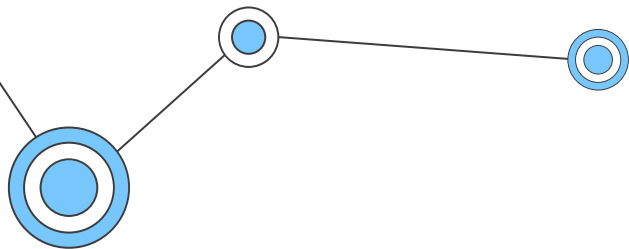


Arquitetura de Software

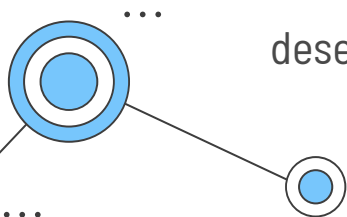
Professor: Ralf Lima

Informações do módulo

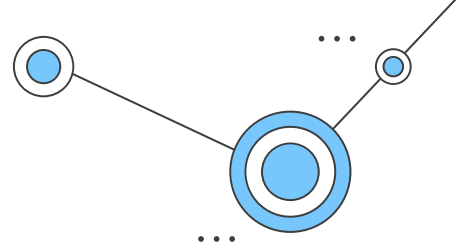
- São seis encontros com carga horária de 4 horas cada, totalizando 24 horas;
- Todas as sextas-feiras: **18 de julho e 01, 08, 15, 22 e 29 de agosto.**
- Nossas aulas terão o seguinte padrão:
 - As quatro primeiras teoria e atividades;
 - A quinta uma apresentação;
 - A sexta uma avaliação com carga horária de quatro horas englobando todos os conceitos abordados no módulo.



- Desenvolvedor desde 2008;
- Instrutor desde 2008;
- Formado em Sistemas para Internet;
- Pós-graduado em Marketing Digital;
- Atualmente tenho uma empresa que atua em dois segmentos, sendo eles: ensino e desenvolvimento de sistemas.



Empresas atendidas

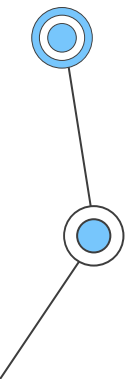
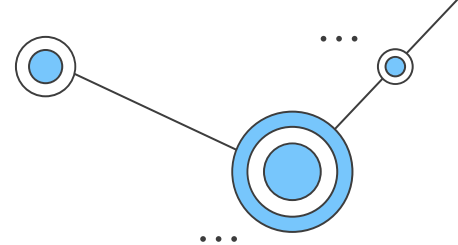


CERVEJARIA

ambev



Treinamentos realizados



Conteúdos abordados

01

API's

Rest, JSON, SOAP, gRPC e GraphQL.

02

Padrões de desenvolvimento

Design Patterns, MVC, DDD, TDD e Hexagonal architecture.

03

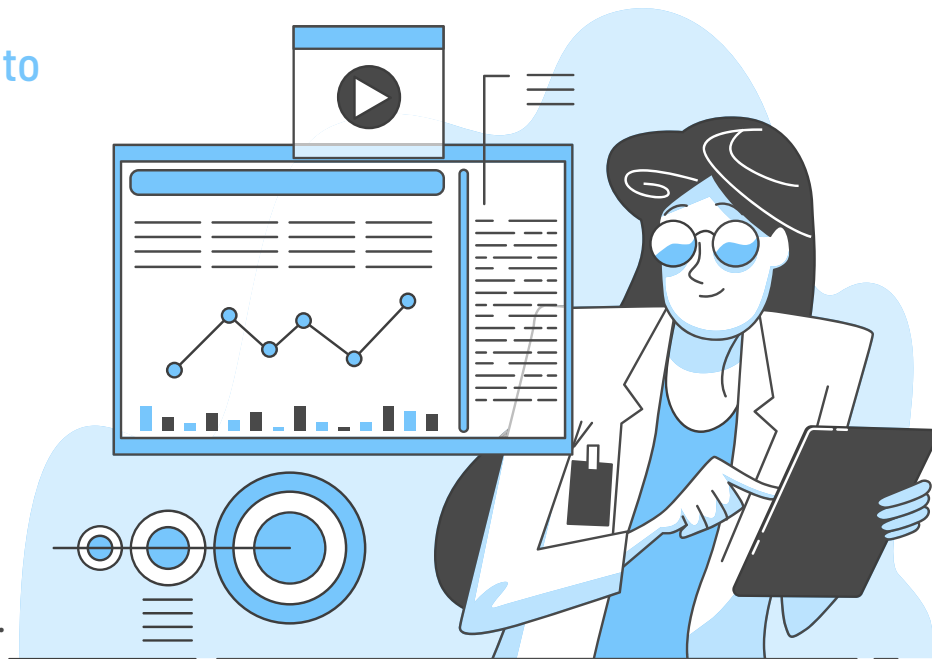
Segurança web

HTTPS, CORS, CSP, JWT e OAuth.

