1차 프로젝트

목차

팀 소개

- 팀명
  - ICE(Infinite Coding Experts)
  - 영원히 코딩하는 전문가들 이라는 의미
- 팀원
  - 이지윤
  - 최연우(리더)
  - 한아름

프로젝트 개요

▼ 프로젝트 소개

- 프로젝트 설명: 지역별 관광지, 문화시설을 한눈에 제공
- 프로젝트 기간: 2023.08.16 ~ 2023.8.25
- 주요기능
  1. 전국 각지의 관광시설 표시
  2. 카테고리와 위치별로 관광시설을 필터링
  3. 상세정보를 모달로 표시
     - 관광시설 상세정보
     - 댓글
     - 좋아요
  4. 로그인/로그아웃
  5. 회원가입
  6. 상세 모달에서 댓글을 조회한다.(로그인 불필요)
  7. 댓글에서 작성, 수정, 삭제를 할 수 있다.(로그인 필요)
  8. 상세 모달에서 좋아요, 싫어요 조회 (로그인 불필요)

9. 좋아요, 싫어요 작성(로그인 필요)

- 사용한 주요 기술
  - Oracle DB
  - PL/SQL
  - Flask
  - Vue3
  - Bootstrap5
  - pinia

▼ 시연 시나리오

- 홈페이지에 접속한다.
- 맵위에 관광지가 핀으로 표시된다.
- 카테고리와 위치에 따라 필터링할 수 있다.
- 특정 핀을 클릭하면 상세모달이 표시된다.
- 모달에는 관광시설에 대한 정보, 댓글, 좋아요를 표시한다.
- 상세 모달에서 댓글창에는 '로그인 후 사용하실 수 있습니다' 안내 표시
- 회원가입한다.
- 로그인한다.
- 상세페이지에서 댓글을 작성한다.
- 댓글이 등록된다.
- 좋아요를 클릭한다.
- 페이지를 새로고침하여도 저장된 정보가 표시된다.

데이터베이스

▼ 테이블 리스트

▼ Facility(시설)

- id: pk
- name: 관광시설명
- address: 시도명칭(ex, 서울특별시)
- latitude: 위도
- longitude: 경도

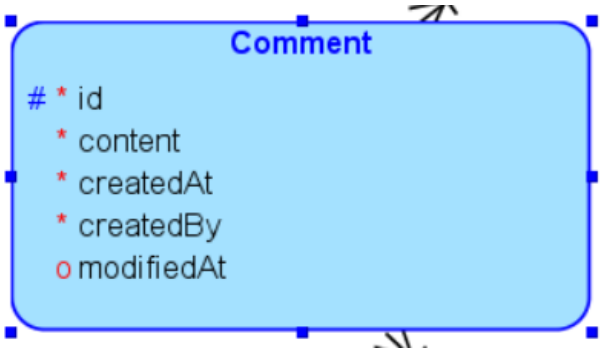
▼ Facility_Info(시설 상세)


Facility_Info
* id
* name
o name_branch
* category1
* category2
o category3
* address_region
o address_city
o address_state
o address_province
o house_number
o street_name
o building_number
* latitude
* longitude
o postal_code
o road_address
o lot_address
o phone_number
o website
o blog_url
o facebook_url
o instagram_url
o closed_days
o operating_hours
o free_parking_available
o paid_parking_available
o admission_fee
o accessible_entrance
o wheelchair_rental_available
o accessible_restroom
o accessible_parking_available
o large_parking_available
o guide_dog_accommodation
o braille_guide_available
o audio_guide_korean
o last_updated_date

▼ User(사용자)

▼ Comment(댓글)

- id: pk
- content: 내용
- createdAt: 작성일
- createdBy: 작성자
- modifiedAt: 수정일



▼ Preference(좋아요, 싫어요)

