UTH UNIVERSITY OF TRANSPORT HOCHIMINH CITY

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
THÀNH PHỐ HỒ CHÍ MINH

Báo cáo

Môn : Xây dựng phần mềm hướng đối tượng

Đề tài : Gateway API – etc: Kong A, Ocelect,...

Mã học phần: 010112201002

ThS. Nguyễn Văn Chiến

UTH UNIVERSITY OF TRANSPORT HOCHIMINH CITY

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI THÀNH PHỐ HỒ CHÍ MINH

Thành viên nhóm :

2251120351-Trần Minh Hoàng	Thuyết trình & Slide Design
2251120223 -Huỳnh Công Luyện	Điều phối,tester, demo
2251120226 Nguyễn Đình Mạnh	API Gateway & Docker
2251120212 - Trần Võ Quang Huy	Backend & Chat Service
2251120382 - Trần Văn Tài	Backend & Login Service

1 Vấn đề trong hệ thống phân tán :

- Nhiều Service → Nhiều Endpoint
- Bảo mật & Xác thực rải rác
- Khó quản lý Log, Giám sát
- Hệ thống phức tạp, khó mở rộng

Vấn đề Phát sinh

- Client không thể quản lý hết Endpoint

  VD: Ứng dụng khách hàng (Mobile, Web) phải trực tiếp biết và gọi nhiều địa chỉ URL khác nhau để tổng hợp dữ liệu
  (ví dụ: gọi /api/users, /api/orders, /api/payments).
- Điều này làm tăng sự phức tạp cho phía Client, khiến việc thay đổi kiến trúc nội bộ trở nên khó khăn.

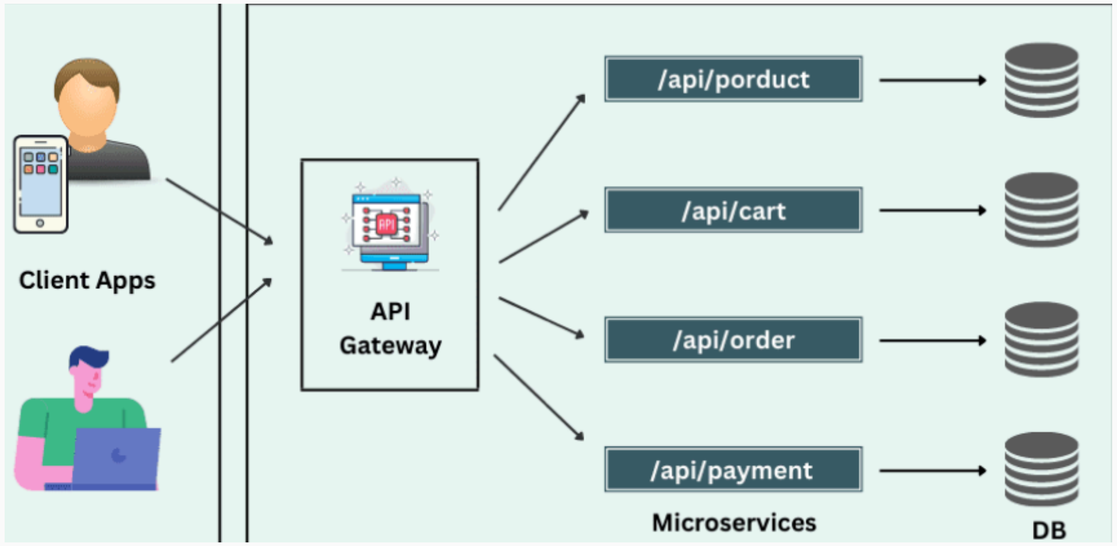
Giải pháp

- Hệ thống cần một điểm vào duy nhất – một "cửa ngõ chung" – để:
- Đơn giản hóa giao tiếp cho Client.
- Tập trung các chức năng ngang hàng như Bảo mật, Định tuyến và Giám sát.
- Cửa ngõ chung này là API Gateway.

