**1. Tổng quan về GraphQL Server**

GraphQL là một ngôn ngữ truy vấn (query language) cho API, cho phép client yêu cầu chính xác dữ liệu mà họ cần từ server, thay vì nhận toàn bộ dữ liệu như REST API. Server GraphQL trong mã nguồn của bạn được xây dựng bằng **Apollo Server** tích hợp với **Express** và hỗ trợ cả HTTP và WebSocket (cho subscription).

**Cách hoạt động của GraphQL Server trong mã nguồn:**

* **Schema (typeDefs)**: Định nghĩa cấu trúc dữ liệu và các thao tác mà client có thể thực hiện (Query, Mutation, Subscription).
* **Resolvers**: Xử lý logic để trả về dữ liệu hoặc thực hiện các thay đổi dựa trên yêu cầu từ client.
* **Context**: Cung cấp thông tin ngữ cảnh (như thông tin người dùng từ token) cho các resolver.
* **WebSocket**: Hỗ trợ subscription để gửi dữ liệu theo thời gian thực.
* **Middleware**: Sử dụng Express để xử lý các yêu cầu HTTP, upload file, và CORS.

Server được khởi động trong file cuối cùng (startServer), kết nối với MongoDB, và lắng nghe các yêu cầu tại một cổng (mặc định là 4000).

**2. Các thành phần chính**

**a. typeDefs (Type Definitions)**

typeDefs là nơi định nghĩa **schema** của GraphQL, bao gồm các kiểu dữ liệu (types), truy vấn (Query), thay đổi (Mutation), và đăng ký (Subscription). Đây là "hợp đồng" giữa client và server, mô tả dữ liệu có thể truy vấn hoặc thay đổi.

* **Ví dụ trong mã nguồn:**

graphql

CollapseWrapCopy

type User {

id: ID!

username: String!

email: String!

profilePicture: String

createdAt: String!

followerCount: Int!

followingCount: Int!

isFollowed: Boolean

}

* + User là một kiểu dữ liệu với các trường như id, username, email, v.v.
  + Dấu ! biểu thị trường bắt buộc (non-nullable).
* **Các thành phần chính trong typeDefs:**
  + **Scalar**: Các kiểu dữ liệu cơ bản như ID, String, Int, Float, Date, Upload.
  + **Object Types**: Các kiểu phức tạp như User, Video, Comment.
  + **Union**: Kết hợp nhiều kiểu, ví dụ LikeTarget = Video | Comment.
  + **Query**: Các truy vấn để lấy dữ liệu.
  + **Mutation**: Các thao tác để thay đổi dữ liệu.
  + **Subscription**: Các đăng ký để nhận cập nhật thời gian thực.

**b. Query**

Query là các thao tác để **lấy dữ liệu** từ server. Client gửi yêu cầu và server trả về dữ liệu theo định dạng được định nghĩa trong typeDefs.

* **Ví dụ trong mã nguồn:**

graphql

CollapseWrapCopy

type Query {

getUser(id: String!): User

getUserVideos(id: String!, page: Int!, limit: Int!): [Video!]!

getVideo(id: ID!): Video

}

* + getUser(id: String!): Lấy thông tin người dùng theo id, trả về kiểu User.
  + getUserVideos: Lấy danh sách video của người dùng với phân trang (page, limit).
* **Resolver tương ứng:**

javascript

CollapseWrapCopy

getUser: async (\_, { id }, { user }) => {

return userService.getUser(id, user.id);

},

getUserVideos: async (\_, { id, page, limit }, { user }) => {

return videoService.getUserVideos(id, page, limit);

},

* + Resolver gọi các dịch vụ (userService, videoService) để xử lý logic và trả về dữ liệu.

**c. Mutation**

Mutation là các thao tác để **thay đổi dữ liệu** trên server, như thêm, sửa, xóa.

* **Ví dụ trong mã nguồn:**

graphql

CollapseWrapCopy

type Mutation {

registerUser(username: String!, email: String!, password: String!, avatarFile: Upload): AuthPayload!

uploadVideo(title: String!, videoFile: Upload!, thumbnailFile: Upload, category: ID, tags: [String!]): Video!

likeVideo(targetId: ID!): Boolean!

