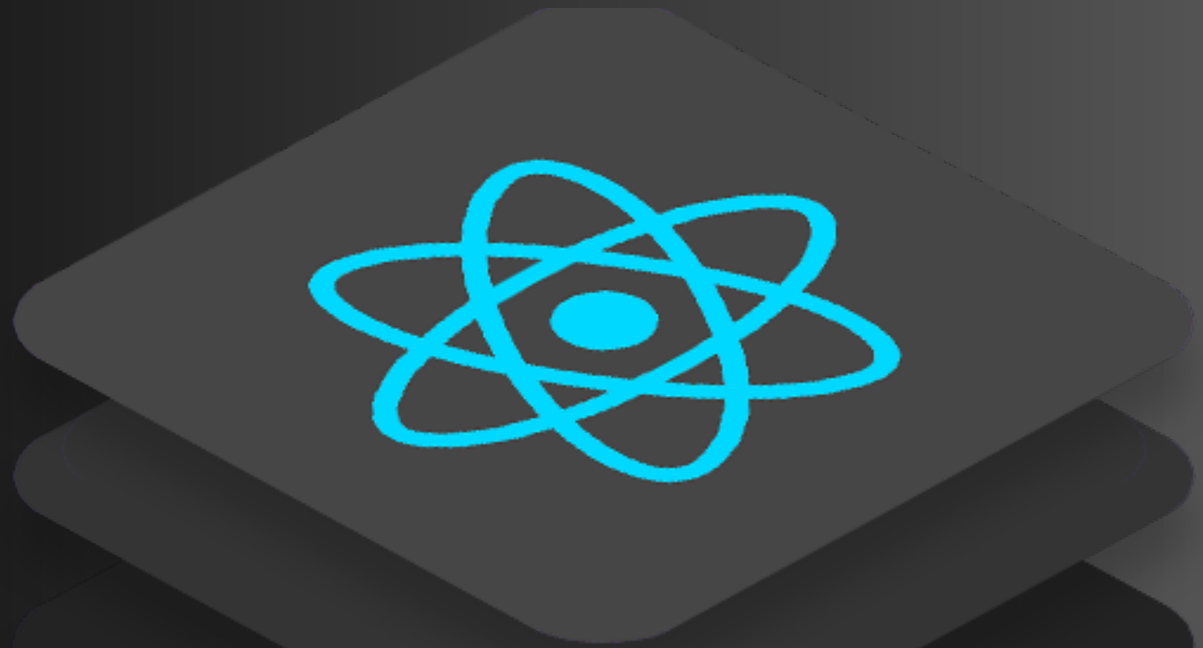


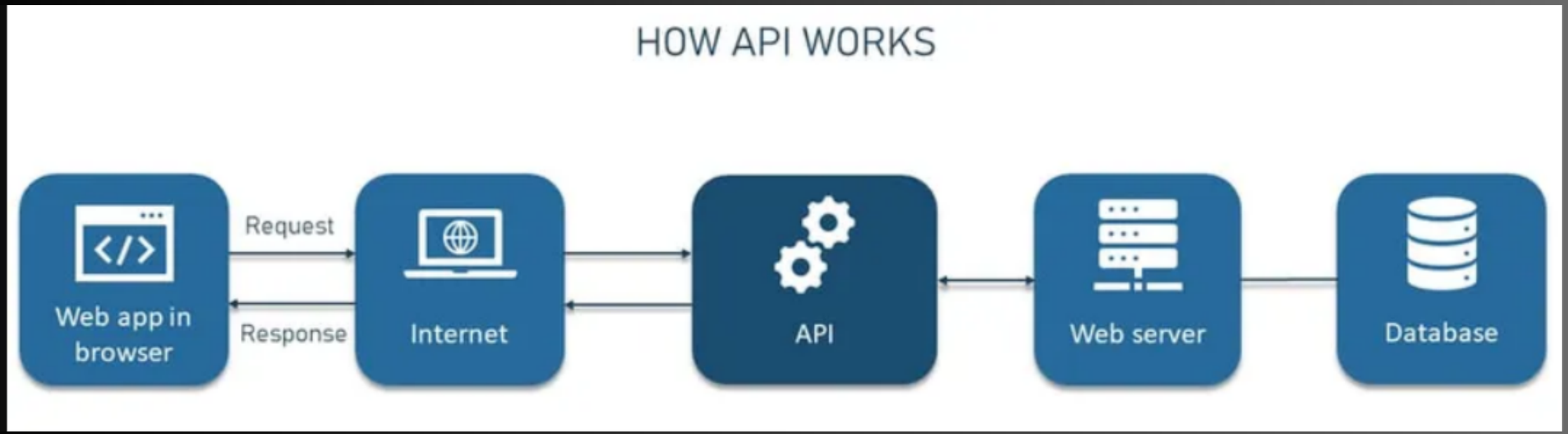
React

Frontend

Conexão ao backend

No React, a comunicação com o banco de dados é sempre feita por meio de uma API.

