Tài liệu Thiết kế Kỹ thuật: RESTful API Login

# 1. Giới thiệu

Tài liệu này trình bày chi tiết thiết kế kỹ thuật cho tính năng đăng nhập (Login) sử dụng RESTful API của hệ thống. Mục tiêu của tính năng là:  
- Xác thực người dùng với quy trình bảo mật cao.  
- Hỗ trợ đăng nhập cho Admin: sử dụng email và password.  
- Hỗ trợ đăng nhập cho Customer: sử dụng email, password và tích hợp đăng nhập qua Google, Facebook.

# 2. Phạm vi

Tính năng đăng nhập này áp dụng cho:  
- Admin: Nhân viên quản trị hệ thống.  
- Customer: Người dùng cuối của hệ thống.

# 3. Yêu cầu chức năng

## 3.1 Yêu cầu chung

Hệ thống phải xác thực và phân quyền người dùng dựa trên thông tin đăng nhập. Sinh ra token (ví dụ: JWT) sau khi đăng nhập thành công để thực hiện các giao tiếp sau này. Hỗ trợ xác thực OAuth cho các nhà cung cấp Google và Facebook. Các dữ liệu nhạy cảm (như mật khẩu) phải được mã hóa theo tiêu chuẩn bảo mật.

## 3.2 Yêu cầu đối với Admin

- Admin đăng nhập bằng email và password.  
- Kiểm tra tính hợp lệ của email và password.  
- Trả về thông tin admin cùng với token khi đăng nhập thành công.

## 3.3 Yêu cầu đối với Customer

- Đăng nhập bằng email và password cho tài khoản nội bộ.  
- Đăng nhập qua Google và Facebook:  
 - Xác thực token được cấp từ nhà cung cấp OAuth.  
 - Nếu người dùng chưa tồn tại, thực hiện đăng ký tự động.  
 - Trả về token và thông tin khách hàng khi đăng nhập thành công.

# 4. Kiến trúc hệ thống

## 4.1 Thành phần chính

- API Gateway: Tiếp nhận và định tuyến các yêu cầu từ client.  
- Authentication Service: Xử lý đăng nhập, xác thực và sinh token.  
- User Service: Quản lý dữ liệu người dùng (Admin và Customer).  
- OAuth Integration Module: Tích hợp với Google và Facebook để xác thực người dùng đăng nhập qua mạng xã hội.

## 4.2 Luồng xử lý

### 4.2.1 Đăng nhập Admin

1. Admin gửi yêu cầu POST đến `/api/admin/login` với email và password.  
2. API Gateway chuyển tiếp yêu cầu đến Authentication Service.  
3. Authentication Service xác thực thông tin với cơ sở dữ liệu.  
4. Nếu hợp lệ, sinh JWT token và trả về cho client.  
5. Nếu không hợp lệ, trả về lỗi 401 Unauthorized.

### 4.2.2 Đăng nhập Customer qua email/password

1. Customer gửi yêu cầu POST đến `/api/customer/login` với email và password.  
2. Authentication Service xác thực thông tin người dùng.  
3. Nếu hợp lệ, sinh JWT token và trả về cho client.  
4. Nếu không hợp lệ, trả về lỗi thích hợp.

### 4.2.3 Đăng nhập Customer qua Google/Facebook

1. Customer gửi yêu cầu POST đến `/api/customer/login` với các trường:  
 - provider: "google" hoặc "facebook"  
 - token: token OAuth được cấp từ nhà cung cấp.  
2. OAuth Integration Module xác thực token với Google hoặc Facebook.  
3. Nếu token hợp lệ:  
 - Nếu người dùng đã tồn tại: tiến hành đăng nhập và sinh token nội bộ.  
 - Nếu người dùng chưa tồn tại: thực hiện đăng ký người dùng, sau đó sinh token.  
4. Trả về token cùng thông tin người dùng.

# 5. Chi tiết API

## 5.1 API cho Admin

URL: `/api/admin/login`

Method: POST

Headers: `Content-Type: application/json`

Request Body:

{  
 "email": "admin@example.com",  
 "password": "password123"  
}

Success Response:  
Status Code: 200 OK  
Response Body:  
{  
 "accessToken": "jwt\_token\_here",  
 "refreshToken": "refresh\_token\_here",  
 "admin": {  
 "id": "admin\_id",  
 "email": "admin@example.com",  
 "name": "Admin Name"  
 }  
}

Error Responses:  
Status Code: 401 Unauthorized  
Response Body:  
{  
 "error": "Invalid credentials"  
}  
Status Code: 400 Bad Request (nếu thiếu thông tin bắt buộc)

## 5.2 API cho Customer

### 5.2.1 Đăng nhập qua Email/Password

URL: `/api/customer/login`

Method: POST

Headers: `Content-Type: application/json`

Request Body:

{  
 "email": "customer@example.com",  
 "password": "password123"  
}

Success Response:  
Status Code: 200 OK  
Response Body:  
{  
 "accessToken": "jwt\_token\_here",  
 "refreshToken": "refresh\_token\_here",  
 "customer": {  
 "id": "customer\_id",  
 "email": "customer@example.com",  
 "name": "Customer Name"  
 }  
}

