

포트폴리오 2018

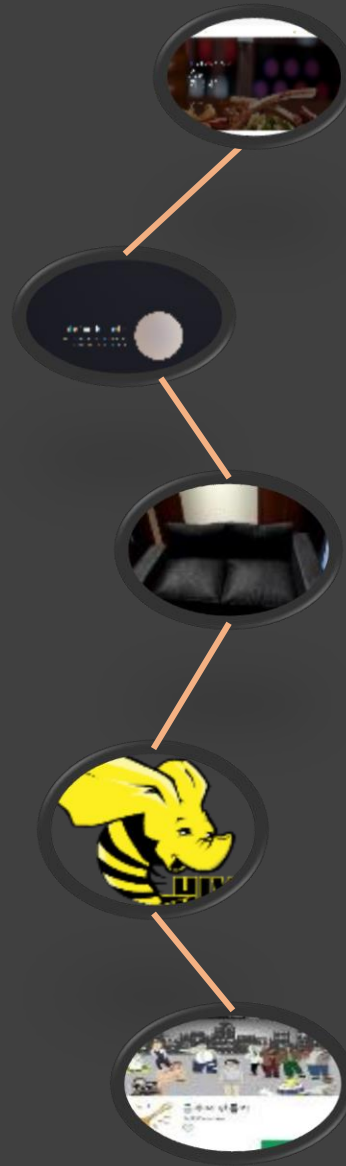
권혁내



Index



Kwonhyucknae



3page -**Spring**을 이용한
웹 어플리케이션

12page -**Beacon**을 이용한
대학교 출결 어플리케이션

21page -증강현실을 이용한
가구배치 어플리케이션

41page -**Hadoop**의 **Hive**를 이용한
빅데이터 처리

49page -유니티를 이용한
2D 게임 어플리케이션

Spring Framework를 이용한 웹어플리케이션



클릭시 깃허브와
연결됩니다.



Github 웹 어플리케이션 부분.

WEB



DB



Framework



Spring Framework를 이용한 웹 어플리케이션

개발기간/개발인원

2018.03~2018.04/1명

분야

웹 어플리케이션

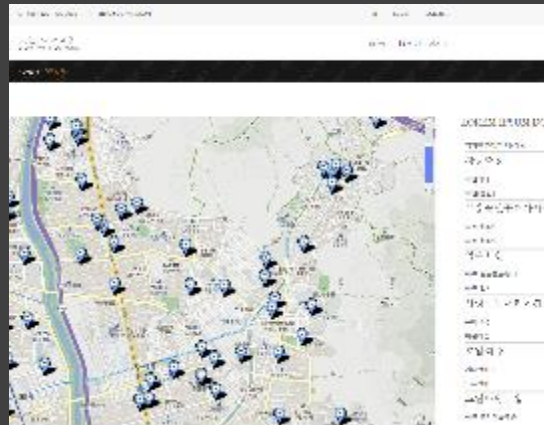
맡은 역할

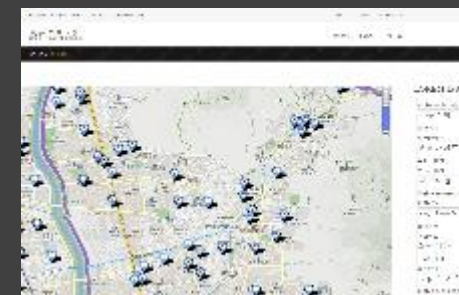
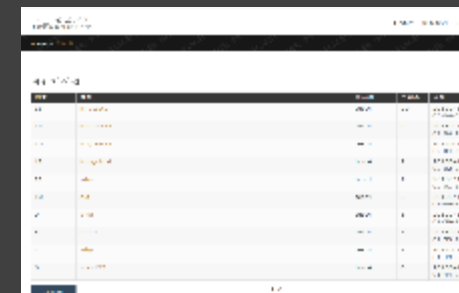
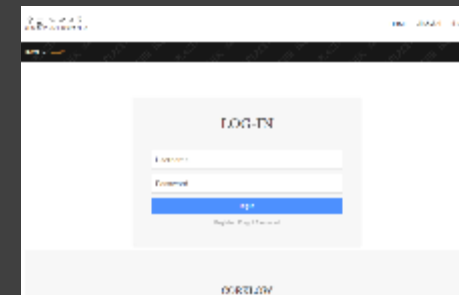
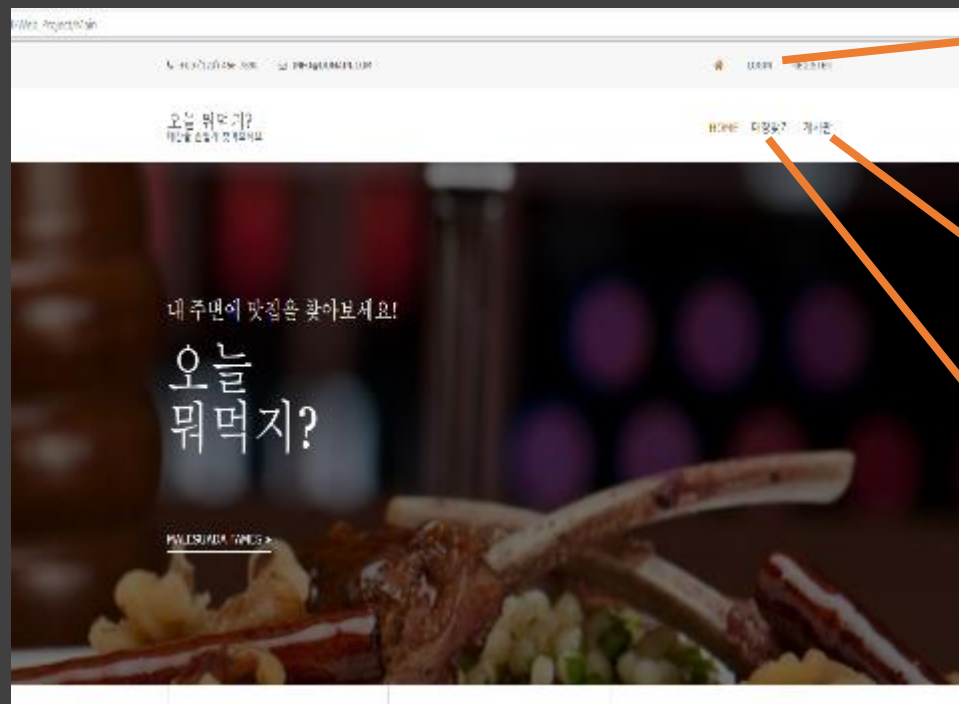
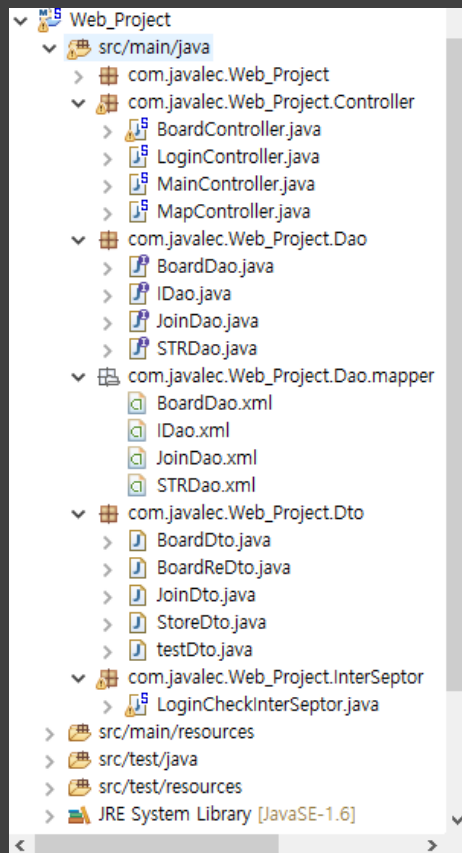
스프링MVC레이아웃 구축, 웹 제작,데이터베이스 관리

플랫폼/ 개발언어 / 개발환경

Windows 10/ Java , JSP , Html, CSS , jquery, Ajax, SQL / Oracle , Eclipse, Spring Framework ,Mybatis

사용자의 현재 위치를 위도와 경도로 받아 지도로 표시한 후 현재 위치와 가까운 매장을 마커로 표시해줬습니다.
매장 데이터는 공공데이터 포털 사이트에서 받아온 데이터를 오라클에 저장 후 Mybatis 를 이용해 쿼리문을 작성 후 데이터를 받아와 사용 했습니다.
그 외 게시판 기능에서는 게시판 목록과 글쓰기 글 수정, 댓글 기능을 구현 했습니다.





웹의 첫 화면입니다.

스프링 MVC 구조를 사용했습니다. 요청을 받는 컨트롤러에서는 모델 역할을 하는 Dao 에서 데이터를 받아올 수 있고 받아온 데이터를 뷰 역할을 하는 JSP에 보낼 수 있습니다.

모델 역할을 하는 Dao 패키지에서는 인터페이스를 만들고 Dao.Mapper 패키지에서 Mybatis를 이용해 데이터를 받아왔습니다. 받아온 데이터는 Dto 패키지로 테이블의 속성에 맞게 받아왔습니다.

메인화면 > 게시판 > 자유게시판

자유게시판

번호	제목	글쓴이	조회수	날짜
15	imgtest2	test4	20	20180418-03-04-02
14	imgtest11	test4	4	20180418-02-04-19
13	imgtest11	test4	3	20180418-02-04-09
12	imgtest	test4	2	20180418-02-02-50
11	why	test4	1	20180418-02-01-38
10	did	test4	2	20180418-01-59-53
9	ajwl	test4	1	20180418-01-57-27
8	hmm	test4	1	20180418-01-53-25
7	why	test4	1	20180418-01-49-53
6	zzzz123	test4	0	20180418-01-44-27

글쓰기

1 2

메인화면 > 게시판 > 자유게시판

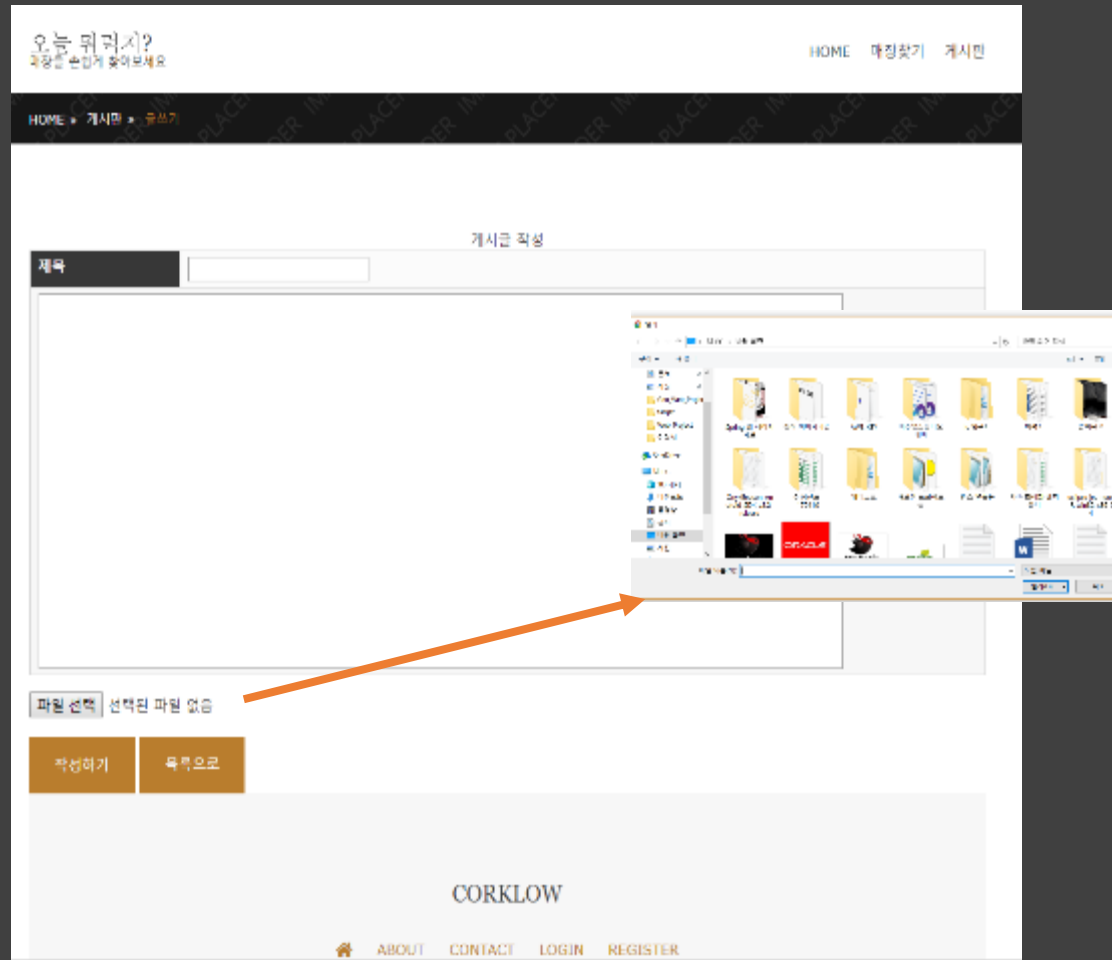
자유게시판

번호	제목	글쓴이	조회수	날짜
5	zzzz	test4	0	20180418-01-42-57
4	test	test4	0	20180418-01-42-02
3	test	test4	0	20180418-01-41-52
2	zzz	test4	54	20180415-20-39-15
1	zzz	test4	11	20180415-20-33-47