Tal prática visa separar o acesso ao BD.



O que é uma API?

API significa "Application Programming Interface".
É uma interface que permite que dois programas se comuniquem entre si.

Em outras palavras, uma API é como um tradutor que permite que dois programas que falam línguas diferentes se entendam.

Exemplos de APIs

Alguns exemplos de APIs populares incluem:

- A API do Facebook, que permite que aplicativos de terceiros acessem dados do Facebook, como fotos e amigos
- A API do Google Maps, que permite que aplicativos de terceiros acessem mapas e direções
- A API do Twitter, que permite que aplicativos de terceiros acessem tweets e usuários

HTTP

O protocolo HTTP é um protocolo de comunicação entre clientes e servidores web. Ele é utilizado para transferir dados, como páginas web, imagens, vídeos e dados.

Princípios do HTTP

O HTTP é baseado nos seguintes princípios:

- Cliente-servidor: O cliente (navegador) envia requisições para o servidor, que as responde.
- Request-response: Cada requisição é independente da anterior, e cada resposta é independente da anterior.
- Sem estado: O servidor não armazena informações sobre o cliente entre requisições.
- Texto simples: O HTTP usa texto simples para comunicar dados, o que permite que ele seja usado por qualquer aplicativo.

Métodos HTTP

O HTTP define quatro métodos principais para fazer requisições:

- GET: Obter um recurso.
- POST: Enviar dados para um servidor.
- PUT: Atualizar um recurso.
- DELETE: Remover um recurso.

Cabeçalhos HTTP

Os cabeçalhos HTTP são pares de chave-valor que fornecem informações adicionais sobre as requisições e respostas HTTP. Eles são usados para uma variedade de propósitos, incluindo:

- Indicar o tipo de conteúdo da requisição ou resposta. O cabeçalho Content-Type é usado para indicar o tipo de conteúdo que está sendo enviado ou recebido. Por exemplo, o valor "text/html" indica que o conteúdo é uma página web.
- Fornecer credenciais de login. O cabeçalho Authorization é usado para fornecer credenciais de login para o servidor. Isso é usado para autenticar o cliente e permitir que ele acesse recursos protegidos.
- Definir as configurações de cache para a resposta. O cabeçalho Cache-Control é usado para definir as configurações de cache para a resposta. Isso permite que o cliente armazene a resposta no cache para evitar que ela precise ser baixada novamente.

- **Content-Type:** O cabeçalho Content-Type é um cabeçalho obrigatório para todas as requisições e respostas HTTP. Ele indica o tipo de conteúdo que está sendo enviado ou recebido. O valor do cabeçalho Content-Type é um tipo MIME, que é um tipo de conteúdo definido pelo IANA (Internet Assigned Numbers Authority). Alguns tipos MIME comuns incluem:
  - text/html: Página web
  - image/jpeg: Imagem JPEG
  - application/json: Dados JSON

- **Authorization:** O cabeçalho Authorization é um cabeçalho opcional. Ele é usado para fornecer credenciais de login para o servidor. O valor do cabeçalho Authorization é um token de autenticação que é gerado pelo cliente. O servidor usa o token para autenticar o cliente e permitir que ele acesse recursos protegidos.

- **Cache-Control:** O cabeçalho Cache-Control é um cabeçalho opcional. Ele é usado para definir as configurações de cache para a resposta. O valor do cabeçalho Cache-Control é uma lista de instruções que o cliente deve seguir ao armazenar a resposta no cache. Algumas instruções comuns do cabeçalho Cache-Control incluem:
  - max-age: Especifica o tempo máximo que a resposta pode ser armazenada no cache.
  - no-cache: Indica que a resposta não deve ser armazenada no cache.
  - must-revalidate: Indica que a resposta deve ser atualizada sempre que o cliente acessar o recurso.

