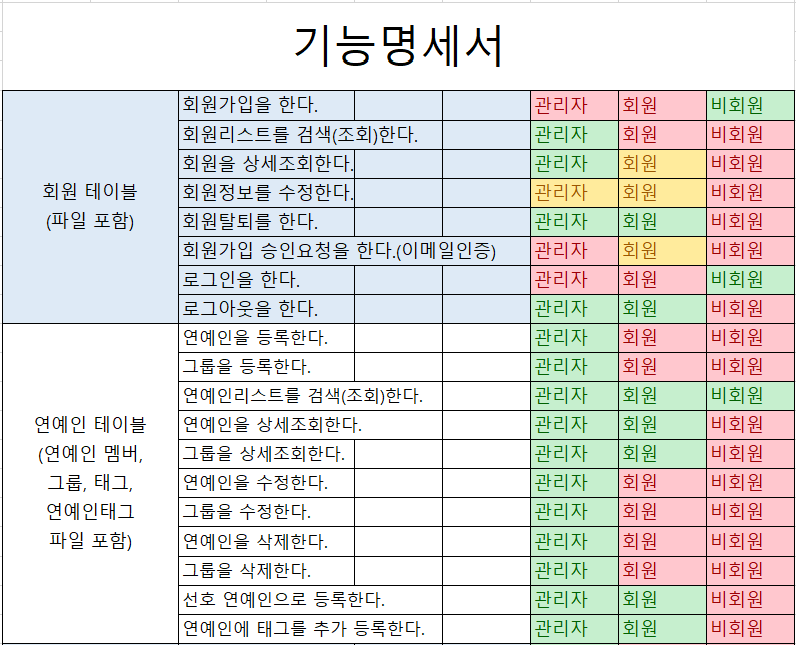
1. 주제 및 설명

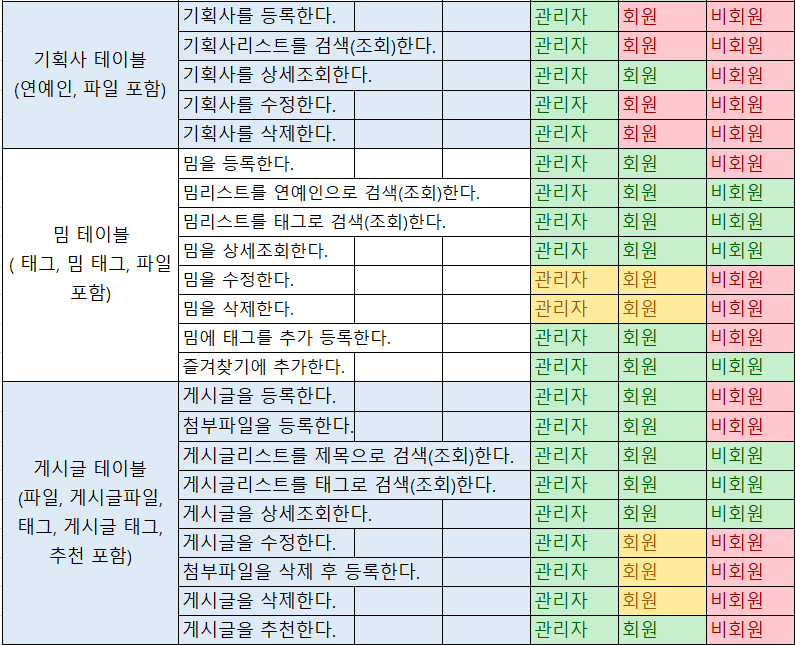
최근 K-POP시장이 성장함에 따라 K-POP 수요 시장의 규모 또한 커졌습니다. 이러한 수요를 해결하기 위해 수많은 연예인들이 등장하게 되지만, 연예인에 대한 정보가 난립함에 따라 원하는 연예인을 검색하기 힘들어져 특정 연예인만 주목받고 많은 수의 연예인들이 주목받지 못하는 상황이 되었습니다. 이러한 수요와 공급의 불균형을 해소하기 위해 소비자에게 취향에 맞는 연예인을 추천해 주는 서비스인 ‘입덕 도우미’를 설계하게 되었습니다.

