**THỰC HÀNH BUỔI 3**

# Mục tiêu

* Lý thuyết Business Analysis (BA): Quy trình phân tích yêu cầu, user story mapping, chuyển đổi sang API design
* Phân tích TODO App: Stakeholder analysis, use cases, user stories, data requirements, API endpoints design
* Thiết kế cơ sở dữ liệu đơn giản (ERD)
* Thiết kế API endpoints theo chuẩn RESTful

# Yêu cầu chuẩn bị trước

* Hoàn thành Buổi 1-2: Hiểu về mã nguồn mở và Git
* Kiến thức cơ bản về HTTP methods (GET, POST, PUT, DELETE)
* Hiểu khái niệm cơ bản về cơ sở dữ liệu quan hệ
* Có công cụ vẽ sơ đồ ([draw.io](http://draw.io), Lucidchart, hoặc giấy bút)

# Lý thuyết

**Mô hình phát triển phần mềm theo Agile/SCRUM**

A diagram of a sprint

AI-generated content may be incorrect.

## Business Analysis trong Phát triển Phần mềm

### Business Analysis là gì?

* Quá trình hiểu và phân tích nhu cầu kinh doanh để chuyển đổi thành yêu cầu kỹ thuật
* Bridge giữa stakeholders và development team
* Đảm bảo xây dựng đúng sản phẩm mà người dùng cần

### Quy trình Business Analysis:

* Stakeholder Analysis: Xác định ai sẽ sử dụng hệ thống
* Problem Definition: Định nghĩa vấn đề cần giải quyết
* Requirements Gathering: Thu thập yêu cầu chức năng và phi chức năng
* Use Case Development: Mô tả các tình huống sử dụng
* User Story Writing: Viết câu chuyện người dùng
* Technical Specification: Chuyển đổi thành yêu cầu kỹ thuật

A group of blue icons

AI-generated content may be incorrect.

### Kanban Workflow Analysis

**Kanban Principles:**

* **Visual Management**: Thể hiện work items trong columns
* **Work In Progress Limits**: Hạn chế số lượng tasks đang làm
* **Flow Optimization**: Tối ưu hóa throughput và cycle time
* **Continuous Improvement**: Kaizen methodology



# Hướng dẫn thực hành

## Phân tích actors

Các tác nhân (Actors):

• User: Người dùng thông thường có thể quản lý tasks của mình

• Admin: Quản trị viên có thể quản lý users và toàn bộ hệ thống

Yêu cầu chức năng:

1. Quản lý **người dùng**: Đăng ký, đăng nhập, quản lý profile

2. Quản lý **Board**: Tạo, xem, sửa, xóa boards (workspace)

3. Quản lý **Task**: CRUD tasks, di chuyển giữa các trạng thái, sắp xếp thứ tự

## Chiến lược thiết kế RESTful API

* **Resource-based URLs**: **/users, /boards, /tasks**
* **HTTP Methods**: GET (đọc), POST (tạo), PUT (cập nhật toàn bộ), PATCH (cập nhật một phần), DELETE (xóa)
* **Status Codes**: 200 (OK), 201 (Created), 400 (Bad Request), 401 (Unauthorized), 404 (Not Found)
* **Consistent Response Format**: JSON với cấu trúc thống nhất

