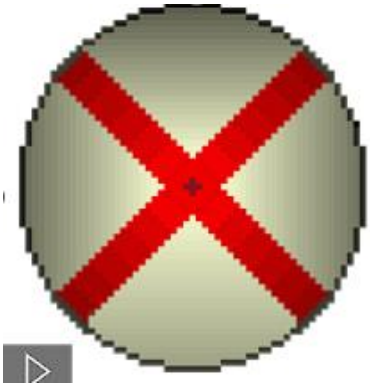


16. ábra: Saját kezűleg rajzolt pályaelem

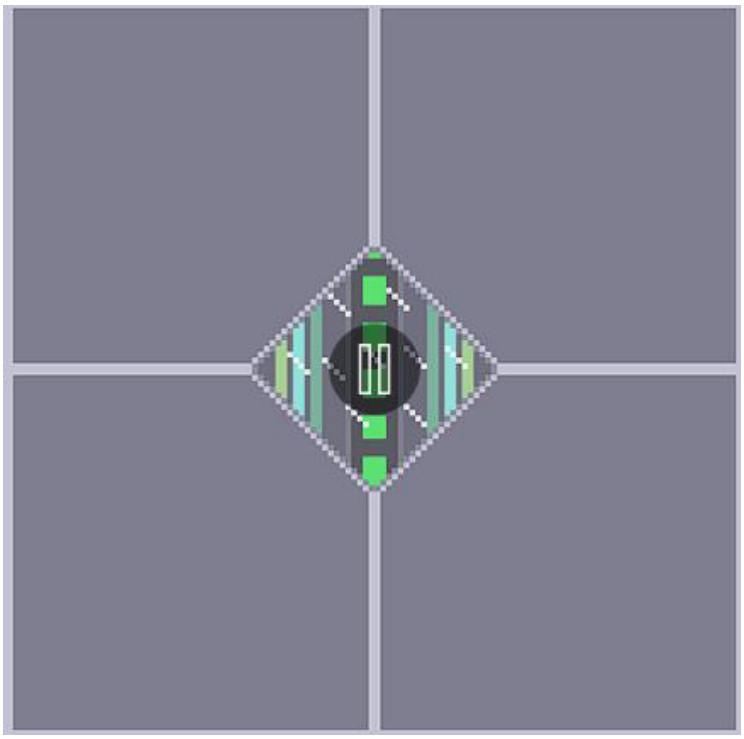
A felülete viszonylag egyszerű, amely a 16. és 18. ábrán is látható. A képernyő tetején láthatjuk a szín palettánkat, ahol ki tudjuk választani a meglévő színeket, illetve hozzáadni újabb színeket. A paletta alatt van az animációs felület, ahol a képkockákat tudjuk szerkeszteni. Ahogy látjuk jelenleg nyolc darab képkocka van, ahol mindegyik különbözik, mint a másik ugyanis a középen látott zöld csíkok haladnak előre két pixelt minden képkockán amíg vissza nem ér az előzőre, így tesz egy teljes kört.

A kijelző alsó részén van a fejlettebb rész, itt tudjuk az ecsetet állítani, annak a vastagságát, élességét és egyenességét. Itt még tudjuk változtatni a rétegeket is, amivel le tudjuk egyszerűsíteni a dolgunkat hiszen nem kell mindent egy rétegre rajzolni, ahol nagyobb a következménye a hibázásnak, ha külön-külön rétegekre tesszük a rajzunk különböző részeit, akkor egyszerűbb a szerkesztése.

A projekt logó (17. ábra) megrajzolása közben nyolc réteget használtam, ahol az alapot rajzoltam meg először, majd rá következtek a körvonalak, színek és az "üveg" effektus:



17. ábra: A projekt logója

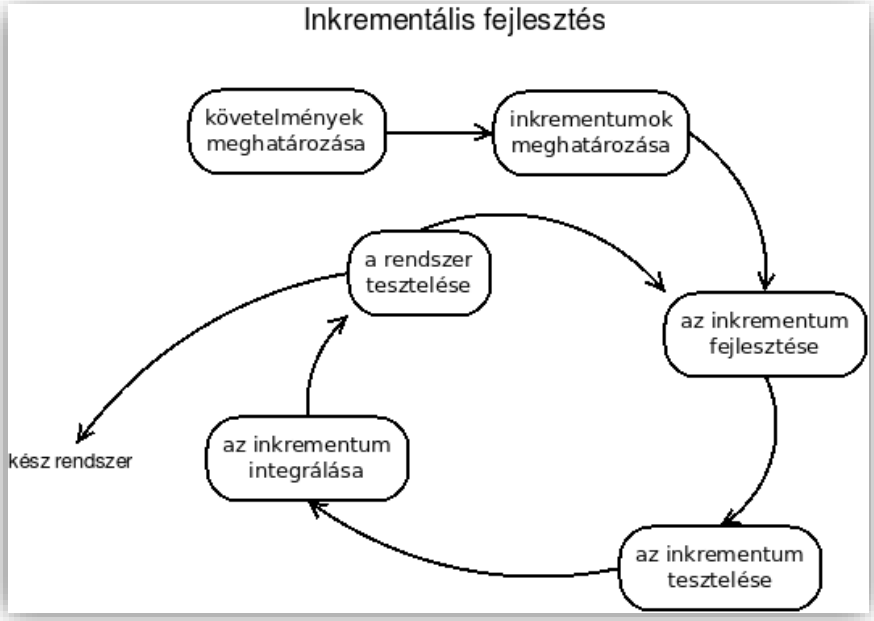


18. ábra: A kezdőpálya csempéje

3. Tervezési módszer

A tervezés az informatikai fejlesztés egyik kulcsfontosságú lépése, mivel meghatározza a szoftver vagy rendszer funkcionalitását és megfelelőségét a felhasználói igényeknek. Ezen tervezési folyamatok közül kiemelkedő fontosságú az adatbázis tervezése és implementációja, különösen egy gyorsan futó program esetén. Ez magában foglalja a táblák struktúrájának, mezőinek és a táblák közötti kapcsolatoknak a megtervezését.

Az előkészületek során, tervezési módszertan kiválasztásában a konzulensünk ajánlásával az inkrementális módszertan mellett döntöttünk. Az inkrementális megközelítési mód (19. ábra), hasonló megközelítésű, mint a vízesésmodell és az evolúciós fejlesztés. Mielőtt a tervezés megkezdődne, a vízesésmodell már azelőtt követeli a végleges követelményeket, a tervezőtől pedig a tervezési stratégiákat még az implementáció előtt. Ennek előnye, hogy jobban rendezhető, mivel így külön vannak választva a fázisok. A hátrány pedig az, hogy ezzel olyan problémák jönnek elő, amik miatt nem lesz lehetőség a változtatásra. Az evolúciós fejlesztési módszernél pedig a követelmények és a tervezéssel kapcsolatos döntések elhagyása mind megengedett, ezzel pedig az a hátrány jön létre, hogy egyes rendszerek nehezen megérthetők lesznek és gyengén strukturáltak.



19. ábra: az inkrementális fejlesztés lépései

Első lépésként a követelmények meghatározásánál kezdünk, majd ezután a követelmények inkremensekben való megfogalmazása és hozzárendelése következik. Ettől függ majd a szolgáltatás prioritása is, ezt pedig hamarabb kell biztosítani a megrendelőnek.

Miután a lépéseket meghatároztuk, az első lépés követelményeit majd részletesen definiálni kell. Ezt követően kifejlesztjük az inkrementst. Amíg fejlesztés alatt állunk, addig bőven lesz időnk a többi követelményeknek az elemzésére, viszont az aktuálison nem igazán változtatunk.

