# PIGEON

# 알림장 WEB SERVICE

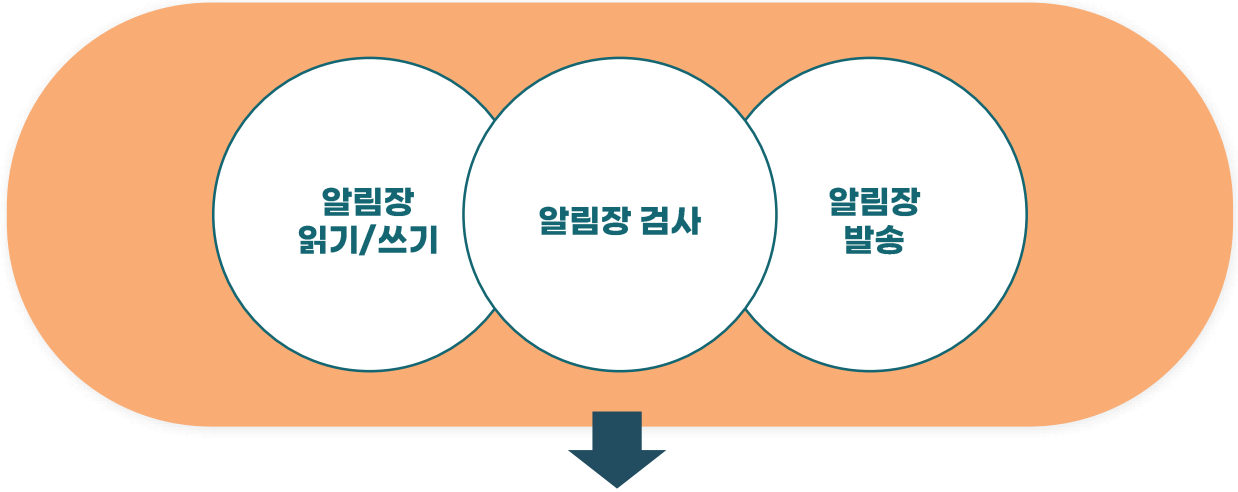
# 목차 table of contents



# 1

# 개요

# :D 서비스 개요




# PISEON: 알림장

오프라인보다 온라인이 활발해진 세상에서,
온라인으로 보다 편리하게 알림장을 작성 / 관리 할 수 있도록 해주는
**알림장 웹 서비스**입니다.

# :D 개발 개요

## 주요 기능

- 로그인 (소셜 간편로그인 + 단방향 암호화)
- 웹 푸시 기능(FCM + 서비스 워커)
- 게시물 CRUD 기능 (Amazon S3 버킷)
- 각 회원의 게시물 읽음/안 읽음 확인 기능
- 클래스 별 회원 관리 기능
- 게시물에 대한 댓-대 댓글 기능

## 개발 기간

1. 기획 및 설계 : 24.04.01 ~ 24.04.10
2. 기능 및 디자인 : 24.04.10 ~ 24.05.15
3. 디버깅 및 기능 정리 : 24.05.15 ~ 24.05.28

