

TravelLog

프로젝트명: 여행기록

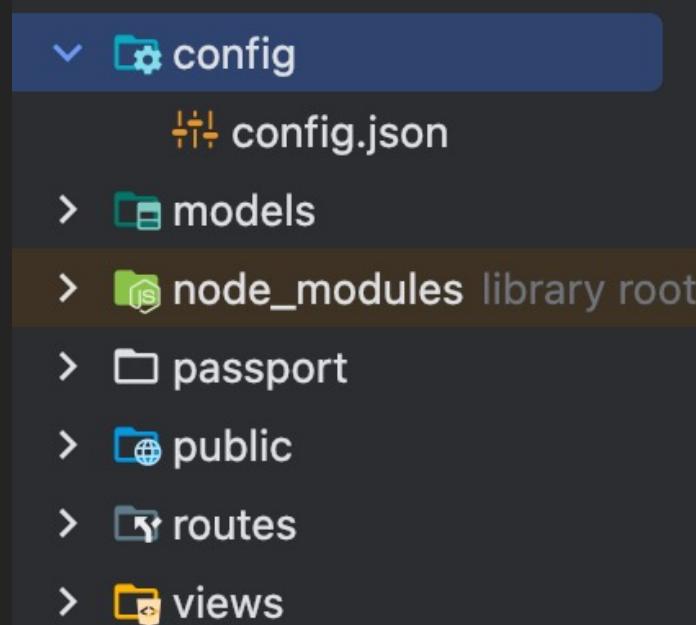
내가 다녀온 여행지를
지도에서 검색하고, 그
곳에서의 기록을 작성한다.

고급웹프로그래밍
60202455 나경호



config

- DB 설정 config 파일



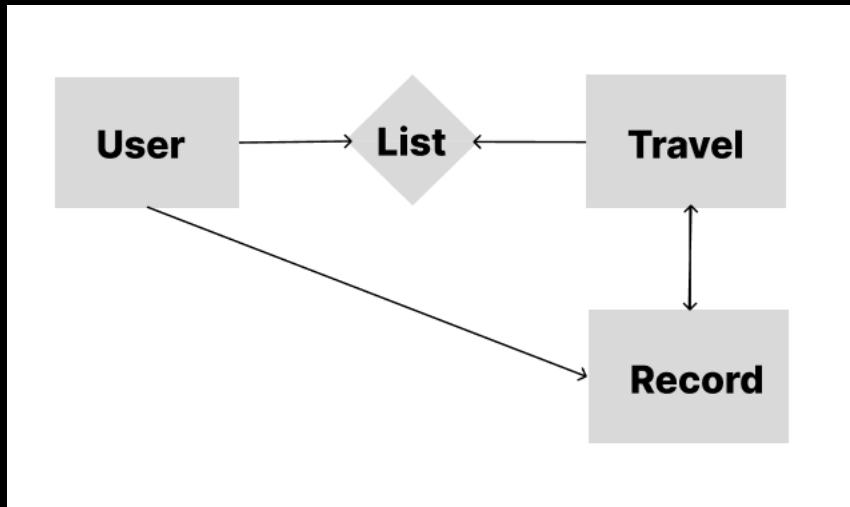
.env

- 프로젝트를 실행했던 환경 변수입니다.

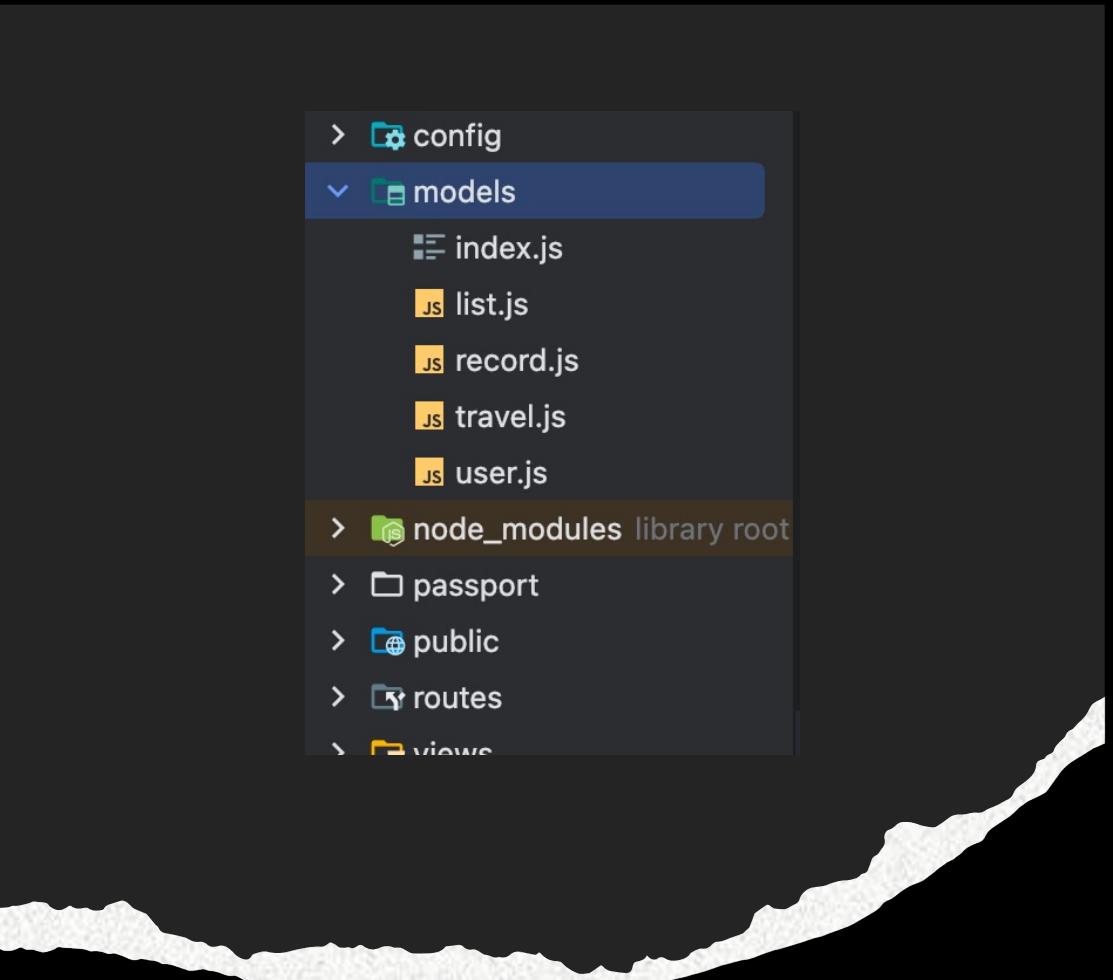
```
PORT=3000
COOKIE_SECRET=secret
jsAppKey=4b656b81a4c9537326c8205936
4f571c
KAKAO_AppKey=7211b3568a685672d9e7
6795132ef25e
MYSQL_USERNAME=root
MYSQL_PASSWORD=Gh852852!
MYSQL_DATABASE=webboard
MYSQL_HOST=127.0.0.1
```

models

- Sequelize로 사용할 DB 관계를 정의 및 설정합니다.



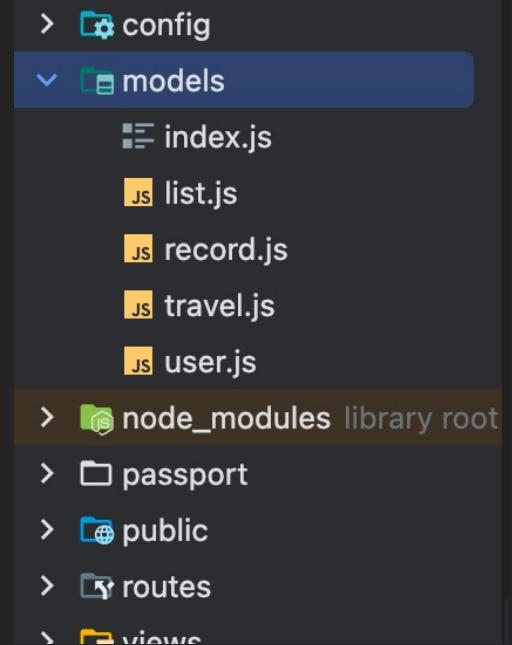
Chen 방식 ERD



models

list.js

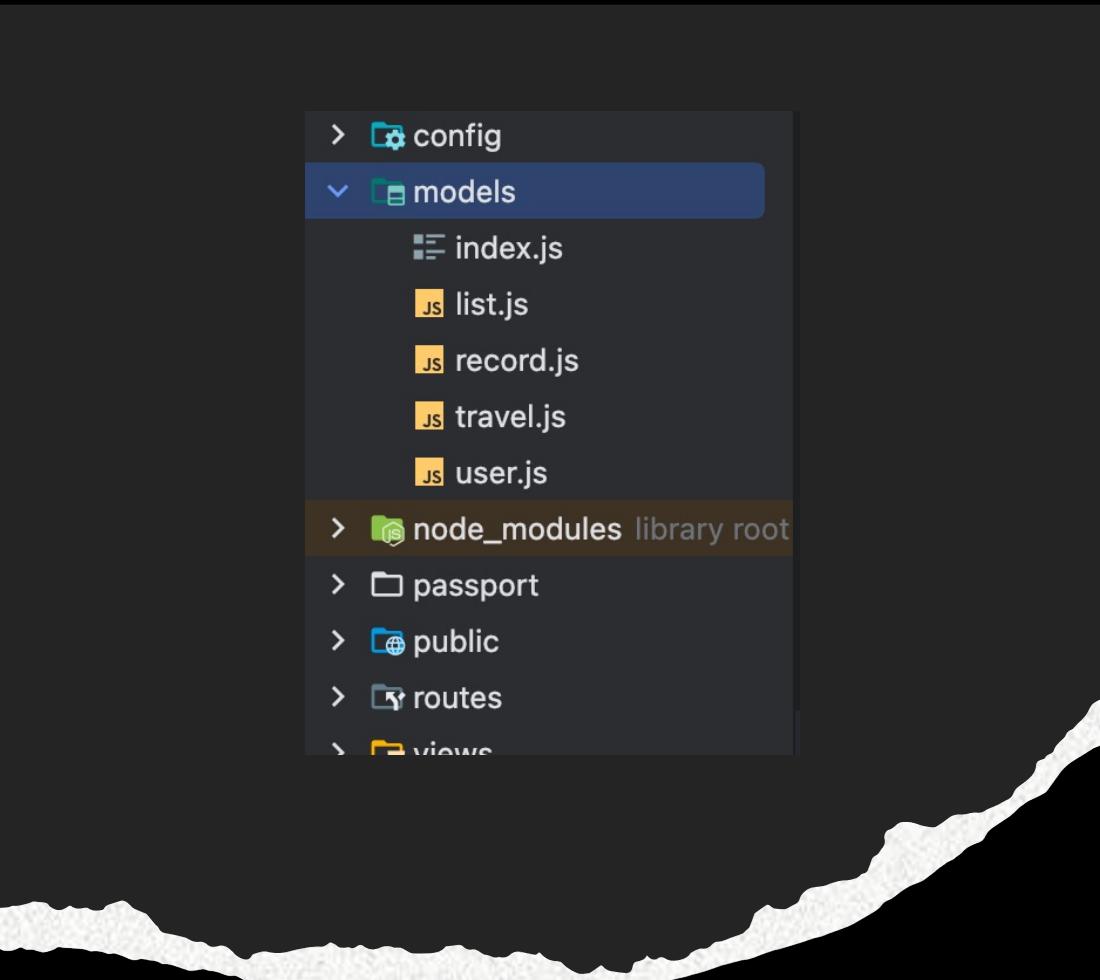
- List 모델을 정의하고 여행지 정보를 저장하며 사용자 및 여행지와 다대다 관계를 설정합니다.



