CTT226 – ỨNG DỤNG PHÂN TÁN

Hệ thống Quản lý Bệnh viện

Đồ án nhóm Học kỳ III – 2024-2025

21HTTT – Ho Chi Minh City University of Science - VNUHCM

GV phụ trách: TS. Nguyễn Trường Sơn, ThS. Phạm Minh Tú

Nhóm UDPT-06

✦

Thành phố Hồ Chí Minh, Tháng 8 năm 2025

Bảng thông tin chi tiết nhóm

Mã nhóm: UDPT-06 **Tên nhóm:** UDPT-06 **Số lượng:** 3

MSSV	Họ tên	Email	Điện thoại
21127004	Trần Nguyễn An Phong	tnaphong21@clc.fitus.edu.vn	
21127135	Diệp Hữu Phúc	dhphuc21@clc.fitus.edu.vn	
21127456	Võ Cao Trí	vctri21@clc.fitus.edu.vn	

Bảng 1. Thông tin thành viên nhóm UDPT-06.

Mục lục

Contents

Yêu cầu của Đồ án/Bài tập

Loại bài tập: Lý thuyết – Thực hành – Đồ án – Bài tập.
Ngày bắt đầu: 2025-06-10. **Ngày kết thúc:** 2025-08-29.

1 Phân tích nghiệp vụ

- Xác định các tác nhân (actors): Bệnh nhân, bác sĩ, quản trị viên,....
- Xác định các trường hợp sử dụng (use cases): Đặt lịch khám, quản lý bệnh nhân, quản lý đơn thuốc, gửi thông báo,....
- Xây dựng sơ đồ Use Case: Giúp hình dung các chức năng hệ thống cần có.

2 Thiết kế kiến trúc hệ thống

Chọn mô hình microservices:

- Đề xuất chia nhỏ hệ thống theo chức năng (functional decomposition).
- Sử dụng RabbitMQ để giao tiếp bất đồng bộ giữa các dịch vụ, đảm bảo tính nhất quán và hiệu năng.

Các loại giao tiếp giữa dịch vụ:

- Đồng bộ: Khi cần phản hồi ngay (ví dụ: xác nhận đăng ký lịch khám).
- Bất đồng bộ: Khi không yêu cầu phản hồi tức thì (ví dụ: gửi thông báo nhắc lịch).

3 Thiết kế cơ sở dữ liệu

Đề xuất: Chia thành các cơ sở dữ liệu riêng cho từng dịch vụ để tránh phụ thuộc. Các bảng có thể cần:

- Bệnh nhân (Patient): ID, tên, tuổi, giới tính, số điện thoại.
- Lịch khám (Appointment): ID, mã bệnh nhân, bác sĩ, thời gian, trạng thái.
- Đơn thuốc (Prescription): ID, mã bệnh nhân, thuốc, liều lượng.

4 Gợi ý công nghệ

- Web API: Java Spring Boot.
- Giao tiếp bất đồng bộ: RabbitMQ. Giao tiếp đồng bộ: REST API.
- Cơ sở dữ liệu: MySQL/PostgreSQL – Dữ liệu quan hệ (bệnh nhân, lịch khám). MongoDB – Dữ liệu không quan hệ (log, thông báo).

5 Yêu cầu báo cáo và nộp bài

1. Báo cáo phân tích: Sơ đồ Use Case, sơ đồ tuần tự (Sequence Diagram). Mô tả nghiệp vụ và các chức năng chính.

2. Báo cáo thiết kế: Sơ đồ kiến trúc microservices. Cách sử dụng RabbitMQ để truyền thông điệp giữa các dịch vụ.
3. Mã nguồn: Đẩy lên GitHub với cấu trúc rõ ràng và README.md hướng dẫn.
4. Video demo: Giới thiệu tổng quan hệ thống (3-5 phút). Demo các chức năng chính.

1 Phân tích nghiệp vụ

1.1 Mô tả nghiệp vụ và Lược đồ use case

1. Quản lý bệnh nhân

- Đăng ký bệnh nhân mới: Điều dưỡng – Văn phòng tiếp nhận thông tin bệnh nhân mới, nhập liệu vào hệ thống, đồng thời hệ thống tự động tạo tài khoản đăng nhập cho bệnh nhân và hồ sơ bệnh án ban đầu.
- Tra cứu / Cập nhật hồ sơ bệnh án: Bác sĩ và Nhân viên y tế có thể tìm kiếm, xem, hoặc cập nhật thông tin bệnh án của bệnh nhân để phục vụ cho quá trình khám chữa bệnh. Nhưng Bệnh nhân chỉ có thể tự xem hồ sơ của mình.

2. Quản lý lịch khám

- Đặt lịch khám: Bệnh nhân hoặc điều dưỡng chọn bác sĩ và khung giờ trống, hệ thống kiểm tra tính hợp lệ và lưu lịch hẹn.
- Hủy lịch khám: Bác sĩ có thể hủy lịch nếu trước thời gian khám một khoảng hợp lệ; hệ thống cập nhật trạng thái và gửi thông báo hủy đến các bên liên quan.

3. Quản lý toa thuốc

- Kê thuốc: Sau khi khám, bác sĩ tạo toa thuốc, lựa chọn thuốc từ danh mục, hệ thống lưu lại và liên kết với hồ sơ bệnh án.
- Cập nhật toa thuốc: Dược sĩ hoặc bác sĩ có thể cập nhật trạng thái (ví dụ: đã phát thuốc) hoặc thay đổi thông tin toa thuốc.
- Xem toa thuốc: Bệnh nhân hoặc nhân viên y tế có thể xem chi tiết toa thuốc kèm danh sách thuốc và liều lượng.

4. Quản lý dịch vụ xét nghiệm

- Yêu cầu dịch vụ: Trong buổi khám, bác sĩ có thể yêu cầu bệnh nhân thực hiện các dịch vụ xét nghiệm hoặc chẩn đoán hình ảnh; thông báo yêu cầu dịch vụ sẽ được gửi đến nhân viên kỹ thuật liên quan.

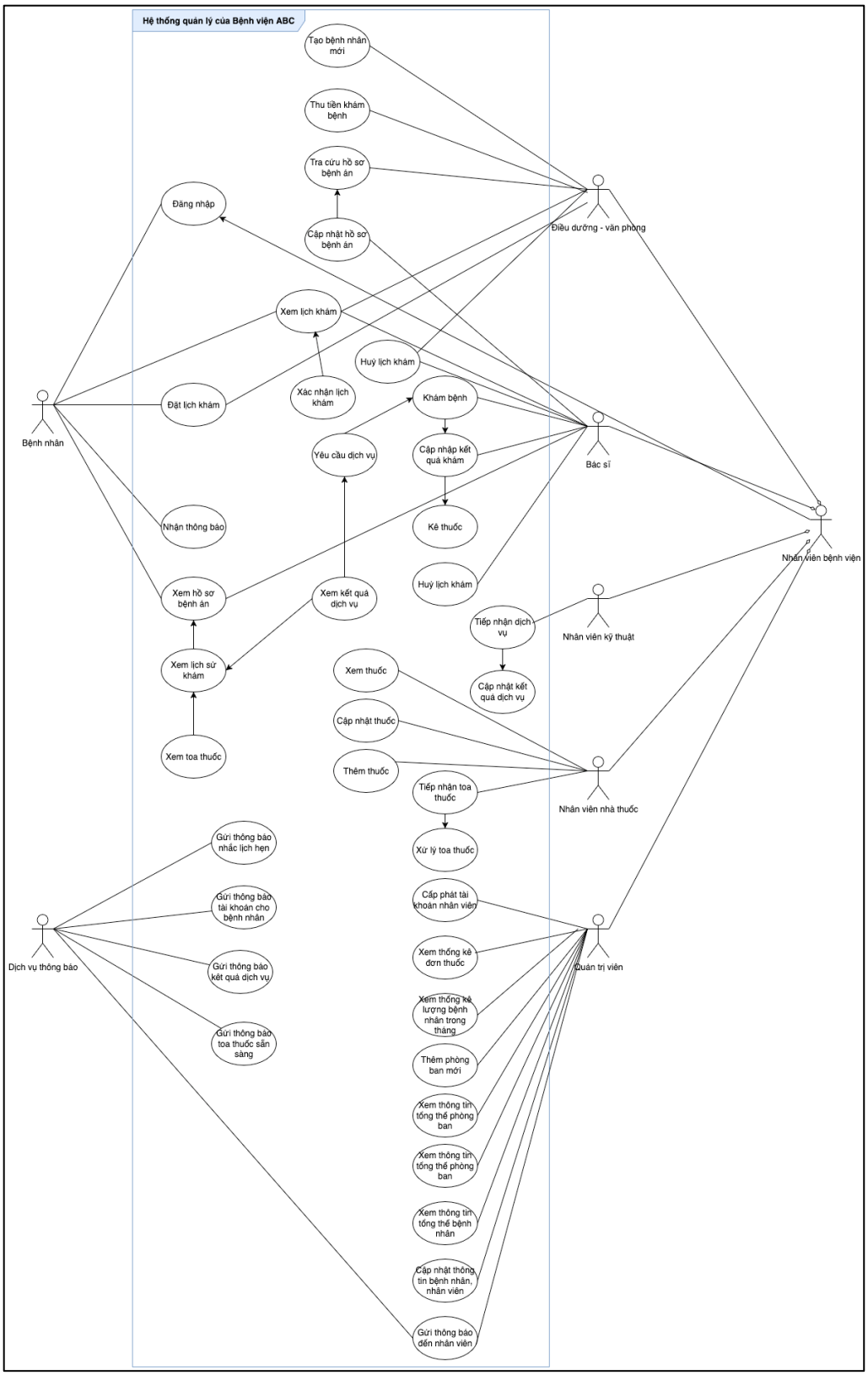
- Cập nhật kết quả dịch vụ: Nhân viên kỹ thuật có thể cập nhật thông tin, trạng thái của dịch vụ đã yêu cầu sau khi nhận được kết quả.
- Xem kết quả dịch vụ: Khi kết quả có sẵn, bác sĩ hoặc bệnh nhân có thể xem trực tiếp trong hệ thống; kết quả có thể bao gồm tệp đính kèm như hình ảnh, PDF báo cáo.

