

중간결과 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

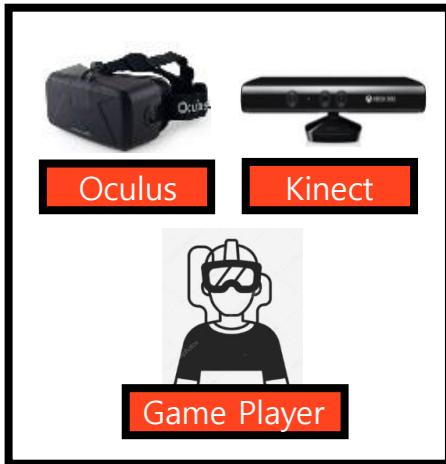
목차

1. 개요
2. 음원 전처리
3. CNN_LSTM Model
4. Unity VR Game
5. Server

01 개요

사용자 요구 사항 상세 설계

VR Game Player



VR Game Player



1 VR Game Play

Unity로 기능 구현

2 점수, 랭킹, 음악 노트 확인

웹 플랫폼 & 서버 구현

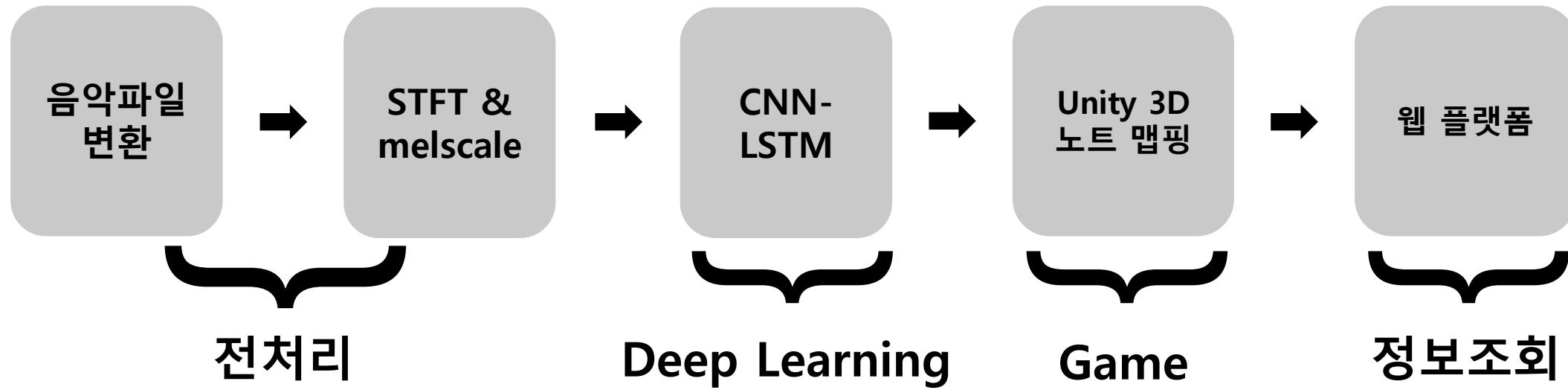
3 비트 노트 맵 제작

음악파일 → 비트 노트 맵

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

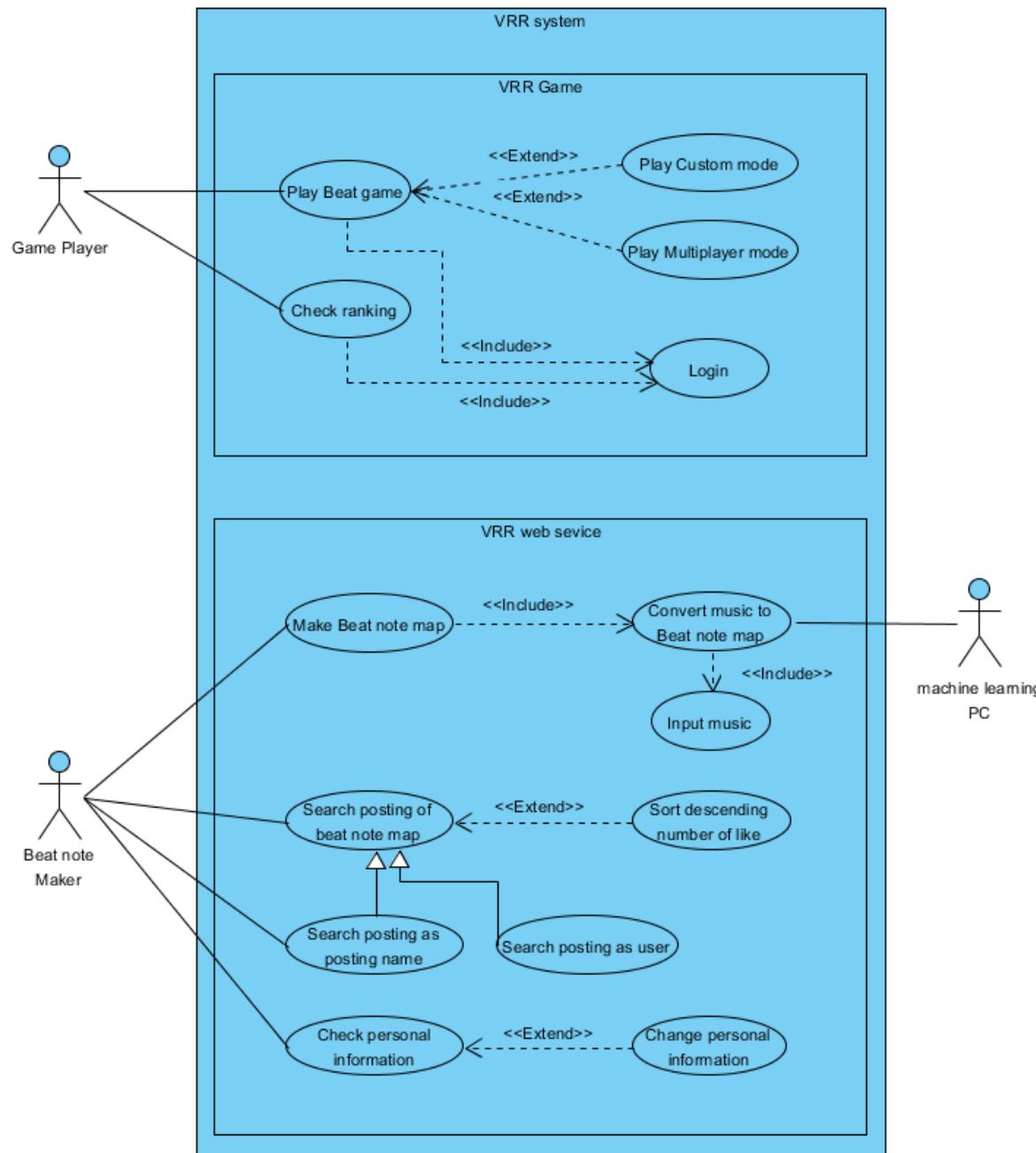
01 개요

주요 요소 기술

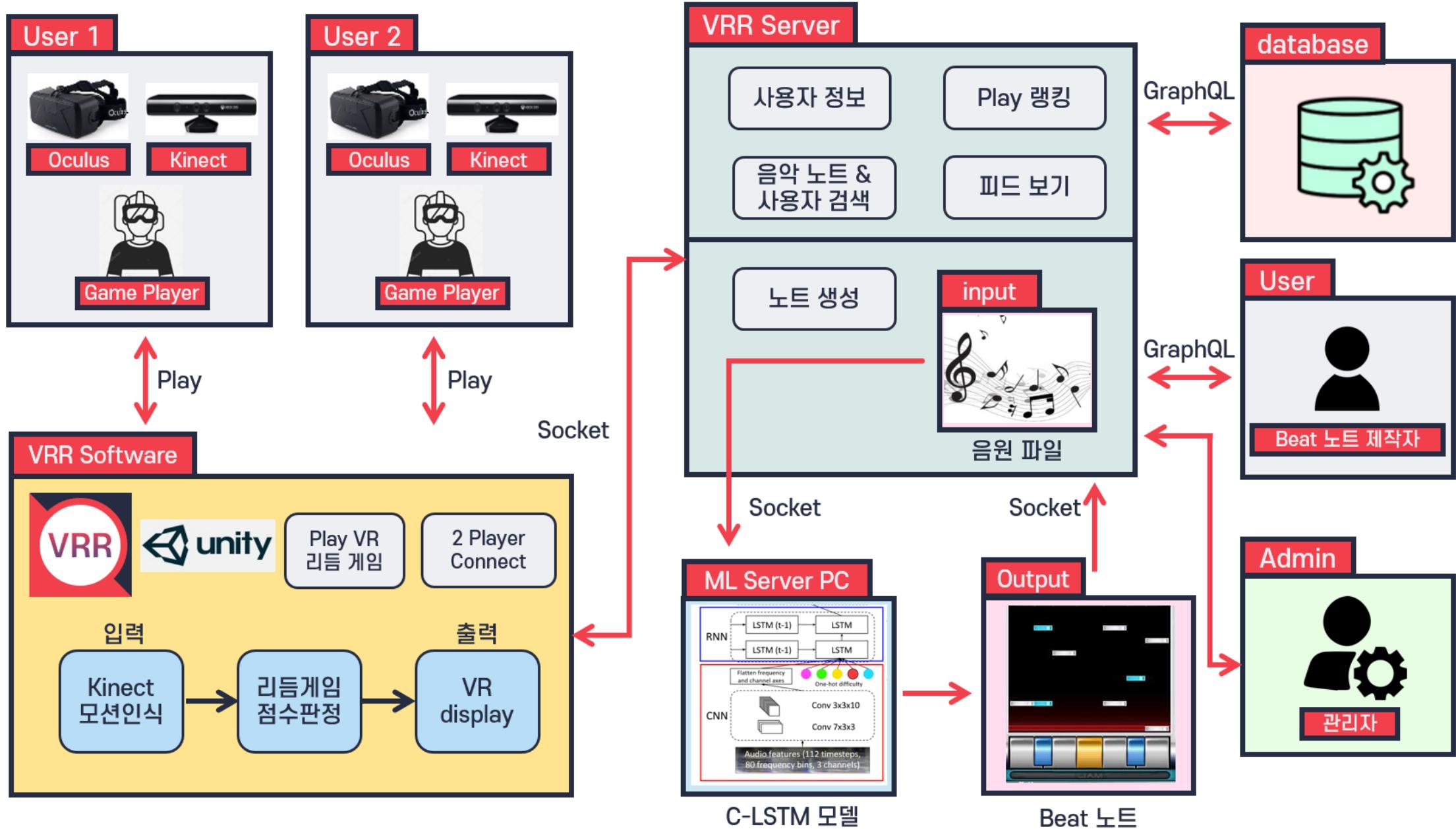


01 개요

Use Case Diagram



01 개요



02 음원 전처리

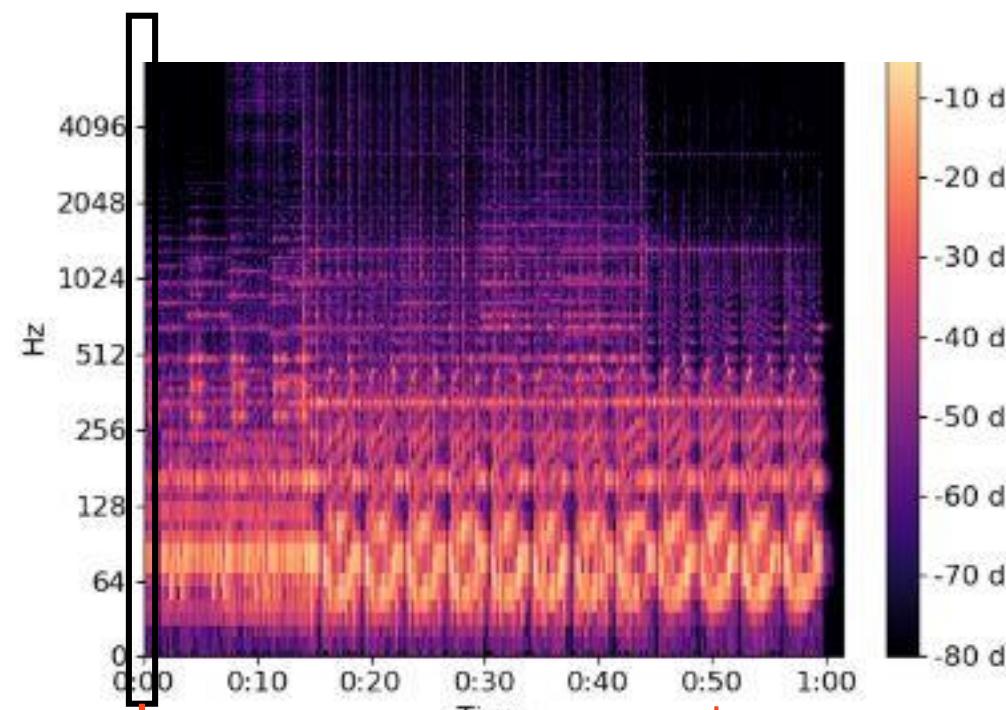
전처리 및 Deep Learning 변경사항

- 1채널에 대한 전처리 stft → melscale에서 3채널에 대한 전처리 stft->melscale->sampling
- 1채널 CNN 모델-> 3채널 CNN-LSTM
- 한번에 들어가는 음악파일 크기 조정 20초->10초->2.5초 (비교 중)

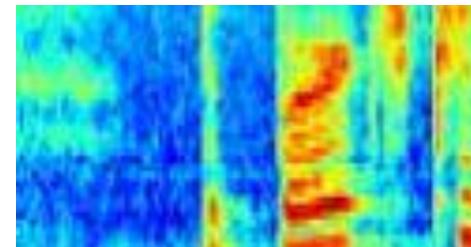
02 음원 전처리

전처리 과정

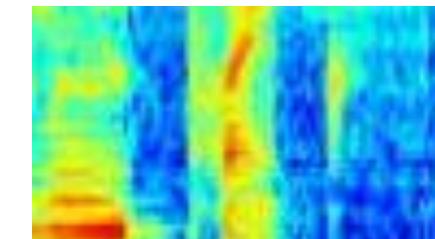
(1) Stft (None, feature)



(2) 3.6ms 단위로 melscale(1, 80)



(None, 80)



(3) stft,melscale이 완료된 3개의 채널을 합친다

02 음원 전처리

전처리 클래스

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

02 음원 전처리

코드 설명

```
class preprocess:
    def __init__(self, music):
        self.music = music

    def stft(self, music, n_fft_size):
        self.music = music
        samples, sample_rate = librosa.load(path=music, sr = 44100, mono = True, duration = 3)
        D = np.abs(librosa.stft(samples, n_fft=n_fft_size, hop_length=160))
        return D

    def mel_scale(self):
        data = [0]*3
        channel = self.three_channel()
        for k in range(0, len(channel)):
            T = np.transpose(channel[k])
            mfccs = np.matrix(librosa.feature.mfcc(y = T[0], n_mfcc=80))
            mel_scale = np.transpose(mfccs)
            for i in range(1, len(T)):
                mfccs = librosa.feature.mfcc(y = T[i], n_mfcc=80)
                mfccs_t = np.transpose(mfccs)
                mel_scale = np.vstack((mel_scale, mfccs_t))
            arr = np.array(mel_scale)
            data[k] = arr.tolist()
        return data
```

02 음원 전처리

코드 설명

```
def three_channel(self):
    #368, 736, 1488
    music = self.music
    D1 = self.stft(music,368)
    D2 = self.stft(music,736)
    D3 = self.stft(music,1488)
    #return len(D1), len(D2), len(D3)
    return [D1,D2,D3]
```

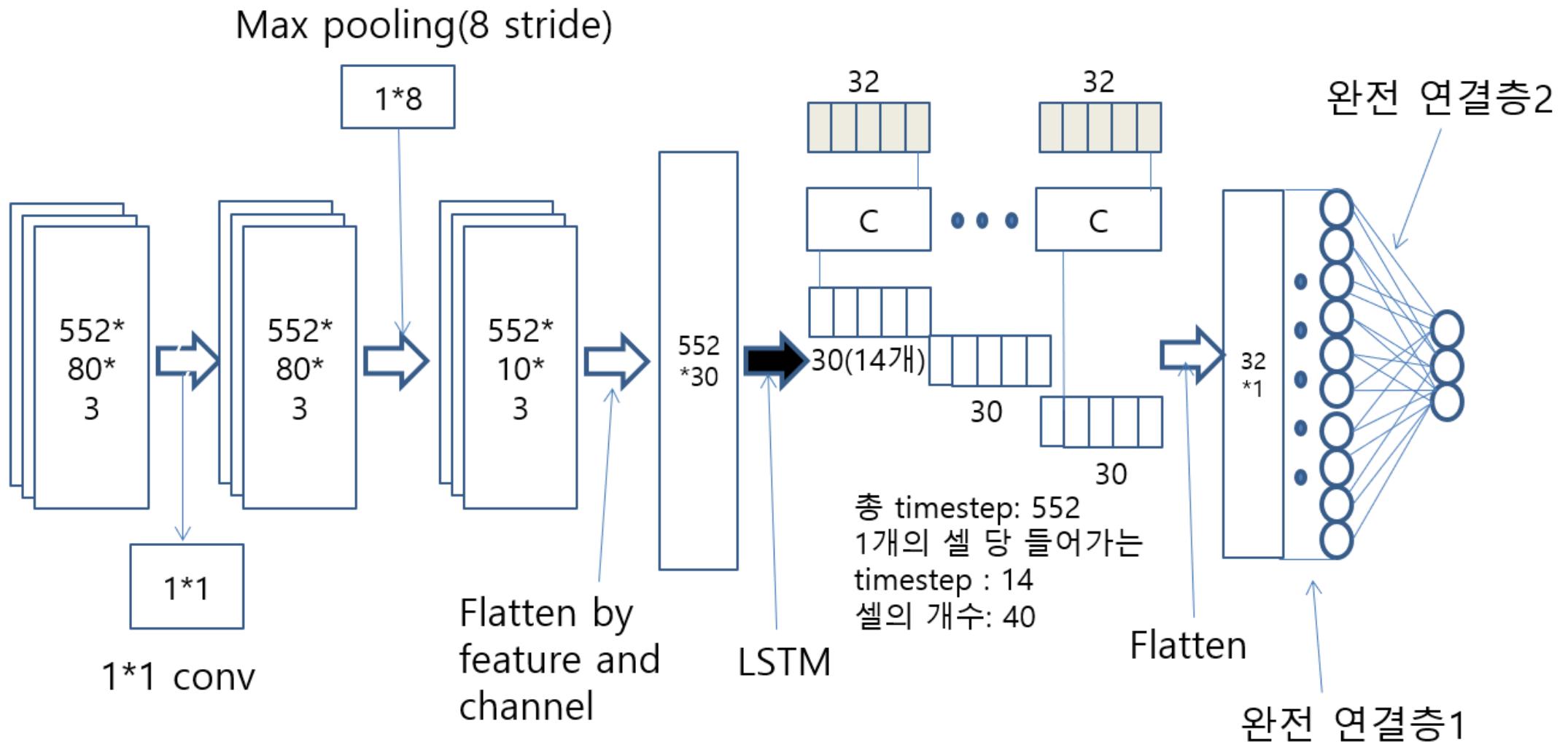
```
def preprocess_output(self):
    melscale_data = self.mel_scale()
    channel_one = [melscale_data[0][0]]
    channel_two = [melscale_data[1][0]]
    channel_three = [melscale_data[2][0]]

    T = np.vstack((channel_one, channel_two, channel_three))
    tmp = np.vstack((channel_one, channel_two, channel_three))
    new_T = np.vstack(([T], [tmp]))

    for i in range(1,int(len(melscale_data[0])/2)):
        channel_one = [melscale_data[0][i]]
        channel_two = [melscale_data[1][i]]
        channel_three = [melscale_data[2][i]]
        tmp = np.vstack((channel_one, channel_two, channel_three))
        new_T = np.vstack((new_T,[tmp]))
    return new_T
```

