VRR

최종결과 발표 및 시연

VR rhythm game with Beat Note automatic generation service using deep learning

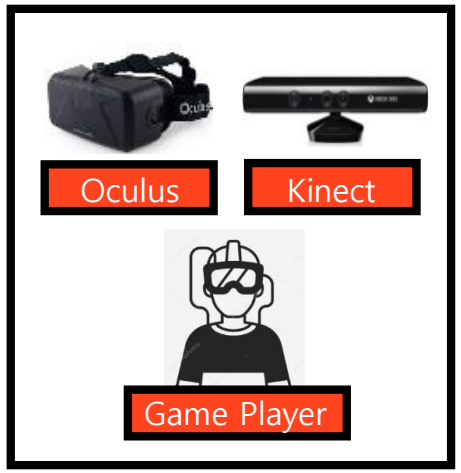
팀 장 박영준
팀 원 문명기
팀 원 김세진
팀 원 이호찬
팀 원 조동철

목차

01 개요

사용자 요구 사항 상세 설계

VR Game Player



1 VR Game Play

Unity로 기능 구현

2 점수, 랭킹, 음악 노트 확인

웹 플랫폼 & 서버 구현

VR Game Player

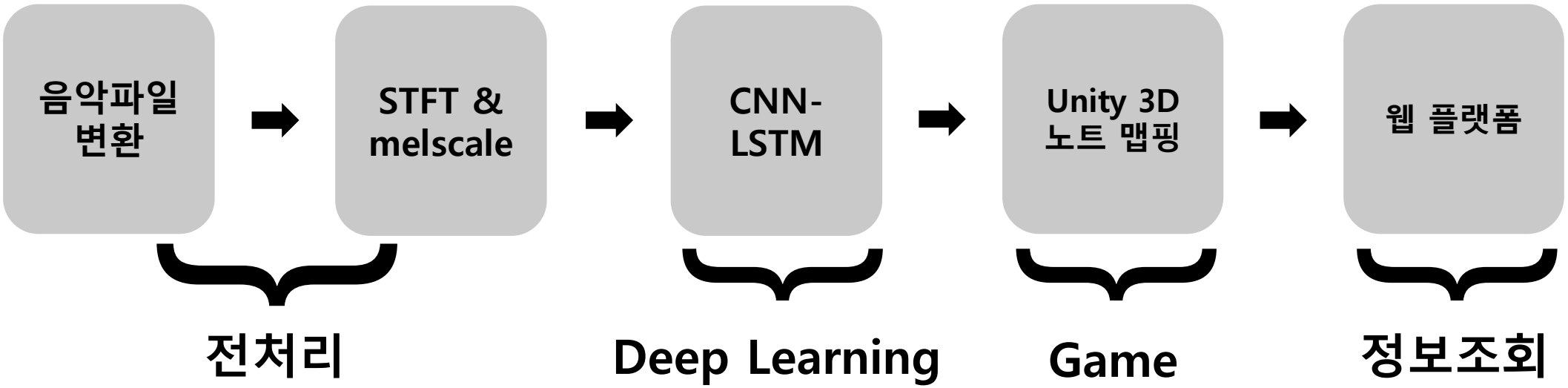
3 비트 노트 맵 제작

음악파일 → 비트 노트 맵

1) 음악파일 전처리
2) Deep Learning Model 적용
3) Unity UI 노트 배치

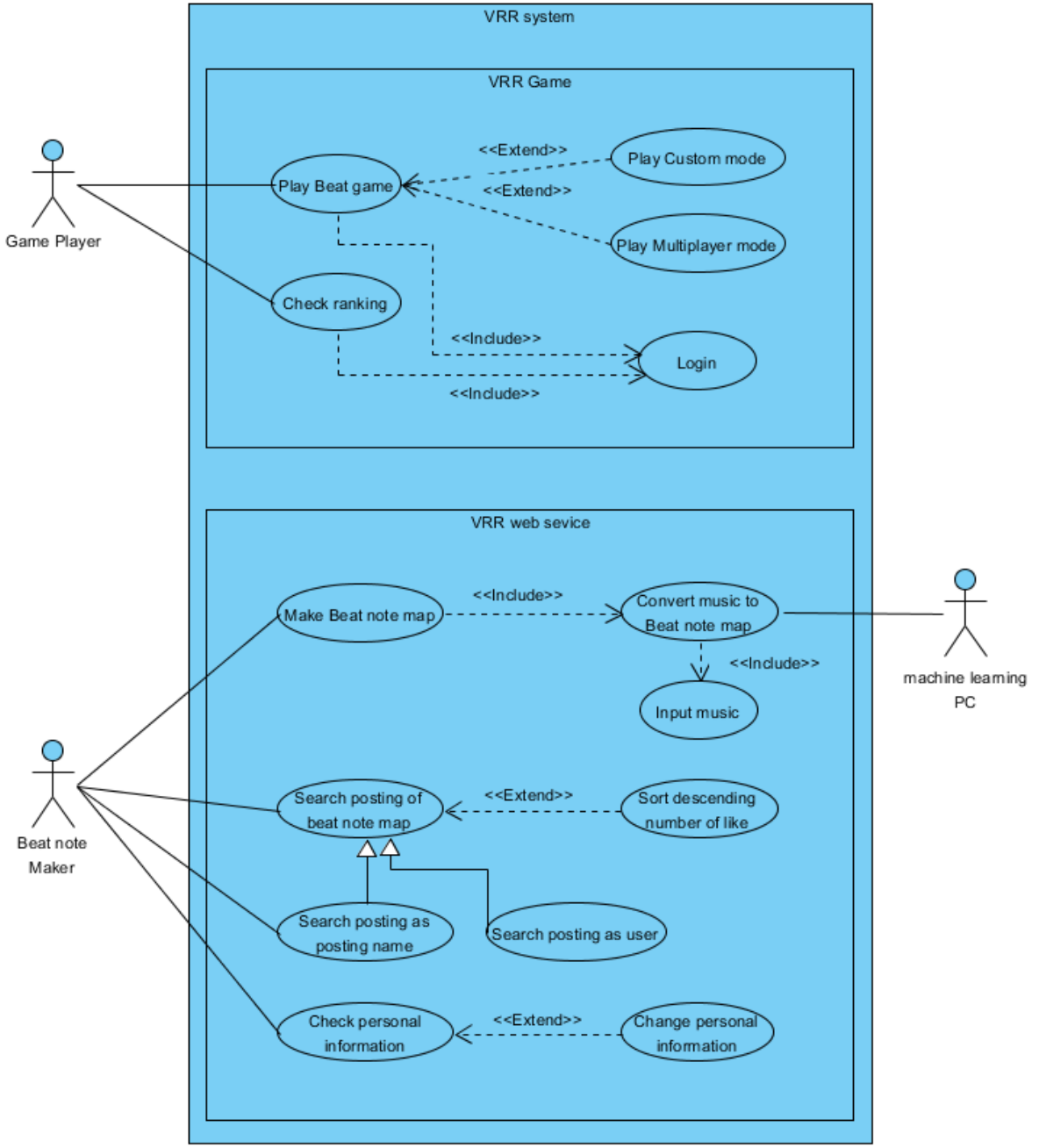
01 개요

주요 요소 기술

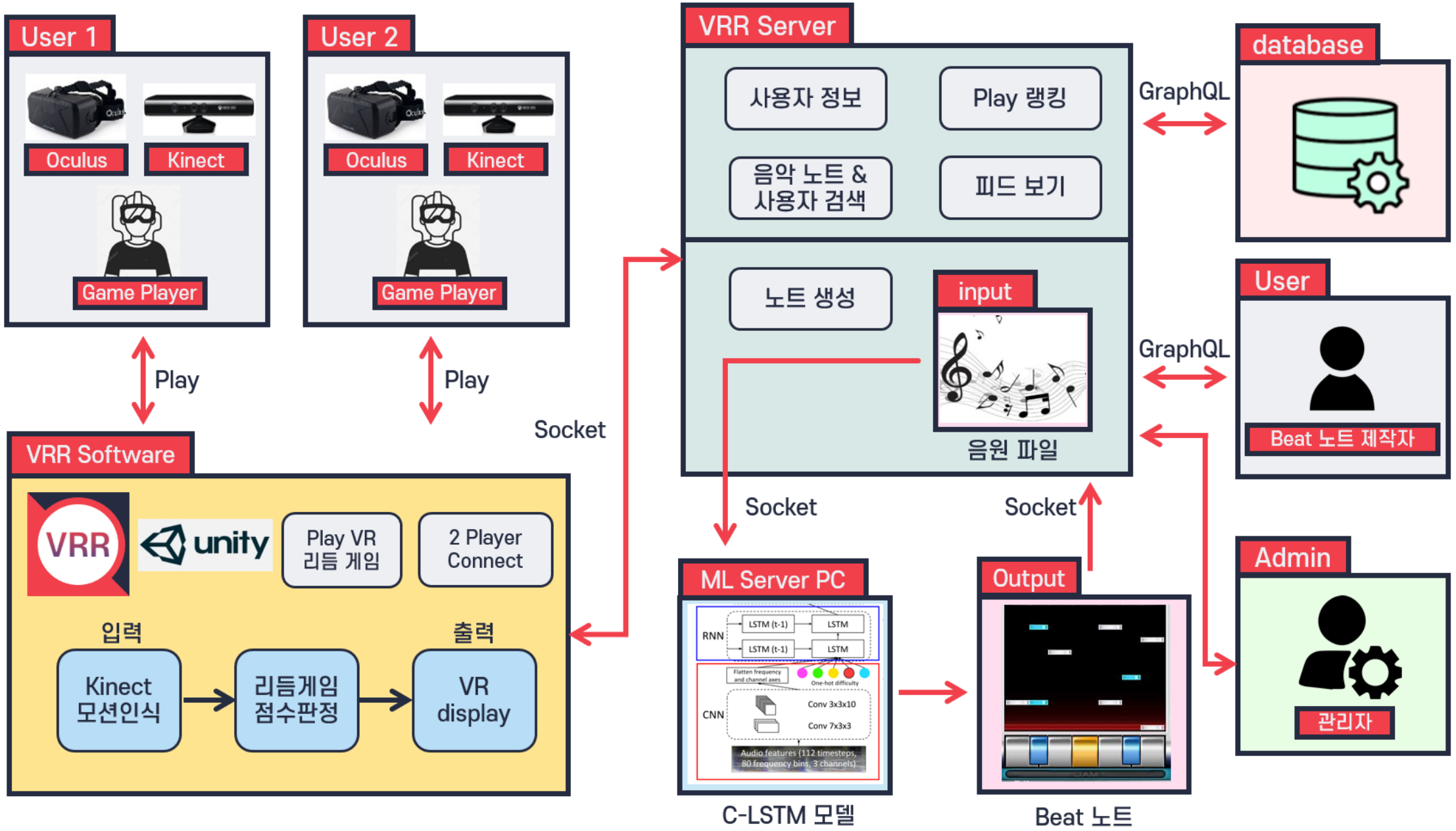

음악파일 변환
STFT & melscale
CNN-LSTM
Unity 3D 노트 맵핑
웹 플랫폼
전처리
Deep Learning
Game
정보조회

01 개요

Use Case Diagram



01 개요


User 1
Oculus
Kinect
Game Player
User 2
Oculus
Kinect
Game Player
Play
Play
VRR Server
사용자 정보
Play 랭킹
음악 노트 & 사용자 검색
피드 보기
노트 생성
input
음원 파일
GraphQL
database
GraphQL
User
Beat 노트 제작자
Socket
VRR Software
VRR
unity
Play VR 리듬 게임
2 Player Connect
입력
출력
Kinect 모션인식
리듬게임 점수판정
VR display
Socket
Socket
ML Server PC
RNN
LSTM (t-1)
LSTM
LSTM (t-1)
LSTM
CNN
Conv 3x3x10
Conv 7x3x3
C-LSTM 모델
Output
Beat 노트
Admin
관리자

02 음원 전처리

전처리 및 Deep Learning 변경사항

- 1채널에 대한 전처리 stft → melscale 에서 3채널에 대한 전처리 stft->melscale->sampling

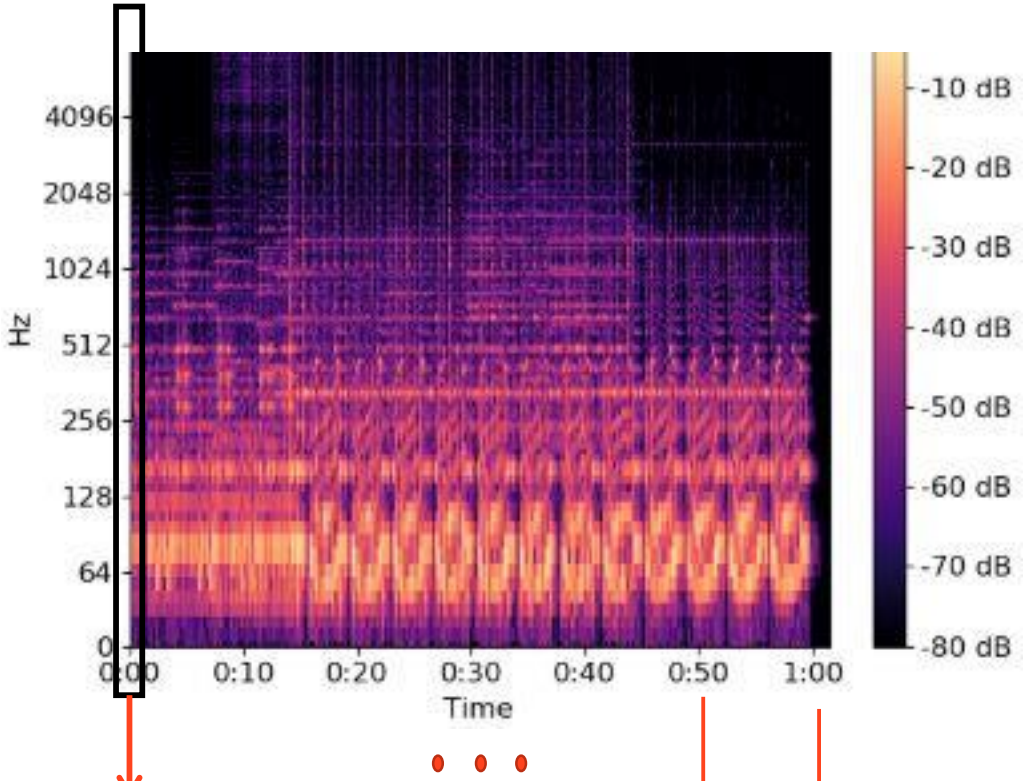
- 1채널 CNN 모델-> 3채널 CNN-LSTM

- 한번에 들어가는 음악파일 크기 조정 20초->10초->2.5초 (비교 중)

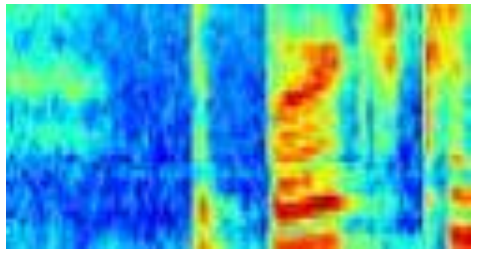
02 음원 전처리

전처리 과정

(1) Stft (None, feature)

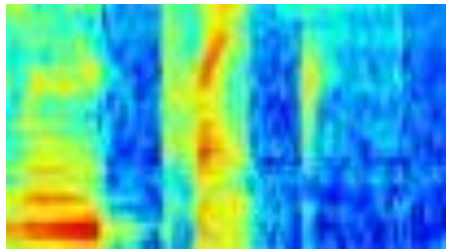


(2) 3.6ms단위로 melscale(1, 80)



(None, 80)

(3) stft,melscale이 완료된 3개의 채널을 합친다



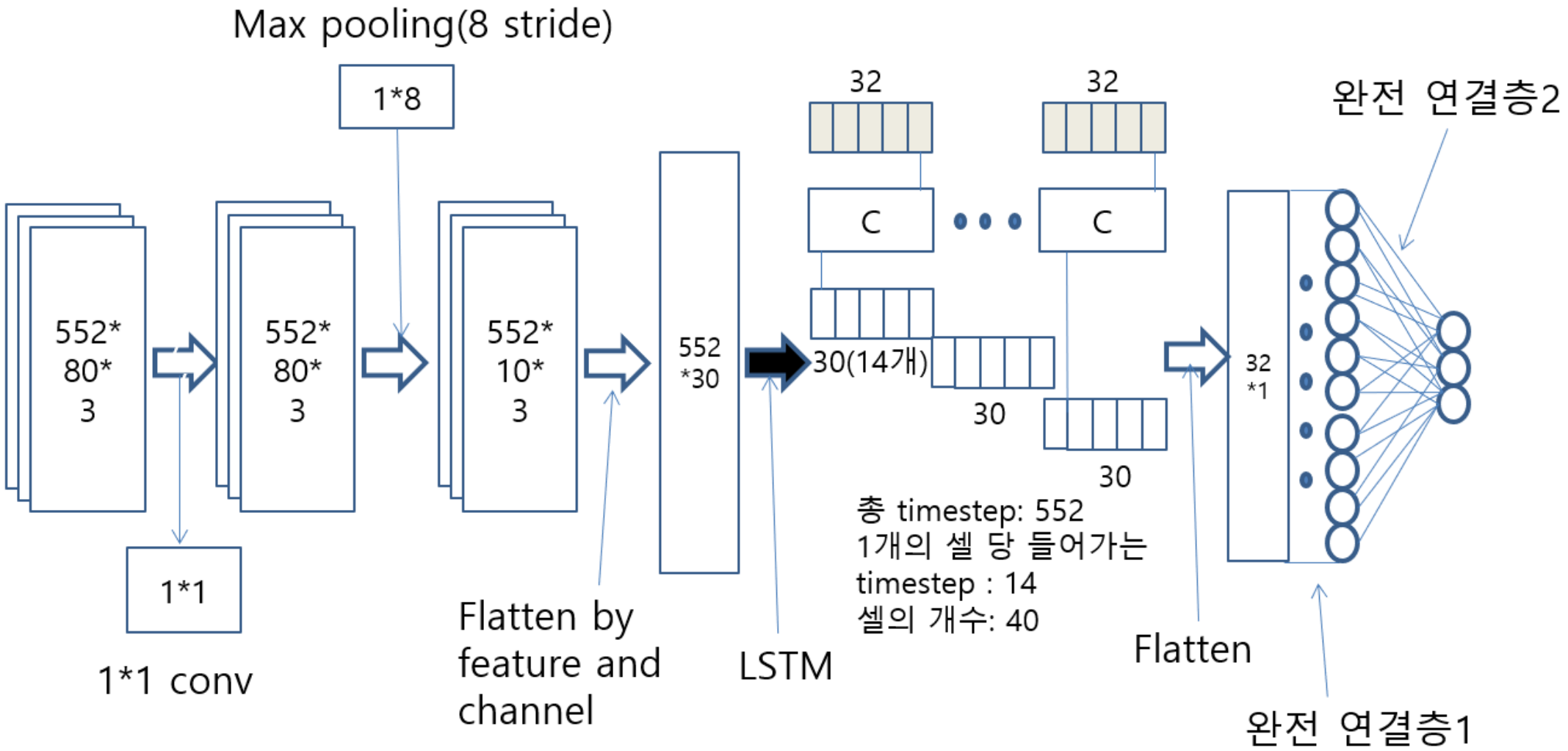
02 음원 전처리

전처리 클래스

- Stft
- Three_channel
  - 23,46,93ms에 대해 stft적용
- Mel_scale
  - Three_channel에서 적용한 값들에 melscale적용
- Preprocess_output
  - Melscale이 끝난 세 개의 채널을 합쳐서 return

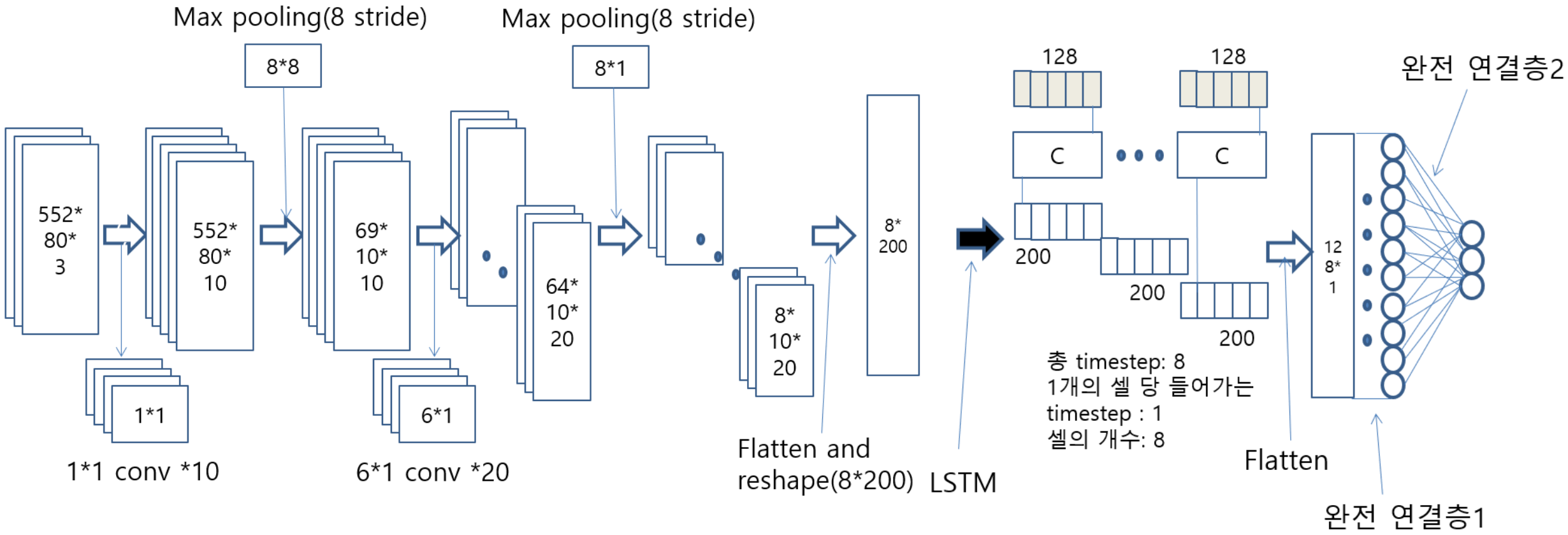
03 CNN_LSTM Model

CNN_LSTM 모델 1



03 CNN_LSTM Model

CNN_LSTM 모델 2



03 CNN_LSTM Model

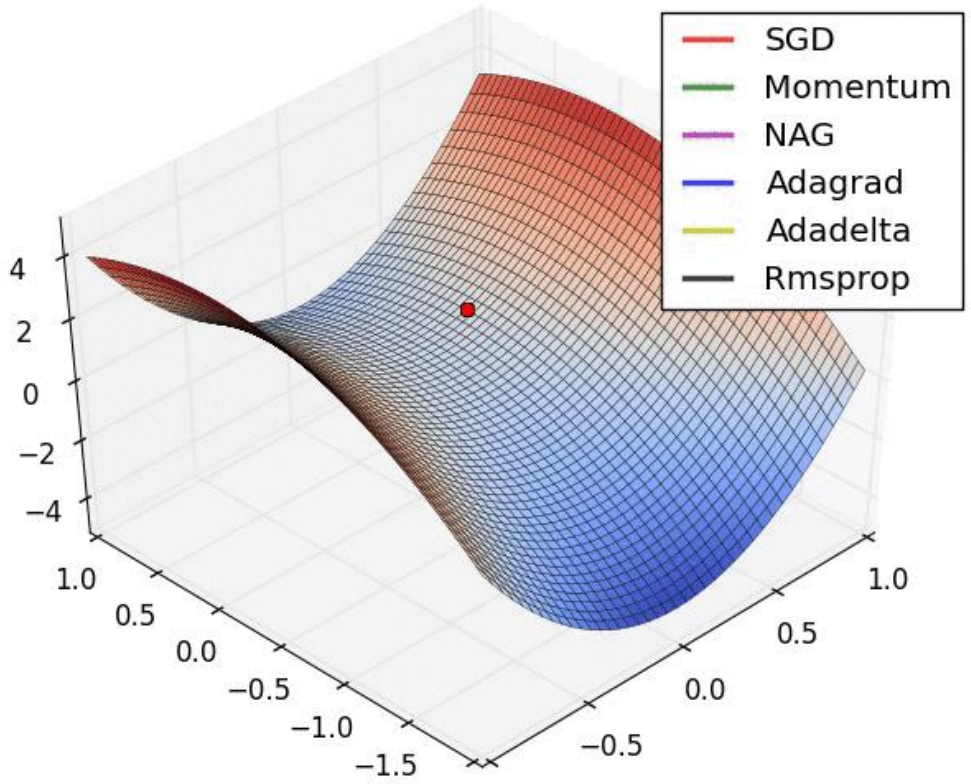
모델의 성능과 일반화 능력 향상을 위한 방법

- 한번에 모델에 들어가는 음악파일의 크기(조정중)
- CNN-LSTM모델 (조정중)
- 최적화 함수
- Dropout 빈도, 시점, 크기
- 스태킹 순환층, 양방향 순환층

03 CNN_LSTM Model

Optimizer

- 목표: 학습속도를 빠르고 안정적으로 만든다.
- Rmsprop
- Adagrad
- Adam



출처 : Gradient Descent Optimization Algorithms

03 CNN_LSTM Model

스태킹 순환층(Stacking recurrent layers)

- 목표: overfitting이 일어날 때 까지 네트워크 용량 늘리기
- 방법: 층의 유닛을 늘리거나 층을 추가한다.
  (고전적인 방법) 향상되는 정확도와 계산 비용의 교차점 파악
- 단점: 계산비용 증가

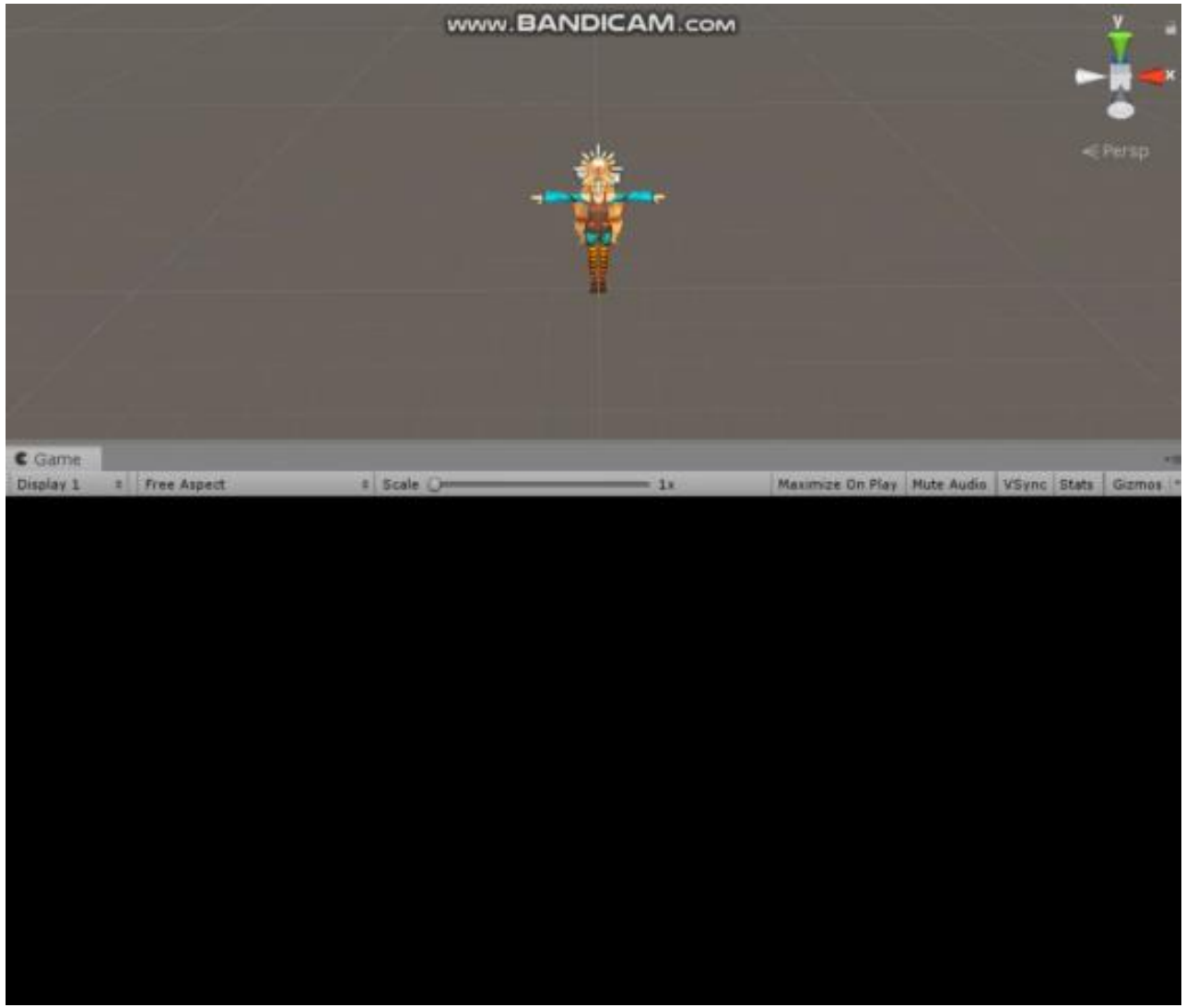
03 CNN_LSTM Model

양방향 순환층(Bidirectional recurrent layers)