## Xác thực người dùng

### Các phương pháp xác thực phổ biến

* **Mật khẩu (Password-Based Authentication):** Người dùng nhập tên đăng nhập và mật khẩu để truy cập ứng dụng; cách này đơn giản nhưng dễ bị tấn công nếu mật khẩu yếu hoặc bị lộ.
* **Xác thực bằng mã OTP (One-Time Password):** Sử dụng mã xác thực gửi qua email, SMS hoặc ứng dụng như Google Authenticator, thường dùng kết hợp với mật khẩu để tăng bảo mật (2FA).
* **Xác thực dựa trên token:** Sử dụng mã xác thực (token) như JWT cho các ứng dụng web/API, giúp giảm phụ thuộc vào phiên server và thuận tiện cho các hệ thống phân tán.
* **Xác thực sinh trắc học:** Nhận diện vân tay, khuôn mặt hoặc giọng nói, rất tiện lợi nhưng đòi hỏi phần cứng hỗ trợ và ít ứng dụng rộng rãi trên web thông thường.
* **Xác thực đa yếu tố (MFA):** Kết hợp nhiều phương pháp như mật khẩu, OTP, token hoặc sinh trắc học để tăng độ bảo mật; thường dùng trong dịch vụ tài chính hoặc bảo mật cao.
* **Đăng nhập một lần (SSO):** Cho phép người dùng đăng nhập một lần và truy cập nhiều dịch vụ nhờ giao thức như OAuth, SAML, OpenID Connect; rất tiện lợi với doanh nghiệp hoặc hệ sinh thái ứng dụng lớn.
* **Xác thực HTTP (Basic/Digest):** Gửi thông tin xác thực qua header HTTP, phù hợp cho API hoặc ứng dụng nhỏ, nhưng kém bảo mật hơn các phương pháp hiện đại.

### JWT là gì?

*Xem thêm tại:* [*https://www.jwt.io/introduction#what-is-json-web-token*](https://www.jwt.io/introduction#what-is-json-web-token)

JWT về mặt kỹ thuật là một cơ chế để xác minh chính chủ với dữ liệu mã hóa dạng JSON. Nó là một chuỗi biến đổi, có thể chứa một lượng dữ liệu không giới hạn (không giống như một cookie) và nó đã được mã hóa bằng chữ ký.

JWT là một công nghệ tuyệt vời để xác thực API và ủy quyền từ máy chủ đến máy chủ.

Nó không phải là một lựa chọn tốt cho các session.

Ví dụ về 1 JWT Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjEzODY4OTkxMzEsImlzcyI6ImppcmE6MTU0ODk1OTUiLCJxc2giOiI4MDYzZmY0Y2ExZTQxZGY3YmM5MGM4YWI2ZDBmNjIwN2Q0OTFjZjZkYWQ3YzY2ZWE3OTdiNDYxNGI3MTkyMmU5IiwiaWF0IjoxMzg2ODk4OTUxfQ.uKqU9dTB6gKwG6jQCuXYAiMNdfNRw98Hw\_IWuA5MaMo

Trông phức tạp là thế nhưng nếu hiểu, cấu trúc của một JWT chỉ đơn giản như sau:

**<base64-encoded header>.<base64-encoded payload>.<base64-encoded signature>**

**Header**

Header bao gồm hai phần chính: loại token (mặc định là JWT - Thông tin này cho biết đây là một Token JWT) và thuật toán đã dùng để mã hóa (HMAC SHA256 - HS256 hoặc RSA).

**Payload**

Payload chứa các claims. Claims là một các biểu thức về một thực thể (chẳng hạn user) và một số metadata phụ trợ. Có 3 loại claims thường gặp trong Payload: **registered**, **public** và **private** claims.

* Registered claims: Đây là một tập hợp các claims được xác định trước không bắt buộc nhưng được khuyến nghị. Ví dụ: iss (issuer), exp (expiration time), sub (subject), aud (audience)...
* Public claims: Được định nghĩa bởi người dùng. Nhưng để tránh xung đột, chúng phải được xác định trong I[ANA JSON Web Token Registry](https://www.iana.org/assignments/jwt/jwt.xhtml) hoặc được định nghĩa là URI chứa namespace chống xung đột.
* Private claims: Đây là các custom clams được tạo để trao đổi thông tin giữa các bên và thỏa thuận sử dụng chúng

**Signature**

Chữ ký Signature trong JWT là một chuỗi được mã hóa bởi header, payload cùng với một chuỗi bí mậ, do bản thân Signature đã bao gồm cả header và payload nên Signature có thể dùng để kiểm tra tính toàn vẹn của dữ liệu khi truyền tải.