}

* + registerUser: Đăng ký người dùng mới, trả về AuthPayload (token và thông tin người dùng).
  + uploadVideo: Tải video lên, trả về thông tin video.
  + likeVideo: Thích một video, trả về Boolean (thành công hay không).
* **Resolver tương ứng:**

javascript

CollapseWrapCopy

registerUser: async (\_, { username, email, password, avatarFile }) => {

return userService.registerUser(username, email, password, avatarFile);

},

uploadVideo: async (\_, { title, videoFile, thumbnailFile, category, tags }, context) => {

if (!context.user) throw new Error('Bạn phải đăng nhập để tải video lên');

const video = await videoService.uploadVideo(context.user.id, title, videoFile, thumbnailFile, category, tags);

return video;

},

* + Resolver kiểm tra quyền (đăng nhập), gọi dịch vụ, và trả về kết quả.

**d. Subscription**

Subscription cho phép client **đăng ký nhận cập nhật thời gian thực** khi có sự kiện xảy ra (như bình luận mới, tin nhắn mới). Nó sử dụng WebSocket thay vì HTTP.

* **Ví dụ trong mã nguồn:**

graphql

CollapseWrapCopy

type Subscription {

commentAdded(videoId: ID!): Comment!

newMessage(conversationId: ID!): Message!

newNotification: Notification!

}

* + commentAdded: Nhận thông báo khi có bình luận mới cho video.
  + newMessage: Nhận tin nhắn mới trong hội thoại.
  + newNotification: Nhận thông báo mới.
* **Resolver tương ứng:**

javascript

CollapseWrapCopy

commentAdded: {

subscribe: (\_, { videoId }, context) => {

return pubsub.asyncIterator(`COMMENT\_ADDED\_${videoId}`);

},

},

newMessage: {

subscribe: (\_, { conversationId }, context) => {

return pubsub.asyncIterator(`NEW\_MESSAGE\_${conversationId}`);

},

},

* + Sử dụng pubsub (Publish-Subscribe) để phát và nhận sự kiện. Khi một sự kiện xảy ra (ví dụ: addComment), server phát tín hiệu qua pubsub.publish, và client nhận qua WebSocket.
* **Ví dụ phát sự kiện:**

javascript

CollapseWrapCopy

addComment: async (\_, { videoId, content, parentCommentId }, context) => {

const newComment = await commentService.addComment(videoId, content, parentCommentId, context.user.id);

pubsub.publish(`COMMENT\_ADDED\_${videoId}`, { commentAdded: newComment, videoId });

return newComment;

},

**3. Cách GraphQL Server hoạt động**

1. **Khởi động Server:**
   * File cuối cùng (startServer) khởi tạo Express, Apollo Server, và WebSocket Server.
   * Kết nối MongoDB qua mongoose.connect.
   * Tích hợp middleware như graphqlUploadExpress (upload file), cors, và morgan (logging).
2. **Xử lý yêu cầu:**
   * **HTTP**: Client gửi Query/Mutation qua endpoint /graphql. Apollo Server phân tích yêu cầu, gọi resolver tương ứng, và trả về dữ liệu.
   * **WebSocket**: Client kết nối qua WebSocket để đăng ký Subscription. Khi có sự kiện, server gửi dữ liệu qua pubsub.
3. **Xác thực:**
   * Token được gửi qua header Authorization. Hàm verifyToken kiểm tra và trả về thông tin người dùng (context.user) hoặc lỗi (context.tokenError).
4. **Phân giải dữ liệu:**
   * Resolver gọi các dịch vụ (userService, videoService, v.v.) để lấy hoặc thay đổi dữ liệu từ MongoDB.
5. **Thời gian thực:**
   * Subscription sử dụng graphql-ws và PubSub để gửi cập nhật khi có sự kiện (ví dụ: bình luận mới, tin nhắn mới).

**4. Ví dụ luồng hoạt động**

**Truy vấn video:**

* Client gửi:

graphql

CollapseWrapCopy

query {

getVideo(id: "123") {

title

user {

username

}

}

}

* Server:
  + Gọi resolver getVideo.
  + videoService.getVideo("123") trả về video.
  + Resolver Video.user lấy thông tin người dùng.
  + Trả về: { "title": "My Video", "user": { "username": "john" } }.

