Leonie Adelwoehrer

Quarkus Full-Stack Übung: Naturschutzgebiet-Management

Aufgabenstellung

Erstellen Sie ein Backend mit **Quarkus** auf Basis der gegebenen Datenbankstruktur. Dazu sollen eine **REST-API**, **DTOs**, **JPQL-Abfragen**, sowie **WebSockets** für Echtzeit-Aktualisierungen implementiert werden. Im Frontend setzen Sie eine Angular-Anwendung um, die mit dem Backend kommuniziert.

Backend-Anforderungen (Quarkus + Hibernate + REST API)

1. Datenbankmodell in Quarkus umsetzen

- Implementieren Sie die Datenbanktabellen als **JPA-Entities**.
- Nutzen Sie **Relationen** (OneToMany, ManyToOne, ManyToMany) entsprechend der gegebenen Struktur.
- Benennen Sie die Felder sinnvoll um (siehe Liste mit besseren Bezeichnungen).

2. REST-API für CRUD-Operationen entwickeln

Erstellen Sie eine REST-API mit folgenden **CRUD-Operationen** für die wichtigsten Entitäten:

- **Naturschutzgebiet:**
  - Anlegen, Bearbeiten, Löschen, Abrufen aller Gebiete oder eines einzelnen Gebiets.
  - Filterung nach Standort (JPQL-Abfrage).

    Naturschutzgebiet-API (`/naturschutzgebiete`)

    ✅ **GET** `/naturschutzgebiete` → Alle Naturschutzgebiete abrufen.
    ✅ **GET** `/naturschutzgebiete/{id}` → Ein Naturschutzgebiet mit Details abrufen.
    ✅ **POST** `/naturschutzgebiete` → Neues Naturschutzgebiet anlegen.
    ✅ **PUT** `/naturschutzgebiete/{id}` → Ein Naturschutzgebiet aktualisieren.
    ✅ **DELETE** `/naturschutzgebiete/{id}` → Ein Naturschutzgebiet löschen.
    ✅ **GET** `/naturschutzgebiete?ort={ort}` → Nach Standort filtern (JPQL-Abfrage).

- **Tiere:**
  - Tiere in einem bestimmten Naturschutzgebiet abrufen.

- Tiere anlegen, bearbeiten und löschen.

Tier-API (`/tiere`)

- ✅ **GET** `/tiere` → Alle Tiere abrufen.
- ✅ **GET** `/tiere/{id}` → Ein Tier mit Details abrufen.
- ✅ **POST** `/tiere` → Ein neues Tier anlegen.
- ✅ **PUT** `/tiere/{id}` → Ein Tier aktualisieren.
- ✅ **DELETE** `/tiere/{id}` → Ein Tier löschen.
- ✅ **GET** `/tiere/naturschutzgebiet/{gebietId}` → Alle Tiere in einem Naturschutzgebiet

- **Giftpflanzen:**
  - Anlegen, Bearbeiten, Löschen, Abrufen.
  - Abrufen nach Toxizitätsstufe (JPQL-Abfrage).
  - Berechnung der durchschnittlichen Ausbreitungsrate aller Giftpflanzen in einer Region.

Giftpflanzen-API (`/giftpflanzen`)

- ✅ **GET** `/giftpflanzen` → Alle Giftpflanzen abrufen.
- ✅ **GET** `/giftpflanzen/{id}` → Eine spezifische Giftpflanze abrufen.
- ✅ **POST** `/giftpflanzen` → Eine neue Giftpflanze anlegen.
- ✅ **PUT** `/giftpflanzen/{id}` → Eine Giftpflanze aktualisieren.
- ✅ **DELETE** `/giftpflanzen/{id}` → Eine Giftpflanze löschen.
- ✅ **GET** `/giftpflanzen/toxizitaet/{stufe}` → Giftpflanzen nach Toxizitätsstufe abrufen.
- ✅ **GET** `/giftpflanzen/region/{region}` → Giftpflanzen in einer Region abrufen.
- ✅ **GET** `/giftpflanzen/ausbreitung/durchschnitt` → Durchschnittliche Ausbreitungsrate

- **Forschung & Wissenschaftler:**
  - Wissenschaftler anlegen und Forschungsprojekte abrufen.
  - Forscher mit bestimmten Tierarten verknüpfen.

