Roadmap di Sviluppo: Step-by-Step

Ecco la procedura consigliata per affrontare il progetto in modo ordinato. Non cercate di scrivere tutto il codice in una volta: procedete per piccoli passi e verificate che ogni passo funzioni prima di passare al successivo.

Step 1: Preparazione dell'Ambiente Database (Podman)

Prima di scrivere una singola riga di Java, dovete avere un database funzionante.

1. **Container Run:** Avviate un container PostgreSQL usando **Podman**
    - Assicuratevi di esporre la porta standard 5432.
    - Impostate una password per l'utente postgres (o create un utente dedicato).
    - *Esempio comando:* podman run --name inventory-db -e POSTGRES_PASSWORD=mysecret -p 5432:5432 -d postgres
2. **Verifica Connessione:** Usate un client SQL (come DBeaver, pgAdmin o la tab "Database" di IntelliJ) per connettervi al database.
    - Create un database vuoto chiamato inventory_db.
3. **Creazione Schema:** Scrivete ed eseguite gli script SQL (DDL) per creare le 4 tabelle (office, asset_type, asset, software_license) e la tabella di join (asset_licenses).
    - *Consiglio:* Definite bene le Primary Key e le Foreign Key ora.

Step 2: Inizializzazione del Progetto Spring Boot

1. Andate su **Spring Initializr** (start.spring.io).
2. Configurate il progetto:
    - **Project:** (Gradle).
    - **Language:** Java.
    - **Spring Boot:** Versione 3.x (stabile).
    - **Packaging:** Jar.
    - **Java:** 17.
3. Selezionate le dipendenze essenziali:
    - Spring Web (per le API REST).
    - Spring Data JPA (per il DB).
    - PostgreSQL Driver (il connettore).
    - Lombok (per ridurre il codice ripetitivo).
4. Generate, scaricate e aprite il progetto nel vostro IDE.

Step 3: Connessione Applicazione-Database

1. Aprite il file src/main/resources/application.yaml

2. Inserite le credenziali per collegarvi al vostro container Podman:
    - URL JDBC (es. jdbc:postgresql://localhost:5432/inventory_db).
    - Username e Password.
3. **Smoke Test:** Avviate l'applicazione (metodo main).
    - Se parte senza errori rossi in console e vedete "Started Application...", la connessione al DB è riuscita.

Step 4: Mapping delle Entità

1. Create un package model o entity.
2. Create le classi Java (Office, Asset, etc.) che rispecchiano le tabelle create allo Step 1.
3. Usate le annotazioni JPA (@Entity, @Table, @Id, @Column, @ManyToOne, ecc.) per mappare i campi.
4. *Verifica:* Riavviate l'applicazione. Se avete sbagliato i nomi delle colonne o le relazioni, Hibernate vi darà errore all'avvio. Correggete finché l'app non parte pulita.

Step 5: Sviluppo Verticale (Iterazione)

Non fate tutti i Controller insieme. Completate **una entità alla volta** (iniziate da quella più semplice, es. Office).

1. **Repository:** Create l'interfaccia OfficeRepository che estende JpaRepository.
2. **Service:** Create OfficeService dove scrivete la logica (es. getAllOffices, createOffice).
3. **Controller:** Create OfficeController ed esponete gli endpoint (es. @GetMapping, @PostMapping).
4. **Test Manuale:** Aprite **Postman**, create la Collection e provate a salvare un Ufficio e poi a rileggerlo.
    - *Funziona?* -> Commit su Git.
    - *Non funziona?* -> Debug (guardate i log).

Step 6: Le Relazioni Complesse

Solo quando le entità semplici (Office, AssetType) funzionano, passate a Asset.

1. Implementate il salvataggio di un Asset (che richiede ID di Ufficio e Tipo esistenti).
2. Implementate la relazione Molti-a-Molti per le licenze software.

Step 7: Rifinitura

1. Aggiungete i log (log.info, log.error) nei punti critici del Service.
2. Pulite il codice (rimuovete import non usati, commenti inutili).
3. Verificate che tutto il codice (nomi variabili, commenti javadoc) sia in **Inglese**.

Requisiti Tecnici e Standard di Sviluppo

Il rispetto dei seguenti standard è vincolante per l'accettazione del progetto.

Stack Tecnologico

- **Linguaggio:** Java 17 o superiore.
- **Framework:** Spring Boot 3.x.
- **Gestione Dipendenze:** Gradle.
- **Database:** PostgreSQL.
- **ORM:** Spring Data JPA.

Best Practices & Code Quality

- **Lingua del Codice:** L'intero codebase deve essere scritto rigorosamente in **Inglese**.
    - **Naming:** Nomi di variabili, metodi e classi devono essere in inglese (es. purchaseDate, findAssetsByOffice).
    - **Commenti:** Tutti i commenti al codice e la Javadoc devono essere in inglese professionale.
- **Architettura a Livelli:** Separazione netta tra Controller, Service e Repository.
- **DTO Pattern:** È vietato esporre le Entity JPA direttamente nei Controller. Utilizzare classi DTO per disaccoppiare il database dalle API.
- **Logging:** Utilizzo di SLF4J (livelli INFO ed ERROR) al posto di System.out.println.

Versionamento e Consegna

- **Git Repository:** Il progetto deve essere creato e mantenuto su un repository Git personale
- **Commit Policy:** I messaggi di commit devono essere in inglese e significativi. È richiesto un utilizzo frequente dei commit (non un unico "upload" finale).
- **Deliverables:**
    - Link al repository Git.
    - Collection di Postman (configurata con variabili d'ambiente per il base_url) che copra tutti gli endpoint.