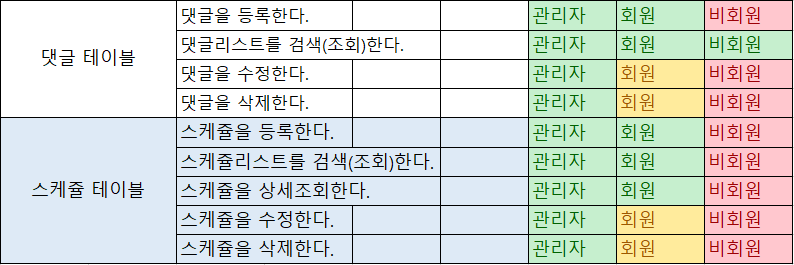
‘입덕 도우미’는 말 그대로 ‘입덕’을 도와준다는 뜻으로, ‘입덕’이란 말은 ‘무언가에 관심을 가지고 좋아하기 시작한다’는 뜻의 신조어입니다. 기본적으로는 사용자가 원하는 취향의 연예인을 찾을 수 있도록 취향 별로 연예인을 검색할 수 있는 기능을 제공합니다. 검색 뿐만 아니라 연예인에 대한 관심을 한층 더 끌어올려주는 영상과 같은 밈이나 연예인 관련 게시글 등의 여러 서비스를 제공합니다.

2. 주요 기능

해당 프로젝트의 주요 기능들은 다음과 같습니다. 색깔별로 권한 범위가 다르며, 초록색일 경우 모든 권한, 노란색일 경우 일부 권한, 빨간색일 경우 제한된 권한을 가집니다.







3. 사용한 SQL

각 기능별 사용한 SQL은 다음과 같습니다.

[회원가입]

insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('이름', '경로',CURRENT\_DATE()); insert into MEM\_TBL ( MEM\_ID, MEM\_PWD, MEM\_NICK, MEM\_NM, FILE\_IDX) values ('아이디', hex(aes\_encrypt('암호 문자열', '암호화 키')), '닉네임', '이름', LAST\_INSERT\_ID());

[회원리스트 조회]

select MEM\_ID, MEM\_NICK, MEM\_NM from MEM\_TBL limit 10 offset N;

[회원 상세조회]

select M.MEM\_ID, M.MEM\_NICK, M.MEM\_NM, M.MEM\_STAT, M.ACVT, F.FILE\_PATH from MEM\_TBL M inner join FILE\_TBL F on M.FILE\_IDX = F.FILE\_IDX where M.MEM\_ID = '아이디';

[회원정보 수정]

update MEM\_TBL M LEFT OUTER JOIN FILE\_TBL F ON M.FILE\_IDX = F.FILE\_IDX SET M.MEM\_PWD = hex(aes\_encrypt('암호 문자열', '암호화 키')), M.MEM\_NICK = '닉네임', M.MEM\_NM = '이름', F.FILE\_PATH = '파일경로' where M.MEM\_ID = '아이디';

[회원 탈퇴]

update MEM\_TBL set ACVT = 0 WHERE MEM\_ID = '아이디';

[회원가입 승인요청(이메일인증)]

update MEM\_TBL set MEM\_STAT = '회원' where MEM\_ID = '아이디';

[로그인]

select M.MEM\_ID, M.MEM\_NICK, M.MEM\_NM, M.MEM\_STAT, F.FILE\_PATH from MEM\_TBL M inner join FILE\_TBL F on M.FILE\_IDX = F.FILE\_IDX where M.MEM\_ID = 'ID' and M.ACVT = TRUE and M.MEM\_PWD = hex(aes\_encrypt('PWD', '암호화 키'));

[연예인 등록]

insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('파일이름', '파일경로', CURRENT\_DATE()); insert into STAR\_TBL(STAR\_NM, STAR\_TYP, STAR\_DBT\_DATE, CMPY\_NM, FILE\_IDX) values ('이름', 'STAR', CURRENT\_DATE(), '소속사', LAST\_INSERT\_ID()); insert into STAR\_MEM\_TBL(STAR\_MEM\_HGHT, STAR\_MEM\_AGE, STAR\_MEM\_BLD, STAR\_MEM\_WGHT, STAR\_MEM\_BIRTH, STAR\_IDX, GRP\_IDX) values ('키', '나이', '혈액형', '몸무게', '출생', LAST\_INSERT\_ID(), '그룹인덱스');

