**프로젝트 기술서**

| 프로젝트 명: STIRRING |

목차

[**1)** **프로젝트 개요** 3](#_Toc5378020)

[1.1 프로젝트 목적 3](#_Toc5378021)

[1.2 서비스 비전 3](#_Toc5378022)

[1.3 기술 스택 3](#_Toc5378023)

[1.4 테스트 서비스 환경 3](#_Toc5378024)

[**2)** **설계** 4](#_Toc5378025)

[2.1 데이터 오브젝트 4](#_Toc5378026)

[2.2 Simple Usecase 5](#_Toc5378027)

[2.3 페이지 구성 7](#_Toc5378028)

[2.4 기타 9](#_Toc5378029)

[**3)** **구현** 10](#_Toc5378030)

[3.1 프론트엔드 10](#_Toc5378031)

[3.2 백엔드 11](#_Toc5378032)

[3.3 데이터베이스 11](#_Toc5378033)

[**4)** **결과 및 상세 참조** 12](#_Toc5378034)

[4.1 프레젠테이션: 12](#_Toc5378035)

[4.2 GitHub: 12](#_Toc5378036)

[4.3 테스트 서비스 링크: 12](#_Toc5378037)

[4.4 프로젝트 기술서: 12](#_Toc5378038)

# **프로젝트 개요**

## 프로젝트 목적

1. 본 프로젝트는 웹 서비스에 대한 이해와 개발역량 향상을 목적으로 하는 개인 프로젝트이다.
2. Vue.js / Node.js 등 대표적인 웹 프레임워크를 이용하여 웹 서비스를 구현한다.

## 서비스 비전

1. STIRRING은 사진 데이터를 자유롭게 공유하는 웹 서비스이다.
2. 세상 속에서 순간을 담은 사진을 통해 신선함을 발견한다.
3. 개발 참조 모델: [Unsplash.com](https://unsplash.com), [Pixabay.com](https://pixabay.com)

## 기술 스택

|  |  |
| --- | --- |
| Front-End | Vue.js, Vuex, Vue-router, Vue-material, Axios |
| Back-End | Node.js, Express, JWT, Multer, NodeMailer, REST API |
| Database | MongoDB, Mongoose |

## 테스트 서비스 환경

1. 웹 서비스: AWS S3 Bucket
2. API Server: 개인PC Node.js Server
3. Database: 개인PC MongoDB Server

# **설계**

## 데이터 오브젝트

1. User: 사용자 정보

|  |  |  |
| --- | --- | --- |
| 속성 | 데이터 타입 | 설명 |
| \_id | string | User 구별을 위한 임의의 ID값, 자동생성, Unique |
| email | string(email) | 유저 이메일, 회원가입 시 입력, 로그인, Unique |
| password | string(password) | 유저 패스워드, 회원가입 시 입력, 로그인 |
| name | string | 유저 네임, 회원가입 시 입력 |
| image | string | 유저 프로필용 이미지, 파일 이름만 저장 |
| posts | [string] | 유저가 게시한 post 리스트 |
| followings | [string] | 유저가 follow하는 유저 수, 리스트 |
| followers | [string] | 유저를 follow하는 유저 수, 리스트 |
| collections | [string] | 유저가 만든 collection 리스트 |
| interests | [string] | 유저가 관심가지는 tag 리스트, 최대 5개 |
| likes | [string] | 유저가 like 표시한 post 리스트 |
| createdAt | date | 유저가 회원가입한 일시 |
| updatedAt | date | 유저가 정보를 수정한 일시 |
|  |  |  |

1. Post: 게시물 정보

|  |  |  |
| --- | --- | --- |
| 속성 | 데이터 타입 | 설명 |
| \_id | string | Post 구별을 위한 임의의 ID값, 자동생성, Unique |
| author | string(id), string(name) | 이 post를 작성한 User(id, name) 데이터 |
| data | data | 업로드된 사진(동영상)의 정보를 담은 객체 |
| tags | [string] | post가 담은 정보를 나타내는 tag, 최대 5개 |
| likes | number | post에 like 표시한 user의 수 |
| downloads | number | post의 data를 내려 받기 한 수 |
| views | number | post를 자세히 보기한 user의 수 |
| createdAt | date | post가 게시된 일시 |
| updatedAt | date | post가 수정된 일시 |
|  |  |  |

