

Микросервисы на Python

Постройте систему, которая сохраняет понятное поведение даже при сбоях соседних сервисов.

120

академических часов

40 + 80

занятия + самоподготовка

5 + 1

проектов модулей + итог

Зачем проходить этот курс

- На практике сравните синхронные вызовы, очереди задач и поток событий.
- Отработаете сбои: таймауты, повторную доставку и недоступность компонентов.
- Подготовите демонстрируемый стенд с архитектурой, тестами и наблюдаемостью.

Кому подойдёт

Python backend-разработчикам, которые хотят перейти от одного API к взаимодействию нескольких сервисов.

Что нужно на старте

Нужны уверенный Python, ООП, HTTP/REST, базовый SQL и тесты. Желательно знакомство с Docker и одним web-фреймворком.

Онлайн • Живые занятия • Группа до 8 человек

Понятный путь к результату

40 ак. ч. с преподавателем + 80 ак. ч. самостоятельной работы. Один академический час - 45 минут. Практика и проекты входят в указанный объём.

Модуль	Занятия	Самост.
01. Границы сервисов и рабочий стенд	4	8
02. HTTP API и хранение данных	8	16
03. Интеграции и устойчивость	6	12
04. События и брокеры сообщений	10	20
05. Интеграция, наблюдаемость и защита	12	24

Как устроена практика

На занятиях разбираются подход и ключевые решения. После каждого модуля вы завершаете мини-проект, проверяете его по критериям и оформляете результат. Итоговый проект объединяет эти работы; это один развиваемый кейс, а не дополнительная нагрузка сверх 120 часов.

Инструменты курса

Python 3 • Flask • SQLAlchemy • PostgreSQL • HTTP • RabbitMQ / pika • Kafka • pytest • Docker Compose

Ведущий преподаватель

Сергей Валерьевич Герасименко - технический директор и основатель IT-компаний «ТЕРАКОР» и «ТЕРАКОР Digital», ведущий преподаватель и практикующий эксперт.

Изучаете. Применяете. Проверяете.

МОДУЛЬ 01 / 4 ак. ч. занятий + 8 ак. ч. самостоятельно

Границы сервисов и рабочий стенд

- Монолит и микросервисы: когда разделение оправдано
- Автономность, владение данными и сетевые отказы
- Конфигурация, зависимости и Docker Compose
- Flask API, health endpoint и схема взаимодействий

ПРОЕКТ ПОСЛЕ МОДУЛЯ / Каркас платформы

Запустить два минимальных сервиса и описать границы данных и ответственности.

Готово, когда: Стенд запускается по инструкции; у каждого сервиса есть health endpoint и отдельная роль.

МОДУЛЬ 02 / 8 ак. ч. занятий + 16 ак. ч. самостоятельно

HTTP API и хранение данных

- Flask: маршруты, слои и обработка ошибок
- SQLAlchemy, PostgreSQL и миграции
- Валидация, OpenAPI и версионирование
- Аутентификация, авторизация и интеграционные тесты

ПРОЕКТ ПОСЛЕ МОДУЛЯ / Сервис заказов

Реализовать создание, просмотр и изменение заказа с проверкой доступа.

Готово, когда: Чужой заказ недоступен; ошибки предсказуемы; миграции и CRUD проверены тестами.

Изучаете. Применяете. Проверяете.

МОДУЛЬ 03 / 6 ак. ч. занятий + 12 ак. ч. самостоятельно

Интеграции и устойчивость

- HTTP-клиент, таймауты и бюджет повторов
- Идемпотентность и circuit breaker
- Компенсации Saga на небольшом сценарии
- Correlation ID, REST и обзор JSON-RPC

ПРОЕКТ ПОСЛЕ МОДУЛЯ / Надёжное резервирование

Связать заказ и резервирование, обработать таймаут и отказ внешнего сервиса.

Готово, когда: Повтор запроса не удваивает резерв; отказ фиксируется; сценарий компенсации воспроизводим.