**Thêm bình luận:**

* Client gửi:

graphql

CollapseWrapCopy

mutation {

addComment(videoId: "123", content: "Great video!") {

id

content

}

}

* Server:
  + Resolver addComment kiểm tra đăng nhập, thêm bình luận qua commentService.
  + Phát sự kiện qua pubsub.publish.
  + Client đăng ký commentAdded nhận thông báo thời gian thực.

**5. Kết luận**

* **typeDefs**: Định nghĩa cấu trúc và API.
* **Query**: Lấy dữ liệu.
* **Mutation**: Thay đổi dữ liệu.
* **Subscription**: Cập nhật thời gian thực.
* Server sử dụng Apollo Server, Express, và WebSocket để xử lý yêu cầu linh hoạt, hỗ trợ cả HTTP và thời gian thực, với xác thực qua token và kết nối MongoDB.

TỔNG QUAN HỆ THỐNG   
  
  
  
  
Tôi sẽ điều chỉnh lại mô tả hệ thống dựa trên yêu cầu của bạn: loại bỏ **API Gateway**, **Google Auth**, và **Email Service**, đồng thời tập trung vào việc video được lưu trên **AWS S3** và server chỉ trả về link. Dưới đây là mô tả hệ thống đã được sửa đổi:

**1. Tổng quan hệ thống**

Hệ thống là một ứng dụng web (có thể là nền tảng chia sẻ video hoặc mạng xã hội) với các thành phần chính:

* **Client**: Được xây dựng bằng **React** và sử dụng **Apollo Client** để giao tiếp với server qua GraphQL.
* **Server**: Được xây dựng bằng **Node.js**, **Express**, và **Apollo Server** để xử lý các yêu cầu GraphQL.
* **Cơ sở dữ liệu**: Sử dụng **MongoDB** để lưu trữ dữ liệu.
* **Lưu trữ tệp**: Sử dụng **AWS S3** để lưu trữ các tệp tĩnh như video, hình ảnh, và server chỉ trả về URL của tệp.
* **Giao tiếp thời gian thực**: Sử dụng **WebSocket** để hỗ trợ các tính năng thời gian thực (Subscription).

**Sơ đồ điều chỉnh** (dựa trên yêu cầu):

* Loại bỏ **API Gateway**: Client giao tiếp trực tiếp với server qua HTTP và WebSocket.
* Loại bỏ **Google Auth** và **Email Service**: Không có xác thực qua Google và không gửi email.
* Video được lưu trên **AWS S3**, server chỉ trả về link (URL) để client truy cập.

**2. Các thành phần chính và luồng hoạt động**

**a. Client: React Apollo Client GraphQL**

* **Công nghệ**:
  + **React**: Thư viện JavaScript để xây dựng giao diện người dùng.
  + **Apollo Client**: Thư viện quản lý GraphQL, giúp client gửi các truy vấn (Query), thay đổi (Mutation), và đăng ký (Subscription) tới server.
* **Chức năng**:
  + Giao diện người dùng (UI) được xây dựng bằng React, cho phép người dùng tương tác với hệ thống (xem video, đăng bình luận, gửi tin nhắn, v.v.).
  + Apollo Client quản lý các yêu cầu GraphQL:
    - **Query**: Lấy dữ liệu (ví dụ: danh sách video, thông tin người dùng).
    - **Mutation**: Thay đổi dữ liệu (ví dụ: đăng video, thích video, gửi tin nhắn).
    - **Subscription**: Nhận cập nhật thời gian thực (ví dụ: thông báo bình luận mới, tin nhắn mới).
* **Giao tiếp**:
  + Gửi yêu cầu **HTTP** (Query/Mutation) trực tiếp tới server qua endpoint /graphql.
  + Sử dụng **WebSocket** để kết nối với server và nhận dữ liệu thời gian thực (Subscription).
  + Nhận URL của video từ server và truy cập trực tiếp trên AWS S3 để phát video.

