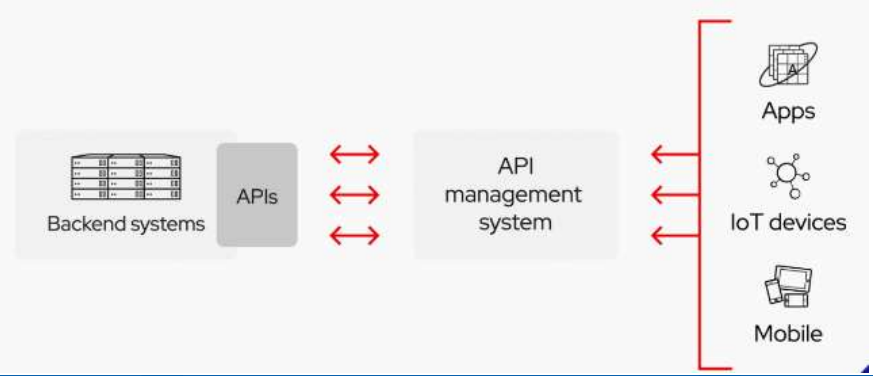
API - Interface de Programação de Aplicações

É um conjunto de rotinas que define como o cliente e o servidor podem interagir. Nela são definidos as URLs, métodos, parâmetros, formatos de dados (JSON, XML, etc) e outros detalhes que os desenvolvedores precisam seguir para interagir corretamente com o servidor.

Por meio do uso de APIs, é possível que diferentes aplicações conversem entre si, por exemplo, um front-end interagir com um back-end.



Protocolo HTTP

Os clientes se comunicam com servidores utilizando o protocolo HTTP. Para isso são enviadas requisições para uma API, as quais são compostas por elementos como:

- URL: indica o alvo da requisição (recurso);

- Método: define a ação que deve ser executada, como obter, excluir ou criar um recurso;

- Informações adicionais: para fornecer algum filtro ou parâmetro no recurso, por exemplo ordenar os resultados de forma decrescente.

Métodos HTTP

Os métodos vão definir o tipo de ação a ser executada, por exemplo:

GET - para recuperar dados
POST - para enviar dados
PUT/PATCH - para atualizar dados
DELETE - para excluir dados

Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como HTTP Verbs (Verbos HTTP)

CÓDIGOS DE RETORNO HTTP

200 -> OK: o servidor retorna os dados solicitados com sucesso
201 -> Created: um novo recurso foi criado com sucesso
204 -> No content: solicitação foi correta, mas o servidor não retorna encontrou conteúdos
400 -> Bad Request: solicitação contém dados inválidos ou está mal formada
401 -> Unauthorized: cliente deve fornecer autenticação para acessar o recurso
403 -> Forbidden: o acesso ao recurso é proibido.
404 -> Not found: recurso solicitado não foi encontrado no servidor.
405 -> Method not allowed: método incorreto, trocou GET por POST, por exemplo
500 -> Internal Server error: servidor encontrou um erro interno ao processar a solicitação
503 -> Service Unavailable: servidor não está disponível para lidar com a solicitação devido a sobrecarga ou manutenção

Criação da primeira API

```
// Importando o modulo
const express = require('express')
// Executando a função que terá as funcionalidades
const app = express()

// URLs (rotas)
app.get('/api', (req, res) => {
    res.status(200).send('Olá, mundo!')
})

app.post('/api', (req, res) => {
    res.status(200).send('Requisição de POST recebida!')
})

// Método listen indica a porta que o servidor vai rodar e um callback
para quando iniciar
app.listen(3000, () => console.log('O servidor está rodando'))
```

Passagem de parâmetros

Podemos receber informação via API de algumas formas, vimos as seguintes:

- Parâmetros de rota (req.params): cria-se um caminho na API separado por ‘/’, onde é possível passar uma informação:

  /api/:nomeDoParametro

  /api/**3**

- Parâmetros de consulta/query (req.query): são passados no caminho utilizando o sinal de ‘?’ e uma consulta:

  /api**?usuario=ruan**

- Parâmetros de corpo/body (req.body): são passados via body, fora da URL.

Acessar o body – forma 1

Nas solicitações de **POST**, **PUT** e **PATCH** que costumam precisar de muitas informações, é comum passarmos isso via body em formato JSON.

Para acessar o body, podemos instalar o pacote body-parser, importá-lo e depois utilizar a propriedade da requisição: req.body

```
const bodyParser = require('body-parser')
app.use(bodyParser.json())

...

app.post('/api', (req, res) => {
    const body = req.body
    res.status(200).send(`Recebemos seus dados ${body.nome}!`)
})
```

Acessar o body – forma 2

O express nas versões mais recentes permite transformar o body para JSON sem a necessidade do body-parser, basta substituir bodyParser.json() por express.json()

```
app.use(express.json())

...

app.post('/api', (req, res) => {
    const body = req.body
    res.status(200).send(`Recebemos seus dados ${body.nome}!`)
})
```