WEBSERVICES & RESTFUL TECHNOLOGIES

DEVELOPER EXPERIENCE

EDUARDO GALEGO

4

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

4 DEVELOPER EXPERIENCE

O que vem à sua mente quando você ouve a palavra “Web Services”? Para muitos surge uma porção de palavras técnicas, como REST, WSDL e HTTP, entre outras.

Entretanto, nos últimos anos, o conceito de Web Services tem sido encarado pelo mercado através de um sentido de Negócio: empresas ganham dinheiro expondo seus *softwares* como um serviço. Trata-se de um novo canal de interação com os clientes.

Para exemplificar: imagine que sua empresa está desenvolvendo um e-Commerce e, dentre as opções de pagamento, quer oferecer a modalidade de “Pagamento por Pix”. Para viabilizar isto, precisamos mapear toda a integração entre o Banco Central e as instituições de pagamento, um trabalho extremamente complexo.

Por se tratar de uma tarefa que não pertence ao *core* do negócio da empresa, existe a possibilidade de terceirizar esta tarefa, deixando que outro *software* faça o trabalho mais difícil. Neste sentido, a empresa contrata um serviço de outra empresa, responsável por expor um Web Service para a execução do pagamento por Pix.

Assim que o usuário escolher esta modalidade de pagamento, o E-Commerce interage com o Web Service da empresa contratada, que ficará responsável por toda a complexidade da operação. Em contrapartida, a empresa contratada cobra um valor de assinatura ou por consumo.

Grandes empresas se destacaram neste novo modelo de negócio, sendo a AWS uma das pioneiras e líderes do mercado. Em meados dos anos 2000, a empresa passou a oferecer serviços de infraestrutura utilizando Web Services como base. Para saber mais, leia o artigo *How AWS came to be*, que está disponível nas referências.

Outra importante leitura neste aspecto de Web Service como Negócio é o livro *The API Product Mindset*, que também está disponível na seção de Referências.

Neste modelo de negócio contemporâneo, um novo agente passa a ganhar espaço: o desenvolvedor. Ele é o responsável pela construção das aplicações consumidoras de Web Service, viabilizando assim toda a integração.

Assim, é importante passar a olhar com mais atenção no que diz respeito à perspectiva do desenvolvedor. Deve-se levar em conta questões como:

- A documentação do Web Service está clara, completa e acessível?
- O Web Service segue padrões de mercado, como o REST?
- Consigo fazer um teste rápido no Web Service?
- Se encontrar erros, consigo acesso fácil e rápido aos *logs*?
- Em caso de erros ou dúvidas, consigo acesso rápido à equipe técnica?

É neste sentido que surge então o conceito de **Developer Experience**, também conhecido como **DX**.

Na mesma linha do *User Experience* (que tem como objetivo medir e melhorar a experiência do usuário), o ***Developer Experience*** mede a qualidade da experiência do desenvolvedor na utilização dos serviços e estabelece metas para melhorar e surpreender, visando aumentar a adesão à utilização do serviço.

Claramente, existem também desenvolvedores que criam *softwares* nas empresas. Para eles, o Developer Experience é pensado na forma de colaboradores internos. Neste capítulo, vamos abordar com mais detalhes a experiência dos desenvolvedores externos, uma visão mais próxima do cliente.

Na abordagem de Web Services como um Negócio, o serviço pode ser nomeado como *API* (acrônimo de *Application Programming Interface* ou interface de programação de aplicação). Este termo já existe bem antes dos Web Services, quando as plataformas de programação ofereciam uma forma de exportar funcionalidades em forma de bibliotecas. Mas o termo voltou a ser utilizado atualmente no contexto dos Web Services.

A Figura “Aspectos do DX para melhorar a experiência do desenvolvedor” apresenta diversos aspectos que podem melhorar a experiência do desenvolvedor de APIs.

