Tables

1. **properties**:
    - id (PK)
    - title (NOT NULL)
    - owner_id
2. **bookings**:
    - id (PK)
    - property_id (FK to properties)
    - users_id (FK to users)
    - start_date (NOT NULL)
    - end_date (NOT NULL)
3. **users**:
    - id (PK)
    - name (NOT NULL)

API Design

Property Endpoints

- **POST** /api/properties: Create new property
- **GET** /api/properties: Get paginated list of properties
- **GET** /api/properties/{id}: Get property by ID
- **PUT** /api/properties/{id}: Update property
- **DELETE** /api/properties/{id}: Delete property

Technical Stack

- **Backend Framework**: Spring Boot
- **Database**: PostgreSQL (implied by JPA annotations)
- **ORM**: Spring Data JPA
- **Build Tool**: Maven
- **Validation**: Jakarta Validation

Implementation Design

1. **Layered Architecture**: Separation of concerns with clear boundaries between controller, service, and repository layers.
2. **DTO Pattern**: Use of Data Transfer Objects (PropertyDTO, BookingDTO)
3. **Pagination**: Implemented for property listings to support large datasets.
4. **Validation**:
    - Field-level validation (e.g., @NotBlank, @NotNull)
    - Temporal validation (@FutureOrPresent for booking dates)