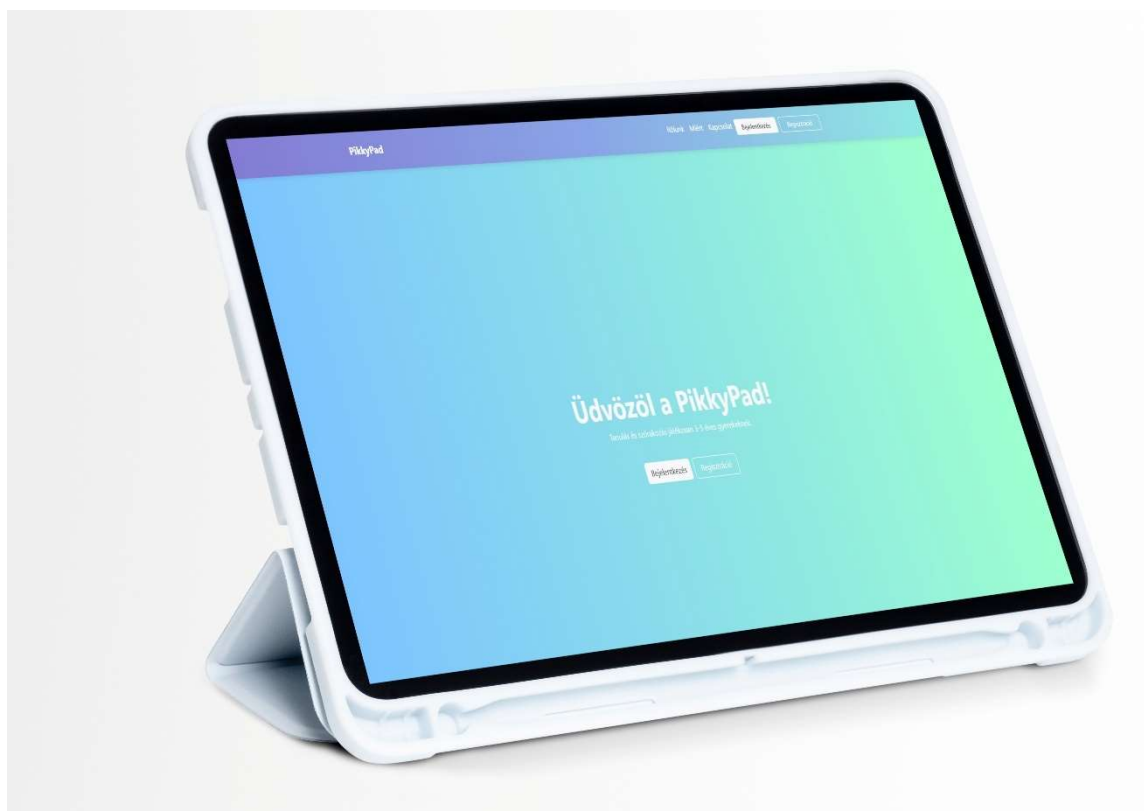


PikkyPad

Készítette:

Heszler István

Bogdán Csaba



Tartalomjegyzék

1. Frontend komponensek

- ChildDashboardComponent
- ChildCardComponent
- AuthModalComponent
- LoginComponent
- RegisterComponent
- ChildGamesComponent
- ChildPlayComponent
- HomePrivateComponent
- LandingComponent
- AdminChildrenComponent
- AdminGameComponent
- AdminLayoutComponent
- AdminUserComponent
- ColoringGameComponent
- PuzzleGameComponent
- MemoryGameComponent

2. Backend

- API végpontok
- Adatbázis struktúra
- RegisterRequest – validáció
- LoginRequest – validáció
- AuthController
- BannerController
- ResponseController
- ChildController
- GameController
- StatisticsController
- AdminController
- SuperAdminController
- MailController

3. Tesztelés

- Frontend tesztelés
- Backend tesztelés(Insomnia)

1.ChildDashboardComponent

A ChildDashboardComponent az egyik legfontosabb komponens, amely felelős a gyermekek kezeléséért és a játékkal kapcsolatos interakciókért.

Főbb feladatok:

- Gyermekek megjelenítése: A gyermekek adatainak listázása, valamint új gyermek hozzáadása.
- Játék hozzárendelése és eltávolítása: A szülő képes hozzárendelni vagy eltávolítani játékokat a gyermekekhez.
- Űrlapok kezelése: A gyermekek hozzáadásához és szerkesztéséhez űrlapokat biztosít.
- Hibakezelés és értesítések: A Toastr segítségével az alkalmazás értesíti a felhasználót az elvégzett műveletekről.

Kód elemzés:

- Űrlapkezelés: A gyermekek hozzáadásához és szerkesztéséhez a FormBuilder segítségével dinamikusan létrehozott formák használata biztosítja a validációkat.
- Játék hozzárendelése: A játékok hozzárendeléséhez a gyermekekhez egy egyszerű választás szükséges, ahol a felhasználó eldöntheti, hogy melyik játékot szeretné hozzárendelni egy adott gyermekhez.
- API hívások: A gyermekek és játékok adatainak kezelése szolgáltatásokon keresztül történik (ChildService és GameService).

Fontos funkciók:

- Gyermekek hozzáadása és szerkesztése: Az űrlap segítségével új gyermekeket lehet hozzáadni, illetve a meglévőket szerkeszteni.
- Játékok kezelése: A játékok a gyermekekhez rendelhetők, és eltávolíthatók, ezzel segítve a szülőket számára, hogy kezeljék, melyik gyermekhez melyik játék van hozzárendelve.

HTML:

- Bootstrap alapú navigációs sáv
- Jobb oldalon három navigációs gomb
 - o Logout metódus hívódik meg a kijelentkezéskor.

Gyermek Hozzáadása

- Reaktív űrlap a gyermekek nevének és korának bevitelére.
 - o Név kötelező mező
 - o Kor kötelező mező
- A gomb felirata dinamikusan változik: Hozzáadás vagy Módosítás.

Játékok

- Kártyás megjelenítés a játékok számára
- Minden játék alatt legördülő lista a gyermekek kiválasztásához.

2.ChildCardComponent

A ChildCardComponent komponens az egyes gyermekek megjelenítésére szolgál az alkalmazásban. Mivel a komponens csak a vizuális megjelenítéssel foglalkozik, az interakciók a szülő komponens (pl. ChildDashboardComponent) felé irányulnak.

A komponens funkciója:

- Gyermekek adatainak megjelenítése: A gyermekek egy-egy kártyán jelennek meg, amely az alapvető információkat tartalmazza, mint például név, életkor, és a hozzárendelt játékok.
- Szerkesztés és törlés lehetősége: A kártyán található gombok segítségével a felhasználó szerkesztheti vagy törölheti a gyermeket, vagy akár eltávolíthatja a játékot a gyermektől.
- Játék eltávolítása: A komponens lehetővé teszi a játékok eltávolítását a gyermekekről, ha szükséges.

Komponens elemzés

Inputok és Outputok:

- @Input() child: any;; A gyermek adatait a szülő komponens küldi át a ChildCardComponent számára. Ezt a gyermek információt a kártya megjelenítésére használjuk.

- `@Output() edit = new EventEmitter<any>();`: Amikor a felhasználó szerkeszteni szeretné a gyermeket, ezt az eseményt aktiválja. Az esemény az adatokat visszaküldi a szülő komponensnek a szerkesztéshez.
- `@Output() delete = new EventEmitter<number>();`: A gyermek törléséhez az id-t átadja a szülő komponensnek, amely felelős a gyermek törléséért az adatbázisban.
- `@Output() removeGame = new EventEmitter<{ childId: number, gameId: number }>();`: Ha a felhasználó eltávolít egy játékot a gyermektől, ezt az eseményt küldi, amely tartalmazza a gyermek és a játék azonosítóját.

Template:

A template-ban biztosítani kell az interakciókat, például a szerkesztéshez és törléshez szükséges gombokat, valamint az egyéb információk megjelenítését.

HTML

- Statikus avatar profilkép jelenik meg kör alakba.
- Gyermek neve és életkora jelenik meg a kártya közepén
- Szerkesztés: kattintáskor az aktuális child objektumot jeleníti meg a gyermekfelvételi mező helyén.
- Törlés: a gyermek id-jét továbbítja törlésre
- Játék indítása: routerLink navigáció child-play oldalra.

3.AuthModalComponent

Az AuthModalComponent a bejelentkezési és regisztrációs modális ablakot kezeli. A komponens felelős a felhasználók hitelesítéséért, és az autentikációval kapcsolatos logikát kezeli, beleértve a sikeres bejelentkezés utáni navigációt. A loginForm űrlap segítségével a felhasználó be tud jelentkezni, és ha a bejelentkezés sikeres, akkor az alkalmazás a megfelelő felhasználói jogosultságok alapján navigál.

A komponens funkciója:

- Bejelentkezési űrlap: A felhasználó beírhatja a felhasználónevet és a jelszót a bejelentkezéshez.
- Navigáció a sikeres bejelentkezés után: A rendszer a felhasználó szerepe alapján navigál a megfelelő oldalra.
- Hibakezelés: A bejelentkezés vagy egyéb hiba esetén a rendszer értesíti a felhasználót.
- Regisztrációs mód váltás: Lehetőséget ad a regisztrációra a bejelentkezési mód mellett.

Komponens elemzése:

Propertik:

- loginForm: FormGroup;: Ez az űrlap csoport a bejelentkezéshez szükséges mezőkkel (felhasználónév és jelszó).
- activeTab: 'login' | 'register' = 'login';: Meghatározza, hogy a modális ablak bejelentkezési vagy regisztrációs nézetben van.

Függőségek:

- AuthService: Az autentikációs szolgáltatás, amely végrehajtja a bejelentkezést, a regisztrációt és más kapcsolódó funkciókat.
- Router: Az alkalmazás navigálásáért felelős szolgáltatás.
- ToastrService: A felhasználói értesítések megjelenítéséhez használt könyvtár.
- RegisterComponent: A regisztrációs formot tartalmazó komponens, amelyet a felhasználó a regisztrációhoz használhat.

Módszerek:

- onLogin():
 - o A bejelentkezési adatokat (felhasználónév és jelszó) elküldi az AuthService szolgáltatásnak.
 - o Sikeres bejelentkezés után elmenti a token-t a localStorage-ba, majd lekéri az aktuális felhasználói adatokat.
 - o A felhasználó szerepe alapján navigál a megfelelő oldalra: admin felhasználókat az admin oldalra, a többieket a privát kezdőoldalra.
 - o Ha hiba történik, akkor egy értesítést küld a felhasználónak.

HTML:

- Célja: felhasználók bejelentkezésének vagy regisztrációjának kezelése.
 - o Bejelentkezéskor: `aktiveTab===login`
 - o Regisztrációkor: `aktiveTab ===register`
- Bejelentkezési űrlap 2 mezővel rendelkezik
 - o email
 - o password
 - o A beküldésekor az `onLogin()` metódus hívódik.
- Regisztráció külön komponensben van kiszervezve.

4.LoginComponent

A LoginComponent a felhasználó bejelentkezési folyamatát kezeli az Angular alkalmazásban. A bejelentkezéshez az email és jelszó megadását várja, és az autentikációs szolgáltatás (AuthService) segítségével ellenőrzi a bejelentkezési adatokat. Ha sikeres a bejelentkezés, a felhasználót a jogosultságának megfelelő oldalra irányítja, például az admin felületre, ha adminisztrátori szerepe van, vagy a privát kezdőlapra.

Funkciók:

Form űrlap:

- email: Kötelező mező, amely az email cím validálását végzi.
- password: Kötelező mező, amely a jelszót várja.

Metódusok:

- `onSubmit()`:
 - o Ellenőrzi, hogy az űrlap érvényes-e.
 - o Ha érvényes, a bejelentkezési adatokat elküldi az AuthService-nek.
 - o A bejelentkezés után lekéri a felhasználó adatait az AuthService-től, hogy meghatározza a felhasználó szerepét.
 - o Ha a felhasználó admin vagy superadmin, az alkalmazás az admin oldalt nyitja meg.
 - o Ha más szerepe van, a privát kezdőlapra irányítja.

- o Ha bármilyen hiba történik, a rendszer értesíti a felhasználót Toastr értesítésekkel.

Függőségek:

- AuthService: Az autentikációs logikát kezeli, például a bejelentkezést és a felhasználói adatok lekérését.
- Router: A navigációért felelős szolgáltatás, amely a felhasználót a megfelelő oldalra irányítja.
- ToastrService: Felhasználói értesítések megjelenítésére szolgáló könyvtár, amely segít az információk megjelenítésében.

HTML:

- Az onSubmit() metódus hívódik meg a beküldéskor.
- A form változó egy FormGroup, amely 2 mezőt tartalmaz: email és password.
- Email mező:
 - o Kötelező mező, email típusú.
 - o Placeholder példacímme.
- Jelszó mező:
 - o Szintén kötelező mező.
 - o Típus: password.
 - o Kiegészítő szöveg segít a helyes jelszó megadásában.
- Beküldés gomb
 - o Aktiválódik, ha az űrlap valid.

5.RegisterComponent

A RegisterComponent a felhasználók regisztrációját kezeli az Angular alkalmazásban. A komponens tartalmazza a szükséges mezőket, mint a felhasználói név, email, jelszó és a jelszó megerősítése. A regisztrációs űrlap helyes kitöltése után a AuthService szolgáltatás segítségével történik a regisztrációs kérés.

Funkciók:

Form űrlap:

- name: Kötelező mező a felhasználói névhez.
- email: Kötelező mező, amely az email cím validálását végzi.
- password: Kötelező mező, amely a jelszót várja.
- confirm_password: Kötelező mező, amely a jelszó megerősítésére szolgál.

Metódusok:

- onSubmit():
 - o Ellenőrzi, hogy az űrlap érvényes-e.
 - o Ha érvényes, a regisztrációs adatokat elküldi az AuthService-nek.
 - o A regisztráció után értesítést küld a felhasználónak a sikeres regisztrációról (Toastr).
 - o Ha hiba történik, a hibákat a válasz alapján jeleníti meg, és a felhasználónak visszajelzést ad a hiba okáról.
 - o A regisztráció után az alkalmazás visszairányítja a felhasználót a kezdőlapra.

Függőségek:

- AuthService: A regisztrációt és az autentikációval kapcsolatos logikát kezeli.
- ToastrService: Felhasználói értesítések megjelenítésére szolgáló könyvtár.
- Router: A navigációt kezeli, hogy a felhasználót a megfelelő oldalra irányítsa.

