

Tolna Vármegyei Szakképzési Centrum
Tolna Vármegyei SzC Apáczai Csere János Technikum és
Kollégium

Vizsgaremek

Készítették: Kéri Bence, Tóth Milán, Váradi Barnabás

Dombóvár

2023 - 2024

Tolna Vármegyei Szakképzési Centrum
Tolna Vármegyei SzC Apáczai Csere János Technikum és
Kollégium

Szakma megnevezése: Szoftverfejlesztő és tesztelő

A szakma azonosító száma: 5 0613 12 03

Vizsgaremek

Webáruház videójátékoknak

Készítették: Kéri Bence, Tóth Milán, Váradi Barnabás

Dombóvár

2023 - 2024

Tartalomjegyzék

I. Bevezető, a feladat rövid ismertetése	4
II. A felhasználói dokumentáció	
1. A program általános specifikációja	5
2. Rendszerkövetelmények	6
3. A program telepítése	7
4. A program használatának a részletes leírása	7
III. A fejlesztői dokumentáció	
1. Témaválasztás indoklása	12
2. Az alkalmazott fejlesztői eszközök	13
3. Tervezési módszer	13
4. Adatmodell leírása	13
5. Részletes feladat-specifikáció, algoritmusok	16
6. Forráskód	19
7. Tesztelési dokumentáció	20
8. Továbbfejlesztési lehetőségek	20
IV. Összegzés	20

I. Bevezető, a feladat rövid ismertetése

Hosszas gondolkodás után egy olyan webáruházat álmodtunk meg, ami különböző játékok megvásárlására ad lehetőséget.

Az internet elterjedésével emelkedett az online vásárlások száma. A 2000-es években kezdte hódító útját az Ebay és az Amazon is. Nem muszáj ilyen messzire mennünk, elég ha csak a 2019-es évet nézzük, amikor kitört a Covid járvány és rengeteg bevásárlóközpont és üzlet zárt be hosszú időre. Mindenki otthon maradt, ennek köszönhetően rengeteg vállalkozás csődbe ment, akik nem rendelkeztek webáruházzal.

A webáruházak előnye, hogy otthonról, néhány kattintással megvehettünk bármit, bármikor, ahelyett, hogy elutaznánk az üzlethez és kivárnánk a sorunkat. Egyszerűbb, gyorsabb és átláthatóbb vásárlást eredményez, ami jó a felhasználónak és az áruháznak. A webáruházak alacsony üzemeltetési költségei miatt versenyképes árakkal rendelkeznek, így a felhasználó olcsóbban jut hozzá egy termékhez. A webáruházakban személyre szabott ajánlatok várnak, mivel nyomon követik a vásárlási szokásainkat.

A projektben sok nehézséget küzdöttünk le, mivel az egyszerűnek hangzó dolgokat néha bonyolult megvalósítani. Így néhány dologról le kellett mondanunk, például a bankkártyás fizetésről, különböző bejelentkezési lehetőségekről (facebook, google) és idő hiányában a sötét/világos/színvak témáról is. A munkánkat nehezítette, hogy nem mindenki rendelkezett elég tapasztalattal a feladatokhoz, így a hibáinkból tanultunk. Ráadásul több olyan eleme volt a projektünknek, amivel még nem találkoztunk, így saját magunknak kellett mindennek utánanézni, ami rengeteg idővel és munkával járt.

A kommunikáció hiányos volt az elején, de idővel fejlődött és jobb lett. A csapatmunkát erősítette, hogy nem volt komolyabb konfliktus közöttünk, ha volt is, mindig meg tudtuk beszélni. Mindenki megcsinálta a rábízott feladatot és amikor szükségünk volt, segítettünk egymásnak. A távolság leküzdése érdekében Discordon kommunikáltunk.

A munkamegosztásra hatással volt a lemaradásunk, amit sikeresen leküzdöttünk.

Kéri Bence

- Adatbázis
- Jira (11.10-ig)
- Volt projekt manager
- *Frontend (HTML, CSS), amikor be kellett hoznunk a lemaradásunkat*

Tóth Milán

- Jira (11.14-től)
- Jelenlegi project manager
- Backend (JAX-RS, Postman, Wildfly)
- Verziókezelés (Github)
- *Frontend (HTML, CSS, JavaScript), amikor be kellett hoznunk a lemaradásunkat*
- *Designer (Figma), amikor be kellett hoznunk a lemaradásunkat*

Váradi Barnabás

- Figma
- Frontend (HTML, CSS, JavaScript, Postman)
- Designer (Figma)

II. A felhasználói dokumentáció

1. A program általános specifikációja

Webáruházunk célja a videójátékok értékesítése különböző platformokra. Az oldal regisztrációt kínál, amely lehetővé teszi a felhasználók számára, hogy nyomon kövessék rendeléseiket. Az oldal főbb funkciói között szerepel a kosár, szűrők és rendezési lehetőségek, valamint az admin felület, ahol új játékokat lehet hozzáadni, meglévőket módosítani vagy törölni.

A főoldal keresőmezővel rendelkezik, amely azonnal megjeleníti a keresett játékokat, a kattintás után pedig a termékek oldalra irányít. A felhasználóknak lehetőségük van bejelentkezésre és regisztrációra, valamint egy felhasználói felületre, ahol módosíthatják személyes adataikat vagy törölhetik fiókukat. A

főoldalon egy forgóképes galéria jelenik meg, amely három véletlenszerűen kiválasztott játékot mutat be. Minden játék adatai között szerepel a kép, név, ár, akció esetén az eredeti ár, és egy "hozzáadás a kosárhoz" gomb.

A kosárban a felhasználók láthatják és törölhetik a kiválasztott játékokat, majd a "rendelés leadása" gomb megnyomásával leadhatják rendelésüket. A rendelés leadása után megjelenik egy üzenet és a rendelés az adatbázisban is rögzítésre kerül. A játék oldal részletes információkat nyújt egy kiválasztott játékról, beleértve a képet, nevet, eszközt, platformot, korhatárt, árakat és elérhetőségi állapotot.