04

Padrões de arquitetura

Monolitos, Microserviços, SOA, Serveless e Service Mesh.



Conteúdos da aula de hoje

- Compreender o que é arquitetura de software;
- Como a arquitetura de software pode contribuir na elaboração de projetos;
- Conceitos básicos de SCRUM;
- Conceito de APIs;
 - REST;
 - SOAP;
- gRPC;
- GraphQL.

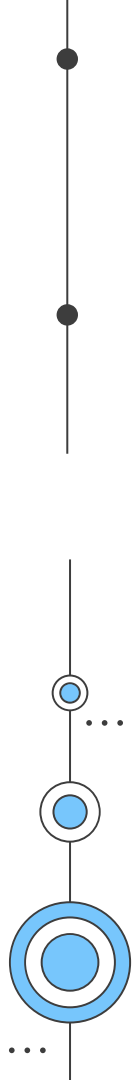
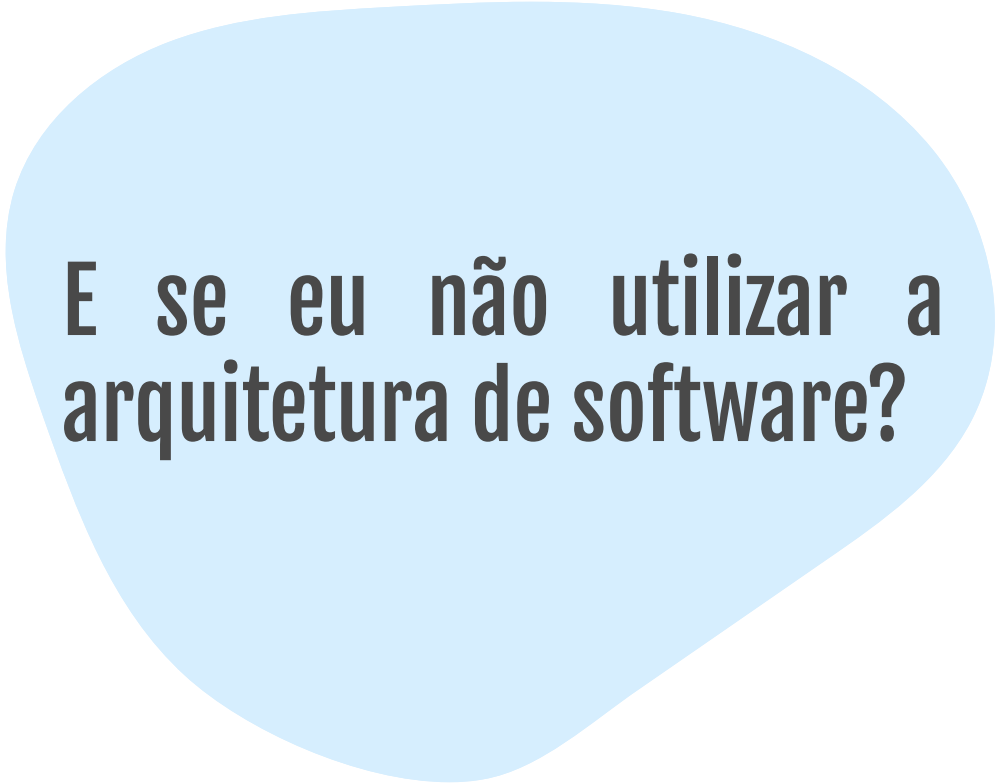

...

O que é Arquitetura de Software?



O que é arquitetura de software?