5. Thông báo

- Nhắc lịch khám: Hệ thống có thể gửi thông báo nhắc lịch hẹn đến bệnh nhân qua email, SMS, hoặc push notification.
- Thông báo kết quả: Khi đơn thuốc hoặc kết quả xét nghiệm đã sẵn sàng, hệ thống có thể tự động gửi thông báo cho bệnh nhân.

6. Báo cáo và thống kê

- Báo cáo đơn thuốc theo năm: Quản trị viên có thể xem thống kê số lượng đơn thuốc theo từng tháng trong năm.
- Thống kế số lượng bệnh nhân: Quản trị viên có thể lấy toàn bộ hồ sơ bệnh án để đưa ra các phân tích về số lượng bệnh nhân và dịch vụ theo tháng và năm.



Hình 1. Lược đồ Use case nghiệp vụ cho Hệ thống quản lý của Bệnh viện ABC.

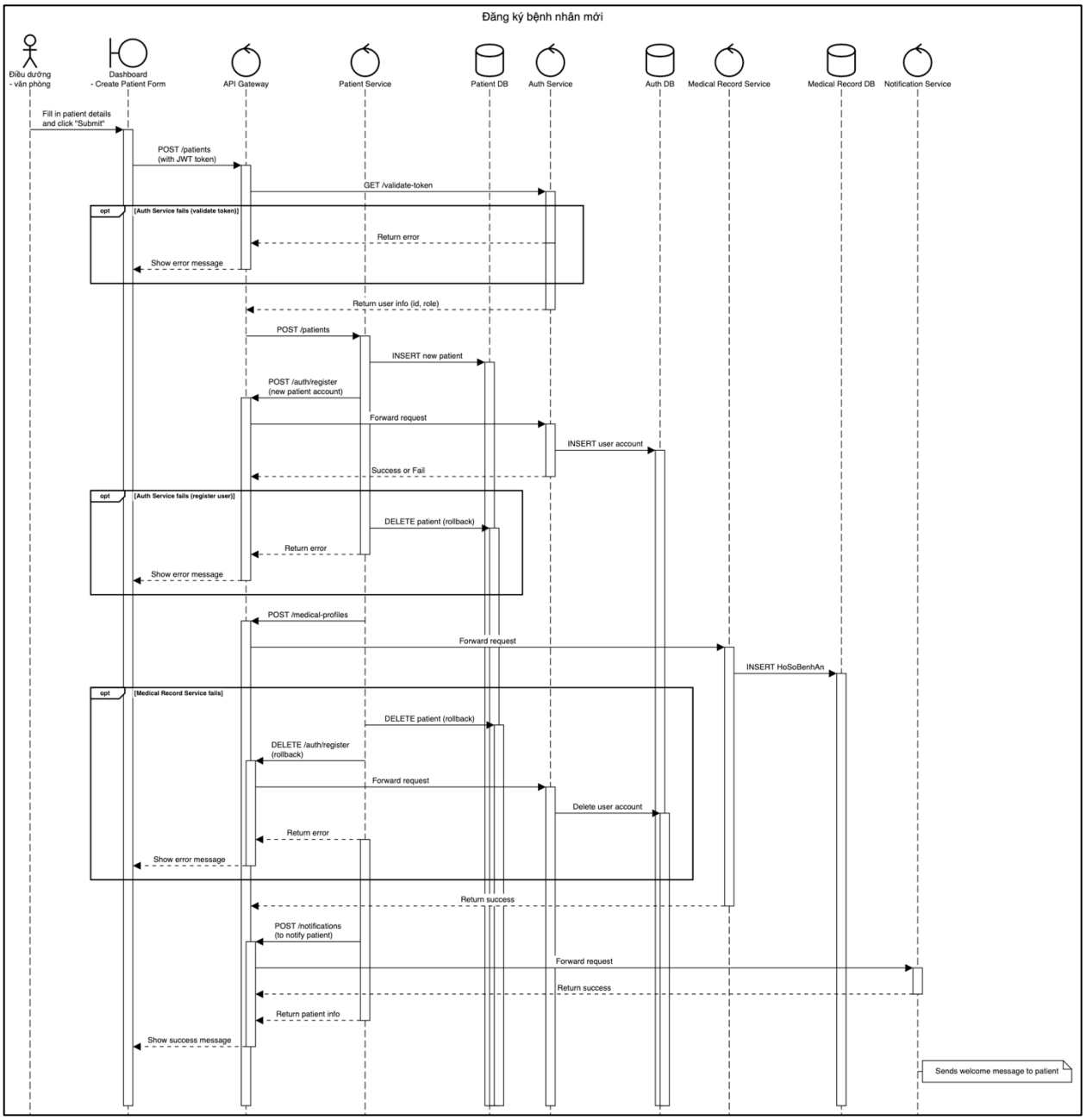
1.2 Sequence diagrams

Để mô tả chi tiết luồng (flow) tương tác giữa các tác nhân và hệ thống, nhóm đã xây dựng tập hợp sơ đồ tuần tự (sequence diagram) đại diện cho các nhóm chức năng chính. Thay vì vẽ một diagram riêng cho từng use case, các sơ đồ được chọn lọc và gộp theo nguyên tắc:

- Đại diện – Mỗi sơ đồ thể hiện một hoặc nhiều use case tiêu biểu cho từng phân hệ chính của hệ thống.
- Tránh trùng lắp – Các use case có luồng xử lý gần giống được gộp chung thành sơ đồ với các nhánh thay thế (alt flow).
- Đầy đủ – Bao quát toàn bộ các chức năng chính đã xác định trong sơ đồ use case, bao gồm cả các luồng đồng bộ (REST API) và bất đồng bộ (RabbitMQ).