HTML:

- Angular Reactive Forms alapú validált regisztrációs űrlap.
- A form nevű FormGroup mezői: name, email, password, confrim_password.
- Beküldésekor az onSubmit() metódus fut le.
- 4 mező van amit ki kell tölteni
 - Név
 - Email
 - Jelszó
 - Ismétlő jelszó
- Beküldés gomb: akkor aktiválódik ha az űrlap valid.

6.ChildGame

A ChildGamesComponent felelős azért, hogy megjelenítse egy adott gyermekhez rendelt játékokat és lehetővé tegye a játékok indítását. Az alábbiakban részletesen bemutatom a komponens működését és a kulcsfontosságú funkcióit.

Funkciók:

Alapvető felépítés:

- childId: Ez a változó tárolja az aktuálisan megjelenített gyermek ID-ját, amely az URL-ből (route paraméter) kerül kinyerésre.
- games: Ez a tömb tárolja azokat a játékokat, amelyeket az adott gyermekhez rendeltek.

Metódusok:

- ngOnInit():
 - A komponens betöltődésekor fut le, és kinyeri a gyermek ID-ját az URL paramétereiből.
 - Ezt követően meghívja a loadGamesForChild() metódust, hogy betöltse a gyermekhez rendelt játékokat.
- loadGamesForChild():
 - A GameService segítségével lekéri az adott gyermekhez rendelt játékokat a backendből.

- o Ha a kérés sikeres, az adatokat elmenti a games tömbbe, és egy sikeres értesítést jelenít meg a felhasználónak.
- o Ha a kérés hibás, hibaüzenetet jelenít meg a felhasználónak.
- startGame():
 - o Ez a metódus akkor hívódik meg, amikor a felhasználó elindít egy játékot.
 - o A kiválasztott játék ID-ját és a gyermek ID-ját átadja a router query paramétereiben, és navigál a /child-play oldalra, ahol a játék elindul.

HTML:

- A games tömb alapján *ngFor ciklussal jönnek létre a kártyák.
- Minden kártya tartalmaz:
 - o Játék címet
 - o leírást
 - o egy Indítás gombot amely startGame(game.id) metódust hívja meg.

7.ChildplayComponents

A ChildPlayComponent egy komplex komponens, amely kezeli a játékok lejátszását egy gyermek számára. Az alábbiakban részletesen bemutatom a komponens működését és annak kulcsfontosságú funkcióit.

Funkciók:

Alapvető felépítés:

- games: Tárolja a gyermekhez rendelt játékokat.
- currentGameIndex: Az aktuálisan lejátszott játék indexét tartalmazza.
- timer: A játékhoz beállított időzítő másodpercben (1 óra, vagyis 3600 másodperc).
- intervallId: Az időzítő kezelő ID-ja, hogy leállíthassuk, amikor a játék véget ér.

Metódusok:

- `ngOnInit():`
 - o Amikor a komponens betöltődik, lekéri a gyermek ID-ját a URL query paramétereiből (`childId`).
 - o Ha a gyermek ID nem található, a felhasználó automatikusan visszairányításra kerül a `/home-private` oldalra.
 - o A `gameService.getGamesForChild()` segítségével lekéri az adott gyermekhez rendelt játékokat.
 - o Ha nincs játék, akkor egy információs értesítést kap a felhasználó. Ha vannak játékok, elindul a játékidőzítő.
- `currentGame():`
 - o Az aktuális játékot adja vissza az index alapján (`currentGameIndex`).
- `isMemoryGame(), isColoringGame(), isPuzzleGame():`
 - o Ezek a metódusok ellenőrzik, hogy az aktuális játék típusa megfelel-e a memóriajáték, színezőjáték vagy puzzle játéknak.
- `nextGame()` és `previousGame():`
 - o Ezek a metódusok segítenek navigálni a játékok között (a játékok listáján belül előre vagy hátra).
- `startTimer():`
 - o Elindítja a visszaszámláló időzítőt, ami minden másodpercben csökkenti a timer változó értékét. Ha az idő lejár, automatikusan leállítja a játékot, és a felhasználó visszakerül a `home-private` oldalra.
- `formatTime():`
 - o Az időt formázza percek és másodpercek formájában, például `1:30` a `90` másodperces időszakra.
- `goBack():`
 - o A felhasználó visszaléphet a kezdőoldalra, és a timer is leállítódik.

HTML

- Ez a nézet biztosítja a gyermek számára a kiválasztott játék futtatását, időkövetéssel.
- A fejléc tartalmazza a játék címét, az aktuális időt ami a játékból hátra maradt.
- A kiválasztott játékok közötti lépkedést `previousGame()/nextGame` metódussal valósult meg.

8.Home-privateComponents

A `HomePrivateComponent` a szülők számára biztosítja a hozzáférést a gyermekek kezeléséhez és a beállítások módosításához. Az alábbiakban részletesen ismertetem a komponens működését és annak kulcsfontosságú funkcióit.

Funkciók és működés:

Alapvető felépítés:

- `parentName`: Tárolja a szülő nevét, amit a gyermekek adatai közül olvasunk ki.
- `children`: Tárolja a szülőhöz rendelt gyermekek listáját, amelyet a `ChildService` segítségével töltünk le.
- `childService`: A gyermekek adatainak lekérdezésére szolgáló szolgáltatás.
- `toastr`: Az értesítések kezelésére szolgáló könyvtár, amely értesítéseket jelenít meg a felhasználónak.

Metódusok:

- `ngOnInit()`:
 - o A komponens betöltődésekor meghívja a `fetchChildren()` metódust, hogy lekérje a szülőhöz tartozó gyermekeket.
- `fetchChildren()`:
 - o A `childService.getChildren()` metódus segítségével lekéri a gyermekek adatait.
 - o A válaszban található első gyermek adatából kinyeri a szülő nevét, és ezt elmenti a `parentName` változóba.

- o Ha a gyermekek listája üres, akkor alapértelmezettként a "Szülő" szöveget állítja be.
- `logout()`:
 - o A `logout()` metódus a felhasználó kijelentkezését kezeli.
 - o Az aktuális token-t eltávolítja a helyi tárolóból, és a felhasználót visszairányítja a kezdőoldalra.
 - o Ha a kijelentkezés sikeres, akkor sikeres üzenetet jelenít meg. Ha hiba lép fel, akkor hibát jelez.
- `goToChildDashboard()`:
 - A felhasználót átirányítja a gyermekek kezelésére szolgáló oldalra (`child-dashboard`).

HTML

- Navigációs sáv tartalmazza a három fő navigációt:
 - o Főoldal – visszairányít a kezdőlapra (`/home-private`)
 - o Gyermekek – a gyermekek kezelésére szolgáló oldalra navigál (`/child-dashboard`).
 - o Kijelentkezés – a `logout()` metódus hívásával kijelentkezik a felhasználótól.
- Tartalom boks: rövid bemutató az oldalról felhasználóknak.

9.landingComponents

A `LandingComponent` a bejelentkezési és regisztrációs felületek kezelésére szolgál. A komponens célja, hogy lehetőséget adjon a felhasználóknak a bejelentkezésre és a regisztrációra, az ezekhez szükséges modálok kezelése révén. Az alábbiakban részletesen ismertetem a komponens működését.

Funkciók és működés:

Alapvető felépítés:

- `showLogin`: Logikai változó, amely meghatározza, hogy a bejelentkezési modál legyen látható vagy sem.
- `showRegister`: Logikai változó, amely meghatározza, hogy a regisztrációs modál legyen látható vagy sem.
- `openLogin()`: Ez a metódus a bejelentkezési modál megjelenítésére szolgál. Ha meghívják, a bejelentkezési modál láthatóvá válik, és a regisztrációs modál eltűnik.

- `openRegister()`: Ez a metódus a regisztrációs modál megjelenítésére szolgál. Ha meghívják, a regisztrációs modál láthatóvá válik, és a bejelentkezési modál eltűnik.
- `closeModals()`: Ez a metódus eltünteti mindkét modált, beállítva a `showLogin` és `showRegister` változókat `false` értékre.

HTLM:

- Navigációs pontok
 - Rólunk gomb a navbaron - `#about`
 - Miért gomb a navbaron - `#why`
 - Kapcsolat - `#contract`
 - Bejelentkezés a modális ablakon – `openLogin()` metódust hívja meg.
 - Regisztráció a modális ablakon – `openRegister()` metódust hívja meg.

10.Admin-childrenComponents

A `AdminChildrenComponent` az adminisztrátori felület egyik komponense, amely a gyermekek kezelésére szolgál. Itt a rendszergazdák képesek listázni a gyermekeket, valamint törölni őket. A komponens az adminisztrációs felület kezelésére szolgál, és az alábbiakban részletesen ismertetem a működését.

Funkciók és működés:

Alapvető felépítés:

- `children`: Ez a változó tárolja az összes gyermek adatait, amelyeket az adminisztrátor a felületen megtekinthet.
- `loadChildren()`: Ez a metódus felelős a gyermekek adatainak betöltéséért. Az `adminService.getChildren()` segítségével lekérdezi az adatokat a backendről, és a választ a `children` tömbbe helyezi.
- `deleteChild(id: number)`: Ez a metódus a gyermek törlésére szolgál. Ha a felhasználó megerősíti a törlést egy visszaigazoló párbeszédablakban, akkor az `adminService.deleteChild(id)` metódus segítségével törli a gyermeket az adatbázisból, és újratölti a gyermekek listáját.

Toastr értesítések:

- Sikeres gyermek törlés: Ha a gyermek sikeresen törlődik, akkor egy értesítést jelenít meg a felhasználónak (`this.toastr.success()`).
- Hiba törléskor: Ha a törlés során hiba történik, akkor hibát jelez (`this.toastr.error()`).
- Gyermek betöltése hiba: Ha a gyermek adatainak betöltése nem sikerül, akkor egy hibaüzenetet jelenít meg.

HTML

- Címsorban : Gyermek kezelése
- A tartalom egy táblázatban jelenik meg.
- Fejléc oszlopai: ID, Név, Életkor, Szülő neve.
- Az adatok `*ngFor` ciklussal jelennek meg.

11.Admin-gameComponents

A `AdminGameComponent` egy adminisztrátori felület, amely lehetővé teszi a játékok kezelését, beleértve azok hozzáadását, frissítését és törlését. Az alábbiakban bemutatom a komponens működését és logikáját.

Funkciók és működés:

Formuláris kezelés:

A komponens egy `gameForm` nevű űrlapot használ, amely három mezőt tartalmaz:

- `title`: A játék neve (kötelező mező)
- `description`: A játék leírása (nem kötelező)
- `category`: A játék kategóriája (kötelező mező)

Az űrlapot a `ReactiveFormsModule` segítségével hozzuk létre, és validáljuk a kötelező mezőket.

Alapvető működés:

- `loadGames()`: Betölti a játékokat a backendből az `gameService.getGames()` metódus hívásával, és a választ a games tömbben tárolja.
- `onSubmit()`: Az űrlap adatainak mentésére szolgál. Ha az űrlap érvényes, akkor:
 - o Ha van `editId`, akkor egy meglévő játékot frissítünk az `updateGame()` metódussal.
 - o Ha nincs `editId`, akkor új játékot adunk hozzá az `addGame()` metódussal.
 - o Minden sikeres művelet után a játékok újratöltődnek, és az űrlap törlődik.
- `editGame(game: any)`: Ez a metódus egy meglévő játék szerkesztéséhez szolgál. Ha a játékot szerkesztjük, akkor az űrlap mezőit az adott játékkal tölti fel.
- `deleteGame(id: number)`: A játék törlésére szolgáló metódus. A törlés előtt megerősítő párbeszédablak jelenik meg, majd az `deleteGame()` metódus törli a játékot a backendből.
- Sikeres mentés vagy frissítés: A felhasználó értesítést kap, hogy a játék sikeresen hozzá lett adva vagy frissítve (`this.toastr.success()`).
- Hiba a mentés vagy frissítés során: Ha a mentés vagy frissítés nem sikerül, hibát jelez (`this.toastr.error()`).
 - o Sikeres törlés: A törlés sikeres, és az admin felhasználó értesítést kap.
 - o Hiba törléskor: A törlés során fellépett hiba esetén hibás értesítést kap.

HTML

- Cél: Új játék rögzítése vagy meglévő szerkesztése.
- Mezők:
 - o Title – játékok címe(kötelező)
 - o category – kategória megnevezése
 - o description – opcionális leírás
- Gombok
 - o Hozzáadás/Frissítés
 - o Mégse
- Játékok listázása a `ngFor` ciklussal.
- Oszlopok: ID, Cím, Kategória, Leírás, Műveletek
- Műveleti gombok
 - o `editGame` - játék betöltése szerkesztése űrlapra.
 - o `deleteGame` – játék végleges törlése.

12. Admin-layoutComponents

A AdminLayoutComponent egy adminisztrátori felületet biztosít, amely tartalmazza a napra kész időt és dátumot, valamint a bejelentkezett felhasználó nevét, és lehetőséget ad a kijelentkezésre. Itt van, hogyan működik a komponens:

Funkciók és működés:

- Idő és Dátum Frissítés:
 - o A komponens folyamatosan frissíti az időt és a dátumot a setInterval() függvény használatával, így mindig naprakész információt mutat.
 - o Az idő másodpercenként frissül a currentTime változóban, és a dátum is az currentDate változóban.
- Bejelentkezett Felhasználó Neve:
 - o A komponens a localStorage-ban tárolt tokenet ellenőrzi, és ha a token jelen van, akkor egy HTTP kérésen keresztül lekéri az aktuális felhasználó nevét az AuthService.getCurrentUser() segítségével.
 - o Ha a kérés sikeres, beállítja a felhasználó nevét a userName változóba, különben az alapértelmezett "Admin" nevet használja.
- Kijelentkezés:
 - o A logout() metódus felelős a felhasználó kijelentkeztetéséért. A kijelentkezéshez a következő lépések történnek:
 - o A token törlésre kerül a localStorage-ból.
 - o A felhasználó átirányításra kerül a kezdőoldalra (/).
 - o Ha a logout kérés hiba történik, az alkalmazás akkor is a kezdőoldalra navigál.

HTML

- Ez a komponens biztosítja az admin felületet fő elrendezését, oldalsávval és dinamikusan betölthető tartalommal(router-outlet)
- Oldalsáv
 - Fejléc: Admin panel címsor.
 - Navigációs linkek:
 - /admin/user – felhasználó kezelése
 - /admin/games – Játékok kezelése
 - /admin/children – gyermekek kezelése
- Óra blokk:aktuális dátum és idő megjelenítése currentDate és currentTime
- Kijelentkezés gomb: logout() metódus hívja, amely kijelentkezteti az admint.

