

WEATHER WEAR

임종혁 이동연 이지연



Table of Contents

- 01. 프로젝트 소개
- 02. 분석 / 설계
- 03. 서비스 시연
- 04. 기술소개
- 05. 후기

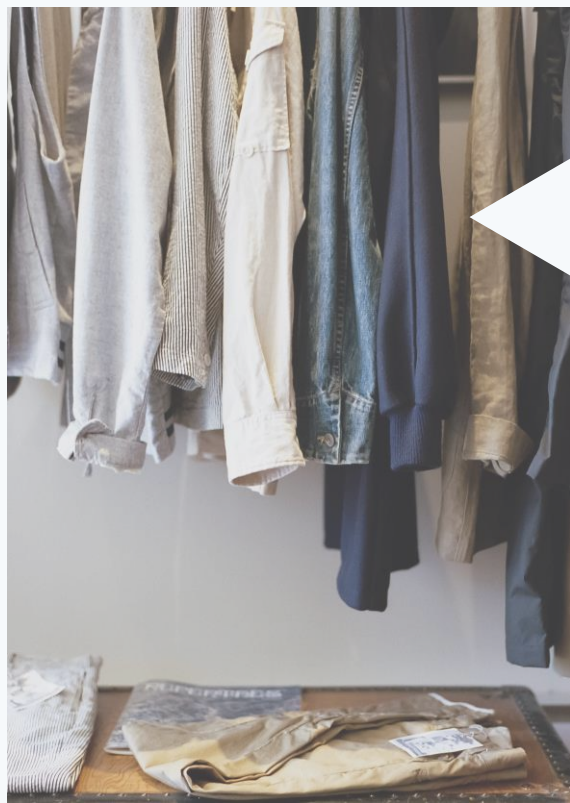
프로젝트 소개





매일 아침, 무슨 옷을 입어야 할지 고민될 땐

WEATHER WEAR



날씨 맞춤 자동 코디 추천

외투를 입어야 할까? 반팔 하나만 입기엔 춥지 않을까?
WEATHER WEAR가 현재 날씨에 맞는 코디를
자동으로 추천해드립니다

내 취향에 맞는 맞춤형 코디

회원 가입 단계에서 마음에 드는 옷을 골라주세요
날씨에 따라 취향 저격 맞춤형 코디를 제공해드립니다

사람들과 소통하는 패션 커뮤니티

패션에 관심이 많은 사람들과 자유롭게 소통해보세요
나의 코디를 공유하고 다른 사람의 OOTD도 살펴보세요

역할

종혁

- 시큐리티 JWT 적용
- RESTapi 구성
- 게시판 에디터 적용
- Vue.js 프레임워크 적용

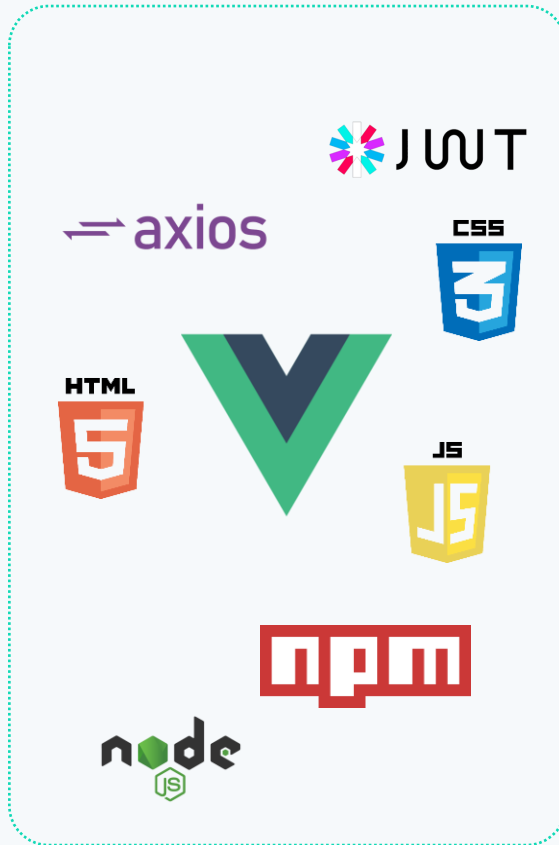
지연

- 게시판 기능 구현
- 날씨 api 적용
- 관리자 대시보드 기능 구현
- 이미지 업로드
- Vue.js 프레임워크 적용

동연

- 코디 이미지 특성 예측 모델 개발
- 이미지 특성 예측 Api 개발
- 사용자 별 코디 추천 Api 개발

개발환경



VIEW

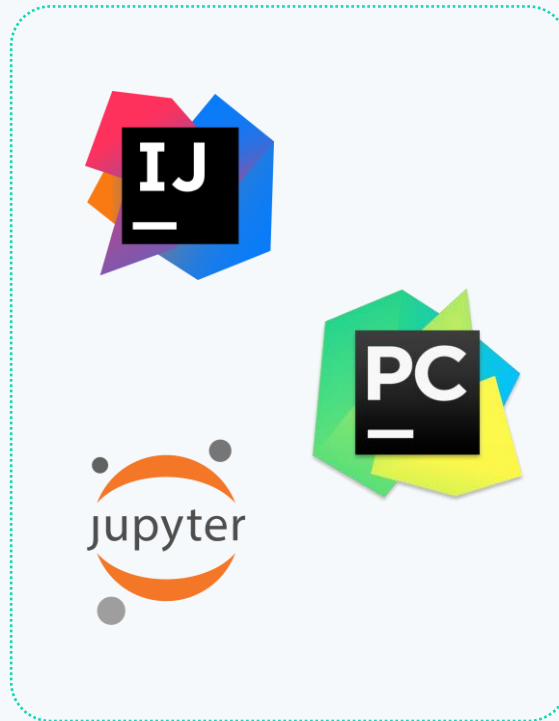


CONTROLLER

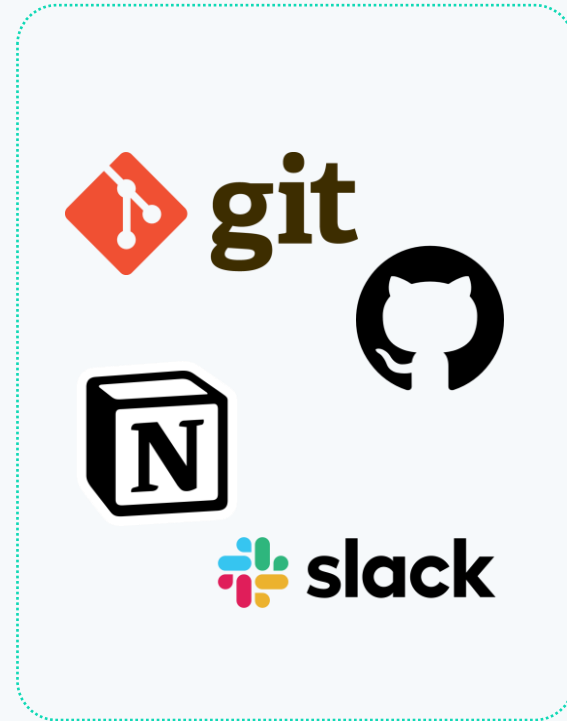


DB

개발환경



IDE



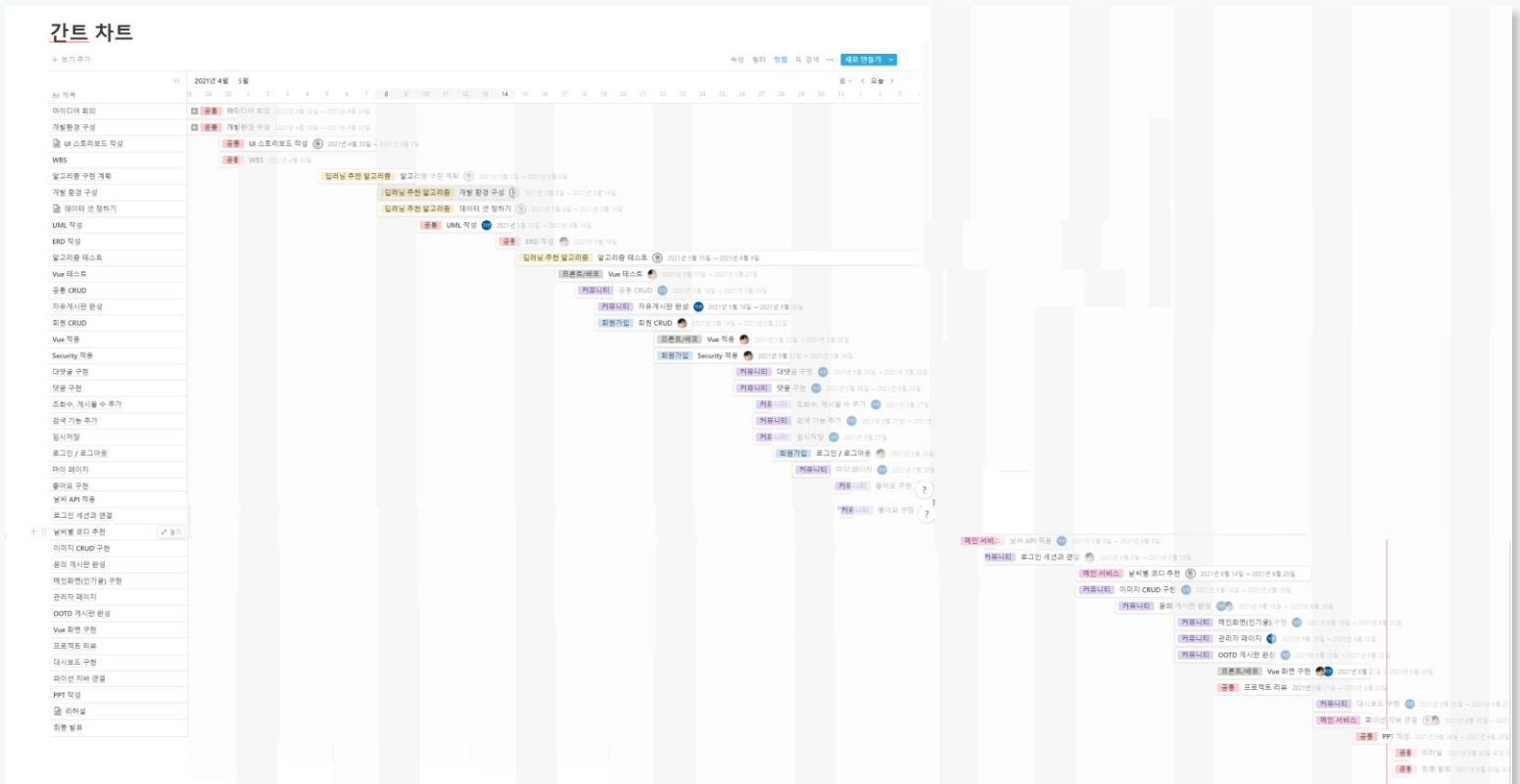