models

record.js

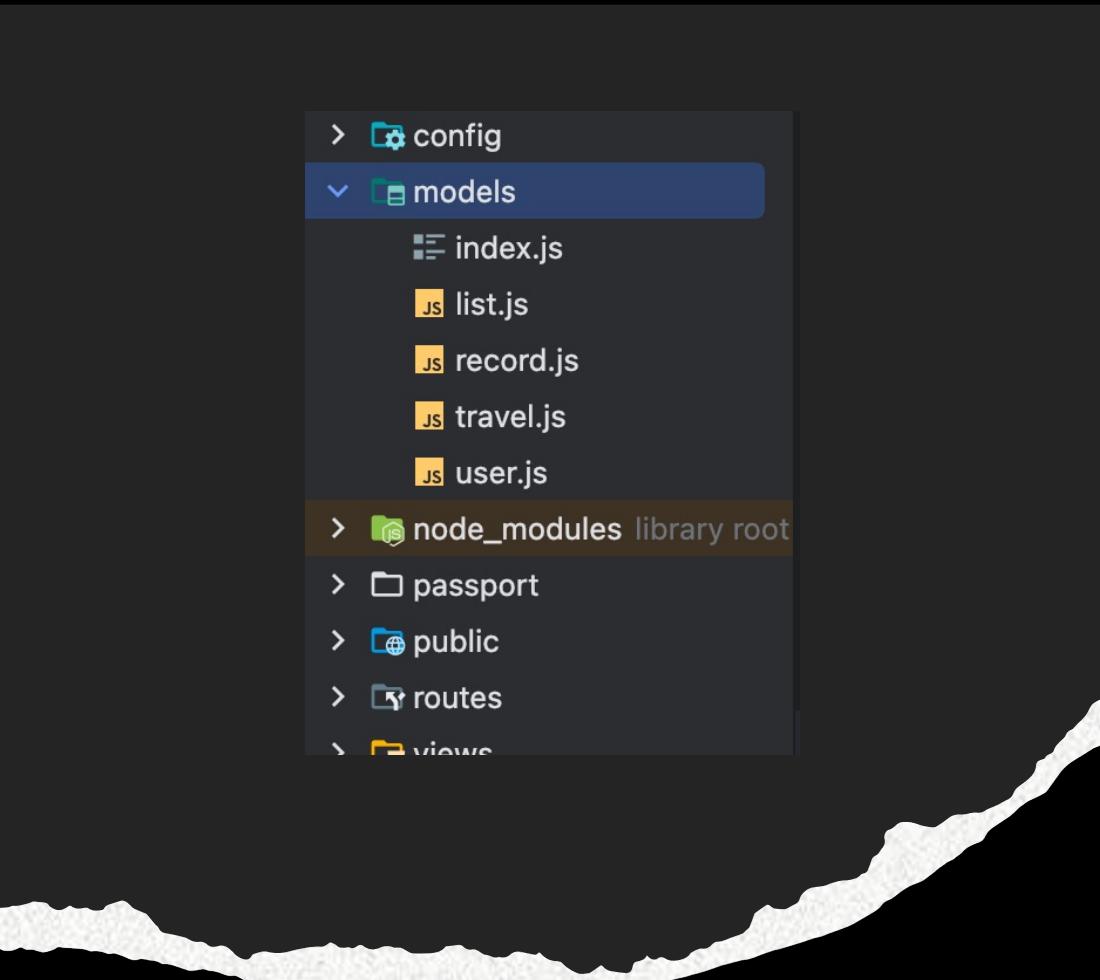
- Record 모델을 정의하고 사용자 및 여행지와 연관된 기록지 데이터를 저장합니다.



models

travel.js

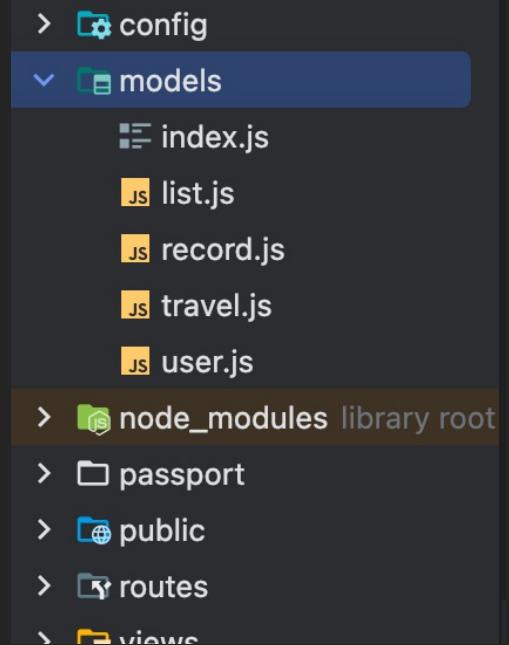
- Travel 모델은 여행지 정보를 저장하고, 관련된 기록을 연결하여 연관성을 정의합니다.



models

user.js

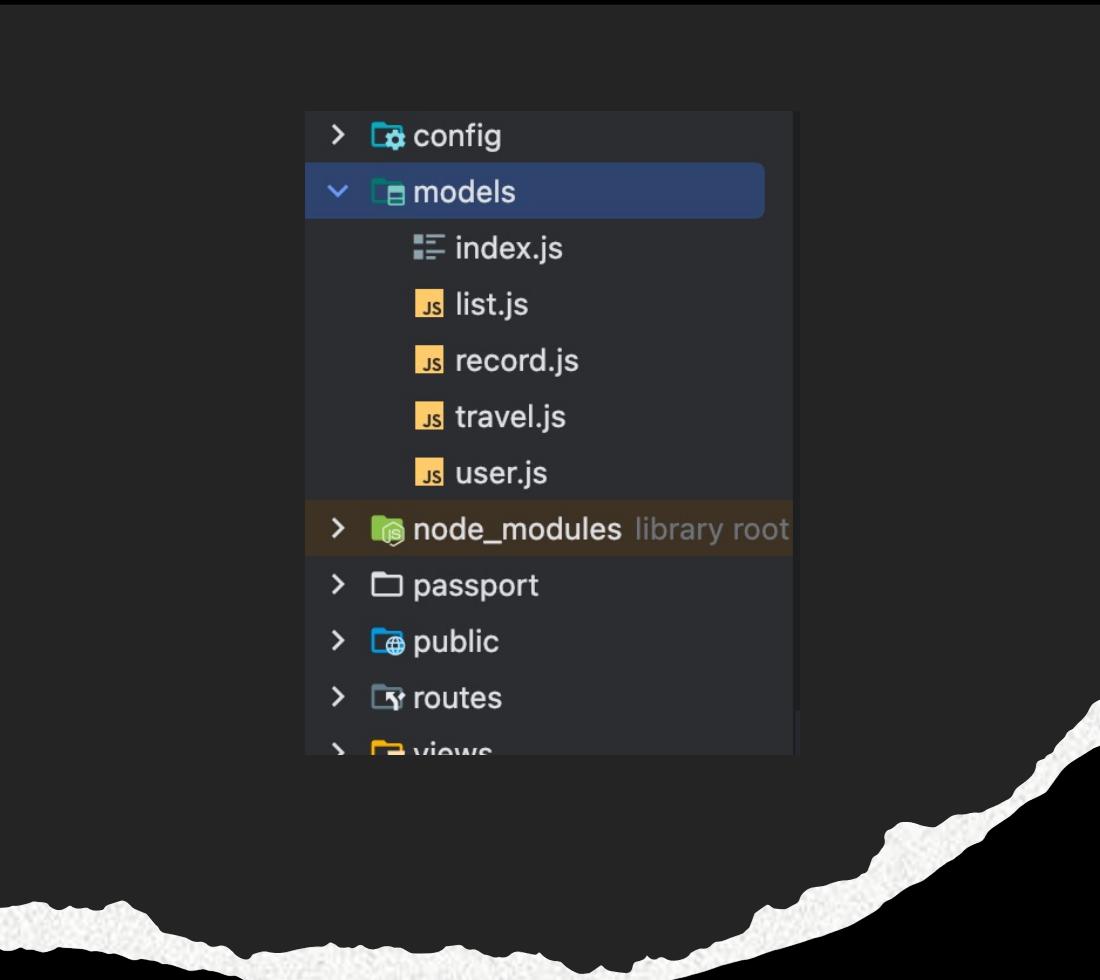
- User 모델은 회원 정보를 저장하며,
각 회원이 작성한 여러
기록(Record)과의 연관성을 정의합니다.



models

index.js

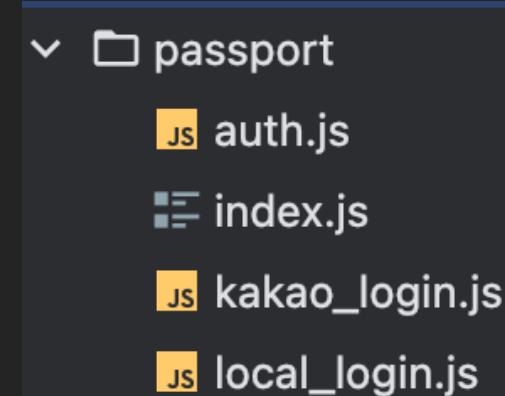
- Sequelize를 사용하여 데이터베이스 모델을 초기화하고 관계를 설정합니다.
- 각각의 모델 파일(User, Travel, List, Record)을 불러와 초기화하고, 데이터베이스 연결 설정합니다.



passport

kakao_login.js

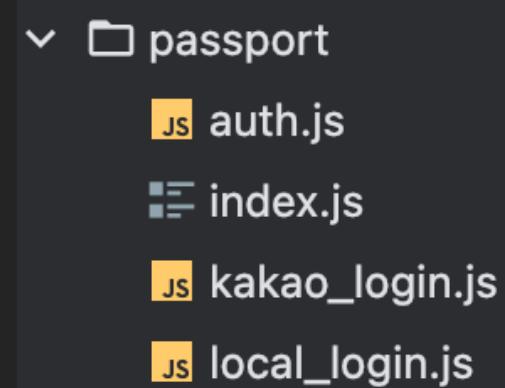
- Passport 모듈을 사용하여 Kakao 계정으로 로그인하는 기능입니다.
- 사용자가 Kakao 계정으로 인증을 시도하면, 해당 사용자의 프로필 정보와 이메일을 데이터베이스에서 조회합니다.
- 이미 등록된 사용자라면 로그인을 처리하고, 없는 경우 새로운 사용자로 데이터베이스에 저장합니다.



passport

local_login.js

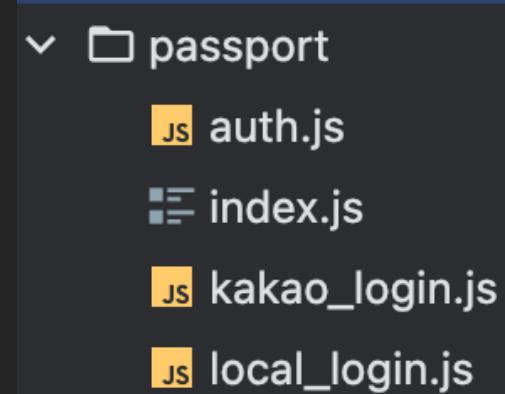
- Passport 모듈을 사용하여 local 로그인을 구현했습니다. 사용자가 이메일과 비밀번호를 입력하면, 데이터베이스에서 해당 이메일을 가진 사용자를 찾습니다.
- 이메일이 존재하면 비밀번호를 저장된 비밀번호와 비교하여 인증을 처리합니다.
- 인증이 성공하면 사용자 객체를 Passport에 전달하여 로그인을 완료합니다.



passport

index.js

- 사용자 인증 및 세션 관리를 설정합니다.
- serializeUser 함수는 사용자 ID를 세션에 저장하고, deserializeUser 함수는 세션에서 ID를 이용해 사용자를 조회합니다.
- local() 함수는 로컬 로그인 전략을 설정하고, kakao() 함수는 카카오 로그인 전략을 설정합니다.

