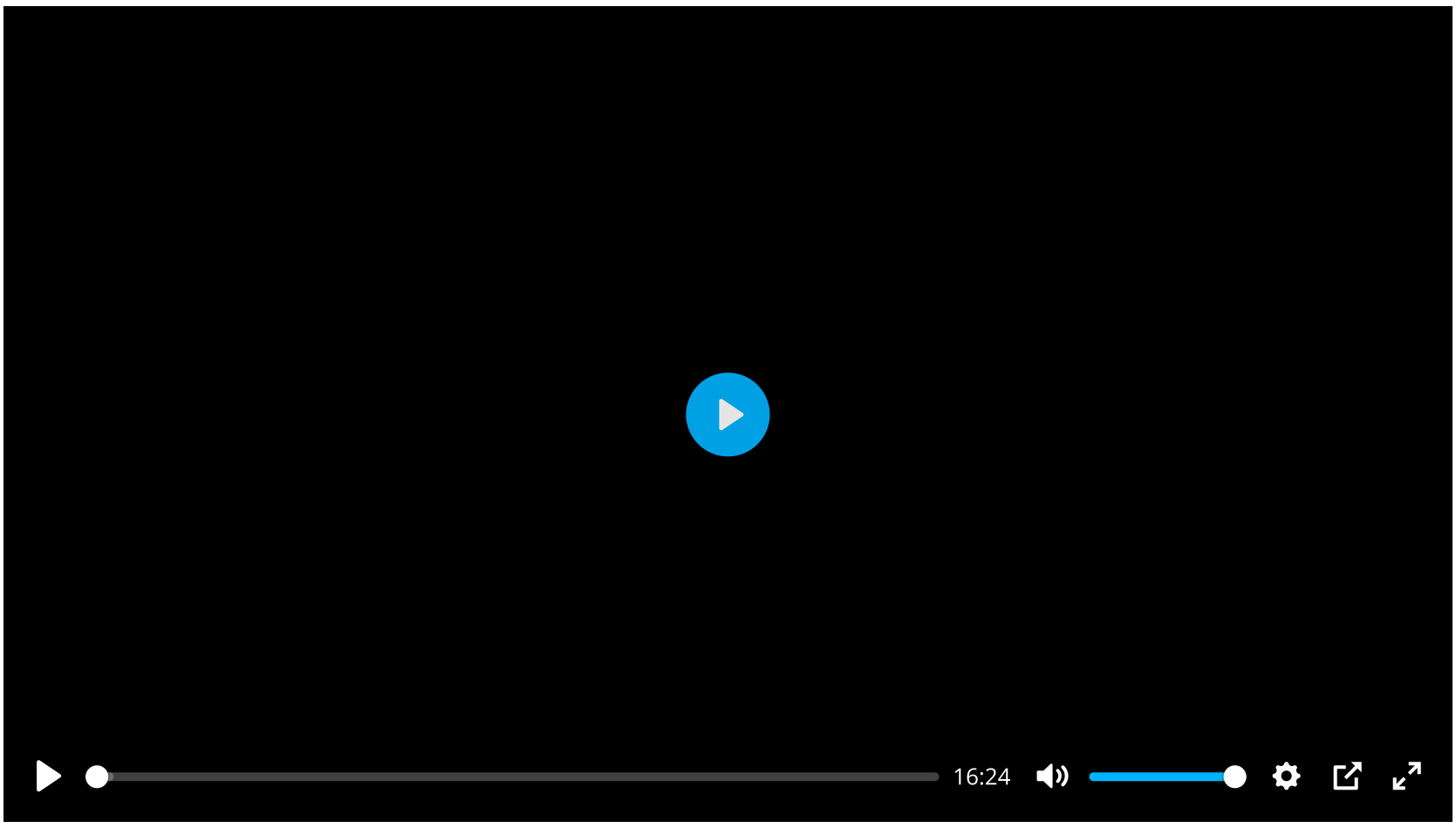
2.4 Web серверы, создание API



Текстовая расшифровка видео:

ИЗ ЧЕГО СОСТОЯТ СЕРВИСЫ

План:

- Representational State Transfer;
- GraphQL;
- OpenAPI (Swagger);
- Swagger UI;
- API Gateway.

Representational State Transfer (REST)

Одним из первых концептов, с которым вы можете столкнуться является «REST». REST расшифровывается как «Representational State Transfer». Это не конкретный механизм или спецификация, описывающая то, как нужно писать веб-сервисы, а набор общих архитектурных правил, которые являются рекомендованными способами реализации.

Нюансы:

- Многие рекомендуют REST к использованию, потому что он привносит хорошие концепции.
- Официальной спецификации, описывающей REST, не существует.

Базовые идеи:

- Протокол HTTP поддерживает большое количество методов даже несмотря на то, что в вебе в основном используются только POST и GET.
- Эти методы условно ложатся на управление объектами и коллекциями этих объектов, а URL явно их идентифицирует:

/api/v1/animals/123/ < – конкретный объект «животное»;

/api/v1/animals/ < – коллекция объектов «животное»;

- Частый вариант применения методов: GET – чтение, POST – создание, PUT – обновление, DELETE – удаление;
- В идеале сервер не должен хранить никакое состояние (сессию), а вся информация для обработки должна передаваться в запросе.