**= 소요 기간 : 약 2달**

## 개발 인원 및 역할

- 인원 : **개인 프로젝트 (1인)**
- 역할 : 기획 / 설계 / 기능 / 디자인

# 2

# 설계 및 아키텍쳐

# :D 시스템 구성 및 기술 스택

## FRONT-END



## BACK-END




## DATABASE

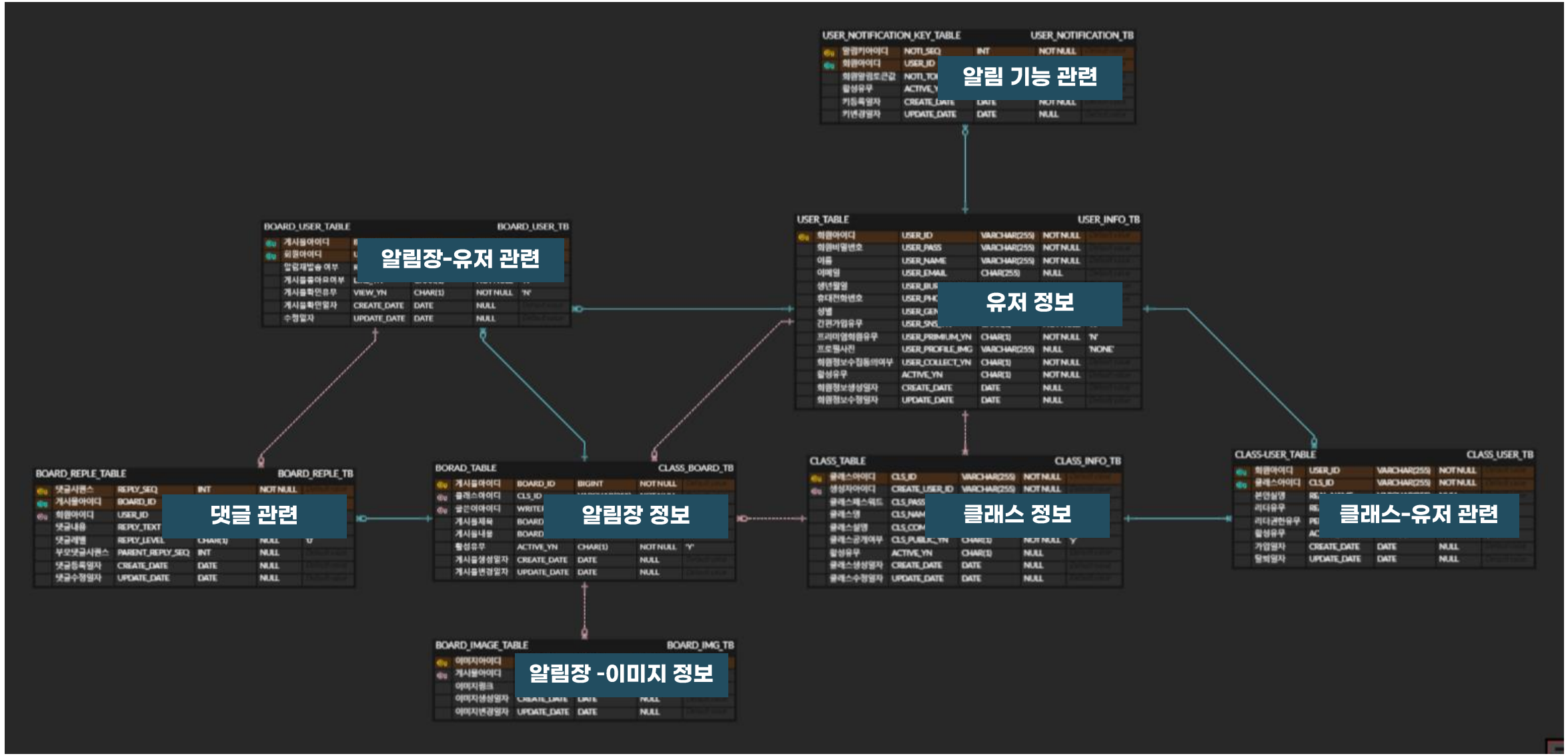

## TOOL

* Eclipse IDE 24. 03
* VS code
* MYSQL Workbench 8.0

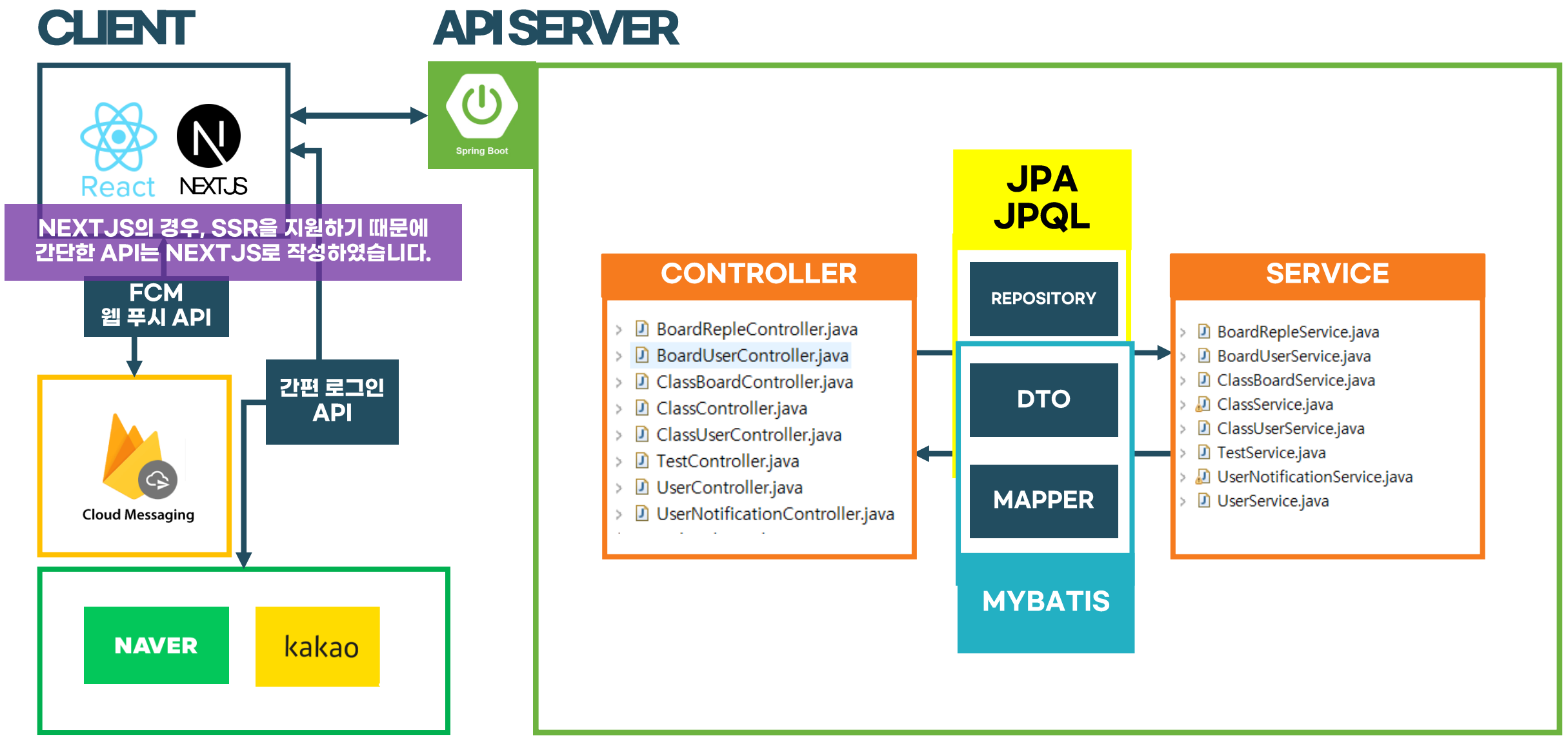
## OTHERS SKILL

- REACT QUILL
- REACT-HOOK-FORM
- NEXT-AUTH
- APP ROUTER
- AMAZON S3
- FRONT CLOUD
- SPRING-SECURITY
- FCM
- SERVICE WOKER
- AXIOS
- TAILWIND CSS
- LOMBOK
- NGROK
- LOCALTUNNEL