Minden oldalon elérhető a keresőmező, bejelentkezés és regisztráció gomb, valamint a felhasználó lenyíló fül. A lábléc az oldalt készítő tagok neveit tartalmazza. Az oldalon található egyéb funkciók közé tartozik a beállítások, amelyek a felhasználót a saját oldalára navigálják, valamint a kijelentkezés, amely törli a tokent a localStorage-ból.

2. Rendszerkövetelmények

Hardver

Olyan számítógép, amely képes futtatni a következő programokat, szoftvereket:

- phpMyAdmin 5.1.1
- mysql 15.1
- Wildfly 26.1.1.Final
- Apache Netbeans 20
- Java 17.0.6 LTS

Windows operációs rendszerű számítógépeken a java 17 miatt a windows 10 futtatásához szükséges hardverek felelnek meg.

Processzor	1 Ghz vagy annál gyorsabb
Memória	1GB/32 bit, 2GB/64 bites
Háttértár	16GB/ 32bit 20GB/64 bit
Grafikus kártya	DirectX 9 vagy annál újabb
Kijelző	800x600 felbontás

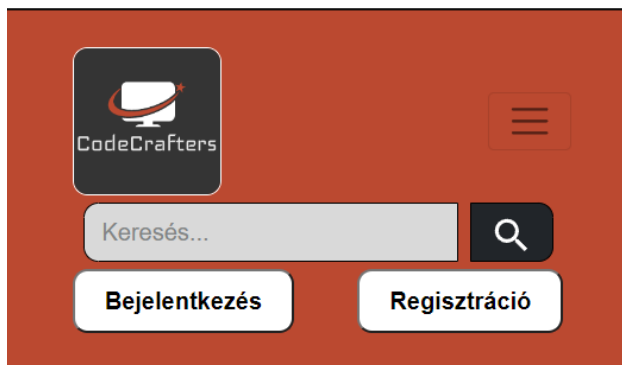
Szoftver

Windows operációs rendszerű számítógépeken a java 17 miatt a minimum Windows 10.

3. A program telepítése

A webalkalmazás használatához csak egy böngésző és megfelelő internetkapcsolat szükséges.

4. A program használatának részletes leírása



Ha megérkezünk az oldalra akkor a főoldal jelenik meg. A megjelenés változik attól függően hogy milyen eszközön van. A weboldalhoz használtam Bootstrapet a verziója v5.0.2. A navbar és a keresőhöz a bootstrap oldalán megtalálható navbárt és keresőt használok.

A kereső ha rá kattintasz a keresés gombra akkor megnyitja a termékek oldalt. Ha nincs az input mezőben semmi akkor mindent megjelenít kivétel ha nem írtál bele. Ha írtál valamit a input mezőbe akkor az lesz eltárolva a localStorage-ba és után megnyílik az termékek oldal a **window.location.href**-el.

A regisztráció és bejelentkezés gomb ezen gombok addig látszódnak amíg nem vagy bejelentkezve. És mindkettő át visz egy oldalra.

Felhasználónév

Vezetéknév Keresztnév

Email

Jelszó

Jelszó Újra

☐ Egyetértek a Felhasználási feltételekkel

Regisztrálj

Már van fiókja? [Belépés](#)

A regisztráció oldalon. Minden mezőt kötelező kitölteni.

És a felhasználási feltételek mutat egy linkre ami megnyitja a felhasználási feltételek oldalt.

A jelszó minimum követelményei:

Legyen benne kis- nagy betű, speciális karakter, 8 hosszú

és az e-mailben kell

szerepelni-e kell @-nak.

Felhasználási feltételek

Regisztráció
A webshop szolgáltatásainak igénybevétele előtt regisztráció szükséges. A webshop szolgáltatásainak igénybevétele előtt a felhasználónak kell fogadnia a felhasználási feltételeket.

Felhasználói adatok
A regisztrációt követően a személyes adatokat bizalmasan kezeljük. Személyes adatnak tekintjük a felhasználó nevét, email címét és jelszavát.

Hozzáférés
A felhasználó felelőssége a személyes fiókjának biztonságát megőrizni.

Kötelező adatok
A regisztráció során kötelező adatok kitöltése szükséges a szolgáltatásokhoz való hozzáféréshez.

Kijelentkezés
A felhasználó köteles biztonságosan kijelentkezni a fiókjából a használat befejezésekor.

Jogosultságok
Az adminisztrátor jogosultságokkal rendelkező felhasználója az oldal tartalmának kezeléséért.

Elérhetőség
A webshop nem vállal felelősséget az esetleges szolgáltatások kieséseikért vagy technikai problémákért eredő kellemetlenségeikért.

Felhasználási feltételek változása
A webshop fenntartja a jogot a felhasználási feltételek egyoldali módosítására, melyről értesítést küld a felhasználóknak.

Fetchet használtam a regisztrációhoz (Post).

Sikeres regisztráció vagy bejelentkezés esetén a főoldalra kerülünk a javascript segítségével. Ami ugyanabban az ablakban nyílik meg.

Felhasználónév

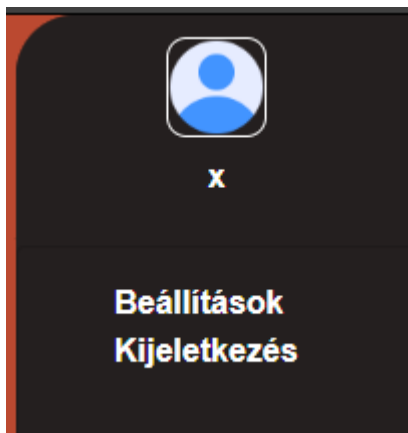
Jelszó

Bejelentkezés

Még nincs fiókja? [Regisztráljon itt.](#)

Ha a Bejelentkezésre kattintunk vagy a bejelentkezés gombra a navbár-ban akkor láthatunk egy Bejelentkezési oldalt. Itt szintén fetchet használtam és Post-ot. Viszont itt a visszakapott adatait a felhasználónak localStorage tárolom.

