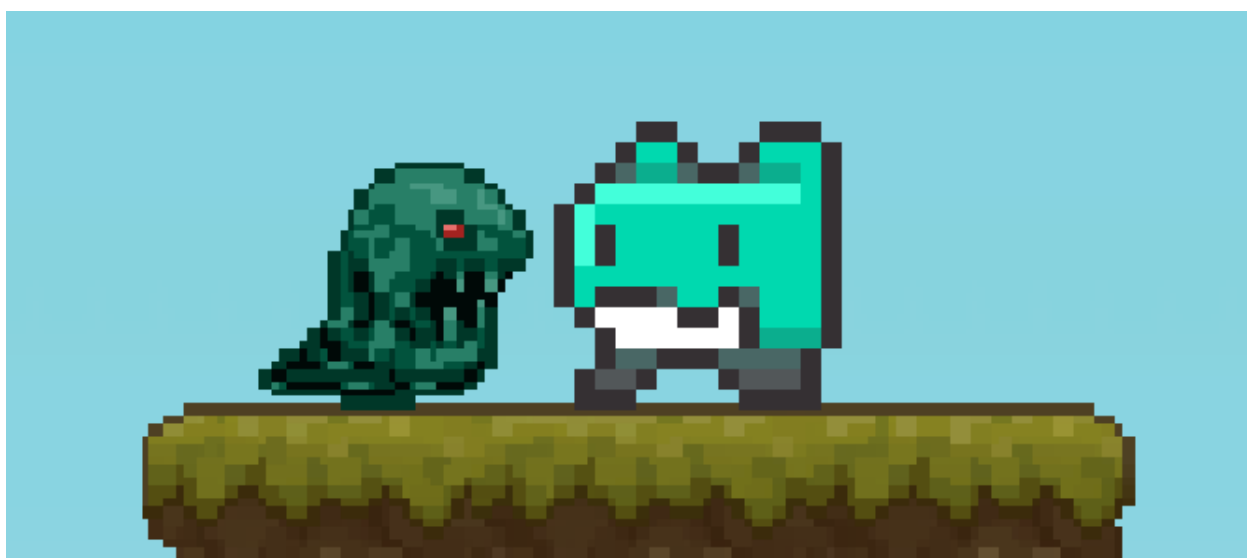


Mudskip the Slime

Pannonhalmi Főapátság

Szegedi SOB Technikuma



Szakképesítés:

Szoftverfejlesztő és -tesztelő
(506131203)

Témavezető: Rédei Dávid

Készítők:

Nagy Máté Károly
Rózsa Kevin
Szántó Tamás Lajos

Szeged, 2025

A projekt rövid összefoglalása.....	4
A projekt témájának, céljának részletes kifejtése.....	4
A projekt tervezés szempontjai, alkalmazott módszereinek bemutatása.....	4
Projekt feladatainak megosztása.....	7
Projekt ütemezés és alkalmazott módszereinek bemutatása.....	8
A felhasználói felület részletezése, felhasznált programok, hálózatok.....	10
Az Adatbázis és a REST API részletes bemutatása.....	24
Eszközök.....	32
REST API tesztelése.....	35
Összegzés.....	39
Források.....	40

A projekt rövid összefoglalása

Projekt címe: Mudskip the Slime

Célunk egy szórakoztató játék készítése ahol a játékosok össze hasonlíthatjuk tudásukat, ami a High Score rendszer segítségével tudnak nyomon követni.

A projekt témájának, céljának részletes kifejtése

A **Mudskip the Slime** egy videójáték, amelyben a játékos egy slime-ot irányít, aki különféle akadályokat küzd le. A projekt neve a főszereplő karakterből ered, aki egy egyedi mozgás mechanikával rendelkező slime, és a játék alatt fő célja hogy megtalálja háziállat Skipper.

Rövid és hosszú távú célokat tűztünk ki. **Rövid távú célunk**, hogy létrehozzunk egy jól működő, szórakoztató platformert egyedi játékelemekkel és kihívásokkal, amely könnyen kezelhető, de változatos mechanikákat tartalmaz. A játékosnak különböző akadályokat kell leküzdenie, ugrálnia, és a környezet adottságait kihasználva eljutnia a pályák végére.

Hosszú távú célunk, hogy a **Mudskip the Slime** egy szerethető és továbbfejleszthető játék legyen, amely esetleg további tartalmakkal, új pályákkal és bővített világ építéssel gazdagodik. A játék későbbi verzióiban több kihívást és változatos szinteket szeretnénk bevezetni, hogy a játékelmény folyamatosan friss és izgalmas maradjon.

A projekt tervezés szempontjai, alkalmazott módszereinek bemutatása

A fejlesztése során egy ötletbörzét tartottunk, hogy meghatározzuk a játék vizuális stílusát és játékmeneti irányvonalát. Két fő koncepció merült fel. Az első egy letisztult, pasztell színvilágú világ volt, amely nyugodt, minimalista hangulatot árasztott. Ezt azonban elvetettük, mert nem illett a platformer műfaj dinamikusabb játék menetéhez.

Végül egy vibrálóbb, mesésebb megoldás mellett döntöttünk, amely élénk színeket és kontrasztos elemeket használ. A játék háttere világoskék égbolttal és lebegő föld szigetekkel teremt egyedi atmoszférát, míg a karakterek és tárgyak (például az almák) jól elkülönülnek a környezettől. A pálya tervezés során törekedtünk arra, hogy az akadályok ne csak vizuálisan illeszkedjenek a világba, hanem aktív szerepet játszanak a játékmenetben is.

Az irányítás kialakításakor több lehetőséget is teszteltünk, végül egy intuitív vezérlési rendszert hoztunk létre, amely könnyen tanulható, de kellően precíz ahhoz, hogy a platformer kihívások teljesíthetők maradjanak. A játék akadályai és platformjai úgy lettek megtervezve, hogy folyamatosan fenntartsák a játékos figyelmét, miközben a cél, Skipper elérése mindig elérhető távolságban marad.

SWOT elemzés – Mudskip the Slime

A projekt hatékony működésének elemzésére a SWOT analízist alkalmaztuk. Segítségével feltérképeztük a játék erősségeit, gyengeségeit, lehetőségeit és a potenciális fenyegetéseket.

A projekt erősségei:

- **Egyedi mozgás mechanika:** A játék fő mechanikája, az ugrás és a dash, egyedi élményt biztosít a platformerek között.

- **Intuitív irányítás:** Az egyszerű, de mély mechanikák miatt könnyen tanulható, mégis kihívást jelentő játékmenet.
- **Kiegyensúlyozott pályatervezés:** Az akadályok úgy lettek kialakítva, hogy fokozatosan vezessék be az új kihívásokat, így biztosítva a folyamatos fejlődés érzését.

A projekt gyengeségei:

- **Nincs harcrendszer:** A játék nem tartalmaz ellenfeleket, ami egyes játékosok számára csökkentheti az izgalmat.
- **Korlátozott mechanikai variáció:** Bár az alap mechanika egyedi, hosszabb távon szükséges lehet a változatosság növelése, hogy fenntartsa a játékosok érdeklődését
- **Célközönség szűkösége:** A harc nélküli, akadálykerülő platformerek réteg közönséghez szólnak, így nehezebb lehet szélesebb játékosbázist megszólítani.

A projekt lehetőségei:

- **Bővíthető mechanikák:** Új környezeti interakciók (pl. mozgó platformok, speciális felületek) bevezetésével növelhető a változatosság.
- **Speedrun támogatás:** A játék egyedi mozgás mechanikája kiváló alap lehet a speedrun közösség számára, ha megfelelő időmérő rendszert építünk be.
- **Cross-platform elérhetőség:** Ha a játékot több platformra is kiadjuk (PC, konzol, mobil), az jelentősen növelheti a játékosbázist.

A projekt fenyegetései:

- **Fokozott verseny a platformer műfajban:** Sok indie platformer jelenik meg, így nehéz kiemelkedni a tömegből.
- **Játékosok gyors lemorzsolódása:** Ha a játék nem tartalmaz elég változatosságot vagy kihívást, a játékosok hamar elveszíthetik az érdeklődésüket.
- **Technikai akadályok:** A játék fizikai motorjának megfelelő finomhangolása kihívást jelenthet, különösen az egyedi mozgás mechanika miatt.

