Ход выполнения дипломной работы

31.01.2024

1. Не читая задания на диплом, внимательно смотрим видеоурок «Консультация по дипломной работе профессии «Python-разработчик», проведенный Азаматом Искаковым, одним из руководителем дипломного прооекта на скорости 1.5х.

Конспект встречи

Мотивационные заметки:

Нет нерешаемых проблем

Дробите задания на более мелкие

Хвалите себя

Слушайте видео и подкасты с экспертами

Гуглите на английском

Дипломный проект

Автоматизация закупок

Создать Django проект

Проработать модели данных

Создать API для покупателей и поставщиков

1000 строк кода в итоге

Модели данных – все, что касается поставщиков, покупателей и связями между ними. Вспомнить нужно релятивистские (реляционные) базы данных. Связи внешние, многие ко многим, один ко многим

Создать API для покупателей и поставщиков.

Касательно поставщиков необходимо реализовать импорт товаров, чтобы предоставить возможность вашим поставщикам предоставлять их список товаров, присущих их магазину в некотором структурированном виде (yml, json). В файле – массив информации. Ваша система его принимает, после чего появляется этот список товаров уже в вашей системе.

Для покупателей должна быть предусмотрена возможность просматривать товары, магазины и оформлять заказы на товары, которые покупателю понравились.

Готовый бэкэнд займет около 1000 строк кода . Это вполне нормально.

Серверный рендеринг для Python – это не очень хорошо. Python – среда относительно медленная. При клиентском рендеринге мы отправляем серверу тот объем информации, который необходим в виде json файла.

Отправляя коллеге xls файл, мы не отправляем им саму программу excel.

Авторизация, регистрация и логин – оставим на второй план в рамках нашего проекта.

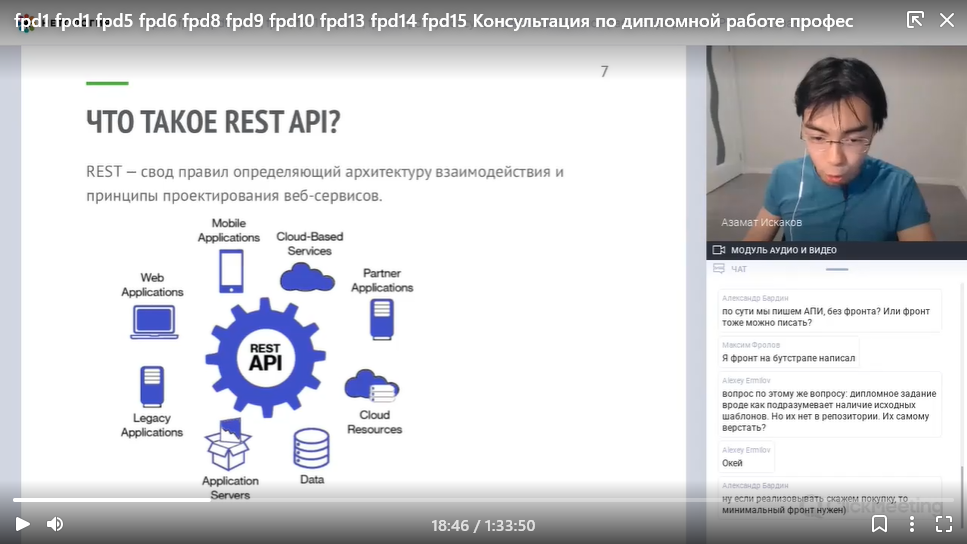
Есть список товаров на сайте, которые показываются клиенту без регистрации и авторизации на сайте. При этом просто viewha отправляет queryset. Т.е. у вас есть некая информация в базе данных. Вы создали т.н. queryset запрос в базу данных. Этот queryset можно представить структурированно в виде json файла. Против каждого поля запроса (допустим, name, quantity, price и т.д.) есть содержание этого поля в виде величины стоимости, описания товара и т.д. Т.е вы отправляете структурированный json файл клиенту, а на стороне клиента это может быть мобильное приложение, веб-фронтэнд, view.js, java script framework, сторонний API сервис от любого сервера, которому нужна информация, она ему нужна именно в таком виде. Именно в таком виде (json) вы ее и выполняете. А как ее представить уже – клиент отвечает сам, т.е. это может быть просто у него мобильное приложение, оно уже само знает как эту информацию представить ей. Это может быть веб-контент, там может быть всё красиво java scriptom описано, но вас это уже не волнует.

По сути мы пишем API без фронта. Бутстрап – это красивая среда, в которой будут хорошо работать формы, кнопки и т.д.

Когда мы используем Django forms, мы ограничиваем себя взаимодействием сервер – веб-браузер, а при использования API-сервиса, мы расширяем этот список взаимодействий.

