RESTful Api and DB Architecture

BE: 1번째 세션

NEXT X LIKELION 김동민

환영합니다!

Backend란?

백엔드의 길은...

2022 백엔드 로드맵 && 우리의 커리큘럼

[2022 백엔드 로드맵]

- 영문 : https://roadmap.sh/backend
- 한글 : https://imsoncod.tistory.com/24

[2학기 커리큘럼]

- https://www.notion.so/minqrii/f5c1291f15994809a427567f7d62faed

오늘의 목표

일찍 끝내기(?)

1. DRF(Django REST Framework)란?

1-1. REST api란?

1-2. Postman 사용하기

2. 요구사항에 맞춰 DB 구조 작성하기(실습)

2-1. ERD란?

2-2. ERD를 Django models.py로 구현하기

DRF란?

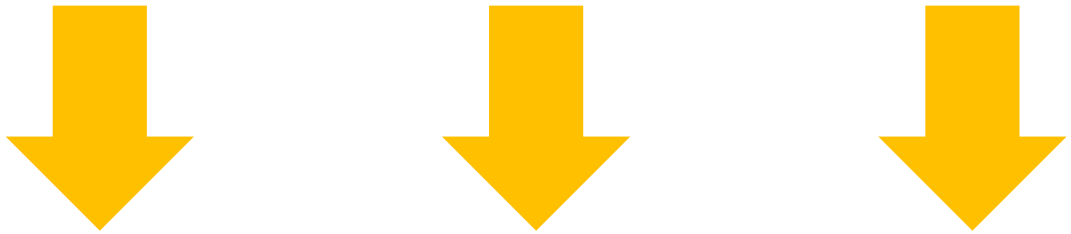
Django REST Framework

DRF란?

Django REST Framework

*“The Django REST framework (DRF) is a toolkit built on top of the Django web framework that reduces the amount of code you need to write to create **REST** interfaces.”*

출처: https://realpython.com/lessons/building-drf-overview/



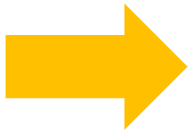
***REST** interface(API) 작성에 도움을 주는,*
Django web Framework 기반의 toolkit

REST api란?

REST란?

{ REST API }

Representational State Transfer API



“자원(resource)의 표현(representation)에 의한 상태(state)를 전달하는 API”

REST란?

자원, 자원의 표현, 상태

- **자원(resource) : 해당 소프트웨어가 관리하는 모든 것**
  - ✓ 텍스트, 그림, ……

- **자원의 표현(representation) : 그 자원을 표현하기 위한 이름**
  - ✓ DB의 학생 정보가 자원(resource)이라면, 그 자원의 표현을 `students`라 정하자!

- **상태(state) : 자원의 정보**
  - ✓ 데이터가 요청되는 시점의 자원의 상태(정보)

REST api란?

REST란?

{ REST API }

Representational State Transfer API

“자원(resource)의 표현(representation)에 의한 상태(state)를 전달하는 API”

“자원의 이름”

“자원의 정보”

“자원을 이름으로 구분하여 해당 자원의 정보를 주고받는 것!”

REST의 구성요소

자원, 행위, 표현

1. 자원(resource)

2. 행위(Verb)

3. 표현(Representation of Resource)

REST의 구성요소: 자원

자원, 행위, 표현

- **자원 : URI (Uniform Resource Identifier)**
  - ✓ URL(Uniform Resource Locator)의 상위 개념!
  - ✓ Server에 존재하는 모든 자원의 고유한 ID
  - ✓ Client는 이 URI를 이용해서 자원을 지정하고, 해당 자원의 정보에 대한 조작을 server에 요청
  - ✓ Ex) /students/1

REST의 구성요소: 자원

자원, 행위, 표현

- **행위 : HTTP Methods**
  - ✓ GET : 정보를 “조회”하는 method
  - ✓ POST : 정보를 “등록”하는 method
  - ✓ PUT : 정보를 “수정”하는 method
  - ✓ DELETE : 정보를 “삭제”하는 method
  - ✓ ***cf. HEAD, OPTIONS, PATCH, CONNECT***

REST의 구성요소: 자원

자원, 행위, 표현

- **표현 : 자원의 표현 방식**
  - ✓ JSON
  - ✓ XML
  - ✓ TEXT
  - ✓ RSS
  - ✓ ……

2주차에서 자세히 다룰 예정입니다 ☺

REST의 개념 끝!