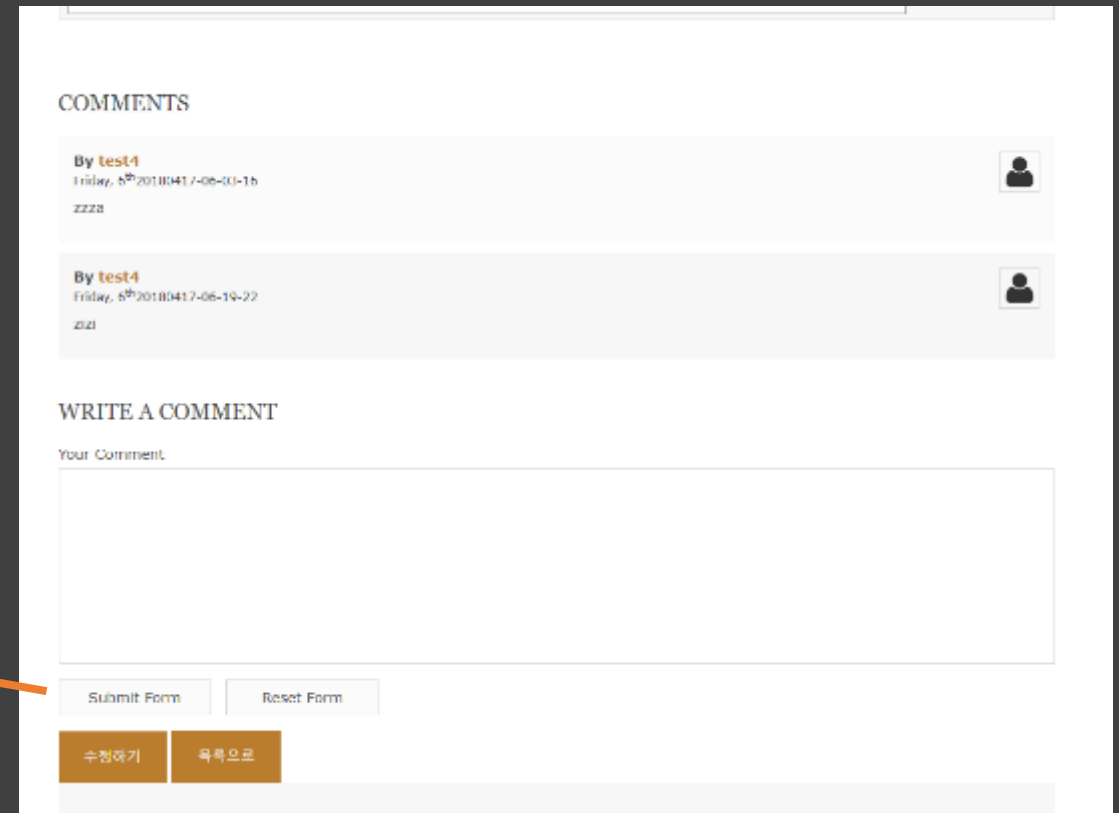
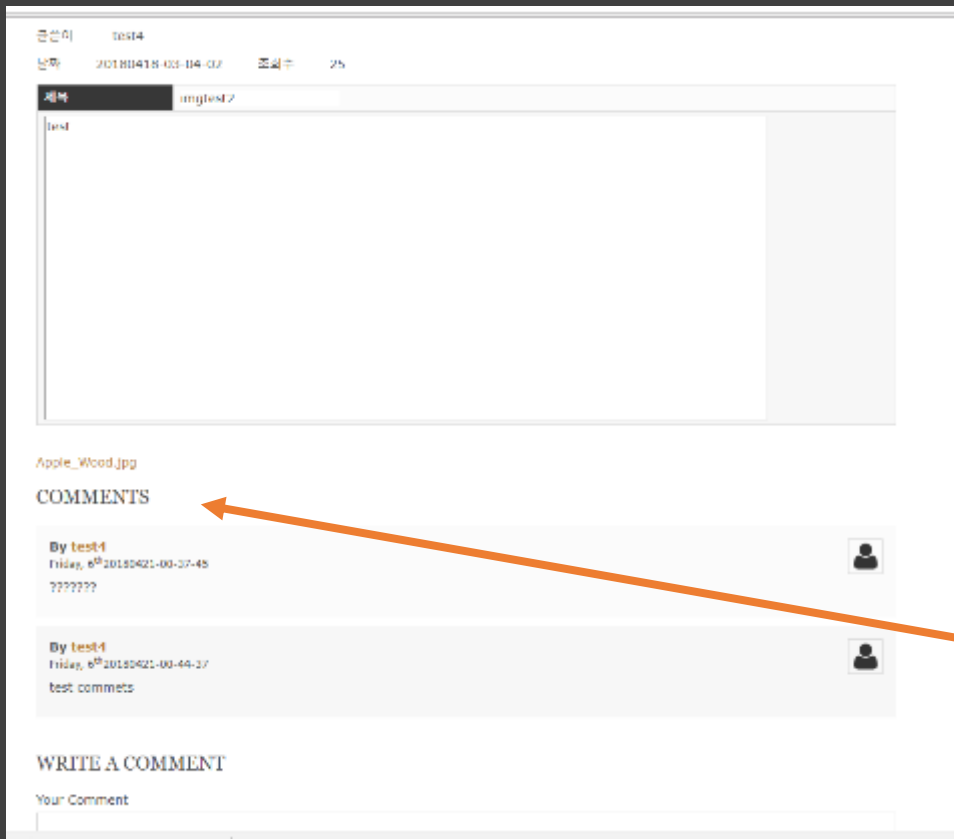
글쓰기

1 2

게시글 전체 목록입니다. 글쓰기 기능과 글을 볼 수 있고 한 페이지에 글은 10개가 표시 됩니다. 10개가 넘으면 다음 페이지 숫자가 생기고 숫자를 클릭하면 페이지 번호를 넘겨줘서 다음 페이지가 화면에 나타나도록 했습니다.



게시글 작성 페이지 입니다. 로그인 되어 있어야 작성이 되고 안 되어있으면 인터셉터를 통해 로그인 페이지로 이동합니다. 게시글은 제목과 글을 작성할 수 있고 파일을 업로드 할 수 있습니다. 파일은 서버에 저장되고 DB에는 파일명을 저장했습니다.



게시글 읽기 페이지 입니다. 제목과 글을 볼 수 있고 , 업로드된 파일이 있다면 파일명을 클릭시 다운로드 할 수 있습니다. 게시글에 댓글을 작성 할 수 있고 게시글을 작성한 사람과 세션의 아이디 값이 같다면 게시글을 수정할 수 있습니다.

HOME > 게시판 > 수정

게시글 작성

제목

imgtest2

test

Apple_Wood.jpg
파일 선택 선택된 파일 없음

삭제

작성하기 목록으로

CORKLOW

게시글 수정 페이지 입니다.

파일에 대해선 경우의 수를 나눠줬습니다.

1. 업로드 된 파일이 없을 때 파일을 업로드하는 경우
2. 업로드 된 파일이 없을 때 파일을 업로드 하지 않는 경우
3. 업로드 된 파일이 있을 때 아무것도 건들지 않은 경우
4. 업로드 된 파일이 있을 때 파일을 삭제하는 경우
5. 업로드 된 파일이 있을 때 새로운 파일을 올리는 경우

파일이 변경 될 경우와 파일을 삭제 할 경우에는 서버에 저장되어 있는 원래 파일을 삭제 한 후 새로운 파일을 경로에 저장했습니다.

Beacon을 이용한 대학교 출결 어플리케이션



클릭시 깃허브와
연결됩니다.



Github 안드로이드 부분.



Github 서버 부분.

i-beacon



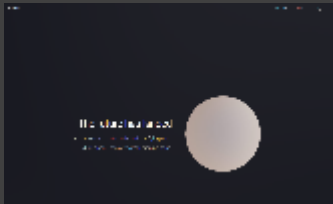
Android



Server



Web



Beacon을 이용한 대학교 출결 어플리케이션

개발기간/개발인원

2017.04~2017.06/1명

분야

IOT(Beacon), 서버, 웹, 앱

맡은 역할

비콘과 앱의 연동, 서버 구축, 웹 페이지 구축, 앱 제작

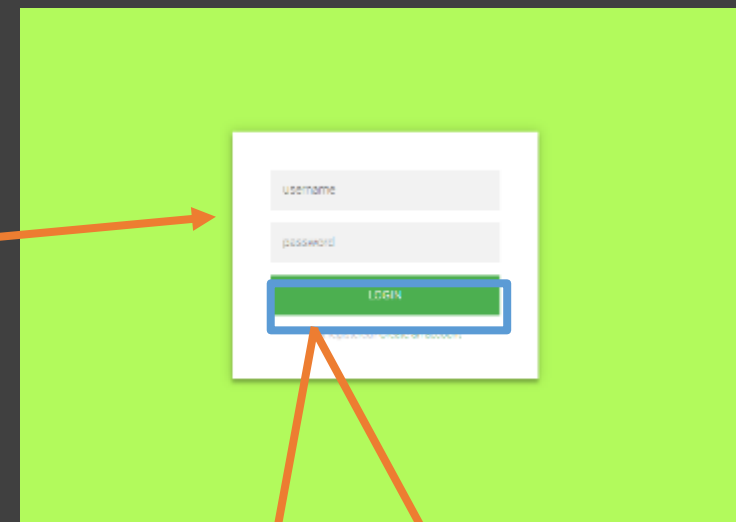
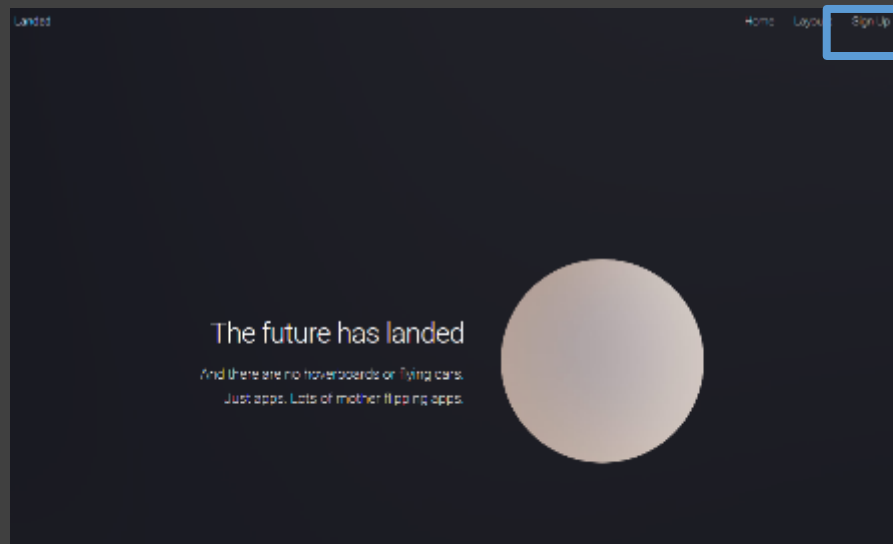
플랫폼/ 개발언어 / 개발환경

Windows 10/ Java , SQL, Html / Android Studio, Mysql , AWS, Eclipse , ERMaste , Beacon estimate Library