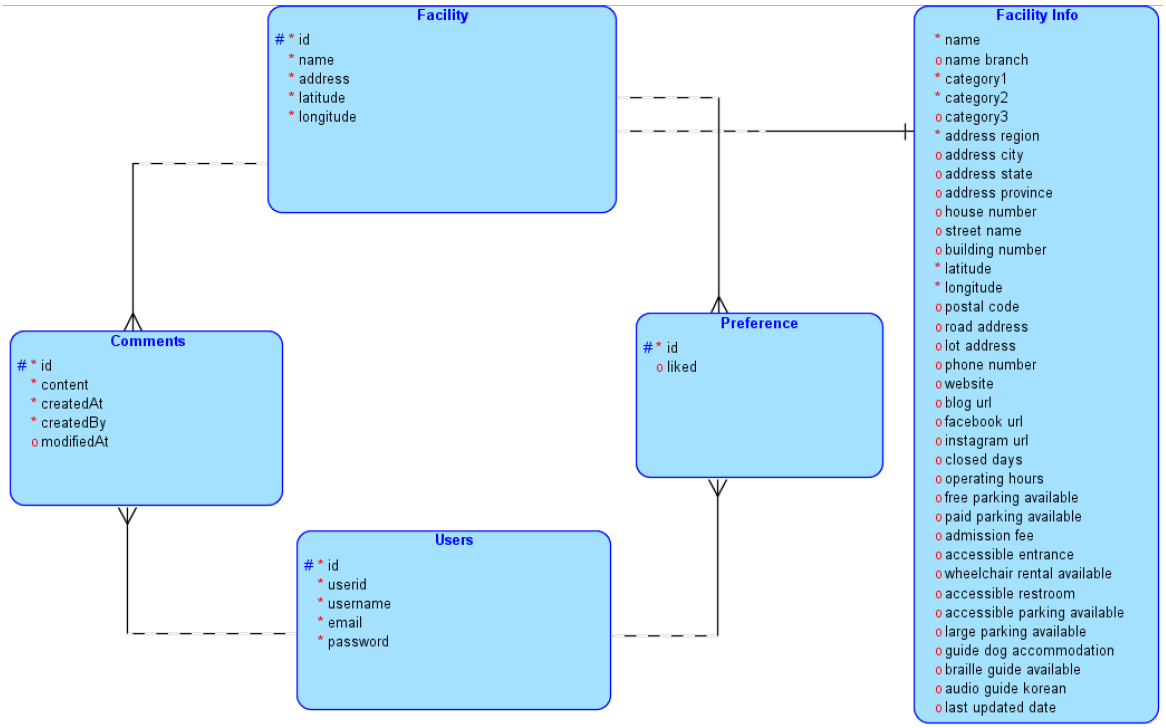
- id: pk
- liked: 좋아요

▼ 관계 요구사항 분석

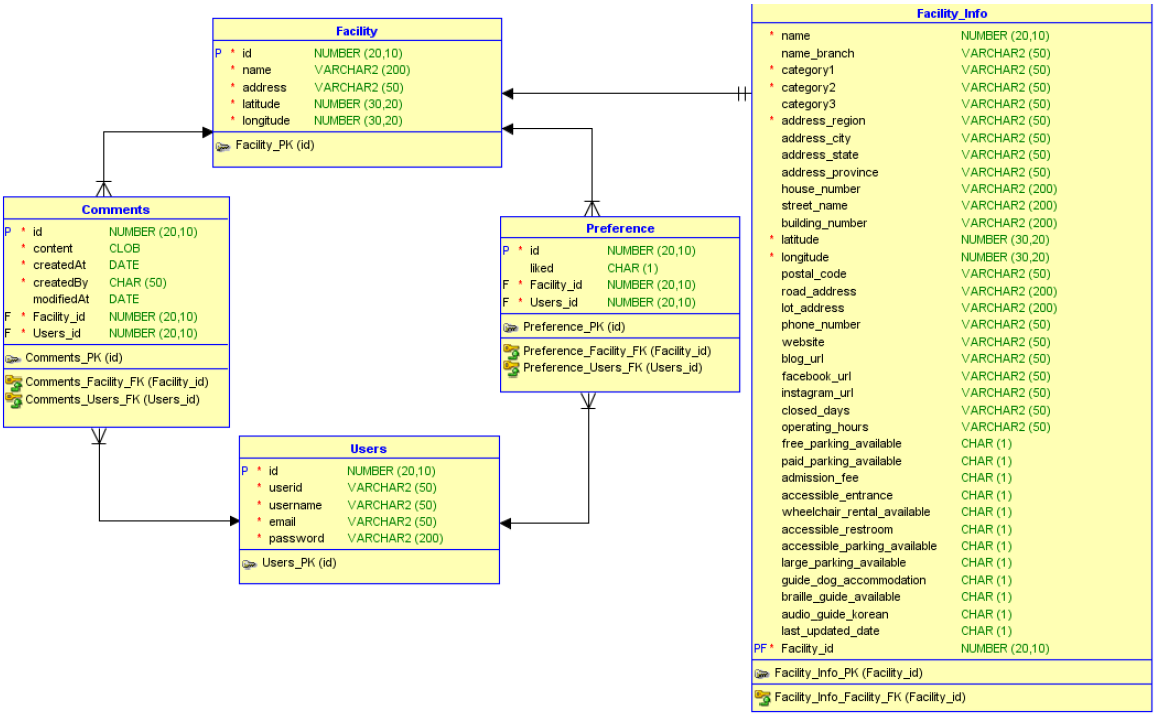
1. Facility테이블과 Facility_Info는 1:1 관계를 가진다.
2. Facility 테이블은 Faclity_Info를 필수적으로 가져야한다.
3. Facility_Info 테이블은 Faclity테이블을 필수적으로 가져야한다.
4. Faclity테이블과 Comment테이블은 1:n관계다.
5. Faclity테이블은 0개 이상의 Comment를 가질 수 있다.
6. Facility테이블은 Comment테이블을 필수로 가지지않아도 된다.
7. Comment테이블은 Facility테이블을 필수적으로 가져야한다.
8. User테이블과 Comment테이블은 1:n 관계다.
9. User테이블은 0개 이상의 Comment를 가질 수 있다.
10. User테이블은 Comment 테이블을 필수적으로 가지지않아도 된다.
11. Comment 테이블은 User테이블을 필수적으로 가져야 한다.