HMACSHA256(

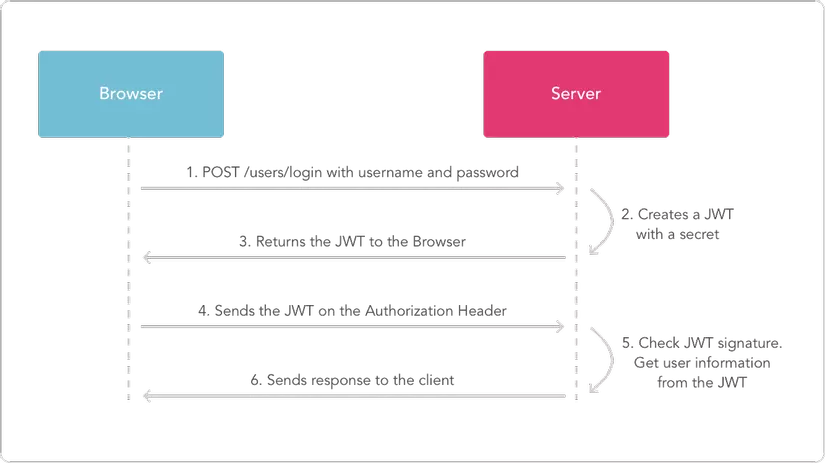
base64UrlEncode(header) + "." +

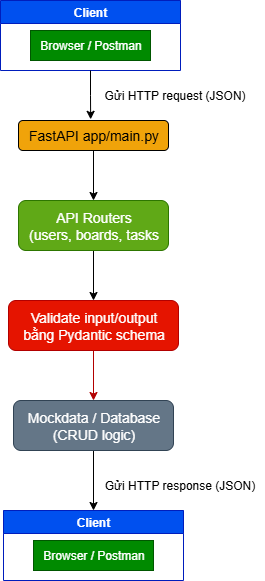
base64UrlEncode(payload),

secret)

Trong xác thực API khi user login server sẽ trả về cho client một JWT. Ở các request sau để có thể truy cập vào tài nguyên người dùng đó thì cần gửi token này lên. Token này sẽ được gửi trong header với key là Authorization và value có dạng **Bearer <token>**.

Sơ đồ dưới đây cho thấy hoạt động xác thực API bằng JWT

****

****

# Các bước chuẩn bị và xây dựng backend của TO app

### Bước 1: Phân tích nghiệp vụ, xác định yêu cầu

* Hiểu rõ project cần quản lý cái gì (nghiệp vụ: User, Board, Task).
* Xác định các tính năng cần thiết (CRUD với từng đối tượng, đăng nhập, phân quyền...).

### Bước 2. Thiết kế API endpoints

* Lên danh sách các endpoint API cần có (ví dụ: /auth/register, /boards, /tasks).
* Định nghĩa cấu trúc request (body, query param, headers).
* Định nghĩa response data (JSON format trả về).

### Bước 3. Thiết kế cơ sở dữ liệu (ERD)

* Xác định các bảng dữ liệu: User, Board, Task.
* Thiết lập các mối quan hệ (Một User có nhiều Board, Một Board có nhiều Task,...).
* Xác định các trường dữ liệu, quy tắc (unique, foreign key, required).

### Bước 4. Chọn công cụ/back-end stack

* Chọn framework (ví dụ: FastAPI cho Python).
* Chọn hệ quản trị database (SQLite, PostgreSQL).
* Chọn công cụ migration (Alembic).

### Bước 5. Thiết lập project & môi trường phát triển

* Tạo cấu trúc thư mục chuẩn (project folder, các module routers, models, schemas).
* Khởi tạo và kích hoạt virtual environment.
* Cài đặt các thư viện cần thiết (fastapi, uvicorn, sqlalchemy, alembic, passlib, python-jose...)

### 6. Thiết kế Pydantic models (validation)

* Xây dựng các Pydantic models cho User, Board, Task để validate input/output.

### 7. Khởi tạo data models (mock hoặc SQLAlchemy)

* Bước đầu có thể sử dụng mock data để test API.
* Khi chuyển sang database thì dùng SQLAlchemy để định nghĩa models.