Ez az elemzés segít abban, hogy jobban megértsük a játékunk helyzetét, és tudatosan tervezzük a fejlesztés következő lépéseit.

Projekt feladatainak megosztása

A projektben résztvevők feladat megosztása

A projekt több rész feladatra osztható fel. Törekedtünk a projekt elkészítése során, hogy a feladatokat egyrészt erősségeink alapján végezzük el, másrészt egymást támogatva, kiegészítve egymás munkáját. Az alábbi táblázat részletesen tartalmazza a projekt tagok nevét, valamint az elvégzett fő feladatokat. A tesztelések és dokumentáció közös feladatunk volt.

Szántó Tamás	Frontend, Tesztelés, Dokumentáció
Nagy Máté Károly	Frontend, Backend, Tesztelés, Dokumentáció
Rózsa Kevin	Frontend, Backend, Tesztelés, Dokumentáció, PPT felelős

Projekt ütemezés és alkalmazott módszereinek bemutatása

1. hét – Projektindítás és alapok

A projekt első hetében létrejön a négy fő komponens: a Unity játék, a Blazor WebAssembly frontend, a MAUI mobilalkalmazás és a C# REST API. A Unity oldalán megvalósul az alap karaktermozgás és a kamera követés. A weboldal Bootstrap alapokra épül, a REST API-ban pedig elindul a felhasználói rendszer kialakítása.

2. hét – Regisztráció és bejelentkezés

A héten elkészül a regisztrációs és bejelentkezési felület minden platformon. A REST API session alapú hitelesítést használ, amit a frontendek és a játék is alkalmaz. Sikeres bejelentkezés után a felhasználói adatok mentésre kerülnek a kliensoldalon.

3. hét – Menü és pályastruktúra

A Unity projektben kialakul a főmenü, a szintválasztó, valamint a külön scene-ekre bontott pályastruktúra (Tutorial + 5 szint). A webes és mobilos frontendeken is megjelenik az alap menürendszer.

4. hét – Pont rendszer

A játékban működésbe lép az alma gyűjtés és a pontszám mentése. A REST API tárolja ezeket az adatokat, amelyeket a weboldal és a mobilapp képes lekérni pályánként, és megjeleníteni a játékos számára.

5. hét – Vélemények kezelése

A rendszer ebben a szakaszban kiegészül egy szöveges véleményküldési lehetőséggel, amelyben a felhasználók magáról a játékról oszthatják meg

tapasztalataikat. A vélemények a REST API-n keresztül kerülnek rögzítésre, de megjelenítésük egyelőre nem történik meg sem a webes, sem a mobilos frontendeken – kizárólag háttéradatként tárolódnak az adatbázisban.

6. hét – Profil és statisztika

A Blazor és MAUI felületeken elérhetővé válik a játékosok saját pontszámainak megtekintése. A bejelentkezett felhasználó csak is saját pontjait is látja. A szükséges adatokat a REST API biztosítja.

7. hét – Ranglista és statisztikák

A héten elkészül a globális ranglista oldal, amely megmutatja a legjobb pontszámokat pályánként. A REST API biztosítja a rendezett adatokat, és a Unity játék végén is megjeleníthető a legjobb eredmény.

8. hét – Design finomítás

A weboldal megjelenése egységesebbé válik, Bootstrap segítségével letisztult, könnyen kezelhető felületet kap. A mobilalkalmazás dizájnya is igazodik a webes megjelenéshez. A Unity játék UI elemei egyszerű, statikus elrendezést használnak.

9. hét – Tesztelés

Funkcionális tesztelés történik minden platformon. Ellenőrizzük a login, highscore mentés, véleményküldés és profiladatok működését. A játék stabil, minden pálya elérhető és működőképes.

10. hét – Hibajavítás és optimalizálás

A héten kijavításra kerülnek a hibák: helytelen válaszok, űrlapvalidációk, session-kezelés problémák. A weboldalon letisztul a felület, gyorsul az adatbetöltés, és csökkennek a hálózati hibák.

11. hét – Teljes összekapcsolás

A Unity játék, a mobilapp és a weboldal teljes összhangban működik a REST API-val.

12. hét – Projektzárás és beadás

Az utolsó héten elkészül a végleges .exe Unity build, a MAUI app tesztelhető formában működik, és a Blazor WebAssembly projekt éles deployra kerül. A teljes dokumentáció és prezentáció összeáll, a projekt leadásra készen áll.

A felhasználói felület részletezése, felhasznált programok, hálózatok

Regisztrációs

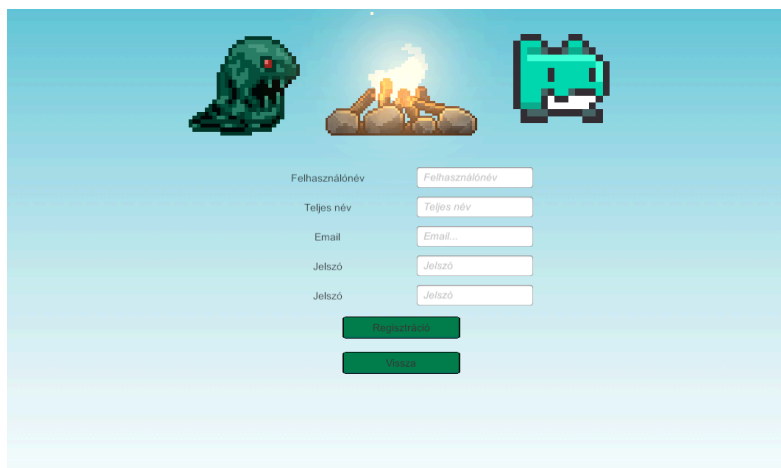
képernyő

A Regisztrációs

képernyő akkor

jelenik meg, amikor a felhasználó a

bejelentkezési képernyőn a "Regisztráció" gombra kattint. Itt a felhasználó új fiókot hozhat létre, és az alábbi adatokat kell megadnia:

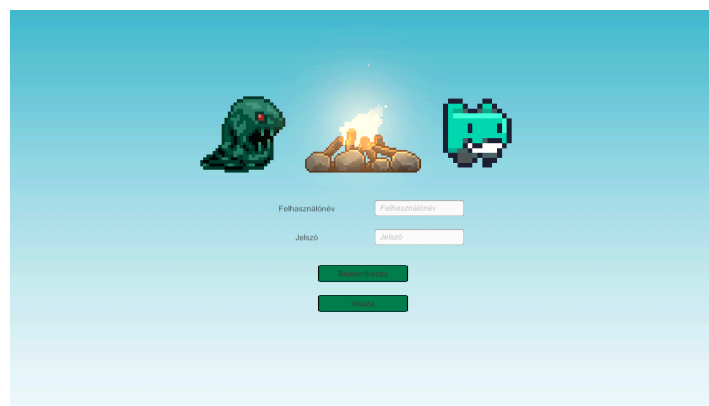


- **Felhasználónév:** Egyedi név, amely azonosítja a felhasználót. Nem tartalmazhat speciális karaktereket vagy szóközöket, és egyedinek kell lennie.
- **Teljes név:** A felhasználó teljes neve, amely opcionális, de segíthet az azonosításban.
- **E-mail cím:** Érvényes e-mail cím, amely szükséges a fiók aktiválásához és a jelszó visszaállításához.
- **Jelszó :** A felhasználó által választott jelszó, legalább 8 karakter hosszúságú, biztonságos jelszó javasolt.
- **Jelszó megerősítése:** A felhasználó által ismételten beírt jelszó a megerősítéshez.

A **Regisztrálás** gombbal hozható létre az új fiók. Ha valamelyik mező nincs megfelelően kitöltve, a rendszer hibaüzenetet ad.

A képernyő alján található a **Vissza** gomb, amely visszavisz a bejelentkezési képernyőre.