2 Khái niệm về API Gateway

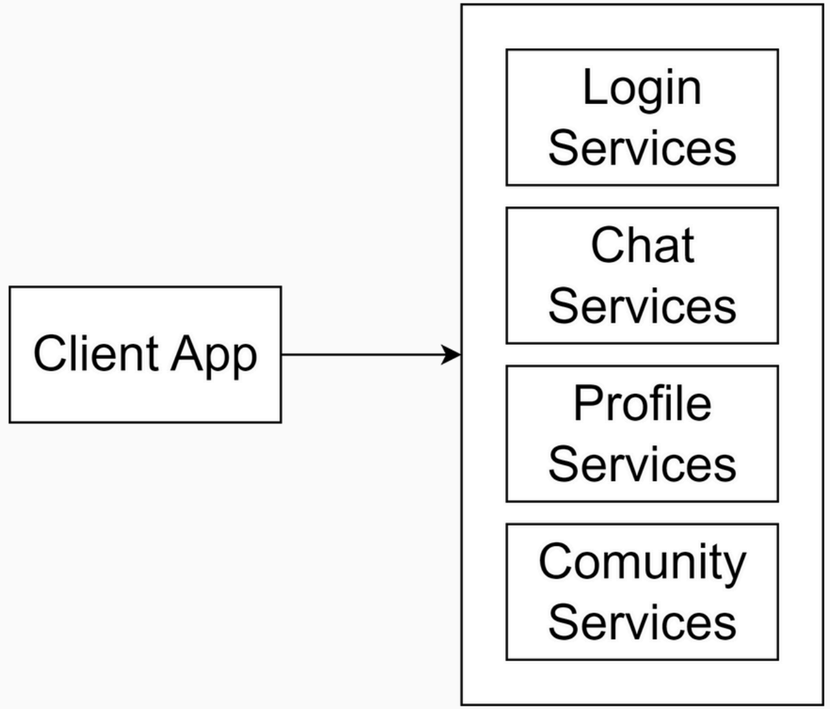
- API Gateway là một đơn vị trung gian giữa máy khách và tập hợp các dịch vụ back-end.
- Nó tiếp nhận tất cả các lệnh gọi API và định tuyến chúng đến một hoặc nhiều dịch vụ back-end phù hợp.

- Nó còn tổng hợp dữ liệu/tài nguyên phù hợp và phân phối đến người dùng một cách thống nhất.



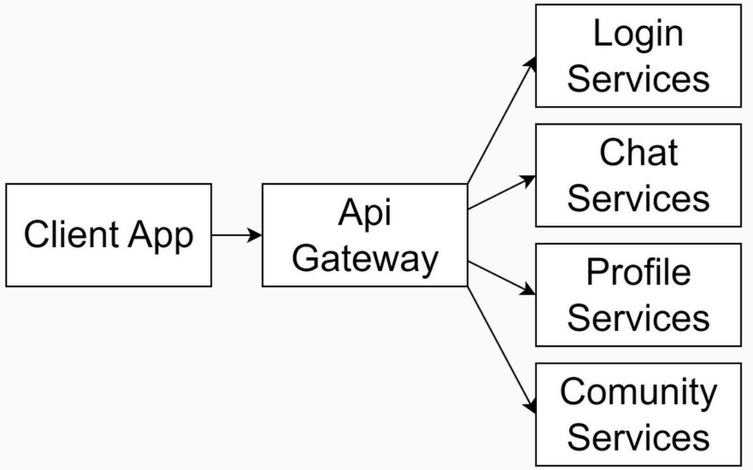
3 Kiến trúc khi không có API Gateway

- Client cần biết tất cả địa chỉ Service: Ứng dụng khách hàng phải cứng hóa (hardcode) và quản lý danh sách tất cả các địa chỉ và cổng (IP/Port) của Microservices.
- Giao tiếp phức tạp: Nếu một dịch vụ thay đổi vị trí, tất cả Client phải được cập nhật và triển khai lại.
- Lộ Cấu trúc Nội bộ: Client nhìn thấy rõ ràng cấu trúc và công nghệ backend, làm giảm khả năng tái cấu trúc (refactor) hệ thống.