# :D ERD 모델링



# :D API 설계 : RESTful API



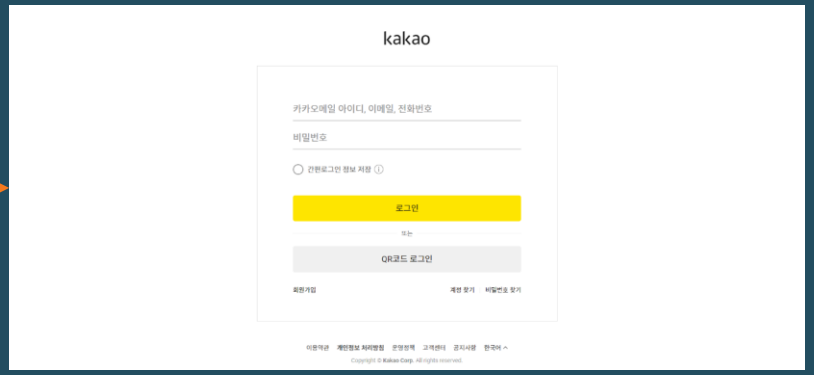
# 3

# 화면흐름도

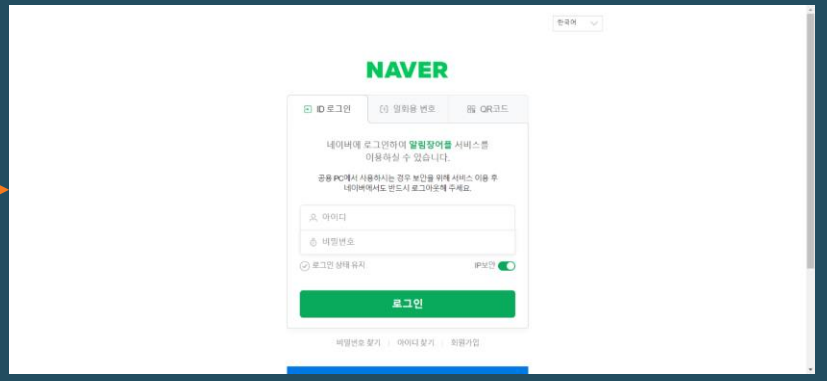
# :D 화면흐름도 - 로그인


회원가입

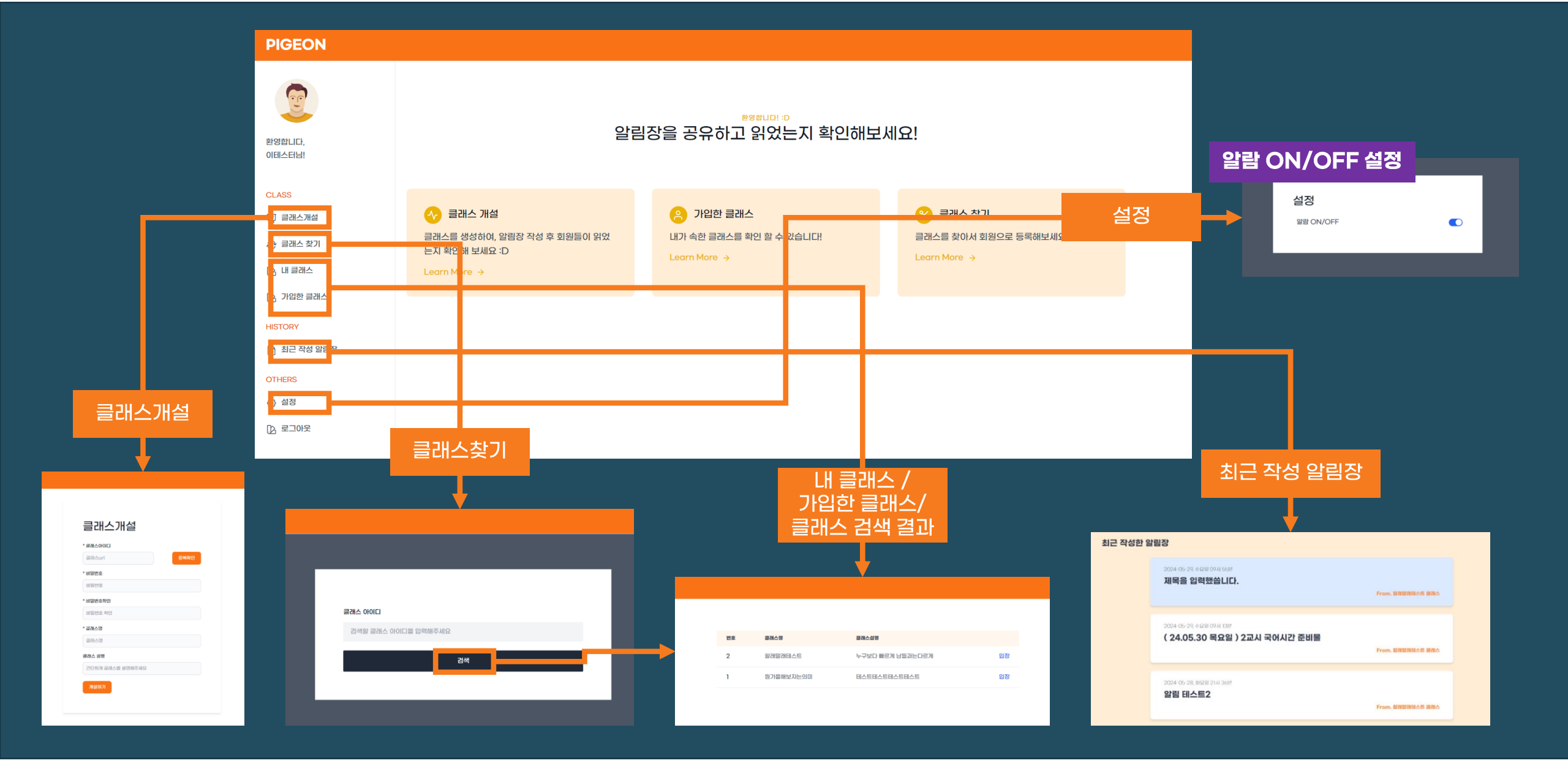

카카오로그인




네이버로그인



# :D 화면흐름도 - 메인


PIGEON
환영합니다,
이테스터님!
CLASS
클래스개설
클래스 찾기
내 클래스
가입한 클래스
HISTORY
최근 작성 알림장
OTHERS
설정
로그아웃
환영합니다! :D
알림장을 공유하고 읽었는지 확인해보세요!
클래스 개설
클래스를 생성하여, 알림장 작성 후 회원들이 읽었는지 확인해 보세요 :D
Learn More →
가입한 클래스
내가 속한 클래스를 확인 할 수 있습니다!
Learn More →
Learn More →
설정
알람 ON/OFF 설정
설정
알람 ON/OFF
클래스개설
클래스찾기
내 클래스 / 가입한 클래스/ 클래스 검색 결과
최근 작성 알림장
클래스개설
클래스 아이디
검색할 클래스 아이디를 입력해주세요
검색
최근 작성한 알림장
제목을 입력했습니다.
( 24.05.30 목요일 ) 2교시 국어시간 준비물
알림 테스트2

# :D 화면흐름도 - 클래스


처음 입장 할 경우
