PUC Minas
Virtual

Estilo Arquitetural de APIs

Marco Mendes

APIs

- Uma API é a interface que uma aplicação de software apresenta a outras aplicações.

- As APIs são os blocos de construção que permitem a **interoperabilidade** para as principais plataformas de negócios na web.

APIs estão em todo lugar

- APIs permitem que identidades sejam criadas e mantidas em contas de software em nuvem, desde seu endereço de e-mail corporativo, a softwares de design colaborativo

- APIs habilitam os o compartilhamento de dados de previsão do tempo para aplicações.

- APIs processam seus cartões de crédito e permitem que as empresas recebem o dinheiro sem preocupações regulatórias.

- No nível pessoal, aplicativos de transporte, comunicação instantânea ou para pedir comida usam APIs.

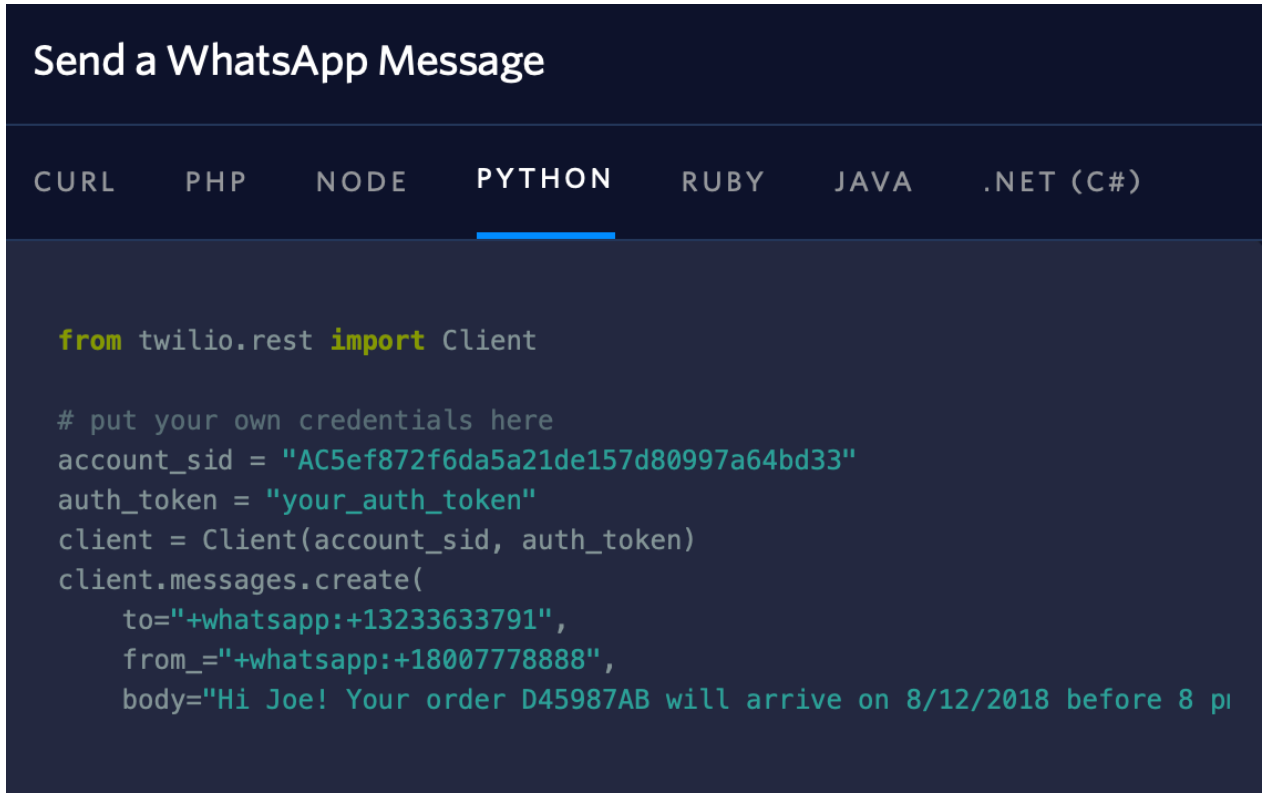
APIs simplificam o uso de outros softwares

Send a WhatsApp Message

CURL PHP NODE PYTHON RUBY JAVA .NET (C#)

```
curl -X "POST" \
    --data-urlencode 'To=whatsapp:+13233633791' \
    --data-urlencode 'From=whatsapp:+18007778888 \
    --data-urlencode 'Body=Hi Joe! Your order D45987AB will arrive on 8/1
    -u {Account Sid}:{Auth Token} \
    https://api.twilio.com/2010-04-01/Accounts/{AccountSID}/Messages.json
```

