МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

КАФЕДРА ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Тема: «Разработка базы данных»

Дисциплина: «Технологии сетевого программирования»

Выполнил: Ашугян Т.А., гр.6303

Проверил: Кашапов А.И.

Самара 2025

СОДЕРЖАНИЕ

[Введение 3](#_Toc199592081)

[1 Описание базы данных 4](#_Toc199592082)

[1.1 Client 4](#_Toc199592083)

[1.2 Master 4](#_Toc199592084)

[1.3 User 5](#_Toc199592085)

[1.4 Service 5](#_Toc199592086)

[1.5 Appointment 5](#_Toc199592087)

[2 Скрипт для заполнения таблиц 7](#_Toc199592088)

[Приложение А 8](#_Toc199592089)

[Ссылка на GitHub с проектом 8](#_Toc199592090)

[Приложение Б 9](#_Toc199592091)

[Скрипт для заполнения таблиц 9](#_Toc199592092)

Введение

AutoService – приложение для записи на услуги в автосервис. Стек технологий приложения: Python, Django, PostgreSQL. Все компоненты конечной системы контейнеризованны с помощью Docker для обеспечения изоляции окружения, удобства развертывания и масштабирования.

Ссылка на GitHub страницу со всеми файлами проекта можно найти в приложении А.

1. Описание базы данных

Основу базы данных составляют 5 таблиц: Users, Client, Master, Service, Appointment. Каждая таблица этой базы данных является представлением определенных сущностей, связанных между собой отношениями. Ниже приведено более подробное описание каждой сущности и связей между ними.

* 1. Client

Хранит данные клиентов

Поля:

* User – связь со встроенной сущностью User;
* Name – имя клиента;
* Phone – номер телефон клиента;
* Last\_login –время последнего подключения клиента.

Сущность связана со встроенной сущностью User отношением один к одному.

Также связана с сущностью Appointment пользователя отношением один ко многим.

* 1. Master

Хранит данные мастеров.

Поля:

* User – связь со встроенной сущностью User;
* Full\_name – ФИО мастера
* Phone – номер телефона мастера;
* Experience – опыт работы мастера;

Сущность связана со встроенной сущностью User отношением один к одному.

Также связана с сущностью Appointment пользователя отношением один ко многим.

* 1. User

Встроенная сущность, которая содержит данные пользователей, взаимодействующих с приложением.

Поля:

* Id – уникальный идентификатор записи;
* Username – уникальное пользовательское имя, которое используется для аутентификации в приложении;
* First\_name – имя пользователя;
* Last\_name – фамилия пользователя;
* Email – электронная почта пользователя;
* Password – пароль от аккаунта пользователя.

Связан с Client и Master.

* 1. Service

Содержит информацию об услугах

Поля:

* Info – информация об услуге;
* Price – цена услуги;

Связана с сущностью Appointment пользователя отношением один ко многим.

* 1. Appointment

Содержит информацию о записях.

Поля:

* Client – записанный клиент;
* Date – дата записи;
* Time – время записи;
* Service – услуга записи;
* Master – мастер, предоставляющий услугу;
* Status – статус записи;

Связана с сущностями Client, Master, Service отношением один ко многим.

1. Скрипт для заполнения таблиц

Для заполнения таблиц данными в основном модуле приложения была создана директория “management/commands”, в которую был помещен скрипт, создающий записи таблиц из текстового файла с данными. Код скрипта приведен в приложении Б. Для их запуска была использована команда “python manage.py <название скрипта>”.

Приложение А

Ссылка на GitHub с проектом

https://github.com/Tigr195/AutoService

Приложение Б

Скрипт для заполнения таблиц

import os

from django.core.management.base import BaseCommand

from Auto\_Service.models import Client, Master, Service, Appointment

class Command(BaseCommand):

help = 'Заполняет базу данных данными из текстового файла'

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

file\_path = 'data.txt' # Путь к файлу с данными

if not os.path.exists(file\_path):

self.stdout.write(self.style.ERROR(f"Файл {file\_path} не найден!"))

return

with open(file\_path, 'r', encoding='utf-8') as file:

for line in file:

line = line.strip()

if not line:

continue

parts = line.split(',')

entity\_type = parts[0]

if entity\_type == 'Client':

self.create\_client(parts[1:])

elif entity\_type == 'Master':

self.create\_master(parts[1:])

elif entity\_type == 'Service':

self.create\_service(parts[1:])

elif entity\_type == 'Appointment':

self.create\_appointment(parts[1:])

else:

self.stdout.write(self.style.WARNING(f"Неизвестный тип сущности: {entity\_type}"))

continue

self.stdout.write(self.style.SUCCESS("База данных успешно заполнена!"))

def create\_client(self, data):

client\_id, name, phone, login, password = data

client = Client(client\_id=client\_id, name=name, phone=phone, login=login)

client.set\_password(password)

client.save()

self.stdout.write(self.style.SUCCESS(f"Создан клиент: {name}"))

def create\_master(self, data):

master\_id, full\_name, phone, experience, login, password = data

master = Master(

master\_id=master\_id,

full\_name=full\_name,

phone=phone,

experience=experience,

login=login

)

master.set\_password(password)

master.save()

self.stdout.write(self.style.SUCCESS(f"Создан мастер: {full\_name}"))

def create\_service(self, data):

service\_id, info, price = data

service = Service(service\_id=service\_id, info=info, price=price)

service.save()

self.stdout.write(self.style.SUCCESS(f"Создана услуга: {info}"))

def create\_appointment(self, data):

appointment\_id, client\_id, date, time, service\_id, master\_id, status = data

client = Client.objects.get(client\_id=client\_id)

service = Service.objects.get(service\_id=service\_id)

master = Master.objects.get(master\_id=master\_id)

appointment = Appointment(

appointment\_id=appointment\_id,

client=client,

date=date,

time=time,

service=service,

master=master,

status=status

)

appointment.save()

self.stdout.write(self.style.SUCCESS(f"Создана запись: {appointment\_id}"))