12. Faclity테이블과 Preference테이블은 1:n 관계다.
13. Faclity테이블은 0개 이상의 Preference를 가질 수 있다.
14. Faclity 테이블은 Preference를 필수적으로 가지지 않아도 된다.
15. Facility테이블은 Preference테이블을 필수로 가지지않아도 된다.
16. Preference테이블은 Facility테이블을 필수적으로 가져야한다.
17. User테이블과 Preference테이블은 1:n 관계다.
18. User테이블은 0개 이상의 Preference를 가질 수 있다.
19. User테이블은 Preference 테이블을 필수적으로 가지지않아도 된다.
20. Preference 테이블은 User테이블을 필수적으로 가져야 한다.

▼ 논리 모델



▼ E-R다이어그램



▼ SQL(테이블 생성)

```
drop table facility cascade constraints;
drop table facility_info cascade constraints;
drop table comments cascade constraints;
drop table preference cascade constraints;
drop table users cascade constraints;
purge recyclebin;

CREATE TABLE facility (
    id                NUMBER(20, 10) GENERATED BY DEFAULT AS IDENTITY NOT NULL PRIMARY KEY,
    name              VARCHAR2(200) NOT NULL,
    address           VARCHAR2(50) NOT NULL,
    latitude          NUMBER(30, 20) NOT NULL,
    longitude         NUMBER(30, 20) NOT NULL
);
CREATE TABLE facility_info (
    name                          VARCHAR2(50) NOT NULL,
    name_branch                   VARCHAR2(50),
    category1                     VARCHAR2(50) NOT NULL,
    category2                     VARCHAR2(50) NOT NULL,
    category3                     VARCHAR2(50),
    address_region                VARCHAR2(50) NOT NULL,
    address_city                  VARCHAR2(50),
    address_state                 VARCHAR2(50),
    address_province              VARCHAR2(50),
    house_number                  VARCHAR2(200),
    street_name                   VARCHAR2(200),
    building_number               VARCHAR2(200),
    latitude                      NUMBER(30, 20) NOT NULL,
    longitude                     NUMBER(30, 20) NOT NULL,
    postal_code                   VARCHAR2(50),
    road_address                  VARCHAR2(200),
    lot_address                   VARCHAR2(200),
    phone_number                  VARCHAR2(50),
    website                       VARCHAR2(50),
    blog_url                      VARCHAR2(50),
    facebook_url                  VARCHAR2(50),
    instagram_url                 VARCHAR2(50),
    closed_days                   VARCHAR2(100),
    operating_hours               VARCHAR2(100),
    free_parking_available        CHAR(1),
    paid_parking_available        CHAR(1),
```

```
    admission_fee                CHAR(1),
    accessible_entrance          CHAR(1),
    wheelchair_rental_available  CHAR(1),
    accessible_restroom          CHAR(1),
    accessible_parking_available CHAR(1),
    large_parking_available      CHAR(1),
    guide_dog_accommodation      CHAR(1),
    braille_guide_available      CHAR(1),
    audio_guide_korean           CHAR(1),
    last_updated_date            CHAR(1),
    facility_id                  NUMBER(20, 10) NOT NULL PRIMARY KEY,
    FOREIGN KEY ( facility_id ) REFERENCES facility ( id )
);

CREATE TABLE users (
    id        INTEGER GENERATED BY DEFAULT AS IDENTITY NOT NULL PRIMARY KEY,
    userid    VARCHAR2(50) NOT NULL UNIQUE,
    username  VARCHAR2(50) NOT NULL,
    email     VARCHAR2(50) NOT NULL UNIQUE,
    password  VARCHAR2(200) NOT NULL
);

CREATE TABLE comments (