13.Admin-userComponents

Az AdminUserComponent a felhasználók kezelésére szolgáló komponens, amely lehetővé teszi a felhasználók betöltését, törlését és adminisztrátori jogosultságokkal történő frissítését. Itt van, hogy hogyan működik a komponens:

Funkciók és működés:

- Felhasználók Betöltése (loadUsers()):
 - Az ngOnInit életciklus-horogban meghívódik a loadUsers() metódus, amely az AdminService getUsers() szolgáltatáson keresztül lekéri a felhasználói adatokat.
 - A válasz (res) sikerességétől függően a felhasználók listáját frissíti. Ha hiba történik, egy hibaüzenetet jelenít meg a ToastrService segítségével.
- Felhasználó Törlése (deleteUser(id: number)):
 - A törlés előtt megerősítést kér a felhasználotól (confirm), hogy biztosan törölni akarja-e a felhasználót.
 - A törlés sikeressége után a felhasználó listája újratöltődik. Ha hiba történik, hibaüzenetet jelenít meg.
- Felhasználó Adminisztrátorrá Lépése (promoteUser(id: number)):
 - Hasonlóan a törléshez, a promoteUser() metódus először megerősítést kér a felhasználotól, hogy adminná akarja-e léptetni a felhasználót.

- o A frissítés sikeressége után a felhasználók listája újra töltődik, és egy visszajelző üzenet jelenik meg.

HTML

- Ez a komponens az admin számára nyújt egy lehetőséget a regisztrált felhasználók áttekintésére és jogosultságainak módosítására.
- *ngFor metódus jeleníti meg a users tartalmát.
- Oszlopok:
 - o ID
 - o Név
 - o Email
 - o Szerep
 - o Műveletek admin funkciókhoz
- Adminná léptetés
 - o prompteUser() metódussal az adott felhasználó szerepe adminisztrátorrá változik.

14.ColoringGameComponents

A ColoringGameComponent egy egyszerű színezős játék komponens, amely lehetővé teszi a felhasználók számára, hogy különböző színekkel színezzék ki a képeken található területeket.

Funkciók és működés

- Színek Lista (colors):
 - o A komponens colors változója tartalmazza a lehetséges színeket, amelyeket a felhasználó kiválaszthat.
 - o Az egyes színek hexadecimális kódok formájában vannak megadva.
- Kiválasztott Szín (selectedColor):
 - o Az aktuálisan kiválasztott szín tárolására szolgáló változó, amely alapértelmezetten a piros (#FF0000).
- Színezett Területek (selectedColors):
 - o A selectedColors objektum tárolja, hogy mely területek (azonosítóval rendelkező elemek) milyen színnel vannak színezve.
 - o Az objektum kulcsa a pathId (ami a terület azonosítója), és az értéke a kiválasztott szín.
- Szín Kiválasztása (onColorClick(pathId: string)):

- o Ez a metódus frissíti a `selectedColors` objektumot az adott `pathId` azonosítóhoz tartozó színnel.
- o Ez akkor történik, amikor a felhasználó egy területre kattint, így az a terület az aktuálisan kiválasztott színt kapja.

HTML

- Színválasztó paletta : a `colors` tömb alapján színes körök jelennek meg, kattintásra beállítjuk a `selecedColor` változót.
- Interaktív rajz felület:
 - o Különböző alakzatok
 - o Az elemel `fill` attribútuma a `selectedColor` értékétől függ.
 - o Kattintásra beállítja a színt az adott elemnek
- Újra gomb – A játék újra indítását kezeli.

15.PuzzleGameComponents

A `PuzzleGameComponent` egy egyszerű, interaktív puzzle játékot valósít meg Angularban. A játékban a felhasználó a képkockákat (puzzle darabokat) húzhatja a megfelelő helyekre, és ellenőrizhetjük, hogy helyesen rendezték-e őket.

Funkciók és működés:

- Képek és Puzzle Darabok
 - o A `imageUrls` változóban tároljuk az elérhető képeket, amelyekből véletlenszerűen választunk egyet, amikor a játék elindul.
 - o A `puzzlePieces` tömb tartalmazza a puzzle darabok pozícióit, valamint azok megfelelő helyét a rácsban.
- Rács Beállításai
 - o A puzzle rácsának méretét a `gridSize` változó határozza meg (például 3x3, tehát 9 darab puzzle rész lesz).
 - o A `pieceSize` változó a puzzle darabok méretét állítja be (100px).

- Kép Kezelése
 - o Az imageUrl változó a kiválasztott kép URL-jét tárolja.
 - o A getRandomImage metódus egy véletlenszerű képet választ a képek közül.
- Puzzle Darabok Inicializálása

Az initializePuzzle metódus létrehozza a puzzle darabokat, meghatározza azok pozícióit, és a darabokat összekeveri a shufflePieces metódussal, hogy a játék kihívást jelentsen.
- Húzás és Áthelyezés
 - o A startDrag metódus elindítja a darab húzását, rögzíti a kezdő pozíciót, és eseménykezelőt ad hozzá a mozgás és az elengedés figyelésére.
 - o A dragPiece metódus frissíti a darab pozícióját a húzás közben.
 - o A stopDrag metódus eltávolítja az eseménykezelőket, amikor a felhasználó elengedi a darabot.
- Megoldás Ellenőrzése
 - o A checkSolution metódus végignézi, hogy a puzzle darabok megfelelő helyeken vannak-e. Ha minden darab a helyére került, akkor egy gratuláló üzenetet jelenít meg.

HTML

- <div class="puzzle-board"> – A kirakó tábla dinamikusan méretezett rows * piecesSize, cols * piecesSize
- *ngFor="let piece of pieces" – A pieces tömb alapján rendereli a kirakódarabokat.
- [ngStyle]="getPieceStyle(piece)" – Beállítja minden darab pozícióját/stílusát.
- (click)="movePiece(piece)" – Kattintással a darab mozgatása történik.
- Új játék gomb – A createPuzzle() új játékelrendezést hoz létre.

16.MemorygameComponents

A MemoryGameComponent egy memóriajátékot valósít meg, ahol a felhasználó párba kell rendezni a kártyákat úgy, hogy a megfelelő párok fel legyenek fordítva. A játék célja, hogy minden párt megtaláljunk, amíg minden kártya a helyére nem kerül.

Funkciók és működés

- Kártyák és játékállapot
 - o A cards tömb tárolja a kártyák adatait, beleértve a kép URL-jét (url), hogy a kártya párosítva van-e (matched), és hogy a kártya le van-e fordítva (flipped).
 - o A flippedCards tömb tartalmazza azokat a kártyákat, amelyeket a felhasználó éppen megfordított.
- Játék Állapotok
 - o A allMatched változó ellenőrzi, hogy minden kártyapár helyesen megtalálásra került-e (nyert a játékos).
- Játék Inicializálása
 - o Az initializeCards metódus inicializálja a kártyákat. Az elérhető képek listáját (imageUrls) duplázzuk, hogy minden képből két kártya legyen. A kártyák véletlenszerűen vannak elrendezve a játéktáblán a sort(() => Math.random() - 0.5) módszerrel.
- Kártyák Megfordítása:
 - o A flipCard metódus felelős a kártyák megfordításáért. Amikor egy kártyát megfordítunk, a flipped érték true-ra változik, és a kártya adatai bekerülnek a flippedCards tömbbe.
 - o Ha két kártya van megfordítva, a checkForMatch metódus ellenőrzi, hogy a két kártya azonos képet tartalmaz-e.
- Párok Ellenőrzése
 - o Ha a két kártya egyezik, a matched értéke true-ra változik, és ellenőrizzük, hogy minden kártya párosítva lett-e a checkForWin metódussal.
 - o Ha a két kártya nem egyezik, akkor egy kis idővel visszafordítjuk őket (1 másodperc), és az flippedCards tömb kiürül.

- Játék Újraindítása
 - o A restartGame metódus újraindítja a játékot, alaphelyzetbe állítja a kártyákat, a flipped kártyákat és a nyert állapotot, majd újra inicializálja a kártyákat.

HTML

- <div class="game-board"> – A játékmező, ahol a kártyák megjelennek
- *ngFor="let card of cards; let i = index" – A cards tömb alapján generálódnak a kártyák
- (click)="flipCard(i)" – Kattintáskor a kiválasztott kártya felfordul.
- Kártyanézet:
 - o Ha felfordítva vagy párosítva van: kép (img).
 - o Egyébként: hátsó oldal(card-back).
- <div *ngIf="allMatched"> – Gratulációs Üzenet, ha minden pár megtalálva.
- Újra gomb – A restartGame újratekdi a játékot

BACKEND

1. API Végpontok

Végpont	Metódus	Azonosítás	Leírás
/register	POST	Nem	Új felhasználó regisztrációja
/login	POST	Nem	Felhasználó bejelentkezése
/logout	POST	Igen	Felhasználó kijelentkezése
/users	GET	Igen (admin)	Felhasználók listázása
/users/{id}/promote-admin	POST	Igen (admin)	Felhasználó adminná léptetése
/admin/users/{id}/deactivate	PATCH	Igen (admin)	Felhasználó deaktiválása
/admin/users/{id}/activate	PATCH	Igen (admin)	Felhasználó aktiválása
/admin/children	GET	Igen (admin)	Összes gyermek listázása
/admin/children/{id}	DELETE	Igen (admin)	Gyermek törlése admin által
/children	GET	Igen	Felhasználó saját gyermekeinek lekérdezése (admin összes)
/children	POST	Igen	Új gyermek hozzáadása
/children/{id}	PUT	Igen	Gyermek adatainak módosítása
/children/{id}	DELETE	Igen	Gyermek törlése
/children/{id}/games	POST	Igen	Játék hozzárendelése gyermekhez
/children/{id}/games	GET	Igen	Gyermek játékainak lekérdezése
/children/{childId}/games/{gameId}	DELETE	Igen	Játék eltávolítása gyermekhez kapcsolva
/games	GET	Igen	Játékok listázása

/games	POST	Igen	Új játék hozzáadása
/games/{id}	PUT	Igen	Játék adatainak módosítása
/games/{id}	DELETE	Igen	Játék törlése
/statistics	GET	Igen	Statisztikák lekérdezése
/statistics	POST	Igen	Statisztikai adat mentése
/user/statistics	GET	Igen	Felhasználó gyermekeinek statisztikái
/parent/statistics	GET	Igen	Szülői statisztikák összesítve
/games/memory-images	GET	Igen	Memóriajáték képek lekérdezése
/user	GET	Igen	Bejelentkezett felhasználó adatainak lekérdezése

Általános működés

A PikkyPad REST API HTTP kéréseket fogad JSON formátumban.

A kényes műveletek (pl. új gyermek hozzáadása, gyermek adatainak módosítása, törlés, játékok kezelése) hitelesítést igényelnek (auth:sanctum token).

Az adminisztrációs funkciókhoz (pl. felhasználók kezelése, adminná léptetés, gyermekek törlése) további admin jogosultság is szükséges (AdminMiddleware).

A regisztráció, bejelentkezés, gyermekek listázása (saját gyermekek) és játékok listázása végpontjai elérhetők minden hitelesített felhasználó számára.

A vezérlést külön kontrollerek (AuthController, AdminController, ChildController, GameController, StatisticsController) végzik.

Az adatokat Eloquent modellek kezelik, az adatbázis táblákkal közvetlen kapcsolatban.

A kommunikáció JSON alapú, minden válasz egységes JSON szerkezetet követ.

2. Adatbázis Struktúra

Users Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Felhasználó egyedi azonosítója (primary key)
name	string	igen	Felhasználó neve
email	string (unique)	igen	Egyedi e-mail cím
email_verified_at	timestamp	nem	E-mail hitelesítési időpont
password	string	igen	Titkosított jelszó
role	enum ('user', 'admin', 'superadmin')	igen	Felhasználói szerepkör
login_counter	integer (default: 0)	nem	Bejelentkezések száma
banning_time	timestamp	nem	Kitiltás ideje
is_active	boolean (default: true)	igen	Aktív státusz
remember_token	string	nem	„Emlékezz rám” token
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

Password Reset Tokens Tábla

Oszlop neve	Típus	Kötelező	Leírás
email	string (primary key)	igen	Token e-mail címmel
token	string	igen	Jelszó-visszaállító token
created_at	timestamp	nem	Token létrehozása

Sessions Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	string (primary key)	igen	Session azonosító
user_id	foreignId (nullable)	nem	Kapcsolódó felhasználó ID
ip_address	string(45)	nem	IP cím
user_agent	text	nem	Böngésző vagy eszköz azonosító
payload	longText	igen	Serializált session adatok
last_activity	integer	igen	Utolsó aktivitás timestamp

Children Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Gyermek egyedi azonosítója (primary key)
user_id	foreignId	igen	Szülő felhasználó ID
name	string	igen	Gyermek neve
age	integer	igen	Gyermek életkora
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

Games Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Játék egyedi azonosítója (primary key)
title	string	igen	Játék címe
description	text	nem	Játék leírása
category	string	igen	Játék típusa
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

Child Game Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Gyermek egyedi azonosítója (primary key)
child_id	foreignId	igen	Kapcsolódó gyermek azonosítója
game_id	foreignId	igen	Kapcsolódó játék azonosítója
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

Statistics Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Gyermek egyedi azonosítója (primary key)
child_id	foreignId	igen	Kapcsolódó gyermek azonosítója
game_id	foreignId	igen	Kapcsolódó játék azonosítója
score	integer	igen	Elért pontszám
play_time	integer	igen	Játékidő másodpercben
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

Personal Access Token Tábla

Oszlop neve	Típus	Kötelező	Leírás
id	integer (auto-increment)	igen	Token egyedi azonosítója
tokenable_type	string	igen	Model típus (pl. App\Models\User)
tokenable_id	integer	igen	Model ID (pl. user ID)
name	string	igen	Token neve
token	string(64)	igen	Egyedi token
abilities	text	nem	Jogosultságok (pl. ["read", "write"])
last_used_at	timestamp	nem	Utolsó használat ideje
expires_at	timestamp	nem	Lejárat ideje
created_at	datetime	nem	Létrehozás ideje
updated_at	datetime	nem	Frissítés ideje

3. RegisterRequest

A RegisterRequest osztály a felhasználói regisztráció során beküldött adatok validálására szolgál.

A FormRequest osztályból öröklődik, és automatikusan ellenőrzi az űrlap mezőit a szerveroldalon, még a tényleges adatbázis műveletek előtt.