- 목표: timestep을 섞거나 방향을 바꾸는 등 학습하는 표현을 변경
- 단점: 박자, 리듬 등을 파악하는 모델에는 적절치 않음

04 Unity Game

Unity VR game 시연 영상

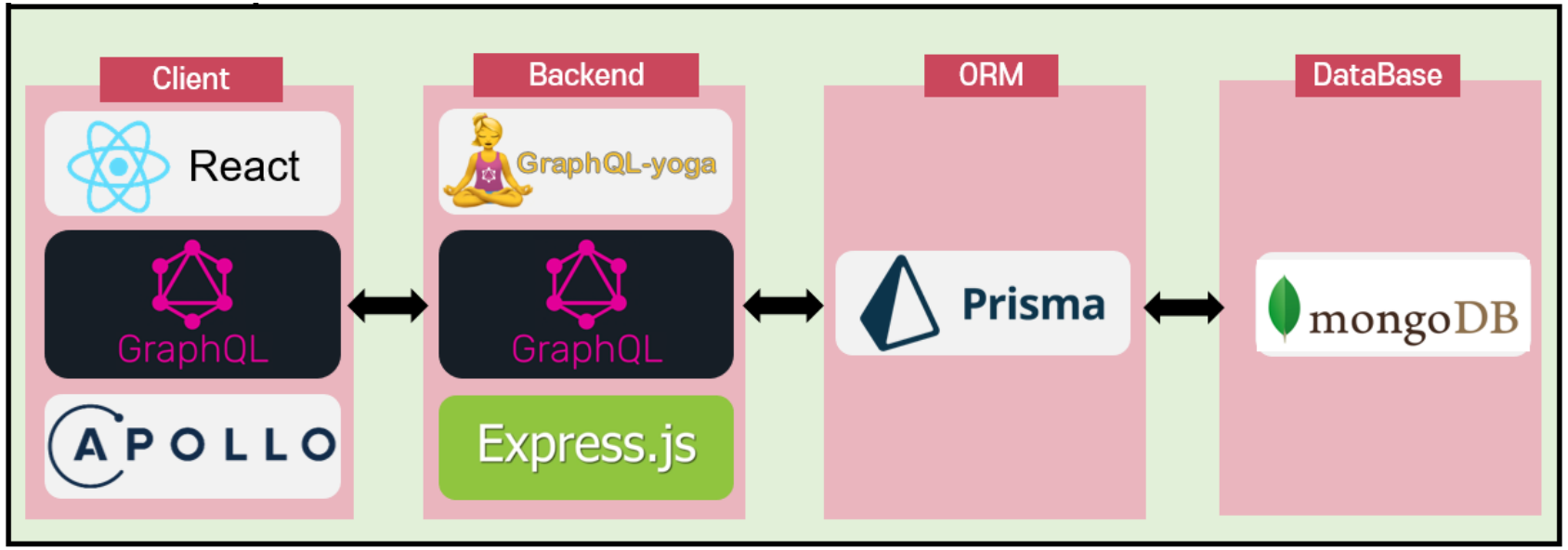


전처리 된 음악 파일 Parsing

```
public class NotesDataParsing
{
    // Get data from path and convert to Json type
    public static JObject getDataNconvert_Json(string path){
        JObject Jdata = new JObject();

        string strData = System.IO.File.ReadAllText(path);
        Jdata = JObject.Parse(strData);

        return Jdata;
    }
```

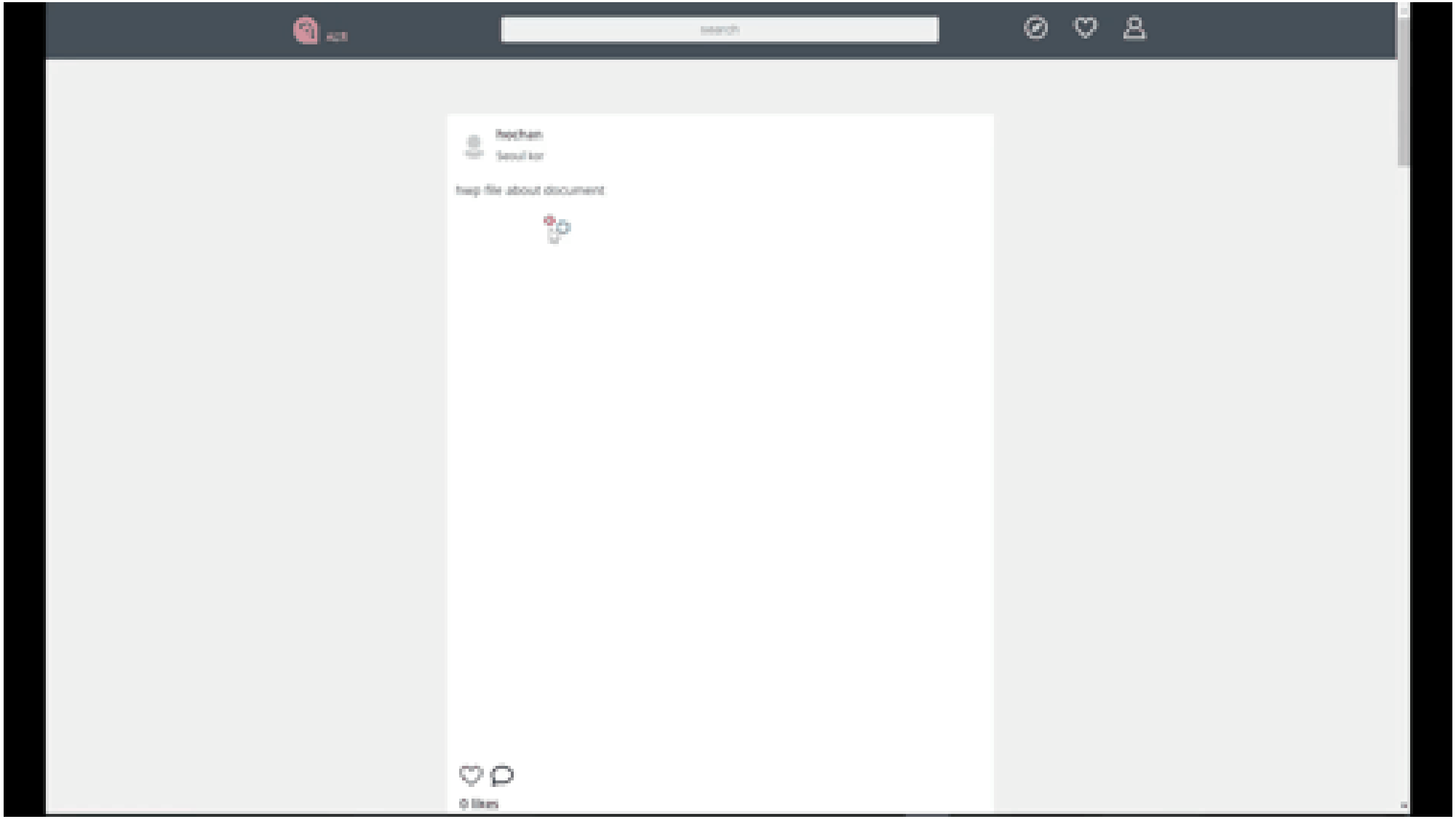
```
// Get each BeatNote data and make notes data list
public static List<BeatNoteClass> parse_data(JObject jdata)
{
    List<BeatNoteClass> data = new List<BeatNoteClass>();


    JArray notes_data = JArray.Parse(jdata["_notes"].ToString());

    foreach(JObject e in notes_data)
    {
        BeatNoteClass temp = new BeatNoteClass();
        temp._time = float.Parse(e["_time"].ToString());
        temp._lineIndex = int.Parse(e["_lineIndex"].ToString());
        temp._lineLayer = int.Parse(e["_lineLayer"].ToString());
        temp._type = int.Parse(e["_type"].ToString());
        temp._cutDirection = int.Parse(e["_cutDirection"].ToString());
        data.Add(temp);
    }

    return data;
}
```

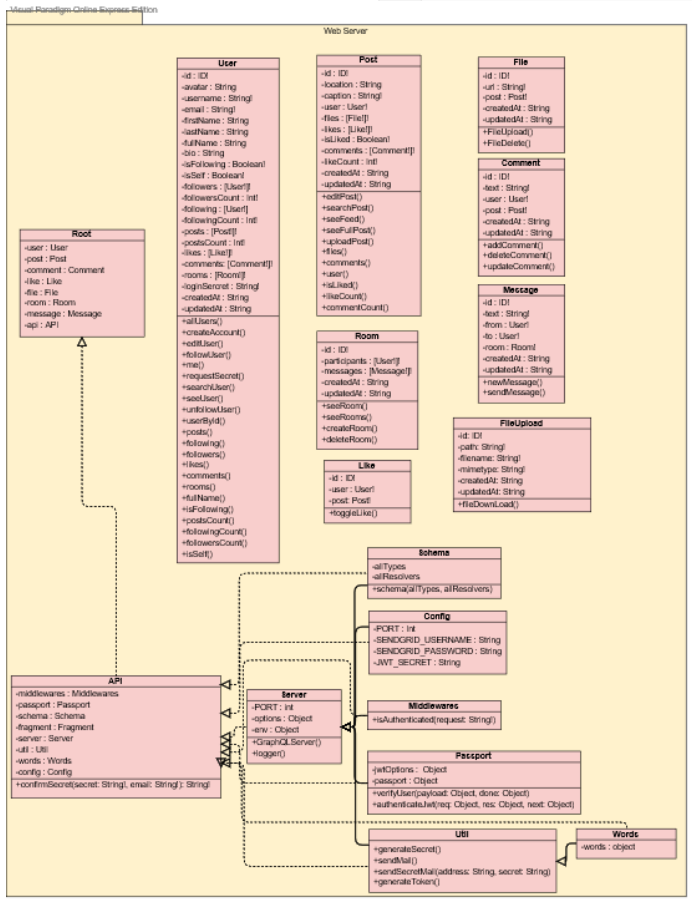
05 Server

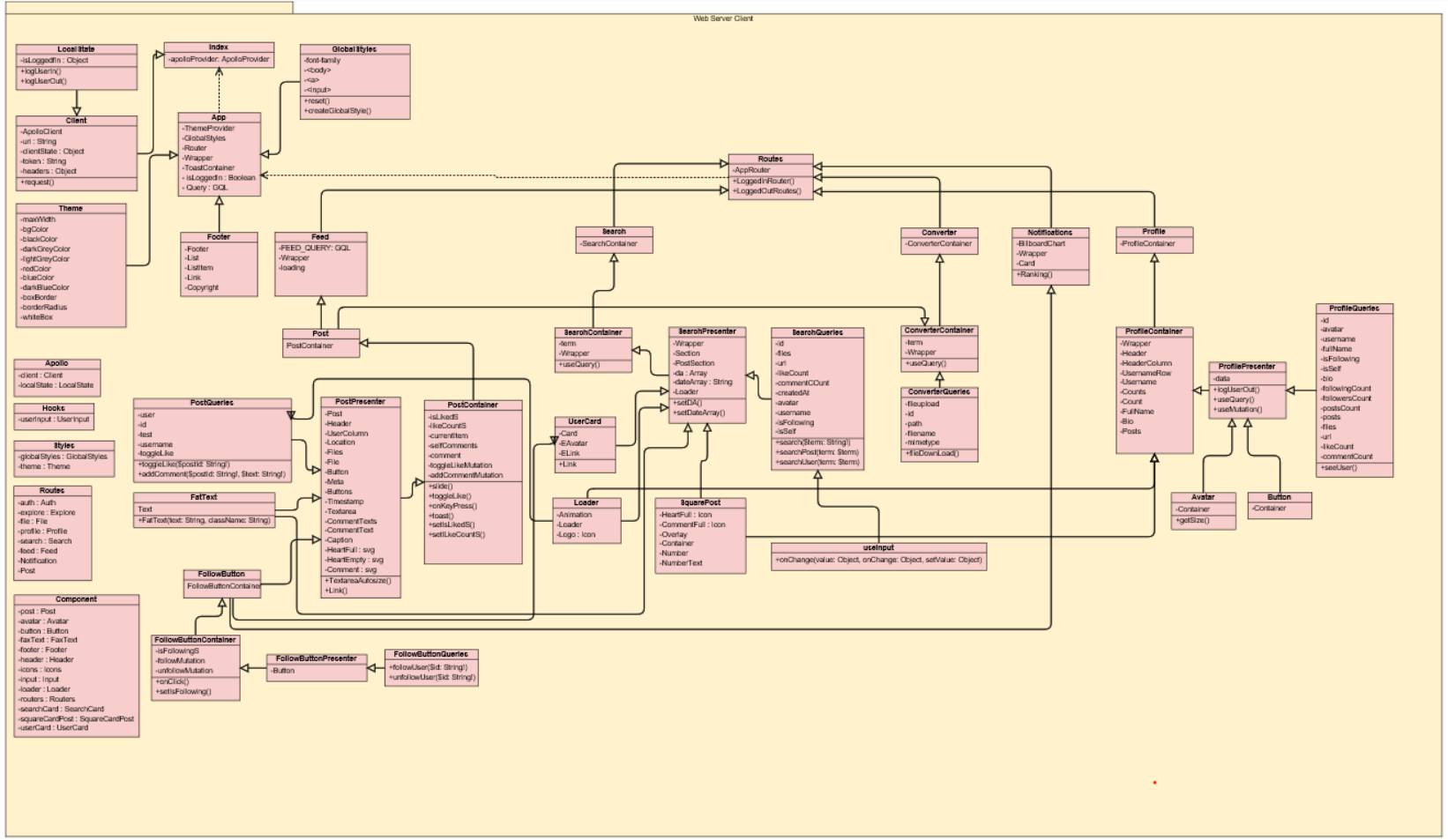


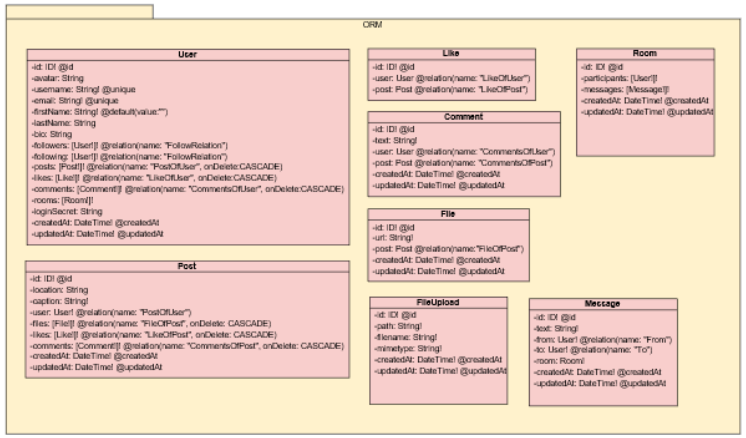
1. ORM
2. Web Server
3. Web Client

VRR 웹 플랫폼 실제 동작 영상

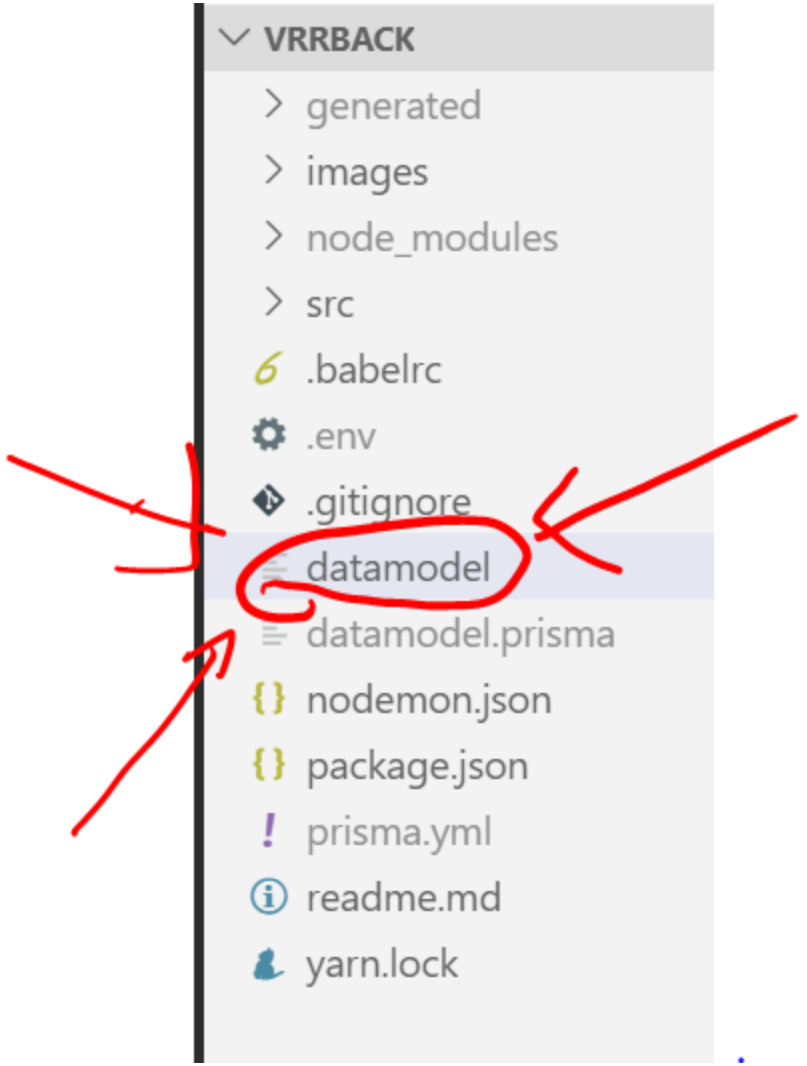


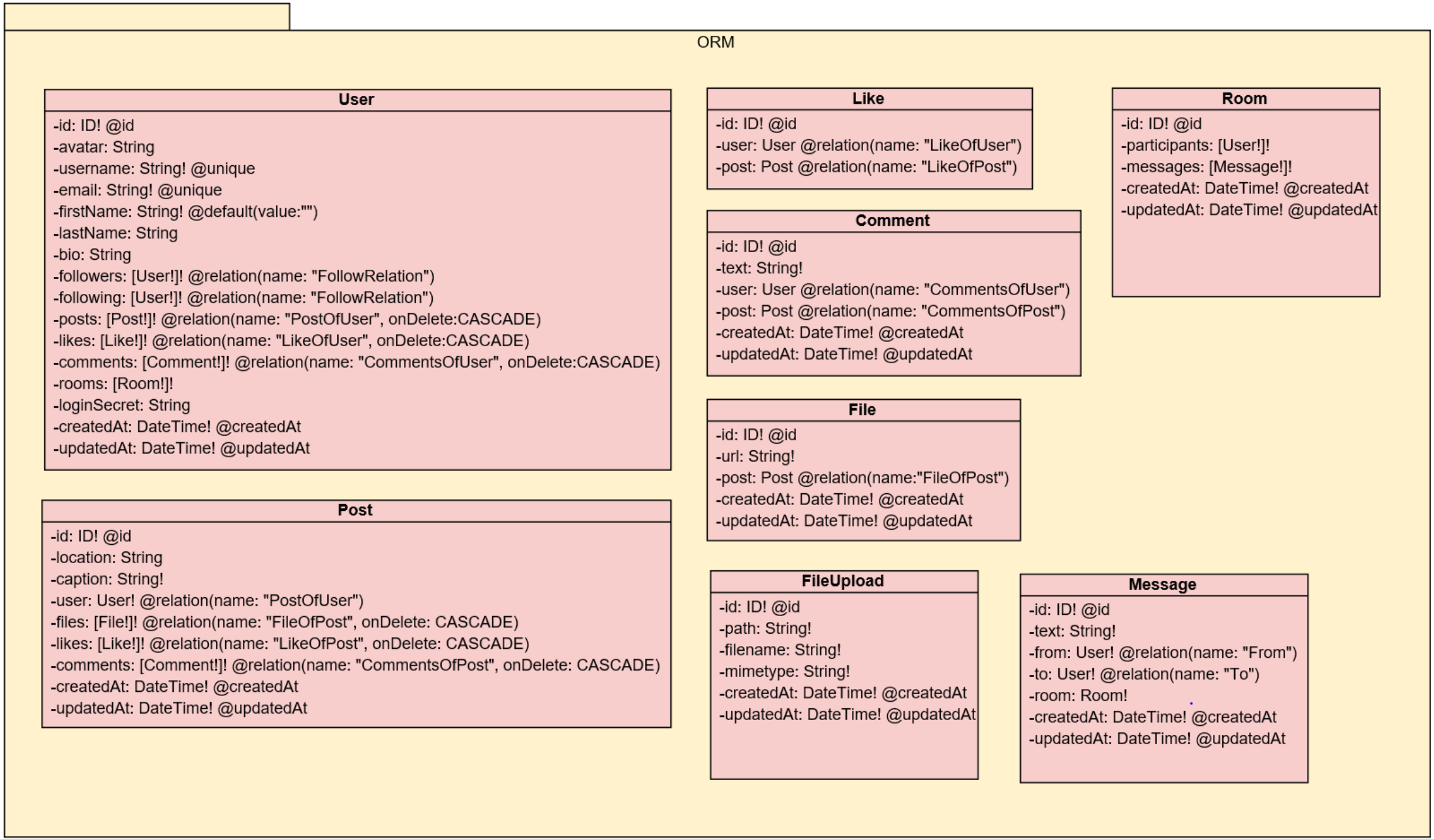






ORM


VRRBACK
generated
images
node_modules
src
.babelrc
.env
.gitignore
datamodel
datamodel.prisma
nodemon.json
package.json
prisma.yml
readme.md
yarn.lock


ORM
User
-id: ID! @id
-avatar: String
-username: String! @unique
-email: String! @unique
-firstName: String! @default(value:"")
-lastName: String
-bio: String
-followers: [User!]! @relation(name: "FollowRelation")
-following: [User!]! @relation(name: "FollowRelation")
-posts: [Post!]! @relation(name: "PostOfUser", onDelete:CASCADE)
-likes: [Like!]! @relation(name: "LikeOfUser", onDelete:CASCADE)
-comments: [Comment!]! @relation(name: "CommentsOfUser", onDelete:CASCADE)
-rooms: [Room!]!
-loginSecret: String
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
Post
-id: ID! @id
-location: String
-caption: String!
-user: User! @relation(name: "PostOfUser")
-files: [File!]! @relation(name: "FileOfPost", onDelete: CASCADE)
-likes: [Like!]! @relation(name: "LikeOfPost", onDelete: CASCADE)
-comments: [Comment!]! @relation(name: "CommentsOfPost", onDelete: CASCADE)
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
Like
-id: ID! @id
-user: User @relation(name: "LikeOfUser")
-post: Post @relation(name: "LikeOfPost")
Comment
-id: ID! @id
-text: String!
-user: User @relation(name: "CommentsOfUser")
-post: Post @relation(name: "CommentsOfPost")
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
File
-id: ID! @id
-url: String!
-post: Post @relation(name:"FileOfPost")
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
FileUpload
-id: ID! @id
-path: String!
-filename: String!
-mimetype: String!
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
Room
-id: ID! @id
-participants: [User!]!
-messages: [Message!]!
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt
Message
-id: ID! @id
-text: String!
-from: User! @relation(name: "From")
-to: User! @relation(name: "To")
-room: Room!
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt

User

-id: ID! @id
-avatar: String
-username: String! @unique
-email: String! @unique
-firstName: String! @default(value:"")
-lastName: String
-bio: String
-followers: [User!]! @relation(name: "FollowRelation")
-following: [User!]! @relation(name: "FollowRelation")
-posts: [Post!]! @relation(name: "PostOfUser", onDelete:CASCADE)
-likes: [Like!]! @relation(name: "LikeOfUser", onDelete:CASCADE)
-comments: [Comment!]! @relation(name: "CommentsOfUser", onDe
-rooms: [Room!]!
-loginSecret: String
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt

```
type User {
  id: ID! @id
  avatar: String
    @default (
      value: "https://icon-library.net/images/default-user-icon/default-user-icon-4.jpg"
    )
  username: String! @unique
  email: String! @unique
  firstName: String! @default(value:"")
  lastName: String
  bio: String
  followers: [User!]! @relation(name: "FollowRelation")
  following: [User!]! @relation(name: "FollowRelation")
  posts: [Post!]! @relation(name: "PostOfUser", onDelete:CASCADE)
  likes: [Like!]! @relation(name: "LikeOfUser", onDelete:CASCADE)
  comments: [Comment!]! @relation(name: "CommentsOfUser", onDelete:CASCADE)
  rooms: [Room!]!
  loginSecret: String
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```

Post

-id: ID! @id
-location: String
-caption: String!
-user: User! @relation(name: "PostOfUser")
-files: [File!]! @relation(name: "FileOfPost", onDelete: CASCADE)
-likes: [Like!]! @relation(name: "LikeOfPost", onDelete: CASCADE)
-comments: [Comment!]! @relation(name: "CommentsOfPost", onDe
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt

```
type Post {
  id: ID! @id
  location: String
  caption: String!
  user: User! @relation(name: "PostOfUser")
  files: [File!]! @relation(name: "FileOfPost", onDelete: CASCADE)
  likes: [Like!]! @relation(name: "LikeOfPost", onDelete: CASCADE)
  comments: [Comment!]! @relation(name: "CommentsOfPost", onDelete: CASCADE)
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```

Like

-id: ID! @id
-user: User @relation(name: "LikeOfUser")
-post: Post @relation(name: "LikeOfPost")

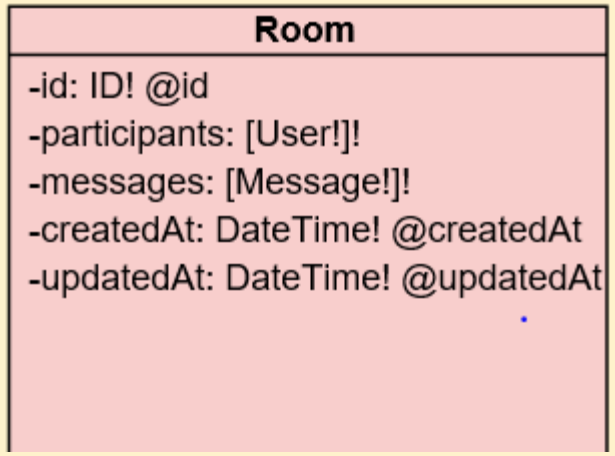
Comment

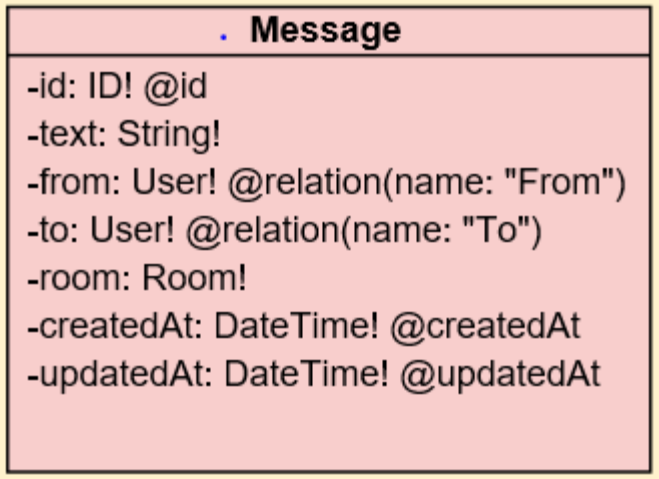
-id: ID! @id
-text: String!
-user: User @relation(name: "CommentsOfUser")
-post: Post @relation(name: "CommentsOfPost")
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt

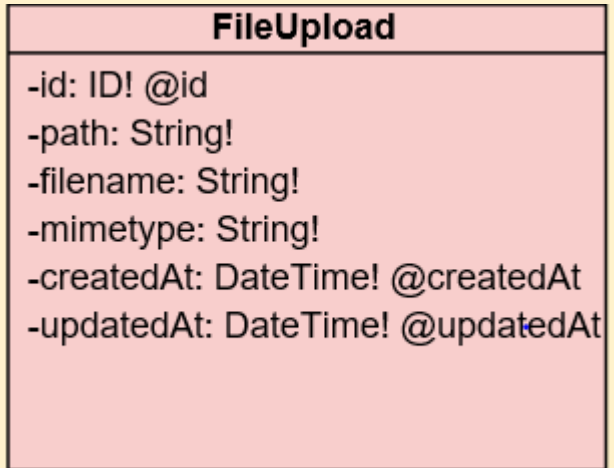
File

-id: ID! @id
-url: String!
-post: Post @relation(name:"FileOfPost")
-createdAt: DateTime! @createdAt
-updatedAt: DateTime! @updatedAt

```
type Like {
  id: ID! @id
  user: User @relation(name: "LikeOfUser")
  post: Post @relation(name: "LikeOfPost")
}

type Comment {
  id: ID! @id
  text: String!
  user: User @relation(name: "CommentsOfUser")
  post: Post @relation(name: "CommentsOfPost")
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}

type File {
  id: ID! @id
  url: String!
  post: Post @relation(name:"FileOfPost")
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```








```
type Room {
  id: ID! @id
  participants: [User!]!
  messages: [Message!]!
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}

type Message {
  id: ID! @id
  text: String!
  from: User! @relation(name: "From")
  to: User! @relation(name: "To")
  room: Room!
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}

type FileUpload {
  id: ID! @id
  path: String!
  filename: String!
  mimetype: String!
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```

Lectures/CAP/2019 - MCLab | Post - Prisma Admin

app.prisma.io/vrr/services/prisma-us1/vrrBack/dev/databrowser

앱 | 한국외국어 대학교... | GraphQL Mutation... | (MongoDB) Mong... | i | GraphQL Schema... | Daily Life | Trello | Microsoft PowerPo... | apollographql/gra... | 아폴로 부스트 마이...

