Báo cáo bài tập lớn

Học phần lập trình hướng đối tượng

Chủ đề : Quản lý bất động sản

Nhóm 1

1. Vũ Minh Hiếu
2. Phạm Duy Hưng
3. Hà Ngọc Huy
4. Hà Thành Long

LỜI NÓI ĐẦU

Ngày nay, với tốc độ đô thị hóa nhanh chóng, nhu cầu quản lý thông tin bất động sản một cách khoa học, chính xác và hiệu quả ngày càng trở nên cấp thiết. Hệ thống **Quản lý Bất động sản** ra đời với mục tiêu giải quyết bài toán về việc lưu trữ, truy xuất và xử lý thông tin liên quan đến các tòa nhà, diện tích cho thuê, người dùng và vai trò trong hệ thống.

Ứng dụng được thiết kế nhằm hỗ trợ các doanh nghiệp, nhà quản lý bất động sản và khách hàng có thể dễ dàng tương tác, cập nhật thông tin và đưa ra các quyết định nhanh chóng. Với khả năng lưu trữ dữ liệu tòa nhà, diện tích cho thuê, và yêu cầu của khách hàng một cách hệ thống, ứng dụng giúp tối ưu hóa quy trình vận hành, tiết kiệm thời gian và chi phí cho doanh nghiệp.

Hệ thống **Quản lý Bất động sản** được xây dựng trên nền tảng công nghệ hiện đại như **Spring Boot**, **Java Core**, và các công cụ hỗ trợ như **Hibernate (JPA)**, **Thymeleaf**, cùng các nguyên lý **lập trình hướng đối tượng (OOP)**. Nhờ đó, ứng dụng không chỉ đảm bảo hiệu suất cao mà còn dễ bảo trì, mở rộng trong tương lai.

Bên cạnh đó, hệ thống phân chia rõ ràng vai trò người dùng (**Admin, Staff, Customer**), mỗi vai trò được phân quyền cụ thể với các chức năng riêng biệt, giúp quản lý dữ liệu một cách minh bạch và hiệu quả. **Admin** đóng vai trò quản trị hệ thống, **Staff** chịu trách nhiệm cập nhật thông tin tòa nhà, và **Customer** tìm kiếm, gửi yêu cầu thuê tòa nhà hoặc diện tích cho thuê phù hợp với nhu cầu của họ.

Việc áp dụng chặt chẽ các nguyên lý **SOLID** và thiết kế hệ thống theo mô hình **RESTful API** giúp hệ thống hoạt động mượt mà, đáp ứng nhu cầu từ giao diện đến logic xử lý backend. Đây là một công cụ mạnh mẽ, giúp nâng cao khả năng quản lý và hỗ trợ ra quyết định nhanh chóng trong lĩnh vực bất động sản.

Với sự phát triển của công nghệ thông tin và nhu cầu số hóa, **ứng dụng Quản lý Bất động sản** hứa hẹn sẽ trở thành một giải pháp hiệu quả, góp phần thúc đẩy sự chuyên nghiệp hóa và hiện đại hóa trong lĩnh vực này.

CHI TIẾT ROLES VÀ THỰC THỂ

I. Role : Admin

Admin là người quản trị hệ thống với quyền hạn cao nhất.

- **Chức năng chính**:
  - Quản lý **toàn bộ dữ liệu** trong hệ thống.
  - Thêm, sửa, xóa thông tin **tòa nhà** và **người dùng**.
  - Quản lý **vai trò (Roles)** và phân quyền cho từng người dùng.
  - Thống kê và báo cáo dữ liệu về **tòa nhà** và **người dùng**.
  - Quản lý các **diện tích cho thuê** trong các tòa nhà.
- **Chi tiết chức năng**:

Chức năng	Mô tả
Thêm/Sửa/Xóa Tòa nhà	Cho phép thêm, chỉnh sửa hoặc xóa thông tin các tòa nhà.
Quản lý người dùng	Tạo, sửa, xóa người dùng và phân quyền.
Quản lý Roles	Tạo các vai trò mới và cấp quyền cho từng vai trò cụ thể.
Thống kê báo cáo	Thống kê thông tin tòa nhà và người dùng dưới dạng biểu đồ và bảng.
Quản lý diện tích thuê	Gán các diện tích thuê cho từng tòa nhà cụ thể.