Ha be vagyunk már jelentkezve akkor a bejelentkezés és regisztráció gomb eltűnik és helyette a felhasználó profil jelenik meg.

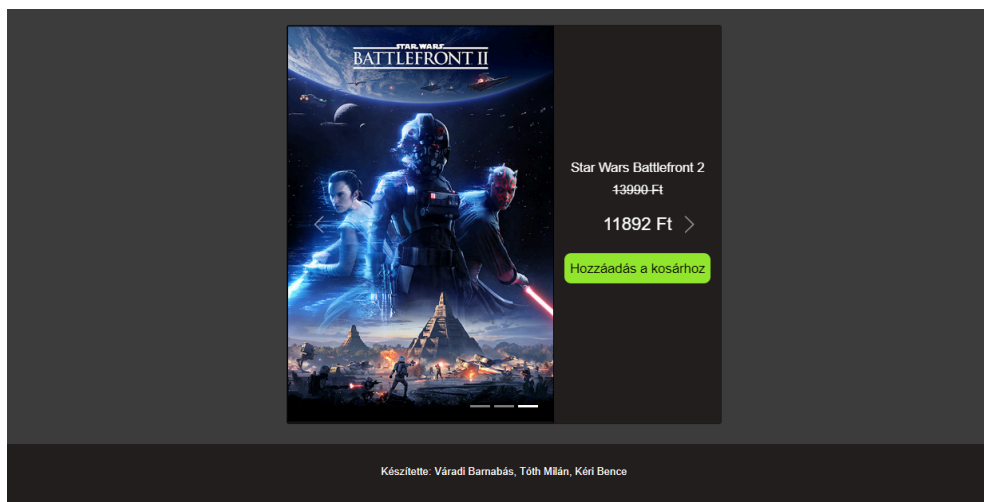


A felhasználó fülben láthatjuk a profilképünk és a nevünk.

Ha rákattintunk akkor két opció van beállítások amin szerkeszthetjük a felhasználó adatait vagy töröljük a magát a felhasználót. A másik opció pedig a kijelentkezés.

Amire ha rákattintunk törli a localStorage-ből a felhasználót.

És a felület eltűnik.



A főoldalra a navbár alatt láthatunk egy carousel-t és egy láblécet. A carousel javascript segítségével 3 random játékot betölt és ezeket lehet látni. Ha

Rákattintunk a hozzáadás gombra akkor a terméket hozzáadjuk a gomb value-ja meg változik kosárban-ra és disabled lesz. A carousel-hez szintén használok fetchet ami egy get kérés.

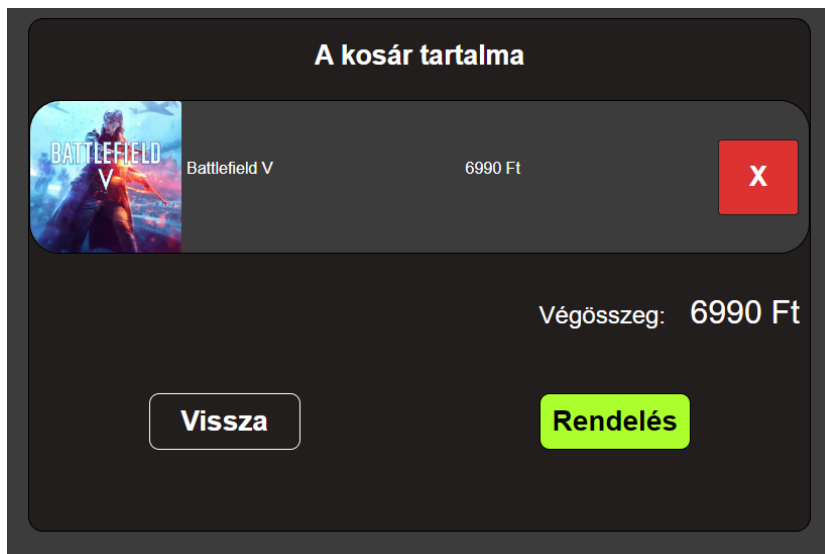
A carousel tartalmaz card elemet amelyekben benne van a játékhoz tartozó kép ami egy link is egyben ami elvezet az adott játék oldalra ami így nézne ki:

```
<a href="/jatek/jatek.html?id=${adat[i].id}">
```

És persze név, ár ami át van húzva ha van akciós ár, akciós ár és egy hozzáadás gomb ami hozzáadja a terméket a kosárhoz.



Ha erre rákattintunk akkor megnyílik a kosároltal.



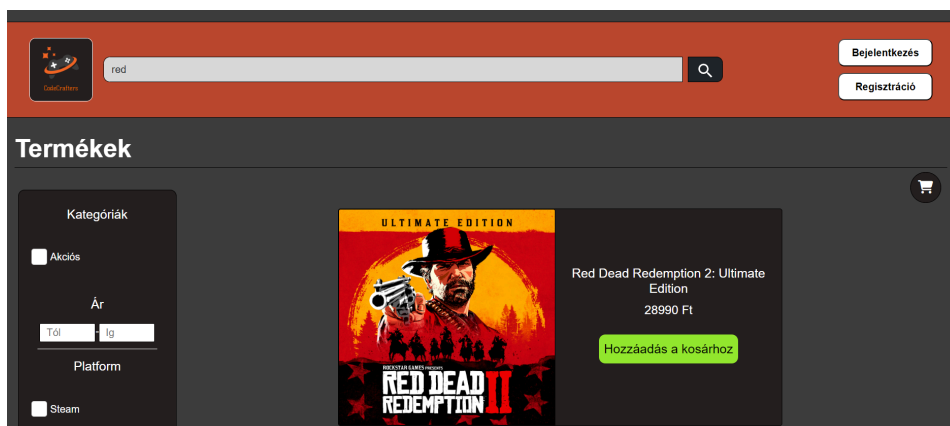
A kosár oldalon láthatunk egy vissza gombot ez a gomb hivatkozik előzmények alapján az előző oldalra

(javascript:history.back()).

