SZAKDOLGOZAT

|  |
| --- |
| Váci Szakképzési Centrum  Boronkay György Műszaki Technikuma és Gimnáziuma |
| **VIZSGAREMEK**  Arcalite |
| Konzulens: Készítők:  Gyombolainé Cserny Zsuzsanna Bakó Borka  Bali István Gábor  Katona Bálint  Vác  2025 |

Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra. Szakdolgozatunk a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium szoftverfejlesztő és -tesztelő technikus képzésén készítettük. Tudomásul vesszük, hogy szakdolgozatunkat a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium tárolja.

Bakó Borka Bali István Gábor Katona Bálint

Konzultációs lap

Vizsgázók neve:

* Bakó Borka
* Bali István Gábor
* Katona Bálint

Szakdolgozat címe: **Arcalite**

Program által nyújtott szolgáltatások:

* Platformer játék
* A játékbeli haladás nyomonkövetése a weboldalon
* Tudástár a játékban lévő ellenfelekhez és tárgyakhoz
* E-mailek (visszaigazoló, jelszó módosítás)

|  |  |  |
| --- | --- | --- |
| **Sorszám** | **A konzultáció időpontja** | **Konzulens aláírása** |
| 1. | 2024. október 11. |  |
| 2. | 2024. november 15. |  |
| 3. | 2024. december 13. |  |
| 4. | 2025. január 17. |  |
| 5. | 2025. február 14. |  |
| 6. | 2025. március 14. |  |

A szakdolgozat beadható: A szakdolgozatot átvettem:

Vác, 2025. Vác, 2025.

Konzulens A szakképzést folytató

intézmény felelőse

Tartalomjegyzék

[Hallgatói nyilatkozat 3](#_Toc196876882)

[Konzultációs lap 4](#_Toc196876883)

[Tartalomjegyzék 5](#_Toc196876884)

[1. Témaválasztás 7](#_Toc196876885)

[2. Fejlesztői dokumentáció 8](#_Toc196876886)

[2.1. Fejlesztői környezet 8](#_Toc196876887)

[2.1.1. Visual Studio Code 8](#_Toc196876888)

[2.1.2. XAMPP 8](#_Toc196876889)

[2.1.3. Visual Studio 8](#_Toc196876890)

[2.1.4. GODOT 8](#_Toc196876891)

[2.2. Használt technológiák, nyelvek 9](#_Toc196876892)

[2.2.1. HTML 9](#_Toc196876893)

[2.2.2. CSS 9](#_Toc196876894)

[2.2.3. JavaScript 9](#_Toc196876895)

[2.2.4. jQuery 9](#_Toc196876896)

[2.2.5. JSON 9](#_Toc196876897)

[2.2.6. Bootstrap 10](#_Toc196876898)

[2.2.7. PHP 10](#_Toc196876899)

[2.2.8. HTTP, REST API 10](#_Toc196876900)

[2.2.9. MySQL 10](#_Toc196876901)

[2.2.10. C# 11](#_Toc196876902)

[2.2.11. Git, GitHub 11](#_Toc196876903)

[2.2.12. ChatGPT 11](#_Toc196876904)

[2.3. Adatszerkezet 12](#_Toc196876905)

[2.3.1. Alapadatok 12](#_Toc196876906)

[2.3.2. Kapcsolati ábra 12](#_Toc196876907)

[2.3.3. Táblák 13](#_Toc196876908)

[2.4. A weboldal felépítése 21](#_Toc196876909)

[2.4.1. A weboldal szerkezete és formázás 21](#_Toc196876910)

[2.4.2. Saját felugróablakok 21](#_Toc196876911)

[2.4.3. Az API és a weboldal közti kommunikáció 22](#_Toc196876912)

[2.4.4. Egyéb funkcionalitás a weboldalon 22](#_Toc196876913)

[2.5. Az API felépítése 23](#_Toc196876914)

[2.5.1. A cél 23](#_Toc196876915)

[2.5.2. Hibakezelés és közös metódusok 23](#_Toc196876916)

[2.5.3. A végpontok 24](#_Toc196876917)

[2.6. Az alkalmazás felépítése 35](#_Toc196876918)

[2.6.1. MainNode.cs 35](#_Toc196876919)

[2.6.2. A Game mappa 35](#_Toc196876920)

[2.6.3. A Game\Enemies mappa 36](#_Toc196876921)

[2.6.4. A Game\Enemies\Bosses mappa 36](#_Toc196876922)

[2.6.5. A Game\Enemies\Projectiles mappa 36](#_Toc196876923)

[2.6.6. A Globals mappa 36](#_Toc196876924)

[2.6.7. A Maps mappa 36](#_Toc196876925)

[2.6.8. A Mechanics mappa 36](#_Toc196876926)

[2.6.9. A Menus mappa 36](#_Toc196876927)

[2.7. Tesztelés 37](#_Toc196876928)

[2.7.1. Backend tesztelés 37](#_Toc196876929)

[2.7.2. Frontend tesztelés 42](#_Toc196876930)

[2.7.3. A játék tesztelése 42](#_Toc196876931)

[2.8. Továbbfejlesztési lehetőségek 43](#_Toc196876932)

[3. Felhasználói dokumentáció 44](#_Toc196876933)

[3.1. A program rövid ismertetése 44](#_Toc196876934)

[3.2. A program feltelepítése 45](#_Toc196876935)

[3.2.1. A játék telepítése 45](#_Toc196876936)

[3.2.2. A weboldal üzembe helyezése 45](#_Toc196876937)

[3.2.3. Az adatbázis felállítása 45](#_Toc196876938)

[3.3. Használati útmutató 46](#_Toc196876939)

[3.3.1. A játék irányítása 46](#_Toc196876940)

[3.3.2. A weboldal használata 46](#_Toc196876941)

[4. Mellékletek 47](#_Toc196876942)

# Témaválasztás

A feladatunk egy asztali alkalmazás, egy weboldal, és egy ezekkel kommunikáló adatbázis elkészítése volt.

Az asztali alkalmazás témájához sok különféle ötletünk támadt, de egyikük sem ragadta meg az érdeklődésünk. Ám egyszer felmerült egy kérdés: „Lehet-e játékot készíteni az asztali alkalmazásunknak?”. Helyeslő választ kapván már el is dőlt, hogy egy játék lesz a projektünk középpontjában. Ezt a döntést nagyban elősegítette, hogy a csapat tagjai is érdekeltek a játékfejlesztésben, hármunk közül ketten is ebben a szakirányban folytatják tanulmányaikat.

A játék jellege és témája volt a következő kérdés. Az általunk ismert játékok jellegét, kinézetét és játékmódját kezdtük el böngészni ötletekért, mérlegelve, hogy mit lennénk képesek határidőre, jó minőségben elkészíteni. Választásunk a játék jellegét illetően a „platformer” stílusra esett. Ilyen jellegű játékokban tipikusan különféle „platformokon” ugrálva kell haladnunk, közben ellenségekkel harcolnunk stb. Kinézetre a kétdimenziós, oldalnézetes stílust választottuk, mivel ezzel már eleinte is rengeteg ötletünk támadt a játék megjelenését illetően. Játékunk főszereplőjeként egy mágust választottunk, akinek a különböző pályákon átkelve, szörnyeket legyőzve kell utat találnia a végső céljához. Ez a cél egyelőre titok, a játékot kijátszva viszont megtudhatjuk!

Már csak a weboldal kérdése maradt hátra. Átgondolva, hogy egy számítógépes játékhoz milyen weboldal illik, két célt adtunk az oldalnak: egy tudástár biztosítása a játékról, illetve felhasználói fiókunk kezelése. A tudástár egy egyedibb megoldást kapott: csak arról tudhatunk meg információkat, amivel már találkoztunk. Épp ezért, illetve játékunk akár más eszközről való folytatása érdekében gondoltuk ideillőnek a felhasználói fiókok implementálását. Az oldalon tudunk regisztrálni, bejelentkezni, kezelni a fiókunk, és a játékba bejelentkezve képesek vagyunk folytatni akár máshol elkezdett vagy folytatott játékunkat is.

