****

ZÁRÓDOLGOZAT

Készítették:

Kiss Tamás

Kovács Levente

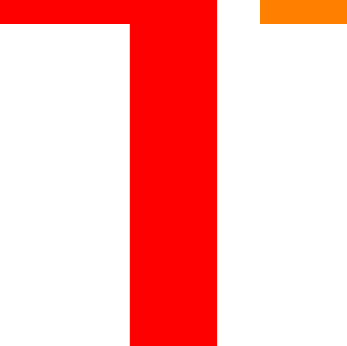
Kurucz Milán

Miskolc

2023.

**Miskolci SZC Kandó Kálmán Informatikai Technikum**

Az 5-0613-12-03 számú Szoftverfejlesztő és -tesztelő technikus szakképesítés záródolgozat

****

Tetris-project

Készítették:

Kiss Tamás

Kovács Levente

Kurucz Milán

Miskolc

2023.

**Tartalom**

[**TÉMAVÁLASZTÁS** 5](#_Toc133943653)

[**FELHASZNÁLT TECHNOLÓGIÁK** 6](#_Toc133943654)

[**1. MongoDB** 6](#_Toc133943655)

[**2. ASP.NET Core Web API** 7](#_Toc133943656)

[**3. NET MAUI** 8](#_Toc133943657)

[**4. Angular** 8](#_Toc133943658)

[**5. TypeScript** 9](#_Toc133943659)

[**6. SCSS** 10](#_Toc133943660)

[**7. JSON Web Token** 11](#_Toc133943661)

[**8. Bootstrap** 11](#_Toc133943662)

[**9. Mockaroo** 12](#_Toc133943663)

[**10. GitHub** 13](#_Toc133943664)

[**11. Trello** 13](#_Toc133943665)

[**12. Discord** 14](#_Toc133943666)

[**13. Visual Studio Code** 15](#_Toc133943667)

[**14. Visual Studio 2022** 16](#_Toc133943668)

[**ADATBÁZIS** 18](#_Toc133943669)

[**BACKEND** 19](#_Toc133943670)

[**1. Kapcsolódás az adatbázishoz** 19](#_Toc133943671)

[**2. Modellek** 19](#_Toc133943672)

[**3. Service-ek** 20](#_Toc133943673)

[**4. Automatizált e-mail küldés** 21](#_Toc133943674)

[**5. Controller-ek** 23](#_Toc133943675)

[**6. Authentication és authorization** 24](#_Toc133943676)

[**7. Összegzés** 24](#_Toc133943677)

[**ASZTALI ALKALMAZÁS** 25](#_Toc133943678)

[**1. Menü** 25](#_Toc133943679)

[**2. Új játék** 26](#_Toc133943680)

[**3. Statisztika és Bolt** 28](#_Toc133943681)

[**4. Beállítások és Kilépés** 28](#_Toc133943682)

[**FRONTEND** 29](#_Toc133943683)

[**1. Home oldal** 29](#_Toc133943684)

[**2. Shop oldal** 30](#_Toc133943685)

[**3. Shop-editor oldal** 30](#_Toc133943686)

[**4. Statistics oldal** 30](#_Toc133943687)

[**5. Users oldal** 31](#_Toc133943688)

[**6. User-editor oldal** 31](#_Toc133943689)

[**7. Profil oldal** 32](#_Toc133943690)

[**8. Privacy Policy oldal** 32](#_Toc133943691)

[**9. Sign-up oldal** 32](#_Toc133943692)

[**10. Login oldal** 33](#_Toc133943693)

[**11. Elfelejtett jelszó** 33](#_Toc133943694)

[**12. Reszponzivitás** 33](#_Toc133943695)

[**13. Kódminőség** 34](#_Toc133943696)

[**14. Tesztelés** 34](#_Toc133943697)

[**TOVÁBBFEJLESZTÉSI TERVEK** 35](#_Toc133943698)

[**1. ASP.NET Core Web API** 35](#_Toc133943699)

[**2. NET MAUI** 35](#_Toc133943700)

[**3. Frontend** 35](#_Toc133943701)

[**IDÉZETT FORRÁSMUNKÁK** 36](#_Toc133943702)

# **TÉMAVÁLASZTÁS**

A 2 éves szoftverfejlesztői tanulmányunk első éve végén egyik csapattagunknak ötletként felmerült egy hasonló alapokkal rendelkező játék, egy Tetris elkészítése, ám akkor még csak szimpla asztali alkalmazásként, sokkal egyszerűbb módon. Ez a gondolat akkor nem vetett gyökeret, talonban megmaradt.

Ez így is volt egészen a 2. évünk szeptemberéig, amikoris a csapatba szerveződés során az ötlet bekerült a közös kalapba, majd onnan az asztalra.

Végül a felvetődött ötletek közül projektmunkánk fő témájának a Tetris-t választottuk, mint világszinten ismert játékformát. Ez már az elején egy számunkra megfelelő témának tűnt a megbeszélések után, mivel mind Frontend szinten, mind a Backend részén részletgazdagon elkészíthető egy ilyen játék.

* Frontend terén kidolgozható akár a felhasználói fiókok kezelése, felhasználói jogosultságok, baráti felkérések és barátlista, vásárlások kezelése, statisztikai fülek megjelenítése
* Backendben kidolgozhatjuk az authentikációt és authorizációt, automatizált e-mail küldést állíthatunk be, az adatbázis összeköttetését a Frontenddel és az Asztali alkalmazásunkkal.
* Az Asztali alkalmazásban megvalósíthatjuk a játékhoz szükséges bejelentkezési felületet, a játékmenüt, amely a felhasználót egy karnyújtásnyira választja el a játékunktól, a boltrendszert, melyben a megszerzett érméit költheti el, kedvére nézheti a statisztikáit és teljesítményeit.

A játéktípus, hátterét illetően egy világszinten ismert, és nagy múlttal rendelkező blokkalapú játék, amelynek első verziója már 1984-ben játszható volt. A játék fő eleme a táblán megjelenő alakzatok egy sorba rendezése, majd ezzel a sor törlése és pontok/egyéb jutalmak szerzése.

Funkciókat illetően a hosszabb és könnyed játékot segítve rendelkezésünkre állhat egy adott blokk eltárolása későbbi felhasználásra, ennek cseréje az éppen aktuálissal, vagy akár az, hogy látjuk a következő elemet.

Ezeket a funkciókat tartalmazta az első hivatalosan megjelent Tetris is, majd ezekre építkezett a mögötte álló közösség. Így tettünk mi is, a projektünkben a fentebb említett funkcionalitásokat, illetve azok továbbfejlesztéseit dolgoztuk ki részletesen, akár saját grafikai elemekkel megtámogatva.

Bár a project folyamán többször dolgoztunk közösen, a feladatokat három részre osztottuk. A három terület elég jól elkülönül egymástól:

* Kiss Tamás – Backend (ASP.NET Core Web API)
* Kurucz Milán – Asztali alkalmazás (.NET MAUI)
* Kovács Levente – Frontend (Angular)

# **FELHASZNÁLT TECHNOLÓGIÁK**

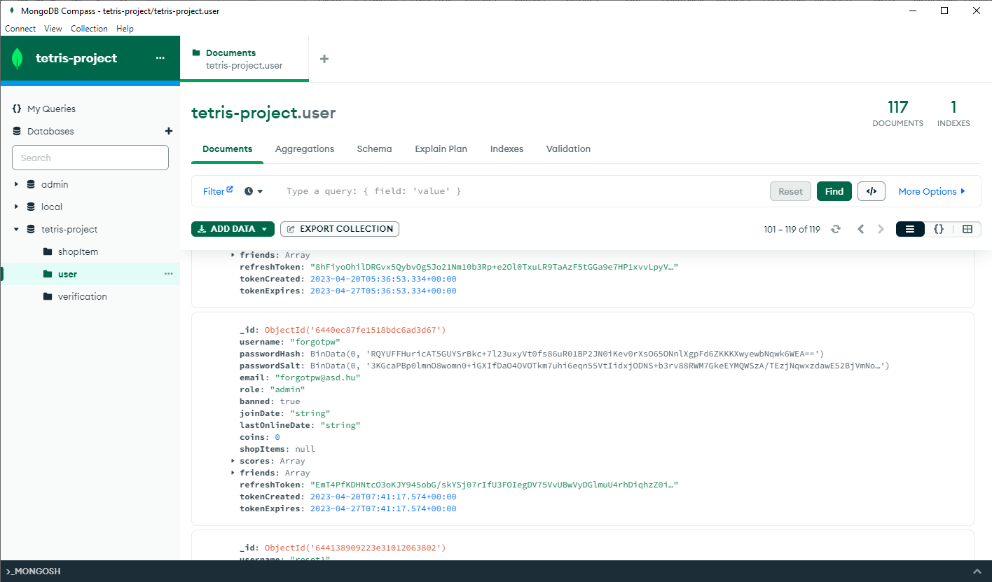
****

## **1. MongoDB**

A MongoDB egy dokumentumadatbázis, amely rendelkezik a kívánt skálázhatósággal és rugalmassággal, valamint a szükséges lekérdezéssel és indexeléssel.

* A MongoDB rugalmas, JSON-szerű dokumentumokban tárolja az adatokat, ami azt jelenti, hogy a mezők dokumentumról dokumentumra változhatnak, és az adatszerkezet idővel változhat.
* A dokumentummodell az alkalmazáskódban lévő objektumokhoz illeszkedik, így az adatokkal könnyű dolgozni.
* Az ad hoc lekérdezések, az indexelés és a valós idejű aggregálás hatékony lehetőségeket biztosít az adatok eléréséhez és elemzéséhez.
* A MongoDB alapvetően elosztott adatbázis, így a nagyfokú rendelkezésre állás, a horizontális skálázás és a földrajzi elosztás beépített és könnyen használható.
* A MongoDB használata ingyenes.[[1]](#footnote-1)

**Miért döntöttünk a MongoDB mellett?**

A legfőbb érv a MongoDB mellett, hogy egyszerűen, ingyenesen és számunkra megfelelő kapacitással biztosít felhő alapú tárhelyet. Így nem vagyunk a lokális hálózatra korlátozva, hanem bárhonnan gyorsan elérhetjük az adatbázisunkat. Továbbá a NoSQL mivolta lényegesen nagyobb szabadságot nyújt számunkra, így egyszerűsítve a bonyolultabb adatszerkezetekkel való munkát.

1. ábra

A user kollekció MongoDB Compass-ban

## **2. ASP.NET Core Web API**

Az ASP.NET Core Web API egy keretrendszer HTTP-alapú szolgáltatások építésére C# nyelven. Lehetővé teszi a fejlesztők számára, hogy RESTful webszolgáltatásokat hozzanak létre, amelyeket alkalmazások és eszközök széles köre vehet igénybe.