03 CNN_LSTM Model

CNN_LSTM 모델 1



03 CNN_LSTM Model

Feature 위주의 conv, maxpooling 적용

Layer (type)	Output Shape	Param #
conv2d_64 (Conv2D)	(None, 80, 3, 1)	2
max_pooling2d_63 (MaxPooling)	(None, 10, 3, 1)	0
flatten_56 (Flatten)	(None, 30)	0
reshape_36 (Reshape)	(None, 1, 30)	0
lstm_30 (LSTM)	(None, 1, 32)	8064
dropout_8 (Dropout)	(None, 1, 32)	0
flatten_57 (Flatten)	(None, 32)	0
dense_43 (Dense)	(None, 32)	1056
dense_44 (Dense)	(None, 3)	99
Total params:	9,221	
Trainable params:	9,221	
Non-trainable params:	0	

03 CNN_LSTM Model

코드 설명

```
from sklearn.model_selection import train_test_split

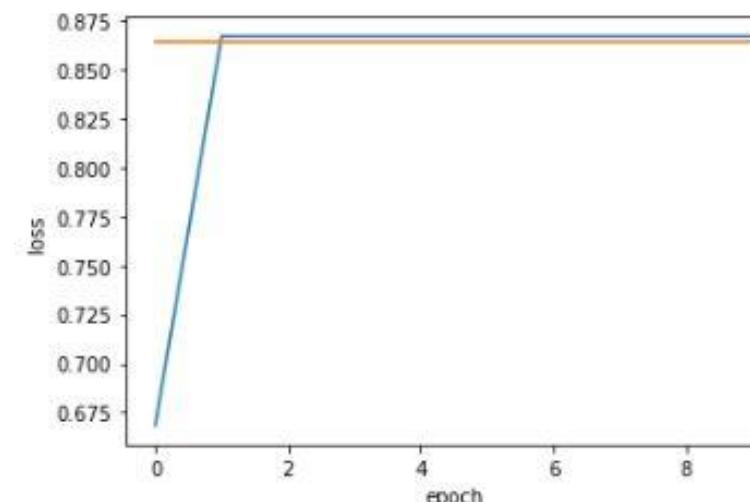
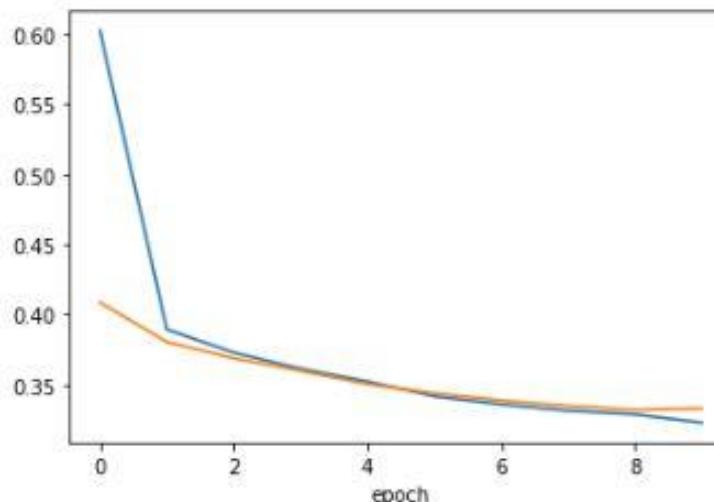
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import LSTM
from keras.layers import Dense, Dropout, Activation, Flatten

model = Sequential()
model.add(Conv2D(1,(1,1), activation='relu', padding = 'same', input_shape =(80,3,1)))
model.add(MaxPooling2D((8,1)))
model.add(Flatten())
model.add(Reshape((1,30)))
model.add(LSTM(32,batch_input_shape=(None,14,3),return_sequences=True))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(32, activation='sigmoid'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.build((None,80,3,1))
model.summary()
```

03 CNN_LSTM Model

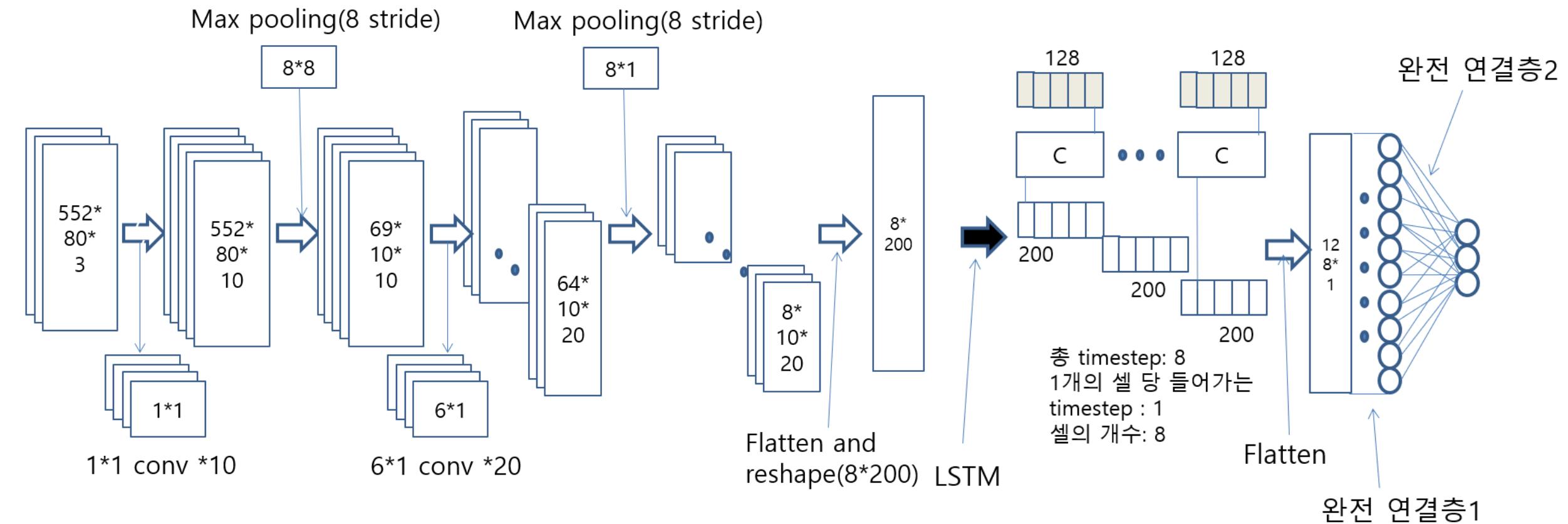
실행 결과

```
552/552 [=====] - 2s 3ms/step - loss: 0.6027 - acc: 0.6685 - val_loss: 0.4086 - val_acc: 0.8647
Epoch 2/10
552/552 [=====] - 0s 822us/step - loss: 0.3893 - acc: 0.8671 - val_loss: 0.3804 - val_acc: 0.8647
Epoch 3/10
552/552 [=====] - 0s 733us/step - loss: 0.3728 - acc: 0.8671 - val_loss: 0.3687 - val_acc: 0.8647
Epoch 4/10
552/552 [=====] - 0s 816us/step - loss: 0.3613 - acc: 0.8671 - val_loss: 0.3598 - val_acc: 0.8647
Epoch 5/10
552/552 [=====] - 0s 772us/step - loss: 0.3522 - acc: 0.8671 - val_loss: 0.3503 - val_acc: 0.8647
Epoch 6/10
552/552 [=====] - 0s 760us/step - loss: 0.3415 - acc: 0.8671 - val_loss: 0.3438 - val_acc: 0.8647
Epoch 7/10
552/552 [=====] - 0s 741us/step - loss: 0.3359 - acc: 0.8671 - val_loss: 0.3386 - val_acc: 0.8647
Epoch 8/10
552/552 [=====] - 0s 750us/step - loss: 0.3313 - acc: 0.8671 - val_loss: 0.3347 - val_acc: 0.8647
Epoch 9/10
552/552 [=====] - 0s 742us/step - loss: 0.3287 - acc: 0.8671 - val_loss: 0.3318 - val_acc: 0.8647
Epoch 10/10
552/552 [=====] - 0s 750us/step - loss: 0.3228 - acc: 0.8671 - val_loss: 0.3330 - val_acc: 0.8647
```



03 CNN_LSTM Model

CNN_LSTM 모델 2



03 CNN_LSTM Model

Time_axis 위주의 conv, maxpooling 적용

Layer (type)	Output Shape	Param #
conv2d_59 (Conv2D)	(None, 552, 80, 10)	40
max_pooling2d_58 (MaxPooling)	(None, 69, 10, 10)	0
conv2d_60 (Conv2D)	(None, 69, 10, 20)	1220
max_pooling2d_59 (MaxPooling)	(None, 8, 10, 20)	0
flatten_49 (Flatten)	(None, 1600)	0
reshape_32 (Reshape)	(None, 8, 200)	0
lstm_27 (LSTM)	(None, 8, 128)	168448
flatten_50 (Flatten)	(None, 1024)	0
dense_37 (Dense)	(None, 128)	131200
dense_38 (Dense)	(None, 3)	387
=====		
Total params: 301,295		
Trainable params: 301,295		
Non-trainable params: 0		

03 CNN_LSTM Model

코드 설명

```
from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import LSTM
from keras.layers import Dense, Dropout, Activation, Flatten

model2 = Sequential()
model2.add(Conv2D(10,(1,1), activation= 'relu', padding = 'same', input_shape =(552,80,3)))
model2.add(MaxPooling2D((8,8)))
model2.add(Conv2D(20,(6,1), activation= 'relu', padding = 'same', input_shape =(64,10,20)))
model2.add(MaxPooling2D((8,1)))
model2.add(Flatten())
model2.add(Reshape((8,200)))

model2.add(LSTM(128,batch_input_shape=(None,2,3),return_sequences=True))
model2.add(Flatten())
model2.add(Dense(128, activation='sigmoid'))
model2.add(Dense(3, activation='sigmoid'))
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model2.build((552,80,3))
model2.summary()
```

03 CNN_LSTM Model

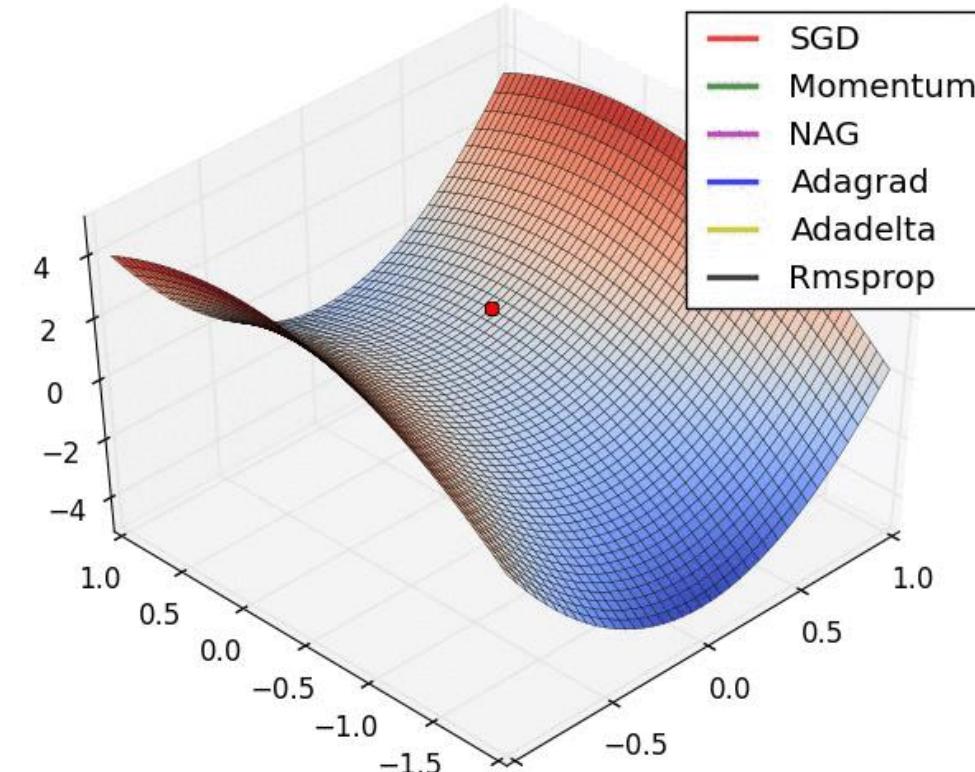
모델의 성능과 일반화 능력 향상을 위한 방법

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

03 CNN_LSTM Model

Optimizer

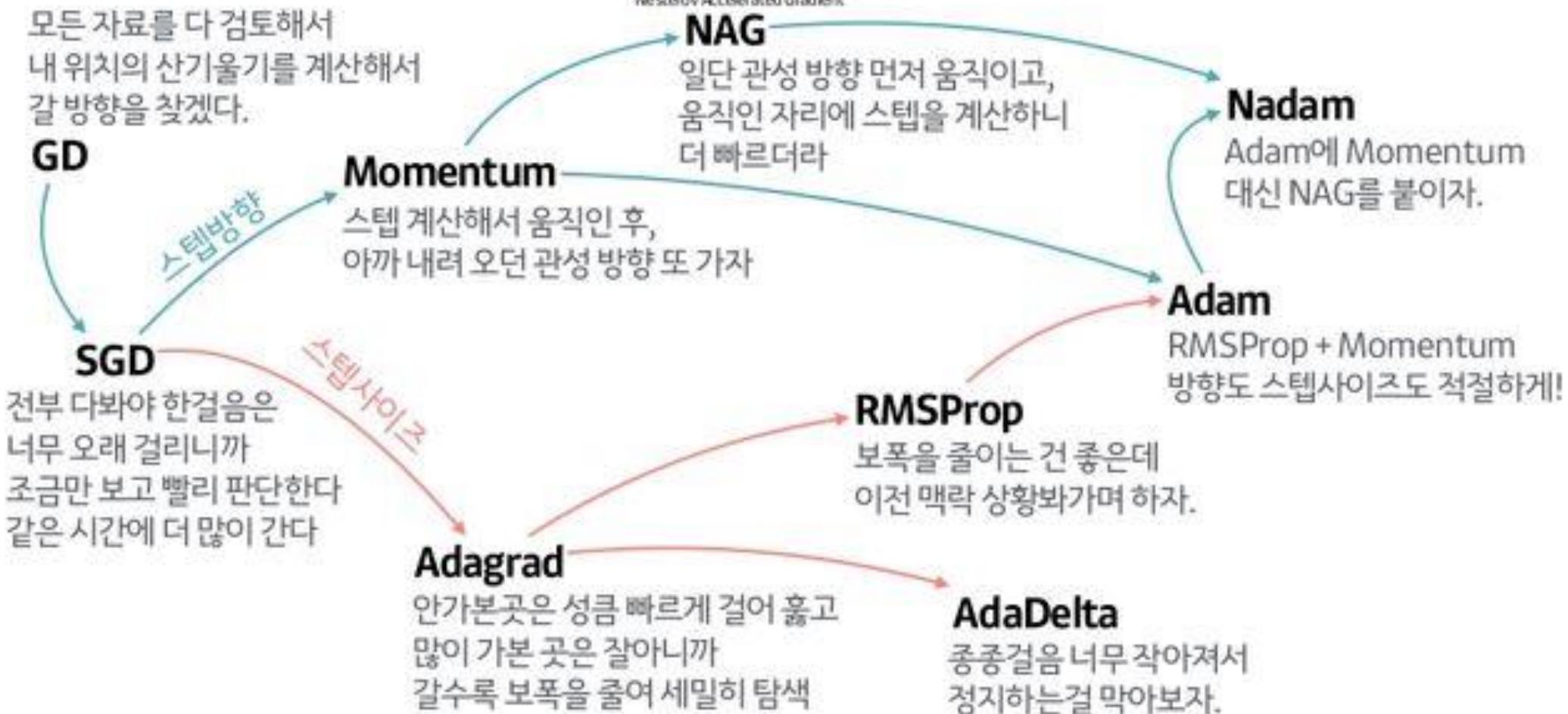
- 목표: 학습속도를 빠르고 안정적으로 만든다.
- Rmsprop
- Adagrad
- Adam



출처 : Gradient Descent Optimization Algorithms

03 CNN_LSTM Model

Optimizer



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

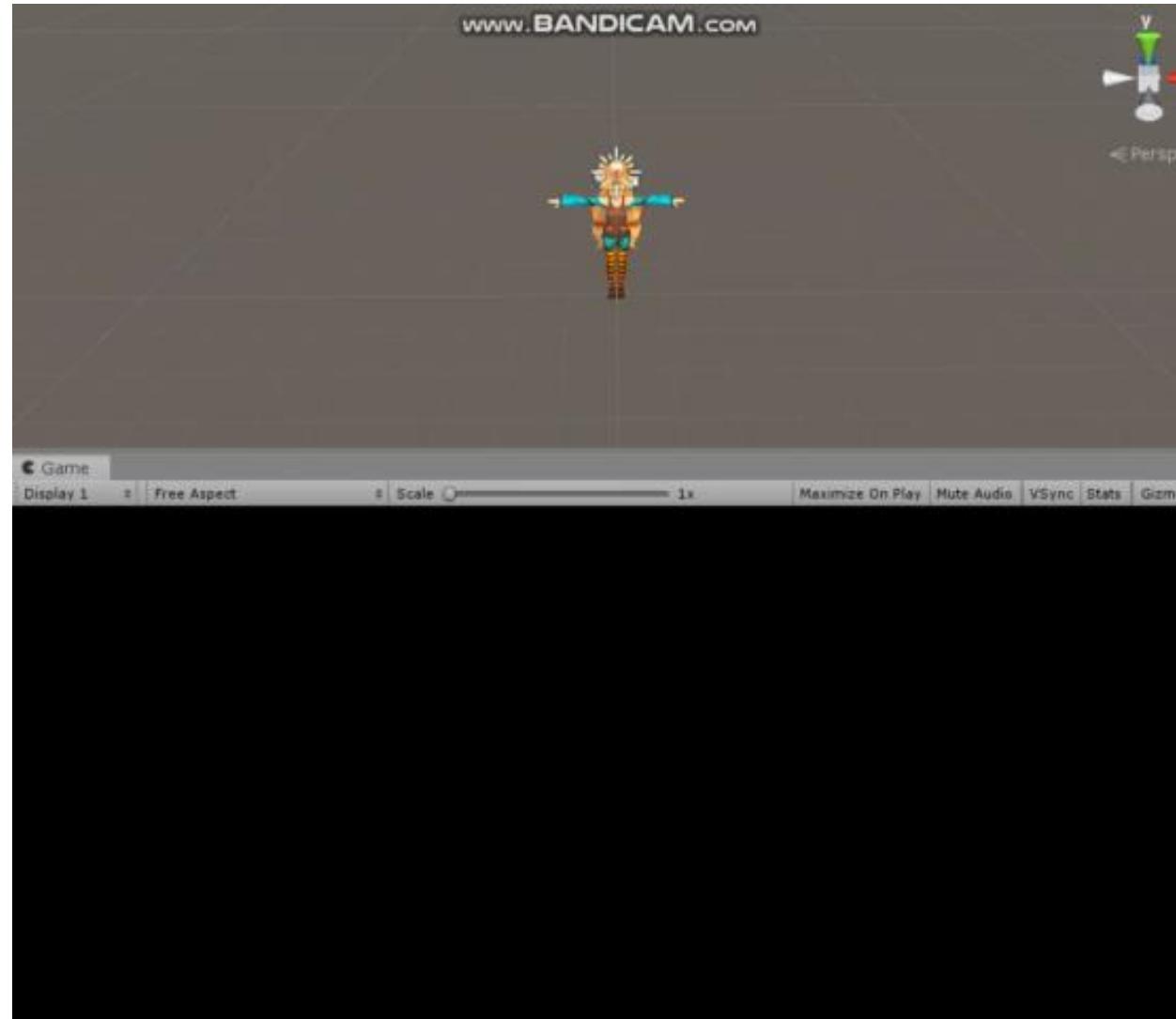
03 CNN_LSTM Model

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



04 Unity Game

전처리 된 음악 파일



Directory structure

```
| -Cover.jpg  
| -info.dat  
| -Normal.dat  
| -Hard.dat  
| -Expert.dat  
| -song.egg
```

