# Тестовое задание на Python Backend разработчика

**Задача:**

Реализовать сервис, который принимает и отвечает на HTTP запросы.

**Функционал:**

1. В случае успешной обработки сервис должен отвечать статусом 200, в случае любой ошибки — статус 400.

2. Сохранение всех объектов в базе данных.

3. Запросы

- GET /city/ — получение всех городов из базы;
- GET /city/city_id/ — получение города по id;
- GET /city/city_id/street/ — получение всех улиц города; (city_id — идентификатор города)
- POST /shop/ — создание магазина; Данный метод получает json с объектом магазина, в ответ возвращает id созданной записи.
- GET /shop/?street=&city=&open=0/1 — получение списка магазинов.

    I. Метод принимает параметры для фильтрации. Параметры не обязательны. В случае отсутствия параметров выводится все магазины, если хоть один параметр есть, то по нему выполнятся фильтрации.

    II. Важно! В объекте каждого магазина выводится название города и улицы, а не id записей.

    III. Параметр open: 0 - закрыт, 1 - открыт. Данный статус определяет исход из параметров «Врем открытия», «Врем закрытия» и текущего времени сервера.

**Объекты**:

Магазин:

- Название
- Город
- Улица
- Дом
- Время открытия
- Время закрытия

Город:

- Название

Улица:

- Название
- Город

!! Замечание: поле id у объектов не указаны, но подразумевается, что они есть.

!! Важно: выстроить связи между таблицами в базе данных.

**Инструменты:**

- Фреймворк для обработки http запросов Django + Django Rest Framework или FastAPI
- Реляционная БД (PostgreSQL - предпочтительно, MySQL и тд)
- Запросы в базу данных через ORM (ORM на выбор).
- Использование Docker, сервис должен запускаться с помощью docker-compose up.

**Сдача задания:**

Ссылка на репозиторий, который содержит ваш проект и README

- Фамилия и имя
- Тестовое задание Python
- Описание проект
- Подготовительные действия (установки, настройки и т.д) для успешной работы проекта
- Информация о доступах (логины/пароли и т.д.)
- Описание, как запустить ваш проект.