A keretrendszer a .NET Core futási időre épül, és hatékony funkciókat biztosít az routing-hoz, a szerializáláshoz és a hibakezeléshez. Támogatja továbbá az olyan modern webes szabványokat, mint az OData, a WebSockets és a SignalR.

Az ASP.NET Core Web API könnyű mikroszolgáltatások és összetett vállalati szintű alkalmazások létrehozására egyaránt használható.

Az ASP.NET Core a következő előnyöket kínálja:

* Egységes történet a webes felhasználói felület és a webes API-k építéséhez.
* Tesztelhetőségre tervezett architektúra.
* A Razor Pages egyszerűbbé és produktívabbá teszi az oldalközpontú projectek kódolását.
* A Blazor lehetővé teszi a C# használatát a böngészőben a JavaScript mellett. Megoszthajuk a szerver- és kliensoldali alkalmazáslogikát a .NET-tel írva.
* Lehetőség a fejlesztésre és futtatásra Windows, macOS és Linux alatt.
* Nyílt forráskódú és közösségközpontú.
* Modern, kliensoldali keretrendszerek és fejlesztési munkafolyamatok integrálása.
* Távoli eljáráshívás (RPC) szolgáltatások gRPC használatával történő hosztolásának támogatása.
* Felhőalapú, környezetalapú konfigurációs rendszer.
* Beépített függőségi injektálás.
* Könnyű, nagy teljesítményű és moduláris HTTP-kéréskezelő pipeline.
* A következőkön való hosztolás képessége:
  + Kestrel
  + IIS
  + HTTP.sys
  + Nginx
  + Apache
  + Docker
* Oldalankénti verziókezelés.
* A modern webfejlesztést egyszerűsítő eszközrendszer.[[2]](#footnote-2)

## **3. NET MAUI**

A .NET MAUI nyílt forráskódú, és a Xamarin.Forms továbbfejlesztése, amely a mobilról az asztali esetekre is kiterjed, a teljesítmény és a bővíthetőség érdekében az alapoktól kezdve újjáépített UI-vezérlőkkel. Ha korábban a Xamarin.Forms programot használtuk platformfüggetlen felhasználói felületek készítésére, sok hasonlóságot fogunk észrevenni a .NET MAUI-nál. Vannak azonban különbségek is. A .NET MAUI használatával egyetlen projekt segítségével multi-platform alkalmazásokat hozhatunk létre, de szükség esetén hozzáadhatunk platformspecifikus forráskódot és erőforrásokat is. A .NET MAUI egyik fő célja, hogy lehetővé tegye az alkalmazás logikájának és felhasználói felületének minél nagyobb részét egyetlen kódbázisban megvalósítani.[[3]](#footnote-3)

2. ábra

.NET Multiplatform App UI (MAUI)

https://learn.microsoft.com/en-us/dotnet/maui/media/what-is-maui/maui-overview.png?view=net-maui-7.0

## **4. Angular**

Az Angular egyike a két nagy, modern, nyílt forráskódú JavaScript keretrendszernek, amelyet a Google fejleszt és tart fenn. A népszerű keretrendszert Single-page webes alkalmazások (SPA) és dinamikus weboldalak építésére használják. Nagy előnye, hogy a komponensek újrafelhasználhatóak más projektekben is.

Az Angular egy TypeScriptre épülő fejlesztési platform. Platformként a következőket tartalmazza:

* Komponensalapú keretrendszer skálázható webes alkalmazások építéséhez.
* Jól integrált könyvtárak gyűjteménye, amelyek a funkciók széles skáláját fedik le, beleértve a routing-ot, az űrlapkezelést, az ügyfél-kiszolgáló kommunikációt és még sok mást.
* Egy fejlesztői eszközkészlet, amely segít a kód fejlesztésében, építésében, tesztelésében és frissítésében.

Az Angularral egy olyan platform előnyeit használhatjuk ki, amely az egy fejlesztőre szabott projektektől a vállalati szintű alkalmazásokig skálázható. Úgy lett kialakítva, hogy a frissítés a lehető legegyszerűbb legyen, így minimális erőfeszítéssel kihasználhatjuk a legújabb fejlesztések előnyeit. A legjobb az egészben, hogy az Angular ökoszisztéma több mint 1,7 millió fejlesztő, könyvtárszerző és tartalomkészítő változatos csoportjából áll.[[4]](#footnote-4)

Az Angular használat sok előnye közül az egyik legnagyobbnak a rengeteg elérhető csomag (package) bizonyult. Az előre elkészített package-ek használatával a kódolás egyszerűbbé és gyorsabbá vált. Sok számunkra teljesen új csomagot sajátítottunk el dokumentáció alapján, és használtunk fel a project-ünkben.

3. ábra

Angular package.json dependencies részlet

## **5. TypeScript**

A TypeScript egy olyan nagy teljesítményű programozási nyelv, amely a JavaScript típusos superset-je, azaz JavaScriptre épül, további funkciókat ad hozzá és egyszerű JavaScriptre fordul le. A Microsoft fejleszti, először 2012-ben jelent meg.

A TypeScript további szintaxist ad a JavaScripthez, hogy szorosabb integrációt támogasson a szerkesztővel, hogy korán észlelje a hibákat a fejlesztő szerkesztőjében.  
A TypeScript kód JavaScriptre fordul le, és bárhol futtatható, ahol JavaScript fut: A böngészőben, Node.js-ben, a Deno-n és az alkalmazásaiban.  
A TypeScript megérti a JavaScriptet, és típuskövetkeztetést használ, hogy további kód nélkül nagyszerű eszközöket biztosítson.[[5]](#footnote-5)

A TypeScript számos funkciót tartalmaz, például osztályokat, modulokat, interfészeket, enumokat és egyebeket. Emellett jobb eszköztámogatást is nyújt, beleértve a kódszerkesztőket és az IDE-ket, ami megkönnyíti a nagyméretű alkalmazások fejlesztését és karbantartását.

A typescript fejlesztés során végig ’strict’ módot használtunk, ezzel is törekedve a lehető legtisztább kódra.

## **6. SCSS**

Az SCSS (Sassy CSS) egy CSS preprocessor, amely kibővíti a CSS szintaxist, és olyan fejlett funkciókat biztosít, mint a változók, függvények, mixinek és a beágyazás.

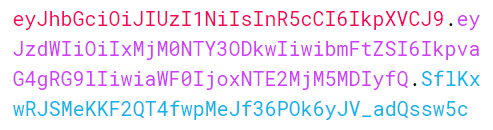
Lehetővé teszi a fejlesztők számára, hogy szervezettebb és karbantarthatóbb CSS-kódot írjanak változók és újrafelhasználható kódrészletek használatával, amelyek az egész kódbázisban felhasználhatók. Emellett az SCSS megkönnyíti az összetett CSS-stíluslapokkal való munkát, mivel lehetővé teszi a stílusok könnyebb szervezését és kezelését.

Az SCSS normál CSS-kóddá fordul, amely aztán használható a webes alkalmazásokban. Általában népszerű front-end eszköztárakkal együtt használják, mint például a Bootstrap.

4. ábra

Egy SCSS függvény és használata

## **7. JSON Web Token**

A JSON Web Token (JWT) két fél közötti információátvitel biztonságos eszköze. Gyakran használják webes alkalmazásokban a felhasználói authentikációhoz és authorizációhoz, valamint biztonságos webes API-k építéséhez. A token tartalmaz egy payload-ot, vagy claim-eket, amelyek kulcs-érték párokból és egy aláírásból állnak. A payload információkat tartalmaz a felhasználóról, például az azonosítóját, szerepét vagy jogosultságait, és egy titkos kulccsal van aláírva. A JWT-k önállóak, ami azt jelenti, hogy minden szükséges információ magában a tokenben található, így a token könnyen továbbítható a különböző rendszerekben lévő felek között.

5. ábra

JSON Web Token (JWT)



## **8. Bootstrap**

A Bootstrap egy népszerű front-end eszköztár, amely leegyszerűsíti a reszponzív és mobilbarát weboldalak tervezésének folyamatát. A Twitter által fejlesztett Bootstrap egy nyílt forráskódú keretrendszer, amely CSS és JavaScript komponenseket tartalmaz reszponzív weboldalak létrehozásához.

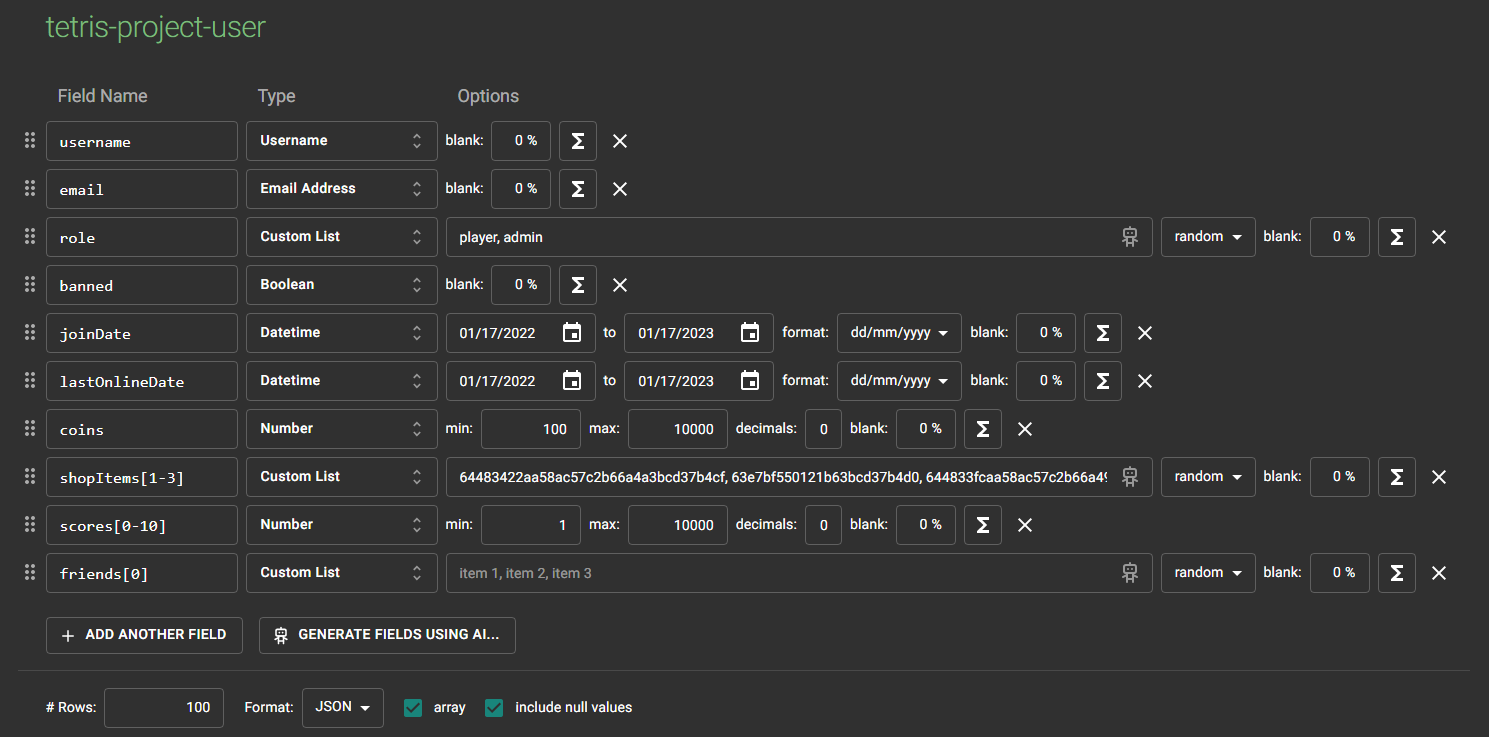
Számos előre elkészített tervezési elemet biztosít, például gombokat, űrlapokat, navigációs menüket, tipográfiát és rácsrendszert, amelyek könnyen testre szabhatók és kombinálhatók, hogy reszponzív és vizuálisan vonzó weboldalakat hozhassunk létre. Tartalmaz továbbá beépített JavaScript modulokat interaktív funkciók, például modal-ok, carousel-ek és legördülő menük létrehozásához.

