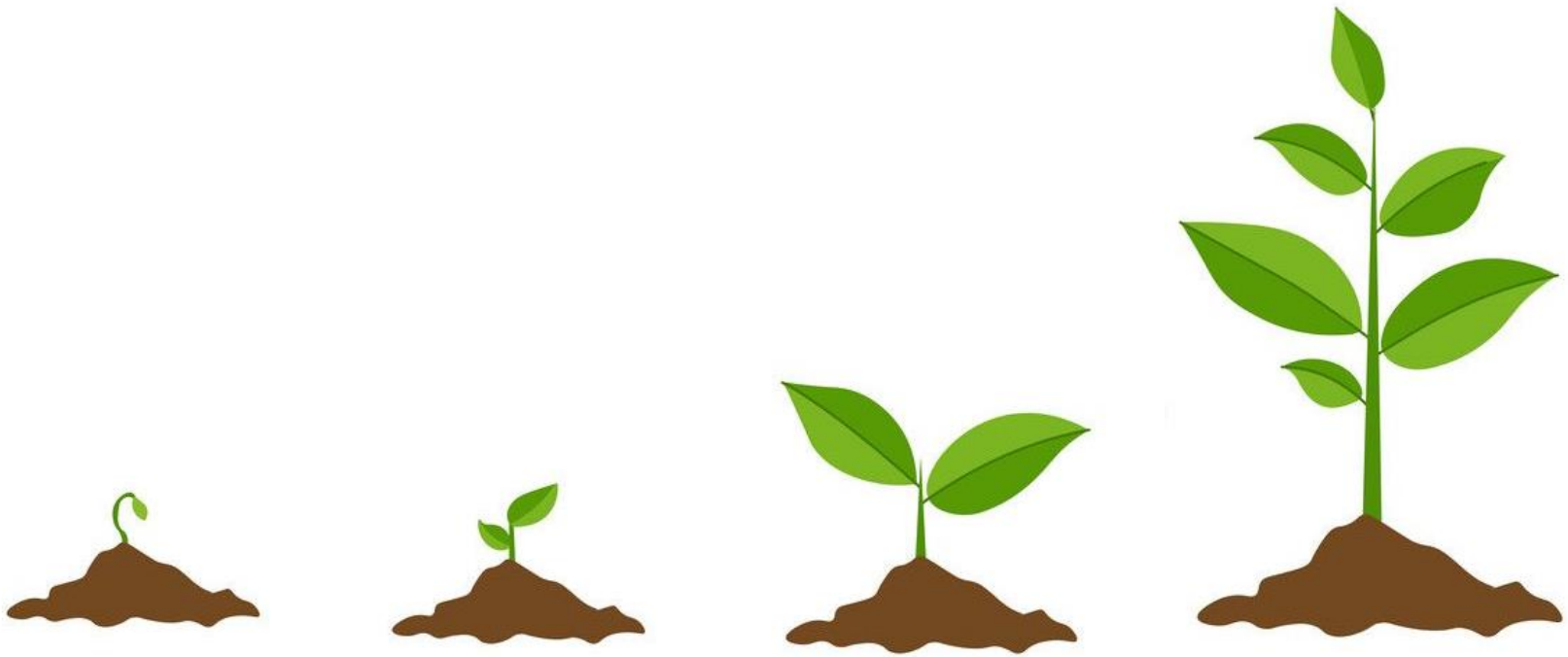
뼛속까지 개발자
김재원

순서



자기소개 >> 주요이력 >> 기술스펙 >> 프로젝트

1. 자기소개

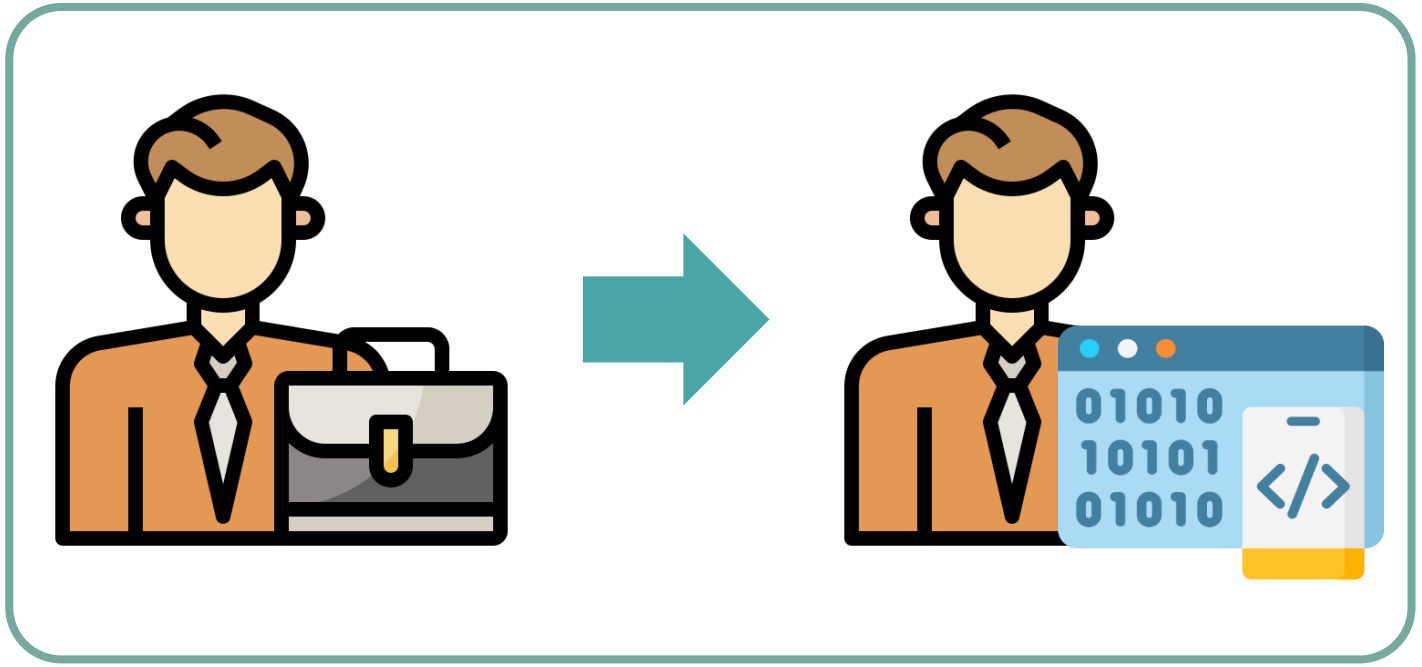
자기소개

- **Developer** 김재원의 포트폴리오 입니다.

- 대학교 전공 강의 중 Database론과 통계학을 이수하면서 Data 관련 전문 자격증 취득 목표
- SAS 자격증 및 정보처리기사 취득
- 위 경험을 통해 IT 전문가가 되길 희망

- 흥미 → 호기심 → 목표
- 흥미가 호기심이 되었고, 호기심은 곧 **뛰어난 개발자**가 되고자 하는 목표로 바뀜

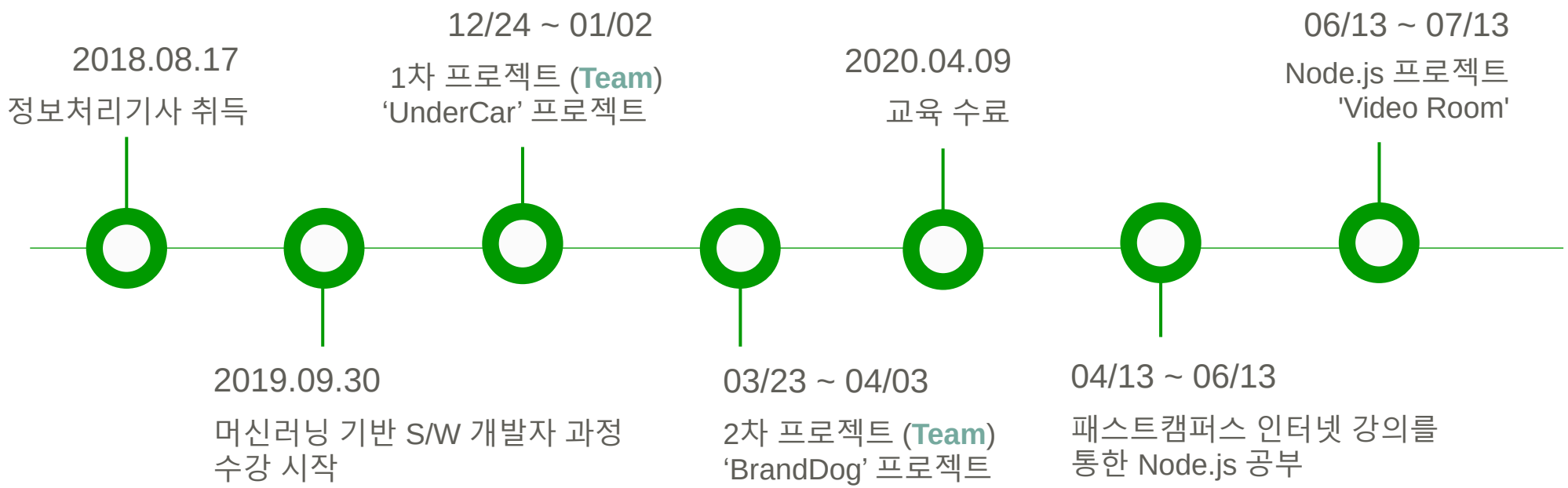
- Java, JavaScript, Python을 배움
- 그 중에 JavaScript의 다재 다능함에 빠지게 됨
- 이 후 JavaScript 런타임 중 하나인 **’Node.js**’를 공부하여 ‘Node.js’를 이용한 **서버 개발자**가 되기 위해 배움을 지속 중



이제 지금까지 제가 노력해온 결과물들을 보여드리겠습니다.