# Fejlesztői dokumentáció

## Fejlesztői környezet

### Visual Studio Code

A Visual Studio Code (VSC) egy nyílt forráskódú kódszerkesztő alkalmazás. Nagy mértékben testre szabható, illetve rengeteg kiegészítő tartalom készült hozzá, melyekkel könnyen fejleszthetünk a legtöbb programozási nyelven. Mi a weboldal HTML vázához, formázásához, a funkcionalitást biztosító JavaScript szkriptekhez, illetve az adatbázissal kommunikáló PHP RestAPI fejlesztéséhez használjuk.

### XAMPP

A XAMPP egy nyílt forráskódú és platformfüggetlen webszerver-szoftver csomag. A csomag fő elemei – amelyek miatt mi is használjuk – az Apache szerver, PHP támogatás, a MariaDB adatbázis-motor, illetve a phpMyAdmin adatbázis-kezelőfelület.

### Visual Studio

A Visual Studio egy fejlesztői kódszerkesztő, amit kód szerkesztésére, debug-olására, de akár alkalmazásunk publikálására is használhatunk. Az IDE a standard kódszerkesztésen és debug-oláson felül tartalmaz fordítókat, kódkiegészítő, grafikus tervező, és még sok egyéb hasznos eszközzel, melyek elősegítik a fejlesztési élményünket. Ebben a projektben a játék kódjának megírásához használtuk.

### GODOT

A Godot egy játékfejlesztő motor, amellyel egy egységes felületen készíthetünk akár két‑, akár háromdimenziós platformfüggetlen játékokat. A motor egy mindent átfogó eszközcsomagot kínál, hogy a fejlesztők a játék fejlesztésére fókuszálhassanak, és ne kelljen már megoldott problémákat újból megoldani. A program teljesen ingyenes és nyílt forráskódú, többnyire a közösség által fejlesztett.

## Használt technológiák, nyelvek

### HTML

A HTML (HyperText Markup Language) egy struktúrált leírónyelv, a weboldalak szerkezetének kialakításához használt szabvány. Általában együtt használatos a CSS és JavaScript nyelvekkel, mely három a webfejlesztés alapjaként szolgál. Míg a HTML a struktúra kialakítására használják, a CSS-t a formázásra, a JavaScript-et pedig a funkcionalitás kialakítására használják.

### CSS

A CSS (Cascading Style Sheets) egy stíluslapnyelv, mellyel a weboldalak megjelenítését és formázását alakíthatjuk ki. Lehetővé teszi, hogy a fejlesztők a formázást és a struktúrát elválasszák egymástól, továbbá, hogy a weboldalak különféle környezetekben is egységesen jelenjenek meg.

### JavaScript

A JavaScript egy dinamikus szkriptnyelv, amelyet általában weboldalak fejlesztéséhez használnak. Az egyik legelterjedtebb és legjelentősebb programozási nyelv a webfejlesztésben. Lehetővé teszi, hogy a weboldal interaktív legyen és dinamikusan reagáljon a felhasználó bemenetére. Emellett teljes értékű alkalmazásokat, játékokat is lehet benne készíteni, ilyenkor a weboldal egy megjelenítő szerepet vet fel. Mi a JavaScriptet a weboldal interaktív részének kialakításán felül az adatok API-ból történő lekérésére és betöltésére használjuk.

### jQuery

A jQuery egy könnyűsúlyú, gyors és keresztplatformos JavaScript könyvtár, amelyet főleg webfejlesztés során használnak. Célja, hogy egyszerűbbé és hatékonyabbá tegye a JavaScript alapú webfejlesztést. Népszerűsége és kényelme miatt sok webfejlesztő használja azáltalános feladatok megkönnyítésére és az alkalmazások gyors fejlesztésére.

### JSON

A JSON (JavaScript Object Notation) egy könnyen olvasható adatcsere formátum, amely gépi feldolgozásra alkalmas. A JSON adatokat egy szöveges formátumban ábrázolja, és adatstruktúrákat, például objektumokat és tömböket képes reprezentálni. Mi az API-ban vesszük hasznát, mivel mind a PHP, mind a JavaScript támogatja a formátumot, beépített funkcióival könnyen átalakítható és kezelhető a továbbított adat.

### Bootstrap

A Bootstrap egy ingyenes és nyílt forráskódú frontend keretrendszer weboldalak és webalkalmazások fejlesztéséhez. Elsősorban HTML, CSS és JavaScript alapú. A Bootstrap célja, hogy segítsen az egyenletes és gyorsan fejlődő webes projektek létrehozásában, minimalizálva a tervezési időt és növelve a projekt hatékonyságát. A keretrendszer egy nagyon népszerű eszköz a webfejlesztők körében, mivel lehetővé teszi az egyszerű és gyorsan fejlődő webes projektek létrehozását, miközben biztosítja a reszponzív design és a konzisztens megjelenés előnyeit.

### PHP

A PHP egy általános szerveroldali szkriptnyelv, amelyet dinamikus weblapok készítésére fejlesztették ki. A hagyományos HTML lapokkal ellentétben a kiszolgáló a PHP-kódot nem küldi el az ügyfélnek, hanem a kiszolgáló oldalán a PHP-értelmező motor dolgozza fel azt. A programokban lévő HTML elemek érintetlenül maradnak, de a PHP kódok lefutnak. A kódok végezhetnek adatbázis-lekérdezéseket, létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli kiszolgálókkal. A PHP-kódok kimenete a megadott HTML elemekkel együtt kerül az ügyfélhez.

### HTTP, REST API

A REST (Representational State Transfer) egy elterjedt architektúra az alkalmazás-programozási interfészek (API-k) tervezéséhez és megvalósításához. A REST API-k az erőforrásokat (adatokat, szolgáltatásokat) ábrázolják, és az HTTP protokollt használják a kliens és a szerver közötti kommunikációhoz. A REST API-k állapotmentesek, azaz a kliens kérésének tartalmaznia kell az összes szükséges információt. A REST API-k népszerűségét az egyszerűség, az egységes interfész és a skálázhatóság adja, amely lehetővé teszi a nagyobb terhelések kezelését.

### MySQL

A MySQL egy ingyenes, nyílt forráskódú relációs adatbázis-kezelő rendszer. A MySQL könnyen használható, széles körben elterjedt és skálázható, ami azt jelenti, hogy nagy adatmennyiségeket is képes hatékonyan kezelni. Az adatok egyszerű kezelése és a megbízhatósága miatt a MySQL-t széles körben használják az üzleti alkalmazásokban, weboldalakban, mobilalkalmazásokban, játékokban és még sok más területen.

### C#

A C# (C-sharp) egy modern, objektumorientált programozási nyelv. Elsősorban a .NET platformhoz kapcsolódik, és az egyik leggyakrabban használt nyelv a Windows alkalmazások, webalkalmazások és szolgáltatások fejlesztéséhez. A C#-t széles körben használják üzleti alkalmazások, játékok, webalkalmazások, adatszolgáltatások és még sok más fejlesztésére.

### Git, GitHub

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver, vagy másképpen egy szoftverforráskód-kezelő rendszer, amely a sebességre helyezi a hangsúlyt. A GitHub ingyenes internetes szolgálatatás, amely a Git segítségével szoftverfejlesztési verziókövetés-szolgáltatást nyújt. A grafikus felület segítségével egyszerűen hozhatunk létre és kezelhetünk Git repository-kat. A projektünk menedzsmentjének nagy része a GitHub beépített Projects felületével lett végezve.