VRR
1 service running

Services | Servers | Settings

H

vrrBack / dev

- Prisma Admin
- Metrics
- Deployment History
- Playground

HTTP ENDPOINT
https://us1.prisma.sh/vrr/vrrBacl

WS ENDPOINT
wss://us1.prisma.sh/vrr/vrrBack/

New Query

Search queries..

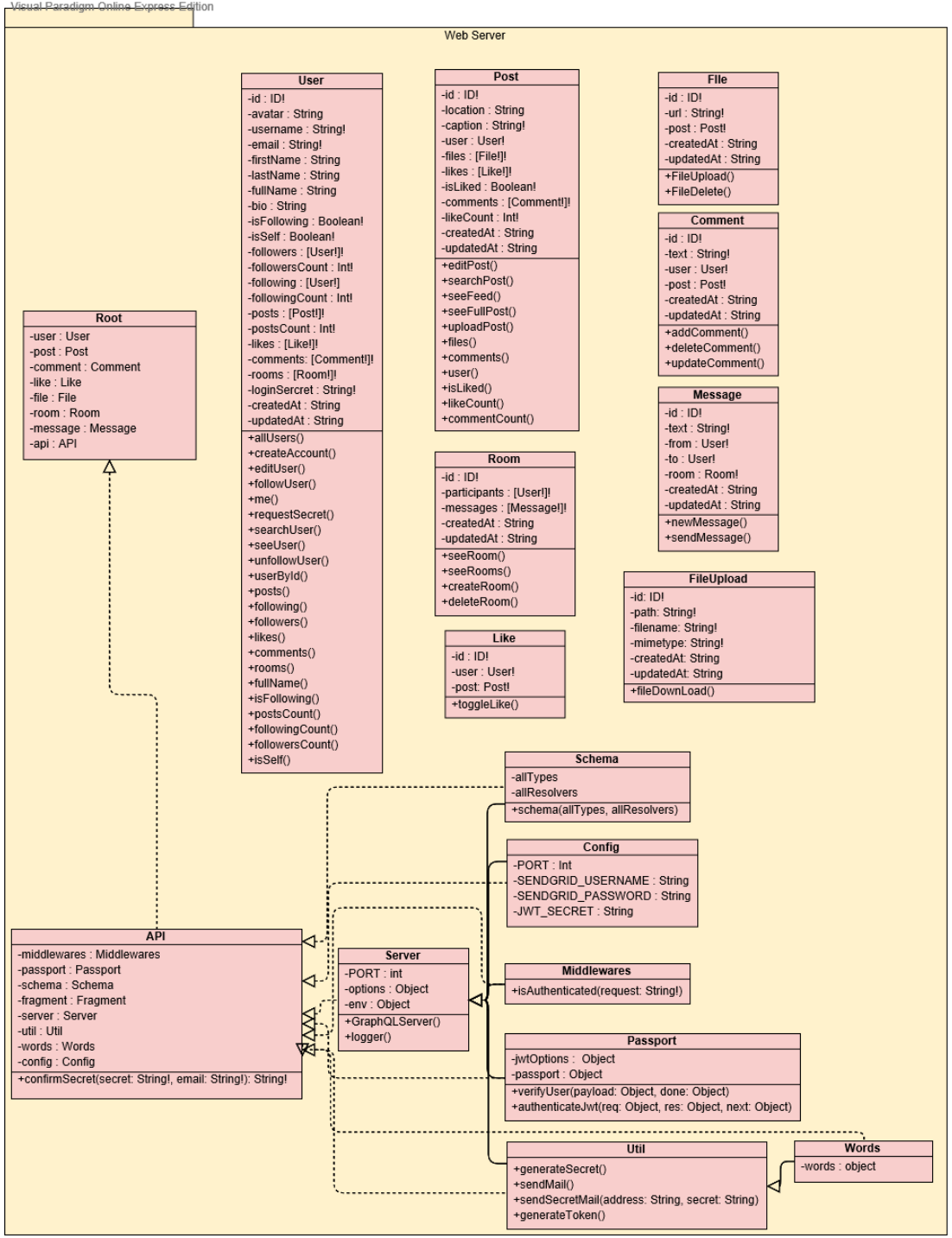
Post *	0
Like	0
Comment	0
File	0
Room	0
Message	0
FileUpload	0
User	0

Post > Rerun

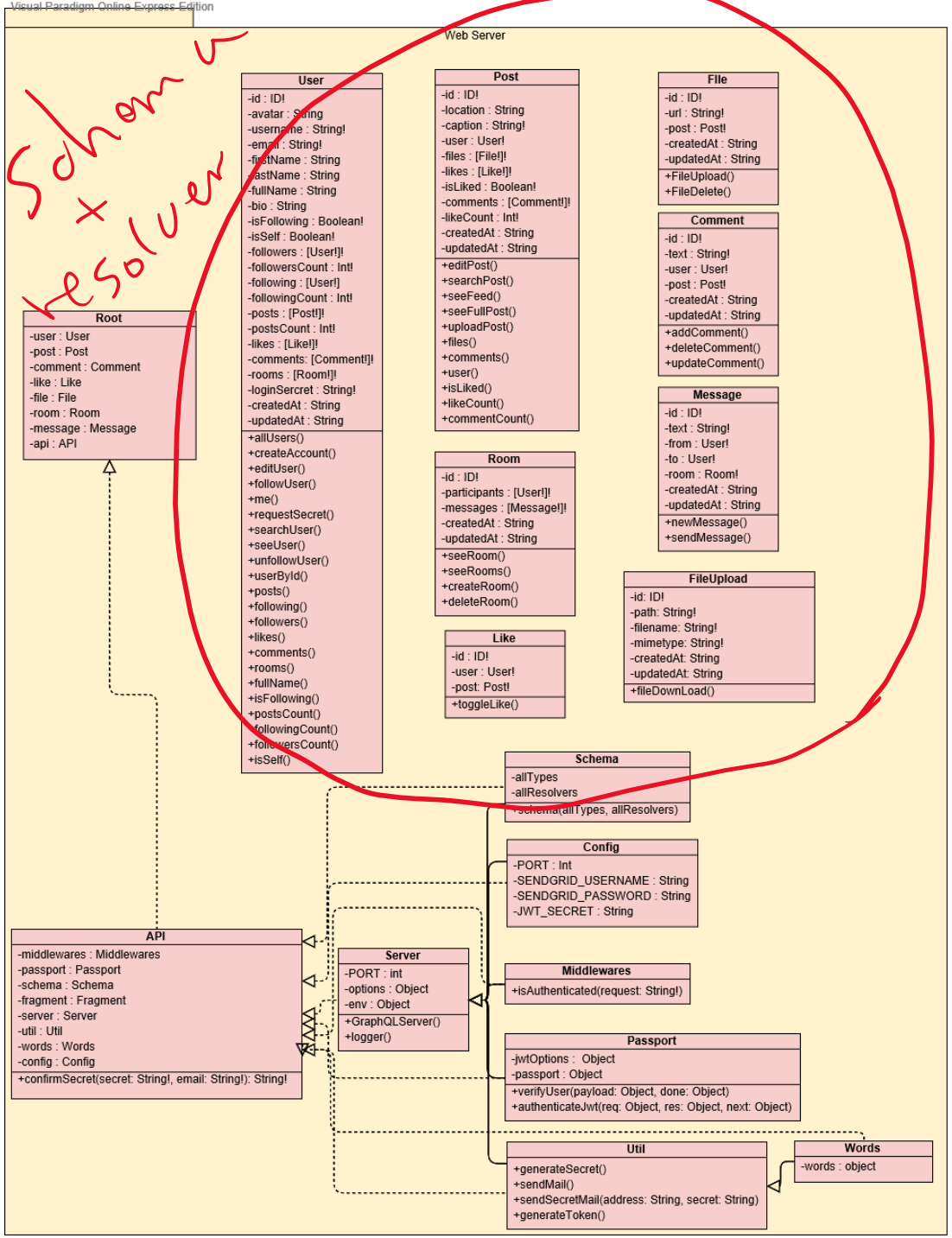
0 Post

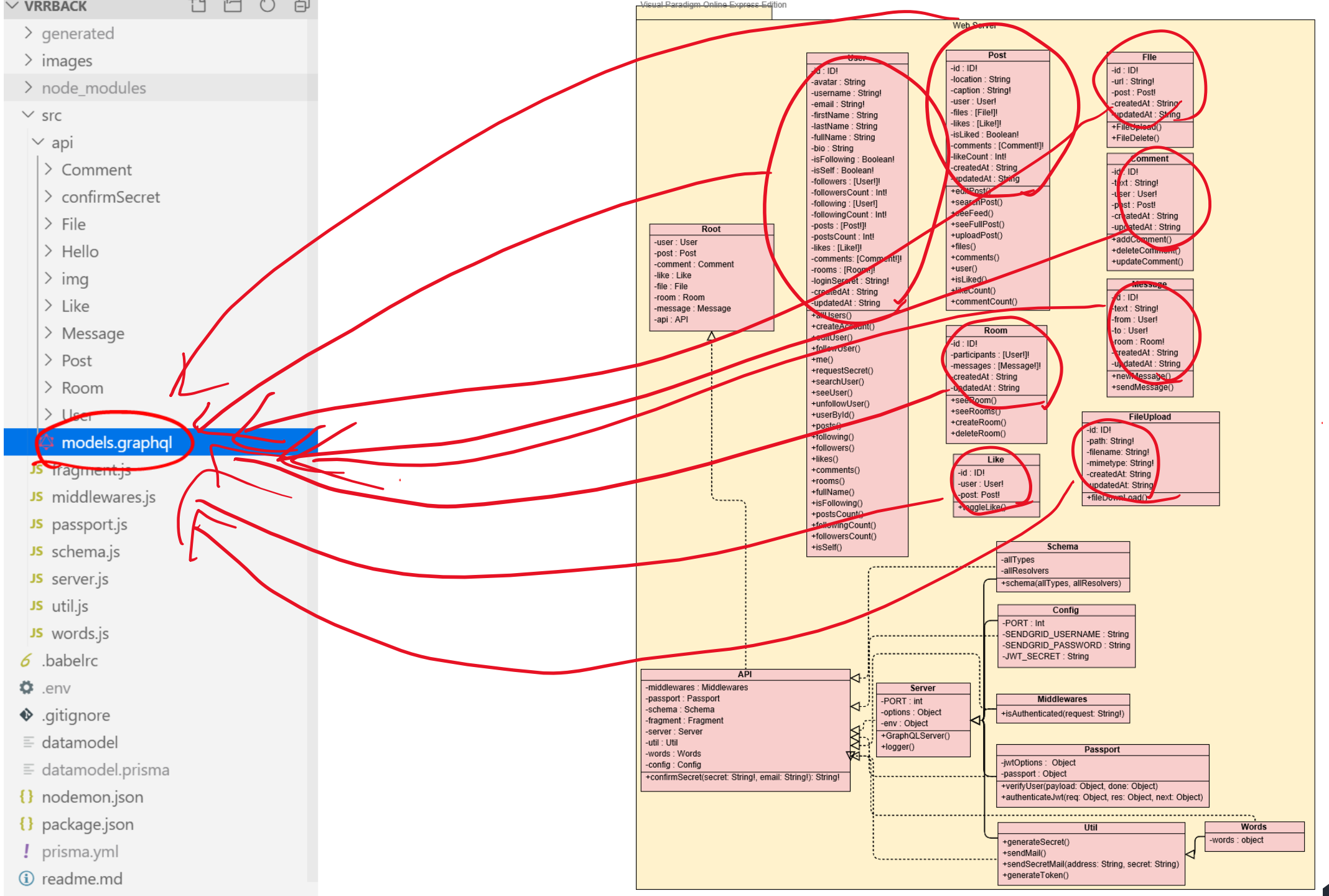
ID!	String	String!	User!
id	location	caption	user

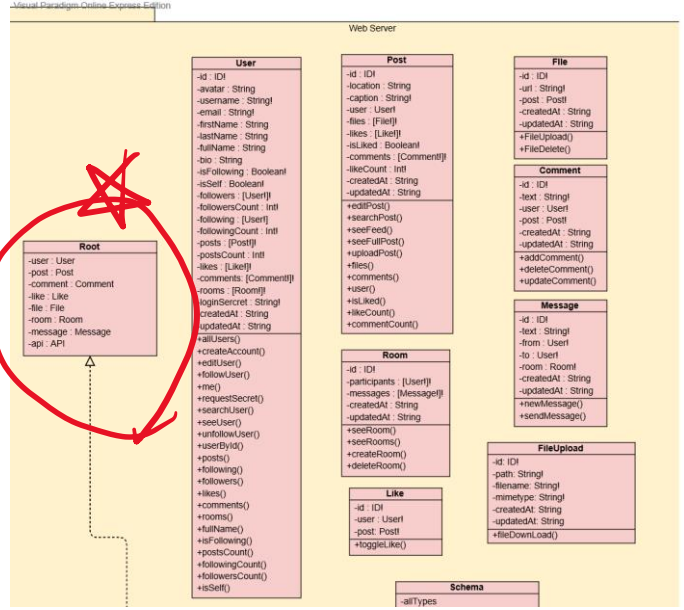
Web Server

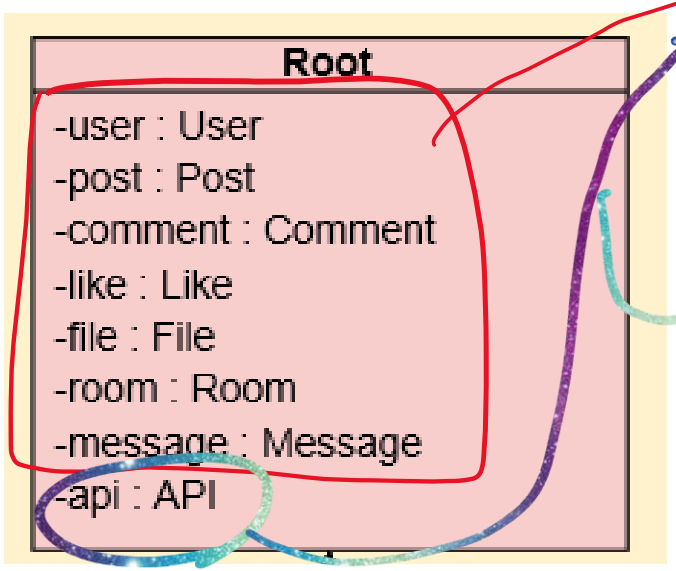

Visual Paradigm Online Express Edition
Web Server
User
-id : ID!
-avatar : String
-username : String!
-email : String!
-firstName : String
-lastName : String
-fullName : String
-bio : String
-isFollowing : Boolean!
-isSelf : Boolean!
-followers : [User!]!
-followersCount : Int!
-following : [User!]
-followingCount : Int!
-posts : [Post!]!
-postsCount : Int!
-likes : [Like!]!
-comments: [Comment!]!
-rooms : [Room!]!
-loginSercret : String!
-createdAt : String
-updatedAt : String
+allUsers()
+createAccount()
+editUser()
+followUser()
+me()
+requestSecret()
+searchUser()
+seeUser()
+unfollowUser()
+userById()
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
Post
-id : ID!
-location : String
-caption : String!
-user : User!
-files : [File!]!
-likes : [Like!]!
-isLiked : Boolean!
-comments : [Comment!]!
-likeCount : Int!
-createdAt : String
-updatedAt : String
+editPost()
+searchPost()
+seeFeed()
+seeFullPost()
+uploadPost()
+files()
+comments()
+user()
+isLiked()
+likeCount()
+commentCount()
FIle
-id : ID!
-url : String!
-post : Post!
-createdAt : String
-updatedAt : String
+FileUpload()
+FileDelete()
Comment
-id : ID!
-text : String!
-user : User!
-post : Post!
-createdAt : String
-updatedAt : String
+addComment()
+deleteComment()
+updateComment()
Message
-id : ID!
-text : String!
-from : User!
-to : User!
-room : Room!
-createdAt : String
-updatedAt : String
+newMessage()
+sendMessage()
Root
-user : User
-post : Post
-comment : Comment
-like : Like
-file : File
-room : Room
-message : Message
-api : API
Room
-id : ID!
-participants : [User!]!
-messages : [Message!]!
-createdAt : String
-updatedAt : String
+seeRoom()
+seeRooms()
+createRoom()
+deleteRoom()
Like
-id : ID!
-user : User!
-post: Post!
+toggleLike()
FileUpload
-id: ID!
-path: String!
-filename: String!
-mimetype: String!
-createdAt: String
-updatedAt: String
+fileDownLoad()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object

Web Server


Visual Paradigm Online Express Edition
Web Server
Schema + Resolver
Root
-user : User
-post : Post
-comment : Comment
-like : Like
-file : File
-room : Room
-message : Message
-api : API
User
-id : ID!
-avatar : String
-username : String!
-email : String!
-firstName : String
-lastName : String
-fullName : String
-bio : String
-isFollowing : Boolean!
-isSelf : Boolean!
-followers : [User!]!
-followersCount : Int!
-following : [User!]
-followingCount : Int!
-posts : [Post!]!
-postsCount : Int!
-likes : [Like!]!
-comments: [Comment!]!
-rooms : [Room!]!
-loginSercret : String!
-createdAt : String
-updatedAt : String
+allUsers()
+createAccount()
+editUser()
+followUser()
+me()
+requestSecret()
+searchUser()
+seeUser()
+unfollowUser()
+userById()
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
Post
-id : ID!
-location : String
-caption : String!
-user : User!
-files : [File!]!
-likes : [Like!]!
-isLiked : Boolean!
-comments : [Comment!]!
-likeCount : Int!
-createdAt : String
-updatedAt : String
+editPost()
+searchPost()
+seeFeed()
+seeFullPost()
+uploadPost()
+files()
+comments()
+user()
+isLiked()
+likeCount()
+commentCount()
Flle
-id : ID!
-url : String!
-post : Post!
-createdAt : String
-updatedAt : String
+FileUpload()
+FileDelete()
Comment
-id : ID!
-text : String!
-user : User!
-post : Post!
-createdAt : String
-updatedAt : String
+addComment()
+deleteComment()
+updateComment()
Message
-id : ID!
-text : String!
-from : User!
-to : User!
-room : Room!
-createdAt : String
-updatedAt : String
+newMessage()
+sendMessage()
Room
-id : ID!
-participants : [User!]!
-messages : [Message!]!
-createdAt : String
-updatedAt : String
+seeRoom()
+seeRooms()
+createRoom()
+deleteRoom()
Like
-id : ID!
-user : User!
-post: Post!
+toggleLike()
FileUpload
-id: ID!
-path: String!
-filename: String!
-mimetype: String!
-createdAt: String
-updatedAt: String
+fileDownLoad()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object


VRRBACK
generated
images
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
server.js
util.js
words.js
.babelrc
.env
.gitignore
datamodel
datamodel.prisma
nodemon.json
package.json
prisma.yml
readme.md
Visual Paradigm Online Express Edition
Web Server
Root
-user : User
-post : Post
-comment : Comment
-like : Like
-file : File
-room : Room
-message : Message
-api : API
User
-id : ID!
-avatar : String
-username : String!
-email : String!
-firstName : String
-lastName : String
-fullName : String
-bio : String
-isFollowing : Boolean!
-isSelf : Boolean!
-followers : [User!]!
-followersCount : Int!
-following : [User!]
-followingCount : Int!
-posts : [Post!]!
-postsCount : Int!
-likes : [Like!]!
-comments: [Comment!]!
-rooms : [Room!]!
-loginSecret : String!
-createdAt : String
-updatedAt : String
+allUsers()
+createAccount()
+editUser()
+followUser()
+me()
+requestSecret()
+searchUser()
+seeUser()
+unfollowUser()
+userById()
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
Post
-id : ID!
-location : String
-caption : String!
-user : User!
-files : [File!]!
-likes : [Like!]!
-isLiked : Boolean!
-comments : [Comment!]!
-likeCount : Int!
-createdAt : String
-updatedAt : String
+editPost()
+searchPost()
+seeFeed()
+seeFullPost()
+uploadPost()
+files()
+comments()
+user()
+isLiked()
+likeCount()
+commentCount()
File
-id : ID!
-url : String!
-post : Post!
-createdAt : String
-updatedAt : String
+FileUpload()
+FileDelete()
Comment
-id : ID!
-text : String!
-user : User!
-post : Post!
-createdAt : String
-updatedAt : String
+addComment()
+deleteComment()
+updateComment()
Message
-id : ID!
-text : String!
-from : User!
-to : User!
-room : Room!
-createdAt : String
-updatedAt : String
+newMessage()
+sendMessage()
Room
-id : ID!
-participants : [User!]!
-messages : [Message!]!
-createdAt : String
-updatedAt : String
+seeRoom()
+seeRooms()
+createRoom()
+deleteRoom()
FileUpload
-id: ID!
-path: String!
-filename: String!
-mimetype: String!
-createdAt: String
-updatedAt: String
+fileDownload()
Like
-id : ID!
-user : User!
-post: Post!
+toggleLike()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object





VRRBACK

- generated
- images
- node_modules
- src
  - api
    - Comment
    - confirmSecret
    - File
    - Hello
    - img
    - Like
    - Message
    - Post
    - Room
    - User
    - models.graphql
    - fragment.js
    - middlewares.js
    - passport.js
    - schema.js
    - server.js
    - util.js
    - words.js
  - .babelrc
  - .env
  - .gitignore
  - datamodel
  - datamodel.prisma
  - nodemon.json
  - package.json
  - prisma.yml
  - readme.md

```
type User {
  id: ID!
  avatar: String
  username: String!
  email: String!
  firstName: String
  lastName: String
  fullName: String
  bio: String
  isFollowing: Boolean!
  isSelf: Boolean!
  followers: [User!]!
  followersCount: Int!
  following: [User!]
  followingCount: Int!
  posts: [Post!]!
  postsCount: Int!
  likes: [Like!]!
  comments: [Comment!]!
  rooms: [Room!]!
  longinSecret: String!
  createdAt: String
  updatedAt: String
}

type Post {
  id: ID!
  location: String
  caption: String!
  user: User!
  files: [File!]!
  likes: [Like!]!
  isLiked: Boolean!
  comments: [Comment!]!
  likeCount: Int!
  commentCount: Int!
  createdAt: String
  updatedAt: String
}
```

ORM

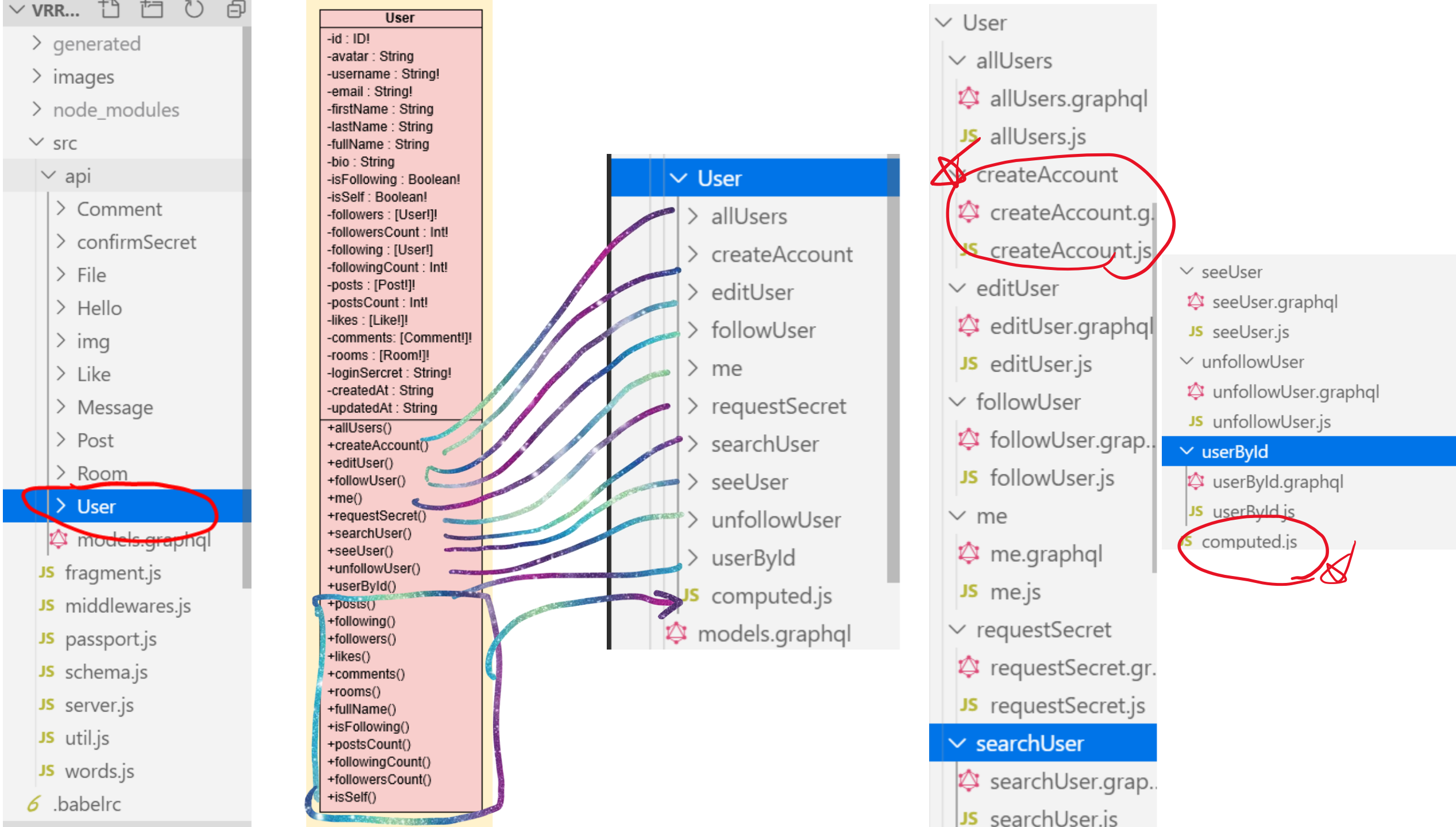
```
type User {
  id: ID! @id
  avatar: String
    @default (
      value: "https://icon-library.net/images/default-user-icon/default-user-icon-4.jpg"
    )
  username: String! @unique
  email: String! @unique
  firstName: String! @default(value:"")
  lastName: String
  bio: String
  followers: [User!]! @relation(name: "FollowRelation")
  following: [User!]! @relation(name: "FollowRelation")
  posts: [Post!]! @relation(name: "PostOfUser", onDelete:CASCADE)
  likes: [Like!]! @relation(name: "LikeOfUser", onDelete:CASCADE)
  comments: [Comment!]! @relation(name: "CommentsOfUser", onDelete:CASCADE)
  rooms: [Room!]!
  loginSecret: String
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```