Amennyiben az inkremens kész van, bizonyos funkciókat már be is eépíthetjük. Ezzel több tapasztalatokat szerezhetünk a rendszerről és majd a későbbi követelményekhez segítségre is lehetnek. Így a félkész inkrementst akár el is elküldhetjük a megrendelőnek tesztelés céljából. Ezután a kívánt változtatásokat is el lehet végezni. Az új inkremens elkészülésekor a már meglévő inkrementseket integrálni kell az újakkal. Ezzel végül elkészül a teljes rendszer.

Az inkrementációs fejlesztés előnyei röviden a következő néhány pontban lehetne összefoglalni:

- A megrendelőnek nem kell megvárnia, míg a teljes rendszer elkészül. Már az első inkremensnél követheti a folyamatokat, így a szoftver használhatóvá válik még menet közben is.
- A megrendelők az előző inkrementseket bátran használhatják, így tapasztalatokat szerezhetnek.
- Kevesebb kockázat van arra, hogy az elképzelt projekt nem fog működni
- Kisebb a hiba esélye, mivel a rendszer legfontosabb szolgáltatásai lesznek a legtöbbet tesztelve.

Mindezek ellenére ennek a fejlesztésnek is megvannak a maga hátrányai. Fontos lenne, hogy az inkremensek megfelelően kisméretűek legyenek és minden inkrementációs lépésnek szolgáltatni kell valami rendszerfunkciót.

A weboldal elkészítését elsősorban a követelmények behatárolásával kezdtem. Hiszen fontos tisztázni, hogy mire lesz szükségem a fejlesztés során. A megvalósításhoz számos funkcionális és nem funkcionális követelményeknek kell megfelelnie. A funkcionális követelmények meghatározásával leírhatjuk, hogy a rendszernek milyen funkciókkal kell rendelkezni, hogyan kell működnie. A

követelményt leíró specifikációnak tartalmaznia kell a megrendelő által igényelt összes szolgáltatást, ezt a teljesség elvének nevezik. A másik az ellentmondás-mentesség, ami azt mondja ki, hogyne legyenek ellentmondó meghatározások. Tegye lehetővé a tulajdonos számára a weboldal tartalmának a karbantartását. Ez alatt a funkciók megfelelő feladat ellátását értem.

Regisztráció: a rendszernek engedélyeznie kell az új felhasználó és új admin regisztrálását. Bejelentkezése: a rendszernek engedélyeznie kell a felhasználó, az admin és a tulajdonos bejelentkezését a megfelelő felhasználónév, jelszó és szerepekör megadásával. Saját adatok: a rendszernek engedélyeznie kell, hogy a felhasználó megtekinthesse személyes adatait. Továbbá, a felhasználó a személyes adatain változtathasson, mint például email címén vagy jelszaván. Azt is engedélyeznie kell, hogy kijelentkezhessen felhasználói fiókjából.

A játék tervezése nem volt egyszerű feladat. Rengeteg mindent át kellett néznünk és beszélnünk. Ezért először megbeszélést tartottunk, hogy ki mivel fog foglalkozni. Bráz Bálint már foglalkozott a Unity-vel, ezért evidens volt, hogy ő fogja a játék programozás részét vezetni. Az ő javaslatára kezdtük el a játék készítéséhez szükséges programok kereséséhez. A modellezéshez és az animációk készítéséhez jól meg kellett fontolnunk, hogy milyen szoftvereket válasszunk. Utána kellett néznünk, hogy a Unity milyen grafikai programokkal kompatibilis, és milyen fájlokat támogat. Mivel Trischler Gergő rendelkezik művészi vénával, és már volt tapasztalata a rajzolással, ezért az ő javaslatára választottuk a Pixel Stúdió alkalmazást.

Mindezek után, második lépésként ki kellett találnunk, hogy milyen legyen a játék kategóriája. Órákon át ötleteltünk, hogy kitaláljuk, mégis milyen lenne az ideális játék, amellyel mi is szívesen játszanánk, illetve belefér az időkeretbe is. Mivel a csapatból többen is játszanak roguelike játékokkal, így a döntés hamar megszületett a kategóriát illetően. Miután kitaláltuk a játék kategóriáját, jöhetett a témaválasztás és a történet. Mivel ezekben a típusú játékokban nem a történeten van a hangsúly, ezért csak egyszerű háttértörténettel álltunk elő, amely megadja a játék hangulatát. A fejlesztés a játék nevének kitalálásával folytatódott, majd a jöhetett a játék menüjének és funkcióinak megtervezése.

A játék tervének elkészítése után nekiláthattunk a weboldal tervezésnek. Bár a játékon belül is lehetséges a regisztráció, azért létrehoztunk egy weboldalt, ahol ugyan szintén lehet regisztrálni, azonban itt a játékos az elért eredményeit is megtekintheti. Igyekeztünk a játék stílusához megfelelő oldalt tervezni, amelyen a felhasználók a fent említett funkciókon kívül még arról is tudakozódhattak, hogy miről szól a játék, a weboldal egyúttal egyfajta kedvcsinálóként is szolgál.

3.1 OOP megvalósítása

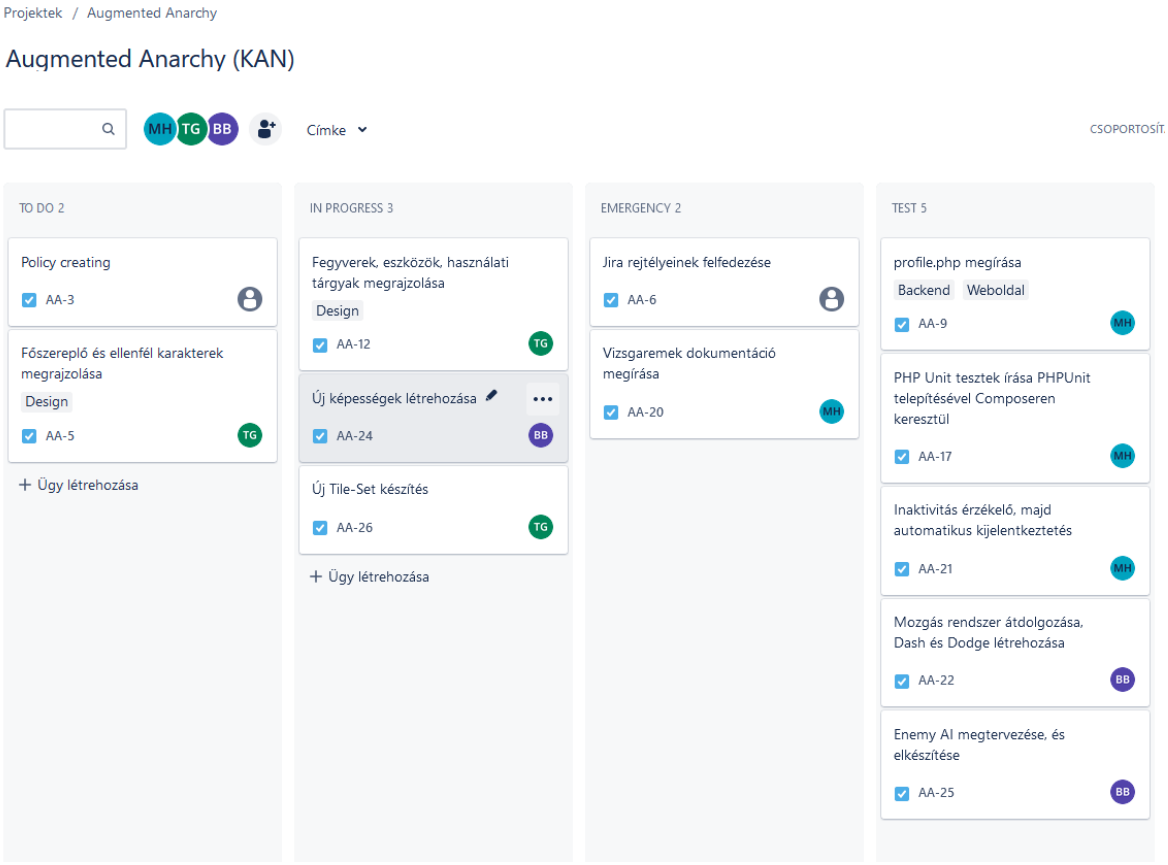
Mind a PHP, mind a C# kód objektumorientált programozási (OOP) elveket követ, bár eltérő nyelveken és környezetekben valósulnak meg.