### ChatGPT

A ChatGPT (Generative Pre-trained Transformer) az OpenAI mesterséges intelligencia (MI) kutató laboratórium által kifejlesztett chatbot, mely a felhasználókkal való folyamatos kommunikáció automatizálása során értelmezőmodelleket használ, melyek segítségével a bevitt információkat azonnal interaktívan kezeli. Segítségével a sokszor hosszas keresést, interneten kutakodást lerövidíthetjük, egy-egy problémára másodperceken belül választ kaphatunk. Emellett kisebb algoritmusokat, mint tesztadatok generálása, vagy bizonyos adatok, esetleg kódrészletek helyességének ellenőrzése, is könnyebb, és még inkább gyorsabb elvégezni a ChatGPT segítségével.

## Adatszerkezet

Adatbázisunk célja a felhasználók adatainak, játékbeli állásuknak, illetve a játék általános információinak tárolása

### Alapadatok

CREATE DATABASE arcalite CHARACTER SET utf8;

|  |  |
| --- | --- |
| MySQL kliens verzió | 10.4.28 - MariaDB |
| Adatbázis neve | arcalite |
| Tárolómotor | InnoDB |
| Alapértelmezett illesztés | UTF-8 |

### Kapcsolati ábra

### Táblák

#### A profile tábla

A felhasználók regisztrációs adatait tárolja, illetve az összes játszott időt.

CREATE TABLE profile (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(64),

password VARCHAR(128),

played TIME DEFAULT 0,

email VARCHAR(128),

deletedAt DATETIME DEFAULT NULL

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| username | Szöveg |  | A fiók felhasználóneve |
| password | Szöveg |  | A fiók jelszava (PASSWORD függvénnyel hash-elve) |
| email | Szöveg |  | A fiók e-mail címe |
| played | Idő |  | Az összes játszott idő |
| deletedAt | Dátum és idő |  | A fiók törlésének időpontja |

#### Az achievement tábla

Az elérhető mérföldköveket tartja számon. Egyelőre nincs sokféle adat benne, későbbi fejlesztések miatt lett létrehozva.

CREATE TABLE achievement (

id INT PRIMARY KEY AUTO\_INCREMENT,

image VARCHAR(128)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| image | Szöveg |  | A mérföldkőhöz tartozó ikonkép helyét tárolja a szerveren. |

#### A proach tábla

Az adott felhasználó által elért mérföldköveket tartja számon.

CREATE TABLE proach (

profileid INT,

achievementid INT,

CONSTRAINT pk\_proach PRIMARY KEY (profileid, achievementid),

CONSTRAINT fk\_proach\_profile FOREIGN KEY (profileid) REFERENCES profile(id),

CONSTRAINT fk\_proach\_achievement FOREIGN KEY (achievementid) REFERENCES achievement(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| profileid | Egész | PK, FK | A fiók kódja. Kapcsolat a *profile* táblával. |
| achievementid | Egész | PK, FK | A mérföldkő kódja. Kapcsolat az *achievement* táblával. |

#### A player tábla

A felhasználó által készített karakterek adatait tárolja.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| name | Szöveg |  | A karakter neve |
| hp | Egész |  | A karakter életpontjainak maximális értéke |
| mp | Egész |  | A karakter manapontjainak maximális értéke |
| profileid | Egész | FK | A hozzátartozó fiók kódja. Kapcsolat a *profile* táblával. |
| avatarid | Egész | FK | A karakter avatárképének kódja. Kapcsolat az *avatar* táblával. |
| levelid | Egész | FK | A legutoljára elért szint kódja. Kapcsolat a *level* táblával. |
| playtime | Idő |  | A karakterrel játszott idő. |

*(a létrehozó kódrészlet a következő oldalon)*

#### Az avatar tábla

CREATE TABLE player (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(64),

hp INT,

mp INT,

profileid INT,

avatarid INT,

levelid INT,

playtime TIME DEFAULT 0,

CONSTRAINT fk\_player\_avatar FOREIGN KEY (avatarid) REFERENCES avatar(id),

CONSTRAINT fk\_player\_level FOREIGN KEY (levelid) REFERENCES level(id),

CONSTRAINT fk\_player\_profile FOREIGN KEY (profileid) REFERENCES profile(id)

);

A karaktereknek választható avatár képek elérési útját tárolja.

CREATE TABLE avatar (

id INT PRIMARY KEY AUTO\_INCREMENT,

splash VARCHAR(128),

image VARCHAR(128)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Mezőnév | Típus | Kulcs | Leírás |
| id | Egész | PK |  |
| splash | Szöveg |  | Az avatárhoz tartozó splash art elérési útja a szerveren. |
| image | Szöveg |  | Az avatárhoz tartozó kép elérési útja a szerveren. |

#### A level tábla

A szintekhez tartozó képek elérési útját tárolja.

CREATE TABLE level (

id INT PRIMARY KEY AUTO\_INCREMENT,

image VARCHAR(128)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| image | Szöveg |  | A szinthez tartozó kép elérési útja a szerveren. |

#### Az enemy tábla

A játékban lévő ellenfelek adatait tárolja.

CREATE TABLE enemy (

id INT PRIMARY KEY AUTO\_INCREMENT,

hp INT,

image VARCHAR(128)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| hp | Egész |  | Az ellenfél életpontjainak maximális értéke. |
| image | Szöveg |  | Az ellenfélhez tartozó kép elérési útja a szerveren. |

#### Az enemplay tábla

A karakter által már felfedezett ellenfeleket tárolja. Az adott karakter „csak ezeket ismeri”.

CREATE TABLE enemplay (

enemyid INT,

playerid INT,

CONSTRAINT pk\_enemplay PRIMARY KEY (enemyid, playerid),

CONSTRAINT fk\_enemplay\_enemy FOREIGN KEY (enemyid) REFERENCES enemy(id),

CONSTRAINT fk\_enemplay\_player FOREIGN KEY (playerid) REFERENCES player(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| playerid | Egész | PK, FK | A karakter kódja. Kapcsolat a *player* táblával. |
| enemyid | Egész | PK, FK | Az ellenfél kódja. Kapcsolat az *enemy* táblával. |

#### Az item tábla

A játékban megszerezhető tárgyak adatait tárolja

CREATE TABLE item (

id INT PRIMARY KEY AUTO\_INCREMENT,

image VARCHAR(128)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| image | Szöveg |  | A tárgyhoz tartozó kép elérési útja a szerveren. |

#### Az itemplay tábla

A karakter által már megszerzett/felfedezett tárgyakat tárolja. Az adott karakter „csak ezeket ismeri”.

CREATE TABLE itemplay (

itemid INT,

playerid INT,

CONSTRAINT pk\_itemplay PRIMARY KEY (itemid, playerid),

CONSTRAINT fk\_itemplay\_item FOREIGN KEY (itemid) REFERENCES item(id),

CONSTRAINT fk\_itemplay\_player FOREIGN KEY (playerid) REFERENCES player(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| playerid | Egész | PK, FK | A karakter kódja. Kapcsolat a *player* táblával. |
| itemid | Egész | PK, FK | A tárgy kódja. Kapcsolat az *item* táblával. |

#### A lang tábla

Az elérhető nyelveket tárolja.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| id | Egész | PK |  |
| name | Egész |  | A nyelv megnevezése |

#### Az achdesc tábla

CREATE TABLE lang (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(64)

);

A mérföldkövek nevét és leírását tárolja adott nyelveken.

CREATE TABLE achdesc (

achievementid INT,

languageid INT,

name VARCHAR(255),

description TEXT,

CONSTRAINT pk\_achdesc PRIMARY KEY (achievementid, languageid),

CONSTRAINT fk\_achdesc\_achievement FOREIGN KEY (achievementid) REFERENCES achievement(id),

CONSTRAINT fk\_achdesc\_languages FOREIGN KEY (languageid) REFERENCES lang(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| achievementid | Egész | PK, FK | A mérföldkő kódja. Kapcsolat az *achievement* táblával. |
| languageid | Egész | PK, FK | A nyelv kódja. Kapcsolat a *lang* táblával. |
| name | Szöveg |  | A mérföldkő megnevezése az adott nyelven. |
| description | Szöveg |  | A mérföldkő leírása az adott nyelven. |

#### Az avatardesc tábla

Az avatárok nevét és leírását/háttértörténetét tárolja adott nyelveken.