**b. Server: Node Apollo Server GraphQL Express**

* **Công nghệ**:
  + **Node.js**: Môi trường runtime để chạy mã JavaScript phía server.
  + **Express**: Framework web để xử lý các yêu cầu HTTP.
  + **Apollo Server**: Thư viện GraphQL để xử lý các yêu cầu Query, Mutation, và Subscription.
* **Chức năng**:
  + Xử lý các yêu cầu GraphQL từ client:
    - **Query**: Truy vấn dữ liệu từ MongoDB (ví dụ: lấy danh sách video, thông tin người dùng).
    - **Mutation**: Thay đổi dữ liệu (ví dụ: đăng video, thêm bình luận, theo dõi người dùng).
    - **Subscription**: Gửi cập nhật thời gian thực qua WebSocket (ví dụ: thông báo bình luận mới, tin nhắn mới).
  + Tích hợp với **MongoDB** để lưu trữ và truy xuất dữ liệu.
  + Tích hợp với **AWS S3** để lưu trữ các tệp tĩnh (video, hình ảnh), và chỉ trả về URL của tệp cho client.
* **Giao tiếp**:
  + Nhận yêu cầu trực tiếp từ client qua HTTP (Query/Mutation) hoặc WebSocket (Subscription).
  + Truy vấn hoặc cập nhật dữ liệu trong **MongoDB**.
  + Lưu trữ tệp (video, hình ảnh) trên **AWS S3** và trả về URL.

**c. Database: MongoDB**

* **Công nghệ**: MongoDB là cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng tài liệu JSON.
* **Chức năng**:
  + Lưu trữ các thông tin như:
    - Thông tin người dùng (User).
    - Video (Video) – bao gồm metadata và URL của video trên S3.
    - Bình luận (Comment).
    - Hội thoại và tin nhắn (Conversation, Message).
    - Thông báo (Notification).
  + Server truy vấn hoặc cập nhật dữ liệu trong MongoDB thông qua các dịch vụ (như userService, videoService).
* **Luồng**:
  + Apollo Server gửi truy vấn MongoDB để lấy hoặc cập nhật dữ liệu.
  + Dữ liệu (bao gồm URL của video trên S3) được trả về và gửi lại cho client.

**d. AWS S3**

* **Công nghệ**: AWS S3 (Simple Storage Service) là dịch vụ lưu trữ đám mây của Amazon.
* **Chức năng**:
  + Lưu trữ các tệp tĩnh như video, hình ảnh (thumbnail, avatar).
  + Khi người dùng tải video lên (qua Mutation uploadVideo), server lưu tệp vào S3, lưu URL của tệp trong MongoDB, và trả về URL cho client.
  + Client sử dụng URL để truy cập trực tiếp tệp trên S3 (ví dụ: phát video).
* **Luồng**:
  + Server nhận tệp từ client (qua graphql-upload).
  + Tệp được tải lên S3, và URL được lưu vào MongoDB.
  + Server trả về URL cho client, và client truy cập tệp trực tiếp từ S3.

**e. WebSocket Link**

* **Chức năng**:
  + Hỗ trợ giao tiếp thời gian thực cho các Subscription.
  + Sử dụng giao thức WebSocket để gửi dữ liệu từ server tới client ngay khi có sự kiện.
* **Luồng**:
  + Client đăng ký Subscription (ví dụ: commentAdded, newMessage).
  + Server sử dụng PubSub để phát sự kiện (qua pubsub.publish).
  + Dữ liệu được gửi qua WebSocket tới client.

**3. Luồng hoạt động chi tiết (Ví dụ: Tải video lên và xem video)**

**a. Người dùng tải video lên**

1. Client (React) gửi Mutation uploadVideo qua HTTP trực tiếp tới server:

graphql

CollapseWrapCopy

mutation {

uploadVideo(title: "My Video", videoFile: <file>, thumbnailFile: <file>, category: "123", tags: ["fun"]) {

id

title

videoUrl

}

}

1. Server (Apollo Server) nhận yêu cầu:
   * Resolver uploadVideo kiểm tra xác thực (token từ context.user).
   * Sử dụng videoService để xử lý: video và thumbnail được tải lên AWS S3.
   * URL của video và thumbnail được lưu vào MongoDB cùng với metadata (title, category, tags, v.v.).
   * Server trả về thông tin video, bao gồm videoUrl (URL trên S3):

json

CollapseWrapCopy

{

"id": "video123",

"title": "My Video",

"videoUrl": "https://s3.amazonaws.com/bucket/video123.mp4"