    id           NUMBER(20, 10) GENERATED BY DEFAULT AS IDENTITY NOT NULL PRIMARY KEY,
    content      CLOB NOT NULL,
    createdat    DATE NOT NULL,
    createdby    CHAR(50) NOT NULL,
    modifiedat   DATE,
    facility_id  INTEGER NOT NULL,
    users_id     INTEGER NOT NULL,
    FOREIGN KEY (facility_id) REFERENCES facility(id),
    FOREIGN KEY (users_id) REFERENCES users ( id )
);

CREATE TABLE preference (
    id           NUMBER(20, 10) GENERATED BY DEFAULT AS IDENTITY NOT NULL PRIMARY KEY,
    liked        CHAR(1),
    facility_id  INTEGER NOT NULL,
    users_id     INTEGER NOT NULL,
    FOREIGN KEY ( facility_id ) REFERENCES facility ( id ),
    FOREIGN KEY ( users_id ) REFERENCES users ( id )
);
```

▼ PL/SQL(CURD)

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- facility
--------------------------------------------------------------------------------
-- spec
CREATE OR REPLACE PACKAGE facility_crud_package AS
    FUNCTION sel_fac_fun (
        p_category facility.category%TYPE,
        p_address facility.ADDRESS%TYPE
    ) RETURN SYS_REFCURSOR;
END facility_crud_package;
/
-- body
CREATE OR REPLACE PACKAGE BODY facility_crud_package AS
    FUNCTION sel_fac_fun (
        p_category facility.category%TYPE,
        p_address facility.ADDRESS%TYPE
    ) RETURN SYS_REFCURSOR IS
        facility_cur SYS_REFCURSOR;
    BEGIN
        OPEN facility_cur FOR
            SELECT *
            FROM facility
            WHERE (p_category IS NULL OR category = p_category)
            AND (p_address IS NULL OR address = p_address);
        RETURN facility_cur;
    END sel_fac_fun;
END facility_crud_package;
```

```
/
-- test
set serveroutput on
DECLARE
    v_facility_cur SYS_REFCURSOR;
    v_facility facility%ROWTYPE;
BEGIN
    v_facility_cur := facility_crud_package.sel_fac_fun;
    LOOP
        FETCH v_facility_cur INTO v_facility;
        EXIT WHEN v_facility_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('User ID: ' || v_facility.name);
    END LOOP;
    CLOSE v_facility_cur;
END;









--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- facility_info
--------------------------------------------------------------------------------
-- spec
CREATE OR REPLACE PACKAGE facility_info_crud_package IS
    FUNCTION sel_fac_info_fun (p_facility_id facility_info.facility_id%type) RETURN SYS_REFCURSOR;
END facility_info_crud_package;
/
-- body
CREATE OR REPLACE PACKAGE BODY facility_info_crud_package IS
    FUNCTION sel_fac_info_fun (p_facility_id facility_info.facility_id%type)
    RETURN SYS_REFCURSOR
    IS v_facilities_cursor SYS_REFCURSOR;
    BEGIN
        OPEN v_facilities_cursor FOR
        SELECT * FROM facility_info WHERE facility_id = p_facility_id;
        RETURN v_facilities_cursor;
    END sel_fac_info_fun;
END facility_info_crud_package;
/
-- test
set serveroutput on
DECLARE
    v_facility_cur SYS_REFCURSOR;
    v_facility facility_info%ROWTYPE;
BEGIN
    v_facility_cur := facility_info_crud_package.sel_fac_info_fun(1);
    LOOP
        FETCH v_facility_cur INTO v_facility;
        EXIT WHEN v_facility_cur%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('User ID: ' || v_facility.name);
    END LOOP;
    CLOSE v_facility_cur;
END;