### 8. Tạo các router/endpoint & logic xử lý

* Tạo các route cho từng module (auth.py cho xác thực, boards.py, tasks.py,...).
* Viết logic xử lý từng API (CRUD, phân quyền, kiểm tra input...).

### 9. Thiết lập authentication và authorization

* Tạo các endpoint register/login, sinh JWT token.
* Sử dụng middleware/dependency để kiểm tra token khi truy cập các route bảo vệ.

### 10. Test API bằng Swagger UI và Postman

* Kiểm tra từng endpoint.
* Xác thực các flow: đăng nhập, tạo task, phân quyền...

### 11. Tối ưu hóa (error handling, security, logging)

* Bổ sung error response chuẩn, validate dữ liệu kỹ càng.
* Hash password, bảo vệ dữ liệu nhạy cảm.
* Thêm logging nếu cần kiểm soát debug.

## Phân tích nghiệp vụ chi tiết

### Xác định các đối tượng nghiệp vụ (Entities)

**Entity 1: User (Người dùng)**

**Thuộc tính**:

id: Định danh duy nhất

username: Tên đăng nhập (duy nhất)

email: Email (duy nhất, có thể null)

password\_hash: Mật khẩu đã mã hóa

full\_name: Họ tên đầy đủ

role: Vai trò (user/admin)

is\_active: Trạng thái tài khoản

created\_at: Thời gian tạo

updated\_at: Thời gian cập nhật cuối

**Nghiệp vụ**:

* Đăng ký tài khoản mới
* Đăng nhập với username/password
* Xem và cập nhật thông tin cá nhân
* Đổi mật khẩu

**Entity 2: Board (Bảng công việc)**

**Thuộc tính**:

id: Định danh duy nhất

name: Tên board

description: Mô tả board

owner\_id: ID của user sở hữu

is\_public: Board công khai hay riêng tư

created\_at: Thời gian tạo

updated\_at: Thời gian cập nhật cuối

**Nghiệp vụ**:

* Tạo board mới
* Xem danh sách boards của user
* Cập nhật thông tin board
* Xóa board (kèm theo tất cả tasks)
* Chia sẻ board với users khác (advanced)

**Entity 3: Task (Công việc)**

**Thuộc tính**:

id: Định danh duy nhất

title: Tiêu đề task

description: Mô tả chi tiết

status: Trạng thái (todo/in\_progress/done)

priority: Độ ưu tiên (low/medium/high)

position: Vị trí sắp xếp trong cùng status

board\_id: ID của board chứa task

assigned\_to: ID user được giao task (có thể null)

due\_date: Ngày hết hạn (có thể null)

created\_at: Thời gian tạo

updated\_at: Thời gian cập nhật cuối

**Nghiệp vụ**:

* Tạo task mới trong board
* Cập nhật thông tin task
* Di chuyển task giữa các trạng thái
* Sắp xếp lại thứ tự tasks
* Gán task cho user
* Xóa task

### Entity Relationship Diagram (ERD)

A diagram of a user and task

AI-generated content may be incorrect.

## Thiết kế API Endpoints

# User Management

POST /auth/register # Đăng ký

POST /auth/login # Đăng nhập

GET /users/me # Xem profile

PUT /users/me # Cập nhật profile

# Board Management

GET /boards # Lấy danh sách boards

POST /boards # Tạo board mới

GET /boards/{id} # Lấy chi tiết board

PUT /boards/{id} # Cập nhật board

DELETE /boards/{id} # Xóa board

# Task Management

GET /tasks?board\_id={id} # Lấy tasks theo board

POST /tasks # Tạo task mới

GET /tasks/{id} # Lấy chi tiết task

PUT /tasks/{id} # Cập nhật task

DELETE /tasks/{id} # Xóa task

PATCH /tasks/{id}/move # Di chuyển task (status/position)

### Authentication Endpoints

**1. Đăng ký tài khoản mới**

POST /auth/register

**Request Body**:

{  
 "username": "johndoe",  
 "email": "john@example.com",  
 "password": "securepassword123",  
 "full\_name": "John Doe"  
}