CREATE TABLE avatardesc (

avatarid INT,

languageid INT,

name VARCHAR(255),

description TEXT,

CONSTRAINT pk\_avatardesc PRIMARY KEY (avatarid, languageid),

CONSTRAINT fk\_avatardesc\_avatar FOREIGN KEY (avatarid) REFERENCES avatar(id),

CONSTRAINT fk\_avatardesc\_languages FOREIGN KEY (languageid) REFERENCES lang(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| avatarid | Egész | PK, FK | Az avatár kódja. Kapcsolat az *avatar* táblával. |
| languageid | Egész | PK, FK | A nyelv kódja. Kapcsolat a *lang* táblával. |
| name | Szöveg |  | Az avatár megnevezése az adott nyelven. |
| description | Szöveg |  | Az avatár leírása az adott nyelven. |

#### Az enemydesc tábla

Az ellenfelek nevét és leírását tárolja adott nyelveken.

CREATE TABLE enemydesc (

enemyid INT,

languageid INT,

name VARCHAR(255),

description TEXT,

CONSTRAINT pk\_enemydesc PRIMARY KEY (enemyid, languageid),

CONSTRAINT fk\_enemydesc\_item FOREIGN KEY (enemyid) REFERENCES enemy(id),

CONSTRAINT fk\_enemydesc\_languages FOREIGN KEY (languageid) REFERENCES lang(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| enemyid | Egész | PK, FK | Az ellenfél kódja. Kapcsolat az *enemy* táblával. |
| languageid | Egész | PK, FK | A nyelv kódja. Kapcsolat a *lang* táblával. |
| name | Szöveg |  | Az ellenfél megnevezése az adott nyelven. |
| description | Szöveg |  | Az ellenfél leírása az adott nyelven. |

#### Az itemdesc tábla

A tárgyak nevét és leírását tárolja adott nyelveken.

CREATE TABLE itemdesc (

itemid INT,

languageid INT,

name VARCHAR(255),

description TEXT,

CONSTRAINT pk\_itemdesc PRIMARY KEY (itemid, languageid),

CONSTRAINT fk\_itemdesc\_item FOREIGN KEY (itemid) REFERENCES item(id),

CONSTRAINT fk\_itemdesc\_languages FOREIGN KEY (languageid) REFERENCES lang(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| itemid | Egész | PK, FK | A tárgy kódja. Kapcsolat az *item* táblával. |
| languageid | Egész | PK, FK | A nyelv kódja. Kapcsolat a *lang* táblával. |
| name | Szöveg |  | A tárgy megnevezése az adott nyelven. |
| description | Szöveg |  | A tárgy leírása az adott nyelven. |

#### A saves tábla

Az adott karakter mentési fájljait tárolja el.

CREATE TABLE saves (

playerid INT,

time DATETIME DEFAULT CURRENT\_TIMESTAMP,

save LONGBLOB,

CONSTRAINT pk\_saves PRIMARY KEY (playerid, time),

CONSTRAINT fk\_saves\_player FOREIGN KEY (playerid) REFERENCES player(id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| **Mezőnév** | **Típus** | **Kulcs** | **Leírás** |
| playerid | Egész | PK, FK | A karakter kódja. Kapcsolat az *player* táblával. |
| time | Dátum és idő |  | A mentés időpontja. |
| save | Longblob |  | A mentési állomány. |

## A weboldal felépítése

### A weboldal szerkezete és formázás

A weboldal 5 darab HTML-fájlból áll, viszont a formázás egy központi CSS fájlban, a *style.css*-ben található, illetve a formázáshoz kapcsolódó JavaScript függvények a *script* mappában a *styleScript.js*-be kerültek.

A weboldalon használt színek többségét CSS-változókkal adjuk meg, és ezek segítségével alakítottunk ki egy sötét és világos témát az oldalnak. Ezt az oldal tetején a menüsávon található kapcsolóval állíthatjuk át. A kapcsoló a háttérben meghívja a *themeChange* függvényt, mely egy cookie-ban eltárolja a jelenleg kiválasztott témát, majd a *setTheme* függvény meghívásával beállítja azt. Az oldal *body* eleme kap egy *bsTheme* adattagot, mely értéke a jelenlegi téma, és amely alapján állítja be a CSS a változóit.

### Saját felugróablakok

A weboldalon hibaüzenetek megjelenítésére, illetve a profilműveletekhez szükséges kommunikációra saját készítésű felugróablakokat használunk. Maga a felugróablak egy div, benne egy címmel, paragrafussal, szöveges bemeneti mezővel és három gombbal. A közös *common.js* szkriptfájlban találhatjuk a felugróablak működéséhez és használatához szükséges függvényeket:

* **showPopup**: Az oldal tetejére csúsztatja a felugróablakot.
* **closePopup**: Az oldal teteje fölé csúsztatja („elrejti”) a felugróablakot.
* **setupPopup**: A felugróablak mindig megjelenő részeinek beállítását végzi el. Ezek a címszöveg, az üzenet, a „*Mégsem*” gomb beállítása, illetve a többi gomb funkciójának reset-elése.
* **setAlert**: Egy bemenet nélküli felugróablak beállítását végzi el. Paraméterekből beállítja a „*Rendben*” gomb tényleges szövegét, funkcióját, illetve elrejti a nemhasznált részeket.
* **setPrompt**: Egy bemenetet váró felugróablak beállítását végzi el. Paraméterekből beállítja az „*Elküld*” gomb tényleges szövegét, funkcióját, illetve megjeleníti a szükséges elemeket, és elrejti a nemhasználtakat. A paraméterek között szerepel a bemenet helyességét ellenőrző függvény, illetve a helytelen bemenetre visszaadott hibaüzenet is.
* **Alert**: A bemenet nélküli felugróablak használatát lehetővé tevő függvény. Meghívja a megjelenítést, beállítást és az esetleges előző ablak bezárását is.
* **Prompt**: A bemenetet váró felugróablak használatát lehetővé tevő függvény. Meghívja a megjelenítést, beállítást és az esetleges előző ablak bezárását is.

### Az API és a weboldal közti kommunikáció

Az API és a kliens közti kommunikációt JavaScript-tel kezeljük, a jQuery *$.ajax()* függvényének segítségével.

Több végpontnak is szüksége van információkra, mint a felhasználó azonosítója, neve, vagy a megjelenítés nyelvének azonosítója (jelen állapotban mindig 1, azaz magyar). Ezeket az oldal bejelentkezéstől kezdve cookie-kban (sütikben) tárolja, és elérésükhöz a *common.js* állományban áll rendelkezésünkre a *getCookie* függvény, mely a megadott nevű cookie értékét, illetve, ha nincs ilyen nevű cookie, *null* értéket ad vissza.

A tényleges API kérések függvényei a weboldalakhoz tartozó szkriptekben találhatóak.

|  |  |
| --- | --- |
| **Fájl** | **Feladat** |
| index.js | A főoldalon megjelenő karakterek adatait kéri le és tölti be. |
| login.js | A bejelentkezéshez, regisztrációhoz és új jelszó kéréshez szükséges bemenet-ellenőrzéseket és API-hívásokat intézi. |
| profile.js | A profil oldalon megjelenő adatokat kéri le és tölti be. |
| ranking.js | A rangsor táblákba kerülő adatokat kéri le és tölti be. |
| wiki.js | A lexikon adatait kéri le és tölti be. |

### Egyéb funkcionalitás a weboldalon

A menüsávon a bejelentkezésfüggő elemek beállítását elvégző függvények a *common.js* fájlban találhatóak. Itt ellenőrzi az oldal, hogy van-e *userid* eltárolva: ha nincs, a *Lexikon* menüpont nem lesz kattintható, illetve a *Bejelentkezés* menüpont fog megjelenni; ha van *userid*, a *Lexikon* elérhető lesz, illetve a *Bejelentkezés* helyett a *Profil* oldalra tudunk eljutni a menüből.