Hard.dat - 메모장

```
{"_version":"2.0.0","_BPMChanges":[],"_events":[{"_time":0,"_type":4,"_value":3},{"_time":1,"_ty  
{"_time":28,"_type":4,"_value":3},{["_time":28.75,"_type":1,"_value":7},{["_time":29.5,"_type":1,"  
_value":3},{["_time":60.75,"_type":0,"_value":3},{["_time":61.5,"_type":0,"_value":3},{["_time":61.5,  
5,"_type":1,"_value":7},{["_time":76,"_type":1,"_value":7},{["_time":76,"_type":8,"_value":0},{["_tim  
94,"_type":1,"_value":7},{["_time":94.75,"_type":1,"_value":7},{["_time":95.5,"_type":1,"_value":7],  
"_value":0},{["_time":116,"_type":9,"_value":0},{["_time":116,"_type":8,"_value":0},{["_time":120,"_  
}, {"_time":138,"_type":3,"_value":2},{["_time":138.25,"_type":2,"_value":2},{["_time":138.25,"_type  
_type":3,"_value":5},{["_time":156,"_type":0,"_value":1},{["_time":156,"_type":1,"_value":1},{["_tim  
_value":0},{["_time":176,"_type":3,"_value":0},{["_time":176,"_type":0,"_value":0},{["_time":176,"_  
_value":0},{["_time":192,"_type":9,"_value":0},{["_time":192,"_type":0,"_value":1},{["_time":192,"_  
1,"_value":3},{["_time":210,"_type":0,"_value":7},{["_time":210,"_type":1,"_value":7},{["_time":210.  
}],
```

Normal.dat - 메모장

```
{"_version":"2.0.0","_BPMChanges":[],"_events":[{"_time":0,"_type":4,"_value":3},{"_time":1,"_ty  
{"_time":28,"_type":4,"_value":3},{["_time":28.75,"_type":1,"_value":7},{["_time":29.5,"_type":1,"  
_value":3},{["_time":60.75,"_type":0,"_value":3},{["_time":61.5,"_type":0,"_value":3},{["_time":61.5,  
5,"_type":1,"_value":7},{["_time":76,"_type":1,"_value":7},{["_time":76,"_type":8,"_value":0},{["_tim  
94,"_type":1,"_value":7},{["_time":94.75,"_type":1,"_value":7},{["_time":95.5,"_type":1,"_value":7],  
"_value":0},{["_time":116,"_type":9,"_value":0},{["_time":116,"_type":8,"_value":0},{["_time":120,"_  
}, {"_time":138,"_type":3,"_value":2},{["_time":138.25,"_type":2,"_value":2},{["_time":138.25,"_type  
_type":3,"_value":5},{["_time":156,"_type":0,"_value":1},{["_time":156,"_type":1,"_value":1},{["_tim  
_value":0},{["_time":176,"_type":3,"_value":0},{["_time":176,"_type":0,"_value":0},{["_time":176,"_  
_value":0},{["_time":192,"_type":9,"_value":0},{["_time":192,"_type":0,"_value":1},{["_time":192,"_  
1,"_value":3},{["_time":210,"_type":0,"_value":7},{["_time":210,"_type":1,"_value":7},{["_time":210.  
}],
```

04 Unity Game

전처리 된 음악 파일 분석

강남스타일 export.dat 분석

```
{  
    "_version": "2.0.0",  
  
    /*배경 효과 이벤트*/  
    "_events": [{"_time": 0, "_type": 0, "_value": 1}....]  
    time      시간  
    type      배경변화 유형  
    value     배경 값
```

```
/*배치된 비트 노트에 관한 데이터*/  
    "_notes": [{"_time": 5, "_lineIndex": 2, "_lineLayer": 0, "_type": 1, "_cutDirection": 1}  
    ....]  
    time      시간  
    lineIndex 좌우위치 (왼쪽부터 0123)  
    lineLayer 상하위치 (밑에서부터 012)  
    type      노트 색 (빨 : 0, 파 : 1)  
    cutdirection 자르는 방향  
                (아래 : 0 위 : 1 오른쪽 : 2 왼쪽 : 3 대각선 : 4, 5, 6, 7)
```

/*장애물에 관한 데이터*/
 "_obstacles": [{"_time": 39.75, "_lineIndex": 2, "_type": 0, "_duration": 1.5, "_width": 2}....]
 time 시간
 lineindex 시작위치
 type 장애물 유형
 duration 지속시간
 width 장애물 넓이

04 Unity Game

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;
    }

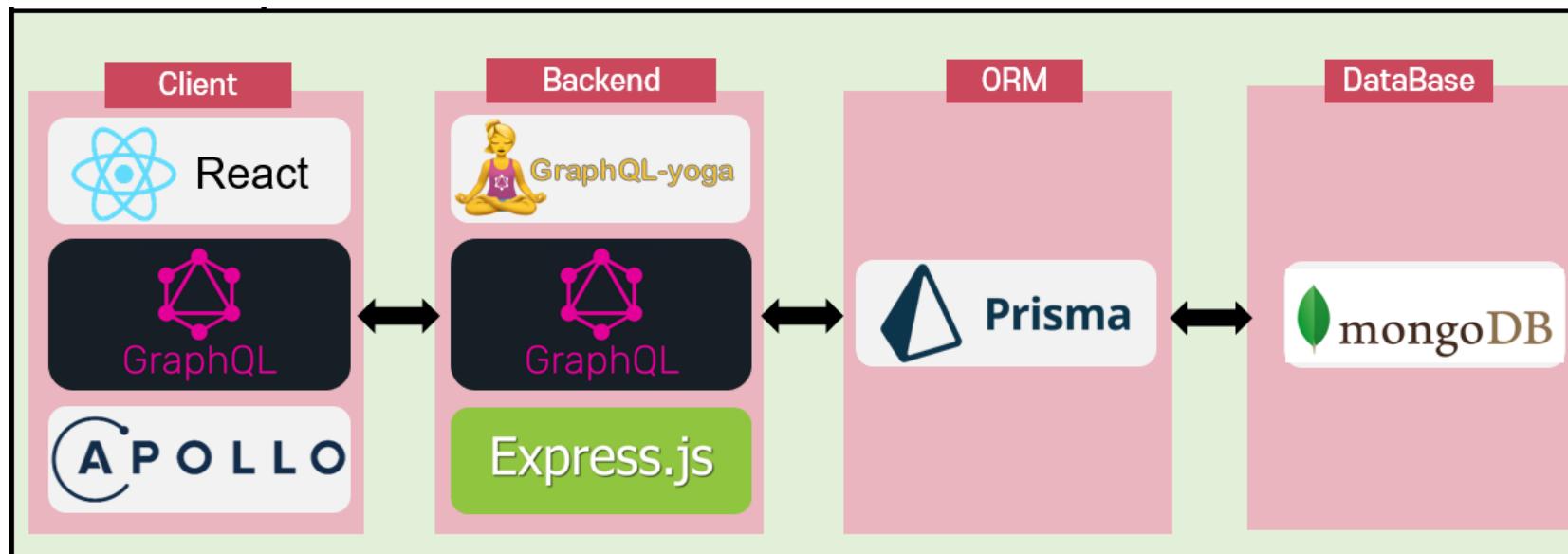
    // Get each BeatNote data and make notes data list
    public static List<BeatNoteClass> parse_data(JObject jdata)
    {
        List<BeatNoteClass> data = new List<BeatNoteClass>();

        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



100

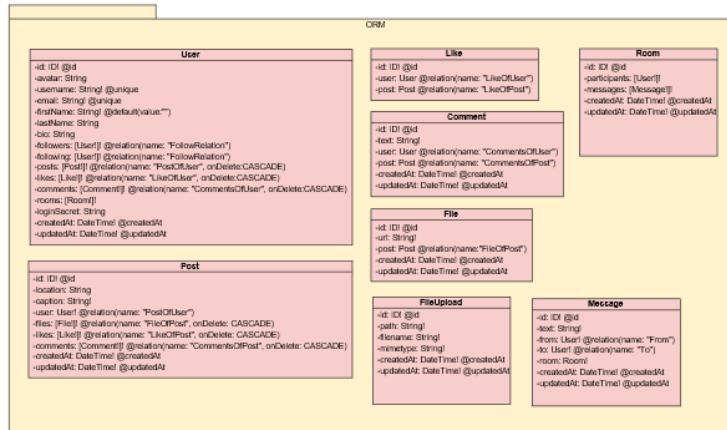
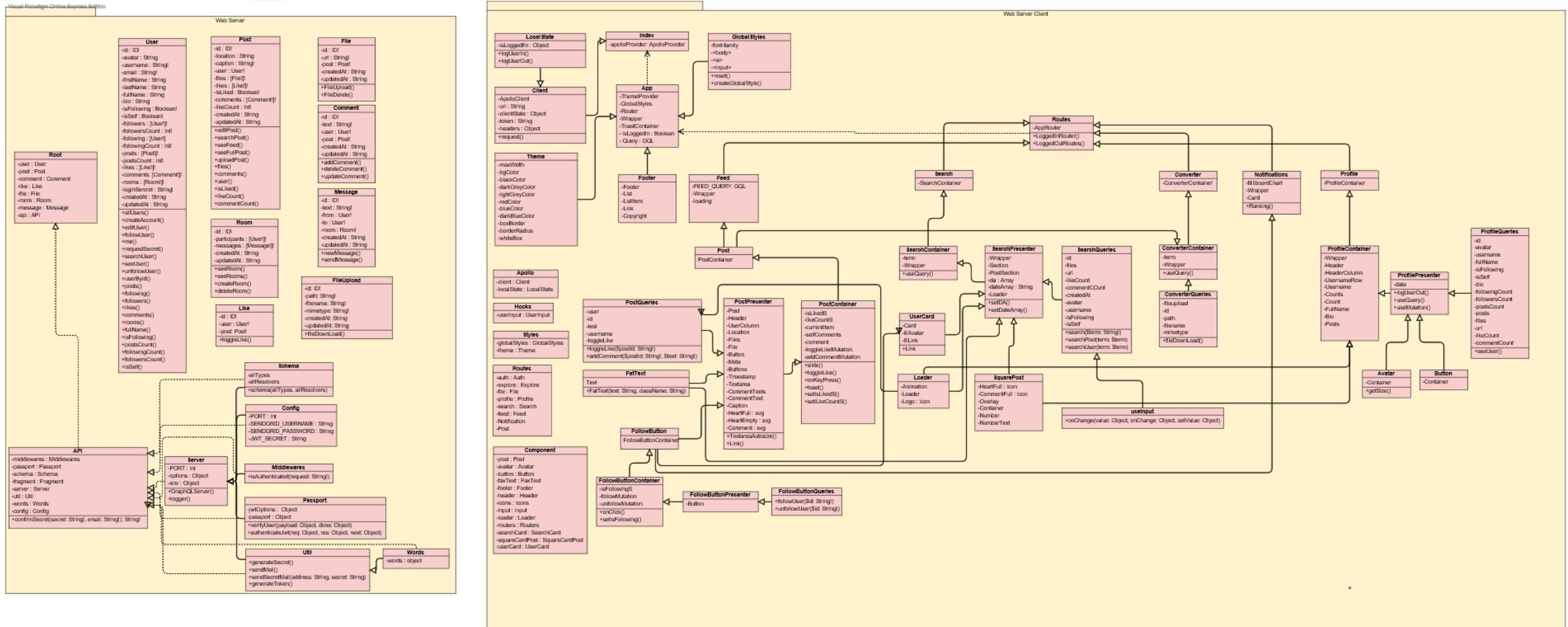
Search

 Report
 Resolution

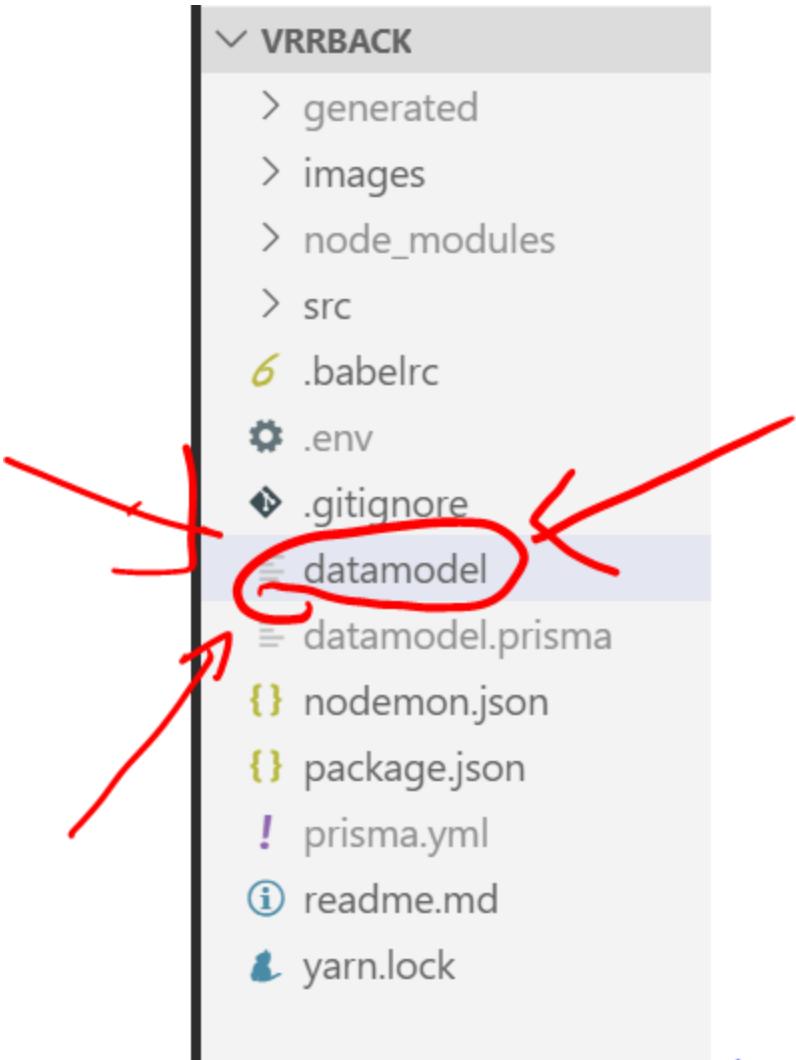
Drop file about document



0 likes



ORM



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", onDelete:CASCADE)
-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", onDelete: CASCADE)
-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  
}
```

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
```