--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- users
```

```
--------------------------------------------------------------------------------
-- spec
create or replace package users_crud_package
is
    procedure register_user(
        ps_userid users.userid%TYPE,
        ps_username users.username%TYPE,
        ps_email users.email%TYPE,
        ps_password users.password%TYPE
    );
    function authenticate_user(
        ps_userid users.userid%type,
        ps_password users.password%type
    ) return SYS_REFCURSOR;
    procedure validate_exist_user(
        ps_userid users.userid%type
    );
end users_crud_package;
/
-- body
create or replace package body users_crud_package
is
    procedure register_user(
        ps_userid users.userid%TYPE,
        ps_username users.username%TYPE,
        ps_email users.email%TYPE,
        ps_password users.password%TYPE
    )
    is
    begin
        validate_exist_user(ps_userid);
        INSERT INTO users (userid, username, email, password)
        VALUES (ps_userid, ps_username, ps_email, ps_password);
    end register_user;

    function authenticate_user(
        ps_userid users.userid%type,
        ps_password users.password%type
    ) return SYS_REFCURSOR
    is pc_user_info SYS_REFCURSOR;
    begin
        open pc_user_info for
        select * from users where userid = ps_userid and password = ps_password;
        return pc_user_info;
    end authenticate_user;

    procedure validate_exist_user(
        ps_userid users.userid%type
    )
    is
        v_count number;
    begin
        select count(*) into v_count
        from users
        where userid = ps_userid;

        if v_count > 0 then
            RAISE_APPLICATION_ERROR(-20001, 'This user already exists.');
        end if;
    exception
        when NO_DATA_FOUND then
            null;
    end validate_exist_user;
end users_crud_package;
/
-- test
set serveroutput on
DECLARE
    v_user_info SYS_REFCURSOR;
    v_user users%ROWTYPE;
BEGIN
-- register test
--    users_crud_package.register_user('good1id', 'good1name', 'good1@example.com', 'good1pass');

-- login test
```

```
    v_user_info := users_crud_package.authenticate_user('jiyoonid', 'jiyoonpass');
    LOOP
        FETCH v_user_info INTO v_user;
        EXIT WHEN v_user_info%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('User ID: ' || v_user.userid);
        DBMS_OUTPUT.PUT_LINE('Username: ' || v_user.username);
        DBMS_OUTPUT.PUT_LINE('Email: ' || v_user.email);
        END LOOP;

    CLOSE v_user_info;
END;









--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- comments
--------------------------------------------------------------------------------
-- spec
CREATE OR REPLACE PACKAGE comments_curd_package
is
    FUNCTION sel_com_fun (pn_facility_id comments.facility_id%TYPE)
    RETURN SYS_REFCURSOR;
    PROCEDURE ins_com_poc (
        ps_content IN comments.content%TYPE,
        ps_createdby IN comments.createdby%TYPE,
        pn_facility_id IN comments.facility_id%TYPE,
        pn_users_id IN comments.users_id%TYPE
    );
    PROCEDURE upd_com_poc (
        ps_content IN comments.content%TYPE,
        pn_id IN comments.id%TYPE
    );
    PROCEDURE del_com_poc(pn_id comments.id%TYPE);
    PROCEDURE validate_exist_user(ps_users_id comments.users_id%TYPE);
    PROCEDURE validate_exist_facility(pn_facility_id comments.facility_id%TYPE);
end comments_curd_package;
-- body
CREATE OR REPLACE PACKAGE BODY comments_curd_package
IS
    FUNCTION sel_com_fun (pn_facility_id comments.facility_id%TYPE)
    RETURN SYS_REFCURSOR
    IS p_comment_list SYS_REFCURSOR;
    BEGIN
        OPEN p_comment_list FOR
        SELECT * FROM comments WHERE facility_id = pn_facility_id;
        RETURN p_comment_list;
    END sel_com_fun;