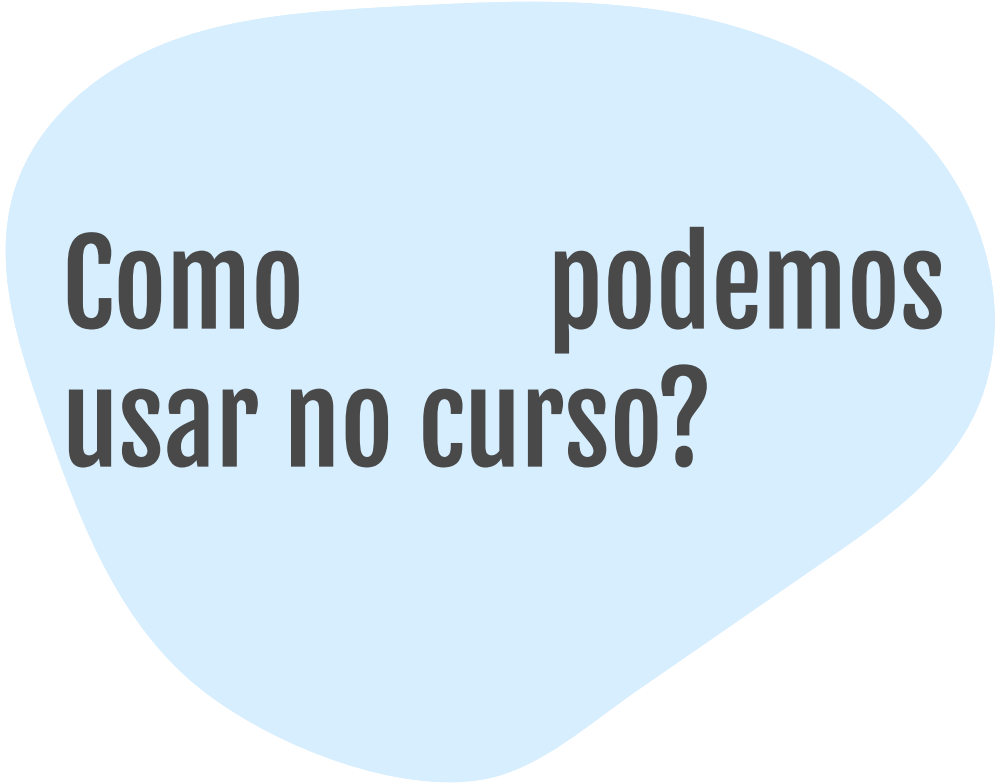

- Saber quais tecnologias utilizar, exemplo: linguagens de programação, frameworks, bancos de dados, tipo de hospedagem...
- Arquitetar (pensar) nos módulos do software e eventuais expansões;
- Gerenciar as regras de negócios;
- Cuidar do cronograma dos integrantes do projeto;
- Acompanhamento contínuo da(s) equipe(s);
- Garantir a documentação dos projetos.



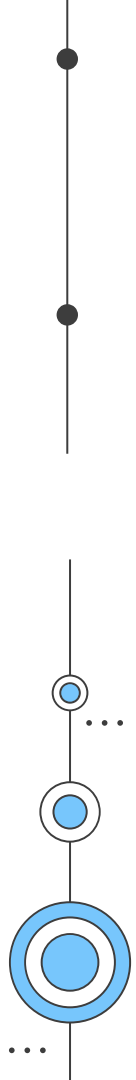
**E se eu não utilizar a
arquitetura de software?**

E se eu não utilizar a Arquitetura de Software?

- Softwares com regras de negócios difíceis de serem compreendidas;
- Prazos de entregas não serem cumpridos;
- Performance comprometida;
- Manutenção complexa, lenta e com dificuldade para encontrar desenvolvedores capacitados para implementar;
- Falta de padronização nos sistemas;
- Desenvolver projetos com falhas nas regras de negócios.



**Como podemos
usar no curso?**



Como podemos usar no curso?



Como podemos usar no curso?

- Saber trabalhar com o GitHub (branches);
- Utilizar softwares para gerenciamento de projetos como o Trello;
- Utilizem o Swagger (OpenAPI) para criar documentações;
- Deixar as tarefas de cada integrante da equipe bem definidas;
- Utilizar a metodologia SCRUM.

