Select.jsx

Áttekintés

A `Select.jsx` egy React komponens, amely lehetővé teszi a felhasználók számára, hogy egy legördülő menüből kiválasszanak egy szobát, és megjelenítse annak részleteit. A komponens az alábbi főbb részekből áll:

1. **Importálások**:
   - React hook-ok: `useState` és `useEffect`.
   - `axios` HTTP kérésekhez.
   - CSS és Bootstrap a stílusokhoz.
   - Egy egyedi komponens: `TablazatSzoba`.
2. **Állapotkezelés**:
   - `rooms`: A szervertől lekért szobák listáját tárolja.
   - `selectedRoom`: Az aktuálisan kiválasztott szoba azonosítóját tárolja.
   - `roomDetails`: A kiválasztott szoba részleteit tárolja.
3. **Adatok lekérése**:
   - A komponens betöltésekor lekéri a szobák listáját a `http://localhost:3001/szobak` címről, és frissíti a `rooms` állapotot.
   - Amikor egy szobát kiválasztanak, lekéri annak részleteit a `http://localhost:3001/szoba/${szazon}` címről, és frissíti a `roomDetails` állapotot.
4. **Eseménykezelés**:
   - `handleSelect`: Kezeli a legördülő menü változás eseményét. Frissíti a `selectedRoom` állapotot és lekéri a kiválasztott szoba részleteit.
5. **Megjelenítés**:
   - Egy legördülő menü, amely szoba opciókkal van feltöltve.
   - A szoba részleteinek megjelenítése egy táblázatban, ha egy szoba ki van választva.
   - A `TablazatSzoba` komponens beillesztése, amelynek átadja a `selectedRoom` prop-ot.

Kód részletezése

Importálások

```
import { useState, useEffect } from "react";
import axios from "axios";
import "../css/App.css";
import 'bootstrap/dist/css/bootstrap.min.css';
import 'tachyons';
import TablazatSzoba from "./TablazatSzoba";
```

- A szükséges modulok és stílusok importálása.

Állapotkezelés

```
const [rooms, setRooms] = useState([]);
const [selectedRoom, setSelectedRoom] = useState("");
const [roomDetails, setRoomDetails] = useState(null);
```

- Három állapot változó definiálása: `rooms`, `selectedRoom`, és `roomDetails`.

Adatok lekérése

```
useEffect(() => {
  axios.get('http://localhost:3001/szobak')
    .then(response => setRooms(response.data))
    .catch(error => console.error("Hiba történt a szobák lekérésekor:",
error));
}, []);
```

- A `useEffect` hook segítségével a komponens betöltésekor lekéri a szobák listáját.

Eseménykezelés

```
const handleSelect = async (event) => {
  const szazon = event.target.value;
  setSelectedRoom(szazon);

  if (!szazon) {
    setRoomDetails(null);
    return;
  }
  try {
    const response = await
axios.get(`http://localhost:3001/szoba/${szazon}`);
    setRoomDetails(response.data);
  } catch (error) {
    console.error("Hiba történt a szoba adatainak lekérésekor:", error);
    setRoomDetails(null);
  }
};
```

- A `handleSelect` függvény kezeli a legördülő menü változását, és lekéri a kiválasztott szoba részleteit.

Megjelenítés

```
return (
  <div>
    <select onChange={handleSelect} value={selectedRoom}>
```

```
      <option value="">Válassz egy szobát</option>
      {rooms.map(room => (
        <option key={room.szazon} value={room.szazon}>{room.sznev}</option>
      ))}
    </select>
    {roomDetails && (
      <div>
        <h3>Szoba adatai</h3>
        <table className="table table-striped">
          <thead>
            <tr>
              <th scope="col">Szoba neve</th>
              <th scope="col">Ágyak száma</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>{roomDetails.sznev}</td>
              <td>{roomDetails.agy}</td>
            </tr>
          </tbody>
        </table>
        <TablazatSzoba selectedRoom={selectedRoom} />
      </div>
    )}
  </div>
);
```

- A legördülő menü és a szoba részleteinek megjelenítése.