По сути нам не нужно создавать пользовательский интерфейс для приложения. А нужно создать бэкэнд, который отдает массивами в виде json файла информацию в сторону клиента или в сторону поставщика. На их стороне можно уже построить отдельное фронтэнд приложение как вам удобно будет. Это может быть тот же бутстрап, любой java script framework, который вы сможет освоить или просто отдать фронтэнд программисту, чтобы он для вас это выстроил. Все зависит от того, как вы себя позиционируете. Если вы хотите быть фуллстек разработчиками, не только специализироваться на бэкэнде, естественно вы можете построить фронтэнд приложение на вебе. Если вы заинтересованы в построении хорошего бэкэнд-сервиса и при этом отлично задокументировать его для фронтэнд-разработчика, чтобы он прекрасно смог разобраться, то инструкция к дипломному проекту вам хорошо в этом поможет.

REST – свод правил, определяющий архитектуру взаимодействия и принципы проектирования веб-сервисов.



Протестировать покупку можно при помощи POSTMAN.

“Энтузиазм – наше всё, остальное можно погуглить”

REST-сервисы – это про некую универсальность. Т.е. говоря о том, что вы создаете некий единый веб-сервис, который будет прекрасно работать как с мобильными приложениями, так и с фронтэнд аппликейшнами, так и с какими-то базами данных и другими сторонними облачными или API сервисами. REST – это то, что вы создаете некий единый сервис, работающий по какой-то определенной архитектуре и посредством его можно взаимодействовать с любыми точками в сети интернет. API может быть любой версии. Это может легко отслеживаться и создаваться.

Для каждого типа операции используется свой метод HTTP-запроса:

Получение – GET

Добавление – POST

Модификация – PUT

Удаление – DELETE

GET /rest/users – получение инфо о всех пользователях

GET /rest/users/125 – получение инфо о пользователе с id=125

POST /rest/users – добавление нового пользователя

PUT /rest/users/125 – изменение инфо о пользователе с id=125

DELETE rest/users/125 – удаление пользователя с id=125

Каждый отдельный запрос нужен для определенной операции. GET запрос используется для получения списка товаров или магазинов. POST и PUT запросы используются для добавления и модификации. DELETE запрос используется для удаления информации.

Ответ от сервера зачастую приходит в виде json или xml файла. Для запросов GET длина, в которую вы можете вместить запрос ограничена 256 символами.

Xml файл менее удобен для человеческого глаза. Json для этого прекрасно подходит. Большинство сервисов в современном вебе зачастую используют json в виде ответов от сервера.

В Django большая часть задач уже решена. Django – простой по освоению фрэймворк для того, чтобы создавать API сервисы. Пользоваться старыми программами не рекомендуется. Инструменты Django постоянно поддерживаются и обновляются.

Практика

Линукс рекомендуется для применения при разработке сервисов. Лучше программировать и разворачивать сервис на линуксе. Pycharm автоматизирует многие рутинные процедуры.

Смотрим практику, повторяем за Азаматом и записываем наиболее важные моменты.

Первоначально необходимо проверить работает ли проект в браузере.

Виртуальная машина vmware workstation – для развертки линукса.

Джанго прекрасно работает.

Модель shop говорит о представлении полей внутри магазина (название, статус принимает ли он заказы)

Модель product (имя, категория, количество, цена)

Установить djangorestframework с помощью файла зависимостей requirements.txt.

УКАЗАТЬ В РАЗДЕЛЕ INSTALLED\_APPS В ФАЙЛЕ SETTINGS.PY!!!

**Файл serializers.py – сущность, которая будет «кушать» ваши модели, информацию из базы данных, и представляет ее в виде структурированного словаря, по сути, json файла.**

Вьюшки скармливают querysets serializeram, чтобы соответственно забирать у них результат.

Если удаляем магазин и удаляются все его товары, то это CASCADE.

Для начала мигрируем нашу модель в базу данных командой python manage.py makemigrations

Далее python manage.py migrate

У сериалайзера будет множество объектов (many=True)

При очередном GET запросе создается новый телефон.

Для корректного указания shop в json файле, в классе Product создаем объект shop=ShopSerializer() и тогда его id, name и state будут отображаться в результате запроса (json).

queryset = Product.objects.all().select\_related(‘shop’) – экономит запрос в базу данных.

Если нужна валидация данных. Это если данные приходят извне.

POSTMAN нужен для того, чтобы протестировать сервис и разговаривать с фронтэнд разработчиком на одном языке.

В settings.py указываем ALLOWED\_HOSTS = [‘\*’], чтобы мы могли отправлять запросы к сервису с любого компьютера.

serializer.is\_valid() – валидация

Валидация идет в сериалайзере.

POSTMAN – отличный способ тестирования API сервиса.