TrackNest Dokumentáció

Tartalomjegyzék

---

1. Áttekintés

A rendszer három fő technológiai pillérre épül: Node.js alapú backend (Express keretrendszerrel), React alapú frontend, valamint MongoDB adatbázis. A fejlesztés során kiemelt szempont volt a modularitás, a bővíthetőség, valamint a felhasználói és adminisztrátori funkciók szétválasztása.

A backend felelős az üzleti logika megvalósításáért, az adatok biztonságos kezeléséért, a hitelesítésért (JWT token alapú), valamint a REST API végpontok kiszolgálásáért. A szerver oldali kód Mongoose ORM-et használ a MongoDB-vel való kommunikációhoz, így biztosítva az adatmodellek validációját és az adatok integritását.

A frontend egy reszponzív, felhasználóbarát React alkalmazás, amely lehetővé teszi a felhasználók számára a bejelentkezést, termékek hozzáadását, listázását, valamint – adminisztrátori jogosultsággal – azok törlését is. Az alkalmazás állapotkezelése egyszerű, jól átlátható, a komponensek között világos felelősségi körök vannak.

Az adatbázis (MongoDB) NoSQL alapú, így rugalmasan bővíthető, és jól illeszkedik a modern webes alkalmazások igényeihez. A rendszer támogatja a többfelhasználós működést, a jogosultságok kezelését, valamint a tipikus hibák és edge case-ek kezelését is.

A projekt célja, hogy egy könnyen telepíthető, fejleszthető, átlátható és biztonságos nyilvántartó rendszert biztosítson, amely alkalmas lehet akár oktatási, akár kisebb vállalati környezetben történő használatra is. A dokumentáció részletesen bemutatja a rendszer felépítését, működését, valamint a fejlesztéshez és üzemeltetéshez szükséges lépéseket.

2. Backend (Node.js, Express)

Szerver felépítése

A backend a `server/` mappában található. Az `index.js` a fő belépési pont, amely elindítja az Express szervert, beállítja a middleware-ket, és csatlakozik a MongoDB-hez. A szerver támogatja a CORS-t, JSON body-k feldolgozását, és naplózást is.

Főbb fájlok:

- `server/index.js` : szerver indítása, útvonalak regisztrálása, adatbázis kapcsolódás
- `server/middleware/auth.js` : hitelesítési middleware, JWT token ellenőrzés
- `server/models/User.js` , `server/models/Product.js` : Mongoose modellek
- `server/createUser.js` : adminisztrátori felhasználó létrehozása

Middleware-k

A middleware-k, mint például az `auth.js` , a kérések feldolgozása előtt futnak le. Az `auth.js` például ellenőrzi a felhasználó jogosultságát JWT token alapján. Ha a token érvénytelen vagy hiányzik, a szerver 401-es hibát ad vissza.

Modellek

A modellek a MongoDB-ben tárolt adatok szerkezetét írják le. A `User.js` és `Product.js` Mongoose sémákat tartalmaznak, amelyek meghatározzák a felhasználók és termékek mezőit, validációkat, indexeket.

Példa (User modell):

```
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['admin', 'user'], default: 'user' }
});
module.exports = mongoose.model('User', UserSchema);
```

Példa (Product modell):

```
const mongoose = require('mongoose');
const ProductSchema = new mongoose.Schema({
  name: { type: String, required: true },
  description: String,
  price: { type: Number, required: true },
  quantity: { type: Number, default: 0 }
});
module.exports = mongoose.model('Product', ProductSchema);
```

API végpontok részletesen

- **POST** `/api/login` – bejelentkezés, visszaad JWT tokent
- **POST** `/api/products` – új termék hozzáadása (auth szükséges)
- **GET** `/api/products` – termékek listázása
- **DELETE** `/api/products/:id` – termék törlése (admin jogosultság)

Példa válaszok:

```
// Sikeres bejelentkezés
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6..."
}
// Hibás bejelentkezés
{
  "error": "Hibás felhasználónév vagy jelszó"
}
```

Jogosultságkezelés és hibakezelés

A middleware ellenőrzi a JWT tokent, és a felhasználó szerepét. Az adminisztrátori műveletekhez (pl. törlés) csak admin jogosultság szükséges. Hibák esetén a szerver megfelelő HTTP státuszkódot és hibaüzenetet ad vissza.

