day91-3thproject-kakaologinapi

태그	
날짜	@2023년 2월 6일

https://vlee.kr/4896

프로젝트 생성

- npx create-react-app kakao-login

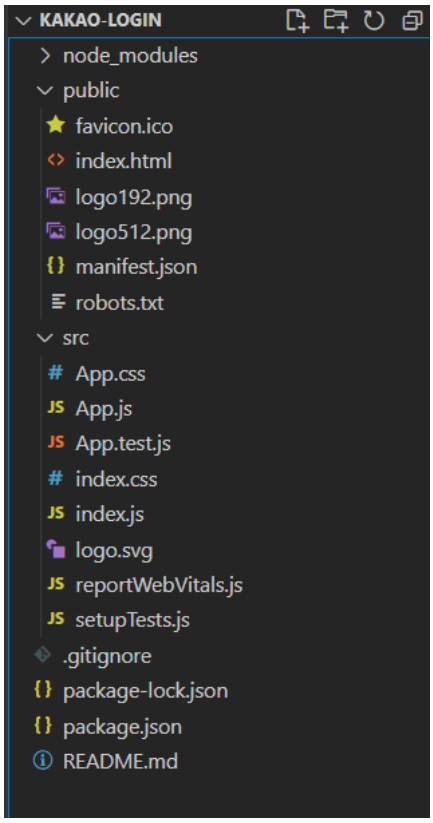
프로젝트 확인

- cd kakao-login

카카오 로그인에 사용될 Dependency 패키지들을 설치

- npm install react-router-dom axios qs

프로젝트 디렉도리로 이동하여 아래와 같은 기본 파일 목록을 확인



App.js - 필요없는 내용은 지우고 먼저 hello 가 출력되도록 수정

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