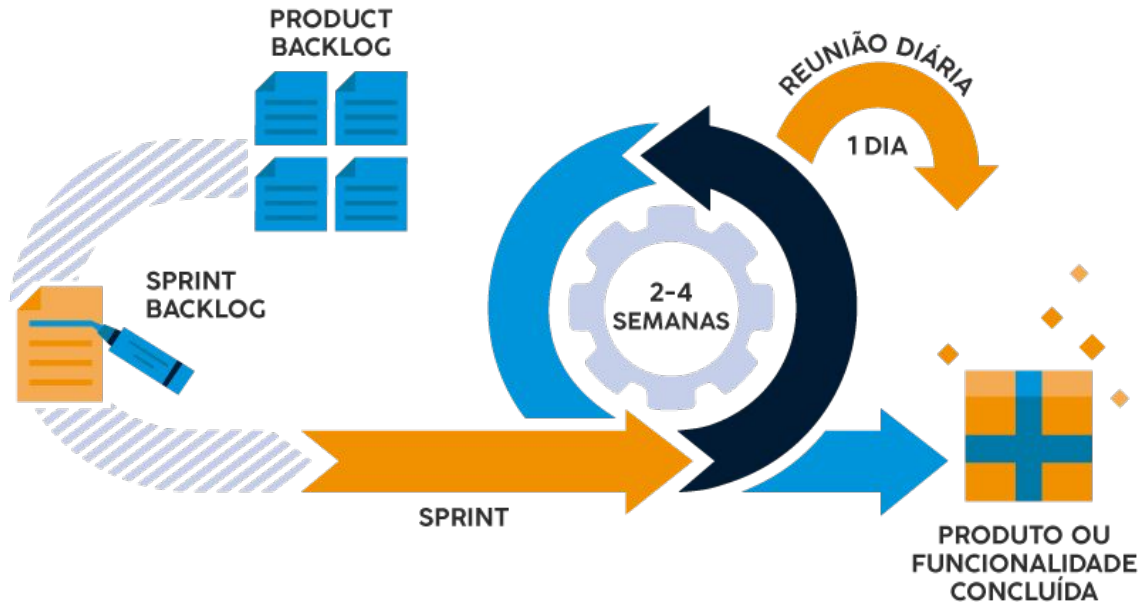
...



SCRUM?

Sobre o SCRUM

Scrum é uma estrutura leve que ajuda pessoas, equipes e organizações a gerar valor por meio de soluções adaptáveis para problemas complexos.



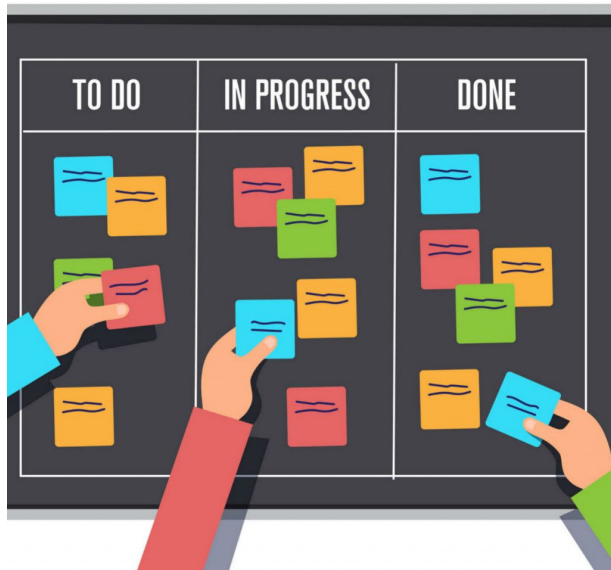
Como gerenciar sprints?

Há diversas ferramentas, veja abaixo as principais:



Exemplo de Sprint

Muitas Sprints utilizam o padrão Kanban, que facilita saber o que está pendente, o que está sendo feito e o que já foi feito.



Alguns termos do SCRUM

- Product Owner: 0 cliente;
- Scrum Master: Responsável por gerenciar toda a metodologia ágil do projeto;
- Scrum Team: Equipe.

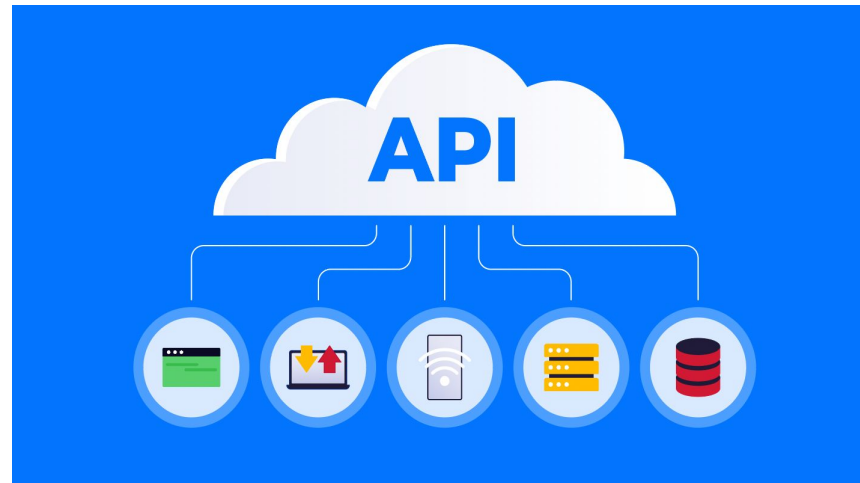
...