1. FileData: 게시된 사진의 정보(DB의 Collections으로 생성하지는 않음/ Post에 귀속)

|  |  |  |
| --- | --- | --- |
| 속성 | 데이터 타입 | 설명 |
| filename | string | 사진(동영상)이 업로드 될 때 주어지는 이름, 자동생성 |
| size | number | 사진(동영상)의 용량 |
| resolution | json | 사진의 해상도, width \* height |
| format | string | 사진(동영상)의 데이터 포맷, 확장자 |
| location | location | 사진(동영상)에 저장된 위치정보, 옵션 |
|  |  |  |

1. Tag: 게시물에 표시될 #태그, Type: Set(String)
2. Collection

|  |  |  |
| --- | --- | --- |
| 속성 | 데이터 타입 | 설명 |
| \_id | string | collection 구별을 위한 임의의 ID값, 자동생성, Unique |
| title | string | collection 제목 |
| posts | [string] | collection에 등록된 post 리스트 |
| author | { id: string, name: string } | collection을 생성한 user 정보 |
| createdAt | date | collection을 생성한 일시 |
| updatedAt | date | collection을 수정한 일시 |
|  |  |  |

## Simple Usecase

주황색 영역: 개발 완료, 빨간색 영역: 개발 중 취소된 부분, 노란색 영역: 개발하지 못한 부분

1. User

|  |  |  |
| --- | --- | --- |
| UC\_ID | UC\_Name | UC\_Description |
| U01 | 회원가입 | 회원가입을 할 수 있다. (local, google, facebook)  email: email 형식, 6~254, 중복체크, 가입 후 email 인증  password: 6~20자, 공백 불가, 비밀번호 확인  name: 2~20자, 특수문자 불가, 끝 공백 불가 |
| U02 | 로그인 | 로그인 할 수 있다. (local, google, facebook) |
|  | with Google |  |
|  | with Facebook |  |
| U03 | 로그아웃 | 로그아웃 |
| U04 | 비밀번호 찾기 | 이메일 인증 후 비밀번호 변경 2단계로 연결 |
| U05 | 비밀번호 변경 | 1단계: 이전 비밀번호 확인  2단계: 새 비밀번호 입력(비밀번호 확인 입력) |
| U06 | 회원 탈퇴 | 비밀번호로 인증 후 탈퇴  게시물 삭제, 컬렉션 삭제, 팔로우/팔로워 삭제 |
| U07 | 회원 정보 조회 | 회원정보 조회 |
| U08 | 프로필 이미지 등록 | 프로필 이미지 등록  미 등록 시 기본 이미지 |
| U09 | 프로필 이미지 삭제 |  |
| U10 | 프로필 이미지 수정 | 등록을 덮어쓰기는 방식으로 구현 |
| U11 | 팔로잉 | A follows B: A Folloings에 B 추가, B Followers에 A추가 |
| U12 | 언팔로잉 |  |
| U13 | 팔로잉 리스트 조회 |  |
| U14 | 팔로워 수 조회 |  |
| U15 | 관심태그 조회 |  |
| U16 | 관심태그 추가 | 최대 5개 |
| U17 | 관심태그 삭제 |  |
| U18 | 유저 검색 | 이름으로 검색  이메일로 검색 |
|  |  |  |