A PHP kódban az objektumorientált programozás megvalósítása funkciók és azok csoportosítása révén történik. A Bejelentkezes() és Regisztracio() függvények objektumok, amelyek a felhasználók bejelentkezését és regisztrációját kezelik. Ezek az objektumok kezelik az adatokat, és azokon különböző műveleteket hajtanak végre.

A C# kód sokkal nyíltabban illeszkedik az objektumorientált programozás paradigmájához. A ApiRequestExample osztály egy teljes osztályként jelenik meg, amelynek saját adattagjai és metódusai vannak. Ez az osztály a Unity játékobjektumokkal is interakcióba lép, például a GameObject objektumokat használja az interfészének kezelésére. Emellett a különböző műveletek is osztályon belüli metódusokként vannak definiálva, például a SendPostRequest() vagy a showError().

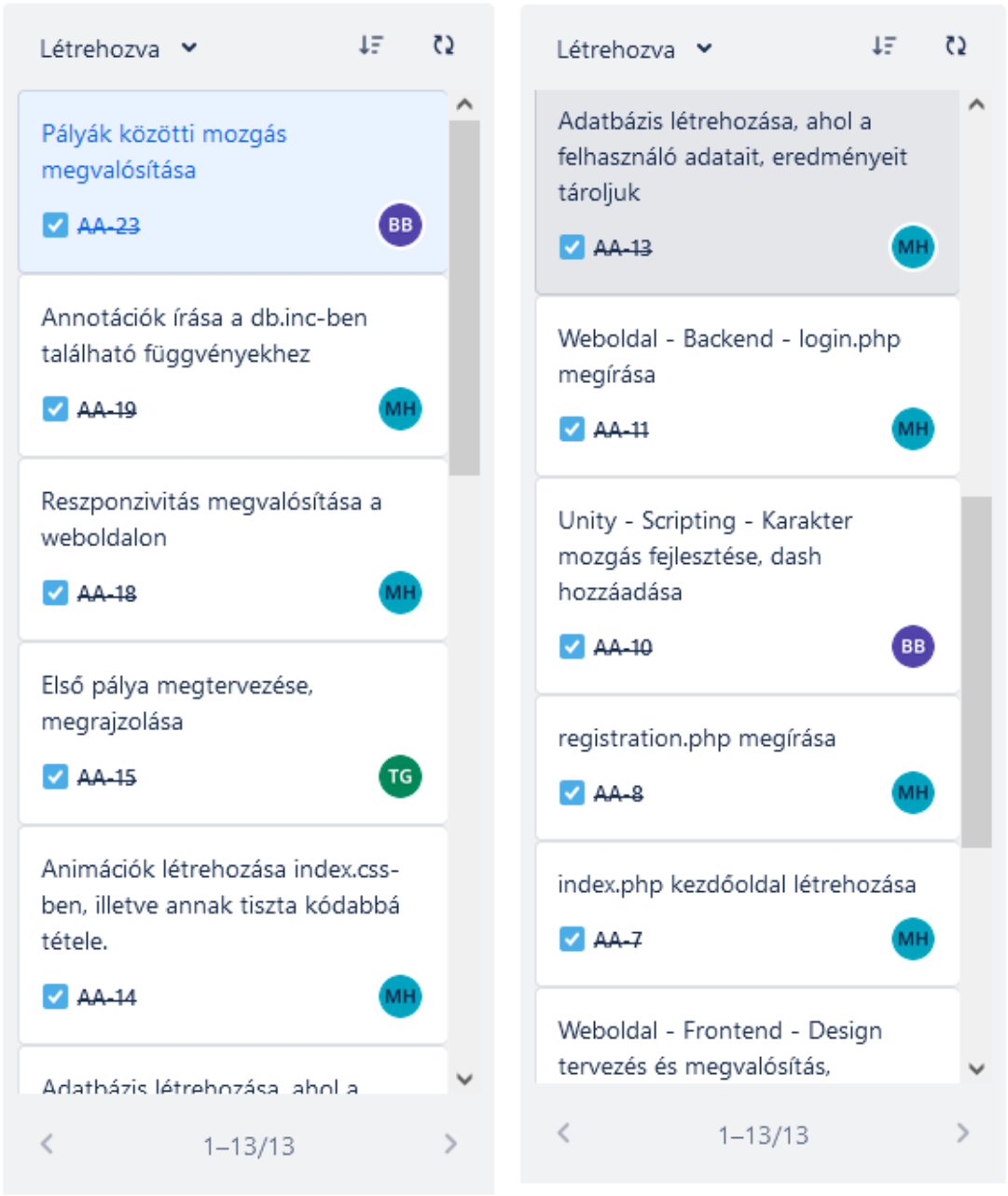
3.2 Projektmenedzsment szoftver

Tekintettel arra, hogy a tervezés során számos tényezőt kellett figyelembe vennünk, ezért a Jira projektmenedzsment szoftvert használtuk a folyamatok összesítéséhez és a feladatok ütemezéséhez. A 20. ábrán a csapatunk munkamegosztását igyekeztünk bemutatni.



20. ábra: Feladatok elosztása a Jira projektmenedzsment szoftverben

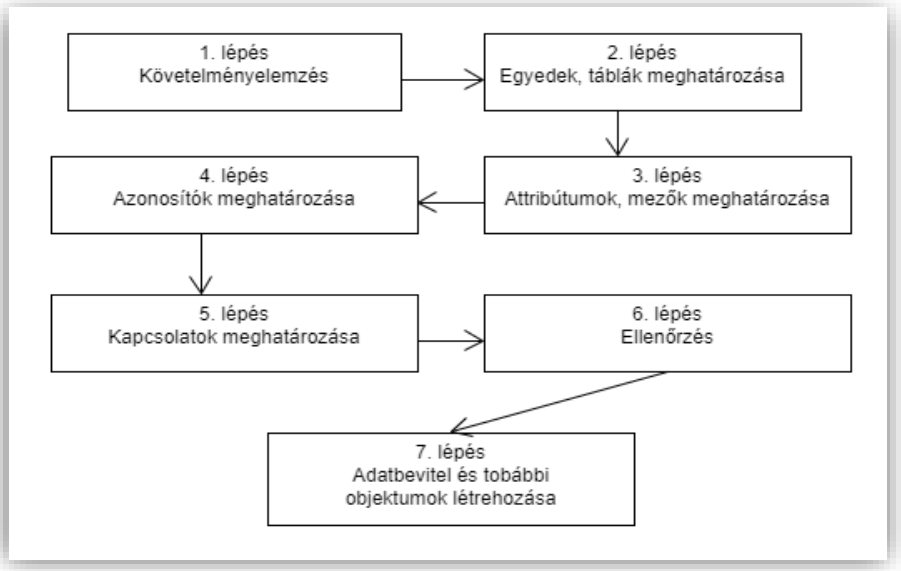
A Jira használata során akár határidőket is beállíthatunk az egyes részfeladatokhoz, melyekhez kommenteket is fűzhetünk, hogy az adott feladatból mi az, ami elkészült, és mi az, ami még hátra van. Az egyes oszlopok pedig jelzik, hogy az adott munkafolyamat milyen fázisban van. A másik nagy előnye a programnak, hogy a projekt állapotát mindannyian láthatjuk. Így mindannyian látjuk, hogy ki mivel haladt. A 21. ábrán pedig egy példát láthatunk arra, hogy összességében mely feladatok állnak készen.