Forschungs-API (`/forschung`)

- ✅ **GET** `/forschung` → Alle Forschungsprojekte abrufen.
- ✅ **POST** `/forschung` → Eine neue Forschung anlegen.
- ✅ **DELETE** `/forschung/{tierId}/{wissenschaftlerId}` → Forschung entfernen.
- ✅ **GET** `/forschung/wissenschaftler/{id}` → Forschungsprojekte eines Wissenschaftlers abrufen.
- ✅ **GET** `/forschung/tiere/{art}` → Wissenschaftler finden, die mit einer bestimmten Tierart forschen.

- **Trackinggerät & Wanderungen:**
  - Trackinggeräte einem Tier zuweisen.

- Bewegungsdaten abrufen (Filterung nach Zeitraum).

Wanderungen-API (`/wanderungen`)

- ✅ **GET** `/wanderungen` → Alle Wanderungen abrufen.
- ✅ **POST** `/wanderungen` → Eine neue Wanderung speichern.
- ✅ **GET** `/wanderungen/tier/{tierId}` → Wanderungen eines bestimmten Tieres abrufen.
- ✅ **GET** `/wanderungen/zeitraum?start={start}&end={end}` → Wanderungen in einem Zeitraum

3. DTOs für die Kommunikation nutzen

- Erstellen Sie **DTO-Klassen** für die Rückgabe und zum Anlegen/Ändern von Daten.
- Beispiel:
  - GiftpflanzeDTO (enthält Name, Toxizitätsstufe, Beschreibung).
  - WanderungDTO (enthält Datum, Region, Tier-Name).
  - NaturschutzgebietSummaryDTO (Name, Ort, Anzahl der Tiere).

4. Komplexe JPQL-Abfragen einbauen

- Durchschnittliche Toxizitätsstufe aller Giftpflanzen berechnen.
- Alle Wanderungen eines bestimmten Tieres in einem Zeitraum abrufen.
- Alle Wissenschaftler, die mindestens mit zwei Tierarten forschen.

5. WebSockets für Echtzeit-Updates einbinden

- Senden Sie **Live-Updates**, wenn ein neues Tier hinzugefügt oder eine Wanderung gespeichert wird.
- Frontend soll sich automatisch aktualisieren, wenn neue Daten verfügbar sind.

Frontend-Anforderungen (Angular)

1. Routing & Navigation

- Mindestens 4 Seiten:
    - Liste aller Naturschutzgebiete
    - Detailansicht eines Naturschutzgebiets mit seinen Tieren
    - Verwaltung der Giftpflanzen
    - Forschung & Wissenschaftler

2. Kommunikation mit Backend

- **GET-Requests**, um Listen anzuzeigen.
- **POST/PUT/DELETE**, um Daten zu erstellen oder zu ändern.
- Validierung von Formularen mit Fehlermeldungen.

3. Listen & Tabellen darstellen

- Anzeige der Tiere eines Naturschutzgebiets als Tabelle (mit Angular Material).
- Giftpflanzen in einer Karte (z. B. mit Leaflet.js).
- Wissenschaftler und ihre Forschung als Liste.

4. Formulare mit Validierung

- Neues Naturschutzgebiet anlegen.
- Giftpflanzen-Daten bearbeiten (Pflichtfelder, Drop-down für Stufen).
- Forschungseinträge zuweisen.

5. WebSockets zur Live-Aktualisierung nutzen

- Automatische Aktualisierung, wenn eine neue Wanderung erfasst wird.
- Benachrichtigung, wenn ein Tier hinzugefügt wird.

Bonus (Optional)

- **Authentifizierung mit JWT** (z. B. Admin darf Daten ändern, User nur lesen).
- **CSV-Export der Giftpflanzendaten**.
- **Dynamische Diagramme für Toxizitätsstufen**.