1. Post

|  |  |  |
| --- | --- | --- |
| UC\_ID | UC\_Name | UC\_Description |
| P01 | 게시하기 | data 등록 후 게시물 생성  Tags: 최대 5개 |
| P02 | 수정하기 | Tags 수정, 최대 5개 |
| P03 | 삭제하기 |  |
| P04 | 자세히보기 | views 증가 |
| P05 | 리스트보기 | 자세히보기로 연결 |
| P06 | 검색하기 | 태그로 검색  유저로 검색 |
| P07 | Author 연결 | 유저 정보 조회로 연결 |
| P08 | like | 토글, User like Post: user likes에 post 추가, post likes에 user 추가 |
| P09 | unlike | 토글, User like Post: user likes에 post 삭제, post likes에 user 삭제 |
| P10 | 파일 다운로드 | Public 링크로 연결, downloads 증가 |
| P11 | collection 추가 | 컬렉션 리스트 모달로 연결 |
| P12 | collection 삭제 | 컬렉션 리스트 모달로 연결 |
| P13 | views 증가 | 자세히 보기 시 증가 |
| P14 | 공유하기 | 링크 복사 |
| P15 |  |  |

1. Collection

|  |  |  |
| --- | --- | --- |
| UC\_ID | UC\_Name | UC\_Description |
| C01 | 리스트 보기 | 컬렉션 생성, 삭제 / 포스트 추가, 제거 로 연결 |
| C02 | 컬렉션 생성 | 타이틀: String, 최대 30자 |
| C03 | 컬렉션 삭제 |  |
| C04 | 포스트 추가 | 컬렉션에 추가, 컬렉션에서 삭제로 토글 |
| C05 | 포스트 삭제 | 컬렉션에서 삭제, 컬렉션에 추가로 토글 |
| C06 |  |  |

1. 기타 미분류

|  |  |  |
| --- | --- | --- |
| UC\_ID | UC\_Name | UC\_Description |
| E01 | 언어 변경 | 다국어 지원: English, 한국어 |
| E02 | 반응형 페이지 | 뷰 크기에 맞는 레이아웃 지원(Mobile, PC) |
| E03 | 인기 태그 갱신 | 검색 페이지의 인기태그가 자동으로 갱신되는 프로세스 |
| E04 | 더 보기 버튼 | 최초 로딩 이후 스크롤이 내려가면 버튼을 통해 추가 데이터를 로딩한다. |
| E05 | 히스토리 유지 | 뒤로가기 시 스크롤 및 데이터를 유지해야 한다. |
|  |  |  |

## 페이지 구성

1. HEADER
   1. 서비스 로고(메인 링크)
   2. 홈 버튼
   3. 검색 버튼 (게시물, 사용자, 컬렉션)
   4. 업로드 버튼
   5. 로그인 버튼 (비 로그인 시)
   6. 회원가입 버튼 (비 로그인 시)
   7. 로그아웃 버튼 (로그인 시)
   8. 마이페이지 버튼 (로그인 시)
   9. 언어 선택 링크
2. FOOTER
   1. 서비스 로고
   2. 저작권 관련
   3. 문의 이메일
   4. 서비스 소개 링크
   5. GitHub 링크
3. 메인
   1. 서비스 메인 (검색 링크, 서비스 소개 링크)
   2. 게시물 리스트
4. 검색
   1. 검색어 입력 부분
   2. 검색어 입력 가이드
   3. 인기 검색어
   4. 검색 결과(게시물, 사용자, 컬렉션)
5. 업로드
   1. 이미지 선택 영역
   2. 태그 입력 영역
   3. 오류 텍스트 영역
   4. 닫기, 제출 버튼
6. 게시물 자세히 보기
   1. 사용자(작가) 정보(링크)
   2. 좋아요 버튼
   3. 컬렉션 추가 버튼
   4. 다운로드 버튼
   5. 게시물 이미지 영역
   6. 게시물 정보 영역
   7. 게시물 삭제 버튼(작가가 로그인 된 사용자 일 때)
7. 컬렉션 자세히 보기
   1. 컬렉션 제목
   2. 컬렉션 제목 변경 버튼
   3. 사용자(작가) 정보(링크)
   4. 컬렉션 내 게시물 리스트
8. 컬렉션 추가 모달 창
   1. 내 컬렉션 리스트(토글 버튼: 추가, 제거)
   2. 새로운 컬렉션 만들기 버튼
   3. 모달 닫기 버튼
9. 사용자 정보 보기
   1. 사용자 프로필 이미지
   2. 사용자 이름
   3. 사용자 메뉴(본인일 때)
   4. 팔로우 버튼(팔로워 인원 수)
   5. 관심 태그
   6. 사용자의 게시물 리스트
   7. 사용자의 컬렉션 리스트
   8. 사용자가 좋아한 게시물 리스트
   9. 사용자가 팔로잉하는 다른 사용자 리스트
10. 사용자 메뉴
    1. 프로필 이미지 번경 -> 파일 선택
    2. 비밀번호 변경(링크)
    3. 관심 태그 변경(모달 창)
    4. 로그아웃
    5. 회원탈퇴(링크)
11. 비밀번호 변경
    1. 현재 비밀번호
    2. 새 비밀번호
    3. 변경 버튼 -> 현재 비밀번호 검증 후 변경
12. 회원탈퇴
    1. 이메일 입력
    2. 비밀번호 입력
    3. 계정 삭제 버튼 -> 로그인 된 이메일, 비밀번호 비교 후 삭제
13. 로그인
    1. 이메일 입력, 비밀번호 입력
    2. 로그인 버튼
    3. 회원가입 버튼(링크)
    4. 비밀번호 찾기 버튼(링크) -> 이메일 입력 -> 이메일 인증 -> 새 비밀번호 입력
14. 회원가입
    1. 이름, 이메일, 비밀번호 입력
    2. 회원가입 버튼 -> 이메일 중복 체크 후 회원가입 승인