21. ábra: Jira kimutatások az elkészült részfeladatokról

4. Adatmodell

Az adatmodell az adatbázis szerkezetét fogja meghatározni. Az adatok típusát, kapcsolatát, a feltételeket és a műveleteket egyben magába foglalja. Egy jól működő adatbázist először meg kell tervezni. Meg kell határoznunk, hogy melyik tulajdonságot szeretnénk tárolni az adatbázisunkban, majd ezek alapján kellene az egyedtípusokat és az adatbázis felépítését definiálni. Ezt a tervezést hat lépésben lehet meghatározni, ezt a 22. ábrán mutatjuk be:



22. ábra: Az adatbázis tervezés lépései

A normalizálás az olyan adatok rendszerezését jelenti, ami az adatbázisban található. Létrehozhatunk táblákat és kapcsolatokat létesíthetünk közöttük, ezeket szabályokkal tehetjük csak meg. Ennek az célja, hogy az adatokat védje és rugalmasabbá tegye őket. Itt a redundanciák és az inkonzisztens függőségek kiküszöbölése is a cél. Az olyan adatok, amik redundánsak, a lemezterületet feleslegesen foglalják, ezzel karbantartási problémákat okozhatnak. Ha több adatot kell módosítani, azt minden helyen pontosan ugyanúgy kell elvégezni, ahogy az előzőknél tettük. Ezt pedig a két táblámnak az összekötésében alkalmaztuk.

A normalizálásnak is van egy-két normálformája, ezeket az alábbiakban felsorolásként meg is említeném:

- (1NF) - ha minden sorban pontosan egy attribútum érték áll
- (2NF) - ha minden másodlagos attribútum teljesen függ a kulcstól
- (3NF)
    1. akkor és csak akkor, ha 2NF-ben van és egyetlen másodlagos attribútum sem függ tranzitíven a kulcstól
    2. akkor és csak akkor, ha minden másodlagos attribútum funkcionálisan függ egymástól és teljesen függ a kulcstól
    3. akkor és csak akkor, ha 2NF-ben van és a másodlagos attribútumok között nincsen funkcionális függőség

- Boyce/Codd - minden elsődleges attribútum teljes funkcionális függőségben van azokkal a kulcsokkal, melyeknek nem része, csak kulcs jellegű relációs függőségek vannak a relációban.

A relációs adatmodell egyrészt definiálja az adatszerkezeteket. Fontos még kiemelni, hogy a relációs adatmodell a logikai adatbázis kereteit határozza meg. A relációs adatbázisséma egy attribútum halmaz, amelyhez név tartozik. Minden attribútumhoz pedig tartozik egy értékkészlet, amelyből felveheti értékeit. Az attribútum halmaznál, amelyen a tábla minden sora különbözik, azt viszont szuperkulcsnak nevezzük.

Ha pedig minimális a szuperkulcs akkor azt pedig kulcsnak nevezzük. Egy relációsémában több kulcs is előfordulhat, ezért egyet ki kell jelölni, ami az elsődleges kulcsunk lesz. Ha egy attribútum alkotja a kulcsot, azt egyszerű kulcsnak nevezzük, ha több attribútum alkotja a kulcsot, azt pedig összetett kulcsnak hívjuk. A külső kulcs pedig az, amikor az attribútum a másik sémának az elsődleges kulcsára hivatkozik.

Kapcsolatnak nevezzük azt, amikor az egyedtípusok között viszony van. A kapcsolat mindig valóságos objektumok közötti viszonyt fejez ki, hiszen az egyed ilyen objektumokat képvisel. A kapcsolódó egyedek között általában nem egyenrangú a viszony, hanem lehet egyfajta irányításról is beszélni. A kapcsolat meghatározója a tulajdonos és a kapcsolat másik oldalán lévő egyed, vagy egyedek, a tagok. Ahogy a normalizálásnál is, az egyedkapcsolatot is alkalmaztam az adatbázisaimnál.

Az adatbázisban (23. ábra) három fő tábla található: a Users, a Players és a Mapscores. Ezek az adattáblák különböző információkat tárolnak a játékhoz kapcsolódóan.

Users Tábla:

- **id (Primary Key):** Egyedi azonosító a felhasználókhoz, ami alapján azonosíthatók.
- **nev:** A felhasználói nevet tartalmazza, egyedi a rendszeren belül.
- **email:** A felhasználó email címét tárolja.
- **jelszo:** A felhasználó jelszavát titkosított formában tárolja.

- **regisztracioDatuma:** A regisztráció időpontját tárolja, amely segít nyomon követni a felhasználók csatlakozását a rendszerhez.
- **modositasDatuma:** A legutóbbi módosítás dátuma, amely nyomon követi a felhasználói adatok változásait.

Mapscores Tábla:

- **record_id (Primary Key):** Egyedi azonosító a pályaszereplésekhez, ami alapján azonosíthatók.
- **map_id:** Az adott pálya azonosítója, amelyhez a pontszámok tartoznak.
- **user_id (Foreign Key):** Idegen kulcs, amely kapcsolódik a Users táblához, jelezve, hogy mely felhasználóhoz tartozik az adott pontszám.
- **map_time:** Az eltelt időt tárolja egy adott pálya teljesítése során.
- **health:** A játékos egészségét tárolja az adott pontszám rögzítésekor.
- **score:** A játékos által elért pontszámot rögzíti az adott pályán.

Az idegen kulcs (user_id) biztosítja a kapcsolatot a két tábla között, amely lehetővé teszi, hogy a pályaszereplések egyértelműen hozzárendelődjenek a felhasználókhoz. A Mapscores tábla lehetővé teszi a játékbéli teljesítmények rögzítését és nyomon követését, míg a Users tábla az egyes felhasználók adatait tárolja, beleértve a regisztrációs információkat és az azonosítókat is.