해당 클래스에서 사용할 이름
인식 가능한 실명을 입력해주세요
클래스 비밀번호
비밀번호
제출
클래스 삭제 / 탈퇴
클래스 리더일 경우 : 삭제
회원일 경우 : 탈퇴
클래스를 정말 삭제하시겠습니까 ?
예
아니요
클래스를 정말 탈퇴하시겠습니까 ?
예
아니요
알림장 작성
PIGEON
할래말래테스트
클래스 아이디 : testclass2
클래스삭제
알림장작성
클래스회원
환영합니다,
이테스터님!
CLASS
클래스개설
클래스 찾기
내 클래스
가입한 클래스
HISTORY
최근 작성 알림장
OTHERS
로그아웃
설정
2024-05-28, 화요일 21시 36분
확인완료
알림장 by 설명입니다
2024-05-29, 수요일 09시 13분
( 24.05.30 목요일 ) 2교시 국어시간 준비물
미확인
알림장 by 설명입니다
2024-05-26, 일요일 17시 30분
디자인 테스트3
확인완료
알림장 by 설명입니다
1 2 3 4 5
내가 읽었는지 확인가능
알림장 읽기
알림장작성
제목
제목을 입력하세요
저장
로그아웃
등록
알림장이 도착했어요! alive-easy-weevil.ngr... 오후 4:36
메시지를 눌러 알림장을 확인해주세요
PIGEON
localhost:3000 • 지금
알림장이 도착했어요!
메시지를 눌러 알림장을 확인해주세요
PIGEON
알림장 등록 시,
현재 가입된 클래스 회원
모두에게 알림
PIGEON
( 24.05.30 목요일 ) 2교시 국어시간 준비물

