

# Web technológiák 2.

Féléves feladat

*Videójáték nyilvántartó rendszer*

Készítette: **Gerőcs Gergő**

Neptun kód: **FEU2E5**

# 1 Tartalomjegyzék

2	Felhasználói felület.....	3
2.1	Üdvözlés .....	3
2.2	Bejelentkezés .....	4
2.3	Adminisztrátori menü .....	5
2.3.1	Ügyintézők Listázása.....	6
2.3.2	Ügyintézők Létrehozása .....	6
2.3.3	Ügyintézők Módosítása .....	7
2.3.4	Adminisztrátorok Listázása .....	7
2.3.5	Adminisztrátorok Létrehozása.....	8
2.3.6	Adminisztrátorok Módosítása.....	8
2.3.7	Adminisztrátorok Törlése .....	9
2.4	Ügyintézői menü.....	9
2.4.1	Játékok Listázása .....	10
2.4.2	Játékok Keresése.....	10
2.4.3	Játékok Létrehozása.....	11
2.4.4	Játékok Módosítása.....	12
2.4.5	Játékok Törlése .....	12
3	Kód .....	13
3.1	Model.....	13
3.2	Entity.....	14
3.3	Controller .....	17
3.4	Component.....	20
3.5	Service .....	23

## 2 Felhasználói felület

### 2.1 Üdvözlés

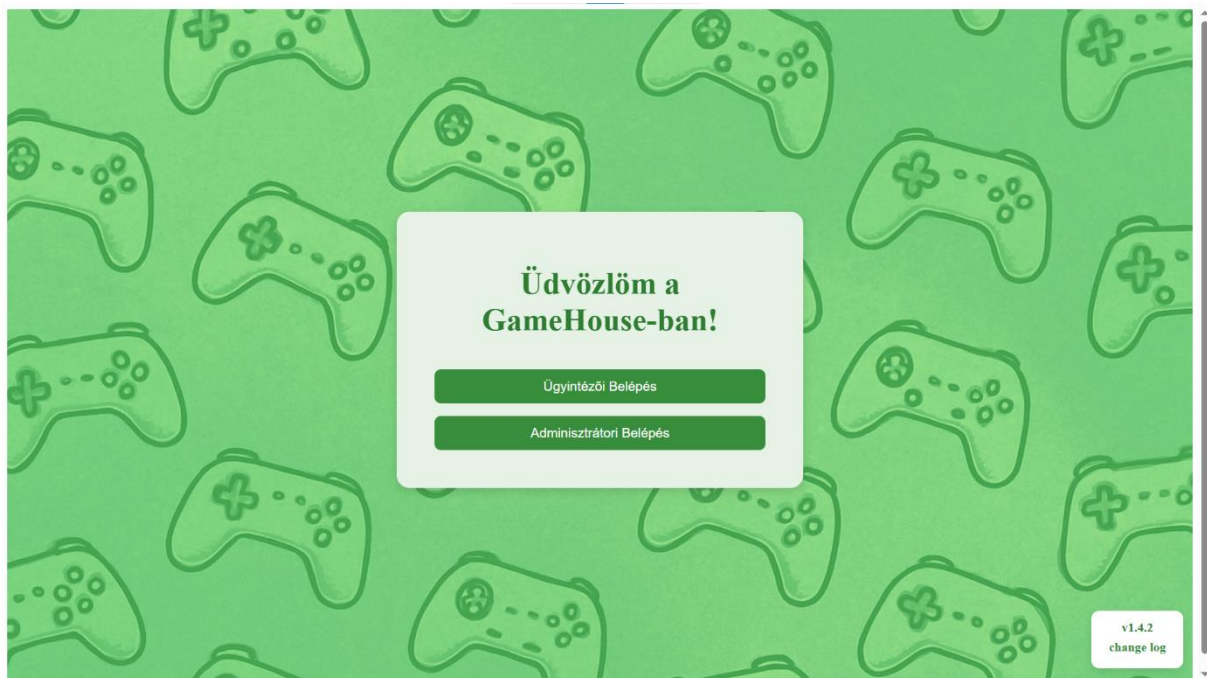
A GameHouse egy online videójáték nyilvántartó rendszer, melyet ügyintézők és adminisztrátorok használnak.

