Бауманский учебный центр «Специалист»

Программа курса Асинхронное программирование на основе веб - фреймворка FastAPI

Код курса: **ФАСТАПИ**

Тема	Ак. часов
Модуль 1. Основы асинхронного программирования Мы познакомимся с основные концепциями и принципами асинхронного программирования. Изучим примеры асинхронного программирования с использованием библиотеки asyncio. Разберем что такое GIL. • Введение в асинхронность. • Основные преимущества asyncio. • В каких задачах полезно асинхронное программирование. • В каких задачах бесполезно асинхронное программирование. • GIL. • IO bound и CPU bound.	**5**
Модуль 2. Awaitable объекты Мы познакомимся с корутинами, разберем ключевые слова async и await. Изучим, что такое цикл событий. • Coroutines / Корутины. • Awaitable объекты. • Ключевое слово await. • Цикл событий (event loop). • Task / Задачи.	**5**
Модуль 3. Введение в API Мы познакомимся с API, архитектурой REST API. Изучим основные HTTP методы и коды ответа сервера. • Что такое API и его типы? • Изучаем REST. Архитектура и лучшие практики. • HTTP методы и идемпотентность. • Коды состояний HTTP.	**4**
Модуль 4. Знакомство с FastAPI Мы познакомимся с основами FastAPI и создадим свое первое API, возвращающее "Hello World". Также мы подробно изучим процесс создания конечных точек с использованием системы маршрутизации в FastAPI. И научимся валидировать параметры, получаемые из пути и тела запросов. • Что такое FastAPI? • HelloWorld на FastAPI. • Знакомство с документацией API. • Маршрутизация в FastAPI. • Валидация параметров маршрутов (Path, Query).	**6**

Модуль 5. Создание CRUD-приложения на FastAPI Мы спроектируем и реализуем простой CRUD проект. Познакомимся с основными методами запросов GET, POST, PUT, DELETE, научимся создавать и валидировать модели класса Pydantic. Изучим моделирование ответов, обработку ошибок и коды состояния. Обсудим, как можно использовать шаблоны Jinja для создания представлений и вывода ответов из API. • Проектирование и реализация REST API (GET, POST, PUT, DELETE). • Модели Pydantic и их валидация. • Модели ответов и обработка ошибок. • Использование шаблонов Jinja и форм в FastAPI.	**4**
Модуль 6. Внедрения зависимостей Мы рассмотрим шаблон внедрения зависимостей, используемый FastAPI для управления экземплярами и структурой проекта с помощью директивы Depends() и сторонних модулей расширения. • Внедрение функций и классов зависимости. • Зависимости в операциях пути и глобальные зависимости.	**4**
Модуль 7. Готовое веб приложение (блог / task scheduler) Мы рассмотрим основные возможности FastAPI, научимся правильно структурировать проект. Узнаем об использовании APIRouter для организации проектов. Создадим подключение к БД через SQLAlchemy, научимся использовать Alembic для выполнения миграций. Научимся использовать сессии и напишем свои первые запросы, которые в дальнейшем мы сделаем асинхронными. Реализуем аутентификацию используя HTTP Basic Auth и научимся генерировать JWT токен. В дальнейшем будем использовать его, для проверки прав доступа к различным конечным точкам нашего API. • Структурирование FastAPI, APIRouter и написание моделей Pydantic. • Подключение к базе данных используя SQLAlchemy. • Введение в модели SQLAlchemy. • Отношения таблиц в SQLAchemy. • Миграции в SQLAlchemy, знакомство с Alembic. • Написание запросов к БД используя SQLAlchemy, часть 1. • Написание запросов к БД используя SQLAlchemy, часть 2. • Добавление асинхронности, переход на PostgreSQL. • Добавление асинхронности в запросах SQLAlchemy. • Реализация аутентификации используя HTTP Basic Auth. • Реализация аутентификации с помощью JWT. • Управление правами доступа в FastAPI.	**8**
Модуль 8. Docker + FastAPI и деплой проекта Мы рассмотрим основные концепции Docker и на примере создадим свой контейнер с Python проектом. Далее мы упакуем FastAPI, PostgreSQL/MySQL, NGINX/Traefik в контейнеры и подготовим проект к деплою. Выполним деплой с помощью PaaS сервиса. • Знакомство с Docker. • Основные команды Docker. • Создание образов и запуск контейнера. • Контейнеризация Django и написание Dockerfile. • Введение в Docker Compose. • Docker Compose на примере FastAPI, PostgreSQL. • Подготовка к деплою, Gunicorn и NGINX в Docker Compose. • Настройка Docker-Compose для статических и медиа файлов. • Деплой проекта.	**4**
Аудиторная нагрузка в классе с преподавателем	**40 ак.ч. + 40 ак.ч. самостоятельно**

По окончании обучения на курсе проводится итоговая аттестация. Аттестация проводится в виде теста на

последнем занятии или на основании оценок практических работ, выполняемых во время обучения на курсе.

Контакты центра

Для частных лиц
+7 (495) 232-32-16

Для организаций
+7 (495) 780-48-44

E-mail
info@specialist.ru