II. Role : Quản lý

Staff là nhân viên phụ trách quản lý và cập nhật thông tin tòa nhà.

- **Chức năng chính**:
  - Cập nhật thông tin các **tòa nhà**.
  - Quản lý và cập nhật **diện tích cho thuê**.
  - Xử lý các **yêu cầu từ khách hàng** liên quan đến tòa nhà.
- **Chi tiết chức năng**:

Chức năng	Mô tả
Cập nhật thông tin Tòa nhà	Chỉnh sửa các thông tin như địa chỉ, diện tích…

Quản lý diện tích thuê	Cập nhật các diện tích còn trống hoặc đã cho thuê.
Xử lý yêu cầu khách hàng	Tiếp nhận và phản hồi các yêu cầu từ phía khách hàng (Customer).

III. Role : Khách hàng

Customer là khách hàng sử dụng hệ thống để tìm kiếm thông tin tòa nhà và gửi yêu cầu thuê.

- **Chức năng chính**:
  - Tìm kiếm và xem thông tin **tòa nhà** theo các tiêu chí như **giá thuê, diện tích, địa chỉ**.
  - Gửi **yêu cầu thuê tòa nhà** hoặc diện tích cụ thể.
  - Xem lịch sử và trạng thái các yêu cầu đã gửi.
- **Chi tiết chức năng**:

Chức năng	Mô tả
Tìm kiếm tòa nhà	Lọc và tìm kiếm tòa nhà dựa trên diện tích, giá thuê, địa chỉ.
Gửi yêu cầu thuê	Gửi thông tin yêu cầu thuê diện tích hoặc tòa nhà cụ thể.
Xem lịch sử yêu cầu	Xem lại các yêu cầu thuê đã gửi và trạng thái phản hồi.

IV. Thực thể

Tất cả các thực thể đều kế thừa từ lớp **AbstractDTO**, lớp này cung cấp các thuộc tính dùng chung như:

- id: Long
- createdDate: Date
- createdBy: String
- modifiedDate: Date
- modifiedBy: String

1. **BuildingDTO** (mở rộng từ AbstractDTO)

   Mô tả: Đại diện cho thông tin tòa nhà trong hệ thống.

   Thuộc tính:

   - name: String
   - street: String

- ward: String
- district: String
- numberOfBasement: Long
- floorArea: Long
- rentArea: String
- managerName: String
- managerPhone: String
- serviceFee: String
- rentPrice: Long
- overtimeFee: String
- direction: String
- structure: String
- note: String
- image: String

2. **UserDTO** (mở rộng từ AbstractDTO)
   Mô tả: Quản lý thông tin người dùng và vai trò.
   Thuộc tính:

   - userName: String
   - fullName: String
   - password: String
   - status: Integer
   - roles: List<RoleDTO>
   - roleName: String
   - roleCode: String

3. **RoleDTO** (mở rộng từ AbstractDTO)
   Mô tả: Đại diện cho vai trò của người dùng.
   Thuộc tính:

   - userName: String
   - fullName: String

4. **CustomerDTO** (mở rộng từ AbstractDTO)
   Mô tả: Thông tin khách hàng trong hệ thống.
   Thuộc tính:

   - name: String
   - managementStaff: String
   - customerPhone: String
   - email: String
   - demand: String
   - status: String
   - companyName: String

5. **AssignmentBuildingDTO**
   Mô tả: Gán tòa nhà cho nhân viên.
   Thuộc tính:

- buildingId: Long
- staffs: List<Long>

6. **DistrictDTO**
   Mô tả: Thông tin quận/huyện liên quan đến tòa nhà.
   Thuộc tính:

   - buildingIds: Long[]

7. **PasswordDTO** (mở rộng từ AbstractDTO)

   Mô tả: Xử lý yêu cầu đổi mật khẩu của người dùng.

   Thuộc tính:

   - oldPassword: String
   - newPassword: String
   - confirmPassword: String

V. Các công nghệ sử dụng trong bài

Frontend:

- Giao diện người dùng sử dụng **JavaServletPage** để hiển thị dữ liệu.