이제 RESTful한 API를 설계해 보아요!!

RESTful API의 설계 규칙

URI, HTTP METHOD

✓ URI는 **정보의 자원을 표현**하여야 한다!

✓ 자원에 대한 **행위는 HTTP Method로 표현**하여야 한다!

RESTful API의 설계 규칙

1학기 때 배운 방식을 개선해보자!

GET /list

POST /new

GET /detail/{post_pk}

POST /edit/{post_pk}

POST /delete/{post_pk}

RESTful API의 설계 규칙

1학기 때 배운 방식을 개선해보자!

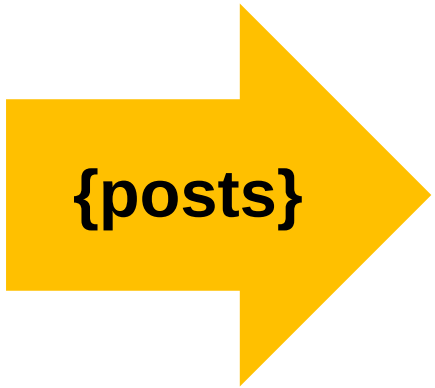
GET /list

POST /new

GET /detail/{post_pk}

POST /edit/{post_pk}

POST /delete/{post_pk}



GET /posts

POST /posts

GET /posts/{post_pk}

PUT /posts/{post_pk}

DELETE /posts/{post_pk}

그런데……

내가 설계한 API, 잘 동작할까?

Api가 잘 작동하는지 확인하려면…

버튼 만들기 → 버튼 누르면 요청이 보내지도록 코드 짜기

→ 버튼 눌러 요청 보내기 → 받은 응답 print하기

‘간편하게 테스트해 볼 수 없을까…?’