4 Kiến trúc khi có API Gateway

- Ứng dụng khách hàng (Mobile/Web) chỉ cần biết duy nhất địa chỉ và cổng của API Gateway (ví dụ: https://api.yourdomain.com:8000)
- Gateway đóng vai trò là điểm truy cập duy nhất (Single Entry Point) cho mọi loại yêu cầu.



Khi nhận yêu cầu, Gateway sẽ tự xử lý logic định tuyến nội bộ:

- Yêu cầu .../api/login được Gateway chuyển đến Login Service (có thể đang chạy ở cổng nội bộ 5001).
- Yêu cầu .../api/messages được Gateway chuyển đến Messaging Service (có thể đang chạy ở cổng nội bộ 5002).

Lý do cốt lõi cho việc sử dụng API Gateway:

- Che giấu Cấu trúc Nội bộ
- Đơn giản hóa Triển khai
- Lợi ích: Dù bạn có 10 service hay 100 service, Client không bao giờ cần cập nhật cấu hình của mình miễn là địa chỉ Gateway không đổi

5 Các chức năng chính :

- Định tuyến (Routing)
- Xử lý Bảo mật và Ủy quyền (Authentication & Authorization)
- Điều tiết tốc độ (Rate Limiting)
- Hợp nhất yêu cầu (Request Aggregation)
- Chuyển đổi Giao thức/Định dạng

Mục tiêu của API gateway :

- Đơn giản hóa phía Client
- Điểm Truy Cập Duy Nhất
- Hợp nhất Yêu cầu
- Tách biệt và Bảo vệ Backend
- Định tuyến linh hoạt (Routing)
- Kiểm soát và Quản lý Lưu lượng
- Ẩn cấu trúc nội bộ
- Tập trung Bảo mật
- Điều tiết tốc độ
- Ghi nhật ký và Giám sát

6 Thách thức của API Gateway

Điểm thắt cổ chai và điểm lỗi duy nhất :

API Gateway là điểm truy cập duy nhất, nên nếu nó gặp sự cố, toàn bộ hệ thống sẽ ngừng hoạt động (Single Point of Failure - SPOF).
Nó có thể trở thành nút thắt cổ chai nếu không được mở rộng hoặc tối ưu hóa đúng cách.

Tăng Độ Trễ (Latency):

Gateway là một tầng trung gian mới, thêm vào một độ trễ nhỏ (overhead) cho mỗi yêu cầu.

Phức tạp trong vận hành:

- Việc quản lý và cấu hình logic định tuyến, bảo mật, và giới hạn tốc độ tại Gateway khiến nó trở nên phức tạp.
- Cần triển khai Gateway với tính sẵn sàng cao (HA - High Availability) và tự động hóa quy trình.