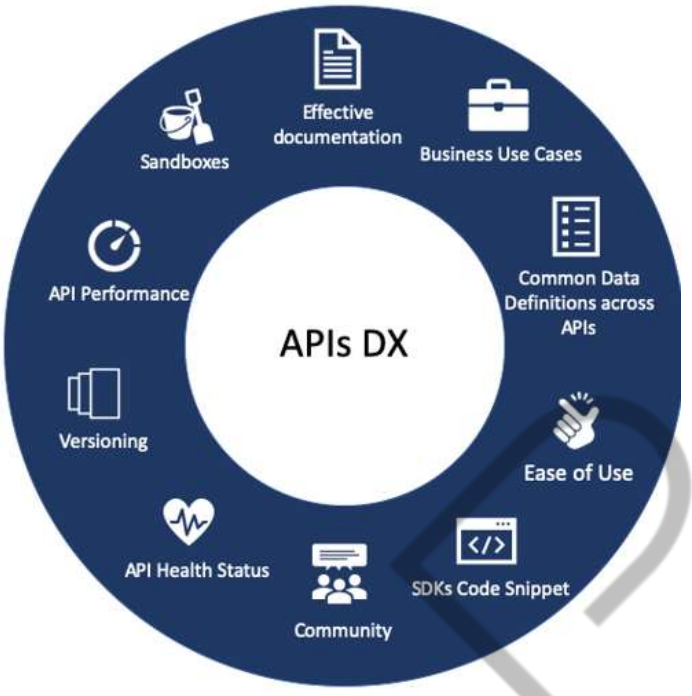

Figura 4.1 – Aspectos do DX para melhorar a experiência do desenvolvedor
Fonte: NeosAlpha (2022)

Tendo isso em mente, os tópicos mais relevantes serão detalhados nas próximas sessões.

4.1 FACILIDADE DE USO

A experiência do desenvolvedor cai à medida que ele tenta usar uma API e é surpreendido com algum erro. Isto é frustrante, pois demonstra que ele não sabe utilizar a API e/ou o serviço não funciona como esperado.

Como visto anteriormente, para aqueles que trabalham com DX, é preciso entender os motivos por trás desta má experiência e prover melhorias. Uma destas melhorias é a padronização da API.

Por vezes, o desenvolvedor da aplicação consumidora está se deparando com um erro porque conhece e está utilizando os padrões REST, sendo que a API não está.

Um exemplo: o usuário está querendo criar um novo recurso PRODUCT, porém está tomando o seguinte erro na chamada `POST /api/product`: 405 Method Not Allowed. Entretanto, a ação para criar um novo produto foi implementada da seguinte forma: `GET /api/product?action=create`.

Alguém pode argumentar: “A ação de criar existe e está descrita na documentação. Culpado é o desenvolvedor que não leu a documentação!”. De fato, não estamos discutindo se a API funciona ou não, mas como o desenvolvedor se sente utilizando-a.

Ele (e milhares de outros desenvolvedores) conhecem REST e provavelmente já desenvolveram outras integrações. Assim, ele se sentirá mais confiante consumindo uma API que segue o padrão, tornando as chances de pegar um erro menores.

Desta forma, quanto maior a experiência do desenvolvedor acerca de uma API, maior as chances de ela ser escolhida pela empresa e adquirida. E quanto maior o consumo, maior a receita da empresa.

Logo, a regra de ouro é: siga o máximo possível os padrões de mercado, assim mais e mais desenvolvedores irão se sentir “em casa” ao operar suas APIs.

Outras dicas para aumentar a facilidade de uso de sua API são:

1. Em caso de erros do consumidor, instrua o desenvolvedor sobre o que fazer. Exemplo: instrua quais campos têm seu preenchimento obrigatório ao retornar um erro *422 Unprocessable Entity*.
2. Instrua quais são as ações possíveis de serem chamadas utilizando HATEAOS. Para saber mais, acesse o artigo *HATEOAS Driven REST APIs*,disponível nas referências.
3. Evite o uso exagerado de recursos aninhados. Por exemplo: `GET /api/orgao/1/local/12/servico/12/agendas`. Opte por utilizar *query params*.

4.2 DOCUMENTAÇÃO DE APIS

Discutimos na disciplina algumas das diferenças entre SOAP e REST e vimos que o SOAP possui uma forma nativa para especificação do contrato: o WSDL.

O REST, no entanto, nasceu sem uma forma de especificação. A ideia era que, pela própria natureza da especificação, seria mais fácil de usar. Os desenvolvedores iriam “aprender” a utilizar o Web Service à medida que fossem utilizando-o. Todavia, no dia a dia, a necessidade de documentação surgiu.

O WADL (acrônimo de *Web Application Description Language*) surgiu com o objetivo de descrever qualquer recurso disponível na Web baseado em HTTP. Trata-se de uma especificação da W3C. Para mais detalhes, acesse a especificação em (https://www.w3.org/Submission/wadl/).