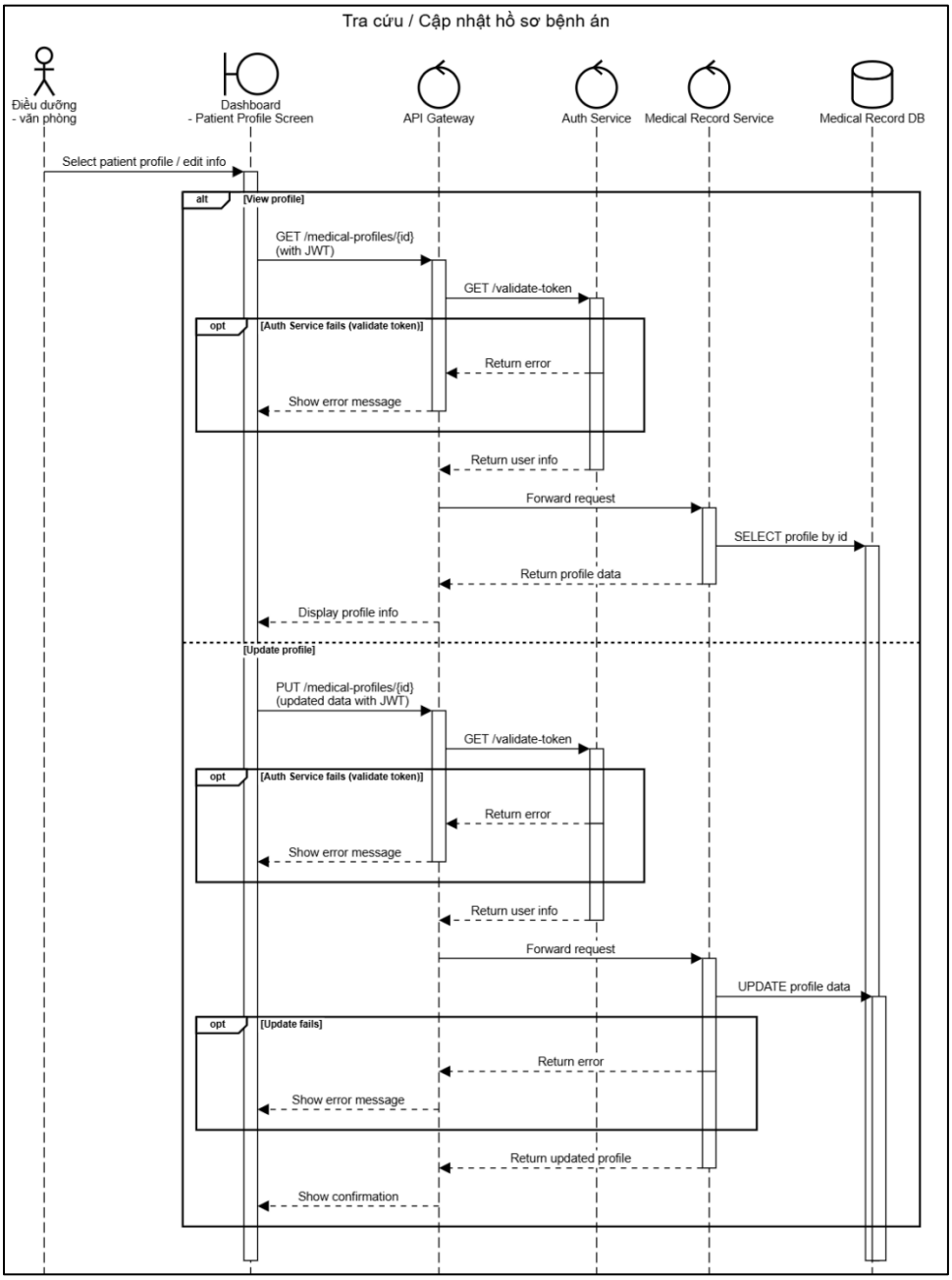
Do việc validate token dùng Auth Service luôn có cùng một flow nên để gọn gàng, nhóm chỉ vẽ chi tiết nhánh `opt` ở Hình 2 và Hình 3; ở các sơ đồ còn lại tuy chỉ để cặp GET-return nhưng sẽ được ngầm hiểu nhánh tương tự vẫn tồn tại.

1.2.1 Đăng ký bệnh nhân mới



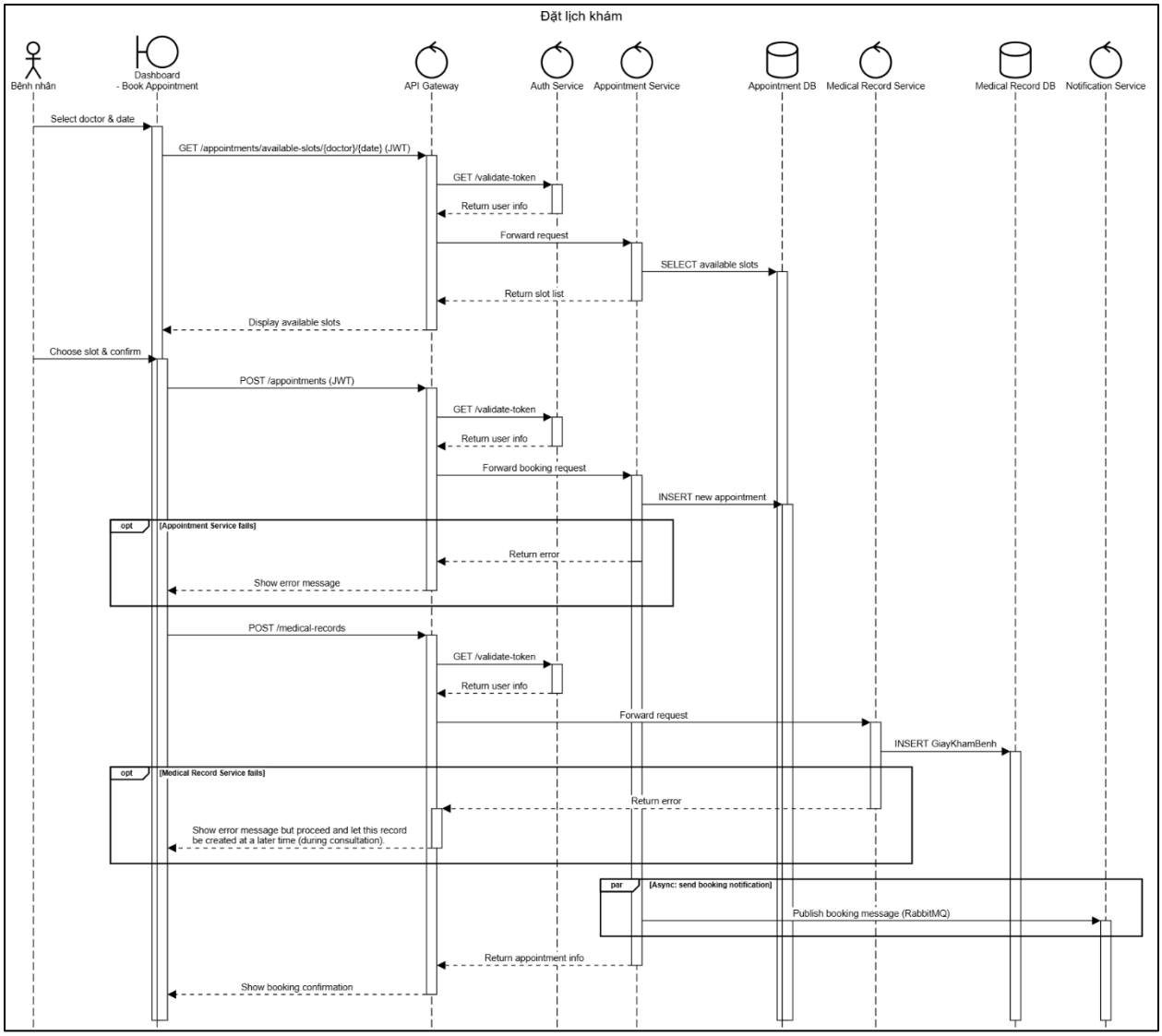
Hình 2. Sơ đồ tuần tự cho Đăng ký bệnh nhân mới.

1.2.2 Tra cứu / Cập nhật hồ sơ bệnh án



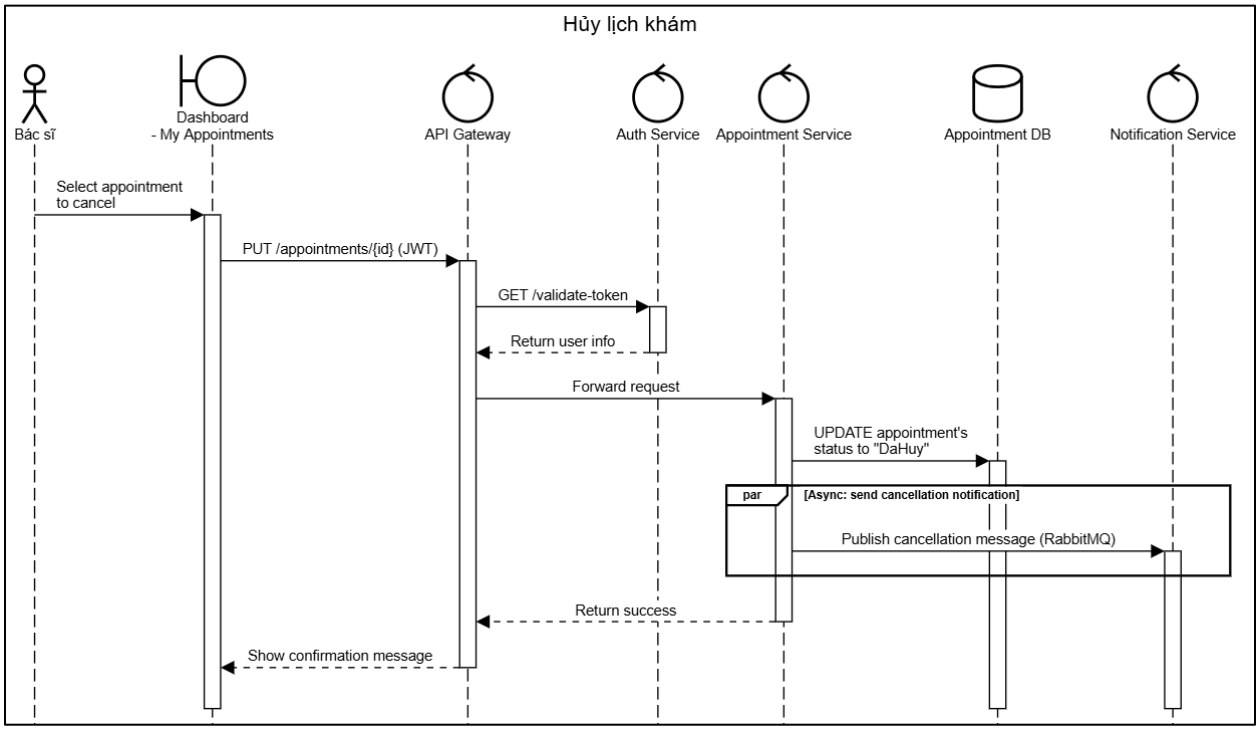
Hình 3. Sơ đồ tuần tự cho Tra cứu hoặc Cập nhật hồ sơ bệnh án.

