**Отчет о выполнении лабораторной работы №1**

Якухин Иван, 6303-010302D

*Идея: приложение с карточками для запоминания английских слов*

1. **Настройка БД**

Изначально при создании Django приложения я использовал базу данных Postgres развернутую локально на компьютере. После я создал образ БД в Docker и, запустив контейнер с БД, перенес туда данные с помощью экспорта/импорта(pg\_dump/pg\_restore).

Dockerfile выглядит следующим образом:

FROM postgres:latest  
WORKDIR postgres/  
EXPOSE 5432

Определяем базовое изображение, название директории, и внутренний порт для Postgres.

Создаем Docker образ командой build -t cardiogram-postgres .

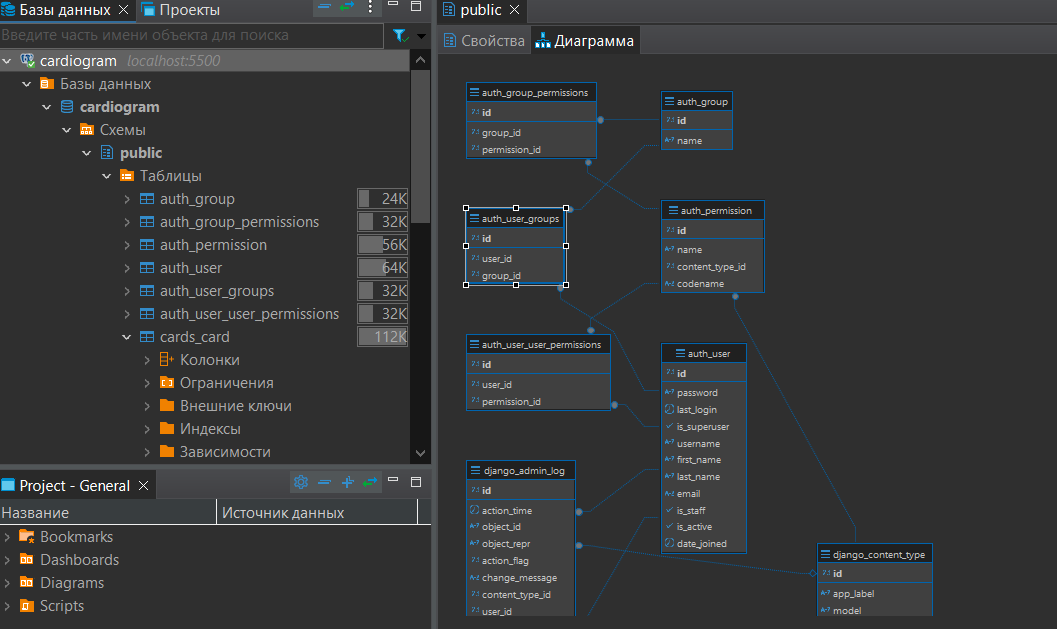
Запускаем контейнер с указанием файла с переменными среды run --env-file cardiogram/.env --name cardiogram-postgres-container -p 5500:5432 cardiogram-postgres

Внешний порт был изменен, чтобы не спутать этот сервер с другим при использовании например psql.

Копируем файл дампа в Docker контейнер из системы docker cp ../cardio.sql cardiogram-postgres-container:cardio.sql

Запускаем команду pg\_restore для импорта БД docker exec -i cardiogram-postgres-container pg\_restore -U postgres -d cardiogram ../cardio.sql

Проверяем доступ к БД через DBeaver.



1. **Разработка ORM моделей в DjangoORM**

Для слегка доработанной схемы БД из прошлой ЛР были созданы модели описывающие все сущности, использованные в схеме.

Модель сущности “Карта”:

class Card(models.Model):  
 card\_id = models.AutoField(primary\_key=True)  
 deck\_id = models.ForeignKey(Deck, models.CASCADE, null=False)  
 front\_text = models.CharField(max\_length=30, unique=True, null=False)  
 back\_text = models.CharField(max\_length=30, unique=True, null=False)  
 example\_usage = models.TextField(unique=True, null=True)  
  
 class Meta:  
 verbose\_name = 'Карта'  
 verbose\_name\_plural = 'Карты'  
  
 def \_\_str\_\_(self):  
 return self.front\_text

Модель сущности “Колода”:

class Deck(models.Model):  
 deck\_id = models.AutoField(primary\_key=True)  
 deck\_name = models.CharField(max\_length=100, unique=True, null=False)  
 description = models.TextField(unique=True, null=True)  
  
 class Meta:  
 verbose\_name = 'Колода'  
 verbose\_name\_plural = 'Колоды'  
  
  
 def \_\_str\_\_(self):  
 return self.deck\_name

Модель сущности “Прогресс пользователя”:

class Progress(models.Model):  
 progress\_id = models.AutoField(primary\_key=True)  
 user\_id = models.ForeignKey(get\_user\_model(), models.CASCADE, null=False)  
 card\_id = models.ForeignKey(Card, models.CASCADE, null=False)  
 attempts = models.IntegerField()  
 successful\_attempts = models.IntegerField()  
  
 class Meta:  
 verbose\_name = 'Прогресс'  
 verbose\_name\_plural = 'Прогресс'  
 unique\_together = [('user\_id','card\_id')]

Модель сущности пользователя реализована в Django “из коробки”, её я менять не стал.

Изменения были отправлены в базу данных с помощью миграций Django, командами python manage.py makemigrations и python manage.py migrate.

1. **Модель пользователя**

Как было сказано ранее, модель пользователя в Django уже реализована и хеширование паролей также встроено в нее изначально. Django использует функцию PBKDF2, которая в свою очередь использует хеш функцию sha256. Также Django предоставляет возможность сменить функцию хеширования, но меня устроила и эта.

Чтобы продемонстрировать что пароль хешируется ч вывожу в admin панель в таблицу с данными пользователей хеш пароля в том виде, в котором он хранится в БД.

1. **Функционал для работы с моделями**

Для заполнения БД я попросил deepseek сформировать для меня 2 txt фала с английскими словами и их переводами, а также файл с тестовыми данными пользователей.

После этого я создал файл миграции и поместил в него код для заполнения БД тестовыми данными. Пример файла:

def fill\_db(apps, schema\_editor):  
 cards = apps.get\_model('cards', 'Card')  
 decks = apps.get\_model('decks', 'Deck')  
 path = os.path.dirname(\_\_file\_\_)  
 eng = open(path+'/nature\_eng.txt', 'r')  
 ru = open(path+'/nature\_ru.txt', 'r', encoding='utf-8')  
 length = len(eng.readlines())  
 eng.seek(0)  
 for i in range(length):  
 b\_text, desc = ru.readline().split(" - ")  
 cards.objects.create(deck\_id = decks.objects.get(deck\_name = "Nature"), front\_text = eng.readline(),  
 back\_text = b\_text, example\_usage = desc)  
 eng.close()  
 ru.close()  
  
class Migration(migrations.Migration):  
  
 dependencies = [  
 ('cards', '0004\_auto\_20250308\_1933'),  
 ('decks', '0003\_alter\_deck\_deck\_id')  
 ]  
  
 operations = [  
 migrations.RunPython(fill\_db)  
 ]

Для работы с данными в Django предусмотрена admin панель. Я добавил в нее таблицы с используемыми данными и немного отредактировал их вид для лучшей читаемости.