API's

API's

API ou Application Programming Interface, são sistemas que podem ser acessados por aplicações front-end e back-end.



API's REST

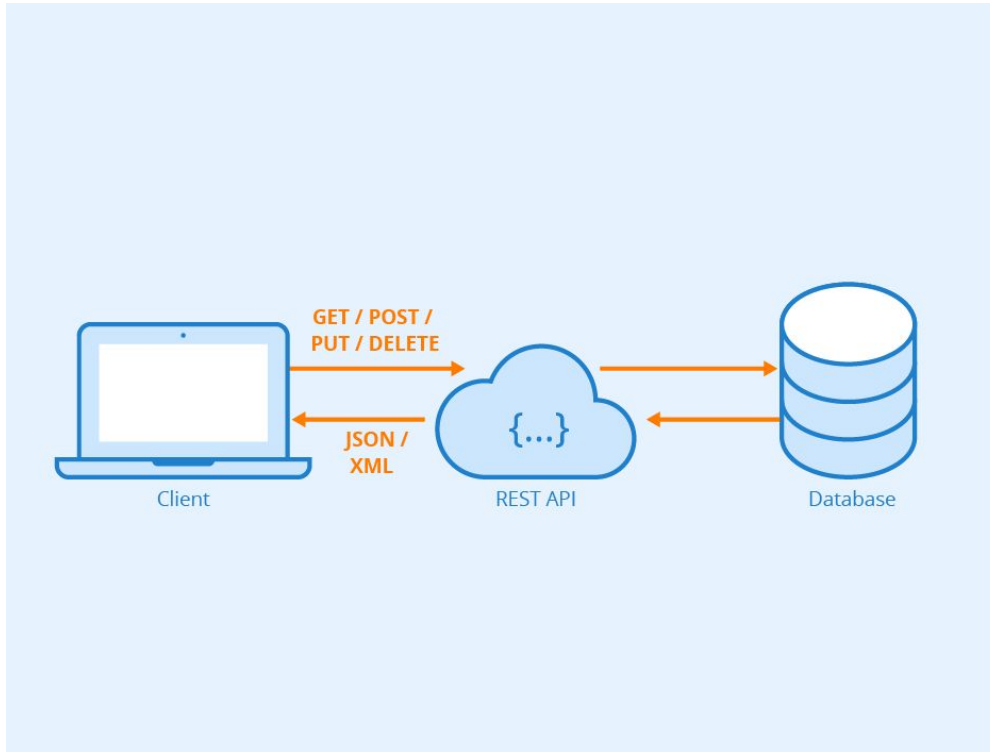
API's REST

API REST (Representational State Transfer), é uma arquitetura de software que dispõe de requisições HTTP para realizar diversos tipos de funcionalidades, como:

- Cadastramentos (**POST**);
- Seleções (**GET**);
- Alterações (**PUT ou PATCH**);
- Exclusões (**DELETE**).

...

API's REST



REST vs RESTful

- REST: Trabalha com o padrão HTTP para gerenciar requisições;
- RESTful: Utiliza como base o REST, especifica endpoints e retornos, exemplos:
 - <http://www.ralflima.com/api/clientes/cadastrar>
 - Poderá retornar um status 201 - CREATED;
 - Poderá retornar um status 400 - BAD REQUEST;
 - <http://www.ralflima.com/api/cursos/remover?codigo=5>
 - Poderá retornar um status 200 - OK;
 - Poderá retornar um status 404 - NOT FOUND.

Tecnologias para criar API's REST

- **C#** - Utilizando o ASP.NET Core, módulo web API;
- **Java/Kotlin** - Spring;
- **Python** - Django e Flask;
- **PHP** - Laravel, Slim e CakePHP;
- **NodeJS** - ExpressJS. ...

Tecnologias para consumir API's REST

- **JavaScript** ou **TypeScript;**
- **Angular;**
- **React;**
- **VueJS.**

...