Descrevemos em WADL utilizando o formato XML. Segue um exemplo no Código-fonte “Exemplo de um WADL descrevendo um Web Service REST de livros” de um arquivo WADL.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="books">
      <method name="GET"/>
      <resource path="{bookId}">
        <param required="true"
               style="template"
               name="bookId"/>
        <method name="GET"/>
      </resource>
    </resource>
  </resources>
</application>
```

Código-fonte 4.1 – Exemplo de um WADL descrevendo um Web Service REST de livros
Fonte: Elaborado pelo autor (2022)

Já o OpenAPI (também chamado de OAS, acrônimo de *OpenAPI Specification*) foi criado a partir da especificação Swagger 2.0 (renomeado em 2015). Atualmente, ele está na versão 3.1.0. Para saber mais detalhes, acesse a especificação em (https://spec.openapis.org/oas/latest).

É possível trabalhar com OpenAPI tanto no formato de dados JSON quanto YAML. No Código-fonte “Trecho de um OpenAPI em YAML descrevendo um Web Service REST de alunos” é possível observar um trecho de um arquivo no formato YAML.

```yaml
openapi: 3.0.2
info:
  title: exemplo
  description: Simples exemplo de OpenAPI.
  version: 0.0.1
  servers:
  - url: http://api.fiap.com.br/v1
    description: Produção
paths:
  /alunos:
    get:
      summary: Retorna uma lista de alunos
      responses:
        '200':    # status code
          description: Sucesso
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

Código-fonte 4.2 – Trecho de um OpenAPI em YAML descrevendo um Web Service REST de alunos
Fonte: Elaborado pelo autor (2022)

Em uma abordagem API First para projetos de *software*, antes de qualquer implementação, é definida a interface, ou seja, a documentação. A partir daí, duas novas frentes de trabalho podem ser executadas de maneira paralela: a implementação do serviço (normalmente especificado como a etapa de *back-end*) e a utilização do serviço a partir da construção de uma aplicação consumidora (etapa de

front-end). Nesta última etapa, pode-se utilizar um Mock da API, ou seja, as chamadas aos métodos retornam valores fixos.

Esta abordagem pode acelerar o projeto de *software*, uma vez que a equipe de *front-end* não precisa aguardar a conclusão da implementação do *back-end* para iniciar seus trabalhos.

Entretanto, diversos *softwares* já possuem sua implementação de API sem que a devida documentação tenha sido feita. Para estes casos, há diversos *plugins* geradores de documentação. Isto é muito útil, pois parte da documentação pode ser anotada no código-fonte e a documentação completa é construída juntamente com o artefato de *software*.

Uma lista extensa dos geradores foi catalogada e se encontra em (https://openapi-generator.tech/docs/generators/).

Há também um exemplo de projeto que utiliza-se de um gerador de documentação para Java Spring em (https://github.com/prof-eduardo-galego/fiap.scjo.agenda).

4.3 VERSIONAMENTO

Abordamos o versionamento de Web Services REST em nossa primeira aula. O que queremos aqui, nesta seção, é ressaltar a importância da construção e manutenção de um ***Release Notes*** para melhorar a experiência do Desenvolvedor.

O Release Notes é um catálogo de alterações no *software* por versão. Ele descreve as alterações no contexto não-técnico, ou seja, é mais facilmente compreensível por alguém que não possui linguajar técnico.

É um pouco diferente do ***Changelog***, que por vezes acaba sendo um documento interno, o qual detalha os aspectos técnicos da correção ou melhoria e é voltado para os desenvolvedores do serviço. Para saber mais a diferença sobre *Changelog* e *Release Notes*, veja o artigo *The Difference Between A Changelog and a Release Note*, disponível na seção Referências.

Para o desenvolvedor que consome serviços, este catálogo melhora sua visão sobre os aprimoramentos (seja tanto correções de erros quanto novas funcionalidades) que os serviços vêm recebendo, além de auxiliar na decisão para melhorar a própria aplicação consumidora. Trata-se, portanto, de um importante painel de comunicação entre provedores e consumidores de serviços.

4.4 DEVELOPER PORTAL

Grande parte da estratégia de DX falha quando não há um canal de comunicação entre os desenvolvedores e a empresa responsável por prover o serviço. Não é difícil encontrar no mercado empresas que ainda insistem em estabelecer este canal por meio de e-mails ou mensagens eletrônicas.

Assim, é necessário que haja um canal de comunicação ativo e com constante interação que permita ao desenvolvedor de aplicações consumidoras aprender sobre o serviço, seja interagindo dinamicamente, acessando uma documentação bem estruturada e/ou interagindo com a comunidade por meio de um fórum de discussões.