Backend:

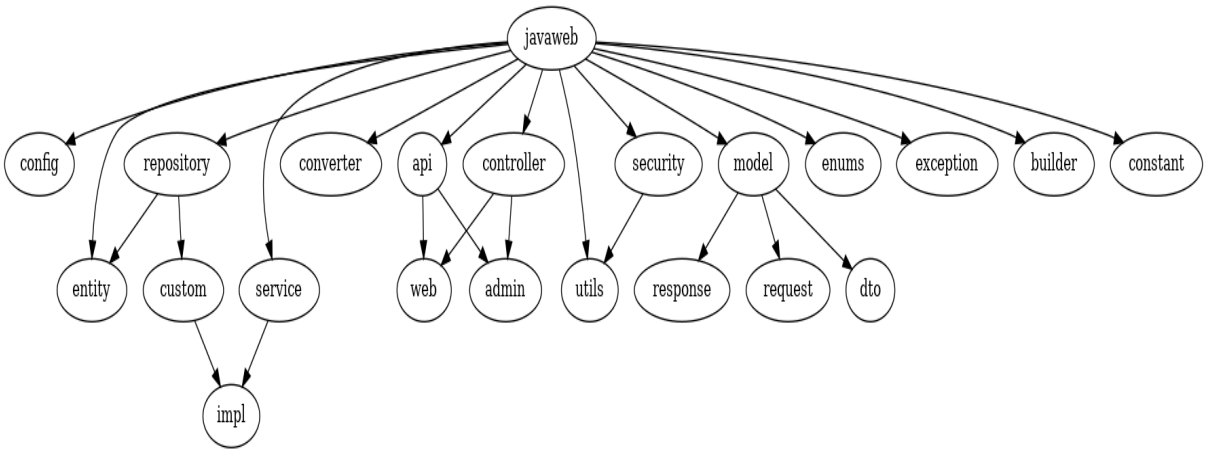
- **Spring Boot** được sử dụng làm framework chính để phát triển server backend.
  - File SpringBootWebApplication.java là entry point chính cho ứng dụng.
  - Các controller xử lý API được triển khai trong các file như:
    - BuildingAPI.java, UserAPI.java, NewAPI.java.
- Các dịch vụ và repository tương tác dữ liệu:
  - Repository: BuildingRepository.java, UserRepository.java.
  - Service: BuildingServiceImpl.java, UserService.java.
- **Lombok** được sử dụng để tự động hóa mã nguồn thông qua các annotation như @Data, @Getter, @Setter.

Xử lý Dữ liệu và Serialization:

- **Gson**: Xử lý chuyển đổi JSON cơ bản.
- **Jackson**: Kết hợp trong Spring Boot để serialize và deserialize dữ liệu JSON.
- Dữ liệu được cấu trúc và chuyển đổi trong các DTO:
  - BuildingDTO.java, UserDTO.java, ResponseDTO.java.