```

```
        hello
    </div>
  );
}

export default App;
```

프로젝트 실행

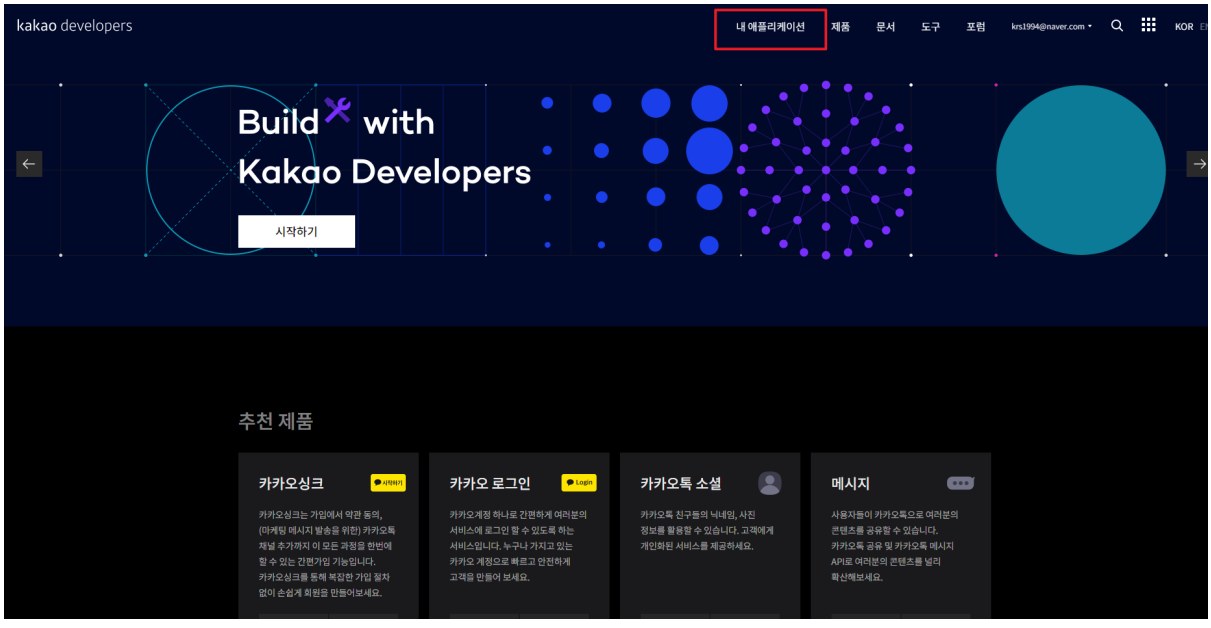
- npm start

프로젝트를 실행하고 브라우저에서 http://localhost:3000 으로 접속하면 아래와 같은 메인 페이지를 확인할 수 있다.

카카오 애플리케이션 생성

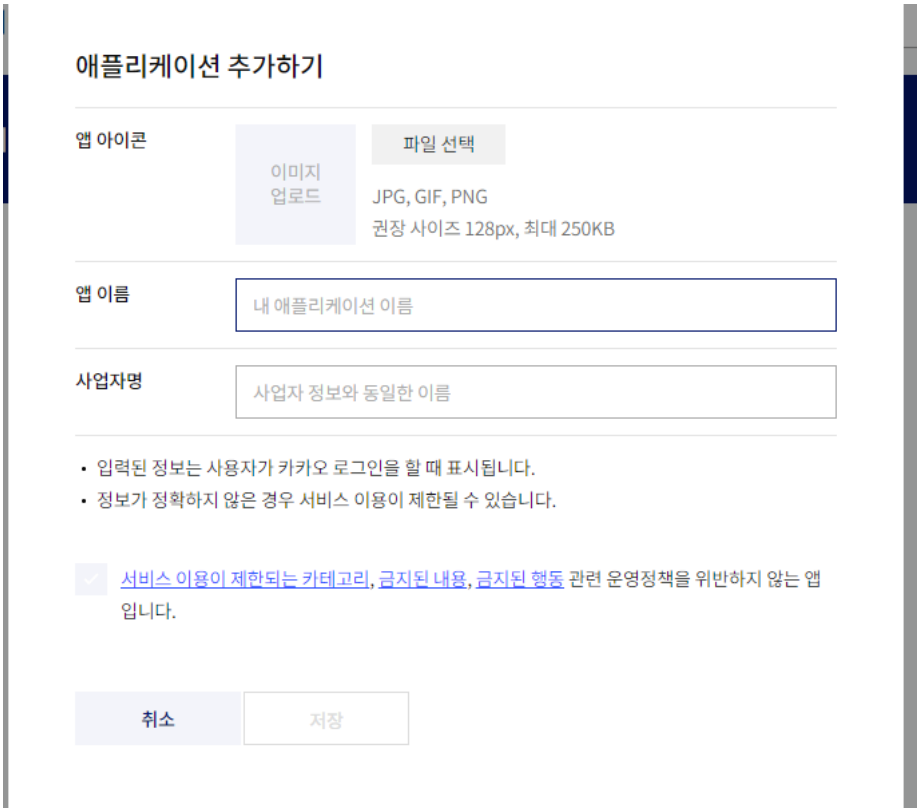
카카오 로그인 기능을 React프로젝트에 구현하기 위해 카카오 개발자 사이트에 만들어둔 Web Application 등록을 한다.