1.2.3 Đặt lịch khám



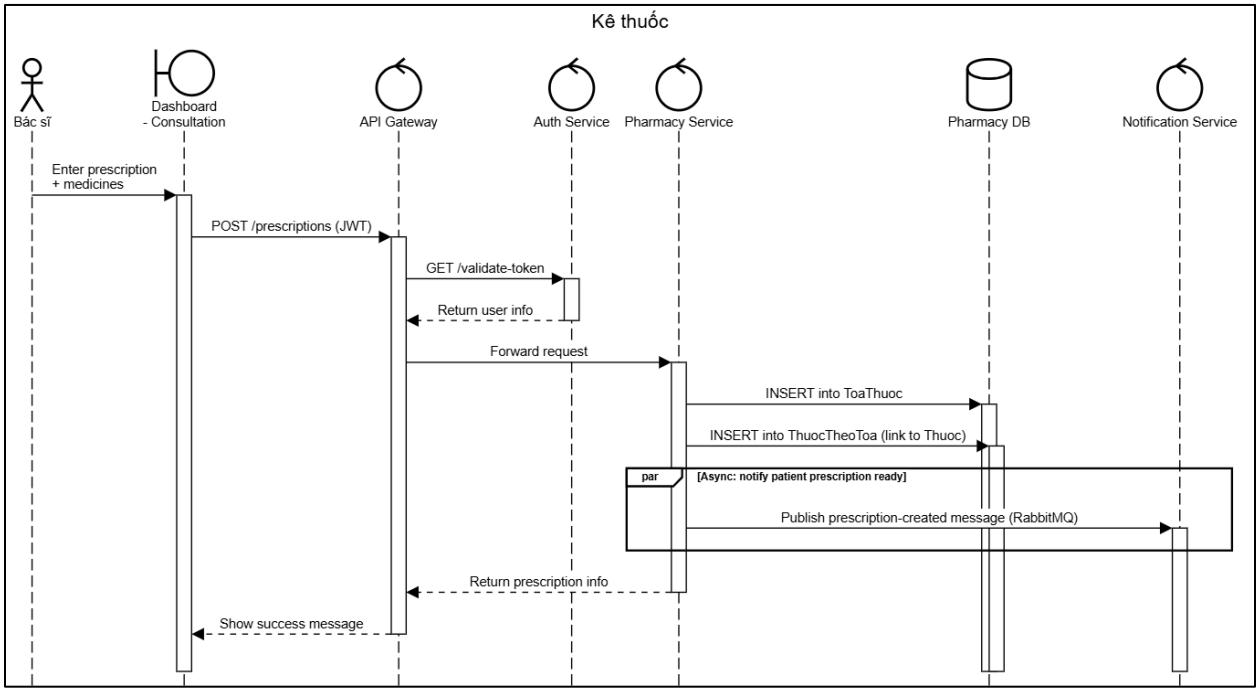
Hình 4. Sơ đồ tuần tự cho Đặt lịch khám.

1.2.4 Hủy lịch khám

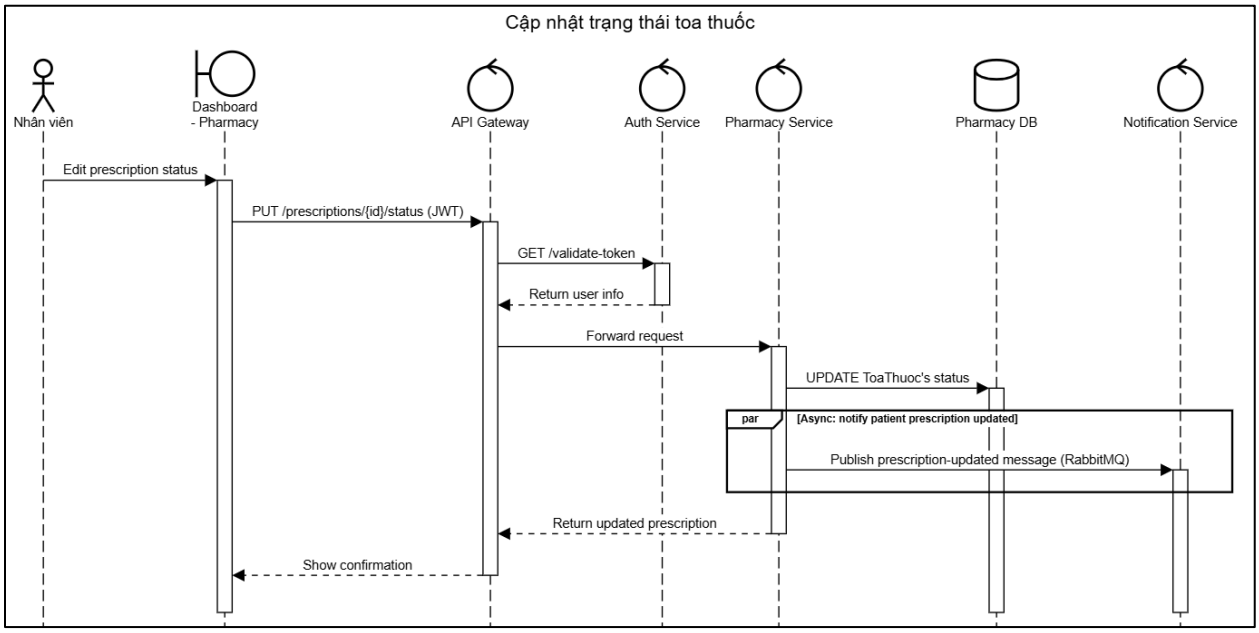


Hình 5. Sơ đồ tuần tự cho Hủy lịch khám.

1.2.5 Kê thuốc & Cập nhật toa thuốc

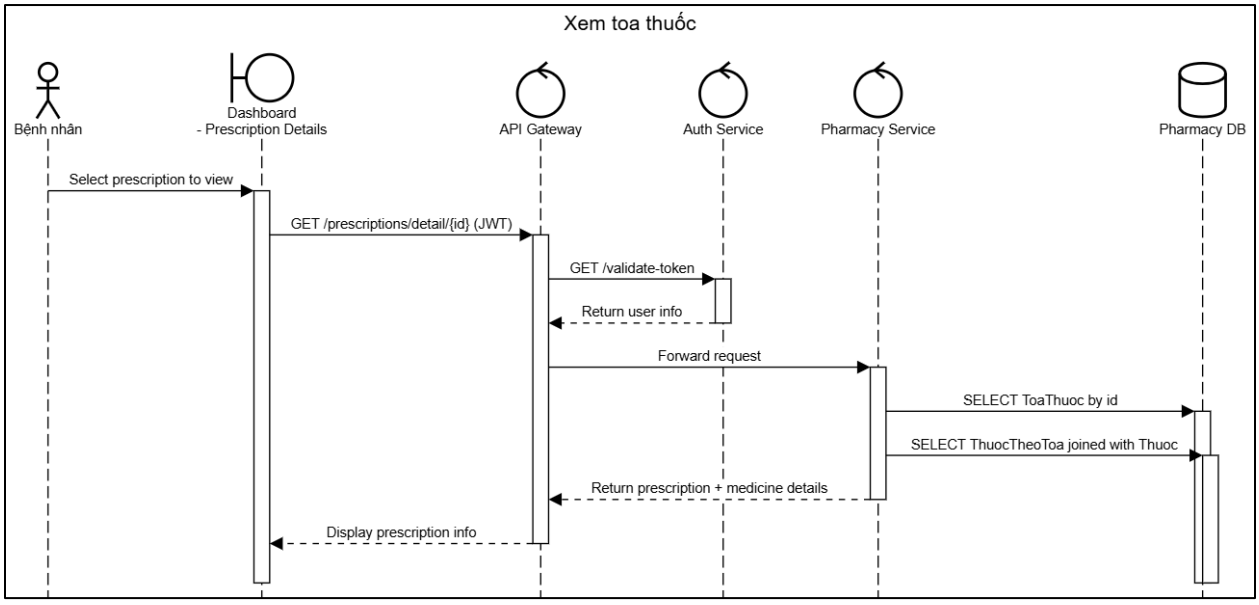


Hình 6. Sơ đồ tuần tự cho Kê thuốc.