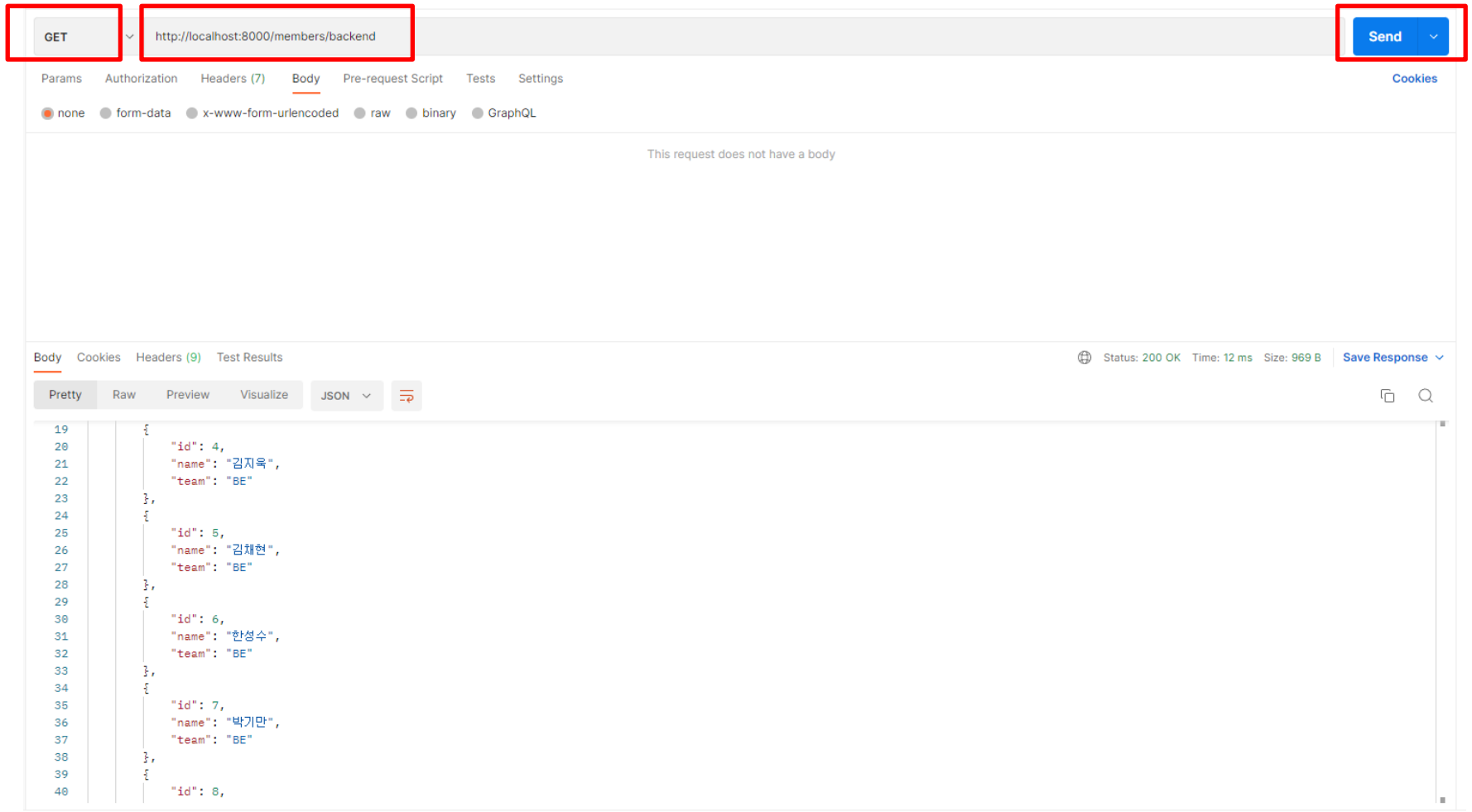
Postman

설계한 API 테스트하기

**“API 개발을 보다 빠르고 쉽게 구현할 수 있도록 도와주며,
개발된 API를 테스트하여 문서화 또는 공유할 수 있도록 도와주는 플랫폼”**

Postman

Postman 실습



직접 사용해 봅시다!! ☺

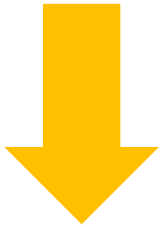
1교시 끝!

이제 2교시로!

ERD란?