Az ügyintézők játékokat tudnak hozzáadni, módosítani, listázni és törölni, illetve különböző paraméterek alapján tudnak keresni.

Az adminisztrátorok az ügyintézők adatait tudják szerkeszteni, illetve új ügyintézőt hozzá adni, más adminisztrátorok adatait szerkeszteni, csak a superAdmin tudja.

Az alkalmazás elindításakor a felhasználó az üdvözlő menübe kerül, innen kiválaszthatja, hogy ügyintézői vagy adminisztrátori bejelentkezést.

Az adott gombra kattintva a felhasználó az adott bejelentkező felületre kerül.



## 2.2 Bejelentkezés

Az ügyintézőknek, valamint az adminisztrátoroknak saját bejelentkezési felülete van.

Az ügyintézőknek jelszóra és felhasználónévre van szükségük a belépéshez, ezeket nem saját maguk, hanem az adminisztrátorok adják meg.



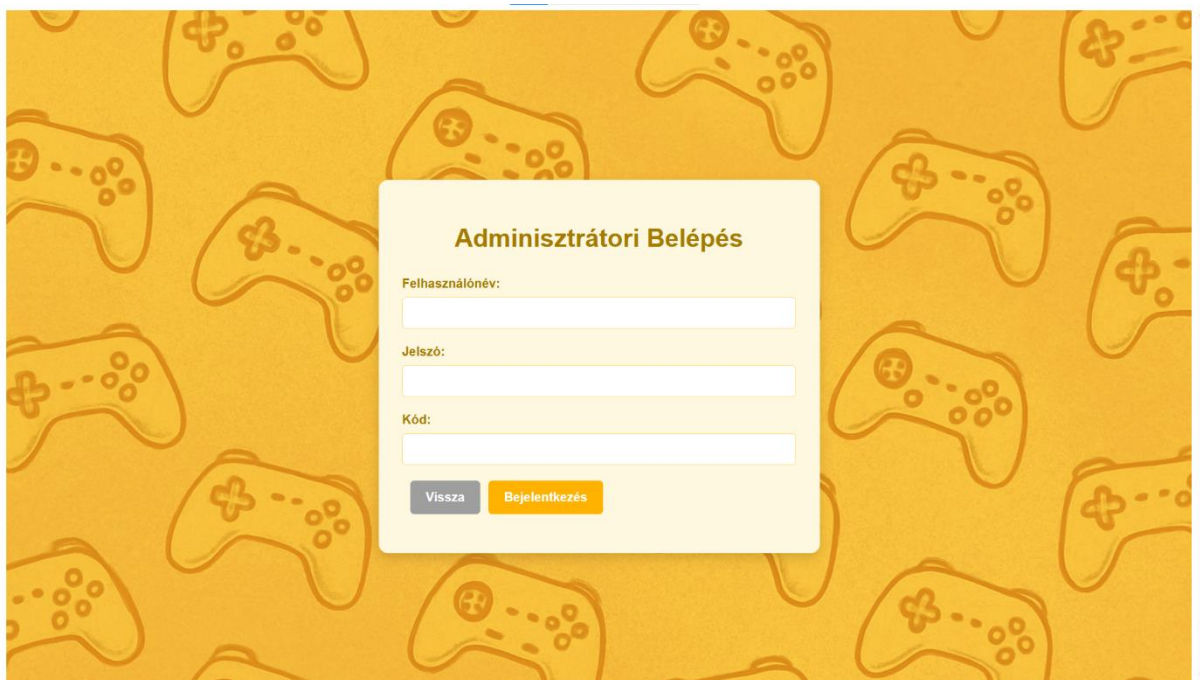
Ügyintézői Belépés

Felhasználónév:

Jelszó:

Vissza Bejelentkezés

Az adminisztrátoroknak jelszóra, felhasználónévre és egy kódra van szükségük.