Bejelentkezési képernyő

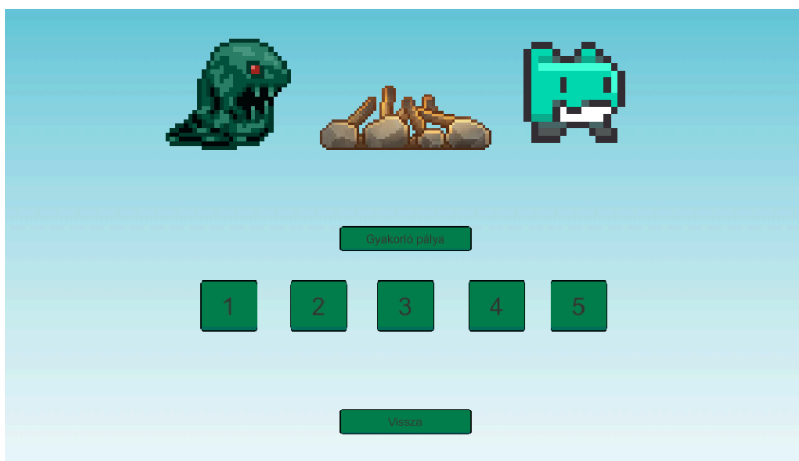


A **Bejelentkezési képernyő** akkor jelenik meg, amikor a felhasználó a játék indításakor vagy a jelentkezési oldalra navigál. A képernyő célja, hogy lehetővé tegye a felhasználó számára, hogy belépjen meglévő fiókjába.

Az alábbi mezők találhatók a jelentkezési képernyőn:

1. **Felhasználónév** : A felhasználó által korábban regisztrált egyedi felhasználóneve, amely azonosítja őt a rendszerben. A felhasználónevet a megfelelő mezőbe kell beírni.
2. **Jelszó** : A felhasználó által választott jelszó, amely a fiók biztonságos hozzáférését biztosítja. A jelszó mezőben a felhasználó titkosítva látja a beírt karaktereket.
3. **Bejelentkezés gomb**: A felhasználó a **Bejelentkezés** gombra kattintva próbál jelentkezni a rendszerbe a megadott felhasználónév és jelszó párosával. Ha a jelentkezési adatok helyesek, a felhasználó hozzáférést nyer a fiókjához, és további funkciókhoz.
4. **Vissza gomb**: A képernyő alján található **Vissza** gomb segítségével a felhasználó visszakerülhet az előző képernyőre (például a regisztrációs oldalra), ha még nem rendelkezik fiókkal, vagy szeretne másik funkcióra navigálni. A vissza gomb nem törli a beírt adatokat, így a felhasználó könnyedén átválthat másik képernyőre.

Pályaválasztó képernyő

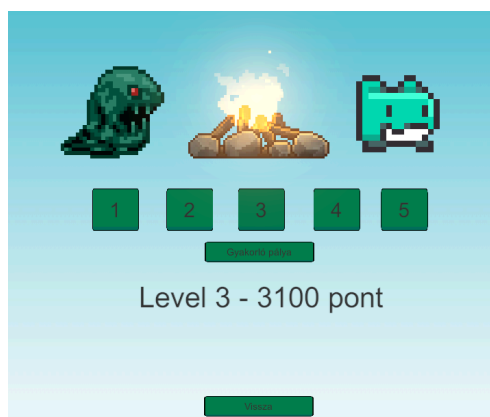


A **Pályaválasztó képernyő** akkor jelenik meg, miután a felhasználó belépett a játékba. Itt a felhasználó választhat a különböző pályák között, és elindíthatja a kívánt szintet. A képernyőn található elemek a következők:

- **Gyakorló pálya gomb:** Ezzel a gombbal a felhasználó elindíthatja a játék bevezető oktatóját, amely segít megismerkedni a játék mechanikájával.
- **1-5 pálya gombok:** Minden egyes gomb egy-egy pályát képvisel, amely a következő számokkal van jelölve. A felhasználó kiválaszthatja, hogy melyik pályát szeretné játszani, és elindíthatja a választott szintet. A pályák számozása a játék sorrendjét tükrözi.
- **Vissza gomb:** A felhasználó visszatérhet a főmenübe a "Vissza" gomb megnyomásával, ha nem kíván pályát választani, vagy inkább más funkciót szeretne elérni a menüben.

Ez a képernyő lehetőséget ad a felhasználónak arra, hogy könnyedén válasszon a különböző pályák között, és elindítsa a játékot a kívánt szinten.

Highscore képernyő



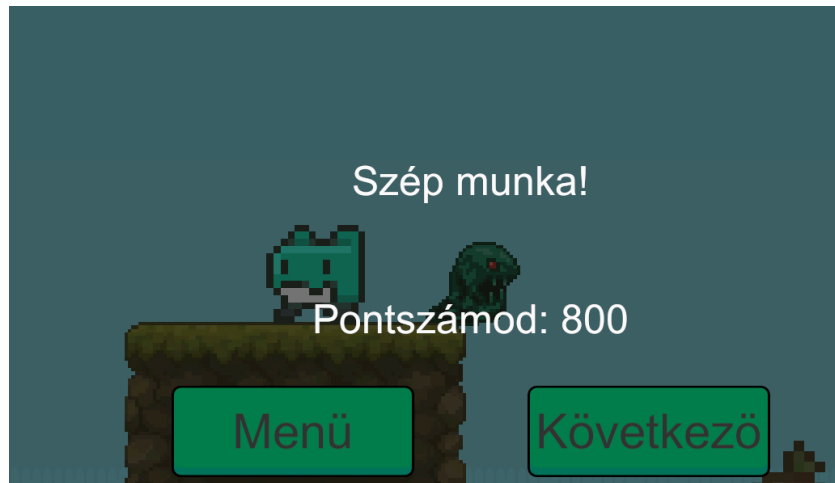
Ez a képernyő lehetőséget ad a bejelentkezett játékosnak arra, hogy megnézze saját legjobb pontszámait minden pályán.

Elérhető elemek:

- 1–5 gombok: A játékos ezekre kattintva lekérheti az adott pályán elért saját pontszámát.
- Gyakorló pálya gomb: A tutorial szinten elért pont jelenik meg.
- Vissza gomb: Visszairányít a ranglista menübe vagy főmenübe.

A képernyő alján mindig megjelenik az aktuálisan kiválasztott pálya és a hozzá tartozó legmagasabb pontszám, pl. Level 3 - 3100 pont. Ez az oldal kizárólag a saját eredmények megtekintésére szolgál – más játékosok pontszámai itt nem jelennek meg.

Szint teljesítési képernyő



A játékos a pálya végére érve a **szint teljesítési képernyőn** (angolul: *Finish Panel*) látja a teljesítés eredményét.

Elérhető elemek:

- **Szép munka!** – Gratuláció a játékosnak a pálya befejezéséért.
- **Pontszámod: [érték]** – Megjeleníti, hány pontot ért el a játékos az adott szinten.
- **Menü gomb** – Vissza irányít a főmenübe vagy a pályaválasztó képernyőre.
- **Következő gomb** – Tovább lép a következő elérhető pályára.

Ez a képernyő összefoglalja a játékos teljesítményét, és lehetővé teszi a továbblépést vagy a kilépést a menübe.

Szünet képernyő



A játék során bármikor elérhető a **szünet menü**, amely lehetőséget ad a játékosnak a játékmenet ideiglenes megszakítására és további lehetőségek közötti választásra.

Elérhető funkciók:

- **Folytatás** – Visszatérés a játékba pontosan onnan, ahol a szüneteltetés megtörtént.
- **Újraindítás** – A jelenlegi pálya újrakezdése az elejétől.
- **Vissza a Menübe** – Visszairányít a főmenübe vagy a pályaválasztó képernyőre.
- **Kilépés a Windowsba** – A játék azonnali bezárása és visszatérés az operációs rendszerhez.

Ez a képernyő segíti a játékosokat a játék megszakításában, újrakezdésében vagy kilépésében, anélkül hogy elveszítenék az irányítást vagy a haladásukat.

Karakter Mozgása és Animációja



A rendszer lehetővé teszi a karakter vízszintes mozgását, ugrását, valamint az interakciót a játék világgal, például almák összegyűjtését.