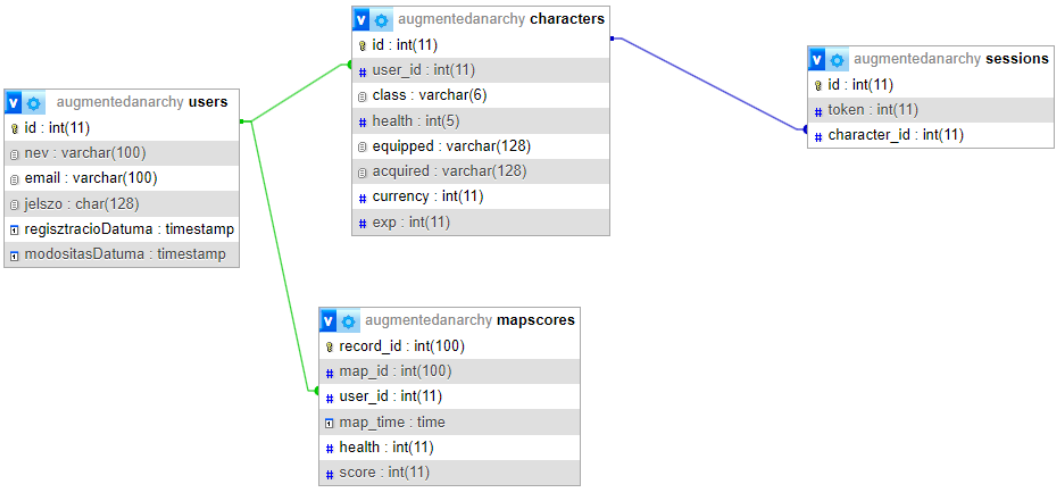
Players Tábla:

- **user_id (Idegen Kulcs):** Kapcsolódik a Users táblához, jelzi, hogy melyik felhasználóhoz tartozik a játékos karakter.
- **character_id (Primary Key):** Egyedi azonosító a játékos karakterekhez, amelyeket a felhasználók létrehoztak vagy birtokolnak.
- **skills:** A karakter készségeit vagy képességeit tárolja, például speciális képességek vagy tulajdonságok.
- **inventory:** A karakter tárgyainak vagy eszközeinek listáját tárolja, amelyekkel a játékos rendelkezik.

A Players tábla lehetővé teszi a felhasználók számára, hogy egyedi játékos karaktereket hozzanak létre a játék során, és arra is lehetőséget nyújt, hogy különböző képességeket és felszereléseket rendeljenek hozzájuk.

Sessions Tábla:

- **id:** Egyedi azonosító a session tárolásához.
- **token:** Bejelentkezéskor ez alapján kéri le az adatbázisból, hogy mely tárgyakkal, képességekkel rendelkezik a karakter.
- **character_id (Idegen kulcs):** Kapcsolódik a characters táblához, azt jelzi, hogy a játékos melyik karakterével játszik.



23. ábra: Adatbázis-kapcsolat

Az adatbázisban több tárolt eljárás is található, amelyek különböző funkciókat látják el az adatok kezelésében és manipulálásában. Ezek az eljárások közvetlenül a MariaDB adatbázisban futtathatók, és lehetővé teszik a rendszer működését a következő módokon:

- Felhasználói adatok módosítása: Például e-mail cím vagy jelszó változtatása, valamint felhasználónév módosítása.
- Bejelentkezés validálása: Ellenőrzi, hogy a megadott felhasználónév és jelszó egyezik-e a tárolt adatokkal.
- Felhasználó regisztrációja: Új felhasználók regisztrálása az adatbázisba, azonosítók és egyéb adatok létrehozása.
- Pályaszereplések kezelése: Az adott felhasználó által elért eredmények rögzítése és lekérdezése az adatbázisból.

Mivel a felhasználók bizonyos adatait használjuk, illetve tároljuk, ezért a

weboldalon történő regisztrációkor a felhasználónak bele kell egyezni, hogy elfogadja az adatkezelési tájékoztatót, melyet a GDPR (24. ábra) alapján mintáztunk. A GDPR rendelet az Európai Parlament által létrehozott jogszabály, az elnevezés rövidítése az angol General Data Protection Regulation kifejezés szava, ami magyarul annyit jelent, hogy: Általános Adatvédelmi Rendelet. Ezt pedig a szabályzat fogja majd a felhasználók számára biztosítani ezt az adatvédelmet.

24. ábra: GDPR-adatvédelem

5. Részletes feladatspecifikáció, algoritmusok

Mivel manapság az internetes tartalmakat többnyire telefonról és tablet eszközről interneteznek, ezért olyan weboldalakat célszerűbb létrehozni, amelyek ezeknek az eszközöknek a képernyőméretével kompatibilis. Erre lett feltalálva a reszponzív webszerkesztés. Ennek a jelentése annyi, hogy ahol a weboldal automatikusan alkalmazkodik ahhoz az eszközhöz, amelyen megjelenítik, ezzel optimális felhasználói élményt nyújt. A reszponzív webszerkesztés lehetőséget ad arra, hogy ugyanaz a weboldal eszköztől, operációs rendszertől, böngészőtől függetlenül bármilyen környezetben tökéletesen jelenjen meg.

A reszponzív webszerkesztés nem más, mint egy weboldal készítési eljárás. A hagyományos weboldalkészítéssel szemben a fejlesztőnek már a grafikai tervezés időszakában egészen másfajta megközelítést kell alkalmaznia. Mivel eszköz független weboldalt tervezünk, ezért a grafikai tervet kiegészíti több olyan terv, amely bemutatja, a különböző felbontású eszközökön az oldal hogyan fog megjelenni. Tehát többfajta elrendezést és megjelenést tervezünk, az alkalmazott technológia teszi lehetővé, hogy a weboldal a megjelenés pillanatában az eszköz felbontásához igazodó változatot

jelenítse meg.

Ahhoz, hogy ennek a követelménynek megfeleljen a weboldalunk, megpróbáltunk arra törekedni, hogy minden eszközön olvasható és használható lehessen a weboldal.

Ezt a reszponzivitást a **@media screen** paranccsal oldottuk meg. Itt beállítottam a képernyő minimum méretét képpontban és ahhoz állítottuk a háttereket és az elemeket. Így a böngésző kicsinyítésekor, amikor eléri ezt a minimum értéket, a tartalom magát szabályozza a megadott méretre.

6. Forráskód

Ahhoz, hogy lehetővé tegyük a játék, a weboldal és az adatbázis közötti kommunikációt és adatcserét API-t használunk. A 25. ábrán látható PHP kódrészlet egy szerveroldali API-t valósítanak meg, amely kezeli az üzenetkéréseket, amik a szerverhez érkeznek.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    /**
     * Kezeli a bejelentkezési kérést.
     */
    if(isset($_POST['mode']) && $_POST['mode'] == 'login') {
        if(isset($_POST['username']) && isset($_POST['password'])) {
            $username = $_POST['username'];
            $password = $_POST['password'];

            $result = Bejelentkezes($username, $password);
            if($result == "TRUE"){
                $response = $result;
            }
            else{
                $response = "Nem létezik ilyen felhasználónév és jelszó párosítás!";
            }
        } else {
            $response = "Hiányzó paraméter!";
        }
    }
```

25. ábra: Megfelelő típusú-e a kérés?

Első lépésként a fenti kód megvizsgálja, hogy az érkező kérés típusa POST-e vagy sem ($_SERVER['REQUEST_METHOD'] === 'POST'). Ez azért fontos, mert a különböző műveletekhez csak POST kérések fogadhatók el.