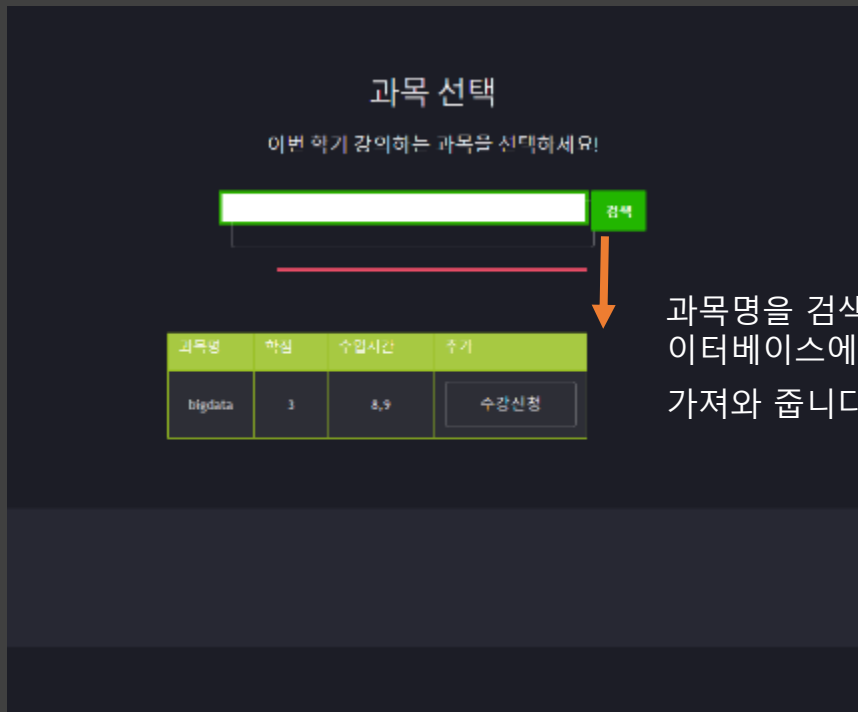
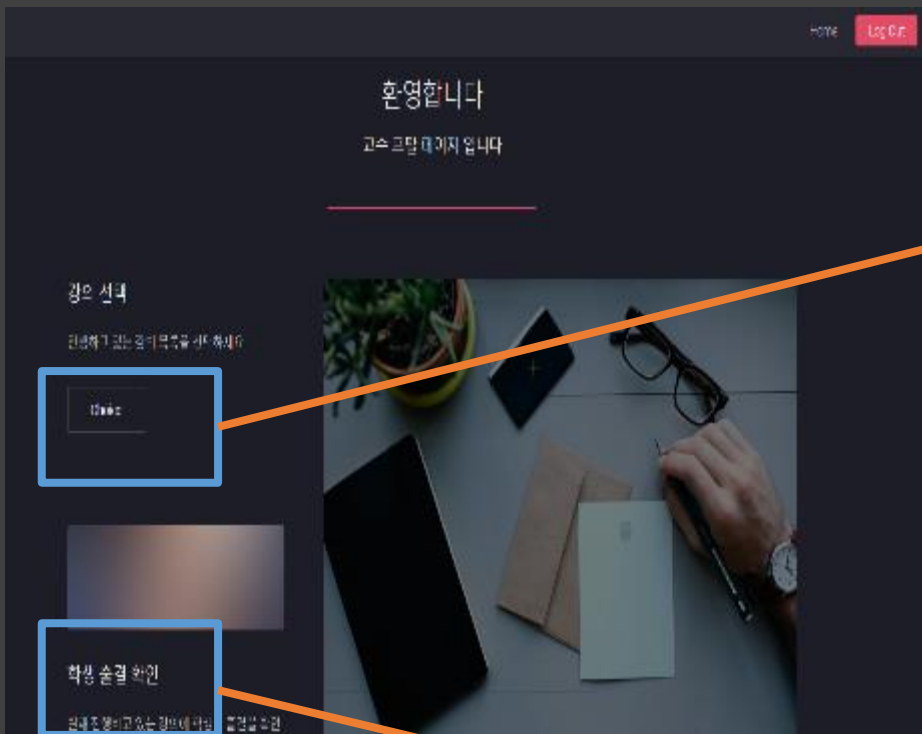
블루투스를 켜진 모바일 기기와 비콘과의 통신을 통해 학생이 강의실에 들어가면 어플리케이션을 통해 스스로 출석체크를 하는걸 목표로 했습니다.

AWS를 이용해 서버를 구축하고 그 안에 Mysql을 설치해 웹과 앱에서 데이터를 사용 할 수 있게 했습니다.

웹에서는 출결 현황 및 수강신청을 할 수 있고 앱에서는 학생 개개인이 출석을 할 수 있고, 수강 신청한 시간표와 출결 현황을 볼 수 있게 했습니다.

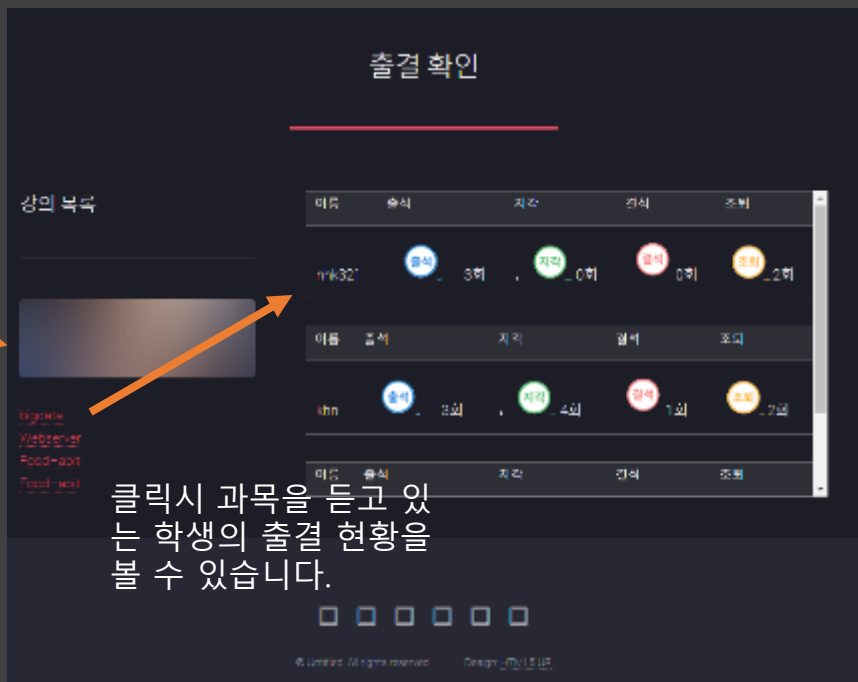


웹의 첫 화면입니다.
로그인 페이지로 넘어갈 수 있고,
등록한 정보에 따라서 로그인을 하면
학생페이지 또는 교수페이지로 이동합니다.

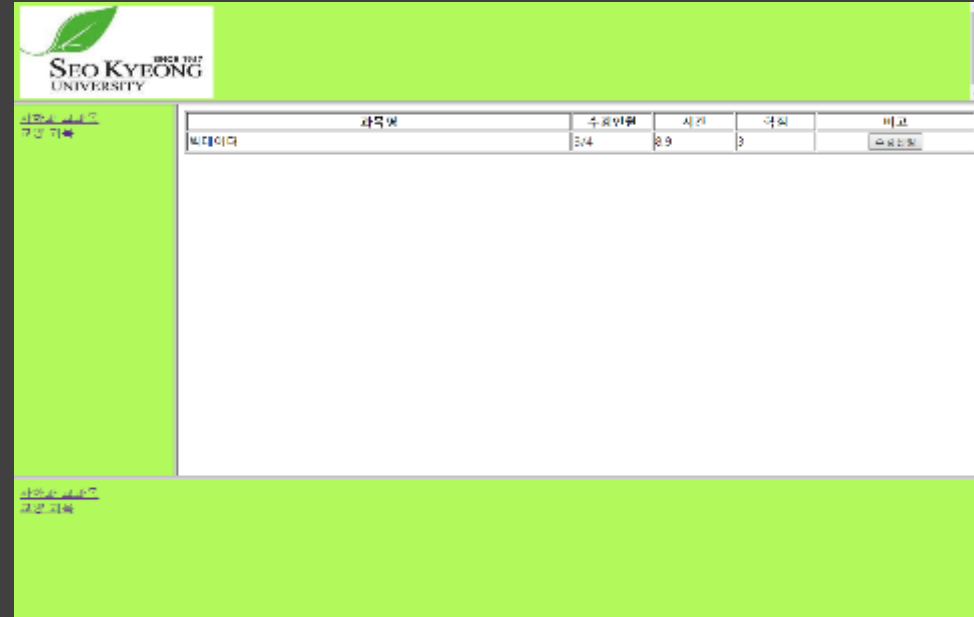
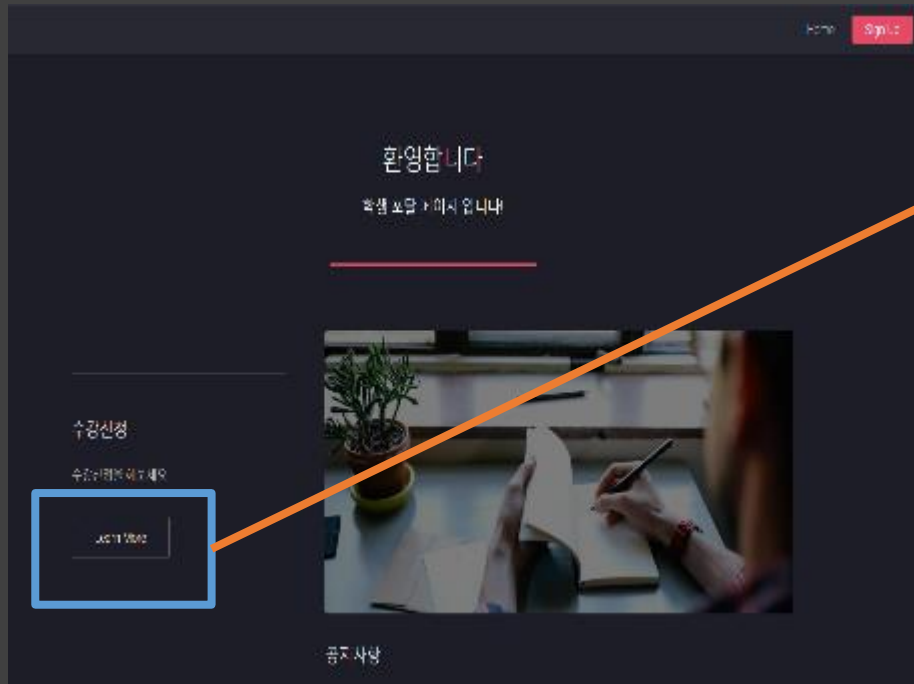


과목명을 검색하면 데이터베이스에서 정보를 가져와 줍니다.

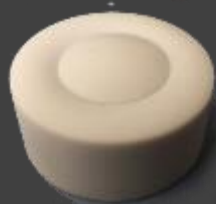
웹에서 교수님 아이디로 로그인할 때 보여지는 페이지입니다. 현재 진행하고 있는 강의를 선택할 수 있고, 그 강의를 듣고있는 학생의 출결 현황을 볼 수 있습니다.



클릭시 과목을 듣고 있는 학생의 출결 현황을 볼 수 있습니다.



웹에서 학생 아이디로 로그인할 때 보여지는 페이지입니다. 간단한 수강신청 기능을 가지고 있습니다.



수업이 있고
비콘과의 거리가 먼 경우



수업이
없는 경우



i-beacon 과 어플리케이션과
연결을 해졌습니다.
비콘과 모바일 기기간의 거리를
측정해서 경우의 수 에 따라서
이벤트를 처리했습니다.

i-beacon개발시에는
Estimote 라이브러리를 이용했
습니다.