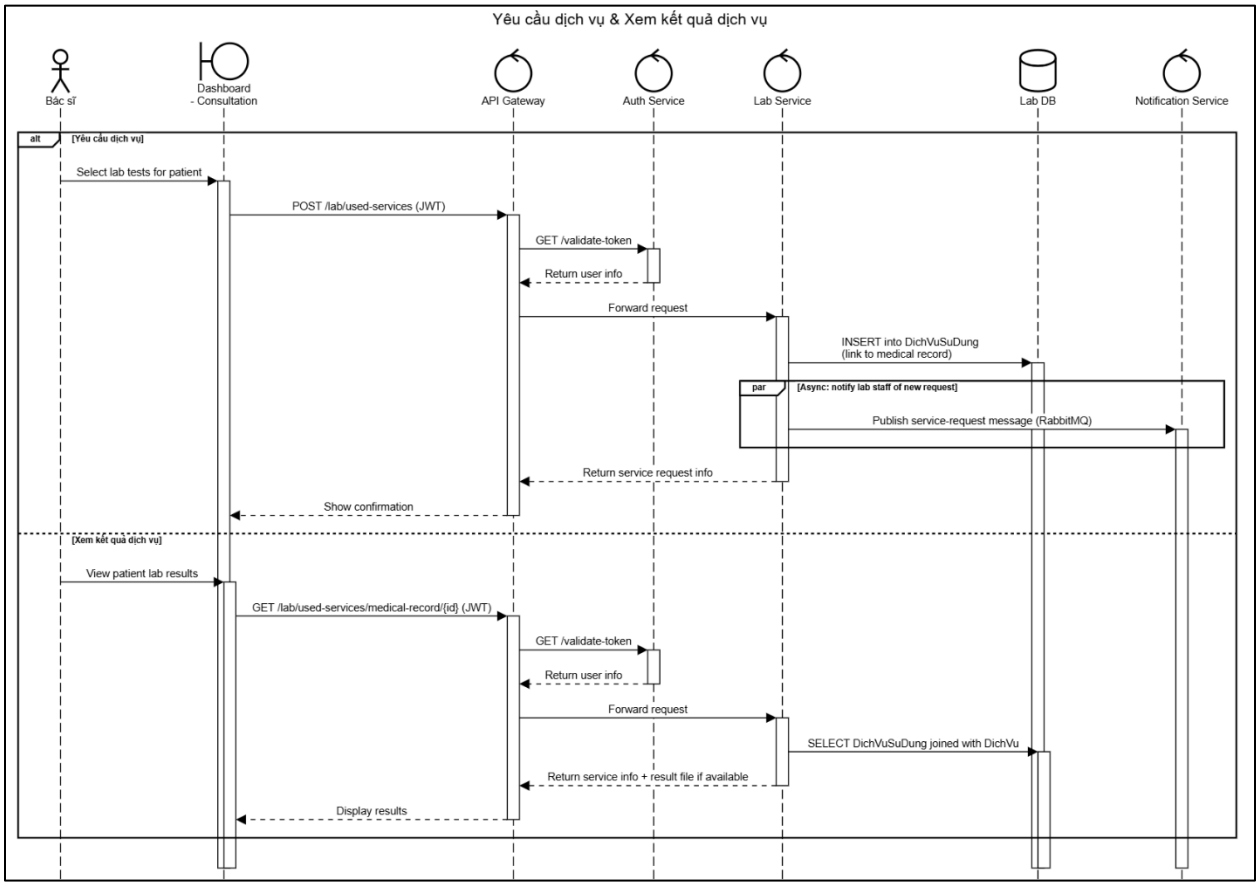
Hình 7. Sơ đồ tuần tự cho Cập nhật trạng thái toa thuốc.

1.2.6 Xem toa thuốc



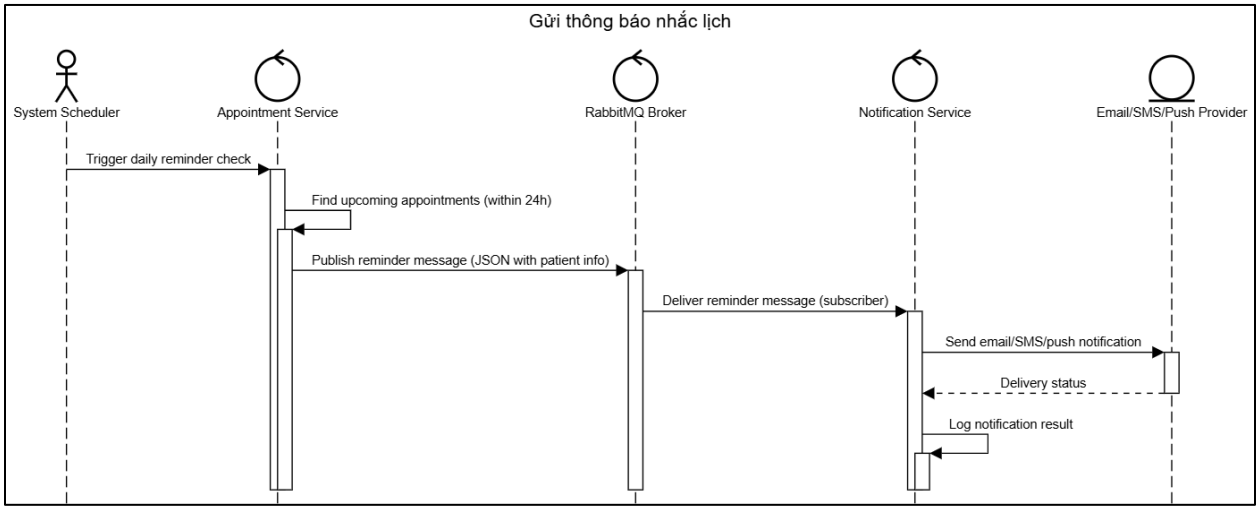
Hình 8. Sơ đồ tuần tự cho Xem toa thuốc.

1.2.7 Yêu cầu dịch vụ & Xem kết quả dịch vụ



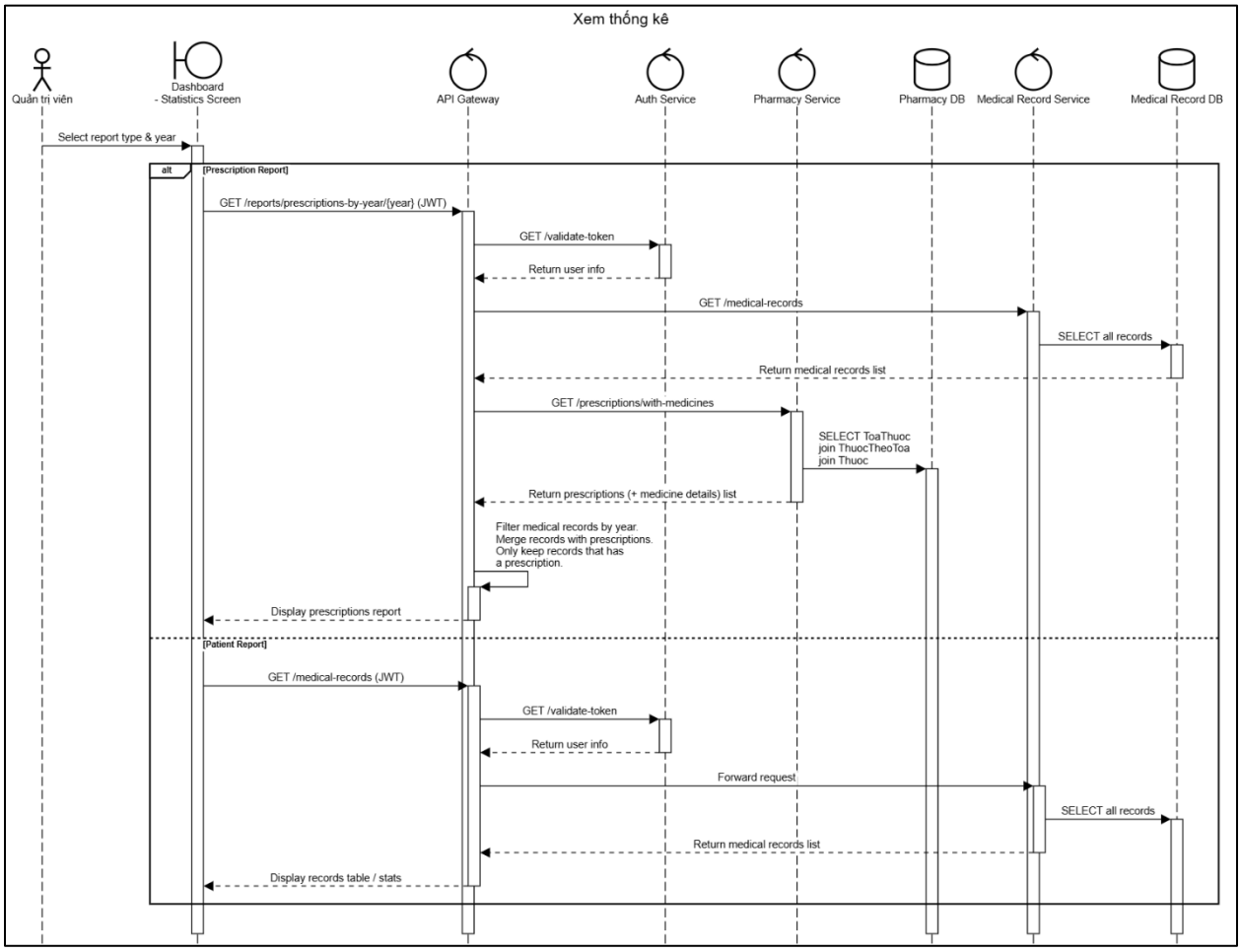
Hình 9. Sơ đồ tuần tự cho Yêu cầu dịch vụ và Xem kết quả dịch vụ.

