FSD01 - APi Gateway & Integration
Atividade Final - Teórica

GRUPO D
Andrade Alves Sampaio
David Stefano Aranda da Silva
Irai Fernandes Daniele
Marília Gabriela Pires Matos
Paolo Angelo Martins

O que é uma API?
Uma API surge como um dos principais meios de integração no desenvolvimento de Software sendo um conjunto de definições e protocolos que viabilizem a comunicação de solução ou serviço sem a necessidade de conhecimento prévio de como os mesmos foram implementados. Ela existe como uma forma de contrato, que controla como os desenvolvedores irão interagir com a solução, criando um ambiente mais seguro e controlado, além de principalmente garantir uma comunicação mais simplificada e ágil.

Quando usar?
Visando a série de benefícios de uma API, a sua principal utilização pode ser para economia de trabalho através do uso de uma já existente, onde pode-se agregar valor a solução trazendo recursos mais completos, muitas vezes desenvolvidos por especialistas na funcionalidade desejada.
A forma contratual que uma API se apresenta é perfeita para uso privado, provendo facilidade para diversas equipes trabalharem em conjunto, trazendo maior controle e organização para os projetos.
Também é uma ótima forma de monetizar soluções, trazendo a possibilidade de diversas abordagens, tais como por acesso ou até mesmo por uso.

Modelos de Maturidade
Os modelos de maturidade de uma API consistem em quatro níveis, classificados de 0 à 3, sendo eles:

- Nível 0
  Sendo o ponto de partida para a implementação, se dá através do uso de HTTP como mecanismo de interação e não usa nenhum dos pilares para sua construção.
- Nível 1
  É implementado através do uso de recursos, sendo assim, ao invés de todos os pedidos responderem através de um único endpoint, os recursos passam a responder de forma individual, baseando-se no uso das URLs.
- Nível 2
  Este é um nível onde os verbos HTTP são essenciais para um implementação adequada, porém não basta somente utilizar GET/POST/PUT para atendê-lo, até porque os mesmos já aparecem em níveis anteriores, sendo então necessário levar outros aspectos em consideração. Um deles é o de Idempotencia, que se baseia em garantir os métodos GET, HEAD, DELETE e PUT mantenham o servidor no mesmo

estado, não resultando em nenhum efeito colateral e como consequência, se uma requisição idêntica for realizada diversas vezes, o efeito deve ser o mesmo. Outro ponto a ser levado em consideração para atingir o nível 2 é a segurança de cada método HTTP, garantindo que os métodos GET e HEAD sejam seguros, ou seja, não alterem o estado do servidor e classificando os demais como PUT, DELETE, POST ou PATCH como inseguros.
Por último temos os códigos de resposta HTTP que devem ser utilizados corretamente de acordo com o resultado obtido. Tais como: Respostas de informação(100-199), Respostas de Sucesso(200-299), Redirecionamentos (300-399), Erros do Cliente (400-499), Erros do servidor(500-599). Além da resposta com códigos adequados, é importante que a API responda com algumas informações adicionais, como por exemplo em caso de sucesso, devolver 201 acompanhado de um código para que o cliente possa acessar o estado atual deste recurso futuramente e até mesmo poupá-lo de uma requisição a mais.

- Nível 3
  O último nível de maturidade de uma API pode ser atingido através do controle de hipermídia, permitindo que um documento descreva o estado do elemento e seu relacionamento com outros. Neste caso é devolvida uma coleção para que o cliente consiga identificar o que de fato está implementado e não necessariamente precise ter um conhecimento prévio, podendo até mesmo utilizar essa coleção como ponto de partida. Em resumo está introduzindo a descoberta, disponibilizando um protocolo auto-documentado.

Como documentar

Como uma excelente forma de documentação de API, temos o Swagger, sendo uma ferramenta OpenSource que visa auxiliar os desenvolvedores no processo de criação, documentação e consumo,padronizando todo o processo e descrevendo os recursos que uma API deve possuir, tais como os ENDPOINTS disponíveis, o formato dos dados recebidos e retornados, códigos HTTP e os métodos de autenticação. A grande vantagem do uso do Swagger é a possibilidade da documentação evoluir no mesmo ritmo da API, uma vez que todo o processo é feito de forma automática pela ferramenta, gerando posteriormente uma interface padronizada e amigável.