Kijelentkezéskor a már sokszor említett cookie-kat egyszerűen töröljük (felhasználó azonosító és -név), erre a *common.js* fájlban található a *logout* függvény.

## Az API felépítése

### A cél

A weboldal, illetve a játék és az adatbázis közötti kommunikációt hivatott az API ellátni. Az API PHP nyelven íródott, a visszatérési adatokat JSON formátumban adja vissza.

### Hibakezelés és közös metódusok

Mivel minden végpont meghívásakor az első lépés a bemenő adatok ellenőrzése, az adatbázis kapcsolat felállítása, illetve hibák esetén hasznos visszajelzés küldése, ezért ezek két „közösen használt” PHP-fájlba kerültek.

A *config.php* a bemenő adatok validálására (*hasProperFields* és *checkValidity*) és a visszatérési értékek visszaadására (*ReturnMessage*, *ReturnResult*, *ReturnQuery*) való függvényeket tartalmazza. Utóbbiak mind valamiféle adatot (akár egy üzenetet, akár adatbázisból lekért adathalmazt), mind egy HTTP-válaszkódot is visszaadnak, így hibaüzenetek küldésére is ezeket használják a végpontok.

A *connection.php* az adatbázis kapcsolatot hozza létre a *mysqli* osztály példányosításával. Ezzel központilag, egy helyen lehet megadni a kapcsolathoz szükséges információkat (*$CONFIG* tömb), mint az adatbázis neve, a szerver címe stb. Kapcsolódási hiba esetén hibát dob vissza, melyben megadja az SQL hibakódot és -üzenetet is.

#### A config.php által küldhető hibakódok

|  |  |  |
| --- | --- | --- |
| Válaszkód | Üzenet | Megjegyzés |
| 500 | Hiba az SQL lekérdezésben: (Hibakód: …) … |  |
| 405 | Hiba az API-hívásban. | Hibás metódus esetén |
| 400 | Hiba az API-hívásban. | Hibás bemenet esetén |

Ezeket a hibakódokat a többi végpont is adhatja, ott nincsenek ezek külön felsorolva.

### A végpontok

#### login.php

Feladat: Bejelentkeztetés

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| email | szöveg | A bejelentkezéshez használt e-mail cím. |
| password | szöveg | A fiókhoz tartozó jelszó. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| id | egész | A fiók egyedi azonosítója. |
| username | szöveg | A fiókhoz tartozó felhasználónév. |
| email | szöveg | A fiókhoz tartozó e-mail cím. |

Hibakódok:

|  |  |
| --- | --- |
| Válaszkód | Üzenet |
| 401 | Nincs ehhez az e-mail címhez regisztrált fiók! |
| 401 | Hibás jelszó! |
| 401 | Több fiók bejelentkezési adatai egyeznek! |

#### register.php

Feladat: Az újonnan regisztráló felhasználó adatainak rögzítése.

Metódus: **POST**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| username | szöveg | A fiókhoz tartozó felhasználónév |
| email | szöveg | A fiókhoz tartozó e-mail cím. |
| password | szöveg | A fiókhoz tartozó jelszó. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| code | egész | HTTP válaszkód |
| message | szöveg | A válaszüzenet |

Hibakódok:

|  |  |
| --- | --- |
| Válaszkód | Üzenet |
| 409 | Már van fiók regisztrálva ehhez az e-mail címhez! |
| 409 | Már van fiók ilyen felhasználónévvel! |
| 500 | Hiba az adatok feltöltése közben. |

#### email\_exists.php

Feladat: Ellenőrzi, hogy egy adott e-mail címhez tartozik-e aktív felhasználó.

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| email | szöveg | A fiókhoz tartozó e-mail cím. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| exists | logikai | Létezik-e a megadott e-mail címhez tartozó fiók. |
| username | szöveg | A fiókhoz tartozó felhasználónév. Ha az *exists* hamis, NULL. |
| email | szöveg | A fiókhoz tartozó e-mail cím. Ha az *exists* hamis, NULL. |

Hibakódok:

A *config.php* által adhatók.

#### new\_password.php

Feladat: Új jelszó beállítása.

Metódus: **POST**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| email | szöveg | A fiókhoz tartozó e-mail cím. |
| password | szöveg | A beállítandó jelszó. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| code | egész | HTTP válaszkód |
| message | szöveg | A válaszüzenet |

Hibakódok:

|  |  |
| --- | --- |
| Válaszkód | Üzenet |
| 500 | Hiba a frissítés közben! Az új jelszó nem került beállításra! |

#### profile.php

Feladat: A bejelentkezett felhasználó karakteradatainak lekérése.

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| userid | egész | A lekérni kívánt fiók egyedi azonosítója. |
| langid | egész | A lekért adatok nyelve. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| - | tömb[karakteradat] | Egy tömb a lekért karakteradatokkal. |

*A karakteradatok felépítése:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| id | egész | A karakter egyedi azonosítója. |
| name | szöveg | A karakter neve. |
| hp | egész | A karakter maximum életpontja. |
| mp | egész | A karakter maximum manapontja. |
| level | egész | A karakter legmagasabb elért szintje. |
| playtime | idő | A karakterrel játszott idő. |
| image | szöveg | A karakterhez tartozó splashart. |

Hibakódok:

A *config.php* által adhatók.

#### update.php

Feladat: E-mail cím vagy felhasználónév változtatás.

Metódus: **POST**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| userid | egész | A fiókhoz tartozó egyedi azonosító |
| username | szöveg | Az új felhasználónév. Ha üres szöveg, ignorálja a végpont. |
| email | szöveg | Az új e-mail cím. Ha üres szöveg, ignorálja a végpont. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| username | szöveg | A frissített rekordhoz tartozó felhasználónév. |
| email | szöveg | A frissített rekordhoz tartozó e-mail cím. |

Hibakódok:

A *config.php* által adhatók.

#### delete\_profile.php

Feladat: Fióktörlés

Metódus: **POST**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| userid | egész | A fiókhoz tartozó egyedi azonosító |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| code | egész | HTTP válaszkód |
| message | szöveg | A válaszüzenet |

Hibakódok:

|  |  |
| --- | --- |
| Válaszkód | Üzenet |
| 500 | Hiba a törlés során! A profil nem került törlésre! |

#### index.php

Feladat: A játékban játszható karakterek adatainak lekérése.

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| langid | egész | A lekért adatok nyelve |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| - | tömb[karakteradat] | Egy tömb a lekért karakteradatokkal. |

*A karakteradatok felépítése:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| id | egész | A karakter egyedi azonosítója. |
| image | szöveg | A karakterhez tartozó avatárkép elérési útja a szerveren. |
| name | szöveg | A karakter neve. |
| description | szöveg | A karakterről szóló leírás. |

Hibakódok:

A *config.php* által adhatók.

#### ranking.php

Feladat: A ranglistákhoz szükséges adatok lekérése.