Adminisztrátori Belépés

Felhasználónév:

Jelszó:

Kód:

Vissza Bejelentkezés

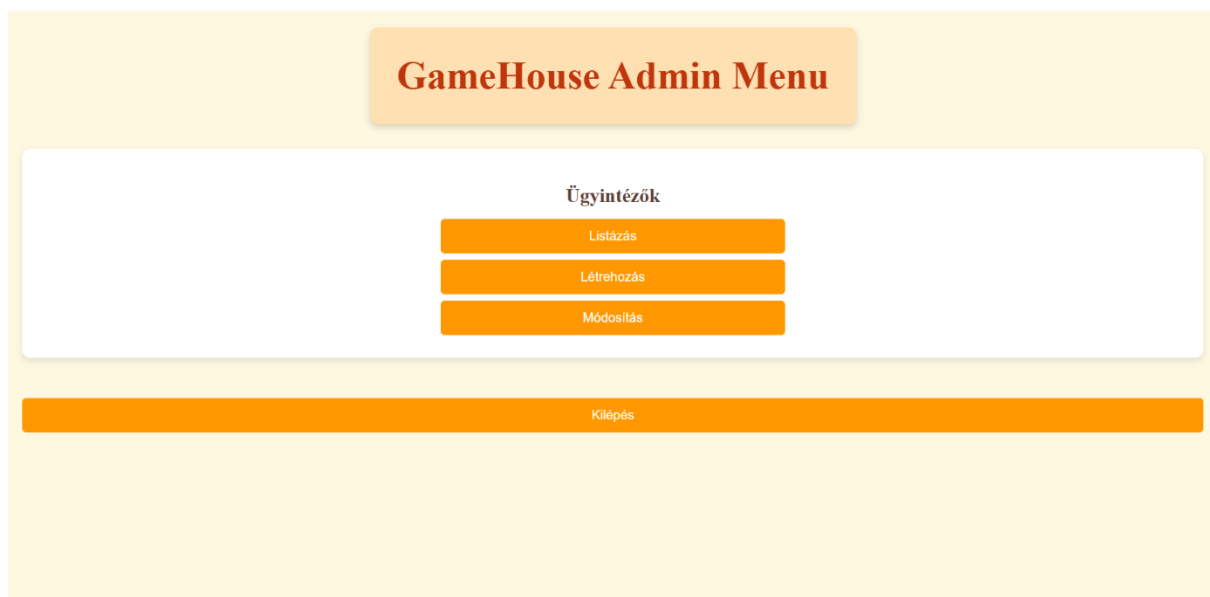
## 2.3 Adminisztrátori menü

Ha az adminisztrátor superAdmin-ként lép be, akkor elérhetővé válik az adminisztrátorok szerkesztése, létre lehet hozni új adminisztrátor fiókot, módosítani, listázni, illetve törölni, a már létezőt.

SuperAdmin fiókot nem lehet létrehozni, automatikusan jön létre indításkor, továbbá nem lehet módosítani és törölni sem.



Ha az adott illető, sima adminisztrátorként lép be, akkor számára csak az ügyintézőkkel kapcsolatos menük érhetőek el.



### 2.3.1 Ügyintézők Listázása

Ebben a menüben az adminisztrátorok, láthatják felsorolva az ügyintézőket, és azok adatait.

Dolgozók listája

ID	Felhasználónév	Aktiv
1	ugyintezo1	Igen
2	ugyintezo2	Igen
3	ugyintezo3	Igen

Vissza

### 2.3.2 Ügyintézők Létrehozása

Ezen menüpontban az adminisztrátoroknak lehetőségük van, új ügyintézői fiók létrehozására.

Új dolgozó létrehozása

Felhasználónév:

Jelszó:

Létrehozás

Vissza

### 2.3.3 Ügyintézők Módosítása

A módosítás felületen lehetséges az ügyintézők adatait módosítani, például kaphatnak új jelszót vagy felhasználónevet. A módosítás Id alapján történik.