Értékelés

- **Funkcionalitás**: A komponens jól kezeli a szobák listájának és részleteinek lekérését és megjelenítését. Az eseménykezelés és az állapotkezelés megfelelően van implementálva.
- **Kódminőség**: A kód jól strukturált és olvasható. Az állapotkezelés és az aszinkron műveletek kezelése megfelelő.
- **Felhasználói élmény**: A felhasználói felület egyszerű és könnyen használható. A Bootstrap és a saját CSS használata biztosítja a megfelelő megjelenést.

Tablazat.jsx

Áttekintés

A `Tablazat.jsx` egy React komponens, amely egy táblázatot jelenít meg a szobák listájával. A komponens az alábbi főbb részekből áll:

1. **Importálások**:
   - React hook-ok: `useState` és `useEffect`.
   - `axios` HTTP kérésekhez.
   - CSS és Bootstrap a stílusokhoz.
2. **Állapotkezelés**:
   - `data`: A szervertől lekért szobák listáját tárolja.
3. **Adatok lekérése**:
   - A komponens betöltésekor lekéri a szobák listáját a `http://localhost:3001/szobak` címről, és frissíti a `data` állapotot.
4. **Megjelenítés**:
   - Egy táblázat, amely a szobák nevét és az ágyak számát jeleníti meg.

Kód részletezése

Importálások

```
import { useState, useEffect } from "react";
import axios from "axios";
import '../css/App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'tachyons';
```

- A szükséges modulok és stílusok importálása.

Állapotkezelés

```
const [data, setData] = useState([]);
```

- Egy állapot változó definiálása: `data`, amely a szobák listáját tárolja.

Adatok lekérése

```
useEffect(() => {
  axios.get("http://localhost:3001/szobak")
    .then(response => setData(response.data))
    .catch(error => console.error("Hiba:", error));
}, []);
```

- A `useEffect` hook segítségével a komponens betöltésekor lekéri a szobák listáját.

Megjelenítés

```
return (
  <div>
    <h2 className="text-center my-4">Szoba lista</h2>
    <table className="table table-striped">
      <thead>
        <tr>
          <th>Szoba neve</th>
          <th>Ágyak száma</th>
        </tr>
      </thead>
      <tbody>
        {data.map((row) => (
          <tr key={row.szazon}>
            <td>{row.sznev}</td>
            <td>{row.agy}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
);
```

- A szobák listájának megjelenítése egy táblázatban, amely tartalmazza a szoba nevét és az ágyak számát.

Értékelés

- **Funkcionalitás**: A komponens jól kezeli a szobák listájának lekérését és megjelenítését. Az eseménykezelés és az állapotkezelés megfelelően van implementálva.
- **Kódminőség**: A kód jól strukturált és olvasható. Az állapotkezelés és az aszinkron műveletek kezelése megfelelő.
- **Felhasználói élmény**: A felhasználói felület egyszerű és könnyen használható. A Bootstrap és a saját CSS használata biztosítja a megfelelő megjelenést.

Javaslatok

- **Hiba kezelés**: Érdemes lenne a felhasználói felületre is kiírni a hibákat, nem csak a konzolra, hogy a felhasználók is értesüljenek a problémákról.
- **Loading állapot**: Érdemes lenne egy "Loading" állapotot is bevezetni, hogy a felhasználók lássák, amikor az adatok betöltődnek.
- **Reszponzivitás**: A táblázat reszponzív megjelenítése érdekében érdemes lehet további CSS osztályokat hozzáadni.

TablazatSzoba.jsx

Áttekintés

A `TablazatSzoba.jsx` egy React komponens, amely egy táblázatot jelenít meg a kiválasztott szoba foglaltsági adataival. A komponens az alábbi főbb részekből áll:

1. **Importálások**:
   - React hook-ok: `useState` és `useEffect`.
   - `axios` HTTP kérésekhez.
   - CSS és Bootstrap a stílusokhoz.
2. **Állapotkezelés**:
   - `data`: A szervertől lekért foglaltsági adatokat tárolja.
   - `loading`: A betöltési állapotot jelzi.
3. **Adatok lekérése**:
   - A komponens betöltésekor és a `selectedRoom` változásakor lekéri a foglaltsági adatokat a `http://localhost:3001/foglaltsag/${selectedRoom}` címről, és frissíti a `data` állapotot.
4. **Megjelenítés**:
   - Egy táblázat, amely a foglaltsági adatokat jeleníti meg, vagy megfelelő üzeneteket jelenít meg, ha nincs adat vagy éppen töltődik.