협업 TOOL

분석 설계

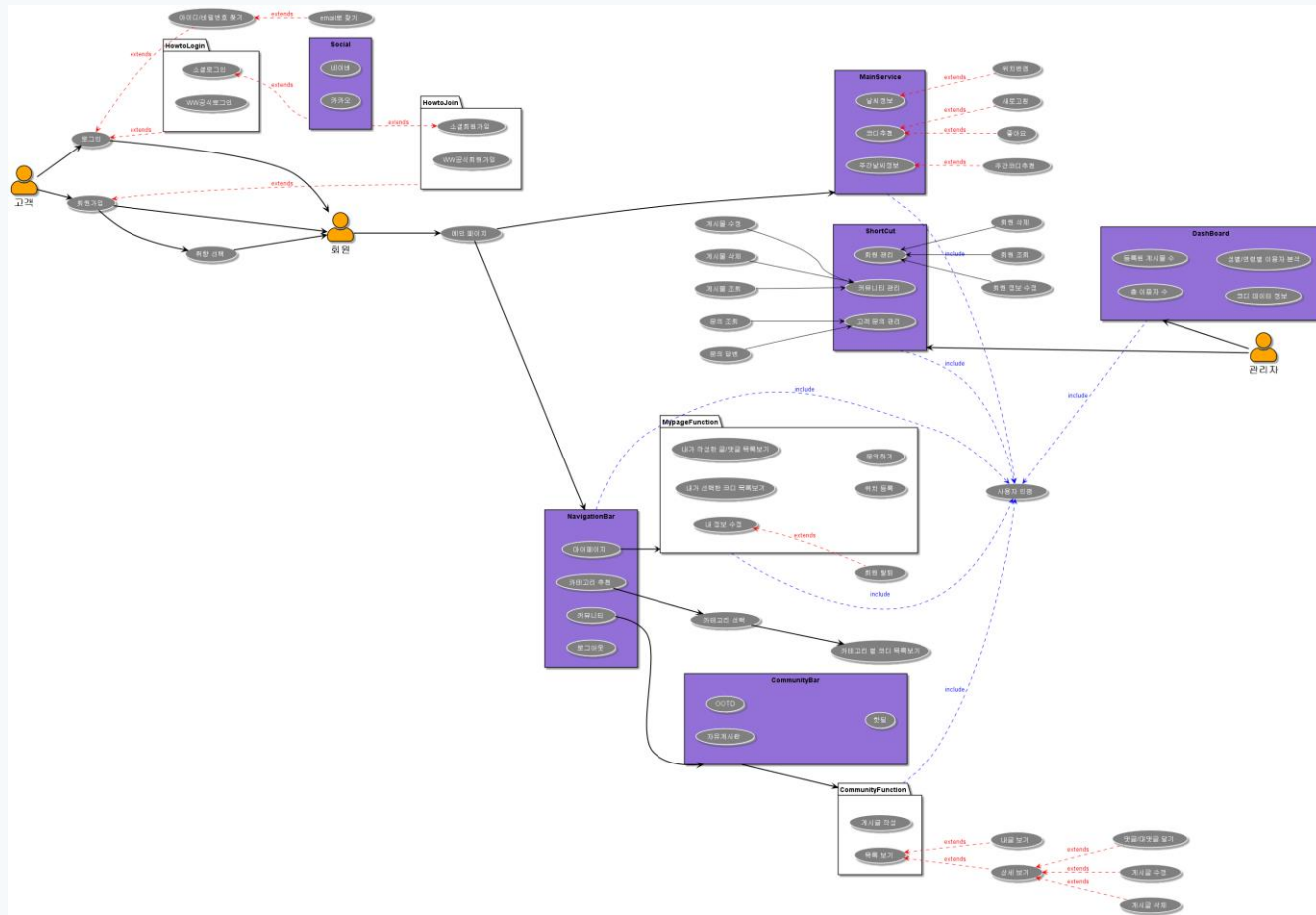


분석 설계

Gantt Chart

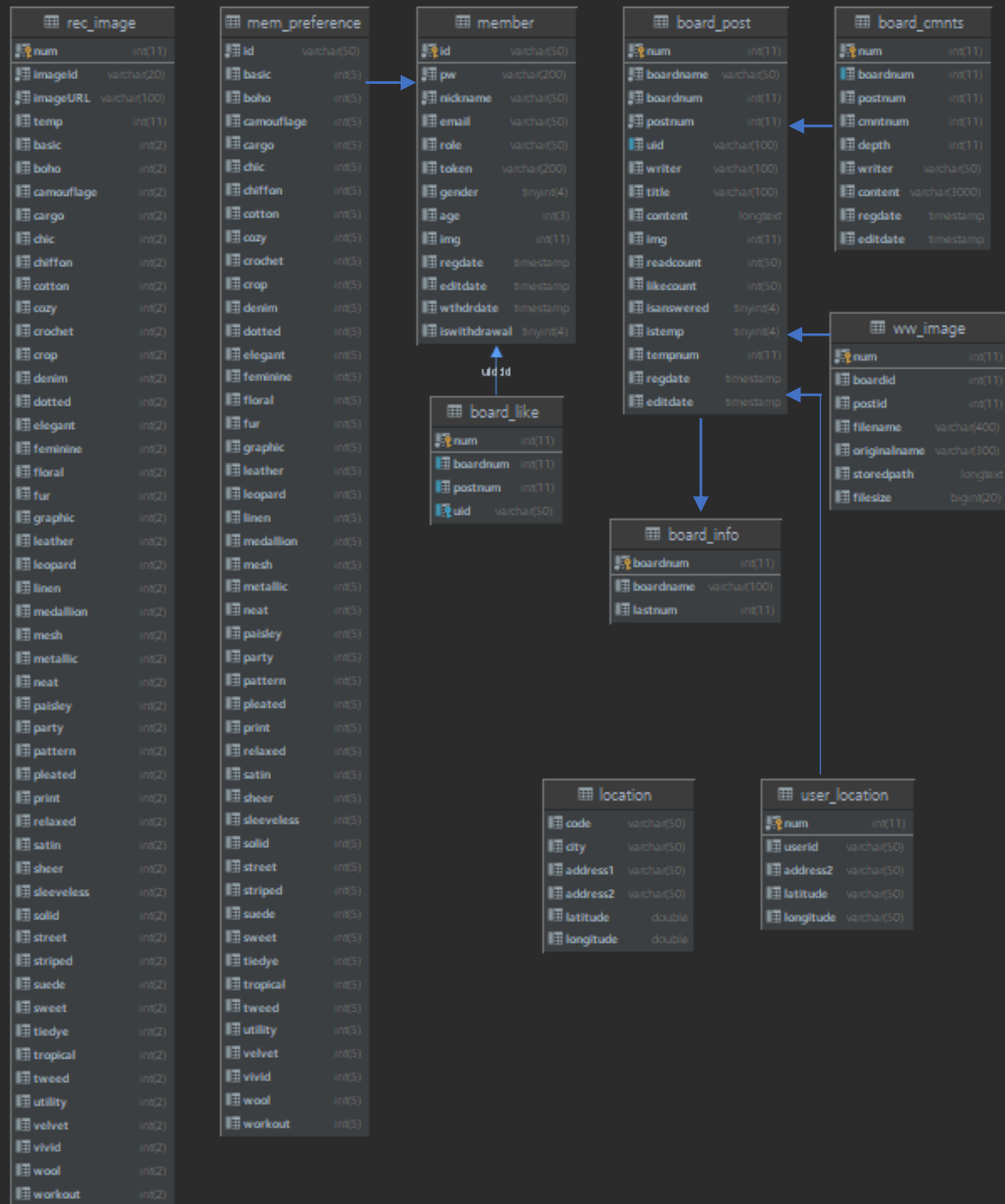


Use Case Diagram



분석 설계

ERD



분석 설계

UI Storyboard

스토리보드 규칙

기능	명명 규칙	설명
내비게이션 바	NV	Navigation bar
서비스 소개	IN	Index
코드 추천	RC	Recommendation
커뮤니티	CO	Community
마이페이지	UI	User Info
로그인	SI	Sign In
회원가입	CA	Create Account
아이디/패스워드 찾기	FA	Find Account
대시보드	DB	Dash Board
회원관리	MM	Member Management
커뮤니티 관리	CM	Community Management
고객문의 관리	IM	Inquiry Management

스토리보드 UI 표시 아이콘

이미지 박스

드롭다운

인포 텍스트

(버튼
활성)







(버튼
비활성)

프로필

라디오버튼

체크박스

스토리보드 영역 표시 및 기타사항

번호(영역)	 		좌 → 우, 상 → 하 순서로 넘버링
번호			좌 → 우, 상 → 하 순서로 넘버링
선택적 노출영역	-		광고 배너 등 특정상황에 노출되는 영역 표시
UI	폰트: Noto Sans CJK, 8pt, 10pt	Description	폰트: Noto Sans CJK, 9pt

Subject	메인 페이지 (회원 로그인)
Page ID	RC-01
Path	
Description	<p>로그인</p> <p>오늘의 날씨</p> <p>코드 추천</p> <p>날짜</p> <p>기온</p> <p>다음페이지 계속</p>