Validációs Szabályok

Mező	Szabályok	Leírás
name	required, min:3, max:20, regex:/^[pL\s]+\$/u	A név megadása kötelező, legalább 3, maximum 20 karakter, csak betűket és szóközt tartalmazhat.
email	required, email:rfc,dns, unique:users,email	Az email cím megadása kötelező, érvényes formátumban kell lennie, és egyedinek kell lennie az users táblában.
password	required, min:8, regex kisbetű, nagybetű, szám, speciális karakter	A jelszónak kötelező minimum 8 karakterből állnia, tartalmaznia kell kis- és nagybetűt, számot, speciális karaktert.

confirm_password	required, same:password	A jelszó megerősítése kötelező, meg kell egyeznie a jelszó mezővel.
------------------	-------------------------	---

Hibaüzenetek

name.required | "Név megadása kötelező."

name.min | "A név túl rövid."

name.max | "A név túl hosszú."

name.alpha | "A név csak betűket tartalmazhat, ékezetrel vagy anélkül."

email.required | "Az email cím megadása kötelező."

email.email | "Kérjük, érvényes email címet adjon meg."

email.unique | "Ez az email cím már használatban van."

password.required | "A jelszó megadása kötelező."

password.min | "A jelszó túl rövid."

password.regex | "A jelszónak tartalmaznia kell kisbetűt, nagybetűt, számot és speciális karaktert."

confirm_password.required | "A jelszó megerősítése kötelező."

confirm_password.same | "A megadott jelszavak nem egyeznek."

4. LoginRequest

A LoginRequest osztály a felhasználói bejelentkezési adatok validációját végzi.

Csak két mező meglétét és formátumát ellenőrzi: email és jelszó.

Validációs szabályok

Mező		Szabályok	Leírás
email required, email		Az email cím megadása kötelező, és érvényes email formátumú kell legyen.	
password	required	A jelszó megadása kötelező.	

Hibaüzenetek

Hiba	Üzenet
email.required	"Email megadása kötelező."
email.email	"Érvénytelen email formátum."
password.required	"Jelszó megadása kötelező."

Sikertelen validáció esetén, a rendszer JSON válasszal reagál.

5. AuthController

Az AuthController kezeli a teljes regisztrációs, bejelentkezési, kijelentkezési folyamatot. Biztonsági funkciók: fiókaktiválás, sikertelen login blokkolás, e-mail értesítések. Az összes válasz JSON formátumban történik, egységesen a ResponseController segítségével.

RegisterRequest - Új felhasználó regisztrációja

Validálja a kérés adatait a RegisterRequest alapján (név, email, jelszó, jelszó megerősítés).

Új felhasználót hoz létre az users táblában:

name

email

password (bcrypt-tel titkosítva)

Megpróbál egy üdvözlő e-mailt küldeni (UserMail@sendWelcomeMail).

Ha az email küldés sikertelen, logolja a hibát, de nem akadályozza meg a regisztrációt. Amennyiben hiba van, 422-es hibakóddal válaszol a RegisterRequest osztályban megadott üzenetekkel.

LoginRequest – Felhasználó bejelentkezése

Validálja az email és jelszó mezőket a LoginRequest alapján. Megpróbál bejelentkezni az Auth::attempt() segítségével.

Sikeres bejelentkezés esetén:

Ellenőrzi, hogy a felhasználó aktív-e (is_active mező). Ha nem aktív: 403 hibakóddal hibaüzenetet küld. Ellenőrzi, hogy van-e kitiltás (banning_time), ha a kitiltás még érvényes: 405 hibakód és visszatér a tiltás lejáratí időpontjával. Ha minden rendben, új API tokent generál (createToken()), login számlálót nullázza (resetLoginCounter), kitiltási időt törlési (resetBanningTime).

Sikertelen bejelentkezés esetén:

Megnöveli a sikertelen belépési próbálkozások számát (`setLoginCounter`). Ha a próbálkozások száma meghaladja a 3-at, beállít egy kitiltási időt (1 perc), és Email értesítést küld (`UserMail@AllertMail`). 401 hibakóddal válaszol: "Hibás email vagy jelszó (le lett tiltva egy időre)".

Amennyiben csak egyszerű hiba van (pl. rossz jelszó), szintén 401 hibakódot ad vissza: "Hibás email vagy jelszó".

Logout – Felhasználó kijelentkezése

A bejelentkezett felhasználó (`auth('sanctum')->user()`) aktuális tokenjét törli (`currentAccessToken()->delete()`).

JSON válasz:

```
{
  "success": true,
  "data": "felhasználó neve",
  "message": "Sikeres kijelentkezés"
}
```

6. BannerController

A BannerController osztály célja, hogy a felhasználók belépési próbálkozásait és ideiglenes kitiltását kezelje. Segítségével korlátozható, hogy hány sikertelen próbálkozás után tiltódjon le egy fiók egy rövid időre. Minden metódus az users táblában dolgozik az adott email cím alapján. A login folyamat során közvetlenül hívódik meg (AuthController).

getLoginCounter

Lekérdezi a megadott email címhez tartozó felhasználó bejelentkezési kísérleteinek számát. Megkeresi a felhasználót az email cím alapján. Ha talál, visszaadja a login_counter értékét. Ha nem talál, 0-t ad vissza.

setLoginCounter

Megnöveli a megadott email-hez tartozó felhasználó sikertelen belépési próbálkozásainak számát. Megkeresi a felhasználót az email cím alapján. Ha talál, az `increment("login_counter")` metódussal eggyel növeli a login_counter értékét. Mentés automatikusan történik. A tiltás logikájában játszik szerepet.

resetLoginCounter

Visszaállítja a login_counter értékét 0-ra a megadott email-hez tartozó felhasználónál. Megkeresi a felhasználót az email cím alapján. Ha talál, a login_counter értékét 0-ra állítja, majd menti. Sikeres bejelentkezés után hívódik meg.

getBanningTime

Lekérdezi a megadott email-hez tartozó felhasználó aktuális kitiltási időpontját (banning_time). Megkeresi a felhasználót az email cím alapján. Ha talál, visszaadja a banning_time értékét. Ha nem talál vagy nincs kitiltva, nullát ad vissza. A rendszer ellenőrzi, hogy egy adott felhasználó még tiltás alatt áll-e.

setBanningTime

Beállít egy új kitiltási időt a megadott email-hez tartozó felhasználónál. Megkeresi a felhasználót az email cím alapján. Beállítja a banning_time mezőt az aktuális időpont +1 perc értékre (Carbon::now()->addMinutes(1)), majd elmenti a változást. Tipikusan 3 sikertelen próbálkozás után hívódik meg automatikusan. Ideiglenes tiltást valósít meg.

resetBanningTime

Eltávolítja a kitiltási időt a megadott email-hez tartozó felhasználónál. Megkeresi a felhasználót az email cím alapján. Nullázza a banning_time mezőt, majd menti a változást. Sikeres belépés után automatikusan resetelésre kerül.

7. ResponseController

A ResponseController célja, hogy egységes JSON válaszformátumot biztosítson az API végpontok számára. Használata garantálja, hogy a sikeres és sikertelen válaszok is konzisztens szerkezetben érkezzenek a frontend felé.

sendResponse (\$data, \$message)

Sikeres kérésre adott egységes szerkezetű válasz. Létrehoz egy JSON választ, amely a következőket tartalmazza:

success: true (mindenképp true, mivel sikeres válasz)

data mező: a visszaadandó adatok

message mező: szöveges üzenet (pl. "Sikeres bejelentkezés", "Adatok sikeresen lekérve")

HTTP státuszkód: 200 OK

Minden sikeres adatküldéshez használható, mint például regisztráció, belépés, adatlekérdezés. Ez biztosítja az egységes API válaszszerkezetet.

sendError (\$error, \$errorMessage)

Sikertelen kérésre adott egységes szerkezetű hiba válasz. Létrehoz egy JSON választ, amely tartalmazza:

success: false (fix érték sikertelenség esetén)

message mező: rövid hibaüzenet (pl. "Hibás jelszó", "Felhasználó nem található")

(opcionálisan) errorMessage: részletes hibaüzenet(ek), ha vannak

HTTP státuszkód:

Alapértelmezett: 404 Not Found

Átadható más kód is pl. 422 (validációs hiba), 403 (hozzáférés megtagadva) stb. Ha nincs részletezés akkor csak succes és message mező lesz.

8. ChildController

A ChildController teljes körűen kezeli a gyermekek adminisztrációját: létrehozás, olvasás, módosítás, törlés. Játékok hozzárendelése és eltávolítása pivot táblán keresztül történik. Jogosultsági szintek megfelelően ellenőrizve (szülő vs admin). JSON válaszok egységes szerkezettel segítik a frontend integrációt.

Child_query (Request \$request)

Gyermekek adatainak lekérdezése a bejelentkezett felhasználó jogosultsága szerint.

Admin szerepkör:

Ha a felhasználó admin szerepű:

Az összes gyermek és szülő neve lekérdezésre kerül.

Szülői:

Csak a saját gyermekei, és azokhoz tartozó játékok listázódnak.

add_child(Request \$request)

Új gyermek létrehozása a bejelentkezett szülő számára.

Validálja a kérést:

name: kötelező, string, max 255 karakter

age: kötelező, integer, 3–5 év között

Létrehoz egy új gyermek rekordot, az aktuális felhasználó (user_id) alatt.

update(Request \$request, \$id)

Gyermekek adatainak frissítése.

Validálja a kérést:

name: opcionális, ha megadott: string

age: opcionális, integer, 3–5 év között

Megkeresi a gyermeket ID alapján.

Jogosultság ellenőrzés:

Szülő: csak a saját gyermekét módosíthatja.

Frissíti a megadott mezőket. Jogosultság hiánya esetén 403-as hibakóddal tér vissza.

destroy(Request \$request, \$id)

Gyermek törlése az adatbázisból. Megkeresi a gyermeket ID alapján.

Jogosultság ellenőrzés:

Szülő: csak saját gyermek törölhető.

Admin: bármely gyermek törölhető.

Sikeres törlés esetén visszaad egy üzenetet. Jogosultság hiánya esetén 403-as hibakóddal tér vissza.

assignGame(Request \$request,\$child_id)

Játék hozzárendelése egy gyermekhez. Validálja, hogy a game_id létezik-e a games táblában. Megkeresi a gyermeket ID alapján. A games() kapcsolat segítségével hozzákapcsolja a megadott játékot.

getGamesForChild(\$id)

Egy adott gyermekhez tartozó játékok lekérdezése. Megkeresi a gyermeket ID alapján, majd visszaadja a gyermekhez rendelt összes játékot.

removeGame(\$child_id, \$game_id)

Egy játék eltávolítása egy gyermekhez rendelt játékok közül. A bejelentkezett szülő saját gyermekei közül keresi ki az adott gyermeket. Leválasztja (detach()) a megadott játékot a kapcsolatból.

9. GameController

A GameController kezeli az összes játék CRUD műveletet: létrehozás, olvasás, frissítés, törlés. A getMemoryImages() külön funkcióval kiszolgálja a memóriajátékban szükséges képeket. JSON válaszok egységes szerkezettel segítik a frontend integrációt.

game_query()

Az összes játék adatainak lekérdezése az adatbázisból. Lekéri az összes rekordot a games táblából (Game::all()), majd visszaadja az összes játék adatait JSON válaszban.

add_game(Request \$request)

Új játék hozzáadása az adatbázishoz.

Validálja a bemeneti adatokat:

title kötelező, max 255 karakter

description opcionális

category kötelező

Létrehoz egy új játék rekordot a games táblában- majd visszaküldi az új játék adatait JSON válaszban.

update(Request \$request, \$id)

Meglévő játék adatainak módosítása.

Validálja a bemeneti adatokat:

title opcionális, ha megadott: max 255 karakter

description opcionális

age_group opcionális, integer típus

Megkeresi az adott játékot ID alapján (Game::find(\$id)).

Ha nem található:

404 hibaüzenet visszaküldése.

Ha megtalálható:

Frissíti a megadott mezőket.

Mentés után visszaküldi az új adatokat JSON válaszban.

destroy(\$id)

Játék törlése az adatbázisból. Megkeresi az adott játékot ID alapján (Game::find(\$id)).

Ha nem található:

404 hibakóddal tér vissza.

Ha megtalálható:

Törli a játék rekordot.

Visszaküld egy sikeres törlés üzenetet.

getMemoryImages()

A memóriajátékhoz szükséges képek listájának lekérdezése a szerverről. A public/img/games/memory könyvtárban található képfájlokat beolvassa (File::files). Mindegyik képre teljes elérési útvonalat (asset()) generál. A képek URL-jeit visszaadja egy tömbben JSON válaszban.

10.StatisticsController

Ez a kontroller felelős a gyermekekhez tartozó játékmenet-statisztikák rögzítéséért és lekérdezéséért.

A statisztikák célja, hogy visszajelzést adjanak a gyermekek fejlődéséről.

get_statistics()

Minden statisztikai rekord lekérdezése az adatbázisból (admin vagy fejlesztési célokra használható). A Statistics::all() segítségével lekéri az összes rekordot a statistics táblából. Nem szűr jogosultság szerint, minden adatot visszaad.

save_Statistics(Request \$request)

Új statisztikai rekord mentése a gyermek adott játékához.

Validáció:

Mező		Kötelező	Leírás
game_id	igen	Létező játék azonosító (exists:games,id)	
child_id		igen	Létező gyermek azonosító (exists:children,id)
score		igen	Elért pontszám, egész szám
play_time	igen	Játékidő másodpercben	

Validálja a bemeneti adatokat, és létrehoz egy új rekordot a statistics táblában. Visszaküldi a mentett adatokat JSON válaszban.

userStatistics (Request \$request)

Az aktuális bejelentkezett felhasználó gyermekeihez tartozó részletes statisztikai adatok lekérdezése.

Jogosultság: Csak bejelentkezett szülők számára elérhető (auth:sanctum middleware).

Lekérdezi a saját gyermekek statisztikáit (nem másokét).

Működés:

Lekéri a bejelentkezett felhasználót (Auth::user()).

Lekérdezi a gyermekek ID-it. Visszaadja a statisztikákat, játékokkal együtt (with('game')).

parentStatistics(Request \$request)

Az aktuális bejelentkezett szülő gyermekeihez tartozó összesített statisztikák lekérdezése.

Lekéri az aktuális felhasználót (Auth::user()).

Lekérdezi az ő gyermekeit.

Csoportosítva visszaadja:

child_id

hány játékot játszott (games_played)

összpontszám (total_score)

11.AdminController

Az AdminController célja az admin jogosultságú felhasználók számára elérhető műveletek biztosítása. Ide tartozik a felhasználók lekérdezése, jogosultságmódosítás, deaktiválás, újraaktiválás, valamint gyermekek adminisztratív kezelése. Minden metódus csak admin vagy superadmin számára elérhető (AdminMiddleware által védett útvonalakon).

index()

Az összes felhasználó és a hozzájuk tartozó gyermekek lekérdezése. Lekéri az összes User-t és az azokhoz tartozó children kapcsolatot (User::with('children')). Visszaadja őket JSON válaszban.

deactivateUser(\$id)

Egy nem-admin felhasználó deaktiválása (fiók felfüggesztése) admin által.