    PROCEDURE ins_com_poc (
        ps_content comments.content%TYPE,
        ps_createdby comments.createdby%TYPE,
        pn_facility_id comments.facility_id%TYPE,
        pn_users_id comments.users_id%TYPE
    )
    IS
    BEGIN
        validate_exist_user(pn_users_id);
        validate_exist_facility(pn_facility_id);
        INSERT INTO comments (content, createdat, createdby, modifiedat, facility_id, users_id)
        VALUES (ps_content, SYSDATE, ps_createdby, SYSDATE, pn_facility_id, pn_users_id);
    END ins_com_poc;

    PROCEDURE upd_com_poc (
        ps_content IN comments.content%TYPE,
        pn_id IN comments.id%TYPE
    )
```

```
    IS
    BEGIN
        UPDATE comments
        SET content = ps_content,
            modifiedat = SYSDATE
        WHERE id = pn_id;
    END upd_com_poc;

    PROCEDURE del_com_poc(pn_id comments.id%TYPE)
    IS
    BEGIN
        DELETE FROM comments
        WHERE id = pn_id;
    END del_com_poc;

    PROCEDURE validate_exist_user(ps_users_id comments.users_id%TYPE)
    IS
        v_count NUMBER;
    BEGIN
        SELECT count(*) INTO v_count
        FROM users
        WHERE id = ps_users_id;

        IF v_count = 0 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Userid does not exist.');
        END IF;
    EXCEPTION
        when NO_DATA_FOUND THEN
            null;
    END validate_exist_user;

    PROCEDURE validate_exist_facility(pn_facility_id comments.facility_id%TYPE)
    IS
        v_count NUMBER;
    BEGIN
        SELECT count(*) INTO v_count
        from facility
        where id = pn_facility_id;

        if v_count = 0 then
            RAISE_APPLICATION_ERROR(-20001, 'This facility does not exist.');
        end if;
    exception
        when NO_DATA_FOUND then
            null;
    end validate_exist_facility;

end comments_curd_package;
-- test
set serveroutput on
DECLARE
    v_comment_list SYS_REFCURSOR;
    v_comment comments%ROWTYPE;
BEGIN
--    comments_curd_package.ins_com_poc(
--        'this is comment',
--        'goodid',
--        999999999,
--        1);
    v_comment_list := comments_curd_package.sel_com_fun(3);
    LOOP
        FETCH v_comment_list INTO v_comment;
        EXIT WHEN v_comment_list%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('User ID: ' || v_comment.content);
        END LOOP;
    CLOSE v_comment_list;

--    comments_curd_package.upd_com_poc(
--        'this is comment',
--        1,
--        1);
--    comments_curd_package.del_com_poc(1, 1);
END;
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- preference
--------------------------------------------------------------------------------
-- spec
create or replace package preference_curd_package
is
    FUNCTION sel_pre_poc(pn_facility_id preference.facility_id%TYPE)
    RETURN SYS_REFCURSOR;
    FUNCTION sel_pre_me_fun(pn_facility_id preference.facility_id%TYPE, pn_users_id preference.users_id%TYPE)
    RETURN preference.liked%TYPE;
    PROCEDURE ins_pre_poc(
        pc_liked preference.liked%TYPE,
        pn_facility_id preference.facility_id%TYPE,
        pn_users_id preference.users_id%TYPE
    );
    PROCEDURE upd_pre_poc(
        pc_liked preference.liked%TYPE,
        pn_id preference.id%TYPE
    );
end preference_curd_package;
-- body
create or replace package body preference_curd_package
is
--    FUNCTION sel_pre_poc(pn_facility_id preference.facility_id%TYPE)
--    RETURN SYS_REFCURSOR;
    FUNCTION sel_pre_me_fun(pn_facility_id preference.facility_id%TYPE, pn_users_id preference.users_id%TYPE)
    RETURN preference.liked%TYPE
    IS v_liked preference.liked%TYPE;
    BEGIN
        select liked into v_liked from preference where facility_id = pn_facility_id AND users_id = pn_users_id;
        return v_liked;
    END sel_pre_me_fun;