**Response Success (201)**:

{  
 "id": 1,  
 "username": "johndoe",  
 "email": "john@example.com",  
 "full\_name": "John Doe",  
 "role": "user",  
 "created\_at": "2025-01-15T10:30:00Z"  
}

**2. Đăng nhập và nhận token**

POST /auth/login

**Request Body**:

{  
 "username": "johndoe",  
 "password": "securepassword123"  
}

**Response Success (200)**:

{  
 "access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
 "token\_type": "bearer",  
 "expires\_in": 3600,  
 "user": {  
 "id": 1,  
 "username": "johndoe",  
 "full\_name": "John Doe",  
 "role": "user"  
 }  
}

### User Management Endpoints

**1. Lấy thông tin user hiện tại**

GET /users/me

**Headers**: **Authorization: Bearer {token}**  
**Response Success (200)**:

{  
 "id": 1,  
 "username": "johndoe",  
 "email": "john@example.com",  
 "full\_name": "John Doe",  
 "role": "user",  
 "is\_active": true,  
 "created\_at": "2025-01-15T10:30:00Z",  
 "updated\_at": "2025-01-15T10:30:00Z"  
}

**2. Cập nhật thông tin user hiện tại**

PUT /users/me

**Headers**: **Authorization: Bearer {token}**  
**Request Body**:

{  
 "email": "newemail@example.com",  
 "full\_name": "John Smith"  
}

**3. Đổi mật khẩu**

PATCH /users/me/password

**Headers**: **Authorization: Bearer {token}**

**Request Body**:

{  
 "current\_password": "oldpassword",  
 "new\_password": "newpassword123"  
}

### Board Management Endpoints

**1. Lấy danh sách boards của user**

GET /boards

**Headers**: **Authorization: Bearer {token}**

**Query Parameters**:

* page (optional): Trang hiện tại (default: 1)
* limit (optional): Số lượng per page (default: 10)  
  **Response Success (200)**:

{  
 "boards": [  
 {  
 "id": 1,  
 "name": "Personal Tasks",  
 "description": "My personal task management",  
 "owner\_id": 1,  
 "is\_public": false,  
 "created\_at": "2025-01-15T11:00:00Z",  
 "updated\_at": "2025-01-15T11:00:00Z",  
 "tasks\_count": 5  
 }  
 ],  
 "total": 1,  
 "page": 1,  
 "limit": 10  
}

**2. Tạo board mới**

POST /boards

**Headers**: **Authorization: Bearer {token}**

**Request Body**:

{  
 "name": "Work Project",  
 "description": "Tasks for the new project",  
 "is\_public": false  
}

**3. Lấy chi tiết board cùng tasks**

GET /boards/{board\_id}

**Headers**: **Authorization: Bearer {token}**

**Response Success (200)**:

{  
 "id": 1,  
 "name": "Personal Tasks",  
 "description": "My personal task management",  
 "owner\_id": 1,  
 "is\_public": false,  
 "created\_at": "2025-01-15T11:00:00Z",  
 "updated\_at": "2025-01-15T11:00:00Z",  
 "tasks": [  
 {  
 "id": 1,  
 "title": "Setup development environment",  
 "status": "todo",  
 "priority": "high",  
 "position": 0  
 }  
 ]  
}

**4. Cập nhật board**

PUT /boards/{board\_id}

**Headers**: **Authorization: Bearer {token}**

**Request Body (JSON):**

{

"name": "Tên board mới",

"description": "Mô tả cập nhật (không bắt buộc)",

"is\_public": true

}

**Response thành công (200 OK):**

{

"id": 1,

"name": "Tên board mới",

"description": "Mô tả cập nhật (nếu có)",

"owner\_id": 123,

"is\_public": true,

"created\_at": "2025-01-15T11:00:00Z",

"updated\_at": "2025-01-16T09:30:00Z"