2. 주요이력

주요이력


2018.08.17
정보처리기사 취득
2019.09.30
머신러닝 기반 S/W 개발자 과정
수강 시작
12/24 ~ 01/02
1차 프로젝트 (Team)
‘UnderCar’ 프로젝트
03/23 ~ 04/03
2차 프로젝트 (Team)
‘BrandDog’ 프로젝트
2020.04.09
교육 수료
04/13 ~ 06/13
패스트캠퍼스 인터넷 강의를
통한 Node.js 공부
06/13 ~ 07/13
Node.js 프로젝트
'Video Room'

주요이력

- 머신러닝 기반 S/W 개발자 이수 내역

  - 기간: 2019.09.30 ~ 2020.04.09 (6개월)
  - 강의명: 머신러닝 기반 S/W 개발자
  - 기관: 경영기술개발교육원
  - 수강내용
    - 개발 입문 :
      JAVA → JSP 웹 프로그래밍 → Spring Framework
    - DB :
      Oracle (SQL 활용)
      MongoDB (NoSQL)
    - 웹표준
      HTML 5, CSS, JavaScript, JQuery
    - Python
      Python 기초, 데이터 분석, 시각화, 머신러닝
      Web Framwork(Django)

- Fast Campus Node.js 강의

  - 기간: 2020.04.13 ~ 2020.06.13 (2개월)
  - 인터넷 강의를 이용하여 Node.js 기본 지식 습득
  - 강의 내용:
    - JavaScript 최신 문법(ES6)
    - Express.js를 이용한 서버 구축
    - Promise, async - await를 이용한 비동기 처리
    - Docker 기본 개념 및 사용
    - 다양한 Design Pattern
  - 추가
    - AWS EC2를 이용한 Linux 사용
    - AWS EC2 인스턴스에서 Docker 실습

학습 관련 Github: https://github.com/angelica127/DevLectureNote

3. 기술스펙

기술스펙

Front-End

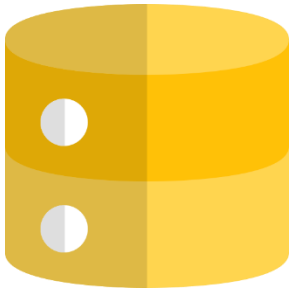
- HTML, CSS, JS 웹 퍼블리싱
- Ajax 활용 비동기통신

- HTML
- CSS
- Bootstrap
- JavaScript
- JQuery
- Ajax
- Fetch

Back-End

- MVC 패턴의 웹 개발
- CRUD 데이터 처리
- 데이터 수집, 정제, 분석

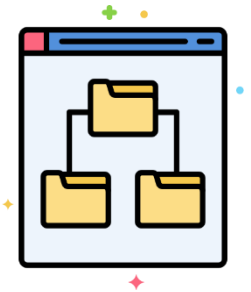
- Java
- Python
- Spring Framework
- Django
- Node.js
- Express.js
- Apache Tomcat
- nginx



Data Base

- CRUD 처리
- 데이터 모델링 처리 가능
- NoSQL 기초

- Oracle
- MySQL
- MongoDB



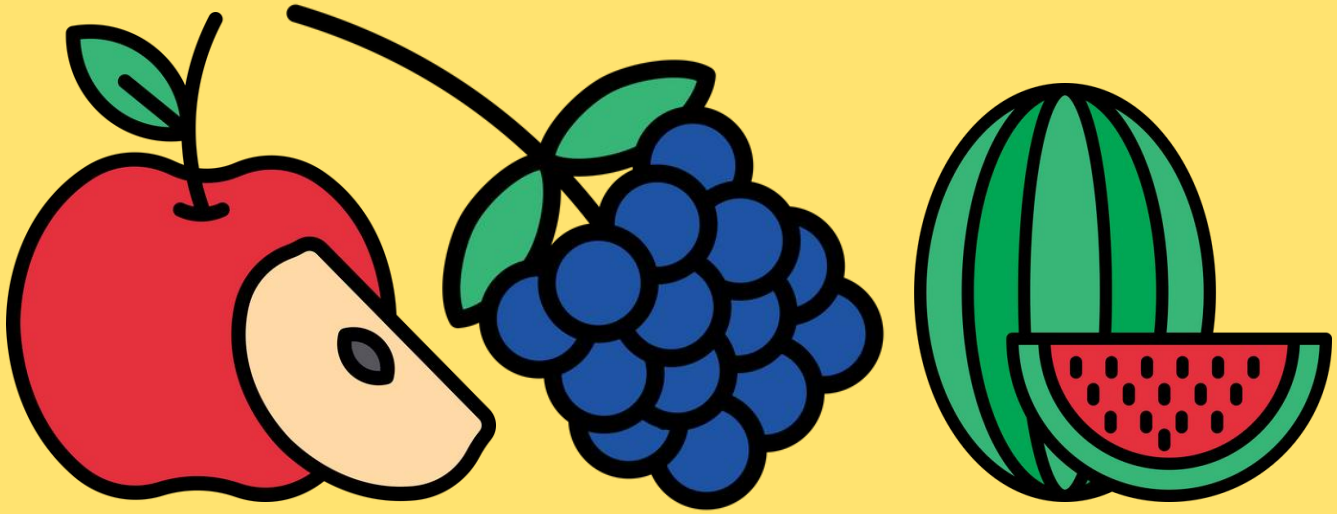
Version Control

- 프로젝트 진행하면서 형상관리 경험
- 팀원과의 협업 경험
- git 기본 사용법 숙지

- SVN
- Github

기타

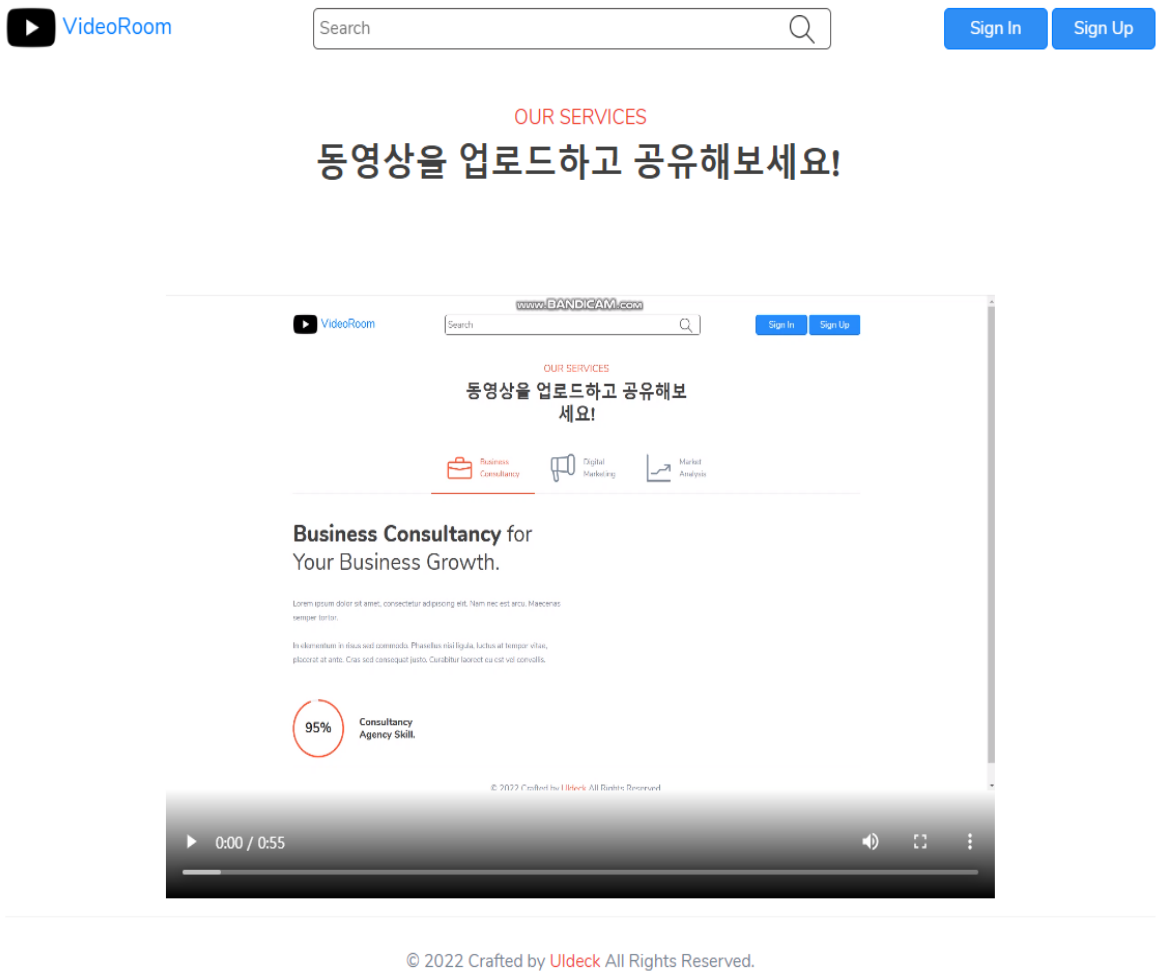
- AWS
- Docker
- Linux



4.1 Video Room

4.1 Node.js Project

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)