Neste sentido, as empresas oferecem ao público um Portal de Desenvolvedor, normalmente chamado de Developer Portal ou simplesmente DevPortal. Alguns exemplos de Portais são:

- Uber: https://developer.uber.com/
- IFood: https://developer.ifood.com.br/pt-BR/
- GetNet: https://developers.getnet.com.br/
- Catho: https://desenvolvedores.catho.com.br/

Em um DevPortal, normalmente encontramos as seguintes funcionalidades:

1. Acesso à documentação de APIs, permitindo que o desenvolvedor possa interagir com os serviços em um ambiente ***sandbox***. Neste ambiente, os dados recuperados são fictícios, os dados armazenados são periodicamente descartados e as operações são apenas *mocks*, ou seja, não expressam a realidade.
2. Também na documentação, podemos extrair trechos de código que expressam a chamada a um determinado serviço, em diversas linguagens de programação. Conhecido como ***snippets***, estes pequenos trechos facilitam o trabalho do desenvolvedor na criação da aplicação consumidora, já que ele pode se aproveitar destes trechos em seu *software*.

3. O desenvolvedor consegue registrar sua aplicação e obter as credenciais de acesso. Desta forma, conseguimos acelerar o processo de ***Onboarding*** de novas aplicações por meio da automatização de processos.
4. Acesso a fóruns de discussões, páginas com conteúdo educativo sobre os serviços e para abertura de chamados, em caso de erros ou dúvidas mais específicas.
5. Painéis para o desenvolvedor ver a performance da API, acesso aos *logs* de erros, páginas com os *releases notes* e páginas com as manutenções futuras, entre outros.

Para aqueles que trabalham com DX, o DevPortal fornece uma visão de acompanhamento de adesão dos desenvolvedores aos serviços. Assim, é possível determinar quais serviços possuem baixa taxa de conversão, ou seja, o desenvolvedor testou a API, mas não adquiriu a licença nem a promoveu para um ambiente produtivo.

Trata-se, portanto, de uma importante ferramenta para medir e melhorar a experiência do desenvolvedor.

CONCLUSÃO

Neste capítulo conhecemos o conceito Developer Experience, que tem como objetivo medir e melhorar a experiência do desenvolvedor. Também exploramos alguns aspectos para melhorar tal experiência, tais como utilização de padrões de mercado, documentações efetivas e versionamento. Finalizamos a aula conhecendo o Developer Portal, uma ferramenta imprescindível para criar um canal de comunicação entre desenvolvedores e melhorar sua experiência.

REFERÊNCIAS

APIGEE GOOGLE CLOUD. **The API Product Mindset**: How to move fast, delight customers, and continually innovate to thrive in today's economy. 2018. Disponível em: <https://cloud.google.com/files/apigee/apigee-api-product-mindset-ebook.pdf>. Acesso em: 03 ago. 2022.

BITBAND. **The Difference Between A Changelog and a Release Note**. 2021. Disponível em: <https://www.bitband.com/blog/the-difference-between-a-changelog-and-a-release-note>. Acesso em: 03 ago. 2022.

CATHO. **Portal dos Desenvolvedores**. 2022. Disponível em: <https://desenvolvedores.catho.com.br/developers-integration-portal>. Acesso em: 03 ago. 2022.

GETNET. **O que é a plataforma digital?** 2022. Disponível em: <https://developers.getnet.com.br>. Acesso em: 03 ago. 2022.

IFOOD. **Estamos revolucionando o universo da alimentação**. Esse é só começo! 2022. Disponível em: <https://developer.ifood.com.br/pt-BR/>. Acesso em: 03 ago. 2022.

GUPTA, L. **HATEOAS Driven REST APIs**. 2021. Disponível em: <https://restfulapi.net/hateoas/>. Acesso em: 03 ago. 2022.

MILLER, R. **How AWS came to be**. 2016. Disponível em: <https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/>. Acesso em: 03 ago. 2022.

NEOSALPHA. **API Developer Experience DX**. 2022. Disponível em: <https://www.neosalpha.com/api-developer-experience/>. Acesso em: 03 ago. 2022.

OPENAPI. **Generators List**. 2022. Disponível em: <https://openapi-generator.tech/docs/generators/>. Acesso em: 03 ago. 2022.

OPENAPI. **OpenAPI Specification v3.1.0**. 2021. Disponível em: <https://spec.openapis.org/oas/latest>. Acesso em: 03 ago. 2021.

UBER. **Solicitações de viagens** - Fazer uma viagempara seus usuários. 2022. Disponível em: <https://developer.uber.com>. Acesso em: 03 ago. 2022.

W3C. **Web Application Description Language**. 2009. Disponível em: <https://www.w3.org/Submission/wadl/>. Acesso em: 03 ago. 2022.

GLOSSÁRIO

REST	REpresentational State Transfer
API	Application Programming Interface
DX	Developer Experience
OAS	OpenAPI Specification
DevPortal	Developer Portal