```
    else if(isset($_POST['mode']) && $_POST['mode'] == 'register') {
        if(isset($_POST['email']) && isset($_POST['username']) && isset($_POST['password'])){
            $email = $_POST['email'];
            $username = $_POST['username'];
            $password = $_POST['password'];

            $result = Regisztracio($email, $username, $password);
            if($result == "TRUE"){
                $response = $result;
            }
            else{
                $response = "Már létezik ilyen felhasználó névvel vagy email címmel profil!";
            }
        } else {
            $response = "Hiányzó paraméter!";
        }
    }

    else {
        $response = "Nem létezik ilyen típusú kérés!";
    }
} else {
    $response = "Nem megfelelő kérési metódus!";
}
```

26. ábra: Milyen a kérés típusa?

A 26. ábrán láthatjuk, ha a kérés POST típusú, akkor belépünk az elágazásba, ahol megnézzük, hogy milyen műveletet kérnek (login vagy register).

Ha a kérés bejelentkezés (login), akkor ellenőrizzük, hogy a felhasználónév (username) és a jelszó (password) adatok megérkeztek-e az üzenetben (isset($_POST['username']) és isset($_POST['password'])). Ha igen, akkor ezeket kinyerjük az üzenetből, majd meghívjuk a Bejelentkezes() függvényt, amely ellenőrzi a bejelentkezési adatokat. A választ a $response változóban tároljuk.

Ha a kérés regisztráció (register), akkor hasonlóan ellenőrizzük az email cím (email), felhasználónév (username) és jelszó (password) meglétét az üzenetben. Ha minden adat rendelkezésre áll, meghívjuk a Regisztracio() függvényt, amely létrehozza az új felhasználót az adatbázisban. A választ szintén a $response változóban tároljuk.

Ha egyik művelet sem feleltethető meg (else ág), akkor egy hibaüzenetet állítunk be a $response-ban, hogy nincs ilyen típusú kérés.

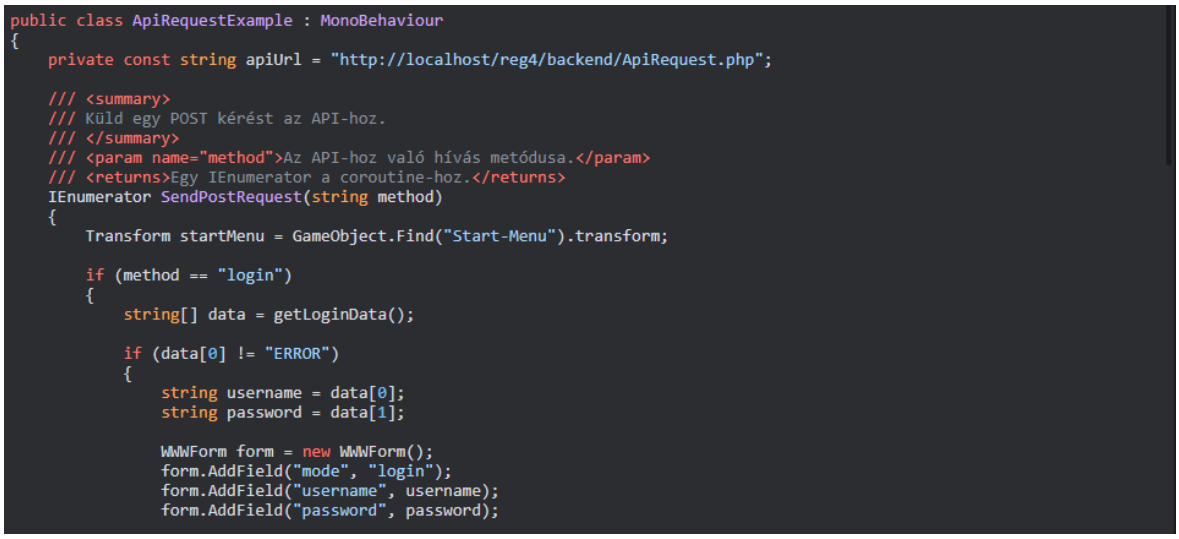
Végül, ha a beérkezett kérés nem POST típusú (else ág), akkor szintén hibát jelzünk a $response-ban, hogy nem megfelelő kérési metódus.

```php
header('Content-Type: application/json; charset=utf-8');
echo json_encode($response, JSON_UNESCAPED_UNICODE);

/**
 * Ellenőrzi a bejelentkezési adatokat.
 *
 * @param string $username A felhasználónév.
 * @param string $password A jelszó.
 * @return string A bejelentkezés ellenőrzésének eredménye.
 */
function Bejelentkezes($username, $password) {
    include 'inc/db.inc';
    $result = validateLogin(csatlakozas(), $username, $password);
    return $result;
}

/**
 * Regisztrál egy új felhasználót.
 *
 * @param string $email Az email cím.
 * @param string $username A felhasználónév.
 * @param string $password A jelszó.
 * @return string A regisztrációs folyamat eredménye.
 */
function Regisztracio($email, $username, $password) {
    include 'inc/db.inc';
    $result = registerUser(csatlakozas(), $email, $username, $password);
    return $result;
}

?>
```

27. ábra: A válasz formátuma

A választ (27. ábra) végül JSON formátumban küldjük vissza a kliensnek (header('Content-Type: application/json; charset=utf-8')), ahol a $response tömböt JSON formátumba alakítjuk (json_encode($response, JSON_UNESCAPED_UNICODE)). Ez a válasz tartalmazza a kliensnek az adott művelet végrehajtásának eredményét vagy esetleges hibáját.

A következőkben a C# kód kerül részletezésre, amely az Unity alkalmazás része, és az API-hoz való kommunikációt valósítja meg. Az kód célja, hogy felhasználói felületet biztosítson a felhasználóknak a regisztrációhoz és bejelentkezéshez, valamint kapcsolódni az előzőleg említett szerveroldali API-hoz, hogy végrehajtsa ezeket az akciókat.



```
public class ApiRequestExample : MonoBehaviour
{
    private const string apiUrl = "http://localhost/reg4/backend/ApiRequest.php";

    /// <summary>
    /// Küld egy POST kérést az API-hoz.
    /// </summary>
    /// <param name="method">Az API-hoz való hívás metódusa.</param>
    /// <returns>Egy IEnumerator a coroutine-hoz.</returns>
    IEnumerator SendPostRequest(string method)
    {
        Transform startMenu = GameObject.Find("Start-Menu").transform;

        if (method == "login")
        {
            string[] data = getLoginData();

            if (data[0] != "ERROR")
            {
                string username = data[0];
                string password = data[1];

                WWWForm form = new WWWForm();
                form.AddField("mode", "login");
                form.AddField("username", username);
                form.AddField("password", password);
```

28. ábra: OOP megvalósítás

A 28. ábrán látható a kód fő részét képező osztály, amely a Unity MonoBehaviour-ból származik (ApiRequestExample). Ez az osztály felelős a POST kérések küldéséért az API számára, attól függően, hogy a felhasználó bejelentkezni vagy regisztrálni próbál-e.

```
                using (UnityWebRequest www = UnityWebRequest.Post(apiUrl, form))
                {
                    yield return www.SendWebRequest();

                    if (www.result != UnityWebRequest.Result.Success)
                    {
                        Debug.LogError("Hiba a kérés küldésekor: " + www.error);
                    }
                    else
                    {
                        Debug.Log("Sikeres kérés!");
                        Debug.Log("Válasz: " + www.downloadHandler.text);

                        bool isResponseTrue = www.downloadHandler.text.Replace("\"", "").ToLower() == "true";
                        Debug.Log("A válasz igaz vagy hamis? " + isResponseTrue);

                        if (isResponseTrue)
                        {
                            GameObject mainMenu = startMenu.Find("MainMenu").gameObject;
                            GameObject loginScreen = startMenu.Find("Login_Screen").gameObject;

                            mainMenu.SetActive(true);
                            loginScreen.SetActive(false);
                        }
                        else
                        {
                            showError(www.downloadHandler.text.Replace("\"", ""));
                        }
                    }
                }
            }
        }
```

29. ábra: Adatok küldése és fogadása az API-tól

A 29. ábrán a SendPostRequest metódus küldi el a POST kéréseket az API számára a megadott módszerrel (login vagy register). Az elküldött adatokat a felhasználói felületen bevitt adatokból szerzi be, például a felhasználónévből és jelszóból. Ezeket az adatokat egy formába helyezi (WWWForm), majd elküldi az API-hoz az UnityWebRequest.Post

segítségével. A választ az API-tól a downloadHandler.text-ből olvassa ki, majd ennek megfelelően reagál.