Vantagens

- Uma API pode ser consumida por outras API's;
- Uma API pode ser consumida em qualquer tipo de plataforma, desde que tenha acesso a internet;
- API's possuem boa performance;
- API's podem trocar informações, independente a linguagem ou framework desenvolvidos, pois os dados transitados são JSON, Texto, Requisição de Formulário, XML, entre outros...

Desvantagens

- Precisa de conhecimento mais aprofundado de uma linguagem de programação ou framework, além dos conceitos de REST;
- Maior possibilidade de falhas na segurança;
- Performance comprometida, caso o desenvolvedor não crie uma boa regra de negócios;
- Se houver falha em uma requisição, todo o sistema pode parar de funcionar;
- Trabalha apenas com o padrão HTTP.

Exemplos práticos de API Rest

```
<defs>
  <linearGradient x1="100%" y1="0%" x2="0%" y2="100%" id="media-control">
    <stop stop-color="#06101F" offset="0%"/>
    <stop stop-color="#1D304B" offset="100%"/>
  </linearGradient>
</defs>
<rect width="800" height="450" rx="8" fill="url(#media-control)" fill-rule="evenodd">
</svg>
<div class="media-control">
  <svg width="96" height="96" viewBox="0 0 96 96" xmlns="http://www.w3.org/2000/svg">
    <defs>
      <linearGradient x1="87.565%" y1="15.875%" x2="37.565%" y2="84.125%" id="media-control">
        <stop stop-color="#FFF" stop-opacity="0.5" offset="0%"/>
        <stop stop-color="#FFF" offset="100%"/>
      </linearGradient>
      <filter x="-500%" y="-500%" width="1000%" height="1000%" filterUnits="userSpaceOnUse">
        <feOffset dy="16" in="SourceAlpha" result="shadowOffsetOuter"></feOffset>
        <feGaussianBlur stdDeviation="24" in="shadowOffsetOuter" result="shadowBlurOuter"></feGaussianBlur>
        <feColorMatrix values="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0" type="matrix" in="shadowBlurOuter" result="shadowMatrixOuter"></feColorMatrix>
      </filter>
    </defs>
    <rect width="96" height="96" rx="16" fill="url(#media-control)" fill-rule="evenodd">
    </rect>
  </svg>
</div>
```



API's SOAP



API's SOAP

API's SOAP (Simple Object Access Protocol), é um tipo mais antigo de API, utilizando como base o XML para enviar e receber dados.

O foco em API's SOAP é no envio de mensagens, em especial e-mails, devido seu alto nível de segurança e estabilidade.

...

Vantagens

- SOAP oferece protocolos de segurança incorporados, tais como WS;
- As APIs oferecem opções incorporadas para o tratamento de erros;
- Trabalha com diversos tipos de protocolos, como: HTTP, HTTPS, TCP, SMTP, e XMPP
- SOAP suporta a automatização quando utilizado com uma linguagem de programação....

Desvantagem

- Menos performático que uma API REST;
- Trabalha com padrão XML;
- Curva de aprendizagem elevada;
- SOAP não dispõe de todas as requisições HTTP que o REST;
- Necessita de uma estrutura de hardware mais avantajada.

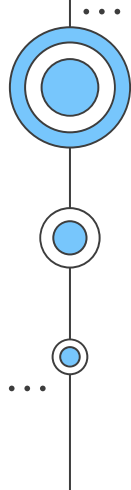


REST e SOAP

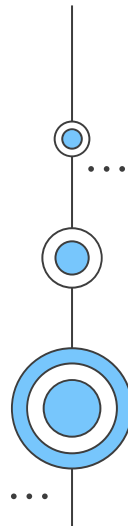


Diferenças entre REST e SOAP

- SOAP API funciona com SOAP (Simple Object Access Protocol), enquanto o REST API funciona com o protocolo REST (Representational State Transfer).
- SOAP API transmite mensagens em formato XML padrão devido ao seu estado, enquanto que o REST API não segue um formato de dados devido à sua ausência de estado.
- SOAP API é mais estruturada, enquanto que REST utiliza dados em forma volumosa.
- SOAP suporta o formato de dados XML, enquanto a API REST suporta texto simples, XML, HTML, JSON, etc.



gRPC



gRPC

Você já ouviu falar em HTTP? Bem, a ideia é termos a transferência de dados entre um servidor e cliente.



gRPC

API's REST e SOAP não utilizam o padrão gRPC, que é o HTTP 2, fazendo com que as requisições sejam menos performáticas.