A Bootstrap egyik legfontosabb jellemzője a rácsrendszere, amely lehetővé teszi a fejlesztők számára, hogy reszponzív elrendezéseket hozzanak létre, amelyek alkalmazkodnak a különböző képernyőméretekhez és eszközökhöz. A rácsrendszer 12 oszlopos elrendezésen alapul, melyek szélessége egyenlően oszlik el a befogadó elem szélességéhez képest. Ezen oszlopok segítségével a weboldal elemei dinamikusan átméretezhetők a felhasználói eszköztől függően, így a tervezési igényeinknek megfelelő egyéni elrendezéseket hozhatunk létre.

Összességében egy erőteljes és rugalmas keretrendszer, amely leegyszerűsíti a reszponzív és mobilbarát weboldalak készítésének folyamatát, így a webfejlesztők és -tervezők népszerű választása.

A weboldalunk szinte minden felületén használtuk a Bootstrap 5.0-s verzióját, törekedve arra, hogy ahol lehet, bootstrap osztályokat használjunk. Az osztályok mellett a beépített elemeket is használtuk. Bootstrap kompenens az alapja többek között a navigációs sávnak, legördülő menünek, kártyáknak, modalnak. Az előre elkészített komponensek használatánál ügyeltünk arra, hogy kellően testre szabjuk azokat, így egyedi megjelenést biztosítva az oldalunk számára. Ezekkel a módszerekkel lényegesen rövidebbé és áttekinthetőbbé vált a kódunk.

## **9. Mockaroo**

A Mockaroo egy webalapú eszköz, amely segít a felhasználóknak valósághű mintaadatok létrehozásában az alkalmazások teszteléséhez és prototípusok készítéséhez. Segítségével testreszabott adatkészleteket hozhatunk létre különböző adattípusok, például nevek, címek, telefonszámok, dátumok és egyéb adatok felhasználásával. Az eszköz lehetővé teszi a felhasználók számára, hogy speciális adatformátumokat konfiguráljanak, adatkorlátokat állítsanak be, és akár valósághű, véletlenszerű adathalmazokat is létrehozhatnak. Többek között támogatja a JSON, SQL, CSV, XML, vagy egyedi formátumokat is. Általában fejlesztők, tesztelők és tervezők használják mintaadatok létrehozására szoftverfejlesztési, tesztelési és prototípuskészítési célokra.

6. ábra

A User sémánk a Mockaroo oldalán

## **10. GitHub**

A GitHub egy webes platform, amely lehetővé teszi a felhasználók számára a szoftverfejlesztési projekteken való együttműködést. Felhőalapú tárhelyszolgáltatást nyújt a verziókezelési rendszerhez a Git segítségével, amely egy elosztott verziókezelő rendszer. A GitHub segítségével a fejlesztők repositorikat (mappákat vagy könyvtárakat, amelyekben kódot és kapcsolódó fájlokat tárolnak) hozhatnak létre, nyomon követhetik a kódjukban bekövetkezett változásokat, és együtt dolgozhatnak más fejlesztőkkel ugyanazon a projekten. Emellett projektmenedzsmenthez, kódellenőrzéshez és hibakövetéshez is biztosít eszközöket. A GitHub a nyílt forráskódú szoftverfejlesztés népszerű platformjává vált, és nagy fejlesztői közösséggel rendelkezik.

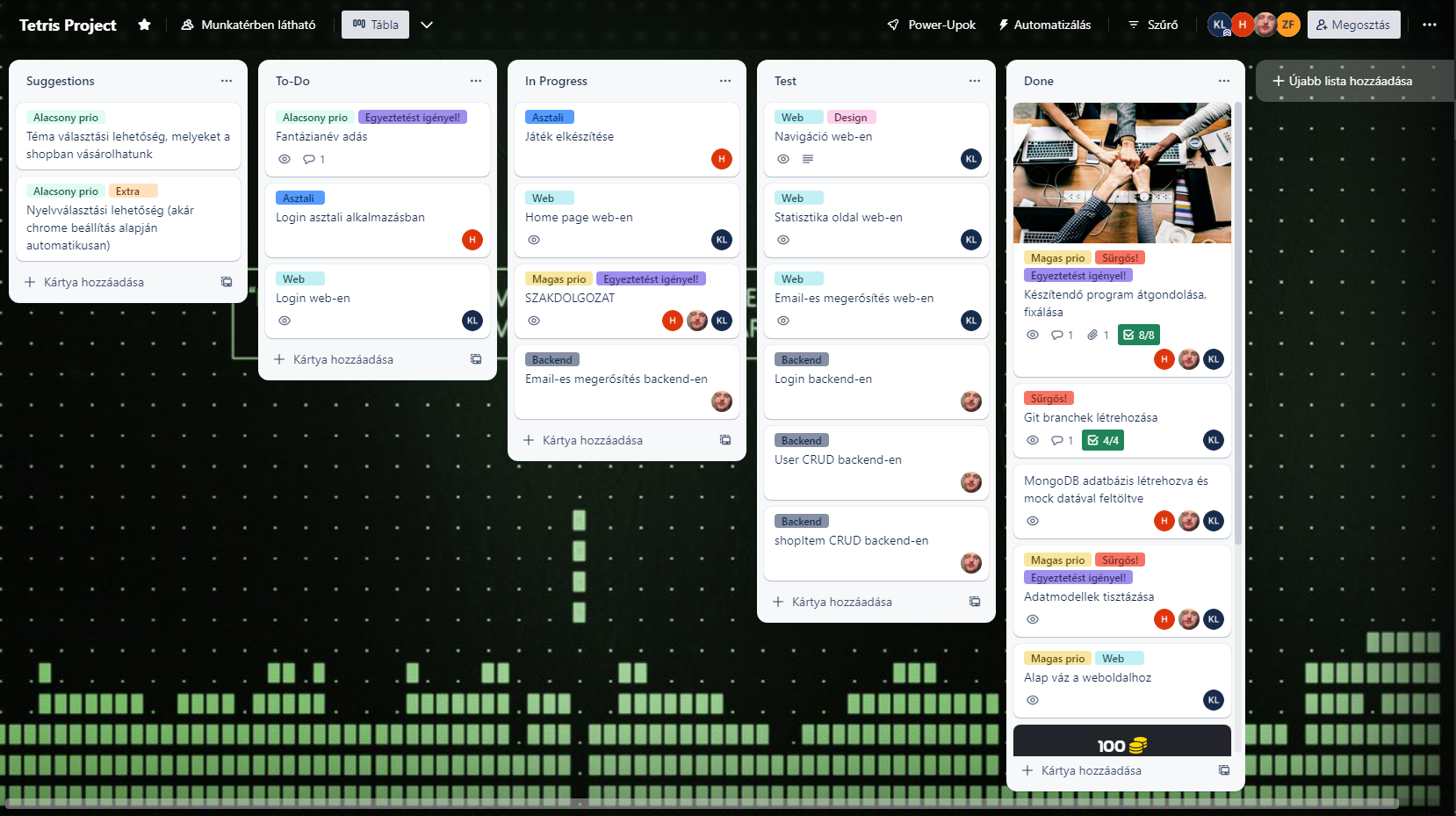
A project készítése során végig használtuk a GitHubot, ide töltve fel a már elkészült kódrészleteket. Több különböző branch-et is létrehoztunk, alapvetően egy main – dev – fejlesztők branch-ei struktúrát követve. Így mindenkinek volt egy saját branch-e, amelyből időnként az elkészült kódot a dev-be merge-öltük, míg meg nem győződtünk arról, hogy a kód teljesen jól működik. Miután a kód működése felől megbizonyosodtunk a dev branch tartalmát a main-be merge-öltük.

Az alap struktúrában szereplő branch-eken kívül létrehoztunk még egy-egy ideiglenes branch-et is a nagyobb feladatok számára, mint például a bejelentkezés. Erre azért volt szükség, mert nem mindig voltunk teljesen biztosak egy probléma megoldásában, kipróbáltunk számunkra új csomagokat és módszereket, így könnyedén előfordulhatott, hogy egy korábbi verzióra kellett visszaállni, amely így könnyen elérhető maradt a saját branch-ünkben.

## **11. Trello**

A Trello egy webalapú projektmenedzsment- és együttműködési eszköz, amely táblákkal, listákkal és kártyákkal segíti a csapatokat a feladatok és munkafolyamatok szervezésében és rangsorolásában. Rugalmas és vizuális eszköz, amely számos célra használható, például személyes feladatok kezelésére, projektek szervezésére, a haladás nyomon követésére és a csapatmunkák koordinálására.

A Trellóban minden egyes tábla egy projektet vagy témát képvisel, és az egyes táblákon belül a felhasználók listákat hozhatnak létre, amelyek a feladatok különböző szakaszait vagy kategóriáit képviselik. A felhasználók ezután az egyes listákon belül kártyákat hozhatnak létre, amelyek az egyes feladatokat jelképezik, és amelyeket a táblán belül mozgathatnak, hogy tükrözzék a státuszukban vagy prioritásukban bekövetkezett változásokat. A kártyákhoz leírások, mellékletek, esedékességi időpontok és egyéb részletek is tartozhatnak, így megkönnyítve a csapattagok számára az együttműködést és a szervezettséget.

Széles körben használják a legkülönbözőbb méretű és iparágú csapatok. Webböngészőn vagy mobilalkalmazáson keresztül is elérhető. Ingyenes és fizetős csomagokat is kínál, a fizetős felhasználók számára pedig további funkciók és integrációk állnak rendelkezésre.