Lekérdezi a felhasználót ID alapján. Ha nem található: 404 Not Found hibát küld vissza. Ha a felhasználó admin vagy superadmin, akkor a rendszer megtiltja a deaktiválást (403 Forbidden). Adminisztrátor nem deaktiválhat másik admin vagy superadmin fiókot. Ha a felhasználó role mezője user, akkor beállítja az is_active = false értéket. Elmenti a változást, és visszatér egy sikeres üzenettel.

activateUser(\$id)

Egy korábban deaktivált felhasználó újraaktiválása. Lekérdezi a felhasználót ID alapján. Ha megtalálható, is_active = true, mentés. Amennyiben a felhasználó nem található, 404-es hibakóddal tér vissza

promoteToAdmin(\$id)

Egy felhasználó adminná léptetése. Lekérdezi a felhasználót ID alapján. Ha nem található akkor 404-es hibát küld. Ha a felhasználó már admin akkor 400-as a válasz ("Ez a felhasználó már admin."),ellenkező esetben role = 'admin', mentés.

listChildren()

Az összes gyermek lekérdezése, szülői adatokkal együtt (admin számára). Child::with('user') segítségével minden gyermek és kapcsolódó szülő betöltése. A válasz JSON formában történik.

deleteChild(\$id)

Egy adott gyermek törlése az adatbázisból (jelen esetben admin által). Gyermek lekérdezése ID alapján. Ha nem található akkor 404-es hibát küld. Egyébként törlés, és sikeres válasz visszaadása.

12.SuperAdminController

Ez a kontroller kizárólag superadmin jogosultságú felhasználók számára biztosít speciális műveleteket.

Fő célja az adminok listázása, valamint adminisztrátorok előléptetése superadmin szerepkörbe. Hozzáférés csak akkor engedélyezett, ha a felhasználó szerepköre superadmin.

listAdmins()

Az összes admin és superadmin szerepkörű felhasználó listázása. A users tábla lekérdezése, ahol a role értéke admin vagy superadmin. Az eredmény egy tömbként visszatér, amely tartalmazza az admin jogosultságú felhasználókat.

promoteToSuperAdmin(\$id)

Egy meglévő admin felhasználó előléptetése superadmin szerepkörbe. A rendszer megkeresi a felhasználót az adott ID alapján. Ha a felhasználó nem található → 404 Not Found hiba. Ha a felhasználó nem admin (pl. sima user) → 403 Forbidden hiba. Ha a felhasználó admin, akkor beállítja a role mezőt superadmin-ra. Elmenti a változást, és sikeres válasz üzenetet küld vissza.

13.MailController

A MailController jelen esetben két célból küld automatikus leveleket a felhasználóknak:

1. Regisztráció után üdvözlő levél (sendWelcomeMail)
2. Túl sok sikertelen bejelentkezés után figyelmeztető levél (sendMail)

Mindkét esetben a Laravel Mail::to(...)->send(...) metódusát használja, amely egy megfelelő Mailable osztály segítségével sablon alapján generálja a levél tartalmát.

Metódus	Milyen esemény váltja ki?	Küldött sablon	Mailable osztály
sendWelcomeMail(\$email, \$name)	Sikeres regisztráció	welcome.blade.php	WelcomeMail
sendMail(\$user, \$time)	3 sikertelen bejelentkezés után (ideiglenes tiltás)	alert.blade.php	AllertMail

14.Child Modell

A Child modell a gyermekeket reprezentálja az alkalmazásban. A modell a children nevű adatbázis-táblához kapcsolódik, és tárolja a gyermek nevét, életkorát, illetve azt, hogy melyik szülőhöz tartozik.

Attribútumok:

- name: a gyermek neve
- age: a gyermek életkora
- user_id: a szülő (felhasználó) azonosítója – idegen kulcs a users táblára

Kapcsolatok:

- user()

Típus: BelongsTo

Minden gyermek egy szülőhöz tartozik (kapcsolat a users táblával).

- games()

Típus: BelongsToMany

Egy gyermek több játékot is játszhat, a kapcsolatot a child_game pivot tábla kezeli.

További Kezelés:

- getAvatarUrlAttribute()

Automatikusan visszaadja a gyermek avatar képének URL-jét.

Ha nincs megadva avatar, akkor egy alapértelmezett képet (dog_avatar.jpg) ad vissza.

További beállítások:

- fillable: name, age, user_id – tömeges adatkitöltés engedélyezett ezekre.

A fillable → azt szabályozza, hogy mely mezőket lehet tömegesen (pl. create()-tel) biztonságosan menteni a modellben. Ez Laravel beépített védelme az ellen, hogy véletlenül vagy rosszpindulatúan olyan mezők is elmentődjenek, amiket nem lenne szabad (pl. role, admin, is_active stb.).

- appends: avatar_url – JSON válaszokban automatikusan megjelenik.

15.Game Modell

A Game modell az alkalmazásban a játékokat reprezentálja. A games nevű adatbázistáblához kapcsolódik, és minden egyes játék alapinformációit tárolja: cím, leírás és kategória.

Attribútumok:

- title: a játék címe
- description: szöveges leírás, mit tartalmaz vagy hogyan működik a játék
- category: a játék típusa (pl. memória, logikai, színezős)

Kapcsolatok:

- children()

Típus: BelongsToMany

Egy játékhoz több gyermek is kapcsolódhat, és egy gyermek is több játékot játszhat. A kapcsolatot a child_game pivot tábla kezeli.

További kezelés:

- getPuzzleImageUrlAttribute()

Ez az accessor metódus automatikusan hozzáad egy puzzle_image_url mezőt a JSON válaszokhoz, ami a kirakós játék képének URL-jét adja vissza.

Statikus fájlútvonalat ad vissza: img/games/puzzle/lionpuzzle.jpg

További beállítások:

- fillable: title, description, category – ezek a mezők tömegesen kitölthetők (pl. create() vagy update() használatával).
- appends: puzzle_image_url – ez a mező automatikusan megjelenik a JSON válaszokban.

16. Statistics Modell

A Statistics modell a gyermekek játékhöz kapcsolódó teljesítményadatait tárolja. Ez a modell szolgál a pontszámok és a játékidő rögzítésére és lekérdezésére. A statistics nevű adatbázis-táblához kapcsolódik.

Attribútumok:

- child_id: a gyermek azonosítója (idegen kulcs a children táblához)
- game_id: a játék azonosítója (idegen kulcs a games táblához)
- score: elért pontszám
- play_time: játékidő másodpercben

Kapcsolatok:

- child()

Típus: BelongsTo

Minden statisztikai rekord egy gyermekhez tartozik.

- game()

Típus: BelongsTo

Minden statisztikai rekord egy adott játékhoz kapcsolódik.

További beállítások:

- fillable: child_id, game_id, score, play_time – ezek a mezők tömegesen kitölthetők, például a create() metódus segítségével.

17. User Modell

A User modell az alkalmazás felhasználóit reprezentálja. Ez lehet szülő, admin vagy superadmin szerepkörű felhasználó. A modell a Laravel Authenticatable osztályából öröklődik, így alkalmas hitelesítésre, jelszó kezelésére, valamint token-alapú API-hitelesítésre (Sanctum).

Attribútumok:

- name: a felhasználó neve

- email: e-mail cím (egyedi)
- password: jelszó (bcrypt-tel titkosított)
- role: szerepkör (user, admin, superadmin)
- email_verified_at: e-mail megerősítés időpontja (opcionális)
- remember_token: automatikus bejelentkezéshez

Kapcsolatok:

- children()

Típus: HasMany

Jelentés: egy szülő több gyermekhez kapcsolódhat a children táblában (kapcsolómező: user_id).

További metódusok:

- isAdmin()
- Visszatér: true, ha a felhasználó szerepköre admin vagy superadmin
- Használat: jogosultság-ellenőrzés API útvonalaknál
- isSuperAdmin()
- Visszatér: true, ha a felhasználó superadmin
- Használat: kizárólagos superadmin funkciók ellenőrzésére

További beállítások:

- fillable: name, email, password, role – ezek engedélyezettek tömeges mentésre.
- hidden: password, remember_token – ezek nem jelennek meg JSON válaszban.
- casts: a password mezőt automatikusan hasheli, az email_verified_at mezőt datetime típusú alakítja.

18.AdminMiddleware

A AdminMiddleware célja, hogy megvédje azokat az útvonalakat, amelyek kizárólag admin vagy superadmin jogosultságú felhasználók számára elérhetők.

Szerepe a rendszerben:

- Ellenőrzi, hogy a bejelentkezett felhasználó rendelkezik-e admin jogosultsággal.
- Ha nem, a kérést megakadályozza, és 403 Forbidden hibát küld vissza.
- Ha igen, továbbengedi a kérést a következő réteg felé (\$next(\$request)).

Működés:

- A `handle()` metódus lekéri az aktuális felhasználót a kérésből (`$request->user()`).
- Meghívja a `User` modell `isAdmin()` metódusát, amely `true`-t ad vissza, ha a felhasználó admin vagy superadmin.
- Ha a felhasználó nincs bejelentkezve, vagy nem admin: HTTP státusz kód: 403 Forbidden
- Ha az ellenőrzés sikeres a kérés továbblép a következő feldolgozási lépésre.

19.AppServiceProvider

Az `AppServiceProvider` a Laravel alkalmazás globális konfigurációs pontja. Ebben a fájlban határozhatók meg olyan jogosultsági szabályok (gate-ek), amelyek az alkalmazás egészére érvényesek. Ezek a szabályok a rendszer különböző pontjain (pl. middleware-ek, view-k, kontrollerek) hivatkozhatók a Gate mechanizmus segítségével. A jelenlegi implementáció célja az adminisztrációs szerepkörök meghatározása és szabályozása.

Admin Gate

A `admin` nevű jogosultsági szabály akkor engedélyez műveleteket, ha a felhasználó szerepköre `admin` vagy `superadmin`. Ezt a gate-et azoknál a műveleteknél alkalmazza az alkalmazás, ahol alap szintű admin jogosultság szükséges (pl. gyermekek kezelése, játékok létrehozása, felhasználók listázása).

Superadmin Gate

A `superadmin` gate csak abban az esetben ad engedélyt, ha a bejelentkezett felhasználó szerepköre kizárólag `superadmin`. Ezt a szabályt kiemelten biztonságos műveletekhez használjuk, például más adminok előléptetéséhez vagy a legmagasabb szintű rendszerbeállításokhoz való hozzáféréshez.

20.Seederek

A rendszer indulásakor a `DatabaseSeeder` és a hozzá kapcsolódó `GameSeeder` gondoskodik az alapadatok betöltéséről. Ezek a seederek fejlesztéshez, teszteléshez vagy bemutatókhoz biztosítanak előre létrehozott felhasználókat és játékokat.

DatabaseSeeder

A `DatabaseSeeder` célja az alkalmazás induláskori alapfeltöltése. A `run()` metódusban három felhasználó kerül létrehozásra:

- Egy `Test User` gyári adatokkal (`factory`)
- Egy `Admin` felhasználó fix jelszóval és `admin` szerepkörrel
- Egy `SuperAdmin` felhasználó fix jelszóval és `superadmin` szerepkörrel

A `password` mezők `bcrypt` titkosítással kerülnek tárolásra (`Hash::make()`).

A seeder a felhasználók létrehozása után meghívja a `GameSeeder` osztályt is, amely játékadatokat tölt be az adatbázisba.

GameSeeder

A GameSeeder feladata három előre definiált játék létrehozása a games táblában.

A Game::insert() metódus segítségével tömbként rögzíti a játékokat.

A létrehozott játékok:

- Memória játék

Kép párosító játék, amely fejleszti a vizuális memóriát és a megfigyelőképességet.

- Puzzle játék

Kirakós játék, amely a logikai gondolkodást és a formafelismerést ösztönzi.

- Színező játék

Formák kiszínezése, amely a kéz-szem koordinációt és a színfelismerést támogatja.

Mindhárom játékhoz automatikusan beállításra kerül a created_at és updated_at mező az aktuális időponttal (now()).

Seeder Indítás

A seederek futtatása a php artisan db:seed paranccsal történik. Ez létrehozza az alapértelmezett adminisztrációs fiókokat és játékokat. Fejlesztés során bármikor újrafuttatható az adatbázis feltöltéséhez.

TESZTELÉS

Felhasználói Tesztelés

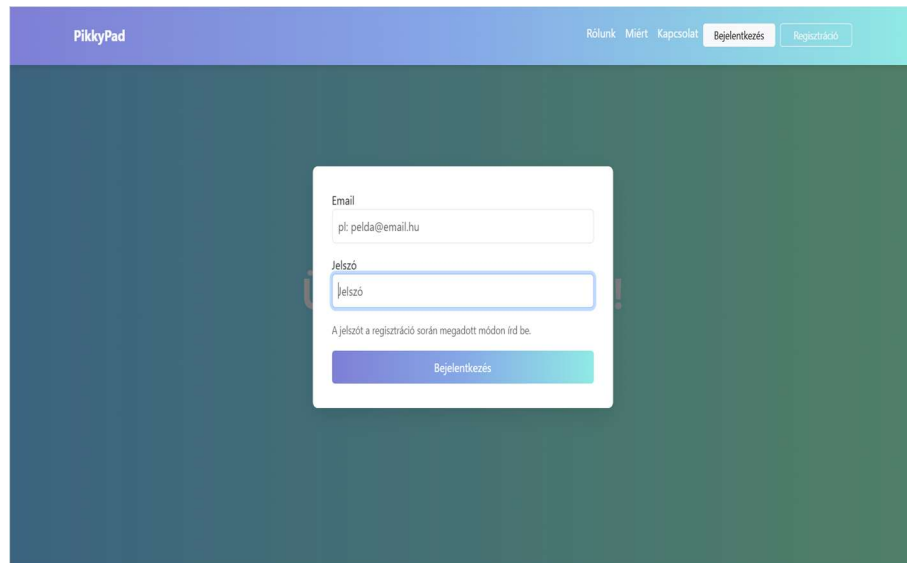
Tesztelés célja

A tesztelés célja annak ellenőrzése, hogy a végfelhasználó (szülő) egyszerűen, hibamentesen tudja használni az alkalmazás fő funkcióit, ideértve a regisztrációt, bejelentkezést, gyermekek kezelését, játékok indítását, valamint a kijelentkezést. A cél az esetleges hibák, használhatósági problémák feltárása és a felhasználói élmény javítása.