Error Responses:  
Status Code: 401 Unauthorized  
Response Body:  
{  
 "error": "Invalid credentials"  
}  
Status Code: 400 Bad Request

### 5.2.2 Đăng nhập qua Google/Facebook

URL: `/api/customer/login`

Method: POST

Headers: `Content-Type: application/json`

Request Body (ví dụ với Google):

{  
 "provider": "google",  
 "token": "google\_oauth\_token\_here"  
}

Request Body (ví dụ với Facebook):

{  
 "provider": "facebook",  
 "token": "facebook\_oauth\_token\_here"  
}

Success Response:  
Status Code: 200 OK  
Response Body:  
{  
 "accessToken": "jwt\_token\_here",  
 "refreshToken": "refresh\_token\_here",  
 "customer": {  
 "id": "customer\_id",  
 "email": "customer@example.com",  
 "name": "Customer Name",  
 "provider": "google" // hoặc "facebook"  
 }  
}

Error Responses:  
Status Code: 401 Unauthorized  
Response Body:  
{  
 "error": "Invalid or expired OAuth token"  
}  
Status Code: 400 Bad Request

# 6. Mô hình dữ liệu

## 6.1 Bảng Admin

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| id | UUID | Mã định danh admin |
| email | String | Email đăng nhập |
| password | String | Mật khẩu đã mã hóa |
| name | String | Tên admin |
| created\_at | Timestamp | Thời gian tạo |
| updated\_at | Timestamp | Thời gian cập nhật |

## 6.2 Bảng Customer

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| id | UUID | Mã định danh khách hàng |
| email | String | Email khách hàng |
| password | String | Mật khẩu (có thể null nếu đăng nhập social) |
| name | String | Tên khách hàng |
| provider | String | Nguồn đăng nhập: local, google, facebook |
| created\_at | Timestamp | Thời gian tạo |
| updated\_at | Timestamp | Thời gian cập nhật |

# 7. Bảo mật và xác thực

- Mã hóa mật khẩu: Sử dụng bcrypt hoặc các thuật toán tương đương.  
- JWT Token: Dùng JSON Web Token (JWT) cho các phiên đăng nhập.  
- Xác thực OAuth: Tích hợp xác thực qua API của Google và Facebook để kiểm tra token.  
- HTTPS: Bắt buộc tất cả giao tiếp giữa client và server thông qua HTTPS.

# 8. Xử lý lỗi

Các lỗi được xử lý theo mã trạng thái HTTP:  
- 400 Bad Request: Yêu cầu không hợp lệ hoặc thiếu dữ liệu cần thiết.  
- 401 Unauthorized: Thông tin đăng nhập sai hoặc token không hợp lệ.  
- 500 Internal Server Error: Lỗi phía máy chủ.  
Mỗi lỗi phải trả về thông tin chi tiết dưới dạng JSON để client có thể xử lý.

# 9. Kiểm thử và đảm bảo chất lượng

- Unit Tests: Kiểm thử từng hàm xử lý đăng nhập, xác thực và sinh token.  
- Integration Tests: Kiểm thử toàn bộ luồng đăng nhập từ client đến server.  
- Security Tests: Thực hiện kiểm thử bảo mật như SQL Injection, XSS, CSRF, và brute force.

# 10. Ghi chú và mở rộng

- Hệ thống cần linh hoạt mở rộng cho các phương thức đăng nhập khác (ví dụ: Twitter, LinkedIn).  
- Định kỳ cập nhật các thuật toán mã hóa và quy trình bảo mật theo tiêu chuẩn mới.  
- Ghi log chi tiết các lần đăng nhập thất bại để hỗ trợ việc giám sát và xử lý vi phạm bảo mật.

# 11. Phụ lục

## 11.1 Quy trình đăng nhập qua OAuth

1. Client lấy token từ nhà cung cấp (Google/Facebook).  
2. Client gửi token này kèm theo thông tin `provider` đến API của hệ thống.  
3. Hệ thống xác thực token với API của Google hoặc Facebook.  
4. Nếu xác thực thành công:  
 - Kiểm tra xem người dùng đã tồn tại chưa.  
 - Nếu chưa, tạo mới tài khoản khách hàng.  
 - Sinh JWT token nội bộ và trả về cho client.  
5. Nếu xác thực không thành công, trả về lỗi 401 Unauthorized.

## 11.2 Ví dụ mã code (Pseudo-code)

def login\_admin(email, password):  
 admin = find\_admin\_by\_email(email)  
 if admin and verify\_password(password, admin.password):  
 token = generate\_jwt(admin.id)  
 return {"accessToken": token, "admin": admin}  
 else:  
 raise UnauthorizedError("Invalid credentials")  
  
def login\_customer(email=None, password=None, provider=None, oauth\_token=None):  
 if provider in ["google", "facebook"]:  
 user\_info = verify\_oauth\_token(provider, oauth\_token)  
 customer = find\_or\_create\_customer(user\_info, provider)  
 else:  
 customer = find\_customer\_by\_email(email)  
 if not customer or not verify\_password(password, customer.password):  
 raise UnauthorizedError("Invalid credentials")  
 token = generate\_jwt(customer.id)  
 return {"accessToken": token, "customer": customer}