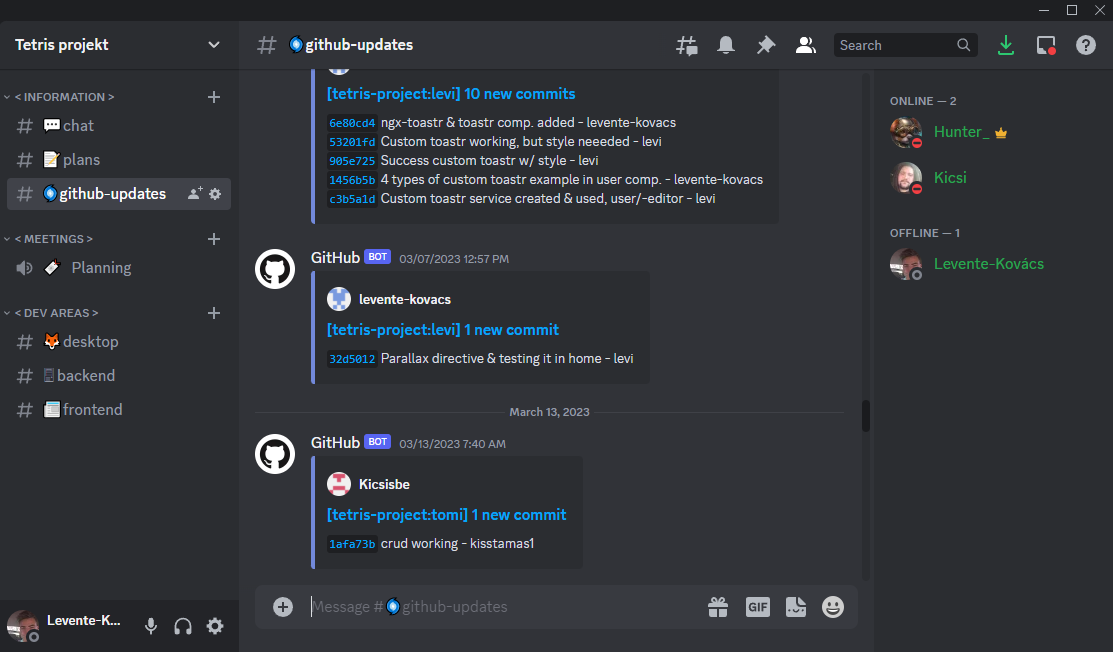
7. ábra

A csapat Trello-ja

## **12. Discord**

A Discord egy ingyenes hang-, videó- és szöveges csevegőalkalmazás, amelyet játékosok számára fejlesztettek ki, de a legkülönbözőbb közösségek használják. Lehetővé teszi a felhasználók számára, hogy szervereket hozzanak létre vagy csatlakozzanak hozzájuk, amelyek olyan virtuális terek, ahol a felhasználók valós időben kommunikálhatnak és együttműködhetnek text, voice, vagy video chat formában.

Csapatunk elsősorban Discord-on tartotta a kapcsolatot. Létrehoztunk egy saját, erre a célra kialakított szervert. A különböző területek külön chat felületeket kaptak, így megvolt a helye a tervezésnek, munkának is, és a kötetlen beszélgetésnek, akár meme-eknek is.

Az egész Discord használtban az egyik kedvencünk a szerverre meghívható GitHub bot volt, ami minden változást naplózott, amely GitHub-on történt (commit, merge, branch műveletek, stb.), egy külön erre a célra kialakított szobában.

8. ábra

A Discord szerverünk és a GitHub bot számára kialakított szoba

## **13. Visual Studio Code**

A Visual Studio Code (VS Code) ötvözi egy forráskód-szerkesztő egyszerűségét a hatékony fejlesztői eszközökkel, például az IntelliSense kódkiegészítéssel és hibakereséssel.

Támogatja a legelterjedtebb operációs rendszereket, mint a macOS, a Linux és a Windows - így platformtól függetlenül azonnal belevághatunk a munkába.

A program középpontjában egy villámgyors forráskód-szerkesztő áll, amely tökéletes a mindennapi használathoz. A több száz nyelv támogatásával a VS Code segít az azonnali produktivitásban a szintaxis-kiemeléssel, a zárójel-illesztéssel, az automatikus behúzással, a doboz-kiválasztással, a snippetekkel és még sok mással. Az intuitív billentyűparancsok, az egyszerű testreszabhatóság és a közösség által biztosított billentyűparancs-ábrázolások segítségével könnyedén navigálhatunk a kódban.[[6]](#footnote-6)

Érdekesség, hogy a VS Code teljes mértékben webes technológiával készült, mint például a TypeScript, HTML, CSS, NodeJs; jól mutatva ezzel a web sokoldalúságát.

A frontend fejlesztése során hatalmas szolgálatot tettek a program beépített billentyűparancsai. Ismeretükkel és használatukkal jelentős mennyiségű időt spóroltunk. Ezekből a kombinációkból alább listáztuk a számunkra leghasznosabbakat, amelyeket leggyakrabban használtunk.

|  |  |
| --- | --- |
| **Hasznos VS Code Billentyűparancsok** | |
| CTRL + P | Megnyitja a palettát egy fájl kereséséhez |
| CTRL + SHIFT + TAB | Navigáció a megnyitott állományok (tabok) között |
| CTRL + Ö | Megnyitja a terminált |
| CTRL + C | Megszakítja az éppen futó műveletet a terminálban |
| CTRL + SHIFT + F | Keresés a teljes projektben |
| CTRL + D | Kijelölést ad hozzá a következő találathoz |
| CTRL + F2 | Kijelöli az összes egyező előfordulást |
| CTRL + F  CTRL + H | Keresés és csere |
| ALT + SHIFT + Lefele nyíl | Megduplázza a sort |
| ALT + Felfele nyíl | Feljebb viszi a kijelölt sort eggyel |
| ALT + Lefele nyíl | Lejjebb viszi a kijelölt sort eggyel |
| Kattintás + ALT + SHIFT + Egér fel/le mozgatása | A metszett sorokhoz kurzorokat ad függőleges vonalban |
| TAB | Beljebb húzza a kijelölt sorokat |
| SHIFT + TAB | Kijjebb húzza a kijelölt sorokat |
| CTRL + K  CTRL + C | Kikommentezi a kijelölt sorokat |
| CTRL + K  CTRL + U | Kiveszi kommentből a kijelölt sorokat |
| Ctrl + K  CTRL + S | Megnyitja a beépített billentyűparancsok listáját |

## **14. Visual Studio 2022**

A Visual Studio 2022 a Microsoft által kifejlesztett integrált fejlesztőkörnyezet (IDE) legújabb verziója, amely különböző típusú alkalmazások - többek között asztali, webes, mobil és felhőalapú alkalmazások - készítésére és hibakeresésére szolgál. Ez a Visual Studio 2019 utódja, és hivatalosan 2022 februárjában jelent meg.

A Visual Studio 2022 számos új funkciót és fejlesztést tartalmaz, amelyek célja a fejlesztők termelékenységének és együttműködésének javítása. A figyelemre méltó funkciók közé tartozik a megújult felhasználói felület a jobb kereséssel és navigációval, a továbbfejlesztett IntelliSense és kódelemző eszközök, a jobb Git-integráció, a továbbfejlesztett hibakeresési képességek, valamint a .NET és a C++ legújabb verzióinak támogatása.

Az alapfunkciók mellett a Visual Studio 2022 számos speciális eszközt is tartalmaz a különböző típusú fejlesztésekhez, például a Xamarint a mobilalkalmazások fejlesztéséhez, az Azure-eszközöket a felhőalapú alkalmazásokhoz és a Unity-t a játékfejlesztéshez. Összességében a Visual Studio 2022 egy nagy teljesítményű és sokoldalú IDE, amely segíthet a fejlesztőknek a minőségi szoftverek hatékonyabb létrehozásában.

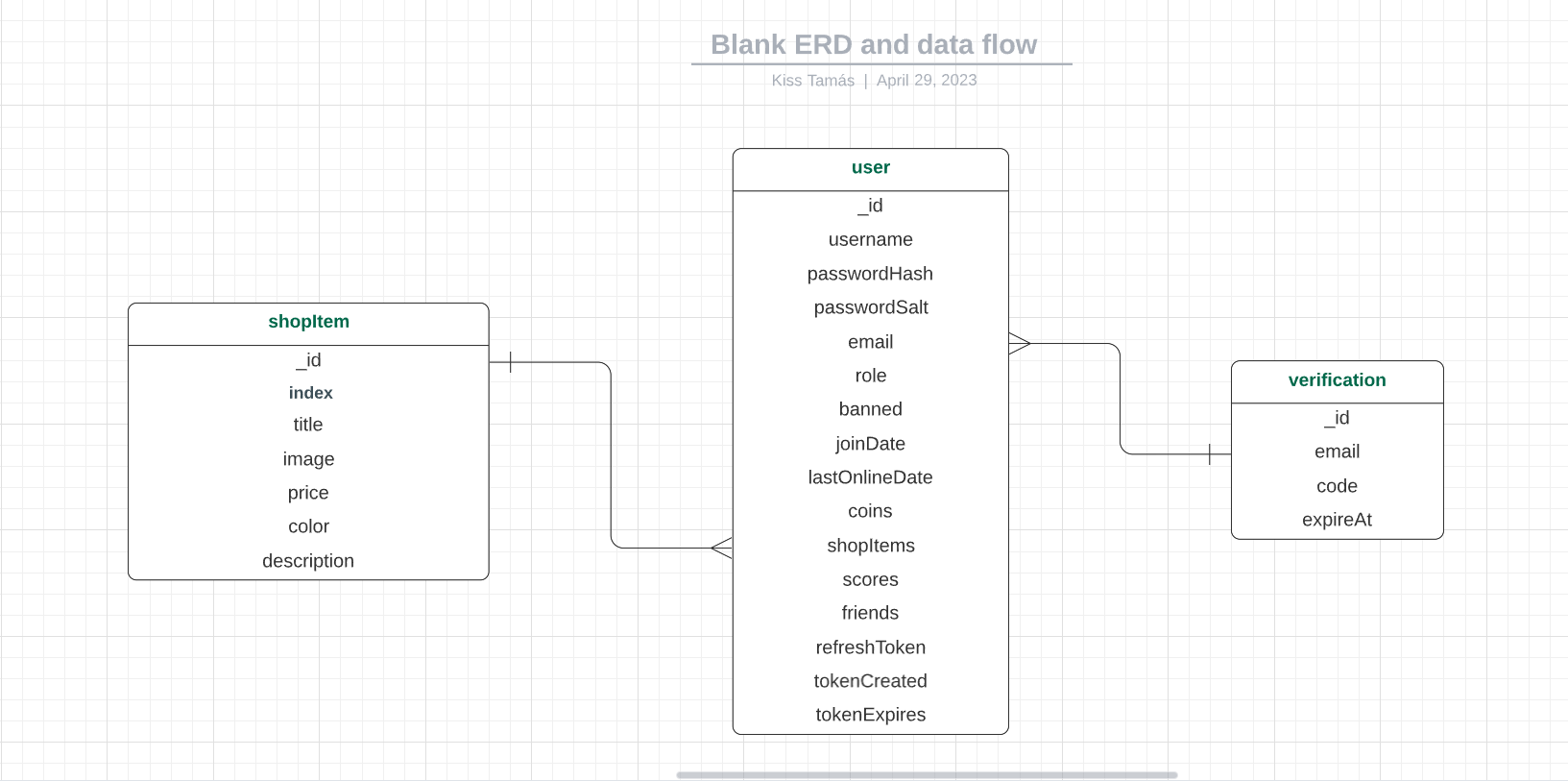
# **ADATBÁZIS**

Adatbázisnak a MongoDB mellett döntöttünk, elsősorban azért, mert az Atlas szolgáltatásuk a legfejlettebb felhőalapú adatbázis-szolgáltatás a piacon.[[7]](#footnote-7) Így nincs az adatbázisunk egy helyi hálózatra, vagy egyetlen számítógépre korlátozva, hanem bárhonnan gyorsan el tudjuk érni azt.