Entity Relationship Diagram

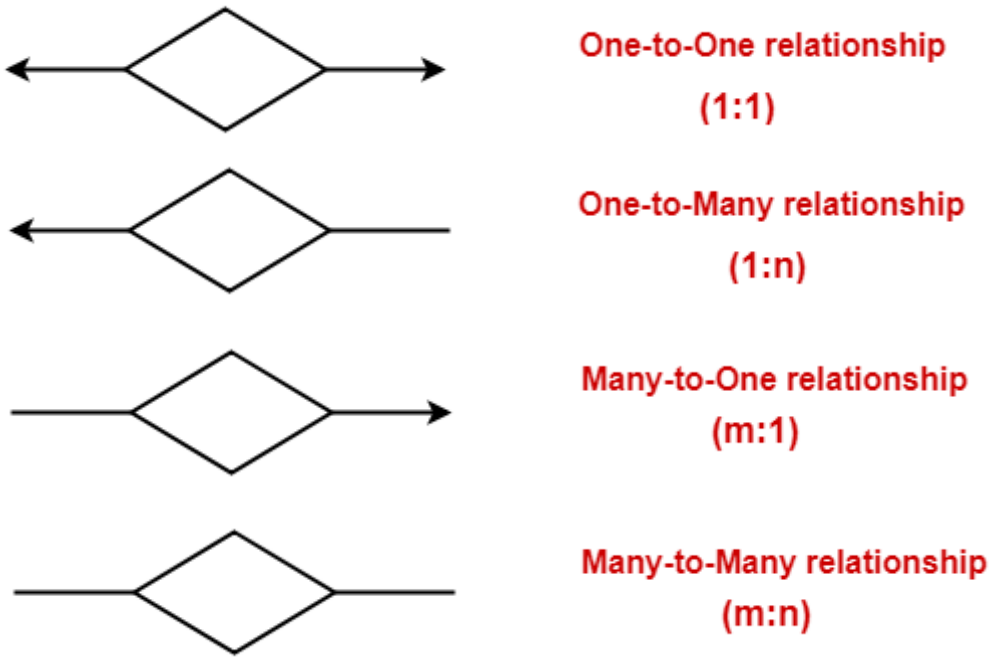
Entity Relationship Diagram



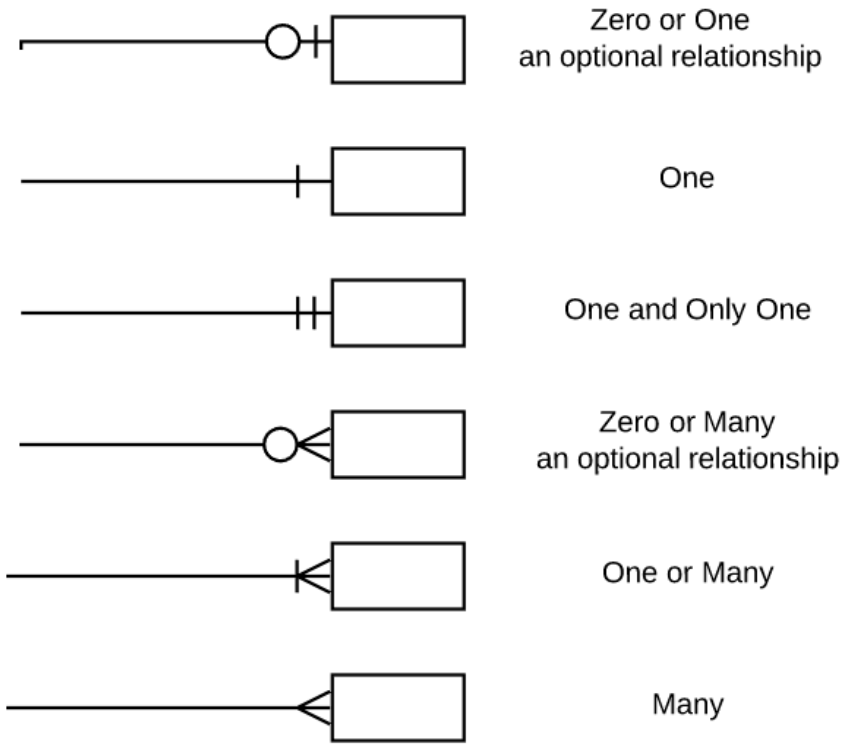
*“엔티티(데이터베이스의 테이블) 간의 관계(**relationship**)을 표현한 다이어그램”*