APIs simplificam o uso de outros softwares

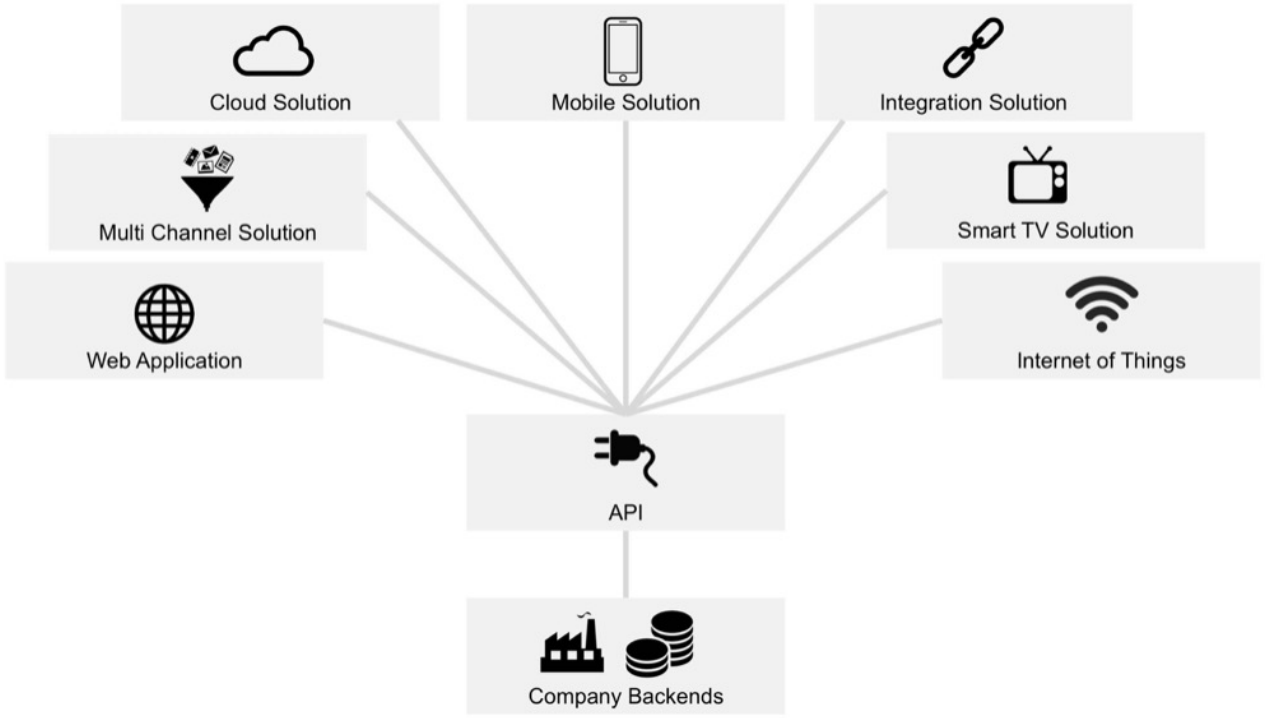


PUC Minas
Virtual

Tipos Comuns de APIs

Marco Mendes

Tipos de APIs



Fonte: API Architecture - The Big Picture for Building APIs, Matthias Biehl

APIs para Soluções Móveis

- O número de dispositivos móveis e tablets superou o número de computadores. Os aplicativos para celular são diferentes dos aplicativos de desktop tradicionais, já que a maioria dos aplicativos móveis não é autônoma nem autossuficiente.

- Os aplicativos precisam se conectar aos servidores na Internet para serem utilizáveis ou, pelo menos, serem utilizáveis em todo o seu potencial.

- Os dados entregues pelas APIs precisam ser leves e particionados. Isso garante que a API possa ser consumida por dispositivos com capacidade de processamento limitada e largura de banda limitada de conexão à Internet.