}

**5. Xóa board và tất cả tasks**

DELETE /boards/{board\_id}

**Headers**: **Authorization: Bearer {token}**

**Response Success (200)**:

{  
 "message": "Board deleted successfully",  
 "deleted\_tasks\_count": 3  
}

### Task Management Endpoints

**1. Lấy danh sách tasks**

GET /tasks

Headers: **Authorization: Bearer {token}**

Query Parameters:

* board\_id (required): ID của board
* status (optional): Lọc theo trạng thái
* priority (optional): Lọc theo độ ưu tiên
* assigned\_to (optional): Lọc theo user được gán

Response Success (200):

{  
 "tasks": [  
 {  
 "id": 1,  
 "title": "Setup development environment",  
 "description": "Install Python, FastAPI, and required tools",  
 "status": "todo",  
 "priority": "high",  
 "position": 0,  
 "board\_id": 1,  
 "assigned\_to": 1,  
 "due\_date": "2025-01-20T00:00:00Z",  
 "created\_at": "2025-01-15T12:00:00Z",  
 "updated\_at": "2025-01-15T12:00:00Z"  
 }  
 ]  
}

**2. Tạo task mới**

POST /tasks

Headers: **Authorization: Bearer {token}**

Request Body:

{  
 "title": "New Task",  
 "description": "Task description",  
 "board\_id": 1,  
 "priority": "medium",  
 "assigned\_to": null,  
 "due\_date": "2025-01-25T00:00:00Z"  
}

**3. Lấy chi tiết task**

GET /tasks/{task\_id}

Headers**: Authorization: Bearer {token}**

Request Parameters:

* task\_id (int, path parameter): ID của task cần lấy chi tiết.

**Response thành công (200 OK):**

{

"id": 1,

"title": "Setup development environment",

"description": "Install Python, FastAPI, and required tools",

"status": "todo",

"priority": "high",

"position": 0,

"board\_id": 1,

"assigned\_to": 2,

"due\_date": "2025-01-20T00:00:00Z",

"created\_at": "2025-01-15T12:00:00Z",

"updated\_at": "2025-01-15T12:30:00Z"

}

**4. Cập nhật task**

PUT /tasks/{task\_id}

**Request Body**:

{  
 "title": "Updated Task Title",  
 "description": "Updated description",  
 "priority": "high",  
 "due\_date": "2025-01-30T00:00:00Z"  
}

**5. Di chuyển task (thay đổi status và/hoặc position)**

PATCH /tasks/{task\_id}/move

**Mô tả**: **Request Body**:

{  
 "status": "in\_progress",  
 "position": 2  
}

**6. Gán task cho user**

PATCH /tasks/{task\_id}/assign

**Request Body**:

{  
 "assigned\_to": 2  
}

**7. Xóa task**

DELETE /tasks/{task\_id}

Headers**: Authorization: Bearer {token}**

**Request Parameters:**

* task\_id (int, path parameter): ID của task cần xóa.

**Response thành công (200 OK):**

{

"message": "Task deleted successfully",

"deleted\_task\_id": 1

}

### Advanced Endpoints (Optional)

Tìm kiếm tasks theo keyword

**GET /tasks/search**  
**Query Parameters**:

* q: Từ khóa tìm kiếm
* board\_id (optional): Tìm trong board cụ thể

**GET /boards/{board\_id}/stats**

**Mô tả**: Thống kê tasks trong board  
**Response**:

{  
 "total\_tasks": 10,  
 "todo": 3,  
 "in\_progress": 4,  
 "done": 3,  
 "overdue\_tasks": 1,  
 "high\_priority\_tasks": 2  
}

**6. Response Format Standards**

**6.1 Success Response Format**

{  
 "data": { ... }, // Dữ liệu trả về  
 "message": "Success", // Thông báo (optional)  
 "timestamp": "2025-01-15T12:00:00Z"  
}

**6.2 Error Response Format**

{  
 "error": {  
 "code": "VALIDATION\_ERROR",  
 "message": "Invalid input data",  
 "details": [  
 {  
 "field": "username",  
 "message": "Username already exists"  
 }  
 ]  
 },  
 "timestamp": "2025-01-15T12:00:00Z"  
}