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| type | szöveg | A ranglista szempontja. |
| langid | egész | A lekért adatok nyelve. |

Kimenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| - | tömb[adat] | Egy tömb a lekért adatokkal. |

*Az adatok felépítése, ha a type „Profile”:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| Felhasználónév | szöveg | A fiókhoz tartozó felhasználónév. |
| Játékidő | idő | A játékos összegzett játékideje. |
| Elért\_mérföldkövek | egész | A játékos által elért mérföldkövek száma. |
| Végigjátszások | egész | A játékos végigvitt játékainak száma. |

*Az adatok felépítése, ha a type „GameThrough”:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| Karakter | szöveg | A karakter neve. |
| Profil | szöveg | A karakterhez tartozó fiók neve. |
| Avatár | szöveg | A karakterhez tartozó avatár neve. |
| Játékidő | idő | A karakterrel játszott idő. |
| Elért\_szint | egész | A karakterrel elért legmagasabb szint. |
| Felfedezett\_ellenfelek | egész | A karakterrel felfedezett ellenfelek száma. |
| Felfedezett\_tárgyak | egész | A karakterrel felfedezett tárgyak száma. |

Hibakódok:

A *config.php* által adhatók.

#### wiki.php

Feladat: A lexikonhoz szükséges adatok lekérése.

Metódus: **GET**

Bemenet:

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| userid | egész | A lekérni kívánt fiók egyedi azonosítója. |
| langid | egész | A lekért adatok nyelve. |
| request\_type | szöveg | A lekérés típusa. |

Kimenet:

*Ha a request\_type „STATISTICS”:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| enemyFound | egész | A játékos által már felfedezett ellenfelek száma. |
| enemyAll | egész | Az ellenfelek összdarabszáma. |
| itemFound | egész | A játékos által már felfedezett tárgyak száma. |
| itemAll | egész | A tárgyak összdarabszáma. |

*Egyébként:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| - | tömb[adat] | Egy tömb a lekért karakteradatokkal. |

*Az adatok felépítése, ha a request\_type „ENEMY”:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| id | egész | Az ellenfél egyedi azonosítója. |
| name | szöveg | Az ellenfél neve. |
| hp | egész | Az ellenfél maximum életpontja. |
| desc | idő | Az ellenfél leírása. |
| image | szöveg | Az ellenfélhez tartozó kép. |

*Az adatok felépítése, ha a request\_type „ITEM”:*

|  |  |  |
| --- | --- | --- |
| Megnevezés | Típus | Leírás |
| id | egész | A tárgy egyedi azonosítója. |
| name | szöveg | A tárgy neve. |
| desc | idő | A tárgy leírása. |
| image | szöveg | A tárgyhoz tartozó kép. |

Hibakódok:

|  |  |
| --- | --- |
| Válaszkód | Üzenet |
| 400 | Hiba az API-hívásban. |

## Az alkalmazás felépítése

A játék forráskódjának általunk írt fájljai a játék mappáján belül a *Script* mappában találhatóak. A feladat megszabásai és tanulmányaink miatt C# nyelven írtuk a forráskódot. A .*cs* fájlok a *Script*-en belül feladatuk szerint csoportosítva vannak. A mappák feladatait az alábbi táblázat szemlélteti.

|  |  |
| --- | --- |
| **Mappa** | **Benne lévő fájlok feladata** |
| Game | A játék elemeinek, szereplőinek működését biztosító szkriptek. |
| Globals | Minden más által elérhető funkciók szkriptjei. Ilyen pl. az adatbázis-kapcsolatot létesítő fájl vagy a mentést és betöltést végző fájl. |
| Maps | A pályák váltásának kezelését végző szkriptek. |
| Mechanics | A játékban található mechanikák és logikák kezelését végző szkriptek. |
| Menus | A különféle menük és egyéb GUI elemek kezelését végző szkriptek. |

A szkriptfájlok mellett egy *.cs.uid* fájl is található, mely a GODOT játékmotor miatt kell, mivel az ezalapján ismeri fel és használja a *.cs* fájlokat.

### MainNode.cs

A GODOT-ban a játék mindegyik eleme egy úgy nevezett *Node*. Ez gyakorlatban egy *Node2D* nevű osztályból való származottságot jelenti. A *MainNode* a játék indításakor hozza létre a fő *Node*-ot, mely alá a játék többi eleme tartozni fog. Emellett a töltőképernyő és a főmenü *Node*-jait is legenerálja, illetve elindítja az adatok betöltését.

### A Game mappa

#### BasicProjectile.cs

#### ChargeProjectile.cs

#### Item.cs

#### Player.cs

#### SpellOracle.cs

### A Game\Enemies mappa

### A Game\Enemies\Bosses mappa

### A Game\Enemies\Projectiles mappa

### A Globals mappa

#### ConfigFileHandler.cs

#### DBConnector.cs

A fájlnak alapvetően három feladata van: Az első az adatbázissal való kapcsolat létrehozása. Az ehhez szükséges adatok (szervercím, port, adatbázisnév stb.) az osztály változóiban vannak eltárolva, melyek segítségével épül fel a connection string. Ez alapján az osztály a MysqlConnection osztály példányosításával létrehozza a kapcsolatot. A másik két funkció is ezt használja.

A második feladat a bejelentkezéskor a felhasználó adatainak lekérése. Lekérésre kerül a játékos azonosítója és felhasználóneve, illetve a külön játékmeneteinek adatai (név, játszott idő, stb.) és a hozzátartozó legutolsó mentés. A lekért adatok eltárolására a fájlban található két struct segítségevel történik. A UserData a játékos adatainak, illetve a CharacterData a játékos mentéseinek tárolására szolgál.

Az utolsó feladat az újonnan mentett játék feltöltése. \*\*\*(Itt vesztem el) EZ SEM TELJES

### A Maps mappa

### A Mechanics mappa

### A Menus mappa

## Tesztelés

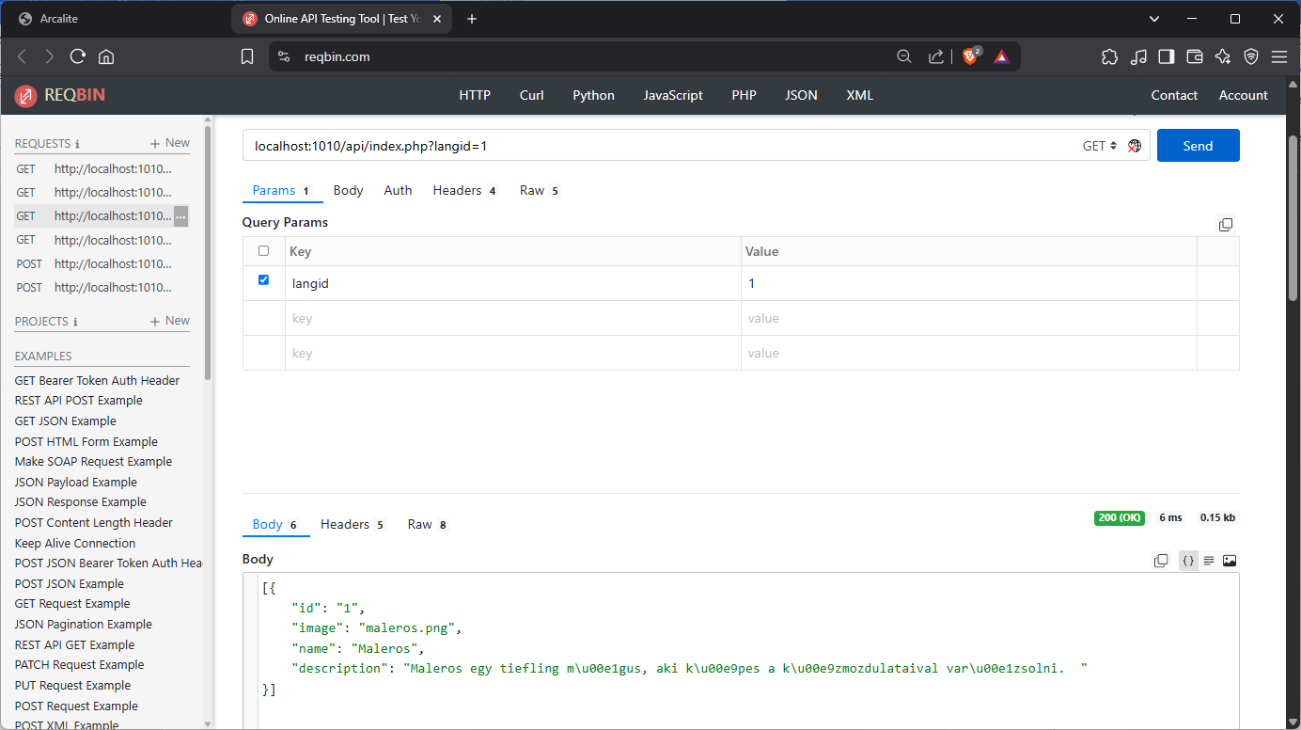
A szoftverfejlesztés nélkülözhetetlen lépése a tesztelés. Ezzel biztosíthatjuk, hogy a fejlesztett szoftver megbízhatóan működjék, illetve megfeleljen a célnak és a megrendelő elvárásainak.