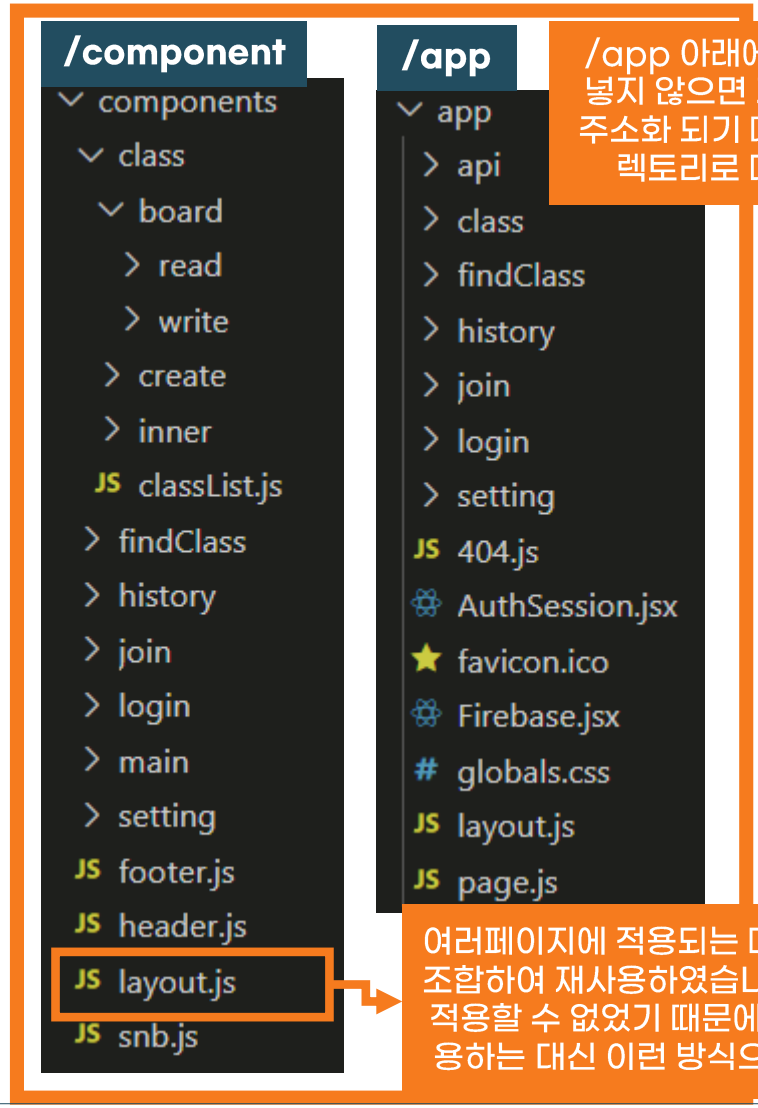
# :D 화면흐름도 - 알림장 (게시물)

# 4

# 문제 해결

# ;D 1. NEXT JS 디렉토리 구조 (FEAT. APP ROUTER)

리액트 환경에서 개발은 처음이고, 특히 AppRouter를 사용하는 Nextjs14에서는 디렉토리 구조에 의해 주소 설정이 이루어졌기 때문에 구조를 처음부터 탄탄하게 셋팅하고자 검색을 많이 해보고, 생각을 많이 했던 부분입니다.
NextJS14 만의 특성화된 장점을 최대한 사용하고 싶었기때문에 기본구조에 충실하게 적용해보았습니다.