작성일	2021-05-07
작성자	이재현
Description	<p>A</p> <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p>

Subject	공동 1내비게이션, footer
Page ID	NV-01, NV-02
Path	
Description	<p>A</p> <p>B</p> <p>C</p> <p>사립자 번호: 101-86-59749</p> <p>대표자: 임종혁</p>
작성일	2021-05-07
작성자	이재현
Description	<p>A</p> <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p>

Subject	커뮤니티 메인
Page ID	CO-01
Path	
Description	<p>A</p> <p>B</p> <p>C</p> <p>D</p> <p>E</p> <p>F</p> <p>G</p> <p>H</p> <p>I</p> <p>J</p> <p>K</p> <p>L</p> <p>M</p> <p>N</p> <p>O</p> <p>P</p> <p>Q</p> <p>R</p> <p>S</p> <p>T</p> <p>U</p> <p>V</p> <p>W</p> <p>X</p> <p>Y</p> <p>Z</p>
작성일	2021-05-07
작성자	임종혁
Description	<p>A</p> <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p> <p>7</p> <p>8</p> <p>9</p> <p>10</p> <p>11</p> <p>12</p> <p>13</p> <p>14</p> <p>15</p> <p>16</p> <p>17</p> <p>18</p> <p>19</p> <p>20</p> <p>21</p> <p>22</p> <p>23</p> <p>24</p> <p>25</p> <p>26</p> <p>27</p> <p>28</p> <p>29</p> <p>30</p> <p>31</p> <p>32</p> <p>33</p> <p>34</p> <p>35</p> <p>36</p> <p>37</p> <p>38</p> <p>39</p> <p>40</p> <p>41</p> <p>42</p> <p>43</p> <p>44</p> <p>45</p> <p>46</p> <p>47</p> <p>48</p> <p>49</p> <p>50</p> <p>51</p> <p>52</p> <p>53</p> <p>54</p> <p>55</p> <p>56</p> <p>57</p> <p>58</p> <p>59</p> <p>60</p> <p>61</p> <p>62</p> <p>63</p> <p>64</p> <p>65</p> <p>66</p> <p>67</p> <p>68</p> <p>69</p> <p>70</p> <p>71</p> <p>72</p> <p>73</p> <p>74</p> <p>75</p> <p>76</p> <p>77</p> <p>78</p> <p>79</p> <p>80</p> <p>81</p> <p>82</p> <p>83</p> <p>84</p> <p>85</p> <p>86</p> <p>87</p> <p>88</p> <p>89</p> <p>90</p> <p>91</p> <p>92</p> <p>93</p> <p>94</p> <p>95</p> <p>96</p> <p>97</p> <p>98</p> <p>99</p> <p>100</p>