APIs para Soluções Móveis

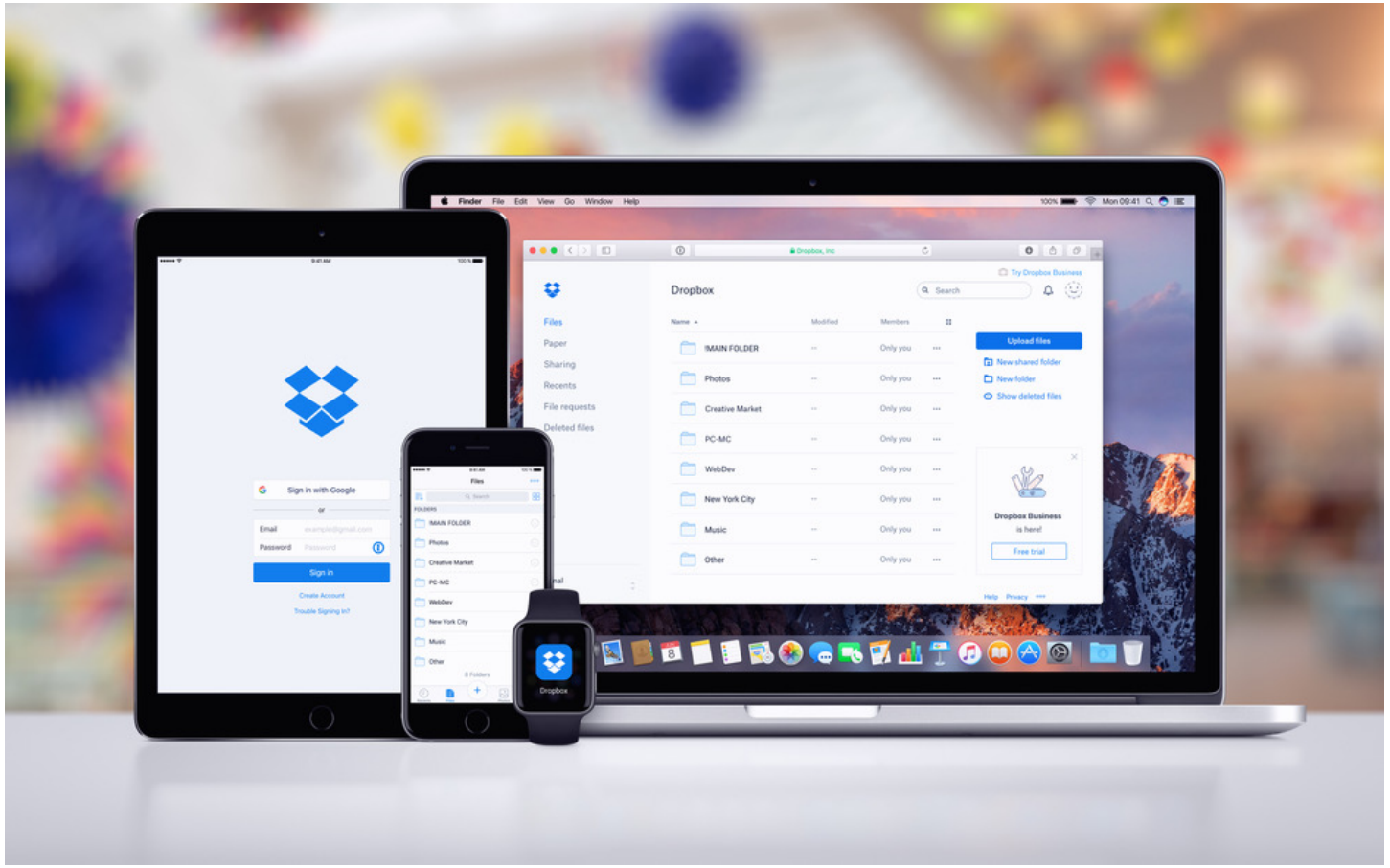
APIs para Soluções de Nuvem

- As soluções em nuvem SaaS normalmente consistem em um aplicativo da Web e APIs. O aplicativo da web é visível para os consumidores.

- Embaixo do capô, as soluções em nuvem geralmente oferecem uma API também, no entanto, a API normalmente permanece sob a superfície. Essa API pode ser usada para conectar o aplicativo de nuvem a outros aplicativos de nuvem para realizar a automação ou para conectar a solução de nuvem a aplicativos móveis e software de desktop.

- O Dropbox é um exemplo para esse tipo de solução em nuvem. A API desta solução de nuvem permite que muitos aplicativos de terceiros se conectem ao Dropbox, incluindo ferramentas de sincronização para dispositivos móveis e desktop.

APIs para Soluções de Nuvem



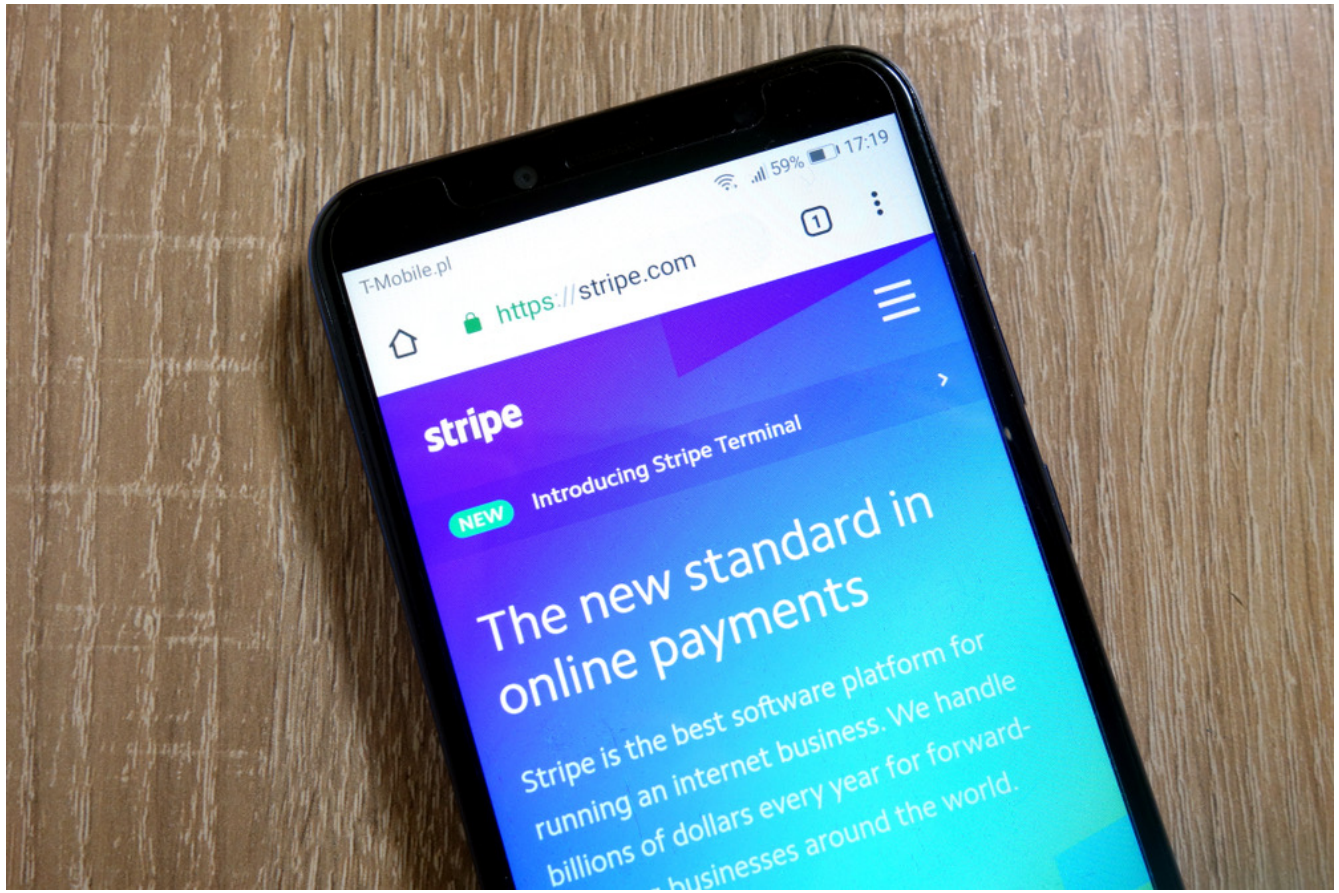
APIs para Soluções de Integração

- APIs fornecem os recursos, que são essenciais para conectar, estender e integrar software. Ao integrar software, as APIs conectam empresas a outras empresas. Eles são usados em soluções de integração de negócios para empresas.