Kód részletezése

Importálások

```
import { useState, useEffect } from "react";
import axios from "axios";
import '../css/App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'tachyons';
```

- A szükséges modulok és stílusok importálása.

Állapotkezelés

```
const [data, setData] = useState(null); // Kezdetben null, hogy az első
betöltésnél is látszódjon az üzenet
const [loading, setLoading] = useState(false);
```

- Két állapot változó definiálása: `data` és `loading`.

Adatok lekérése

```
useEffect(() => {
  if (!selectedRoom) {
```

```
    setData(null); // Ha nincs kiválasztott szoba, biztosan megjelenik az
alapértelmezett üzenet
    return;
  }
  setLoading(true); // Betöltési állapot aktiválása

  axios.get(`http://localhost:3001/foglaltsag/${selectedRoom}`)
    .then(response => {
      setData(response.data.length > 0 ? response.data : []); // Ha nincs
adat, akkor üres tömböt állítunk be
    })
    .catch(error => console.error("Hiba:", error))
    .finally(() => setLoading(false)); // Betöltés vége
}, [selectedRoom]); // A useEffect minden selectedRoom változásnál lefut
```

- A `useEffect` hook segítségével a komponens betöltésekor és a `selectedRoom` változásakor lekéri a foglaltsági adatokat.

Megjelenítés

```
return (
  <div>
    <h2 className="text-center my-4">Szoba foglaltsága</h2>
    {/* Oldal betöltésekor alapértelmezett üzenet jelenjen meg */}
    {data === null ? (
      <p className="text-center text-muted">Válassz egy szobát a foglaltság
megtekintéséhez.</p>
    ) : loading ? (
      <p className="text-center">Adatok betöltése...</p>
    ) : data.length === 0 ? (
      <p className="text-center text-warning">Nincs foglalás erre a
szobára.</p>
    ) : (
      <table className="table table-striped">
        <thead>
          <tr>
            <th>A vendég neve</th>
            <th>Szoba neve</th>
            <th className="text-center">Érkezés dátuma</th>
            <th className="text-center">Távozás dátuma</th>
          </tr>
        </thead>
        <tbody>
          {data.map((row) => (
            <tr key={row.szazon}>
              <td>{row.vnev}</td>
```

```
                <td>{row.sznev}</td>
                <td className="text-center">{row.erk}</td>
                <td className="text-center">{row.tav}</td>
              </tr>
            ))}
          </tbody>
        </table>
      )}
    </div>
);
```

- A foglaltsági adatok megjelenítése egy táblázatban, vagy megfelelő üzenetek megjelenítése, ha nincs adat vagy éppen töltődik.

Értékelés

- **Funkcionalitás**: A komponens jól kezeli a foglaltsági adatok lekérését és megjelenítését. Az eseménykezelés és az állapotkezelés megfelelően van implementálva.
- **Kódminőség**: A kód jól strukturált és olvasható. Az állapotkezelés és az aszinkron műveletek kezelése megfelelő.
- **Felhasználói élmény**: A felhasználói felület egyszerű és könnyen használható. A Bootstrap és a saját CSS használata biztosítja a megfelelő megjelenést.

Javaslatok

- **Hiba kezelés**: Érdemes lenne a felhasználói felületre is kiírni a hibákat, nem csak a konzolra, hogy a felhasználók is értesüljenek a problémákról.
- **Loading állapot**: A betöltési állapot megfelelően van kezelve, de érdemes lehet egy vizuális elemet (pl. spinner) hozzáadni a jobb felhasználói élmény érdekében.
- **Reszponzivitás**: A táblázat reszponzív megjelenítése érdekében érdemes lehet további CSS osztályokat hozzáadni.