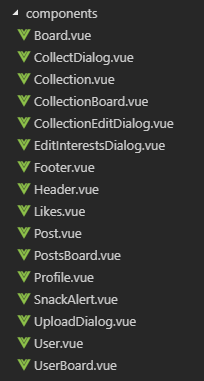
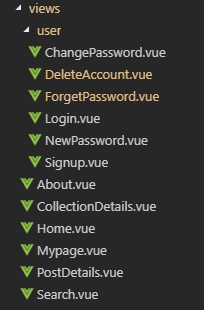
## 기타

1. 다국어 처리: 한국어, 영어 등 각 언어 변경이 가능해야 한다.
2. 반응형 웹 페이지: Mobile, Tablet, Desktop 등 여러 환경에 적절한 UI/UX를 제공해야 한다.
3. 이메일 인증: 비밀번호 찾기를 위해 이메일로의 확인 메일을 전송해야 한다.

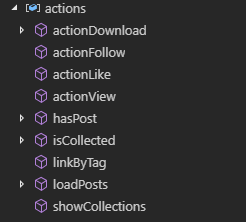
# **구현**

## 프론트엔드

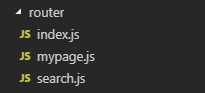
1. Vue.js를 이용하여 각 페이지 및 컴포넌트 단위로 구현

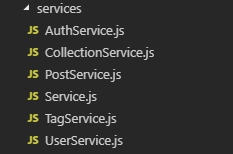
1. 각 컴포넌트 별 동일한 액션은 Mixins를 이용해 각 컴포넌트에 혼합되도록 구현



1. Vue-router를 이용하여 각 페이지를 바인딩하고, 라우팅 구현



1. Vuex를 이용하여 서비스 전체에 필요한 상태, 데이터를 관리하도록 구현
   1. **State**: userInfo, language, count
   2. **Getters**: isLogined, isMe, isLike, isFollow
   3. **Mutations**: login, logout, actionLike, actionUnlike, actionFollow, actionUnfollow, addCount, resetCount
   4. **Actions**: login, logout, actionLike, actionUnlike, createCollection, actionFollow, actionUnfollow
2. Html, CSS, Vue-material을 이용해 간단하고 모던한 UI를 구현했다.
3. Axios를 이용하여 API 서버와 통신하였다.