A kosár tartalmát egy get kérés ami az oldal betöltése után és a törlés után van meghívva újra. A törlés gomb egy piros alapon fehér x. Ha rákattintunk a törlésre akkor azt az elemet törli a kosárból az id alapján amit a get kérésnél kapunk.

A végösszeget a fronted végzi a minden kapott termék amit a get kérésnél kapunk van egy egy ár ami annak függvényében változik hogy akciós-e a termék. Ezután csak összeadjuk az összesnek az árát.

A termékek oldalon van szintén egy fetch és minden játékot fetchel és egy for ciklussal



az adatokat át adjuk egy függvénynek ami elhelyezi a megfelelő részre a az adatokat és utána vissza adja a card-ot egy returnal és bele helyezi a htmlbe inner

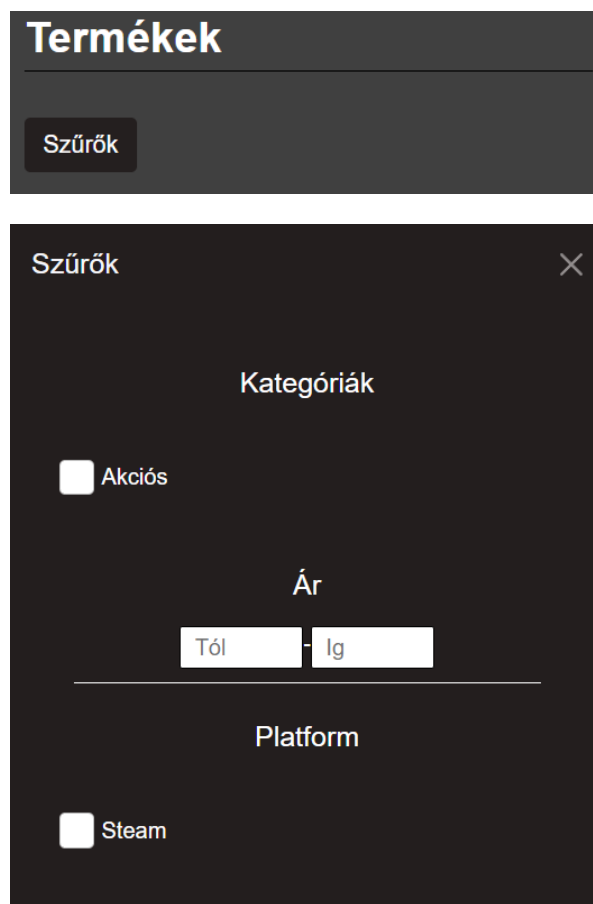
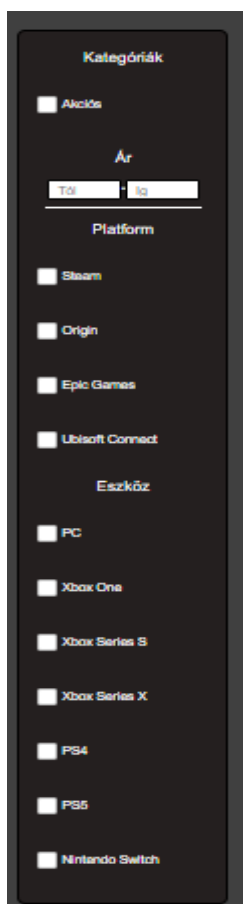
html segítségével.

A card felépítése hasonló a carusel-ben lévőhöz.

```
function createCard(kepPath, nev, ar, akcio, id, url) {
  const akciosAr = akcio > 0 ? Math.round(ar - (ar / 100) * akcio) : null;

  return `
    <div class="card my-4 flex-column flex-lg-row">
      <a href="${url}?id=${id}"></a>
      <div class="card-body">
        <h5 class="card-title">${nev}</h5>
        <p class="card-text ar">${
          akciosAr > 0 ? `<del>${ar} Ft</del>` : `${ar} Ft`
        }</p>
        ${
          akciosAr > 0
            ? `<p class="card-text akciosar">${akciosAr} Ft</p>`
            : ""
        }
        <input id="hozzadas" class="my-2 p-2 btn btn-success fs-5" type="button" value="Hozzáadás a kosárhoz" />
      </div>
    </div>
  `;
}
```

A szűrők mobilon egy gomb segítségével jelenik meg off-canvasból.



Nagyobb képernyőn viszont gomb nélkül is látható.

A képernyő mérete alapján változik hogy melyiket látjuk.

A felhasználó oldalon láthatunk egy form-ot mint a regisztráció vagy bejelentkezés esetében.

A dark-themed user profile form. At the top left is a circular profile picture placeholder with a blue and white icon. Below it are two small 'x' and 'xx' labels. The form contains several input fields: 'Felhasználónév' (Username) with a single 'x' placeholder, 'Vezetéknév' (Surname) and 'Keresztnév' (First Name) each with a single 'x' placeholder, 'Email' with an 'x@' placeholder, 'Jelszó' (Password) and 'Jelszó Újra' (Password Again) both empty. At the bottom are two buttons: a green 'Mentés' (Save) button and a red 'Törlés' (Delete) button.

Láthatjuk a Felhasználó nevünket mind a profilba mind

placeholder-be az inputba. Ugyanígy a vezetéknév és keresztnévet is. Ugyanakkor az emailt látjuk de nem módosíthatjuk. A jelszónak itt is egyeznie kell mint a regisztrációnál. A mentés gombbal mentjük a változtatásokat. A törléssel pedig töröljük a felhasználót.

III. A fejlesztői dokumentáció

1. Témaválasztás indoklása

A fő szempont a KKK elvárásai voltak, emellett a hozzánk legközelebb álló témát helyeztük előtérbe. Az áruházunk RESTful és az adatokon CRUD műveletek is képes végrehajtani.