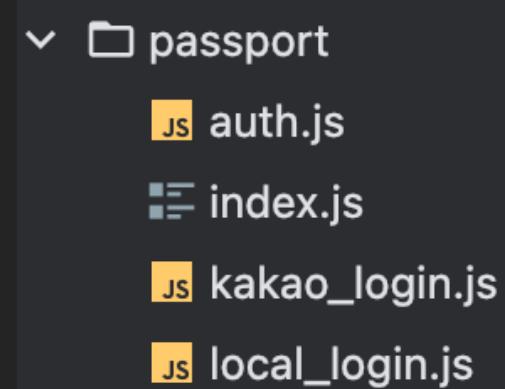


passport

auth.js

isLoggedIn 함수

- req.isAuthenticated()를 사용하여 사용자가 로그인한 상태인지 확인합니다.
- 로그인한 경우 next()를 호출하여 다음 미들웨어로 넘어갑니다.
- 로그인하지 않은 경우 403 Forbidden 상태 코드와 함께 클라이언트에게 알림을 전달하고, 로그인 페이지로 리다이렉트합니다.

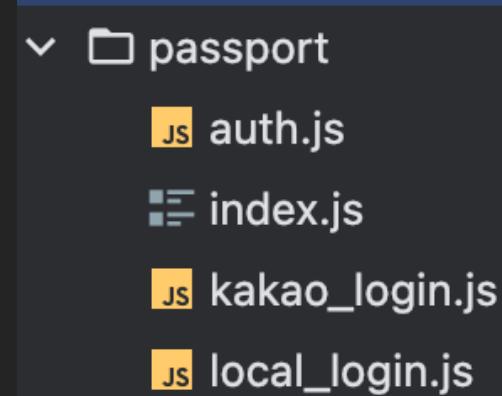


passport

auth.js

isNotLoggedIn 함수

- req.isAuthenticated()를 사용하여 사용자가 로그인한 상태가 아닌지 확인합니다.
- 로그인하지 않은 경우 next()를 호출하여 다음 미들웨어로 넘어갑니다.
- 이미 로그인한 경우 403 Forbidden 상태 코드와 함께 클라이언트에게 알림을 전달하고, 이전 페이지로 되돌아갑니다(history.back()).

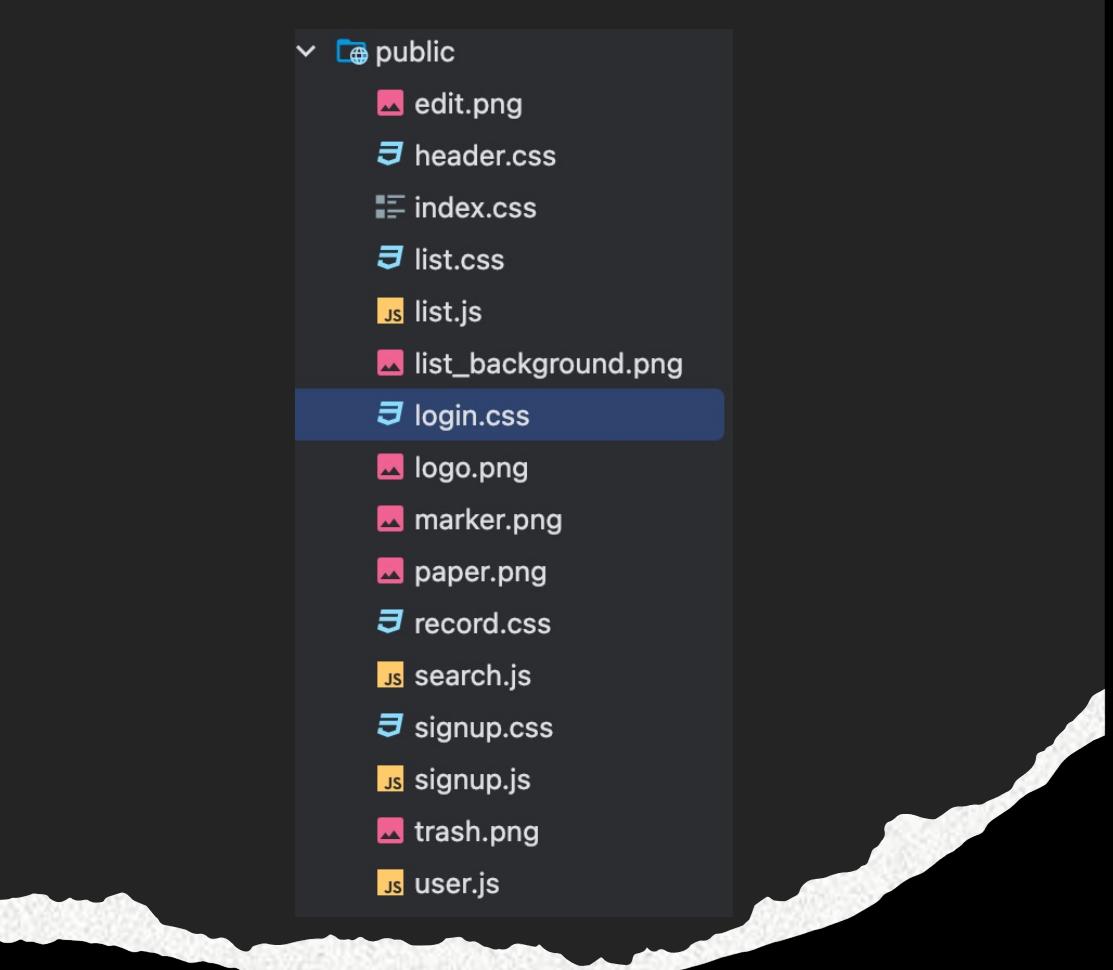


public

프론트(클라이언트) 레벨에서 사용할 js , css
함수를 모아둔 폴더입니다.

Kakao의 지도 API를 사용하여, 키워드를 통해
검색 결과를 관리합니다.

사진 파일들은 logo, 아이콘, 배경 등에 사용합
니다.



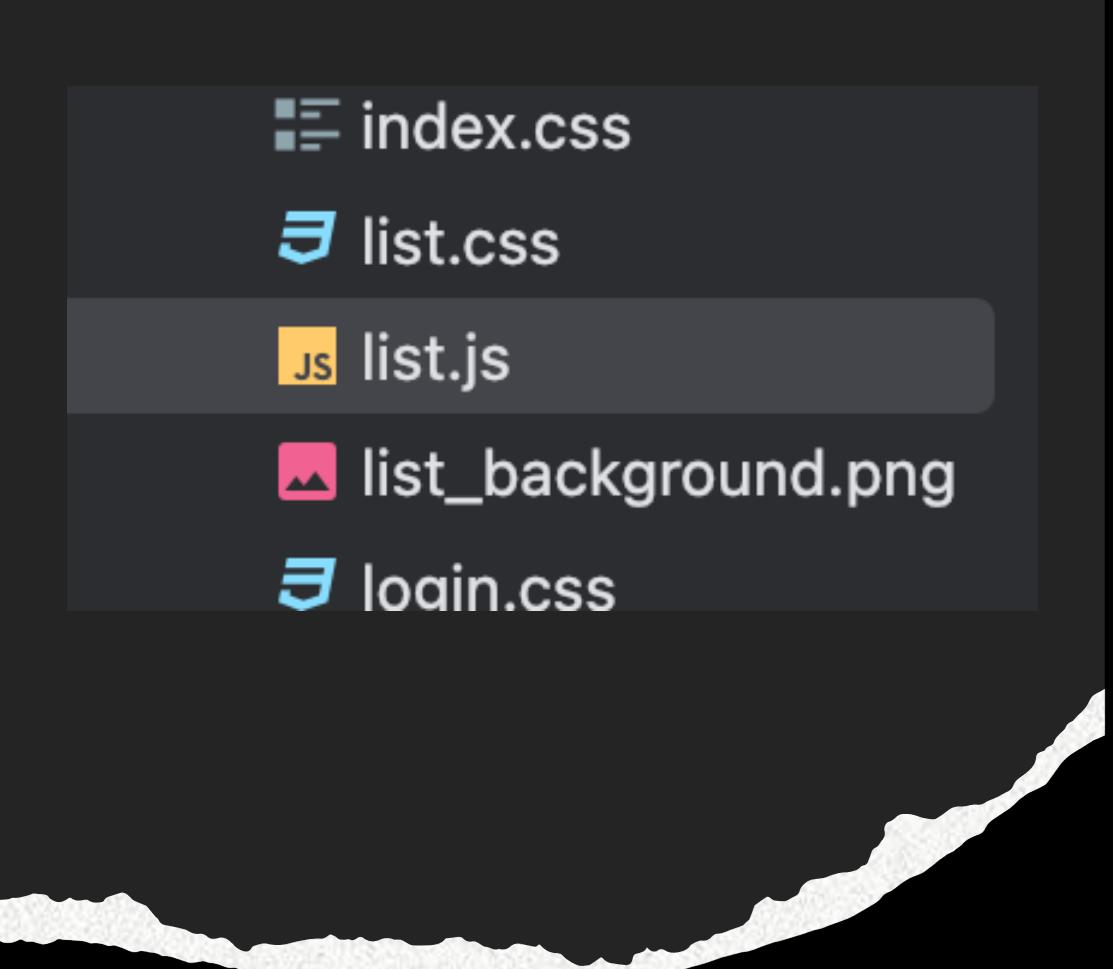
public

list.js

removeTravel(id)

- 호출 시 여행지를 삭제

- /user/list/:id 로 DELETE 요청을 보냅니다.



public

user.js

editName()