GraphQL

GraphQL – альтернатива и противовес REST. Это более современный и модный подход.

Некорректно сравнивать GraphQL и REST. GraphQL часто пытаются продать, как «лучший REST», однако приложения на обоих подходах могут быть функционально эквивалентны друг другу.

GraphQL – спецификация не архитектуры, а языка запросов к бэкенду.

Язык запросов GraphQL выглядит следующим образом:

```
query GetPets {
  pets {
    name
    petType
  }
}
```

```
{
  "data": {
    "pets": [
      {
        "name": "Sandy",
        "petType": "Cat"
      },
      {
        "name": "Hank",
        "petType": "Dog"
      }
    ]
  }
}
```

Это некое подобие JSON или же аналог SQL.

Несмотря на функциональность GraphQL и его возможность установления определенных контрактов взаимодействия, он довольно сложен в реализации.

Не подходит:

GraphQL сложно поддерживать, поэтому не стоит использовать его для небольшого API.

Подходит:

Если необходимы графы и сложные взаимосвязи объектов.

Есть два кейса, где GraphQL использован удачно:

- API Facebook'a;
- API Github'a.

OpenAPI (Swagger)

OpenAPI (Swagger) – это стандарт описания взаимодействия между клиентом и сервером в виде набора полей и объектов, которые будут пересылаться.

Рассмотрим пример:

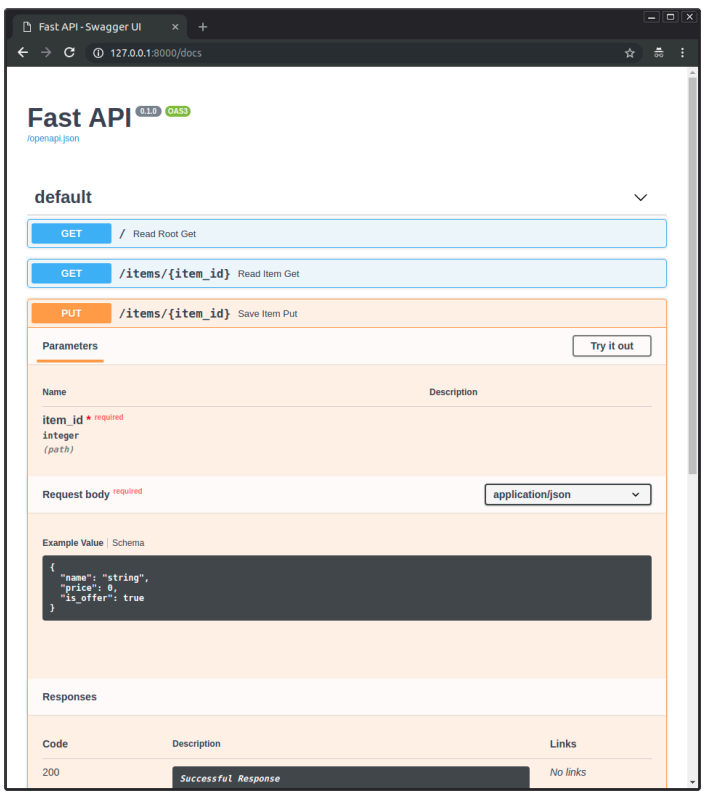
```yaml
openapi: 3.0.0
info:
 title: Sample API
 description: Sample description of a web service
 version: 0.1.9
servers:
 - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
paths:
  /users:
    get:
      summary: Returns a list of users.
      responses:
        '200':   # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
              type: array
              items:
                type: string
```

Если мы посмотрим на пример, то заметим, что у нас есть полноценные метаданные и спецификация на веб-сервис (с полной характеристикой). Это буквально всеобъемлющее описание спецификаций, которое позволяет с разных сторон четко описать требования.

OpenAPI помогает найти общий язык между фронтендом и бэкендом, позволяет разрабатывать их параллельно и независимо.

Существуют как инструменты по генерации кода на основе спеки, так и встроенная в фреймворки поддержка генерации самой спеки по коду приложения.

Swagger UI



Набор инструментария для Swagger включает в себя Swagger UI – веб-интерфейс, который одновременно может выступать в качестве готового HTTP-клиента к UI. Сам по себе он является документацией, где можно посмотреть объекты и методы.

Помимо этого, можно делать тестовые запросы и демонстрировать логику работы API напрямую через Swagger UI.

API Gateway

Это общий механизм авторизации для доступа сразу ко многим сервисам.

Часто API Gateway представляет собой прокси-сервер, позволяющий дополнительно настроить балансировку, маршрутизацию и даже разные элементы уровня приложения.

Мы выносим в единое место сразу решение пачки проблем, то есть, мы можем настроить как балансировку запросов отдельно для сервиса, так и решить более сложные аспекты маршрутизации. Также, мы можем всю авторизацию для всего облака микросервисов настроить в едином месте.

Существуют разные сервисы, например:

- WSO2;
- Kong;
- Amazon API Gateway.

Как вам урок?

Далее