FileUpload

```
-id: ID! @id  
-path: String!  
-filename: String!  
-mimetype: String!  
-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

App 한국외국어 대학교 GraphQL Mutation... MongoDB Mong... GraphQL Schema... Daily Life | Trello Microsoft PowerPo... apollographql/grap... 아풀로·부스트 마이...

VRR 1 service running

Services Servers Settings

vrrBack / dev New Query Rerun

Post > 0 Post

ID!	String	String!	User!
id	location	caption	user

Metrics Deployment History Playground

New Query Search queries..

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

HTTP ENDPOINT

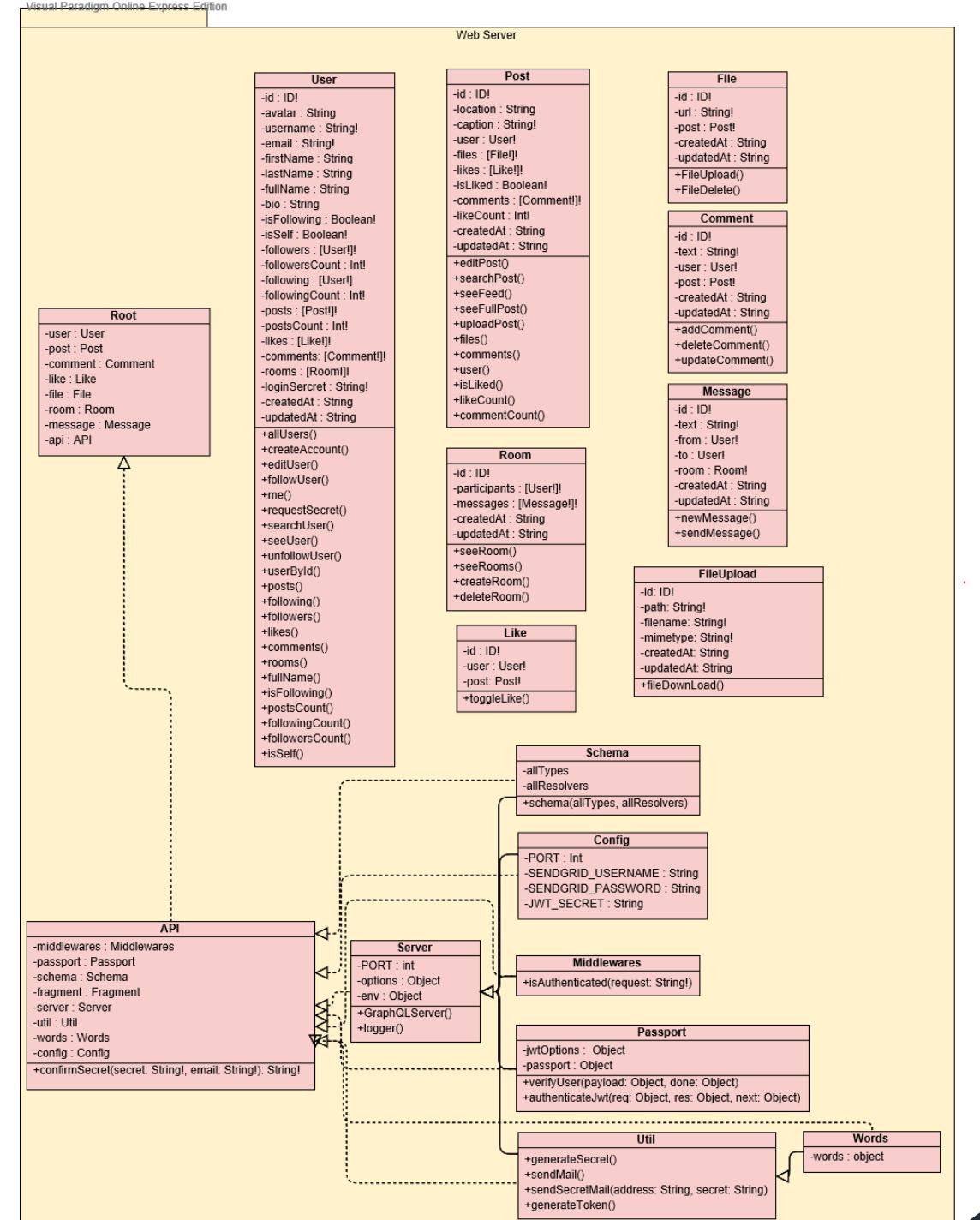
<https://us1.prisma.sh/vrr/vrrBack>

WS ENDPOINT

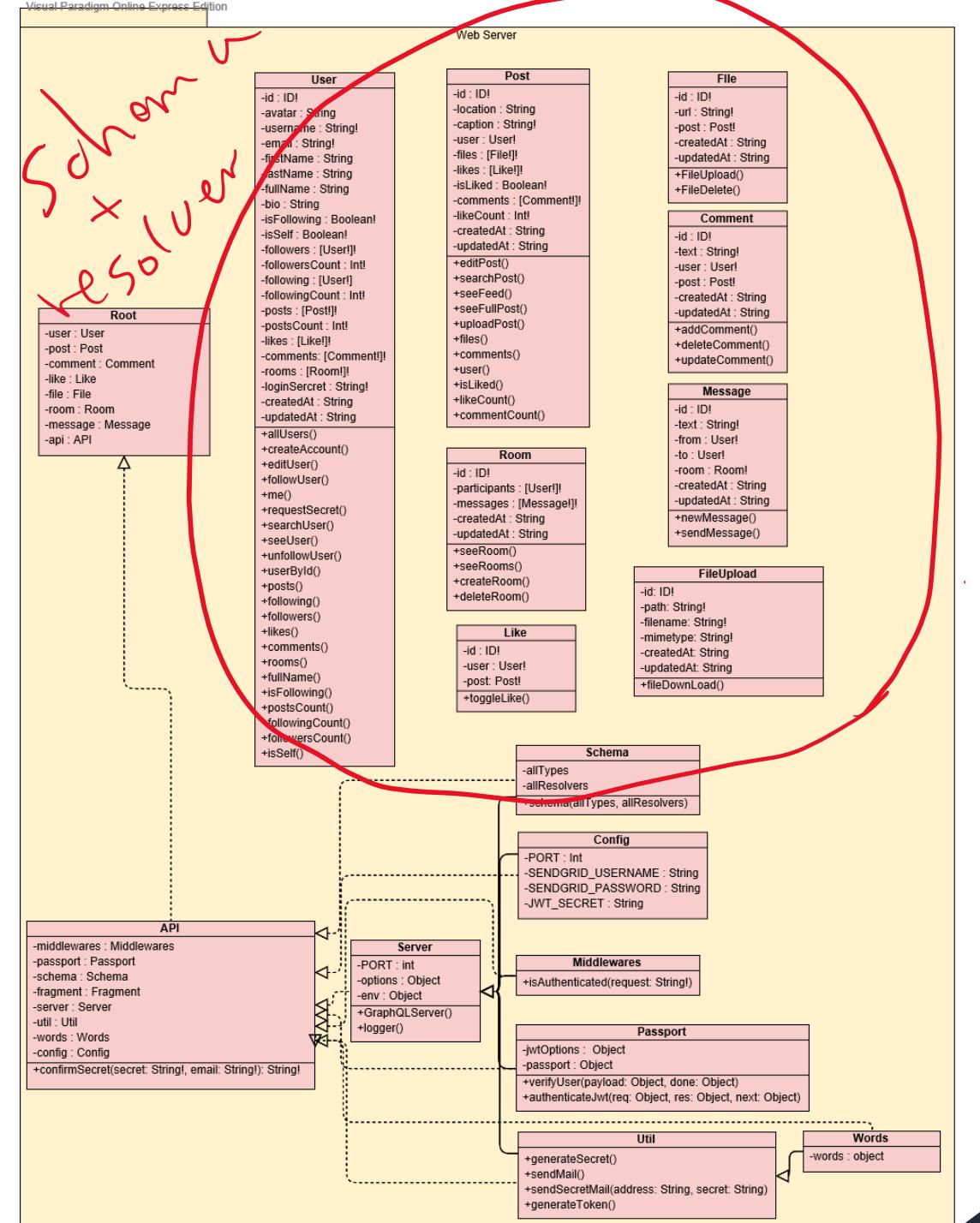
<wss://us1.prisma.sh/vrr/vrrBack/>

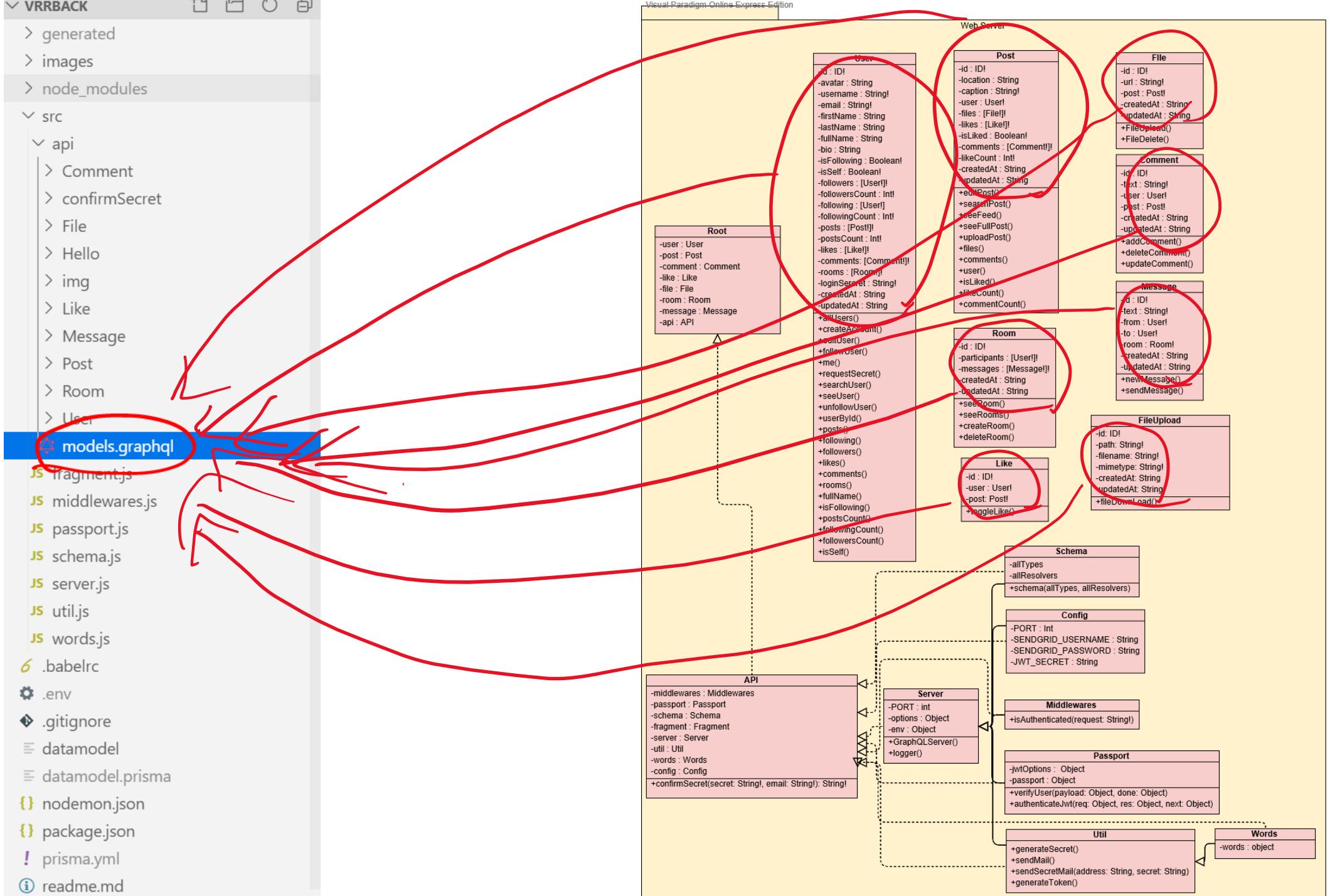


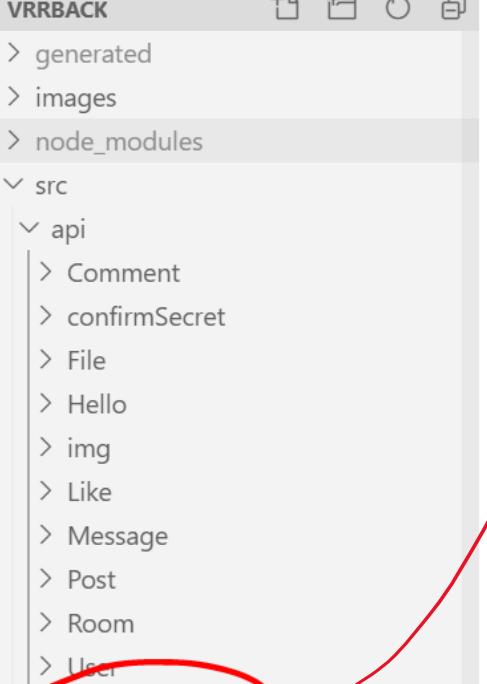
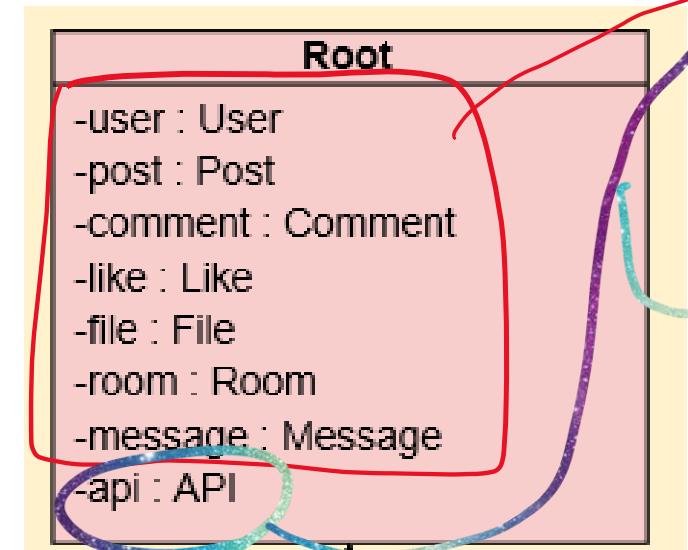
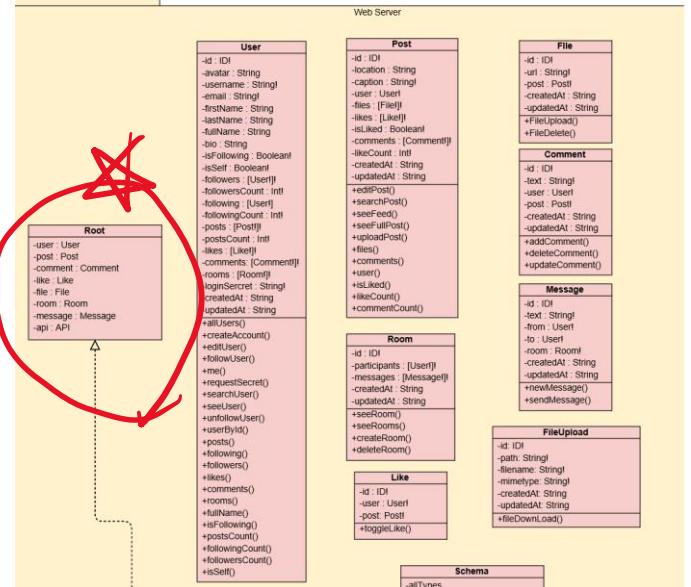
Web Server



Web Server







```
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!]!
  loginSecret: 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

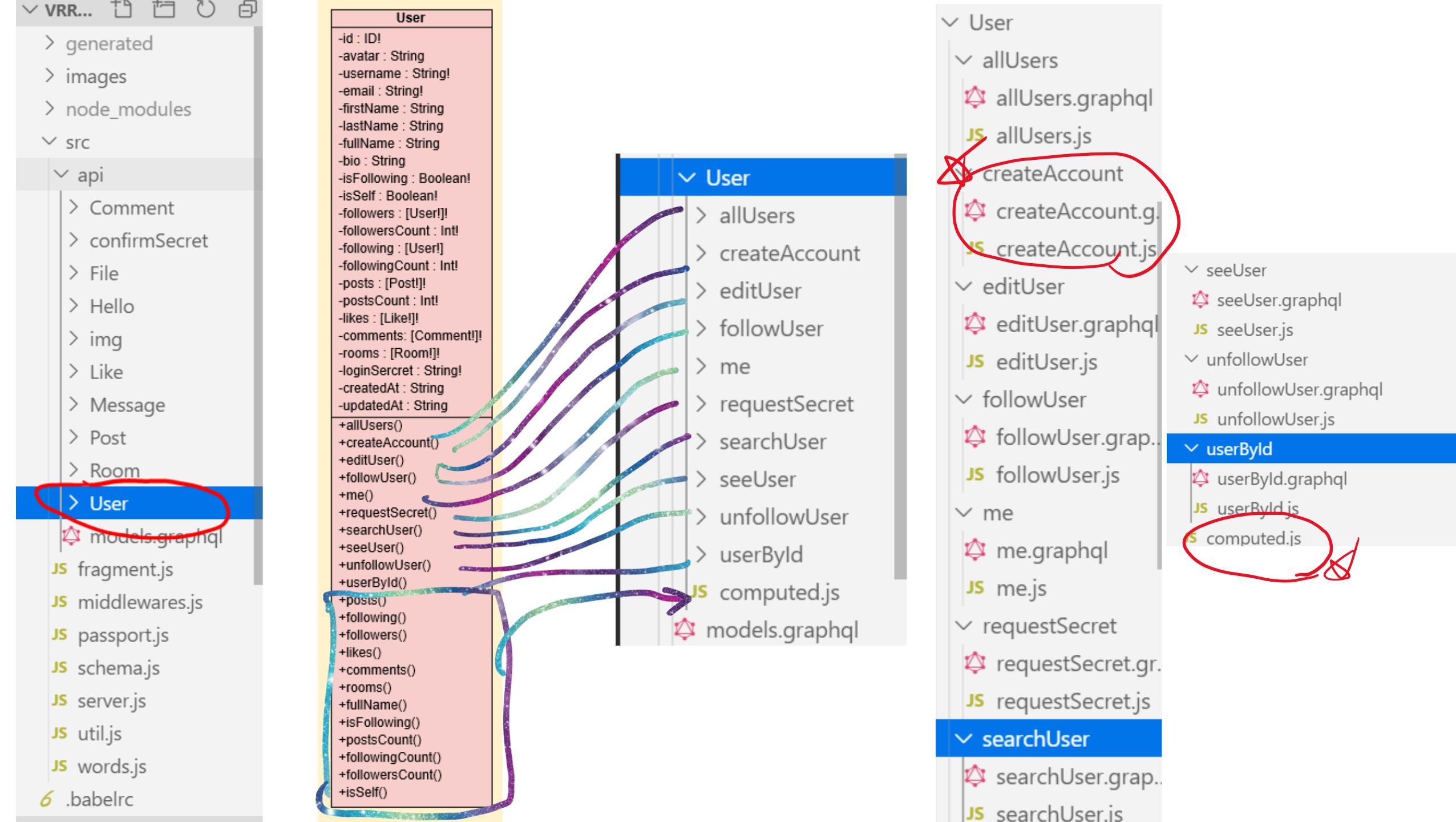
```
type User {  
    id: ID! @id  
    avatar: String  
        @default (  
            value: "https://icon-library.net/images/default-user-icon/default-user-icon-4.jpg"  
        )  
    username: String! @unique  
    email: String! @unique  
    firstName: String! @default(value:"")  
    lastName: 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 graph

```
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  
}
```



createAccount.graphql

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

createAccount.js

```
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
    }
  }
}
```

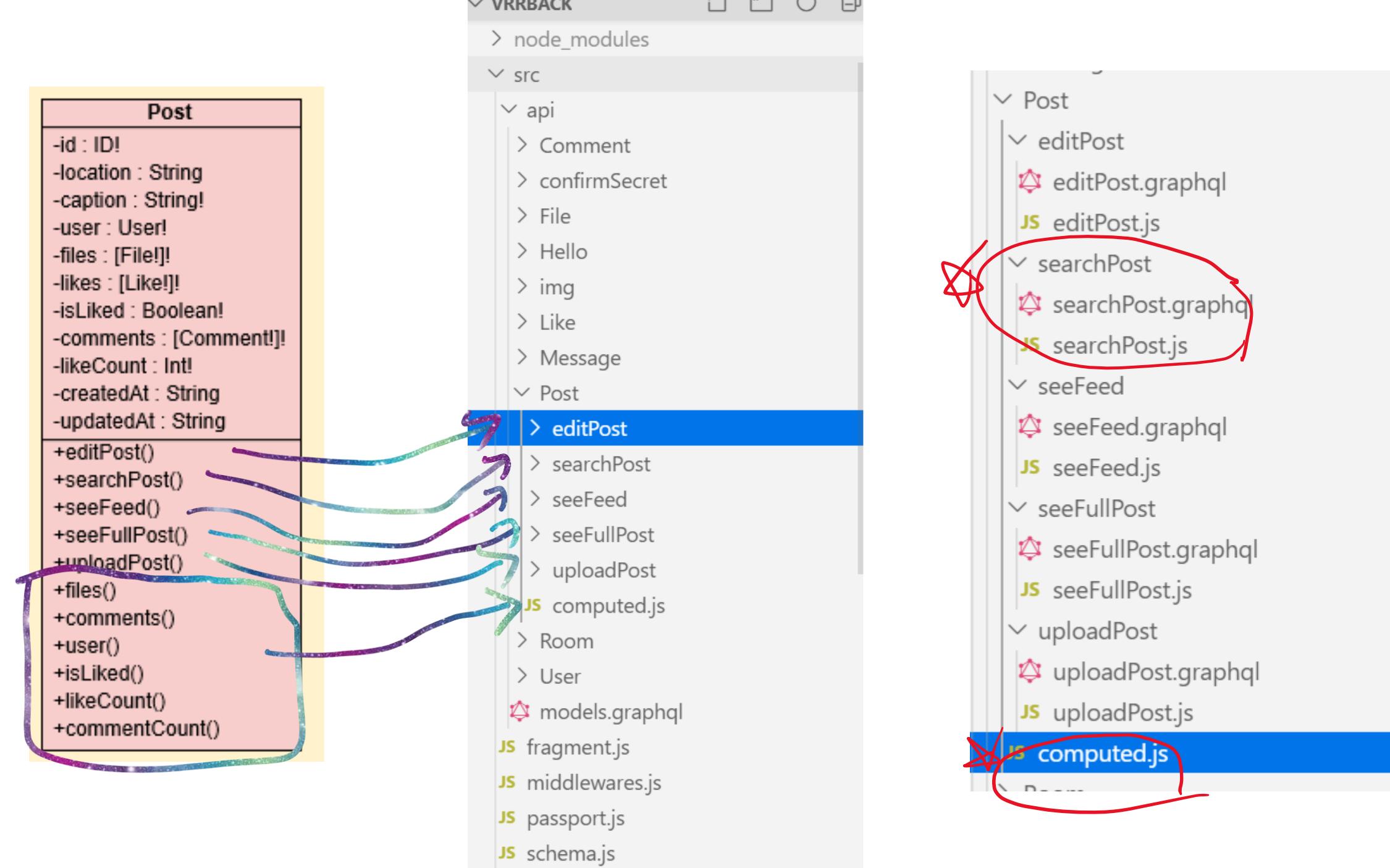
JS computed.js

```
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
      }
    },
  },
}
```

```
+posts()
+following()
+followers()
+likes()
+comments()
+rooms()
+fullName()
+isFollowing()
+postsCount()
+followingCount()
+followersCount()
+isSelf()
```