UI Storyboard 예시

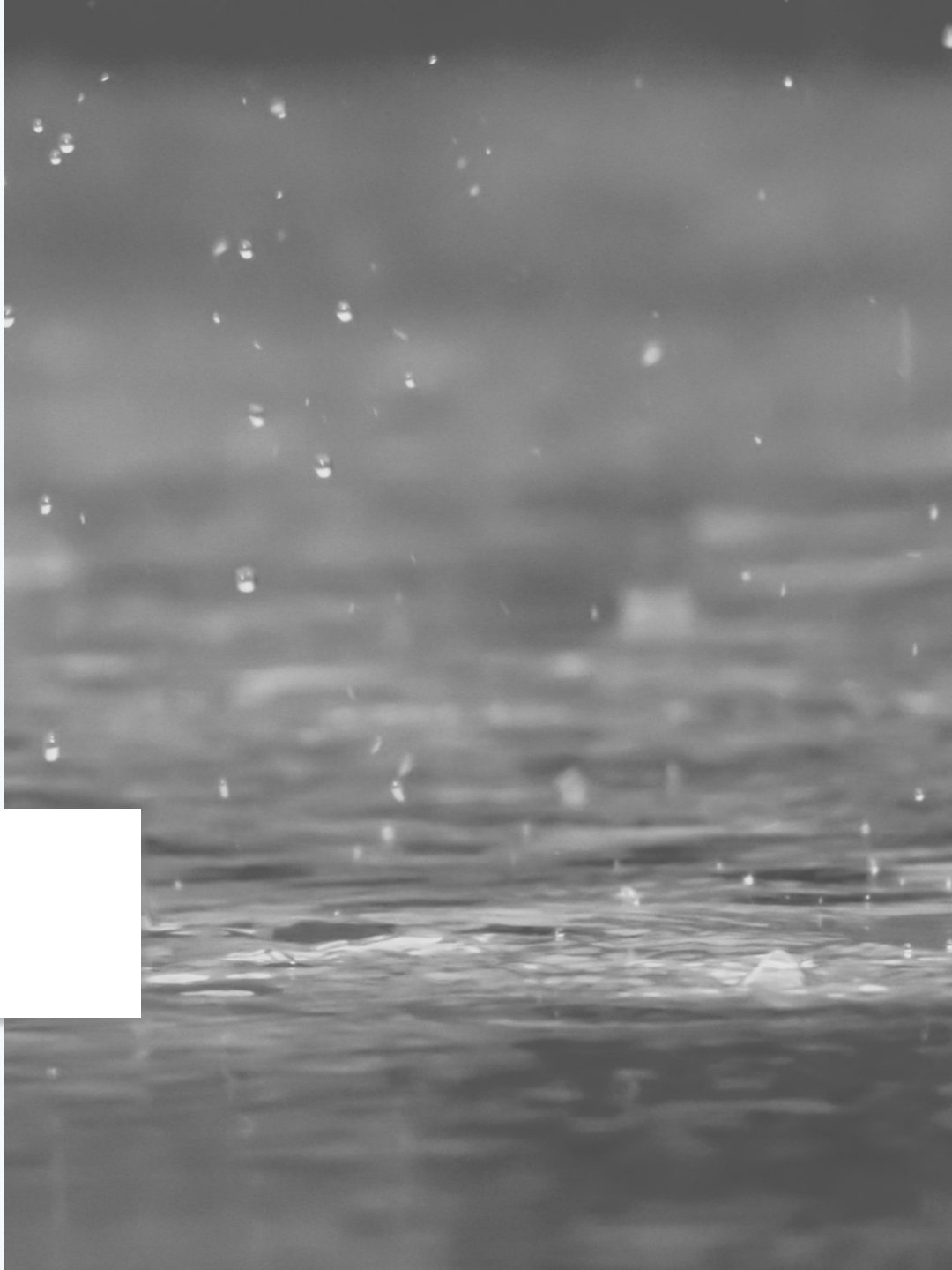
시연



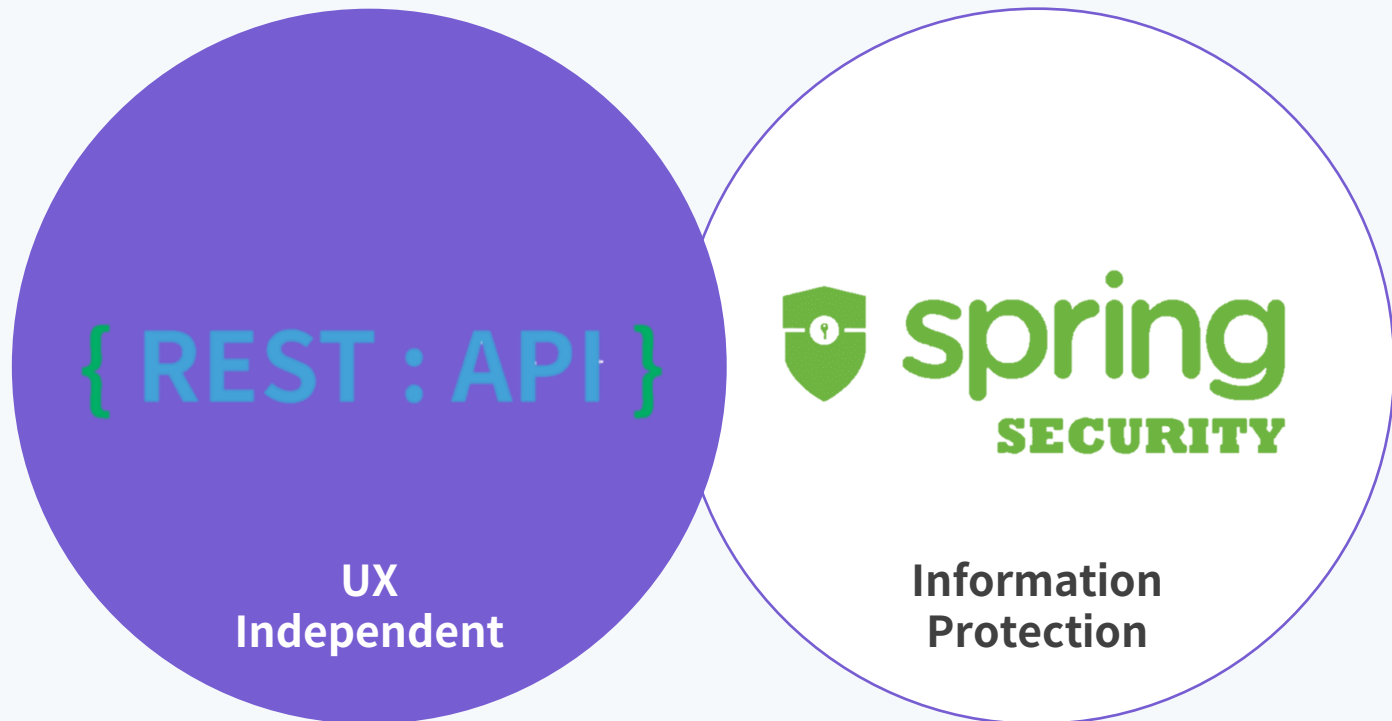
Weather Wear 시연



기술소개

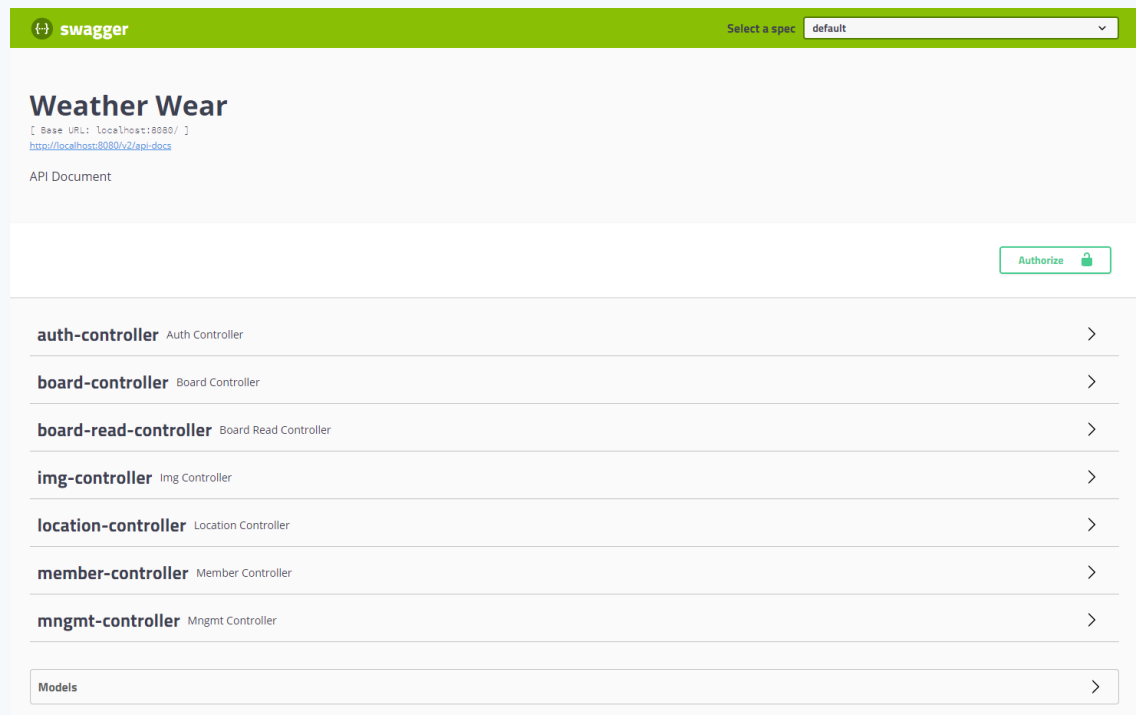


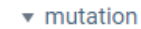
KEY FUNCTIONS



KEY FUNCTIONS | REST API

7개의 카테고리, 46개의 URI





▼ state

- ▼ getters

18

KEY FUNCTIONS | SECURITY



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva  
G4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1Kx  
wRJSMekKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

KEY FUNCTIONS | SECURITY



```
export function setInterceptors(instance) {  
  instance.interceptors.request.use(  
    onFulfilled: function (config : AxiosRequestConfig ) {  
      //Header에 인증 토큰 작성  
      config.headers.Authorization = store.state.token;  
      return config;  
    },  
    onRejected: function (error) {  
      return Promise.reject(error);  
    },  
  );  
  return instance;  
}
```


날씨 API | OpenWeatherMap

```
export function get7daysWeather(lat, lon) {
  return instance.get(`https://api.openweathermap.org/data/2.5/onecall?lat=${lat}&lon=${lon}&exclude=minutely,hourly,alerts&appid=${받은 api key값}&units=metric&lang=kr`);
}

export let weatherData = '';

function saveCoords(coordsObj) {
  localStorage.setItem('coords', JSON.stringify(coordsObj));
}

function handleGeoSuccess(position) {
  console.log(position);
  const latitude = position.coords.latitude; // 경도
  const longitude = position.coords.longitude; // 위도
  const coordsObj = {latitude, longitude};
  saveCoords(coordsObj);
}

function handleGeoError(err) {
  alert("위치를 찾을 수 없습니다.");
}
```

날씨 API | OpenWeatherMap

```
export async function loadCoords() {  
  // localStorage에서 위치정보 가져옴  
  const loadedCoords = localStorage.getItem('coords');  
  
  if (loadedCoords === null) { // 위치 정보가 없으면  
    // 위치 정보 요청 함수  
    navigator.geolocation.getCurrentPosition(handleGeoSuccess, handleGeoError);  
  } else {  
    // json형식을 객체 타입으로 바꿔서 저장  
    const parseCoords = JSON.parse(loadedCoords);  
    const {data} = await get7daysWeather(parseCoords.latitude,  
                                          parseCoords.longitude)  
  
    weatherData = data;  
  }  
}
```