- O negócio de uma empresa pode ser expandido conectando-se os negócios aos parceiros para cima e para baixo na cadeia de valor.

- Como os negócios são executados pela TI, os negócios podem ser mais bem vinculados, integrando os sistemas de TI de um negócio em toda a cadeia de valor aos sistemas de TI de outras empresas, parceiros, funcionários e, é claro, aos clientes.

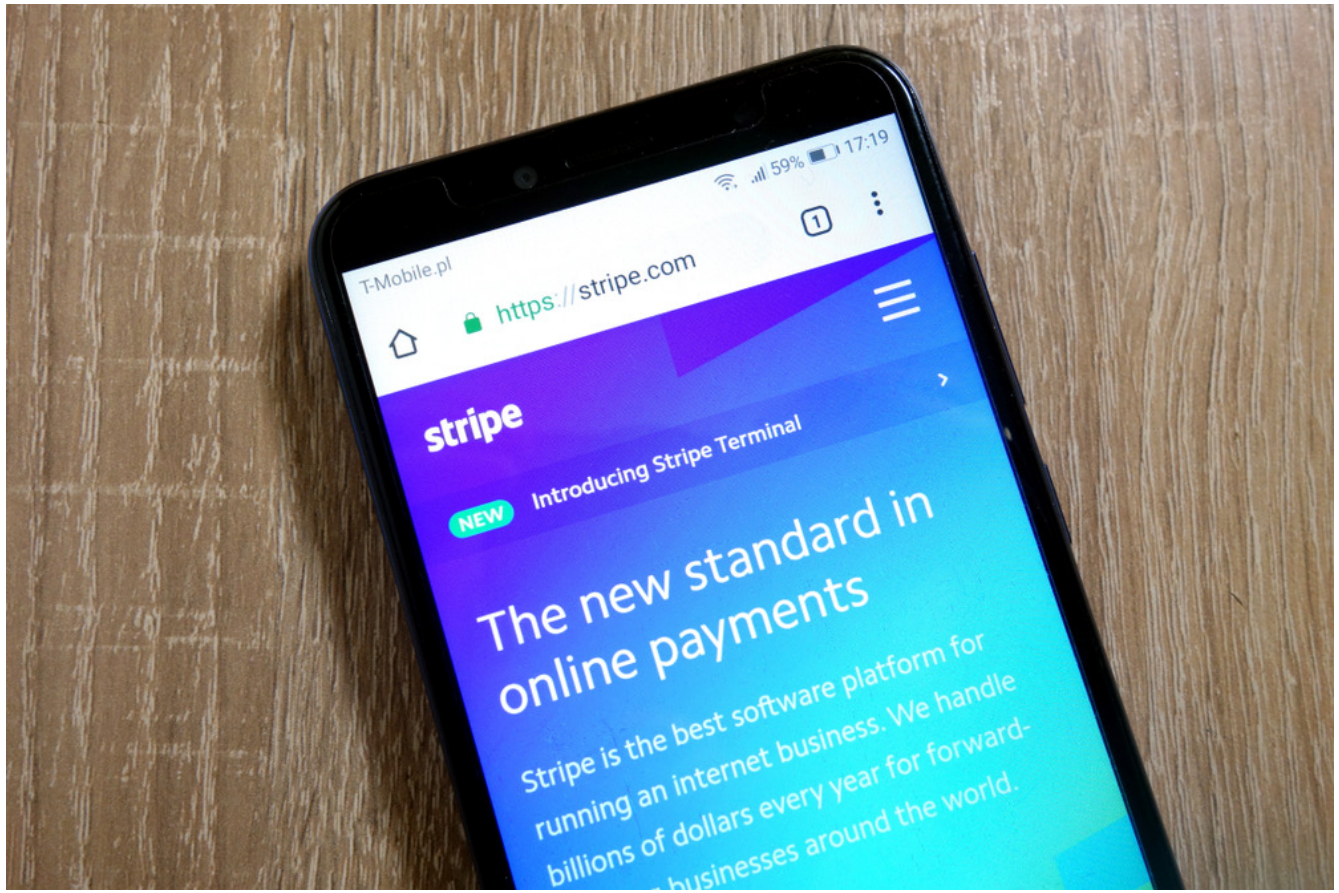
APIs para Soluções de Integração



APIs para Soluções Multi-Canal

- Sistemas diversos oferecem aos clientes a possibilidade de fazer compras em várias plataformas: no celular, na Web ou no tablet.
- Para melhorar a experiência de uso, os mesmos dados e ações do usuário precisam estar disponíveis em todos os dispositivos do usuário, mesmo que sejam construídos em hardware diferente, executem sistemas operacionais diferentes e aplicativos diferentes.
- Soluções Omni-Canal ou soluções multicanais fornecem exatamente isso. Independentemente do canal usado pelos clientes, eles obtêm uma experiência consistente em todos os dispositivos e podem alternar facilmente entre os dispositivos.