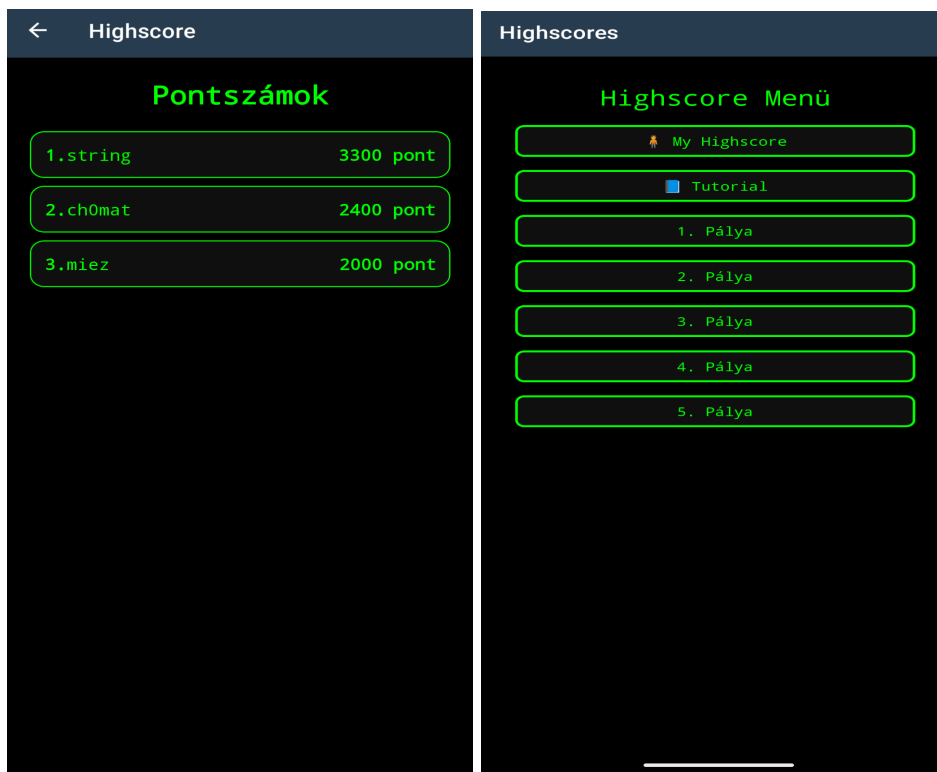
A játékos a megfelelő gombok lenyomásával mozgathatja a karaktert balra és jobbra. A rendszer figyeli a karakter nézetének irányát, és ha szükséges, megfordítja azt. Ha a játékos lenyomja az ugrás gombot és a karakter a földön van, az ugrás végrehajtásra kerül. A rendszer biztosítja, hogy az animációk mindig a karakter állapotának megfelelően jelenjenek meg.

A rendszer folyamatosan figyeli a mozgás állapotát, és ennek megfelelően változtatja az animációkat. Ha a karakter mozog, a séta animáció aktiválódik. Ha a karakter egy almát érint, az alma eltűnik, és a Highscore számlálója növekszik.



Fejlesztési lehetőségek közé tartozik a dupla ugrás, amely lehetőséget adna arra, hogy a játékos még a levegőben is végrehajtson egy ugrást. A sprint funkció pedig lehetővé tenné a mozgási sebesség növelését egy speciális gomb lenyomásával.

Mobil applikáció fejlesztése

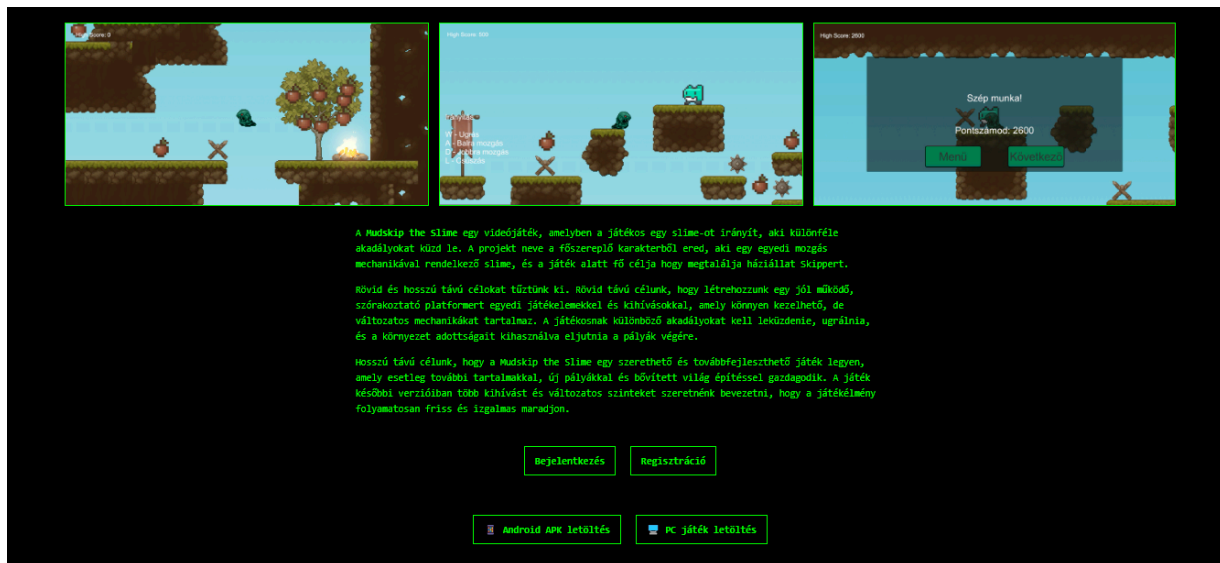


A *Mudskip the Slime* Unity játékhoz egy kiegészítő mobilalkalmazást is készítettünk, amely lehetővé teszi a felhasználók számára, hogy regisztráljanak, bejelentkezzenek, majd megtekintsék az egyes játék szinteken elért pontszámokat. Az alkalmazás felületén a szintek egyszerű, sorszámozott formában jelennek meg – például "Tutorial", "1. pálya", "2. pálya" – egészen az ötödik szintig. A felhasználók ezekhez tartozó gombokra kattintva tekinthetik meg, hogy mennyi pontot értek el az adott szinten.

A fejlesztés a Visual Studio környezetben történt, C# nyelven, a .NET MAUI keretrendszer segítségével. Ez a technológia lehetővé tette egy modern, natív hatású Android-alkalmazás létrehozását, amely könnyen kezelhető és jól illeszkedik a játékhoz.

A felhasználói élmény megtervezésénél a letisztultságra és az egyszerű navigációra törekedtünk. A regisztráció és bejelentkezés folyamata gyors és intuitív, a pontszámok elérése pedig logikusan felépített struktúrán keresztül történik. Az alkalmazás célja, hogy kényelmesen elérhetővé tegye a játékosok számára az elért eredményeket, és méltó kiegészítője legyen a *Mudskip the Slime* játék élményének.

Weboldal fejlesztése



Landing Page – Mudskip the Slime Kezdőoldal

Ez a kezdőoldal (landing page) szolgál belépési pontként a *Mudskip the Slime* játék világába. A célja, hogy röviden bemutassa a játék történetét, hangulatát és lehetőségeit, valamint gyors hozzáférést adjon a legfontosabb funkciókhoz.

Az oldal közepén **négy fő gomb** található, amelyek segítségével a felhasználó elérheti a legfontosabb funkciókat:

Bejelentkezés gomb

- Navigál a *Bejelentkezési oldalra*.

- Itt a korábban regisztrált felhasználók megadhatják felhasználónevüket és jelszavukat.
- Sikeres bejelentkezés után a játékhoz kapcsolódó funkciók (például highscore) elérhetővé válnak.

Regisztráció gomb

- Átvisz a *Regisztrációs oldalra*.
- Itt új felhasználói fiók hozható létre felhasználónév, email és jelszó megadásával.