Webáruházunk adatbázist használ a játékok, rendelések, valamint a felhasználóhoz köthető adatok tárolása érdekében. A backend szolgáltatásait a Java

alapú JAX-RS biztosítja, ami kommunikál a frontenddel és az adatbázissal. A Java egy erősen típusos nyelv, ami megkönnyíti a hibakezelést, így, biztonságos és stabil működést biztosít. A frontendhez nem használtunk keretrendszert, így Vanilla JavaScript-el dolgoztunk, az adatbázis pedig mysql nyelven íródott.

Fontosnak tartottuk a reszponzivitást, amihez a Bootstrap keretrendszert használtuk, így telefonon az elemek átrendeződnek a könnyű navigáció és az olvashatóság érdekében.

2. Az alkalmazott fejlesztői eszközök

- Programnyelvek
 - Frontend: JavaScript
 - Adatbázis: MySql
 - Backend: Java
- Fejlesztői környezet
 - Frontend: Visual Studio Code, PostMan, Figma (design)
 - Adatbázis: MAMP (PHPMyAdmin), diagrams.net
 - Backend: Netbeans, PostMan, WildFly szerver
 - Mindegyik: Git, GitHub Desktop

3. Tervezési módszer

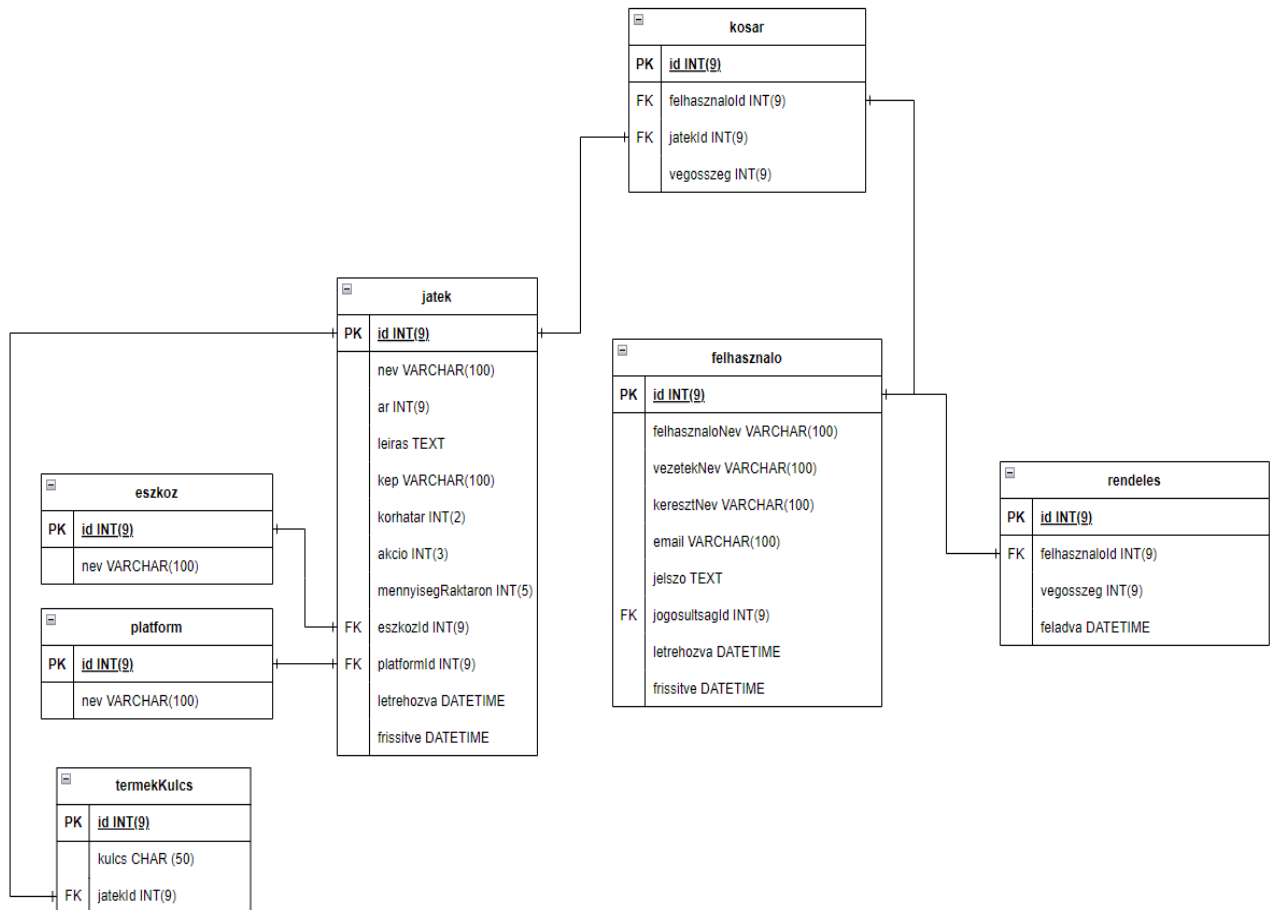
A projekt tervezése során figyelembe vettük, hogy a projektünk minden szegmense alkalmazkodjon a vállalati igényekhez. A frontendben az elsődleges szempont a reszponzivitás volt, így a Bootstrap 5 keretrendszert használtuk. A backend REST alapú szolgáltatásokat nyújt, aminek létrehozásához a JAX-RS keretrendszer volt a legalkalmasabb. Az adatbázisban relációkat használunk, így a MySQL használatára esett a választásunk.

4. Adatmodell leírása

Az adatbázis tervezéshez a diagrams nevű, diagram készítő oldalt választottam. Ezzel a diagram szerkesztővel könnyen átalakíthattam és

rendezhettem a táblákat, oszlopokat és kapcsolatokat. A következő ábrán látható a teljes adatbázis az adattáblák összekapcsolásával.