Az adatbázisunk három kollekcióból áll. (MongoDB-ben az SQL-ben megszokott táblákat kollekcióknak, a rekordokat pedig dokumentumoknak hívják.)

Létrehoztunk egy kollekciót a felhasználóknak, és egyet a boltban vásárolható tárgyaknak (skineknek). E mellett a két kollekció mellett van egy harmadik is, melyet verification névre kereszteltünk és az e-mailes megerősítéshez használt kódot, a fogadó e-mail címet és egy lejárati időpontot tartalmaz. Ebből a kollekcióból a dokumentumok a lejárati időben automatikusan törlődnek. A törlést a MongoDB hajtja végre.

A user kollekciónkat mock adatokkal, azaz mintaadatokkal töltöttük fel, a Mockaroo weboldal segítségével. Így gyorsan tudtunk generálni akár 100 valósághű dokumentumot is a kollekciónkba. Az adatbázisunk NoSQL tulajdonsága megengedi, hogy a mock adatok hiányosabban legyenek feltöltve. Ez azt jelenti, hogy például a jelszó tárolásához szükséges hash-el és salt-al nem rendelkeznek, mivel ezeket a dokumentumokat nem szeretnénk valós bejelentkezett felhasználóként használni.



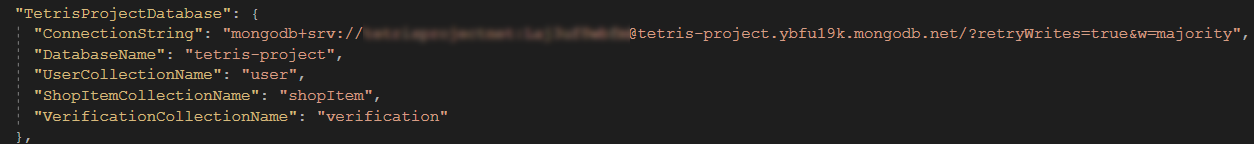
9. ábra

Az adatbázisunk NoSQL data model diagram-ja

# **BACKEND**

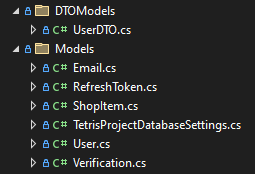
A kétéves szoftverfejlesztői képzésünk alatt főleg C# programozással foglalkoztunk, így érintettük a ASP.NET Core 6 Web API-t is. A vizsgaremek elkezdésekor ezen tanulmányi tapasztalataink alapján döntöttünk a keretrendszer használata mellett.

## **1. Kapcsolódás az adatbázishoz**

A backend-del az adatbázishoz a MongoDB Driver segítségével kapcsolódtunk. Az appsettings.json fájlban létrehoznunk egy JSON objektumot a szükséges adatokkal, majd a program.cs-ben megírtuk a kapcsolódáshoz szükséges kódot az objektum adatainak felhasználásával.

10. ábra

Az appsettings.json file-ban tárolt adatbázis-kapcsolódáshoz szükséges objektum



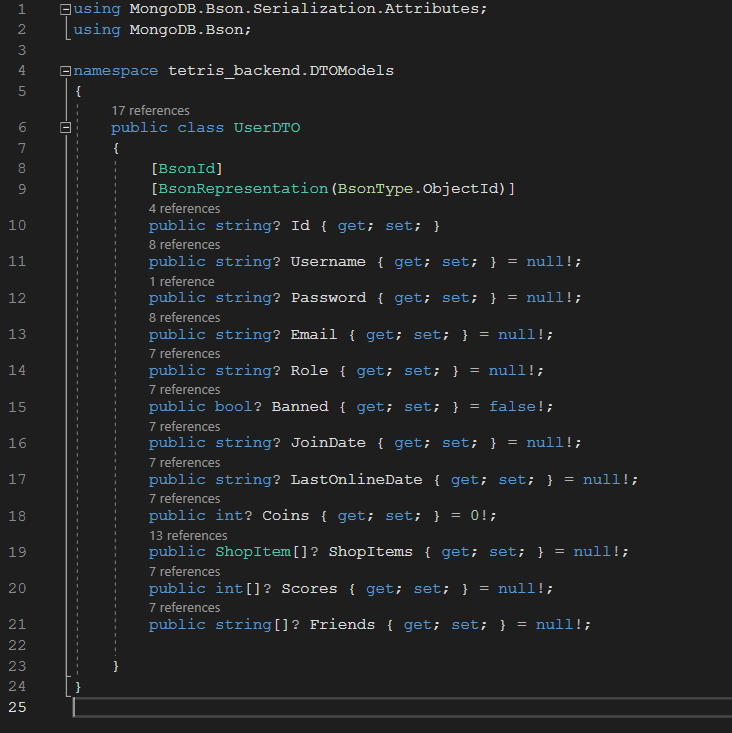
## **2. Modellek**

Az adatmodelljeink számára két különböző mappát is létrehoztunk. Az egyikben a hagyományos osztályokat tároljuk, míg a másikban egy DTO (Data Transfer Object) osztályt.

A hagyományos osztályok az adatbázisban tárolt adatok property-jeit tartalmazzák a megfelelő típusokkal. Például a User model shopItems property-je string[]? típusú, és ShopItem-ek ObjectId-jeit tárolja.

A DTO osztály viszont a frontendre küldeni kívánt szerkezetnek megfelelően lett kialakítva. Például a User DTO osztály ShopItems property-je ShopItem[]? típusú, és már teljes ShopItem objektumokat tárol.

A modellek Id property-jét külön meg kellett jelölnünk, hogy a MongoDB felismerje, hogy ezek ObjectId-k. Adatbevitelnél sosem adunk értéket az Id tulajdonságnak, mivel ezt automatikusan generálja a Mongo. Ezek az ObjectId-k 24 karakterből álló hexadecimális karakterláncok.



11. ábra

UerDTO model

## **3. Service-ek**

Mivel a NodeJS-ben elterjedt Express keretrendszer az ASP.NET Core 6 Web API-ban nem áll rendelkezésünkre, sem az Entity Framework nem támogatja a MongoDB-vel való munkát, így a service-ekben MongoDB Query Language (MQL) nyelven kellett megírnunk az adatbázis-kezelési műveleteket.

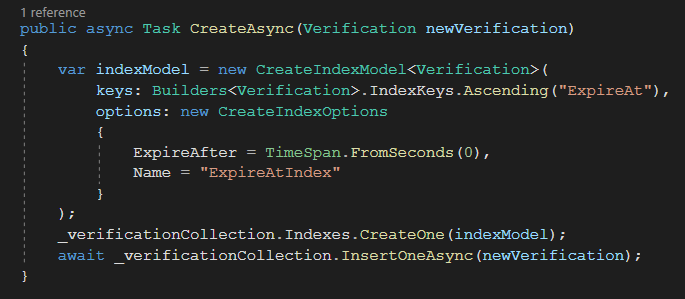
A service-ek tartalmazzák a CRUD összes műveletét, melyek:

1. összes dokumentum lekérdezése
2. egy dokumentum lekérdezése id alapján
3. dokumentum létrehozása
4. dokumentum módosítása
5. dokumentum törlése

A user service-ben lehetőségünk van az id alapú lekérdezés mellett felhasználónév és  
e-mail cím alapú lekérdezésekre is.

Az Email service e-mail küldés funkciót tartalmaz, mellyel a backend-ünk képes automatizált e-mail küldésre. Működéséhez egy egyszerű interface-t írtunk.

A Verfification service CreateAsync metódusa hozzáad egy indexet a dokumentumhoz, amely alapján a MongoDB automatikusan törli a dokumentumot a meghatározott idő eltelte után.

Az MQL nyelv támogatja az aszinkron működést, így összes az adatbázis-kezelési metódusunk aszinkron működésű, mely gyorsítja a kiszolgálási folyamatokat, mivel az egyes folyamatoknak lehetőségük van párhuzamosan végrehajtódni, nem kell megvárniuk a korábbi feladat befejeződését.