A tesztelés elvégzését sokféleképpen megtehetjük, kezdve az egyszerű manuális tesztekkel, ahol mi magunk találunk ki tesztadatokat és teszteljük ki őket, egészen a tesztelésre készített programokig, melyek akár maguknak generálnak tesztadatokat, majd ellenőrzik, hogy az elvárt kimenetet kapták-e. Mi a projekt jellege és keretei alapján a manuális tesztelés mellett döntöttünk.

### Backend tesztelés

Az API teszteléséhez a ReqBin tesztelőalkalmazást használtuk. A ReqBin egy ingyenes online tesztelőeszköz, mely egy böngészős bővítménnyel a helyi szerverre is tud API-hívásokat küldeni. A kezelőfelületen egyszerűen megadható az URL, a meghívás típusa és törzse, illetve egyből látjuk a kérésre kapott választ, így könnyen és gyorsan tudjuk az API-nkat tesztelni. Fontos megjegyezni, hogy az oldal tényleges API-hívásokat intéz, így azok ténylegesen módosíthatják az adatainkat – amennyiben persze ilyen funkciót is kínál az API-nk.

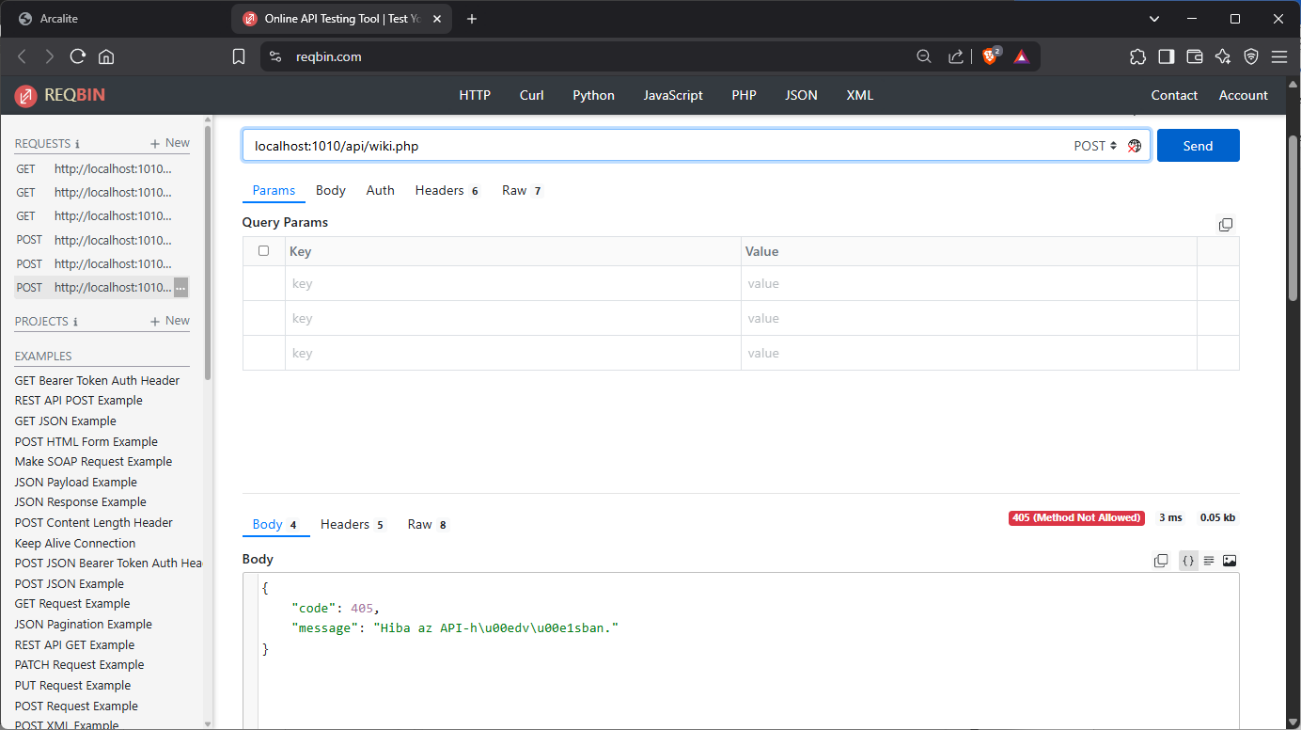
A következő oldalakon példákat hozunk a tesztelési folyamatból.



A ReqBin felülete

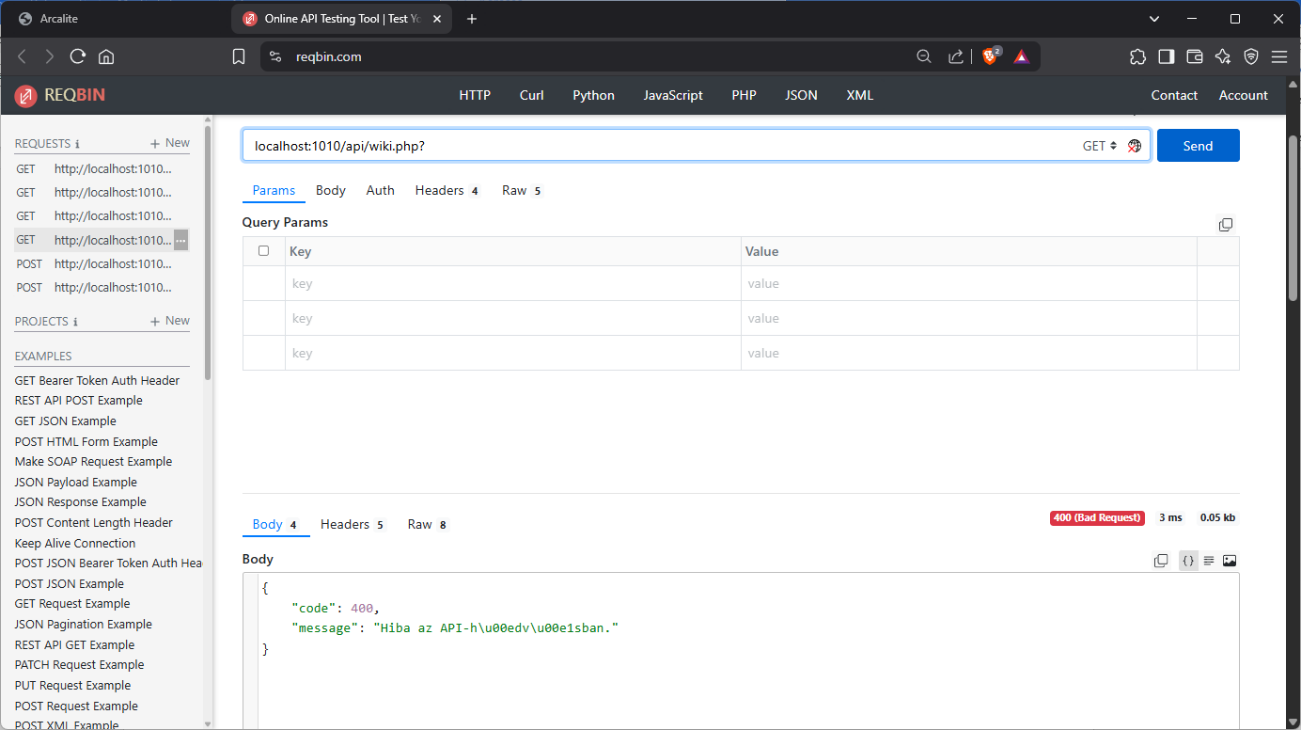
#### 1. teszteset: hibás metódus

Ha egy végpontot nem a megfelelő metódussal hívunk meg, eljárás szerint 405-ös (Method Not Allowed) válaszkódot kell visszaadnia, emellett JSON formátumban úgyszint ezt a 405-ös kódot, illetve a „Hiba az API-hívásban.” üzenetet. A képen látható, hogy az elvárt kimenetet kaptuk.