[그룹 등록]

insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('파일이름', '파일경로', CURRENT\_DATE()); insert into STAR\_TBL(STAR\_NM, STAR\_TYP, STAR\_DBT\_DATE, CMPY\_NM, FILE\_IDX) values ('이름', 'GRP', CURRENT\_DATE(), '소속사', LAST\_INSERT\_ID()); insert into GRP\_TBL(STAR\_IDX, CLB\_SITE) values (LAST\_INSERT\_ID(), '사이트');

[연예인리스트 검색(조회)]

select S.STAR\_IDX, S.STAR\_NM, S.STAR\_TYP, S.STAR\_DBT\_DATE, S.CMPY\_NM, F.FILE\_PATH FROM STAR\_TBL S inner join FILE\_TBL F on S.FILE\_IDX = F.FILE\_IDX limit 10 offset N;

[연예인 상세조회]

select S.STAR\_MEM\_IDX, S.STAR\_MEM\_HGHT, S.STAR\_MEM\_AGE, S.STAR\_MEM\_BLD, S.STAR\_MEM\_WGHT, S.STAR\_MEM\_BIRTH, GS.STAR\_NM, GS.STAR\_TYP from STAR\_MEM\_TBL S right outer join GRP\_TBL G on S.GRP\_IDX = G.GRP\_IDX right outer join STAR\_TBL GS on G.STAR\_IDX = GS.STAR\_IDX where S.STAR\_IDX = '인덱스' and GS.STAR\_TYP='TYP';

[그룹 상세조회]

select G.GRP\_IDX, G.CLB\_SITE from GRP\_TBL G inner join STAR\_TBL S on G.STAR\_IDX = S.STAR\_IDX where G.STAR\_IDX = '인덱스';

[연예인 수정]

update STAR\_MEM\_TBL SM inner join STAR\_TBL SI on SM.STAR\_IDX = SI.STAR\_IDX inner join FILE\_TBL FT on SM.FILE\_IDX = FT.FILE\_IDX set SM.STAR\_MEM\_HGHT = '키', SM.STAR\_MEM\_BLD = '혈액형', SM.STAR\_MEM\_WGHT = '몸무게', SM.STAR\_MEM\_BIRTH = '출생', SI.STAR\_DBT\_DATE = '데뷔날짜', FT.FILE\_PATH = '파일경로', SM.GRP\_IDX = (select \* from (select G.GRP\_IDX from GRP\_TBL G inner join STAR\_TBL S on G.STAR\_IDX = S.STAR\_IDX where S.STAR\_NM = '그룹이름' and S.STAR\_TYP = 'GRP')T) where SM.STAR\_MEM\_IDX = '인덱스' and SI.STAR\_TYP = 'STAR';

[그룹 수정]

update GRP\_TBL GT inner join STAR\_TBL ST on GT.STAR\_IDX = ST.STAR\_IDX inner join FILE\_TBL FT on ST.FILE\_IDX = FT.FILE\_IDX set CLB\_SITE = '사이트', ST.STAR\_DBT\_DATE = '데뷔날짜', FT.FILE\_PATH = '파일 경로' where GT.GRP\_IDX = '그룹인덱스';

[연예인 삭제]

delete ST, SMT, FT from STAR\_TBL ST inner join STAR\_MEM\_TBL SMT on ST.STAR\_IDX = SMT.STAR\_IDX inner join FILE\_TBL FT on ST.FILE\_IDX = FT.FILE\_IDX where ST.STAR\_IDX = '인덱스';

[그룹 삭제]

delete ST, GT, FT from STAR\_TBL ST inner join GRP\_TBL GT on ST.STAR\_IDX = GT.SRP\_IDX inner join FILE\_TBL FT on ST.FILE\_IDX = FT.FILE\_IDX where ST.GRP\_IDX = '그룹인덱스';

[선호 연예인으로 등록]

insert into LIKE\_STAR\_TBL(STAR\_IDX, MEM\_ID) values ('연예인인덱스', '회원아이디');