APIs para Soluções Multi-Canal



APIs para Soluções IoT

- A internet das coisas é composta de dispositivos físicos com uma conexão à internet. Os dispositivos são controlados por software por meio de seus atores ou os dispositivos podem coletar dados por meio de seus sensores.
- Assim, o dispositivo em si não precisa ser "inteligente", no entanto, ele pode se comportar como um dispositivo inteligente.

APIs para Soluções IoT

- O dispositivo (um sensor ou atuador) se conecta a funções inteligentes, que são expostas na internet por meio de APIs.

- Exemplos de tais soluções API incluem vestíveis inteligentes, carros inteligentes, casas inteligentes ou cidades inteligentes.

APIs para Soluções IoT

PUC Minas
Virtual

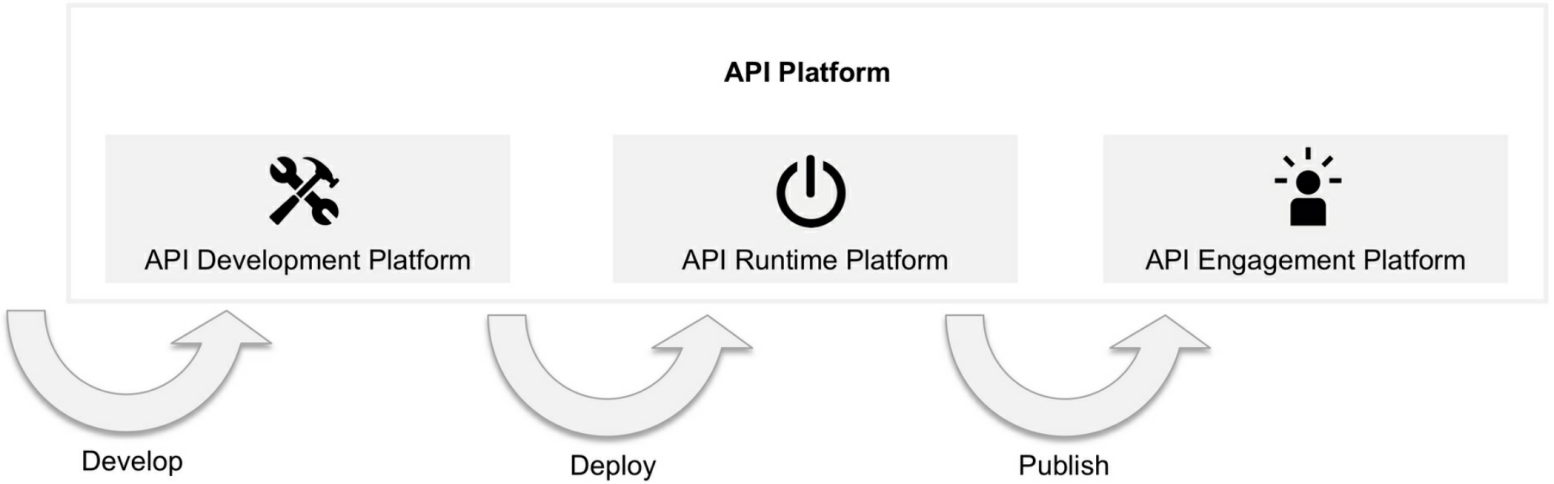
A Agenda do Arquiteto para APIs

Marco Mendes

A agenda do arquiteto para APIs

- Como identificar, escolher e desenhar APIs?
- Como implementar e testar APIs?
- Como documentar e comunicar APIs?
- Como definir tecnologias de APIs?
- Como gerenciar APIs?
- Como descobrir e reusar APIs?

Ciclo de Vida de APIs com Plataformas



Fonte: API Architecture - The Big Picture for Building APIs, Matthias Biehl

O processo de construção de APIs

1. Descobrimos como a maioria dos **consumidores gostaria de usar a nova API**.
2. **Projetamos a API** para que ela se encaixe no portfólio de diferentes APIs que nossa empresa oferece
3. **Escolhemos o estilo arquitetônico**, ou seja, se a API aplica um estilo REST, RPC,SOAP, WebSocket, entre outros.
4. **Projetamos um protótipo da API** usando uma linguagem de descrição da API, como RAML ou Open API (Swagger).

O processo de construção de APIs

5. **Simulamos a API** e criamos um protótipo da API.
6. **Selecionamos a plataforma da API**, que fornece os blocos de construção reutilizáveis para as APIs.
7. **Implementamos e testamos a API**, uma pedaço por vez.
8. **Implantamos e gerenciamos a API** em ambiente de produção
9. **Aumentamos o engajamento** para descoberta e uso das APIs criadas.

PUC Minas
Virtual

Plataformas de APIs

Marco Mendes

Plataforma de Desenvolvimento

- Linguagem para desenhar APIs.
- Ferramenta de desenho e testes de API.
- Ferramenta para gerar documentação e esqueletos de código baseados no desenho da API.
- Biblioteca de blocos de construção da API.
- Linguagem para implementar APIs.
- IDE para desenvolvimento de APIs com editor, depurador e ferramentas de implantação.