```
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
}
```



searchPost.graphql

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

JS searchPost.js

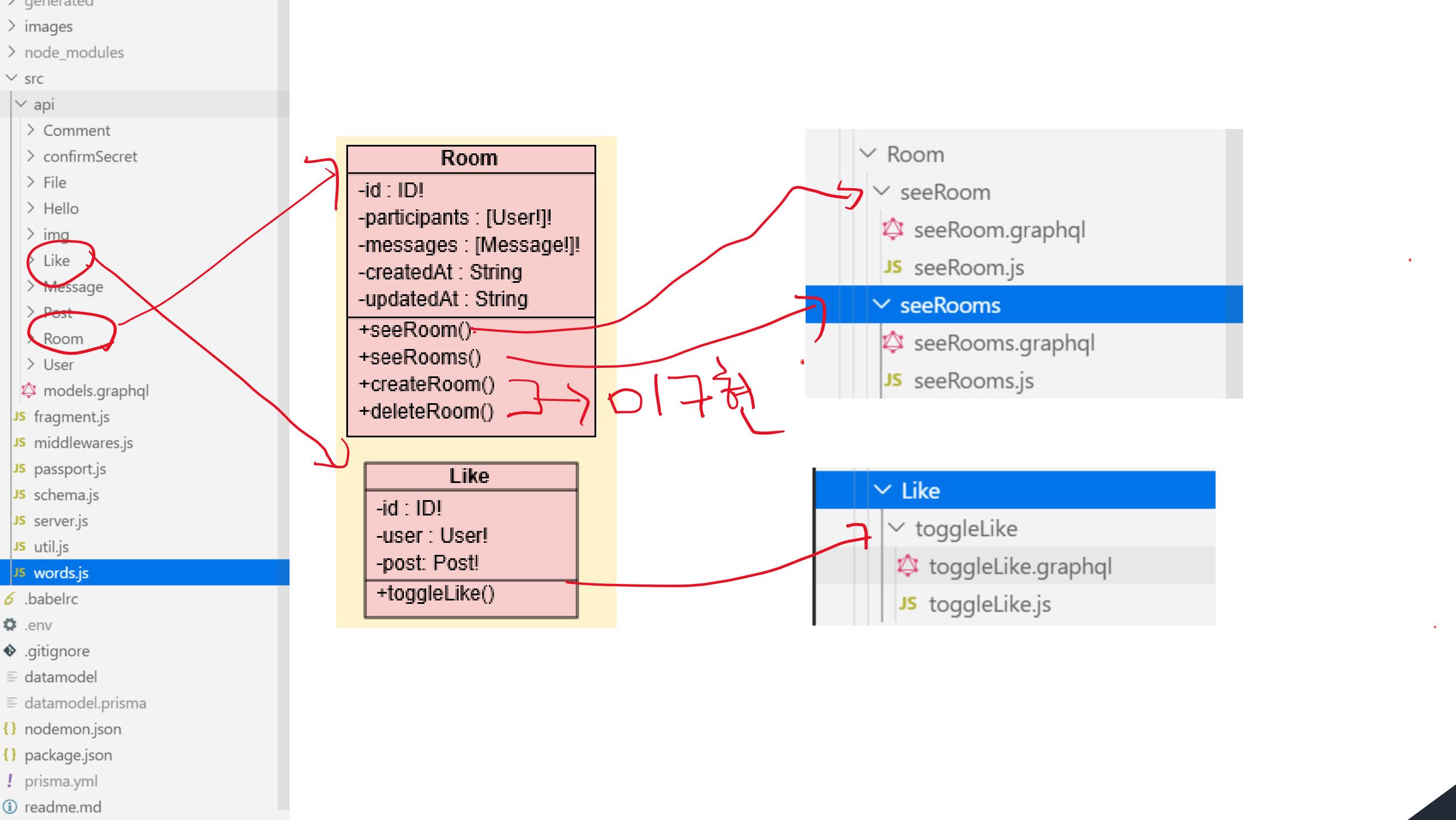
```
import { prisma } from "../../../../../generated/prisma-client";  
  
export default {  
  Query: {  
    searchPost: async(_, args) => prisma.posts(  
      { where:  
        {  
          OR: [  
            { location_starts_with: args.term },  
            { caption_starts_with: args.term }  
          ]  
        }  
      }  
    )  
  }  
}
```

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()
  }
}
```



toggleLike.graphql

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

JS toggleLike.js

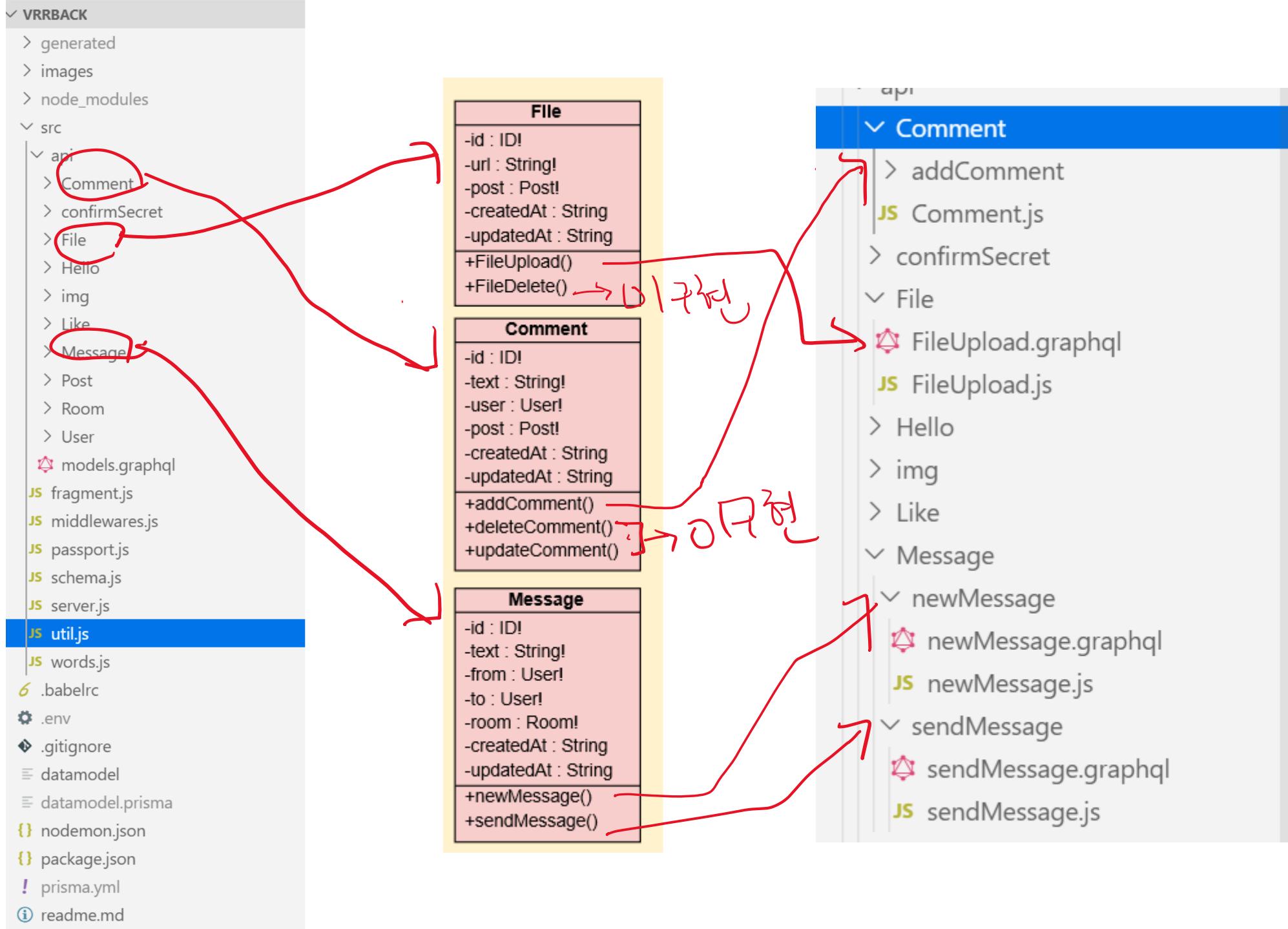
```
import { isAuthenticated } from "../../middlewares"  
import { prisma } from "../../generated/prisma-client"  
  
export default {  
  Mutation: {  
    toggleLike: async (_, args, { request }) => {  
      // passport user identity  
      isAuthenticated(request)  
      const { postId } = args  
      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!  
}
```

JS 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")  
      }  
    }  
  }  
}
```



FileUpload.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!
}
```

FileUpload.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  
    }  
  }  
}
```

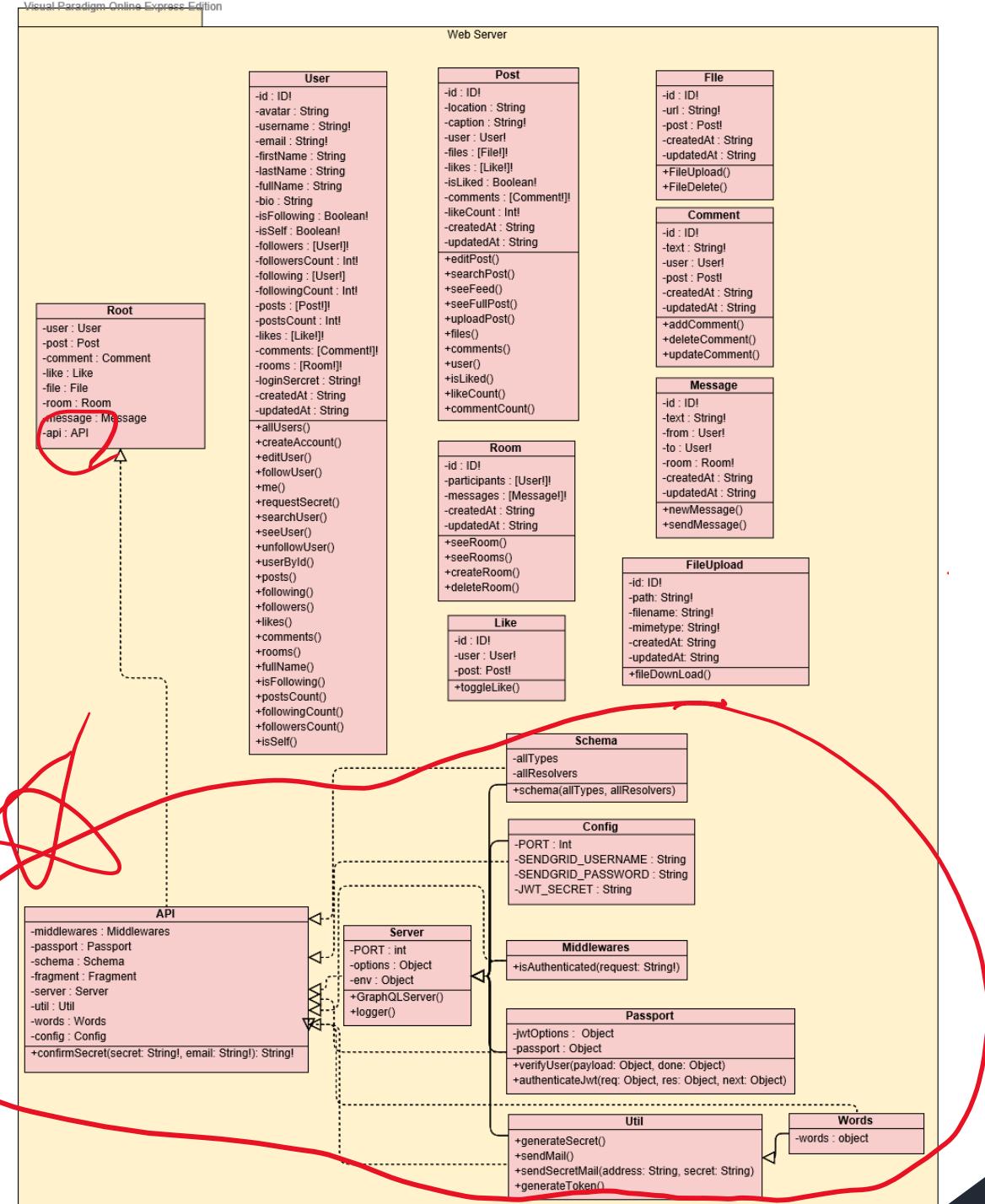
 newMessage.graphql

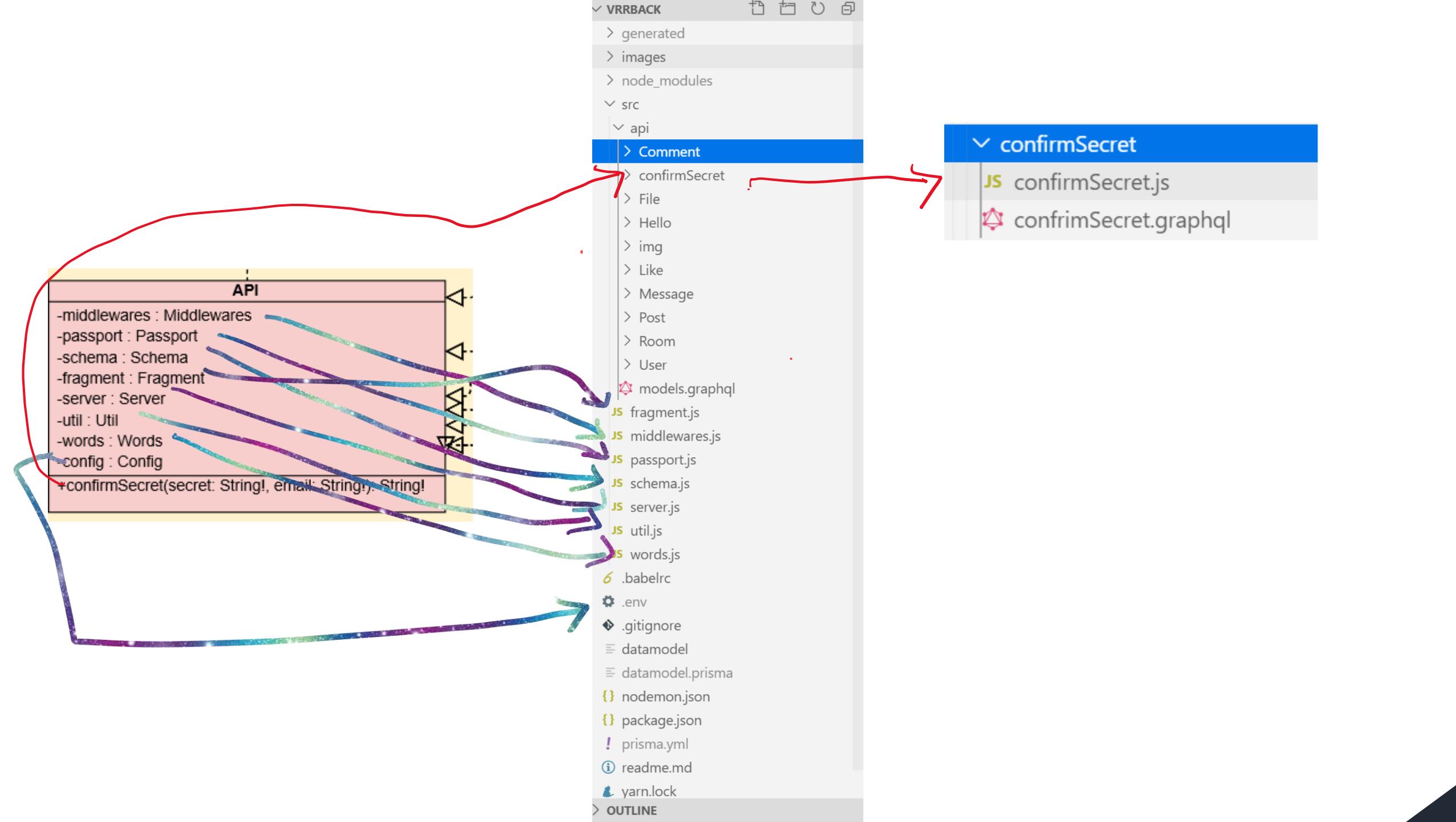
```
type Subscription {  
  newMessage(roomId: String!): Message  
}
```

 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



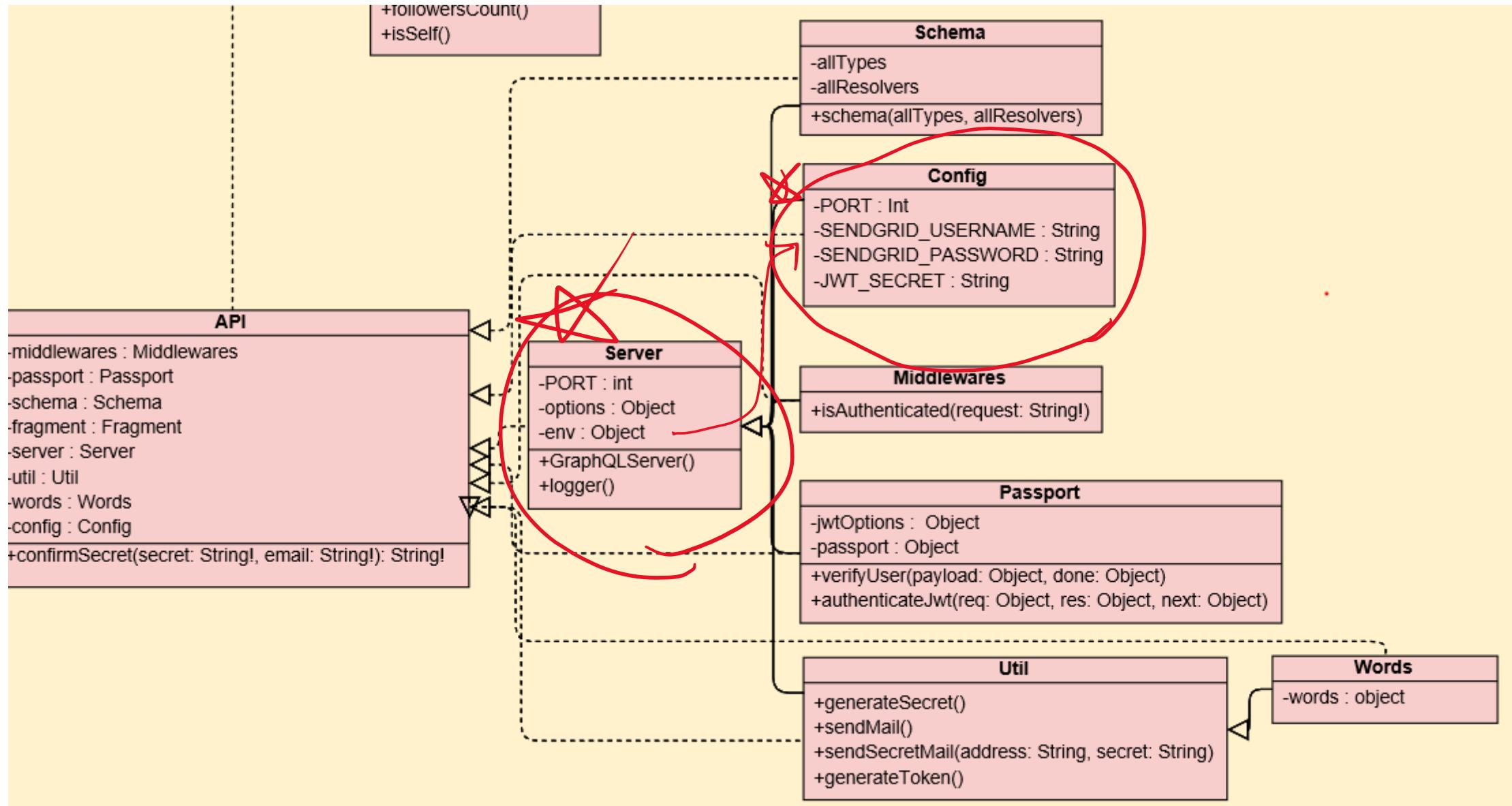


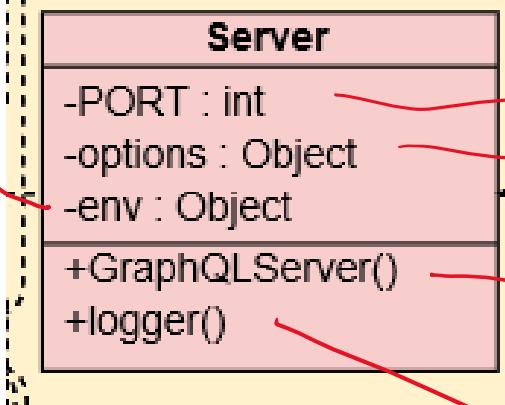
confirmSecret.graphql

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

confirmSecret.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")  
      }  
    }  
  }  
}
```





```
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'

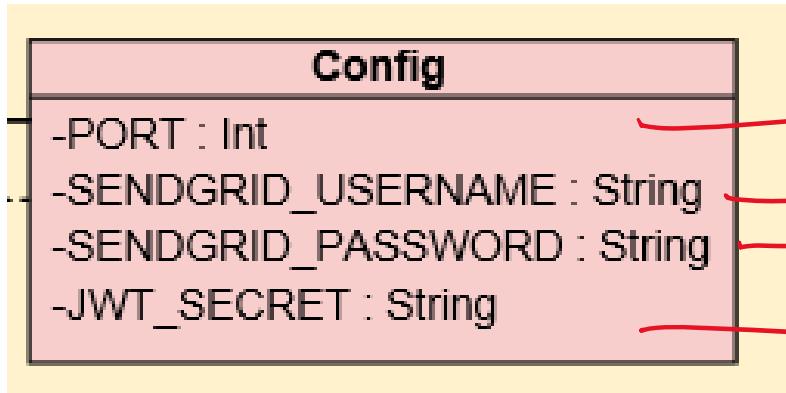
const PORT = process.env.PORT || 4000
const options = {
  port: PORT,
  endpoint: '/graphql',
  playground: '/playground',
  subscriptions: '/subscriptions',
  uploads: {
    maxFileSize: 10000000, // 10 MB
    maxFiles: 10,
  }
}

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

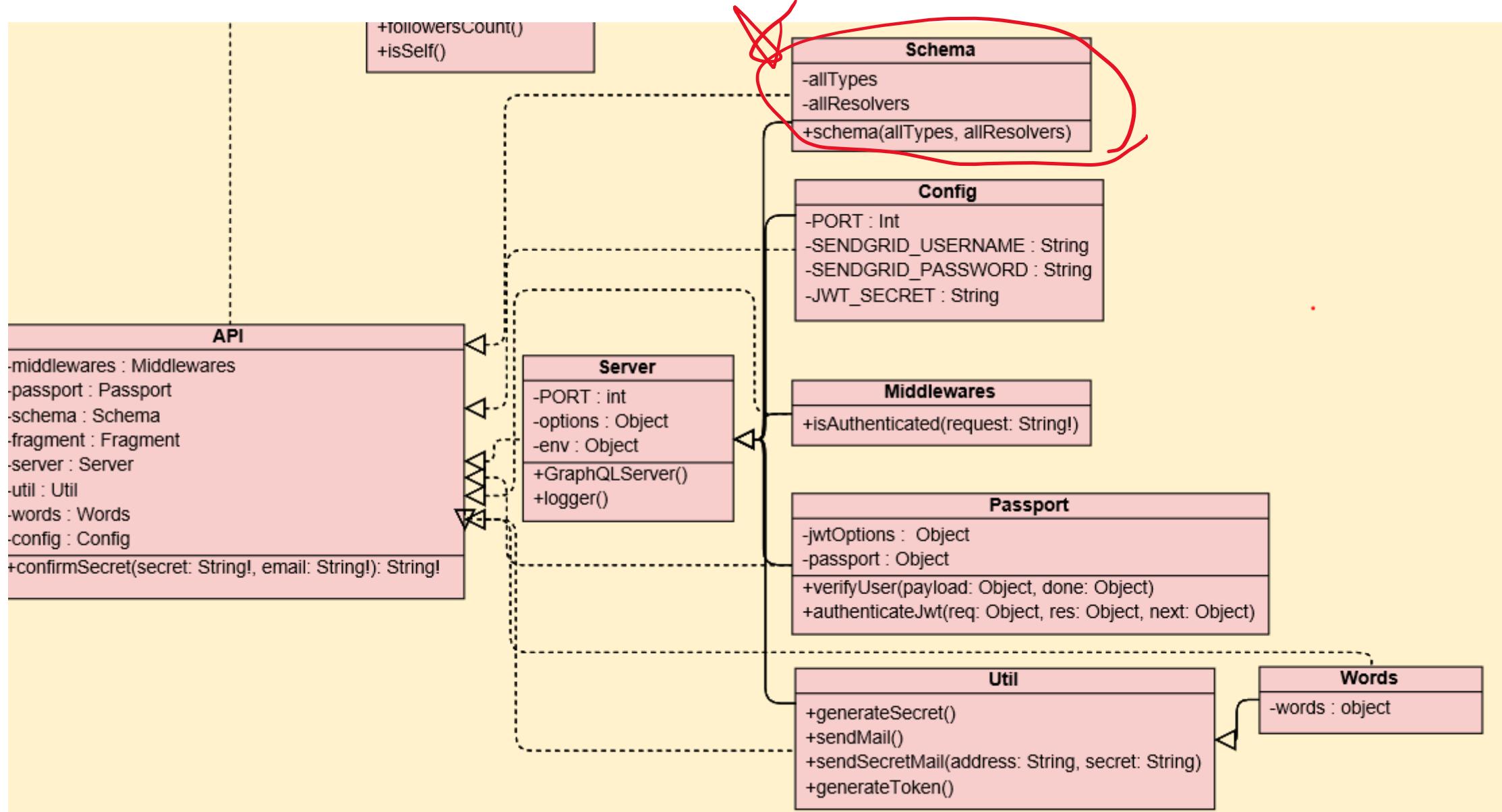
server.express.use(logger("dev"))
server.express.use(authenticateJwt)

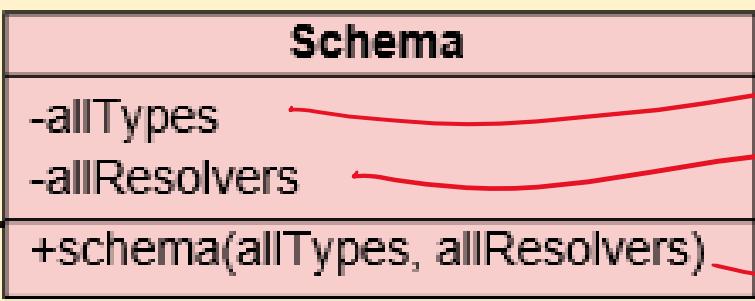
server.express.get('/hello', function(req, res) {
  res.send('hello world');
});

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



PORT = 4000
SENDGRID_USERNAME = hochan
SENDGRID_PASSWORD = ~~dkudsmf1e1!~~
JWT_SECRET = ~~Ab1Amc8bu1NLxsh...VQSFPyKoybeJ~~



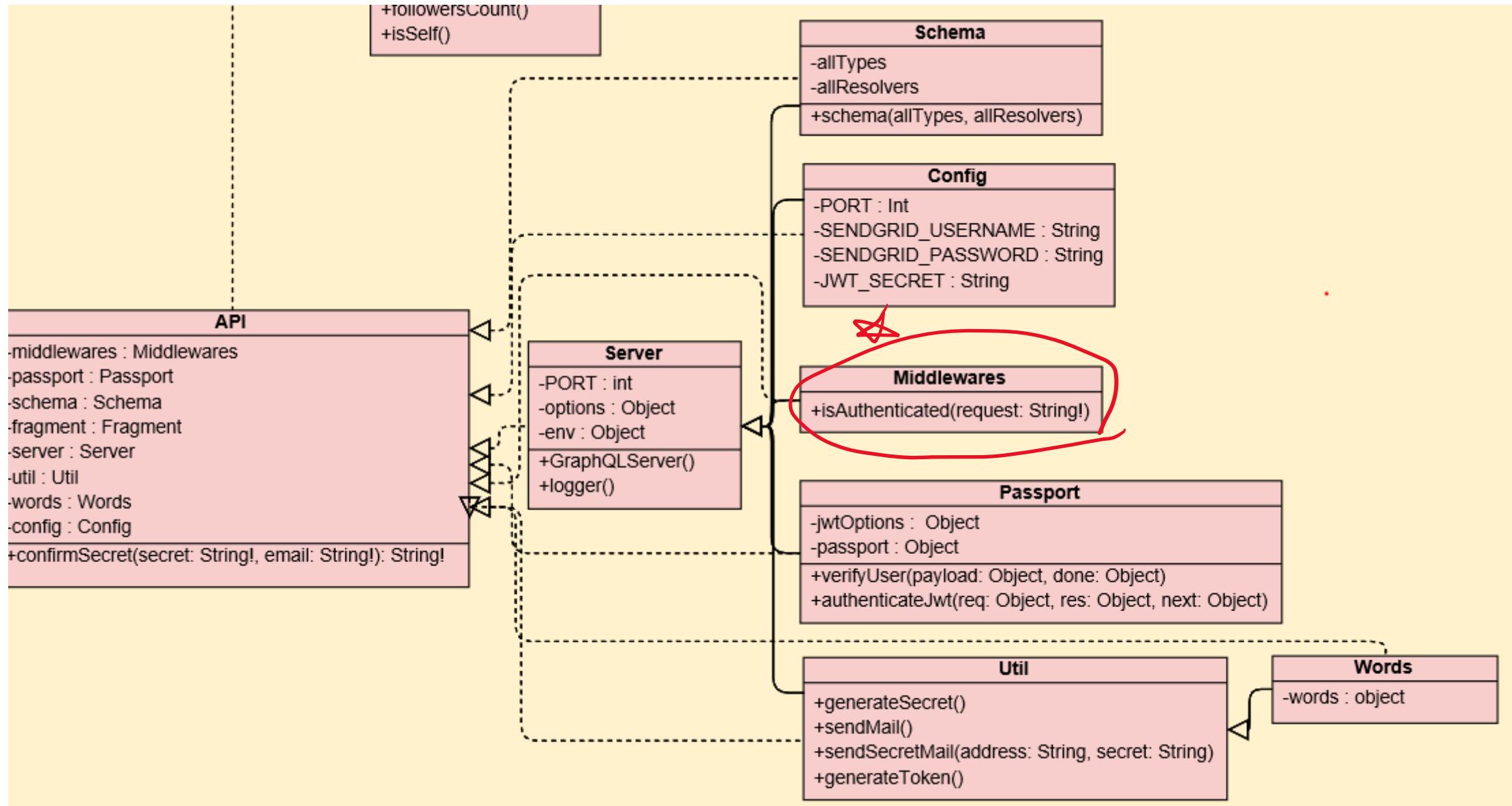