- 프로젝트 명 : **Video Room**
- 프로젝트 기간 : 2020.06.13 ~ 2020.07.13
- 프로젝트 내용 :
  - 동영상 스트리밍 서버 구축(Node.js, MongoDB)
  - Client 접속 서버
    - 클라우드 인프라 플랫폼 이용
    - 하드웨어 인스턴스 (AWS EC2)
  - Service Logic 서버
    - Container Software (Docker)
- Github :
- https://github.com/angelica127/webapp-video-streaming
- 배포 : AWS EC2 인스턴스 http://3.12.212.58/

4.1.1 개발 환경

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

분류	Resource	비고
Back – End	Node.js	12.18.2
Front – End	HTML	HTML5
	CSS	CSS 3
	Bootstrap	Bootstrap 4
	JavaScript	
	Templating Engine	nunjucks (v 3.2.1)
Database	MongoDB	4.2.8
WS	nginx	1.18.0
Deployment	AWS EC2 – Amazon Linux 2 with .Net Core, PowerShell, Mono, and MATE Desktop Environment	
	Docker	19.03.6-ec

4.1.2 Why Use MongoDB ?

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

- 동영상 업로드 처리 방안 고민

  기존 RDB 한계 : File 경로에 대한 Row Data만 저장되는 점

MongoDB GridFS 사용

- 동영상 타입의 데이터 (이진 데이터) 저장 가능
- Meta Data 형식의 추가정보 저장
- MongoDB 외 별도의 저장소 불필요

4.1.3 동영상 업로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

- Node.js의 Package 중 ‘mongodb’ Package를 이용

  → NoSQL의 기본 사용법을 익히기 위해 **MongoDB**의 ODM인 ‘**mongoose**’를 사용하지 않음

- Dropzone.js와 ‘multer’, ‘mongodb’ package를 이용하여 동영상 업로드 처리

- 동영상 업로드 처리 과정

  1. Dropzone.js에 동영상 업로드
  2. 업로드한 file 정보를 토대로 MongoDB에 저장 (GridFs 이용)
  3. MongoDB에 저장 후 사용자가 추가 정보 입력
  4. 추가 정보를 업로드한 파일의 MongoDB Document에 추가
  5. 동영상 화면에서 MongoDB에서 다운받은 파일 Play

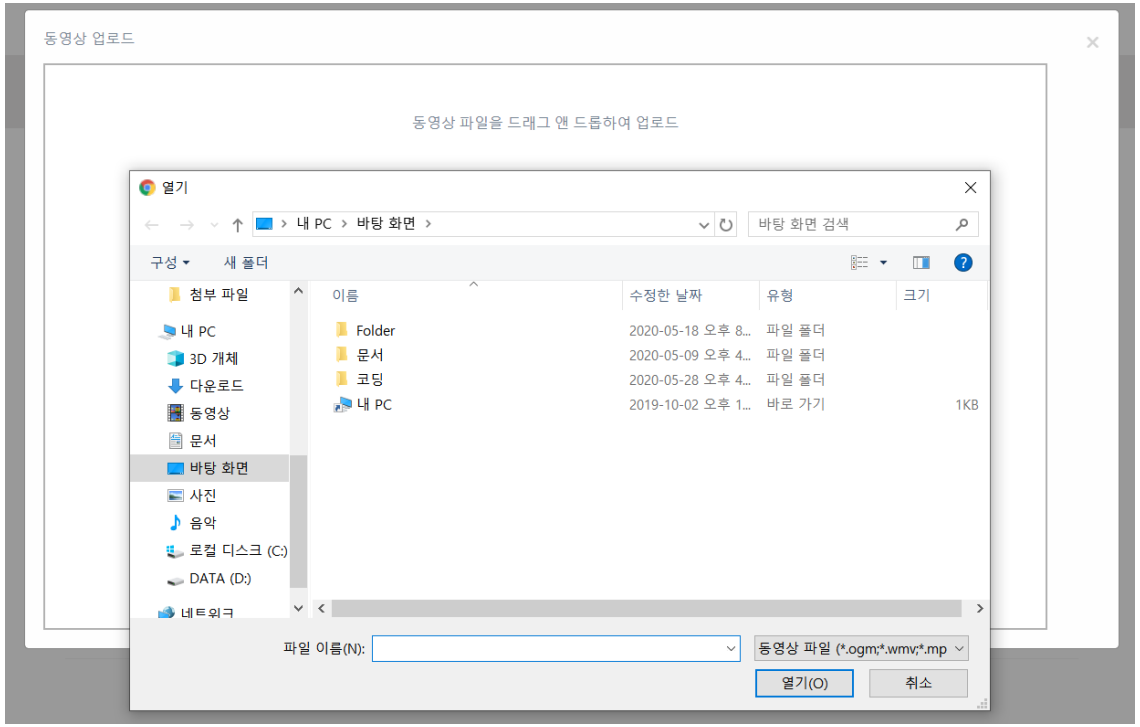
- AWS 배포환경에서 nginx 이용 시 파일 업로드에 관한 설정 필요

  → 현재 설정은 제한 없음으로 설정함

```
http {
    # Set Client upload size - No Limit
    client_max_body_size 0;
```

4.1.3 동영상 업로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)



1. Dropzone.js 화면

```
// Module for File Upload
const multer = require('multer'); // multer모듈 (for 파일업로드)

const videoStorage = multer.diskStorage({
    destination: function(req, file, cb) {
        cb(null, 'uploads/video');
    }
});
const videoUpload = multer({storage: videoStorage});

const imgStorage = multer.diskStorage({
    destination: function(req, file, cb) {
        cb(null, 'uploads/img');
    },
    filename: function(req, file, cb) {
        file.uploadedFile = {
            ext: file.mimetype.split('/')[1],
            name: new Date().valueOf(),
            type: file.mimetype
        }
        cb(null, file.uploadedFile.name + '.' + file.uploadedFile.ext);
    }
});
const imgUpload = multer({storage: imgStorage});

router.post('/upload', videoUpload.single('file'), service.post_upload_video);
```

2. Dropzone.js로 업로드한 file 정보 받기
(Controllers)

4.1.3 동영상 업로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

```
exports.post_upload_video = (req, res) => {
    const file = req.file;
    const id = req.session.video_id;

    Video.videoUpload(file, id, (err, result) => {
        if(err) res.json({code: -1, message: '오류가 발생하였습니다. 다시 시도해 주세요.'});

        const video_id = result;
        const obj = {
            user_id: id,
            video_id: video_id
        }

        Video.findOne(obj, (err, data) => {
            if(err) res.json({code: -1, message: '오류가 발생하였습니다. 다시 시도해 주세요.'});

            // data = Video 정보
            res.json(data);
        }); // end of findOne

        // 서버에 업로드한 파일 삭제
        fs.unlink(file.path, (err) => {
            if(err) throw err;

            console.log('file delete!');
        });

    }); // end of video Upload
} // end of upload_video
```

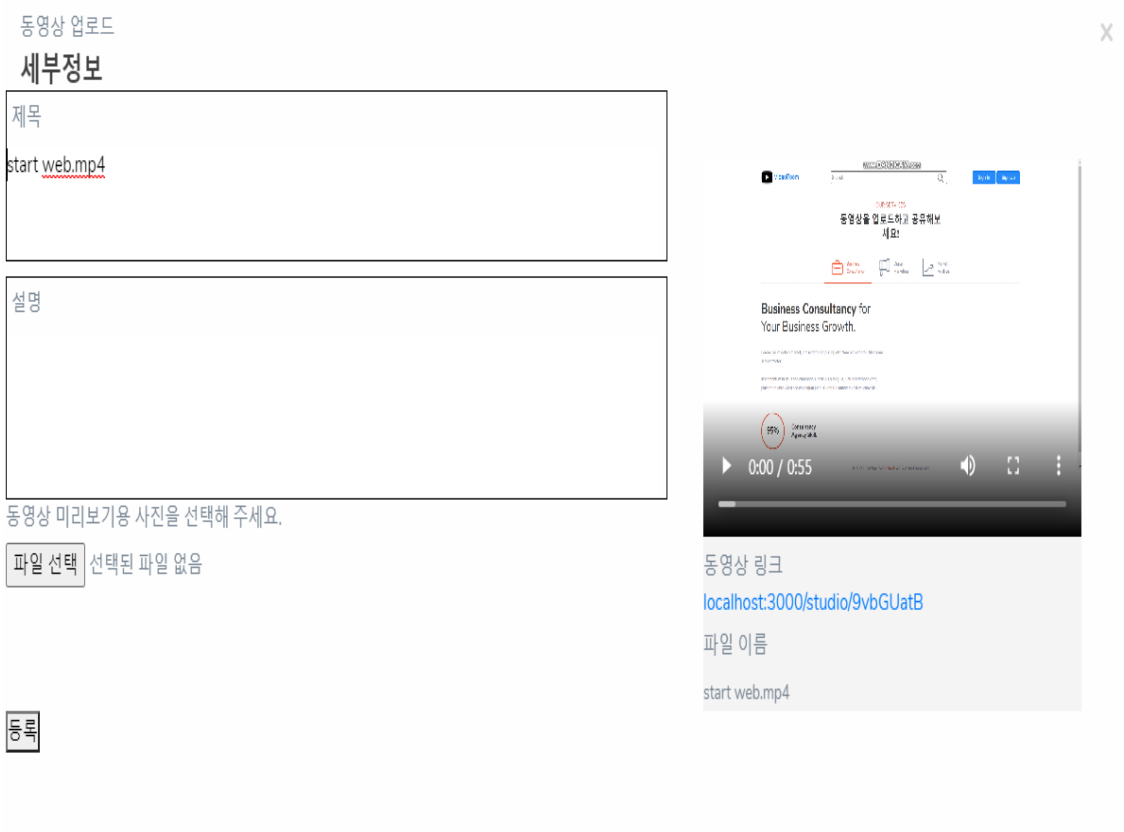
3. 처리 Service 함수
(Services)

```
// Video Upload
videoUpload: (fileObj, id, callback) => {
    Client.connect(process.env.MONGO_URI, {useUnifiedTopology: true}, async (err, client) => {
        if(err) throw err;

        const file = fileObj;
        const user_id = id;
        const video_id = createRandomString();
        const db = client.db(process.env.VIDEO_DB);
        const bucket = new mongo.GridFSBucket(db, {
            bucketName: user_id
        });

        fs.createReadStream(file.path)
        .pipe(bucket.openUploadStreamWithId(video_id, file.originalname))
        .on('error', (err) => {
            if(err) {
                console.log(err);
                client.close();
            }
        })
        .on('finish', () => {
            client.close();
            callback(null, video_id);
        })

    }); // end of Client.connect()
}, // end of videoUpload
```