## 백엔드

1. Node.js, Express를 이용해 웹서버 및 라우팅 구현

|  |  |  |  |
| --- | --- | --- | --- |
| Group | Method | EndPoint | Description |
| Root | GET | api/ | API 동작여부를 체크 |
| Auths | POST | api/auths/login | 로그인 정보를 생성 |
| GET | api/auths/refresh | 토큰 인증 정보 갱신 |
| api/auths/check | 토큰 인증 정보 체크 |
| Users | POST | api/users/ | 새 사용자 생성 |
| api/users/password | 새 비밀번호 등록 |
| GET | api/users/:id | id로 사용자 정보 조회 |
| api/users/ | 모든 사용자 정보 조회 |
| api/users/profile-image/:id | id에 해당하는 사용자 이미지 조회 |
| api/users/email/:email | email 등록 여부 조회(비밀번호 찾기) |
| PUT | api/users/interests/:userId | userId에 해당하는 사용자 관심 변경 |
| api/users/:id | id에 해당하는 사용자 정보 변경 |
| api/users/profile-image/:id | id에 해당하는 사용자 이미지 변경 |
| api/users/password/:id | id에 해당하는 사용자 비밀번호 변경 |
| DELETE | api/users/:email | email에 해당하는 사용자 삭제 |
| Posts | POST | api/posts/ | 게시물 생성 |
| GET | api/posts/ | 모든 게시물 조회 |
| api/posts/:id | id에 해당하는 게시물 조회 |
| PUT | api/posts/:id | id에 해당하는 게시물 변경 |
| DELETE | api/posts/:id | id에 해당하는 게시물 삭제 |
| Collections | POST | api/collections/ | 컬렉션 생성 |
| api/collections/posts | 컬렉션에 게시물 등록 |
| GET | api/collections/ | 모든 컬렉션 조회 |
| api/collections/:ids | ids에 해당하는 컬렉션 조회 |
| PUT | api/collections/:id | id에 해당하는 컬렉션 정보 변경 |
| DELETE | api/collections/posts | 컬렉션에서 게시물 삭제 |
| api/collections/:id | id에 해당하는 컬렉션 삭제 |
| Tags | POST | api/tags/:word | word에 해당하는 태그 등록 |
| GET | api/tags/top5 | 인기 태그 조회 |

1. Multer: 업로드되는 이미지를 구별된 파일 이름을 부여하여 저장한다..
2. Mongoose: MongoDB와의 데이터 매핑, 동작을 구현한다.
3. JWT: 로그인 된 사용자의 인증을 Token을 발행하여 관리한다.
4. NodeMailer: 비밀번호 찾기 기능을 위해 이메일 전송을 담당한다.

## 데이터베이스

1. User
   1. Schema

|  |  |
| --- | --- |
| KEY | VALUE |
| email | { type: String, required: true, unique: true, lowercase: true } |
| password | { type: String, required: true, trim: true } |
| name | { type: String, required: true } |
| image | { type: String, default: 'face-default.png' } |
| posts | [String] |
| followings | [String] |
| followers | [String] |
| collections | [String] |
| interests | { type: [String], uppercase: true } |
| likes | [String] |
| option | { timestamps: true} |

* 1. Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Arguments | Return | Description |
| addPost | id, post, cb |  | 사용자의 posts에 post.\_id 추가 |
| create | user | user | user의 정보로 새 사용자 저장 |
| createCollection | collection | user | 사용자의 collections에 collection 추가 |
| deleteCollection | collection | user | 사용자의 collections에서 collection 삭제 |
| deleteOneByEmail | email | user | email에 해당하는 사용자 삭제 |
| deletePost | post | user | 사용자의 posts에서 post 삭제 |
| findAll | count | users | count만큼의 유저 정보 조회 |
| findById | ids, count | users | count만큼의 ids에 해당하는 사용자 조회 |
| findByWord | word, count | users | count만큼의 word(name, email, interests)에 해당하는 사용자 조회 |
| findOneByEmail | email | user | email에 해당하는 사용자 조회 |
| findOneByIdWithPW | id | user | id에 해당하는 사용자 조회(비밀번호 포함) |
| findOneProfileImageById | id | user | id에 해당하는 사용자 이미지 조회 |
| subtractFF | followings, followers,  accountId | promise | followings, followers에 해당하는 사용자에서 accountId 삭제 |
| update | id, user | user | id에 해당하는 사용자 정보 변경 |
| updateInterests | id, interests | user | id에 해당하는 사용자 관심 변경 |
| updatePassword | id, newPassword | user | id에 해당하는 사용자 비밀번호 변경 |
| updatePasswordByEmail | {email, newPassword} | user | email에 해당하는 사용자 비밀번호 변경 |
| updateProfileImageById | id, filename | user | id에 해당하는 사용자 이미지 번경 |
| verify | password | Boolean | password 일치 여부 확인 |