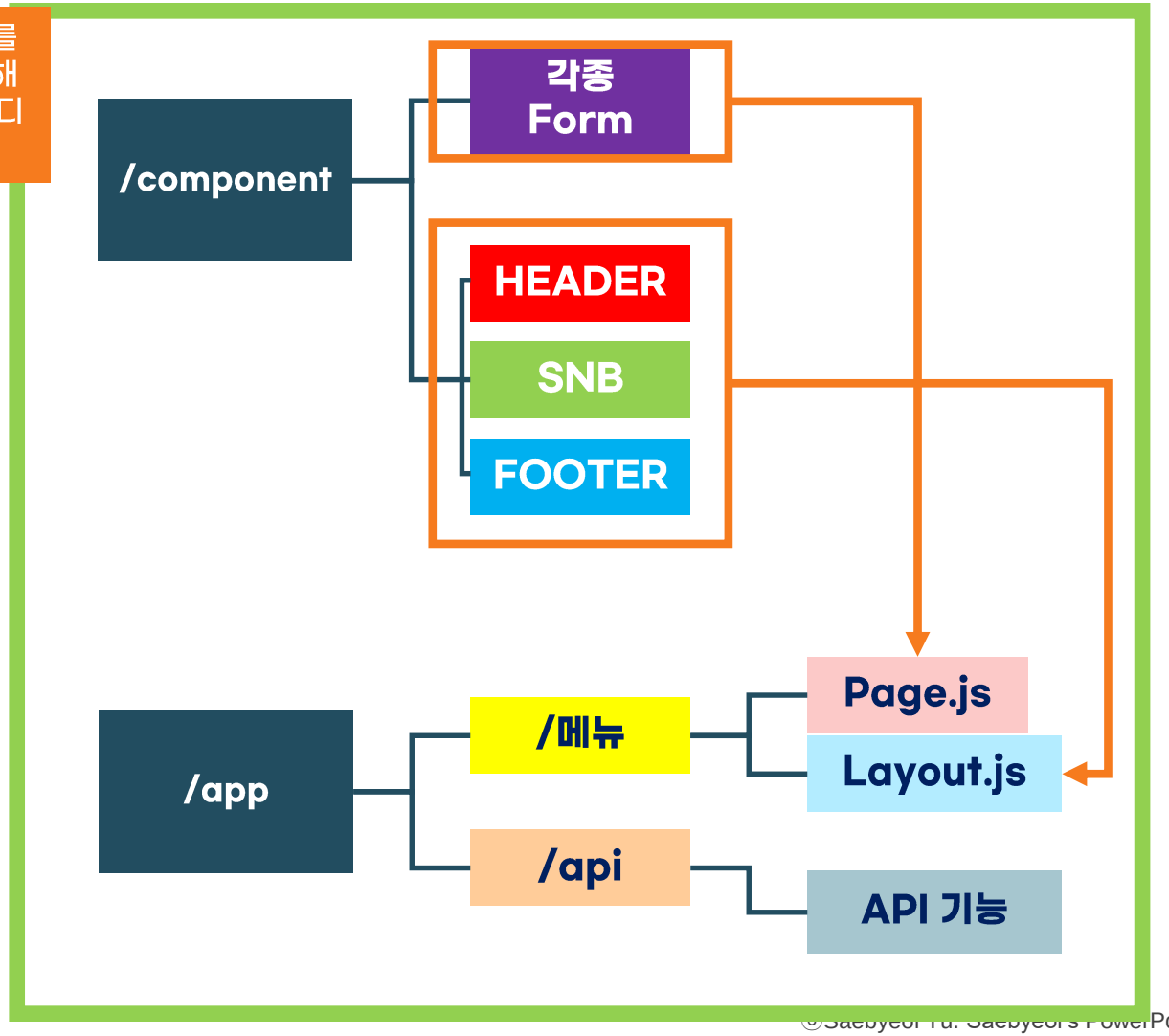






/app 아래에 있는 디렉토리는 언더바(_)를 넣지 않으면 자동으로 app router에 의해 주소화 되기 때문에 component를 상위디렉토리로 따로 빼서 작성해주었습니다.

여러페이지에 적용되는 메인 레이아웃을 따로 조합하여 재사용하였습니다. 몇몇 페이지에는 적용할 수 없었기 때문에 루트 레이아웃에 적용하는 대신 이런 방식으로 적용하였습니다.



# ;D 2. CORS 정책 및 서버 접근 보안 이슈

리액트( 클라이언트 서버 )와 스프링부트( API 서버 ) 도메인 정보가 달라 문제가 생긴 문제입니다.
로컬에서 개발할때는 클라이언트 포트 : 3000 API서버 포트 : 8080 을 사용 했기때문에 문제가 생겼습니다.
SpringSecurity 와 WebMvcConfigurer을 이용하여 해당 도메인의 접근을 API서버에서 허용하여 이를 해결하였습니다.

**WebConfig.java**

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").allowedOrigins("https://alive-easy-weevil.ngrok-free.app").allowedMethods("OPTIONS",
                "GET", "POST", "PUT", "DELETE");
    }

}
```

루트URL을 적어주면 해당 URL의 접근이 허용된다.

해당 하는 메소드의 접근이 허용 된다.

**SecurityConfig.java**

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.securityMatcher("/api", "/error")
            .authorizeHttpRequests((authz) -> authz.dispatcherTypeMatchers(DispatcherType.FORWARD).permitAll()
                    .requestMatchers(HttpMethod.GET, "**").permitAll()
                    .requestMatchers(HttpMethod.POST, "**").permitAll()
                    .requestMatchers(HttpMethod.PUT, "**").permitAll()
                    .requestMatchers(HttpMethod.DELETE, "**").permitAll().anyRequest().authenticated());

    return http.build();
}
```

/api , /error에 대해서
매칭된 메소드 접근을 허용한다.