BIỂU ĐỒ UML

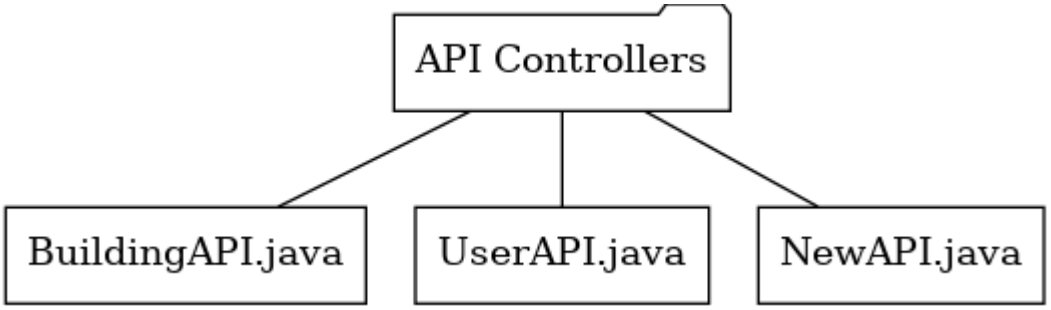
I. Biểu đồ phụ thuộc gói



II. Biểu đồ lớp

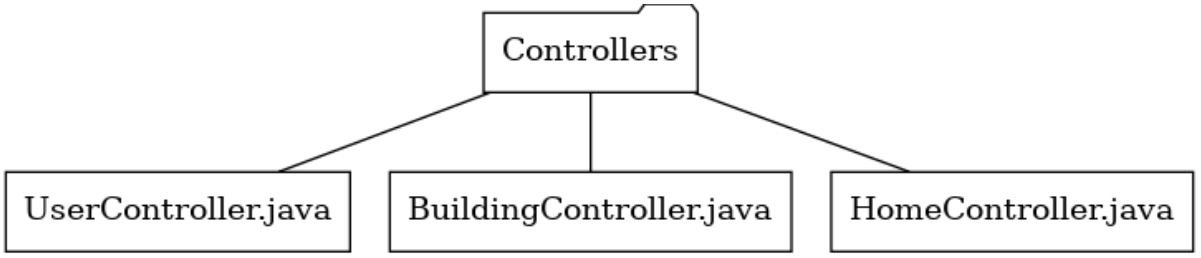
1. API Controllers

   BuildingAPI.java, **UserAPI.java**, **NewAPI.java**: Xử lý yêu cầu HTTP từ client và điều phối dữ liệu giữa các tầng khác nhau.



2. Controllers

   BuildingController.java, **HomeController.java**, **UserController.java**: Quản lý điều hướng giao diện và xử lý các tác vụ liên quan đến người dùng.

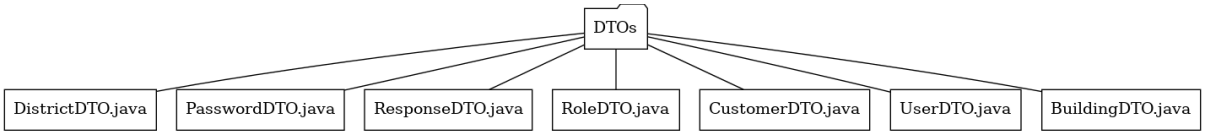


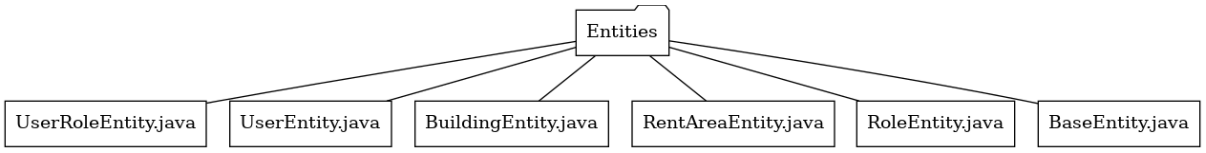
3. DTOs

BuildingDTO.java, **CustomerDTO.java**, **RoleDTO.java**, **UserDTO.java**: Đại diện dữ liệu gửi/nhận giữa client và server.

4. Entities

BaseEntity.java, **UserEntity.java**, **RoleEntity.java**, **BuildingEntity.java**: Các lớp mô tả cấu trúc bảng trong cơ sở dữ liệu.

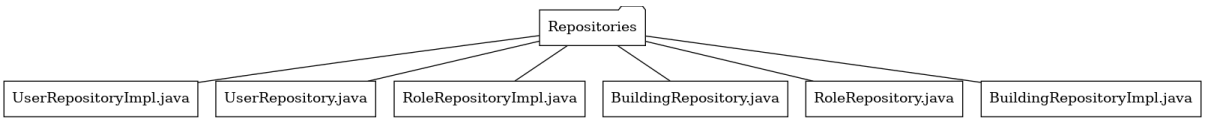




5. Repositories

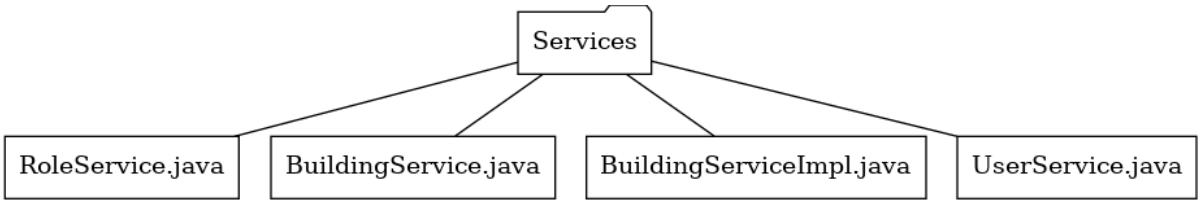
UserRepository.java, **BuildingRepository.java**, **RoleRepository.java**: Giao diện truy vấn dữ liệu từ database.