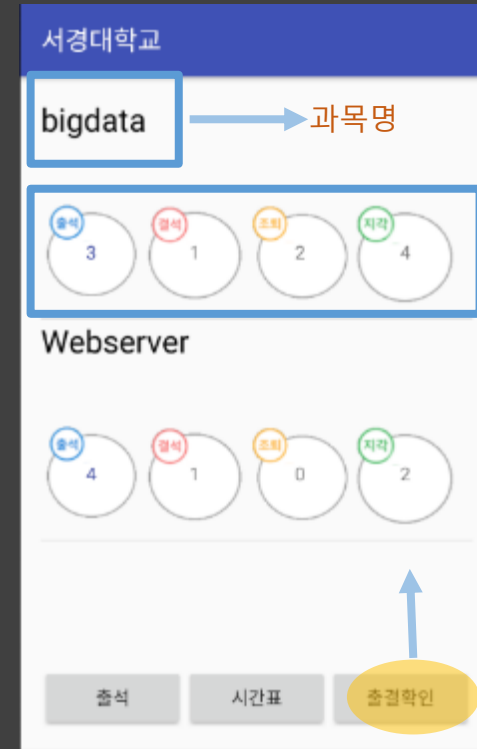
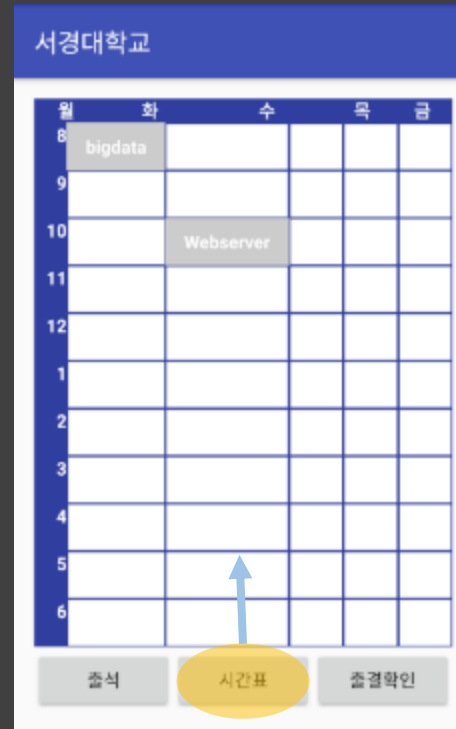
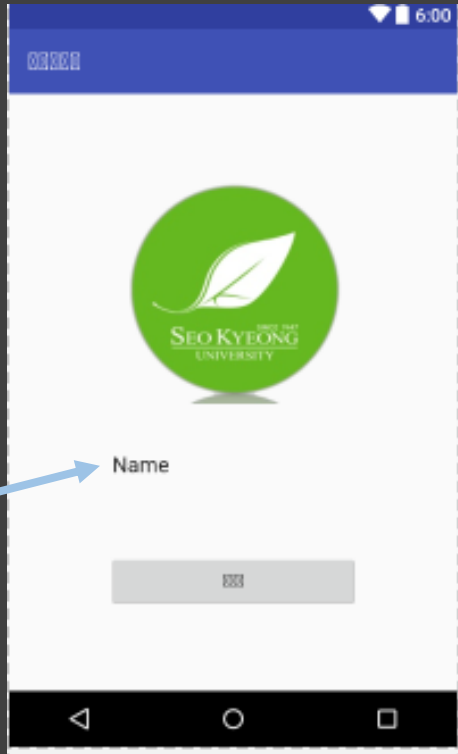
수업이 있고
비콘과의 거리가 가까운 경우



버튼 클릭시
출석이 완료
됩니다.



웹에서 만든
아이디로
로그인

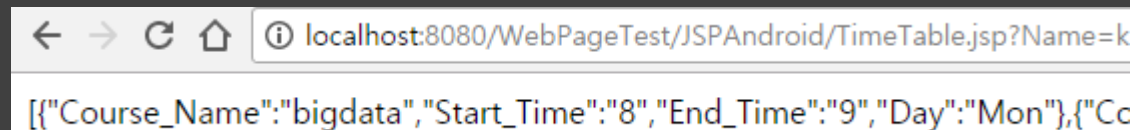
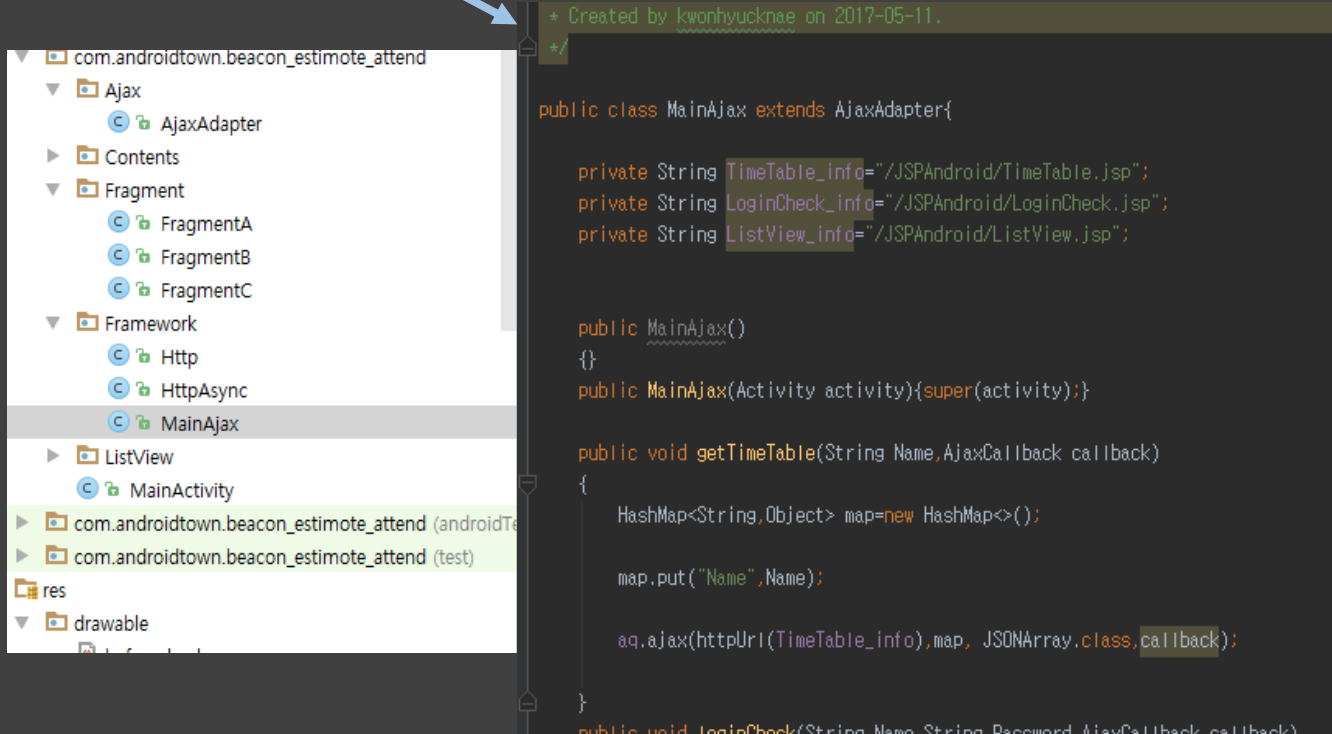


출결 현황

웹에서 회원가입한 아이디로 모바일에서 로그인을 하게 되면 사용자의 정보를 데이터베이스에
서 받아서, 그 사용자가 등록한 수강신청에 따라 시간표를 자동으로 생성해서 보여줬습니다.
출결현황을 클릭하면 듣고 있는 과목의 출결 상태를 볼 수 있습니다.

Ajax로 통신

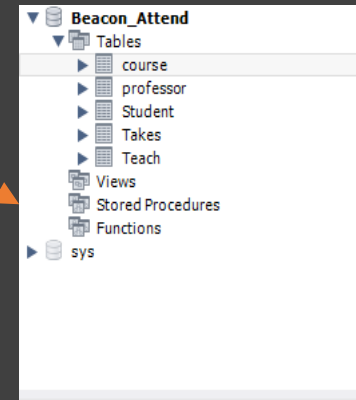
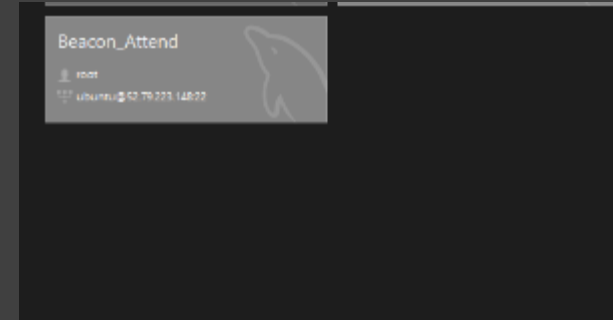
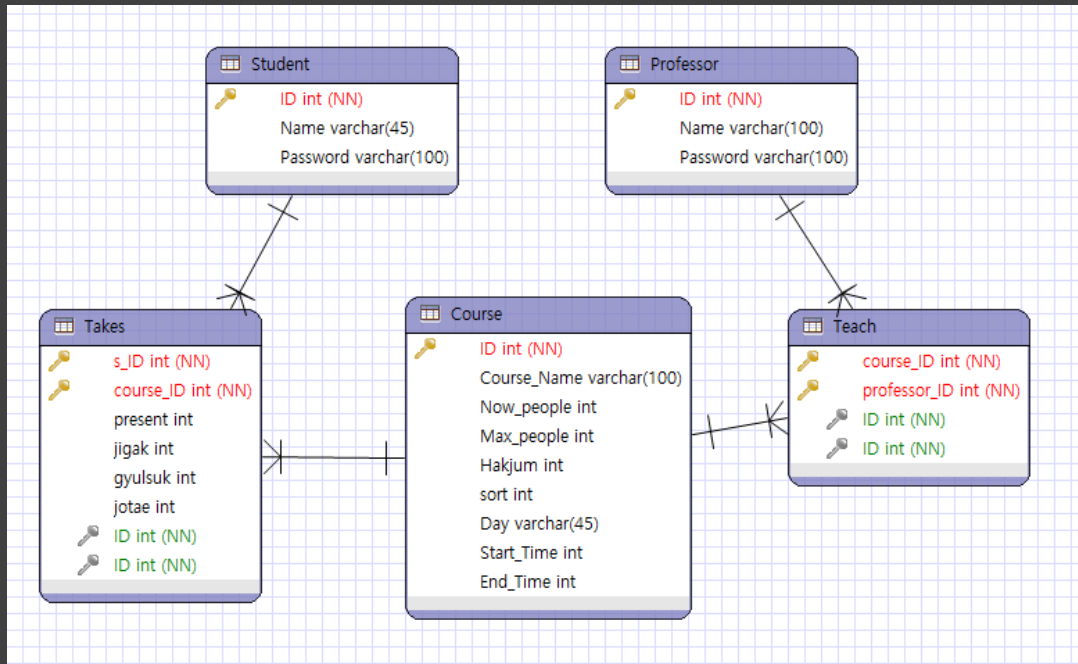
JSON 파싱



```
test = object.getJSONObject(i).get("Course_Name").toString();
sta = object.getJSONObject(i).get("Start_Time").toString();
end = object.getJSONObject(i).get("End_Time").toString();
int startt = Integer.parseInt(sta);
int endt = Integer.parseInt(end);
```

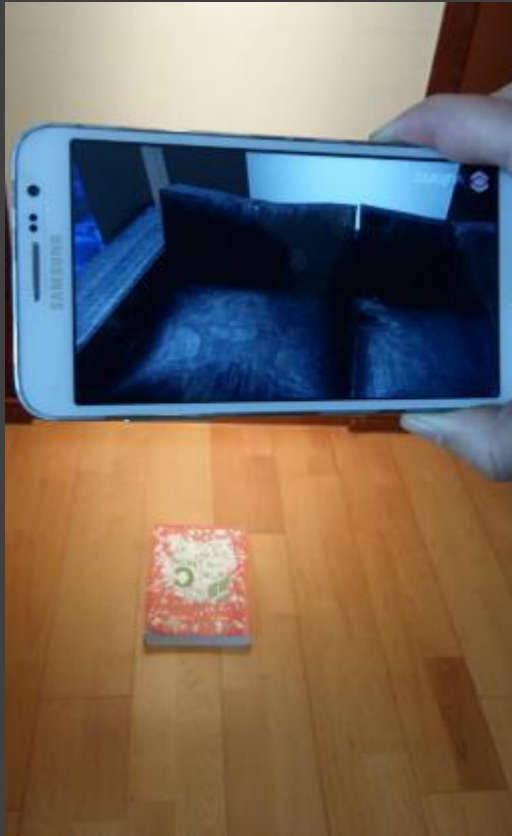
Jsp와의 통신을 Ajax로 사용해봤습니다.
클라이언트에서 데이터를 보낼때 JSON형식으로 파싱해서 보내고
Jsp에서도 데이터를 JSON 형식으로 바꿔서 보내줬습니다.

