

Факультет инфокоммуникационных технологий.

Практическая работа №3 по теме:
«Реализация серверной части на Django REST» по дисциплине:
Web-программирование.

Выполнил: Кирильцев Роман
Группа: K33401

Преподаватель: Говоров Антон Игоревич

Санкт-Петербург, 2021

Цель работы: овладеть практическими навыками реализации серверной части (backend) приложений средствами Django REST framework.

Практическое задание :

Реализовать модель базы данных средствами DjangoORM .

Задание 10

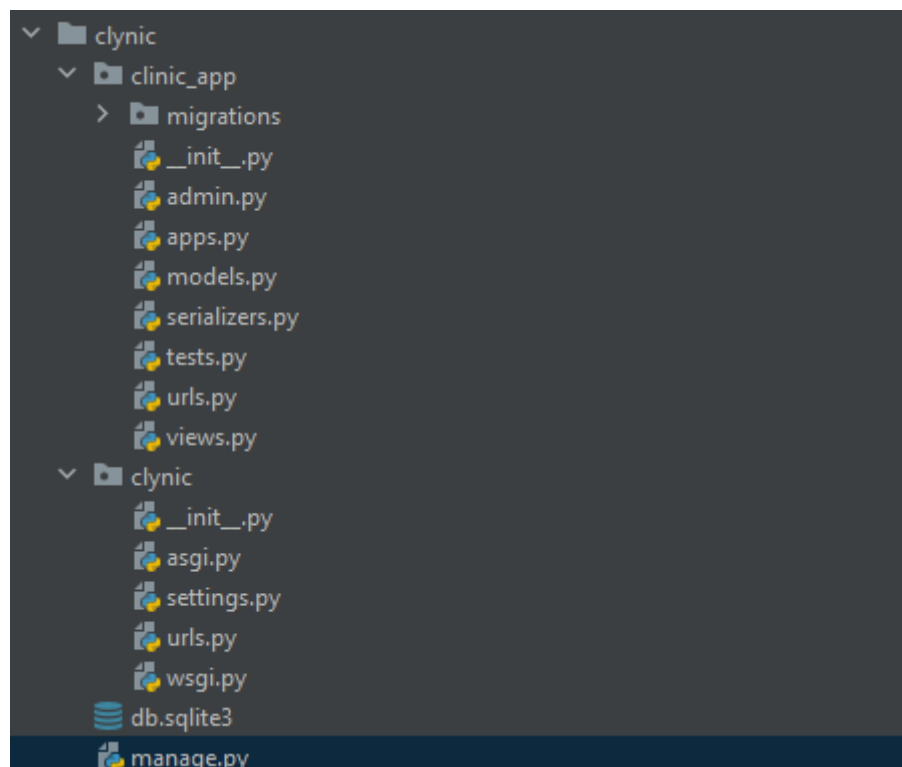
Создать программную систему, предназначенную для администратора лечебной клиники.

Прием пациентов ведут несколько врачей различных специализаций. На каждого пациента клиники заводится медицинская карта, в которой отражается вся информация по личным данным больного и истории его заболеваний (диагнозы). При очередном посещении врача в карте отражается дата и время приема, диагноз, текущее состояние больного, рекомендации по лечению. Так как прием ведется только на коммерческой основе, после очередного посещения пациент должен оплатить медицинские услуги (каждый прием оплачивается отдельно). Расчет стоимости посещения определяется врачом согласно прейскуранту по клинике.

Для ведения внутренней отчетности необходима следующая информация о врач: фамилия, имя, отчество, специальность, образование, пол, дата рождения и дата начала и окончания работы в клинике, данные по трудовому договору. Для каждого врача составляется график работы с указанием рабочих и выходных дней.

Прием пациентов врачи могут вести в разных кабинетах. Каждый кабинет имеет определенный режим работы, ответственного и внутренний телефон.

Структура проекта:



Реализованные сущности:

Django administration

Home > Clinic_App > Users

AUTH TOKEN

Tokens + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

CLINIC_APP

Cabinets + Add

Diagnosiss + Add

Medcards + Add

Meetings + Add

Patients + Add

Prices + Add

Schedules + Add

Users + Add

Создание записи в таблице doctor:

← → ↻ 127.0.0.1:8000/api/doctor/create/

Django REST frameworkКирильцев Роман Сергеевич

Doctor Api / Doctor Create Api

Doctor Create ApiOPTIONSGET

GET /api/doctor/create/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "username": "roman",
    "fio": "Кирильцев Роман Сергеевич",
    "speciality": "",
    "education": "",
    "work_years": 15
  }
]
```

Raw dataHTML form

Username

Required: 150 characters or fewer. Letters, digits and @/+/./_ only.

ФИО

Специальность

Уровень образованияВысшее

Трудовой стаж

POST

Просмотр созданных кабинетов:

inet Api - Django REST fram x +

↻ 127.0.0.1:8000/api/cabinet/

Django REST frameworkКирильцев Роман Сергеевич

Cabinet Api

Cabinet ApiOPTIONSGET

GET /api/cabinet/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "number": 31,
    "owner": "Роман",
    "phone": "+79119706266"
  }
]
```

Интерфейс создания записи:

Meeting Create Api - Django RE

127.0.0.1:8000/api/meeting/create/

aa

Django REST framework

Кирильцев Роман Сергеевич

Meeting List Api / Meeting Create Api

Meeting Create Api

OPTIONS

GET /api/meeting/create/

HTTP 405 Method Not Allowed

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

{

"detail": "Method \"GET\" not allowed."

}

Raw data

HTML form

Дата приема

ДД. ММ. ГГГГ

Время приема

--:--

Текущее состояние

Оплачено

Пациент

Иванов Иван

Иванов Иван

Врач

Кирильцев Роман Сергеевич

Услуга

POST

Фрагмент из models.py

```
manage.py x urfs.py x models.py
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3
4 class Doctor(AbstractUser):...
5
6 class Cabinet(models.Model):...
7
8 class Schedule(models.Model):...
9
10 class Price(models.Model):...
11
12 class Patient(models.Model):...
13
14 class Diagnosis(models.Model):...
15
16 class Medcard(models.Model):
17     id_patient = models.ForeignKey('Patient', verbose_name='Пациент', on_delete=models.CASCADE)
18     id_diagnosis = models.ForeignKey('Diagnosis', verbose_name='Диагноз', on_delete=models.CASCADE)
19     start_date = models.DateField(verbose_name='Дата установки диагноза')
20     status = models.CharField(max_length=60, verbose_name='Статус заболевания')
21
22 class Meeting(models.Model):
23     id = models.AutoField(unique=True, primary_key=True)
24     id_patient = models.ForeignKey('Patient', verbose_name='Пациент', on_delete=models.CASCADE)
25     id_doctor = models.ForeignKey('Doctor', verbose_name='Врач', on_delete=models.CASCADE)
26     id_price = models.ForeignKey('Price', verbose_name='Услуга', on_delete=models.CASCADE)
27     date_meet = models.DateField(verbose_name='Дата приёма')
28     time_meet = models.TimeField(verbose_name='Время приёма')
29     status = models.CharField(max_length=50, verbose_name='Текущее состояние')
30     price_status = models.BooleanField(default=False, verbose_name='Оплачено')
```