# ;D 3. NEXT AHTH 로그인 세션 전역 적용 문제

REACT와 NEXTJS는 처음으로 사용 해보는 것이라, 어떤 방식으로 로그인을 사용 해야될지 몰랐는데, NEXT-AUTH를 사용하면 로그인 ~ 로그인 세션을 쉽게 적용할 수 있다는 것을 알게 되었습니다. 로그인 세션을 전체적으로 적용 시키기위해 session provider를 루트 레이아웃에 적용 시킬려고 했으나, 클라이언트 컴포넌트에서만 적용이 가능해서 방안을 찾던 중 AuthSession.jsx 파일을 따로 두고 하위레이아웃을 감싸는 방식으로 사용 하니까 동작이 잘 되었습니다.

**AuthSession.jsx**

Session provid는 'use client' 가 아닐 때는 동작이 불가능

```jsx
'use client';

import { SessionProvider } from "next-auth/react";

export default function AuthSession({ children }) {

    return (
        <SessionProvider>
            {children}
        </SessionProvider>
    )
}
```

**Root layout**

```jsx
AuthSession from "/src/app/AuthSession";
import Firebase from "/src/app/Firebase";

export const metadata = {
  title: "알림장",
  description: "알림장을 공유하고 검사하자!",
  icons: {
    icon: "./favicon.ico",
  }
};

export default function RootLayout({ children }) {
  return (
    <html lang="ko">
      <AuthSession>
      <Firebase>
        <body>
          {children}
        </body>
      </Firebase>
      </AuthSession>
    </html>
  )
}
```

어플리케이션 전역에 적용

# ;D 4. USE FORM 사용 시 Datepicker의 연결 에러

REACT-HOOK-FORM의 USE FORM으로 회원가입 폼의 INPUT 태그를 효율적으로 처리 할려고 했으나,
외부 라이브러리인 REACT-DATEPICKER 이용 시 USE FORM이 인식하지 못했습니다. controller를 이용하여 USE FORM에 연결 할 수 있었습니다.

**joinForm.js**

**Controller로 한번 감싸고
name, id를 셋팅 후, control 옵션을 주고,
Render 옵션 내부에 기존의 폼을 넣어준다.**

기존의 react-datepicker 폼

```
<Controller
    name="burnDate"
    id="burnDate"
    control={control}
    render={({ field }) => (
        <DatePicker
            showIcon
            placeholderText='  클릭하여 날짜선택'
            locale={ko}
            dateFormatCalendar='yyyy년 M월'
            maxDate={new Date()} // maxDate 이후 날짜 선택 불가
            selected={field.value}
            onChange={(date) => field.onChange(date)}
            dateFormat="yyyy-MM-dd"
            className="bg-gray-50 border border-gray-300 text-g
            showYearDropdown
            scrollableYearDropdown
            yearDropdownItemNumber={150}
            required
        />
    )}
/>
```

# ;D 5. S3 - FRONT CLOUD 접근 에러

알림장의 첨부 이미지를 저장하기위해 S3 BUCKET을 사용하였습니다.
S3 도메인이 노출되는 것은 보안상 위험이 있을 거 같아, FRONT CLOUDE를 연동하여 해당 이미지에 액세스하는 방식으로 이용했습니다.
S3 정책 설정 문제 때문에 해당 파일에 접근하는데 문제가 많이 생겼고, 아래 CORS 공유 설정과 버킷 정책 설정을 통해 해결하였습니다.

**해당 버킷에 대한 CORS 공유 설정**

```
"AllowedHeaders": [
    "*"
],
"AllowedMethods": [
    "HEAD",
    "GET",
    "PUT",
    "POST",
    "DELETE"
],
"AllowedOrigins": [
    "http://localhost:3000",
    "https://alive-easy-weevil.ngrok-free.app"
],
"ExposeHeaders": [
    "ETag",
    "x-amz-server-side-encryption",
    "x-amz-request-id",
    "x-amz-id-2"
```

입력한 정보를 통해
S3서버에 접근을 허용한다

**S3 버킷 정책 설정**

프론트 클라우드로 접근 가능하게하는 정책

```
{
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::alram-app-bucket/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::6078
                }
```