```
import path from 'path'
import { makeExecutableSchema } from 'graphql-tools'
import { fileLoader, mergeResolvers, mergeTypes } from 'merge-graphql-schemas'

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
```



Middlewares

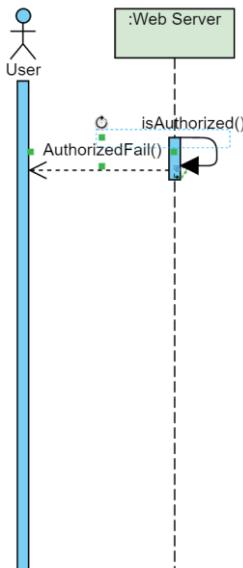
```
+isAuthenticated(request: String!)
```

JS middlewares.js

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

JS server.js

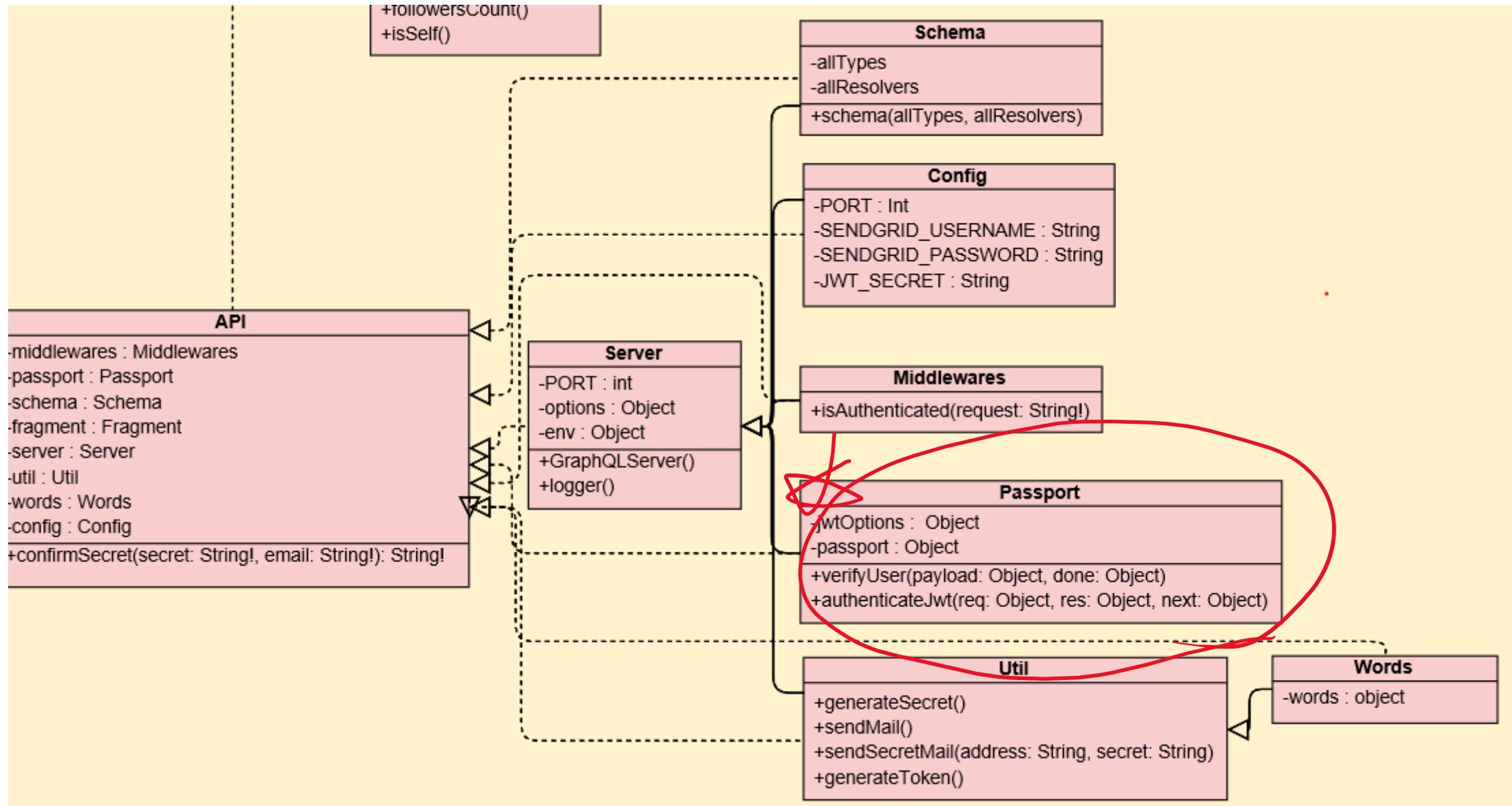
```
const server = new GraphQLServer({ schema,
  context: ({request}) => ({request, isAuthenticated})
})
```

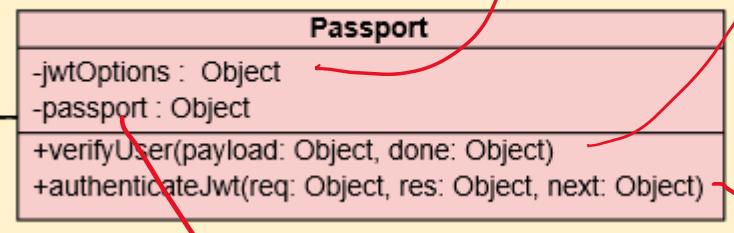


JS seeRoom.js

```
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")
      }
    }
  }
}
```





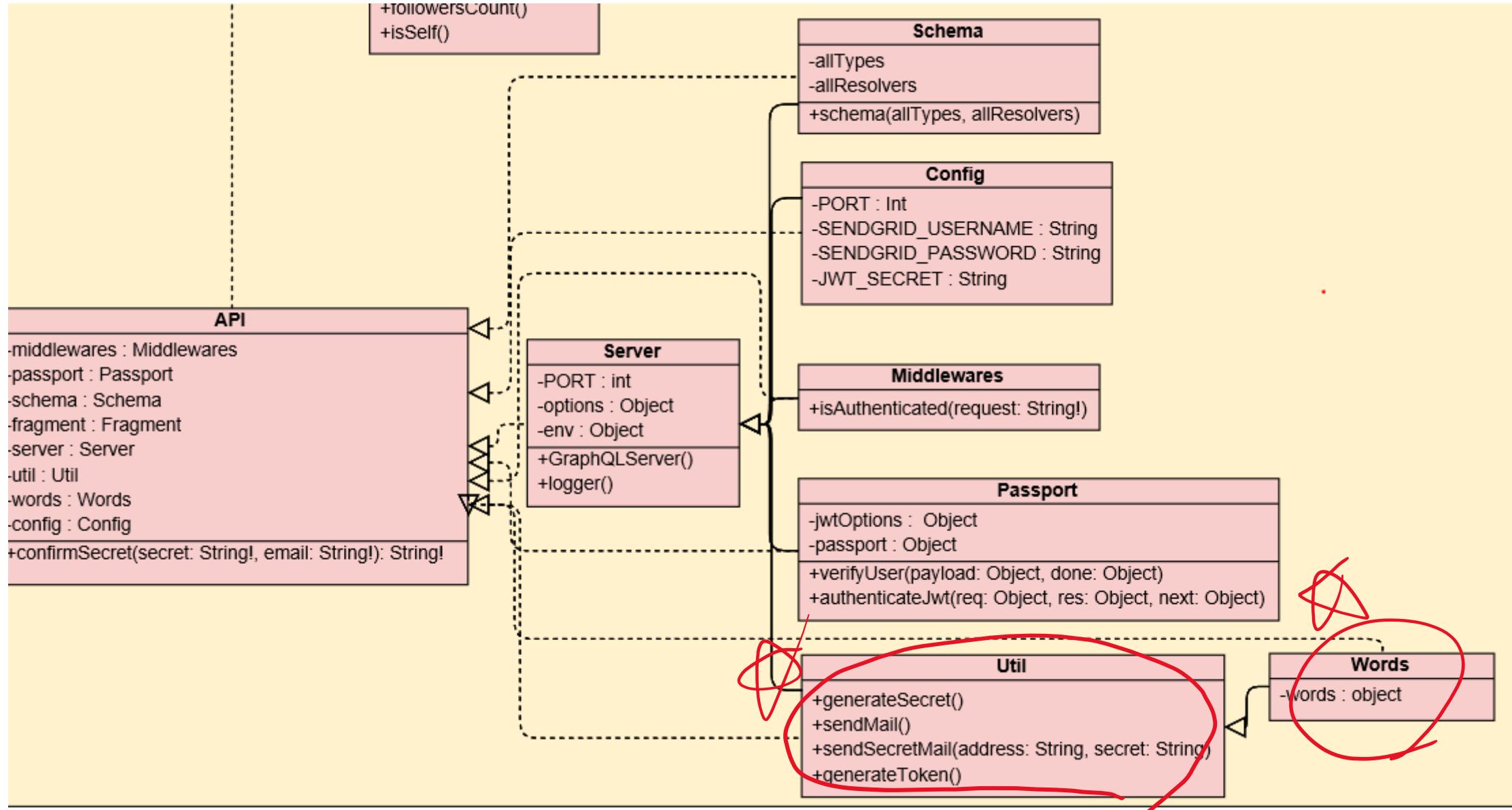
```
import passport from "passport"
import { Strategy, ExtractJwt } from "passport-jwt"
import { prisma } from '../generated/prisma-client'

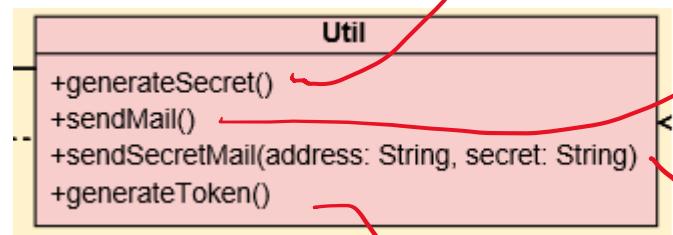
const jwtOptions = {
  jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
  secretOrKey: process.env.JWT_SECRET
}