1.2.8 Gửi thông báo nhắc lịch



Hình 10. Sơ đồ tuần tự cho Gửi thông báo nhắc lịch.

1.2.9 Xem thống kê



Hình 11. Sơ đồ tuần tự cho Xem thống kê.

2 Thiết kế hệ thống

2.1 Microservices

Hệ thống được xây dựng theo kiến trúc Microservice, trong đó mỗi dịch vụ đảm nhiệm một chức năng nghiệp vụ độc lập.

- **Ngôn ngữ & Framework:** Python (FastAPI) cho hiệu năng cao, hỗ trợ async, và dễ triển khai REST API.
- **Cơ sở dữ liệu:** PostgreSQL cho dữ liệu quan hệ, mỗi dịch vụ có CSDL riêng để đảm bảo độc lập và tránh phụ thuộc trực tiếp.
- **Triển khai:** Docker Compose quản lý các container của từng service, giúp môi trường triển khai nhất quán, dễ cài đặt và khởi động toàn bộ hệ thống.

- **Mục tiêu thiết kế:** Mở rộng dễ dàng, chịu lỗi tốt, tách biệt chức năng, triển khai và bảo trì độc lập.

Bằng cách gộp use cases liên quan (Hình 1) và xem xét các thành phần của sequence diagrams (1.2 Sequence diagrams), nhóm tổng hợp được những service như sau:

Service	Chức năng chính	CSDL quản lý	Giao tiếp với
API Gateway	Điều phối request từ client, xác thực token, định tuyến đến service đích.	Không lưu dữ liệu	Tất cả microservice
Auth Service	Quản lý tài khoản, đăng nhập/đăng ký, xác thực token, phân quyền.	Auth DB	API Gateway, Patient Service, Staff Service
Patient Service	Quản lý thông tin bệnh nhân, đăng ký bệnh nhân mới.	Patient DB	API Gateway, Auth Service, Medical Record Service, RabbitMQ
Medical Record Service	Quản lý hồ sơ bệnh án, giấy khám bệnh.	Medical Record DB	API Gateway, Patient Service
Appointment Service	Quản lý lịch khám, tìm slot trống, xác nhận/hủy lịch.	Appointment DB	API Gateway, Medical Record Service, RabbitMQ,
Pharmacy Service	Quản lý thuốc và toa thuốc.	Pharmacy DB	API Gateway, Medical Record Service, RabbitMQ
Lab Service	Quản lý dịch vụ xét nghiệm, kết quả xét nghiệm.	Lab DB	API Gateway, Medical Record Service, RabbitMQ
Notification Service	Gửi thông báo qua email/ SMS/push, lưu trữ thông báo.	Notification DB	API Gateway, RabbitMQ
Staff Service	Quản lý thông tin nhân sự (bác sĩ, điều dưỡng,...).	Staff DB	API Gateway, Auth service, RabbitMQ
RabbitMQ	Nhận yêu cầu gửi thông báo từ các service khác và phân phối lại cho Notification service.	Không	Notification Service, Patient Service, Appointment Service, Pharmacy Service, Staff Serivce

Bảng 2. Mô tả các microservice thành phần.

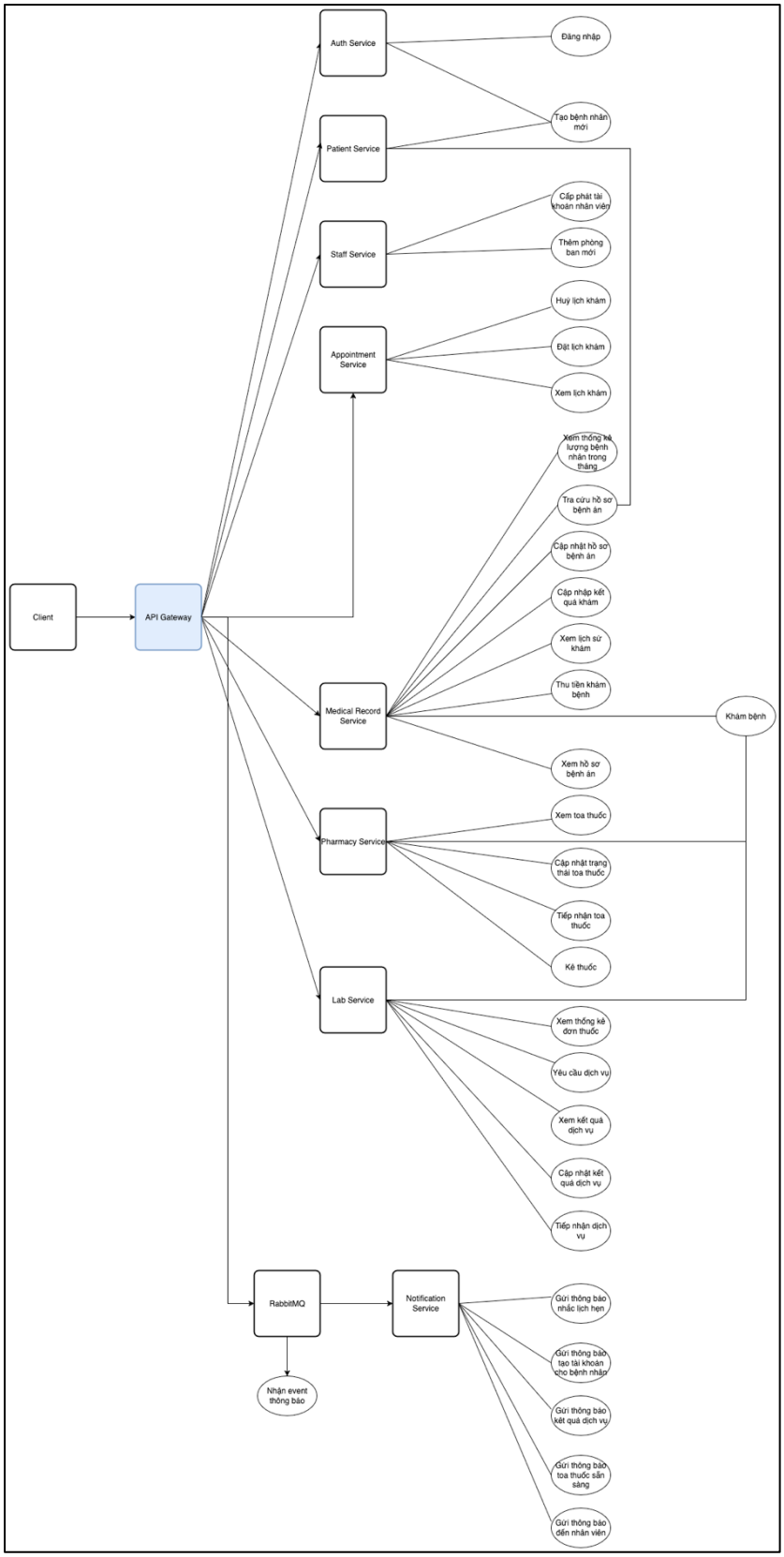
Việc **giao tiếp giữa các service** thực hiện theo 2 hướng sau:

- **Đồng bộ** (REST API): Tất cả request đi qua API Gateway, được định tuyến đến service đích qua HTTP(S). Ví dụ: POST /patients → API Gateway → Patient Service.
- **Bất đồng bộ** (RabbitMQ): Dùng cho các sự kiện không cần phản hồi ngay, ví dụ nhắc lịch khám, thông báo kết quả xét nghiệm, thông báo đơn thuốc đã sẵn sàng và gửi thông báo cho nhân viên. RabbitMQ giúp giảm độ trễ và tách biệt các service.

Cách tiếp cận này giúp giảm bớt công việc cho service gửi sự kiện khi nó không phải chờ đợi quá trình xử lý hoàn tất và có thể ủy thác các task cần nhiều thời gian (notification, scheduling) cho RabbitMQ. Đồng thời, các service nhận sự kiện qua RabbitMQ có thể xử lý khi chúng sẵn sàng, không bị ép phải thực hiện ngay lập tức. Bên cạnh đó, hệ thống sẽ dễ dàng mở rộng được và đảm bảo các sự kiện vẫn xử lý ổn định ngay cả khi một service gặp lỗi.