관리자페이지 | JPA를 활용한 데이터 생성

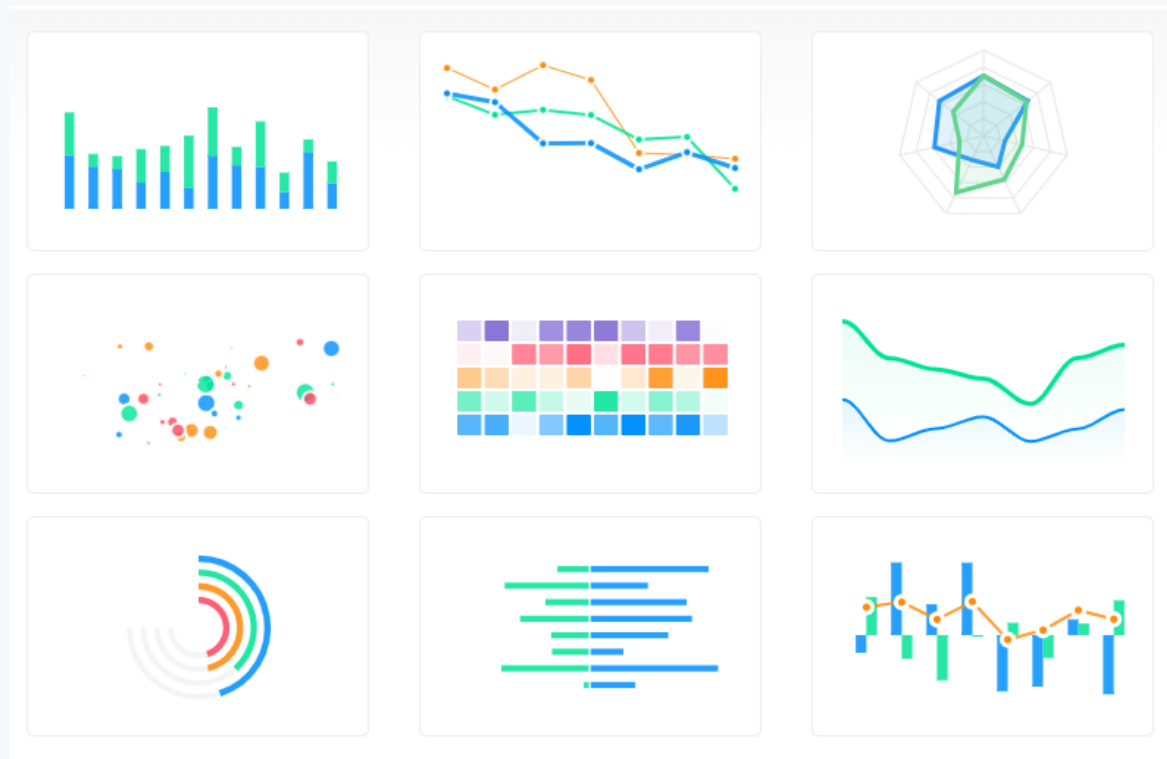
```
// 총 회원수
int countAllByIdIsNotNull();
// 성별, 연령별 회원 수
int countAllByGenderAndIswithdrawalIsFalseAndAgeBetween(boolean gender, int start, int end);
// 총 회원 중 가입탈퇴 수
int countAllByIswithdrawalIsFalse();
int countAllByIswithdrawalIsTrue();
// 일일 가입 탈퇴 수
int countAllByIswithdrawalIsFalseAndRegdateIsBetween(LocalDateTime start, LocalDateTime end);
int countAllByIswithdrawalIsTrueAndWthdrdateIsBetween(LocalDateTime start, LocalDateTime end);
```

관리자페이지 | HashMap 형태로 데이터 출력

```
public HashMap memberInfos() {  
    HashMap<String, Object>infos = new HashMap<>();  
    infos.put("cntMembers", memberService.cntMembers());  
    infos.put("cntWthdrFalse", memberService.cntWthdrFalse());  
    infos.put("cntWthdrTrue", memberService.cntWthdrTrue());  
    infos.put("cnt7days", memberService.cntToday());  
    return infos;  
}
```

ApexChart 라이브러리

Modern interactive opensource chart



ApexChart 라이브러리

01 npm install -> 라이브러리 선언

```
npm install apexcharts --save  
npm install vue-apexcharts --save
```

```
import Vue from 'vue'  
import VueApexCharts from 'vue-apexcharts'  
Vue.use(VueApexCharts)  
Vue.component('apexchart', VueApexCharts)
```


ApexChart 라이브러리

02 출력하고자 하는 데이터 입력

```
var bar = {
  chart: {
    id: '게시판 별 포스팅 수',
    type: 'bar',
    height: 380,
    width: '100%',
    stacked: true,
  },
  plotOptions: {
    bar: {
      columnWidth: '45%',
    }
  },
  colors: ['#00D8B6', '#008FFB'],
  series: [{
    name: "최신 포스팅 수",
    data: [this.cntRecentFB, this.cntRecentOOTD],
  }],
  xaxis: {
    categories: ['자유게시판', 'OOTD'],
    labels: {
      show: true,
    },
    axisBorder: {
      show: false
    }
  }
}
```

ApexChart 라이브러리

03 출력하고자 하는 위치에 차트로 렌더링

```
        show: false
      },
      axisTicks: {
        show: false
      },
      labels: {
        style: {
          colors: '#78909c'
        }
      }
    },
    title: {
      text: '게시판 별 주간 포스팅 수',
      align: 'left',
      style: {
        fontSize: '18px'
      }
    }
  }

  }
  var elBar = document.getElementById("bar");
  var chartBar = new ApexCharts(elBar, bar);
  await chartBar.render();
```

의상 특징 인식 API



의상 특징 인식 API | 학습데이터 전처리



DeepFashion: Attribute Prediction Data Set

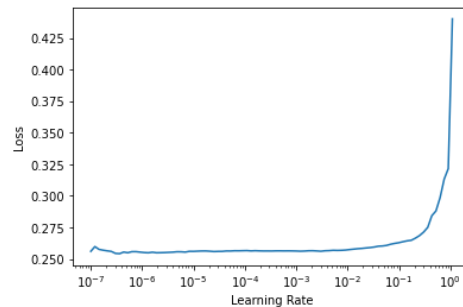
총 289,222장의 이미지 중 59,000장
47개의 label

의상 특징 인식 API | 모델생성

```
In [24]: learn = cnn_learner(dls, resnet34, loss_func=LabelSmoothingBCEWithLogitsLossFlat(),
                             metrics=metrics, opt_func=opt_func).to_fp16()
learn.fine_tune(2)
```

```
In [25]: learn.unfreeze()
learn.lr_find()
```

```
Out [25]: SuggestedLRs(lr_min=1.318256749982538e-07, lr_steep=1.5848931980144698e-06)
```



```
In [36]: learn.fit_one_cycle(20, lr_max=slice(2e-7, 1e-4))
```

epoch	train_loss	valid_loss	fbeta_score	accuracy_multi	time
0	0.255221	0.250692	0.461737	0.966072	03:46
1	0.254102	0.249641	0.461270	0.966248	03:46
2	0.254827	0.249696	0.463054	0.966717	03:47
3	0.254398	0.249173	0.467232	0.966530	03:46
4	0.253013	0.248348	0.470602	0.966417	03:47
5	0.252080	0.248248	0.470756	0.966800	03:46
6	0.252670	0.248751	0.476192	0.966568	03:46
7	0.251195	0.248570	0.480184	0.966764	03:46
8	0.250887	0.247764	0.481248	0.966893	03:48
9	0.250388	0.247953	0.481098	0.966490	03:46
10	0.250142	0.247124	0.481248	0.967015	03:46
11	0.249958	0.247551	0.485149	0.966825	03:46
12	0.250324	0.247568	0.487004	0.966947	03:47
13	0.249781	0.247235	0.488340	0.966970	03:47
14	0.249579	0.247236	0.488134	0.966336	03:46
15	0.249711	0.246896	0.488608	0.966690	03:47
16	0.249507	0.247144	0.490238	0.966990	03:46
17	0.249805	0.246532	0.488017	0.967271	03:46
18	0.249457	0.246748	0.488470	0.966884	03:46
19	0.249202	0.246840	0.488806	0.967012	03:47