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()
```





```
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]}`

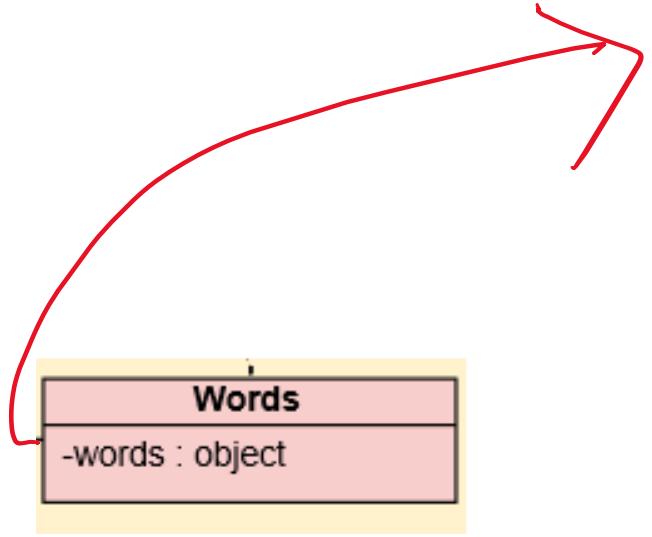
}

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)
```



문서를 보면 찾을 수 있다

6 .babelrc

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

프로젝트 폴더에 있는 파일

{ nodemon.json

```
{  
  "ext": "js graphql"  
}
```

prisma ORM 사용 가능

▼ generated

```
  ▼ prisma-client  
    TS index.d.ts  
    JS index.js  
    JS prisma-schema.js
```

작동하기

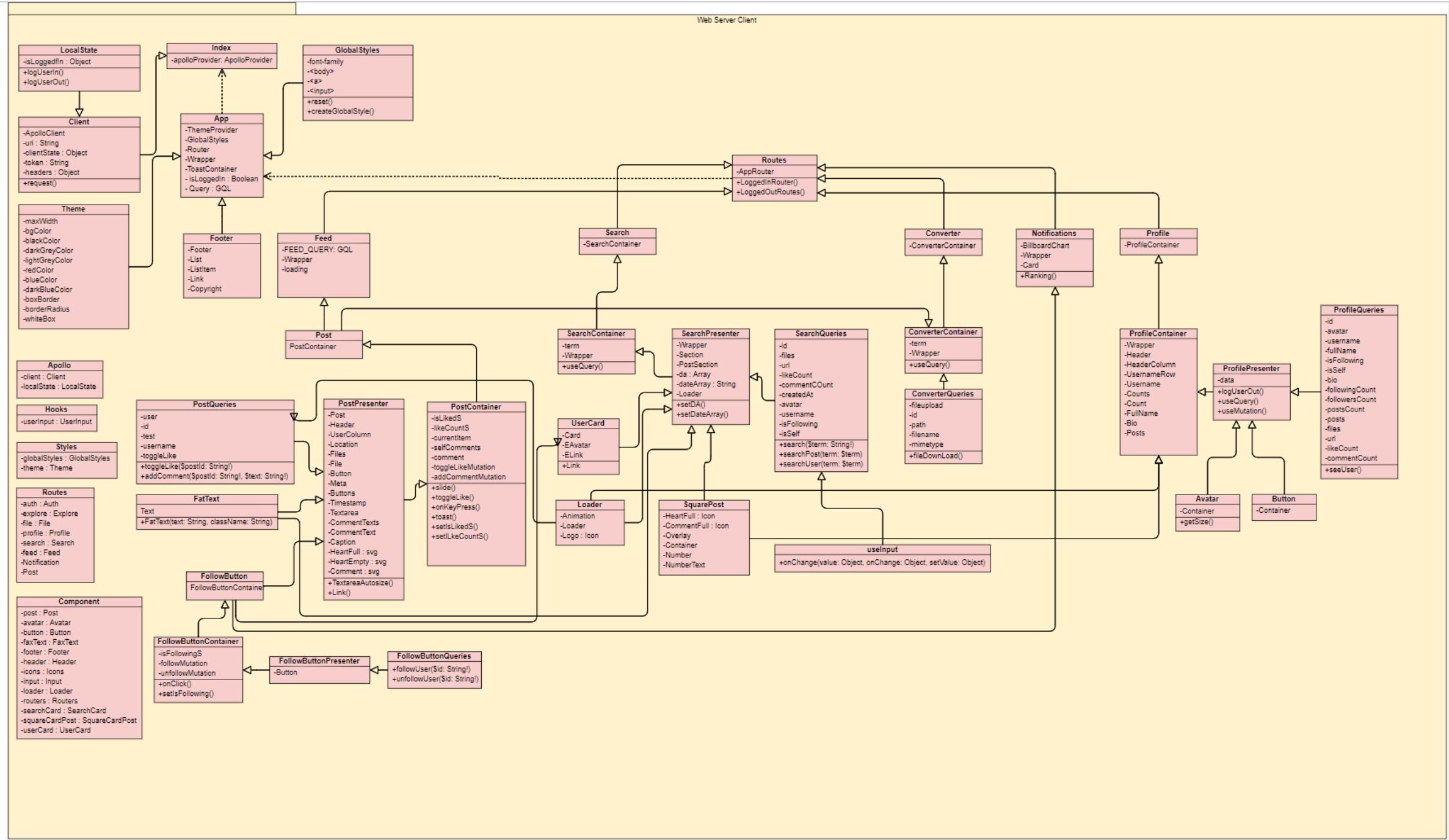
! prisma.yml

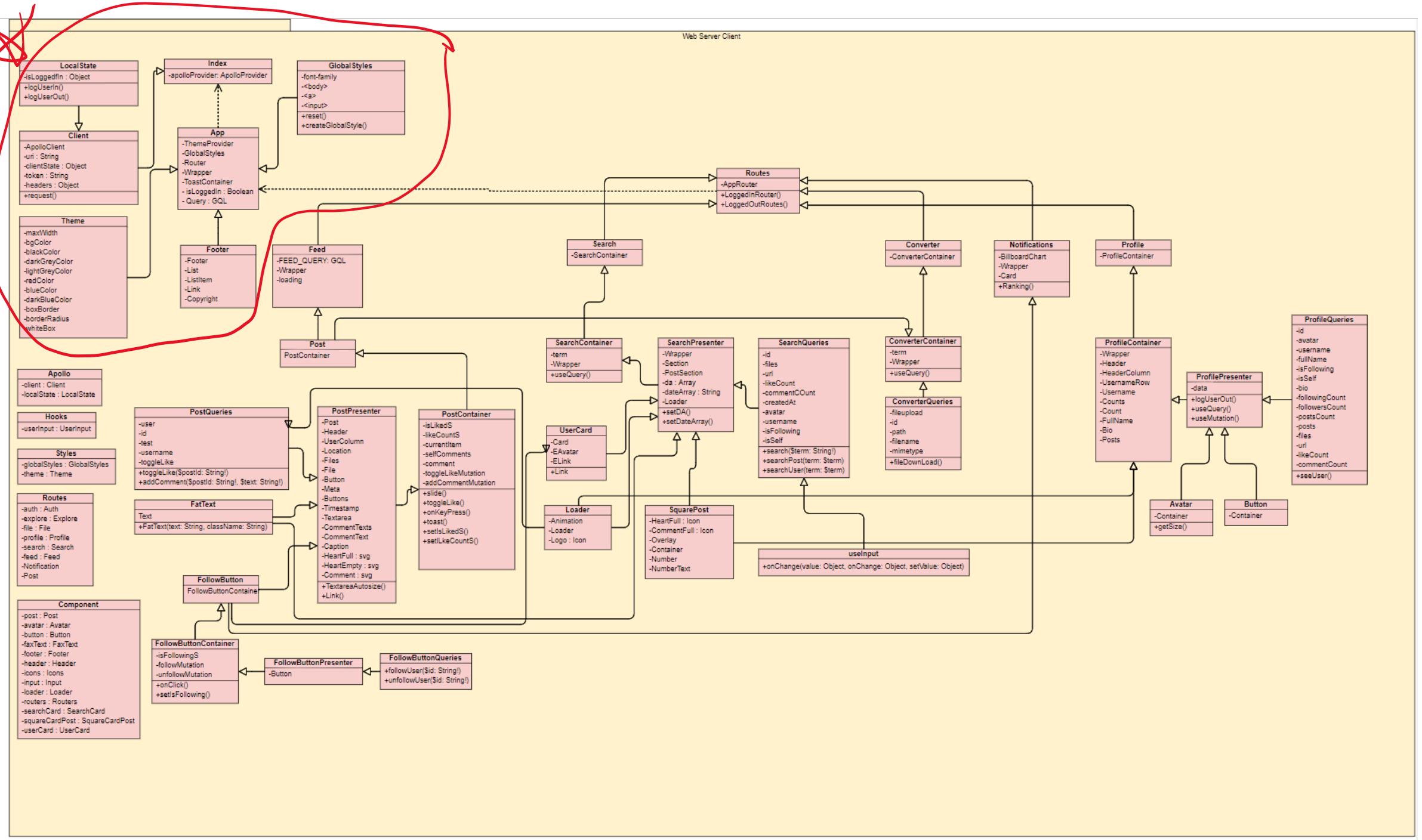
```
endpoint: https://us1.prismadb.net/dev  
datamodel: datamodel.prisma
```

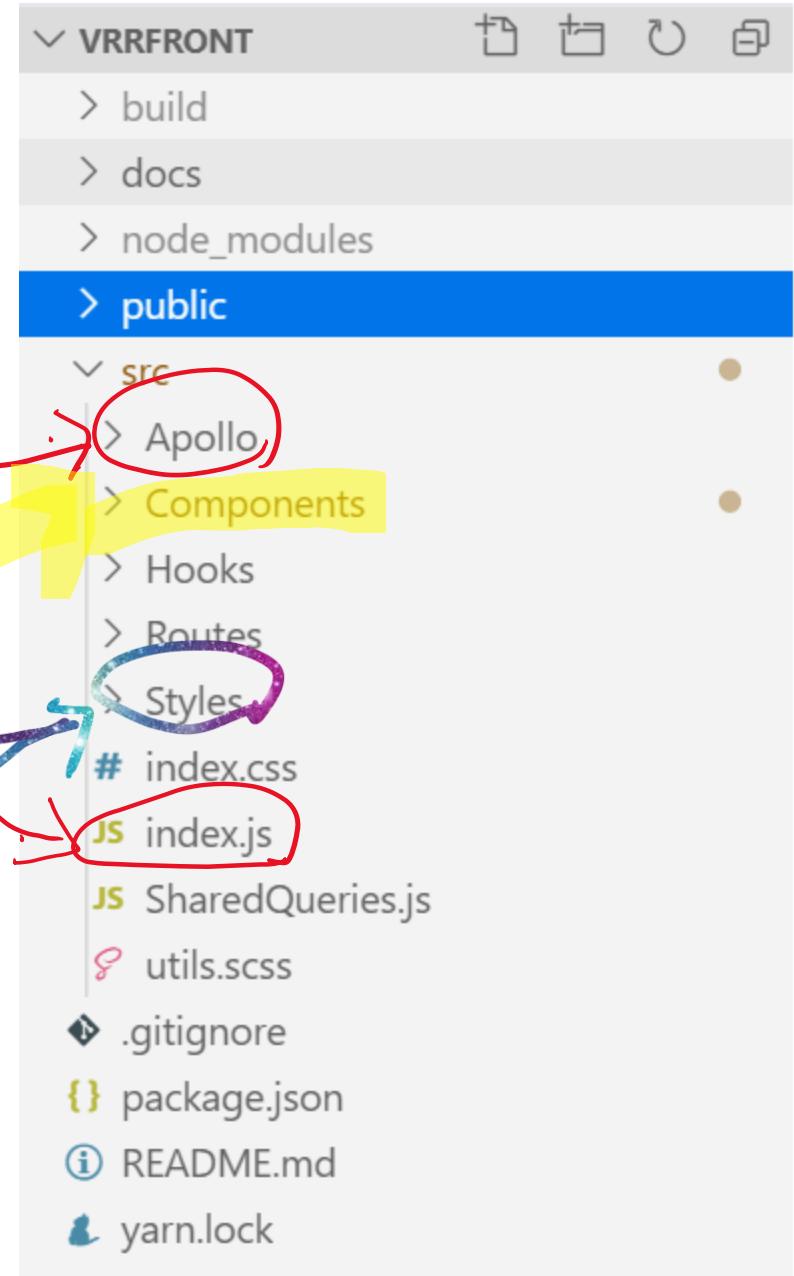
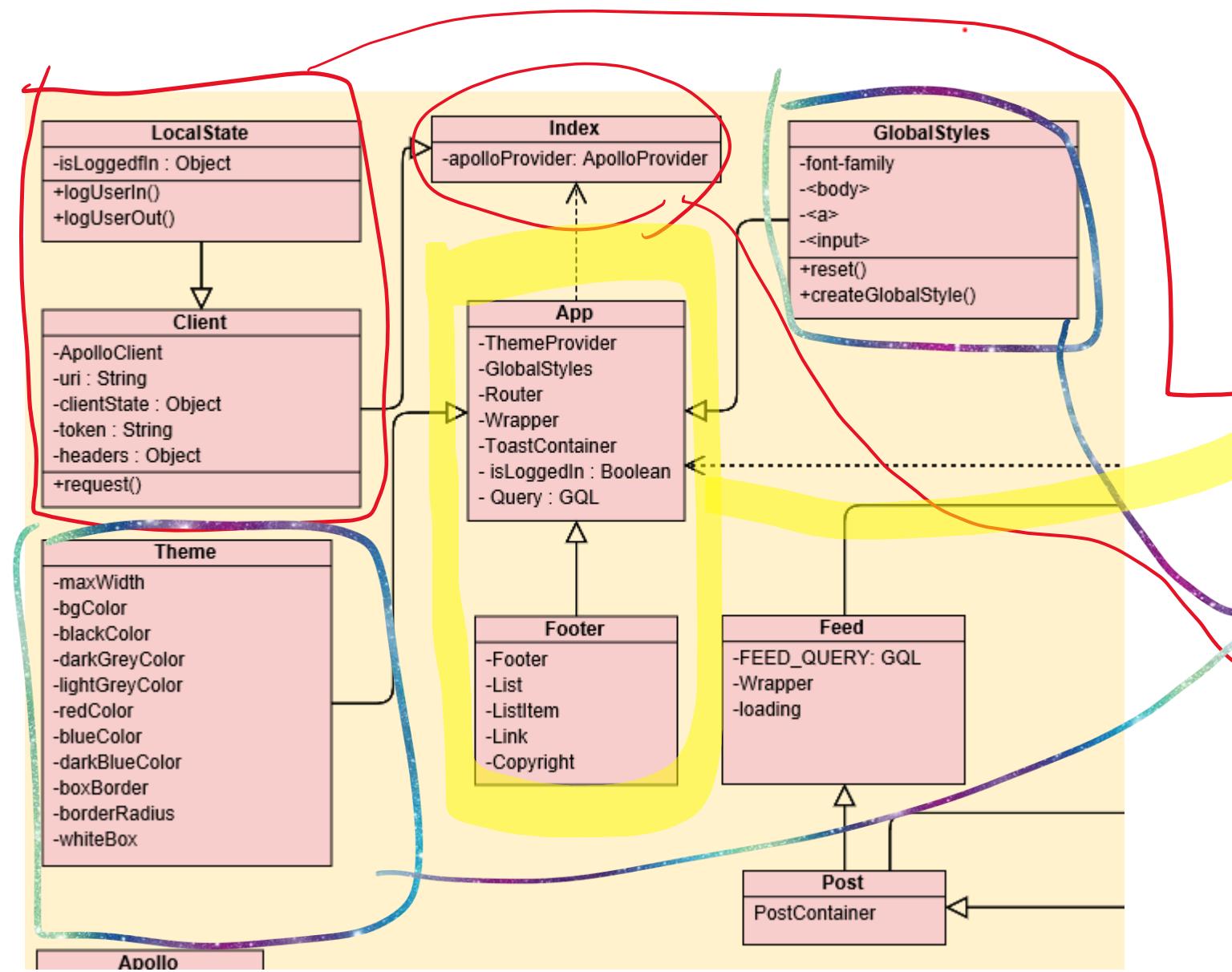
generate:

```
  - generator: javascript-client  
    output: ./generated/prisma-client/
```

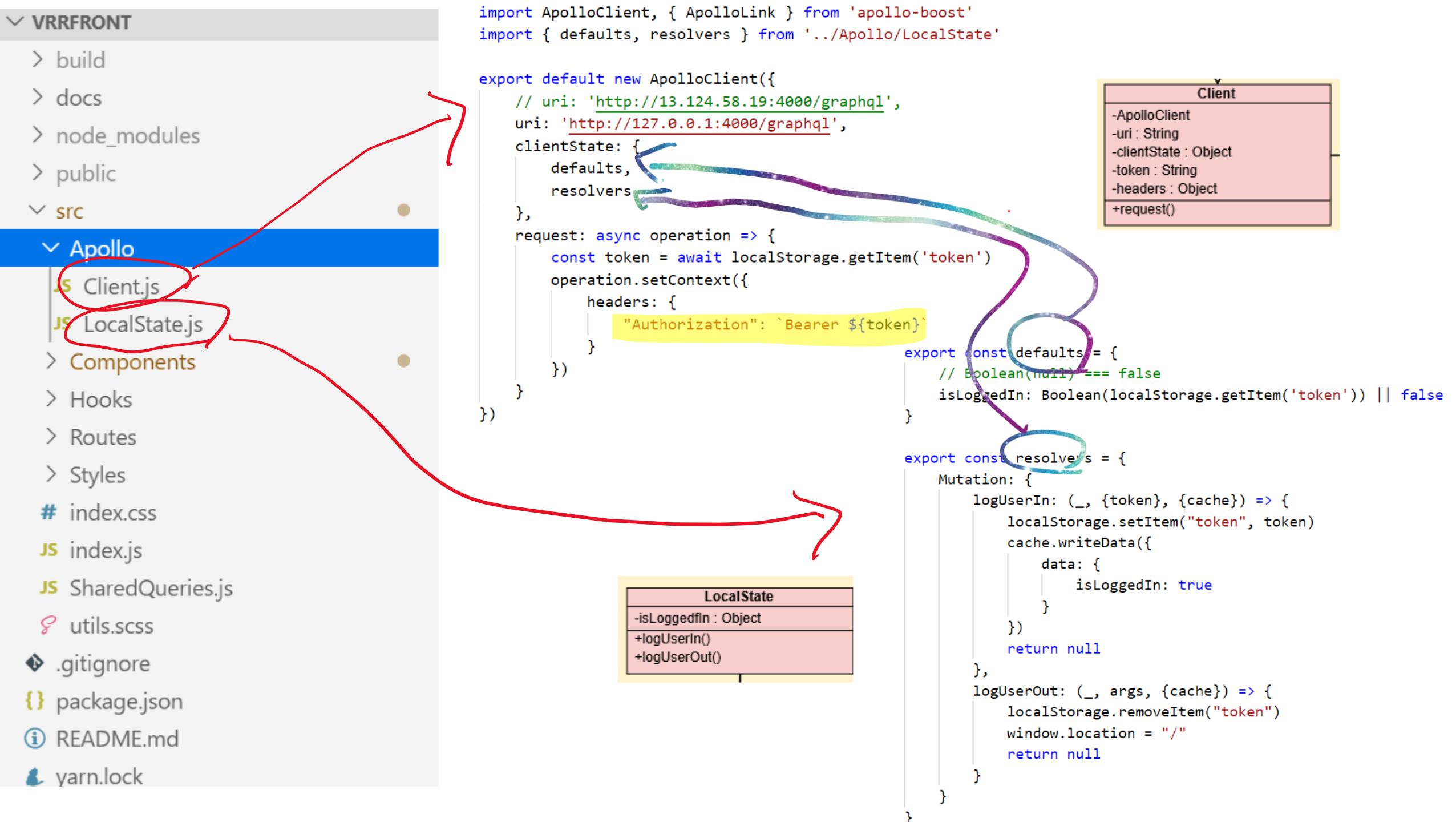
Web Client











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

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local Storage and Session Storage for the URL <http://localhost:3000>. In the Local Storage table, there is one entry for the key 'token' with the value 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNrMWppMWVsMDdvYTkwYjQwazNkazJhaGYiLCJpYXQiOjE1NzA2NzExNjF9.zwmg8ZKnh4G7xjE-DDg13Vkc35gCwXwVksJcDvoQe-Y'. A hand-drawn purple oval highlights the 'token' variable in the code above and the corresponding row in the storage table.

Key	Value
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNrMWppMWVsMDdvYTkwYjQwazNkazJhaGYiLCJpYXQiOjE1NzA2NzExNjF9.zwmg8ZKnh4G7xjE-DDg13Vkc35gCwXwVksJcDvoQe-Y

✓ VRRFRONT

- > build
- > docs
- > node_modules
- > public
- ✓ src
 - > Apollo
 - > Components
 - > Hooks
 - > Routes
 - ✓ Styles
 - JS GlobalStyles.js
 - JS Theme.js
 - # index.css
 - JS index.js
 - JS SharedQueries.js
 - ♀ utils.scss
 - ❖ .gitignore
 - { package.json
 - ⓘ README.md
 - 牢固树立