```
type Post {
  id: ID! @id
  location: String
  caption: String!
  user: User! @relation(name: "PostOfUser")
  files: [File!]! @relation(name: "FileOfPost", onDelete: CASCADE)
  likes: [Like!]! @relation(name: "LikeOfPost", onDelete: CASCADE)
  comments: [Comment!]! @relation(name: "CommentsOfPost", onDelete: CASCADE)
  createdAt: DateTime! @createdAt
  updatedAt: DateTime! @updatedAt
}
```

model.graphql

```
type User {
  id: ID!
  avatar: String
  username: String!
  email: String!
  firstName: String
  lastName: String
  fullName: String
  bio: String
  isFollowing: Boolean!
  isSelf: Boolean!
  followers: [User!]!
  followersCount: Int!
  following: [User!]
  followingCount: Int!
  posts: [Post!]!
  postsCount: Int!
  likes: [Like!]!
  comments: [Comment!]!
  rooms: [Room!]!
  longinSecret: String!
  createdAt: String
  updatedAt: String
}
```

```
type Post {
  id: ID!
  location: String
  caption: String!
  user: User!
  files: [File!]!
  likes: [Like!]!
  isLiked: Boolean!
  comments: [Comment!]!
  likeCount: Int!
  commentCount: Int!
  createdAt: String
  updatedAt: String
}
```

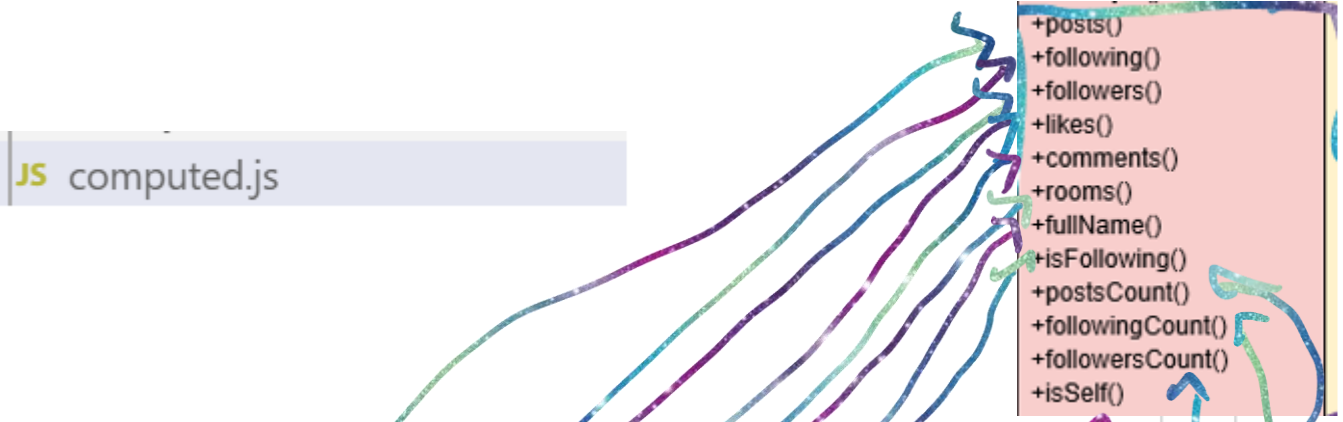

VRR...
generated
images
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
server.js
util.js
words.js
.babelrc
User
-id : ID!
-avatar : String
-username : String!
-email : String!
-firstName : String
-lastName : String
-fullName : String
-bio : String
-isFollowing : Boolean!
-isSelf : Boolean!
-followers : [User!]!
-followersCount : Int!
-following : [User!]
-followingCount : Int!
-posts : [Post!]!
-postsCount : Int!
-likes : [Like!]!
-comments: [Comment!]!
-rooms : [Room!]!
-loginSercret : String!
-createdAt : String
-updatedAt : String
+allUsers()
+createAccount()
+editUser()
+followUser()
+me()
+requestSecret()
+searchUser()
+seeUser()
+unfollowUser()
+userById()
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
User
allUsers
createAccount
editUser
followUser
me
requestSecret
searchUser
seeUser
unfollowUser
userById
computed.js
models.graphql
User
allUsers
allUsers.graphql
allUsers.js
createAccount
createAccount.g.
createAccount.js
editUser
editUser.graphql
editUser.js
followUser
followUser.grap..
followUser.js
me
me.graphql
me.js
requestSecret
requestSecret.gr.
requestSecret.js
searchUser
searchUser.grap..
searchUser.js
seeUser
seeUser.graphql
seeUser.js
unfollowUser
unfollowUser.graphql
unfollowUser.js
userById
userById.graphql
userById.js
computed.js

createAccount.graphql

```graphql
type Mutation {
    createAccount(
        username: String!,
        email: String!,
        firstName: String,
        lastName: String,
        bio: String
    ): Boolean!
}
```

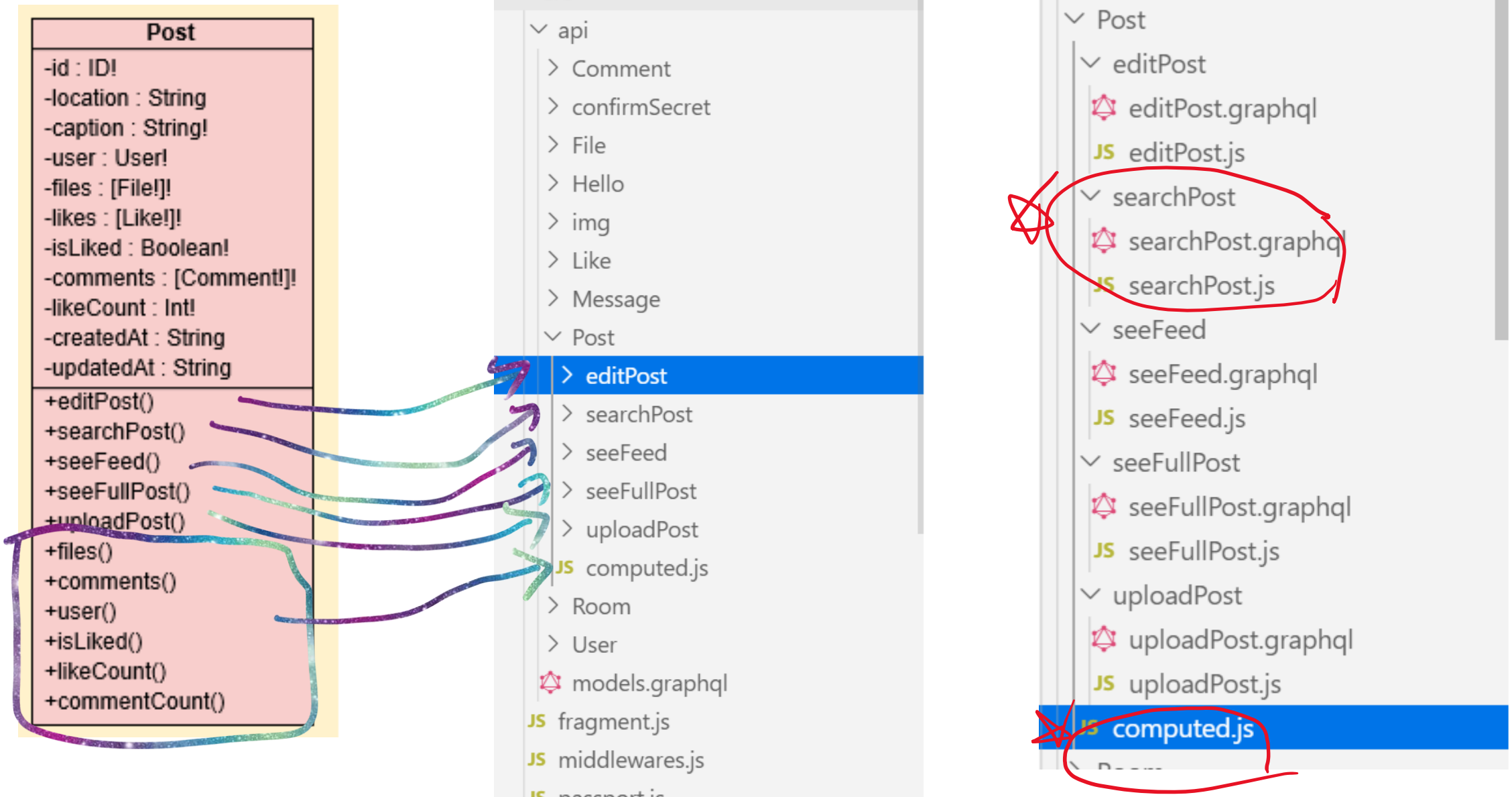
JS createAccount.js

```javascript
import { prisma } from "../../../../generated/prisma-client";

export default {
    Mutation: {
        createAccount: async(_, args) => {
            const { username, email, firstName = "", lastName, bio } = args;
            const exists = await prisma.$exists.user({
                OR: [
                    {
                        username
                    },
                    { email }
                ]
            })
            if (exists){
                throw Error("There username / email is already taken")
            }
            await prisma.createUser({
                username,
                email,
                firstName,
                lastName,
                bio
            })
            return true
        }
    }
}
```



JS computed.js

```
import { prisma } from "../../../generated/prisma-client"

export default {
    User: {
        posts: ({id}) => prisma.user({ id }).posts(),
        following: ({id}) => prisma.user({ id }).following,
        followers: ({id}) => prisma.user({ id }).followers,
        likes: ({id}) => prisma.user({ id }).likes(),
        comments: ({id}) => prisma.user({ id }).comments(),
        rooms: ({id}) => prisma.user({id}).rooms(),
        fullName: (parent) => {
            return `${parent.firstName} ${parent.lastName}`
        },
        isFollowing: async (parent, args, {request}) => {
            const { user } = request
            const { id: parentId } = parent
            const exist = await prisma.$exists.user({
                AND: [{id: parentId}, {followers_some: {id: user.id}}
            })
            try {
                return exist
            } catch (error) {
                console.log(error)
                return false
            }
        },
        postsCount: ({ id }) =>
            prisma
            .postsConnection({ where: { user: { id } } })
            .aggregate()
            .count(),
        followingCount: ({ id }) =>
            prisma
            .usersConnection({ where: { followers_some: { id } } })
            .aggregate()
            .count(),
        followersCount: ({ id }) =>
            prisma
            .usersConnection({ where: { following_none: { id } } })
            .aggregate()
            .count(),
        isSelf: (parent, _, {request}) => {
            const { user } = request
            const {id: parentId } = parent
            return user.id === parentId
        }
    }
}
```

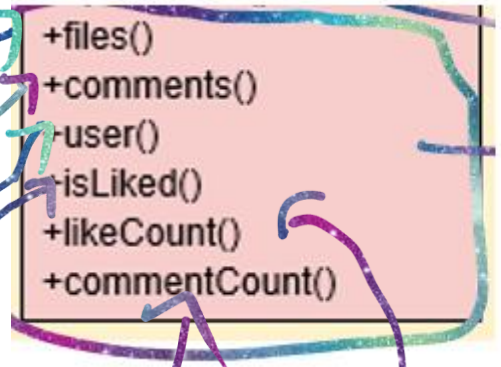

Post
-id : ID!
-location : String
-caption : String!
-user : User!
-files : [File!]!
-likes : [Like!]!
-isLiked : Boolean!
-comments : [Comment!]!
-likeCount : Int!
-createdAt : String
-updatedAt : String
+editPost()
+searchPost()
+seeFeed()
+seeFullPost()
+uploadPost()
+files()
+comments()
+user()
+isLiked()
+likeCount()
+commentCount()
VRRBACK
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
editPost
searchPost
seeFeed
seeFullPost
uploadPost
computed.js
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
Post
editPost
editPost.graphql
editPost.js
searchPost
searchPost.graphql
searchPost.js
seeFeed
seeFeed.graphql
seeFeed.js
seeFullPost
seeFullPost.graphql
seeFullPost.js
uploadPost
uploadPost.graphql
uploadPost.js
computed.js

searchPost.graphql

```
type Query {
    searchPost(term: String!): [Post!]!
}
```

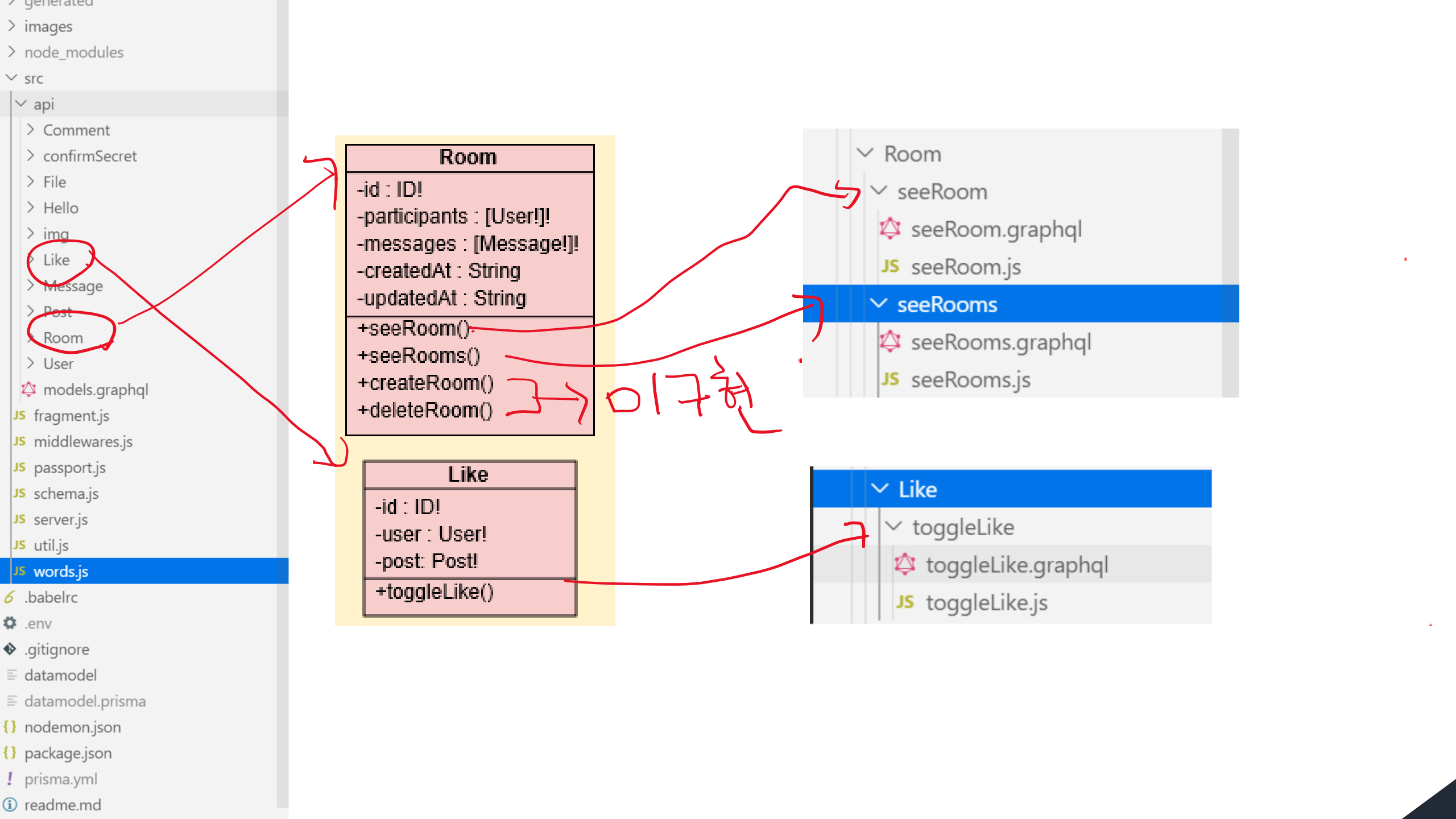
searchPost.js

```
import { prisma } from "../../../../generated/prisma-client";

export default {
    Query: {
        searchPost: async(_, args) => prisma.posts(
            { where:
                {
                    OR: [
                        { location_starts_with: args.term },
                        { caption_starts_with: args.term }
                    ]
                }
            }
        )
    }
}
```



JS computed.js

```
import { prisma } from "../../../generated/prisma-client"

export default {
    Post: {
        files: ({id}) => prisma.post({ id }).files(),
        comments: ({id}) => prisma.post({ id }).comments(),
        user: ({id}) => prisma.post({ id }).user(),

        isLiked: async(parent, _, {request}) => {
            const { user } = request
            const { id } = parent
            return prisma.$exists.like({
                AND: [
                    {
                        user: {
                            id: user.id
                        }
                    },
                    {
                        post: {
                            id
                        }
                    }
                ]
            })
        },

        likeCount: (parent) =>
            prisma
            .likesConnection({
                where: { post: { id: parent.id } }
            })
            .aggregate()
            .count(),
        commentCount: (parent) =>
            prisma
            .commentsConnection({
                where: { post: { id: parent.id } }
            })
            .aggregate()
            .count()
    }
}
```


images
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
server.js
util.js
words.js
.babelrc
.env
.gitignore
datamodel
datamodel.prisma
nodemon.json
package.json
prisma.yml
readme.md
Room
-id : ID!
-participants : [User!]!
-messages : [Message!]!
-createdAt : String
-updatedAt : String
+seeRoom()
+seeRooms()
+createRoom()
+deleteRoom()
Like
-id : ID!
-user : User!
-post: Post!
+toggleLike()
Room
seeRoom
seeRoom.graphql
seeRoom.js
seeRooms
seeRooms.graphql
seeRooms.js
Like
toggleLike
toggleLike.graphql
toggleLike.js

toggleLike.graphql

```graphql
type Mutation {
    toggleLike(postId: String!): Boolean!
}
```

toggleLike.js

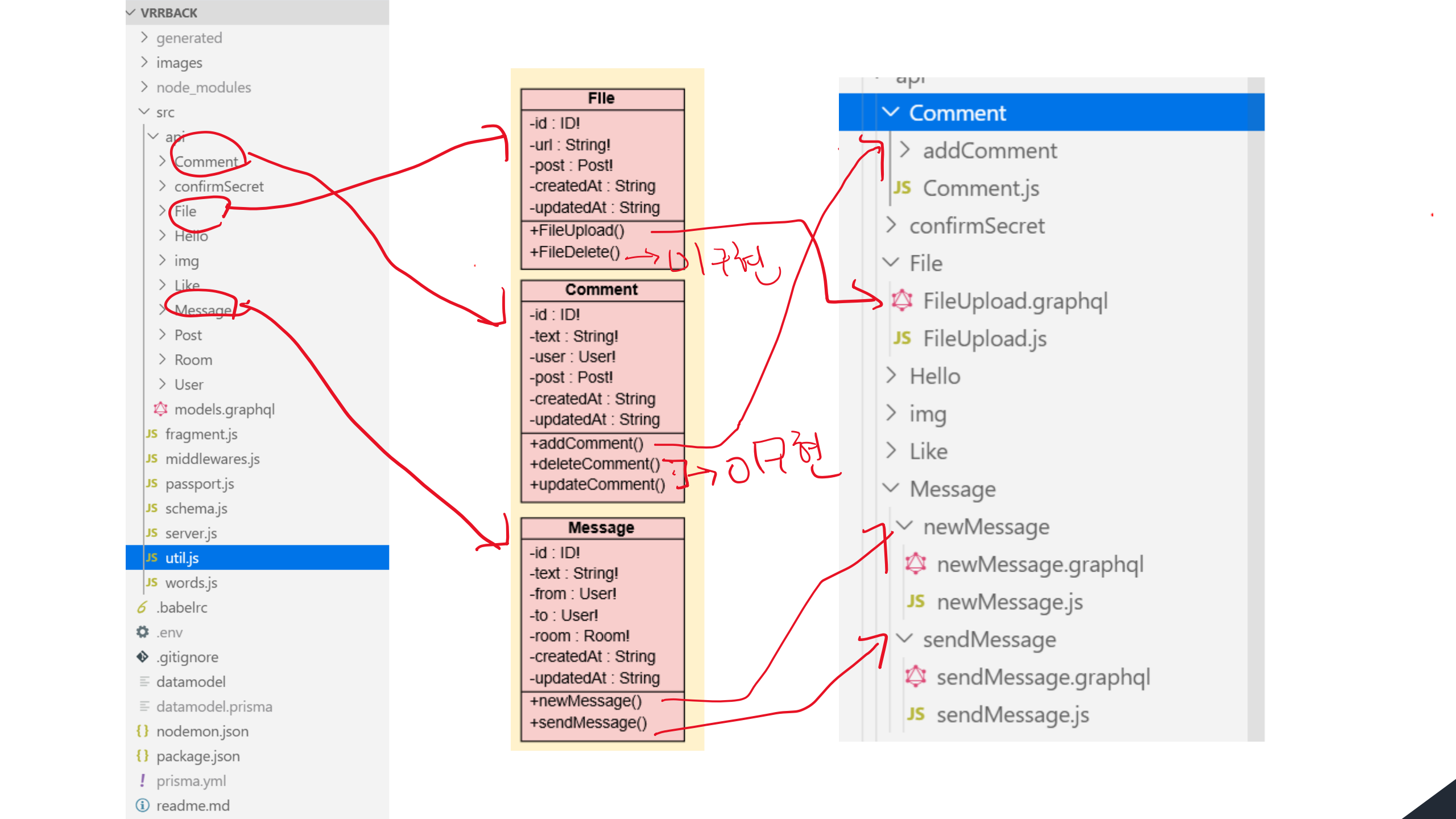
```js
import { isAuthenticated } from "../../../middlewares"
import { prisma } from "../../../../generated/prisma-client"

export default {
    Mutation: {
        toggleLike: async (_, args, { request }) => {
            // passport user identity
            isAuthenticated(request)
            const { postId } = args
            const { user } = request
            const { filterOptions } = {
                AND: [
                    {user:{
                        id: user.id
                    }},
                    {
                        post: {
                            id: postId
                        }
                    }
                ]
            }

            try {
                const existingLike = await prisma.$exists.like(filterOptions)

                if(existingLike){
                    await prisma.deleteManyLikes(filterOptions)
                } else {
                    await prisma.createLike({
                        user: {
                            connect: {
                                id: user.id
                            }
                        },
                        post: {
                            connect: {
                                id: postId
                            }
                        }
                    })
                }
                return true
            } catch (error){
                console.log(error)
                return false
            }
        }
    }
}
```

seeRoom.graphql

```
type Query {
    seeRoom(id: String!): Room!
}
```

seeRoom.js

```
import { prisma } from "../../../../generated/prisma-client"
import { ROOM_FRAGMENT } from "../../../fragment"

export default {
    Query: {
        seeRoom: async (_, args, {request, isAuthenticated}) => {
            isAuthenticated(request)
            const { id } = args
            const { user } = request
            const canSee = await prisma.$exists.room({
                participants_some: {
                    id: user.id
                }
            })
            if(canSee) {
                return prisma.room({ id }).$fragment(ROOM_FRAGMENT)
            } else {
                throw Error("You can't see this")
            }
        }
    }
}
```


VRRBACK
generated
images
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
server.js
util.js
words.js
.babelrc
.env
.gitignore
datamodel
datamodel.prisma
nodemon.json
package.json
prisma.yml
readme.md
File
-id : ID!
-url : String!
-post : Post!
-createdAt : String
-updatedAt : String
+FileUpload()
+FileDelete()
Comment
-id : ID!
-text : String!
-user : User!
-post : Post!
-createdAt : String
-updatedAt : String
+addComment()
+deleteComment()
+updateComment()
Message
-id : ID!
-text : String!
-from : User!
-to : User!
-room : Room!
-createdAt : String
-updatedAt : String
+newMessage()
+sendMessage()
Comment
addComment
Comment.js
confirmSecret
File
FileUpload.graphql
FileUpload.js
Hello
img
Like
Message
newMessage
newMessage.graphql
newMessage.js
sendMessage
sendMessage.graphql
sendMessage.js

FileUpload.graphql

```graphql
scalar Upload
  type Query {
    uploads: [File2]
  }
  type Mutation {
    UploadFile (file: Upload!): Boolean!
    multipleUpload (files: [Upload!]!): [File2!]!
  }
  type File2 {
    id: ID!
    path: String!
    filename: String!
    mimetype: String!
    encoding: String!
  }
```

JS FileUpload.js

```js
const processUpload = async upload => {
  const { createReadStream, filename, mimetype, encoding } = await upload
  const stream = createReadStream()
  console.log(stream)
//   const { id, path } = await storeUpload({ stream, filename })
  return true
  return recordFile({ id, filename, mimetype, encoding, path })
}


export default {
    // Query: {
    //   uploads: () => db.get('uploads').value(),
    // },
    Mutation: {
      UploadFile: (obj, { file }) => processUpload(file),
      multipleUpload: (obj, { files }) => Promise.all(files.map(processUpload)),
    },
}
```

addComment.graphql

```
type Mutation {
    addComment(text: String!, postId: String!): Comment!
}
```

addComment.js