Foglaltsag.jsx

Áttekintés

A `Foglaltsag.jsx` egy React komponens, amely egy oldal elrendezést biztosít a szobák foglaltsági adatainak megjelenítéséhez. A komponens az alábbi főbb részekből áll:

1. **Importálások**:
   - A `Select` komponens importálása.
2. **Megjelenítés**:
   - Egy konténer, amely két oszlopra van osztva.
   - Az első oszlopban a falusi szálláshely típusainak listája és egy kép található.
   - A második oszlopban a `Select` komponens és egy üzenet található, amely arra kéri a felhasználót, hogy válasszon egy szobát a foglaltsági adatok megjelenítéséhez.

Kód részletezése

Importálások

```
import Select from "./Select.jsx";
```

- A `Select` komponens importálása, amely a szobák kiválasztásáért és azok részleteinek megjelenítéséért felelős.

Megjelenítés

```
function Foglaltsag() {
  return (
    <>
      <div className="container-fluid">
        <div className="row">
          <div className="col-md-6">
            <h3>Falusi szálláshely fajtái</h3>
            <ul>
              <li>Vendégszoba: a vendégek rendelkezésére bocsátható önálló lakóegység, amely egy lakóhelyiségből, és a minősítéstől függően a hozzátartozó mellékhelyiségekből áll.</li>
              <li>Lakrész: önálló épület kettő, illetve több szobából álló lehatárolt része a minősítéstől függően hozzátartozó mellékhelyiségekkel együtt</li>
              <li>Vendégház: önálló épület, több szobával, mellékhelyiségekkel és főzési lehetőséggel rendelkező lakó-, illetve üdülőegység, családok vagy kisebb csoportok elszállásolására.</li>
```

```
              <li>Sátorozóhely: csak valamelyik falusi szálláshely típus
mellett, mintegy azt kiegészítve üzemeltethető az előírt feltételek megléte
esetén. Pl.: falusi vendégház sátorozóhellyel.</li>
            </ul>
            <p><img src="./ketagyas.jpg" width="600px" height="400px"
className="img-thumbnail" alt="turizmus" title="turizmus" /></p>
          </div>
          <div className="col-md-6">
            <h3>A választott fogadó:</h3>
            <Select/>
            <h1>Kérem válasszon egy szobát a foglaltsági adatok
megjelenítéséhez.</h1>
          </div>
        </div>
      </div>
    </>
  )
}
export default Foglaltsag;
```

- A komponens egy konténerben két oszlopot jelenít meg.
- Az első oszlopban egy lista található a falusi szálláshely típusairól, valamint egy kép.
- A második oszlopban a `Select` komponens található, amely lehetővé teszi a felhasználók számára, hogy kiválasszanak egy szobát, valamint egy üzenet, amely arra kéri a felhasználót, hogy válasszon egy szobát a foglaltsági adatok megjelenítéséhez.

Értékelés

- **Funkcionalitás**: A komponens jól strukturált és megfelelően integrálja a `Select` komponenst. Az elrendezés logikus és könnyen követhető.
- **Kódminőség**: A kód jól strukturált és olvasható. Az elrendezés és a stílusok megfelelően vannak alkalmazva.
- **Felhasználói élmény**: A felhasználói felület egyszerű és könnyen használható. Az információk jól vannak elrendezve, és a felhasználók könnyen megtalálják, amit keresnek.

Javaslatok

- **Reszponzivitás**: Érdemes lenne további CSS osztályokat hozzáadni a reszponzív megjelenítés érdekében, hogy a felület különböző képernyőméreteken is jól nézzen ki.
- **Hiba kezelés**: Érdemes lenne a `Select` komponensben megjeleníteni a hibákat a felhasználói felületen, hogy a felhasználók is értesüljenek a problémákról.
- **Interaktivitás**: Érdemes lenne további interaktív elemeket hozzáadni, például egy keresőmezőt a szobák listájához, hogy a felhasználók könnyebben megtalálják a kívánt szobát.