AWS서버에 설치되어있는 Mysql로 외부접속

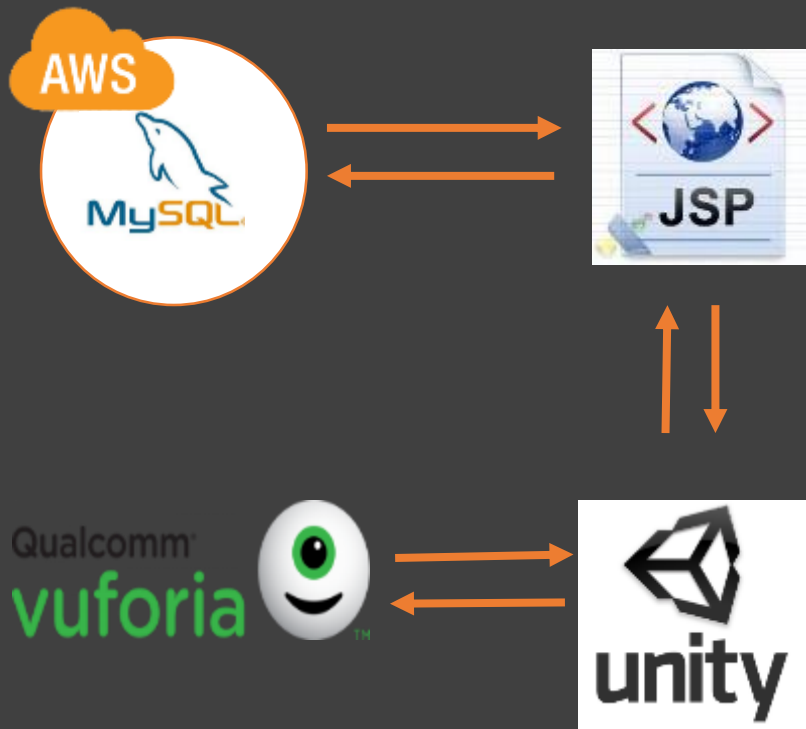


이클립스의 ERMaster 로 데이터베이스를 설계 후,
AWS 서버에 설치되어있는 Mysql로 외부접속 후
데이터베이스 스키마를 만들어졌습니다.

AR을 이용한 가구배치 어플리케이션



Github 유니티 부분.



AR을 이용한 가구배치 어플리케이션

개발기간/개발인원

2017.03~2017.05/1명

분야

AR, 서버, 앱

맡은 역할

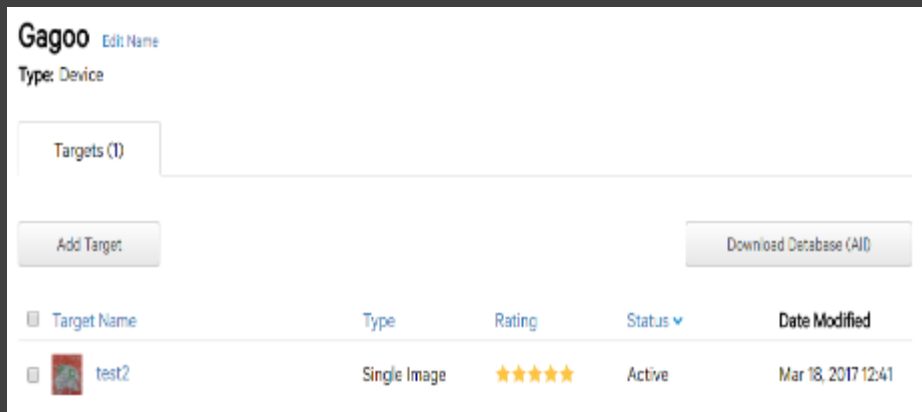
3D모델 증강, 서버 구축, 앱 제작

플랫폼/ 개발언어 / 개발환경

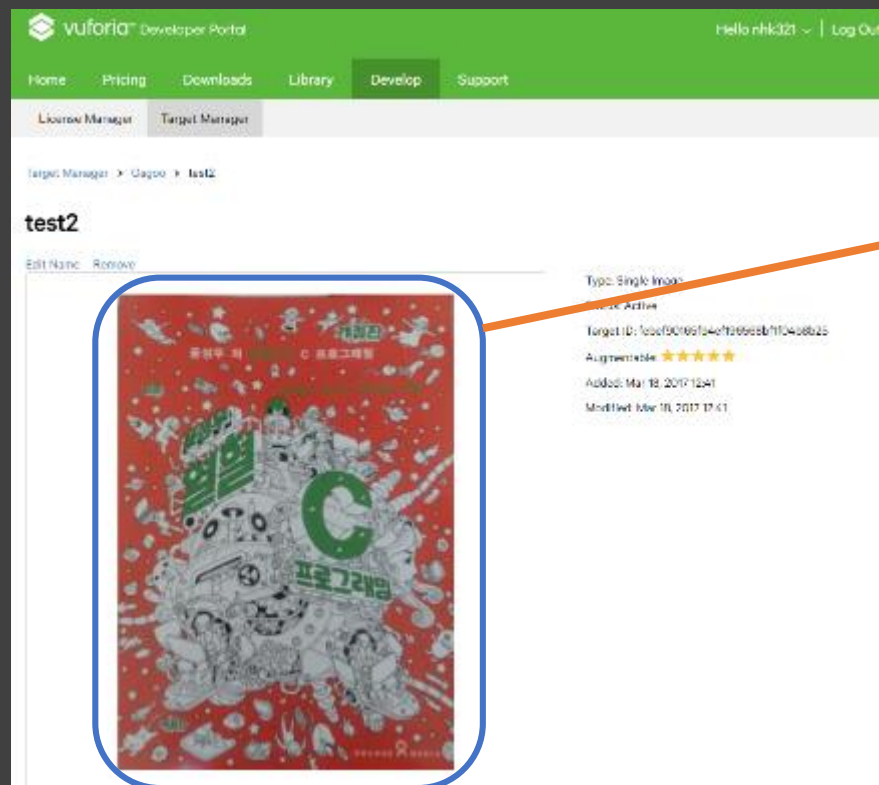
Windows 10/ C# , SQL / Unity, Mysql , AWS, Eclipse ,
ERMaster , OpenCV(Vuforia) ,Vuforia Database

사용자가 원하는 위치에 실제 크기와 같은 3D 가구 모델을 증강시켜 방의 분위기와 가구가 맞는지, 가구의 크기가 방에 맞는지를 어플리케이션을 통해 미리 확인해 볼 수 있게 하는 것을 목표로 했습니다. 유니티에서 뷰포리아 라이브러리를 삽입해 3D 모델을 증강시키는데 사용했습니다. 유니티와 서버와의 통신을 할 때에는 AES 256 암호화 처리를 통해 사용자의 정보를 암호화처리 하고, 사용자 개인만의 커스텀 가구를 만들어 AR로 볼 수 있게 만들었습니다.

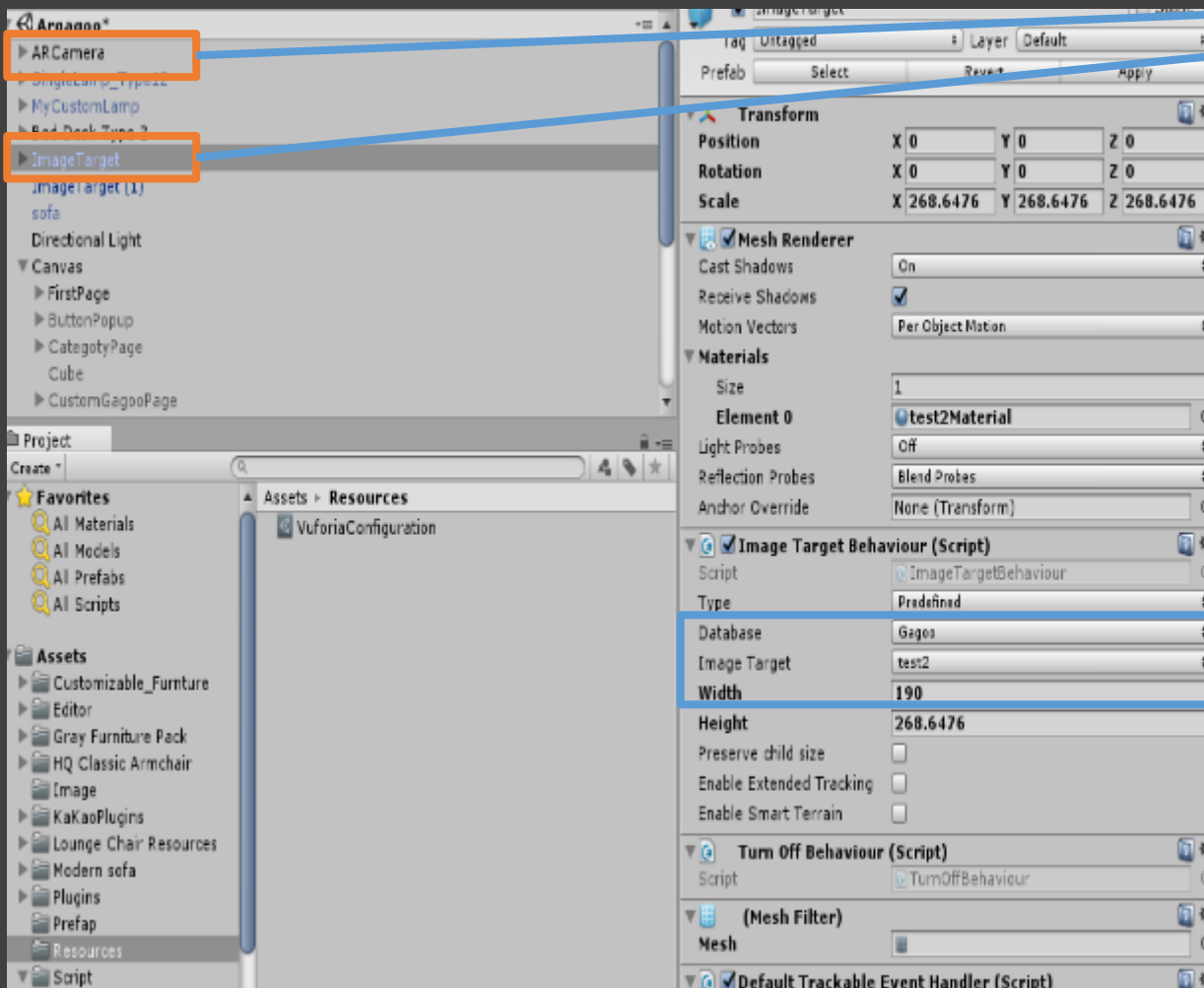




뷰포리아 데이터베이스 페이지에서
데이터베이스를 생성해줬습니다.



뷰포리아 데이터베이스에서
위에 3D모델을 증강 시켜 줄
이미지 타겟을 설정 해 줬습니다.



뷰포리아 라이브러리를 import 시켜준 후
포함 되어 있는 AR Camera와
ImageTarget을 추가 시켜 줬습니다.

뷰포리아 데이터베이스 페이지에서
만들어 줬던 데이터베이스를 선택해주고
이미지 타겟(위에 3D 모델을 증강시킬 이미지)을
선택 해주고 이미지 타겟의 실제 크기를
설정해줬습니다.

로딩 화면



Delta Time 변수를 생성
3초간 화면에 표시 후
사라집니다

시작 화면

IKEA의 CATALOG
어플리케이션의
UI 를 참고했습니다.



메뉴 바 입니다.
메뉴 팝업을 열 수 있고, AR로 전환 하는
기능이 있습니다.

밑으로 스크롤을
할 수 있습니다.



거실



다이닝



주방



침실

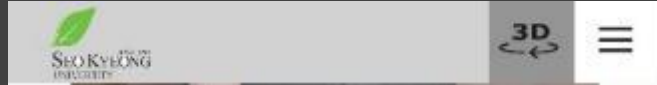
연결

스크롤을 했을 때
나오는 화면입니다.