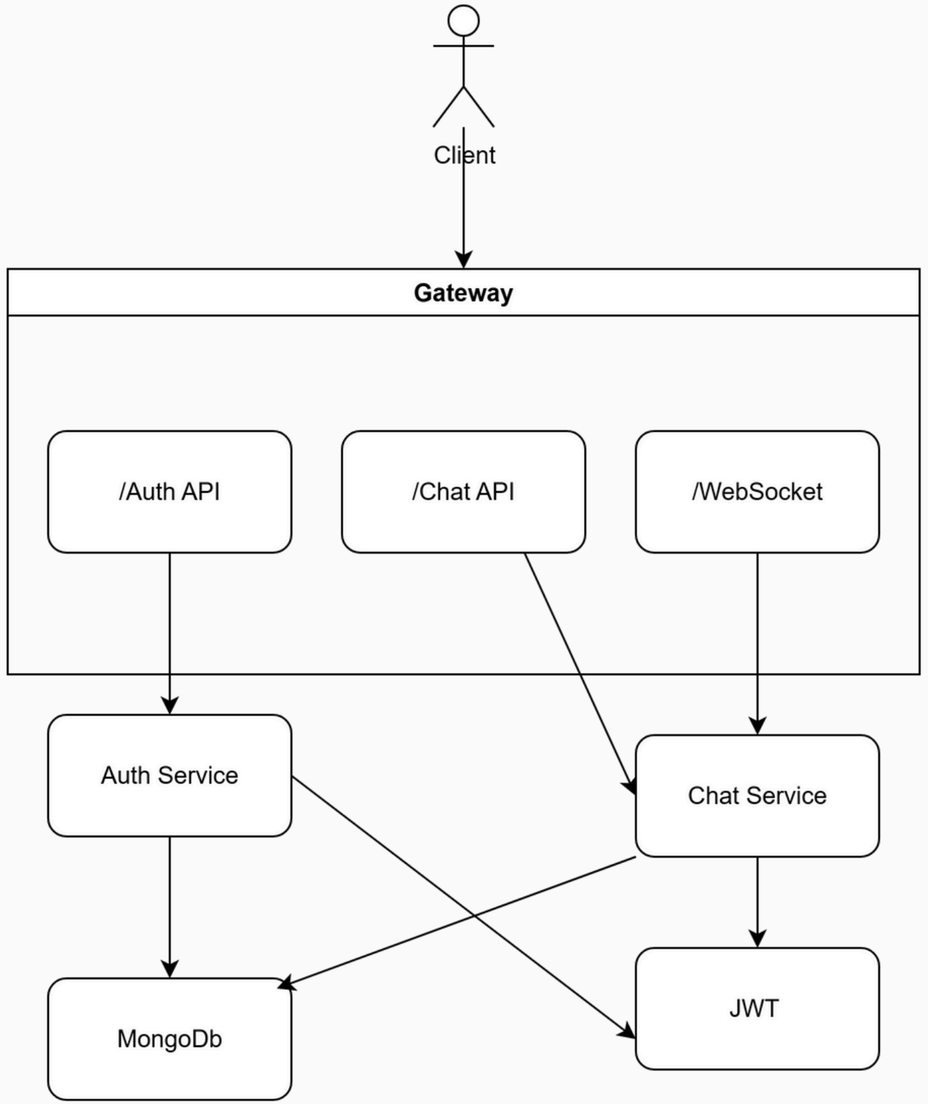
Nguy cơ thành "Monolith Gateway

- Dễ có xu hướng đặt quá nhiều logic nghiệp vụ vào Gateway, biến nó thành một khối kiến trúc khó bảo trì và làm mất đi sự linh hoạt của Microservices.
- Ví dụ: Khi client gọi một