Custom Implementations: **BuildingRepositoryImpl.java**, **RoleRepositoryImpl.java**.



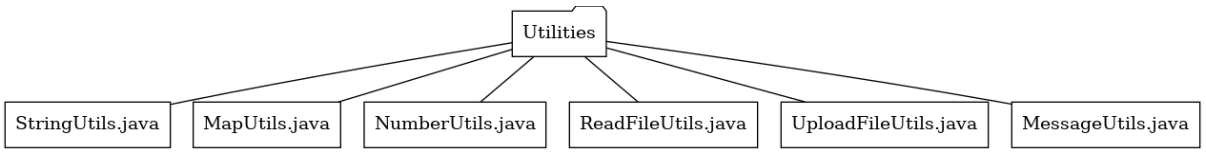
6. Services

BuildingService.java, **UserService.java**, **RoleService.java**: Xử lý logic nghiệp vụ.



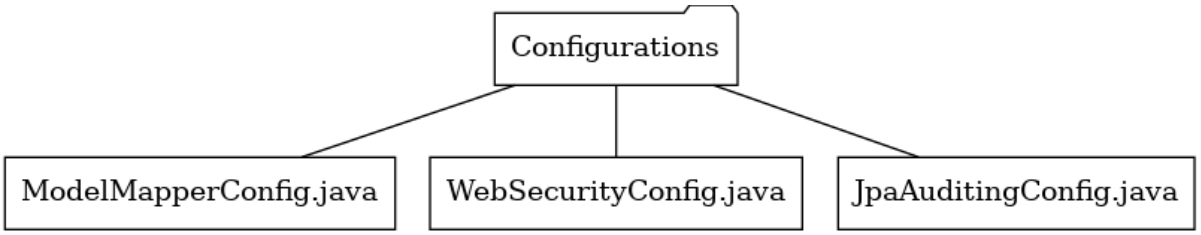
7. Utilities

MapUtils.java, **StringUtils.java**, **UploadFileUtils.java**, **MessageUtils.java**: Các tiện ích hỗ trợ trong quá trình xử lý dữ liệu.



8. Configurations

WebSecurityConfig.java, **JpaAuditingConfig.java**, **ModelMapperConfig.java**: Cấu hình bảo mật và các thành phần Spring Boot.



HƯỚNG DẪN SỬ DỤNG

Video demo : [Link]

1. Clone the repository:

   git clone https://github.com/linhgrr/estate_management.git

2. .Import the SQL file using MySQL stored in the Database folder, and edit the DATABASE connection information in the 'application.properties' file:

   spring.datasource.url = database URL

   spring.datasource.username = account

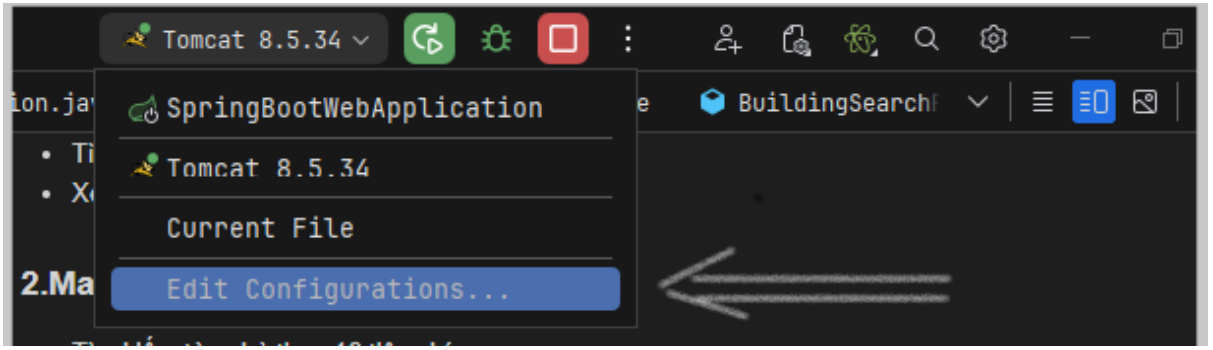
   spring.datasource.password = password