개발자 사이트 https://developers.kakao.com 에 접속해서 카카오 로그인을 위한 애플리케이션을 생성하자.

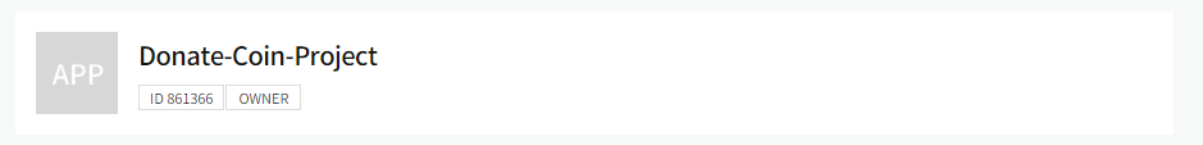


로그인 후 “내 애플리케이션” 클릭

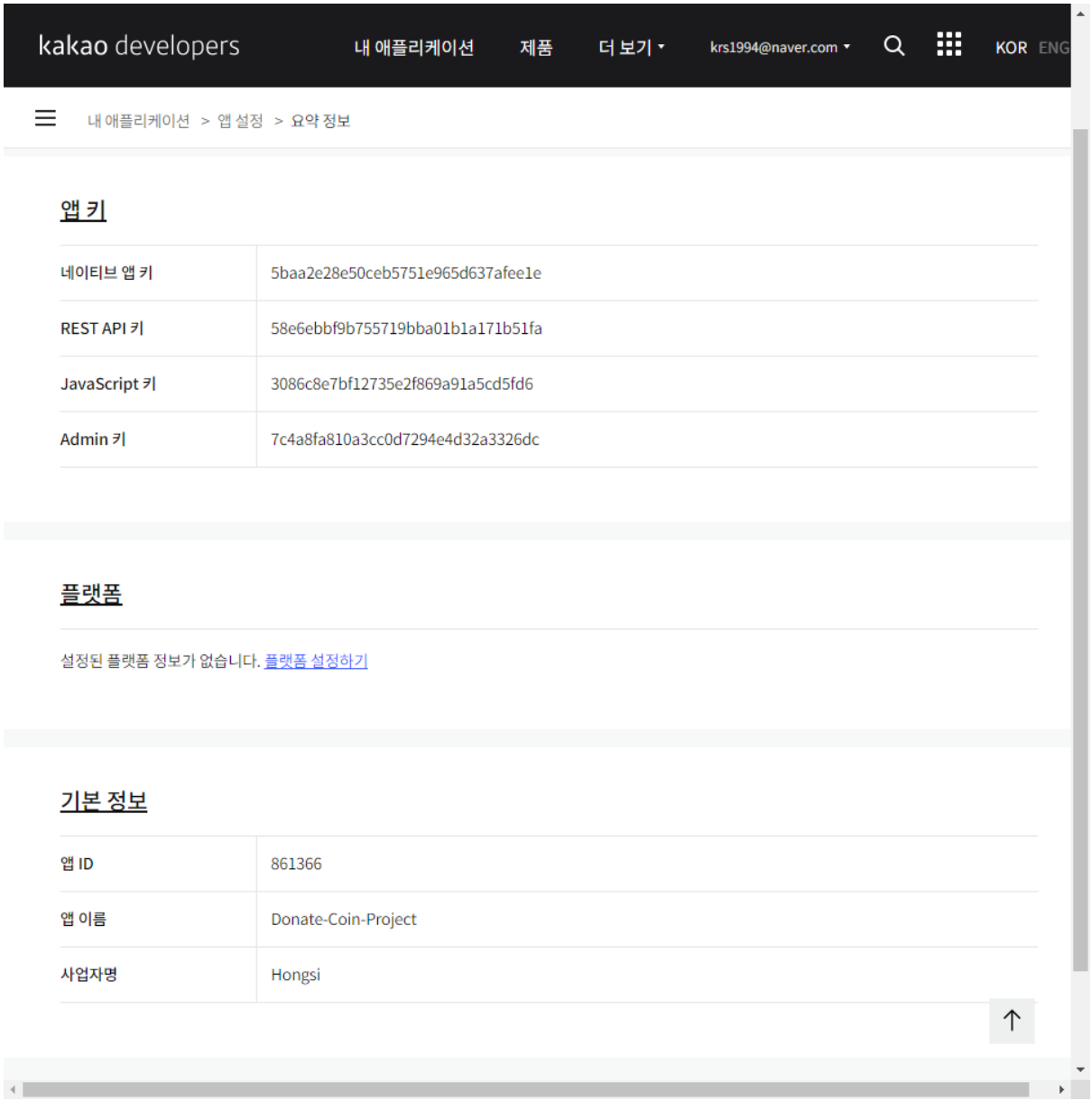
애플리케이션 추가하기 클릭



앱 이름과 사업자명을 기입한다.



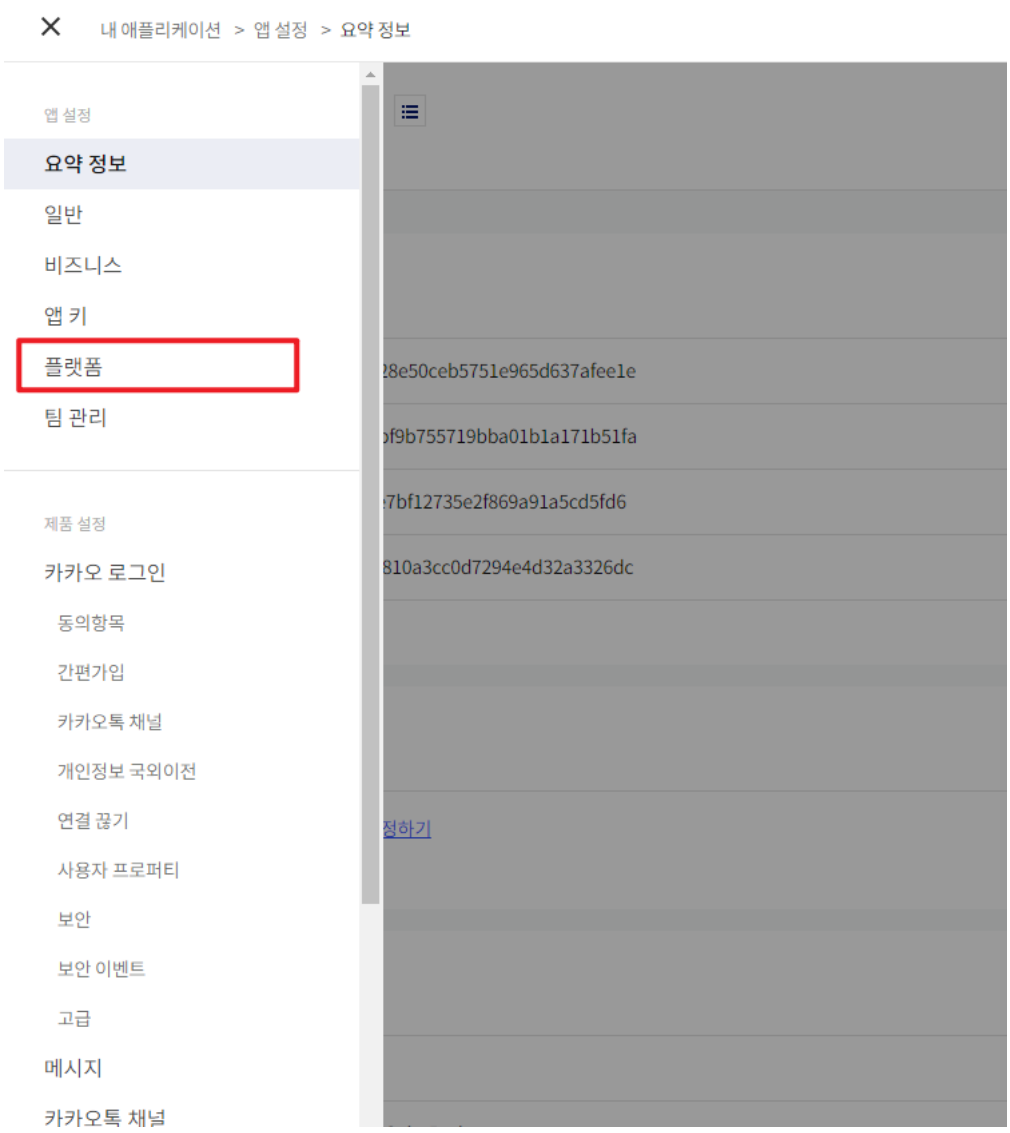
추가 된 모습 클릭하여 상세 정보들을 확인할 수 있다.

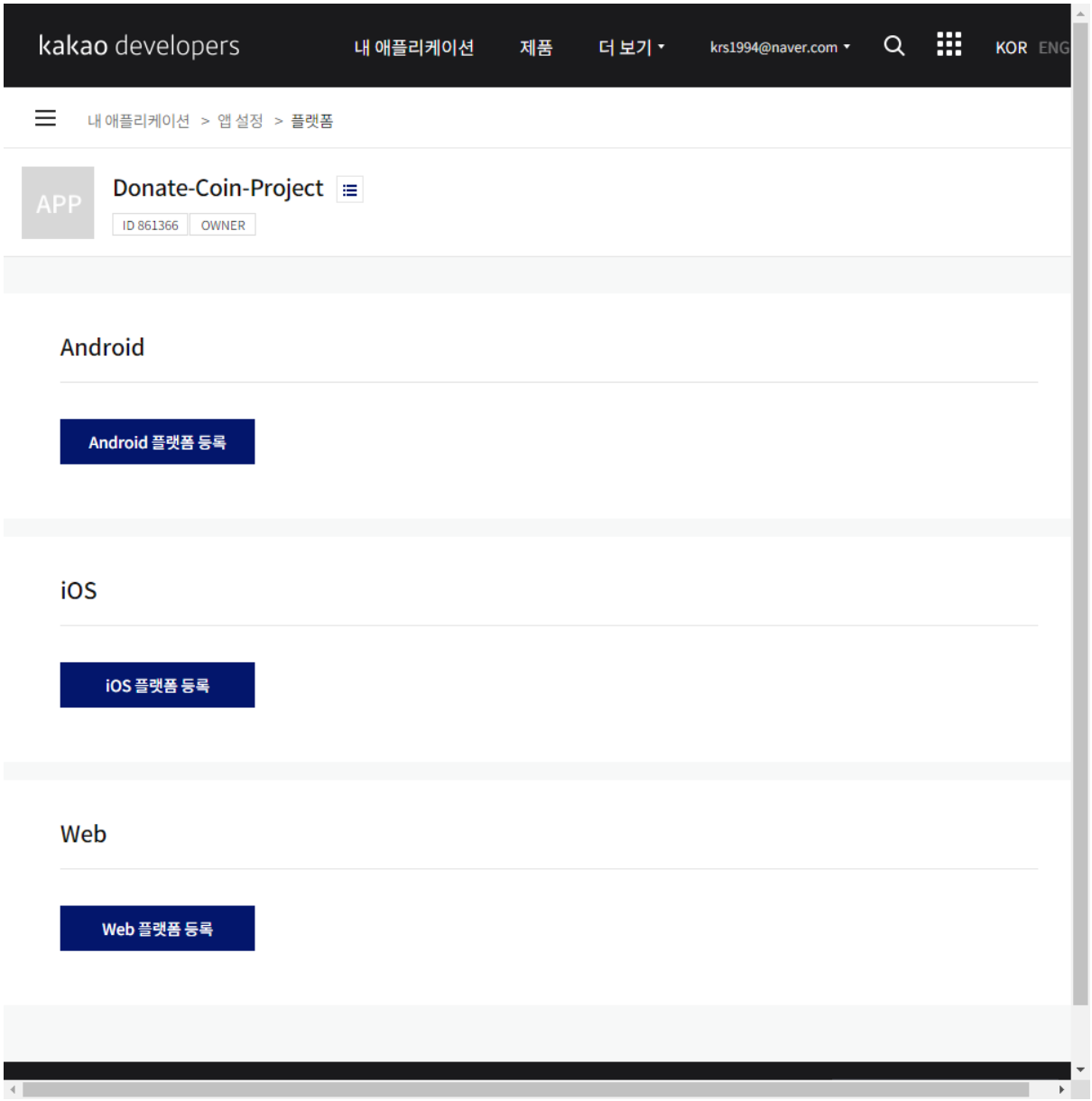


앱 키들을 확인 할 수 있다.

이 앱키들을 이용해서 Web, Android, IOS 등에서 카카오 로그인할 때 인증절차를 받을 수 있다.