```
import { isAuthenticated } from "../../../middlewares";
import { prisma } from "../../../../generated/prisma-client";

export default {
    Mutation: {
        addComment: async(_, args, {request}) => {
            isAuthenticated(request)
            const { text, postId } = args
            const { user } = request
            const comment = await prisma.createComment({
                user:{
                    connect: {
                        id: user.id
                    }
                },
                post: {
                    connect: {
                        id: postId
                    }
                },
                text
            })
            return comment
        }
    }
}
```

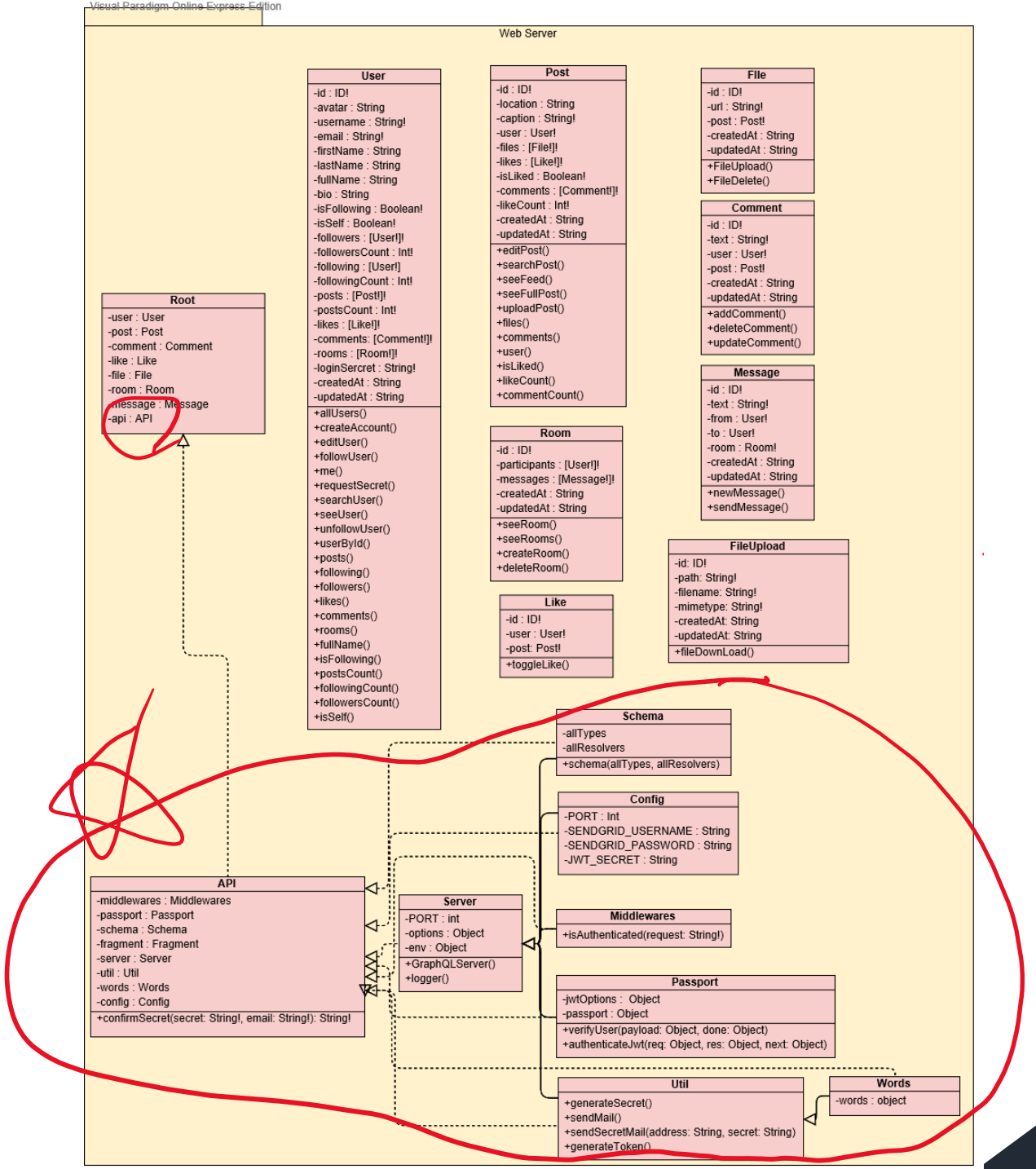
newMessage.graphql

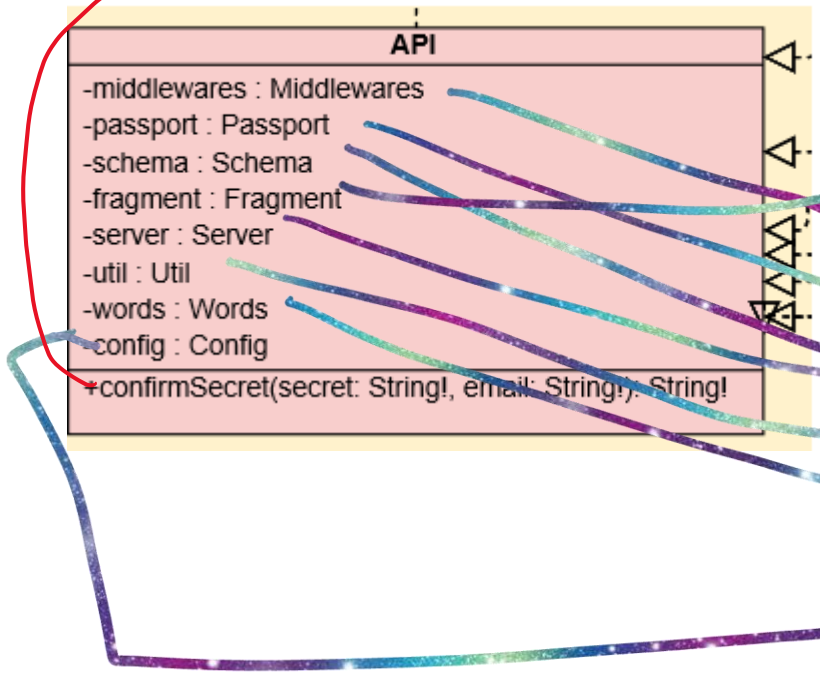
```
type Subscription {
    newMessage(roomId: String!): Message
}
```

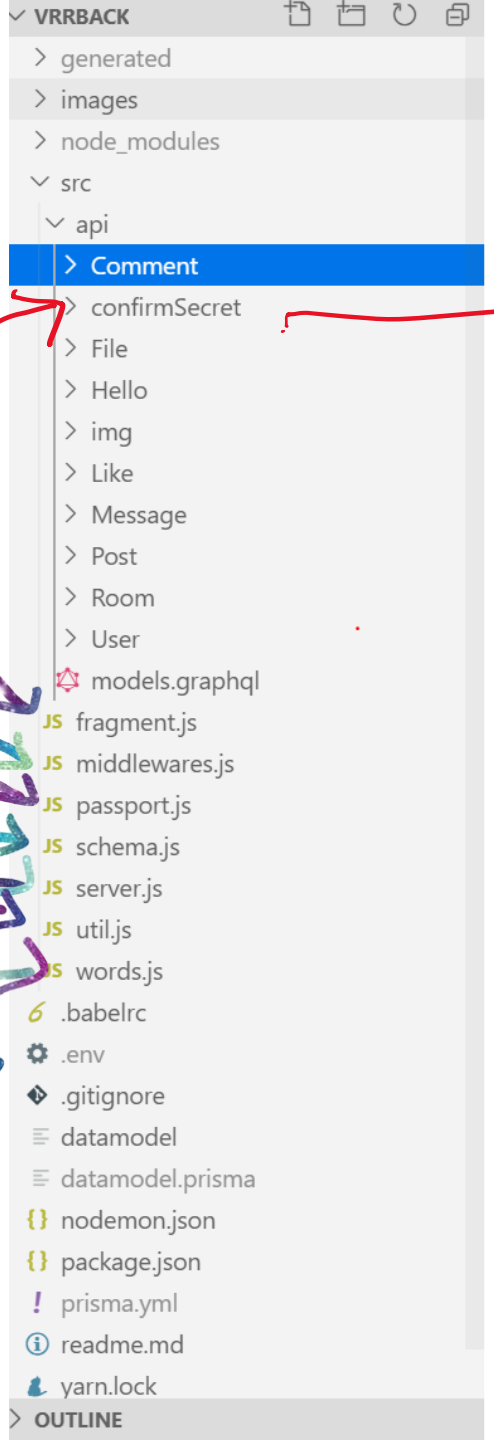
newMessage.js

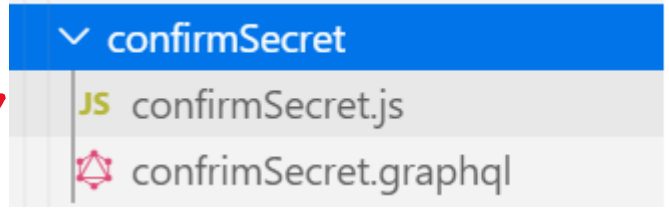
```
import { prisma } from "../../../../generated/prisma-client"

export default {
    Subscription: {
        newMessage: {
            subscribe: (_, args) => {
                const { roomId } = args
                return prisma.$subscribe.message({
                    AND: [
                        { mutation_in: "CREATED" },
                        {
                            node: {
                                room: { id: roomId }
                            }
                        }
                    ]
                }).node()
            },
            resolve: payload => payload
        }
    }
}
```

Web Server


Visual Paradigm Online Express Edition
Web Server
User
-id : ID!
-avatar : String
-username : String!
-email : String!
-firstName : String
-lastName : String
-fullName : String
-bio : String
-isFollowing : Boolean!
-isSelf : Boolean!
-followers : [User!]!
-followersCount : Int!
-following : [User!]
-followingCount : Int!
-posts : [Post!]!
-postsCount : Int!
-likes : [Like!]!
-comments: [Comment!]!
-rooms : [Room!]!
-loginSercret : String!
-createdAt : String
-updatedAt : String
+allUsers()
+createAccount()
+editUser()
+followUser()
+me()
+requestSecret()
+searchUser()
+seeUser()
+unfollowUser()
+userById()
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
Post
-id : ID!
-location : String
-caption : String!
-user : User!
-files : [File!]!
-likes : [Like!]!
-isLiked : Boolean!
-comments : [Comment!]!
-likeCount : Int!
-createdAt : String
-updatedAt : String
+editPost()
+searchPost()
+seeFeed()
+seeFullPost()
+uploadPost()
+files()
+comments()
+user()
+isLiked()
+likeCount()
+commentCount()
FIle
-id : ID!
-url : String!
-post : Post!
-createdAt : String
-updatedAt : String
+FileUpload()
+FileDelete()
Comment
-id : ID!
-text : String!
-user : User!
-post : Post!
-createdAt : String
-updatedAt : String
+addComment()
+deleteComment()
+updateComment()
Message
-id : ID!
-text : String!
-from : User!
-to : User!
-room : Room!
-createdAt : String
-updatedAt : String
+newMessage()
+sendMessage()
Root
-user : User
-post : Post
-comment : Comment
-like : Like
-file : File
-room : Room
-message : Message
-api : API
Room
-id : ID!
-participants : [User!]!
-messages : [Message!]!
-createdAt : String
-updatedAt : String
+seeRoom()
+seeRooms()
+createRoom()
+deleteRoom()
Like
-id : ID!
-user : User!
-post: Post!
+toggleLike()
FileUpload
-id: ID!
-path: String!
-filename: String!
-mimetype: String!
-createdAt: String
-updatedAt: String
+fileDownLoad()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object


API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!


VRRBACK
generated
images
node_modules
src
api
Comment
confirmSecret
File
Hello
img
Like
Message
Post
Room
User
models.graphql
fragment.js
middlewares.js
passport.js
schema.js
server.js
util.js
words.js
.babelrc
.env
.gitignore
datamodel
datamodel.prisma
nodemon.json
package.json
prisma.yml
readme.md
yarn.lock
OUTLINE

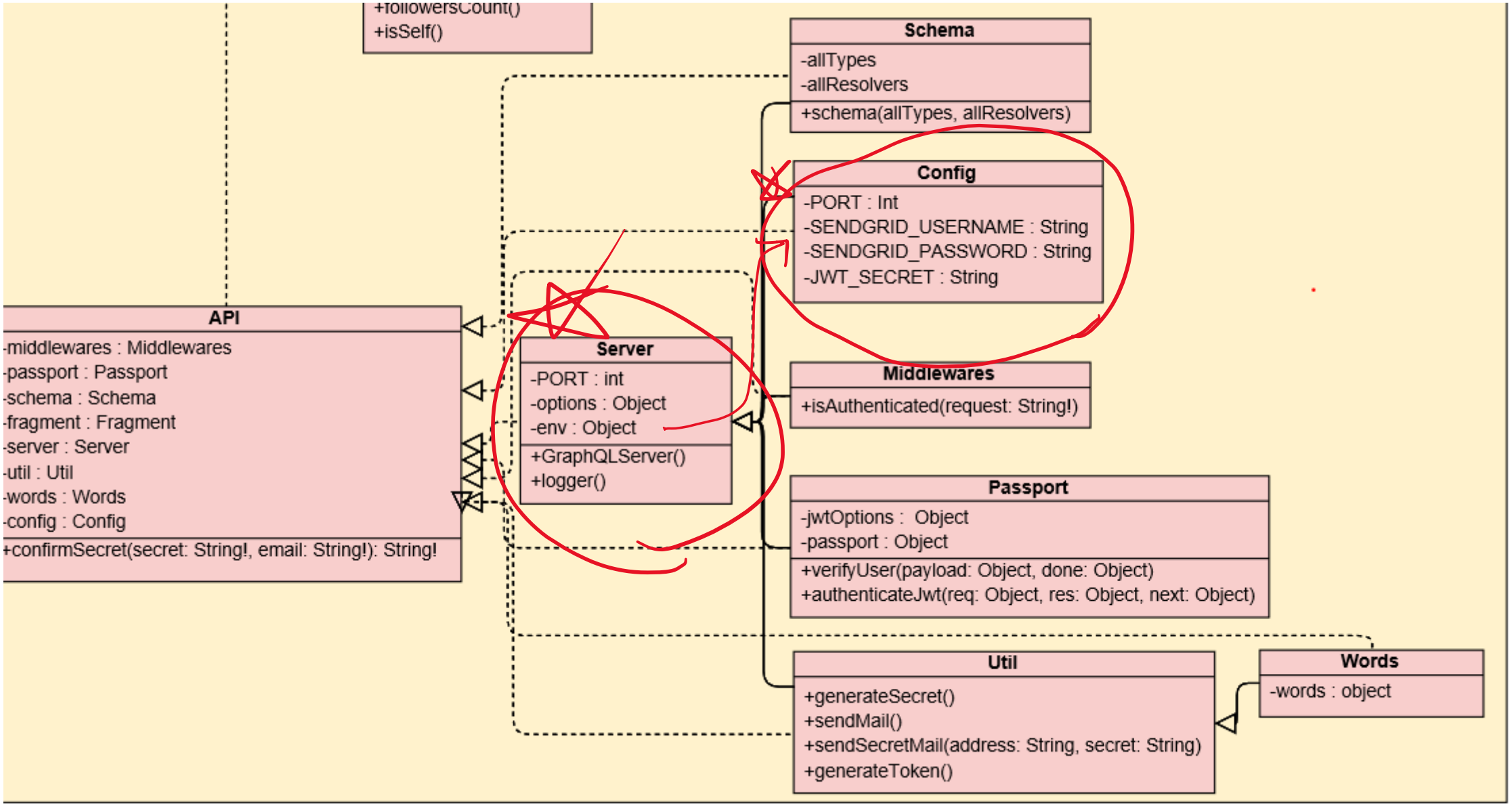

confirmSecret
confirmSecret.js
confrimSecret.graphql

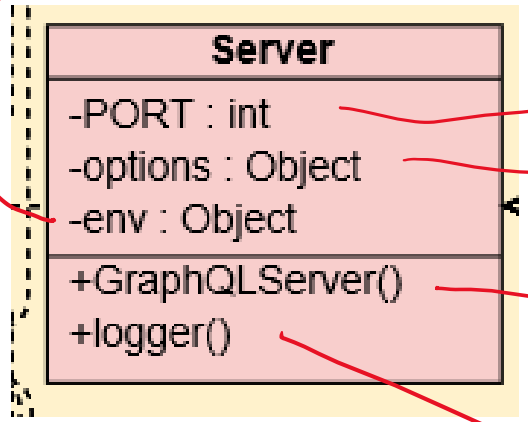
confrimSecret.graphql

```graphql
type Mutation {
    confirmSecret(secret: String!, email: String!): String!
}
```

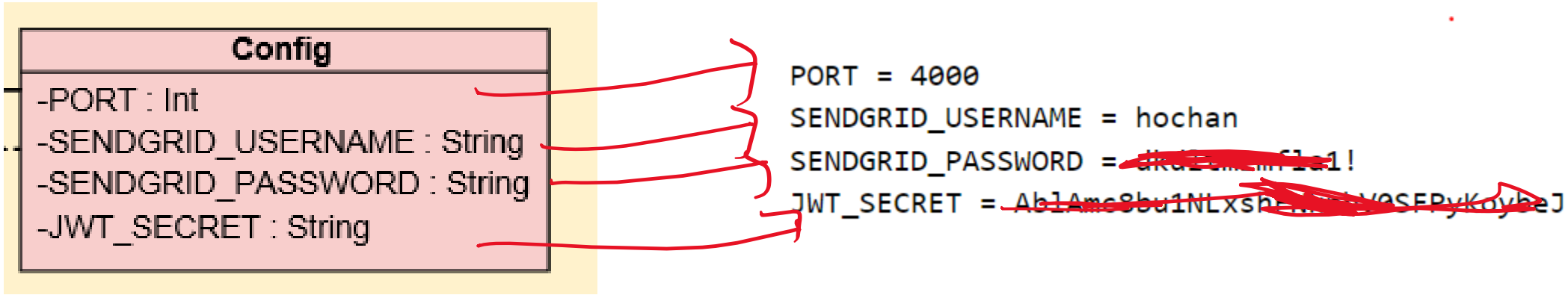
confirmSecret.js

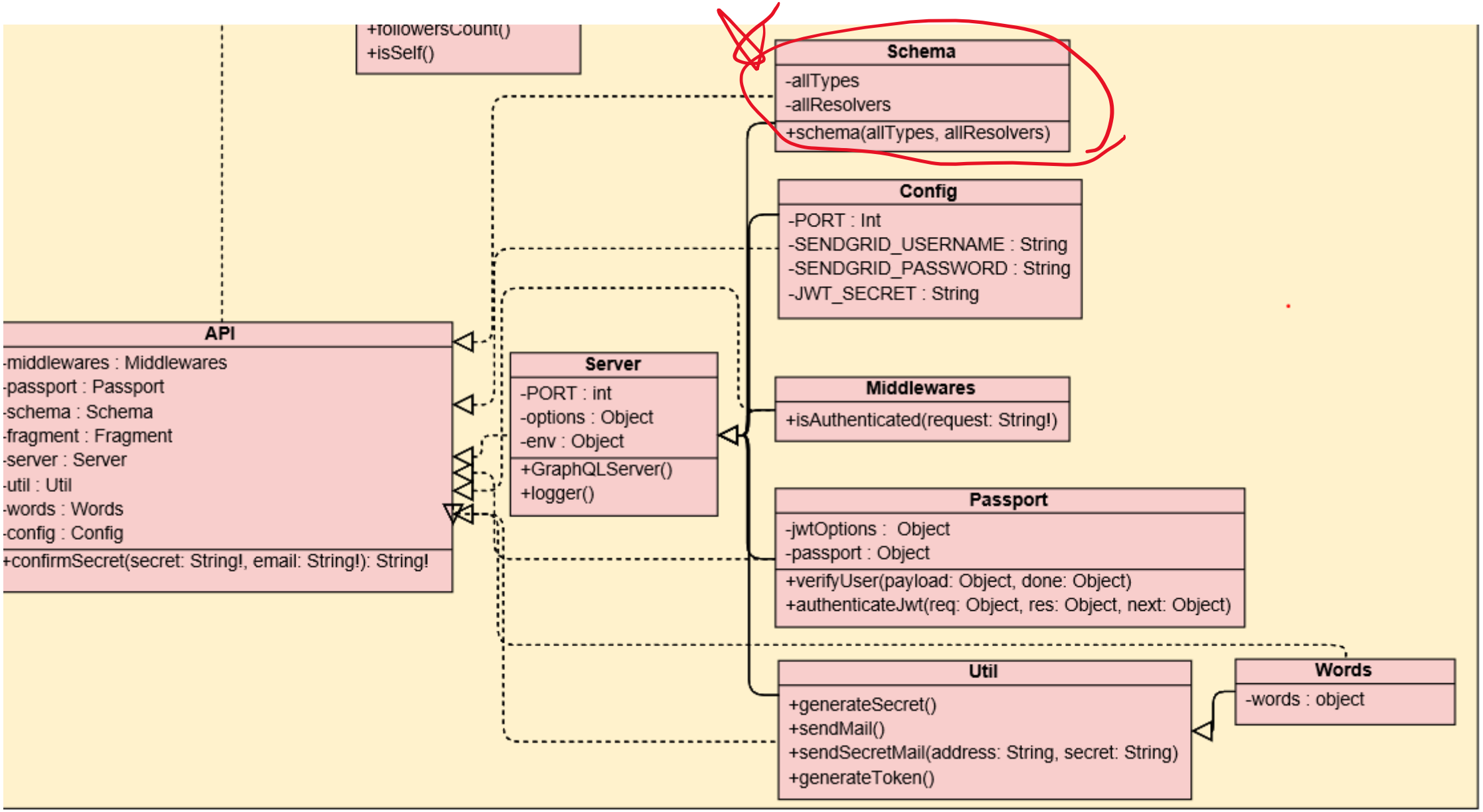
```js
import { prisma } from "../../../generated/prisma-client"
import { generateToken } from "../../util"

export default {
    Mutation: {
        confirmSecret: async(_, args) => {
            const { email, secret } = args
            const user = await prisma.user({ email })
            if(user.loginSecret === secret){
                // delete loginSecret
                await prisma.updateUser({
                    where: { id: user.id },
                    data: { loginSecret: ""}
                })
                return generateToken(user.id)
            } else {
                throw Error("Wrong email/secret combination")
            }
        }
    }
}
```

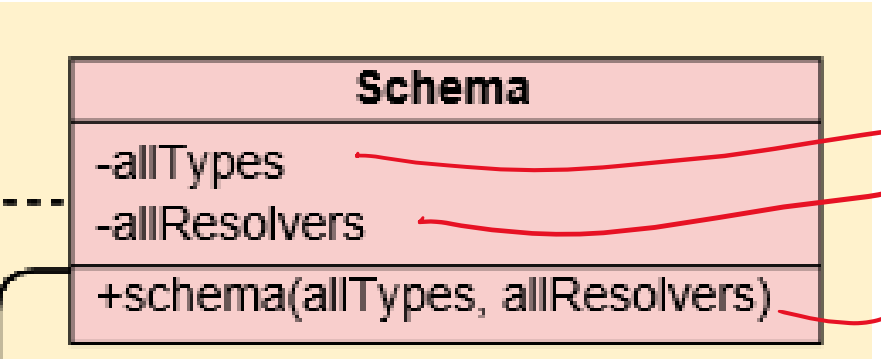

+followersCount()
+isSelf()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object



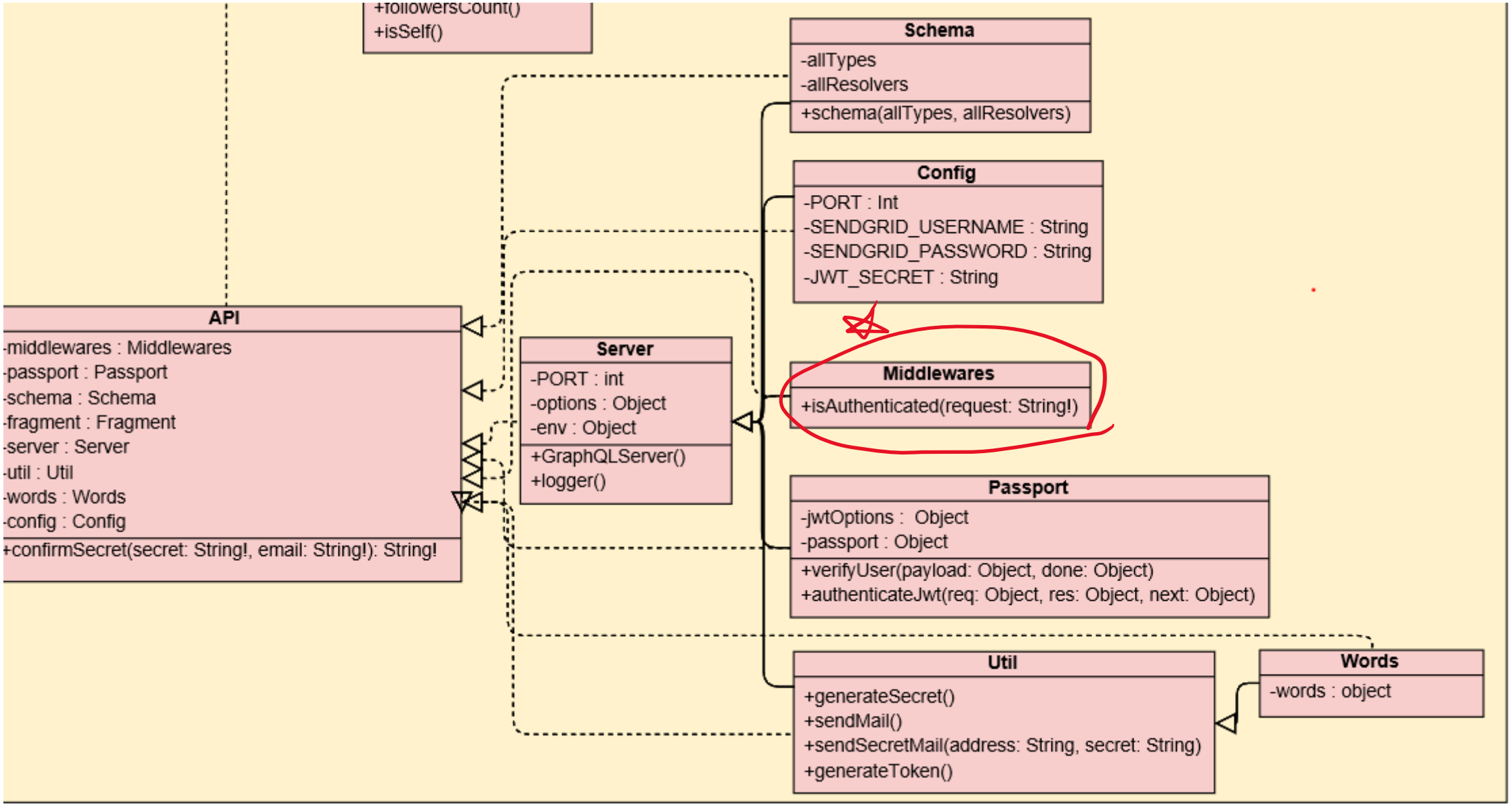

```
import { GraphQLServer } from "graphql-yoga"
import logger from 'morgan'
require('dotenv').config()

import schema from './schema'
import passport from 'passport'
import './passport'
import { authenticateJwt } from "./passport"
import { isAuthenticated } from './middlewares'

const PORT = process.env.PORT || 4000
const options = {
    port: PORT,
    endpoint: '/graphql',
    playground: '/playground',
    subscriptions: '/subscriptions',
    uploads: {
        maxFileSize: 10000000, // 10 MB
        maxFiles: 10},
}

const server = new GraphQLServer({ schema,
    context: ({request}) => ({request, isAuthenticated })
})

server.express.use(logger("dev"))
server.express.use(authenticateJwt)


server.express.get('/hello', function(req, res) {
    res.send('hello world');
  });

server.start(options, () => console.log(`Server running on http://localhost:${PORT}`))
```

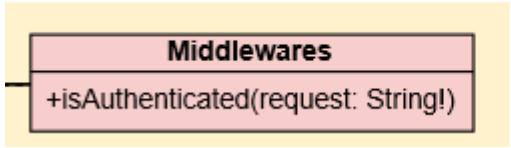

Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
PORT = 4000
SENDGRID_USERNAME = hochan
SENDGRID_PASSWORD =
JWT_SECRET =


+followersCount()
+isSelf()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object




```
import path from 'path'
import { makeExecutableSchema } from 'graphql-tools'
import { fileLoader, mergeResolvers, mergeTypes } from 'merge-graphql-schemas'

const allTypes = fileLoader(path.join(__dirname, '/api/**/*.graphql'))
const allResolvers = fileLoader(path.join(__dirname, '/api/**/*.js'))

const schema = makeExecutableSchema({
    typeDefs: mergeTypes(allTypes),
    resolvers: mergeResolvers(allResolvers)
})

export default schema
```


+followersCount()
+isSelf()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object

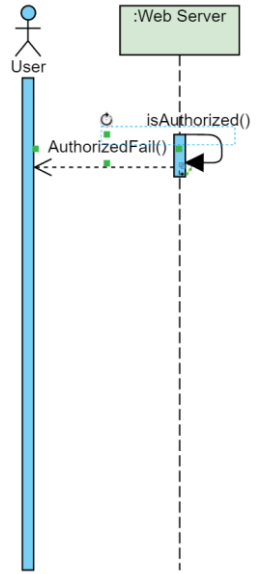


middlewares.js

