Ти си ученик от 12 клас, специалност "Приложно програмиране". Правиш дипломен проект, който е теоретико-практическа разработка по зададена тема. Темата на проекта е зададена с дипломно задание по следния начин:

Тема: Принципи на REST API и създаване на собствено REST API

Входни данни: Дипломният проект е теоретико-практическа разработка на темата, възложена с настоящото индивидуално задание на основание заповеди №-08-139/08.01.2025 г. и №-08-143/08.01.2025 г. на директора на ПГЕЕ, гр. Банско. В съответствие с чл. 1 ал. 5 от Наредба №1 от 19.02.2020г. за организацията и провеждането на изпитите за придобиване на професионална квалификация, разработката трябва да се състои от две части - теоретична част и практическа част.

В теоретична част да се разгледат основите на REST архитектурата – сфера на приложение , методи (GET, POST, PUT, DELETE) и добри практики при реализация на приложения.

Като практическа част да се разработи REST API за приложение за управление на резервации – за хотели, коуъркинг пространства, събития и др.

Практическата част е разработена. Използвани са Python, Django и DRF. Реализирано е малко приложение за записване на обучителни курсове за деца. Два изгледа, Два модела, съответните сериализатори и маршрутизиране - това е включено в практическата разработка. Добавен е и минимален примерен потребителски интерфейс, който илюстрира използването на разработеното API. Трябва да напишем писмената част на проекта, която да е с обем около 30 стр. и поне половината да е разглеждане на въпросите от теоретичната част на заданието, а останалото - описание на практическата разработка. Състави план (съдържание) на писмената част, като добавиш и примерна разбивка за обема на всеки раздел (точка) от плана. Не включвай в плана раздел Използвана литература.

1. **Увод**
   * Обем: 1 страница
2. **Теоретична част**  
   2.1. **Същност на REST архитектурата**  
   - Какво е REST (Representational State Transfer)  
   - История и възникване на REST  
   - Основни принципи на REST  
   - Обем: 2 страници

2.2. **Сфера на приложение на REST**  
- Къде и защо се използва REST  
- REST vs. други архитектурни стилове (SOAP, GraphQL)  
- Примери за реални приложения  
- Обем: 2 страници

2.3. **Основни HTTP методи в REST API**  
- Методи: GET, POST, PUT, DELETE  
- Примери за използването им  
- Други методи (PATCH, OPTIONS)  
- Обем: 3 страници

2.4. **Дизайн на REST API**  
- Добри практики при проектиране на REST API  
- Именуване на ресурси и маршрути  
- Използване на статус кодове  
- Документиране на API  
- Обем: 3 страници

2.5. **Сигурност в REST API**  
- Основни подходи (автентикация, авторизация, HTTPS)  
- Чести уязвимости и защити  
- Обем: 1,5 страници

2.6. **Тестове на REST API**  
- Тестови подходи  
- Популярни инструменти (Postman, Swagger и др.)  
- Обем: 1,5 страници

**Общ обем на теоретична част: 13 стр.**

1. **Практическа част**  
   3.1. **Описание на разработваното приложение**  
   - Цел и функционалност на приложението  
   - Обосновка на избора на технологии  
   - Обем: 1 страница

3.2. **Използвани технологии и инструменти**  
- Python  
- Django  
- Django Rest Framework  
- Кратко описание на всеки инструмент  
- Обем: 2 страници

3.3. **Архитектура на приложението**  
- Модели (описание и схема)  
- Взаимодействие между компонентите  
- Обем: 2 страници

3.4. **Реализация на REST API**  
- Дефиниране на сериализатори  
- Двата изгледа (описание, примери)  
- Маршрутизиране на заявки  
- Обем: 3 страници

3.5. **Потребителски интерфейс**  
- Как е реализиран  
- Примери/screenshots  
- Обем: 1 страница

3.6. **Примери за работа с приложението**  
- Примерни заявки и отговори  
- Работа с Django REST Framework вградената поддръжка за автоматично генериране на документация и тестване през уеб интерфейс  
- Обем: 1 страница

3.7. **Тестове и резултати**  
- Проведени тестове (  
- Констатации  
- Обем: 1 страница

**Общ обем на практическата част: 11,5 стр.**

1. **Заключение**
   * Изводи от проведената работа
   * Препоръки за бъдещо развитие

Обем: 1,5 страници

Супер. Да започнем с генерирането на текста на дипломния проект. Състави текст по с следния план:

2. Теоретична част

2.1. Същност на REST архитектурата

- Какво е REST (Representational State Transfer)

- История и възникване на REST

- Основни принципи на REST

- Обем: 2 страници

2.2. Сфера на приложение на REST

- Къде и защо се използва REST

- REST vs. други архитектурни стилове (SOAP, GraphQL)

- Примери за реални приложения

- Обем: 2 страници

2.3. Основни HTTP методи в REST API

- Методи: GET, POST, PUT, DELETE

- Примери за използването им

- Други методи (PATCH, OPTIONS)

- Обем: 3 страници

2.4. Дизайн на REST API

- Добри практики при проектиране на REST API

- Именуване на ресурси и маршрути

- Използване на статус кодове

- Документиране на API

- Обем: 3 страници

2.5. Сигурност в REST API

- Основни подходи (автентикация, авторизация, HTTPS)

- Чести уязвимости и защити

- Обем: 1,5 страници

2.6. Тестове на REST API

- Тестови подходи

- Популярни инструменти (Postman, Swagger и др.)

- Обем: 1,5 страници

Общ обем на теоретична част: 13 стр.