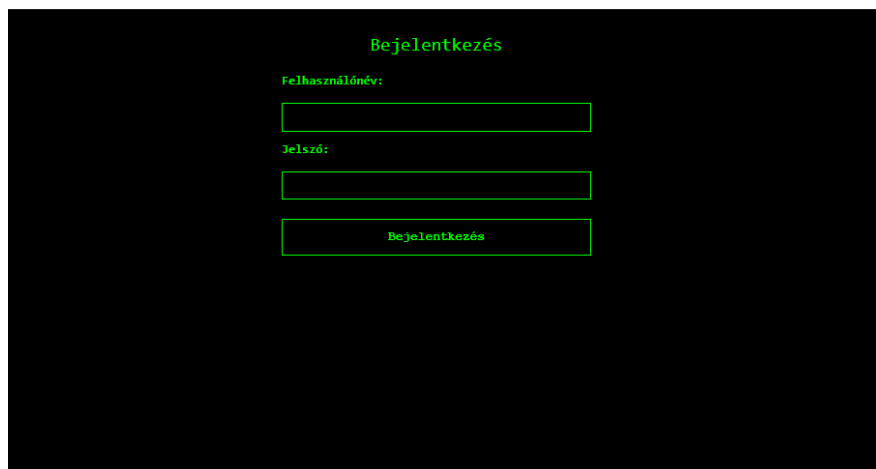
Android APK letöltés gomb

- Letölti a játék Androidos verzióját **.apk** fájlformátumban.
- Ez a fájl telefonra telepíthető, így a játék mobilon is kipróbálható.

PC játék letöltés gomb

- A számítógépes verziót kínálja **.zip** formátumban.
- Letöltés után a játék futtatható Windows rendszer alatt.

Bejelentkezési képernyő



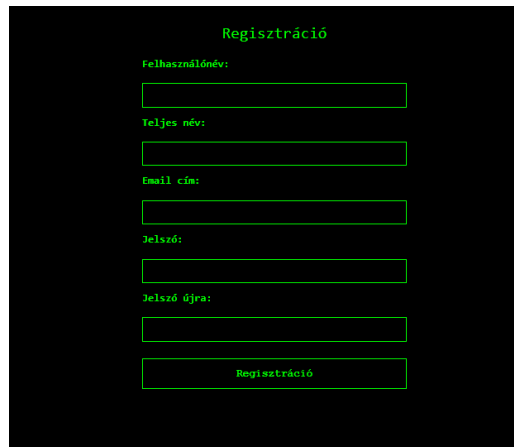
The image shows a login screen with a black background. At the top, the title "Bejelentkezés" is written in red. Below it, the label "Felhasználónév:" is in red, followed by a white rectangular input field. Underneath that, the label "Jelszó:" is in red, followed by another white rectangular input field. At the bottom, there is a white rectangular button with the text "Bejelentkezés" in red.

A bejelentkezési oldal lehetőséget nyújt a regisztrált felhasználók számára, hogy belépjenek fiókjukba. A felhasználónév és jelszó megadása után a **Bejelentkezés** gombbal történik az azonosítás.

Sikeres bejelentkezés esetén a felhasználó a főmenübe kerül. Hibás adatok esetén figyelmeztető üzenet jelenik meg.

A felület célja az egyszerű és gyors hozzáférés biztosítása a játék statisztikáihoz és pályáihoz.

Regisztrációs képernyő



The image shows a registration form titled "Regisztráció" on a dark background. The form contains the following fields and labels:

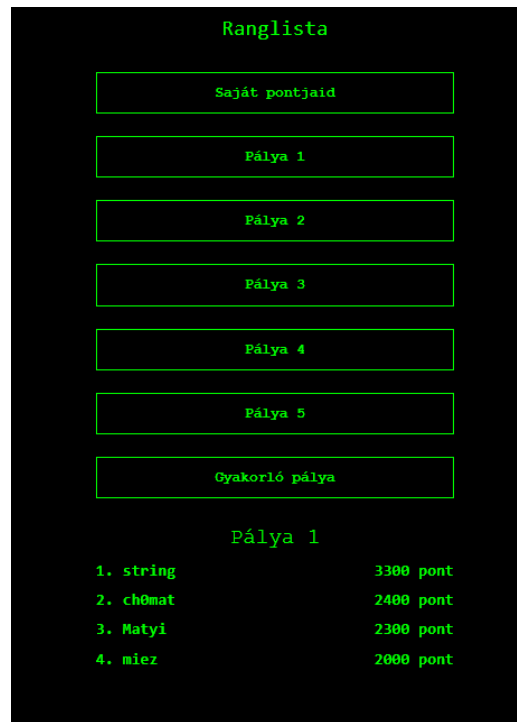
- Felhasználónév: (Username)
- Teljes név: (Full Name)
- Email cím: (Email Address)
- Jelszó: (Password)
- Jelszó újra: (Repeat Password)
- Regisztráció (Register button)

A regisztrációs képernyőn új felhasználók hozhatnak létre fiókot a játék használatához. A felhasználónak a következő mezőket kell kitöltenie:

- **Felhasználónév** – egyedi azonosító, mellyel később bejelentkezik
- **Teljes név** – opcionális mező a felhasználó teljes nevével
- **Email cím** – kötelező, érvényes email cím szükséges
- **Jelszó** – legalább 8 karakter hosszú jelszó
- **Jelszó újra** – megerősítés céljából ismét beírandó

A **Regisztráció** gomb beküldi az adatokat az adatbázisba. Hibás kitöltés esetén figyelmeztető üzenet jelenik meg. Sikeres regisztráció után a felhasználó automatikusan átirányításra kerül a bejelentkezési oldalra.

Highscore Képernyő



A ranglista képernyő célja, hogy a játékosok megtekinthessék saját és más felhasználók pontszámait pályánként.

A képernyő felső részén található gombok:

- **Saját pontjaid** – a bejelentkezett játékos egyéni eredményeit mutatja minden pályán
- **Pálya 1–5** – az adott pályán elért legjobb pontszámokat listázza minden játékosról, csökkenő sorrendben
- **Gyakorló pálya** – a bevezető oktató szint ranglistája

Az aktuálisan kiválasztott pálya neve a ranglista alatt jelenik meg, és alatta sorszámozva, névvel és pontszámmal szerepelnek a legjobbak. Ez a képernyő ösztönzi a versengést és nyomon követhetővé teszi a fejlődést a játékosok számára.

Az Adatbázis és a REST API részletes bemutatása

UserController működése:

User		^
POST	/api/User/register	▼
POST	/api/User/login	▼
PUT	/api/User/update	▼
POST	/api/User/logout	▼
DELETE	/api/User/delete/{id}	▼

Regisztráció

POST /api/User/register

Új felhasználó regisztrációja. Szükséges adatok:

- felhasználónév
- teljes név
- email cím
- jelszó

Sikeres regisztráció esetén az adatok bekerülnek az adatbázisba.

A felhasználók regisztrációjakor a rendszer ellenőrzi, hogy az e-mail cím és a

felhasználónév egyedi-e. Ha valamelyik már létezik, a regisztráció sikertelen. A jelszó titkosítva kerül tárolásra. Sikeres regisztráció esetén a felhasználó adatai mentésre kerülnek, és a rendszer automatikusan rögzíti a **regisztráció dátumát (CreatedAt)**, amely az aktuális UTC időpontot veszi fel.

Bejelentkezés

POST /api/User/login

Bejelentkezés felhasználónév és jelszó megadásával.

Siker esetén visszaigazolást kapunk.

Kijelentkezés

POST /api/User/logout

Kijelentkezés a rendszerből.

Adatok módosítása

PUT /api/User/update

Felhasználói adatok (teljes név, email) frissítése.

Felhasználó törlése

DELETE /api/User/delete/{id}

Felhasználó törlése ID alapján.

A felhasználók regisztrációjakor a rendszer ellenőrzi, hogy az e-mail cím és a felhasználónév egyedi-e. Ha valamelyik már létezik, a regisztráció sikertelen. A jelszó titkosítva kerül tárolásra. Sikeres regisztráció esetén a felhasználó adatai

mentésre kerülnek, és a rendszer automatikusan rögzíti a **regisztráció dátumát (CreatedAt)**, amely az aktuális UTC időpontot veszi fel.

HighscoreController működése:

Highscore		^
GET	/api/Highscore/{dotLevel}	▼
GET	/api/Highscore/my-highscores	▼
POST	/api/Highscore	▼
DELETE	/api/Highscore/{id}	▼
GET	/api/Highscore/by-level	▼

Highscore lekérése szint alapján

GET /api/Highscore/{dotLevel}

A végpont segítségével lekérdezhetjük az adott szinthez tartozó highscore-okat. A dotLevel paraméterben megadjuk, melyik szint eredményeire vagyunk kíváncsiak. A rendszer visszaadja az összes elmentett pontszámot ehhez a szinthez.