[연예인에 태그 추가 등록]

insert into STAR\_TAG\_TBL(STAR\_IDX, TAG\_NM) values ('연예인인덱스', '태그이름');

[기획사 등록]

insert into CMPY\_TBL ( CMPY\_NM , CMPY\_REP , FILE\_IDX ) values ( '기획사1', '대표1', NULL );

[기획사리스트 조회]

select C.CMPY\_NM, C.CMPY\_REP, F.FILE\_PATH from CMPY\_TBL C inner join FILE\_TBL F on C.FILE\_IDX = F.FILE\_IDX LIMIT 10 OFFSET N;

[기획사 상세조회]

select C.CMPY\_NM, C.CMPY\_REP, F.FILE\_PATH from CMPY\_TBL C inner join FILE\_TBL F on C.FILE\_IDX = F.FILE\_IDX where CMPY\_NM='기획사1' and CMPY\_REP = '대표1';

[기획사 수정]

update CMPY\_TBL C LEFT OUTER JOIN FILE\_TBL F ON C.FILE\_IDX = F.FILE\_IDX SET C.CMPY\_REP = '대표2', F.FILE\_PATH = '파일경로' WHERE CMPY\_NM = '기획사1';

[기획사 삭제]

delete from CMPY\_TBL WHERE CMPY\_NM = '기획사1';

[밈(영상) 등록]

insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('짤', '파일경로', CURRENT\_DATE()); insert into MM\_TBL(FILE\_IDX, MEM\_ID) values (LAST\_INSERT\_ID(), '업로드한회원ID'); insert into STAR\_MM\_TBL(STAR\_IDX, MM\_IDX) values ((select STAR\_IDX from STAR\_TBL where STAR\_NM='그룹1' and STAR\_TYP='GRP' and CMPY\_NM='기획사1'),LAST\_INSERT\_ID());

[밈 리스트를 연예인 이름으로 조회]

select MMT.MM\_IDX, MMT.MM\_HIT\_CNT, FT.FILE\_NM, FT.FILE\_PATH, FT.FILE\_CRT\_DATE from MM\_TBL MMT inner join FILE\_TBL FT on MMT.FILE\_IDX = FT.FILE\_IDX where MMT.MM\_IDX in (select SMT.MM\_IDX from STAR\_MM\_TBL SMT inner join STAR\_TBL ST on SMT.STAR\_IDX = ST.STAR\_IDX where ST.STAR\_NM = '연예인이름') limit N, 10;

[밈 리스트르 태그로 조회]

select MMT.MM\_IDX, MMT.MM\_HIT\_CNT, FT.FILE\_NM, FT.FILE\_PATH, FT.FILE\_CRT\_DATE from MM\_TBL MMT inner join FILE\_TBL FT on MMT.FILE\_IDX = FT.FILE\_IDX where MMT.MM\_IDX in (select MM\_IDX from MM\_TAG\_TBL where TAG\_NM = '태그이름') limit N, 10;

[밈 수정]

update from FILE\_TBL set FILE\_NM = '파일이름', FILE\_PATH = '파일경로' where FILE\_IDX in (select FILE\_IDX from MM\_TBL where MM\_IDX = '밈인덱스');

[밈 삭제]

delete FT, MMT from MM\_TBL MMT inner join FILE\_TBL FT on MMT.FILE\_IDX = FT.FILE\_IDX where MMT.MM\_IDX = '밈인덱스';

[밈에 태그를 추가로 등록]

insert into MM\_TAG\_TBL (TAG\_NM, MM\_IDX) values ('태그이름', '밈인덱스');

[게시글 등록]

insert into POST\_TBL(MEM\_ID, STAR\_IDX, POST\_TTL, POST\_DATE, POST\_CON) values ('회원아이디', '연예인인덱스', '게시글제목', SYSDATE(), '게시글내용'); insert into POST\_TAG\_TBL(POST\_IDX, TAG\_NM) values (LAST\_INSERT\_ID(), '태그이름'), (LAST\_INSERT\_ID(), '태그이름2'), ...;