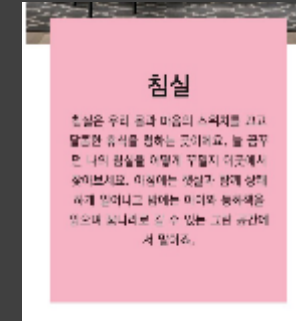
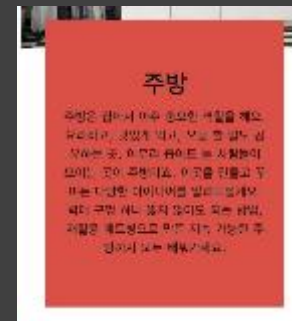
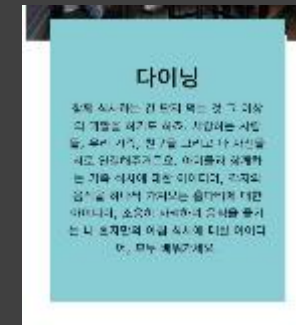
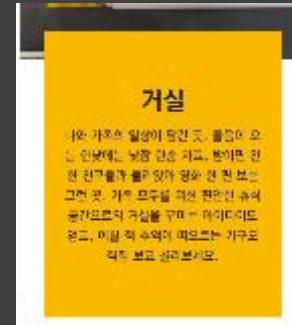
거실

나와 가족의 일상이 담긴 곳. 졸음이 오는 한낮에는 낮잠 한숨 자고, 밤이면 친한 친구들과 둘러앉아 영화 한 편 보는 그런 곳. 가족 모두를 위한 편안한 휴식 공간으로의 거실을 꾸미는 아이디어도 얻고, 어릴 적 추억이 떠오르는 가구도 직접 보고 골라보세요.



메뉴 바에 있는
메뉴 팝업을 클릭하세
요

메뉴 팝업입니다.




각 기능에 맞는 페이
지로 이동합니다.


로그인과 회원가입 기능입니다.

X

아이디

비밀번호

X

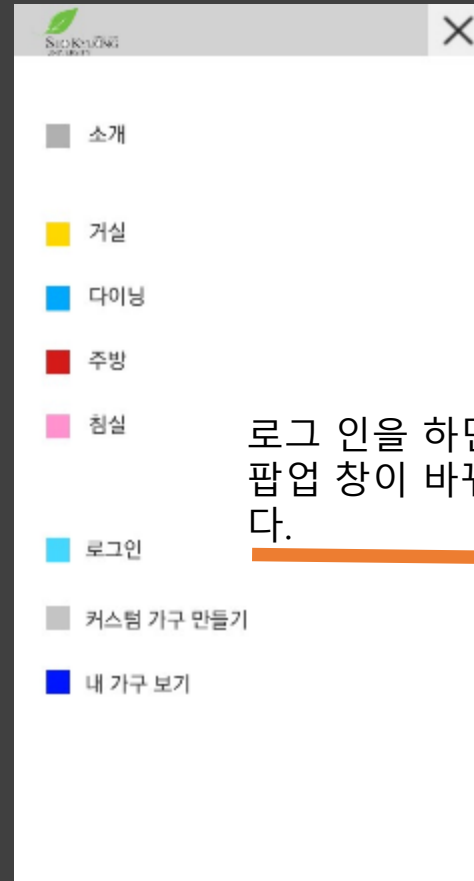
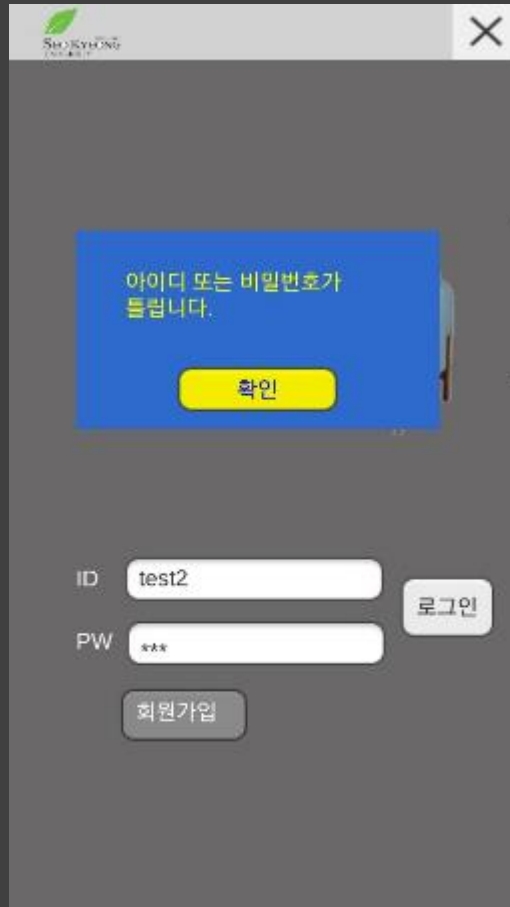


ID

PW

AES 암호화 ,
복호화 처리로 비밀번호
를
암호화 한 후 저장했습
니다

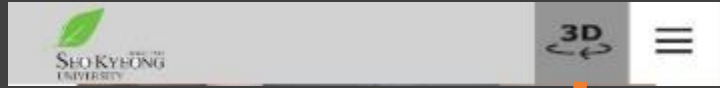
아이디 또는 비밀번호가 다르면 예외처리 로
오류 메시지를 띄워줍니다.



로그인을 하면
팝업창이 바뀌는 것을 볼 수 있습니다.



증강현실 기능 입니다.



메뉴 바에 있는
증강현실 전환 버튼을
클릭하면 증강현실 페
이지로 전환 됩니다.

안내문과 함께
확인 버튼을 누르게 되
면
증강현실을 시작합니다.





Vuforia 증강현실 카메라가 시작되면 이미 Vuforia 데이터베이스에 저장해 놔던 이미지 타겟을 3D 모델을 증강시키하고자 하는 위치에 놓겠습니다.



실제 크기와 비슷한 3D 가구 모델을 증강 시켰습니다.
이미지 타겟에 가까이 가게 되면 가구의 세부적인 모습
을 볼 수 있습니다.

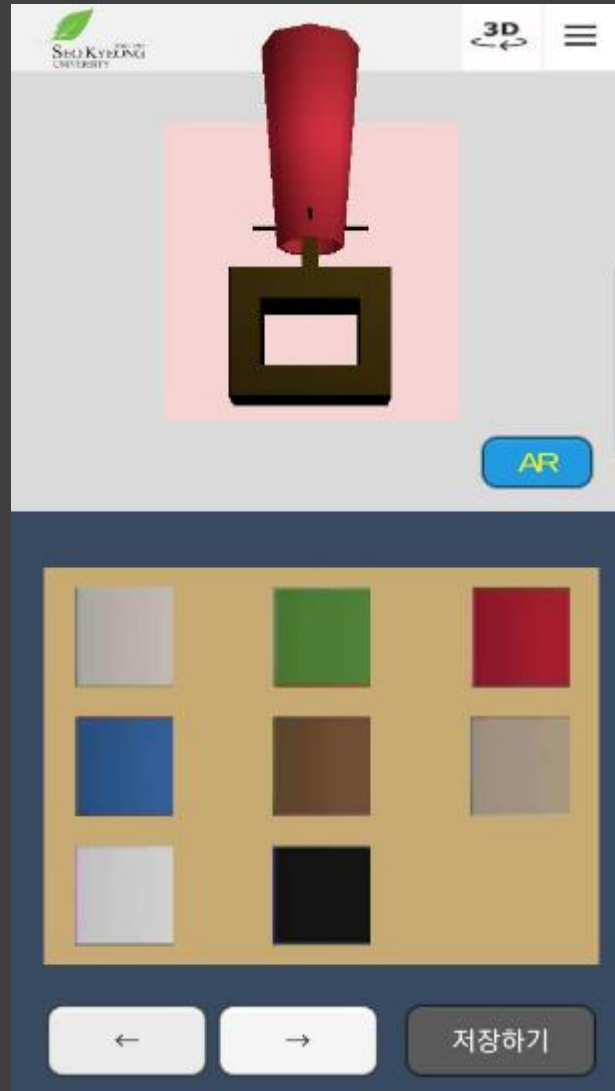


모델에 가까이 다가갔을 경우

나만의 가구 만들기



Custom Furniture 는
팝업 메뉴에서 이동 가능합니다.
로그인 이 필요한 서비스 이고
로그인 이 안 되어있는 경우에는
팝업 창을 띄워서 로그인 을 하게
해줍니다.



Custom Furniture 페이지에서
모델의 모양, 색, 전구의 색깔
맞춤형으로 제작 후 저장 할
있습니다.

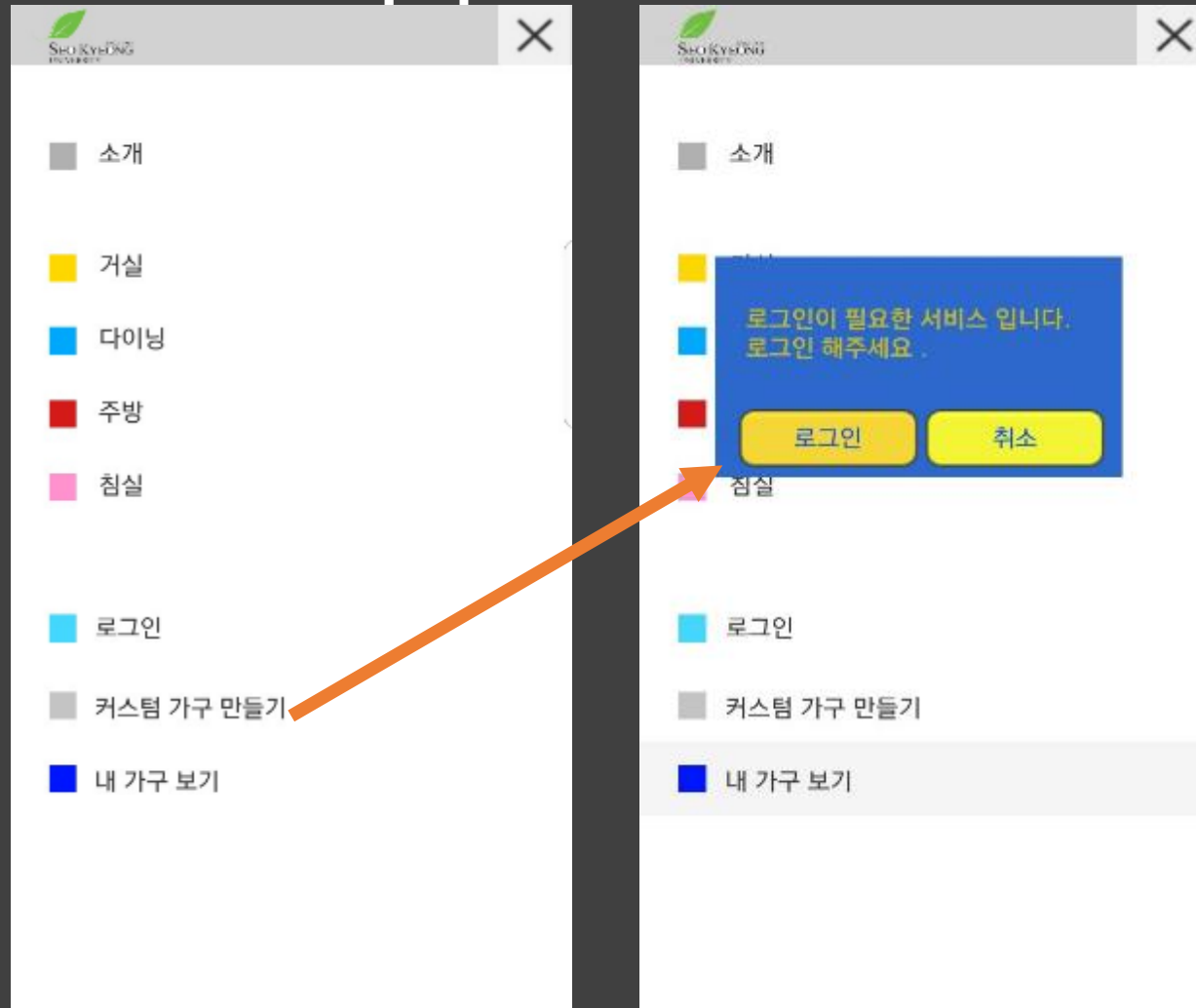


AR로 이동
버튼을 누르면
제작 중이던 나만의
가구를 증강현실로
확인할 수 있습니다.