Antes da criação do HTTP, em 1976 com o avanço dos computadores, diversos estudiosos analisavam como dados poderiam ser compartilhados entre os computadores, sendo assim foi criado o termo RPC (Remote Procedure Call).

O HTTP nas versões 1.X utilizam essa arquitetura até hoje, mas em 2015 a Google criou o gRPC, onde utiliza o HTTP 2.0, garantindo mais velocidade na transição de dados.

...

gRPC

Você pode criar API's REST tranquilamente, mas em alguns projetos a velocidade na requisição das API's pode ser lento, neste caso, o uso o gRPC pode ser interessante.

As ações do gRPC são as mesmas que uma API, podendo ser efetuados cadastros, seleções, alterações ou exclusões, porém lembre-se:

- API's REST ou SOAP utilizam HTTP 1;
- gRPC utilizam HTTP2.

...

Vantagens

- Extremamente rápido para realizar requisições;
- Focado em arquiteturas de microsserviços;
- Recursos disponíveis em várias linguagens (C#, PHP, Java, Python...);
- Mais segurança, pois o gRPC trabalha com informações criptografadas;
- Totalmente gratuito para utilizar nos projetos.

Desvantagens

- Poucas referências sobre a tecnologia;
- Suporte limitado a alguns navegadores e plataformas (web e Android);
- Atualmente é possível trabalhar com 10 linguagens, sendo elas: C#, C++, Dart, Go, Java, Kotlin, Node, Object-C, PHP, Python e Ruby;
- Estrutura de código mais complexa que API's em REST.

...

Navegadores que suportam gRPC

Suporte dos Navegadores



...



GraphQL

GraphQL

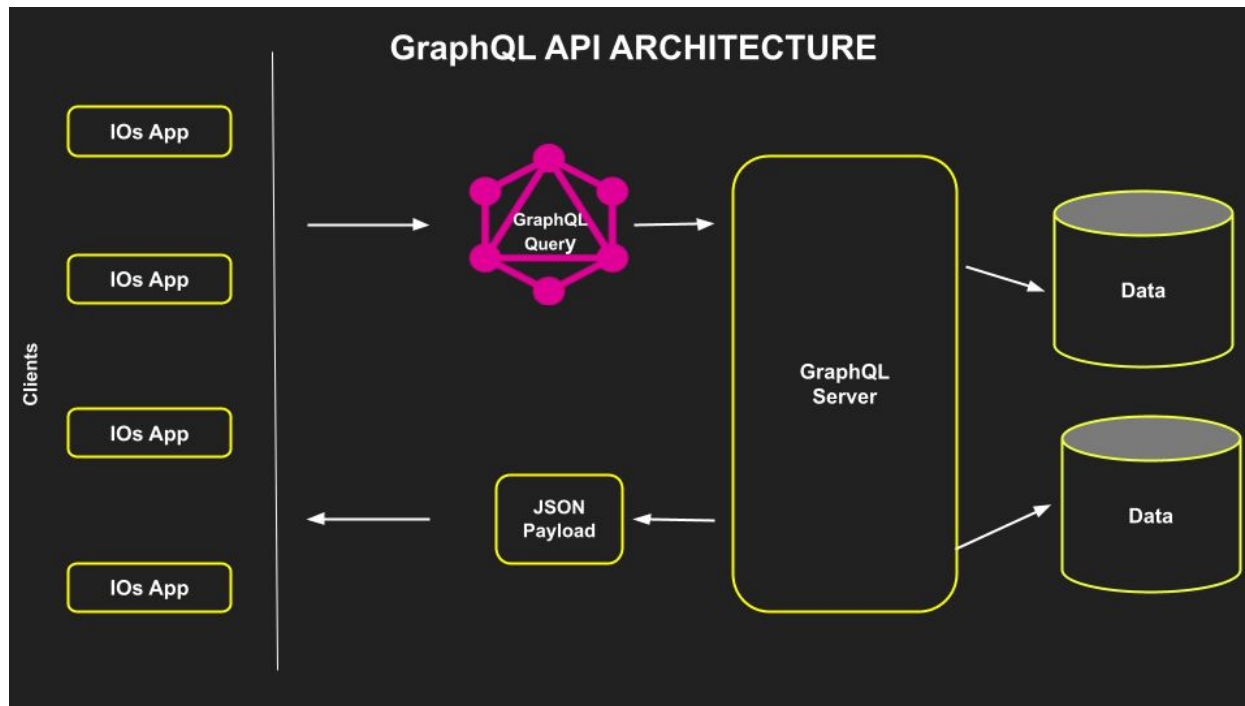
Criado em 2015 pelo Facebook, o GraphQL é uma estrutura de gerenciamento de dados, ela não utiliza o padrão REST.

