Курс: веб-разработка на python и django

Авторы: Михаил Аграновский, Владимир Курилин

Для zavtra.online by Skillfactory

Цель курса:

Дать ключевые навыки по настройке бэкэнд-составляющей ML-продукта с использованием django orm

Результат курса:

Отвечает на вопрос: Что студент сможет сделать, применяя знания, полученные на курсе?

Пройдя курс, студент сможет написать бекенд веб-сервиса на Django + Django Rest Framework, и обратиться из него к любому микросервису, в частности – к ML-модели.

Приобретаемые навыки:

Что студент будет знать, уметь?

- понимание принципов работы WEB
- понимание ролей backend-разработчика и ml-разработчика в web-разработке
- умение выбирать подходящий фреймворк для написания API под конкретную задачу
- навык разработки API на Django + Django Rest Framework. Разработанный проект.
- понимание баз данных, очередей задач
- понимание концепта микросервисной архитектуры. Разработанный проект из связки Django API с микросервисом, выполняющем рассчет с использованием ML
- навык тестирования и отладки кода
- умение работать с инструментами разработки PyCharm, Docker, Postman
- Сможет дополнить резюме строчкой “WEB, Python/Django/Celery/PyTest, Docker”

Контроль достижения результатов (задание в конце модуля):

Приложение на Django + DjangoRestFramework, реализующее REST-API для решения прикладной задачи на выбор студента. Приложение должно взаимодействовать с внешним сервисом. Выбор предметной области остается за студентом.

Пример: приложение на Django реализует фото-альбом с поиском по изображенным предметам. Для классификации предметов на фотографиях приложение отправляет изображения в сервис-предсказатель и получает от него список объектов.

Структура курса:

Модуль 1: Современный WEB

Результат: Студент познакомился с принципами работы web и местом бекенда в структуре web, локализовал место ml-разработки в веб-сервисах. Познакомился с видами api, научился вручную взаимодействовать с сторонними API. Познакомился с классами вычислительных задач и видами web-фреймвоков на Python, научился выбирать web-фреймворк для бекенда под конкретную задачу.

Навыки: WEB, REST API, curl/Postman.

План урока:

Юнит 1. WEB

1. Типовая задача на собесе на проверку на грамотность: Путь одного запроса: http, tsl (https), dns, tcp. Http методы и заголовки. Уровни DNS кеша.
2. Сохранение состояние между запросами. Cookie.
3. HTTP REST и альтернативы (graphQL, web-socket, gRPC)
4. Место бекенда в веб разработке. Генерация html на беке и современный мир. Толстый и тонкий клиент.
5. Что еще может лежать между запросом из клиентского js до бекенда. Backend-for-frontend. Кеши.

Юнит 2. Backend on Python

6. CPU/io-bound tasks
7. выбор фреймворка. Выбор между скоростью работы сервиса и скоростью его разработки/поддержки. Выбор между затратами на горизонтальное масштабирование серверов и затратами на з/п команде разработке за время на более сложные решения.
8. Ускоряем обработку запроса: не срочное ставим на асинхронное выполнение. Очереди задач и Celery.

Задание:
Небольшие практические задания на отработку знаний из теоретических разделов. Студенты выполнят вручную запросы в некоторому REST API, определяет ip-адрес по доменному имени сервиса.

Модуль 2. Основы Django

Результат: Поймём, почему Django популярен, и в какие моменты его удобно использовать. Познакомимся со структурой типового Django-приложения, основными концепциями и научимся подключать дополнительные модули.

Навыки: Django, Django Admin, Django ORM + migrations.

План урока:

Юнит 1. Знакомимся с Django

1. Немного слов о том, почему Django крут, и в каких случаях его стоит использовать.
2. Где искать знания. Django tutorial + Django Docs – да, Stackoverflow – не всегда.
3. Галопом по туториалу:
    a. Смотрим на простое Django приложение: структура, назначение каждого файла
    b. Разбираемся с INSTALLED_APPS + routing
    c. Модели – магия, позволяющая из питона читать и писать в базу
    d. Django Admin. (Вжух, и можно добавлять контент)
    e. Django View

Юнит 2. Django ORM

1. manage.py shell
2. ORM + queries + миграции
3. Редактируем Django проект на лету
    a. Django Forms
    b. Django можно (и нужно) тестировать
4. Модули. Берем и добавляем. (Debug panel/Тема на админку/…)

Задание:
Практические задание на рефакторинг, добавление фичи и исправление ошибок в готовом коде.

Модуль 3. Продолжение изучения Django

Результат: Научимся делать API-шки с помощью DRF, познакомимся с паттерном Mixin, защитим API от несанкционированного доступа. Узнаем как не надо варить очереди, и как надо, реализуем систему с асинхронным взаимодействием.

Навыки: Django Rest Framework, Mixins, REST API, Celery, AMQP.

План урока:

Юнит 1. Связываемся с внешним миром: Django Rest Framework

4. Зачем нужен DRF.
5. Где искать знания. Закрепляем паттерн по чтению документации и туториала.
6. Основные сущности:
    a. Serializer, делаем наш первый API
    b. Запросы и ответы, content-type
    c. Class-based views + Mixins + Generic views
7. Пара слов о безопасности
8. Авторизация

Юнит 2. Очереди задач, Celery

5. Ещё раз про очереди, AMQP, семантики доставки сообщений
6. Знакомимся с Celery
7. Антипаттерны в очередях
8. Проектируем систему с асинхронными взаимодействиями
9. Реализуем наши планы на DRF + Celery

Задание:
Продолжаем развивать кодовую базу проекта, практическое задание на проектирование API и Celery Workers.