Fetch API - Sintaxe

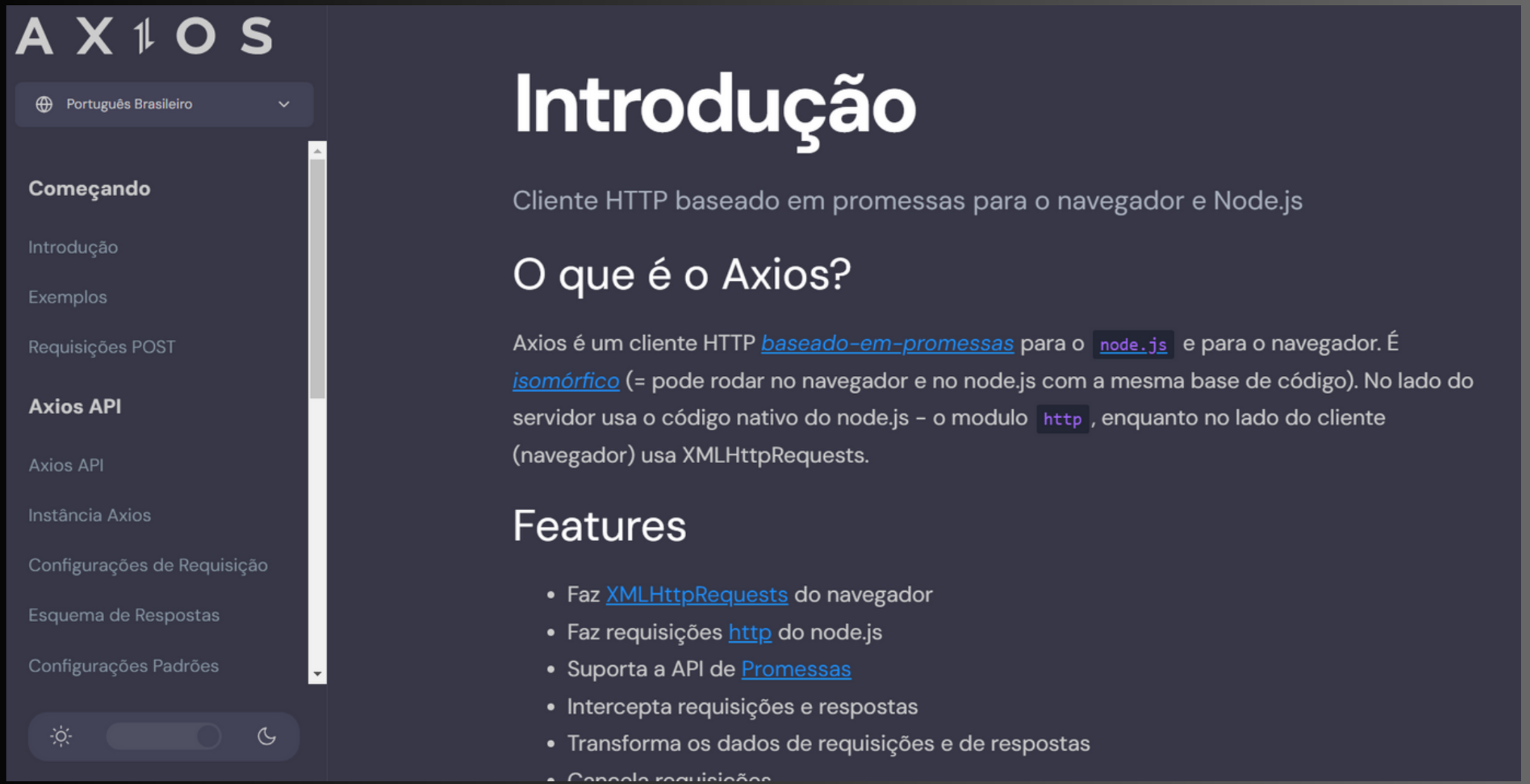
```
let data = {username:"fulano", phone:"8598888-8888"}
fetch("gateway", {
  method: 'POST',
headers: {
'Content-Type': 'application/json',
},
body: JSON.stringify(data),
}
).then((response)=>{
  response.json().then((result)=>console.log(result))
})
```

Fetch API - Sintaxe

```
let data = {username:"fulano", phone:"8598888-8888"}
// isso tem que estar de uma função async
let response = await fetch("gateway", {
  method: 'POST',
headers: {
'Content-Type': 'application/json',
},
body: JSON.stringify(data),
}
)
let result = await response.json()
```

Axios

Biblioteca para manipulação de requisições ao servidor



Vamos à prática