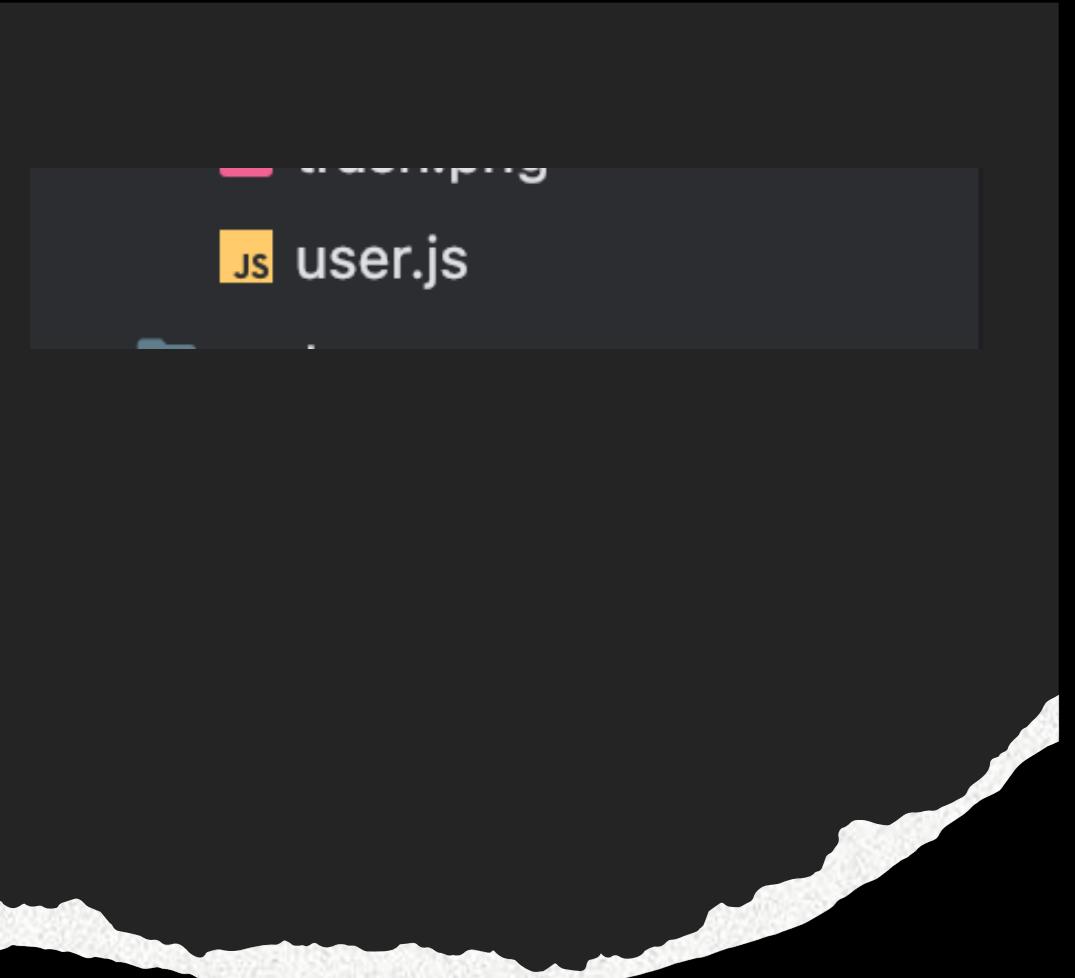
- 사용자의 이름을 업데이트
- /user/myinfo로 PUT 요청을 보내며, 새 이름을 전송합니다.

editPassword()

- 사용자의 비밀번호를 업데이트
- /user/myinfo로 PUT 요청을 보내며, 새 비밀번호를 전송합니다.

withdraw()

- 사용자가 회원탈퇴
- /user/withdraw DELETE 요청을 보냅니다.



public

search.js

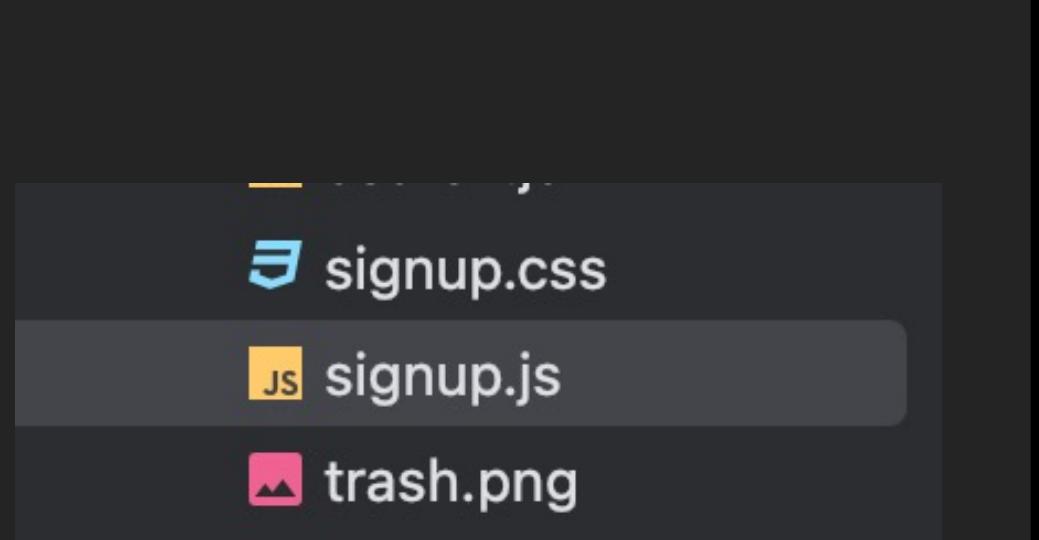
- Kakao 지도 API를 이용해 사용자가 입력한 키워드로 장소를 검색하고, 검색 결과를 목록과 지도에 표시합니다.
- 각 장소를 클릭하면 해당 장소의 이름과 주소를 표시하며, 선택한 장소를 서버로 전송하여 여행지 목록에 추가할 수 있습니다.
- marker와 infowindow 통해 지도에서도 장소를 시각적으로 확인할 수 있으며, 페이지네이션 기능을 통해 여러 페이지의 검색 결과를 관리합니다.



public

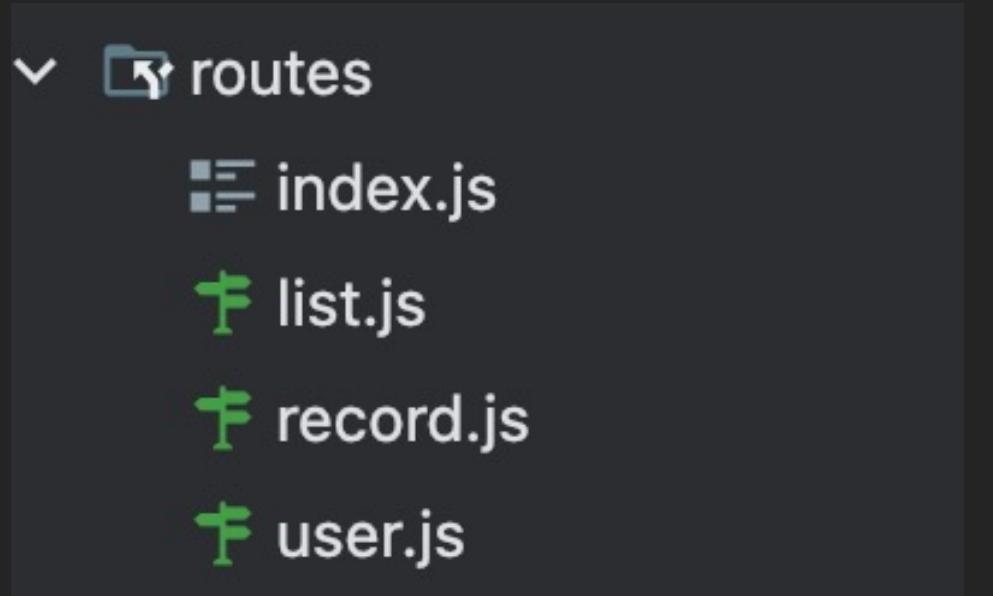
signup.js

- 사용자가 비밀번호와 비밀번호 확인 필드를 입력할 때 실시간으로 비교하여 일치 여부를 검사합니다.
- 비밀번호는 8자리 이상이어야 하며, 일치하지 않거나 조건을 충족하지 않을 경우 적절한 메시지를 출력하고 회원가입 버튼을 비활성화합니다.



routes

- end point를 각각 분리하여 routing하고,
서버에서 수행 해야하는 기능을 제공하는
폴더입니다.



routes

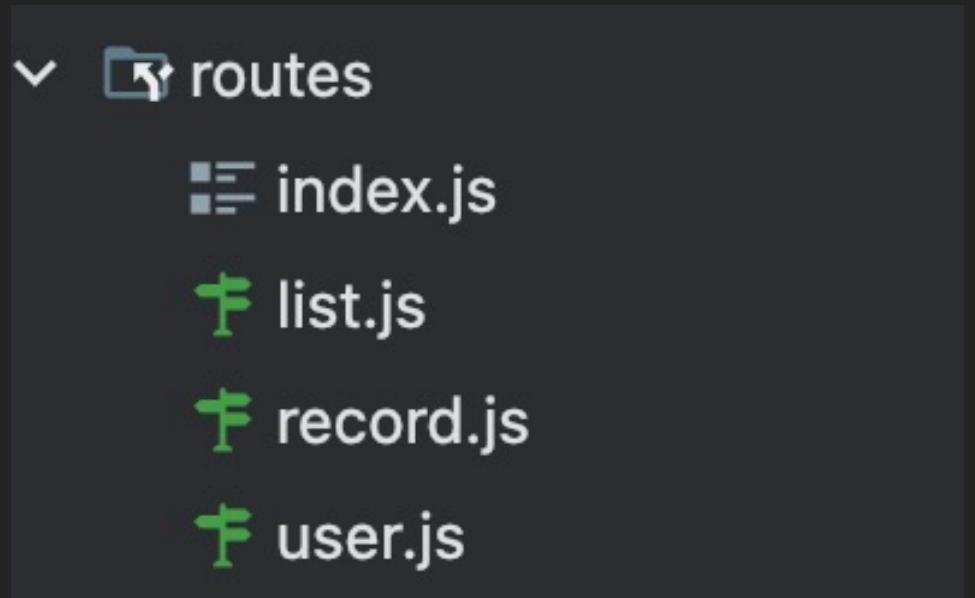
index.js

홈 페이지

- GET /: 홈 페이지를 렌더링합니다.
- 사용자 정보가 있으면 해당 정보를 렌더링합니다.

검색 페이지

- GET /search: 검색 페이지를 렌더링 합니다.
- 사용자 정보가 있으면 해당 정보를 렌더링합니다.



routes

list.js

여행지 리스트 페이지

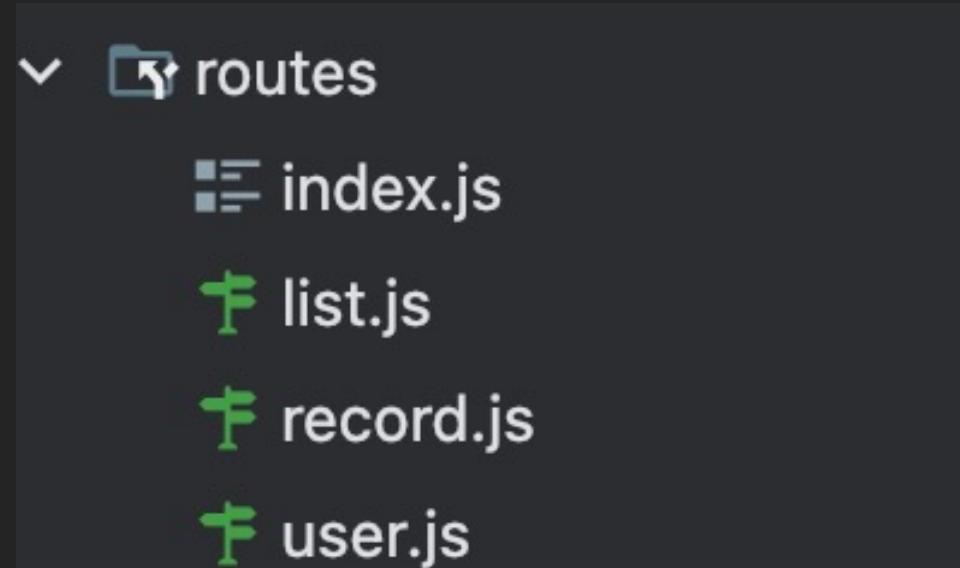
- GET /list: 사용자의 여행지 목록을 조회하여 리스트 페이지를 렌더링합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