```
export const isAuthenticated = (request) => {
    if(!request.user){
        throw Error('You need to log in to perform this action')
    }
    return
}
```

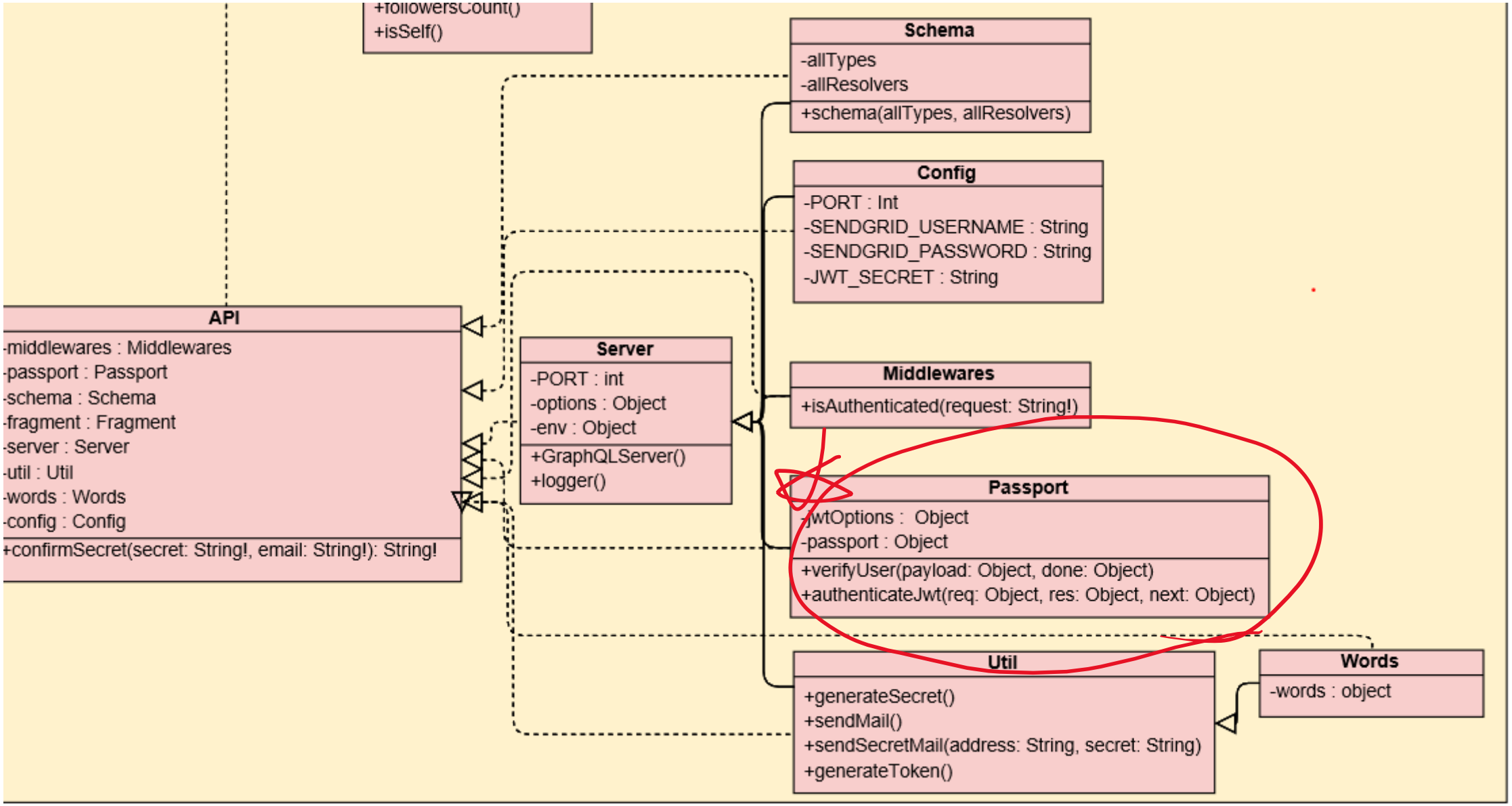
server.js

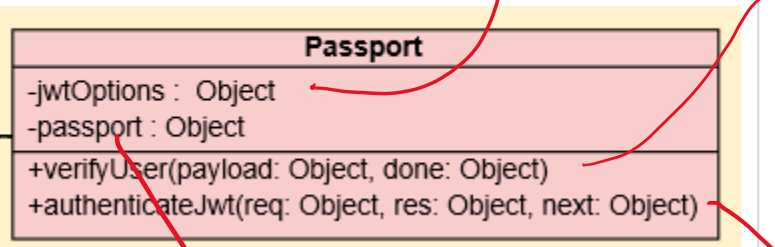
```
const server = new GraphQLServer({ schema,
    context: ({request}) => ({request, isAuthenticated })
})
```



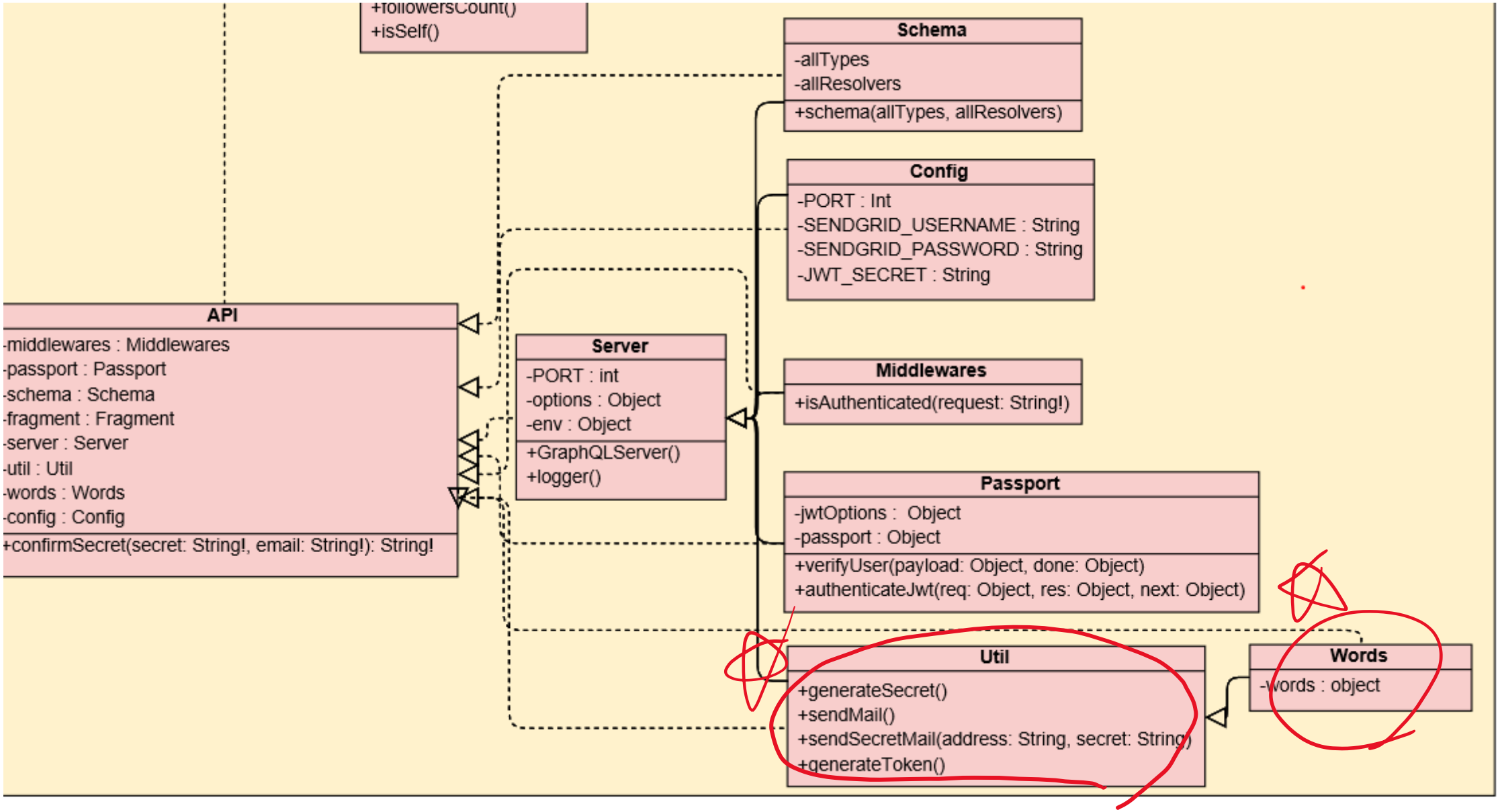
seeRoom.js

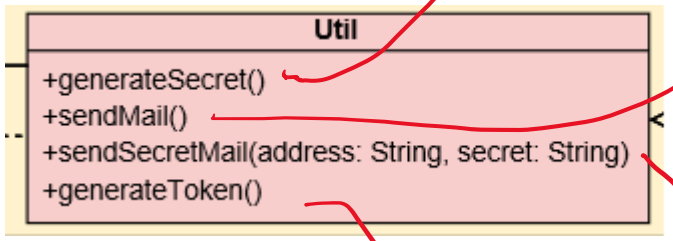
```
import { prisma } from "../../../../generated/prisma-client"
import { ROOM_FRAGMENT } from "../../../fragment"

export default {
    Query: {
        seeRoom: async (_, args, {request, isAuthenticated}) => {
            isAuthenticated(request)
            const { id } = args
            const { user } = request
            const canSee = await prisma.$exists.room({
                participants_some: {
                    id: user.id
                }
            })
            if(canSee) {
                return prisma.room({ id }).$fragment(ROOM_FRAGMENT)
            } else {
                throw Error("You can't see this")
            }
        }
    }
}
```


+followersCount()
+isSelf()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object

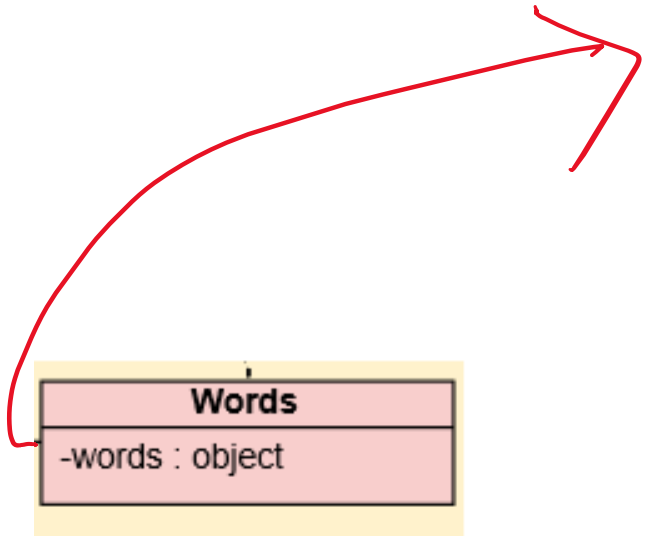



```
import passport from "passport"
import { Strategy, ExtractJwt } from "passport-jwt"
import { prisma } from '../generated/prisma-client'

const jwtOptions = {
    jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
    secretOrKey: process.env.JWT_SECRET
}

const verifyUser = async(payload, done) => {
    try{
        const user = await prisma.user({id: payload.id})
        if(user !== null){
            return done(null, user)
        } else {
            return done(null, false)
        }
    } catch(error) {
        return done(error, false)
    }
}

export const authenticateJwt = (req, res, next) => passport.authenticate('jwt', {sessions: false}, (error, user) => {
    if(user) {
        req.user = user;
    }
    next()
})(req, res, next)

passport.use(new Strategy(jwtOptions, verifyUser))
passport.initialize()
```

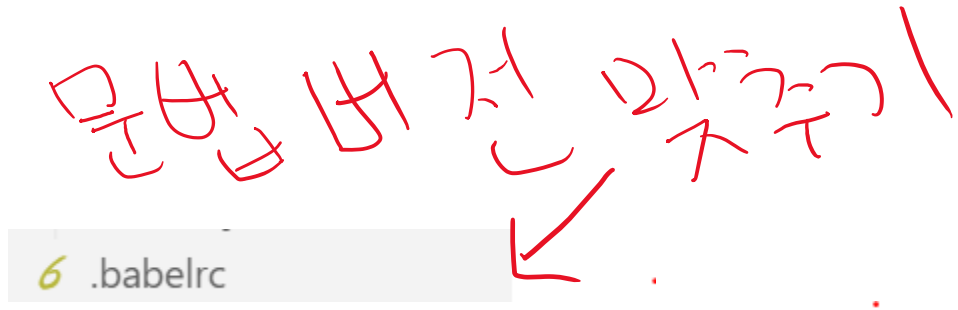

+followersCount()
+isSelf()
Schema
-allTypes
-allResolvers
+schema(allTypes, allResolvers)
Config
-PORT : Int
-SENDGRID_USERNAME : String
-SENDGRID_PASSWORD : String
-JWT_SECRET : String
API
-middlewares : Middlewares
-passport : Passport
-schema : Schema
-fragment : Fragment
-server : Server
-util : Util
-words : Words
-config : Config
+confirmSecret(secret: String!, email: String!): String!
Server
-PORT : int
-options : Object
-env : Object
+GraphQLServer()
+logger()
Middlewares
+isAuthenticated(request: String!)
Passport
-jwtOptions : Object
-passport : Object
+verifyUser(payload: Object, done: Object)
+authenticateJwt(req: Object, res: Object, next: Object)
Util
+generateSecret()
+sendMail()
+sendSecretMail(address: String, secret: String)
+generateToken()
Words
-words : object

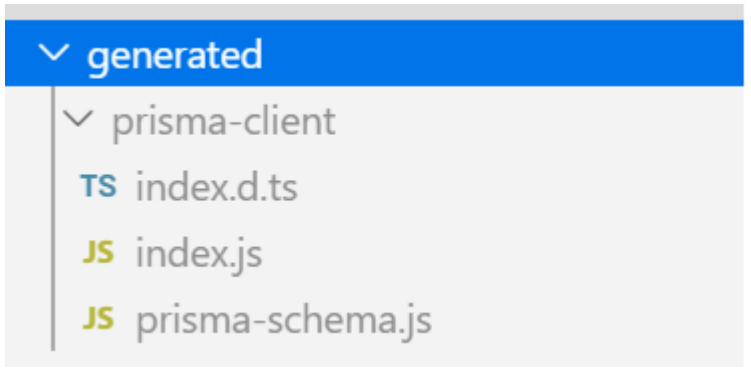


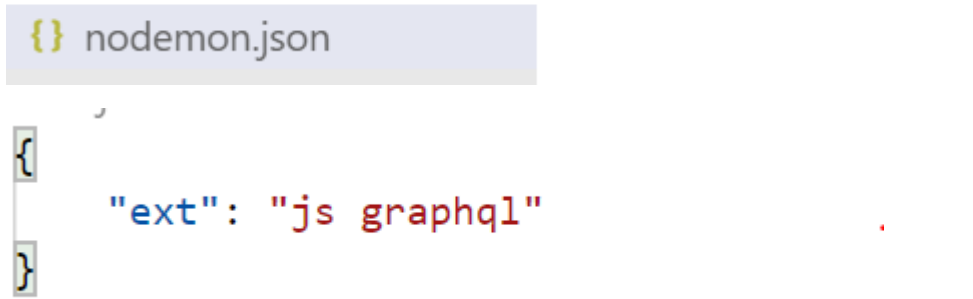

```
require('dotenv').config()

import { adjectives, nouns } from "./words"
import nodemailer from 'nodemailer'
import sgTransport from 'nodemailer-sendgrid-transport'
import jwt from "jsonwebtoken"

export const generateSecret = () => {
    const randomNumber = Math.floor(Math.random() * adjectives.length)
    return `${adjectives[randomNumber]} ${nouns[randomNumber]}`
}

const sendMail = (email) => {
    const options = {
        auth: {
          api_user: process.env.SENDGRID_USERNAME,
          api_key: process.env.SENDGRID_PASSWORD
        }
    }
    const client = nodemailer.createTransport(sgTransport(options))

    return client.sendMail(email)
}

export const sendSecretMail = (address, secret) => {
    const email = {
        from: "ghcksdk.com",
        to: address,
        subject: "Login Secret for Prismagram",
        html: `Hello! Your login secret is <b>${secret}</b>.<br/>Copy paste on the app/website login`
    }
    return sendMail(email)
}

export const generateToken = (id) => jwt.sign({ id }, process.env.JWT_SECRET)
```

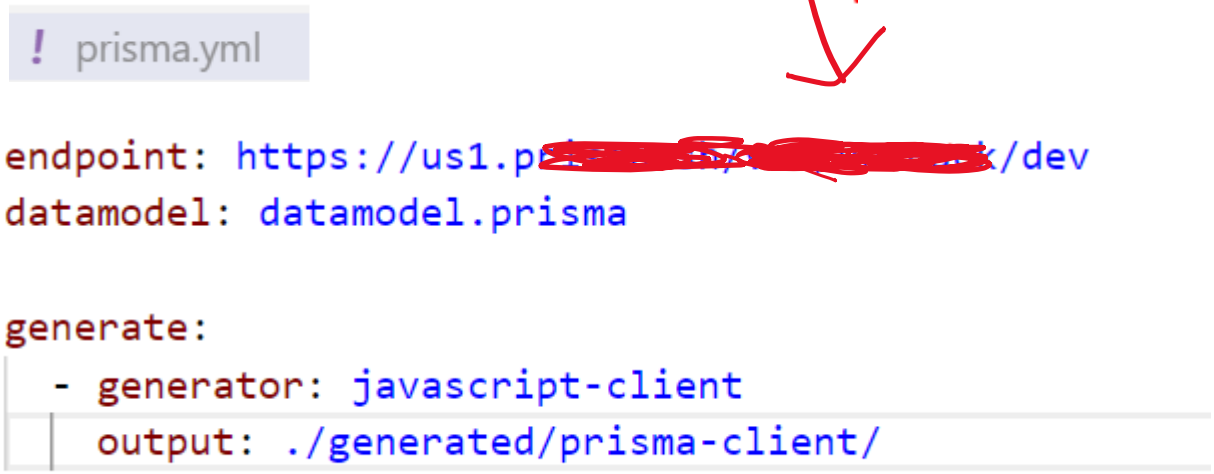