12. ábra

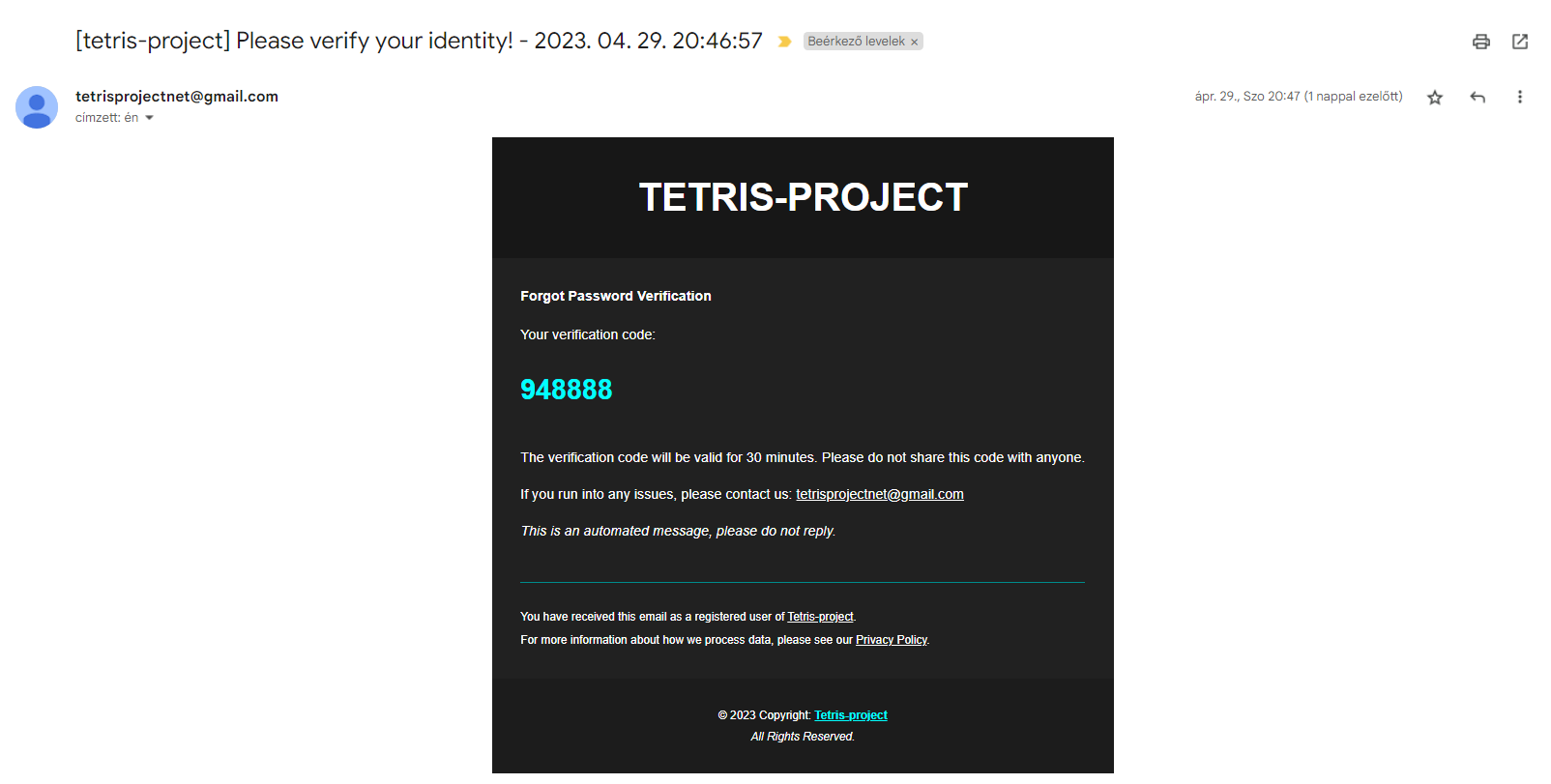
A Verification service CreateAsync metódusa

## **4. Automatizált e-mail küldés**

Az Email service SendEmail metódusát meghívva a backend-ünk egy automatizált e-mailt küld a paramtérben kapott címre. Az üzenetküldéshez a MailKit csomagot használtuk.

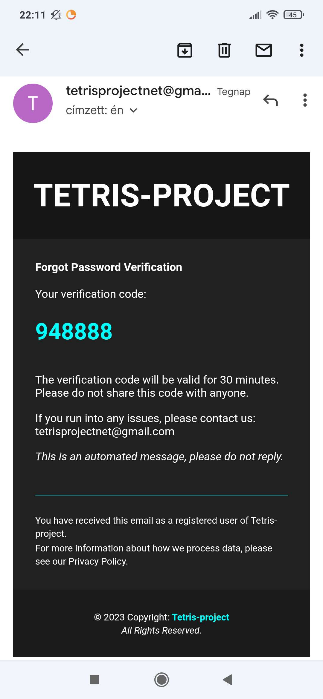
Az e-mailünknek beállíthatunk egy célcímet, egy feladót, mely esetünkben a [*tetrisprojectnet@gmail.com*](mailto:tetrisprojectnet@gmail.com), egy tárgyat, és egy tartalmat. A tárgyban foglaltakat HTML nyelvben kódoltuk le, melyet az e-mail szolgáltatók támogatnak. A HTML-t tartalmazó rész helyes kódolásához néhány dologban el kellett térnünk a mai webprogramozásban megszokott módszerektől. Néhány példa:

* a teljes HTML táblázatos felépítésű
* minden formázás inline
* a Bootstrap class-ok nem működnek
* az svg ikonok nem támogatottak



13. ábra A

A backend által automatikusan küldött e-mail a biztonsági kóddal **asztali** nézetben



13. ábra B

A backend által automatikusan küldött e-mail a biztonsági kóddal **mobil** nézetben

## **5. Controller-ek**

A backend-ünk végpontjai a Controllers mappa fájljaiban vannak definiálva. Az egyes végpontok a megfelelő service megfelelő metódusát (vagy metódusait) hívják meg. A program futtatásakor egy Swagger alkalmazás indul el, melyben egyszerűen tesztelhetjük az összes végpontot.

A kívánt úgynevezett üzleti logikát a controller-ek tartalmazzák, mely többek között:

* óvja az adatbázist a helytelen adatbeviteltől
* óvja az adatbázist a nemkívánt adatvesztéstől
* biztosítja a megfelelő adattovábbítást és fogadást a frontenddel
* hiba esetén megfelelő hibaüzeneteket küld a frontend felé

A „biztosítja a megfelelő adattovábbítást és fogadást a frontenddel” magában foglalja azt is, hogy a Modelleknél említett frontend-hez kapcsolódó DTO osztály és az adatbázishoz kapcsolódó osztály között a controller metódusai oldják meg a szerkezetváltást.

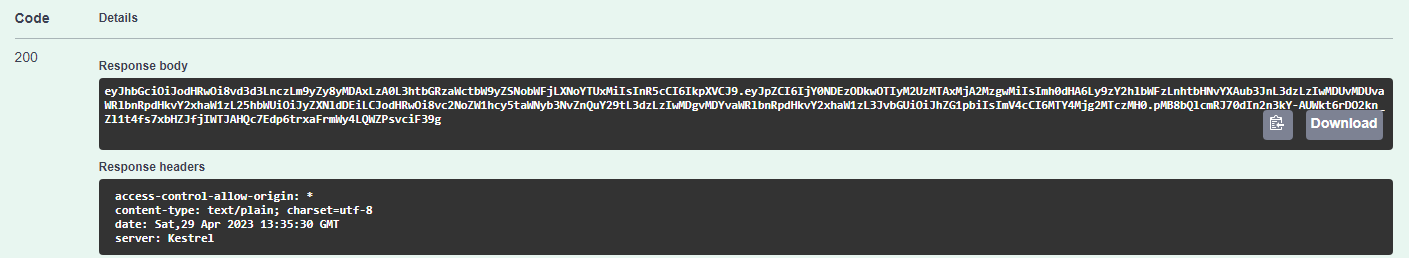
14. ábra

Végpontok a Swagger-ben

## **6. Authentication és authorization**

A végpontjainkat hitelesítéssel és azonosítással védjük, melyet JSON Web Tokenek (JWT) segítségével teszünk. Ez azt jelenti, hogy bizonyos végpontokat csak bejelentkezett, vagy csak az arra jogosult felhasználók érhetnek el.

Az auth controller Login végpontja egy felhasználónevet és egy jelszót vár. A jelszó egy dekódolási folyamaton keresztülmenve kerül hitelesítésre az adatbázisban tárolt passwordHash és passwordSalt alapján, melyeket az API-unk a regisztrációkor generál és tárol el. Megfelelő bejelentkezési adatok esetén a végpont egy három részből álló JWT-t küld a frontend felé string formában, melybe az adott felhasználó azonosításához szükséges adatok bele vannak kódolva.

A védett végpontok csak a JWT megfelelő formában való alkalmazása esetén érhetők el.

15. ábra

A Login metódus válaszában kapott JSON Web Token

## **7. Összegzés**

A fejlesztés folyamán hamar rájöttünk, hogy nem az ASP.NET Core 6 Web API volt a legjobb választás MongoDB adatbázis mellé. Míg a NodeJS-ben rendelkezésre áll az Express keretrendszer, a Web API-ban csak egy MongoDB Driver-t tudtunk használni az adatbázishoz való kapcsolódásra.

Visszatekintve összességében egyszerűbbnek bizonyult a tervezett funkciók elkészítése Web API-val, mint először hittük. A Microsoft dokumentáció nem vált a kedvencünkké, de a keretrendszert használó közösség sok hasznos segítséget nyújtott a program elkészítéséhez.

# **ASZTALI ALKALMAZÁS**

Projektünk felosztásából adódóan magát a játékot egy asztali alkalmazásként terveztük meg elkészíteni, amihez a Microsoft .NET legújabb, 2022-ben debütált MAUI keretrendszerét választottuk. A MAUI segítségével nevéből is adódóan (Multi-platform App UI) megfelelő platformspecifikációk mellett egy kódbázisból fejleszthetünk alkalmazást iOS-re, Androidra, macOS-re, és amit mi fő szempontnak választottunk, Windowsra, melyet a keretrendszer 7.0-s verziójában készítettünk el.

Az alkalmazást elindítva először a logónk, illetve egy bejelentkezési oldal fogad minket, ahol is a weboldalunkkal azonos felhasználónevet és jelszót kell megadnunk adatokként, melyet verifikálva az alkalmazás vagy beléptet, vagy jelzi felénk, hogy bizony elgépelhettünk valamit. A projekt alapnyelvének az angolt választottuk, mivel játékok során ez az elterjedt.

Rögtön jelszóemlékeztető kérésére, vagy ha még nincs felhasználónk, regisztrációra is van lehetőségünk ezen a fülön, mely a további adatok megadásáért a weboldalunkra irányítja a felhasználót. Ezek kitöltése után ezt bezárhatjuk, és visszatérhetünk az alkalmazásba, ahol immár biztos lehetőségünk van a bejelentkezésre.

A bejelentkezéssel kapcsolatos felhasználói authentikáció a Backend-ből kapott JWT Tokennel van megvalósítva.

## **1. Menü**

A játékba való bejelentkezés után a menübe juthatunk el, ahonnan a különböző menüpontokat érhetjük el.

Bal fent itt is megjelenítésre kerül és látható a felhasználó elkölthető érméje, illetve vele szemben a jobb sarokban a saját felhasználónk neve.

Rögtön szembe találjuk magunkat a menüből elérhető menüpontokkal:

* New game
* Statistics
* Shop
* Settings
* Quit

Rögtön a felső, Új játék menüpontot kiválasztva elindíthatjuk magát a játékot. Ezalatt találhatjuk meg a Statisztika és a Shop elemeket, amik a weboldalban elérhető Boltba, illetve annak Statisztika fülére navigálnak el minket. A Beállításokban szabályozhatjuk az alkalmazásspecifikus preferenciáinkat, illetve az alatta található kilépés gombbal zárhatjuk be a programot.