여행지 추가

- POST /list: 여행지를 추가하고, 추가된 여행지를 리스트에 연결합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

여행지 삭제

- DELETE /list/:id: 특정 여행지를 삭제합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.



routes

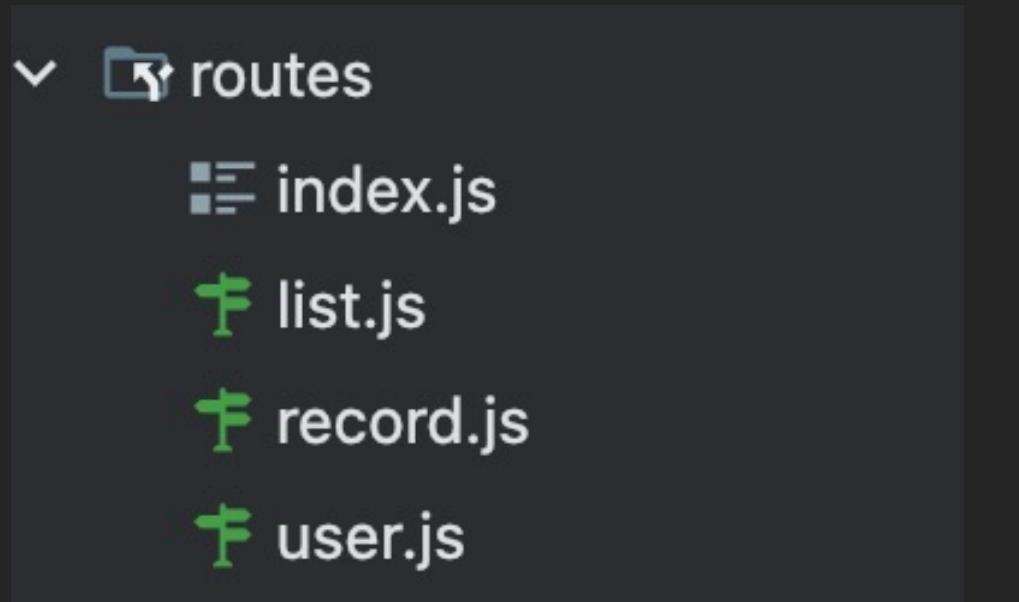
record.js

기록 작성 페이지로 이동

- GET /record/:travelId: 특정 여행지에 대한 기록 작성 페이지를 렌더링합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

기록 저장

- POST /record/:travelId: 특정 여행지에 대한 기록을 저장합니다. 내용 및 사진, 함께한 사람 등이 저장 또는 update 됩니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.



routes

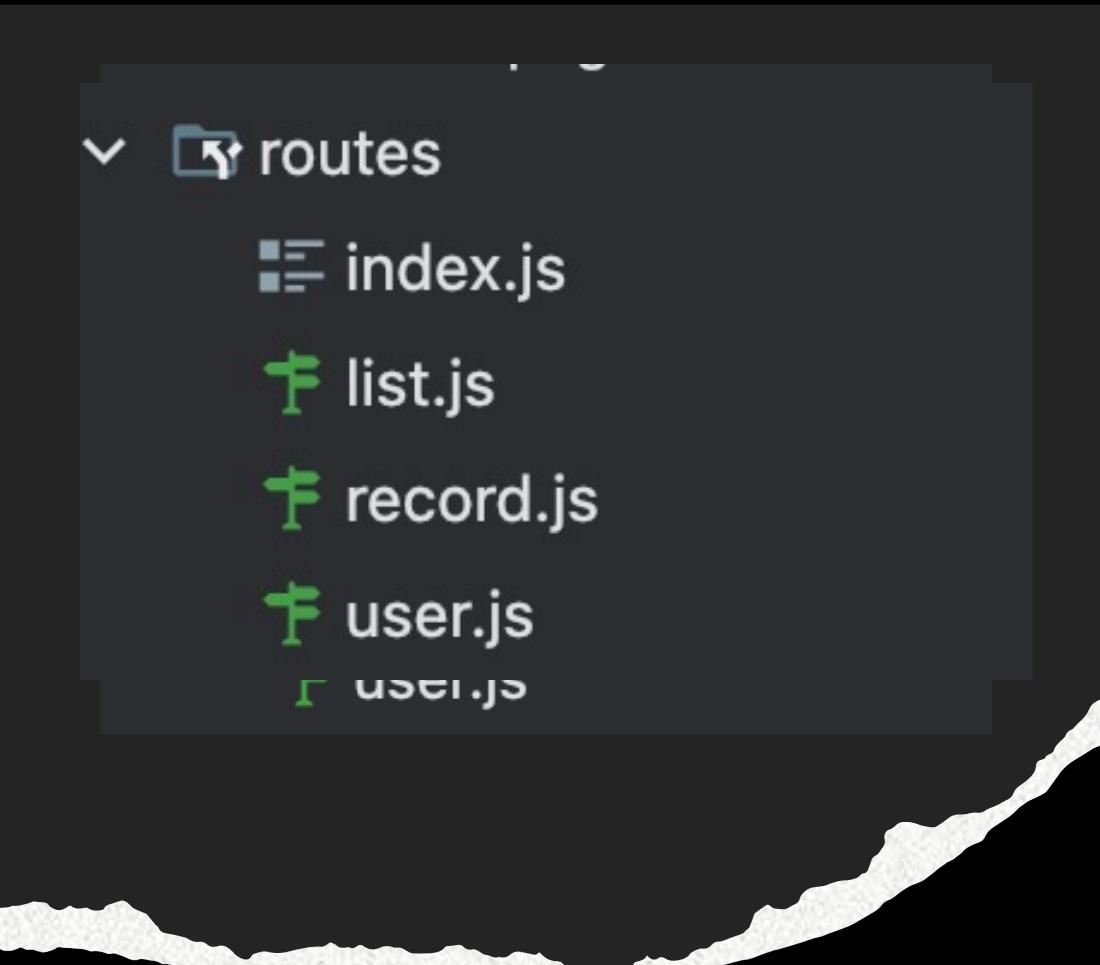
record.js

저장된 기록 페이지로 이동

- GET /record/:travelId/rewrite: 특정 여행지에 대한 저장된 기록 페이지를 렌더링합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

기록 편집 페이지로 이동

- GET /record/:travelId/edit: 특정 여행지에 대한 기록 편집 페이지를 렌더링합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.



routes

user.js

로그인

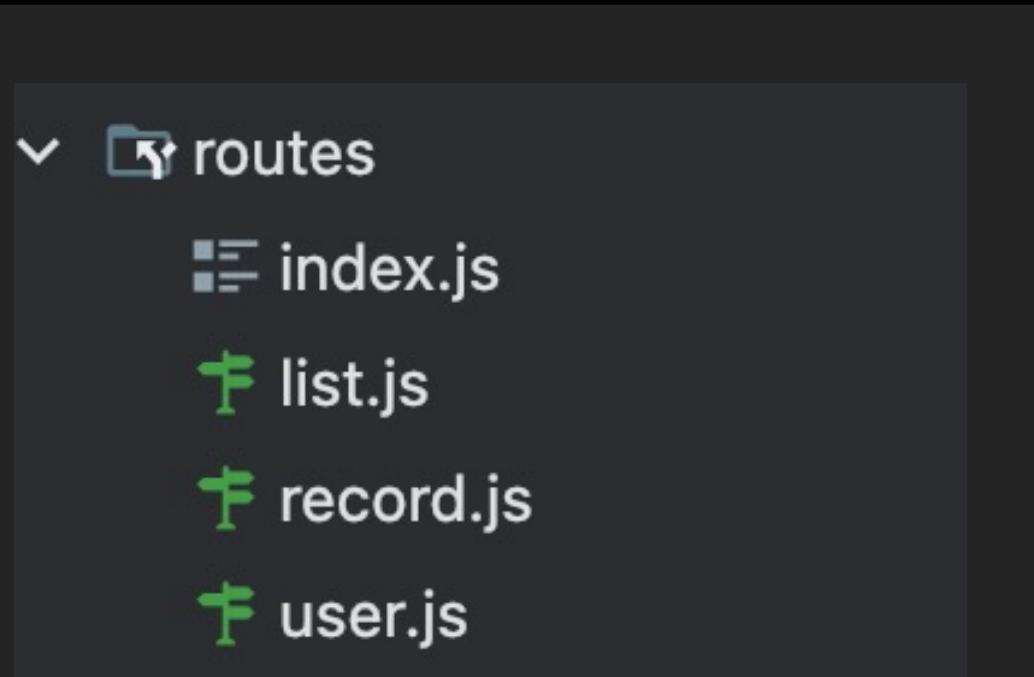
- GET /user/login: 로그인 페이지를 렌더링합니다.
- POST /user/login: 사용자의 로그인을 처리합니다.

카카오 로그인

- GET /user/login/kakao: 카카오 로그인을 위한 요청을 합니다.
- GET /user/login/kakao/cb: 카카오 로그인 콜백을 처리합니다.

로그아웃

- GET /user/logout: 로그아웃을 처리하고 세션을 파기합니다.
- isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

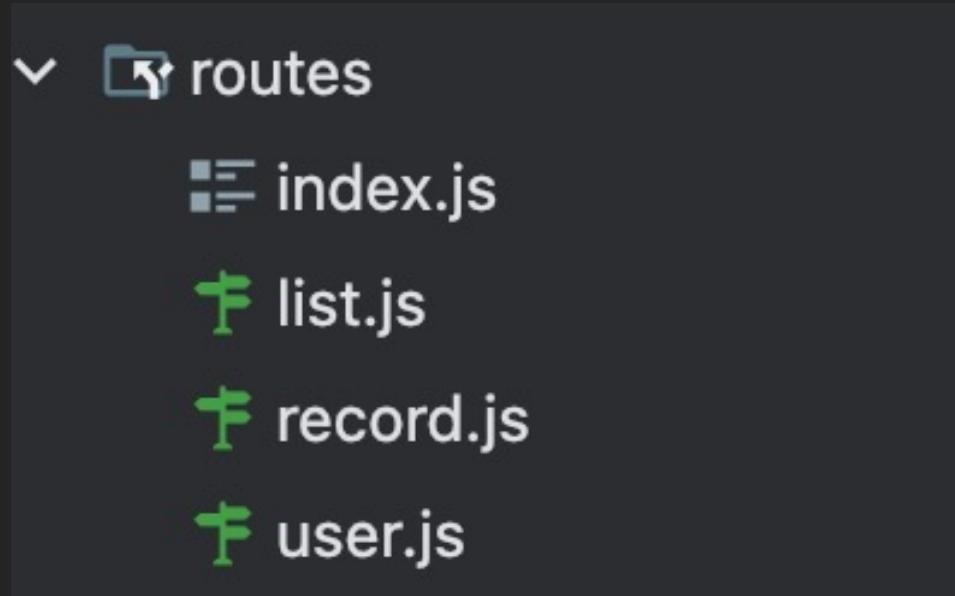


routes

user.js

회원가입

- GET /user/signup: 회원가입 페이지를 렌더링합니다.
- POST /user/signup: 사용자의 회원가입을 처리합니다.



