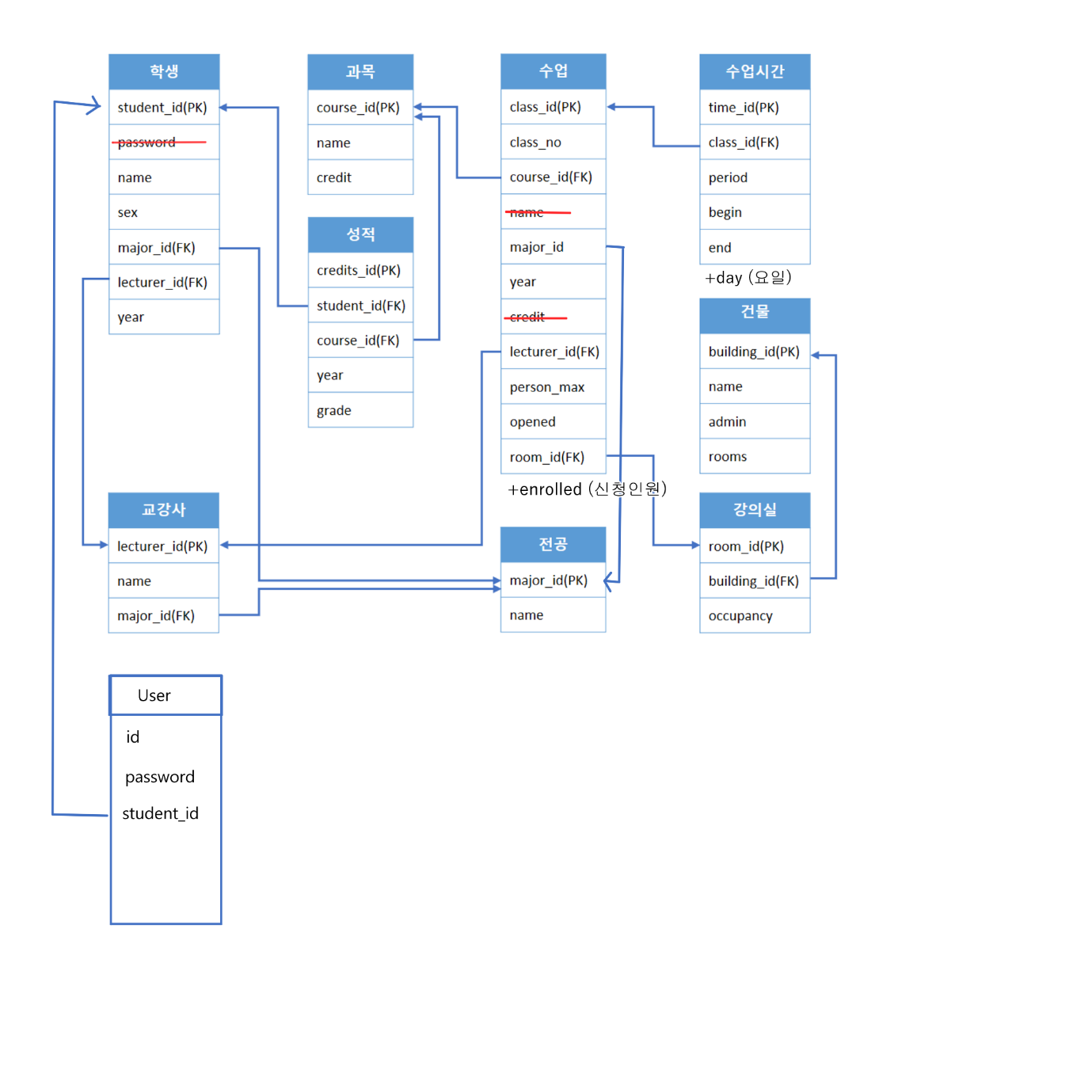
모의 수강신청 사이트 제작

작성자: 2021038131 장준혁

연락처: [ggogono3@naver.com](mailto:ggogono3@naver.com)

시작일: 2022년 10월 13일

# Schema 디자인



실제 구현은 sugang/home/models.py에서 확인 가능하다.

대부분은 과제 가이드라인을 기반으로 일부 수정했다.

수업 스키마의 name과 credit 어트리뷰트는 과목과 중복된다. 따라서 제거했다. 수업시간 스키마에서 begin과 end에서 요일 정보의 중복이 발생한다. 이를 day라는 별도의 어트리뷰트를 만들어서 중복을 제거했다. 수업 테이블에 현재 수강신청을 한 학생 수에 대한 어트리뷰트를 추가했다.

학생의 테이블에서 비밀번호를 암호화하지 않고 그대로 노출하고 있어서 password를 제거했다. PBKDF2 알고리즘으로 SHA256 해시를 적용해서 password를 암호화한 후 별로의 계정 정보(auth\_user) 테이블에 저장한다. 또한 실제 학교 계정에서 초기화되지 않은 계정의 아이디는 학번과 같지만 이후 학번과 다르게 변경할 수 있다. 따라서 로그인하는 아이디도 별도의 테이블에서 관리해야한다.