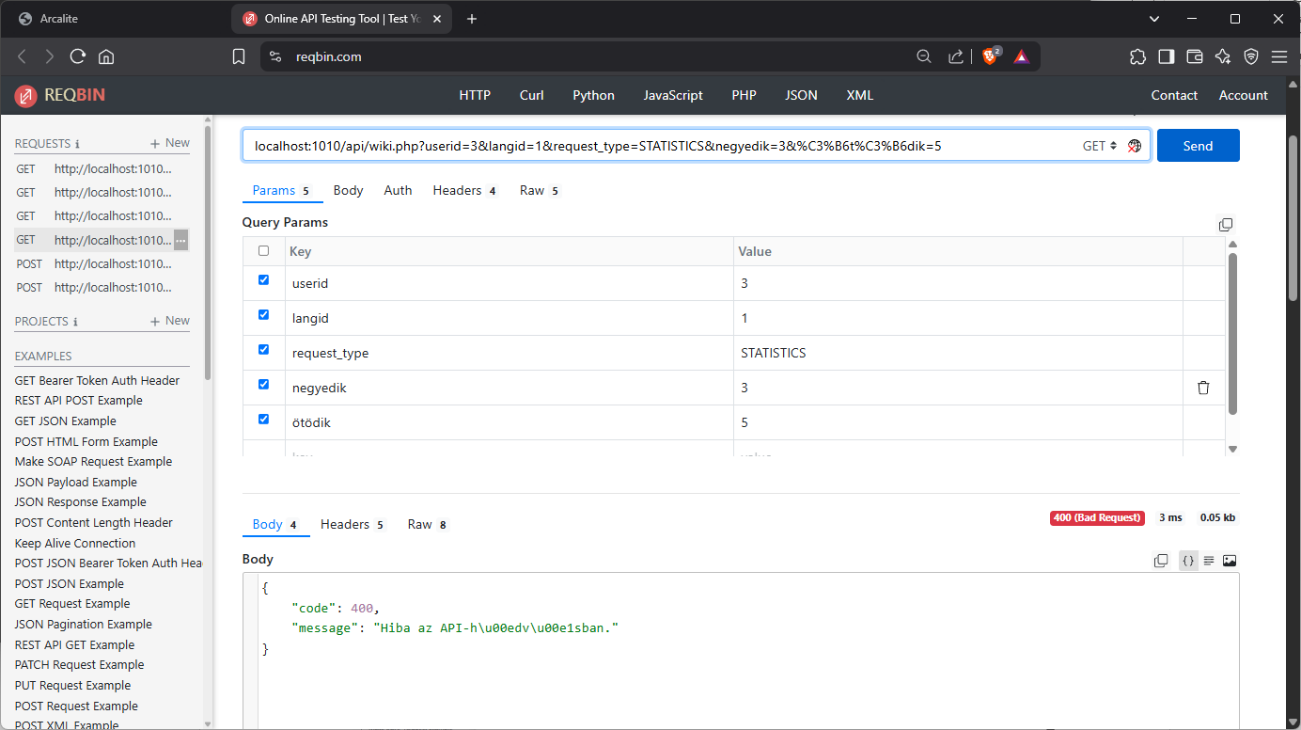


#### 2. teszteset(-csoport): hibás bemenet

Ha a hívás metódusa megfelelő, viszont a bemenő adatokkal van valami probléma, akkor 400-as (Bad Request) kódot kell az API-nak visszaadnia, illetve a „Hiba az API-hívásban.” üzenetet. Az első ilyen tesztnél bemenet nélkül hívtuk meg az API-t:



A második esetben több adatot adunk meg, mint amennyi elvárt. Itt látszik az is, hogy attól független is hibát ad vissza az API, hogy a kért adatok szerepelnek a bemenetben:



A harmadik esetben az elvártnál kevesebb adatot adunk meg:

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A negyedik teszt során megfelelő számú, viszont nem a hívás által elvárt adatokat adjuk meg:

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

#### 3. teszteset: helyes bemenet – egyéb hiba

Ennél a tesztesetnél az API-hívásunk helyes, ám valami egyéb hiba folytán hibaüzenetet kapunk vissza. Ilyen például regisztrációnál, ha már létezik a megadott felhasználónév vagy e-mail cím más (aktív) felhasználónál. Ilyenkor 409-es (Conflict) válaszkódot és a „Már van fiók regisztrálva ehhez az e-mail címhez!” vagy a „Már van fiók ilyen felhasználónévvel!” üzenetet kapjuk vissza.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

#### 4. teszteset: helyes bemenet és kimenet

Végül azt is ellenőrizzük, hogy megfelelő bemenettel helyes adat jön-e ki. Ilyenkor 200-as (OK) (regisztráció esetén 201 (Created)) válaszkódot kapunk, és a lekért adathalmazt, vagy egy sikeres lefuttatást visszaigazoló üzenetet.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

### Frontend tesztelés

#### Adatok megjelenítése

A weboldal fejlesztése során a leginkább kitesztelni való részek – nyilvánvalóan – azok, melyek változó adatokat tartalmaznak. Ilyen a lexikon, a ranglisták és a profil oldal. Itt alapvetően az adatok megjelenítését vizsgáljuk. Főleg a következő eseteket különböztetjük meg:

* ha nincs megjelenítendő adat
* ha az API hibát ad vissza
* ha az „átlagosnál” több adat van
* ha „átlagos mennyiségű” adatot kell megjeleníteni

Az „átlagos mennyiség” változó az oldalaknál, például a lexikonban – mivel ez csak a játékkal együtt bővül – csak általunk kiszámíthatóan sok adattal kell számolnunk. A ranglistáknál, mivel akárhány felhasználó regisztrálhat az oldalra, fontos a megfelelő skálázhatóság. A profil oldalon a felhasználó játékmeneteit soroljuk fel, és mivel elméletileg ez is bővülhet a végtelenségig, itt is fontos szerepet kellett kapjon a skálázhatóság. Viszont valószerűen gondolkodva elmondható, hogy átlagosan több felhasználó lesz az oldalon, mint játékmenete egy felhasználónak, így a ranglistákat általánosan nagyobb adatmennyiségre kellett felkészítenünk, mint a profil oldalt.

#### Reszponzivitás

Az adatok megjelenítésében az is fontos szerepet játszik, hogy különféle méretű eszközökről is megfelelően látszódjanak az adatok. Minden tesztesetnél nem csak a helyes betöltést kell vizsgálni, hanem a kisebb-nagyobb képernyőkön való megjelenést is.

### A játék tesztelése

## Továbbfejlesztési lehetőségek

# Felhasználói dokumentáció

## A program rövid ismertetése

## A program feltelepítése

### A játék telepítése

### A weboldal üzembe helyezése

### Az adatbázis felállítása

## Használati útmutató

### A játék irányítása

### A weboldal használata

# Mellékletek