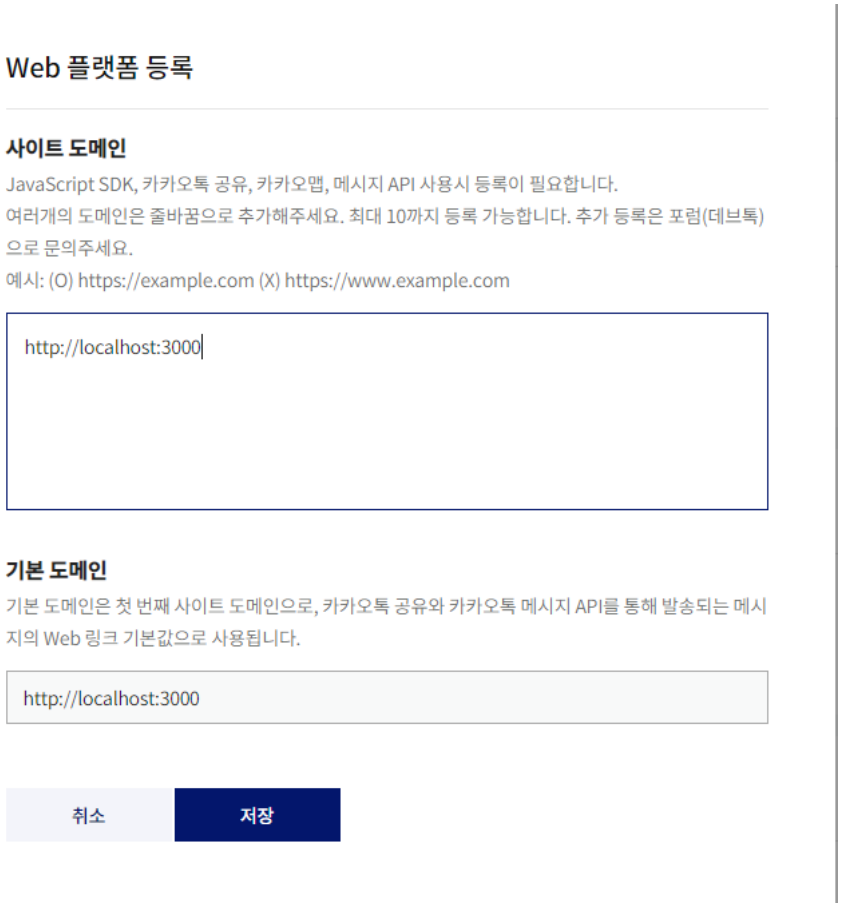
REST API를 이용해서 구현할 예정이다.


내 애플리케이션 > 앱 설정 > 요약 정보
앱 설정
요약 정보
일반
비즈니스
앱 키
플랫폼
팀 관리
제품 설정
카카오 로그인
동의항목
간편가입
카카오톡 채널
개인정보 국외이전
연결 끊기
사용자 프로퍼티
보안
보안 이벤트
고급
메시지
카카오톡 채널
28e50ceb5751e965d637afee1e
bf9b755719bba01b1a171b51fa
7bf12735e2f869a91a5cd5fd6
810a3cc0d7294e4d32a3326dc
정하기

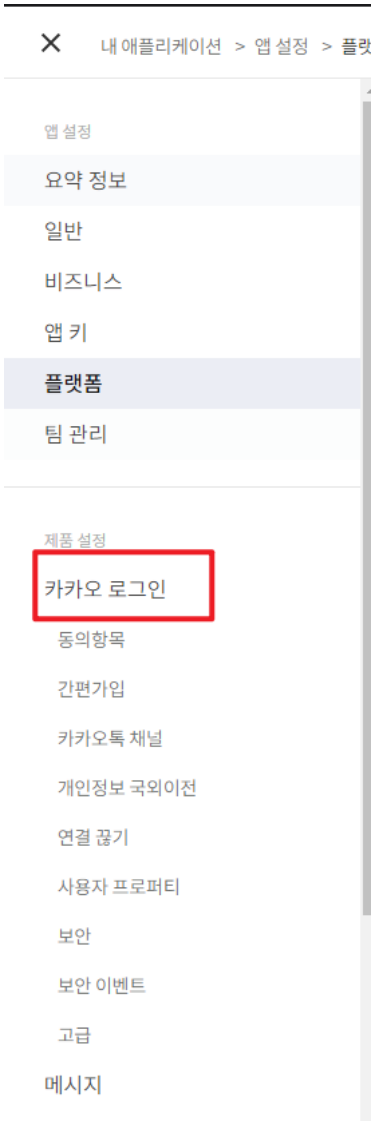