Plataforma de Execução

- Balanceamento de carga, pool de conexão e armazenamento em cache
- Monitoramento, auditoria, log ou *analytics*
- Implantação e manutenção
- Gerenciamento de acesso
- Gerenciamento de configuração

Plataforma de Engajamento

- Gerenciamento de API: configuração e reconfiguração de APIs existentes sem a necessidade de implantação da API
- Descoberta de API: um mecanismo para clientes obterem informações sobre a integração do consumidor de APIs (geração de ID de cliente / chave de aplicativos, console de API interativo) , Funcionalidades sociais, etc.)
- Documentação (Idealmente uma documentação interativa que foi gerada a partir da descrição da API)

PUC Minas
Virtual

Arquitetura de Referência de APIs

Marco Mendes

Arquitetura de Referência de APIs

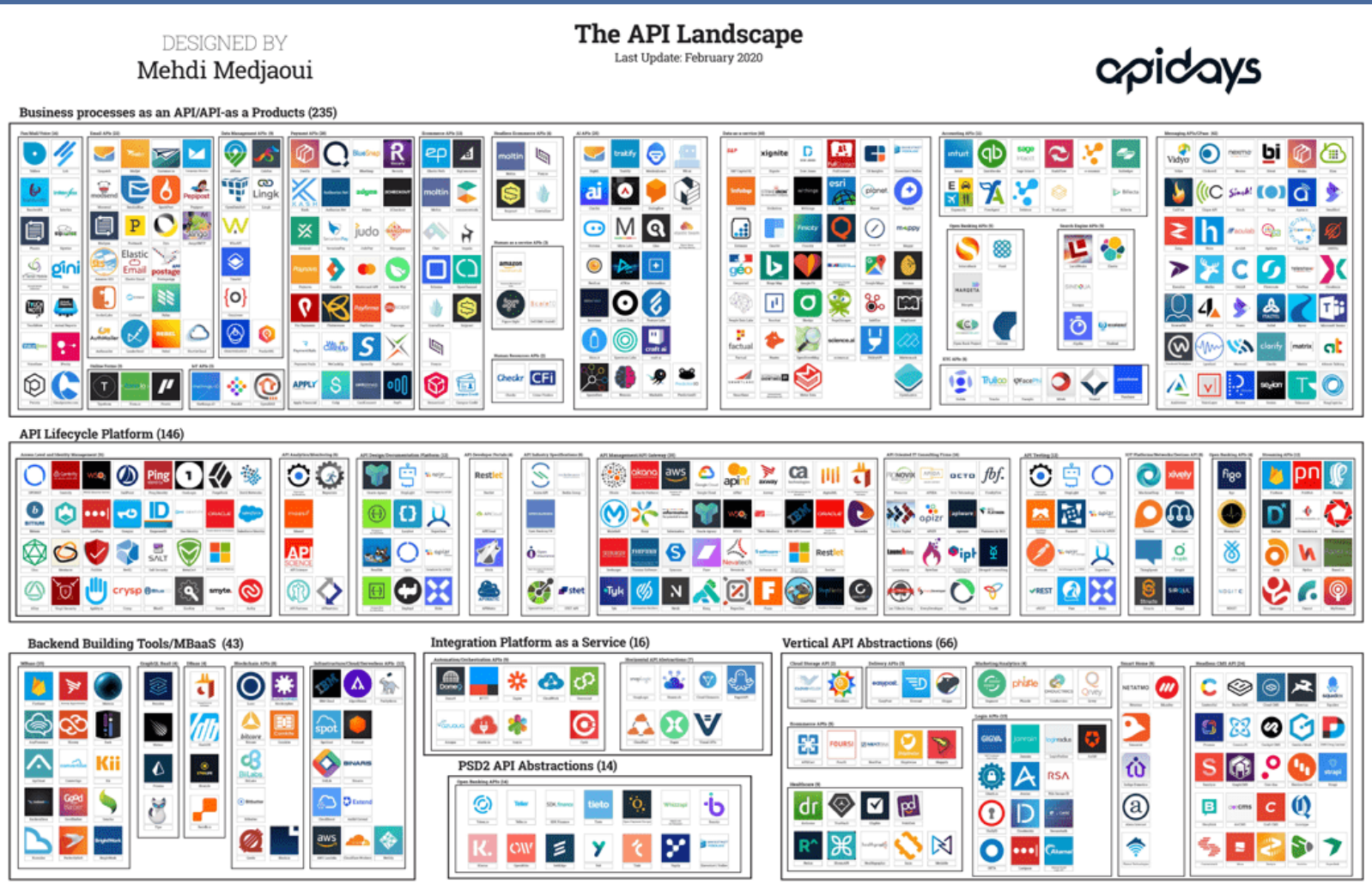
Fonte: API Architecture - The Big Picture for Building APIs, Matthias Biehl

PUC Minas
Virtual

Ferramentas de APIs

Marco Mendes

API Landscape



PUC Minas
Virtual

Táticas de APIs

Marco Mendes

Táticas de APIs

- Mecanismos pragmáticos para a construção e consumo de APIs.
  - REST/HTTP
  - RPC/gRPC
  - WebSocket
  - GraphQL

PUC Minas
Virtual

APIs Baseadas em REST

Marco Mendes

O paradigma REST

A Transferência de Estado Representacional (REST) é a escolha mais popular para o desenvolvimento da API nos últimos anos.

REST

- O paradigma REST é baseado no conceito central de recursos
- Um recurso é uma entidade que pode ser identificada, nomeada, endereçada ou tratada na web.
- AS APIs REST expõem dados como recursos e usam métodos HTTP padrão para representar transações Criar, Ler, Atualizar e Excluir (CRUD) contra esses recursos.
- Por exemplo, a API da Stripe representa clientes, encargos, saldos, reembolsos, eventos, arquivos e pagamentos como recursos.