## **2. Új játék**

Új játék kezdésekor láthatunk egy visszaszámlálást, amely 3-tól indul, ekkor még csak a felkészülés zajlik. A játék során először kapunk egy elemet, amelyet szabadon mozgathatunk és lehelyezhetünk a pálya egy kívánt pontjára a WASD billentyűk segítségével. Ha nem sietünk, akkor pár másodperc múlva magától is eléri az alsó sort, mivel a játék során másodpercenként 1 sorral lentebb megy a jelenlegi elem. A cél az, hogy a sorokat feltöltsük, és ezzel töröljük a lerakott elemeket, pontokra váltva őket.

Az elemeket balra, jobbra illetve lefele a megfelelő (sorban A, D, S) billentyűgombokkal vezérelhetjük, ezeket letartva gyorsabban is mozgathatjuk őket. Egy elem akkor válik lerakottá, ha az adott körben már lefele nem tudjuk tovább mozgatni. Ilyenkor még van 1 másodpercünk oldalirányban változtatni rajta, onnantól a pozíciója állandó.

Minden elem (kivétel az O alakú, sárga négyzet) négyfajta előre definiált állapottal rendelkezik forgatás terén, ezáltal a megfelelő körülmények között szabadon forgathatóak a W billentyűvel. Ezek mellett vannak speciális esetek is, például mikor egy adott elem forgatási helyzete a jelenlegi pozíciókon nem férne el, ezért hiába nyomjuk a W-t, az elem nem forgatható már el.

Egy másik speciális eset, amikor a játék kerete mentén próbálunk forgatni, hiszen ilyenkor előfordulhat, hogy a forgatási pozíciók egyike már a pályán kívülre kerülne, ezért a játék automatikusan bentebb csúsztatja az elemünket egészen addig, amíg az összes elem egy lehetséges pozícióban helyezkedik el.

A képen szöveg, óra, sötét látható

Automatikusan generált leírásHa feltöltünk egy sort, akkor annak tartalma törlődik, és az összes felette lévő sor tartalma lentebb csúszik, ezáltal is több időt adva nekünk a játékra. Ha a sorok feltöltésére nem figyelünk eléggé, kevésbé taktikázunk, akkor hamar feltorlódhatnak az elemek, már pedig, ha nincs már hova mozgatni a következő elemet, a játékos időnek vége.

16. ábra

A játék végén látható Game Over felirat

Amíg tart a játék, a következő körben érkező elemet mindig a Next ablakban láthatjuk, mely az elérhető 7 elem közül véletlenszerűen sorsolódik ki. Ezt az érkezést akár fel is gyorsíthatjuk, ha már az elején kihasználjuk a Hold lehetőségünket, bár ez kevésbé ajánlatos.

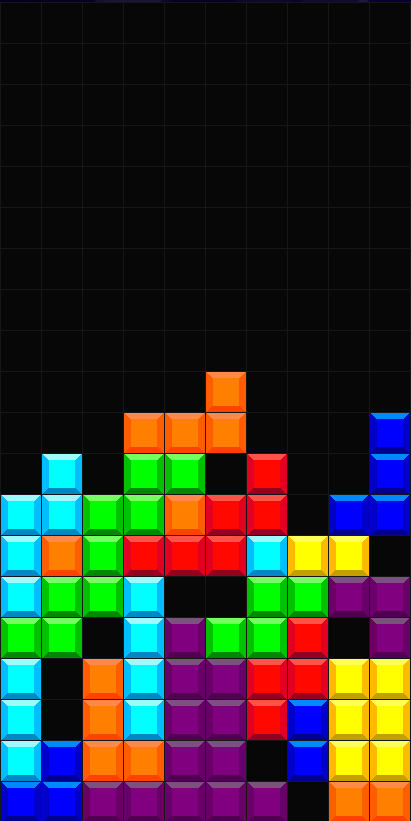
A Hold funkcióval elraktározhatunk egy elemet későbbi felhasználásra, vagy ha már ezt korábban megtettük, akkor a jelenlegi elemünket cserélhetjük ki az ottlévővel egy jobb blokklerakást megpróbálva. Ha az adott körben már cseréltünk elemet, akkor csak a következő körben tehetjük ezt meg újra, azaz a jelenleg nálunk lévő elemet előbb valahova le kell helyeznünk.

Az eddig elért pontszámunkat és a szerzett érméket a jobb alsó sarok jelzi, itt minden feltöltött sor után 65 pontot és 3 coint kapunk, de persze ezeket az értékeket növeli a játékban eltöltött időnk is.

Ha még szeretnénk játszani, a játék újraindítható a vége után, megállítható az Escape billentyű lenyomásával, illetve ebben az esetben a felugró Resume és Quit gombok közül utóbbival vissza kiléphetünk a menübe, míg előbbivel tovább játszhatunk a játékkal.

17. ábra

*Egy játék pillanatnyi állapota*



A játékból kilépve automatikusan elmentésre kerül az imént megszerzett érme, és letároljuk a felhasználó által elért pontszámot, melyet például részletesen elemezve visszanézhet a weboldal Statisztika fülén.

## **3. Statisztika és Bolt**

A menü Statistics gombjára kattintva megnyílik egy új böngészőablak, melyen a bejelentkezési adatainkat megadva részletesen megnézhetjük az előző játékainkat, adatainkat.

Ugyanígy tehetünk a Shop gombbal is, mely során elkölthetjük megszerzett érméinket és egyedi blokkokra cserélhetjük be, amelyeket a játék során használhatunk fel.

## **4. Beállítások és Kilépés**

A Settings gombra kattintva vehetjük lentebb a játék hangerejét, amely szabályozza a menü és a játékon belüli hangok fő hangerejét.

Itt választhatjuk ki a játékon belüli 7 elemre vonatkozó egyedi textúránkat, feltéve, ha korábban vásároltunk hasonlót.

A Quit gombbal tudunk kilépni a játékból, és ezzel bezárni az alkalmazásunkat.



18. ábra

*A teljes elérhető menü, amiben helyet kapott a Kilépés gomb is*

# **FRONTEND**

Mivel a csapatunk egyik tagja, Levente, a korábban elvégzett képzésein Angular-t tanult, így a frontend-ünket ennek a keretrendszernek a segítségével fejlesztettük. Az Angular TypeScript-en (TS) alapul, amely a JavaScript (JS) egy típusos superset-je. A TS a típusossága és több más tulajdonsága miatt is jobban hasonlít a Kandóban legtöbbet tanult C#-hoz, mint az alapja, a JS.

A frontend az Angular 15.0.3-as verziójával, a Node 18.12.1-es verziójával, npm 9.1.3-mal, rxjs 7.4.0-val és a typecsript 4.8.4-es verziójával készült Windows10 alatt.

A weboldalunknak három fő hitelesítési állapota van:

* nincs bejelentkezett felhasználó
* bejelentkezett egy játékos felhasználó
* bejelentkezett egy admin felhasználó

A weblap minden oldala tetején megtalálható egy navbar, melyen az aktuális hitelesítési állapottól függően, a felhasználó által elérhető oldalakon navigálhat. Ez azt jelenti, hogy például a Users oldalt nem tudjuk elérni admin jogosultság nélkül.

A navbar-hoz hasonlóan a footer-ünk is statikus, így megjelenik minden oldalunk legalján. Innen új ablakban megnyíló közösségi média platformokra, vagy a project fejlesztése alatt használt technológiákról szóló oldalakra navigálhatunk.

## **1. Home oldal**

A főoldalunk alig változik a bejelentkezéssel, illetve minden felhasználó ugyan azt látja és használhatja.

Az oldalra navigáláskor egyből az oldal tetején egy parallax effekttel ellátott város képét látjuk. Ha elkezdünk lefelé görgetni, a házak egymástól eltérő sebességgel mozognak a képernyőn, így 3D hatást keltve.

Az oldal következő (a város alatti) része a Shop oldalt népszerűsíti, melynek használatához az oldalon be kell jelentkeznünk.

Tovább görgetve egy frissítési napló részhez érünk, melyen a legfrissebb újításokat, fejlesztéseket, hibajavításokat listáztuk.

## **2. Shop oldal**

A Tetris-Shop-ban skin-eket vehetünk az asztali alkalmazásban megvalósított játékhoz. Hogy vásárolni tudjunk be kell lennünk jelentkezve.

A boltnak három állapota van:

1. bejelentkezés nélkül – látjuk a kártyákat, de a vásárlás funkció le van tiltva
2. bejelentkezett játékos – vásárolhat skin-eket, amíg a coin-jaiból kitelik
3. bejelentkezett admin – vásárolhat, illetve számára extra elemek is megjelennek az oldalon, mint a tétel törlése, tétel módosítása gombok, vagy az új tétel felvétele kártya

Ha egy skin az éppen bejelentkezett felhasználónak már a tulajdonában van, akkor azt már nincs lehetősége megvásárolni.

Ha az oldalra admin jogosultsággal navigálunk, egy kapcsoló segítségével ki- és bekapcsolhatjuk a fentebb említett extra elemek láthatóságát. Így adminként lehetőségünk van megnézni, hogyan fest a bolt egy játékos szemszögéből.

## **3. Shop-editor oldal**

Itt tarthatók karban a bolt tárgyai. Lehetőség van módosítani a skin-eket, törölni őket, vagy egy újjal bővíteni a Shop kínálatát.

Erre az oldalra csak az admin jogosultsággal rendelkező felhasználók navigálhatnak.

Minden skin-nek van egy szín tulajdonsága, ami meghatározza a kártyáján megjelenő háttérszínt is. Ha módosítjuk ezt a szín tulajdonságot, a kártyán megjelenő szín is megváltozik.

A beviteli mezők az adatoknak megfelelő típussal vannak ellátva, és a mentésre szolgáló gomb mindaddig le van tiltva, amíg minden kötelező mező megfelelően nincs kitöltve. Igy nincs lehetőség hibás adat bevitelére.

## **4. Statistics oldal**

Erre az oldalra biztonsági okokból csak bejelentkezett felhasználó navigálhat. Az oldalon az összes felhasználó adataiból készült statisztikák láthatók, bejelentkezés nélkül pedig nincs lehetőség lekérni az összes user adatát.

Az oldal felső részén táblázatokat helyeztünk el, melyek a legjobb statisztikákkal rendelkező felhasználókat tartalmazzák. Bal oldalon az egyetlen játékban valaha elért legjobb pontszámok alapján rendeztük sorba a top 5 játékost, míg jobb oldalon a felhasználók összes lejátszott játékának az átlaga alapján. Középen az a 10 játékos látható, akik játékaik összpontszámát tekintve az élen állnak.

Ha van már legalább hét lejátszott játékunk, akkor a táblázatok alatt egy radar diagram ábrázolja a legutóbbi hét játékunkat az összes játékos legutóbbi játékainak átlagaihoz viszonyítva.

Ha admin jogosultsággal jelentkeztünk be, akkor a radar diagram alatt egy kördiagramon láthatjuk a bolt skin-jeinek eloszlását.