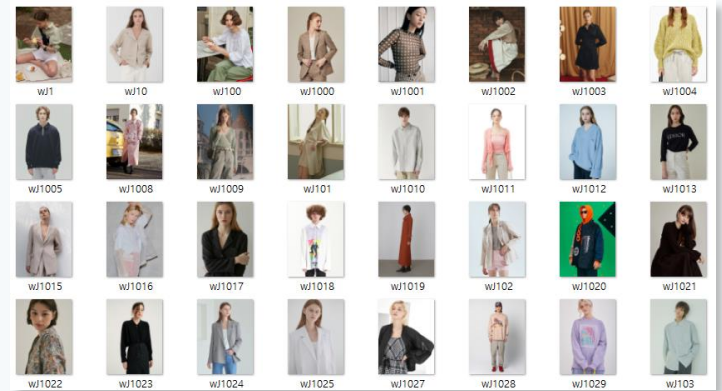
의상 특징 인식 API | 코디 이미지 크롤링

W.CONCEPT.

검색어 활용
이미지 데이터 수집
:: Selenium



중복 이미지 삭제 :: python
부적합 이미지 삭제



약 50,000건의 코디 이미지 수집

의상 특징 인식 API | 예측 및 저장

Multipart/form-data로 신규 코디 이미지 전달

→ 학습 모델을 통해 코디 이미지의 특징 예측

→ 예측 값이 있는 경우 aws S3에 업로드 및 DB 저장

01 신규 이미지 등록



02 이미지 특징 예측

"chiffon", "crochet", "pleated"

03 이미지 데이터 저장

	num	imageURL	camouflage	cargo	chic	chiffon	cotton	...	sweet	tiedye	tropical	tweed	utility	velvet	vivid	wool	workout	list
imageId																		
210629_160833.jpg	111	https://wwuptest.s3.ap-northeast-2.amazonaws.c...	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	print
210629_160908.jpg	112	https://wwuptest.s3.ap-northeast-2.amazonaws.c...	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	print
210629_160922.jpg	113	https://wwuptest.s3.ap-northeast-2.amazonaws.c...	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	print
210629_160944.jpg	114	https://wwuptest.s3.ap-northeast-2.amazonaws.c...	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	floral
210629_160957.jpg	115	https://wwuptest.s3.ap-northeast-2.amazonaws.c...	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	chiffon,crochet,pleated

추천 API | 선호 속성 데이터

코디 이미지에 대한 사용자의 선호도를 파악하여 추천 기준으로 활용

- 회원 가입시 최초 선호 데이터 생성
- 서비스 이용 중 코디 이미지에 대한 좋아요 클릭 시 선호 데이터 업데이트

	basic	boho	camouflage	cargo	chic	chiffon	cotton	cozy	crochet	crop	...	suede	sweet	tiedye	tropical	tweed	utility	velvet	vivid	wool
id																				
a	1	0	0	0	0	0	0	0	0	0	...	2	0	1	0	0	0	0	0	0
d	1	0	0	1	0	0	0	1	0	1	...	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	2	0	0	0	0	...	0	0	0	0	0	0	0	0	0
p	0	0	0	0	0	1	1	0	0	0	...	0	0	0	0	0	0	0	1	0

사용자 별 선호 속성 데이터(mem_preference)

	basic	boho	camouflage	cargo	chic	chiffon	cotton	cozy	crochet	crop	...	suede	sweet	tiedye	tropical	tweed	utility	velvet
imageId																		
210627_220913.jpg	0	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0
210627_221421.jpg	0	0	0	0	0	1	0	0	1	0	...	0	0	0	0	0	0	0
210627_221441.jpg	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
210627_221514.jpg	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0
210627_221534.jpg	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0

코디 별 속성 데이터(rec_image)

추천 API | 선호 속성 데이터

코디 별 속성 데이터(rec_image)에서 날씨에 맞는 데이터 선택 후
rec_image 테이블에 추천할 사용자의 선호 데이터를 추가 한 후
cosine_similarity(sklearn)을 사용하여 유사도 계산

```
imgUserAttr = imgAttr.append(userAttr)

target_index = imgUserAttr.shape[0] - 1
cosine_matrix =
cosine_similarity(imgUserAttr,
imgUserAttr)
sim_index =
cosine_matrix[target_index, :30].reshap
e(-1)
sim_index = sim_index[sim_index !=
target_index]

sim_scores = [(i, c) for i, c in
enumerate(cosine_matrix[target_index])
if i != target_index]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.000000	0.0	0.707107	0.0	0.707107	0.707107	0.707107	0.707107	0.0	0.0	0.0	0.000000	0.707107	0.0	0.223607
1	0.000000	1.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000
2	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
3	0.000000	0.0	0.000000	1.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.000000
4	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
5	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
6	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
7	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
8	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	1.0	1.0	0.0	0.000000	0.000000	0.0	0.000000
9	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	1.0	1.0	0.0	0.000000	0.000000	0.0	0.000000
10	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	1.0	0.000000	0.000000	1.0	0.000000
11	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	1.000000	0.000000	0.0	0.316228
12	0.707107	0.0	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	0.0	0.0	0.0	0.000000	1.000000	0.0	0.000000
13	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	1.0	0.000000	0.000000	1.0	0.000000
14	0.223607	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.316228	0.000000	0.0	1.000000

코디 이미지 + 특정 유저 유사도 테이블

소감