1. Post
   1. Schema

|  |  |
| --- | --- |
| KEY | VALUE |
| author | { id: { type: String, required: true },  name: { type: String, required: true } } |
| data | { filename: { type: String, required: true },  size: Number,  resolution: { width: Number, height: Number },  format: String,  location: String } |
| tags | { type: [String], uppercase: true } |
| likes | { type: Number, default: 0 } |
| downloads | { type: Number, default: 0 } |
| views | { type: Number, default: 0 } |
| option | { timestamps: true} |

* 1. Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Arguments | Return | Description |
| create | post | post | post 정보로 새 게시물 저장 |
| decLikes | likes: post.\_id |  | likes에 해당하는 게시물 likes 1 감소 |
| deleteByAuthorId | authorId |  | authorId가 일치하는 게시물 삭제 |
| deleteOne | id | post | id에 해당하는 게시물 삭제 |
| findAll | condition | posts | condition에 맞는 게시물 조회(작가, 태그 등) |
| findByAuthorId | authorId | posts | authorId가 일치하는 게시물 조회 |
| findById | id | posts | id에 해당하는 게시물 조회 |
| findOndById | id | post | id에 해당하는 게시물 조회 |
| updateOneById | id, post | post | id에 해당하는 게시물 변경 |

1. Collection
   1. Schema

|  |  |
| --- | --- |
| KEY | VALUE |
| title | { type: String, required: true } |
| posts | { type: [String], default: [] } |
| author | { id: { type: String, required: true },  name: { type: String, required: true } } |
| tags | { type: [String], default: [] } |
| option | { timestamps: true } |

* 1. Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Arguments | Return | Description |
| create | title, author | collection | 새 컬렉션 저장 |
| deleteCollections | collections |  | collections에 해당하는 컬렉션 삭제 |
| deleteOne | collection | collection | collection에 해당하는 컬렉션 삭제 |
| findAll | count | collections | count만큼의 컬렉션 조회 |
| findByAuthor | authorId, count | collections | authorId가 일치하는 컬렉션 조회 |
| findByIds | ids, count | collections | ids에 해당하는 컬렌션 조회 |
| findByWords | word, count | collections | word(title, aurhor.name)가 일치하는 컬렉션 조회 |
| findOneAddPost | collection, post | collection | 컬렉션에 post 추가 |
| findOneSubtractPost | collection, post | collection | 컬렉션에 post 삭제 |
| updateOne | collection | collection | 컬렉션 정보 변경 |

1. Tag
   1. Schema

|  |  |
| --- | --- |
| KEY | VALUE |
| word | { type: String, required: true, uppercase: true } |
| option | { capped: { max: 100000, size: 2 \* 20 \* 100000 } }  (최근 검색된 태그를 개수 제한하여 저장) |

* 1. Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Arguments | Return | Description |
| push | word | tag | 검색된 새 태그 저장 |
| getAll |  | tags | 모든 태그 조회 |

# **결과 및 상세 참조**

## 프레젠테이션:

* [https://github.com/Mounc/Stirring/Project\_Presentation(STIRRING).pptx](https://github.com/Mounc/Stirring/blob/master/Project_Presentation(STIRRING).pptx?raw=true)

## GitHub:

* <https://github.com/Mounc/Stirring>

## 테스트 서비스 링크:

* <http://stirring.s3-website.ap-northeast-2.amazonaws.com/>

## 프로젝트 기술서:

* [본 문서](#_top)