플랫폼 등록 화면으로 이동하여 “Web 플랫폼 등록” 을 해준다.

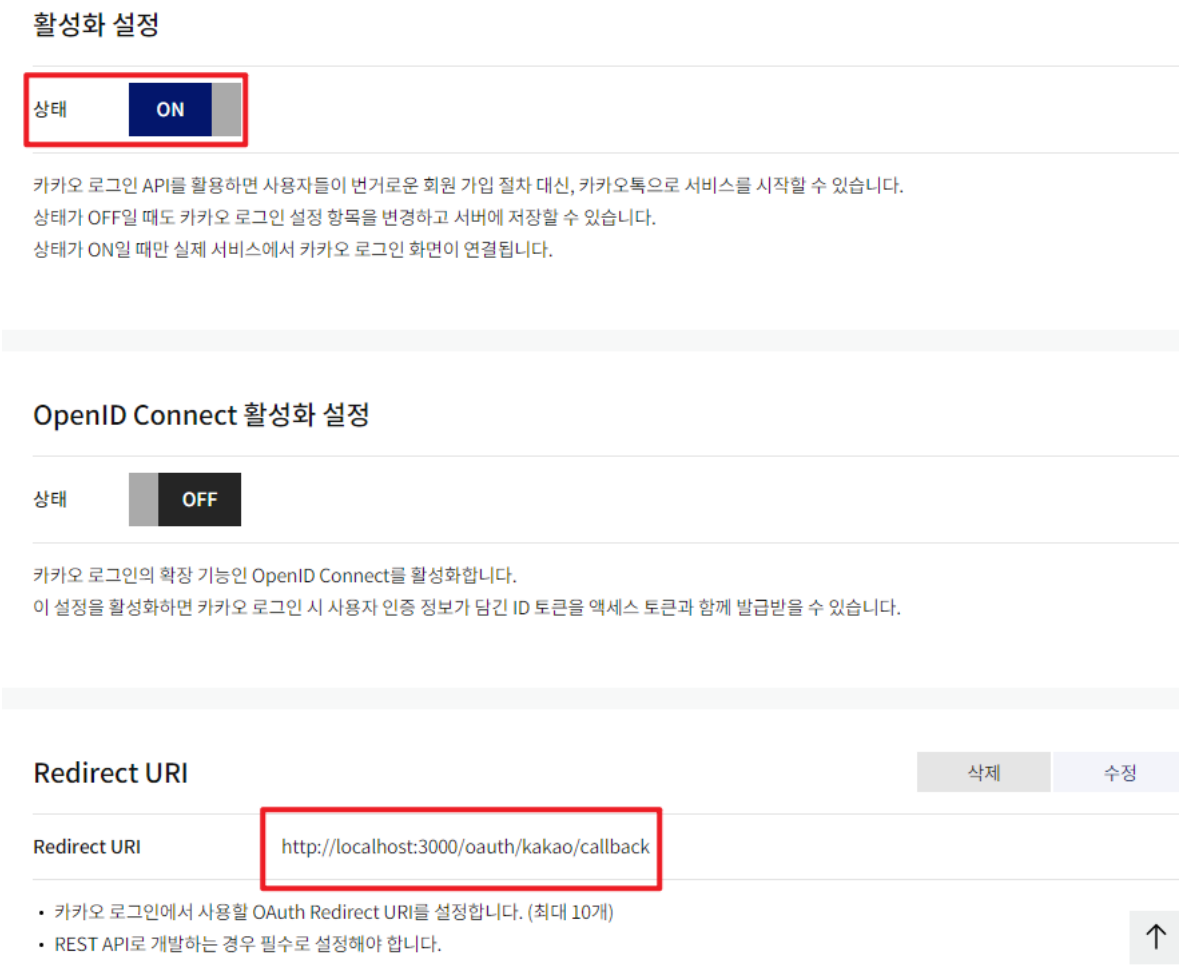
개발할 사이트 도메일을 추가해준다. 아직 개발단계이므로 localhost주소로 입력하자.



다음으로는 카카오 로그인 설정을 활성화 하고 Redirect URI를 추가한다.



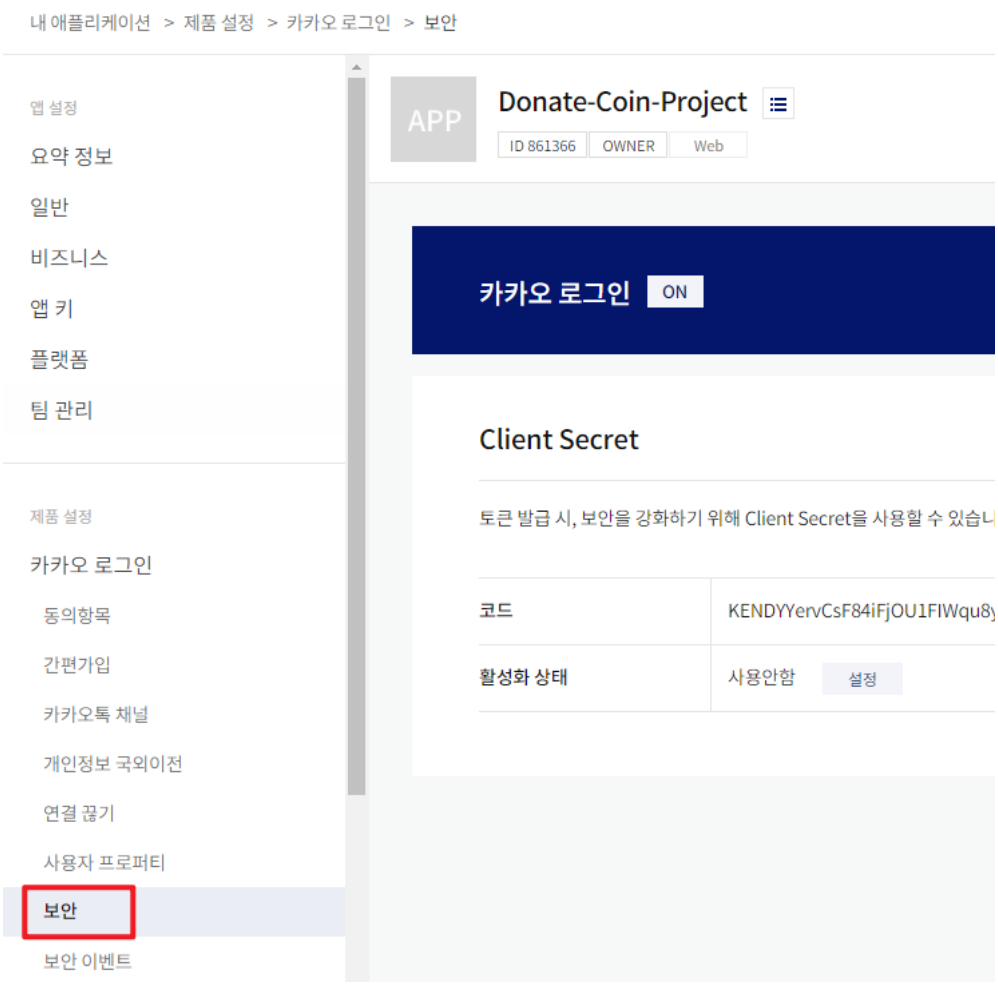
http://localhost:3000/oauth/kakao/callback



Redirect URI는 REST API키를 이용해서 카카오에 “인가 코드”를 받는 callback 주소 이다. 이후 소스를 보면 알겠지만, 카카오 로그인 버튼은