МОДУЛЬ 04 / 10 ак. ч. занятий + 20 ак. ч. самостоятельно

События и брокеры сообщений

- RabbitMQ: очередь задач, ack, retry и dead-letter
- Kafka: ключ, partition, consumer group и offset
- At-least-once, дубликаты и ограничения exactly-once
- Outbox и идемпотентный обработчик

ПРОЕКТ ПОСЛЕ МОДУЛЯ / Доставка уведомлений и событий

Сначала создать очередь уведомлений RabbitMQ, затем поток событий заказов Kafka с outbox.

Готово, когда: Два проекта имеют отдельные запуски; повтор события безопасен; сбой публикации восстанавливается.

Изучаете. Применяете. Проверяете.

МОДУЛЬ 05 / 12 ак. ч. занятий + 24 ак. ч. самостоятельно

Интеграция, наблюдаемость и защита

- Unit, integration и сквозные тесты
- Контейнеры, non-root, секреты и конфигурация
- Структурированные логи, метрики и readiness
- Сценарии отказов, документация и защита

ПРОЕКТ ПОСЛЕ МОДУЛЯ / OrderHub

Объединить сервисы в один стенд и показать заказ от API до события и уведомления.

Готово, когда: Успешный и аварийные сценарии автоматизированы; по correlation ID можно найти путь запроса.

Один кейс - несколько инженерных решений

Сохраняйте версии проекта после каждого этапа: так в портфолио будет виден путь от первого API до взаимодействия сервисов, обработки сбоев и автоматических проверок.

На защите важно не только показать работающий сценарий, но и объяснить границы решения: что произойдёт при повторе, таймауте или недоступности компонента.

Ваш главный кейс в портфолио

OrderHub: микросервисная платформа заказов

Учебная платформа включает сервис заказов, сервис резервирования и обработчик уведомлений. Основной сценарий использует Kafka для событий; RabbitMQ изучается в отдельном проекте очереди задач.

Критерии готовности

- Сервисы владеют своими данными и взаимодействуют через контракты.
- Событие заказа публикуется через outbox; повтор не дублирует бизнес-операцию.
- Сбой резервирования приводит к понятному статусу и компенсации.
- Есть тесты успеха, таймаута, повторной доставки и недоступности обработчика.
- Docker Compose запускает стенд; README описывает диагностику и ограничения.

Что останется после обучения

Репозиторий с исходным кодом, учебные данные или конфигурация стенда, автоматические проверки, инструкция запуска и описание решений. Это материал для демонстрации навыков на работе и техническом собеседовании.

Как представить результат

Покажите задачу, запустите основной сценарий, продемонстрируйте обработку ошибки и объясните один архитектурный выбор. Отдельно зафиксируйте, что можно улучшить следующим шагом.

Итоговый кейс собирается из проектов модулей. Его завершение и подготовка защиты включены в часы последнего модуля.

Что вы сможете делать

- Выделять границы сервисов и владение данными.
- Разрабатывать Flask API с PostgreSQL и проверяемыми контрактами.
- Настраивать ограниченные повторы, идемпотентность и обработку сообщений.
- Объяснять компромиссы согласованности и диагностировать сквозной сценарий.

Практическое обучение в ТЕРАКОР

Живой онлайн-формат позволяет задавать вопросы и разбирать решения. Программа сочетает занятия с преподавателем и самостоятельную практику. После успешного прохождения - сертификат ТЕРАКОР и оформленный проект.

Перед записью

Обсудите исходный уровень, рабочую задачу и время для практики. Для уверенного результата важно проходить модули последовательно и завершать проекты. Курс развивает конкретные навыки; профессиональный уровень зависит также от предыдущего опыта и объёма самостоятельной работы.

Начните с консультации по программе

Подберём стартовый уровень и уточним формат обучения.

Открыть страницу курса → | info@terakor.ru

+7 (960) 352-38-05 • terakor.ru

Сроки старта, стоимость и состав выбранного тарифа уточняются при записи.

Программа оптимизирована на основе страницы курса ТЕРАКОР, изученной 29.09.2026. Это предлагаемая редакция для обновления курса и его описания.

Источник: terakor.ru/catalog/python/microservices/