3. In Project Structure, set:

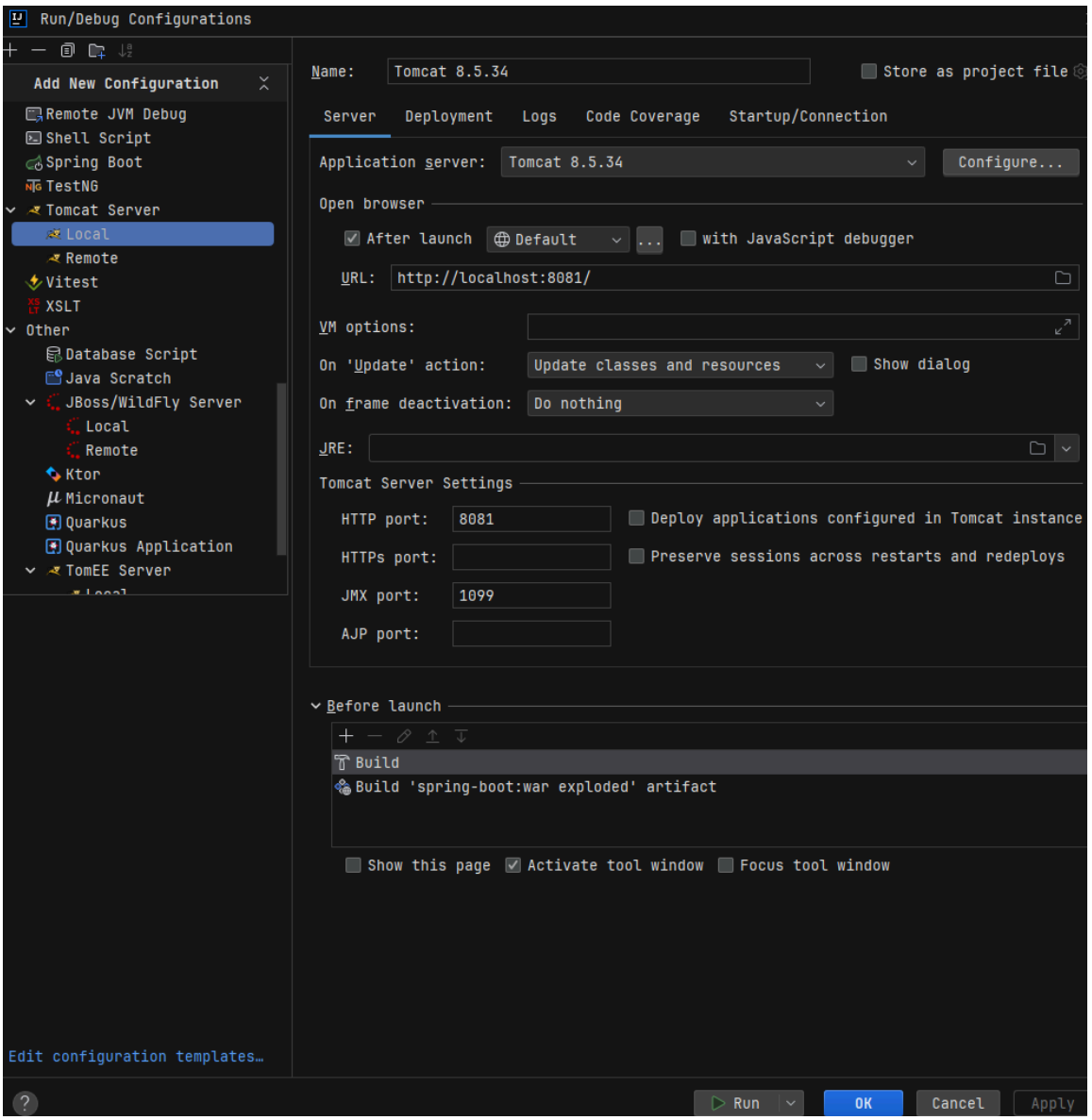
   SDK: 1.8 (OpenJDK)

   Language level: 8 - Lambdas, type annotations, etc.

4. In Run/Debug Configuration, select Edit Configurations...



Setup as shown below:



5. Press RUN to launch the web, a browser window will open automatically, or access it at http://localhost:8081