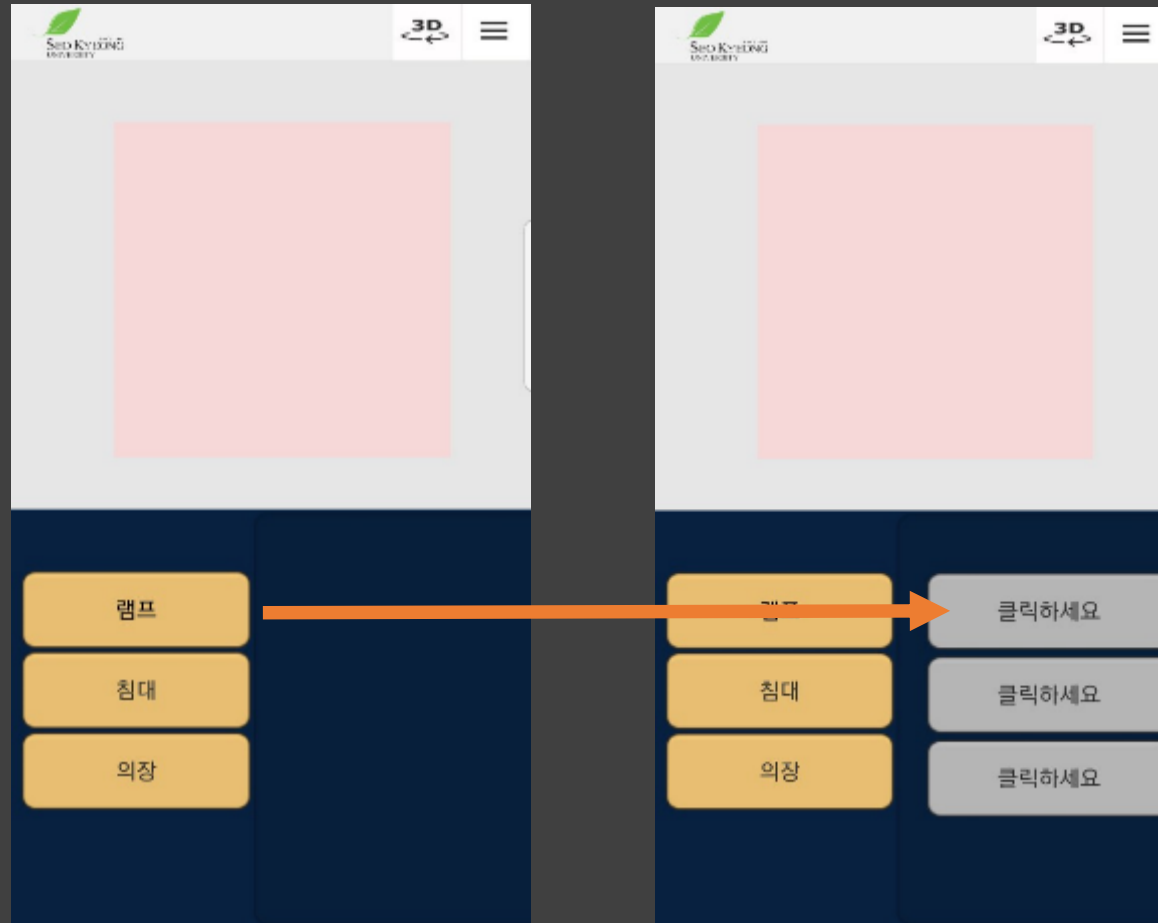
저장하기 버튼을 눌러
가구를 저장해두면
후에 가구를 불러와서
해당 가구를 볼 수 있습니다.

나만의 가구 확인 하기



내 가구 보기도
로그인이 필요한 서비스입니다.

비어있는 첫 화면 입니다.



Lamp 를 클릭하게 되면 내 아이디어의 저장 되어 있는 Lamp가구를 데이터베이스에서 받아와 개수만큼 리스트 뷰를 생성해줍니다.

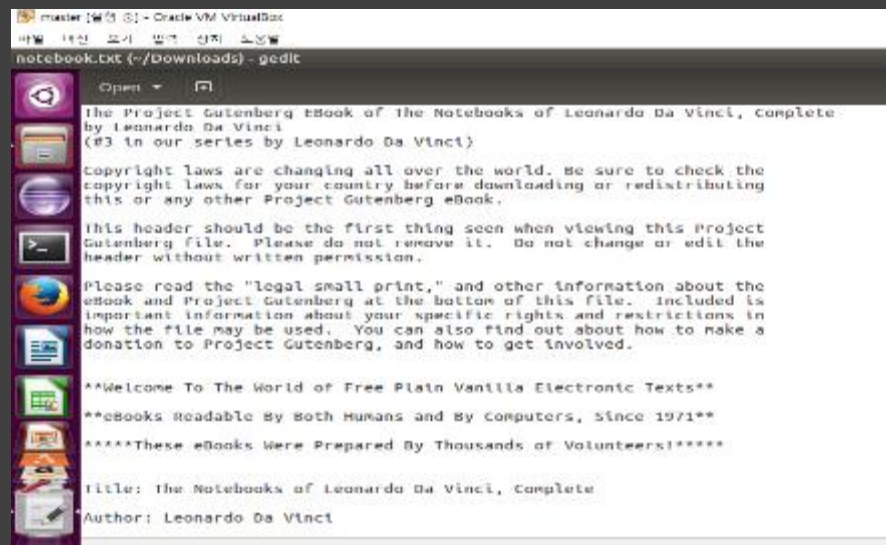


리스트 뷰를 선택하면 저장 되어있던 나만의 가구를 볼 수 있습니다.



가구를 드래그 하면 원하는 방향 대로 360도 회전을 하면서 가구를 볼 수 있습니다.

Hadoop HIVE 를 이용한 빅데이터 처리



Hadoop HIVE 를 이용한 빅데이터 처리



```
Stage-Stage-2: HDFS Read: 5780940 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
the      20583
of       10388
and      7611
in       5226
to       5079
is       4097
a        3797
that     2668
it       2387
which    2387
Time taken: 6.491 seconds, Fetched: 10 row(s)
```

개발기간/개발인원

2017.03~2017.05/1명

분야

빅데이터 처리

맡은 역할

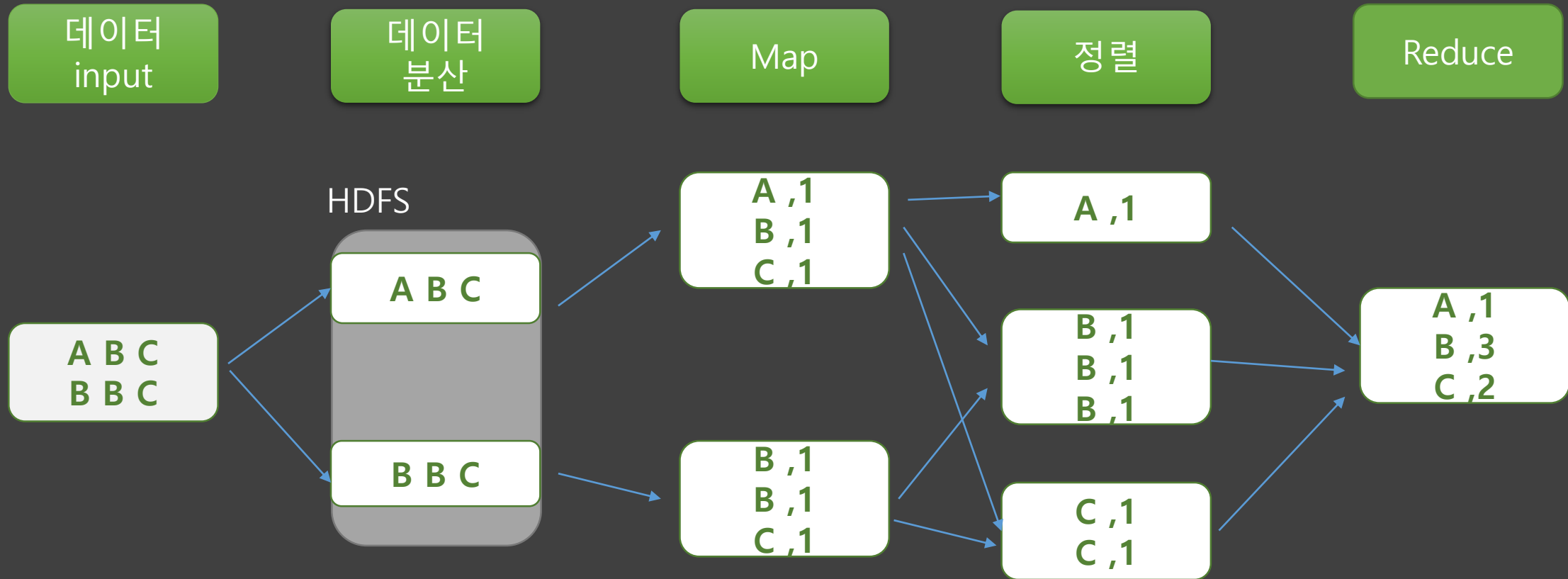
빅데이터 분석

플랫폼/ 개발언어 / 개발환경

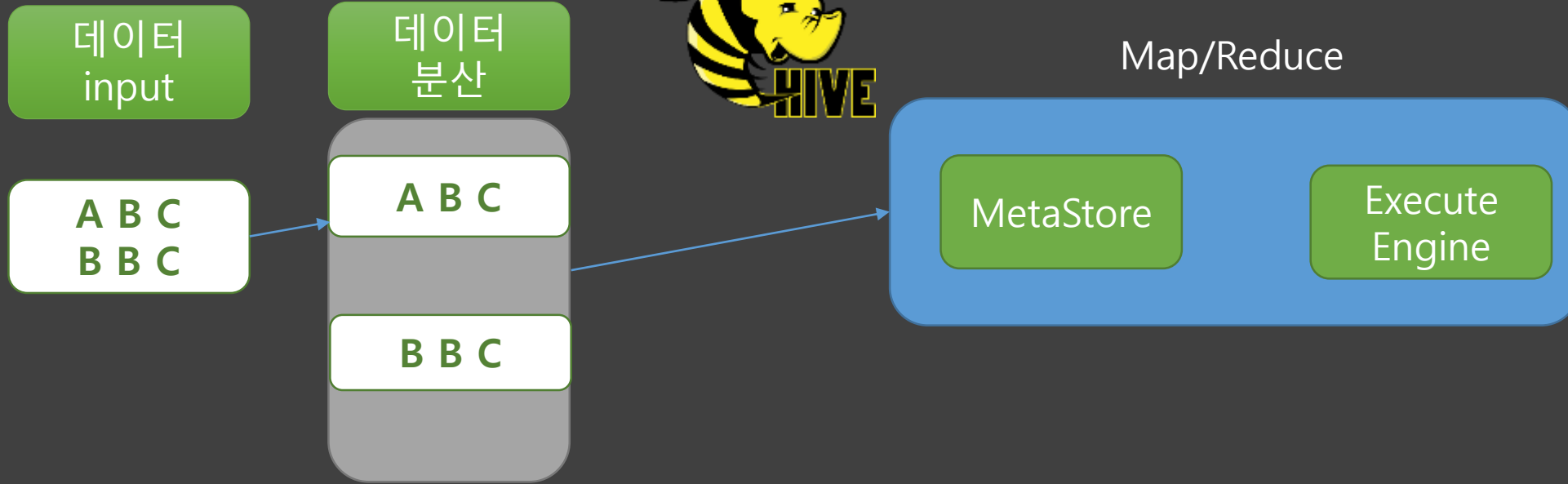
Linux Ubuntu/ SQL / Oracle VM VirtualBox , Hadoop ,
Hive

빅데이터 분산 컴퓨팅 강의를 수강하면서 구현 해본 빅데이터 분석입
니다.

SQL 문으로 Hadoop의 핵심 기능인 Map/Reduce 를 자동으로 적용해
주는 HIVE 를 사용해봤습니다.