Az alábbiakban részletezem a fenti ábra diagramjának részeit és adatbázis szerkezetét.



Mindenhol próbáltam egyértelmű, egyszerű és rövid elnevezéseket használni. Az adatbázist „CodeCraftersDB”-nek neveztem el, így egyértelműsítve, hogy ehhez a projekthez tartozik. A csapat többi tagjával abban egyeztünk meg, hogy a projekt magyar nyelvű legyen, ezért az adatbázisban a táblák, oszlopok és tárolt eljárások magyarul vannak elnevezve. Amint az ábra alapján látható Camel case-t használtam az elnevezésekhez. Több helyen is előfordulhatnak azonos oszlopnevek, de csak ha ugyanazt a szerepet töltik be.

Természetesen minden táblán található elsődleges kulcs, amit „id”-nak neveztem el. Ez a mező szám típusú és maximum 9 karakter hosszú lehet. A táblák közötti kapcsolatokat, ha egy másik táblában szereplő idegen kulcsként, mindig a tábla neve és „id” alapján neveztem el. Például: „felhasznaloid”.

Az első és legfontosabb tábla a „users”, ebben a felhasználók adatait tároljuk. A nevet két részben, vezetéknev és keresztnévként rögzítem. Mindkettőt szöveges formában tárolom el és legfeljebb 100-100 karakter lehet. A felhasználónak még el van tárolva az oldalon lévő felhasználó neve is, ami lehet akármi 100 karakter nagyságig. Szöveges formában van még tárolva az E-mail cím, szintén 100 karakter nagyságú. A felhasználó jelszava SHA lett titkosítva, ami az egyik legbiztonságosabb titkosítási algoritmus, ezt text ként tároljuk. A felhasználók jogosultságát a „jogosultsagld” alapján kezeli az adatbázis, a backed és a frontend is, ez egy karakteres szám formájú: 1 - általános felhasználó, 2 - admin. Időbélyegeket is tárolok, például a „letrehozva” és a „frissitve”.

A „felhasznalo” tábla id-ja több táblával is kapcsolatban van: „kosar - felhasznalold”, „rendeles - felhasznalold”.

A „jatek” tábla, mint products táblaként szolgál. Itt vannak eltárolva a termékek, amiket az oldalon lehet megtekinteni és megvenni. Található a táblán „nev” és „kep” szöveg típusú 100 karakter nagyságúak. A „kep”-ben csak a kép neve van eltárolva, maga a kép frontenden van. Az „ar” (9 karakter hosszú lehet) és „akcio” (3 karakter hosszúságú maximum) szám formában van eltárolva, az „akcio”-ba bevitt adatot, mint százalékot vonjuk ki az „ar”-ból és jelenítjük meg frontenden. Még szám formában tároltam el a „korhatar” és „mennyisegRaktaron” adatokat. Leírás is tartozik minden játékoz, ehhez van a „leiras”, ami textként van eltárolva.

A „jatek” tábla „eszkozld” az „eszkosz” tábla egyik sorára hivatkozik, ugyanúgy a „platformld”, ami a „platform” tábla egy sorára hivatkozik. A „jatek” táblán is tárolok időbélyegeket, például a „letrehozva” és a „frissitve”.

A „jatek” tábla id-ja kapcsolatban van a „termekKulcsok - jatekld” és a „kosar - jatekld” sorokkal.

A „platform” és az „eszkosz” egyszerű táblák, csak egy 100 karakter hosszú szöveges „nev” található bennük.

A termékkulcsot a „termekKulcs” táblában helyeztem el. Itt a „kulcs” szöveg. A kapcsolat a „jatek” táblával arra használok, hogy a „mennyisegRaktaron” sort frissítsem, ami frontenden jelezve van, hogy egy termék elérhető vagy nem.

A „kosar” táblában elmentésre kerül a „jatekld”, amit a felhasználó hozzá tud adni és törölni is tudja, a „vegosszeg”-ben elmentésre kerül az adott játéknak az ára, amit a „jatek” tábla „ar” és „akcio” sorából számolok. Minden kosár egy felhasználóhoz van rendelve a „felhasznalold” sor segítségével.

A „rendeles” tábla „vegosszeg” sorába mentésre kerül a „kosar” táblából a felhasználóhoz rendel összes termék ára. A „feladva” időbélyeg, amiben elmentődik a rendelés feladásnak ideje. Ez a tábla hozzá van kapcsolva a „felhasznalo” táblához a „felhasznalold” sorral.

Az adatbázisban admin jogosultságú felhasználókat csak az adatbázisból manuálisan lehet felvenni, ez a biztonságot szolgálja. A jelszó titkosított és front- és backenden is ellenőrizzük a felhasználó jogosultságát, így szinte lehetetlen bejutni az adatbázisba.

5. Részletes feladatspecifikáció, algoritmusok

Minden webalkalmazás fontos eleme a backend, mivel ő a felelős az adatok kezeléséért. A backend segítségével különböző műveleteket hajthatunk végre az adatokon. Ebben a részben a backend szerepét fogom bemutatni.

Minden egyes HTTP kéréskor a CORS szabályok döntenek el, hogy a frontend mit kaphat a backendtől. A szabályok megváltoztatásához különböző fejléc mezőket kell hozzáadni a kérésekhez. Ezek a fejléc értékek képesek blokkolni felhasználókat a DDOS támadás elkerülése érdekében, emellett meghatározhatjuk, hogy a kliens milyen HTTP kérést futtathat a szerveren és hogy milyen fejlécet fogad el.