ERD 표기법의 종류

Peter Chen, Crowfoot



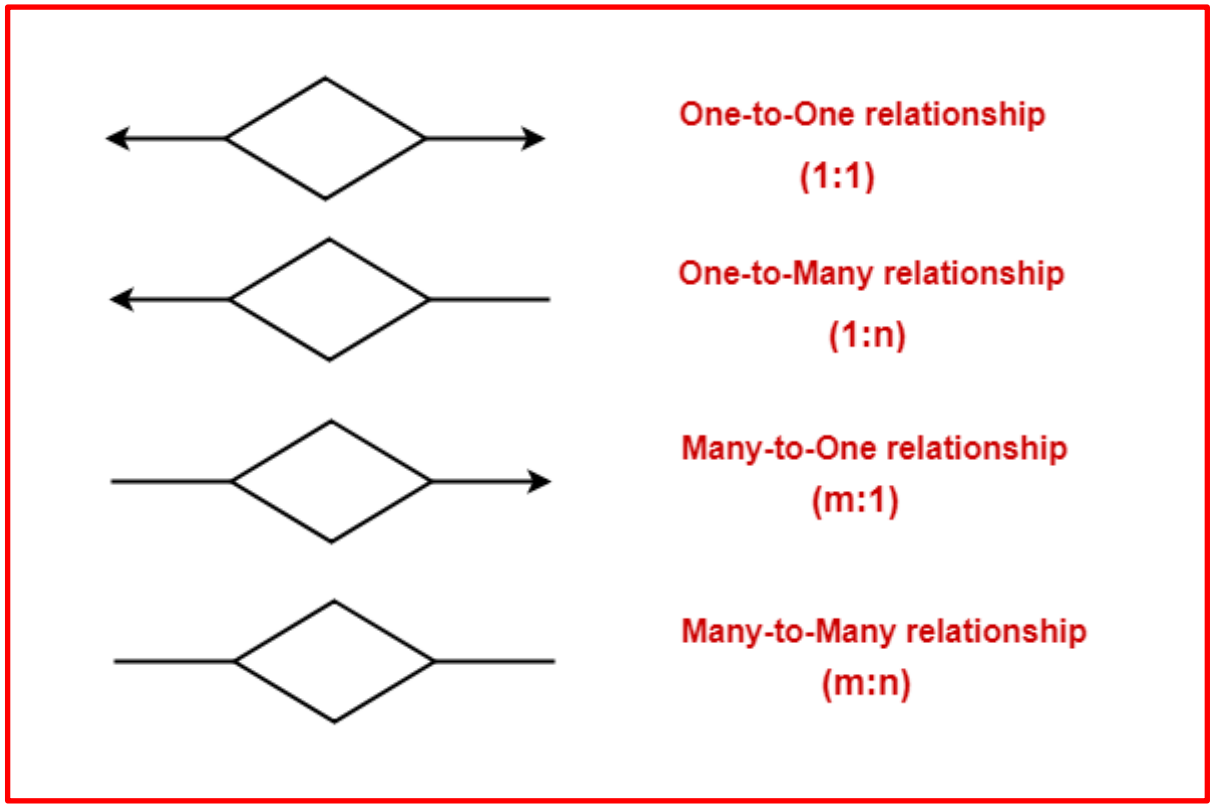
<Peter-Chen>



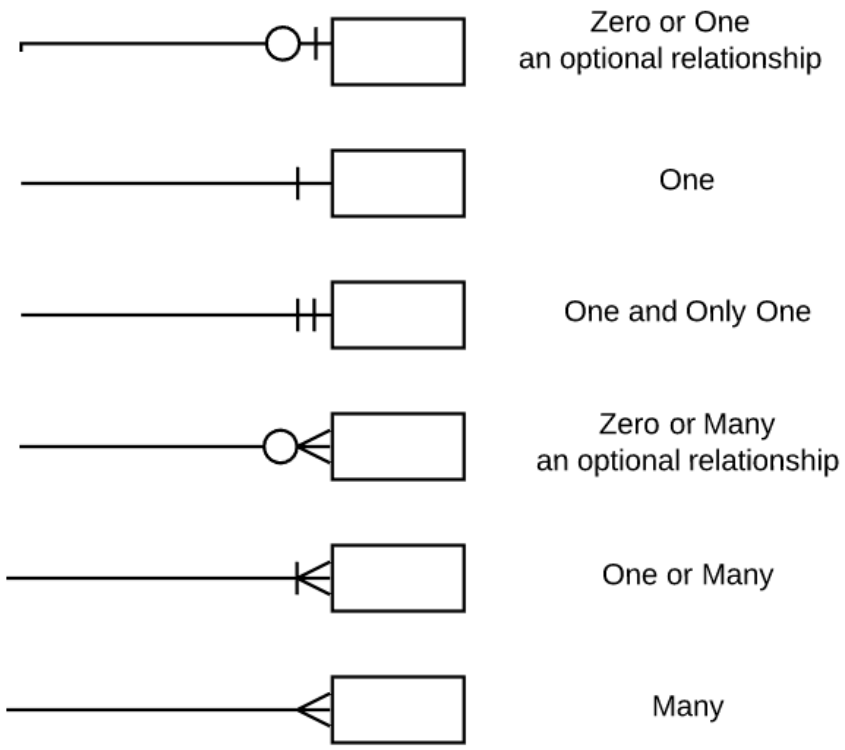
<Crow-foot>

ERD 표기법의 종류

Peter Chen, Crowfoot



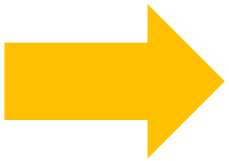
<Peter-Chen>