****

## **5. Users oldal**

A Users oldalra csak és kizárólag admin jogosultsággal rendelkező felhasználók látogathatnak.

Itt tarthatjuk karban a felhasználókat, melyek egy lapozóval ellátott táblázatban vannak listázva. Alapértelmezetten a táblázatnak tíz sora van, de ezt kedvünk szerint módosíthatjuk. A lapozó oldalszáma ehhez igazodva dinamikusan változik.

A táblázatot szűrhetjük a felette bal oldalon elhelyezett beviteli mezők segítségével. A bal oldali legördülő listából kiválaszthatjuk az oszlopot, amelyben keresni szeretnénk, vagy az üres listaelem választásával lehetőségünk van a táblázat összes értéke közt szűrni. A jobb oldali beviteli mezőbe már csak be kell írnunk a kívánt kifejezést, és a táblázatban csak a szűrésnek megfelelő sorok maradnak.

A tábla fejléceire kattintva megváltoztathatjuk az adatok rendezési sorrendjét. Adott fejlécre való első kattintás után az új sorrend a választott oszlop értékeinek betűrendje szerint alakul. Ugyanezen fejlécre való második kattintás után a sorok visszafelé, Z-től A-ig lesznek rendezve. Az alapértelmezett rendezési forma az id alapján betűrendben megjelenés.

Az oszlopok sorrendjét is módosíthatjuk, amennyiben egy oszlop fejlécét az egérrel túlhúzzuk egy másik fejlécen. Az alapértelmezett oszlopelrendezéshez csak frissítenünk kell az alkalmazást.

Az oldalon lehetőségünk van a felhasználók kitiltására, vagy törlésére is. A kitiltott felhasználókat többé nem engedi bejelentkezni a rendszer.

Ha új felhasználót szeretnénk létrehozni, vagy módosítani egyet, akkor a megfelelő gombokra kattintva tovább navigálhatunk a User-editor oldalra.

## **6. User-editor oldal**

Erre a felületre a Users oldalról léphetünk tovább, így ezt is kizárólag admin-ok érhetik el.

Itt módosíthatjuk a felhasználók adatait, vagy regisztrálhatunk új felhasználót, megadva olyan tulajdonságait is, melyekre normál regisztrációkor nem lenne lehetőségünk. Így adhatunk az új user-nek admin jogosultságot, vagy az alapértelmezettől eltérő mennyiségű coin-nal rendelkező fiókot hozhatunk létre.

Ha egy felhasználó módosítása gombra kattintva navigáltunk az oldalra, akkor a felhasználó módosítható adatai mellett látjuk a nem módosítható adatait is, mint például a csatlakozása dátuma, vagy a játékaiban elért eredményei.

## **7. Profil oldal**

A bejelentkezett felhasználók a navbar-juk legördülő menüjén keresztül érhetik el azt az oldalt, ahol módosíthatják az adataikat.

Az admin jogosultsággal rendelkezők módosíthatják a felhasználónevüket, az e-mail címüket, a coin-jaik számát, illetve lefokozhatják a profiljukat játékos jogosultságúra is. Ezzel szemben a játékosok csak a felhasználónevüket módosíthatják.

## **8. Privacy Policy oldal**

Ezt az oldalt is a navbar legördülő menüjéből lehet elérni, ha be vagyunk jelentkezve.

Itt olvashatjuk el az oldal adatvédelmi szabályzatát és irányelveit.

## **9. Sign-up oldal**

A Sign-up oldalon regisztrálhatunk a felületre. Minden mezőt kötelező kitölteni, és nem adhatunk meg olyan felhasználónevet, vagy e-mail címet, amellyel korábban már regisztrált valaki.

Minden mező szigorúan validálva van, így a felhasználónévnek legalább három karakter hosszúnak és az email-címnek helyes formájúnak kell lennie. A jelszó legalább hat karakterből kell álljon, tartalmaznia kell legalább egy kis- és nagybetűt és egy számot. A regisztráció gomb mindaddig le van tiltva, míg minden mező meg nem felel az elvárt követelményeknek.

A működő gombra kattintás után az oldal egy értesítéssel tájékoztat minket a regisztráció eredményéről. Ha hiba lépett fel, akkor egy Error értesítést kapunk a megfelelő üzenettel. Amennyiben a regisztráció sikeres volt, akkor az oldal egy Success értesítést küld, és tovább irányít minket a Login oldalra.

## **10. Login oldal**

Ezen az oldalon jelentkezhetünk be a felületre. Meg kell adnunk a felhasználónevünket és a jelszót. Sikeres bejelentkezés esetén az oldal értesítésben adja ezt tudtunkra, és a Home oldalra navigál, míg sikertelen belépés esetén egy Error értesítést küld a hibának megfelelő üzenettel. Figyeltünk arra, hogy ne áruljunk el túl sokat az esetleges hiba okáról, csak azt, ami feltétlenül szükséges.

Abban az esetben, ha elfelejtettük a jelszavunkat, lehetőségünk van megváltoztatni azt.

## **11. Elfelejtett jelszó**

Olyankor, ha elfelejtettük a jelszavunkat, a Login oldalról tovább navigálva eljuthatunk a Forgot Password az oldalra, melyen az e-mail címünket adhatjuk meg. Amennyiben az e-mail cím valid, és regisztráltak már vele, a Send e-mail gombra kattintva a szerverünk egy automatikus e-mailt küld a megadott címre és tovább küld minket a Verification oldalra.

A Verification oldalon egy hatjegyű számot váró OTP (one-time password) beviteli mező fogad minket, melybe karakterenként is beírhatjuk az e-mailben kapott kódot, vagy egyben is bemásolhatjuk azt. Ha az OTP input megtelik, az alkalmazás automatikusan leellenőrzi azt. Amennyiben a kód helyes, tovább irányít minket a New Password oldalra, ha viszont az azonosítás nem sikerült, akkor egy Error értesítést kapunk a megfelelő üzenettel. Itt is ügyeltünk arra, hogy ne áruljunk el túl sokat az esetleges hiba okáról, csak azt, ami feltétlenül szükséges.

A New Password oldalra jutva sikeresen túljutottunk a hitelesítési folyamaton, és megváltoztathatjuk a jelszavunkat. Az újonnan megadott jelszóra is ugyan azok a feltételek igazak, amelyeket a regisztrálásnál is megkövetel az oldal. Sikeres módosítás esetén értesít minket erről az applikáció, illetve tovább navigál a Login oldalra. Sikertelen módosítás esetén a megszokott Error üzenetet kapjuk.

****

## **12. Reszponzivitás**

Az applikációt Mobile First szemléletmódban fejlesztettük, feltételezve, hogy többször lesz telefonról használva, mint számítógépről. Ez a kódolásban úgy nyilvánult meg, hogy próbáltuk először telefonos nézetben elkészíteni az oldalakat, és a szélesebb képernyőkre csak ezután optimalizáltuk a tartalmat. Így minden felületünk egyaránt megfelelően működik, és e mellett vonzó megjelenéssel is rendelkezik bármilyen képarányú eszközön.

## **13. Kódminőség**

A fejlesztés folyamán végig szem előtt tartottuk a tiszta kód alapelveit.

* Beszédes angol nyelvű elnevezések
* CamelCase alkalmazása az elnevezésekben (pl. avgUserData)
* Az Angular közösség által megszokott jelölések használata (pl. shopItem$)
* Minél jobb olvashatóságra törekvés
* Good és Best practice-ek alkalmazása
* Optimalizáltságra való törekvés

## **14. Tesztelés**

A weboldalt teszteltük is, melyhez Python-ban a Selenium WebDriver-t használtuk. A teszt program egy automatizált tesztfolyamatot futtat az oldalon, mely egymás után végig próbálja az alkalmazás lehetséges állapotait. Ezek a bejelentkezés nélkül elérhető oldalak, a játékosként elérhető oldalak és az admin-ként elérhető felületek.

# **TOVÁBBFEJLESZTÉSI TERVEK**

## **1. ASP.NET Core Web API**

* Kód refaktorálása
* Több végpont
* Még biztonságosabb végpontok
* Szerver oldali lapozó
* HTTP események naplózása
* Konténerizálás

## **2. NET MAUI**

* Milán ezt még írd meg légyszi :D

## **3. Frontend**

* Dark/Light Mode kapcsoló
* Nyelvválasztási lehetőség
* Szélesebb körű statisztikák
* A statisztika táblázatokban a bejelentkezett felhasználó megjelenítése/kiemelése
* Több feature a főoldalon
* Barátlista
* Chat a barátok között
* Profilkép feltöltési lehetőség

# **MELLÉKLETEK**

# **IDÉZETT FORRÁSMUNKÁK**

1. Téma: Mi az az Angular?  
   Webcím: <https://angular.io/guide/what-is-angular>  
   Letöltés időpontja: 2023. 04. 21.
2. Téma: Mi az a MongoDB?  
   Webcím: <https://www.mongodb.com/what-is-mongodb>  
   Letöltés időpontja: 2023. 04. 21.
3. Téma: Miért a VS Code?  
   Webcím: <https://code.visualstudio.com/docs/editor/whyvscode>  
   Letöltés időpontja: 2023. 04. 21.
4. Téma: Mi az a TypeScript?  
   Webcím: https://www.typescriptlang.org/  
   Letöltés időpontja: 2023. 04. 21.
5. Téma: Mi az a .NET MAUI?  
   Webcím: <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0>  
   Letöltés időpontja: 2023. 04. 26.
6. Téma: ASP .NET Core Web Api  
   Webcím: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0>  
   Letöltés időpontja: 2023. 04.26.
7. Téma: Atlas - felhő alapú adatbázis  
   Webcím: <https://www.mongodb.com/atlas/database>  
   Letöltés időpontja: 2023. 04. 30.
8. Felhasználás: 4. ábra - Egy SCSS függvény és használata  
   Téma: Függvények  
   Webcím: <https://sass-lang.com/documentation/at-rules/function>  
   Letöltés időpontja: 2023. 04. 26. - Felhasználva: 4. ábra

1. Mi az a MongoDB? – Webcím: <https://www.mongodb.com/what-is-mongodb> [↑](#footnote-ref-1)
2. Miért válasszuk az ASP.NET Core-t? - Webcím: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0> [↑](#footnote-ref-2)
3. Mi az a .NET MAUI? – Webcím: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0 [↑](#footnote-ref-3)
4. Mi az az Angular? - Webcím: <https://angular.io/guide/what-is-angular> [↑](#footnote-ref-4)
5. Mi az a TypeScript? – Webcím: <https://www.typescriptlang.org/> [↑](#footnote-ref-5)
6. Miért a VS Code? – Webcím: <https://code.visualstudio.com/docs/editor/whyvscode> [↑](#footnote-ref-6)
7. Atlas - felhő alapú adatbázis – Webcím: https://www.mongodb.com/atlas/database [↑](#footnote-ref-7)