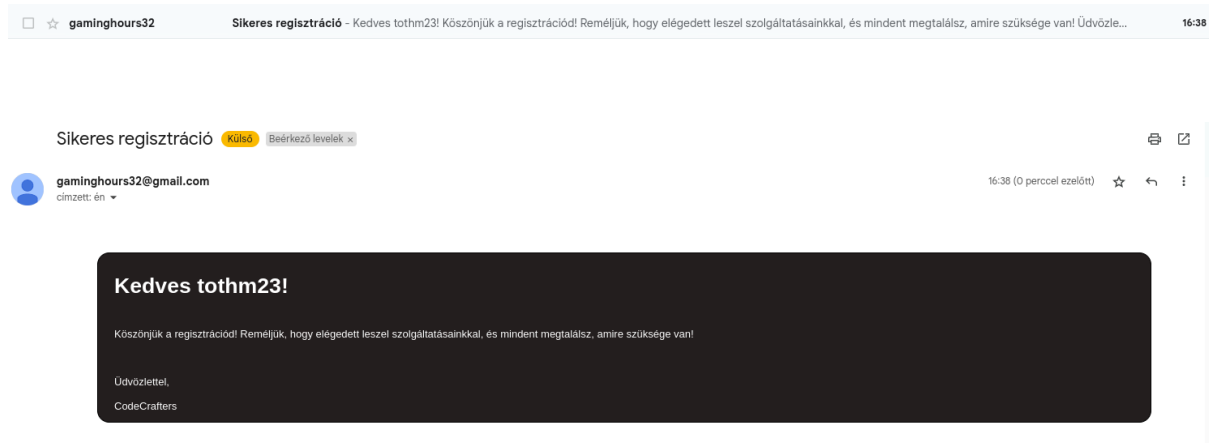
```
/**
 *
 * @author tothm23
 */
@Provider
public class CorsFilter implements ContainerResponseFilter {

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws IOException {
        responseContext.getHeaders().add(key: "Access-Control-Allow-Origin", value: "*");
        responseContext.getHeaders().add(key: "Access-Control-Allow-Methods", value: "GET, POST, PUT, DELETE");
        responseContext.getHeaders().add(key: "Access-Control-Allow-Headers", value: "*");
    }
}
```

Minden webáruház alapvető eleme a belépés és a regisztráció. A regisztrációban egy űrlapot kell kitöltenie a felhasználónak, ahol megadja adatait és elfogadja a felhasználási feltételeket. A frontend küld egy HTTP kérést a JAX-RS web alkalmazás egyik végpontjához. Vanilla Javascriptet használ, a fetch API segítségével.

Az alkalmazás „felhasznalo” végpontja feldolgozza a kérést és kommunikál az adatbázissal. Ha a felhasználó adatai rendben vannak, akkor a felhasználó az

adatbázisban rögzítésre kerül. Ezt követően kap egy emailt az általa megadott email címre.



Az adatok validálása backend szinten lett megvalósítva, amik a felhasználó entitás osztályban szerepelnek. Például a felhasználó csak érvényes email címmel regisztrálhat és a jelszavának meg kell felelnie a követelményeknek (minimum 8 karakter, tartalmaznia kell számot, betűt, speciális karaktert).

```
public static boolean jelszoEllenorzes(String jelszo) throws FelhasznaloException {  
    boolean tartalmazSzamot = false;  
    boolean tartalmazBetut = false;  
    boolean tartalmazSpecialiskaraktert = false;  
  
    for (char c : jelszo.toCharArray()) {  
        if (Character.isDigit(ch: c)) {  
            tartalmazSzamot = true;  
        } else if (Character.isLetter(ch: c)) {  
            tartalmazBetut = true;  
        } else if (!Character.isLetterOrDigit(ch: c)) {  
            tartalmazSpecialiskaraktert = true;  
        }  
    }  
  
    if (jelszo.equals(anObject: "")) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszava lehet üres!");  
    } else if (jelszo.length() > 100) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszava nem lehet 100 karakternél hosszabb!");  
    } else if (jelszo.length() < 8) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszava nem lehet 8 karakternél rövidebb!");  
    } else if (!tartalmazSzamot) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszavának tartalmaznia kell számot!");  
    } else if (!tartalmazBetut) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszavának tartalmaznia kell betűt!");  
    } else if (!tartalmazSpecialiskaraktert) {  
        throw new FelhasznaloException(hiba: "A felhasználó jelszavának tartalmaznia kell speciális karaktert!");  
    } else {  
        return true;  
    }  
}
```

Bármilyen hiba esetén egy kivételt dob, amit visszaküldünk a frontendnek, így tudatjuk a felhasználóval, hogy rossz adatot adott meg.

A regisztrációt követően a felhasználó a bejelentkezés után képes vásárolni játékot. A bejelentkezés során JSON Web Token-eket használunk. A felhasználó megadja a felhasználónevét és jelszavát, ha mindegyik helyes, akkor egy hosszú

karakterláncot küld vissza a "bejelentkezés" végpont. Ez a karakterlánc a felhasználó adatait tartalmazza egy objektumban, amit kódoltunk. Dekódolás után a frontend ezt a böngésző localStorage-ában tárolja. Ez a token 1 óráig érvényes, utána az oldal kilépteti a felhasználót.

A legelső oldal amit a felhasználó lát, az nem más, mint a **főoldal**. A főoldalon egy Carousel elemben 3 véletlenszerű játék jelenik meg, amiket a felhasználó a kosarába helyezhet. Ez a folyamat a következőféleképpen zajlik: A frontend küld egy HTTP kérést a JAX-RS web alkalmazás egyik végpontjához.

```
// GET kérés
fetch(
  "http://localhost:8080/CodeCraftersWebshop-1.0-SNAPSHOT/webresources/fooldal"
)
  .then((valasz) => valasz.json())
  .then((adat) => {
```

Az alkalmazás megfelelő végpontja feldolgozza a kérést és szükség esetén kommunikál az adatbázissal.

```
/**
 * REST Web Service
 *
 * @author tothm23
 */
@Path("/fooldal")
public class FooldalResource {

    public FooldalResource() {
    }

    @GET
    public Response _3veletlenjatek() {
        JSONArray eredmény = FooldalService._3veletlenjatek();
        return Response.status(status: Response.Status.OK).entity(entity: eredmény.toString())
            .type(type: MediaType.APPLICATION_JSON).build();
    }

}
```

Ebben a példában a `@Path("/fooldal")` annotáció megadja az elérési utat, amelyen keresztül a szolgáltatás elérhető lesz. Tehát a szolgáltatás ebben az esetben a `/fooldal` úton lesz elérhető a szerveren. A `@GET` annotáció azt jelzi, hogy a `_3veletlenjatek` metódus egy GET kérést fog kiszolgálni. Ez a metódus a `FooldalService` osztály `_3veletlenjatek()` metódusát hívja meg, aminek eredményét `JSONArray` formájában kapjuk meg. Egy olyan `Response` objektummal tér vissza,

ami beállítja a HTTP státuskódot 200-ra, beállítja a válasz testét (a *JSONArray* szöveges reprezentációját) és a válasz típusát (JSON).