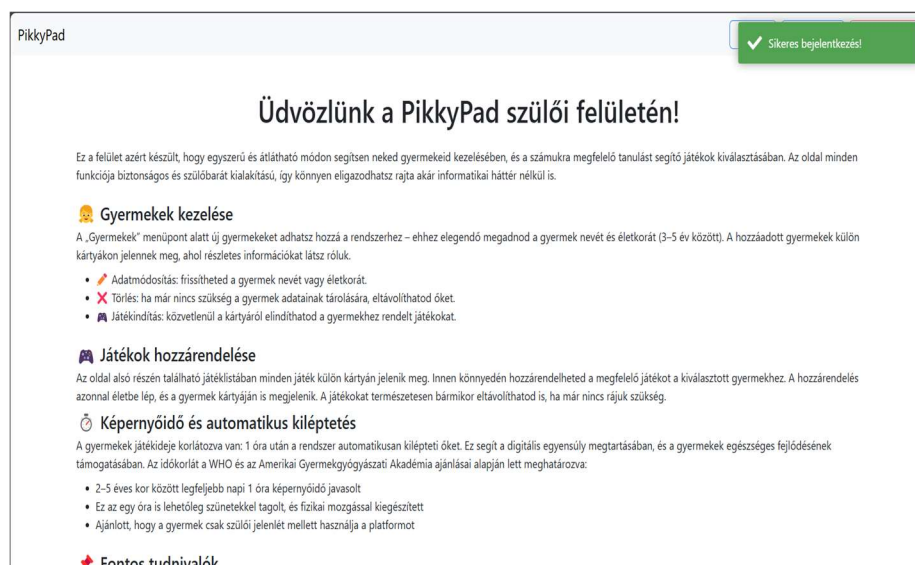
#	Teszteset leírása	Elvárt eredmény	Tesztelt	Sikeres
1	Felhasználó regisztrál érvényes emaillel és jelszóval	Sikeres regisztráció, automatikus bejelentkezés	✓	✓
2	Bejelentkezés létező felhasználóval	Átírányítás a gyermekek felületére	✓	✓
3	Új gyermek hozzáadása (név megadása)	A gyermek megjelenik a listában	✓	✓
4	Gyermek nevének módosítása	A lista frissül, új név látható	✓	✓
5	Gyermek törlése a listából	A gyermek eltűnik a listából	✓	✓
6	Gyermek kiválasztása és játékok megtekintése	A játéklista megjelenik	✓	✓
7	Játék kiválasztása és indítása	A játék elindul vagy részletes nézetre vált	✓	✓
8	Kijelentkezés gomb használata	Visszatérés a bejelentkezési képernyőre	✓	✓

Képernyőképek Frontend Képek

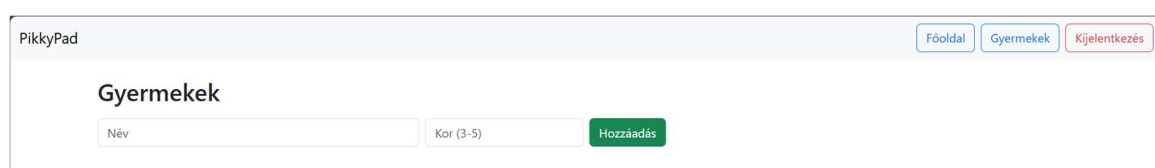
- Bejelentkezési képernyő betöltődik



- Sikeres Bejelentkezés esetén: jobb sarokban felugró ablak „Sikeres bejelentkezés felirattal, át irányít a privát oldalra.



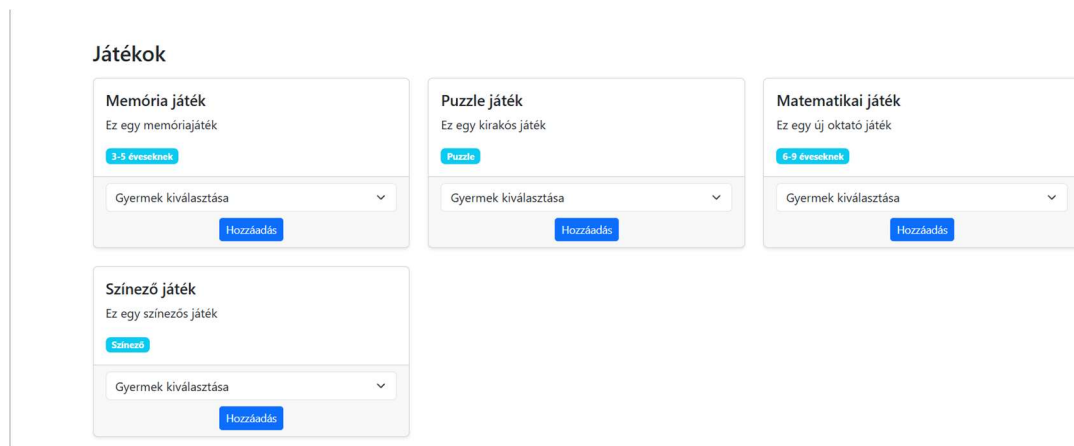
- Gyermekek hozzáadása űrlap sikeresen betöltődik



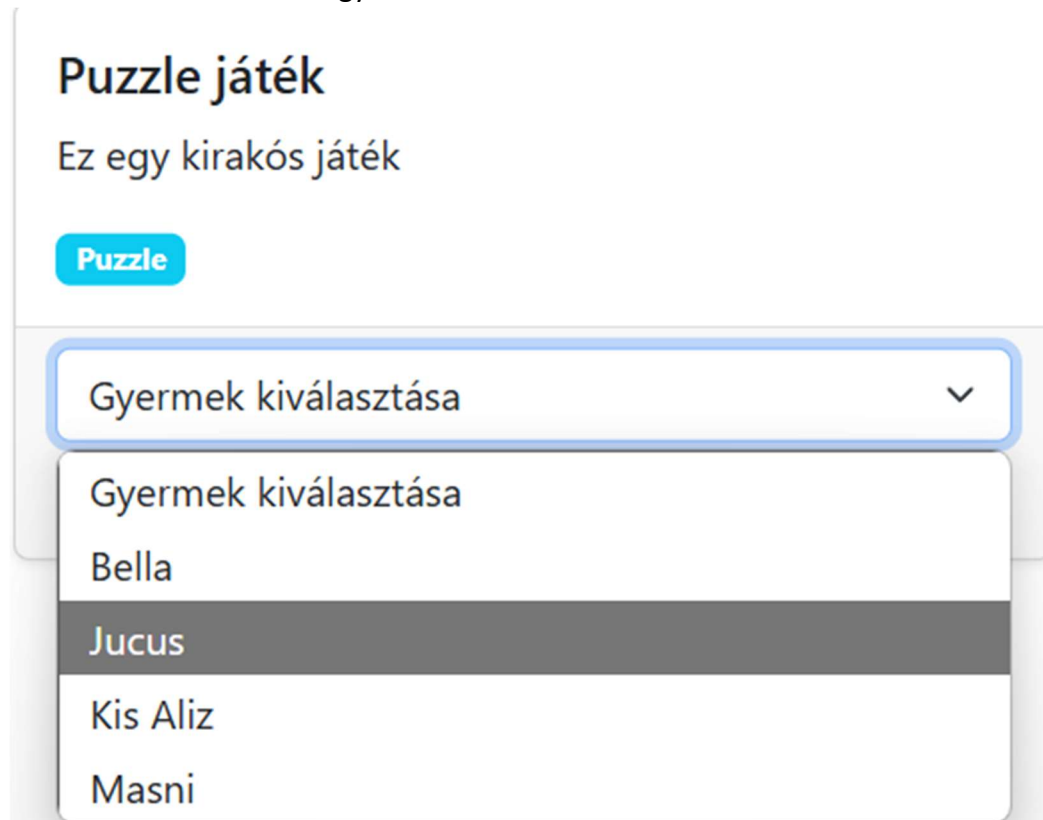
- Sikeres gyermek hozzáadás



- Játék hozzárendelése gyerekekhez



- Játék hozzárendelése gyerekekhez



- Játék törlése gyerektől: X re kattintva törlődik a gyermek játékok listájából.






Jucus

Kor: 6 év

SzerkesztésTörlésJáték indítása

Játékok:

Matematikai játék	
Puzzle játék	
Memória játék	


- Üdvözlő emaild érkezik regisztrációkor.

←

15/3

Üdvözlünk a **PikkyPad**-en!

Beérkező levelek x

**PikkyPad** <pikkypad@gmail.com>
címezett: én

ápr. 12., Szo 12:57 ☆ ☺ ↶ ⋮

Üdvözlünk, Heszler István!

Köszönjük, hogy csatlakoztál a **PikkyPad** közösséghez! 🐶

Mostantól gyermeked játszva tanulhat a **PikkyPad** segítségével – biztonságos és élvezetes környezetben, 3–5 évesekre szabott játékokkal és eszközökkel.

Ha bármikor kérdésed lenne, keress bennünket bátran az info@pikkypad.hu címen!

Üdvözlettel,
🐶 **PikkyPad** csapat

- Fiók tiltása email érkezik

Felhasználó tiltása

Beérkező levelek x

**PikkyPad** <pikkypad@gmail.com>
címezett: én

ápr. 12., Szo 12:42 ☆ ☺ ↶ ⋮

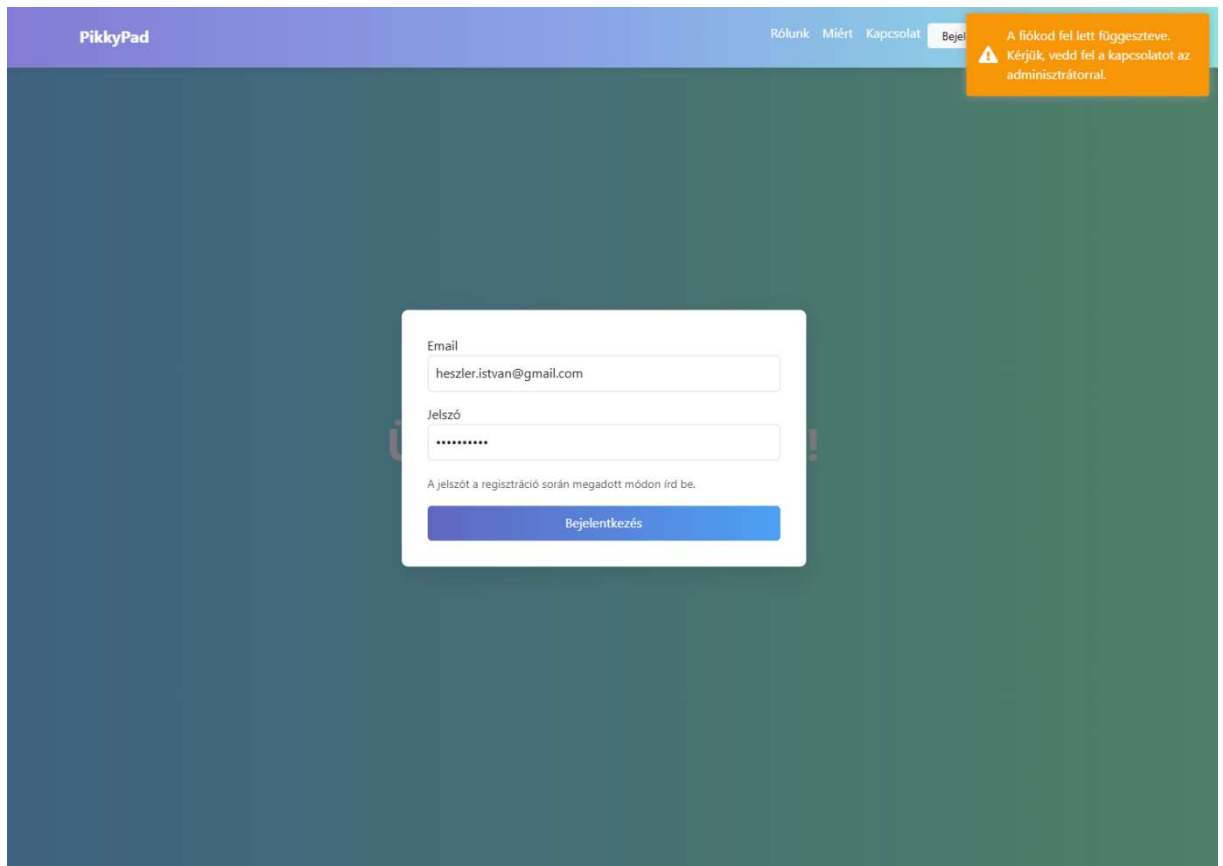
heszler.istvan@gmail.com

Kedves felhasználó: heszler.istvan@gmail.com

Sajnálattal értesítünk, hogy fiókod ideiglenes tiltásra került.

Időpont: 2025-04-12 10:42:16

- Fiókod tiltás alatt.



- Inaktívvá tétel adminként

Admin panel

Felhasználók

Játékok

Gyermekek

2025. 04. 25.
11:01:49

Kijelentkezés

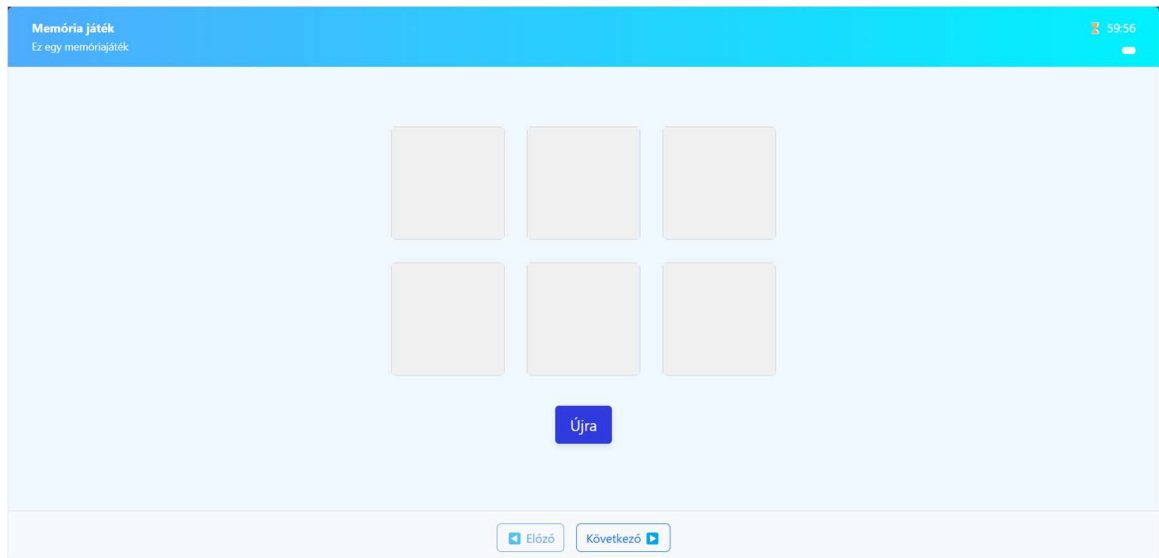
Felhasználók kezelése

✓ Felhasználó inaktívvá.