4. Node.js의 File System과 MongoDB의 GridFS
(Models)

4.1.3 동영상 업로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)



5. 동영상에 대한 추가 정보 입력

```
updateVideoInfo: async (params) => {
    let collection_id = params.col_id;

    const Client = new mongo.MongoClient(process.env.MONGO_URI, {useUnifiedTopology: true});
    await Client.connect();
    const collection = Client.db(process.env.VIDEO_DB).collection(collection_id + '.files');

    // Inof Update

    let result;
    if (params.type === undefined) {
        result = await collection.updateOne({_id: params.fileId}, {$set: {
            title: [params.title],
            explanation: [params.explanation],
            image: [params.image],
        }})

    } else {
        result = await collection.updateOne({_id: params.fileId}, {$push: {
            title: params.title,
            explanation: params.explanation,
            image: params.image,
            type: params.type
        }});

    }

    await Client.close();

    return new Promise((resolve, reject) => {
        if (result.modifiedCount === 1) resolve('success');
        else reject(new Error('update error'));
    });

}, // end of updateVideoInfo
```

6. 동영상 업로드 마무리
(Models)

4.1.4 동영상 다운로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

- 회원 고유의 ID = MongoDB의 File이 저장된 ‘Video’ Database의 Collection Name
  - 회원 고유 ID는 Session에 저장되어 있다.
- 다운받을 File의 ID

위의 2개의 정보를 토대로 업로드한 동영상을 다운로드 한다.

4.1.4 동영상 다운로드 처리

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

```
videoDownLoad: async (params, callback) => {
    const video_id = params.video_id;
    const col_id = params.col_id;
    const type = params.type;
    const ext = type.substr(type.indexOf('/') + 1);

    const Client = new mongo.MongoClient(process.env.MONGO_URI, {useUnifiedTopology: true});
    await Client.connect();
    // video DB에 연결
    const db = Client.db(process.env.VIDEO_DB);

    const bucket = new mongo.GridFSBucket(db, {
        chunkSizeBytes: 1024,
        bucketName: col_id
    });

    const downloadStream = bucket.openDownloadStream(video_id);
    const outputStream = fs.createWriteStream(path.join(__dirname, `../uploads/video/${video_id + '.' + ext}`));
    await pipeline(downloadStream, outputStream);

    await Client.close();

    let message;
    fs.stat(path.join(__dirname, `../uploads/video/${video_id + '.' + ext}`), (err, result) => {
        if(err) {
            message = 'error';
            callback(null, message);
        } else {
            message = 'success';
            callback(null, message);
        }
    });

}, // end of videoDownLoad
```

File이 업로드 된 Collection 지정

1. 'mongodb'의 GridFS Method 중 'openDownloadStream' Mehtod를 통해 Download Stream을 얻는다.

2. fs Module을 이용하여 uploads 폴더 밑의 video 폴더에 동영상을 다운로드 한다.

3. 다운로드 된 파일명은 고유 ID에 확장자를 추가한 것으로 한다.

4. 다운로드 받은 파일을 통해 사용자에게 동영상을 보여준다.

4.1.5 AWS EC2

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

- Nginx를 이용한 Reverse Proxy Server 설정

- Reverse Proxy :
  외부에서 내부 서버가 제공하는 서비스 접근 시, Proxy 서버를 먼저 거쳐서 내부 서버로 들어오는 방식
- nginx의 conf 파일을 알맞게 수정
  - http 블록 중 server 블록 수정

```
# Setting proxy
location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;

        proxy_pass http://127.0.0.1:3000/;
        proxy_redirect off;
}
```

- 80 Port로 Node Application 접근 가능

- Docker 실습환경 구축

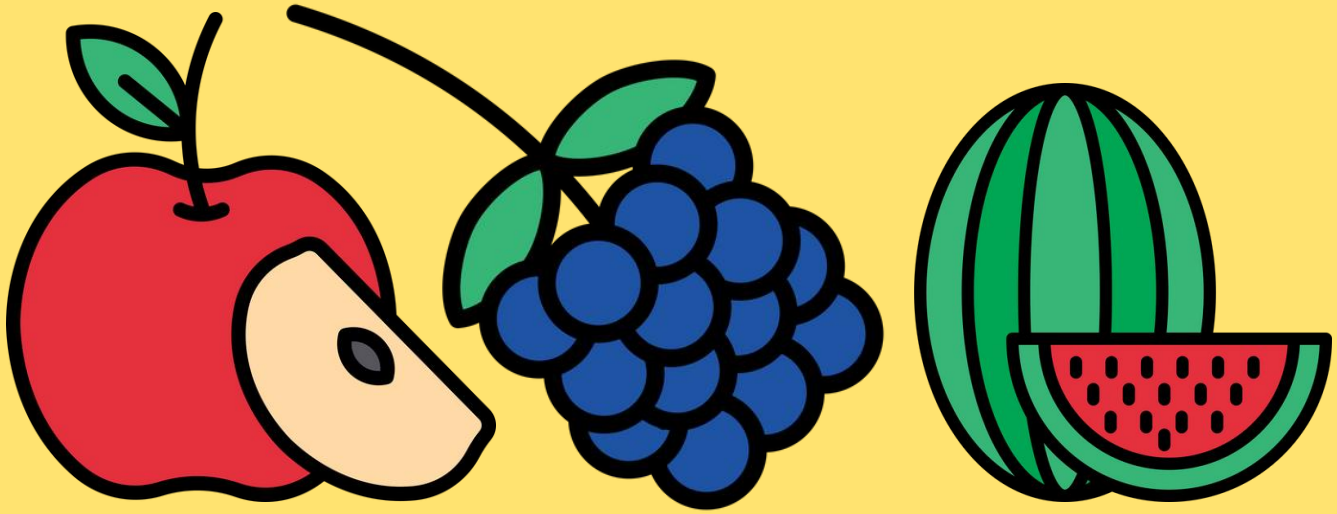
- Docker 설치 후 MongoDB 공식 이미지 Pull
- Pull한 이미지 실행
- docker run –name [container name] –d –p 27017:27017 mongo
  - -d : background 실행
  - -p : 호스트의 27017 port를 docker container의 27017 port로 연결
- docker 구동 확인
  - ps –ef | grep docker
- docker container의 bash 접속
  - docker –i –t [container name] /bin/bash
- docker container에 할당된 IP 확인
  - docker inspect –format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' [컨테이너 이름]

4.1.6 개발 후기

Node.js를 이용한 동영상 스트리밍 서버 구축 (개인 Project)

- **Video Room** 동영상 스트리밍 서버 구축 프로젝트 후기

개발에 대해 배우기 시작한 후 처음으로 나만의 생각을 구현한 첫번째 프로젝트 였다. 누군가의 도움이 없다는 점에서 처음에는 매우 막막했다. 막막함을 이겨내고 첫 발을 떼면서 많은 우여곡절을 겪게 되었다. 모르는 부분과 막히는 부분을 하나씩 찾아가며 하나씩 모습을 갖춰가는 것에 희열을 느꼈다. 하지만 그 반대로 결과물을 만들면서 아쉬움이 매우 컸다. '내가 좀 더 많이 알았다면... 더 좋은 결과물을 얻었을 텐데...'라는 생각이 많이 들었던 기간이었다. 많은 경험을 토대로 나 스스로 만족할 수 있는 결과물을 만들어 보고 싶다는 생각이 많이 든 프로젝트였다.