    PROCEDURE ins_pre_poc(
        pc_liked preference.liked%TYPE,
        pn_facility_id preference.facility_id%TYPE,
        pn_users_id preference.users_id%TYPE
    )
    IS
    BEGIN
        INSERT INTO preference (liked, facility_id, users_id)
        VALUES (pc_liked, pn_facility_id, pn_users_id);
    END ins_pre_poc;

    PROCEDURE upd_pre_poc(
        pc_liked preference.liked%TYPE,
        pn_id preference.id%TYPE
    )
    IS
    BEGIN
        UPDATE preference
        SET liked = pc_liked
        WHERE id = pn_id;
    END upd_pre_poc;
end preference_curd_package;
-- test
set serveroutput on
DECLARE
    v_pre_count NUMBER;
    v_liked preference.liked%TYPE;
BEGIN
    v_liked := preference_curd_package.sel_pre_me_fun(2, 1);
--    preference_curd_package.sel_pre_poc(1, v_pre_count);
    DBMS_OUTPUT.PUT_LINE(v_liked);
--
```

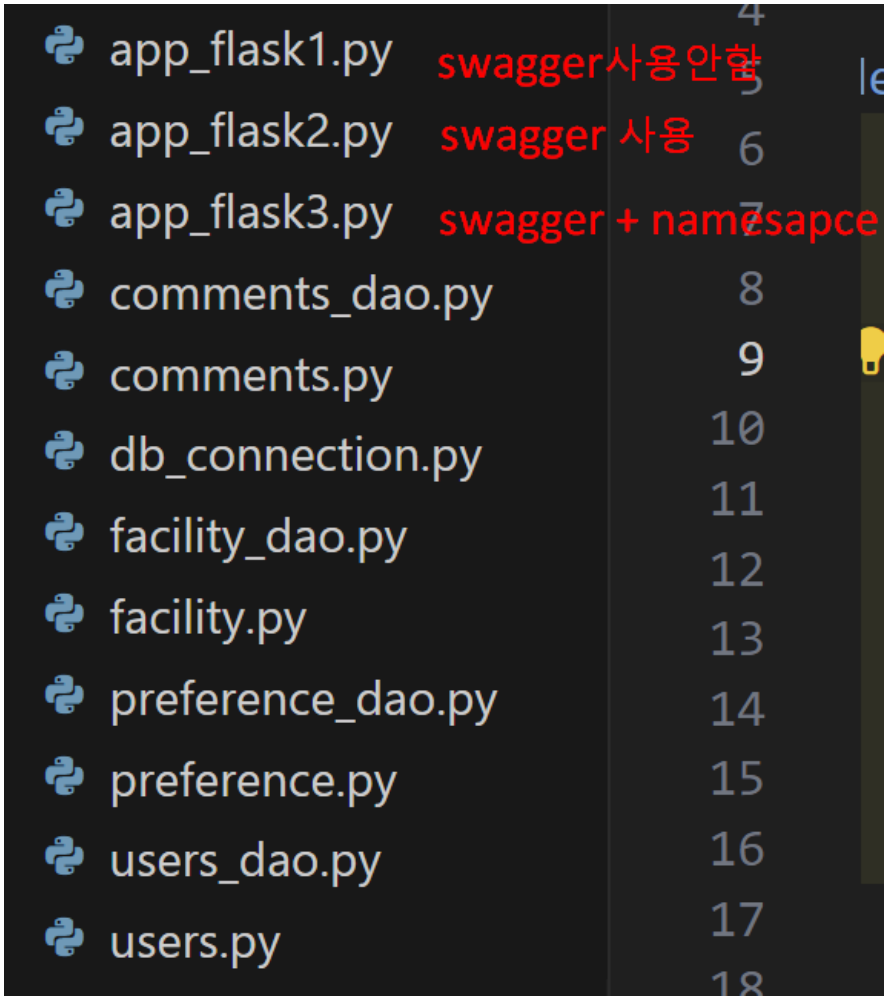
```
--    preference_curd_package.upd_pre_poc('F', 1, 1);
END;
```

백엔드

- python
- flask
- swagger

▼ 구조

파일 구조



아래의 명령코드 실행