3. Практическа част

3.1. Описание на разработваното приложение

- Цел и функционалност на приложението

- Обосновка на избора на технологии

- Обем: 1 страница

3.2. Използвани технологии и инструменти

- Python

- Django

- Django Rest Framework

- Кратко описание на всеки инструмент

- Обем: 2 страници

3.3. Архитектура на приложението

- Модели (описание и схема)

- Взаимодействие между компонентите

- Обем: 2 страници

3.4. Реализация на REST API

- Дефиниране на сериализатори

- Двата изгледа (описание, примери)

- Маршрутизиране на заявки

- Обем: 3 страници

3.5. Потребителски интерфейс

- Как е реализиран

- Примери/screenshots

- Обем: 1 страница

3.6. Примери за работа с приложението

- Примерни заявки и отговори

- Работа с Postman/или друг инструмент

- Обем: 1,5 страници

3.7. Тестове и резултати

- Проведени тестове

- Констатации

- Обем: 1 страница

имай предвид и обема по точки. Пиши като 2. Теоретична част

2.1. Същност на REST архитектурата

- Какво е REST (Representational State Transfer)

- История и възникване на REST

- Основни принципи на REST

- Обем: 2 страници

2.2. Сфера на приложение на REST

- Къде и защо се използва REST

- REST vs. други архитектурни стилове (SOAP, GraphQL)

- Примери за реални приложения

- Обем: 2 страници

2.3. Основни HTTP методи в REST API

- Методи: GET, POST, PUT, DELETE

- Примери за използването им

- Други методи (PATCH, OPTIONS)

- Обем: 3 страници

2.4. Дизайн на REST API

- Добри практики при проектиране на REST API

- Именуване на ресурси и маршрути

- Използване на статус кодове

- Документиране на API

- Обем: 3 страници

2.5. Сигурност в REST API

- Основни подходи (автентикация, авторизация, HTTPS)

- Чести уязвимости и защити

- Обем: 1,5 страници

2.6. Тестове на REST API

- Тестови подходи

- Популярни инструменти (Postman, Swagger и др.)

- Обем: 1,5 страници

Общ обем на теоретична част: 13 стр.

3. Практическа част

3.1. Описание на разработваното приложение

- Цел и функционалност на приложението

- Обосновка на избора на технологии

- Обем: 1 страница

3.2. Използвани технологии и инструменти

- Python

- Django

- Django Rest Framework

- Кратко описание на всеки инструмент

- Обем: 2 страници

3.3. Архитектура на приложението

- Модели (описание и схема)

- Взаимодействие между компонентите

- Обем: 2 страници

3.4. Реализация на REST API

- Дефиниране на сериализатори

- Двата изгледа (описание, примери)

- Маршрутизиране на заявки

- Обем: 3 страници

3.5. Потребителски интерфейс

- Как е реализиран

- Примери/screenshots

- Обем: 1 страница

3.6. Примери за работа с приложението

- Примерни заявки и отговори

- Работа с Postman/или друг инструмент

- Обем: 1,5 страници

3.7. Тестове и резултати

- Проведени тестове

- Констатации

- Обем: 1 страница

Пиши като ученик от 12 клас, специалност "Приложно програмиране". Отчитай обема на всяка точка. Не използвай твърде много подточки и тирета, нека е в по-разказна форма. добави кратки цитати код, където е уместно.

1. **Тема и цел на проекта**  
   Проектът представя изграждането на REST API за управление на обучителни курсове за деца, използвайки Python, Django и Django Rest Framework. Основната цел е да се демонстрират на практика принципите и добрите практики при създаване на уеб услуги.
2. **REST архитектура и приложение**  
   Обяснена е същността на REST като лек и мащабируем архитектурен стил за уеб услуги, който използва стандартни HTTP методи. REST е избран заради своята простота, съвместимост и широко приложение в съвременните уеб и мобилни системи.
3. **Основни HTTP методи**  
   В проекта се използват основните HTTP методи – GET, POST, PUT, DELETE и PATCH, които позволяват стандартни операции върху ресурсите. Всеки метод има ясно предназначение и улеснява комуникацията между клиент и сървър.
4. **Дизайн и сигурност на API**  
   API-то е проектирано с ясни и логични маршрути, използва подходящи статус кодове и добра документация. Включени са мерки за сигурност като валидация на данните, CORS настройки и препоръки за използване на HTTPS и токен-базирана автентикация.
5. **Модели и база данни**  
   Създадени са два основни модела – Course (Курс) и Enrollment (Записване), които отразяват реалните обекти и връзки в системата. Структурата на базата данни е ясна и лесно разширяема.
6. **Реализация на REST API**  
   Използвани са сериализатори, ViewSet-и и автоматична маршрутизация чрез DRF Router, което осигурява бърза и стандартизирана разработка на API-то. Всички CRUD операции са реализирани с минимален код.
7. **Потребителски интерфейс**  
   Към проекта е добавен отделен потребителски интерфейс с Vue.js и Axios, който комуникира с REST API-то. Това позволява реално тестване на системата и демонстрира интеграцията между frontend и backend.
8. **Тестване и резултати**  
   Проведени са тестове както през вградения DRF интерфейс, така и през външния фронтенд, които потвърждават коректната работа на всички основни функционалности. Валидирането на данните и CORS настройките са успешно реализирани.
9. **Възможности за разширяване**  
   Проектът може да бъде надграден с автентикация, по-сложни модели, известия, статистики и други функционалности. Архитектурата позволява лесно добавяне на нови възможности и интеграция с външни системи.
10. **Образователна стойност**  
    Проектът служи като учебен модел за ученици и начинаещи програмисти, показвайки добрите практики при изграждане на REST API и интеграционни решения. Кодът е структуриран и подходящ за б