

Szent István Katolikus  
Technikum és Gimnázium

**Baráth Levente**

**Karsai Mihály Ferenc**

**Klubert Ádám**

**Videójáték terem**

**Szoftverfejlesztés és -tesztelés vizsgaremek**

Sátoraljaújhely, 2022

Tartalom

[Bevezetés 3](#_Toc184029950)

[Felhasználói dokumentáció 4](#_Toc184029951)

[Ajánlott gépigény 4](#_Toc184029952)

[Minimum gépigény: 4](#_Toc184029953)

# Bevezetés

A projekt elsődlegesen a videójáték termekkel rendelkező személyeknek, a retro - játékgyűjtőknek és minden ennek a témának érdeklőinek szól.

Ezeket az eszközöket eltárolhatják a weboldalon, és megnézhetik, hogy mivel rendelkeznek, vagy mire van szükségük.

A ritka cikkekkel való kereskedéshez és a játékról való beszélgetéshez fórumot hoztunk létre – itt megtekinthetik a játékkal kapcsolatos híreket.

Az oldalunkon található egy videójáték cégeket ismertető rész, amely a főoldalon található-ez átvezet az adott konzol és cég történetén és jelenlegi helyzetéhez. Az oldalunkon található egy fórum is, ahol a gyűjtők és az általános felhasználók egymással beszélhetnek és cserélhetnek-vehetnek egymástól. Itt elérhetőek a legújabb videójáték hírek is.

A felület legfontosabb része az úgynevezett raktár, ahol a felhasználók feltölthetik és letárolhatják a saját gépüket -konzoljukat.

**Miért választottuk ezt a témakört?**

A videojátékok korában nőttünk fel, és szeretünk megismerni kultúráját és eredetét.

Ennek a témának az ötlete Ádámtól származik, mert nagyon szereti a retro játékokat (régi és új játékokat is szívesen játszunk/tesztelünk), és úgy gondolta, hogy ezt az érdeklődést hasznosíthatjuk.

Úgy gondoljuk hogy még nincs olyan oldal ahol a felhaszálók egy oldalon feljegyezhetik a saját eszközeiket, az ez iránt érdeklődők megtalálhatják egymást,-cserélhetnek egymás között és el adhatják eszközeiket.

# Felhasználói dokumentáció

Ajánlott gépigény***:***

CPU: 4 magos processzor (2.5 GHz vagy gyorsabb)

RAM: 8 GB

Tárhely: 50-100 GB SSD (a gyors adatkezelés és képek tárolása érdekében)

Operációs rendszer: Windows

Böngésző: Google Chrome, Opera

Internet sebesség: Legalább 20 Mbps (a képek és a slideshow gyors betöltéséhez és zökkenőmentes élményhez)

## Minimum gépigény:

CPU: 2 magos processzor (2.0 GHz vagy gyorsabb)

RAM: 4 GB

Tárhely: Legalább 20 GB SSD (a gyorsabb betöltési idő érdekében)