[첨부파일 등록]

insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('파일이름', '파일경로', SYSDATE()); insert INTO POST\_FILE\_TBL (POST\_IDX, FILE\_IDX) values ('게시글인덱스', LAST\_INSERT\_ID());

[게시글 리스트를 제목으로 검색]

select POST\_IDX, MEM\_ID, POST\_TTL, POST\_DATE, POST\_HIT\_CNT from POST\_TBL where POST\_TTL like '%게시글제목%' limit N, 10;

[게시글 리스트를 태그로 검색]

select PT.POST\_IDX, PT.MEM\_ID, PT.POST\_TTL, PT.POST\_DATE, PT.POST\_HIT\_CNT from POST\_TBL PT inner join POST\_TAG\_TBL PTT on PT.POST\_IDX = PTT.POST\_IDX where PTT.TAG\_NM = '태그이름' limit N, 10;

[게시글 상세조회]

select POST\_CON from POST\_TBL where POST\_IDX = '게시글인덱스'; select FT.FILE\_NM, FT.FILE\_PATH, FT.FILE\_CRT\_DATE from POST\_FILE\_TBL PFT inner join FILE\_TBL FT on PFT.FILE\_IDX = FT.FILE\_IDX where POST\_IDX = '게시글인덱스';

[게시글 수정]

update from POST\_TBL set POST\_TTL = '게시글제목', POST\_DATE = SYSDATE(), POST\_CON = '게시글내용' where POST\_IDX = '게시글인덱스';

[첨부파일 삭제 후 등록]

delete PFT, FT from POST\_FILE\_TBL PFT inner join FILE\_TBL FT where POST\_IDX = '게시글인덱스'; insert into FILE\_TBL(FILE\_NM, FILE\_PATH, FILE\_CRT\_DATE) values ('파일이름', '파일경로', SYSDATE()); insert INTO POST\_FILE\_TBL (POST\_IDX, FILE\_IDX) values ('게시글인덱스', LAST\_INSERT\_ID());

[게시글 삭제]

delete PT, PFT, FT from POST\_TABLE PT inner join POST\_FILE\_TBL PFT on PT.POST\_IDX = PFT.POST\_IDX inner join FILE\_TBL FT on PFT.FILE\_IDX = FT.FILE\_IDX where PT.POST\_IDX = '게시글인덱스';

[게시글 추천]

insert into RECM\_TBL(POST\_IDX, MEM\_ID) select '게시글인덱스', '회원아이디' from DUAL where not exists (select \* from RECM\_TBL where POST\_IDX = '게시글인덱스' and MEM\_ID = '회원아이디');

[댓글 등록]

insert into CMNT\_TBL (CMNT\_CON, CMNT\_DATE, POST\_IDX, MEM\_ID, HIGH\_CMNT\_IDX) values ('댓글내용', SYSDATE(), '게시글인덱스', '회원아이디', '상위댓글인덱스');

[댓글리스트 검색]

select \* from CMNT\_TBL where POST\_IDX = '게시글인덱스' limit N, 10;

[댓글 수정]

update from CMNT\_TBL set CMNT\_CON = '댓글내용', CMNT\_DATE = SYSDATE() where CMNT\_IDX = '댓글인덱스';

[댓글 삭제]

delete from CMNT\_TBL where CMNT\_IDX = '게시글인덱스' or HIGH\_CMNT\_IDX = '게시글인덱스';

[스케쥴 등록]

insert into SCHE\_TBL(SCHE\_NM, SCHE\_DATE, SCHE\_TRM, STAR\_IDX, MEM\_ID) values ('스케쥴이름', '스케쥴시작날짜', '스케쥴기간', '연예인인덱스', '회원아이디');

[스케쥴 리스트 검색]

select \* from SCHE\_TBL where STAR\_IDX = '연예인인덱스' and SCHE\_DATE >= '시작달 1일' and SCHE\_DATE < '다음달 1일';

[스케쥴 수정]

update from SCHE\_TBL set SCHE\_NM = '스케쥴이름', SCHE\_DATE = '시작일', SCHE\_TRM = '스케쥴기간' where SCHE\_IDX = '스케쥴인덱스';

[스케쥴 삭제]

delete from SCHE\_TBL where SCHE\_IDX = '스케쥴인덱스';