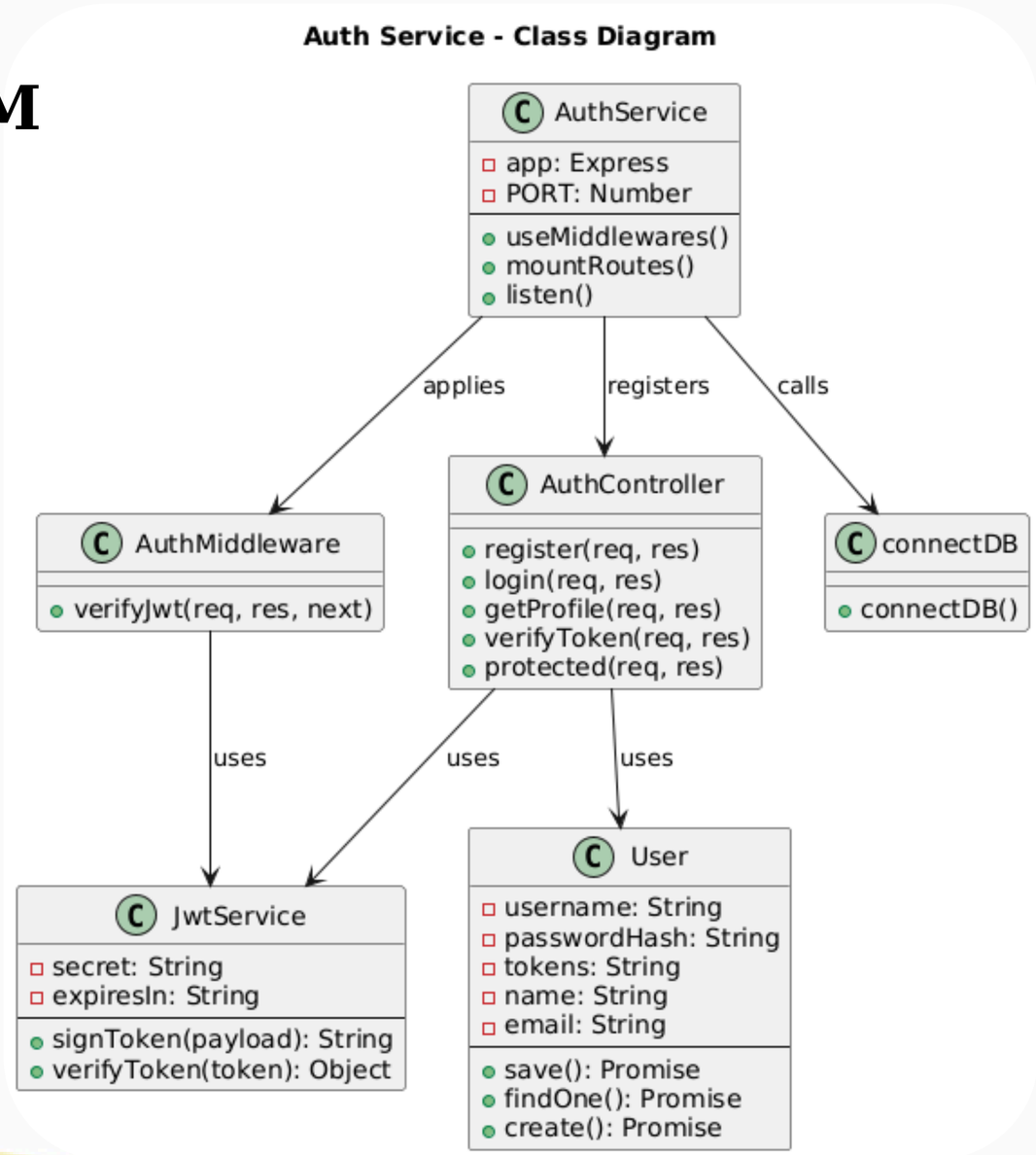
7 Kết luận

- Hệ thống Microservices là kiến trúc mạnh mẽ cho phép mở rộng và chịu lỗi vượt trội. Tuy nhiên, bản chất phân tán của nó tạo ra các thách thức lớn về quản lý, bảo mật và giao tiếp.
- Để tận dụng tối đa lợi ích, cần áp dụng các thực tiễn tốt nhất:
- Giữ Gateway "Mỏng": Chỉ xử lý các chức năng ngang hàng, không chứa logic nghiệp vụ.
- Đảm bảo Khả năng Mở rộng (Scale Ngang): Triển khai Gateway theo cụm để tránh trở thành Điểm Lỗi Duy nhất.