Saját highscore-ok lekérése

GET /api/Highscore/my-highscores

Ez a végpont a bejelentkezett felhasználó összes highscore-ját adja vissza. Így a felhasználó megtekintheti a saját eddig elért eredményeit, szintenként.

Új highscore hozzáadása

POST /api/Highscore

Ezen az útvonalon új highscore adható hozzá az adatbázishoz. A kérésben meg kell adni a szint azonosítóját és az elért pontszámot. Sikeres feltöltés esetén az eredmény mentésre kerül.

Highscore törlése

DELETE /api/Highscore/{id}

Ezzel a végponttal egy meglévő highscore törölhető az adatbázisból az ID alapján. A művelet véglegesen eltávolítja az adott rekordot.

GET /api/Highscore/by-level

Ez a végpont lehetővé teszi, hogy lekérdezd a saját pontszámodat egy megadott szinthez. A kérés során meg kell adni a szint azonosítóját (levelId), és a rendszer visszaadja, hogy mennyi pontot értél el azon a pályán. Ha még nincs pontod az adott szinten, 0 vagy üres válasz érkezik vissza.

ReviewController működése:

Review		^
POST	/api/Review	▼
GET	/api/Review/all	▼
DELETE	/api/Review/{reviewId}	▼

Vélemény beküldése

POST /api/Review

Ezzel a végponttal egy új véleményt lehet rögzíteni az adatbázisban. A kérésben meg kell adni az értékeléshez tartozó adatokat:

- felhasználó azonosító
- értékelés (szám)
- szöveges megjegyzés

Sikeres beküldés esetén az értékelés mentésre kerül, és később megtekinthető.

Összes vélemény lekérése

GET /api/Review/all

Ez a végpont az adatbázisban található összes véleményt visszaadja.

Segítségével minden eddig beküldött értékelés megtekinthető.

Vélemény törlése

DELETE /api/Review/{reviewId}

A megadott reviewId alapján törli az adott véleményt az adatbázisból. A művelet véglegesen eltávolítja a kiválasztott értékelést.

LevelController működése:

Level		^
GET	/api/Level	▼
POST	/api/Level	▼
GET	/api/Level/{id}	▼
PUT	/api/Level/{id}	▼
DELETE	/api/Level/{id}	▼

Összes szint lekérése

GET /api/Level

Ez a végpont az adatbázisban szereplő összes szint (Level) adatait adja vissza.

Segítségével megtekinthetők a rendszerben elérhető pályák.

Új szint hozzáadása

POST /api/Level

Új szint hozzáadására szolgáló végpont. A kérdésben meg kell adni a szint nevét.

Sikeres hozzáadás esetén az új pálya bekerül az adatbázisba, és a továbbiakban elérhető lesz.

Adott szint lekérése azonosító alapján

GET /api/Level/{id}

Ezzel a végponttal egy konkrét szint adatai kérhetők le az adatbázisból az azonosító (id) megadásával.

Szint módosítása

PUT /api/Level/{id}

Ez a végpont lehetőséget ad egy már meglévő szint adatainak módosítására. A PUT kérésben meg kell adni a módosított adatokat, például a szint új nevét.

Szint törlése

DELETE /api/Level/{id}

A megadott id alapján töröl egy szintet az adatbázisból. A törlés véglegesen eltávolítja az adott pályát a rendszerből.

LevelStatsController működése:

LevelStats		^
GET	/LevelStats	▼
PUT	/LevelStats/{levelId}	▼
DELETE	/LevelStats/{levelId}	▼

Összes szint statisztika lekérése

GET /LevelStats

Ez a végpont az adatbázisban szereplő összes szint statisztikai adatait adja vissza.

Segítségével megtekinthető, hogy egyes pályákat hányszor teljesítettek a játékosok.

Szint statisztika frissítése

PUT /LevelStats/{levelId}

Ez a végpont lehetőséget ad egy adott szint statisztikáinak frissítésére. A kérdésben meg kell adni a módosított adatokat, például a teljesítések számát.

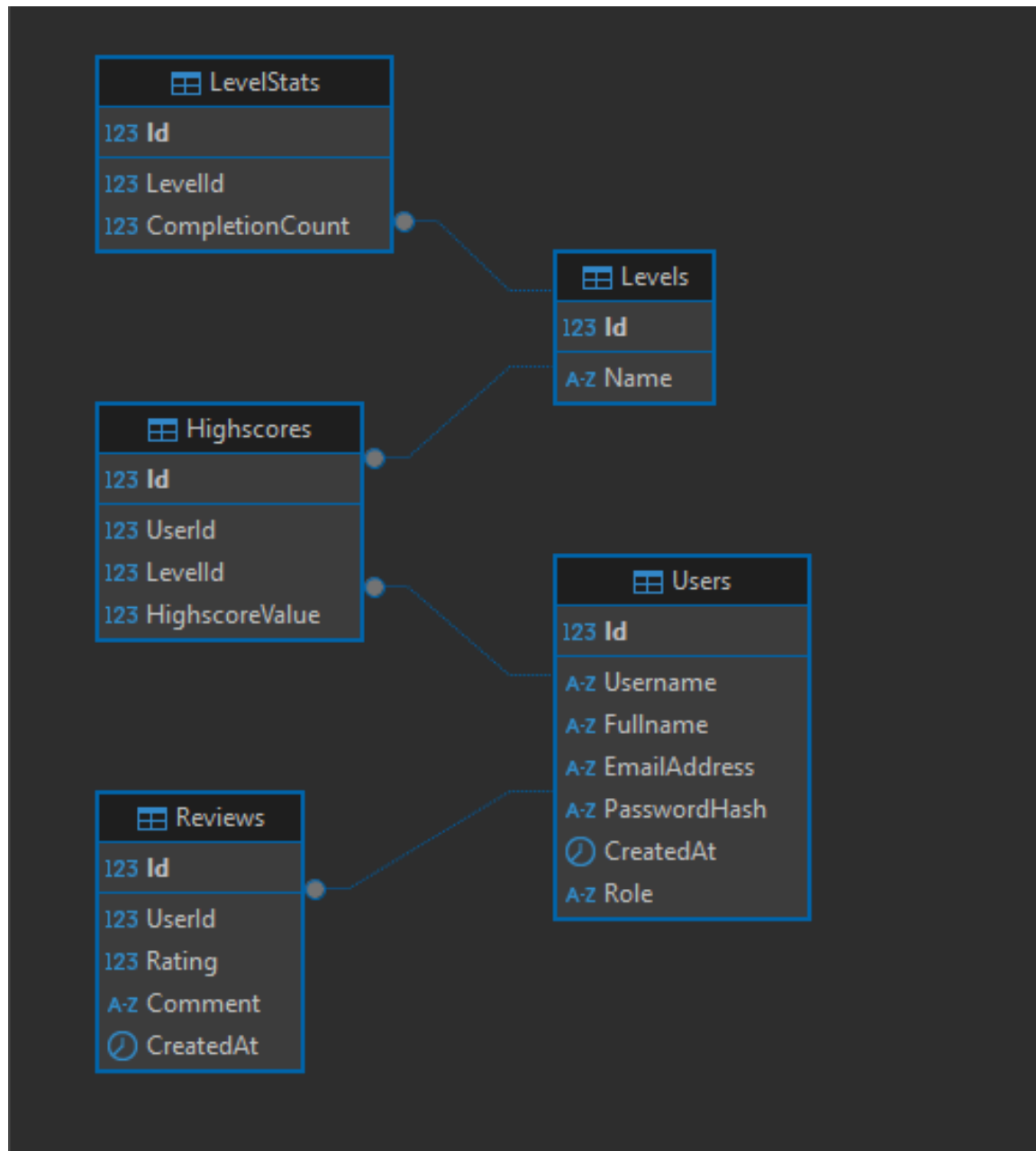
Szint statisztika törlése

DELETE /LevelStats/{levelId}

Ezzel a végponttal egy adott szint statisztikai adatai törölhetők az adatbázisból az azonosító (**levelId**) megadásával.