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory(persistenceUnitName: "com_CodeCraftersWebshop_war_1.0-SNAPSHOTPU");
EntityManager em = emf.createEntityManager();

try {
    StoredProcedureQuery spq = em.createStoredProcedureQuery(procedureName: "3veletlenjatek");

    List<Object[]> eredmeny = spq.getResultList();

    if (!eredmeny.isEmpty()) {
        for (Object[] sor : eredmeny) {
            HashMap<String, Object> jatek = new HashMap<>();

            jatek.put(key: "id", (Integer) sor[0]);
            jatek.put(key: "nev", (String) sor[1]);
            jatek.put(key: "ar", (Integer) sor[2]);
            jatek.put(key: "leiras", (String) sor[3]);
            jatek.put(key: "kep", (String) sor[4]);
            jatek.put(key: "korhatar", (Integer) sor[5]);
            jatek.put(key: "akcio", (Integer) sor[6]);
            jatek.put(key: "mennyisegraktaron", (Integer) sor[7]);
            jatek.put(key: "eszkoz", (String) sor[8]);
            jatek.put(key: "platform", (String) sor[9]);

            jatekok.add(e: jatek);
        }
    }
} catch (Exception e) {
    System.err.println(x: e.getMessage());
} finally {
    em.clear();
    em.close();
    emf.close();
}
```

Ez a kódsor elvégzi a tényleges kommunikációt az adatbázissal. Jelen esetben a JPA (Java Persistence API) segítségével meghívja azt a tárolt eljárást, amely kiválassza a 3 játékot véletlenszerűen, a megfelelő táblából. Az adatbázis-kezelő keretrendszer az *EntityManager* és az *EntityManagerFactory* objektumokat használja az adatokhoz való hozzáférésre. A hibakezelés-ért a *try-catch* ágak felelősek, majd a hibától függetlenül a *finally* block-ban lezárjuk az összes objektumot a szerver erőforrásainak felszabadítása érdekében.

A választ visszaküldi a frontendnek, ahol azt a JavaScript tovább dolgozza és megjeleníti a felhasználói felületen. Ez a három technológia együttműködése segít a webalkalmazások teljeskörű működésében.

6. Forráskód

A projekt teljes forráskódja megtekinthető a Github felületén:

<https://github.com/tothm23/CodeCrafters>

7. Tesztelési dokumentáció

A Postman által generált API teljes dokumentációja:

<https://documenter.getpostman.com/view/29325685/2s9YsDmFaZ>

A JUnit tesztek és eredményeik megtekinthetők a Github felületén:

<https://github.com/tothm23/CodeCrafters>

8. Továbbfejlesztési lehetőségek

Webáruházunk fejlesztésre nyitottak vagyunk. Leginkább az elmaradt dolgokat szeretnénk megvalósítani a későbbiekben.

Ide sorolnám a bejelentkezést Google vagy Facebook fiókkal és a bankkártyás fizetést. A bankkártyás fizetéshez a Simplepay-t vagy a Bariont használnánk. Bankkártya információkat nem tárolnánk az adatbázisban, viszont sikeres fizetés esetén tájékoztatnánk a felhasználót egy e mailben.

A játékok mellett szívesen kínálnánk ajándékkártyákat, amiknek értékét levonnánk a játékból. Megvalósítanánk egy bestsellers opciót, ami a 3 legtöbbet eladott játék vagy ajándék kártyát jelenítené meg.

A regisztrációt úgy bővítenénk, hogy a felhasználó képes legyen profilkép feltöltésére, továbbá képes legyen törölni a profilját. Sikeres bejelentkezést követően a felhasználó láthatná, hogy meddig érvényes a tokene. A regisztrációhoz kapcsolnánk egy születési év mezőt, aminek előnye, hogy a felhasználó a születésnapján kedvezményben részesülne (egyedi CodeCrafters ajándék kártya). Adatbázisunk rögzíti a sikeres regisztráció idejét, így minden évben jutalmaznánk ezt. Az ünnepek idején csökkennének a termékek árai, amit az adatbázisban esemény indítással valósítanánk meg.

Idő hiányában elmaradt az adminisztrációs oldal, ami különböző statisztikákat jelenítené meg a vásárlásokkal kapcsolatban. Az adminisztrátor képes lenne inaktívvá tenni egy felhasználót, aki erről e mail formájában szerezne tudomást.

IV. Összegzés

Együttműködésünk eredményeként hoztuk létre a webáruházat, amely videójátékok értékesítésére szolgál. A projekt célja az volt, hogy kielégítse az online vásárlási igényeket, különösen a Covid-19 járvány idején. A fejlesztés során felmerült

nehézségek ellenére csapatunk sikeresen implementálta a webáruház funkcionalitását, beleértve az email küldést és a felhasználóbarát kezelőfelületet. A dokumentációnk részletesen ismerteti a projekt folyamatát, a felhasznált technológiákat és a webáruház használatának lépéseit. Folyamatosan fejlődünk, ahogy haladtunk a projekttel, úgy bővítettük ismereteinket. A munkából mindenki egységesen kivette a részét, aminek eredménye egy működő webáruház lett.