### Dolgozó adatlap

Adattípus	Érték	Művelet
ID	1	
Felhasználónév	ugyintezo1	Módosítás
Jelszó	*****	Módosítás
Állapot	Aktív	

Inaktíválás Aktíválás

Vissza

### 2.3.4 Adminisztrátorok Listázása

Ezen felületen a jogosultsággal rendelkező felhasználók megtekinthetik az adminisztrátorok listáját, mely tartalmazza a felhasználónevet, Id-t, illetve a státuszt, ami lehet aktív vagy inaktív.

### Adminisztrátorok listája


ID	Felhasználónév	Státusz
2	admin1	Aktív
4	admin2	Aktív
5	admin3	Aktív
6	admin4	Aktív

Vissza a főmenübe

## 2.3.5 Adminisztrátorok Létrehozása

SuperAdminként létrehozható új adminisztrátori fiók, ehhez szükség van a következőkre:


- Felhasználónév
- Kód
- jelszó



A screenshot of a web form titled "Új Admin létrehozása" (New Admin Creation). The form is set against a light yellow background. It contains three input fields: "Felhasználónév:" (Username), "Jelszó:" (Password), and "Kód:" (Code). Below the fields are two buttons: an orange "Létrehozás" (Create) button and a grey "Mégse" (Cancel) button.

## 2.3.6 Adminisztrátorok Módosítása

Ezen felület a superAdmin képes az adminisztrátorok adatainak a módosítására, Id alapján.




A screenshot of a web form titled "Admin módosítása" (Admin Modification). The form is set against a light yellow background. It contains three input fields: "Admin ID:", "Új jelszó:" (New password), and "Új kód:" (New code). Below the fields are two buttons: an orange "Módosítás" (Modification) button and a grey "Vissza" (Back) button.



### 2.3.7 Adminisztrátorok Törlése

Az adminisztrátori fiókok törlésére, szintén a superAdmin képes, ez a törlés, csak státusz váltással jár.



The image shows a web form titled "Admin Törlés / Aktiválás". It features a text input field labeled "Admin ID:". Below the input field are three buttons: a red button labeled "Törlés", a green button labeled "Aktiválás", and a light gray button labeled "Vissza".

## 2.4 Ügyintézői menü

Az ügyintézői menüben lehetséges a játékokkal kapcsolatos műveletek végrehajtása. Ezen műveletek:

- Listázás
- Keresés
- Létrehozás
- Módosítás
- Törlés



The image shows a web interface titled "Ügyintézői Menü". It contains a section labeled "Játékok" with a list of buttons: "Listázás", "Keresés", "Létrehozás", "Módosítás", "Törlés", and "Kilépés".

### 2.4.1 Játékok Listázása

Ezen a felületen láthatóak a játékok kilistázva adataikkal együtt.

Ezek a következők:

- Sorszám
- Cím
- Hozzáadás dátuma
- Kategória
- Platform

A platform és kategória adatot az ügyintéző egy előre megadott listából választhatja ki.

Játékok listája				
Sorszám	Cím	Hozzáadás dátuma	Kategória	Platform
2	POSTAL	2025-05-30	Akcio	PC
3	South Park 1	2025-05-30	Kaland	PS4
4	South Park 2	2025-05-30	Kaland	PS5
5	Postal 4	2025-05-31	Akcio	XboxOne
Vissza				

### 2.4.2 Játékok Keresése

A játékok keresése többféleképpen is történhet, lehet Id, Cím, Kategória és Platform alapján keresni.

Fontos megjegyezni, hogy egymás felett helyezkednek el a keresési feltételek mezői, és ez a fentről lefelé haladó sorrend maga a keresési sorrend.

### Játék keresés

ID:

Cím:

Kategória:

Akcio

Platform:

-- Válassz --

Keresés

Vissza

ID	Cím	Beszerezés dátuma	Kategória	Platform
2	POSTAL	2025-05-30	Akcio	PC
5	Postal 4	2025-05-31	Akcio	XboxOne

### 2.4.3 Játékok Létrehozása

Az ügyintéző hozzáadhat új játékokat, ebben az esetben meg kell adni a játék címét, kategóriáját, illetve a platformot, amin megjelent.

