Архитектуры и шаблоны проектирования Python

Урок 9

Микросервисная архитектура

Плюсы и минусы. Sync или async, REST или сообщения. Переход от монолита к микросервисам.

На этом уроке разберём

- Минусы монолитной архитектуры.
- Плюсы и минусы микросервисной архитектуры.
- Божественные объекты и DDD.
- От монолита к микросервисам.
- Sync/Async, REST/Messages.

Минусы монолитной архитектуры

1. Запуск тестов занимает длительное время (например, несколько дней).
2. Код проекта становится таким большим, что он долго загружается в ide и в нём трудно ориентироваться.
3. Из за множества зависимостей система становится хрупкой и при каждом новом изменении есть риск сломать другие части (частично этого можно и нужно избегать, применяя принципы и паттерны, рассмотренные нами на предыдущих занятиях, но, как показывает практика, с течением времени система всё равно становится более жёсткой).
4. Обычно организационная структура похожа на структуру системы (закон Конвея). Поэтому команда разработки становится огромной и так же жёстко связанной.
5. В бизнес-логике системы появляется всё больше *Божественных объектов*, что ещё крепче связывает части системы.

Плюсы микросервисной архитектуры

- Независимость частей системы и, как следствие, гибкость и отсутствие хрупкости.
- Возможность очень быстро вносить изменения в систему и получать обратную связь от пользователей.
- Отсутствие *Божественных объектов*.

Минусы микросервисной архитектуры

- Трудность развёртывания и интеграционного тестирования.
- Риск превращении системы в распределённый монолит.
- Высокая нагрузка на сеть.

Пример разработки системы

Бизнес-система по ремонту и продаже смартфонов.

Есть сайт с прайс-листом по продаже и ремонту смартфонов, где пользователь может сделать заказ на покупку или ремонт смартфона.

После передачи смартфона в ремонт техник заказывает запчасти и запускает процесс ремонта.

Также можно купить новый смартфон и оплатить различными способами (например, в личном кабинете).

Рассмотрим на примерах

- Связывание из-за возникновения *Божественных объектов*.
- Использование DDD для улучшения системы.
- Переход к микросервисам и создание более гибкой системы.
- Sync- и async-подходы к микросервисам.

Практическое задание

В этой самостоятельной работе тренируем умения:

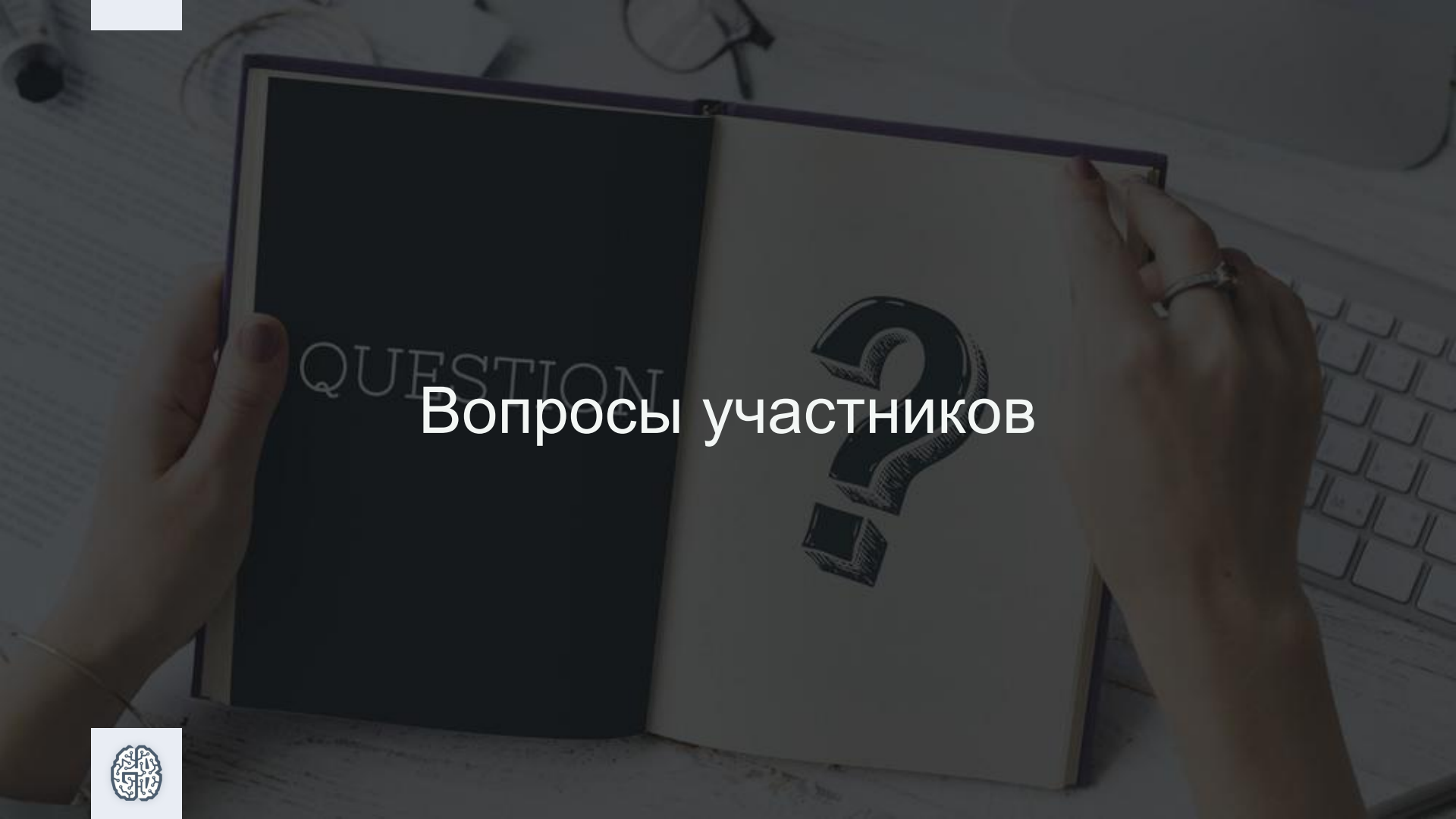
1. Применять на практике знания, полученные на курсе

Смысл:

Для использования всего пройденного в будущих проектах

Последовательность действий:

1. Доделать проект в рамках практических заданий по урокам 1-8. Можно добавить любой новый полезный функционал в wsgi-фреймворк бизнес логику или взаимодействие с базой данных.
2. *Продумать архитектуру проекта для командной разработки (следующий курс)



Вопросы участников