DB Insert 작업의 편의성을 위해 모델들의 의존 관계에 따라 우선순위(Tier)를 부여했다.

티어가 낮은 모델은 더 높은 티어의 모델에 의존적이다.

DB 삽입 명령을 수행할 때 높은 티어부터 실행해야한다.

Tier 1 Model: 전공, 건물, 과목

Tier 2 Model: 교강사, 강의실

Tier 3 Model: 학생, 수업

Tier 4 Model: 성적, 수업시간, +auth\_user

DB 구조를 생성하는 코드는 다음과 같다.

**from** **django.db** **import** models

*# Create your models here.*

**from** **django.template.defaultfilters** **import** default

*# 티어가 낮은 모델은 더 높은 티어의 모델에 의존적이다.*

*# 즉, DB 삽입 명령을 수행할 때 높은 티어부터 실행해야한다.*

*# Tier 1 Model: 전공, 건물, 과목*

*# Tier 2 Model: 교강사, 강의실*

*# Tier 3 Model: 학생, 수업*

*# Tier 4 Model: 성적, 수업시간, +auth\_user*

*# ----------- Tier 1 ---------------*

*# Tier 1 Model: 전공, 건물, 과목*

**class** **Major**(models.Model):

*"""*

*major\_id,name*

*1,건설환경공학과*

*"""*

major\_id = models.IntegerField(primary\_key=**True**)

name = models.CharField(max\_length=100)

**class** **Building**(models.Model):

*"""*

*building\_id,name,admin,rooms*

*305,IT / BT,공과대학,19*

*"""*

building\_id = models.IntegerField(primary\_key=**True**)

name = models.CharField(max\_length=100)

admin = models.CharField(max\_length=100)

rooms = models.IntegerField()

**class** **Course**(models.Model):

*"""*

*course\_id,name,credit*

*CIE3022,철근콘크리트구조설계,3*

*"""*

course\_id = models.CharField(primary\_key=**True**, max\_length=20)

name = models.CharField(max\_length=100)

credit = models.IntegerField()

*# ----------- Tier 2 ---------------*

*# Tier 2 Model: 교강사, 강의실*

**class** **Lecturer**(models.Model):

*"""*

*lecturer\_id,name,major\_id*

*2001001001,조병완,1*

*"""*

lecturer\_id = models.IntegerField(primary\_key=**True**)

name = models.CharField(max\_length=100)

major = models.ForeignKey(Major, on\_delete=models.CASCADE)

**class** **Room**(models.Model):

*"""*

*room\_id,building\_id,occupancy*

*1,305,140*

*"""*

room\_id = models.IntegerField(primary\_key=**True**)

building = models.ForeignKey(Building, on\_delete=models.CASCADE)

occupancy = models.IntegerField()

*# ----------- Tier 3 ---------------*

*# Tier 3 Model: 학생, 수업*

**class** **Student**(models.Model):

*# csv 라이브러리로 파싱 후수동 셋업 필요*

*"""*

*student\_id,password,name,sex,major\_id,lecturer\_id,year*

*2018003125,125125125,정남아,female,44,2001032011,4*

*"""*

student\_id = models.IntegerField(primary\_key=**True**)

*# password : 보안상 auth\_user 테이블로 분리함*

name = models.CharField(max\_length=100)

sex = models.CharField(max\_length=20)

major = models.ForeignKey(Major, on\_delete=models.CASCADE)

lecturer = models.ForeignKey(Lecturer, on\_delete=models.CASCADE)

year = models.IntegerField()

**class** **Class**(models.Model):

*"""*

*class\_id,class\_no,course\_id,name,major\_id,year,credit,lecturer\_id,person\_max,opened,room\_id*

*8831,10003,CIE3022,철근콘크리트구조설계,1,1,3,2001001001,3,2022,169*

*"""*

class\_id = models.IntegerField(primary\_key=**True**)

class\_no = models.IntegerField()

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

*# name = models.CharField(max\_length=100) : DB 스키마에서 삭제 가능 Course에 이미 있음*

*# major\_id: # FK, DB 스키마에서 화살표 빠진거 보충 필요*

major = models.ForeignKey(Major, on\_delete=models.CASCADE)

year = models.IntegerField()

*# credit = models.IntegerField() : DB 스키마에서 삭제 가능 Course에 이미 있음*

lecturer = models.ForeignKey(Lecturer, on\_delete=models.CASCADE)

person\_max = models.IntegerField()

opened = models.IntegerField()

enrolled = models.IntegerField(default=0) *# 추가 가능*

room = models.ForeignKey(Room, on\_delete=models.CASCADE)

*# ----------- Tier 4 ---------------*

*# Tier 4 Model: 성적, 수업시간, +auth\_user(장고 기본 auth DB 이용)*

**class** **Credits**(models.Model):

*"""*

*credits\_id,student\_id,course\_id,year,grade*

*1,2018003125,GEN5026,2022,B0*

*"""*

credits\_id = models.IntegerField(primary\_key=**True**)

student = models.ForeignKey(Student, on\_delete=models.CASCADE)

course = models.ForeignKey(Course, on\_delete=models.CASCADE)

year = models.IntegerField()

grade = models.CharField(max\_length=2)