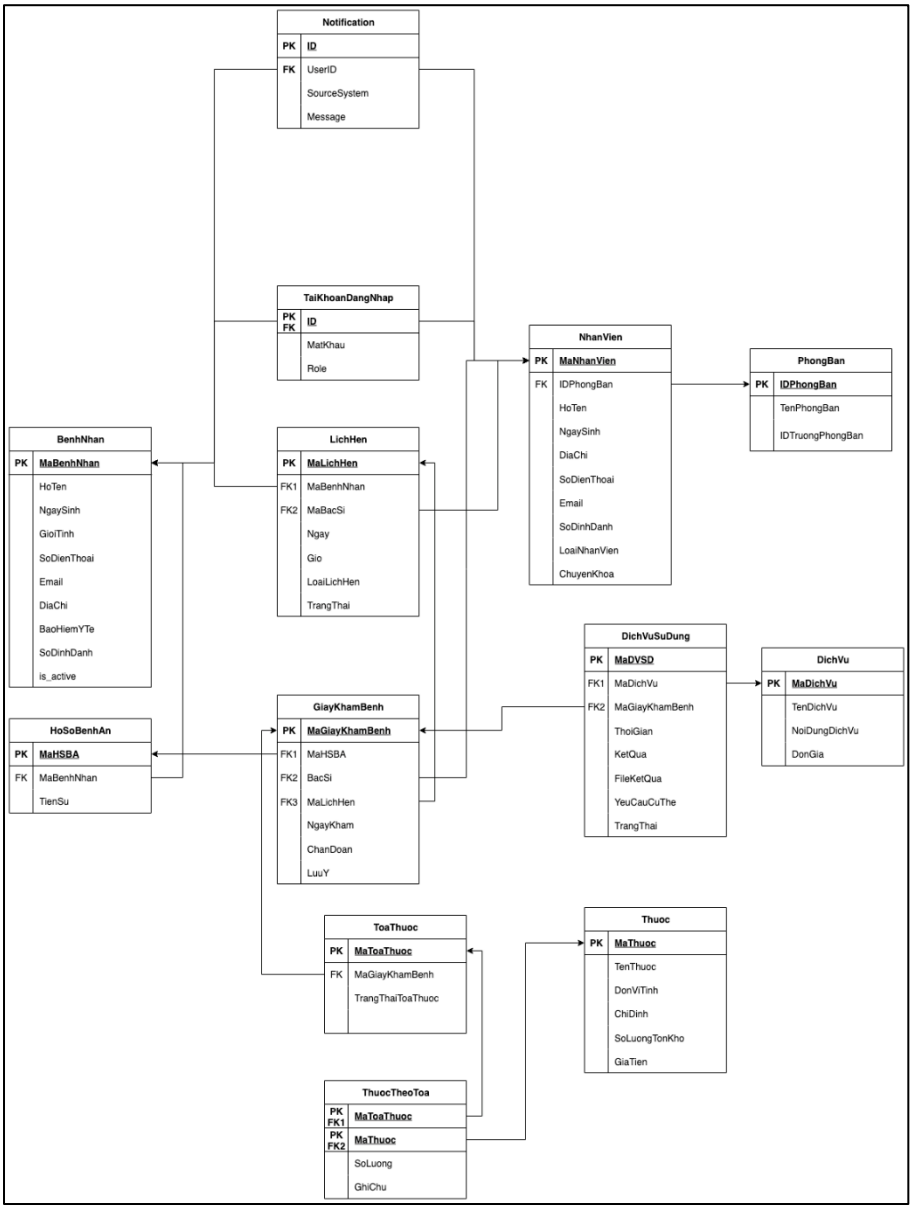
Về **bảo mật và phân quyền**, trừ các route công khai như register, login, và validate-token, mọi request đều thông qua API Gateway để kiểm tra JWT token với Auth Service trước khi được chuyển tiếp. Phân quyền thực hiện dựa trên Role-based Access Control (RBAC) theo vai trò như bệnh nhân, bác sĩ, điều dưỡng, admin. Những thông tin nhạy cảm như email, số điện thoại, và mật khẩu đảm bảo được mã hóa khi lưu trữ. Đồng thời, mỗi service chỉ truy cập CSDL của chính nó, giảm nguy cơ rò rỉ dữ liệu liên dịch vụ.

Hình 12 cho thấy mô hình tổng thể của kiến trúc microservice đã thiết kế, cùng mối tương quan giữa từng microservice và những use case mà nó xử lý.



Hình 12. Sơ đồ kiến trúc microservice.

2.2 Thiết kế dữ liệu



Hình 13. Mô hình dữ liệu mức quan hệ.

Hình 13 chỉ phục vụ cho góc nhìn tổng quan về mọi dữ liệu có trong hệ thống và cách chúng liên quan với nhau. Khi thực hiện cài đặt, sẽ đảm bảo các nguyên tắc thiết kế:

- **Tách biệt CSDL** – Mỗi microservice chỉ sử dụng một CSDL riêng để đảm bảo tính độc lập và khả năng mở rộng.
- **Quan hệ nội bộ** – Quan hệ và ràng buộc chỉ áp dụng bên trong CSDL của service. Ví dụ, Staff Service sẽ có 2 bảng quan hệ là NhanVien và PhongBan.

- **Ràng buộc liên dịch vụ** – Không dùng khóa ngoại liên DB mà sẽ được xử lý ở tầng ứng dụng thông qua API, bởi chính service đó (kiểm tra tính hợp lệ dữ liệu khi insert, update), hoặc ở client trước khi gửi request (tiền xử lý thông tin điền trong form).

Phân loại theo service, chúng ta sẽ có các bảng như sau:

Auth Service – Auth DB: Quản lý thông tin tài khoản và phân quyền.

- TaiKhoanDangNhap: Thông tin tài khoản.

Patient Service – Patient DB: Lưu trữ thông tin cá nhân của bệnh nhân.

- BenhNhan: Thông tin bệnh nhân.

Medical Record Service – Medical Record DB: Quản lý hồ sơ bệnh án và giấy khám bệnh.

- HoSoBenhAn: Hồ sơ bệnh án tổng hợp.
- GiayKhamBenh: Phiếu khám bệnh cho từng lần khám.

Appointment Service – Appointment DB: Quản lý lịch khám và trạng thái đặt hẹn.

- LichHen: Thông tin lịch khám (ngày, giờ, bác sĩ, bệnh nhân, trạng thái).

Pharmacy Service – Pharmacy DB: Quản lý thuốc và đơn thuốc.

- ToaThuoc: Thông tin đơn thuốc.
- ThuocTheoToa: Chi tiết thuốc trong đơn.
- Thuoc: Danh mục thuốc.

Lab Service – Lab DB: Quản lý dịch vụ xét nghiệm và kết quả.

- DichVu: Danh sách dịch vụ xét nghiệm/cận lâm sàng.
- DichVuSuDung: Lịch sử sử dụng dịch vụ.

Staff Service – Staff DB: Quản lý thông tin nhân sự y tế.

- NhanVien: Thông tin nhân viên.
- PhongBan: Danh sách các phòng ban.

Notification Service – Notification DB: Quản lý các thông báo đã được tiếp nhận từ RabbitMQ.

- Notification: Thông tin các thông báo.

2.3 RabbitMQ

RabbitMQ là một message broker mã nguồn mở, hỗ trợ giao tiếp bất đồng bộ giữa các dịch vụ thông qua cơ chế hàng đợi và publish/subscribe. Trong hệ thống này, RabbitMQ đóng vai trò trung gian truyền tải thông điệp giữa các dịch vụ: Appointment, Lab, Pharmacy, và Staff, nhằm gửi yêu cầu thông báo đến Notification service.

Các service gửi thông điệp đến một exchange có tên là **events** với kiểu **fanout**, đảm bảo mọi consumer liên quan đều nhận được thông điệp. Thông điệp được định dạng kiểu chuỗi **`event_name:payload`** và cấu hình với thuộc tính **`delivery_mode=2`** để đảm bảo tính bền vững.

```python
def publish_event(event_name, payload):
    """Enhanced publish_event matching lab service pattern"""
    try:
        print(f"🔄 Attempting to publish event: {event_name}")
        print(f"📝 Payload: {payload}")
        print(f"🔧 RabbitMQ Config: {RABBITMQ_CONFIG}")

        connection = pika.BlockingConnection(
            pika.ConnectionParameters(
                host=RABBITMQ_CONFIG['host'],
                port=RABBITMQ_CONFIG['port'],
                credentials=pika.PlainCredentials(
                    RABBITMQ_CONFIG['user'],
                    RABBITMQ_CONFIG['password']
                )
            )
        )
        channel = connection.channel()

        channel.exchange_declare(
            exchange='events',
            exchange_type='fanout',
            durable=True
        )

        # Publish the event
        message = f"{event_name}:{payload}"
        channel.basic_publish(
            exchange='events',
            routing_key='',
            body=message,
            properties=pika.BasicProperties(
                delivery_mode=2,  # Persistent messages
            )
        )

        print(f"✅ Event published successfully: {event_name}")
        connection.close()
        return True
```

Hình 14. Hàm publish_event cùng cấu hình liên quan để gửi một event đến exchange.