A hozzáadás dátuma automatikusan kerül hozzáadásra, illetve a sorszámot az adatbázis generálja.

### Új játék felvitel

Cím:

Kategória:

Akcio

Platform:

PC

Mentés

Vissza

## 2.4.4 Játékok Módosítása

Ebben a menüben lehet a játékok adatait módosítani azonosító alapján. Ezen adatokat egyesével lehet változtatni.

Módosítható adatok:

- Cím
- Kategória (az előre meghatározott kategóriák közül lehet választani)
- Platform (az előre meghatározott platformok közül lehet választani)

### Videójáték adatlap

Adattípus	Érték	Művelet
Sorszám	3	
Cím	South Park 1	Módosítás
Kategória	Kaland	Módosítás
Platform	PS4	Módosítás

Viszsa

## 2.4.5 Játékok Törlése

A játékok törlését az ügyintéző végzi és sorszám alapján történik.

### Videójáték adatlap

Adattípus	Érték
Sorszám	2
Cím	POSTAL
Kategória	Akcio
Platform	PC

Törlés

Viszsa

## 3 Kód

A GameHouse alkalmazás frontendje Angular keretrendszerrel, TypeScript nyelven készült. A weboldalak egységes és reszponzív megjelenését saját CSS szabályok biztosítják.

A backend TypeScriptben, Node.js környezetben, Express keretrendszerrel, TypeORM leképezéssel és MySQL adatbázissal készült. Az alkalmazás támogatja a session alapú hitelesítést és bcrypt algoritmussal kezeli a jelszavakat.

### 3.1 Model

A projekt gyökerében létrehozott *models* jegyzék index.ts fájlja tartalmazza a kliens-szerveradatátvitelkor használt DTO-kat.

Kód:

```
export interface VersionDTO {
    version: string;
}

export interface GameDTO {
    sorszam: number;
    cim: string;
    beszerzes_datuma: Date;
    kategoria: string;
    platform: string;
}

export interface StaffDTO {
    id: number;
    username: string;
    password: string;
    isActive: boolean;
}

export interface StaffResponseDTO {
    id: number;
    username: string;
    isActive: boolean;
}

export interface AdminDTO {
    id: number;
```

```

    username: string;
    password: string;
    code: string;
    isActive: boolean;
}

export interface AdminResponseDTO {
    id: number;
    username: string;
    isActive: boolean;
}

```

## 3.2 Entity

A feladat elkészítése során 3 entitást használtam:

- Admin: adminisztrátor, tulajdonságai:
  - id, username, password, code, isActive
- Game: videójáték, tulajdonságai:
  - sorszam, cím, beszerzes\_datuma, categoria, platform
- Staff: ügyintéző, tulajdonságai:
  - id, username, password, isActive

Az isActive minden esetben egy igaz/hamis érték, amelyet törléshez használtam, mivel a törlés ebben az esetben, csak státusz váltással jár.

A játékokhoz tartozó categoria és platform mezők csak előre meghatározott értékeket vehetnek fel, mivel ezek enum típusokként kerültek definiálásra a backend oldalon. Így biztosítható, hogy csak a meghatározott értékkészletből lehessen választani, ezzel is növelve az adatintegritást és elkerülve a hibás adatbevitelt.

### Entity-k:

Admin.ts entity

```

@Entity("admin")
export class Admin implements AdminDTO{

    @PrimaryGeneratedColumn()
    id: number;

```

```

@Column()
username: string;

@Column()
password: string;

@Column()
code: string;

@Column({ default: true })
isActive: boolean;
}

```

## Game.ts entity

```

@Entity()
export class Game implements GameDTO{

    @PrimaryGeneratedColumn()
    sorszam: number;

    @Column()
    cim: string;

    @Column({ type: 'date', nullable: false })
    beszerzes_datuma: Date;

    @Column({
        type: 'enum',
        enum: JatekKategoria
    })
    kategoria: JatekKategoria;

    @Column({
        type: 'enum',
        enum: Platform
    })
    platform: Platform;
}

