**APIs**

- É um acrônimo para Interface de Programação de Aplicações (Application Programming Interface), não tem telas propriamente ditas, o front-ed está sendo separado do back-end, é esperado a resposta da interface com base no que solicitamos;

- Uma forma simples de comunicação entre aplicações; não necessariamente o software se conecta a apenas uma API, pode ser conectado a várias;

- Não necessita de telas, respostas geralmente são em JSON;

Ex.: É enviado um texto que contém um usuário (email, nome e senha) e API dá a resposta do usuário cadastrado com sucesso.

- Independente do front-end, a API não possui ligação com o front (isso é uma regra, seguindo a padronização da API em qualquer ambiente);

Ex.: - quero adicionar produto

- Api responde com a cor do botão(btn-color:red)

- Isso não faria sentido **em uma loja**

- A **api só trataria de adicionar o produto**

- Baseada em requisições e resposta;

- Pode ser criada uma API com o Express.

**O que é REST e RESTful**

- **REST** significa Representational State Transfer ou Transferência Representacional de Estado, quando transferimos o estado da nossa aplicação, seja da requisição para a API ou vice-versa, trabalhando só com informações;

**REST** - Padrão que se aplica a uma API

- Estilo de arquitetura que define como as APIs devem ser criadas;

- **Seguindo todos** esses **padrões** a **API é** considerada uma **RESTful**;

- Alguns dos pontos são: respostas uniformes(mensagem de erros e de sucesso padronizadas), sem ligação de estado(front e back sem ligação), cache de respostas e outros.

**SPA**

- Significa Single Page Application;

- Quando a aplicação possui um **front-end separado do back-end**;

- Ou seja, uma **API para back-end**;

- E um framework **front-end JS**, como: React, Vue e Angular;

- Aplicações deste tipo estão dominando o mercado;

- As seções seguintes tem como objetivo trabalhar focadas em SPA, consumir api no front-end;

- A arquitetura anteriormente utilizada é a conhecida como Monolith (front, back-end e banco de dados).

**Verbos HTTP**

- Andam de mãos dadas com as APIs e **fazem parte do REST**;

- Temos métodos como: **GET**(resgatando informação), **POST**(postar, enviar dados), **PUT**(atualização completa de um registro (todos os dados)), **DELETE**(remove, deleta dados), **PATCH**(atualização parcial dos dados) e etc;

- Cada um representa uma ação na API, ou seja, são essenciais;

- É muito importante que as nossas funcionalidades usem métodos corretos, os métodos essenciais para aquela função;

- Enviamos o método através da requisição.

**Criando uma API com Express**

- Para isso, teremos uma tarefa comparada a estrutura com **Handlebars**;

- Basta **instalar o Express** e ele fará tudo sozinho neste ponto;

- Depois será criada uma **rota** **que responde em JSON**, este é o **dado de comunicação** entre aplicação e API;

- É importante a definição do verbo correto, com **GET** ou **POST** por exemplo.

**Terminal -** npm init -y + npm install express nodemon;

**Index.js**

const express = require("express"); //chamando express

const app = express(); //inicializando express

//ler requisição em json

app.use(

  express.urlencoded({

    extended: true,

  })

);

app.use(express.json()); //lendo em json

//rotas - endpoints

app.get("/", (req, res) => {

  res.json({ message: "Primeira rota criada com sucesso" }); //resposta em json

});

app.listen(3000); //executar alguma porta(3000)

**Conhecendo o Postman**

- **Client para testes** de API;

- Podemos criar **o back-end antes ou separado do fron**t só com a ajuda deste software;

- Isso é fundamental quando estamos criando APIs, pois as vezes ficar esperando o front atrapalha o desenvolvimento do back. É ágil;

- Podemos simular verbos (get, post, put), corpo de requisição, inserir headers, tudo que é possível com uma aplicação web.

**Testando rota com Postman**

- Para acessar uma rota com o Postman é preciso **configurar o cliente**;

- Inserir o verbo correto a rota;

- Configurar o endpoint, que é a URL onde a rota foi estabelecida;

- Enviando a requisição, receberemos a resposta.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

**Criando uma rota de POST**

- Utilização do método **POST** do Express;

- Extrair os dados da requisição, acessando **req.body**;

- Como no get é possível **retornar** uma **resposta** como **JSON pela API**.

//rotas - endpoints

app.post('/createproduct', (req, res) => {

  const name = req.body.name;

  const price = req.body.price;

  console.log(name);

  console.log(price);

  res.json({message: `O produto ${name} foi criado com sucesso!`})

});

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

**Adicionando status na resposta**

- Podem ajudar no desenvolvimento da aplicação;

- Exibindo mensagens de sucesso ou erro por exemplo.

- Entre o res e o método json define-se um **número de status HTTP**;

- Isso é feito por meio do método status.

**Postman:**

![Interface gráfica do usuário

Descrição gerada automaticamente com confiança média]() ![Interface gráfica do usuário

Descrição gerada automaticamente com confiança média]()



 res.status(200).json({ message:””})

**Finalizando seção sobre APIs**

- API é **desenvolvida de forma** muito **semelhante** **a** quando criamos **projetos com Handlebars**, por exemplo;

- Temos alterações leves, como: respostas apenas em JSON;

- Isso acaba simplificando as coisas, separando as responsabilidades;