```
export const adjectives = [
    "ad hoc",
    "interesting",
    "outgoing",
    "fearless",
    "fixed",
    "highfalutin",
    "invincible",
    "deserted",
    "ugly",
    "petite",
    "unnatural",
    "curly",
    "delirious",
    "tidy",
    "useless",
    "peaceful",
    "breakable",
    "muddled",
    "wrathful",
    "certain",
    "scientific",
    "deafening",
    "nine",
    "likeable",
    "verdant",
    "overwrought",
    "broken",
    "meaty",
    "alleged",
    "chilly",
    "able",
    "futuristic",
    "enchanted",
    "excited",
    "lavish",
    "tested",
    "dusty",
    "graceful",
    "impolite",
    "gaping",
    "sore",
    "scandalous",
    "exciting",
```

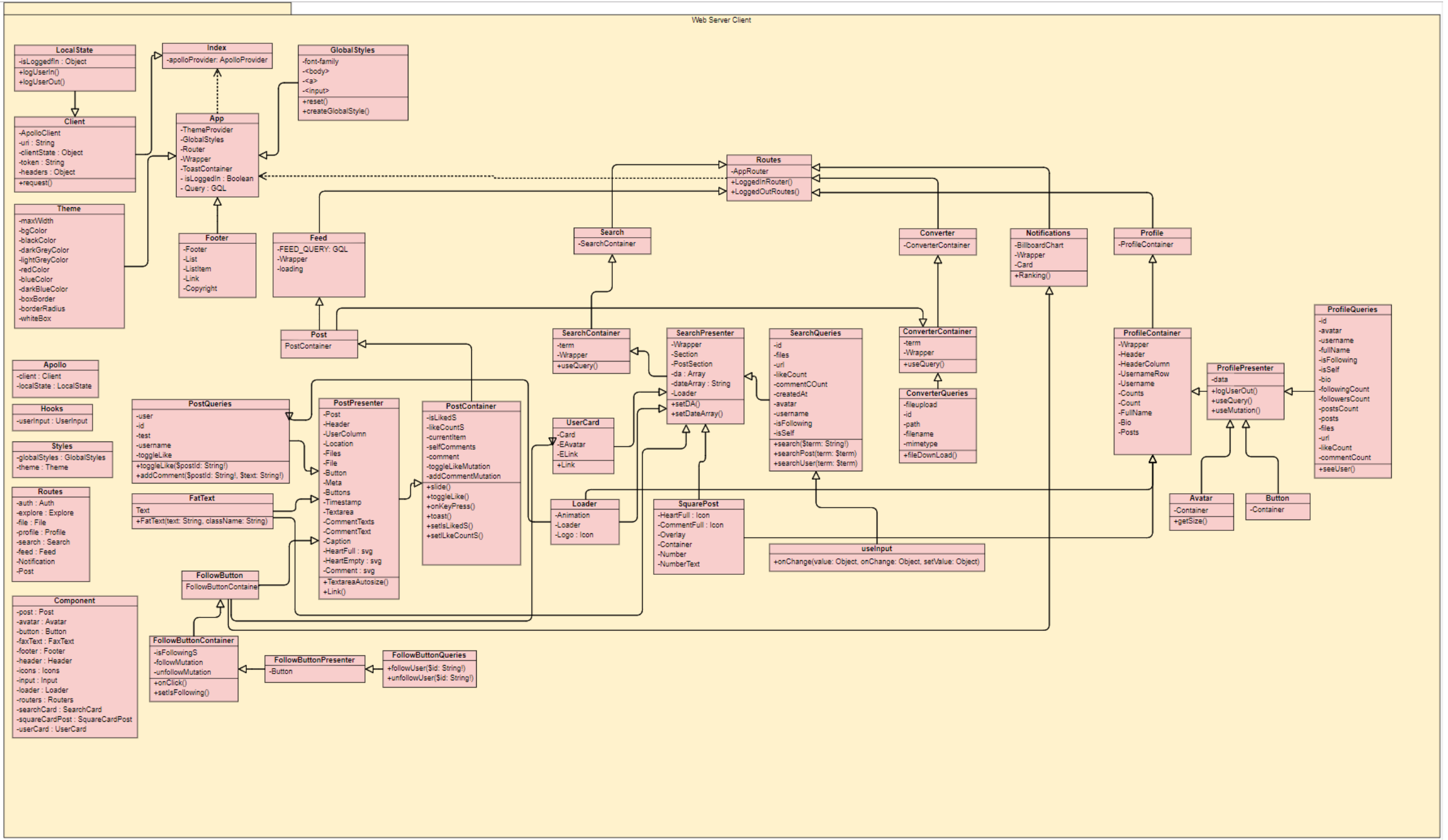

.babelrc
{
"presets": ["@babel/preset-env"]
}

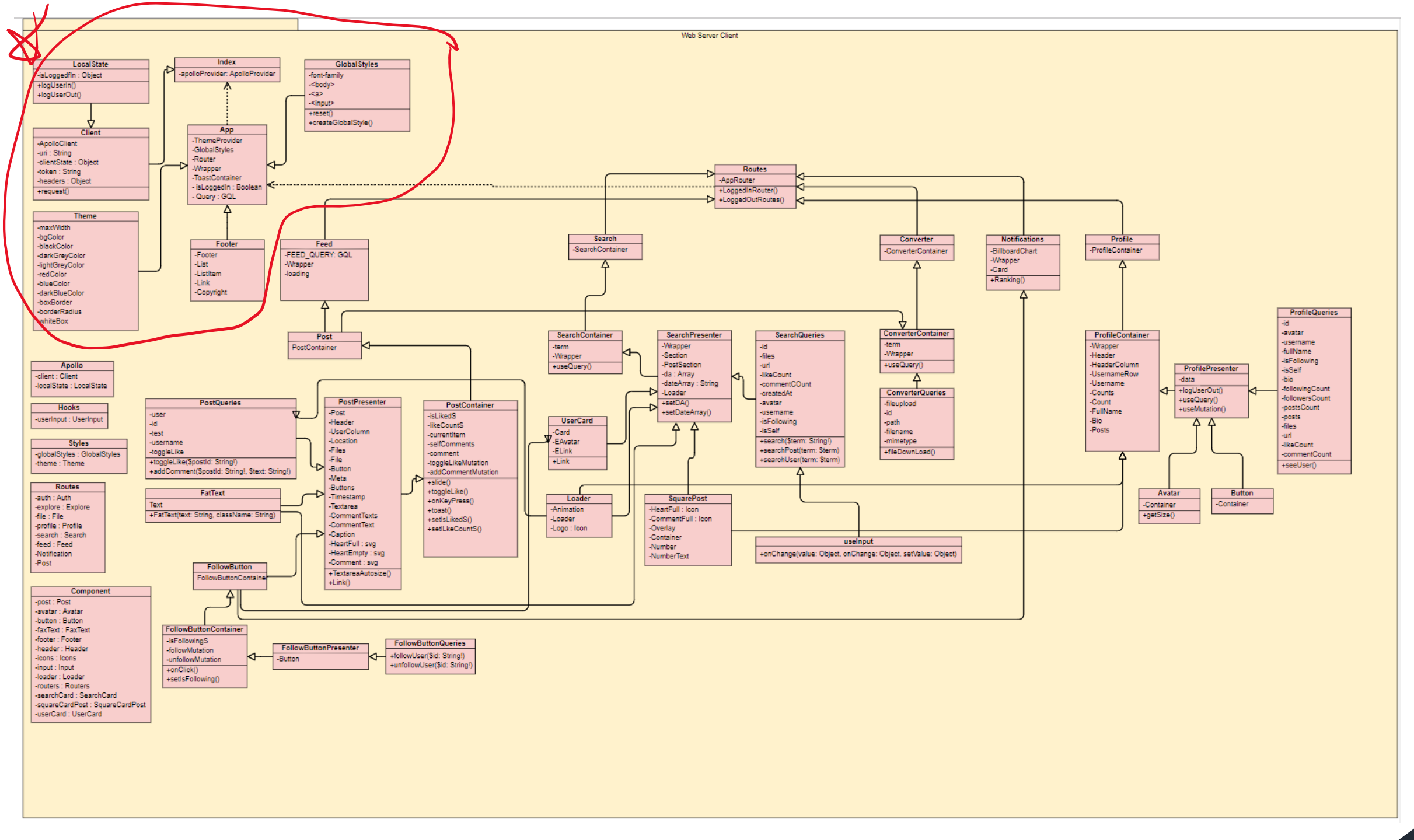

generated
prisma-client
index.d.ts
index.js
prisma-schema.js

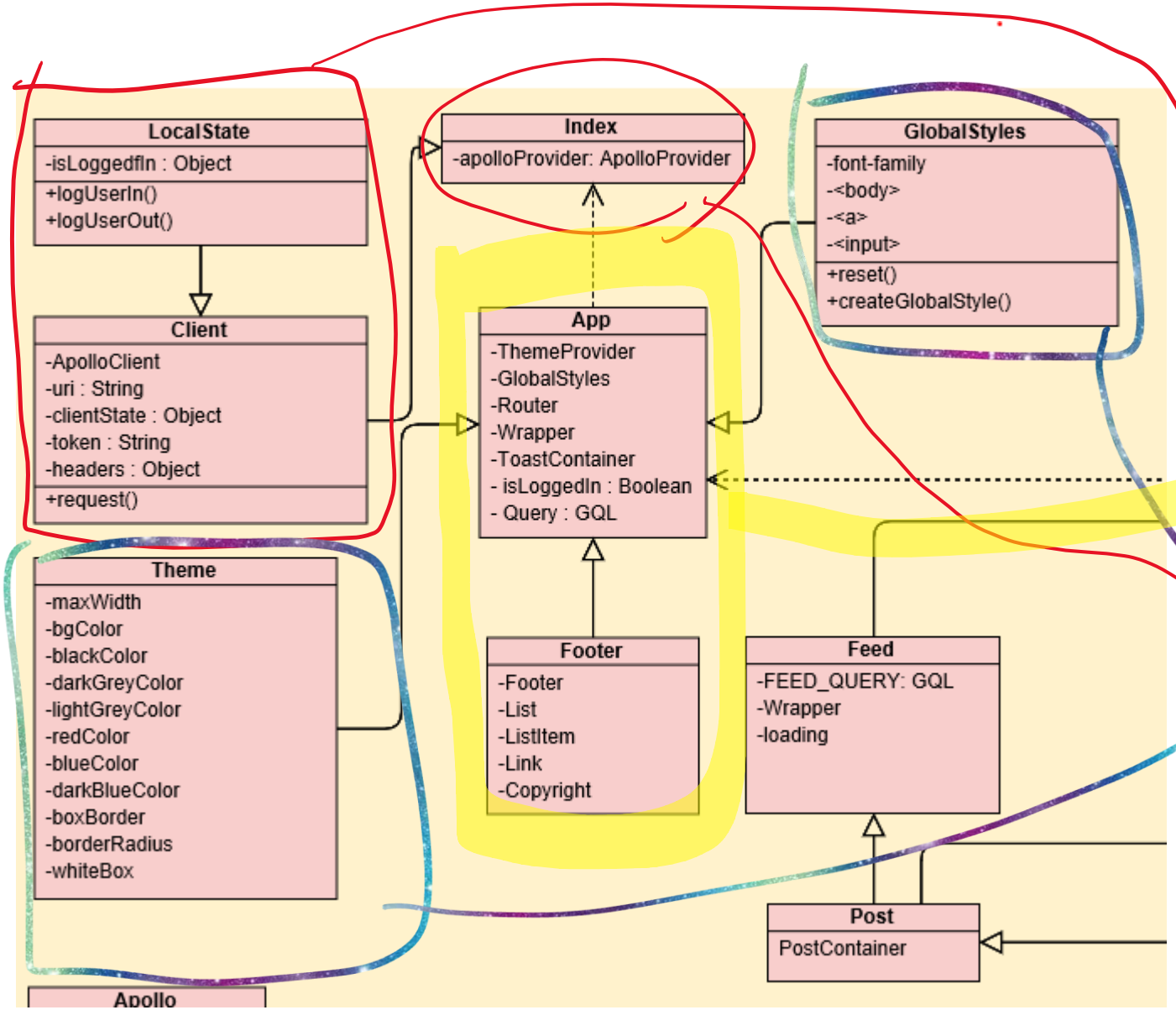

nodemon.json
{
"ext": "js graphql"
}

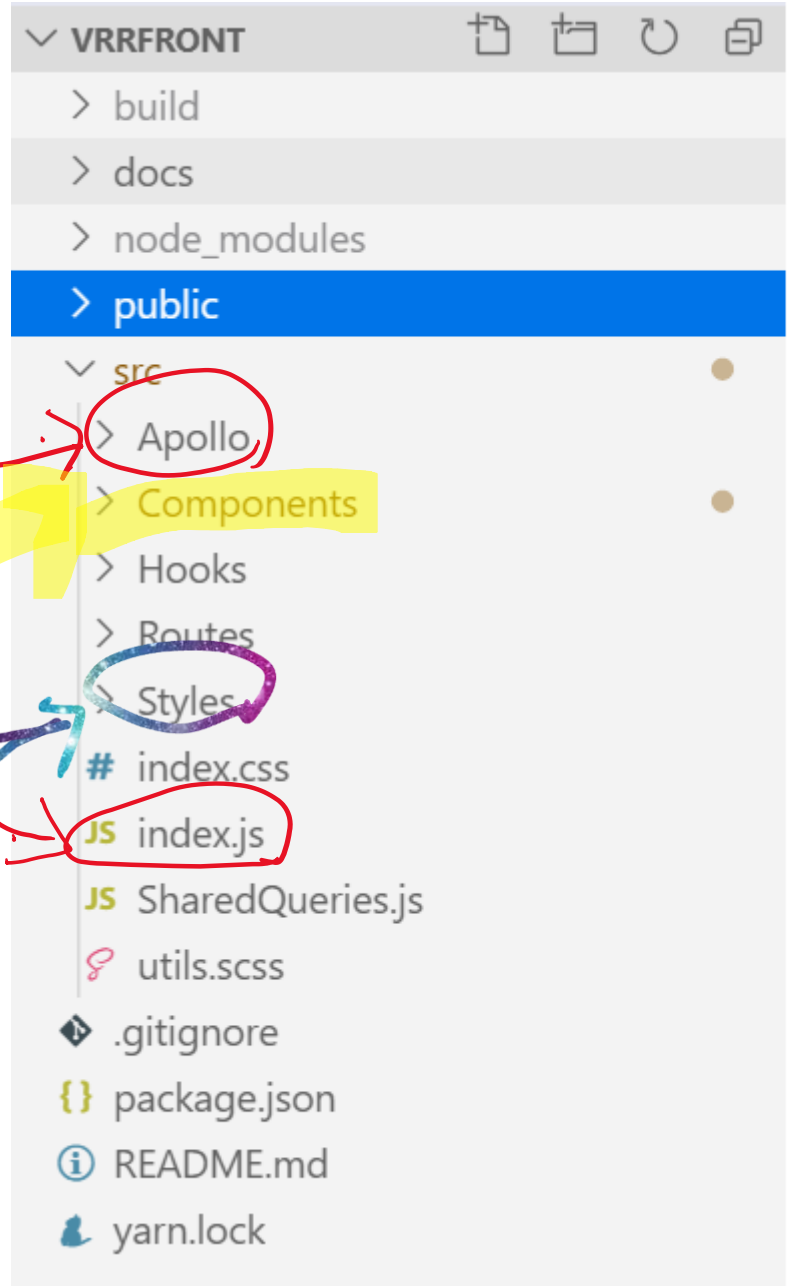

prisma.yml
endpoint: https://us1.p
/dev
datamodel: datamodel.prisma
generate:
- generator: javascript-client
output: ./generated/prisma-client/

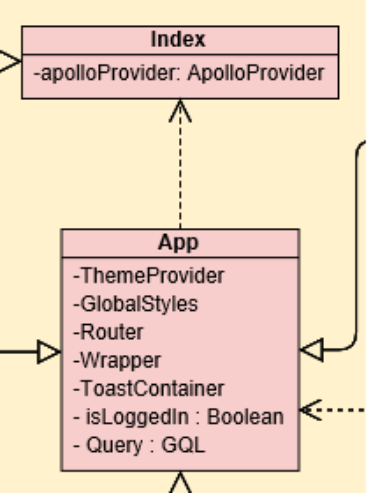
Web Client

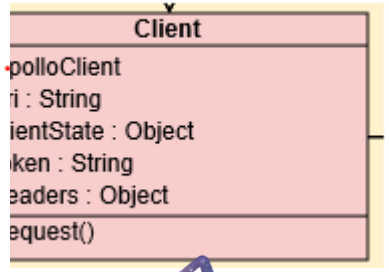

Web Server Client
LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Index
-apolloProvider: ApolloProvider
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Footer
-Footer
-List
-ListItem
-Link
-Copyright
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Search
-SearchContainer
Converter
-ConverterContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer
Post
PostContainer
SearchContainer
-term
-Wrapper
+useQuery()
SearchPresenter
-Wrapper
-Section
-PostSection
-da : Array
-dateArray : String
-Loader
+setDA()
+setDateArray()
SearchQueries
-id
-files
-url
-likeCount
-commentCOunt
-createdAt
-avatar
-username
-isFollowing
-isSelf
+search($term: String!)
+searchPost(term: $term)
+searchUser(term: $term)
ConverterContainer
-term
-Wrapper
+useQuery()
ConverterQueries
-fileupload
-id
-path
-filename
-mimetype
+fileDownLoad()
ProfileContainer
-Wrapper
-Header
-HeaderColumn
-UsernameRow
-Username
-Counts
-Count
-FullName
-Bio
-Posts
ProfilePresenter
-data
+logUserOut()
+useQuery()
+useMutation()
ProfileQueries
-id
-avatar
-username
-fullName
-isFollowing
-isSelf
-bio
-followingCount
-followersCount
-postsCount
-posts
-files
-url
-likeCount
-commentCount
+seeUser()
Apollo
-client : Client
-localState : LocalState
Hooks
-userInput : UserInput
Styles
-globalStyles : GlobalStyles
-theme : Theme
Routes
-auth : Auth
-explore : Explore
-file : File
-profile : Profile
-search : Search
-feed : Feed
-Notification
-Post
PostQueries
-user
-id
-test
-username
-toggleLike
+toggleLike($postId: String!)
+addComment($postId: String!, $text: String!)
PostPresenter
-Post
-Header
-UserColumn
-Location
-Files
-File
-Button
-Meta
-Buttons
-Timestamp
-Textarea
-CommentTexts
-CommentText
-Caption
-HeartFull : svg
-HeartEmpty : svg
-Comment : svg
+TextareaAutosize()
+Link()
PostContainer
-isLikedS
-likeCountS
-currentItem
-selfComments
-comment
-toggleLikeMutation
-addCommentMutation
+slide()
+toggleLike()
+onKeyPress()
+toast()
+setIsLikedS()
+setLkeCountS()
UserCard
-Card
-EAvatar
-ELink
+Link
FatText
Text
+FatText(text: String, className: String)
Loader
-Animation
-Loader
-Logo : Icon
SquarePost
-HeartFull : Icon
-CommentFull : Icon
-Overlay
-Container
-Number
-NumberText
useInput
+onChange(value: Object, onChange: Object, setValue: Object)
Avatar
-Container
+getSize()
Button
-Container
FollowButton
FollowButtonContaine
Component
-post : Post
-avatar : Avatar
-button : Button
-faxText : FaxText
-footer : Footer
-header : Header
-icons : Icons
-input : Input
-loader : Loader
-routers : Routers
-searchCard : SearchCard
-squareCardPost : SquareCardPost
-userCard : UserCard
FollowButtonContainer
-isFollowingS
-followMutation
-unfollowMutation
+onClick()
+setIsFollowing()
FollowButtonPresenter
-Button
FollowButtonQueries
+followUser($id: String!)
+unfollowUser($id: String!)

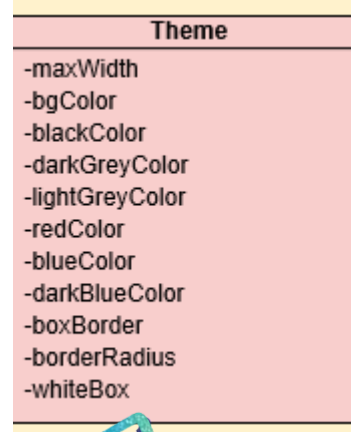

Web Server Client
LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Index
-apolloProvider: ApolloProvider
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Footer
-Footer
-List
-ListItem
-Link
-Copyright
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Search
-SearchContainer
Converter
-ConverterContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer
Post
PostContainer
SearchContainer
-term
-Wrapper
+useQuery()
SearchPresenter
-Wrapper
-Section
-PostSection
-da : Array
-dateArray : String
-Loader
+setDA()
+setDateArray()
SearchQueries
-id
-files
-url
-likeCount
-commentCOunt
-createdAt
-avatar
-username
-isFollowing
-isSelf
+search($term: String!)
+searchPost(term: $term)
+searchUser(term: $term)
ConverterContainer
-term
-Wrapper
+useQuery()
ConverterQueries
-fileupload
-id
-path
-filename
-mimetype
+fileDownLoad()
ProfileContainer
-Wrapper
-Header
-HeaderColumn
-UsernameRow
-Username
-Counts
-Count
-FullName
-Bio
-Posts
ProfilePresenter
-data
+logUserOut()
+useQuery()
+useMutation()
ProfileQueries
-id
-avatar
-username
-fullName
-isFollowing
-isSelf
-bio
-followingCount
-followersCount
-postsCount
-posts
-files
-url
-likeCount
-commentCount
+seeUser()
Apollo
-client : Client
-localState : LocalState
Hooks
-userInput : UserInput
Styles
-globalStyles : GlobalStyles
-theme : Theme
Routes
-auth : Auth
-explore : Explore
-file : File
-profile : Profile
-search : Search
-feed : Feed
-Notification
-Post
PostQueries
-user
-id
-test
-username
-toggleLike
+toggleLike($postId: String!)
+addComment($postId: String!, $text: String!)
FatText
Text
+FatText(text: String, className: String)
PostPresenter
-Post
-Header
-UserColumn
-Location
-Files
-File
-Button
-Meta
-Buttons
-Timestamp
-Textarea
-CommentTexts
-CommentText
-Caption
-HeartFull : svg
-HeartEmpty : svg
-Comment : svg
+TextareaAutosize()
+Link()
PostContainer
-isLikedS
-likeCountS
-currentItem
-selfComments
-comment
-toggleLikeMutation
-addCommentMutation
+slide()
+toggleLike()
+onKeyPress()
+toast()
+setIsLikedS()
+setLkeCountS()
UserCard
-Card
-EAvatar
-ELink
+Link
Loader
-Animation
-Loader
-Logo : Icon
SquarePost
-HeartFull : Icon
-CommentFull : Icon
-Overlay
-Container
-Number
-NumberText
useInput
+onChange(value: Object, onChange: Object, setValue: Object)
Avatar
-Container
+getSize()
Button
-Container
FollowButton
FollowButtonContaine
Component
-post : Post
-avatar : Avatar
-button : Button
-faxText : FaxText
-footer : Footer
-header : Header
-icons : Icons
-input : Input
-loader : Loader
-routers : Routers
-searchCard : SearchCard
-squareCardPost : SquareCardPost
-userCard : UserCard
FollowButtonContainer
-isFollowingS
-followMutation
-unfollowMutation
+onClick()
+setIsFollowing()
FollowButtonPresenter
-Button
FollowButtonQueries
+followUser($id: String!)
+unfollowUser($id: String!)


LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Apollo
Index
-apolloProvider: ApolloProvider
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Footer
-Footer
-List
-ListItem
-Link
-Copyright
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Post
PostContainer


VRRFRONT
build
docs
node_modules
public
src
Apollo
Components
Hooks
Routes
Styles
index.css
index.js
SharedQueries.js
utils.scss
.gitignore
package.json
README.md
yarn.lock





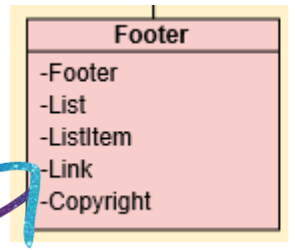


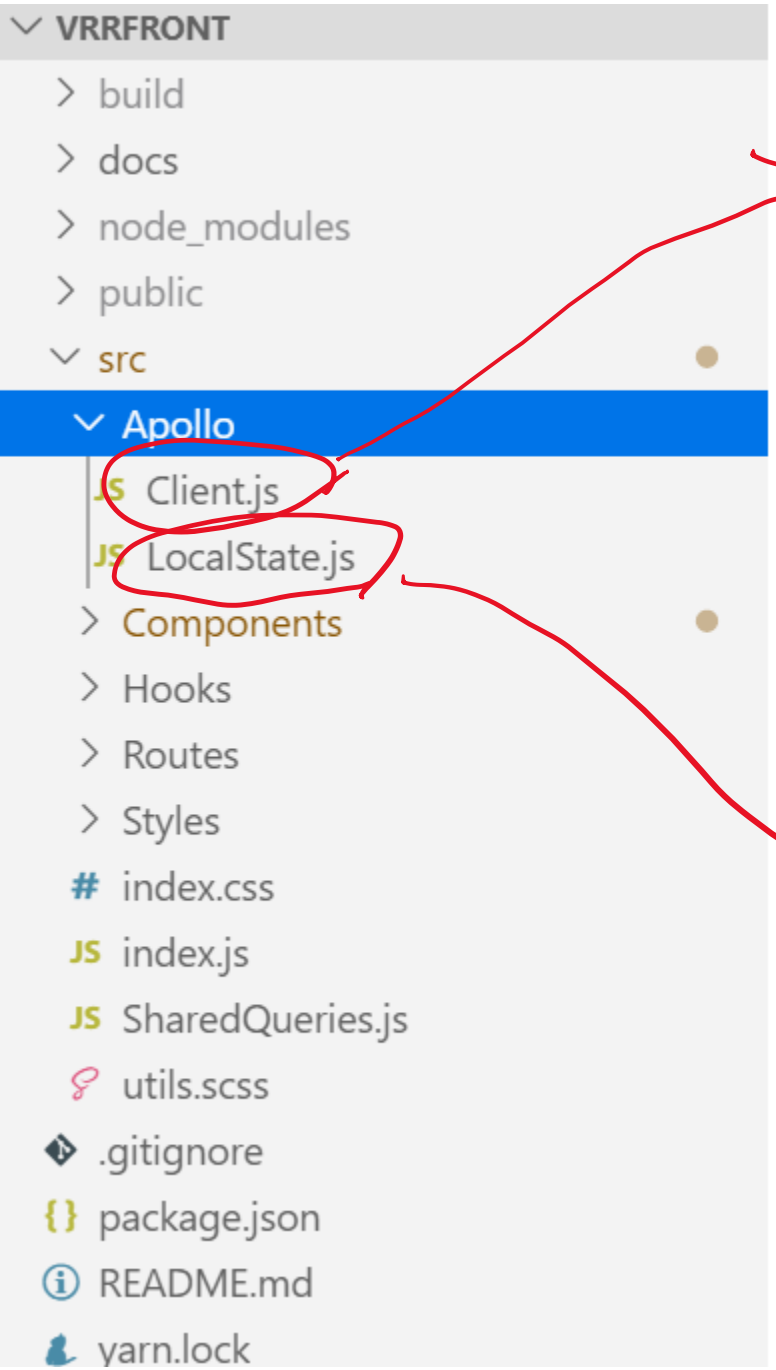
index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './Components/App';

import Client from './Apollo/Client'
import { ApolloProvider } from 'react-apollo-hooks'

ReactDOM.render(
    <ApolloProvider client={Client}>
        <App />
    </ApolloProvider>,
    document.getElementById('root')
);
```

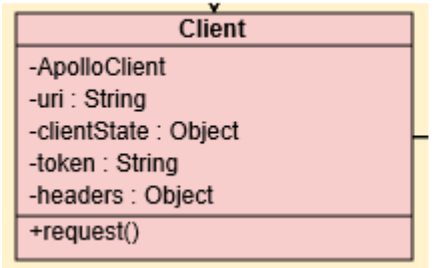
App.js

```
15  const QUERY = gql`
16    {
17      isLoggedIn @client
18    }
19  `
20
21  const Wrapper = styled.div`
22    margin: 0 auto;
23    max-width: ${props => props.theme.maxWidth};
24    width: 100%;
25  `
26  export default () => {
27    const { data: { isLoggedIn } } = useQuery(QUERY)
28
29    return (
30      <>
31        <ThemeProvider theme={Theme}>
32          <>
33            <GlobalStyles/>
34            <Router>
35              <>
36                {isLoggedIn && <Header />}
37                <Wrapper>
38                <Routes isLoggedIn={isLoggedIn}/>
39                <Footer/>
40                </Wrapper>
41              </>
42            </Router>
43            <ToastContainer position={toast.POSITION.BOTTOM_LEFT}/>
44          </>
45        </ThemeProvider>
46      </>
47    )
48  }
49
```

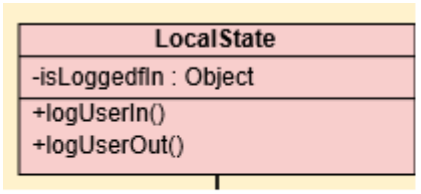




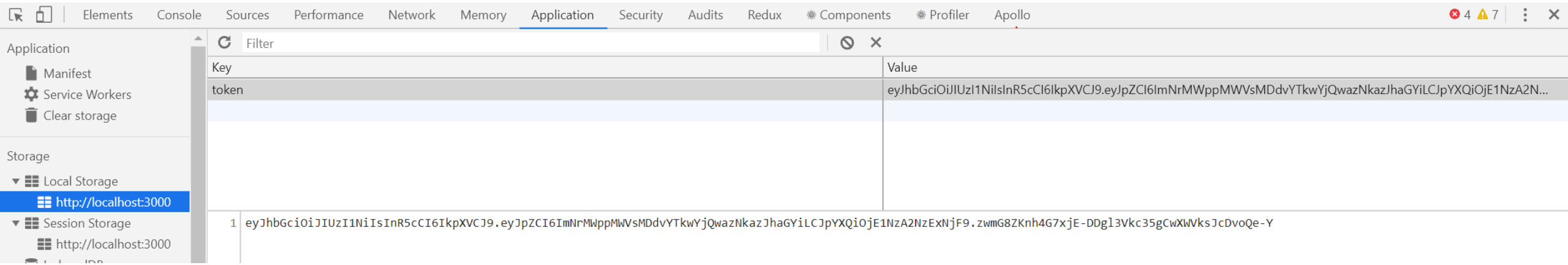

```
import ApolloClient, { ApolloLink } from 'apollo-boost'
import { defaults, resolvers } from '../Apollo/LocalState'

export default new ApolloClient({
    // uri: 'http://13.124.58.19:4000/graphql',
    uri: 'http://127.0.0.1:4000/graphql',
    clientState: {
        defaults,
        resolvers
    },
    request: async operation => {
        const token = await localStorage.getItem('token')
        operation.setContext({
            headers: {
                "Authorization": `Bearer ${token}`
            }
        })
    }
})
```

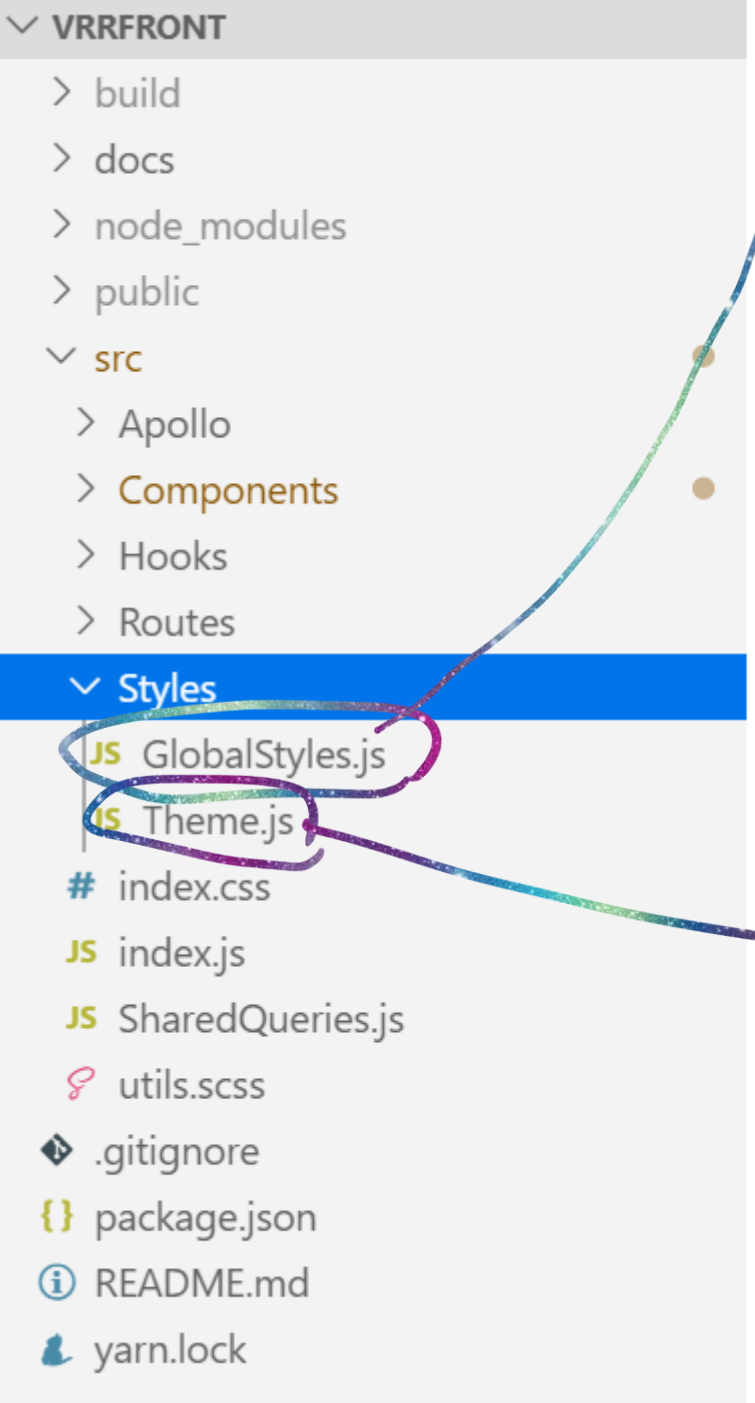



```
export const defaults = {
    // Boolean(null) === false
    isLoggedIn: Boolean(localStorage.getItem('token')) || false
}

export const resolvers = {
    Mutation: {
        logUserIn: (_, {token}, {cache}) => {
            localStorage.setItem("token", token)
            cache.writeData({
                data: {
                    isLoggedIn: true
                }
            })
            return null
        },
        logUserOut: (_, args, {cache}) => {
            localStorage.removeItem("token")
            window.location = "/"
            return null
        }
    }
}
```



```
request: async operation => {
    const token = await localStorage.getItem('token
    operation.setContext({
        headers: {
          "Authorization": `Bearer ${token}`
        }
    })
}
```



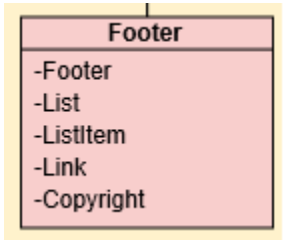



```
import { createGlobalStyle } from "styled-components"
import reset from "styled-reset"

export default createGlobalStyle`
    ${reset};
    @import url('https://fonts.googleapis.com/css?family=Open+Sans:400,600,700&display=swap');
    * {
        box-sizing: border-box;
    }

    body {
        background-color: ${props => props.theme.bgColor};
        color: ${props => props.theme.blackColor};
        font-size: 14px;
        font-family: 'Open Sans', sans-serif;
    }

    a {
        color: ${props => props.theme.blueColor};
        text-decoration: none;
    }
    input:focus {
        outline: none;
    }
`
```

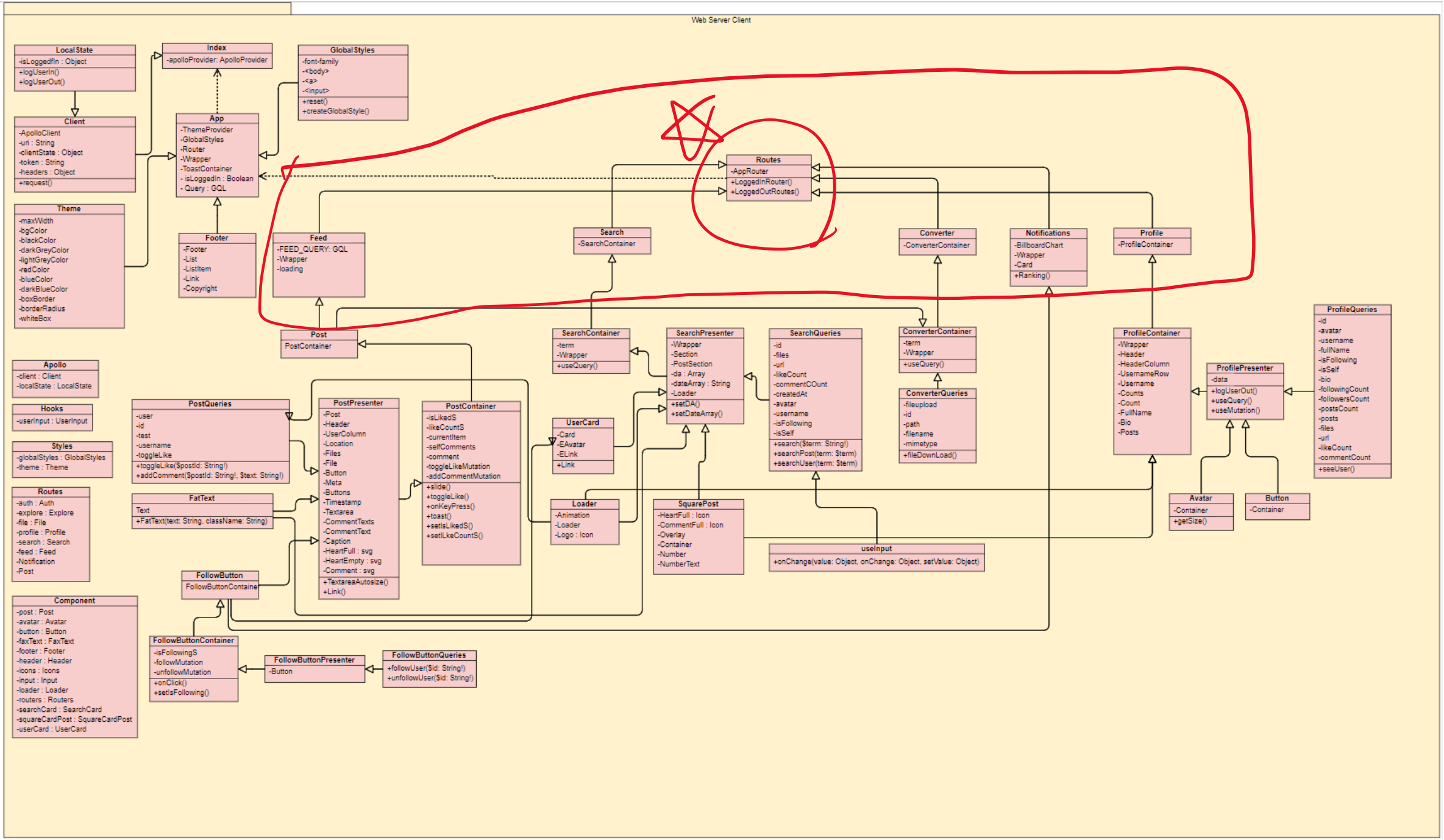
GlobalStyles
-font-family -<body> -<a> -<input>
+reset() +createGlobalStyle()

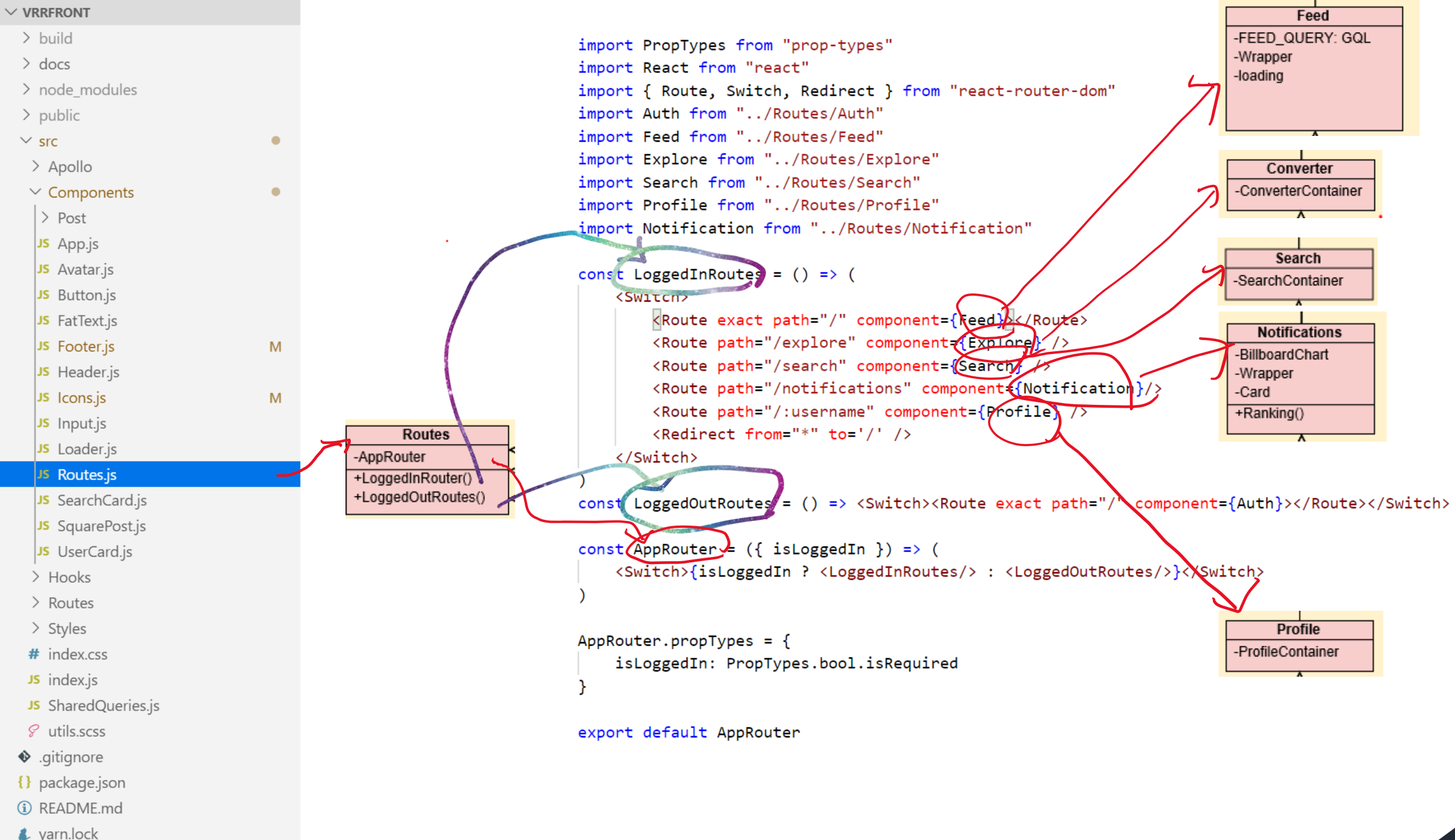
```
const BOX_BORDER = "1px solid #e6e6e6"
const BORDER_RADIUS = "4px"

export default {
    maxWidth: '935px',
    // bgColor: "#FAFAFA",
    bgColor: "#efefef",
    blackColor: "#262626",
    darkGreyColor: "#999",
    lightGreyColor: "#c7c7c7",
    redColor: "#ED4956",
    blueColor: "#3897f0",
    darkBlueColor: "#003569",
    boxBorder: "1px solid #e6e6e6",
    borderRadius: "4px",
    whiteBox: `border: ${BORDER_RADIUS};
               border-radius: ${BOX_BORDER};
               background-color:white;
               `
}
```