Operációs rendszer: Windows

Böngésző: Google Chrome, Opera

Az alkalmazás/oldal indítása

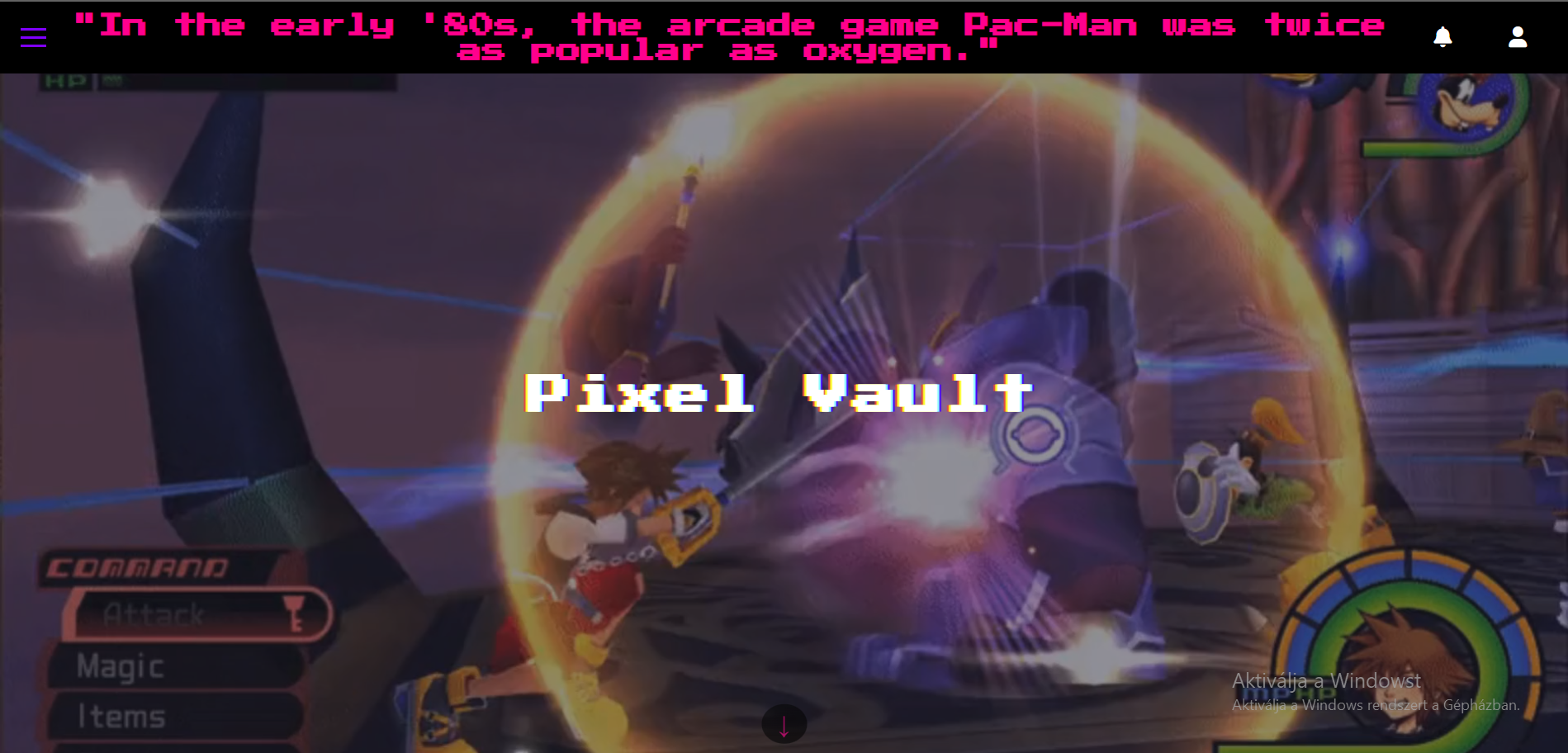
Az oldalt a felhasználó a számítógépen futó, már előre telepített böngészőben a <http://localhost:4200/> URL címen érheti el.

Ajánlott böngészők:

Google Chrome

Microsoft Edge

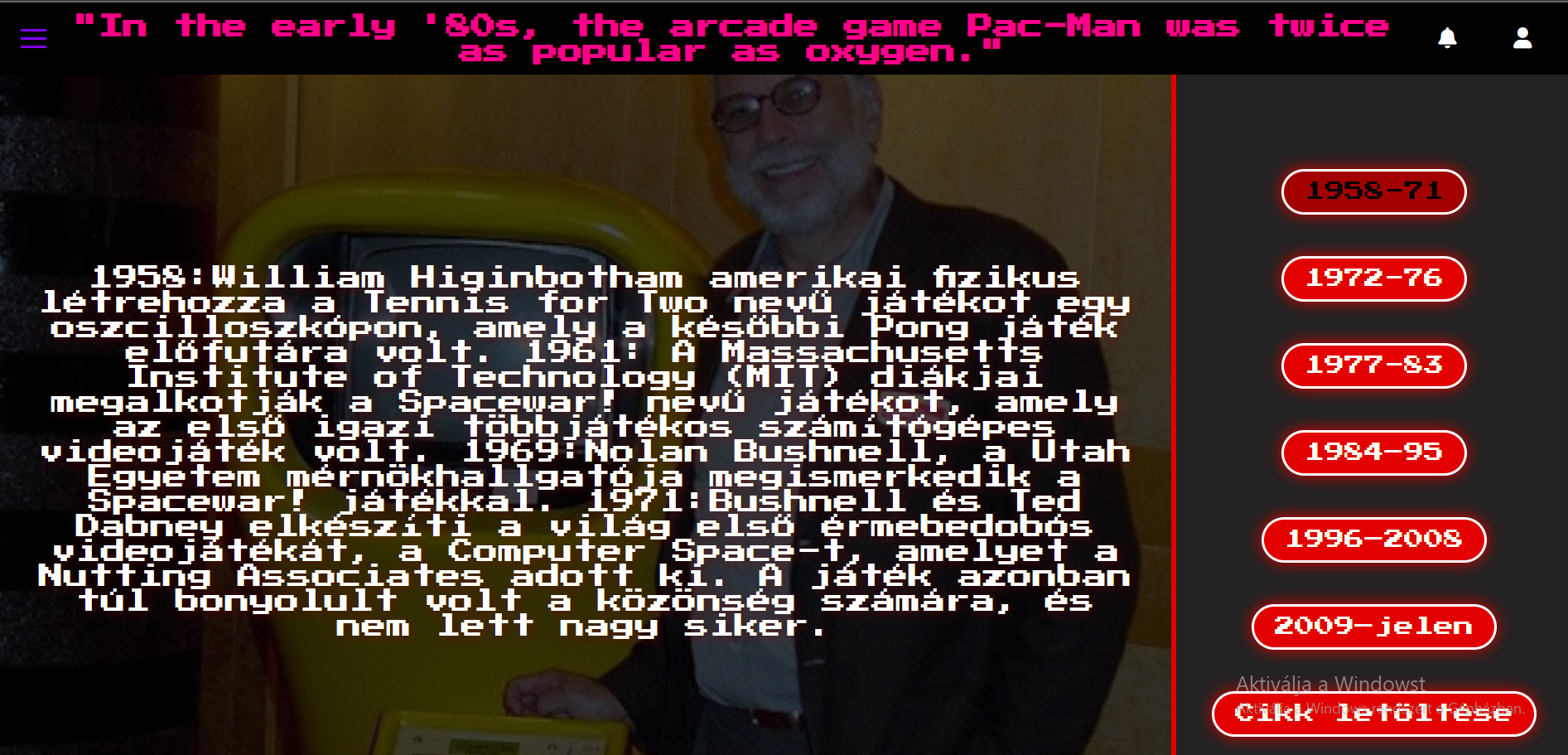
Főoldal

Az URL címen el juthatunk a Főoldalra, ahol egy navigációs sávval érhetjük el az oldal többi részét-funkcióit. A háttérben egy videójátékokból álló videó látható és a „Pixel Vault” felirat, amely az oldal neve. Ha lejjebb görgetünk, vagy az oldalon látható lefele mutató nyílra kattintunk akkor egy beviteli mezőt fogunk látni. Itt a videójáték-terem fenntartó/cég hirdetheti a helyét adatok és kép feltöltésével.  


Konzol ismertető

Ezen az oldalon elérhetőek az adott cégek/márkák története. Ha az adott „kártyára” kattintunk, akkor egy idővonallal találjuk szemben magunkat. Minden egyes évszámra kattintva az adott év történéseit olvashatjuk el, miközben a háttérben egy ahhoz tartozó képet látunk.





Az évszámok alatt van egy cikk letöltése gomb is, amivel az adott tartalmat egy pdf formátumként letölthetünk. A cikkben plusz információk is vannak és itt az összes forrás is megtalálható.



# Fejlesztői Dokumentáció

Backend és Adatbázis Ismertető

A web-applikáció alapját szolgáló adatbázis, és backend-szerver elkészítésének feladata rám esett (Karsai Mihály Ferenc-re). A dokumentum alábbi részében részletesen át fogom vezetni az olvasót, a fent említett rendszer működésén, felépítésén, miközben elmagyarázom a gondolatmenetemet a rendszer kiépítése közben. A dokumentum ezen részének olvasásához és megértéséhez ajánlott a webfejlesztés alapvető ismerete.

Az elkészített Backend architektúrája követi a tradicionális fejlesztési szokásokat, az MVC modellt; Az egész backend struktúra egy „Backend” nevű mappában van elhelyezve, ahol „Controllers”, „Routers”, és „Models” mappák használatával tiszta, és fenntartható kóddal kivitelezzük a szerverrészek közötti kommunikációt, és a kérések küldését, fogadását, az adatbázis fenntartását, abban végzett módosítások végrehajtását és mentését.

# „Response” mappa

A legegyszerűbb fontosabb mappa, amit bővebben ki tudnék fejteni, az a „Response” nevű ’Válasz’ mappa lenne. Ez talán az egyetlen, ami nem konvencionális a struktúránk felépítésében, de a működése egészen egyszerű, használata kényelmes. Minden kérés küldésekor generálódik egy siker, vagy egy hiba üzenet, valamint egy hibakód is. Ez alapján tudunk a felhasználónak visszajelzést adni a kérésük eredményéről.

Viszont a fejlesztés során észrevettem, hogy a „Controllers” mappa, ahol a kéréseket, és lekérdezéseket kezeljük egyre-egyre nehezebben fenntartható volt. Egy dolog ami különösen kitűnt a tömegből az volt, hogy minden alkalommal amikor új választ írtam (vagy siker, vagy hiba kóddal és üzenettel) új hibaüzenetet kellett írni minden esetre, amiket többször újra kellett használni. Például: „Adatok lekérdezése sikertelen volt.”, „Bejelentkezés sikeres volt.”, „Ismeretlen hiba történt.”, stb. Nagyon hamar kialakultak ezeknek az üzeneteknek különböző verzióik, kisebb elírások, kicsit eltérő szóhasználatok, stb, mivel mindegyik hiba, és siker üzenetet abban a pontban írtam meg, amikor a kérést is.