```

## Staff.ts entity

```
@Entity()
export class Staff implements StaffDTO{

    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    username: string;

    @Column()
    password: string;

    @Column()
    isActive: boolean;

}
```

## Enumok:

### platform.enum.ts

```
export enum Platform {
    PC = 'PC',
    PS5 = 'PS5',
    PS3 = 'PS3',
    PS4 = 'PS4',
    PS2 = 'PS2',
    XBOX360 = 'Xbox360',
    XBOXOne = 'XboxOne',
    SWITCH = 'Switch'
}
```

### game-category.enum.ts

```
export enum JatekKategoria {
    AKCIO = 'Akció',
    KALAND = 'Kaland',
    RPG = 'RPG',
    STRATEGIA = 'Stratégia',
    SPORT = 'Sport',
    VERSENY = 'Verseny',
    HORROR = 'Horror',
    PUZZLE = 'Puzzle',
    SZIMULATOR = 'Szimulátor',
}
```



```
MMO = 'MMO',  
PARTY = 'Party',  
ZENEI = 'Zenei',  
INDIE = 'Indie',  
SCIFI = 'Sci-fi',  
FANTASY = 'Fantasy'  
}
```

### 3.3 Controller

A controller osztályok kezelik a HTTP kérések logikáját és kommunikálnak az adatbázissal a repository-kon keresztül. A fő feladataik:

- Kérések (GET, POST, PUT, DELETE) fogadása
- Validáció (pl. kötelező mezők ellenőrzése)
- Jogosultságok kezelése (pl. admin aktív-e)
- Válaszok küldése a kliens felé JSON formátumban
- Hiba esetén megfelelő hibakezelés

Mindegyik controller saját repository-val dolgozik és aszinkron függvényeket használ. A jelszavak és kódok titkosításához a bcrypt csomagot alkalmazzák.

#### AdminController

Feladata: adminisztrátorok kezelése (bejelentkezés, létrehozás, módosítás, törlés, aktiválás).

#### Főbb metódusok:

- loginAdmin: ellenőrzi a bejelentkezési adatokat (username, password, code), bcrypt segítségével ellenőrzi a jelszót és a kódot, session-be rakja a bejelentkezett admin adatait.
- getCurrentAdmin: visszaadja a bejelentkezett admin adatait.
- listAllAdmin: összes aktív admin listázása, superadmin kivételével.
- createAdmin: új admin létrehozása, jelszó és kód bcrypt hash-eléssel.
- modifyAdmin: admin jelszó és/vagy kód módosítása.
- deleteAdmin: admin deaktiválása (nem törlés adatbázisból).
- activateAdmin: admin újraaktiválása.
- handleError: egységes hibakezelő metódus.

## GameController

Feladata: játékok kezelése (létrehozás, lekérés, módosítás, törlés, listázás).

Főbb metódusok:

- createGame: játék létrehozása (cím, kategória, platform), ellenőrzi a kategóriát és platformot.
- getGame: játék keresése ID, cím, kategória vagy platform alapján.
- getGameById: játék lekérése ID alapján.
- modifyGame: játék adatok módosítása (cím, kategória, platform).
- deleteGame: játék törlése adatbázisból.
- getAllGame: összes játék listázása.
- handleError: egységes hibakezelés.

## StaffController

Feladata: dolgozók kezelése (bejelentkezés, létrehozás, lekérés, módosítás, törlés, aktiválás).

Főbb metódusok:

- loginStaff: bejelentkezés felhasználónév és jelszó alapján, bcrypt hash ellenőrzés.
- createStaff: új dolgozó létrehozása, jelszó hash-eléssel.
- getStaffById: dolgozó lekérése ID alapján.
- modifyStaff: dolgozó adatainak módosítása (felhasználónév ellenőrzés, jelszó hash).
- deleteStaff: dolgozó inaktiválása.
- activateStaff: dolgozó újraaktiválása.
- handleError: egységes hibakezelés.