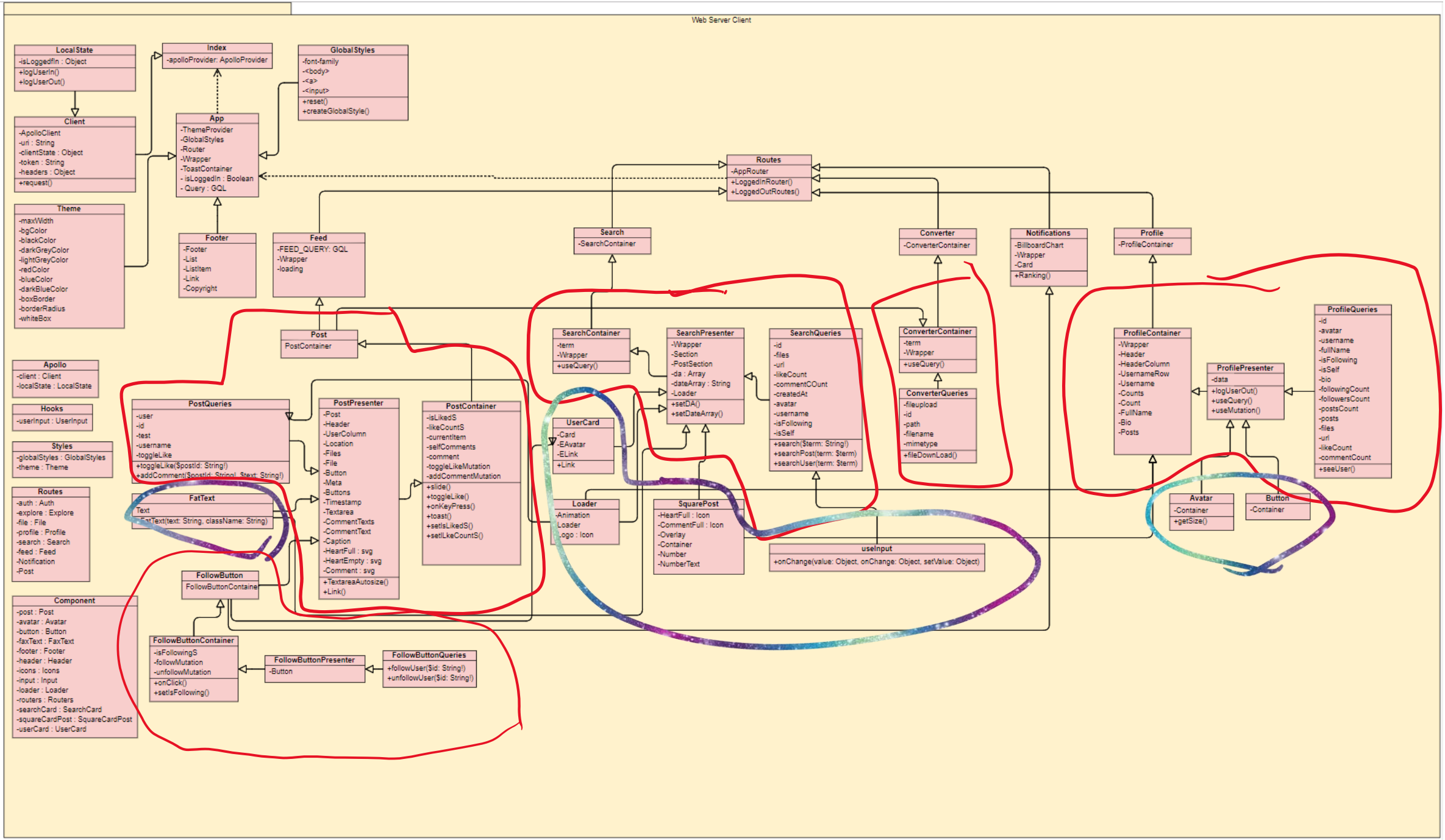
Theme
-maxWidth -bgColor -blackColor -darkGreyColor -lightGreyColor -redColor -blueColor -darkBlueColor -boxBorder -borderRadius -whiteBox

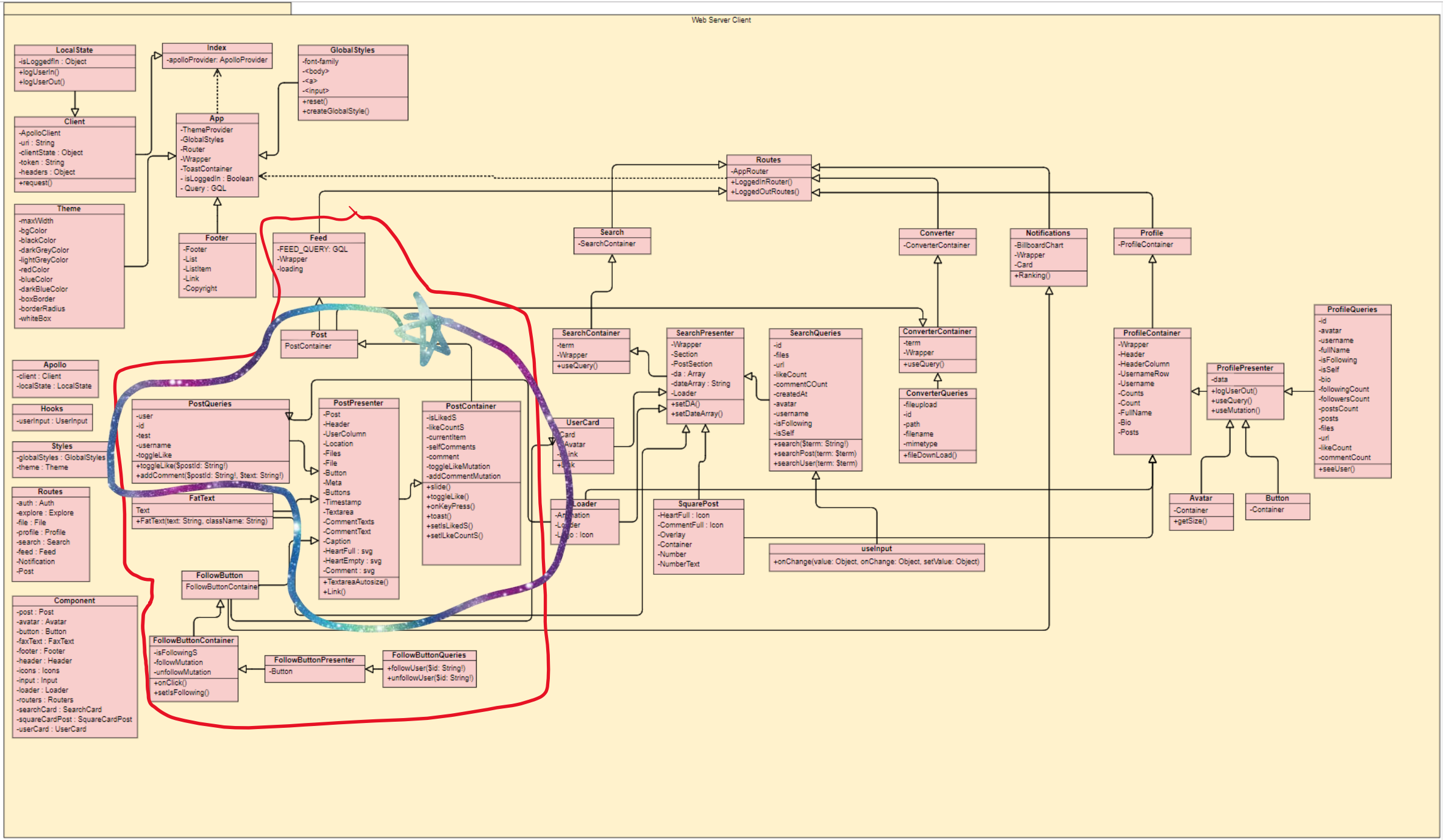


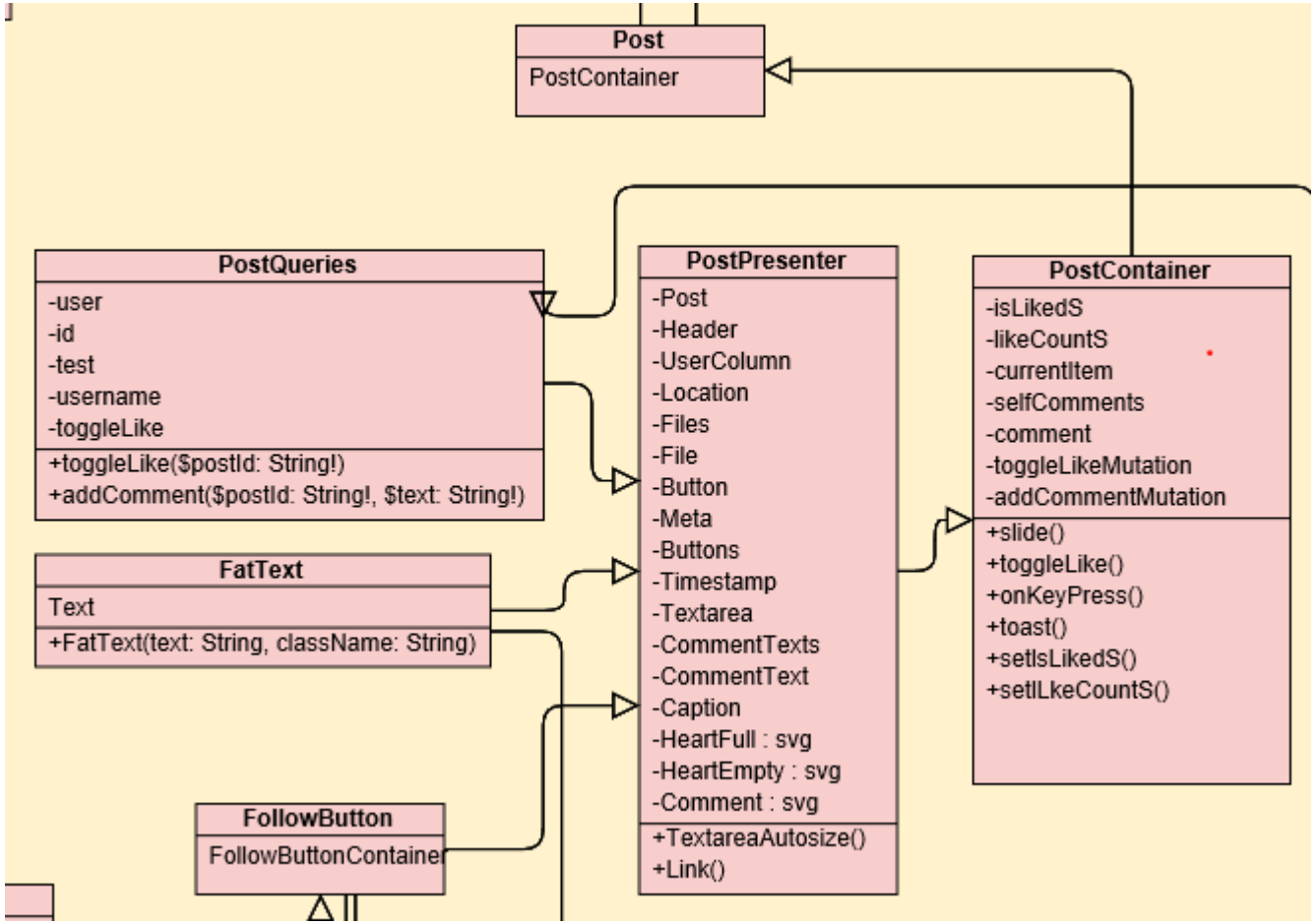

```
    text-transform: uppercase;
    font-weight: 600;
    font-size: 12px;
    margin: 50px 0px;
`

const List = styled.ul`
    display: flex;
`

const ListItem = styled.li`
    &:not(:last-child) {
        margin-right: 16px;
    }
`

const Link = styled.a`
    color: ${props => props.theme.darkBlueColor}
`

const Copyright = styled.span`
    color:${props => props.theme.darkGreyColor}
`

export default () => (
    <Footer>
        <List>
            <ListItem><Link href="#">정보</Link></ListItem>
            <ListItem><Link href="#">개인정보 처리방침</Link></ListItem>
            <ListItem><Link href="#">약관</Link></ListItem>
            <ListItem><Link href="#">디렉터리</Link></ListItem>
            <ListItem><Link href="#">프로필</Link></ListItem>
            <ListItem><Link href="#">해시 태그</Link></ListItem>
            <ListItem><Link href="#">언어</Link></ListItem>
        </List>
        <Copyright>
            VRR {new Date().getFullYear()} &copy;
        </Copyright>
    </Footer>
```


Web Server Client
LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Index
-apolloProvider: ApolloProvider
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Footer
-Footer
-List
-ListItem
-Link
-Copyright
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Search
-SearchContainer
Converter
-ConverterContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer
Post
PostContainer
SearchContainer
-term
-Wrapper
+useQuery()
SearchPresenter
-Wrapper
-Section
-PostSection
-da : Array
-dateArray : String
-Loader
+setDA()
+setDateArray()
SearchQueries
-id
-files
-url
-likeCount
-commentCOunt
-createdAt
-avatar
-username
-isFollowing
-isSelf
+search($term: String!)
+searchPost(term: $term)
+searchUser(term: $term)
ConverterContainer
-term
-Wrapper
+useQuery()
ConverterQueries
-fileupload
-id
-path
-filename
-mimetype
+fileDownLoad()
ProfileContainer
-Wrapper
-Header
-HeaderColumn
-UsernameRow
-Username
-Counts
-Count
-FullName
-Bio
-Posts
ProfilePresenter
-data
+logUserOut()
+useQuery()
+useMutation()
ProfileQueries
-id
-avatar
-username
-fullName
-isFollowing
-isSelf
-bio
-followingCount
-followersCount
-postsCount
-posts
-files
-url
-likeCount
-commentCount
+seeUser()
Apollo
-client : Client
-localState : LocalState
Hooks
-userInput : UserInput
Styles
-globalStyles : GlobalStyles
-theme : Theme
Routes
-auth : Auth
-explore : Explore
-file : File
-profile : Profile
-search : Search
-feed : Feed
-Notification
-Post
PostQueries
-user
-id
-test
-username
-toggleLike
+toggleLike($postId: String!)
+addComment($postId: String!, $text: String!)
PostPresenter
-Post
-Header
-UserColumn
-Location
-Files
-File
-Button
-Meta
-Buttons
-Timestamp
-Textarea
-CommentTexts
-CommentText
-Caption
-HeartFull : svg
-HeartEmpty : svg
-Comment : svg
+TextareaAutosize()
+Link()
PostContainer
-isLikedS
-likeCountS
-currentItem
-selfComments
-comment
-toggleLikeMutation
-addCommentMutation
+slide()
+toggleLike()
+onKeyPress()
+toast()
+setIsLikedS()
+setLkeCountS()
UserCard
-Card
-EAvatar
-ELink
+Link
FatText
Text
+FatText(text: String, className: String)
Loader
-Animation
-Loader
-Logo : Icon
SquarePost
-HeartFull : Icon
-CommentFull : Icon
-Overlay
-Container
-Number
-NumberText
useInput
+onChange(value: Object, onChange: Object, setValue: Object)
Avatar
-Container
+getSize()
Button
-Container
FollowButton
FollowButtonContaine
Component
-post : Post
-avatar : Avatar
-button : Button
-faxText : FaxText
-footer : Footer
-header : Header
-icons : Icons
-input : Input
-loader : Loader
-routers : Routers
-searchCard : SearchCard
-squareCardPost : SquareCardPost
-userCard : UserCard
FollowButtonContainer
-isFollowingS
-followMutation
-unfollowMutation
+onClick()
+setIsFollowing()
FollowButtonPresenter
-Button
FollowButtonQueries
+followUser($id: String!)
+unfollowUser($id: String!)


VRRFRONT
build
docs
node_modules
public
src
Apollo
Components
Post
App.js
Avatar.js
Button.js
FatText.js
Footer.js M
Header.js
Icons.js M
Input.js
Loader.js
Routes.js
SearchCard.js
SquarePost.js
UserCard.js
Hooks
Routes
Styles
index.css
index.js
SharedQueries.js
utils.scss
.gitignore
package.json
README.md
yarn.lock
import PropTypes from "prop-types"
import React from "react"
import { Route, Switch, Redirect } from "react-router-dom"
import Auth from "../Routes/Auth"
import Feed from "../Routes/Feed"
import Explore from "../Routes/Explore"
import Search from "../Routes/Search"
import Profile from "../Routes/Profile"
import Notification from "../Routes/Notification"
const LoggedInRoutes = () => (
<Switch>
<Route exact path="/" component={Feed}></Route>
<Route path="/explore" component={Explore} />
<Route path="/search" component={Search} />
<Route path="/notifications" component={Notification}/>
<Route path="/:username" component={Profile} />
<Redirect from="*" to='/' />
</Switch>
)
const LoggedOutRoutes = () => <Switch><Route exact path="/" component={Auth}></Route></Switch>
const AppRouter = ({ isLoggedIn }) => (
<Switch>{isLoggedIn ? <LoggedInRoutes/> : <LoggedOutRoutes/>}</Switch>
)
AppRouter.propTypes = {
isLoggedIn: PropTypes.bool.isRequired
}
export default AppRouter
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Converter
-ConverterContainer
Search
-SearchContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer


Web Server Client
LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Index
-apolloProvider: ApolloProvider
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Footer
-Footer
-List
-ListItem
-Link
-Copyright
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Search
-SearchContainer
Converter
-ConverterContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer
Post
PostContainer
Apollo
-client : Client
-localState : LocalState
Hooks
-userInput : UserInput
Styles
-globalStyles : GlobalStyles
-theme : Theme
Routes
-auth : Auth
-explore : Explore
-file : File
-profile : Profile
-search : Search
-feed : Feed
-Notification
-Post
PostQueries
-user
-id
-test
-username
-toggleLike
+toggleLike($postId: String!)
+addComment($postId: String!, $text: String!)
FatText
Text
FatText(text: String, className: String)
PostPresenter
-Post
-Header
-UserColumn
-Location
-Files
-File
-Button
-Meta
-Buttons
-Timestamp
-Textarea
-CommentTexts
-CommentText
-Caption
-HeartFull : svg
-HeartEmpty : svg
-Comment : svg
+TextareaAutosize()
+Link()
PostContainer
-isLikedS
-likeCountS
-currentItem
-selfComments
-comment
-toggleLikeMutation
-addCommentMutation
+slide()
+toggleLike()
+onKeyPress()
+toast()
+setIsLikedS()
+setLkeCountS()
FollowButton
FollowButtonContaine
Component
-post : Post
-avatar : Avatar
-button : Button
-faxText : FaxText
-footer : Footer
-header : Header
-icons : Icons
-input : Input
-loader : Loader
-routers : Routers
-searchCard : SearchCard
-squareCardPost : SquareCardPost
-userCard : UserCard
FollowButtonContainer
-isFollowingS
-followMutation
-unfollowMutation
+onClick()
+setIsFollowing()
FollowButtonPresenter
-Button
FollowButtonQueries
+followUser($id: String!)
+unfollowUser($id: String!)
SearchContainer
-term
-Wrapper
+useQuery()
SearchPresenter
-Wrapper
-Section
-PostSection
-da : Array
-dateArray : String
-Loader
+setDA()
+setDateArray()
SearchQueries
-id
-files
-url
-likeCount
-commentCOunt
-createdAt
-avatar
-username
-isFollowing
-isSelf
+search($term: String!)
+searchPost(term: $term)
+searchUser(term: $term)
ConverterContainer
-term
-Wrapper
+useQuery()
ConverterQueries
-fileupload
-id
-path
-filename
-mimetype
+fileDownLoad()
ProfileContainer
-Wrapper
-Header
-HeaderColumn
-UsernameRow
-Username
-Counts
-Count
-FullName
-Bio
-Posts
ProfilePresenter
-data
+logUserOut()
+useQuery()
+useMutation()
ProfileQueries
-id
-avatar
-username
-fullName
-isFollowing
-isSelf
-bio
-followingCount
-followersCount
-postsCount
-posts
-files
-url
-likeCount
-commentCount
+seeUser()
UserCard
-Card
-EAvatar
-ELink
+Link
Loader
-Animation
Loader
-ogo : Icon
SquarePost
-HeartFull : Icon
-CommentFull : Icon
-Overlay
-Container
-Number
-NumberText
useInput
+onChange(value: Object, onChange: Object, setValue: Object)
Avatar
-Container
+getSize()
Button
-Container


Web Server Client
LocalState
-isLoggedIn : Object
+logUserIn()
+logUserOut()
Index
-apolloProvider: ApolloProvider
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()
Client
-ApolloClient
-uri : String
-clientState : Object
-token : String
-headers : Object
+request()
App
-ThemeProvider
-GlobalStyles
-Router
-Wrapper
-ToastContainer
- isLoggedIn : Boolean
- Query : GQL
Routes
-AppRouter
+LoggedInRouter()
+LoggedOutRoutes()
Theme
-maxWidth
-bgColor
-blackColor
-darkGreyColor
-lightGreyColor
-redColor
-blueColor
-darkBlueColor
-boxBorder
-borderRadius
-whiteBox
Footer
-Footer
-List
-ListItem
-Link
-Copyright
Feed
-FEED_QUERY: GQL
-Wrapper
-loading
Search
-SearchContainer
Converter
-ConverterContainer
Notifications
-BillboardChart
-Wrapper
-Card
+Ranking()
Profile
-ProfileContainer
Post
PostContainer
SearchContainer
-term
-Wrapper
+useQuery()
SearchPresenter
-Wrapper
-Section
-PostSection
-da : Array
-dateArray : String
-Loader
+setDA()
+setDateArray()
SearchQueries
-id
-files
-url
-likeCount
-commentCOunt
-createdAt
-avatar
-username
-isFollowing
-isSelf
+search($term: String!)
+searchPost(term: $term)
+searchUser(term: $term)
ConverterContainer
-term
-Wrapper
+useQuery()
ConverterQueries
-fileupload
-id
-path
-filename
-mimetype
+fileDownLoad()
ProfileContainer
-Wrapper
-Header
-HeaderColumn
-UsernameRow
-Username
-Counts
-Count
-FullName
-Bio
-Posts
ProfilePresenter
-data
+logUserOut()
+useQuery()
+useMutation()
ProfileQueries
-id
-avatar
-username
-fullName
-isFollowing
-isSelf
-bio
-followingCount
-followersCount
-postsCount
-posts
-files
-url
-likeCount
-commentCount
+seeUser()
Apollo
-client : Client
-localState : LocalState
Hooks
-userInput : UserInput
Styles
-globalStyles : GlobalStyles
-theme : Theme
Routes
-auth : Auth
-explore : Explore
-file : File
-profile : Profile
-search : Search
-feed : Feed
-Notification
-Post
PostQueries
-user
-id
-test
-username
-toggleLike
+toggleLike($postId: String!)
+addComment($postId: String!, $text: String!)
FatText
Text
+FatText(text: String, className: String)
PostPresenter
-Post
-Header
-UserColumn
-Location
-Files
-File
-Button
-Meta
-Buttons
-Timestamp
-Textarea
-CommentTexts
-CommentText
-Caption
-HeartFull : svg
-HeartEmpty : svg
-Comment : svg
+TextareaAutosize()
+Link()
PostContainer
-isLikedS
-likeCountS
-currentItem
-selfComments
-comment
-toggleLikeMutation
-addCommentMutation
+slide()
+toggleLike()
+onKeyPress()
+toast()
+setIsLikedS()
+setLkeCountS()
UserCard
Card
Avatar
ink
k
Loader
nation
der
o : Icon
SquarePost
-HeartFull : Icon
-CommentFull : Icon
-Overlay
-Container
-Number
-NumberText
useInput
+onChange(value: Object, onChange: Object, setValue: Object)
Avatar
-Container
+getSize()
Button
-Container
FollowButton
FollowButtonContaine
Component
-post : Post
-avatar : Avatar
-button : Button
-faxText : FaxText
-footer : Footer
-header : Header
-icons : Icons
-input : Input
-loader : Loader
-routers : Routers
-searchCard : SearchCard
-squareCardPost : SquareCardPost
-userCard : UserCard
FollowButtonContainer
-isFollowingS
-followMutation
-unfollowMutation
+onClick()
+setIsFollowing()
FollowButtonPresenter
-Button
FollowButtonQueries
+followUser($id: String!)
+unfollowUser($id: String!)



JS index.js

```
import PostContainer from './PostContainer'
export default PostContainer
```

JS PostQueries.js

```
import { gql } from 'apollo-boost'

export const TOGGLE_LIKE = gql`
    mutation toggleLike($postId: String!){
        toggleLike(postId: $postId)
    }
`

export const ADD_COMMENT = gql`
    mutation addComment($postId: String!, $text: String!){
        addComment(postId: $postId, text:$text){
            id
            text
            user {
                username
            }
        }
    }
`
```

JS PostContainer.js

```
    const [isLikedS, setIsLiked] = useState(isLiked)
    const [likeCountS, setLikeCount] = useState(likeCount)
    const [currentItem, setCurrentItem] = useState(0)
    const [selfComments, setSelfComments] = useState([])
    const comment = useInput('')
    const [toggleLikeMutation] = useMutation(TOGGLE_LIKE, {
        variables: {postId: id}
    })
    const [addCommentMutation] = useMutation(ADD_COMMENT,{
        variables: {postId: id, text: comment.value}
    })

return (
    <PostPresenter
        user={user}
        files={files}
        likeCount={likeCountS}
        isLiked={isLikedS}
        comments={comments}
        caption={caption}
        location={location}
        createdAt={createdAt}
        newComment={comment}
        setIsLiked={setIsLiked}
        setLikeCount={setLikeCount}
        currentItem={currentItem}
        toggleLike={toggleLike}
        onKeyPress={onKeyPress}
        selfComments={selfComments}
    />
)
```

JS PostPresenter.js

```
export default({user:{username, avatar}, location, caption, files, isLik
<Post>
    <Header>
        <Avatar size='sm' url={avatar} />
        <UserColumn>
            <Link to={`/${username}`}>
                <FatText text={username}/>
            </Link>
            <Location>{location}</Location>
        </UserColumn>
    </Header>
    <Caption>{caption}</Caption>
    <Files>
        {files && files.map( (file, index) =><File key={file.id} src={fi
    </Files>
    <Meta>
        <Buttons>
            <Button onClick={toggleLike}>
                {isLiked ? <HeartFull/> : <HeartEmpty/>}
            </Button>
            <Button>
                <Comment/>
            </Button>
        </Buttons>
        <FatText text={likeCount === 1 ? '1 like': `${likeCount} likes`}
        {comments && (
            <CommentTexts>
                {comments.map(comment =>(
                    <CommentText key={comment.id}>
                        <FatText text={comment.user.username} />
                        {comment.text }
                    </CommentText>
                ))}
                {selfComments.map(comment =>(
                    <CommentText key={comment.id}>
                        <FatText text={comment.user.username} />
                        {comment.text }
                    </CommentText>
                ))
```

감사합니다.