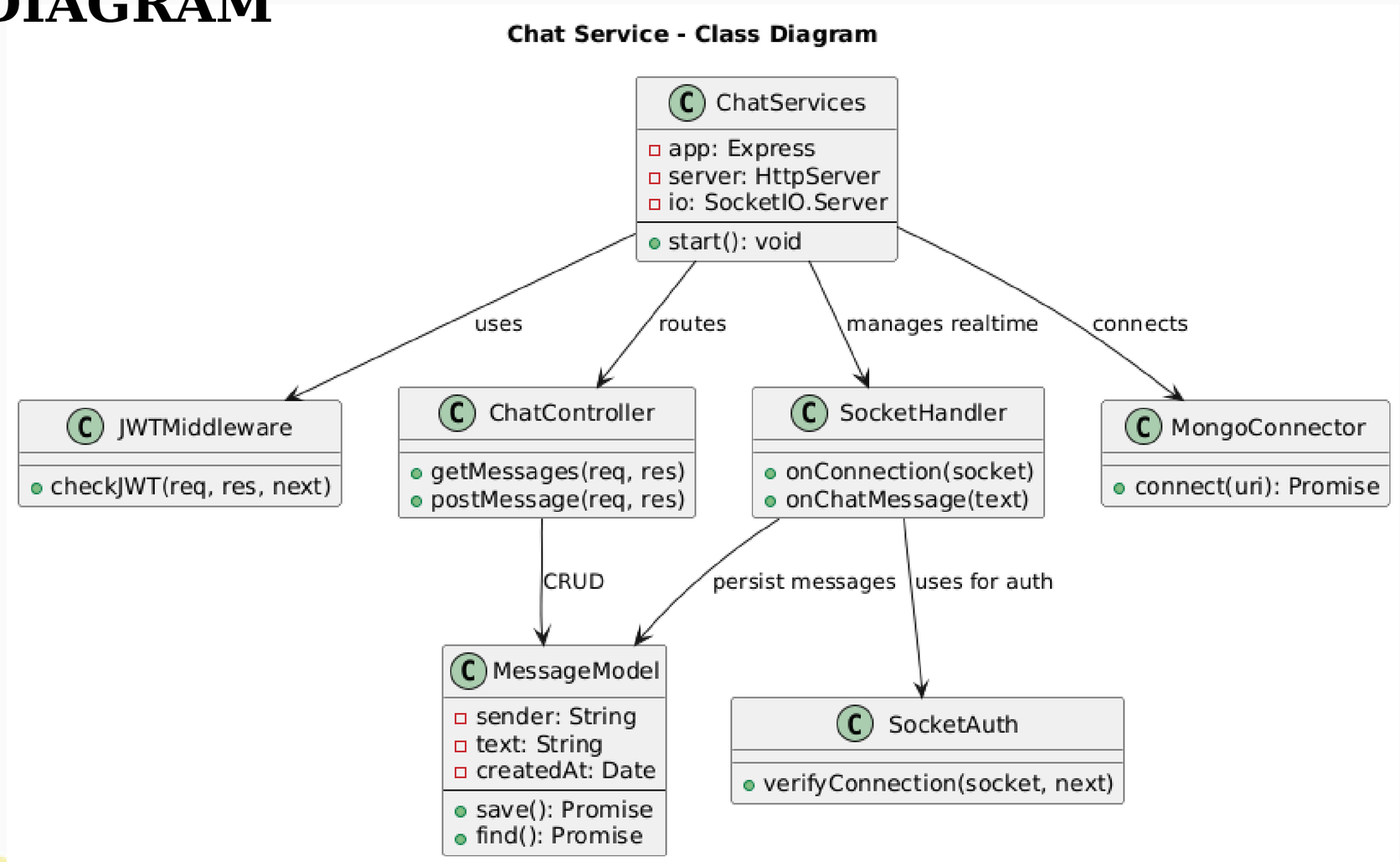
Demo sản phẩm


Client
Gateway
/Auth API
/Chat API
/WebSocket
Auth Service
Chat Service
MongoDb
JWT