routes

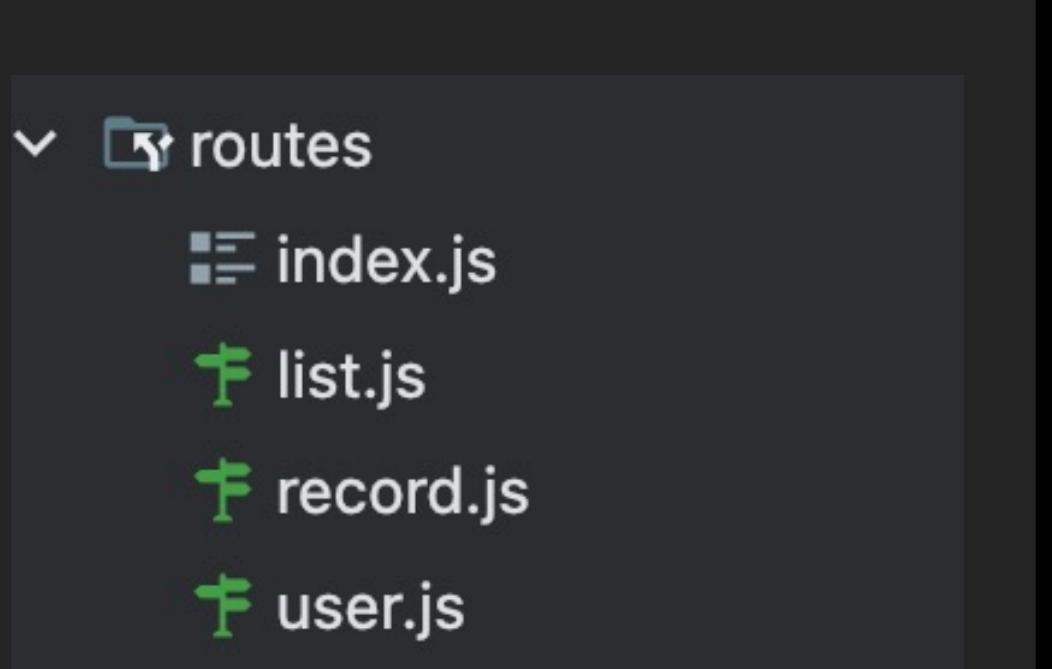
user.js

1. 내 정보

1. GET /user/myinfo: 사용자의 개인 정보 페이지를 렌더링합니다.
2. PUT /user/myinfo: 사용자의 개인 정보를 수정합니다.
3. isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.

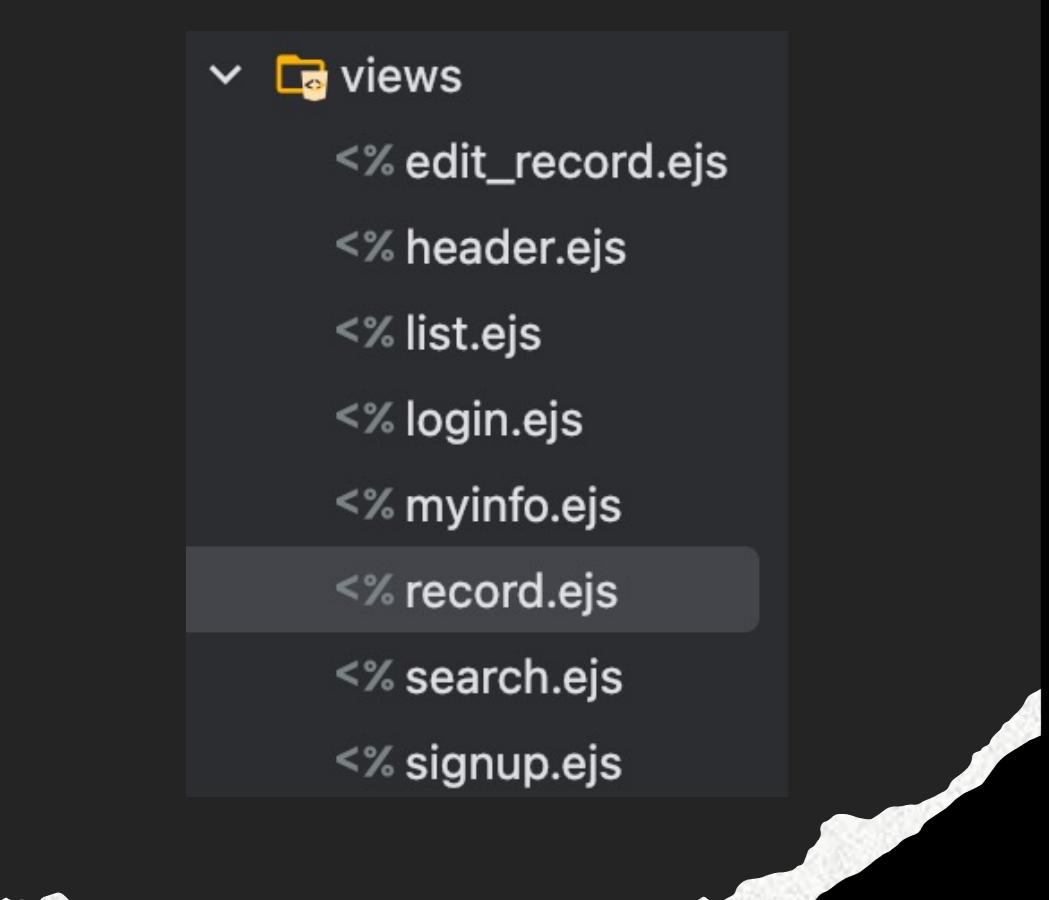
2. 회원탈퇴

1. DELETE /user/withdraw: 사용자의 회원 탈퇴를 처리합니다.
2. isLoggedIn 미들웨어로 인증을 확인한 후에만 접근할 수 있습니다.



views

- template-engine인 ejs를 이용해 각 페이지를 구현한 폴더입니다.



```
<% edit_record.ejs
<% header.ejs
<% list.ejs
<% login.ejs
<% myinfo.ejs
<% record.ejs
<% search.ejs
<% signup.ejs
```

views

- header.ejs: 웹 페이지 상단에 있는 로고와 네비게이션 메뉴를 포함한 헤더를 구성하는 템플릿입니다. 로그인 여부에 따라 다른 네비게이션 메뉴를 렌더링합니다.
- search.ejs: 여행지를 검색하고 결과를 표시하는 페이지입니다. 사용자가 로그인하지 않은 경우 기록 저장 기능을 사용할 수 없음을 안내하며, 검색된 여행지의 정보를 표시합니다.

▼  views

<% edit_record.ejs

<% header.ejs

<% list.ejs

<% login.ejs

<% myinfo.ejs

<% record.ejs

<% search.ejs

<% signup.ejs

views

- login.ejs: 사용자 로그인을 위한 페이지입니다. 이메일과 비밀번호를 입력하여 로그인할 수 있으며, 카카오 계정으로 로그인하는 옵션이 제공됩니다.
- signup.ejs: 새로운 사용자가 회원가입하는 페이지입니다. 닉네임, 이메일, 비밀번호를 입력하여 계정을 생성할 수 있습니다. 비밀번호 확인 기능도 포함되어 있습니다.

views

<% edit_record.ejs

<% header.ejs

<% list.ejs

<% login.ejs

<% myinfo.ejs

<% record.ejs

<% search.ejs

<% signup.ejs

views

- list.ejs: 사용자의 여행 기록을 역순으로 나열하여 보여주는 페이지입니다. 각 기록은 수정 및 삭제할 수 있는 버튼이 포함되어 있습니다.
- edit_record.ejs: 특정 여행 기록을 편집하는 페이지입니다. 기록의 제목과 생성/수정 일시를 표시하고, 내용을 수정할 수 있는 입력란이 제공됩니다. 또한 사진을 업로드하게 되면 새로 생성되는 경우 새로운 사진으로 저장되고, 기존 사진이 있는 경우 새로운 사진을 업로드합니다.

views

<% edit_record.ejs

<% header.ejs

<% list.ejs

<% login.ejs

<% myinfo.ejs

<% record.ejs

<% search.ejs

<% signup.ejs

views

- myinfo.ejs: 사용자의 개인 정보 및 설정을 관리하는 페이지입니다. 닉네임, 이메일을 표시하고 비밀번호 변경 및 회원 탈퇴 기능을 제공합니다.
- record.ejs: 기록의 생성일 및 수정일을 표시하며, 내용과 함께 누구와 함께했는지, 해당 기록의 사진을 확인할 수 있습니다.

views

<% edit_record.ejs

<% header.ejs

<% list.ejs

<% login.ejs

<% myinfo.ejs

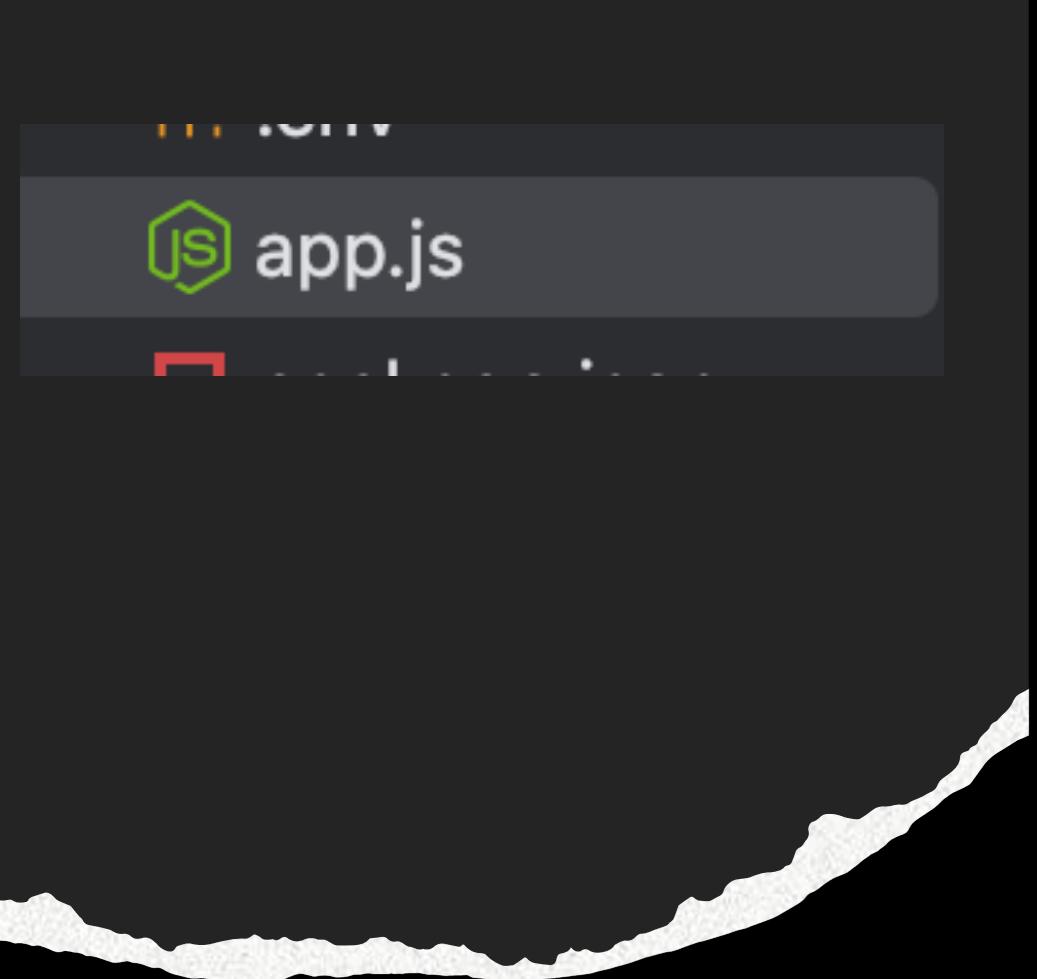
<% record.ejs

<% search.ejs

<% signup.ejs

app.js

- Sequelize를 통해 데이터베이스와 연동
- Passport를 활용하여 사용자 인증을 구현하며,
기본적인 미들웨어들로 method-override,
morgan, cookie-parser 등을 사용
- 정적 파일 제공과 HTTP 요청 처리를 위한 설정
도 포함
- 라우터는 별도 파일로 분리



주제, 핵심기능 및 추가
기능 설명



주제

- 여행을 다녀온 후, 여행지에서의 추억을 기록하고 싶은 때가 있습니다.
- 이럴 때, 여행지를 지도에서 검색하여 위치를 저장하고, 해당 위치에서의 추억을 글과 사진으로 간단하게 기록하는 서비스를 만들었습니다.