Exemplo na API do Stripe

Requisição HTTP para recuperar uma cobrança da API Stripe

```
GET
/v1/charges/ch_CWyutlXs9pZyfD
HOST api.stripe.com
Authorization: Bearer
YNoJ1Yq64iCBhzfL9HNO00fzVrsEjtV
l
```

Exemplo na API do Stripe

Requisição HTTP para enviar uma cobrança na API Stripe

```
POST /v1/charges
HOST api.stripe.com
Content-Type: application/x-www-form-
urlencoded Authorization: Bearer
YNoJ1Yq64iCBhzfL9HNO00fzVrsEjtVl

amount=2000&currency=usd
```

Regras comuns REST – Uso de métodos HTTP

- Métodos HTTP como GET, POST, UPDATE e DELETE informam o servidor sobre a ação a ser realizada.
- Diferentes métodos HTTP invocados na mesma URL fornecem funcionalidades diferentes
- **Criar**: Use o POST, na maioria dos casos, para criar novos recursos.
- **Ler** : Use o GET para ler recursos. Requisições GET nunca mudam o estado do recurso. Elas não têm efeitos colaterais. O GET é idempotente.

Regras comuns REST – Uso de métodos HTTP

-
- **Atualização**: Use o PUT para substituir um recurso e PATCH para atualizações parciais para os recursos existentes. O PUT também é idempotente.
- **Excluir**: Use o DELETE para excluir recursos existentes. O DELETE também é idempotente.

Regras comuns REST – Nomenclatura

- Os recursos fazem parte de URLs, como **/usuarios**.
- Para cada recurso, duas URLs são geralmente implementadas: uma para a coleção, como **/usuarios**, e uma para um elemento específico, como **/usuarios/U123**.
- Nomes são usados em vez de verbos para definir recursos. Por exemplo, em vez de **/lerInformacoesUsuario/U123**, use **/usuarios/U123**.

Regras comuns REST – Códigos de Resposta

- Os códigos de status de resposta HTTP padrão são devolvidos pelo servidor indicando sucesso ou falha.
- Geralmente, códigos na faixa 2XX indicam sucesso, os códigos 3XX indicam que um recurso se moveu, e códigos na faixa 4XX indicam um erro do lado do cliente (como um parâmetro necessário ausente ou muitas solicitações).
- Códigos na faixa 5XX indicam erros do lado do servidor.
- Para saber mais: https://developer.mozilla.org/pt-BR/docs/Web/HTTP

Regras comuns REST – Formatos de resposta

- AS APIs REST podem devolver respostas JSON ou XML. E devido à sua simplicidade e facilidade de uso com JavaScript, o JSON tornou-se o padrão para APIs modernas.
- O XML e outros formatos ainda podem ser suportados para facilitar a adoção para clientes que já estão trabalhando com esses formatos usando APIs semelhantes.
- No Brasil, o XML é comum em APIs do governo para facilitar a governança de dados. (exemplos: NF-E ou E-Social).

Convenções para um CRUD com o REST

Operação	Verbo HTTP	URL: /usuarios	URL: /usuários/123
Criar	POST	Cria um novo usuário	Não é aplicável
Ler	GET	Lista todos os usuários	Recupera o usuário 123
Atualizar	PUT ou PATCH	Atualiza em lote os usuários	Atualiza o usuário 123
Remover	DELETE	Apaga todos os usuários	Apaga o usuário 123

Manipulação de relacionamentos com REST

Verbo HTTP	URI Exemplo	Semântica
POST	`/livros/123/autores`	Criar os autores para o livro 123
GET	`/livros/123/autores/1`	Recupera o autor 1 do livro 123
PUT ou PATCH	`/livros/123/autores/1`	Atualiza os dados do autor 1 do livro 123
DELETE	`/livros/123/autores`	Apaga todos os autores do livro 123

Regras comuns REST para operações não CRUD

Além das operações típicas da CRUD que acabamos de olhar, as APIs REST às vezes precisam representar operações não-CRUD. As seguintes abordagens são comumente usadas nesse caso:

1. Crie uma ação como um subrecurso.

Exemplo: **GET livros/empromocao**

2. Crie uma ação através de parâmetros de entrada

Exemplo: **GET livros?tipo=empromocao&categoria=autoajuda**

REST se tornou o paradigma mais popular para criar APIs nos últimos anos.
Ao implementar APIs REST, considere as melhores práticas e guias de desenho.

PUC Minas
Virtual

RPC e gRPC

Marco Mendes

APIs do Tipo RPC

A Chamada de Procedimento Remoto (RPC) é um dos paradigmas mais simples de API, em que um cliente executa um bloco de código em outro servidor.

Enquanto o REST é sobre recursos, a RPC é sobre ações. Os clientes tipificam um nome e argumentos de método para um servidor e recebem de volta JSON ou XML.

Exemplo da API Slack

```
POST
/api/conversations.archive
HOST slack.com
Content-Type: application/x-
www-form-urlencoded
Authorization:
Bearer xoxp-1650112-jgc2asDae

channel=C01234
```

API de Conversação do Slack

conversations.archive	Archives a conversation.
conversations.close	Closes a direct message or multi-person direct message.
conversations.create	Initiates a public or private channel-based conversation
conversations.history	Fetches a conversation's history of messages and events.
conversations.info	Retrieve information about a conversation.
conversations.invite	Invites users to a channel.
conversations.join	Joins an existing conversation.
conversations.kick	Removes a user from a conversation.
conversations.leave	Leaves a conversation.
conversations.list	Lists all channels in a Slack team.
conversations.members	Retrieve members of a conversation.
conversations.open	Opens or resumes a direct message or multi-person direct message.

