Dokumentácia k webovej aplikácii
Fitness App

Úvod

Fitness App je webová aplikácia navrhnutá na správu jedál, vytváranie tréningových plánov a monitorovanie cvičení. Cieľom aplikácie je poskytnúť používateľom intuitívne nástroje na sledovanie ich stravovania, cvičenia a pokroku. Naša aplikácia ponúka rôzne funkcie, ktoré umožňujú personalizáciu tréningových plánov, správu jedál a detailný prehľad o aktivitách používateľov. Aplikácia sa zameriava na fitness nadšencov, ktorí chcú mať komplexný prehľad o svojich aktivitách a strave.

V porovnaní s konkurenčnými aplikáciami, ako sú MyFitnessPal alebo Strava, ponúka naša aplikácia rozšírené možnosti personalizácie a jednoduchší spôsob integrácie jedál a cvičení do denného režimu.

Funkcie aplikácie

1. Hlavná stránka - Prehľad

- Zobrazenie pridaných jedál spolu s informáciami o kalorickej hodnote.
- Panel pre zobrazenie cvičení:
  - Typ (napr. Outdoor alebo Indoor).
  - Intenzita cvičenia (napr. **LOW**, **MEDIUM**, **HIGH**).
  - Časť tela, ktorú cvičenie posilňuje.
- Prehľad o pokroku: grafy s vývojom váhy, počtu spálených kalórií a vykonaných cvičení.

2. Správa jedál

- **Pridať jedlo:** Možnosť pridávať nové jedlá spolu s ich kalorickými hodnotami a makroživinami.
- **Úprava jedál:** Upravovanie detailov existujúcich jedál (napr. zmena kalorických hodnôt).
- **Odstraňovanie:** Odstránenie nepotrebných záznamov jedál z histórie.
- **Sledovanie kalórií:** Automatické výpočty kalórií a makroživín v závislosti od zadaného množstva jedla.

3. Tvorba tréningového plánu

- Personalizácia tréningov na základe intenzity a preferovaných častí tela.
- Možnosť ukladať a spravovať viaceré tréningové plány.
- Integrácia s fitness zariadeniami (napr. Fitbit, Apple Watch) na sledovanie výkonu v reálnom čase.

4. Zoznam cvičení

- Prehľad dostupných cvičení spolu s ich detailmi:
  - Typ cvičenia (napr. Beh, Posilňovanie).
  - Cieľová oblasť tela (napr. Nohy, Ruky).
  - Doplnkové informácie, ako napr. požiadavky na vybavenie.

5. Profil užívateľa

- Možnosť upraviť osobné údaje užívateľa.
- Prehľad histórie aktivít, výkonu a pokroku v cvičeniach.
- Nastavenie cieľov (napr. dosiahnutie určitej váhy, počet opakovaní).

API Endpoints

1. Autentifikácia

```
router.post("/login", validate(LoginUserSchema), login);
router.post("/register", validate(RegisterUserSchema), register);
router.get("/refreshSession", refreshSession);
router.post("/logout", logout);
router.get("/me", verifyJWT, authenticatedUser);
```

2. Správa jedál (Food)

```
router.get("/food", verifyJWT, getUserFoods);
router.post("/food", verifyJWT, postUserFood);
router.delete("/food/:id", verifyJWT, deleteFoodLog);
router.post("/food/similar", verifyJWT, getSimilarFoods);
router.post("/spoonacular/foodSuggestion", verifyJWT, getSuggestionReq
router.post("/spoonacular/foodInfo", verifyJWT, getFoodInfoRequest);
```

3. Správa cvičení (Exercise)

```
router.get("/exercises", verifyJWT, getUserExercises);
router.get("/exercises/:id", verifyJWT, getExercise);
router.put("/exercises/:id", verifyJWT, validate(UpdateExerciseSchema)
router.post("/exercises", verifyJWT, validate(CreateExerciseSchema), p
router.post("/exercises/owned", verifyJWT, validate(PaginationSchema),
router.post("/exercises/all", verifyJWT, validate(PaginationSchema), g
router.delete("/exercises/:id", verifyJWT, deleteExercise);
```