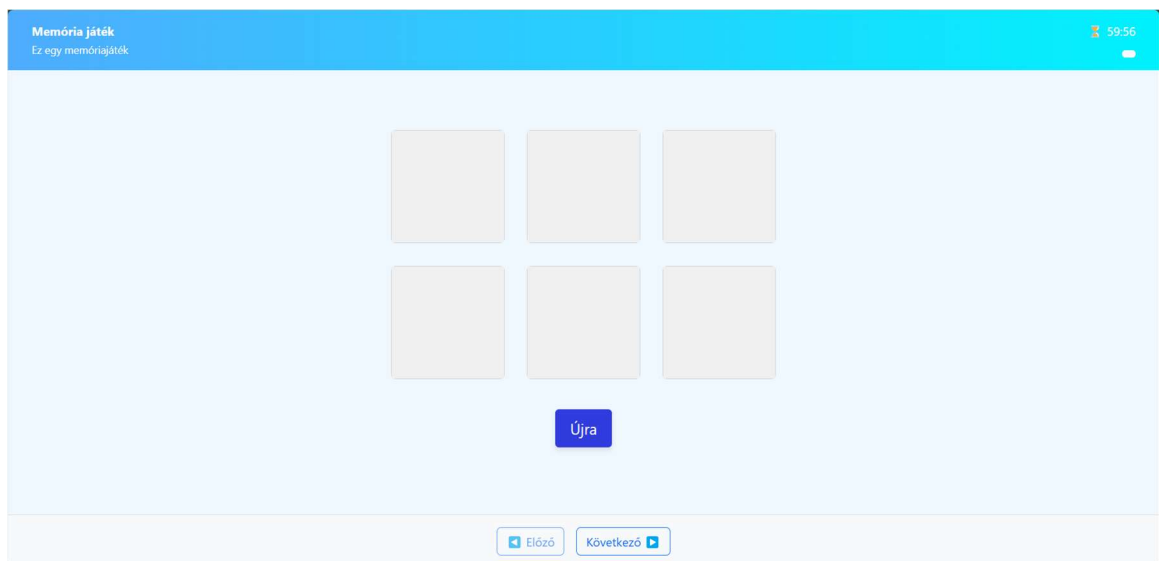
ID	Név	Email	Szerep	Műveletek
1	Test User	test@example.com	user	<div>Adminná léptetés</div> <div>Inaktíválás</div>
2	Admin	admin@example.com	admin	<div>Adminná léptetés</div>
3	SuperAdmin	superadmin@example.com	superadmin	<div>Adminná léptetés</div>
4	Heszler István	heszler.istvan@gmail.com	user	<div>Adminná léptetés</div> <div>Aktiválás</div>

- Játék Indítása

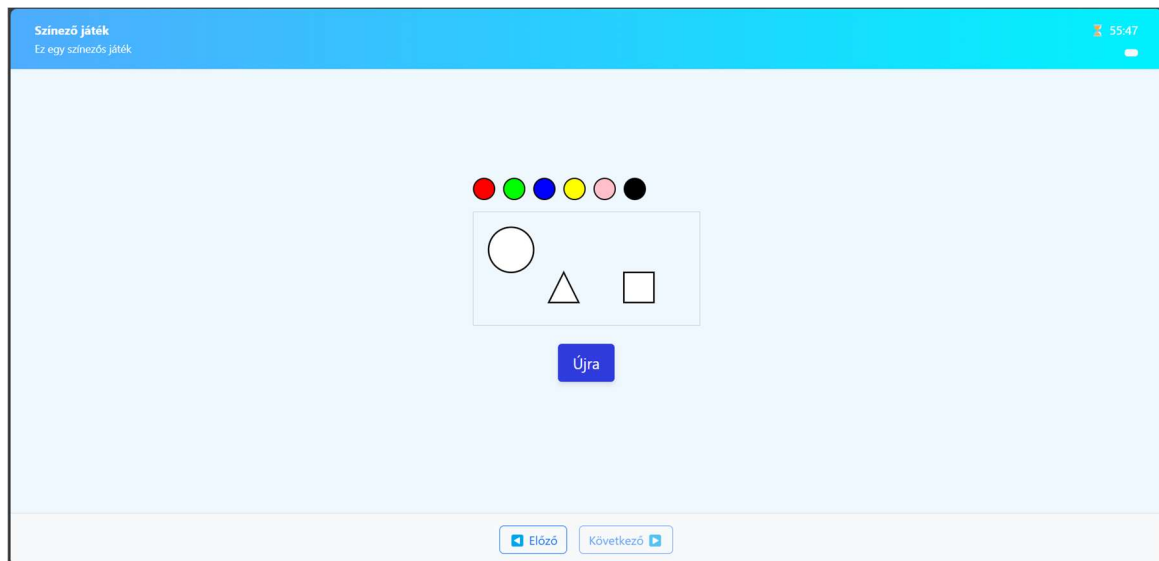
Memória Játék



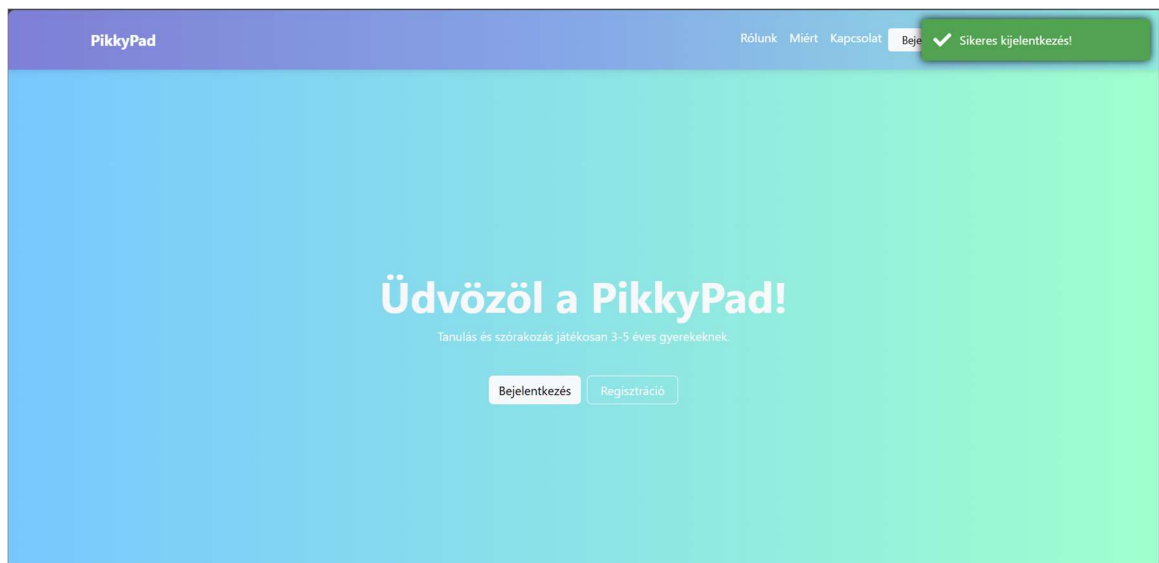
Puzzle



Színező



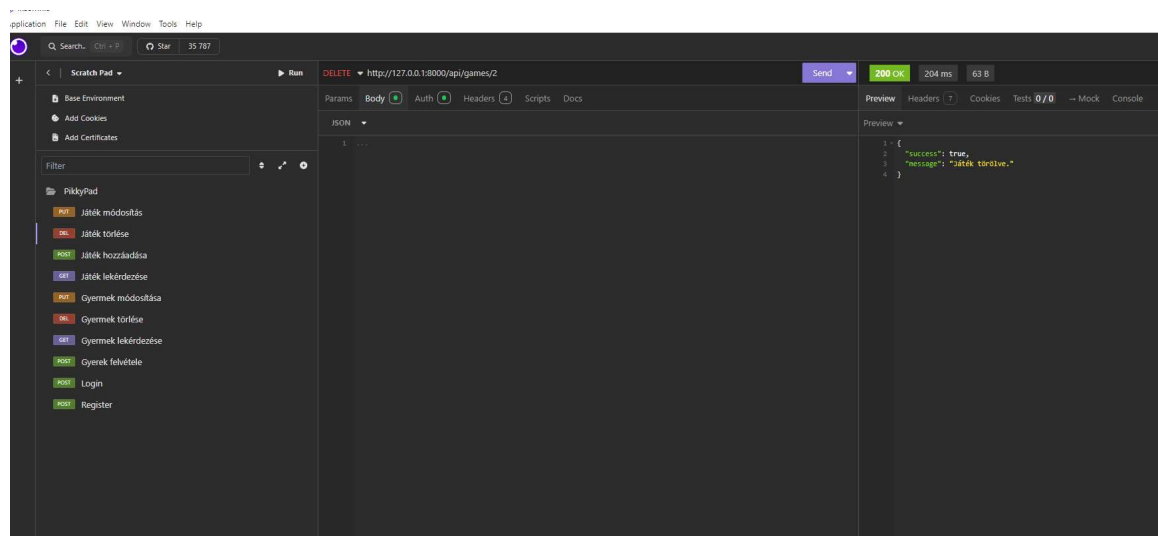
- Sikeres Kijelentkezés



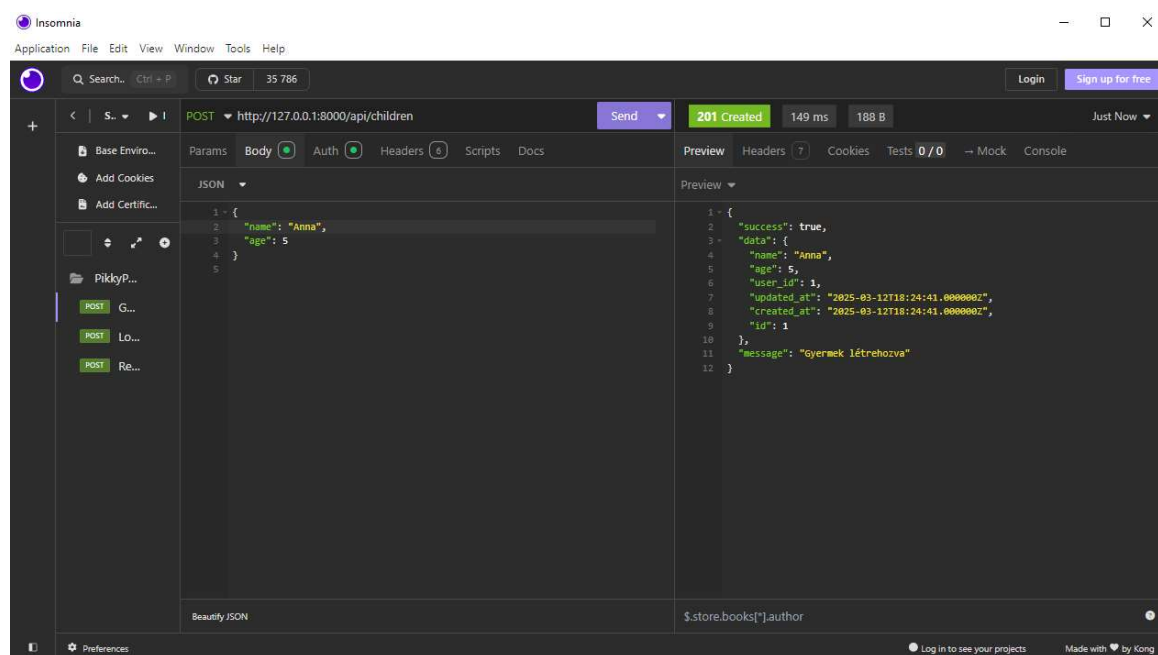
Backend tesztelés

A rendszer egyes REST API végpontját Insomnia segítségével teszteltük. A hitelesítést `Bearer Token` segítségével végeztük, amelyet a bejelentkezés (`/login`) során kaptunk vissza.