Protocolos de APIs em RPC

As APIs estilo RPC não são exclusivas do HTTP.

Existem outros protocolos de alto desempenho que estão disponíveis para APIs estilo RPC, incluindo o Apache Thrift e Google gRPC.

https://thrift.apache.org
https://grpc.io

Vantagens de protocolos RPC

- APIs baseadas em verbos facilitam a organização de contratos extensos (dezenas ou centenas de funcionalidades).
- Fornecem facilidade e extensibilidade para produção de APIs.

Vantagens de protocolos gRPC/Thrift

- APIs específicas RPC possuem performance melhor pois operam obrigatoriamente sobre HTTP/2 e usam protocolos de dados como ProtoBuffer.
- Comparativo:
  - **https://github.com/david-cao/gRPCBenchmarks**

Desvantagens de protocolos gRPC/Thrift

- Curvas de aprendizado de protocolo de dados específicos.
- Curvas de aprendizado de protocolos de transporte específicos como o gRPC.

PUC Minas
Virtual

GraphQL

Marco Mendes

GraphQL

- É uma linguagem de consulta para APIs que ganhou tração significativa recentemente.
- Ela foi internamente pelo Facebook em 2012 e publicamente em 2015 e foi adotada desde então por provedores de API como GitHub, Yelp e Pinterest, entre outros.
- O GraphQL permite que os clientes definam a estrutura dos dados necessários, e o servidor retorna exatamente essa estrutura.
- Referência primária: https://graphql.org

Requisições e respostas dinâmicas

```
query {
  contas {
    nome,
  }
}
```

```
{
    "data": {
        "contas": [
            {
                "nome": "Fulano"
            },
            {
                "nome": "Ciclano"
            }
        ]
    }
}
```

Requisições e respostas dinâmicas

```
query {
  contas {
    nome,
    cpf
  }
}
```

```
{
  "data": {
    "contas": [
      {
        "nome": "Fulano",
        "cpf": "123.456.789-10"
      },
      {
        "nome": "Ciclano",
        "cpf": "987.654.321-10"
      }
    ]
  }
}
```

Vantagens

- AS APIs REST e RPC muitas vezes acabam respondendo com dados que clientes podem nunca usar.
- Com o GraphQL, como os clientes podem especificar exatamente o que precisam os tamanhos dos pacotes podem ser menores.
- As consultas da GraphQL retornam resultados previsíveis, dando aos clientes controle sobre os dados que são devolvidos.

GraphQL

- O servidor define um esquema de dados com tipos aninhados.
- O servidor cria um código que interpreta as requisições que enviam pacotes de dados baseados no esquema de dados, e retorna ou altera as coleções de dados necessárias.

Vantagens

- O GraphQL permite que os clientes aninhem consultas e busquem dados de recursos recursos em uma única solicitação. Sem o GraphQL, isso pode exigir várias chamadas HTTP para o servidor.
- Isso significa que aplicativos móveis usando GraphQL podem ser rápidos, mesmo em conexões de trabalho de rede lenta.

Vantagens

- Você pode adicionar novos campos e tipos a uma API GraphQL sem afetar as consultas existentes. Da mesma forma, depreciar os campos existentes é mais fácil.
- Ao fazer análise de log, um provedor de API pode descobrir quais clientes estão usando um campo. Você pode esconder campos preteridos de ferramentas e removê-los quando nenhum cliente os estiver usando.
- Com AS APIs REST e RPC, é mais difícil descobrir quais clientes estão usando um campo preterido, dificultando a remoção.

Desvantagens

- Projeto do servidor GraphQL pode ser complexo para esquemas de dados muitos complexos.
- Custo do processamento de operações complexas de atualizações (Mutations) pode ser um problema na escalabilidade do servidor.

Desvantagens

- Projeto do servidor GraphQL pode ser complexo para esquemas de dados muitos complexos.
- Custo do processamento de operações complexas de atualizações (Mutations) pode ser um problema na escalabilidade do servidor.

PUC Minas
Virtual

Web Sockets

Marco Mendes

Web Sockets

- O WebSocket é um protocolo usado para estabelecer um canal de comunicação de streaming bidirecional sobre uma única conexão de Protocolo de Controle de Transporte (TCP).
- Embora o protocolo seja geralmente usado entre um cliente web (por exemplo, um navegador) e um servidor, às vezes é usado também para comunicação servidor-servidor.
- Fonte: https://developer.mozilla.org/pt-BR/docs/WebSockets

Vantagens

- Emula canais bidirecionais de comunicação para facilitar conversações.
- Aplicações que demandam atualizações de eventos com frequência fazem uso desse tipo de modelo.
- Exemplos:
  - Slack
  - Trelllo
  - Blockchain

Vantagens

- WebSockets podem habilitar a comunicação *full-duplex* (servidor e cliente podem se comunicar simultaneamente) com baixo custo.
- Além disso, eles são projetados para trabalhar sobre a porta 80 ou 443, permitindo-lhes trabalhar bem com firewalls que podem bloquear outras portas.

Vantagens

- WebSockets são ótimos para dados rápidos e para transmissão ao vivo se você tiver conexões de longa duração.
- Tenha cuidado se você planeja disponibilizá-los em dispositivos móveis ou em regiões onde a conectividade pode ser irregular. Isso porque os clientes devem manter a conexão viva. Se a conexão morrer, o cliente irá precisar reiniciá-la.