Lợi ích của việc sử dụng RabbitMQ cho công việc này bao gồm:

- **Giảm độ phụ thuộc giữa các service** – Các dịch vụ không cần biết chi tiết về Notification service. Notification service cũng không cần chuẩn bị các route trước cho các dịch vụ.
- **Tăng khả năng mở rộng** – Có thể dễ dàng thêm consumer mới mà không cần thay đổi logic gửi.
- **Đảm bảo tính bền vững** – Thông điệp được lưu trữ và tự động gửi lại nếu Notification chưa sẵn sàng xử lý. Giúp giảm trách nhiệm cho các service trong trường hợp Notification service gặp phải down time.
- **Đáng tin cậy** – RabbitMQ là một công cụ được xây dựng mạnh mẽ và đã được công nhận rộng rãi trong ứng dụng thực tế. Dùng RabbitMQ như message broker giúp giảm bớt việc phải xây dựng đầy đủ các trường hợp gửi thông báo lỗi một cách thủ công.

Các nhiệm vụ hiện tại RabbitMQ đang đảm nhận:

- Gửi thông báo cho patient khi được tạo tài khoản.
- Gửi thông báo nhắc lịch khi gần đến hẹn cho patient.
- Gửi thông báo đơn thuốc đã sẵn sàng cho patient.
- Gửi thông báo lab service đã hoàn thành cho patient.
- Gửi thông báo từ admin đến tất cả các staff.

2.4 Client

Phần client được xây dựng theo mô hình MVC nhằm tách biệt rõ ràng giữa xử lý dữ liệu, điều khiển luồng nghiệp vụ, và giao diện người dùng:

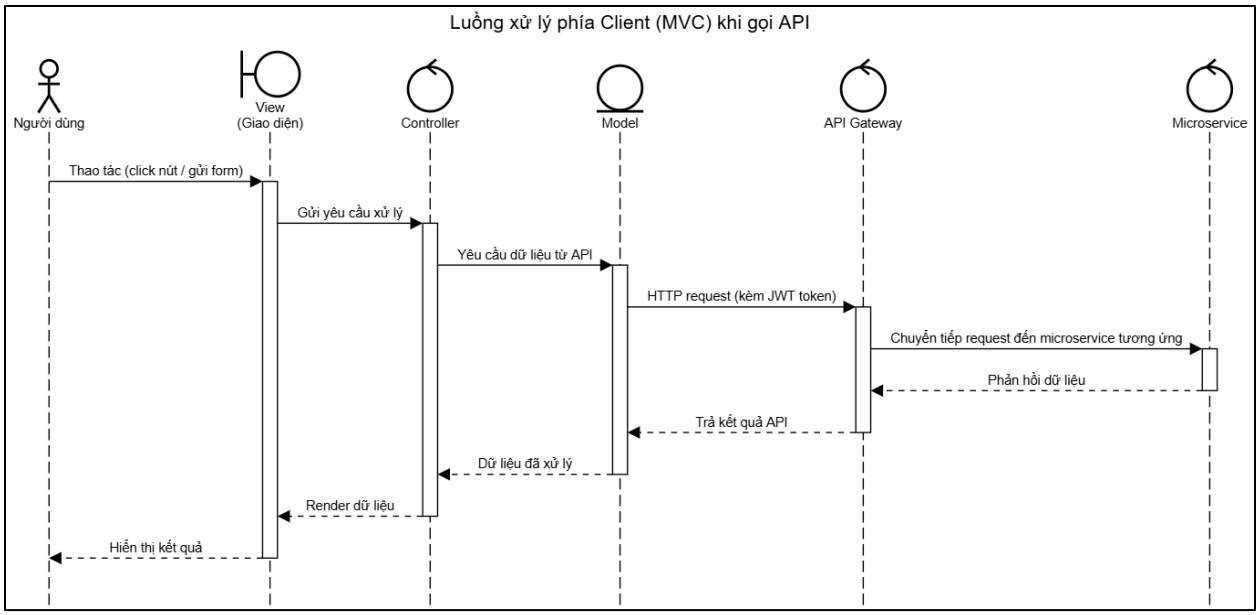
- Model – Quản lý dữ liệu, gọi API từ các microservice thông qua API Gateway, xử lý phản hồi và trả kết quả về cho Controller.
- View – Hiển thị giao diện, nhận dữ liệu từ Controller và render ra trình duyệt.
- Controller – Xử lý yêu cầu từ người dùng (qua View), điều phối gọi Model để lấy hoặc cập nhật dữ liệu, sau đó chọn View phù hợp để hiển thị kết quả.

Sử dụng PHP 8.x (thuộc MAMP); và quản lý phiên đăng nhập dùng PHP session để lưu thông tin đăng nhập và token JWT, token này được gửi kèm trong header Authorization khi gọi API. Ngoài ra, có những thư viện và framework hỗ trợ:

- Bootstrap 5: Tạo giao diện responsive và nhất quán.
- JavaScript + AJAX: Gửi yêu cầu bất đồng bộ đến API Gateway, cải thiện trải nghiệm người dùng.

Để giao tiếp với Backend, tất cả yêu cầu từ client đều được gửi tới API Gateway qua HTTP(S). Token JWT được lấy sau khi đăng nhập thành công, lưu ở session, và gắn vào header

`Authorization: Bearer <token>` trong mỗi request. Controller đóng vai trò định tuyến request từ View sang Model, Model sẽ thực hiện callApi() để gửi dữ liệu và nhận phản hồi từ API Gateway. **Error! Reference source not found.** mô tả luồng xử lý chung khi client muốn gửi request đến một service.



Hình 15. Luồng xử lý chung khi client gọi API.

3 Cài đặt

Source code là thư mục UDPT_QLBV. Chi tiết quy trình cài đặt vui lòng xem trong README.md.

- GitHub repo: https://github.com/phongan1x5/UDPT_QLBV
- Demo video: https://youtu.be/xqwWUNcCvx0

Client được đặt trên http://localhost/UDPT_QLBV/client. Khi gọi microservices, ta dùng base URL của API Gateway là http://localhost:6000 cộng với tên route cho đường dẫn kèm các thông tin cần thiết (header, body, URL parameter).

- Mọi route hỗ trợ được liệt kê ở microservices/api_gateway/routes.py.
- Những tài khoản tạo sẵn có thể được tìm thấy trong helper/setup_hospital_db.sql, tại phần INSERT cho bảng users. Tất cả mật khẩu là **password123**.

Về phía microservices, API Gateway sẽ định tuyến sang URL phù hợp của từng service, và mỗi port cũng được dùng tương ứng để triển khai chính service đó ở Docker.

microservices/api_gateway/database.py

```
MICROSERVICE_URLS = {
    'auth': 'http://auth_service:6001',
    'patient': 'http://patient_service:6002',
    'staff': 'http://staff_service:6003',
    'appointment': 'http://appointment_service:6004',
    'lab': 'http://lab_service:6005',
    'pharmacy': 'http://pharmacy_service:6006',
    'medical_record': 'http://medical_record_service:6007',
    'notification': 'http://notification_service:6008',
    'rabbitmq': 'http://hospital_rabbitmq:5672',

}
```

> Docker compose ps

NAME	...	PORTS
microservices-api_gateway-1	...	0.0.0.0:6000->6000/tcp
microservices-appointment_service-1	...	0.0.0.0:6004->6004/tcp
microservices-auth_service-1	...	0.0.0.0:6001->6001/tcp
microservices-lab_service-1	...	0.0.0.0:6005->6005/tcp
microservices-medical_record_service-1	...	0.0.0.0:6007->6007/tcp
microservices-notification_service-1	...	0.0.0.0:6008->6008/tcp
microservices-patient_service-1	...	0.0.0.0:6002->6002/tcp
microservices-pharmacy_service-1	...	0.0.0.0:6006->6006/tcp
microservices-staff_service-1	...	0.0.0.0:6003->6003/tcp
hospital_rabbitmq	...	0.0.0.0:5672->5672/tcp

Bảng 3. URL của các microservice ở server và thông tin triển khai trên Docker.

Các cấu hình mặc định (env) có thể được tùy chỉnh ở setup.py. Hầu như nhóm giữ các cài đặt có sẵn của ứng dụng, nhưng cần lưu ý do microservices được host trên docker, nên để kết nối với Postgres ở local thì `POSTGRES_HOST=host.docker.internal`.

3.1 Quy trình làm việc

Những buổi họp chính của nhóm được diễn ra định kỳ mỗi tuần một lần vào Thứ 7 sau buổi học lý thuyết tại lớp. Đồng thời, để trao đổi và theo dõi tiến độ công việc, nhóm còn sử dụng Messenger. Nhằm thuận tiện cho việc quản lý và bàn giao sản phẩm, các báo cáo và biểu đồ được đặt trên Onedrive chung, và source code thì commit lên GitHub. Thông tin chi tiết nhiệm vụ từng thành viên đảm nhiệm được liệt kê và cập nhật thường xuyên trong tệp **UDPT-06_PhanCong**.

References

There are no sources in the current document.