## Router

A kontrollerek útvonalhoz rendelése a routes.ts fájlban felépített express Router alapján az index.ts fájlban történik.

```
import express from 'express';
import {VersionController} from '../controllers/version.controller'
import { Router } from 'express';
import { AppDataSource } from '../data-source';
import { StaffController } from '../controllers/staff.controller';
```

```
import { AdminController } from './controllers/admin.controller';
import { GameController } from './controllers/game.controller';

const router = Router();

//Version
const versionController = new VersionController();
router.get('/version', versionController.getVersion);

//game
const gameController = new GameController();
router.get('/getAllGames', gameController.getAllGame);
router.post('/createGame', gameController.createGame);
router.get('/getGame', gameController.getGame);
router.get('/getGameById/:id', gameController.getGameById);
router.put('/modifyGame/:id', gameController.modifyGame);
router.delete('/deleteGame/:id', gameController.deleteGame);

//admin-staff
const staffController = new StaffController();
router.post('/loginStaff', staffController.loginStaff);
router.get('/listAllStaff', staffController.listAllStaff);
router.post('/createStaff', staffController.createStaff);
router.get('/getStaff/:id', staffController.getStaffById);
router.put('/modifyStaff/:id', staffController.modifyStaff);
router.put('/deleteStaff/:id', staffController.deleteStaff);
router.put('/activateStaff/:id', staffController.activateStaff);

//admin
const adminController = new AdminController();
router.post('/loginAdmin', adminController.loginAdmin);
router.get('/getCurrentAdmin', adminController.getCurrentAdmin);
router.get('/listAllAdmin', adminController.listAllAdmin);
router.post('/createAdmin', adminController.createAdmin);
router.put('/modifyAdmin/:id', adminController.modifyAdmin);
router.put('/deleteAdmin/:id', adminController.deleteAdmin);
router.put('/activateAdmin/:id', adminController.activateAdmin);

export { router };
```

### 3.4 Component

A frontend legfontosabb elemeit a következő Angular komponensek alkotják:

#### 1. WelcomeComponent

Kezdőképernyő, ahol a felhasználó kiválaszthatja, hogy ügyintézőként vagy adminisztrátorként szeretne bejelentkezni.

#### 2. StaffLoginComponent

Ügyintézők számára kialakított bejelentkezési felület felhasználónévvel és jelszóval.

#### 3. StaffMainMenuComponent

Ügyintézői vezérlőpult, ahol a játékkezelési funkciók (listázás, keresés, szerkesztés) érhetők el.

#### 4. GameListComponent

Az összes hozzáadott videójáték listáját jeleníti meg táblázatos formában.

#### 5. GameSearchComponent

Keresési felület, ahol szűrhetők a játékok különböző paraméterek alapján.

#### 6. GameCreateComponent

Menüpont, új játékok rendszerbe való felvételéhez.

#### 7. GameModifyComponent

Meglévő játékok adatainak módosítására szolgáló szerkesztő felület.

#### 8. GameDeleteComponent

Játékok törlését lehetővé tevő felület

#### 9. AdminLoginComponent

Adminisztrátorok számára kialakított bejelentkezés felhasználónév, jelszó és biztonsági kód megadásával.

#### 10.AdminCreateComponent

Új adminisztrátori fiók létrehozására szolgáló felület (kizárólag superAdmin számára).

#### 11.AdminModifyComponent

Adminisztrátorok profiljainak (jelszó, e-mail) módosítását teszi lehetővé (kizárólag superAdmin számára).

#### 12.AdminDeleteComponent

Adminisztrátori fiókok törlésére szolgáló felület (superAdmin jogosultság szükséges).

#### 13.AdminListComponent

Az összes regisztrált adminisztrátor listája (a superAdmin kivételével).

#### 14.AdminStaffListComponent

Ügyintézők nyilvántartása alapvető adatokkal.

#### 15.AdminStaffCreateComponent

Menü, új ügyintézők hozzáadásához a rendszerhez.

#### 16.AdminStaffModifyComponent

Ügyintézők adatainak (pl. jelszó) frissítésére szolgáló felület.