Ennek a procedúrának a lebonyolításához, egyszerűsítéséhez és konzisztensebbé tevése igényében a „Response” mappa egy msg.js és egy uzn.js file-t tartalmaz. A(z) msg.js (message) az angol hiba/siker üzenetek fenntartásáért felelős, amíg az uzn.js (üzenet) a magyar verzióért felelős. A felépítésük ugyanaz, hogy a controller file-okban elég legyen beimportálni az uzn.js-t és a(z) msg.js-t, majd hasonló módon lekérdezni. A hiba/siker üzenetek így megváltoznak:  
  
Az alábbiról:  
„message”: „An unknown error has occurred.”;  
„üzenet”: „Egy ismeretlen hiba merült fel.”;  
  
A következőre:  
„message”: msg.user.failure.unknown;  
„üzenet”: uzn.user.failure.unknown;

Az alábbi megoldás sokkal gyorsabbá tette a hiba/siker üzenet küldési folyamatot.

# „Models” Mappa

A következő, szintén egyszerű mappa a „Models”, modellek mappa, ami az adatbázis tábláit tartalmazza. A backend során egyetlen egy sor SQL kódot sem írtam, hanem sequelize, és mysql2 modulokkal konvertáltam a javascript kódot modulokon belül futási időben SQL-é. A Models mappa tehát csak javascript-esített SQL táblákat tárol a „project” adatbázishoz. A következőkben részletezem a különböző táblák felépítését.

# Users.js

Az első a Users.js model, ami a felhasználó adatait tárolja. 3 alapvető adata van (az id-n kívülÍ) egy felhasználónak, a neve, az e-mail címe, és a jelszavának a hash-e (vagyis a kódolva eltárolt jelszava). Ebből mindhárom kötelezően kitöltendő mező. Az e-mail címnek követnie kell a megszokott e-mail cím formátumot, amit a modul kezel is (pl. [pistike123@gmail.hu](mailto:pistike123@gmail.hu)), ez a mező „unique” (egyedi), tehát nem szerepelhet az adatbázisban két felhasználó ugyanazzal az e-mail címmel. Hasonlóan egyedi a felhasználónév is, ami az e-mail címhez hasonlóan használható bejelentkezéshez.

A kötelező mezőkön kívül tárolunk még opcionális adatokat. Például minden felhasználónak van egy isAdmin (admin-e) mezeje, ami alapvetően hamis, de szerver-oldalról bárki adminná tehető. Ezen kívül életkort, és telefonszámot is tárolunk, amit a felhasználó saját maga állít be magának, ha akar. Ez az utóbbi két mező tulajdonképpen jelentéktelen, mivel az oldal egyik része sem változik meg ezeknek az adatoknak a módosulásán.

Érdekesség megjegyezni, hogy nekünk sok hibánk akadt abból, hogy a felhasználó táblában (és később a céges táblában is) a jelszót, mint passwordHash tároltam, és kéréseknél is passwordHash argumentumot várt a backend controllerem, ami sok fejfájást okozott a middle-end-en dolgozó személyünknek.

# Token.js

Ez talán a legegyszerűbb tábla az adatbázisban, egyszerűen a felhasználók által bejelentkezéshez használt tokeneket tárolja. A tokenek a szerver által (jsonwebtoken modullal) generált hosszú kódsorok, ami amíg él a felhasználó bejelentkezve marad. Nálunk jelenleg (ha az olvasás idejéig meg nem változik) 2 óráig tart egy felhasználói token.

# Consoles.js, ArcadeMachines.js, PinballMachines.js

Ezt a hármat egybe vettem, mert a szerepük hasonló. A játékgépeket külön táblákba tároljuk típusuktól függően (nem mintha túl nagy jelentősége lenne a típusuknak, csak egészen egyszerűen így lettek kigyűjtve eredetileg az adatok, tehát különböző táblákra szedtem szét őket.). Itt olyan adatokat tárolunk, mint a név, kiadó, kiadási év, és még esetleg „genre”-t (játék típust).

Ezeket a raktár rész lekérdezi, és kilistázza, valamint hozzá lehet fűzni felhasználókhoz, kb mint egy „könyvtár” rendszer, amivel felhasználók tudják jelezni, hogy ez megvan nekik, vagy esetleg, hogy ezt akarják.