**class** **Time**(models.Model):

*# csv 라이브러리로 파싱 후 수동 셋업 필요*

*"""*

*input:*

*time\_id,class\_id,period,begin,end*

*1,8831,1,1900-01-02T05:30:00.000Z,1900-01-02T07:00:00.000Z*

*"""*

*"""*

*DB:*

*time\_id,class\_id,period,day,begin,end*

*1,8831,1,5,05:30:00,05:30:00*

*"""*

time\_id = models.IntegerField(primary\_key=**True**)

classInfo = models.ForeignKey(Class, on\_delete=models.CASCADE) *# class 는 예약어여서 사용할 수 없음*

period = models.IntegerField()

*# begin = models.DateTimeField() 스키마 수정 필요*

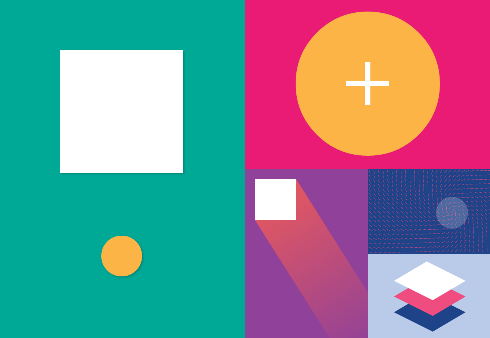
*# end = models.DateTimeField() 스키마 수정 필요*

day = models.IntegerField()

begin = models.TimeField()

end = models.TimeField()

# UI 디자인



구글의 Material Design 가이드라인을 따른다.



CSS 프레임워크인 Bootstrap을 기반으로 작업한다.



Hanyang Blue (#1D2475)를 대표색으로 사용한다. (ex. NavBar 배경색 등에 적용)

반응형 웹 기술을 통해 모바일 친화적인 UI를 구성한다.

(ex. 모바일 화면에서는 가로로 배치된 레이아웃이 세로로 변경된다.

Navbar의 요소들이 모바일 화면에서는 햄버거 버튼으로 축소된다.)

네이버 블로그나 카페처럼 페이지 오른쪽 블록에 사용자 정보를 표시하는 레이아웃을 둔다.

# Input data set

DB 삽입 작업에는 기존에 작성된 CSV 파일을 그대로 이용한다. 파이썬의 CSV 라이브러리를 이용해서 CSV 파일을 읽어온 후 현재 설계된 DB 스키마에 따라 데이터를 직접 파싱해 DB에 데이터를 삽입한다.

아래는 DB Insertion을 위해 CSV를 파싱하는 코드의 일부이다. 길이상 일부만 첨부했다.

Sugang/management/commands/insert\_test\_data.py 파일에서 전체를 확인할 수 있다.

**from** **django.core.management.base** **import** BaseCommand, CommandError

**from** **home.models** **import** \*

**import** **csv**

**class** **Command**(BaseCommand):

help = 'inserts test data set'

**def** handle(self, \*args, \*\*options):

print("insert major.csv")

**with** open('test\_data\csv\major.csv', 'r') **as** csvfile:

r = csv.reader(csvfile)

**for** row **in** r:

**if** row[0] == 'major\_id':

**continue**

major = Major()

*"""*

*major\_id,name*

*1,건설환경공학과*

*"""*

major.major\_id = row[0]

major.name = row[1]

major.save()

# 구현된 코드 설명

장고는 MVT 패턴을 따른다.

서버에 요청이 들어오면 url을 urls.py 파일에 따라 파싱하고 연결된 views.py의 함수로 보내고 views.py는 models.py를 통해 데이터베이스 관련 작업을 하고 유저가 원하는 데이터를 template에 적용해서 렌더링한 후 유저에게 응답을 보낸다.

# 빌드 및 실행 과정

1. 파이썬을 설치한다. 이때 환경변수 등록을 꼭 해야한다. (설치 시 ADD PATH 옵션 체크)
2. requirements.txt 파일에 입력된 라이브러리들을 설치한다. (freeze 명령어 이용해서 한번에 설치 가능)
3. .secrets와 같이 로컬에서 따로 설정해야 하는 파일들은 편의상 미리 만들어서 확장자 뒤에 .example을 붙여 놓았으니 .example을 확장자 뒤에서 제거하고 사용하면 된다.

# 요구사항

# 추가 기능

# URL 설계

// 실제 구현은 sugang/urls.py와 home/urls.py 에 있다.

null: home // 공지사항을 보여주는 메인 페이지, 로그인이 안되 있을 경우 로그인 페이지로 리다이렉션 된다.

login // 로그인

user // 사용자 정보 조회 및 수정

lectures // 강의 목록

lectures?subject=건축 ... // 검색 필터 기능

enroll/<number> // 강의 수강 신청시 post 요청을 보내는 링크

mylectures // 수강신청한 강의들

cancel // 강의 수강 취소시 post 요청을 보내는 링크

timetable // 시간표 확인

chat // ajax를 통한 실시간 채팅 페이지

admin // 관리자 페이지

statistics // 정보 분석 페이지