#### 17.AdminMainMenuComponent

Adminisztrátori irányítópult, ahol elérhetők a felhasználókezelési és rendszerbeállítási lehetőségek.

### Néhány példa:

game-list.component.ts

```
@Component({
  selector: 'app-game-list',
  imports: [NgFor, CommonModule],
  templateUrl: './game-list.component.html',
  styleUrls: ['./game-list.component.css']
})
export class GameListComponent implements OnInit {
  router = inject(Router);
  staffService = inject(StaffService);

  games: GameDTO[] = [];

  ngOnInit() {
    this.loadGames();
  }
}
```

```

loadGames() {
  this.staffService.getAllGames().subscribe({
    next: (res) => {
      this.games = res;
    },
    error: (err) => {
      console.error('Hiba a játékok betöltésekor:', err);
    }
  });
}

goBack() {
  this.router.navigate(['/staff-main-menu']);
}
}

```

## game-list.component.html

```

<div class="container">
  <h2>Játékok listája</h2>

  <table>
    <thead>
      <tr>
        <th>Sorszám</th>
        <th>Cím</th>
        <th>Hozzáadás dátuma</th>
        <th>Kategória</th>
        <th>Platform</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let game of games">
        <td>{{ game.sorszam }}</td>
        <td>{{ game.cim }}</td>
        <td>{{ game.beszerzes_datuma | date:'yyyy-MM-dd' }}</td>
        <td>{{ game.kategoria }}</td>
        <td>{{ game.platform }}</td>
      </tr>
    </tbody>
  </table>

  <button (click)="goBack()">Vissza</button>
</div>

```

## 3.5 Service

A backend API metódusainak meghívásáért a frontend oldalon a `src/app/services` mappában található `Service` osztályok felelnek.

Ezek a szolgáltatások egységes szerkezetet követnek, és HTTP kéréseket (GET, PUT, DELETE, POST) küldenek az `/api` útvonalon.

A proxy konfiguráció révén ezek a kérések automatikusan átirányulnak a `localhost:4200`-ról (ahol a frontend fut) a `localhost:3000`-es portra (ahol a szerver működik).

### Példa:

`admin.service.ts`

```
@Injectable({
  providedIn: 'root'
})

export class AdminService {
  http = inject(HttpClient);

  //Staff methods for admin menu
  createStaff(username: string, password: string) {
    return this.http.post<StaffResponseDTO>('/api/createStaff', { username, password});
  }

  getStaffById(id: number) {
    return this.http.get<StaffResponseDTO>(`/api/getStaff/` + id);
  }

  modifyStaff(id: number, username?: string, password?: string) {
    return this.http.put<StaffResponseDTO>(`/api/modifyStaff/` + id, { username, password});
  }

  deleteStaff(id: number) {
    return this.http.put<{ message: string, staff: StaffResponseDTO }>(`/api/deleteStaff/` + id, {});
  }

  activateStaff(id: number) {
    return this.http.put<StaffResponseDTO>(`/api/activateStaff/` + id, {});
  }
}
```

```

}

listAllStaff() {
    return this.http.get<StaffResponseDTO[]>('/api/listAllStaff');
}

//ADMIN

listAllAdmin() {
    return this.http.get<AdminResponseDTO[]>('/api/listAllAdmin');
}

createAdmin(username: string, password: string, code: string) {
    return this.http.post<AdminResponseDTO>('/api/createAdmin', { username,
password, code });
}

modifyAdmin(id: number, newPassword?: string, newCode?: string) {
    return this.http.put(`/api/modifyAdmin/${ id}, { newPassword, newCode });
}

deleteAdmin(id: number) {
    return this.http.put(`/api/deleteAdmin/${ id}, {});
}

activateAdmin(id: number) {
    return this.http.put(`/api/activateAdmin/${ id}, {});
}

getCurrentAdmin() {
    return this.http.get<{ isSuperAdmin: boolean }>('/api/getCurrentAdmin', {
withCredentials: true })
}
}

```