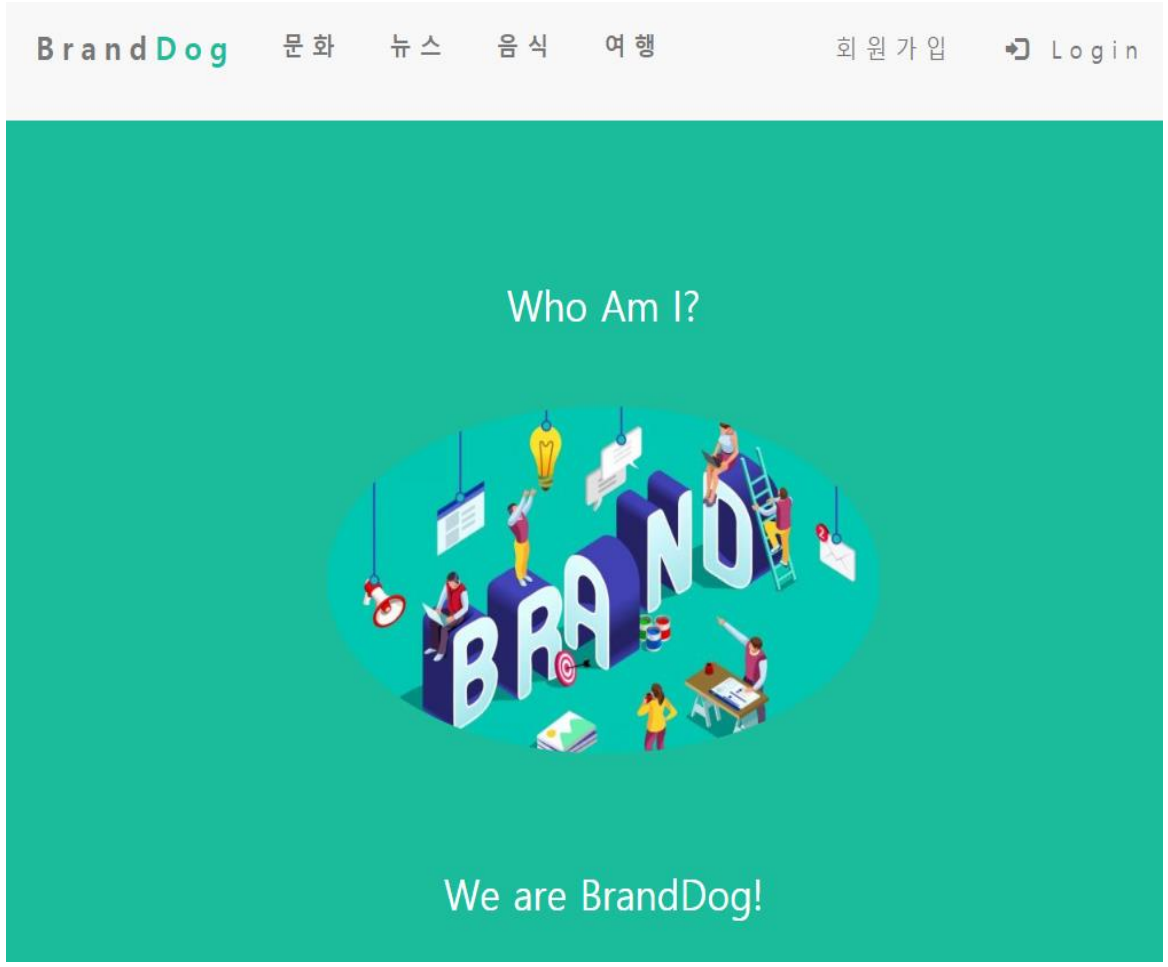


4.2 BrandDog

4.2 Spring Framework Project

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog



- 프로젝트 명 : BrandDog
- 프로젝트 기간 : 2020.04.03 ~ 2020.04.23
- 프로젝트 내용 :

  4가지 모듈에 따라 데이터 수집, 정제, 분석 후 사용자에게 유용한 정보를 제공할 수 있는 검색 사이트 개발

- 맡은 역할 : '**여행**' 모듈 개발 담당
- Github :
  - https://github.com/angelica127/Branddog
  - https://github.com/angelica127/Branddog/tree/master/Source/brandproject/src/main/java/com/branddog/travel

4.2.1 개발 환경

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog

분류	Resource	비고
Back – End	Java	JDK 1.8
	JSP	3.1
	Python	Anaconda3 5.2.0
	Framework	Spring Framework (Spring MVC Project)
Front – End	HTML	HTML5
	CSS	CSS3
	Bootstrap	Bootstrap3
	Sitemesh	
	JavaScript	JQuery
WAS	Apache Tomcat	9.0
DBMS	Oracle	12c
Version Control	Apache Subversion	

4.2.2 E.R.D

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog

BrandDog(ORACLE)

Member

회원

PK	AI	FK	Null	Logical Name	Name	Type
✔	✔			번호	No	INT
			✔	아이디	ID	VARCHAR2(60)
			✔	이름	Name	VARCHAR2(60)
			✔	비밀번호	PW	VARCHAR2(60)

TravelSearchTrend

여행_검색트렌드

PK	AI	FK	Null	Logical Name	Name	Type
✔	✔			번호	No	INT
			✔	시작 날짜	startDate	VARCHAR2(100)
			✔	끝 날짜	endDate	VARCHAR2(100)
			✔	단위	timeUnit	VARCHAR2(10)
			✔	주제어	title	VARCHAR2(300)
			✔	검색어	keywords	VARCHAR2(300)
			✔	데이터	data	VARCHAR2(400)

TravelReWord

여행_관련어

PK	AI	FK	Null	Logical Name	Name	Type
✔	✔			번호	No	INT
			✔	주제어	title	VARCHAR2(300)
			✔	단어1 ~ 단어30	word1 ~ word30	VARCHAR2(300)

- Python으로 데이터 수집, 정제, 분석 후 DB에 저장
- 저장한 데이터를 가져와 시각화

각각의 시각화 도구에 따라 데이터가 달라지면서
매우 간단한 데이터 모델링이 도출됨

4.2.3 Java & Python 연동

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog

Client의 요청 → Python을 이용하여 데이터 수집, 정제, 분석

→ Spring Framework를 통해 Client에 응답 (데이터 시각화)

Java (Spring Framework)와 Python을 어떻게 연동할 지에 대한 고민

- 다양한 방법 Testing

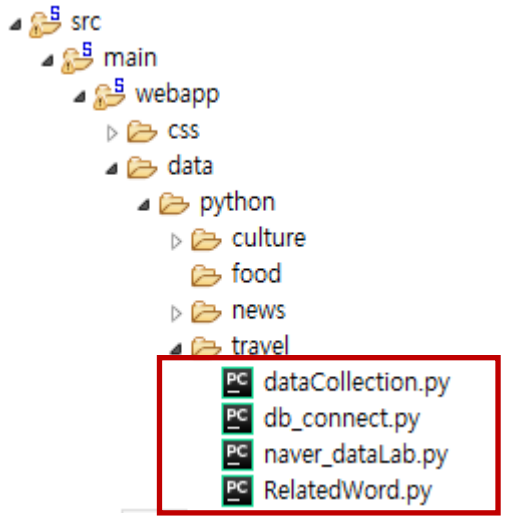
1. Django ----------→ 1. 깊이 있는 공부 부족
2. Jython ----------→ 2. Python 2.7 이후로 업데이트 X
3. JEP ----------→ 3. 많은 오류 발생
4. Java Class 이용

Testing 후 Java의 **ProcessBuilder Class**를 이용하기로 결정

4.2.3 Java & Python 연동

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog




```java
ProcessBuilder builder = new ProcessBuilder(command);
try {
    Process p = builder.start();

    BufferedReader stdOut = new BufferedReader(new InputStreamReader(p.getInputStream()));
    BufferedReader stdError = new BufferedReader(new InputStreamReader(p.getErrorStream()));

    while((result = stdOut.readLine()) != null){
        if(result.indexOf("no") >= 0) {
            strNo = result;
        }
    }

    while((result = stdError.readLine()) != null){
        System.out.println(result);
    }
```

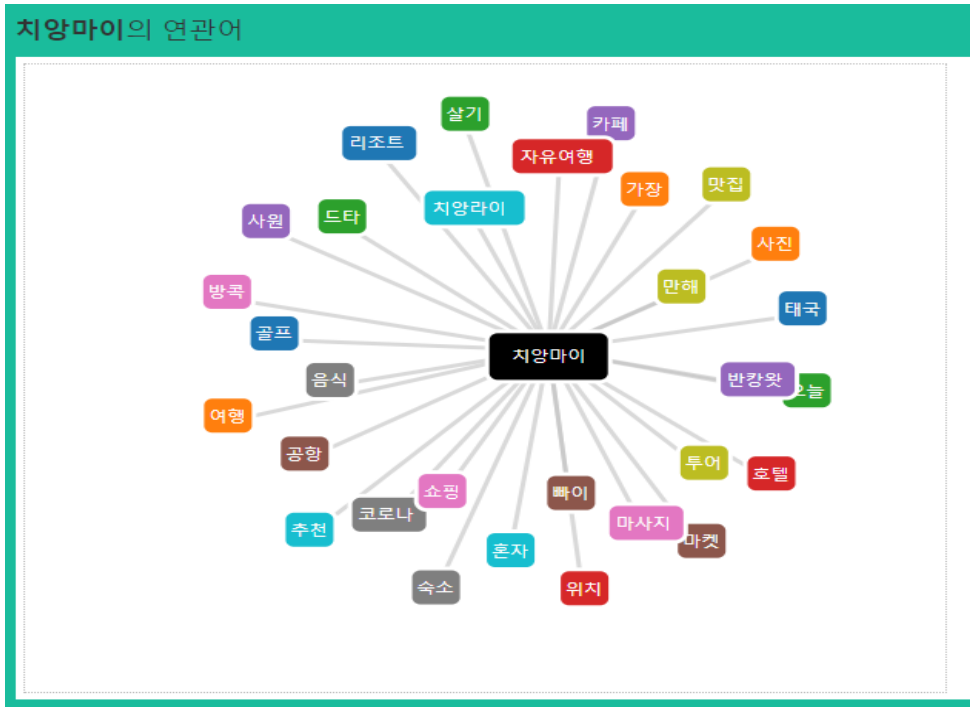
Python 코드를 Java의 **ProcessBuilder Class** 를 이용하여 명령 프롬프트로 코드를 실행한다.