```
https://kauth.kakao.com/oauth/authorize?client_id=${REST_API_KEY}&redirect_uri=${REDIRECT_URI}&response_type=code
```

위와 같은 주소를 호출하고 카카오 서버에서 다시 redirect로 우리가 입력해준 redirect URI로 이동하면서 인가 코드 값을 전달해주는 형태로 진행된다.



다음으로는 REST API를 사용하는 경우 필요한 Client Secret 코드 발급이다.

보안 메뉴에서 코드 발급을 한다.

인가 코드 발급 받기

이제 프로젝트로 돌아와 인가 코드를 발급 받도록 구현해보자.

./src/Auth.js 생성

```
import React from "react";

const Auth =()=>{
    const code = new URL(window.location.href).searchParams.get("code");
    return (
        <div>
            {code}
        </div>
    );
};

export default Auth;

// 코드 내용 Redirect 주소로 전달받은 code값을 추출하여 보여주는 코드이다.
```

App.js 수정

```
import logo from "./logo.svg";
import "./App.css";
import Auth from "./Auth";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";

function App() {
  const REST_API_KEY = "58e6ebbf9b755719bba01b1a171b51fa";
  const REDIRECT_URI = "http://localhost:3000/oauth/kakao/callback";
  const KAKAO_AUTH_URL = `https://kauth.kakao.com/oauth/authorize?client_id=${REST_API_KEY}&redirect_uri=${REDIRECT_URI}&response_type

  return (
    <Router>
    <div className="App">
      <Routes>
        <Route exact path="/" element={<a href={KAKAO_AUTH_URL}>Kakao Login</a>} />
        <Route path="/oauth/kakao/callback" element={<Auth />} />
      </Routes>
    </div>
    </Router>

  );
};

