003-Rust-Diesel

Backend Aufgabe
Anforderungsniveau: Mittel

1 Ziel der Aufgabe

Ziel dieser Aufgabenstellung ist es, Ihre Kenntnisse in moderner Backend-Entwicklung mit Rust zu prüfen. Sie implementieren eine REST-API mit dem Framework Warp und binden eine PostgreSQL-Datenbank an. Optional können Sie für ORM und Authentifizierung Diesel einsetzen und das Projekt dockerisieren.

2 Aufgabenbeschreibung

Erstellen Sie eine Rust-Applikation, die folgende Funktionalitäten bietet:

1. **Grundgerüst der API:**
   - Verwenden Sie das Warp-Framework für Routing und Middleware.
   - Strukturieren Sie Ihr Projekt übersichtlich (z. B. `src/routes`, `src/models`, `src/db`).
2. **Datenbankanbindung:**
   - Richten Sie eine PostgreSQL-Datenbank ein und verbinden Sie diese mit Ihrer App.
   - Optional: Nutzen Sie Diesel als ORM für Schema-Management und Query-Building.
3. **REST-Endpunkte für einen Webshop:**
   - **User Authentication & Management:**
     - Registrierung, Login, Passwort-Reset.
     - JWT-basierte Authentifizierung oder Sessions.
   - **Items mit Lagerbestand:**
     - CRUD-Operationen für Produktdaten (Name, Beschreibung, Preis, Bestand).
   - **Warenkorb:**
     - Warenkorb anlegen, Einträge hinzufügen/entfernen, Menge anpassen.

3 Anforderungen

- **Code-Organisation:** Klare Modulstruktur, saubere Trennung von Verantwortlichkeiten.
- **Typensicherheit:** Rust's Typsystem voll ausnutzen, möglichst keine `unwrap()`-Aufrufe ohne Kontext.
- **Fehlerbehandlung:** Verwenden Sie `Result` und den `?`-Operator; sinnvolle HTTP-Statuscodes.
- **Dokumentation:** Kurz-README mit Aufsetzen-, Build- und Run-Anleitung sowie API-Referenz (z. B. mit OpenAPI oder REST-docs).
- **Tests:** Unit-Tests für Geschäftslogik; Mock-DB oder In-Memory-DB nach Möglichkeit.

4 Optionale Erweiterungen

1. **ORM-Integration mit Diesel:**
   - Automatisches Schema-Migrationssystem.
   - Typed Query Builder für alle Datenbankzugriffe.
2. **Dockerisierung:**
   - `Dockerfile` und `docker-compose.yml` für App und PostgreSQL.

5 Zusätzliche Anforderungen für Experten (Kann-Kriterien)

Die folgenden Punkte sind optionale Kann-Kriterien und keine Muss-Anforderungen:

- **Integrationstests:** Vollständige Testsuite, die jede Route inklusive Auth-Flow gegen eine Testdatenbank abdeckt.
- **Custom-Typen:** Definieren Sie eigene Typen (z. B. für `Money`, `Email`, `Quantity`) mit jeweils eigener `FromRow`-/`ToSql`-Implementierung in Diesel.
- **Sicherheitsaspekte:** Schutz vor SQL-Injection (Prepared Statements), Rate-Limiting, sichere Passwort-Hashes (argon2 oder bcrypt).

- **Bilder-Upload:** Implementieren Sie einen Endpunkt zum Upload von Bildern und speichern Sie die Dateien in einem lokal gemounteten Verzeichnis (z. B. `./uploads`).

6 Abgabe und Bewertung

- Stellen Sie Ihr Projekt in einem öffentlichen Git-Repository bereit und senden Sie uns den Link.
- Beurteilt werden:
  - Funktionalität: Erfüllt die API alle geforderten Endpunkte?
  - Codequalität: Lesbarkeit, Modularität, Rust-Best-Practices.
  - Sicherheit: Sichere Authentifizierung und DB-Zugriffe.
  - Dokumentation: Verständliche README, API-Referenz.
  - Optionale Features: Docker, Diesel-Integration, Tests.

Viel Erfolg bei der Umsetzung!
Stand: 31. Mai 2025