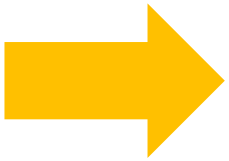


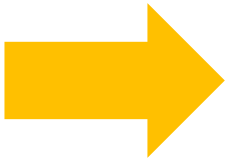
<Crow-foot>

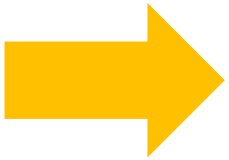
Peter-Chen 표기법

ERD 표기법









“일대일 관계”

“일대다 관계”

“다대일 관계”

“다대다 관계”

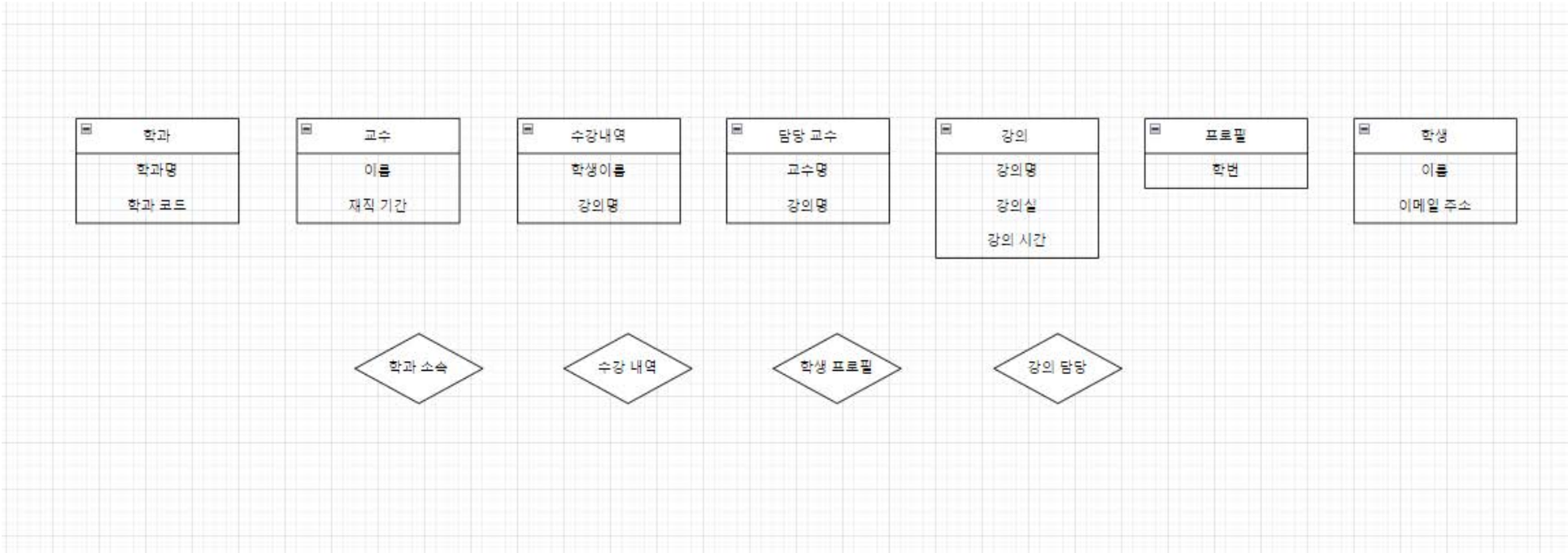
직접 그려봅시다!