핵심기능 및 추가기능 설명

여행 기록

이메일
비밀번호

로그인
카카오 로그인

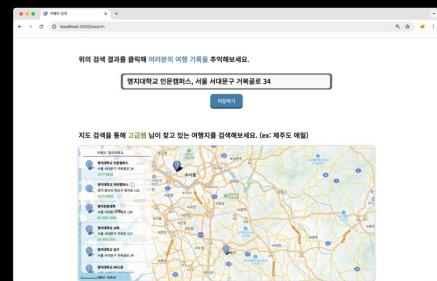
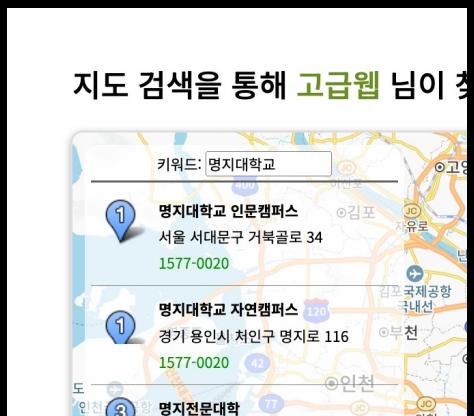
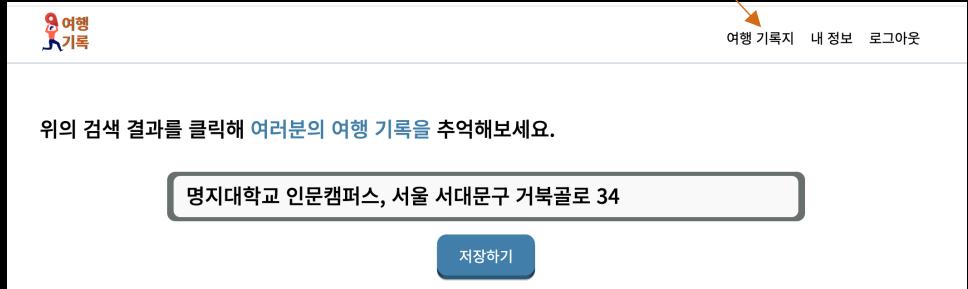
회원가입

이름
이메일
비밀번호
비밀번호 확인

가입하기

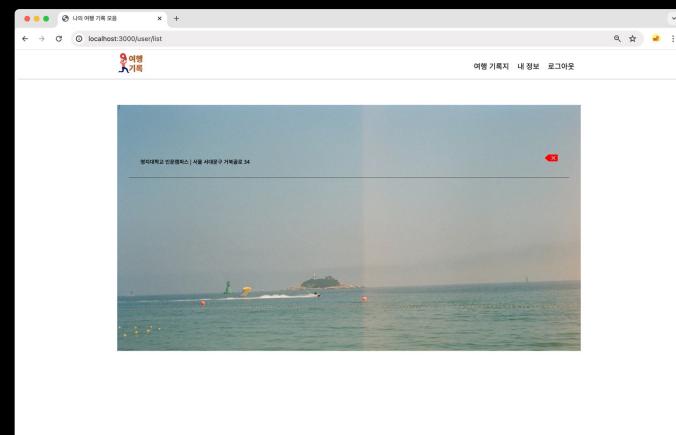
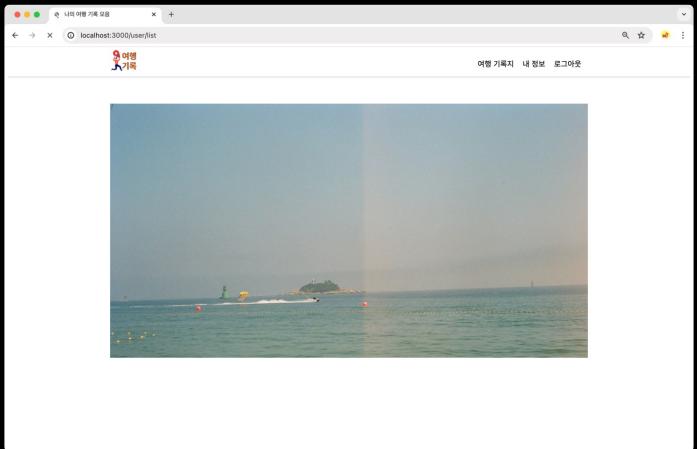
- 로그인
 - passport 모듈로 로그인 구현
 - local, 카카오톡 로그인
- 회원가입
 - 이름, 이메일, 비밀번호로 저장
 - 비밀번호 재확인 기능 추가
 - 비밀번호 8자리 이하인 경우 경고문 표시

핵심기능 및 추가기능 설명



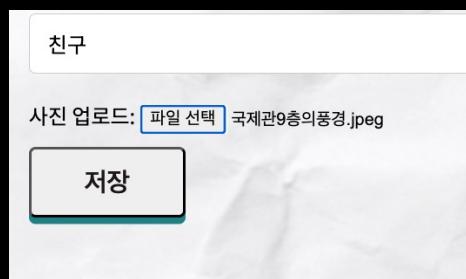
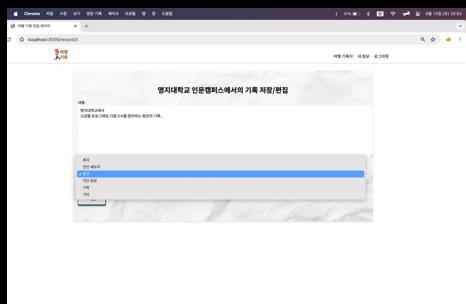
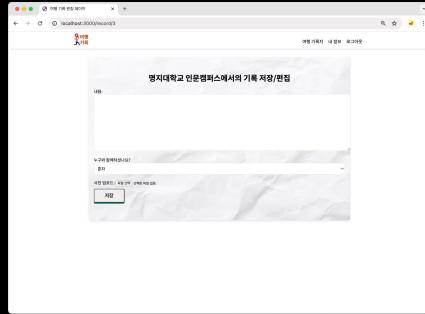
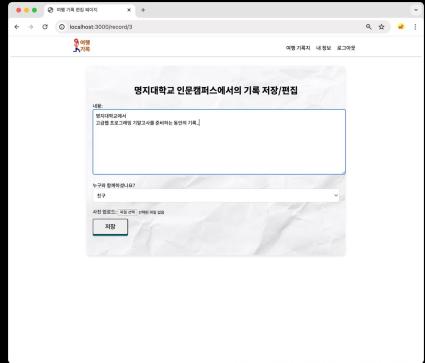
- 카카오 지도 API를 통해 키워드를 입력하면 해당 주소를 검색할 수 있도록 구현
- 클릭시 첫번째 사진처럼 해당 키워드로 검색된 위치명이 표시되고, 만약 주소를 포함한 장소라면 주소까지 표시
- 저장을 버튼을 누르면 여행 기록지 페이지에서 확인 가능

핵심기능 및 추가기능 설명



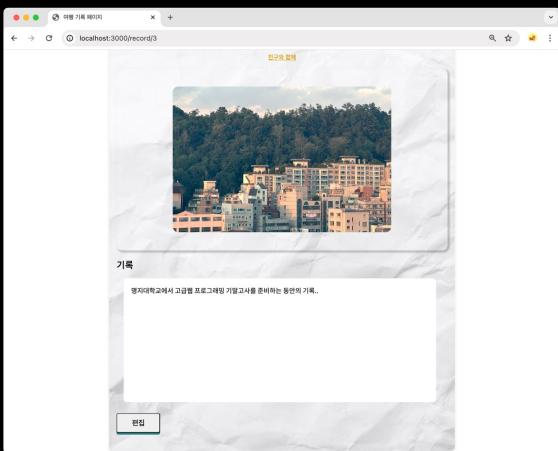
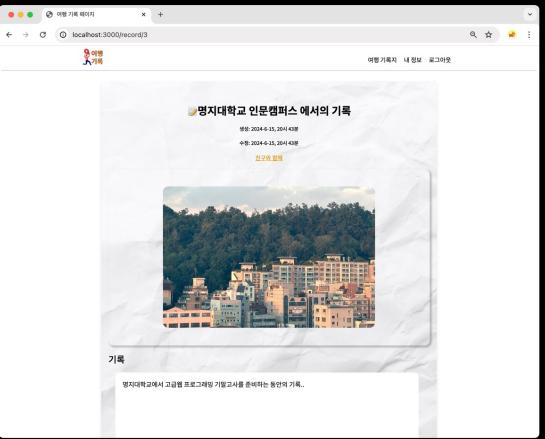
- 여행 기록지 페이지 (title: 나의 여행 기록 모음)의 초기 페이지와 앞선 ppt에서 저장을 누르게 되는 경우 표시되는 화면
- 목록에 저장된 위치를 클릭
- 삭제 기능 구현(앞선 기능 이후 설명)

핵심기능 및 추가기능 설명



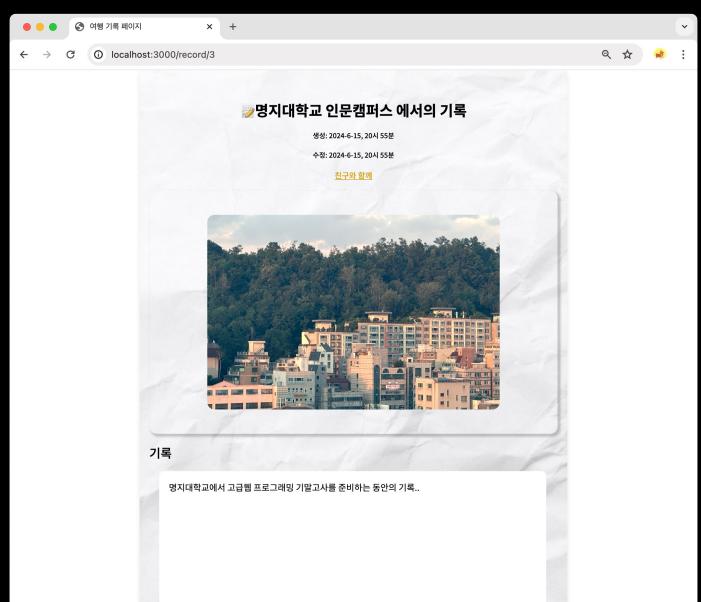
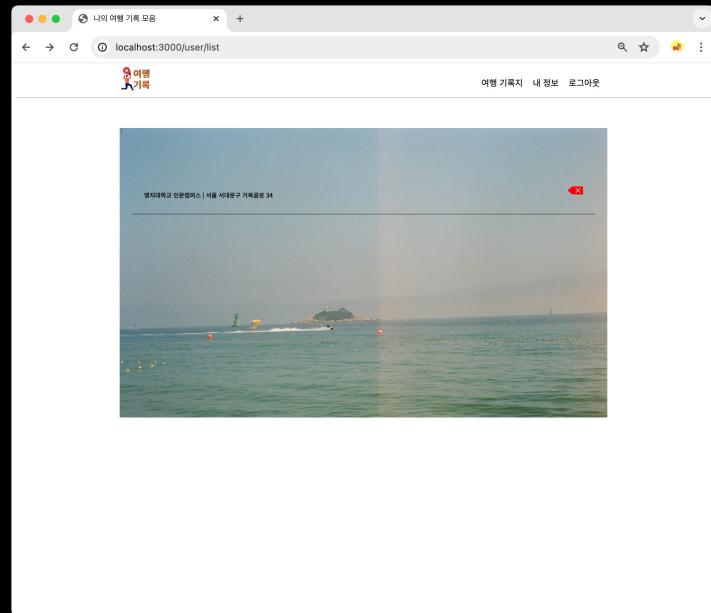
- 기록하고 싶은 내용을 기록
- 함께한 사람 카테고리 선택
- 해당 추억의 사진 업로드
- 저장 시 저장된 화면으로 전환