4. Tréningové plány (Plan)

```
router.post("/training-plans", verifyJWT, createPlan);
router.get("/training-plans", verifyJWT, getPlans);
router.get("/training-plans/:id", verifyJWT, getPlanById);
router.put("/training-plans/:id", verifyJWT, updatePlan);
router.delete("/training-plans/:id", verifyJWT, deletePlan);
```

5. Správa používateľských údajov (User)

```
router.post("/email/sendResetPassword", validate(SendResetPasswordEmai
router.post("/passwordReset", validate(ResetPasswordSchema), resetPassv
router.post("/passwordChange", verifyJWT, validate(ChangePasswordSchem
```

Technické detaily

Frontend

- **Technológie:** HTML, CSS, JavaScript (React alebo Vue).
- **Dizajn:** Minimalistické, tmavé rozhranie s dôrazom na čitateľnosť a prehľadnosť.
- **Komponenty:** Reaktívne komponenty na správu jedál, tréningov a profilov.

Backend

- **Technológia:** Node.js s frameworkom Express.js.
- **Middleware:** `validate`, `verifyJWT`.
- **Štruktúra:** Modularizované s jasne oddelenými routami a kontrolérmi.

Databáza

Aplikácia **Fitness App** využíva relačnú databázu PostgreSQL na správu údajov o používateľoch, jedlách, aktivitách, cvičeniach a tréningových plánoch. K dátovej vrstve pristupujeme cez Prisma ORM, čo umožňuje efektívnu správu a manipuláciu s databázovými objektmi.

Schéma databázy

Databáza obsahuje niekoľko základných modelov, ktoré reprezentujú rôzne entitné oblasti aplikácie. Každý model má definované atribúty a vzťahy, ktoré medzi nimi existujú. Schéma je definovaná v jazyku Prisma a vyzerá nasledovne:

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

Modely databázy

1. `User`

Model `User` reprezentuje používateľa aplikácie. Každý používateľ má unikátny `username` a `email`. Tento model obsahuje aj informácie o hesle, roli používateľa (napr. bežný používateľ alebo administrátor), a jeho priradené jedlá, aktivity, cvičenia a tréningové plány. Model ďalej obsahuje `refreshToken` pre správu relácií používateľa.

```
model User {
  id            String          @id @default(uuid())
  username      String          @unique
  email         String          @unique
  password      String
  role          Role            @default(USER)
  salt          String
  refreshToken  String?
  foodLogs      FoodLog[]
  activityLogs  ActivityLog[]
  exercises     Exercise[]
  trainingPlans Plan[]
  createdAt     DateTime        @default(now())
  updatedAt     DateTime        @updatedAt

  @@index([username, email, refreshToken])
}
```

2. `FoodLog`

Model `FoodLog` uchováva záznamy o jedlách pridaných používateľom. Obsahuje informácie o jednotlivých jedlách, ako je ich názov, kalorická hodnota, množstvo bielkovín, sacharidov a tukov. Každý záznam je spojený s konkrétnym používateľom.

```
model FoodLog {
  id       String   @id @default(uuid())
  user     User     @relation(fields: [user_id], references: [id])
  user_id  String
  foodItem String
  calories Int
  protein  Float
  carbs    Float
```

```
  fat         Float
  date        DateTime  @default(now())
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt
}
```

3. `ActivityLog`

Model `ActivityLog` uchováva informácie o aktivitách používateľa, ako je názov aktivity, jej trvanie v minútach, spálené kalórie a dátum vykonania aktivity. Tento model pomáha používateľom sledovať svoj pokrok v oblasti fyzickej aktivity.

```
model ActivityLog {
  id              String    @id @default(uuid())
  user            User      @relation(fields: [user_id], references: [id
  user_id         String
  activity        String
  duration        Int // in minutes
  caloriesBurned  Float
  date            DateTime  @default(now())
  createdAt       DateTime  @default(now())
  updatedAt       DateTime  @updatedAt
}
```