요구사항으로 ERD 실습하기

ERD 실습 준비

요구사항으로 ERD 실습하기

1. https://app.diagrams.net/ 접속하기
2. [기존 다이어그램 열기]
3. ERD_practice.drawio 파일을 열어주세요 ☺



Client 요구사항

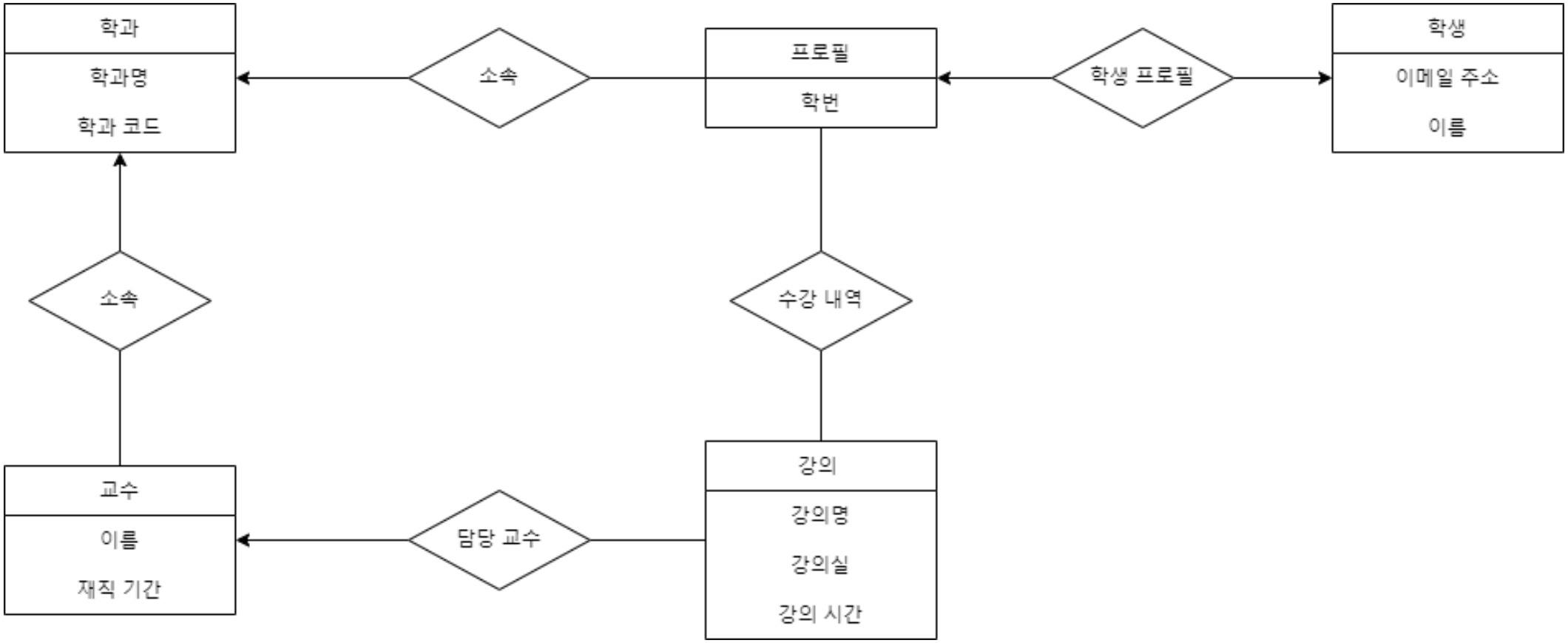
ERD 실습하기

1. 멋사대학교에서는 재학생들의 정보, 개설 강의 정보, 재직 중인 교수들의 정보를 관리해야 합니다.
2. **개설 강의에 대한 재학생들의 수강 내역을 조회할 수 있는 기능이 필요합니다.**
3. 재학생 정보로는 이름, 학번, 이메일 주소와 소속 학과를 관리하지만, 특정 강의의 수강생 목록을 조회할 때는 개인 정보 보호를 위해 학생 프로필 정보인 '학번'만을 조회합니다.
4. 교수자 정보로는 이름, 재직 기간, 소속 학과를 관리합니다.
5. 강의 정보로는 강의명, 강의실 정보, 강의시간을 관리합니다.
6. 하나의 강의는 1명의 교수자가 담당하는 것이 원칙이며, 매 학기 모든 교수자는 최소 1개 이상의 강의를 의무적으로 개설합니다.