export default App;
```

주의!!

[React 에러 일지] ... is not a component. All component children of must be a or

🚫 [...] is not a component. All component children of must be a or react-router-dom v6부터는, Switch 대신 Routes를 사용 Route 안에 component 대신 element 사용 그리고 자식으로는 만 가능하다. 따라서 v6 이전처럼 아래와 같이 코드를 작성하면 에러가 발생하는 것이다.

https://velog.io/@nemo/react-error-routes

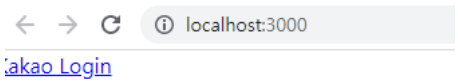
velog

1.

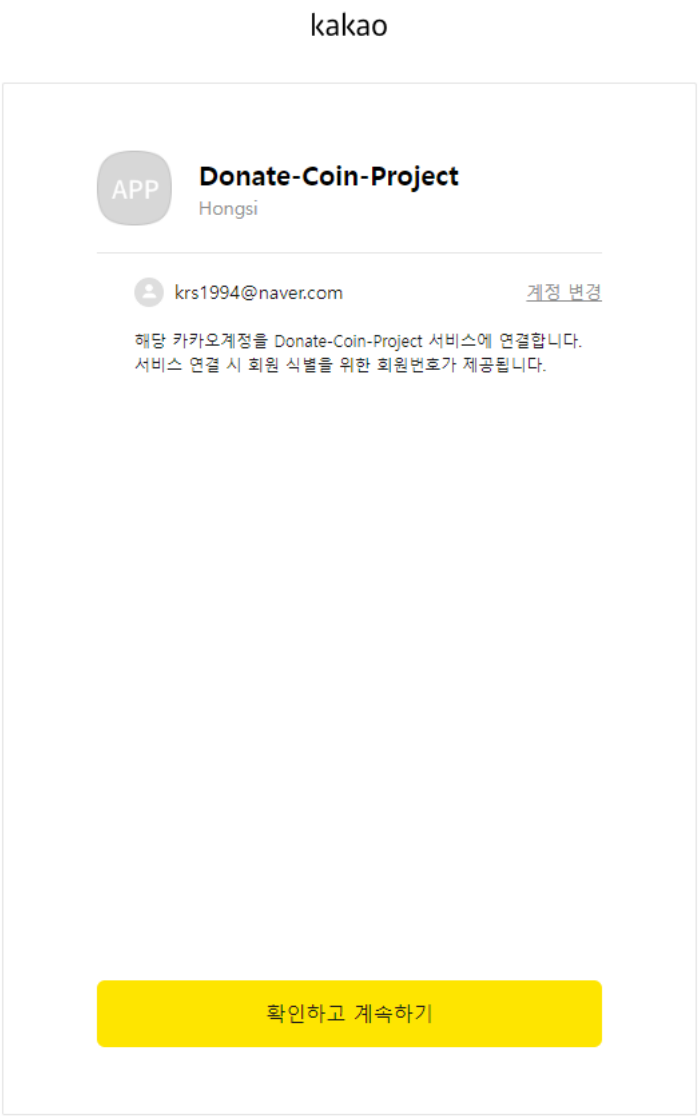
```
<Router>
  <Routes>
      <Route path="" element={} />
  </Routes>
</Router>

위 형태로 구성된다.
Switch 대신 Routes를 사용하고, Router 사이 컴포넌트를 element로
넣어준다.
```

코드를 보면 router를 이용해 kakao로 부터 인가 코드를 받기 위한 주소 링크를 메인 페이지에 띄우고 callback 주소로 “/oauth/kakao/callback” path를 추가하고 해당 path에 Auth page가 연결되도록 하였다.



페이지를 띄워보면 위와 같은 링크가 생성된 걸 볼 수 있다. 링크를 클릭하면 아래와 같이 카카오 계정 연동 페이지가 호출 된다.



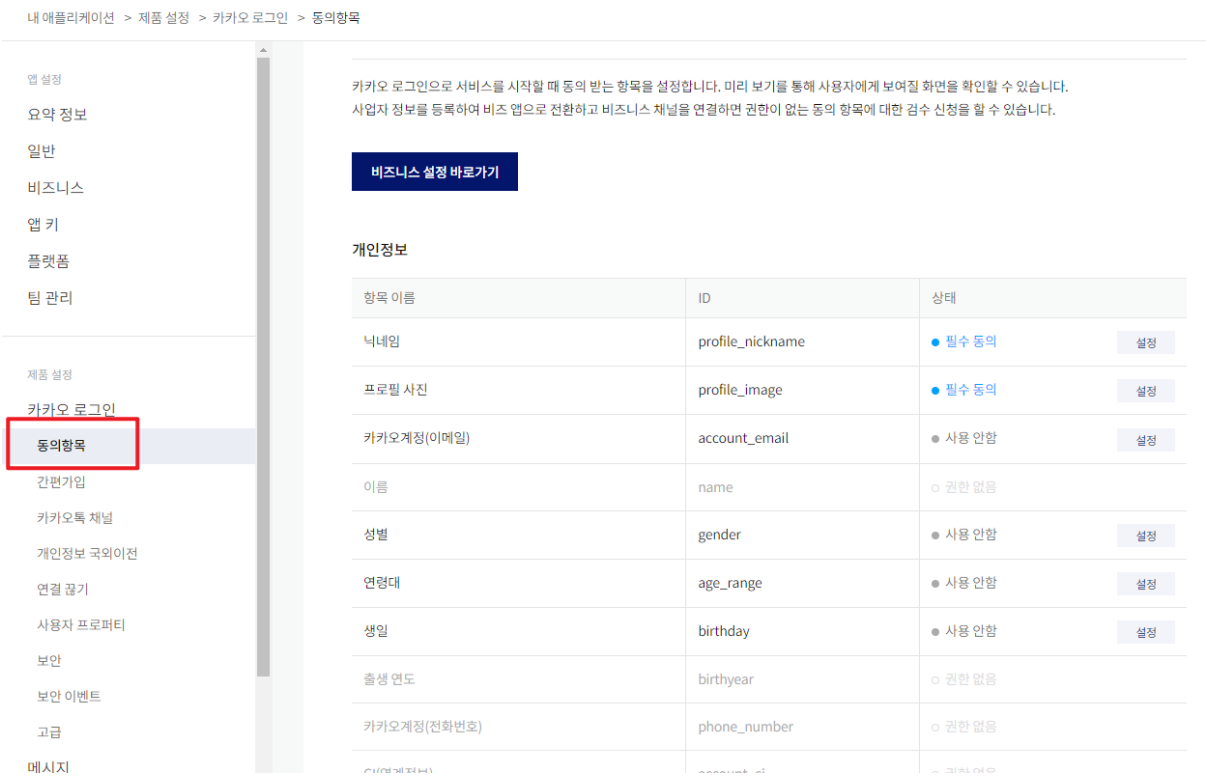
'동의하고 계속하기' 버튼을 누르면 필자가 만든 프로젝트의 callback 주소로 페이지가 다시 이동 된다. 이 때 인가 코드가 같이 전달되고 앞서 Auth.js에서 해당 코드를 출력하게 했으므로 아래와 같은 결과가 나온다.