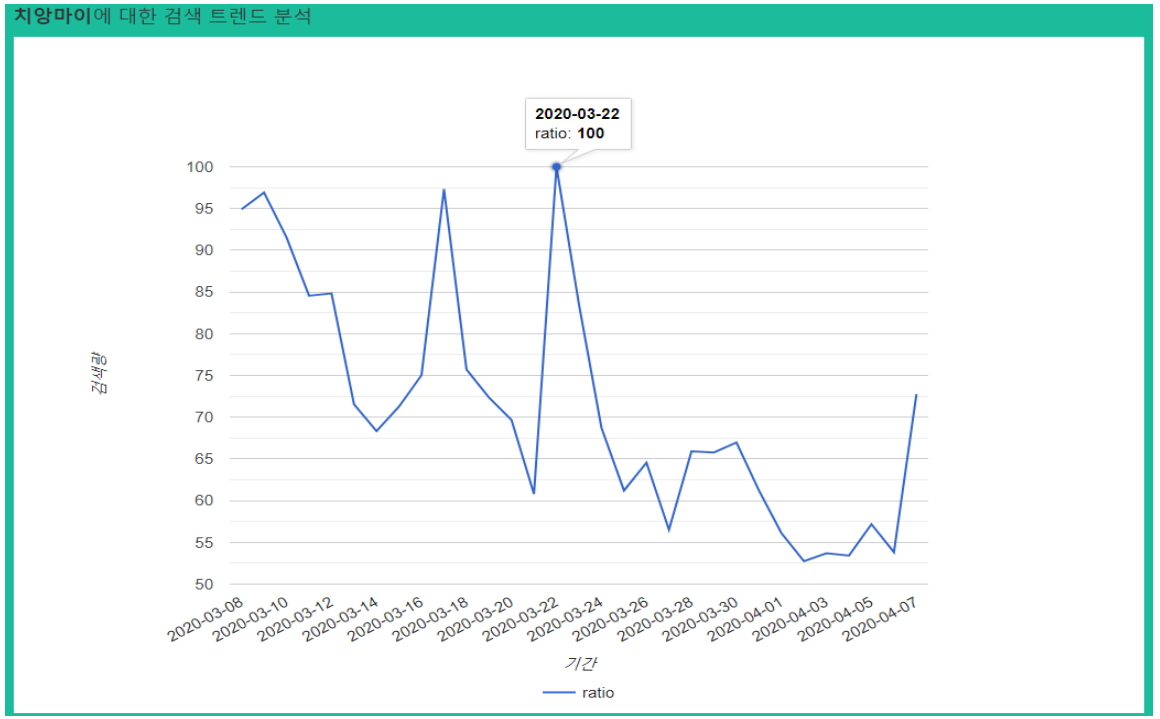
- Github :
  - https://github.com/angelica127/Branddog/blob/master/Source/brandproject/src/main/java/com/branddog/util/TravelPythonConnectUtil.java

4.2.4 개발 내용

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog





Crawling한 데이터를 **d3.js**를 이용하여 시각화

네이버 API를 이용하여 가져온 데이터 **구글차트**로 시각화

Github :https://github.com/angelica127/Branddog/blob/master/Source/brandproject/src/main/webapp/WEB-INF/views/travel/graph.jsp

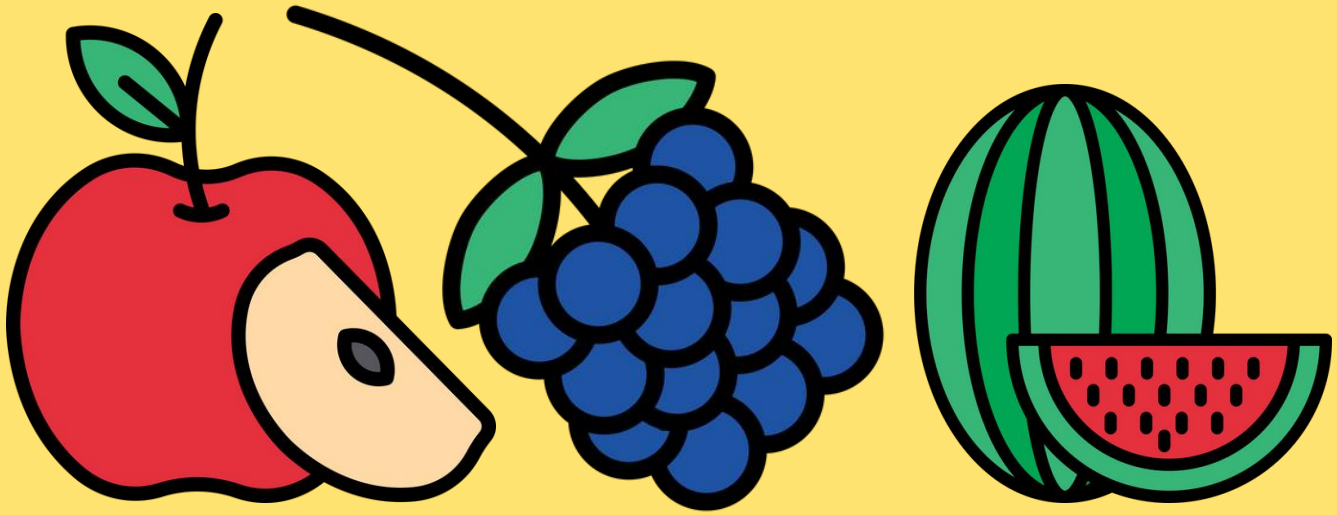
4.2.5 개발 후기

검색어를 활용한 트렌드 정보 제공 사이트 (Team Project)

BrandDog

- BrandDog Website 팀 프로젝트 후기

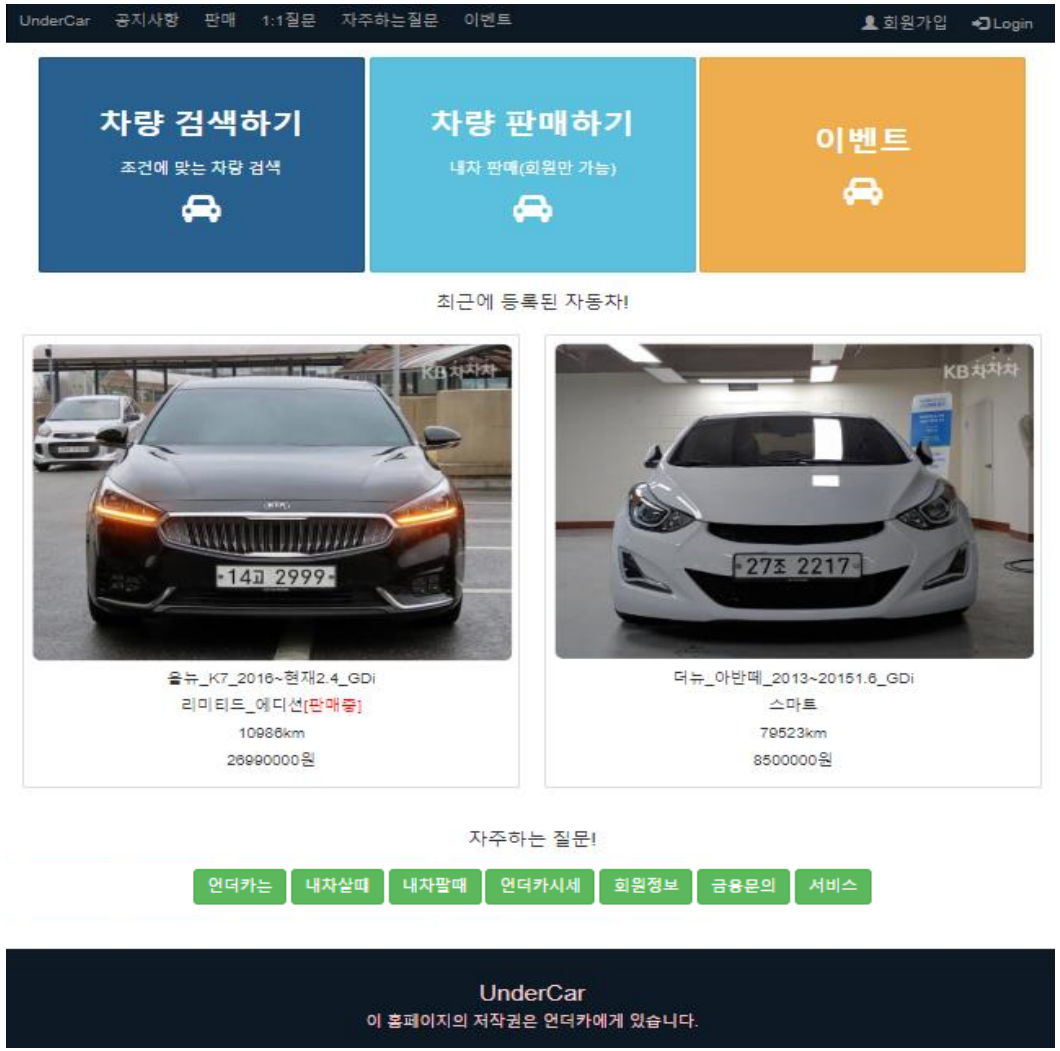
이번 프로젝트는 데이터를 가공하여 내가 원하는 데이터로 만드는 것에 중점을 두고 진행했다. 원하는 것만큼 자유자재로 데이터를 다루기 어려웠지만, 이번 프로젝트의 경험으로 앞으로 데이터를 다루는데 있어서 도움이 될 것 같다. 결과물을 만들어 내는데 있어서 Java와 Python을 연동하여 사용했던 부분도 흥미로웠다. 아직 실력이 부족해서 '효율적인' 코딩이 부족했는데, 지속적인 공부를 통해 보완해야 할 것 같다. 프로젝트를 진행하면서 팀원 간에 원하는 부분이 잘 소통이 되어 순조롭게 진행되어 즐겁게 마무리할 수 있었다.