- ✓ **HINT 1. 모든 엔티티(테이블)을 다 사용하지는 않습니다~!**
- ✓ **HINT 2. 두 번 이상 사용되는 관계표현(다이아몬드 모양!)이 있을 수 있습니다.**

예시 다이어그램

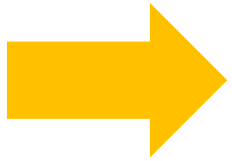
다들 잘 그리셨나요? ☺



DB 설계 내역을 Django models로 구현하기(마지막!)

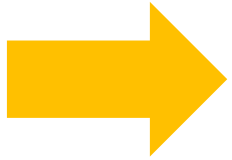
다음 주 예고...!!

"일대일 관계"



```python
class Profile(models.Model):
    student = models.OneToOneField(
        Student,
        on_delete=models.CASCADE,
        primary_key=True
    )
```

"다대다 관계"



```python
class Course(models.Model):
    name = models.CharField(max_length=50)
    classroom = models.CharField(max_length=100)
    time = models.CharField(max_length=100)
    students = models.ManyToManyField(Profile)
```

"일대다 관계"는 다들 아시죠~? ☺

예시 답안

DB 설계 내역을 Django models로 구현하기

```python
class Department(models.Model):
    dept_name = models.CharField(max_length=50)
    dept_code = models.CharField(max_length=30)


class Student(models.Model):
    name = models.CharField(max_length=50)
    email = models.EmailField(max_length=50, unique=True)


class Profile(models.Model):
    student = models.OneToOneField(
        Student,
        on_delete=models.CASCADE,
        primary_key=True
    )
    department = models.ForeignKey(
        Department,
        on_delete=models.SET_NULL, null=True)
```

```python
class Course(models.Model):
    name = models.CharField(max_length=50)
    classroom = models.CharField(max_length=100)
    time = models.CharField(max_length=100)
    students = models.ManyToManyField(Profile)


class Professor(models.Model):
    name = models.CharField(max_length=50)
    years = models.PositiveSmallIntegerField()
    department = models.ForeignKey(
        Department,
        on_delete=models.SET_NULL, null=True)
```

백엔드 첫 번째 미니(?)세션 끝!

수고하셨습니다!!

과제 공지! (~ 9/15 스터디 전까지)

파이팅!

1. 다음 페이지의 [요구사항]에 맞춰 Django 프로젝트에서 models.py를 작성하기
2. DRF 공식문서 tutorial 1 ~ 3 예습하기
3. 유튜브 강의 챕터 1 ~ 6 예습하기 (1시간 정도 분량)

    ~~(자막 없는 거 골라서 죄송해요...)~~

[발표 과제] 1주차 내용 정리 + α 해서 다음 주에 발표하기 ☺ (해당하는 분들만!)

[선택과제]

1. 작성한 models.py에 따라 ERD 그려보기 (캡처해서 제출해 주세요!)
2. Django 공식문서 – ManyToManyField 예습해 보기

서버 요구사항

궁금한 점은 질문 부탁드려요!

1. **멋사 백엔드 스터디에서는 게시판 서비스를 지원하는 서버를 구축하려고 합니다!**
2. 게시글(Post) 및 댓글(Comment)에 대한 CRUD(CREATE, READ, UPDATE, DELETE) 기능이 필요합니다.
3. 게시글은 카테고리로 구분되어야 하며, '카테고리 이름' 혹은 '카테고리 코드'로 특정 카테고리의 게시글만을 조회할 수 있어야 합니다.
4. 사용자(User) 정보를 관리해야 하고, 로그인한 사용자만이 게시글 및 댓글을 작성할 수 있습니다.
5. 사용자 정보로는 사용자의 이름, 아이디, 비밀번호, 그리고 사용자의 전공을 관리합니다.
6. 사용자의 전공 정보로는 본전공 외에 이중/복수/융합/심화 전공 정보를 포함해야 합니다.
7. 로그인한 사용자는 게시글에 "좋아요"를 표시하거나, 해제할 수 있어야 하며, 게시글에는 "좋아요를 누른 사용자 수"가 표시되어야 합니다.

과제 관련 링크

파이팅!

[DRF 공식문서 Tutorial]

https://www.django-rest-framework.org/tutorial/1-serialization/

[유튜브 강의]

https://www.youtube.com/watch?v=B38aDwUpcFc&t=3821s