4. 시연 영상

Youtube : https://youtu.be/PKpa2yCRwCE

Github: https://github.com/jsc2576/ipduckhelper

5. 고찰

[정수찬]

프로젝트 백 엔드 부분을 대부분 담당하면서 모든 쿼리를 작성하는 데 이틀도 걸리지 않아서 생각보다 간단하다고 생각하였다. 하지만 프로젝트를 진행하고 강의가 진행됨에 따라서 트랜잭션 등 더욱 많은 기능들이 필수불가결로 사용되어야 하는 것을 깨달았고 오히려 작성하는 것 보다 프로젝트 흐름에 따라 쿼리를 수정하는 것이 더 오래 걸리게 되었다. 또한 mysql을 사용하지만 이를 맵핑해주는 mybatis를 사용하면서 mysql보다는 mybatis를 공부하는데 더 많은 시간을 쏟아부었고 spring boot의 공부가 얕아서 이에 따라 생기는 버그들을 잡는데도 많은 시간이 사용되었다. 그래서 최근 트랜드에 맞춰 사용함에 따라 더욱 많은 프로그램과 기능들을 공부해야 할 필요성을 느끼게 되었다.

기여도 : 10/10

[정석호]

저는 이번 웹 프로젝트에서 프론트 엔드 개발을 담당하였습니다. 동적 페이지를 만들어 내기 위해 여러 프레임워크 중 VueJS를 선택하였습니다. 각 페이지별 필요한 컴퍼넌트들을 Vue라는 파일로 생성하였고 WebPack이 모든 vue 파일들을 하나의 js로 묶어 결과물이 하나의 페이지로 생성되는 SPA(Single Page Application)를 구현하였습니다. 다만 서버와 클라이언트와의 연동에서 파일 업로드(Multipart Upload) 대해서 문제가 발생하여 이 문제를 해결하는데 시간이 생각보다 많이 소비되어 다른 작업을 진행하지 못하였습니다. 하고 싶었던 부분은 네이버 oAuth 액세스 토큰 연동으로 가능한 로그인과 XSS 보안 및 비밀번호 AES 암호화 등등 보안에 대해서 좀 더 기반을 두고 싶었습니다. 예상치 못한 곳에서 시간을 많이 허비하여 결과물이 매우 만족스럽지 못했습니다. 다만, 프론트 엔드에서는 SPA의 작용으로 첫 접속을 제외한 페이지 이동이 refresh없이 가능하기 때문에 매우 빠르다는 점을 보여줄 수 있습니다. 이번 과제에서 프론트 엔드에서 맡았던 바를 완수하지 못했기 때문에 자체 평가 점수는 10점 만점에 7점이라고 생각합니다.

[강희정]

서비스의 주제 및 메인 아이디어를 구상하고 UX/UI 디자인을 맡았으며, html과 css를 이용해 web publishing을 진행하였습니다. 화면 요소를 html/css파일로 작성할 때 요소들을 적당한 위치에 배치하는 것이 생각보다 까다로웠습니다. firefox의 inspect element창을 이용하여 요소들의 style을 바꿔가면서 요소들을 배치하였습니다. 처음에는 D3.js를 이용해 데이터 시각화를 진행하려 하였으나 google chart가 해당 프로젝트에서의 내용을 더 잘 표현할 수 생각해 google chart로 시각화를 진행하였습니다. 제공하는 샘플은 정적 데이터에 대한 샘플이었지만 database의 data를 가지고 시각화를 진행하게 되면 표현할 수 있는 데이터의 수가 항상 변하게 되고 그에 따른 내용도 변하기에 database와 연동되도록 코드를 수정해야 했습니다. google.visualization.DataTable()이라는 DataTable class에 대해 조사하고, 관련 method들을 찾아보아 시각화 페이지를 구현하였습니다. 서비스의 메인 아이디어를 구상하였고 제가 맡은 부분에서는 작업을 완료했지만 server side쪽과 database 관련 수업임에도 불구하고 database 관련 기능을 사용하지 않았기에 제가 생각하는 자체 평가 점수는 10점 만점 중에 7점이라고 생각합니다.