```
python app_flask3.py
```

첫 진입 페이지인 `app_flask3.py`에서 네임스페이스에 따라 분리

```python
from flask import Flask
from flask_restx import Api
from flask_cors import CORS

import facility
import users
import comments
import preference

app = Flask(__name__)
CORS(app, origins="*")

@app.route('/')
def index():
    return 'Welcome My App'

api = Api(
    app, version='1.0', title='My App API',
    description='A simple My App API',
    doc='/api/'
)

# Add the namespaces
api.add_namespace(facility.ns)
api.add_namespace(users.ns)
api.add_namespace(comments.ns)
api.add_namespace(preference.ns)

if __name__ == "__main__":
    app.run(debug=True)
```

다음 진입페이지인 `facility.py`를 예시로 든다.

`facility.py`에서는 라우팅 처리를 하며 데이터베이스 호출로직은 `facility_dao.py` 에서 한다.

```python
from flask import request
from flask_restx import Namespace, Resource
import facility_dao as facility

ns = Namespace("facility", description="Facility operations")


@ns.route("/all")
class GetAllFacility(Resource):
    def get(self):
        category = request.args.get("category")
        address = request.args.get("address")
        return facility.get_all_facility(category, address)


@ns.route("/")
class GetFacility(Resource):
    def get(self):
        name = request.args.get("name")
        return facility.get_facility(name)
```

```python
from flask import jsonify, make_response
import db_connection as db


def get_all_facility(category, address):
    connection = db.create_connection()
    cursor = connection.cursor()
    facilities_cursor = cursor.callfunc(
        "facility_crud_package.sel_fac_fun", db.CURSOR, [category, address]
    )
    facilities = []
    for row in facilities_cursor:
        row_as_dict = {
            desc[0]: value for desc, value in zip(facilities_cursor.description, row)
        }
        facilities.append(row_as_dict)
    return make_response(jsonify(facilities), 200)
```

프론트앤드

- 사용기술
  - Vue3(composition api)
  - pinia
  - vue-router

    - bootstrap5
    - eslint
    - prettier
- 시연

성장🔥

- 데이터베이스 설계는 처음
    - 논리모델, 관계모델, 물리모델은 굉장히 어렵고 공부해야할 것이 많다는 것을 깨달음.
    - 설계 덕분에 curd를 구현하는데 쉬었음.
    - 설계가 중요하다는 것을 깨달음
- chatGPT의 활용능력 향상
- 공부방법(책을 다 사라)

미완 작업

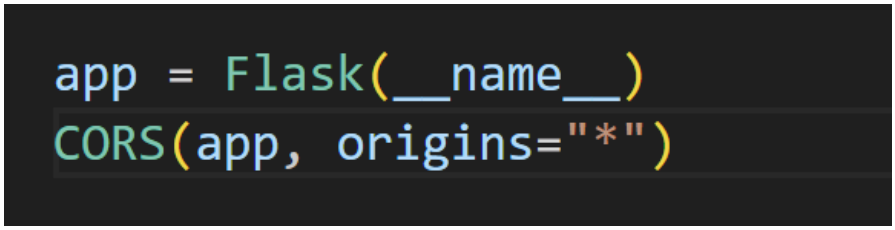
1. 기능구현 완료
    - 좋아요 조회
    - 좋아요 변경
2. map 에러 수정
3. 새로고침 시 로그아웃 처리

아직도 이해되지않는 문제

▼ CORS

백엔드는 5000포트 사용, 프런트앤드는 3000포트를 사용하고 있으므로 CORS 발생

1. flask에 flask_cors를 적용



회원가입, 로그인은 성공하였다. 그러나 댓글작성에서 CORS가 발생했다.

회원가입, 로그인 API와 댓글작성 API는 동일하게 post 메서드를 사용하며 PLSQL부터 백엔드 로직까지 구조가 동일하며 프런트앤드에서의 요청 또한 같은데 왜 댓글작성에서 계속 CORS에러가 발생하는지 이해할 수 없었다.

깃허브의 이슈를 찾아봐도 해결하지 못한사람들이 많았다. 결국 프런트에서 proxy를 설정하여 해결할 수 있었다.

프로젝트를 끝내야했기 때문에 더이상 시간을 지체할 수 없어 proxy로 해결했지만, 아직도 왜 댓글작성에서만 계속 CORS에러가 나는지 이해할 수 없다.