}

1. Client nhận videoUrl và lưu thông tin video.

**b. Người dùng xem video**

1. Client gửi Query getVideo để lấy thông tin video:

graphql

CollapseWrapCopy

query {

getVideo(id: "video123") {

title

videoUrl

user {

username

}

}

}

1. Server nhận yêu cầu:
   * Resolver getVideo gọi videoService.getVideo để truy vấn MongoDB.
   * Lấy thông tin video, bao gồm videoUrl (URL trên S3).
   * Trả về dữ liệu:

json

CollapseWrapCopy

{

"title": "My Video",

"videoUrl": "https://s3.amazonaws.com/bucket/video123.mp4",

"user": { "username": "john" }

}

1. Client sử dụng videoUrl để phát video trực tiếp từ S3 (không cần server trung gian).

**c. Thông báo thời gian thực (ví dụ: bình luận mới)**

1. Client đăng ký Subscription commentAdded:

graphql

CollapseWrapCopy

subscription {

commentAdded(videoId: "video123") {

id

content

user {

username

}

}

}

1. Khi một người dùng khác thêm bình luận (qua Mutation addComment):
   * Server lưu bình luận vào MongoDB.
   * Phát sự kiện qua pubsub.publish:

javascript

CollapseWrapCopy

pubsub.publish(`COMMENT\_ADDED\_video123`, { commentAdded: newComment, videoId: "video123" });

1. Client nhận dữ liệu qua WebSocket và cập nhật giao diện (hiển thị bình luận mới).

**4. Các tính năng chính của hệ thống**

Dựa trên mã nguồn trước đó, hệ thống hỗ trợ:

* **Quản lý người dùng**: Đăng ký, đăng nhập, theo dõi/bỏ theo dõi, cập nhật hồ sơ (không có Google Auth).
* **Quản lý video**: Tải lên, xem, thích, lưu, tìm kiếm video (video lưu trên S3, server trả về URL).
* **Bình luận**: Thêm, thích, trả lời bình luận.
* **Hội thoại và tin nhắn**: Gửi tin nhắn, nhận tin nhắn thời gian thực.
* **Thông báo**: Nhận thông báo thời gian thực (bình luận, lượt thích, video mới).
* **Đề xuất video**: Dựa trên sở thích người dùng (thông qua UserPreference và UserInteraction).
* **Thống kê**: Thống kê video, người dùng, tương tác (ví dụ: getDailyVideoStatistics).

**5. Ưu điểm của kiến trúc**

* **GraphQL**: Client yêu cầu chính xác dữ liệu cần thiết, giảm tải dữ liệu không cần thiết.
* **Thời gian thực**: WebSocket và Subscription hỗ trợ cập nhật tức thì.
* **Lưu trữ hiệu quả**: AWS S3 lưu trữ video, server chỉ trả về URL, giảm tải cho server.
* **Tích hợp đám mây**: MongoDB phù hợp với dữ liệu không cấu trúc, S3 tối ưu cho tệp tĩnh.
* **Đơn giản hóa**: Loại bỏ API Gateway, Google Auth, và Email Service làm giảm độ phức tạp.

**6. Hạn chế**

* **Phức tạp**: GraphQL và WebSocket vẫn phức tạp hơn REST API.
* **Hiệu suất**: WebSocket có thể tiêu tốn tài nguyên nếu có nhiều kết nối đồng thời.
* **Bảo mật**: Cần đảm bảo URL trên S3 được bảo vệ (ví dụ: sử dụng presigned URL) để tránh truy cập trái phép.
* **Xác thực**: Không có Google Auth, hệ thống cần cơ chế xác thực khác (như email/mật khẩu với JWT).

**7. Sơ đồ điều chỉnh**

Dựa trên yêu cầu, sơ đồ mới sẽ như sau:

* **React Apollo Client GraphQL** ↔ **Node Apollo Server GraphQL Express** (HTTP/WebSocket).
* **Node Apollo Server** ↔ **MongoDB** (lưu trữ dữ liệu).
* **Node Apollo Server** ↔ **AWS S3** (lưu trữ video, trả về URL).
* Không có **API Gateway**, **Google Auth**, hoặc **Email Service**.

**8. Kết luận**

Hệ thống đã được đơn giản hóa, tập trung vào giao tiếp trực tiếp giữa client và server qua GraphQL. Video được lưu trên AWS S3, và server chỉ trả về URL để client truy cập trực tiếp. Hệ thống vẫn hỗ trợ các tính năng thời gian thực qua WebSocket và lưu trữ dữ liệu hiệu quả với MongoDB, phù hợp cho một nền tảng chia sẻ video.