Jegyzőkönyv

Web technológiák 1.

Féléves feladat

Készítette: **Emődi Máté**

Neptunkód: **GT43W5**

Dátum: **2025. december**

Tartalomjegyzék

Feladat leírása

A beadandóhoz a 2. Opciót választottam, ami egy React alkalmazás elkészítése volt. A feladatkiírás szerint olyan programot kellett írni, ami képes kezelni az adatokat: listázni, újat felvenni, módosítani és törölni őket.

A feladat részletei:

- Téma: Mivel tetszőleges témát lehetett választani, én egy Hallgatói Nyilvántartót készítettem. Itt hallgatók nevét és Neptun kódját lehet kezelni.
- Működés: A programnak tudnia kell a négy alapműveletet: hozzáadás, listázás, szerkesztés és törlés.
- Adattárolás: A feladat szerint nem volt szükség szerveroldali (backend) háttérre, így az adatokat ideiglenesen a böngésző memóriájában tárolja az alkalmazás.

Struktúra és szerkezet

A projektet a create-react-app paranccsal hoztam létre, ami automatikusan legenerálta a React szabványos mappaszerkezetét. A fejlesztés során a forráskód mappájában (src) dolgoztam.

A projekt könyvtárszerkezete:

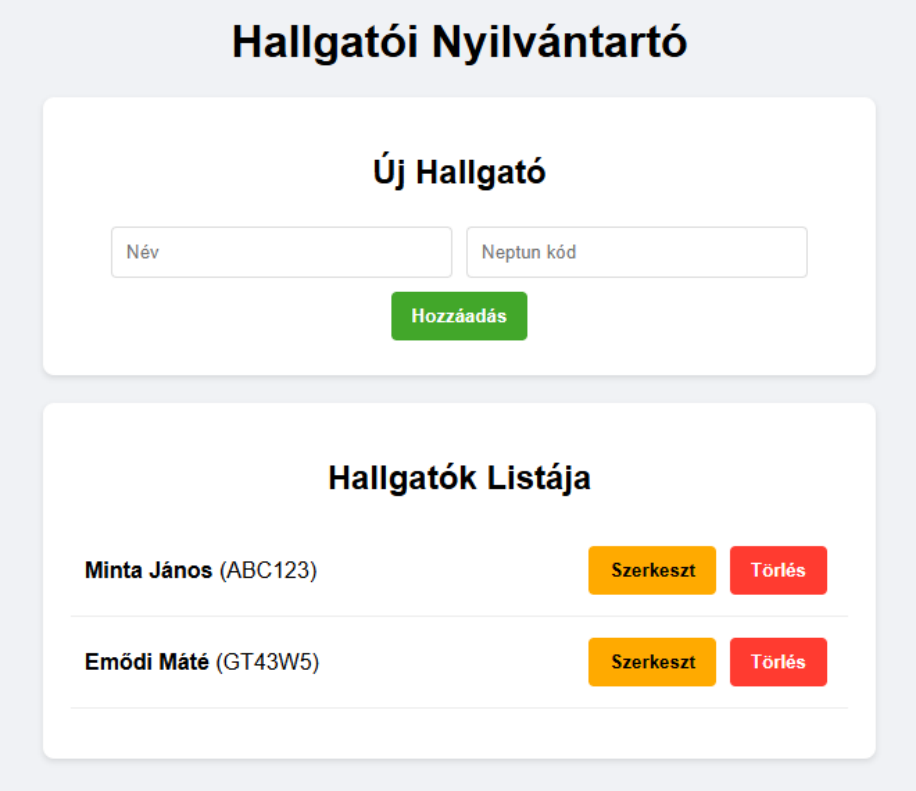
- **node_modules** mappa: Itt találhatóak a projekt működéséhez szükséges előre megírt csomagok és könyvtárak. Ezt a mappa tartalmát a rendszer automatikusan kezeli.
- **public** mappa: Itt vannak a nyilvános fájlok.
    - **index.html:** Ez az egyetlen HTML fájl az egész projektben. A React ebbe szúrja be a tartalmát működés közben.
- **src** mappa: Ez a forráskód mappája, itt történt a tényleges fejlesztés.
    - **App.js:** Ez a fő fájl. Itt található a program teljes logikája és a felület leírása.
    - **App.css:** Ez a stíluslap, ami az alkalmazás kinézetét határozza meg.
    - **index.js:** Összeköti az App.js-t a weboldallal.
- **package.json** fájl: Ez tartalmazza a projekt nevét, verzióját és azt, hogy milyen kiegészítőket használ.

Felhasználói felület

Az alkalmazás felülete egyszerű és átlátható. Egyetlen oldalból áll, amely két fő részre oszlik: felül található az űrlap az adatok kezeléséhez, alul pedig a lista.