4.3 UnderCar

4.3 JSP Web Project

중고차 중매 사이트 개발 (Team Project)



- 프로젝트 명 : **UnderCar**
- 프로젝트 기간 : 2019.12.24 ~ 2020.01.02
- 프로젝트 내용 :

  중고차와 관련하여 판매자와 구매자를 중매할 수 있는 사이트 개발

  JSP(Servlet)을 이용하여 MVC2 Pattern 구현
- 맡은 역할 : '**Message**' 모듈 개발 담당
- Github :
  - https://github.com/angelica127/Undercar
  - https://github.com/angelica127/Undercar/tree/master/undercarcom/src/com/undercar/message

4.3.1 개발 환경

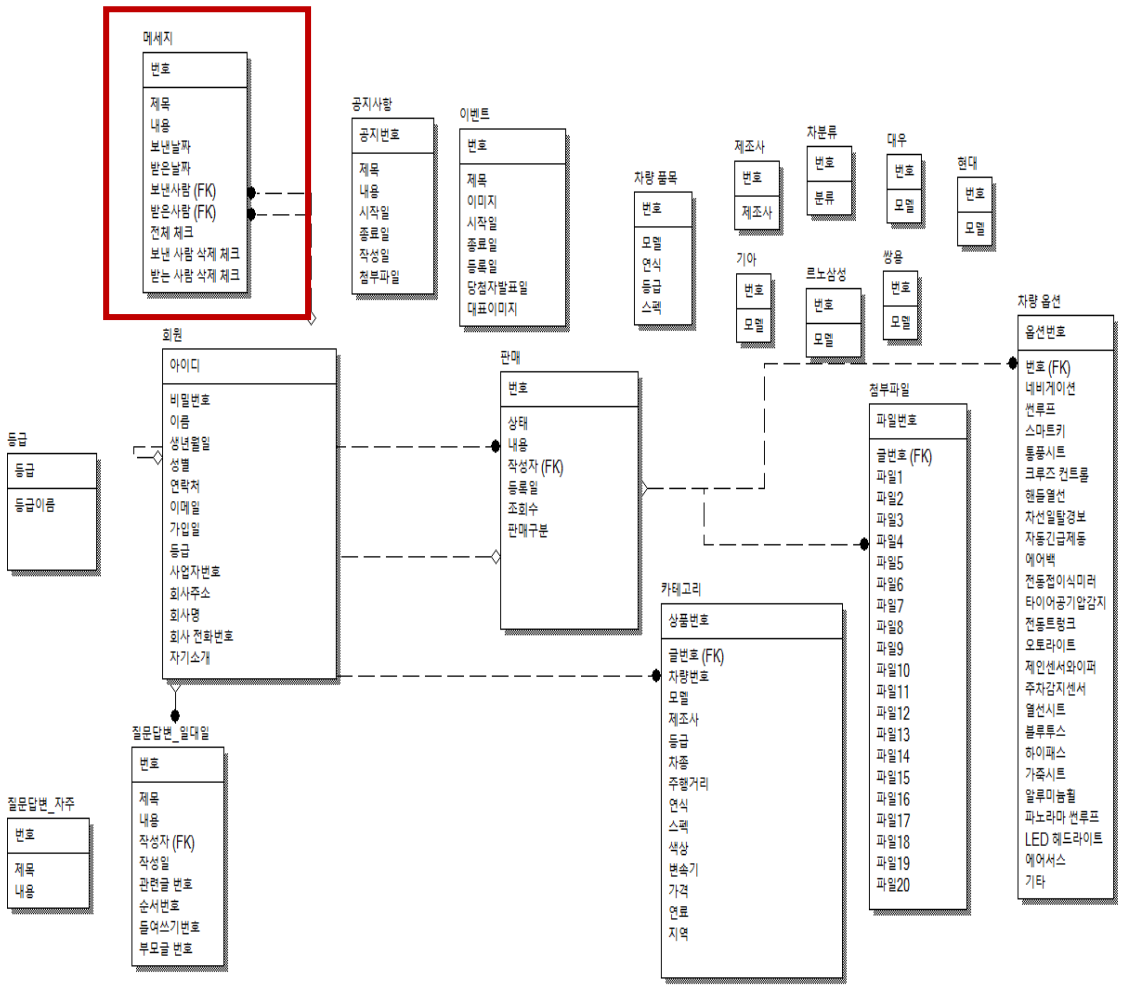
중고차 중매 사이트 개발 (Team Project)

분류	Resource	비고
Back – End	Java	JDK 1.8
	JSP	3.1
Front – End	HTML	HTML5
	CSS	CSS3
	Bootstrap	Bootstrap3
	Sitemesh	
	JavaScript	JQuery
WAS	Apache Tomcat	9.0
DBMS	Oracle	12c
Version Control	Apache Subversion	

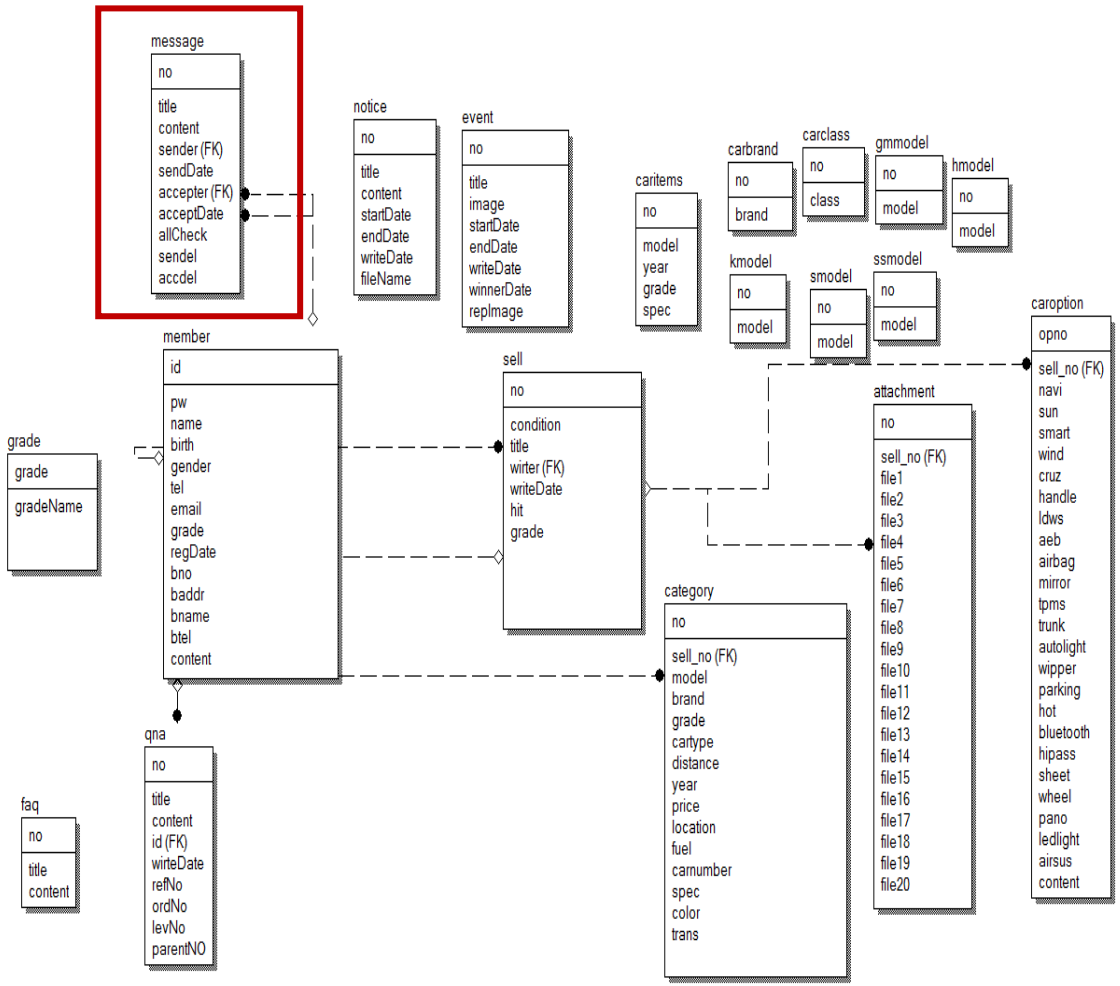
4.3.2 E.R.D

중고차 중매 사이트 개발 (Team Project)