```
/// <summary>
/// Lekéri a bejelentkezési adatokat a felhasználói felületről.
/// </summary>
/// <returns>Egy tömb, amely tartalmazza a felhasználónevet és a hashelt jelszót.</returns>
public string[] getLoginData()
{
    Transform startMenu = GameObject.Find("Start-Menu").transform;
    Transform loginScreen = startMenu.Find("Login_Screen");
    string username = loginScreen.Find("Username Fields").Find("Username").gameObject.GetComponent<TMP_InputField>().te
    string password = ComputeSha512Hash(loginScreen.Find("Password Fields").Find("Password").gameObject.GetComponent<TM

    if (username != "" || password != "")
    {
        return new string[] { username, password };
    }
    else
    {
        showError("Kérlek töltsd ki az összes mezőt!");
        return new string[] { "ERROR" };
    }
}
```

30. ábra: Bejelentkezési adatok küldése a játékon keresztül

A 30. ábrán egy kis részletet láthatunk a regisztrációs és bejelentkezési folyamatból. A getLoginData és getRegistrationData metódusok segítségével a felhasználói felületen bevitt adatokat szedi össze, például a felhasználónevet, jelszót és e-mail címet. Ezeket az adatokat továbbítja a SendPostRequest metódusnak, hogy elküldje az API számára.

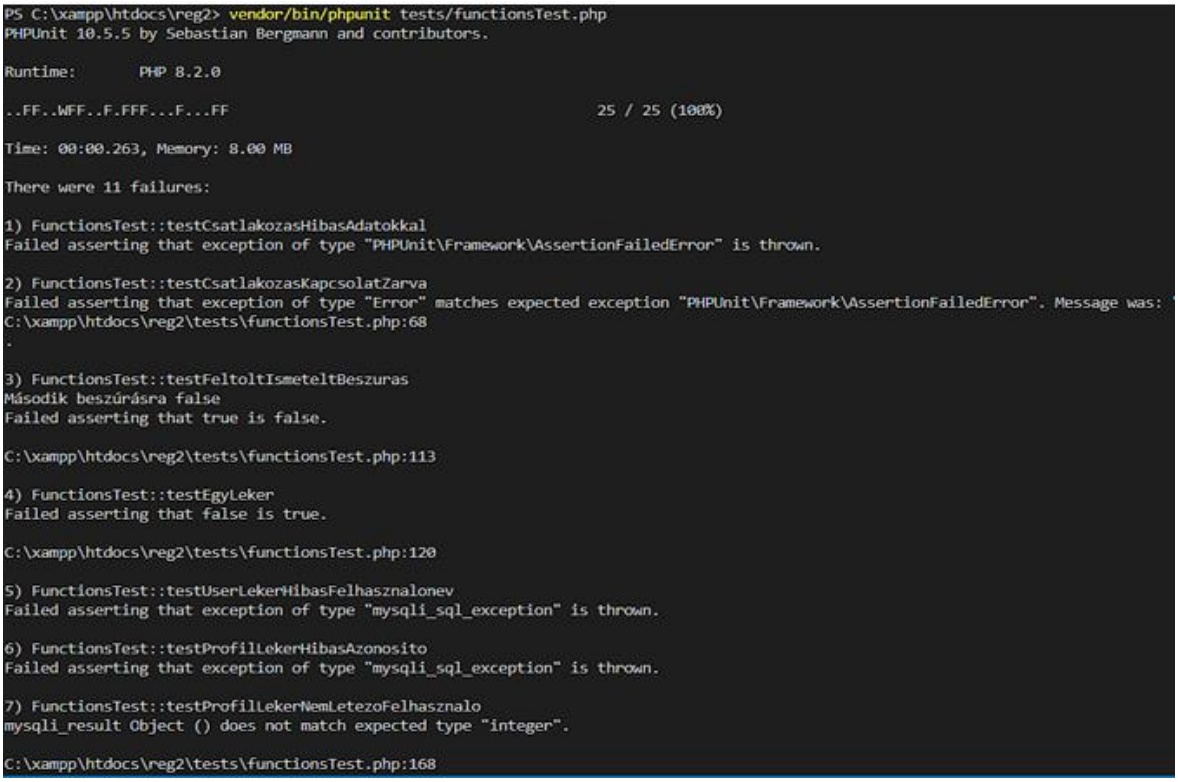
A showError metódus megjeleníti az esetleges hibaüzeneteket a felhasználói felületen, például ha valami nem megfelelően lett kitöltve.

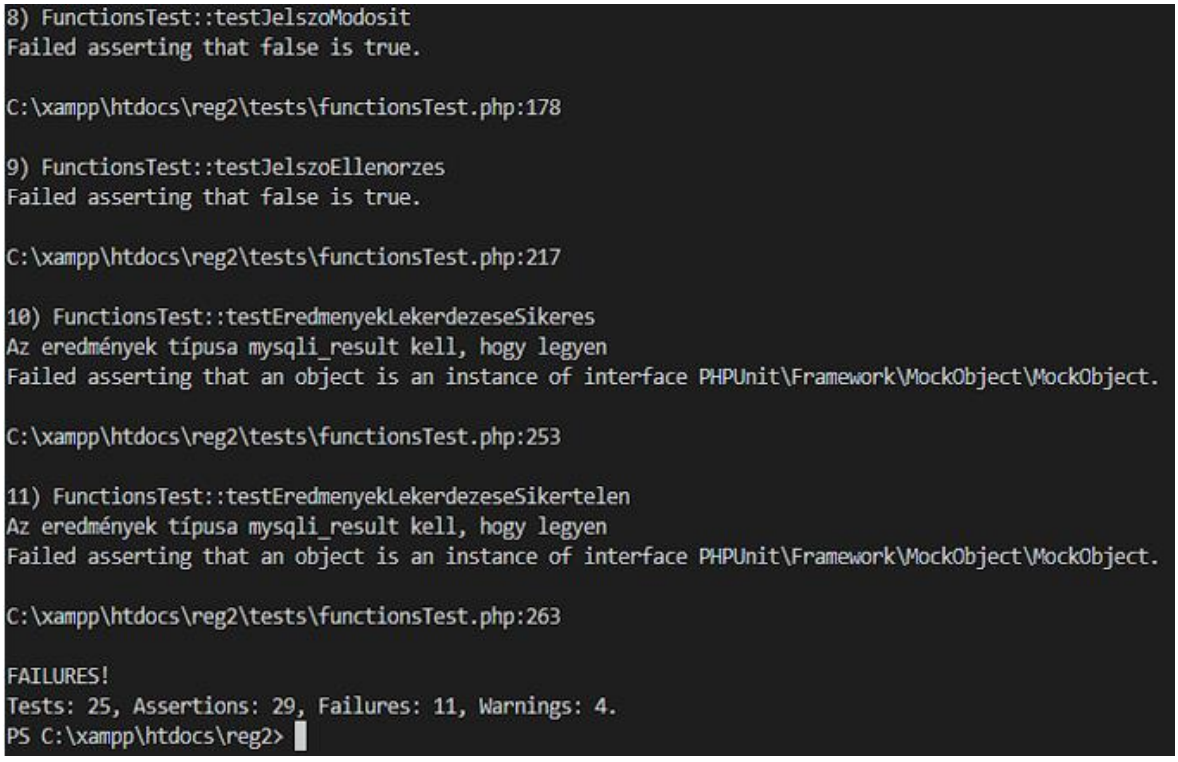
Összességében a C# kód felelős a felhasználói felület és az API közötti kommunikációért, amely segítségével a felhasználók regisztrálhatnak és bejelentkezhetnek egy adott rendszerbe. A PHP kód pedig a szerveroldalon futó API-t valósítja meg, amely kezeli ezeket a kéréseket, ellenőrzi az adatokat és válaszol a kliens felé az eredményekkel. A két kód együttműködve lehetővé teszi a felhasználók számára, hogy biztonságosan kommunikáljanak és interakciót létesítsenek a szerverrel.

7. Tesztelési dokumentáció

A játék és a weblap tesztelése jelentős hányadát tette ki a fejlesztésnek. Gondolunk itt a karakter mozgatására, a pályák vagy a menü egyes részeinek a tesztelésére. A bejelentkezést is tesztelni kellett, hogy a megadott adatokkal engedi-e a belépést vagy sem. A manuális tesztesek elérhetőségét az „Alkalmazott fejlesztői eszközök” Dependency Management részénél tárgyaltuk. Alább részletezzük, hogy mennyi és milyen típusú unit teszt készült a projektről. A 31. ábrán pedig bemutatjuk ezen tesztek sikerességét.

1. **Csatlakozás adatbázishoz (4 unit teszt):**
    - Ellenőrzi, hogy a kapcsolatobjektum megfelelően létrejön.
    - Teszteli a sikeres és sikertelen kapcsolódást.
    - Vizsgálja, hogy a kapcsolat zárása után várt hibakivétel történik-e.
2. **Profil lekérdezése adatbázisból (2 unit teszt):**
    - Ellenőrzi, hogy a lekérdezés objektuma helyesen létrejön.
    - Teszteli, hogy a lekérdezés eredménye nem üres.
3. **Felhasználó műveletek az adatbázisban (6 unit teszt):**
    - Teszteli a felhasználó beszúrását, és ellenőrzi, hogy a visszatérési érték igaz vagy hamis.
    - Vizsgálja az ismételt beszúrás helyes működését.
    - Ellenőrzi az egyedi felhasználó lekérését és annak visszatérési értékét.
    - Teszteli a felhasználó lekérését hibás felhasználónév esetén.
4. **Profil és jelszó műveletek adatbázisban (11 unit teszt):**
    - Teszteli a profil lekérését, és ellenőrzi, hogy a visszatérési érték megfelelő típusú vagy 0, ha nincs találat.
    - Vizsgálja a jelszó módosítását, és ellenőrzi, hogy a visszatérési érték igaz vagy hamis.
    - Teszteli a jelszó ellenőrzését különböző esetekben.
5. **Játékos eredményeinek lekérdezése adatbázisból (2 unit teszt):**
    - Teszteli az eredmények lekérését és ellenőrzi, hogy a visszatérési érték megfelelő típusú vagy hamis, ha a lekérdezés sikertelen.





31. ábra: Unit tesztek

8. Továbbfejlesztési lehetőségek