3. Frontend (React)

A frontend a `src/` mappában található. A fő komponens az `App.tsx`, amely a különböző oldalakat és komponenseket kezeli. A React Router segítségével több oldal (bejelentkezés, terméklista, termék hozzáadás) érhető el.

Főbb komponensek

- `components/login/Login.tsx` – bejelentkezési űrlap, hibakezelés, token tárolás
- `components/productform/AddProductForm.tsx` – termék hozzáadása, validáció
- `components/productlist/ProductList.tsx` – termékek listázása, törlés gomb (csak adminnak)

Állapotkezelés

A React komponensek belső állapotot (`useState`) és effektusokat (`useEffect`) használnak. A bejelentkezés után a JWT token a localStorage-be kerül, és minden védett kérésnél a fejléchez csatolódik.

Kommunikáció a backenddel

Az `api.ts` fájl tartalmazza az axios/fetch alapú HTTP hívásokat, például:

```
import axios from 'axios';
export const login = async (username, password) => {
  return await axios.post('/api/login', { username, password });
};
export const getProducts = async (token) => {
  return await axios.get('/api/products', { headers: { Authorization: `Bearer ${token}` } });
};
```

Példák és tipikus hibák

- Hibás bejelentkezés esetén hibaüzenet jelenik meg.
- Token hiányában a védett oldalak nem elérhetők.
- Admin funkciók csak admin felhasználónak jelennek meg.

4. MongoDB adatbázis

A MongoDB a backenddel Mongoose-on keresztül kapcsolódik. Az adatbázisban két fő gyűjtemény található: `users` és `products`.

Adatmodell

- **User**: felhasználónév, jelszó (hash-elve), jogosultságok
- **Product**: név, leírás, ár, mennyiség

Kapcsolódás

A szerver indításakor a `mongoose.connect()` hívással kapcsolódik a MongoDB-hez.

Példa kapcsolódásra:

```
mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true });
```

Indexek, validáció

A modellekben megadhatók egyedi indexek (pl. felhasználónév), kötelező mezők, értéktartományok. A hibás adatokat a backend elutasítja.

5. Folyamatok és működés

Felhasználói folyamatok

1. A felhasználó megnyitja a webalkalmazást.
2. Bejelentkezik a frontend bejelentkezési űrlapján keresztül.
3. A frontend elküldi a bejelentkezési adatokat a backendnek.
4. A backend ellenőrzi az adatokat, és sikeres hitelesítés esetén JWT tokent ad vissza.
5. A felhasználó a tokennel további védett műveleteket végezhet (pl. termék hozzáadása, listázása).
6. Az adatok a MongoDB-ben tárolódnak.

Adminisztrációs folyamatok

- Új admin létrehozása a `createUser.js` segítségével
- Termékek törlése, módosítása

6. Telepítés és futtatás

Fejlesztői környezet

1. Lépj be a `server/` mappába:

   ```
   cd server
   ```

2. Telepítsd a függőségeket:

   ```
   npm install
   ```

3. Indítsd el a szervert:

```
npm start
```

Frontend

1. Lépj vissza a fő mappába:

   ```
   cd ..
   ```

2. Telepítsd a frontend függőségeit:

   ```
   npm install
   ```

3. Indítsd el a React alkalmazást:

   ```
   npm start
   ```

Éles környezet

- A környezeti változókat `.env` fájlban kell megadni (pl. MONGO_URI, JWT_SECRET)
- A szervert érdemes process managerrel (pl. pm2) futtatni

7. Mellékletek: REST API válaszok, példák

Sikeres termék hozzáadás

```
{
  "message": "Termék sikeresen hozzáadva",
  "product": {
    "_id": "...",
    "name": "Teszt termék",
    "price": 1000,
    "quantity": 5
  }
}
```

Hibás kérés

```
{
  "error": "Hiányzó kötelező mező: név"
}
```