**6.3 Pagination Format**

{  
 "data": [...],  
 "pagination": {  
 "total": 100,  
 "page": 1,  
 "limit": 10,  
 "total\_pages": 10,  
 "has\_next": true,  
 "has\_prev": false  
 }  
}

**7. Status Codes Reference**

|  |  |  |
| --- | --- | --- |
| Code | Meaning | Usage |
| 200 | OK | GET, PUT, PATCH thành công |
| 201 | Created | POST tạo resource thành công |
| 204 | No Content | DELETE thành công |
| 400 | Bad Request | Dữ liệu đầu vào không hợp lệ |
| 401 | Unauthorized | Chưa đăng nhập hoặc token không hợp lệ |
| 403 | Forbidden | Không có quyền truy cập |
| 404 | Not Found | Resource không tồn tại |
| 409 | Conflict | Xung đột dữ liệu (username đã tồn tại) |
| 422 | Unprocessable Entity | Validation error |
| 500 | Internal Server Error | Lỗi server |

**8. Authentication & Authorization**

**8.1 JWT Token Structure**

Header: {  
 "alg": "HS256",  
 "typ": "JWT"  
}  
  
Payload: {  
 "sub": "1", // User ID  
 "username": "johndoe",  
 "role": "user",  
 "exp": 1642694400, // Expiration timestamp  
 "iat": 1642690800 // Issued at timestamp  
}

**8.2 Protected Endpoints**

* Tất cả endpoints (trừ /auth/register, /auth/login) đều yêu cầu authentication
* Header format: Authorization: Bearer {jwt\_token}
* Token có thời hạn (ví dụ: 24 giờ)

**8.3 Role-Based Access Control**

* **User**: Chỉ truy cập boards/tasks của mình
* **Admin**: Truy cập tất cả resources + user management

**9. Validation Rules**

**9.1 User Validation**

* username: 3-20 ký tự, chỉ chứa letters, numbers, underscore
* email: Format email hợp lệ (optional)
* password: Tối thiểu 8 ký tự, có chữ và số
* full\_name: Tối đa 100 ký tự

**9.2 Board Validation**

* name: 1-100 ký tự, không được empty
* description: Tối đa 500 ký tự (optional)

**9.3 Task Validation**

* title: 1-200 ký tự, không được empty
* description: Tối đa 1000 ký tự (optional)
* status: Chỉ nhận "todo", "in\_progress", "done"
* priority: Chỉ nhận "low", "medium", "high"
* position: Số nguyên >= 0

BONUS: Git workflow – Push project lên trên Github

Bước 1: Mở Terminal

Bước 2: Gõ lần lượt các lệnh bên dưới:

### # Add and commit

### git add .

### git commit -m "feat: Business Analysis & OOP Implementation - Session 3

### - Complete business analysis with stakeholders, use cases, user stories

### - Implement Task and TaskManager classes with full CRUD operations

### - Add comprehensive filtering and statistics functionality

### - Map requirements to future API endpoints design"

# Bài tập vận dụng

**Bài tập cơ bản**

1. **Vẽ ERD chi tiết**: Bổ sung thêm các thuộc tính, ràng buộc, indexes
2. **Thiết kế API spec**: Sử dụng OpenAPI/Swagger format
3. **Tạo mock responses**: Chuẩn bị sample data cho tất cả endpoints
4. **Viết validation rules**: Chi tiết cho từng field của mỗi entity

**Bài tập nâng cao**

1. **Mở rộng tính năng**:
   * Comments trên tasks
   * File attachments
   * Task dependencies (task A phải hoàn thành trước task B)
   * Team collaboration (nhiều users cùng board)
2. **Thiết kế API cho tính năng mới**:
   * GET /tasks/{id}/comments
   * POST /tasks/{id}/comments
   * POST /tasks/{id}/attachments
   * GET /boards/{id}/members
   * POST /boards/{id}/members
3. **Security considerations**:
   * Rate limiting endpoints
   * Input sanitization
   * CORS configuration
   * API versioning strategy