Az adatbázis felépítése

A projekthez használt adatbázis 4 főbb entitásból épül fel: **User**, **Level**, **Highscore** és **Reviews**. Az adatbázisban szereplő entitások kapcsolata az alábbi képen látható:



Eszközök

Frontend

A projekt webes felülete a **Visual Studio** fejlesztőkörnyezetben készült, amelyet a .NET támogatása, kényelmes fejlesztői eszközei és integrált hibakeresési lehetőségei miatt választottunk. A frontend fejlesztéshez **Blazor WebAssembly** technológiát használtunk, amely lehetővé tette, hogy modern, komponensalapú, reszponzív weboldalt készítsünk főként C# nyelvben.

A rendszer működéséhez szükséges aszinkron kommunikációhoz egy külön **JavaScript alapú fetchHelper.js fájlt** is készítettünk, amely a REST API végpontok elérését egyszerűsíti, és biztosítja a felhasználói hitelesítéshez szükséges session cookie-k kezelését.

A weboldalt a **Netlify** platformon publikáltuk, mivel ez könnyű telepítést, gyors frissítést és megbízható hosztolást biztosít dinamikus oldalak esetén is. Bár alternatívaként felmerült a GitHub Pages, az nem támogatta volna a dinamikus backend kommunikációt, így a Netlify bizonyult ideálisnak.

A projekt részeként egy **.NET MAUI** alapú **mobilalkalmazást** is készítettünk, amely lehetővé teszi a felhasználók számára, hogy Android eszközökön is használják a rendszer fő funkcióit, például a regisztrációt, bejelentkezést és highscore megtekintést.

Backend

A backend oldalt Visual Studio 2022 segítségével fejlesztettük, C# nyelven, az ASP.NET Core Web API keretrendszer használatával. Ezt a technológiát azért

választottuk, mert rendkívül hatékony, skálázható és kis memóriaigényű megoldást kínál. Az API lehetővé teszi a frontend és más külső rendszerek közti strukturált adatkommunikációt.

Az adatbázis kezeléshez az Entity Framework Core technológiát vettük igénybe, amely objektum-orientált módon képes kezelni az adatokat és segíti az adatbázis migrációk kezelését. Az adatbázisunkat Aiven platformra helyeztük, mely felhő alapú, skálázható és megbízható MySQL-kiszolgálót biztosít. A REST API-t a Render szolgáltatásra telepítettük, amely biztosítja a külső elérhetőséget és az automatikus deploymentet.

A kezdeti próbálkozások során a tesztelést az éles adatbázison végeztük, azonban ez adatvesztéshez vezetett, ezért áttértünk a teszt adatbázisok használatára (InMemoryDatabase). Ez biztosítja, hogy a tesztelés ne befolyásolja az éles adatokat.

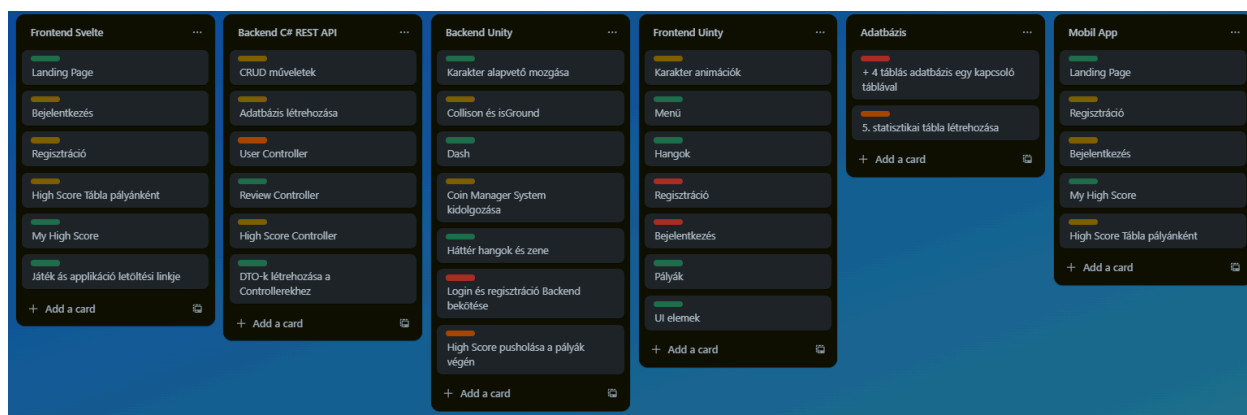
A backend egységteszteléséhez az xUnit keretrendszert alkalmaztuk, ahol részletes naplózással (Console.WriteLine) és külön txt logfájlokba írással követtük a teszt lépéseket. A session alapú hitelesítés teszteléséhez egy saját FakeSession osztályt fejlesztettünk, amely lehetővé tette a bejelentkezés és highscore mentés teljes szimulációját tesztkörnyezetben.

Unity

A projekthez egy különálló 2D platformer játék is tartozik, melyet **Unity** játék motorral és **C#** nyelven készítettünk. A játék REST API-n keresztül kommunikál a backenddel, például regisztráció, bejelentkezés és pontszámok mentése esetén. A session-alapú hitelesítés biztosítja, hogy a játékosok személyre szabott élményben részesüljenek. A játék és a backend közötti kommunikációt JSON formátumú HTTP kérésekkel valósítottuk meg, biztonságos és megbízható adatkezelés mellett.

Kommunikáció és Menedzsment

A csapaton belüli kommunikációhoz Discordot használtunk, amely egyszerű, gyors és megbízható módot biztosított a valós idejű egyeztetésre. A projekt feladatait és azok állapotát Trello táblákon keresztül kezeltük. Ez lehetővé tette az átlátható munkamegosztást, a határidők követését, és a feladatok prioritizálását, különösen az agilis szemlélet mentén.



A projekt során a csapat tagjai Discordon keresztül tartották a kapcsolatot. A platform lehetővé tette a gyors üzenetváltást, hanghívást és fájlmegosztást, ami megkönnyítette az együttműködést.

A legnagyobb kihívást a csapattagok eltérő személyisége és szakmai tapasztalata jelentette. Egyesek önállóan és gyorsan dolgoztak, míg mások több támogatást igényeltek, ami időnként félreértésekhez és lassabb haladáshoz vezetett.

A rendszeres egyeztetések és a türelmes kommunikáció segített abban, hogy jobban megértsük egymást, és fokozatosan kialakult egy működő csapatinamika.

A projekt során nemcsak szakmailag fejlődtünk, hanem csapatmunkában és konfliktuskezelésben is értékes tapasztalatokat szereztünk.

Tesztelés

A **REST API** működésének ellenőrzését manuális teszteléssel végeztük, hogy megbizonyosodjunk arról, hogy minden funkció megfelelően és az elvárt módon működik.

```
1 reference
public void UpdateCheckpoint(Vector2 pos)
{
    checkpointPos = pos;
    Debug.Log("☑ Checkpoint frissítve: " + checkpointPos);
}

1 reference
void Die()
{
    Debug.Log("☠ Meghaltál, respawn indul...");
    StartCoroutine(Respawn(0.5f));
}
```

Ennek a segítségével ellenőriztem, hogy a respawn funkció megfelelően működik-e. Ilyen és ehhez hasonló Debug.Log részekkel a kódunk tele van, így megtudtam nézni, hogy minden funkció tényleges úgy működik ahogy annak kell.

REST API tesztelése

A projekt során automatizált egységteszteket készítettünk a MudskipDB nevű ASP.NET Core backendhez. A tesztelés célja az volt, hogy megbizonyosodjunk arról, hogy a felhasználói regisztráció, bejelentkezés és a highscore mentés helyesen működik. A teszteket a xUnit test keretrendszerrel írtuk meg, és különálló teszt projekten belül futtattuk le, teljesen izolált környezetben.

Az első tesztelési próbálkozás során közvetlenül az éles adatbázisra irányítottuk a DbContext-et, viszont ez súlyos következményekkel járt: a tesztek lefutása után a

Users, Highscores és Levels táblákból az adatok teljesen törlődtek, mivel a tesztlogika részeként adatokat töröltünk és újra létrehoztunk. Ezért a továbbiakban úgy döntöttünk, hogy az összes teszt eset során **InMemory adatbázist** fogunk használni, amely minden teszt futtatásakor új, ideiglenes adatbázist hoz létre.