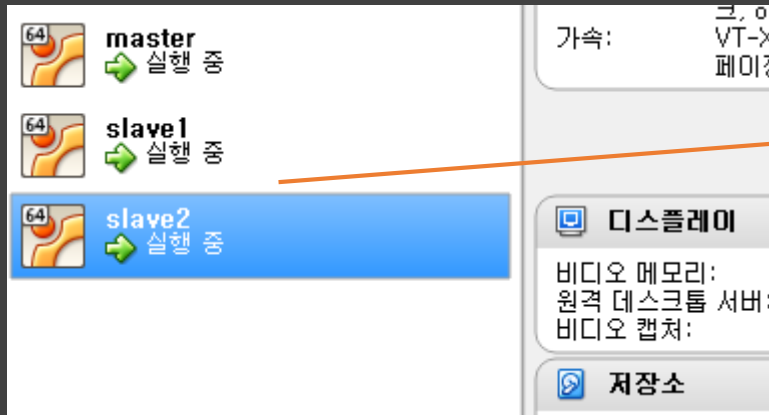
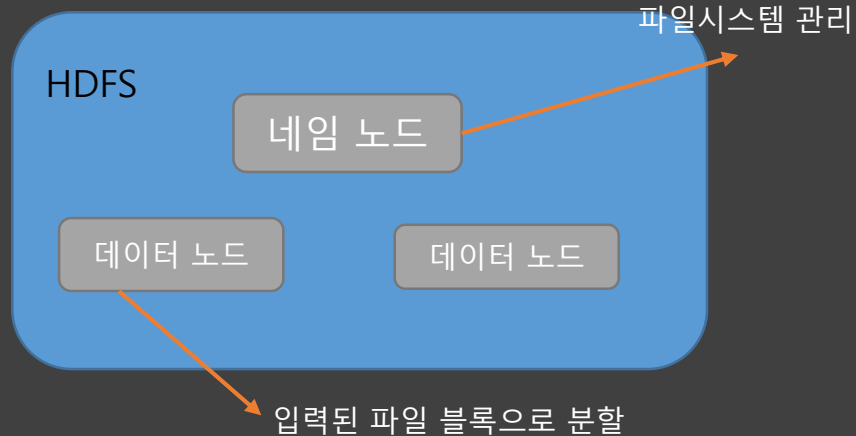


하둡의 중요 기능인 Map/Reduce의 진행 과정입니다.
HDFS 파일 시스템으로 데이터를 분산 처리 한 후
Map으로 데이터를 나눠주고
정렬을 시킨 후에 Reduce로 결과값(WordCount)을 얻습니다.



하둡의 HIVE를 이용할 때의 진행 과정입니다.
데이터를 분산처리하는 HDFS까지는 동일한 과정을 가진 후
DB스키마를 만들어 MetaStore 에 스키마를 저장 한 후
Excute Engine 에서 하둡의 Map Reduce 를 해줍니다.
즉, MetaStore 에서 테이블을 만든 후
SQL 쿼리 문만 적으면 Map/Reduce 된 결과를 얻을 수 있습니다.

먼저 데이터를 분산처리 하기 위해서 HDFS를 만들어 보겠습니다.



Slave 1,2 에 네임노드를 삭제해서 데이터노드로 만들었습니다.

```
NOTICE.txt bin-15 09:20 hadoop data lib map = logs resbin = 10 slaves
coe@master:~/Hadoop/hadoop-2.7.3$ sbin/start-all.sh
This script is deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /home/coe/Hadoop/hadoop-2.7.3/logs/h
```

파일시스템 관리

```
coe@master:~/Hadoop/hadoop-2.7.3$ jps
2887 ResourceManager
2633 SecondaryNameNode
3340 Jps
2158 NameNode
coe@master:~/Hadoop/hadoop-2.7.3$
```

```
coe@slave1:~$ jps
3187 Jps
2444 NodeManager
2269 DataNode
coe@slave1:~$
```

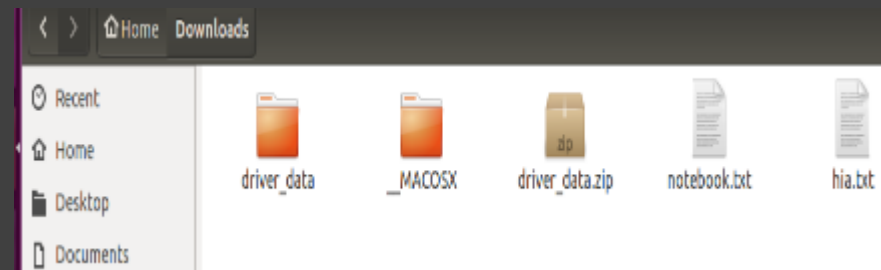
Master, Slave1,2 에서 네임노드, 데이터노드가 제대로 되어있는지 확인 했습니다.

```
coe@slave2:~$ jps
2771 Jps
1929 DataNode
2462 NodeManager
coe@slave2:~$
```

```
2152 Namenode
2332 SecondaryNameNode
coe@master:~/hive$ hive
```

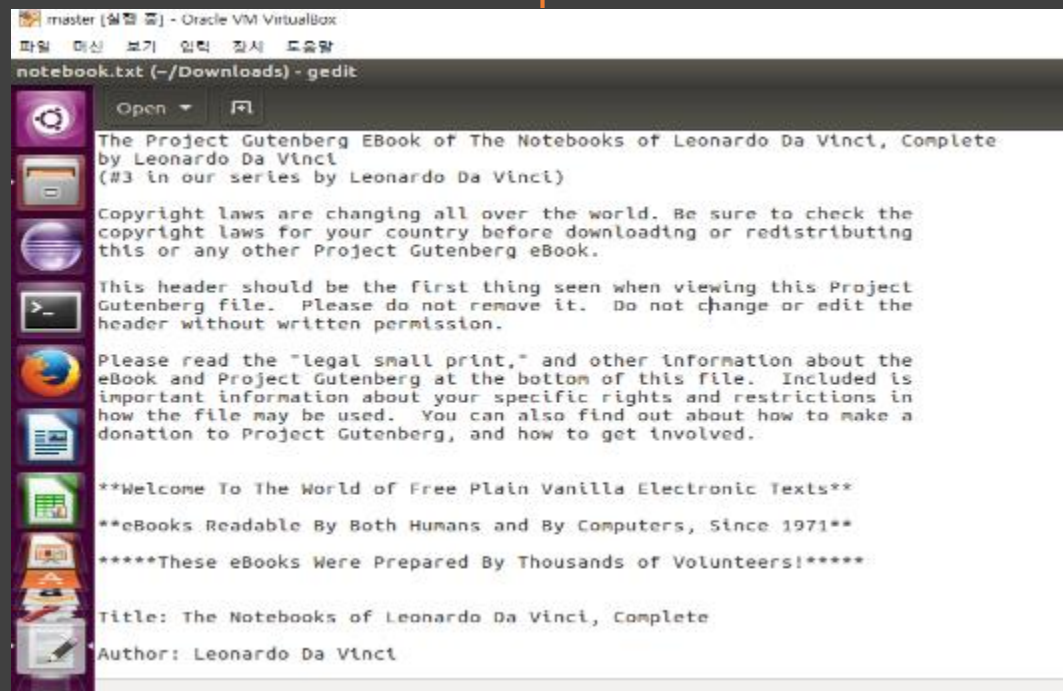
```
Logging initialized using configuration in jar:file:/home/coe/apache-hive-2.1.1-
bin/lib/hive-common-2.1.1.jar!/hive-log4j2.properties Async: true
```

```
hive> show databases;
OK
default
user_db
worddb
Time taken: 11.138 seconds, Fetched: 3 row(s)
hive> use user_db;
OK
Time taken: 0.222 seconds
```



Hive를 실행 시켜 준 후 hive 내부의 데이터 베이스에 접속했습니다.

이제 notebook.txt 안에 들어가 있는 단어를 분석해서 어떤 글자가 가장 많이 사용되었나를 나타내보겠습니다.



```
hive> create table if not exists notebook(word String);
OK
Time taken: 13.402 seconds
```

단어를 저장할 테이블을 만들었습니다.

```
coe@master:~/Hadoop/hadoop-2.7.3$ ls
LICENSE.txt  README.txt  etc          include  libexec  masters  share  use metastore_dli
NOTICE.txt   bin         hadoop_data  lib      logs     sbin     slaves
coe@master:~/Hadoop/hadoop-2.7.3$ cd bin
coe@master:~/Hadoop/hadoop-2.7.3/bin$ ./hdfs dfs -put ~/Downloads/notebook.txt /
user/input
coe@master:~/Hadoop/hadoop-2.7.3/bin$
```

데이터를 분산시켜 저장하기 위해서
Notebook.txt 파일을 hdfs 안에 저장시켰습니다.

```
hive> load data inpath '/user/input/notebook.txt' overwrite into table notebook;
Loading data to table user_db.notebook
OK
Time taken: 5.784 seconds
```

하이프의 만들어 둔 테이블에
HDFS에 저장되어있는 파일을 불러와줬습니다..

```

Time taken: 9.85 seconds, Fetched: 10 row(s)
hive> select word,count(1) as count from(select explode(split(word,' '))as word
from notebook) w where word <> '' group by word order by count desc limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = coe_20170515092003_de626293-20ba-4cfd-8010-24ab0e6133a6
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)

```

하이프에서 select 문을 사용해서
데이터의 단어 마다 count를 해서 어떤 단어가
가장 많이 사용되었는지를 결과로 나타내보게했습니다.

```

Stage-Stage-2: HDFS Read: 5780940 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
the      20583
of       10388
and      7611
in       5226
to       5079
is       4097
a        3797
that     2668
it       2387
which    2387
Time taken: 6.491 seconds, Fetched: 10 row(s)

```

Select 문의 결과.

Unity를 이용한 2D게임 어플리케이션



Github 유니티 부분.

Unity를 이용한 2D게임 어플리케이션



개발기간/개발인원

2016.12~2017.02/3명

분야

게임 프로그래밍

맡은 역할

파일 시스템을 이용한 저장구현, 확률에 따른 주식 기능, 리스트뷰를 통한 구매가능 목록, 캐릭터레벨과 배경 변경

플랫폼/ 개발언어 / 개발환경

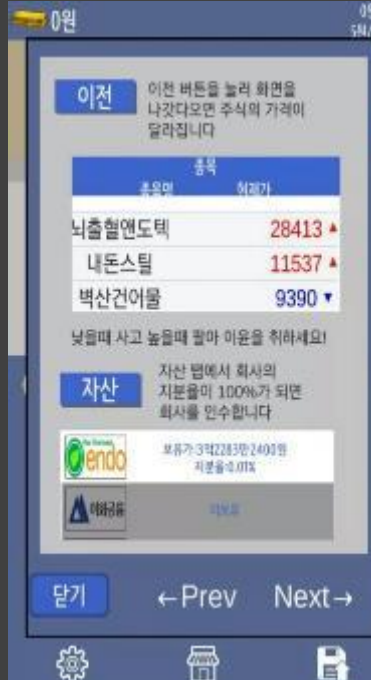
Windows10 / C# / Unity 2D

간단한 조작으로 즐길 수 있는 2D 게임을 만들어 출시해봤습니다.
유니티 내부의 파일을 생성해 데이터를 저장해놓고
유니티의 실행 순서 Awake() -> Start() -> Update() 인 점을 이용해
Awake 에서 파일에 저장되어있는 정보를 불러와 자동저장을 할 수있
게 했습니다.

Unsigned long 을 이용해 1000조 정도의 큰 수를 다뤄봤고
이미지와 텍스트 그리고 버튼이 들어있는 커스텀 리스트뷰를
만들어 구매 가능한 목록들을 보여줬습니다.



유니티 2D를 이용해
단순 조작으로 즐길 수 있는 게임을 만들어 출시 해봤습니다.
화면을 터치하면 돈이 올라가고
리스트뷰를 이용해 구입이 가능한 집과 국가를 나타냈고
주식을 사게되면 랜덤 확률에 따라서
한 주 의 가격이 변동되게 해줬습니다.
돈을 올려 계급을 올리는
단순 2D 게임을 만들어 봤습니다.



```

public void SaveData(){           //저장버튼누르면

    BinaryFormatter bf = new BinaryFormatter ();
    FileStream file = File.Create (Application.persistentDataPath + "/playerInfo.dat");

    PlayerData data = new PlayerData ();

    //A --> B에 할당
    data.money = myMoney.money;
    data.moneyspeed = myMoney.moneyspeed;
    data.moneyTouch = myMoney.touchspeed;
    data.selected_BG = myBGList.Selected_BG;

```

게임 저장은 서버를 이용하지 않고 게임 내부에 파일을 생성시켜서 데이터 값을 저장시켰습니다.
 유니티의 Awake() -> Start() -> Update() 순으로 함수가 실행되는 것을 이용해서 Awake 에서 파일을 불러와 자동 저장을 해줬습니다.



서비스의 유니티 애드를 이용해서 간단한 수익을 내봤습니다.