This document defines a **complete Admin CRUD architecture** for your database schema, designed for **Spring Boot + JPA + Thymeleaf**, aligned with your **enterprise PhilPost red theme**.

---

1. ADMIN SCOPE (What Admin Manages)

Admin has CRUD access to:

Lookup Tables

- Area
- Department
- Job Position
- User Roles
- Intended Viewer Group
- Document Classification

Core Tables

- Office
- Users
- Office Viewer Mapping
- Importable Documents (metadata only)

---

2. PROJECT STRUCTURE (RECOMMENDED)

```
com.yourapp
 ├── config
 │    └── SecurityConfig.java
 ├── controller
 │    └── admin
 │         ├── AreaAdminController.java
 │         ├── DepartmentAdminController.java
 │         ├── JobPositionAdminController.java
 │         ├── RoleAdminController.java
 │         ├── OfficeAdminController.java
 │         ├── UserAdminController.java
 │         ├── ViewerGroupAdminController.java
 │         ├── OfficeViewerAdminController.java
 │         └── DocumentAdminController.java
 ├── entity
 ├── repository
 ├── service
 └── templates
```

```
        └── admin
            ├── area
            ├── department
            ├── office
            ├── users
            └── fragments
```

---

3. ENTITY DEFINITIONS (JPA)

Example: Area Entity

```java
@Entity
@Table(name = "area")
public class Area {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false, length = 100)
    private String name;
}
```

Office Entity

```java
@Entity
@Table(name = "office")
public class Office {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer officeId;

    private String officeName;

    @ManyToOne
    @JoinColumn(name = "area")
    private Area area;

    @ManyToOne
    @JoinColumn(name = "department")
    private Department department;
}
```

User Entity (Security-ready)

```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer userId;

    @Column(unique = true)
    private String username;

    private String password;

    @ManyToOne
    @JoinColumn(name = "role_id")
    private UserRole role;

    @ManyToOne
    @JoinColumn(name = "job_position")
    private JobPosition jobPosition;

    @ManyToOne
    @JoinColumn(name = "office_id")
    private Office office;

    private LocalDateTime lastLogin;
    private LocalDateTime dateCreated;
    private String keywords;
}
```

---

4. REPOSITORIES

```
public interface AreaRepository extends JpaRepository<Area, Integer> {}
public interface OfficeRepository extends JpaRepository<Office, Integer> {}
public interface UserRepository extends JpaRepository<User, Integer> {
    Optional<User> findByUsername(String username);
}
```

---

5. GENERIC ADMIN CRUD CONTROLLER PATTERN

Example: Area Admin Controller

```
@Controller
@RequestMapping("/admin/areas")
@PreAuthorize("hasRole('ADMIN')")
public class AreaAdminController {

    private final AreaRepository repo;

    public AreaAdminController(AreaRepository repo) {
        this.repo = repo;
    }

    @GetMapping
    public String list(Model model) {
        model.addAttribute("areas", repo.findAll());
        model.addAttribute("area", new Area());
        return "admin/area/list";
    }

    @PostMapping("/save")
    public String save(@ModelAttribute Area area) {
        repo.save(area);
        return "redirect:/admin/areas";
    }

    @GetMapping("/delete/{id}")
    public String delete(@PathVariable Integer id) {
        repo.deleteById(id);
        return "redirect:/admin/areas";
    }
}
```

---

6. ADMIN UI – ENTERPRISE STYLE (MATCHES YOUR UPLOAD FORM)

Admin List + Modal Form (Area)

```
<div class="form-container">
    <div class="page-header">
        <h4><i class="fas fa-map me-2"></i>Area Management</h4>
    </div>
```

```
    <div class="page-body">
        <table class="table table-sm align-middle">
            <thead class="table-light">
                <tr>
                    <th>Name</th>
                    <th class="text-end">Action</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="a : ${areas}">
                    <td th:text="${a.name}"></td>
                    <td class="text-end">
                        <a th:href="@{'/admin/areas/delete/' + ${a.id}}"
                           class="btn btn-sm btn-outline-danger">Delete</a>
                    </td>
                </tr>
            </tbody>
        </table>

        <form th:action="@{/admin/areas/save}" method="post" class="mt-4">
            <label class="form-label">New Area</label>
            <input class="form-control" name="name" required>
            <button class="btn btn-primary mt-2">Save</button>
        </form>
    </div>
</div>
```

---

7. USERS CRUD (ADMIN)

Admin Can:

- Create user
- Assign role
- Assign office
- Reset password
- Disable user

Passwords **must be BCrypt encoded**.

```
user.setPassword(passwordEncoder.encode(rawPassword));
```

---

8. OFFICE VIEWER MAPPING CRUD

Purpose: Map **viewer group → offices**

```
@Entity
public class OfficeViewer {
    @Id
    @GeneratedValue
    private Integer officeViewerId;

    @ManyToOne
    private IntendedViewerGroup intendedViewerGroup;

    @ManyToOne
    private Office office;
}
```

Admin UI uses **multi-select** offices per group.

---

9. DOCUMENT ADMIN (METADATA ONLY)

Admin may: - View uploaded documents - Reassign viewer groups - Change classification - Disable document

❌Admin does NOT re-upload files

---

10. SECURITY (IMPORTANT)

```
.antMatchers("/admin/**").hasRole("ADMIN")
```

Roles table drives authorization.

---

11. NEXT STEPS (I CAN DO FOR YOU)

✅Generate **ALL entities** fully ✅Generate **ALL controllers** automatically ✅Generate **admin sidebar & dashboard** ✅Convert this into **generic CRUD generator** ✅Add **DataTables + AJAX**

Tell me next:

A) Spring Boot version? **B)** Thymeleaf only or REST + JS? **C)** Do you want inline edit or modal forms?