O Facebook durante anos utilizou APIs REST, mas notou que realizava muitas requisições, fazendo com que seus servidores tivessem sobrecargas consideradas desnecessárias.

A ideia do GraphQL é simples, realizar em uma única requisição o retorno dos dados que o cliente precisa.

...

GraphQL



Vantagens

- Realiza o retorno dos dados em apenas um endpoint;
- Velocidade superior ao REST para retornar dados;
- Tecnologia gratuita;
- Foco em arquitetura de microsserviços;
- Documentação robusta. ...

Desvantagens

- Curva de aprendizagem;
- Não é possível realizar requisições contendo arquivos (jpg, png, pdf, doc, xls...);
- Ideal para trabalhar com grandes conjuntos de dados, em sistemas pequenos não haverá mudança relevante na performance das consultas;
- Ainda não está difundido nos projetos.

Atividades

Para garantir que cada um de vocês compreendeu os conceitos abordados na primeira aula de arquitetura de software, haverá alguns exercícios.

Os exercícios devem ser enviados para o e-mail: ralflima@gmail.com até dia 31/07.

Dica importante! Nossa última aula será no dia 29/08, neste dia teremos diversas atividades avaliativas, algumas questões terão como base as atividades disponibilizadas na aula de hoje.

Atividades

1. Em suas palavras, descreva o que faz um arquiteto de software e quais as vantagens em utilizar a arquitetura de software em projetos?
 2. Descreva o que é SCRUM e quais são as etapas.
 3. Utilizando o Trello, crie um projeto (tema livre) contendo as três etapas do Kanban e desenvolva um texto explicando seu projeto, tempo de desenvolvimento e como o processo foi dividido.
- ...
- Enviar o texto e uma imagem do Trello mostrando a divisão da Sprint.

Atividades

4. Explique o que é uma API e como é baseada sua estrutura?
5. O que é um endpoint?
6. Explique o que é REST e RESTful.
7. O que é uma API SOAP?
8. Quais as diferenças entre uma API REST e uma API SOAP?

Atividades

9. O que é gRPC? Quais suas vantagens e desvantagens?
10. Explique o que é GraphQL e exemplifique seu uso em algum projeto.
11. Explique quais as funcionalidades de cada uma das requisições:
 - a. POST;
 - b. GET;
 - c. PUT;
 - d. PATCH;
 - e. DELETE.

Atividades

12. Uma API pode ter diversos retornos, explique quando cada um dos retornos deve ser utilizado:
- a. 200 - OK
 - b. 201 - CREATED
 - c. 202 - ACCEPTED
 - d. 400 - BAD REQUEST
 - e. 401 - UNAUTHORIZED
 - f. 404 - NOT FOUND
 - g. 429 - TOO MANY REQUESTS

Referências bibliográficas

- https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/software_architecture_4269A354.html
- <https://www.machado.mg.gov.br/files/concursos/1cf11cf161fe4eb688dfec880d6b4d9.pdf>
- <https://www.atlassian.com/br/agile/scrum>
- <http://www.desenvolvimentoagil.com.br/scrum/>
- <https://blog.contaazul.com/metodologia-scrum>
- <https://www.hostinger.com.br/tutoriais/api-restful>
- <https://www.iugu.com/blog/api-rest-o-que-e>
- <https://www.freecodecamp.org/portuguese/news/tutorial-de-apis-rest-client-rest-servicos-rest-e-chamadas-de-api-explicados-com-exemplos-de-codigo/>
- <https://4success.com.br/api-rest-e-restful/>
- <https://www.redhat.com/pt-br/topics/integration/whats-the-difference-between-soap-rest>
- <https://aws.amazon.com/pt/compare/the-difference-between-soap-rest/>
- <https://blog.tecnospeed.com.br/rest-x-soap/>

Referências bibliográficas

- <https://grpc.io/>
- <https://blog.lsanatos.dev/guia-grpc-1/>
- <https://www.zup.com.br/blog/grpc-o-que-e-beagle>
- <https://graphql.org/>
- <https://www.redhat.com/pt-br/topics/api/what-is-graphql>
- https://medium.com/@emerson_pereira/introdu%C3%A7%C3%A3o-a-graphql-cbd1697784bf
- <https://rockcontent.com/br/blog/http/>
- <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>