<논리 설계>



<물리 설계>



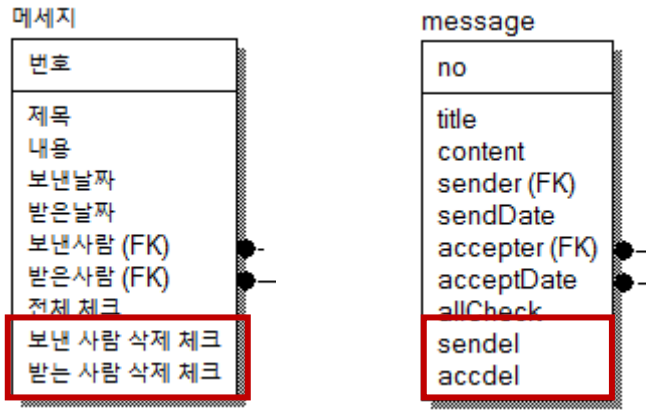
4.3.3 개발 설계

중고차 중매 사이트 개발 (Team Project)

Database 설계

코드 작업 전 '보낸 사람'과 '받는 사람' 중 한 쪽에서 메시지를 삭제 시 **둘 모두에게 메시지 삭제**되는 것을 어떻게 해결할지에 대해 고민

Database Table에 Column 2개 추가



상대방이 삭제 했는지, 안 했는지의 여부를 파악할 수 있는 칼럼을 추가하여 둘 모두 삭제 시 DB에서 최종 삭제 될 수 있도록 처리

```
// 글번호 - String을 int로 바꿈
int getNo = Integer.parseInt(str);
// 글번호에 맞는 MessageDTO 가져오기
MessageDTO msgDto = getDTO(request, getNo);
// session의 아이디와 받은 사람 아이디 일치하는 지 확인
if (loginDto.getId().equals(msgDto.getAccepter())) {
    // sendel이 0인지 확인 -> 0이면 최종 삭제 x
    if (msgDto.getSendel() == 0) {
        // accdel변수를 1로 바꿈, 최종 삭제 x
        Execute.service(updateStatusService, getNo, 1);
```

4.3.4 개발 내용

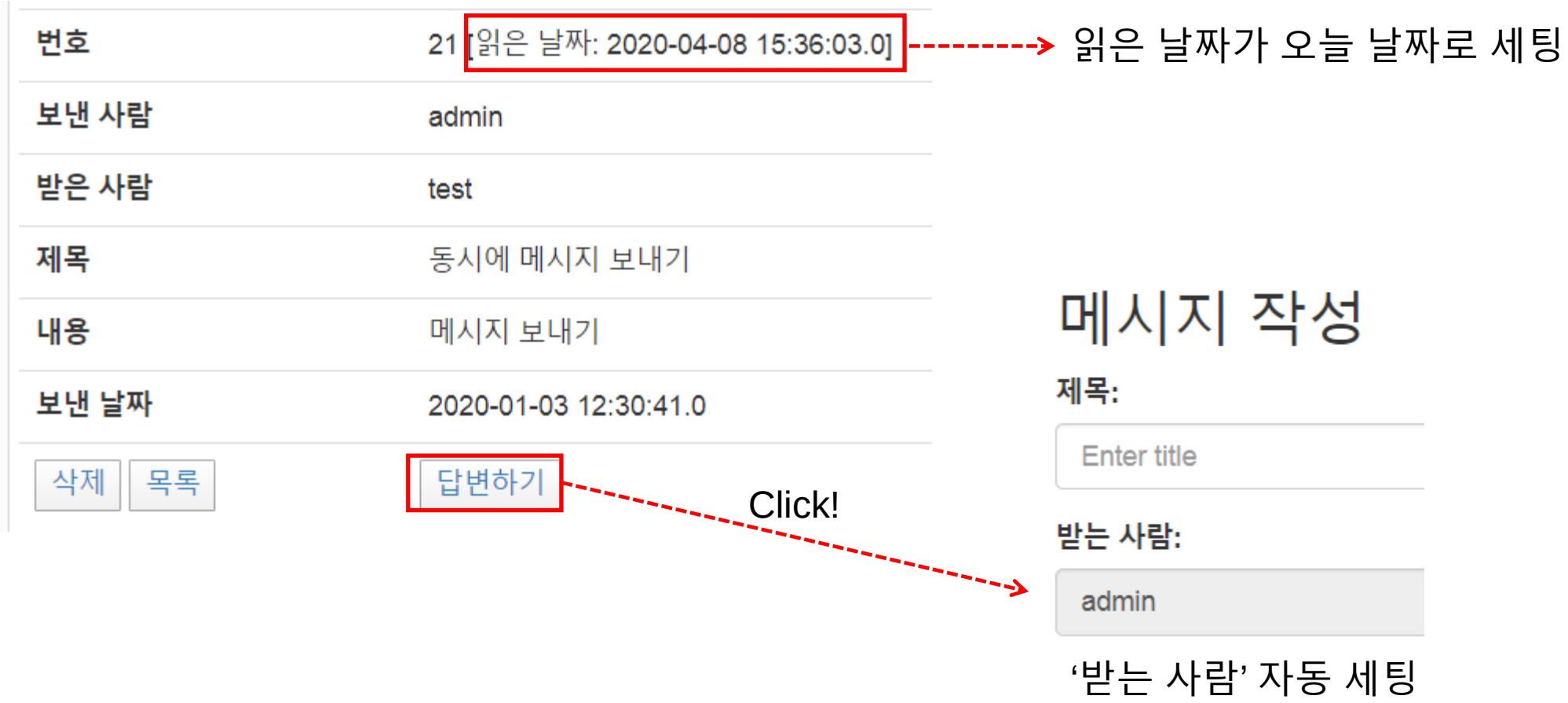
중고차 중매 사이트 개발 (Team Project)

Github :

1. https://github.com/angelica127/Undercar/tree/master/undercarcom/src/com/undercar/message/controller
2. https://github.com/angelica127/Undercar/blob/master/undercarcom/WebContent/WEB-INF/views/message/list.jsp

4.3.4 개발 내용

중고차 중매 사이트 개발 (Team Project)



읽은 날짜가 오늘 날짜로 세팅

'받는 사람' 자동 세팅

4.3.4 개발 내용

중고차 중매 사이트 개발 (Team Project)

	..	CONTENT	SENDDATE	ACCEPTDATE	SENDER	ACCEPTER
1	26	vjh...	20/01/03	(null)	admin	agreetest1
2	25	vjh...	20/01/03	(null)	admin	test
3	24	vjh...	20/01/03	(null)	admin	agreetest
4	23	vjh...	20/01/03	(null)	admin	btest
5	22	메시...	20/01/03	(null)	admin	btest
6	21	메시...	20/01/03	(null)	admin	test
7	20	공지...	20/01/03	(null)	admin	agreetest1
8	19	공지...	20/01/03	(null)	admin	test
9	18	공지...	20/01/03	(null)	admin	agreetest
10	17	공지...	20/01/03	(null)	admin	btest
11	16	자세...	20/01/03	20/01/03	admin	test
12	15	판매...	20/01/03	20/01/03	admin	test
13	14	이번...	20/01/03	20/01/03	test	admin
14	12	새로...	20/01/03	(null)	admin	agreetest1
15	11	새로...	20/01/03	(null)	admin	test
16	9	새로...	20/01/03	(null)	admin	agreetest
17	8	새로...	20/01/03	(null)	admin	btest
18	6	회원...	20/01/03	(null)	admin	agreetest1
19	5	회원...	20/01/03	20/01/03	admin	test
20	3	회원...	20/01/03	(null)	admin	agreetest
21	2	회원...	20/01/03	(null)	admin	btest

Accepter가 메시지를 누르면 본 것으로 처리
오늘 날짜로 AcceptDate가 바뀌게 된다.

4.3.5 개발 후기

중고차 중매 사이트 개발 (Team Project)

- UnderCar Website 팀 프로젝트 후기

처음으로 프로그래밍을 배우면서 처음에는 너무 어려웠다. 시간이 지나면서 짧은 시간에 많은 것을 배워 그것들을 정리할 시간들이 필요했는데 프로젝트를 진행하는 기간 동안 배운 것을 정리하고, 더 다양한 것들을 시도할 수 있는 시간이 되어 좋았다. 부족한 게 많았는데 팀원들이 서로 서로 도와가며 부족하지만 결과물이 나와 재밌었고, 부족한 것을 느낀 만큼 더 열심히 공부해야겠다는 생각이 들었다.

kimjwon127@gmail.com
https://github.com/angelica127

THANK YOU