4. `Exercise`

Model `Exercise` reprezentuje jednotlivé cvičenia, ktoré si používateľ môže pridať do svojho tréningového plánu. Každé cvičenie má názov, typ (napr. beh, posilňovanie), intenzitu a môže byť spojené s konkrétnou časťou tela (napr. nohy, ruky).

```
model Exercise {
  id           String            @id @default(uuid())
  name         String            @unique
  bodyPart     String
  type         String
  owner        User              @relation(fields: [owner_id], referenc
  owner_id     String            @default(uuid())
  description  String?
  logo         String
  intensity    ExerciseIntensity @default(LOW)
```

```
  createdAt    DateTime            @default(now())
  updatedAt    DateTime            @updatedAt
}
```

5. `Plan`

Model `Plan` predstavuje tréningový plán, ktorý používateľ môže vytvoriť. Každý plán obsahuje názov, popis, trvanie a zoznam cvičení priradených k tomuto plánu.

```
model Plan {
  id           String          @id @default(uuid())
  name         String
  description  String
  duration     Int // D   ka v d och
  user         User            @relation(fields: [user_id], references:
  user_id      String
  date         DateTime        @default(now())
  createdAt    DateTime        @default(now())
  updatedAt    DateTime        @updatedAt
  exercises    PlanExercise[] // Prepojenie na PlanExercise
}
```

6. `PlanExercise`

Model `PlanExercise` slúži na prepojenie cvičení s tréningovými plánmi. Tento model uchováva informácie o počte sérií, opakovaní a odpočinku pre každé cvičenie v rámci plánu.

```
model PlanExercise {
  id             String @id @default(cuid())
  trainingPlanId String
  plan           Plan   @relation(fields: [trainingPlanId], references
  exerciseId     String // Cudz   k       na Exercise
  sets           Int
  reps           Int
  rest           Int
}
```

Enumy

1. `ExerciseIntensity`

Enum `ExerciseIntensity` definuje možné úrovne intenzity cvičenia:

- `LOW` - nízka intenzita.
- `MEDIUM` - stredná intenzita.
- `HIGH` - vysoká intenzita.

2. `Role`

Enum `Role` určuje rolu používateľa v aplikácii. Existujú dve možné hodnoty:

- `USER` - bežný používateľ.
- `ADMIN` - administrátor aplikácie.

Indexy a optimalizácia

Na zvýšenie výkonu pri vyhľadávaní používateľských údajov a autentifikácii sú definované indexy na polia `username`, `email` a `refreshToken` v modeli `User`. To umožňuje rýchlejší prístup k týmto často používaným údajom.

1 Testovanie REST API

Testovanie REST API endpointov je dôležitou súčasťou vývoja backendových aplikácií. Na tento účel môžeme použiť nástroje ako `Jest` a `Supertest`, ktoré umožňujú jednoduché a efektívne testovanie HTTP požiadaviek.

1.1 Príprava projektu na testovanie

Pre testovanie pomocou `Jest` je potrebné zabezpečiť, aby projekt podporoval testovanie s modernou syntaxou JavaScriptu. Ak sa používajú ES moduly (t. j. `import/export`), je potrebné nakonfigurovať prostredie tak, aby tieto moduly podporovalo. To zahŕňa:

- Inštaláciu balíčkov `@babel/preset-env` a `babel-jest`, ktoré umožnia transformáciu moderného JavaScriptu.
- Vytvorenie súboru `babel.config.js` s konfiguráciou:

```
module.exports = {
    presets: ['@babel/preset-env'],
};
```

- Aktualizáciu konfigurácie `jest.config.js`, aby používala transformáciu cez `babel-jest`.