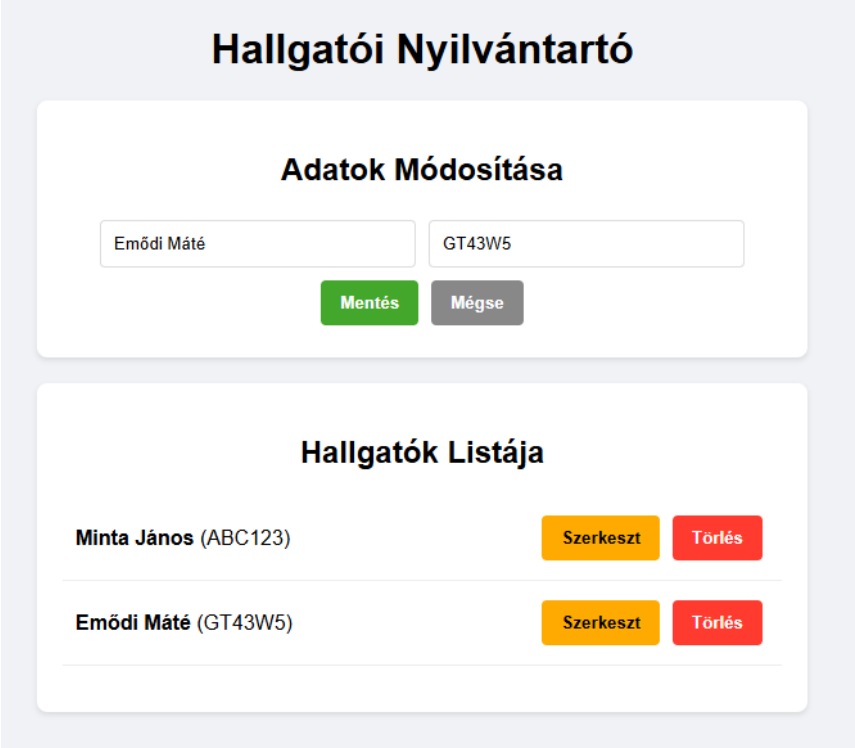
A felület részei:

1. **Adatfelvételi űrlap:** Az oldal tetején két mező található. Ide írhatjuk be az új hallgató adatait. A „Hozzáadás” gombbal mentjük el az adatot, ami azonnal megjelenik a lenti listában.



1. ábra Az alkalmazás alapállapota a listával.

2. **Szerkesztés és Módosítás:** Ha a listában rákattintunk a sárga „Szerkeszt” gombra, az űrlap átvált szerkesztő módba. Ilyenkor a gomb felirata „Mentés”-re változik, és megjelenik egy „Mégse” gomb is. Így javítani tudjuk az elírt adatokat.



2. ábra Szerkesztés közben az űrlap megváltozik.

3. **Törlés:** Minden hallgató neve mellett van egy piros „Törlés” gomb. Erre kattintva az adott sor azonnal eltűnik a listából.

Kód részletek magyarázata

1. **Adatok tárolásának módja:**

```
const [students, setStudents] = useState([
  { id: 1, name: 'Minta János', neptun: 'ABC123' },
  { id: 2, name: 'Emődi Máté', neptun: 'GT43W5' },
]);
const [name, setName] = useState('');
const [neptun, setNeptun] = useState('');
const [isEditing, setIsEditing] = useState(false);
const [currentId, setCurrentId] = useState(null);
```

- A students tömb tartalmazza a hallgatók objektumait ID, név, Neptun kód.
- A name és neptun változók tárolják az űrlapba éppen beírt szöveget.
- Az isEditing és currentId változók vezérlik a szerkesztési módot: ezek jelzik, ha éppen módosítunk egy meglévő adatot, és nem újat viszünk fel.

2. Hozzáadás és módosítás

```
const handleSubmit = (e) => {
    e.preventDefault();
    if (!name || !neptun) return alert("Kérlek tölts ki minden mezőt!");
if (isEditing) {
// Módosítás ha szerkesztés történik
      setStudents(students.map(stu => stu.id === currentId ? { ...stu, name, neptun } : stu));
      setIsEditing(false);
      setCurrentId(null);
    } else {
// Új hozzáadása ha nem történik szerkesztés
      setStudents([...students, { id: Date.now(), name, neptun }]);
    }
    setName('');
    setNeptun('');
  };
```

- Először megakadályozza az oldal újratöltését (e.preventDefault()).
- Ha szerkesztési módban vagyunk (isEditing), akkor a map() függvénnyel végigiterál a listán, és lecseréli a módosított hallgató adatait, a többieket változatlanul hagyja.
- Ha új adatot viszünk fel, akkor a spread operátor (...) segítségével hozzáfűzi az új hallgatót a lista végéhez egy egyedi azonosítóval (Date.now()).

3. Törlés funkció

```
const handleDelete = (id) => setStudents(students.filter(stu => stu.id !== id));
```

A törlés funkció a JavaScript filter() metódusát alkalmazza. A program létrehoz egy új listát, amelybe minden hallgató bekerül, *kivéve* azt az egyet, akinek az azonosítója (id) megegyezik a törlendő elemivel. Ezzel az elem eltűnik a megjelenített listából.