```
JuPQsKMPPn2DkUpUazywjalgqAF08kpsUrZbfhnsrtu04I5K9xy0PKoiAeNZMmejoKjhQgo9c00AAAGGJcO0rg
```

Access Token 발급과 사용자 정보 가져오기

이제 실제 사용자 정보에 접근하기 위해 앞서 받은 인가 코드를 이용해 Access Token을 받고 사용자 정보를 획득해보자.

사용자 정보를 가져오기 위해서는 카카오 개발자 페이지의 동의항목에서 다음과 같이 개인 정보 사용 동의 설정을 해야한다.



./public/index.html 수정

Kakao Javascript SDK를 사용하기 위해 index.html의 <head></head> 태그에 아래 코드를 추가합니다.

REST API를 이용해서 Access Token을 발급 받는 것 까지는 카카오 Javascript SDK 없이 진행하는데 문제가 없으나 사용자 프로필을 가져오기 위한 kapi.kakao.com/v2/user/me URL 도메인에서 데이터를 획득하려면 CORS 정책 위반 에러가 발생합니다. Access Token 까지만 발급 받고 나머지 작업은 Backend에서 진행해도 되지만 본 포스트에서는 React 프로젝트 내에서 사용자 정보를 가져오기 위해 Kakao Javascript SDK를 사용했습니다.

```
<script src="https://developers.kakao.com/sdk/js/kakao.min.js"></script>
```

App.js 수정

```
import logo from "./logo.svg";
import "./App.css";
import Auth from "./Auth";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import Profile from "./profile";
//profile 추가

function App() {
  // 본인의 REST API KEY 값. 요약 정보에서 확인 가능하다.
  const REST_API_KEY = "58e6ebbf9b755719bba01b1a171b51fa";

  const REDIRECT_URI = "http://localhost:3000/oauth/kakao/callback";
  const KAKAO_AUTH_URL = `https://kauth.kakao.com/oauth/authorize?client_id=${REST_API_KEY}&redirect_uri=${REDIRECT_URI}&response_type

  return (
    <Router>
    <div className="App">
      <Routes>
        <Route exact path="/" element={<a href={KAKAO_AUTH_URL}>Kakao Login</a>} />
        <Route path="/oauth/kakao/callback" element={<Auth />} />
        //profile 추가
        <Route path="/profile" element={<Profile />} />
      </Routes>
    </div>
```

```
    </Router>

  );
};

export default App;
```

Auth.js 수정

```
import React from 'react';
import { useEffect } from 'react';
import axios from "axios";
import qs from "qs";
import {useNavigate} from "react-router-dom"

const Auth = () => {
    // 본인의 REST API KEY 값. 요약 정보에서 확인 가능하다.
    const REST_API_KEY = "58e6ebbf9b755719bba01b1a171b51fa";
    const REDIRECT_URI = "http://localhost:3000/oauth/kakao/callback";
    // 본인의 CLIENT SECRET 값
    const CLIENT_SECRET = "KENDYYervCsF84iFjOU1FIWqu8yb1pMj"

    // callback으로 받은 인가코드
    const code = new URL(window.location.href).searchParams.get("code");

    const history = useNavigate();

    const getToken = async () => {
        const payload = qs.stringify({
            grant_type : "authorization_code",
            client_id : REST_API_KEY,
            redirect_uri : REDIRECT_URI,
            code : code,
            client_secret : CLIENT_SECRET,
        });

        try {
            //access token 가져오기
            const res = await axios.post(
                "https://kauth.kakao.com/oauth/token",
                payload
            );

            // kakao javascript SDK 초기화
            window.Kakao.init(REST_API_KEY);
            // access token 설정
            window.Kakao.Auth.setAccessToken(res.data.access_token);
            history.replace("/profile");

        }catch (err) {
            console.log(err);
        }
    };

    useEffect(()=>{
        getToken();
    },[]);
    return null;
};

export default Auth;
```

Auth.js 에서 인가 코드를 이용해서 Access Token을 가져옵니다. 가져온 Access Token을 Kakao 객체에 설정하기 위해 Kakao.Init()함수와 Kakao.Auth.setAccessToken() 함수를 호출합니다. 그리고 /profile 페이지로 이동합니다.

Profile.js 생성

Auth.js에서 Profile.js 페이지를 호출한다.

```
import React, {useEffect, useState} from "react";

const Profile =()=> {
    const [user_id, setUserId] = useState();
    const [nickName, setNickName] = useState();
    const [profileImage, setProfileImage] = useState();

    const getProfile = async()=>{
```

```
        try{
            // Kakao SDK API를 이용해 사용자 정보 획득
            let data = await window.Kakao.API.request({
                url : "/v2/user/me",
            });
            // 사용자 정보 변수에 저장
            setUserId(data.id);
            setNickName(data.properties.nickname);
            setProfileImage(data.properties.profile_image);
        }catch(err){
            console.log(err);
        }
    };

    useEffect(()=>{
        getProfile();
    },[]);
    return(
        <div>
            <h2>{user_id}</h2>
            <h2>{nickName}</h2>
            <img src={profileImage}></img>
        </div>
    );
};

export default Profile;
```

Kakao SDK의 Kakao.API.request를 호출한다. /v2/user/me 경로를 호출하면 사용자 정보를 가져올 수 있다. Request/Response의 상세 규격은 링크페이지 참고

https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api#req-user-info

로그아웃 구현

kakao developers에도 나와있지 않아서 직접 구현해야 했는데

막상 해보니 어찌어찌 해결이 된것 같다.

새로운 컴포넌트를 만드는것 보다 원래 있던 프로파일 컴포넌트에 직접 코딩하였다.

kakao developers

내 애플리케이션 - 고급 - Logout Redirect URI 설정 (로그아웃 시 초기 화면으로 돌아가도록 설정했다. http://localhost:3000)

Profile.js에 추가

```
const KAKAO_LOGOUT_URL = `https://kauth.kakao.com/oauth/logout?client_id=${REST_API_KEY}&logout_redirect_uri=${REDIRECT_URI}`;

return(
  <Link to={KAKAO_LOGOUT_URL}>로그아웃 </Link>
)
```