핵심기능 및 추가기능 설명



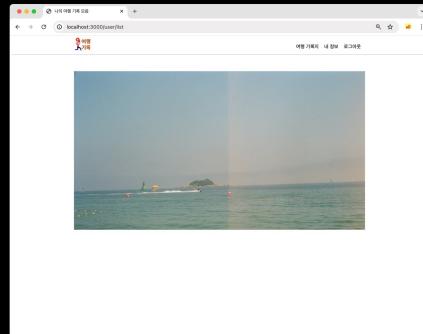
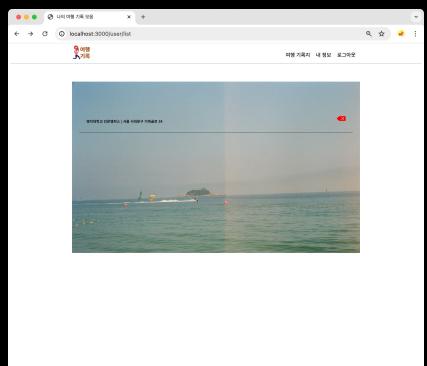
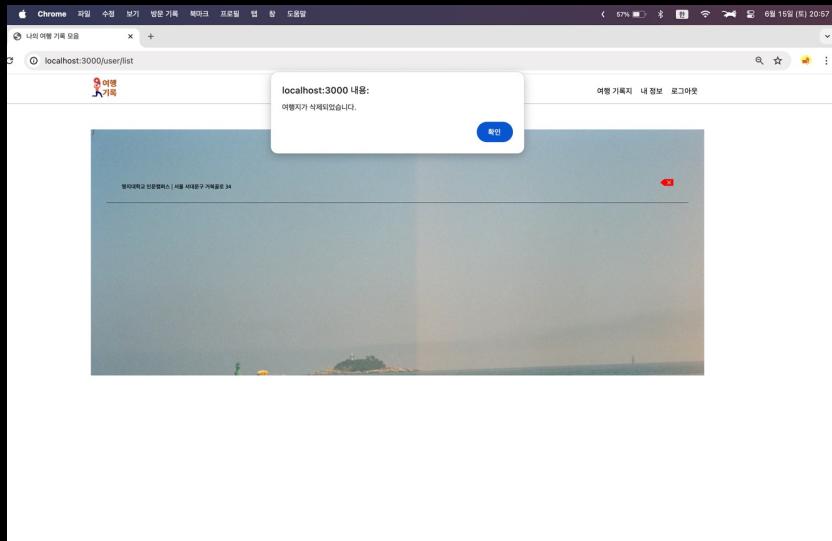
- 기록한 내용을 확인 (편집 불가능)
- 선택한 함께한 사람 표시
- 저장된 사진 적용
- 편집 버튼 클릭시 이전 편집 화면으로 전환

핵심기능 및 추가기능 설명



- 저장이 완료된 후 해당 위치 클릭 시
바로 저장된 화면으로 표시

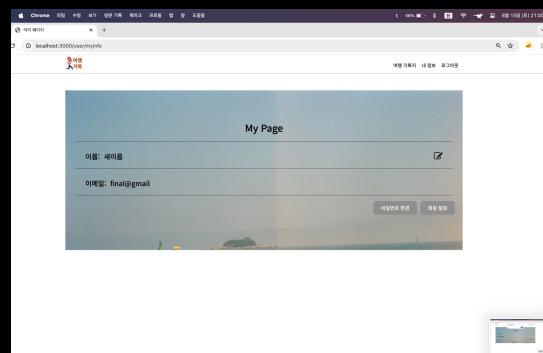
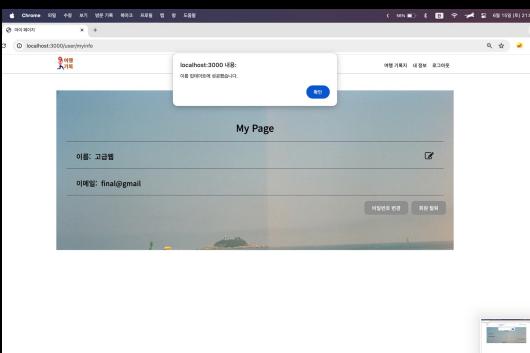
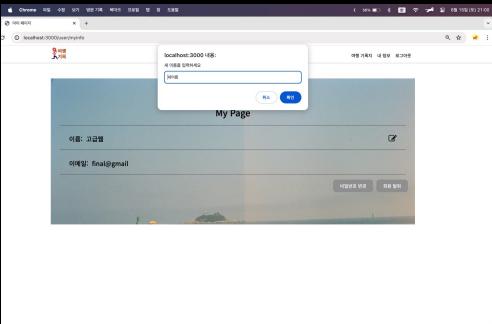
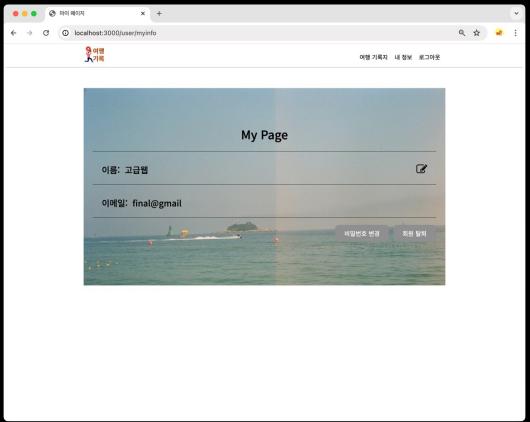
핵심기능 및 추가기능 설명



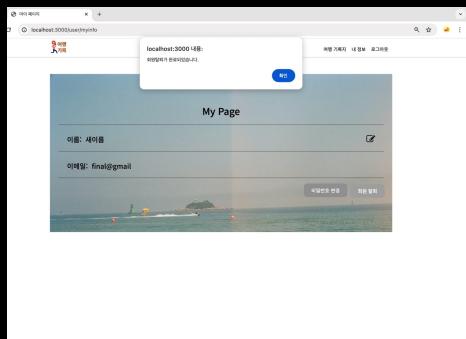
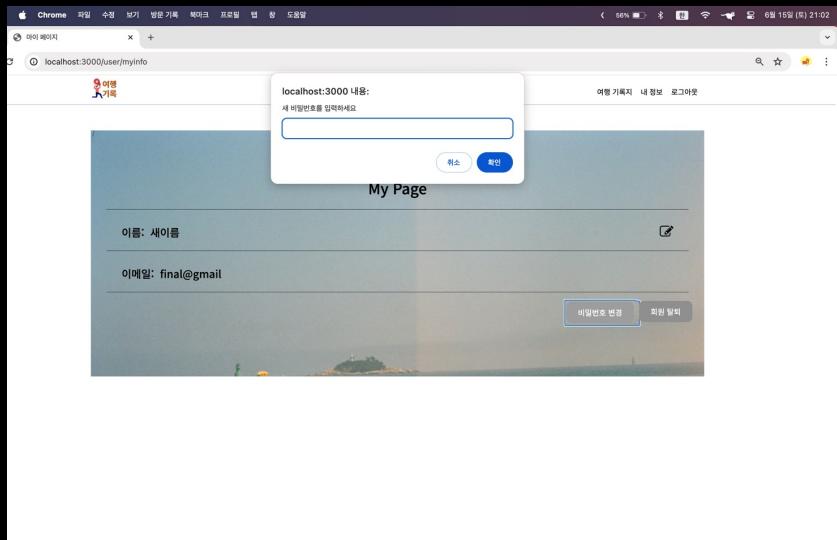
- 빨간색 삭제 버튼 클릭 시 삭제 완료 창 팝업
- 확인 버튼 누르면 redirect 로 reload되면서 업데이트된 화면 표시

핵심기능 및 추가기능 설명

- 이름 옆 편집 버튼 클릭 시 이름 편집 가능

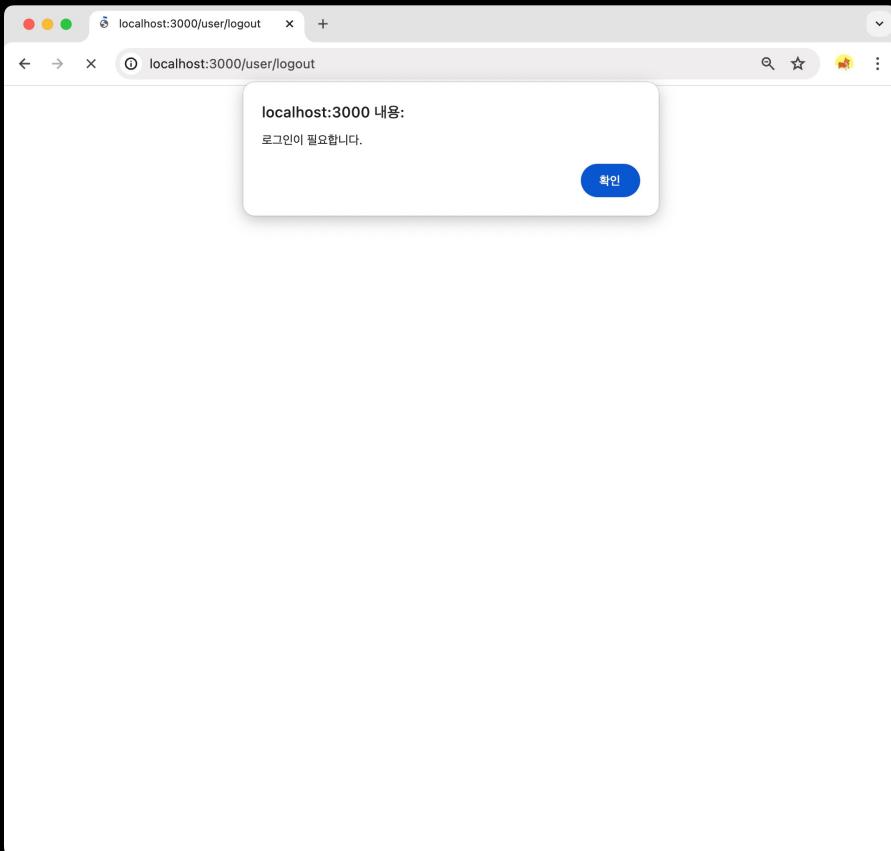


핵심기능 및 추가기능 설명



- 비밀번호 변경 버튼 클릭 시 비밀번호 변경 가능
- 회원탈퇴 버튼 클릭 시 해당 회원삭제
- 이 경우, 초기화면으로 redirect

핵심기능 및 추가기능 설명



- 로그인이 필요한 기능에서 로그인이
되어있지 않은 경우 팝업창으로 알림