A tesztelés során használtuk a FakeSession nevű segéd osztályt, amely egy egyszerű implementációja az ISession interfésznek. Ez lehetővé tette, hogy a tesztek során szimuláljuk a session alapú hitelesítést, így valósághű körülmények között alkalmunk volt tesztelni a sessionre épülő végpontokat is (mint például a bejelentkezés és a highscore mentés).

Két fő teszt esetet készítettünk el részletesen: az első a UserCanLoginSuccessfully, amely egy meglévő felhasználó bejelentkezését és a session beállítását ellenőrzi. A második a PostHighscores_ForAllLevels_Works, amely regisztrál egy új felhasználót, bejelentkezik vele, majd minden pályára (Tutorial, Level 1-5) beküld egy highscore-t. Minden művelet után Console.WriteLine() hívásokat használtunk a részletes naplózáshoz, és minden teszt eset létrehoz egy külön .txt fájlt is, amely tartalmazza a teszt lépéseket és azok eredményeit. Ez lehetővé teszi, hogy a teszt folyamat nem csak géppel, hanem ember által is könnyen nyomon követhető legyen.

Összességében a tesztelési megoldások biztosítják, hogy a rendszer logikája megbízhatóan működjön, anélkül, hogy az éles adatokat kockáztatnánk. A szimulált környezet lehetőséget adott arra, hogy minden edge case-t lefedjünk, és megbizonyosodjunk arról, hogy a session, az adatbázis és az API réteg összhangban működik.

```
Teszt futás: UserCanLoginSuccessfully
16:40:00 - 🗄 Tesztfelhasználó létrehozása az adatbázisban...
16:40:00 - ✅ Felhasználó mentve: loginuser
16:40:00 - 🔑 Bejelentkezés indítása...
16:40:00 - ✅ Bejelentkezés sikeres.
16:40:00 - 🚀 Session beállítva UserID: 1

Teszt futás: PostHighscores_ForAllLevels_Works
16:40:00 - 🗄 Pályák létrehozása adatbázisban...
16:40:00 - ✅ Pályák mentve az adatbázisba.
16:40:00 - 👤 Regisztráció folyamatban...
16:40:00 - ✅ Regisztráció sikeres.
16:40:00 - 🔑 Bejelentkezés folyamatban...
16:40:00 - ✅ Bejelentkezés sikeres.
16:40:00 - 🚀 Bejelentkezett User ID: 1
16:40:00 - 📁 Highscore-ok beküldése folyamatban...
16:40:00 - ✅ Highscore mentve: Tutorial - 200 pont
16:40:00 - ✅ Highscore mentve: Level 1 - 200 pont
16:40:00 - ✅ Highscore mentve: Level 2 - 200 pont
16:40:00 - ✅ Highscore mentve: Level 3 - 200 pont
16:40:00 - ✅ Highscore mentve: Level 4 - 200 pont
16:40:00 - ✅ Highscore mentve: Level 5 - 200 pont
16:40:00 - 📊 Összes highscore rendben elmentve: 6
```

Project későbbi fejlesztése

Mivel ez eredetileg egy iskolai projektmunkaként indult, kezdetben nem gondoltunk arra, hogy saját Asset Packot kellene készíteni vagy csináltatni a játékhoz. Emiatt a fejlesztés korai szakaszában kész, ingyenesen elérhető vagy megvásárolható erőforrásokat használtunk a játék vizuális és hang hatásainak megvalósítására.

Időközben azonban annyira megszerettük ezt a projektet, hogy komolyan fontolóra vettük annak további fejlesztését és esetleges publikálását a Steam platformon. Ahhoz, hogy a játék egyedivé és vizuálisan is karakteresebbé váljon, a későbbiekben szeretnénk egy teljesen saját Asset Packot létrehozni, amely jobban illeszkedik az általunk elképzelt világba. Ez nemcsak az esztétikai élményt növelné, hanem lehetőséget adna arra is, hogy egy igazán egyedi látvány világot és atmoszférát teremtsünk.

A további fejlesztések terén több ötletünk is van. Szeretnénk, ha a játékos a későbbiekben további képességeket szerezhethetne, amelyeket különböző „power-up” rendszeren keresztül lehetne aktiválni. Ezek a képességek dinamikusabbá és izgalmasabbá tennék a játékmenetet, miközben új stratégiai lehetőségeket biztosítanak a játékosok számára.

A játék publikálását illetően két lehetőséget is mérlegelünk. Az egyik opció az, hogy a játékot „Early Access” formában tesszük közzé a Steamen, így a játékosok visszajelzései alapján folyamatosan fejleszthetnénk és finomhangolhatnánk a tartalmat. Ebben az esetben szükség lenne egy részletes **roadmap** elkészítésére, amely bemutatja a játék jövőbeli frissítéseit és a tervezett fejlesztési ütemtervet.

A másik lehetőség egy **Kickstarter kampány** indítása, amely segítségével finanszírozni tudnánk a további fejlesztéseket. Ebben az esetben egy részletes támogatói rendszer kidolgozására lenne szükség, amelyben különböző célelérésekhez kötnénk az egyes fejlesztési mérföldköveket. Ezzel a módszerrel a közösség támogatását és érdeklődését is felmérhetnénk, miközben anyagi forrást is biztosíthatnak a projekt sikeres megvalósításához.

A weboldalunk fejlesztése során számos olyan végpontot is létrehoztunk, amelyeket eddig még nem integráltunk a felhasználói felületbe. Ilyen például a vélemények (review rendszer) kezelése, amely lehetővé tenné, hogy a felhasználók értékeléseket és visszajelzéseket írjanak az egyes pályákról vagy termékekről. Ez nemcsak a közösség aktivitását növelné, hanem segítene abban is, hogy a fejlesztőcsapat pontosabb képet kapjon arról, mely tartalmak tetszenek leginkább a játékosoknak.

További fontos funkció a statisztikák megjelenítése, amely nyomon követi, hogy egy adott pályát hányszor teljesítettek a játékosok. Ezek az adatok elemzése segítene

abban, hogy felmérjük, mely típusú pályák népszerűek, és melyek kevésbé. Ennek alapján célzottabb és a közösség igényeihez jobban igazodó új pályák fejlesztését tudnánk megvalósítani.

Összességében ezeknek a rendszereknek az integrálása nemcsak a felhasználói élményt emelné magasabb szintre, hanem hosszú távon is hozzájárulna a platform sikerességéhez azáltal, hogy tudatosan, adatalapú módon irányítanánk a fejlesztéseket.

Összegzés

Csapatunk kiegyensúlyozottan dolgozott a projekt tervezésén és megvalósításán, folyamatosan ügyelve arra, hogy a projekt életciklusának megfelelő ütemben és a határidőket betartva haladjunk. A kezdeti szakaszban jelentős támogatást kaptunk a tanároktól, ami nagyban hozzájárult a sikeres induláshoz. A tervezési és végrehajtási fázisban Rédei Tanár Úr szakmai iránymutatását követtük, ami segített abban, hogy a projektet magas színvonalon tudjuk kidolgozni.

Kiemelt figyelmet fordítunk az ellenőrzésre, hogy a zárási szakaszt a lehető legjobb minőségben tudjuk befejezni. Összességében egy jól kidolgozott és stabil projektet hoztunk létre, amely továbbfejleszthető, és megfelelő költségtervezéssel akár a valós piacon is megállja a helyét. Különösen hasznos volt, hogy az elméletben megszerzett tudást egy valós projekt során alkalmazhatjuk, ami jól mutatja, hogy szakmai ismereteink folyamatosan fejlődtek és egyre magasabb szintre jutottak.

Források

ChatGPT (OpenAI) – <https://chat.openai.com>

YouTube – <https://www.youtube.com>

Bootstrap – <https://getbootstrap.com>

Netlify – <https://www.netlify.com>

Unity Documentation – <https://docs.unity3d.com/Manual/2D.html>

xUnit – <https://xunit.net>

C# Documentation – <https://learn.microsoft.com/en-us/dotnet/csharp>

Unity Asset Store – <https://assetstore.unity.com>

Aiven – <https://aiven.io>

Render – <https://render.com>