Alternatívne je možné použiť `CommonJS` syntax (`require`), ktorá nevyžaduje ďalšiu konfiguráciu.

1.2 Ukážka testovacieho skriptu

Testovací skript pre `Jest` a `Supertest` môže vyzerať nasledovne:

```
import request from 'supertest';
import { app } from '../path-to-your-express-app';

describe('Testovanie API endpointov', () => {
    it('Vráti status 200 pre GET /food', async () => {
        const res = await request(app).get('/food');
        expect(res.status).toBe(200);
    });
});
```

Tento test kontroluje, či endpoint `/food` odpovedá s korektným stavovým kódom HTTP 200.

1.3 Spustenie testov

Testy je možné spustiť príkazom:

```
npm test
```

Ak testovanie vyžaduje podporu ES modulov, môže byť potrebné nastaviť `"type": "module"` v súbore `package.json` alebo použiť `babel-jest` na transformáciu kódu.

Testovanie REST API endpointov je kľúčové pre zaistenie stability a funkčnosti backendových služieb, pričom `Jest` v kombinácii s `Supertest` poskytuje intuitívne a robustné riešenie pre tieto úlohy.

Nasadenie aplikácie

Aplikácia bude nasadená na cloudovú platformu (napr. AWS, Heroku alebo DigitalOcean) s nasledujúcimi krokmi:

- Konfigurácia prostredia (NODE_ENV, DB_URL).
- Nasadenie backendu (Node.js).
- Nasadenie frontendových súborov (React/Vue).
- Sledovanie výkonu a chýb pomocou nástrojov ako Sentry alebo New Relic.

Bezpečnosť a ochrana dát

Autentifikácia a autorizácia

- Používame JWT (JSON Web Tokens) pre autentifikáciu.
- Hashed password (bcrypt) na ochranu hesiel.
- Validácia vstupov (napr. emailová validácia).

Šifrovanie údajov

- Všetky citlivé dáta sú šifrované pred uložením do databázy.
- HTTPS šifrovanie pre bezpečnú komunikáciu medzi klientom a serverom.