A játék tervezése során nagyon sokat gondolkodtunk, hogy mi legyen az a pont, ahol meghúzzuk a határt és még a határidő letelte előtt be tudjuk fejezni a projektet. Hosszas tanakodás után a játék grafikájából, részletességéből kellett visszább vennünk, és helyette inkább a fő funkciók tökéletes működésére koncentráltunk. Szerencsére a szép menürendszerrel és az egyszerű, de nagyszerű játékmenettel sikerült elkészülnünk. Nyilván, ahhoz, hogy a játékot kiadhassuk, még rengeteget kell dolgoznunk rajta. Főleg azon, hogy a látvány megfeleljen a mai, modern játékok által támasztott magas elvárásoknak.

Az előzetes tervek közt szerepelt, hogy a játékosok még több fegyver közül tudjanak választani, illetve több karakter kinézetet is szerettünk volna. Ezek mellett szerettünk volna még három-négy extra pályát, de ezek inkább a mennyiségi kérdéshez tartoznak, mint a minőségihez.

A statisztika további bővítése szintén nagyon fontos lenne, mivel az emberek szeretik, ha minél részletesebb adatokat látnak arról, hogy mit is értek el a játék során. Ilyen bővítés lenne például az, hogy a játékon belül melyik fegyvert használta a legtöbbször, mennyi sebzést osztott ki vele stb.

A weboldallal kapcsolatban is szeretnénk rengeteg fejlesztést eszközölni. A hírek menüpont alatt folyamatosan bővülő tartalmat szeretnénk a játékosok számára biztosítani, ahol értesülhetnek a legújabb frissítésekről vagy újabb build verziókról. A regisztrációt kibővítenénk még Google fiókos regisztrációval is.

Ennek a játéknak az elkészítését a játékok iránti szenvedélyünk, illetve a csapattagjaink által kedvelt játékok miatt készítettük. A bevezetőben már szó volt róla, hogy a csapattagok nagy kedvelői a műfajnak. Valamint az is szerepet játszott a projekt témájának döntésében, hogy olyan programot szerettünk volna elkészíteni, amelyet később lehetne bővíteni, illetve referenciamunkaként is tudunk használni. A roguelike műfajú játék tökéletesen megfelelt ezeknek a kritériumoknak. A referencia szempontjából igazából lényegtelen a játék műfaja, inkább a komplexitás miatt szerepelhet referenciaként.

IV. ÖSSZEGZÉS

A projektmenedzser sokat fejlődött a csapattagok koordinálásában, illetve a megbeszélt feladatok szétosztásáról és betartatásától. Rendszeres megbeszéléseket szervezett és bonyolított le a csapattagokkal Discordon, illetve az iskolában ő volt a felelős a kapcsolattartásért a tanárokkal, valamint bemutatta nekik, hogy a csapat mivel, mennyit haladt a legutóbbi megbeszélés óta. A csapat kollektíven sokat fejlődött ezekből a készségekből, mivel az előadások során mindenkinek prezentálnia, beszélnie kellett a feladatáról.

A Jirát és GitHubot valamivel többet kellett volna használnunk, de mivel kicsi a projektünk, és mindennap találkozunk a csapattársakkal, ezért szóban sokkal gyorsabban és hatékonyabban tudunk mindent megbeszélni. Illetve, mivel annyira külön-külön területeken dolgoztunk, ezért kevésszer volt olyan, hogy egymás fájljaiban kellett volna átírni valamit, a weblappal például egyikünk foglalkozott, aki tudta magának lokálisan menteni az előrehaladását. Viszont továbbra is úgy gondoljuk, hogy bár kevés átfedés volt a fájlok között, a mentés és verziókezelés továbbra is kulcsfontosságú, főleg, ha a modularitást is szem előtt tartjuk, hogy a projektünk könnyen bővíthető legyen, és akkor még fontosabb lesz a verziókezelés.

Hivatkozások

19. ábra: Sallai, A. SzIT. Oktatás: Programozás: Fejlesztési modellek és módszertanok. https://szit.hu/doku.php?id=oktatas:programozas:fejlesztesi_modellek_es_modszertanok

Hozzáférve: 2024. április 11.

22. ábra: Sallai, A. SzIT. Oktatás: Adatbázis-kezelés tananyag. https://szit.hu/doku.php?id=oktatas:adatbazis-kezeles:tananyag

Hozzáférve: 2024. április 11.

24. ábra: PricewaterhouseCoopers Magyarország Kft. GDPR. PricewaterhouseCoopers. https://www.pwc.com/hu/hu/szolgaltatasok/vallalati_kockazatkezeles/gdpr.html

Hozzáférve: 2024. április 11.