Модуль 4. Базы данных

Результат: Познакомимся с основными подходами к хранению данных, фундаментальными проблемами СУБД.
Познакомимся с практическими аспектами работы PostgreSQL: оптимизация запросов и чтение плана, работа с множеством соединений через pgBouncer.
Узнаем, какие подводные камни возникают при использовании ORM.

Навыки: SQL, noSQL, оптимизация запросов, psql explain/analyse, Индексы, ORM.

План урока:

Юнит 1. Немного теории

1. SQL vs noSQL
2. ACID, транзакции, конкуренция, блокировки, мультиверсионность, WAL. Select_for_update.
3. Нормализация, constraints
4. Распределённые хранилища

Юнит 2. Databases in depth

5. Индексы, устройство и виды.
6. Репликация
7. OLAP vs OLTP, Аналитические базы.
8. PgBouncer.
9. Это не страшно, GUI есть!
10. Explain/Explain analyse.
11. Плюсы и минусы ORM. Оптимизация Django ORM (select/prefetch_related)

Задание:
Побенчмаркать и оптимизировать запрос в PostgreSQL .

Модуль 5. Инфраструктура разработки

Результат: Студент познакомился сервисами и процессами, которые облегчают разработку приложения, снижают риск возникновения ошибок или позволяют выявить часть багов до релиза приложения в продакшен.

Навыки: PyCharm, flake8, black, pytest, poetry, pyenv, docker, docker-compose, zsh

План урока:

1. Расширенные возможности среды разработки PyCharm. Шорткаты, подсказки, рефакторинги, графический интерфейс Git, диаграммы зависимости моделей Django. Получение удовольствия от процесса разработки.
2. Линтеры и форматеры кода: flake8, black. Поддержание кода в чистоте, упрощение code review.
3. Тестирование
    a. Почему тесты не тратят время, а экономят
    b. Виды тестов: интеграционные, юниты и функциональные.
    c. pytest – современный фреймворк для тестирования python
    d. Структура теста: паттерн Arrange, Act, Assert
    e. Принцип DRY: pytest.mark.parametrize
    f. Подготовка данных для теста: pytest fixtures, фабрики FactoryBoy
    g. Тест зависит только от того, что он хочет протестировать: patch
    h. Сравниваем тест курильщика и тест-сьют здорового человека
4. Сборка и запуск проекта: poetry, pyenv, docker, docker-compose
5. Консоль класса “комфорт плюс”
    a. Удобный эмулятор терминала: iterm (OS X) / Tilix (Linux) / ConEmu (Windows)
    b. Удобный шел: zsh

Задание:

1. Запустить flake8 на коде проекта. Исправить не менее 5 замечаний.
2. Запустить black на коде проект. Посмотреть diff и понять причины изменений.
3. Написать интеграционные и юнит тесты на функционал проекта
4. Собрать и запустить проект в docker

Модуль 6. Инфраструктура продакшена

Результат: Студент познакомился с особенностями эксплуатации непрерывно доступного сервиса и способами ее облегчить. Доработал свой проект для упрощения его потенциальной эксплуатации в продакшене.

Навыки: docker, docker-compose, kubernetes, aws lambda.

План урока:

Часть 1. Инфраструктура рантайма

1. Виды облачных инфраструктур: IaaS, PaaS, SaaS. Написание лямбды в Amazon Web Services. Kubernets
2. Деплой: docker registry, релиз-теги в git, бесшовные релизы, библиотека Django zero-downtime-migrations и решаемые в ней проблемы
3. Горизонтальное и вертикальное масштабирование. Балансировка нагрузки, nginx.
4. Graceful degradation. Расселение не связанных элементов бизнес-логики на разные инстансы. Отдельные инстансы для Celery и для администрирвоания (админка, шел, запуск миграций и менеджмент-команд Django)
5. Микросервисы и межсервисное взаимодействие (гормошка обращений по rest, cricket breaker, очереди)
6. Недоступность как явление неизбежна, важно не выходить за пределы SLA

Часть 2. Логи

1. Что писать в логи и что не писать
2. Модуль logging и как писать логи
3. Логирование на продакшене: Kibana

Часть 3. Исключения

1. Жизнь без исключительных ситуаций: обрабатываем все, что можем предвидеть; остаемся готовыми к непредвиденному. Иерархия классов исключений
2. Исключение как способ сформировать отличный от 200 http ответ
3. Логирование ошибок
4. Ошибки на продакшене: Sentry

Часть 4. Отладка

1. Отладка Python: pdb, ipdb, графический дебагер PyCharm
2. Пример починки бага на продакшене: от жалобы пользователя до локализации в коде

Задание:

1. Зарегистрироваться в AWS и написать пару лямбд
2. Провалидировать имеющиеся миграции на безопасность. Написать безопасное добавление нового поля в модель.
3. Проект

a. Написать кастомную ошибку валидации. Подключиться к проекту дебагером и поймать ее.
b. Добавить / отрефакторить логирование
c. Добавить в проект микросервис и взаимодействие с ним. Упрощенная версия: добавить взаимодействие с внешним сервисом