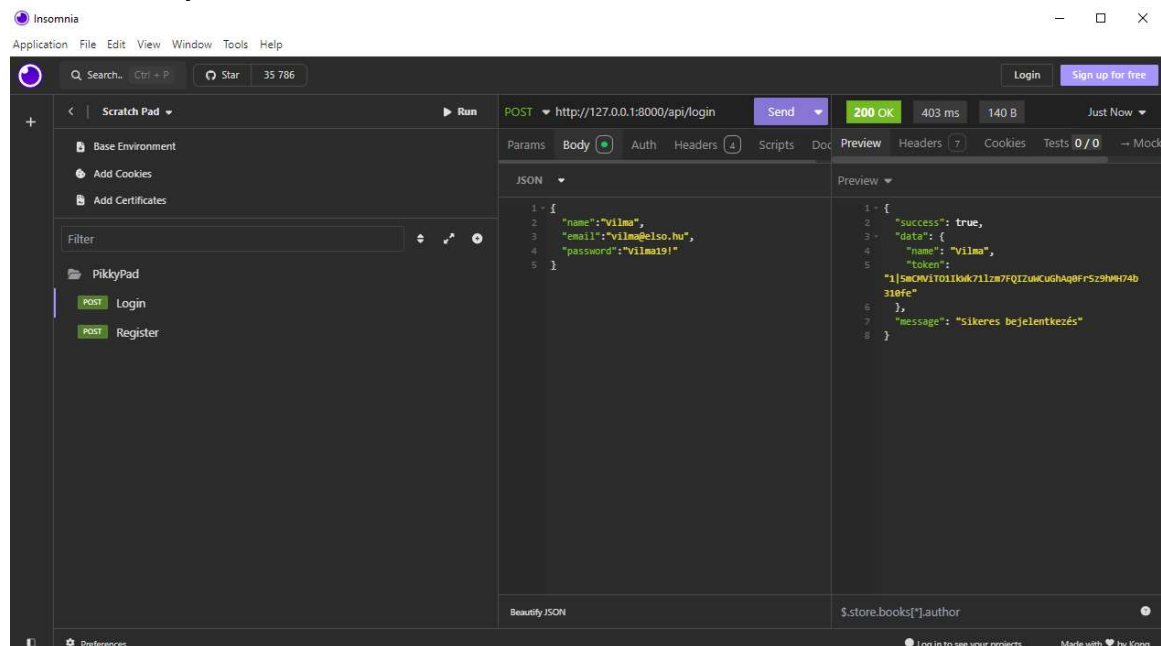
- Játék sikeres törlése



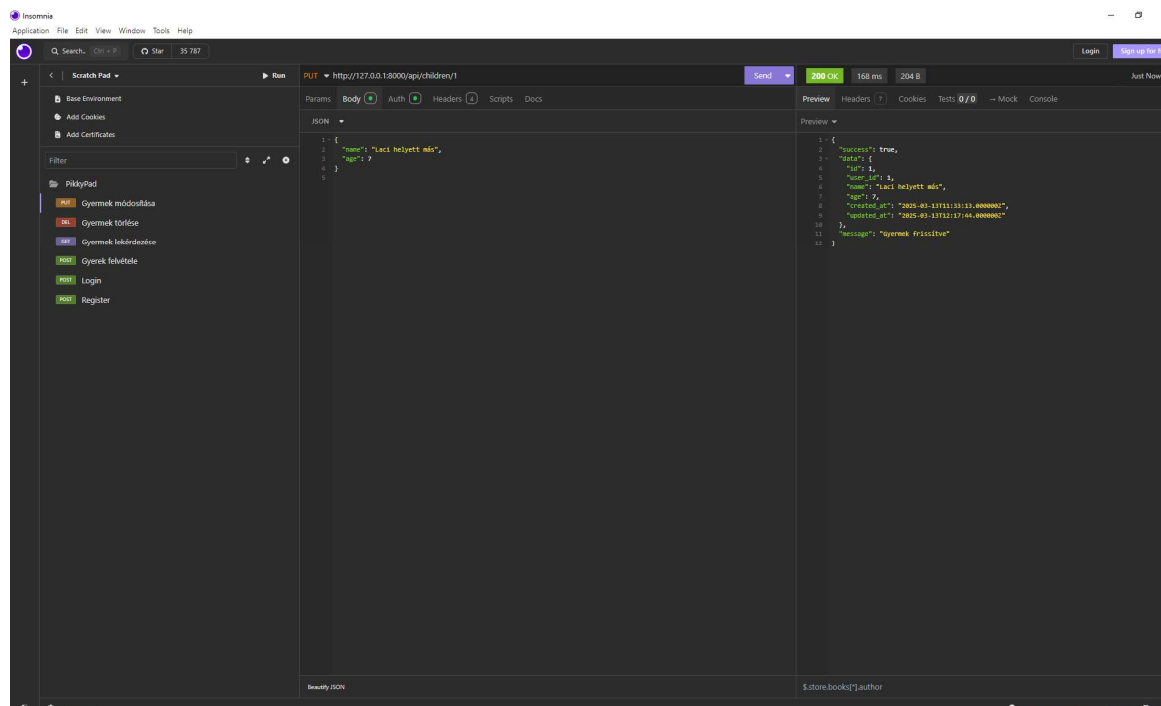
- Gyermek hozzárendelése a profilhoz



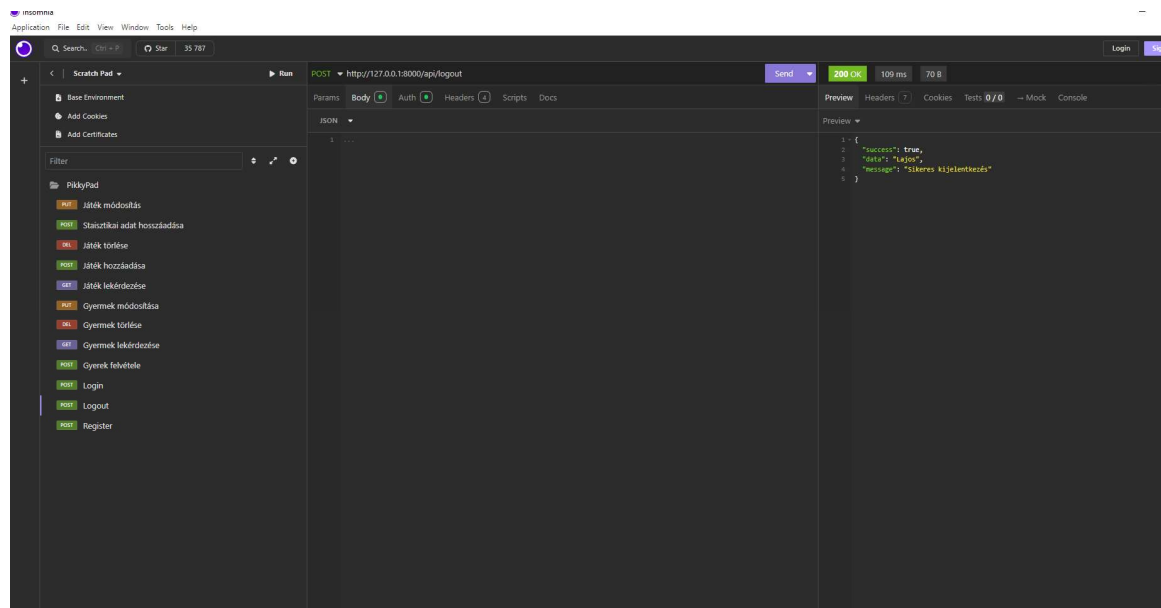
- Sikeres bejelentkezés



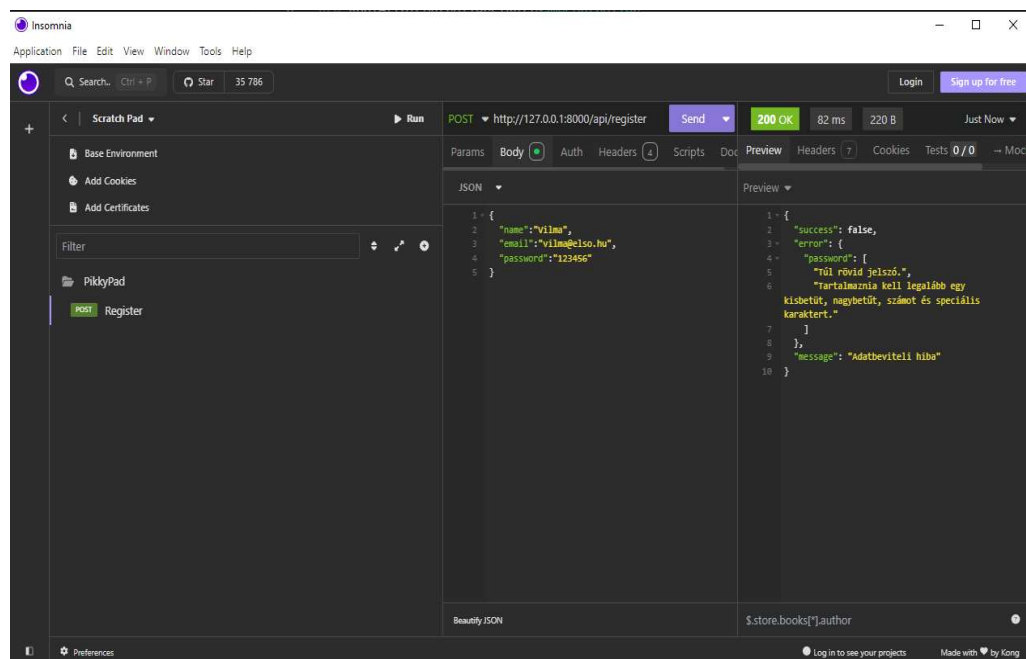
- Gyermek módosítása

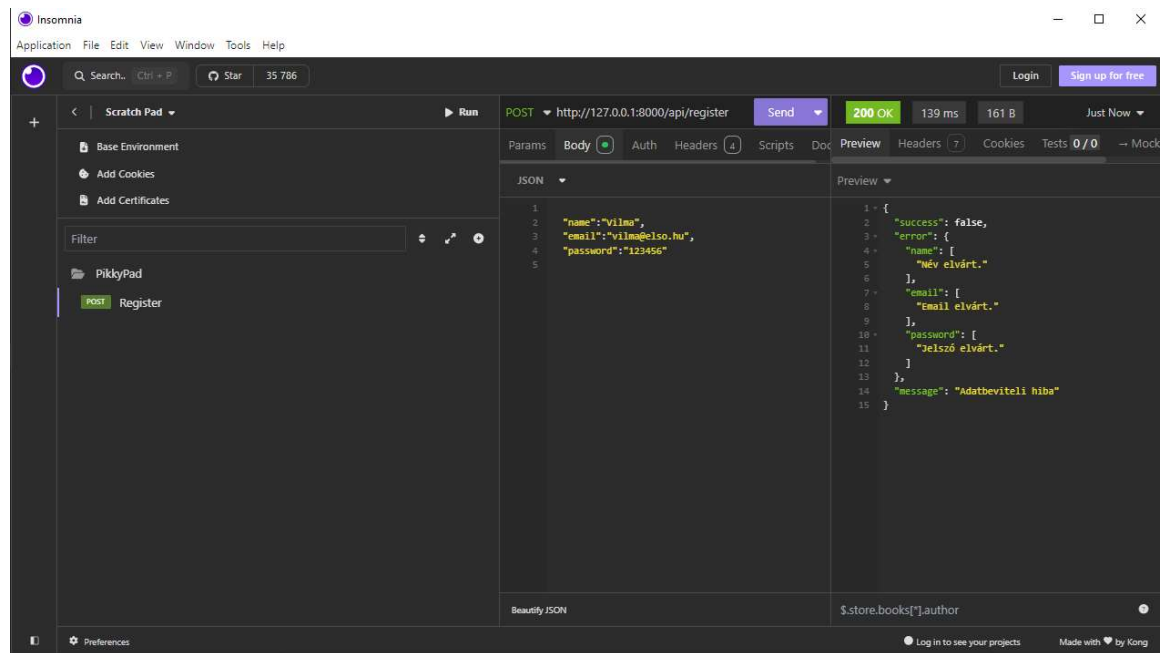


- Sikeres kijelentkezés

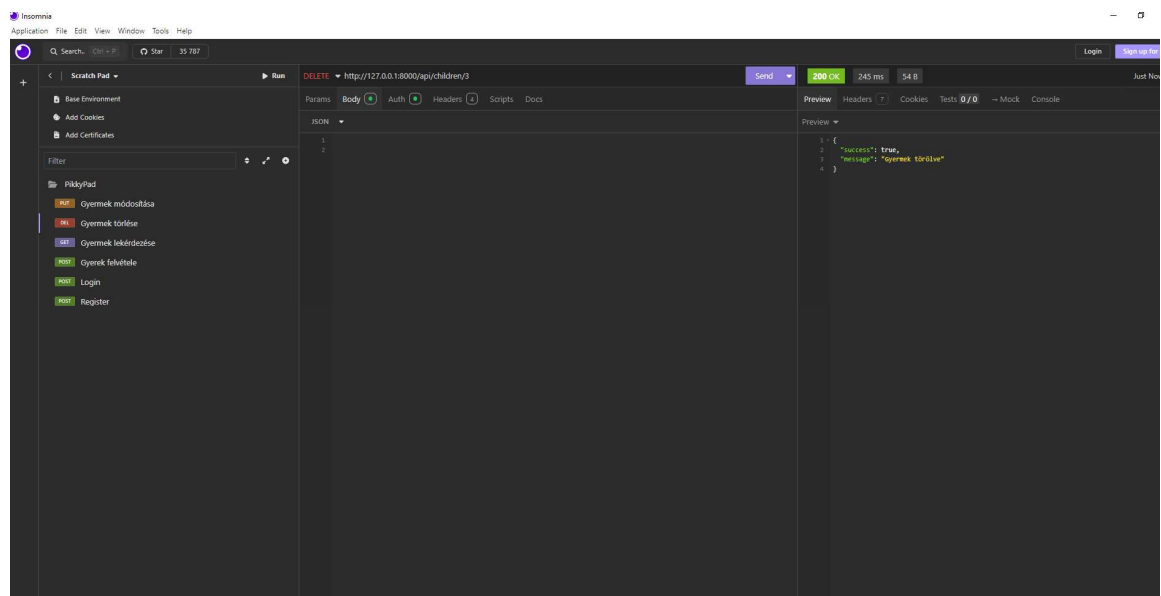


- Regisztrációs hibák: Jelszó hiba.

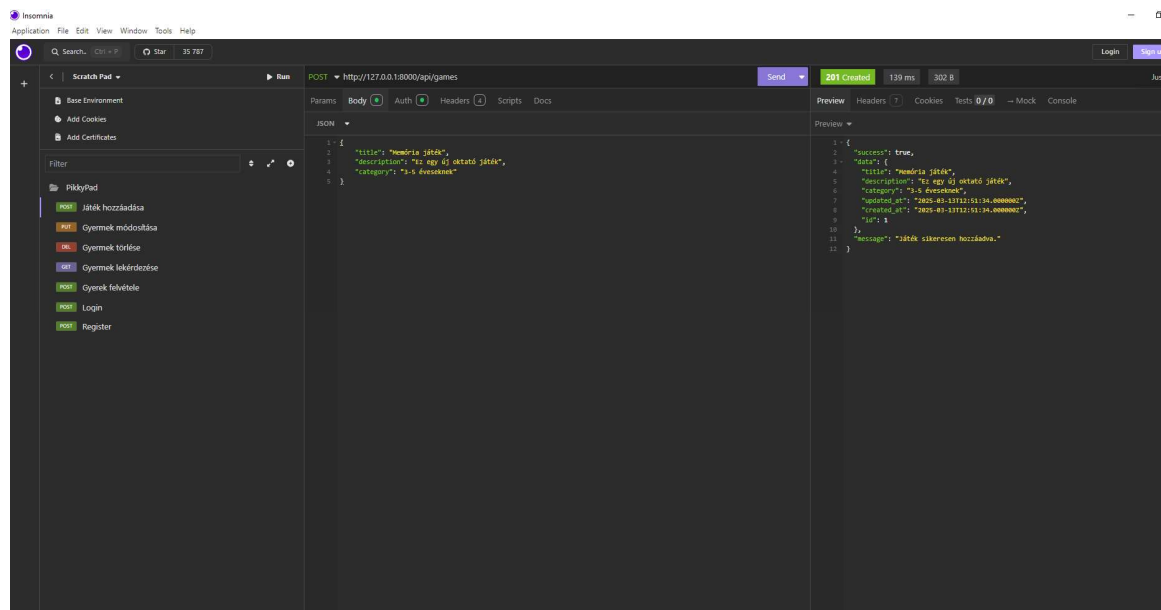




- Gyermek törlése



- Játék sikeres hozzárendelése a gyermekhez.



Megjegyzések

Validációk:

- Gyermek hozzáadásakor a "név" mező kitöltése kötelező.
- Amennyiben üres mezővel próbálkozik a felhasználó, helyesen megjelenik a hibaüzenet.
- Felhasználói élmény (UX) megfigyelések:
- Általános teljesítmény:
- A regisztráció és bejelentkezés gyorsan, hibamentesen történt.
- A gyermekek kezelése (hozzáadás, módosítás, törlés) zökkenőmentesen működött.
- Játékok elindítása során nem tapasztaltunk hibát vagy akadozást.