허용되는 S3 파일 접근 권한

프론트 클라우드의 ARN을 입력해주면
해당하는 프론트 클라우드 도메인으로
S3 버킷에 접근이 가능

프론트 클라우드 + 파일명 으로
이미지 파일에 접근이 가능해진다

```
//s3 관련 설정들
AWS.config.update({
    region: REGION,
    accessKeyId: ACCESS_KEY,
    secretAccessKey: SECRET_ACCESS_KEY,
});


//업로드
const upload = new AWS.S3.ManagedUpload(
    params: {
        ACL: "public-read",
        Bucket: BUCKET_NAME,
        Key: name,
        Body: file,
    },
});
```

```
//이미지 업로드
//업로드 된 이미지 url을 가져오기
const url_key = await upload.promise()
    .then((res) => res.Key)
    .catch((err) => {
        console.log("에러메세지" + err);
    }
    );
```

```
// 가져온 위치에 이미지를 삽입한다
editor.insertEmbed(range.index, "image", CLOUD_FRONT_URL + url_key);
```

# ;D 6. FCM WEB PUSH 백그라운드 , 이벤트 에러

FCM 매뉴얼 대로 사용 할 경우, 네이티브 서비스가 아닌 서비스 워커를 통한 메시지 수신이라, 알림 창의 클릭이벤트와 알람이 울리지 않았습니다.
이를 해결하기 위해, 기존의 정해진 항목대로 데이터를 넣는 방식이 아닌, data 라는 커스텀 객체에 정보를 담아 각 옵션들을 적용하는 방식으로 사용하였습니다.

## 기존 FCM 메뉴얼

notification 객체에 정해진 항목별로 데이터를 전송 해야함.

```
{
  "notification": {
    "title": "FCM Message",
    "body": "This is a message from FCM"
  },
  "webpush": {
    "headers": {
      "Urgency": "high"
    },
    "notification": {
      "body": "This is a message from FCM to web",
      "requireInteraction": "true",
      "badge": "/badge-icon.png"
    }
  }
}
```

FCM 서비스 자체가 설정된 기능을 덮어써서 알람이 울리지 않거나, 이벤트나 동작이 원할하게 이루어 지지 않았습니다.

변경

## /api/notification/route.js

data{ } 에 정보 담기

```
const message = {
  tokens: dev
  data: {
    title,
    body,
    icon,
    image,
    click_action,
  },
```

Cloud Messaging

FCM 서버로 송신 -> 클라이언트에서 수신

## firebase-messaging-sw.js

```
self.addEventListener("push", async (event) => {
  if (event.data) {
    // const { data } = event.dat
    const data = event.data.json().data;
    const options = {
      body: data.body,
      icon: "/icons/logo.PNG",
      // image: data.image,
      data: {
        click_action: data.click_action, // 이 필드는 밑의 클릭 이벤트 처리에 사용됨
      },
    };
    event.waitUntil(self.registration.showNotification(data.title, options));
  }
});
```

data{ } 내부의 정보를 이용

클릭 이벤트, 알림 등
포/백그라운드에서 원활하게 작동 됨.

# ;D 7. 복잡한 조인 시 JPQL 사용의 한계 -> MYBATIS 동시 운용

JPQL 사용시 테이블을 두개만 조인 할 때에는 문제가 없었으나, 테이블을 여러 개 묶는 다중 조인을 할 때 where절이나 on절에 조건을 걸어줘도 데이터가 정확하게 조회 되지 않는 현상이 있어서, 불가피하게 Mybatis를 사용하게 되었습니다. Mybatis는 실무에 있을 때 자주 사용하여, 사용 방식은 어렵지 않았지만 JPA와 동시에 사용해야 했기때문에 설정때문에 서로 충돌하는 문제가 있었고 repository를 두개로 나누어주는 방식을 사용하여 문제를 해결하였습니다.

**AlimAppBack1Application.java**

```
@SpringBootApplication
@MapperScan(basePackages = "com.alim.demo.repository.mybatis")
@EnableJpaRepositories(basePackages = "com.alim.demo.repository.jpa")
public class AlimAppBack1Application {

    public static void main(String[] args) {
        SpringApplication.run(AlimAppBack1Application.class, args);
    }

}
```

JPA와 MYBATIS의
패키지 정보를 명시해준다

**JPA/MYBATIS Repository 분기**

- com.alim.demo.repository.jpa
  - BoardRepleRepository.java
  - BoardUserRepository.java
  - ClassBoardRepository.java
  - ClassRepository.java
  - ClassUserRepository.java
  - TestRepository.java
  - UserNotificationRepository.java
  - UserRepository.java
- com.alim.demo.repository.mybatis
  - BoardClassUserMapper.java
  - ClassUserRepleMapper.java

**MyBatisConfig.java 중**

JPA의 카멜케이스 설정도 동일하게 적용하였습니다.

```
// 언더바 다음 영어 대문자로
org.apache.ibatis.session.Configuration configuration = new org.apache.ibatis.session.Configuration();
configuration.setMapUnderscoreToCamelCase(true);
sqlSessionFactoryBean.setConfiguration(configuration);

return sqlSessionFactoryBean.getObject();
```

# PIGEON
# 알림장 WEB SERVICE


---

긴 글 읽어 주셔서 감사합니다.

아래는 제 개인 프로필 사이트 입니다 :D
많은 관심 부탁드립니다!

https://dldnrud123.github.io/profile/