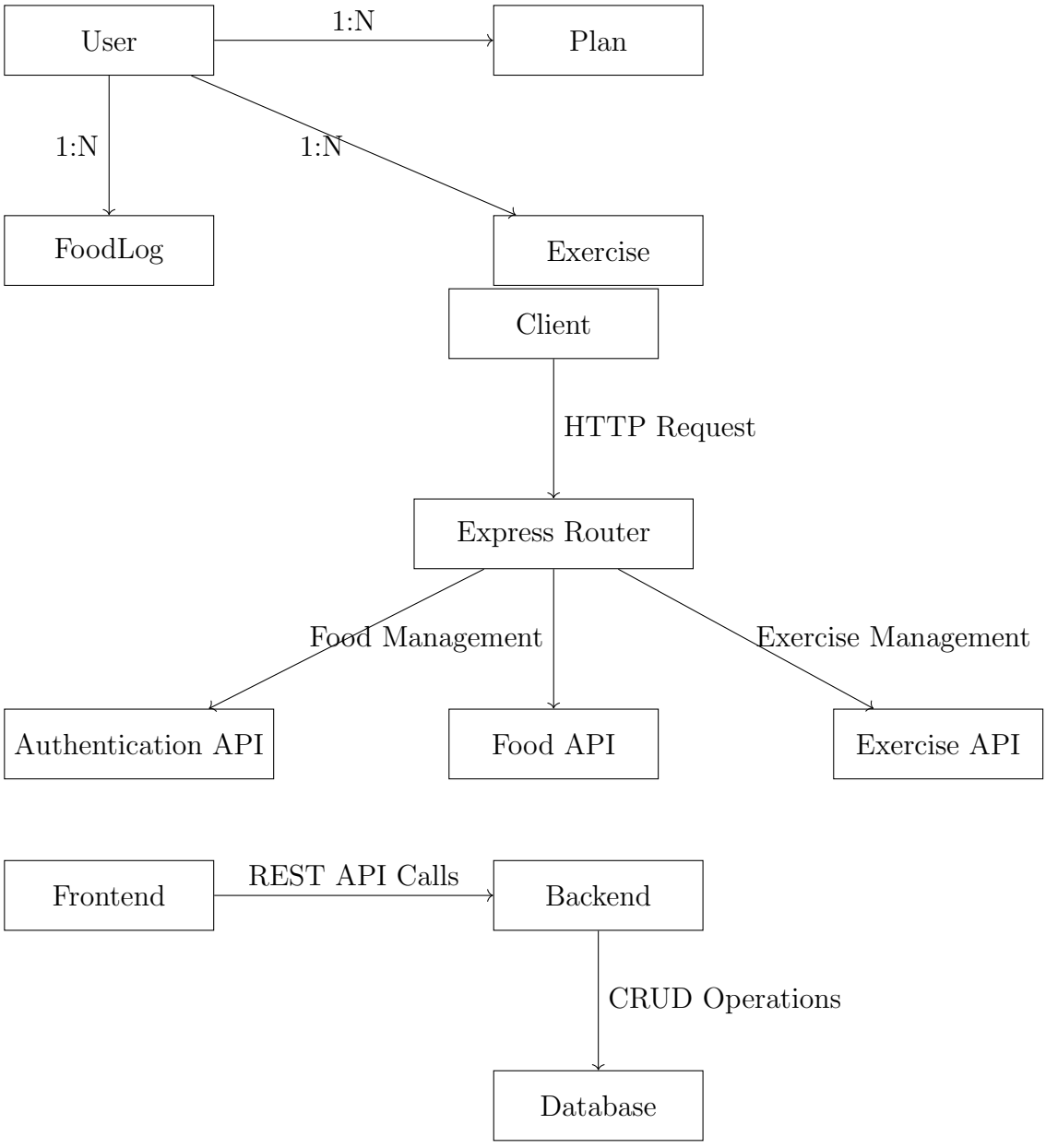
Ochrana proti útokom

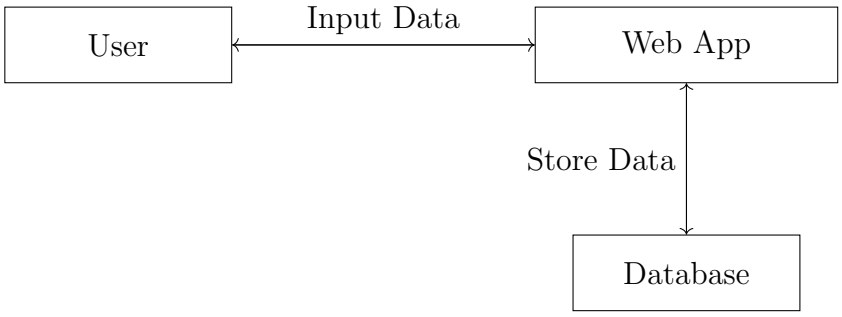
- Ochrana pred CSRF a XSS útokmi.
- Použitie CORS pre zabezpečenie prístupu iba z autorizovaných domén.

Záver a budúci vývoj

Aplikácia **Fitness App** sa neustále vyvíja. V budúcnosti plánujeme pridať nové funkcie ako:

- Integrácia s ďalšími nositeľnými zariadeniami.
- Pokročilé analýzy a reporty o výkone.
- Mobilná aplikácia pre iOS a Android.


User
1:N
Plan
1:N
1:N
FoodLog
Exercise
Client
HTTP Request
Express Router
Food Management
Exercise Management
Authentication API
Food API
Exercise API
Frontend
REST API Calls
Backend
CRUD Operations
Database


User
Input Data
Web App
Store Data
Database