```
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;  
  }
```

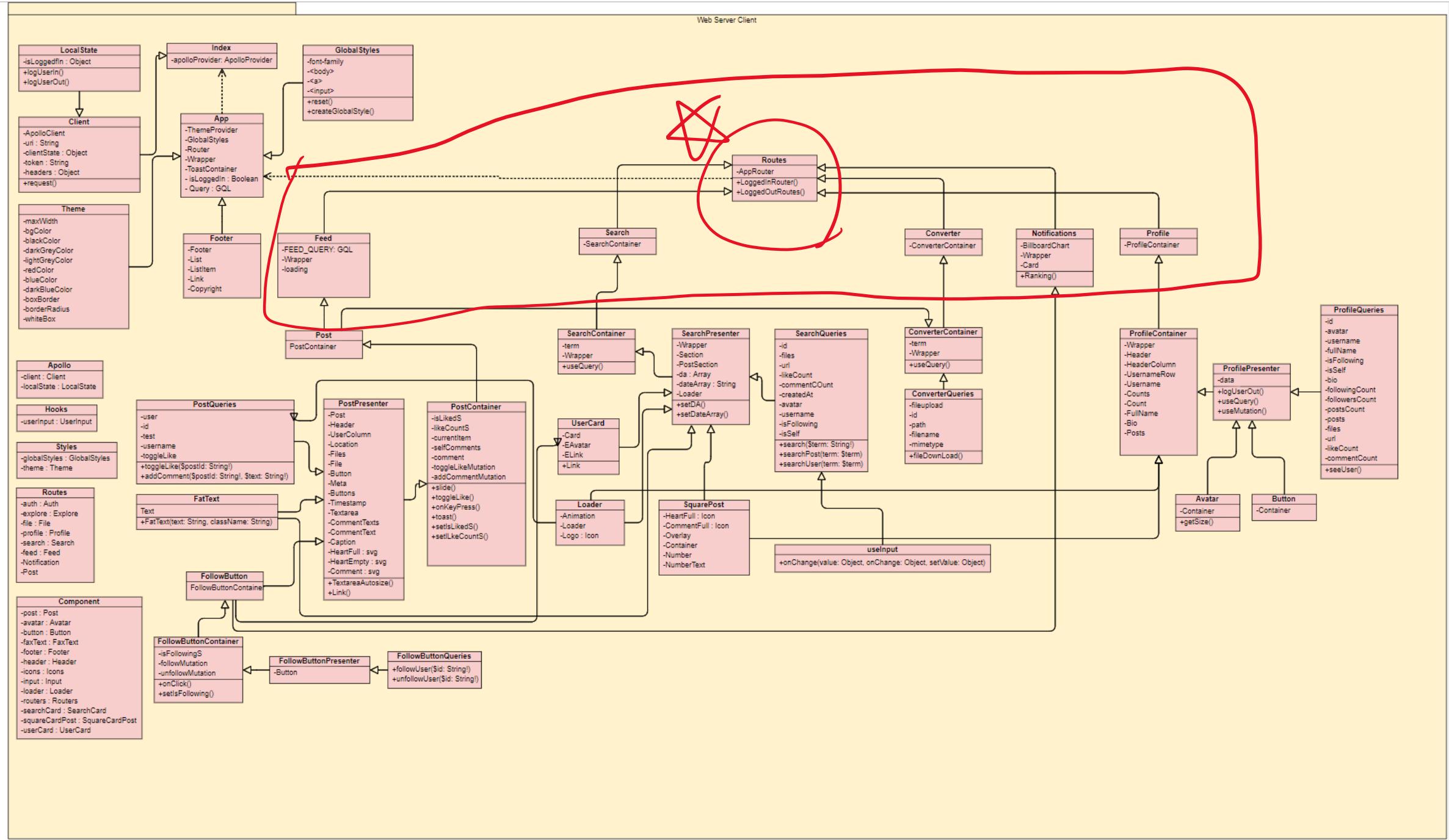
GlobalStyles
-font-family
-<body>
-<a>
-<input>
+reset()
+createGlobalStyle()

```
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

Footer
-Footer
-List
-Listitem
-Link
-Copyright

```
8     text-transform: uppercase;
9     font-weight: 600;
10    font-size: 12px;
11    margin: 50px 0px;
12
13
14  const List = styled.ul`  
15    display: flex;  
16
17
18  const ListItem = styled.li`  
19    &:not(:last-child) {  
20      margin-right: 16px;  
21    }  
22
23
24  const Link = styled.a`  
25    color: ${props => props.theme.darkBlueColor}  
26
27
28  const Copyright = styled.span`  
29    color: ${props => props.theme.darkGreyColor}  
30
31
32  export default () => (  
33    <Footer>  
34      <List>  
35        <ListItem><Link href="#">정보</Link></ListItem>  
36        <ListItem><Link href="#">개인정보 처리방침</Link></ListItem>  
37        <ListItem><Link href="#">약관</Link></ListItem>  
38        <ListItem><Link href="#">디렉터리</Link></ListItem>  
39        <ListItem><Link href="#">프로필</Link></ListItem>  
40        <ListItem><Link href="#">해시 태그</Link></ListItem>  
41        <ListItem><Link href="#">언어</Link></ListItem>  
42      </List>  
43      <Copyright>  
44        VRR {new Date().getFullYear()} &copy;  
45      </Copyright>  
46    </Footer>
```



VRFRONT

```
> build
> docs
> node_modules
> public
< src
  > Apollo
  < Components
    > Post
    JS App.js
    JS Avatar.js
    JS Button.js
    JS FatText.js
    JS Footer.js
    JS Header.js
    JS Icons.js
    JS Input.js
    JS Loader.js
    JS Routes.js
    JS SearchCard.js
    JS SquarePost.js
    JS UserCard.js
  > Hooks
  > Routes
  > Styles
  # index.css
  JS index.js
  JS SharedQueries.js
  ❀ utils.scss
  .gitignore
  { package.json
  ⓘ README.md
  yarn.lock
```

The diagram illustrates the component flow from `Routes.js` to various UI components. Red arrows point from the code snippets to the corresponding components in the sidebar.

- LoggedInRoutes** points to the `Feed` component.
- LoggedOutRoutes** points to the `Profile` component.
- AppRouter** points to the `Converter`, `Search`, and `Notifications` components.

```

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>
    <Route path="/search" component={Search}></Route>
    <Route path="/notifications" component={Notification}></Route>
    <Route path="/:username" component={Profile}></Route>
    <Redirect from="/" to="/" />
  </Switch>
)

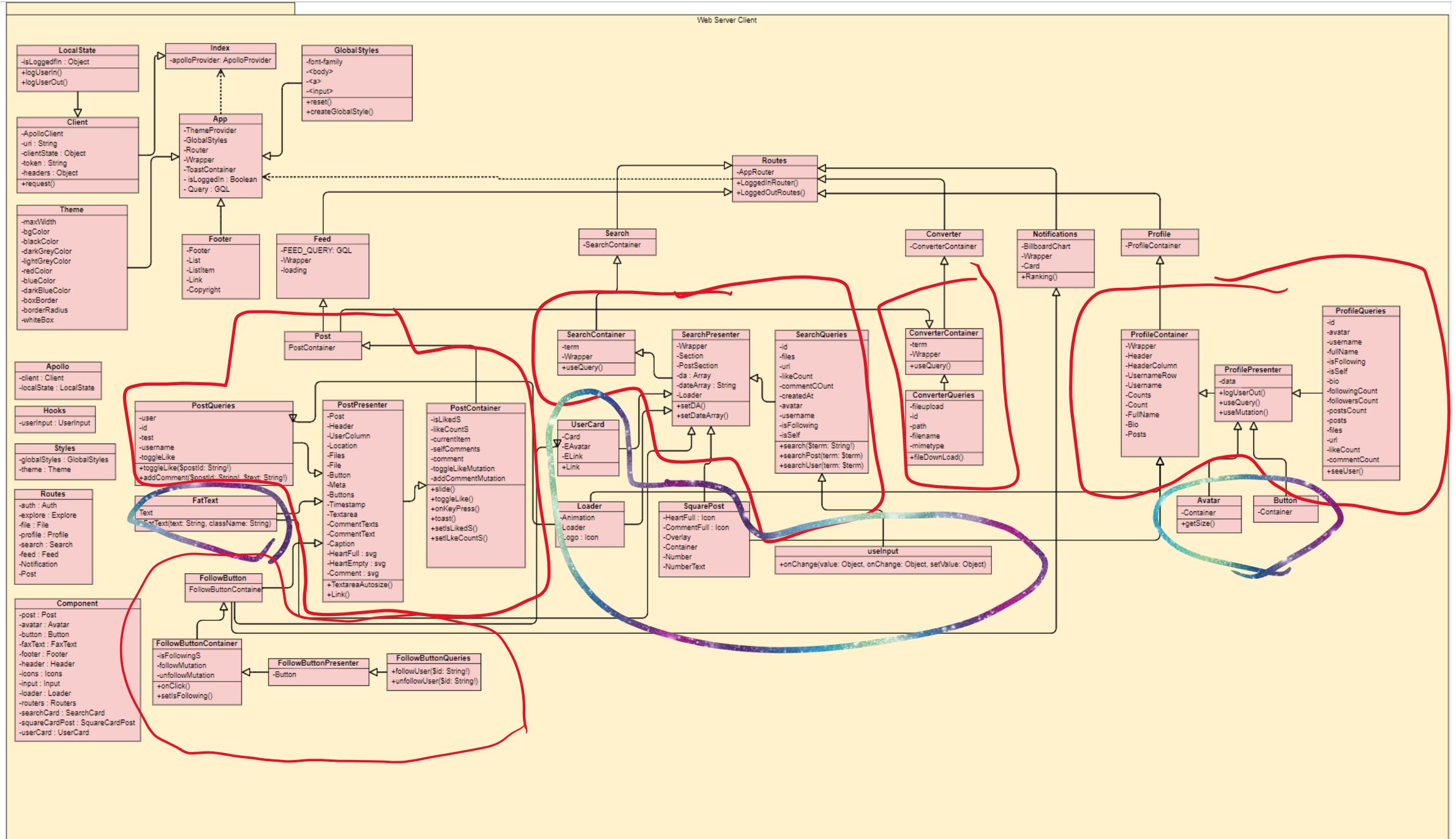
const LoggedOutRoutes = () => <Switch><Route exact path="/" component={Auth}></Route></Switch>

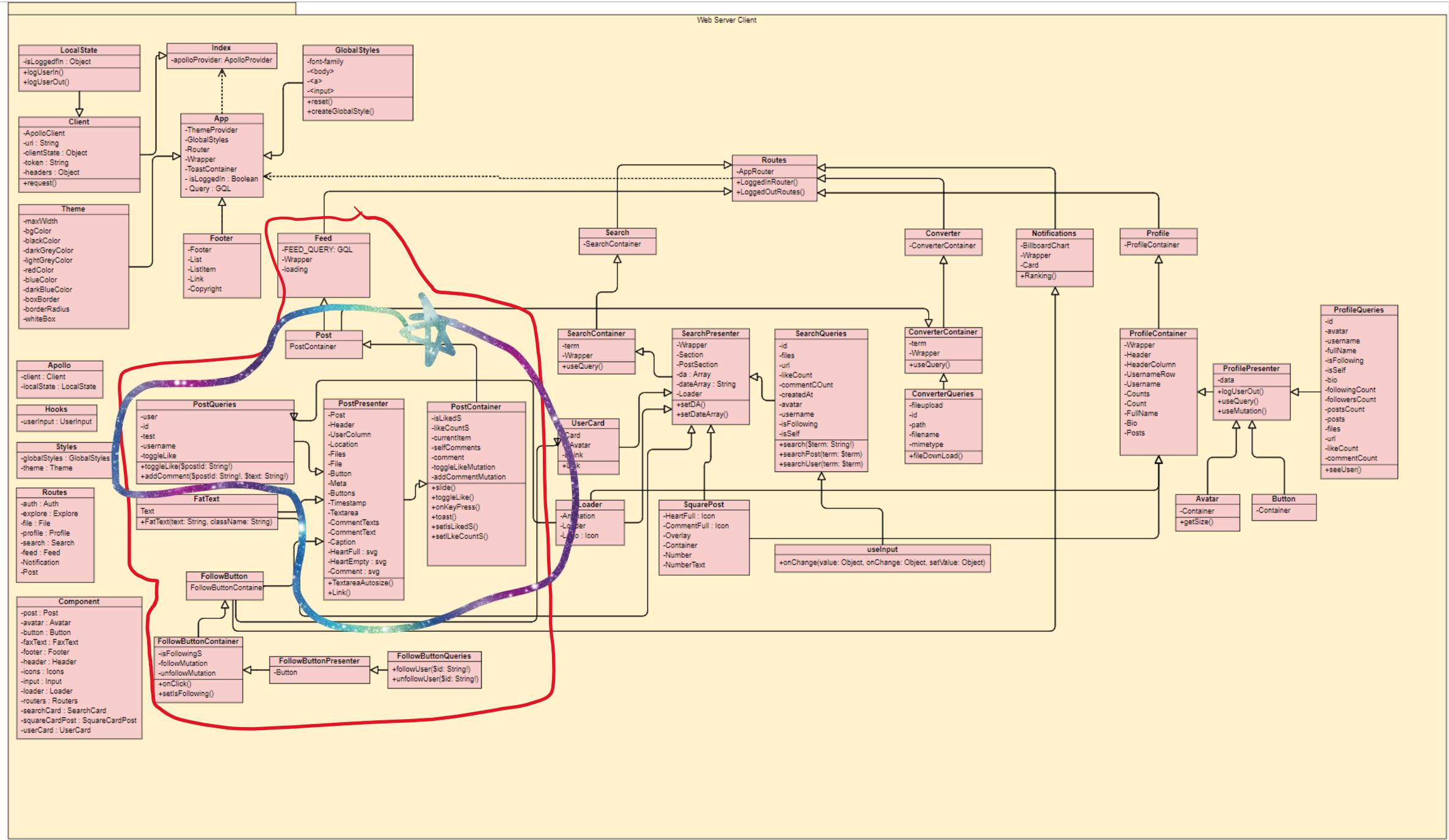
const AppRouter = ({ isLoggedIn }) => (
  <Switch>{isLoggedIn ? <LoggedInRoutes/> : <LoggedOutRoutes/>}</Switch>
)

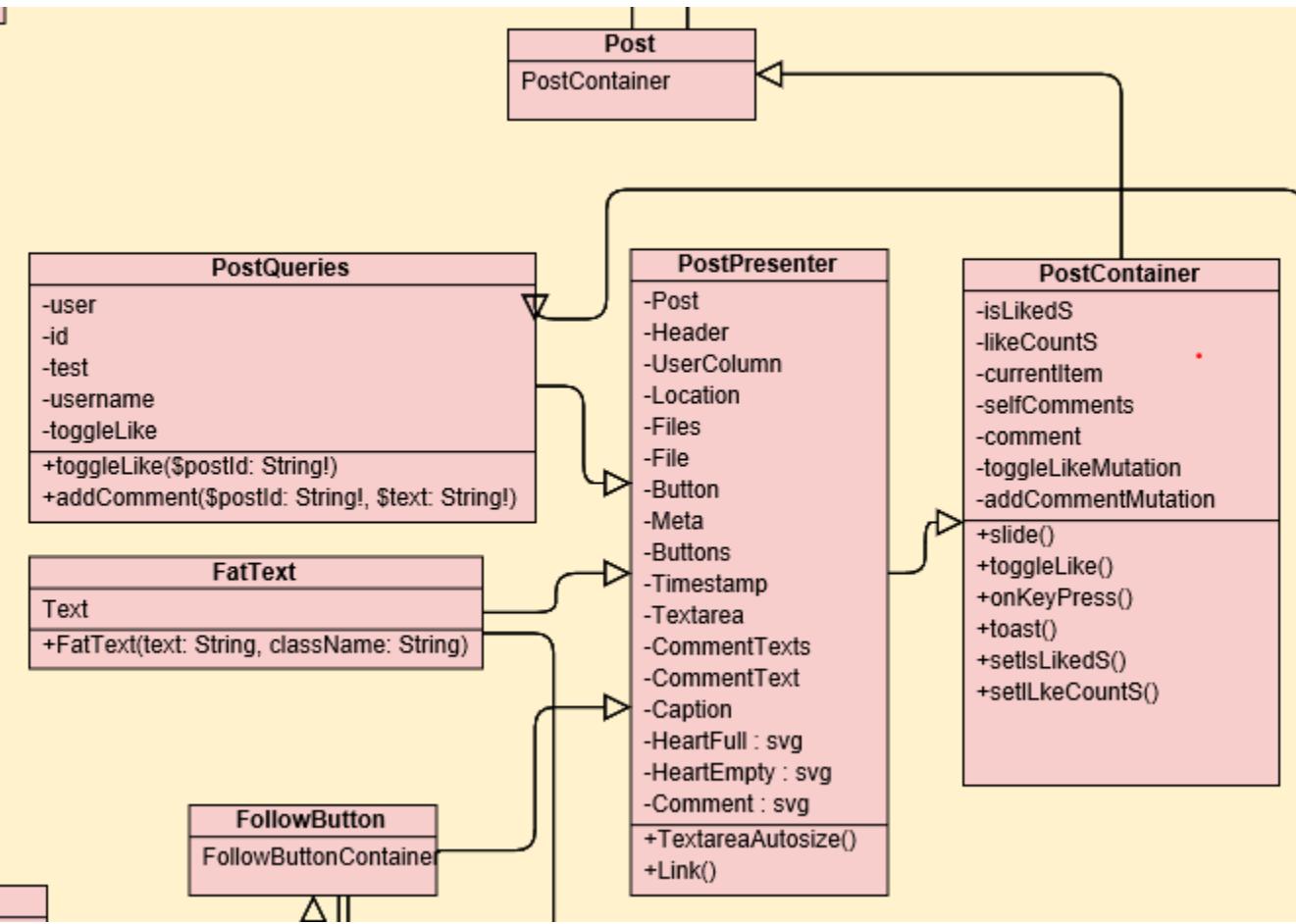
AppRouter.propTypes = {
  isLoggedIn: PropTypes.bool.isRequired
}

export default AppRouter

```







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

```
10  const [isLikedS, setIsLiked] = useState(isLiked)
11  const [likeCountS, setLikeCount] = useState(likeCount)
12  const [currentItem, setCurrentItem] = useState(0)
13  const [selfComments, setSelfComments] = useState([])
14  const comment = useInput('')
15  const [toggleLikeMutation] = useMutation(TOGGLE_LIKE, {
16    variables: {postId: id}
17  })
18  const [addCommentMutation] = useMutation(ADD_COMMENT, {
19    variables: {postId: id, text: comment.value}
20  })

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>
        ))}
      </CommentTexts>
    )}
  </Meta>
```

감사합니다.