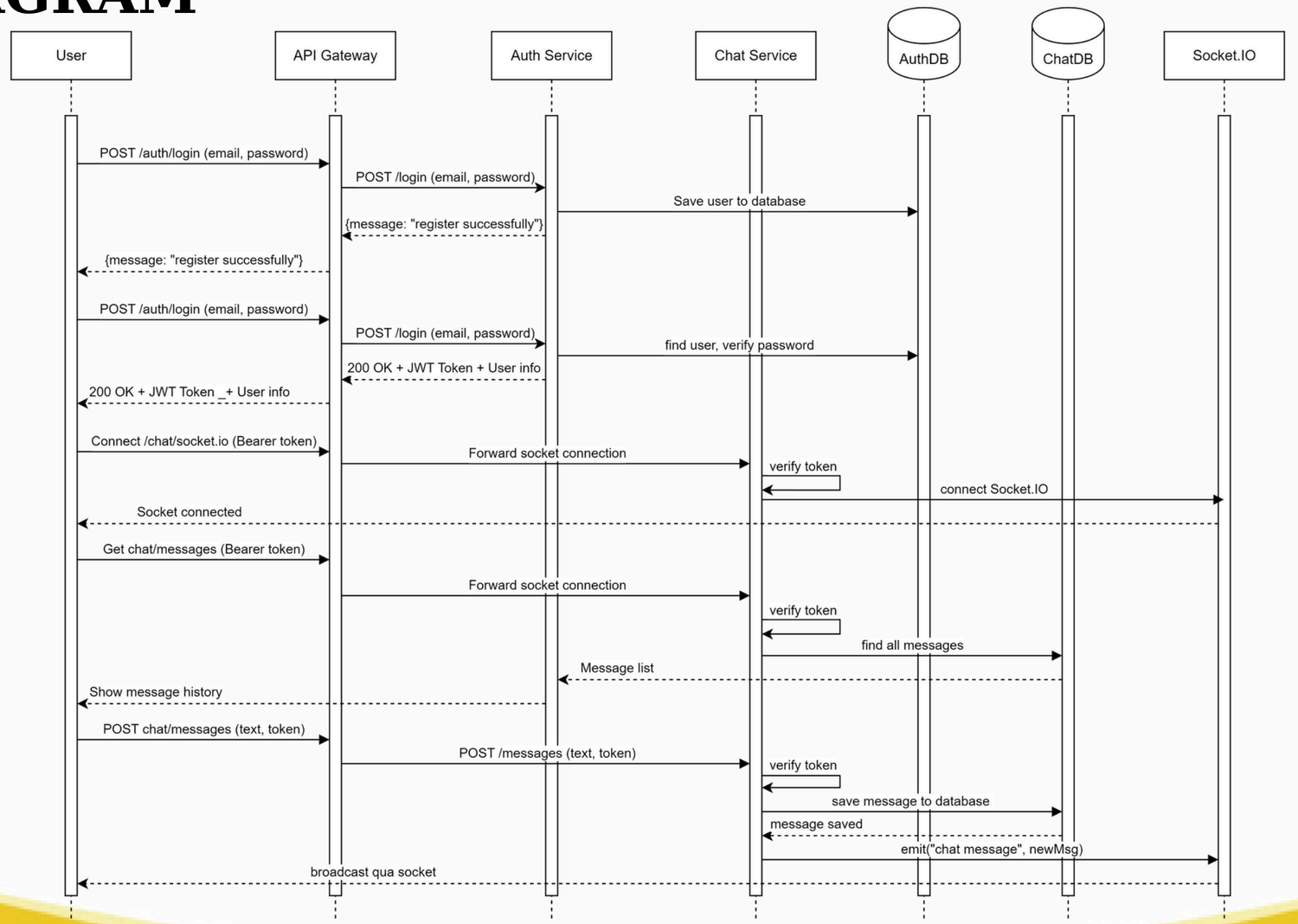
CLASS DIAGRAM



CLASS DIAGRAM



CLASS DIAGRAM



THANKS FOR LISTENING