# Requisições na prática

Orientadora: Kelly Joany de Oliveira Santos

Reprograma: Turma Backend - Mercado Livre

# API REST

# Como funciona?

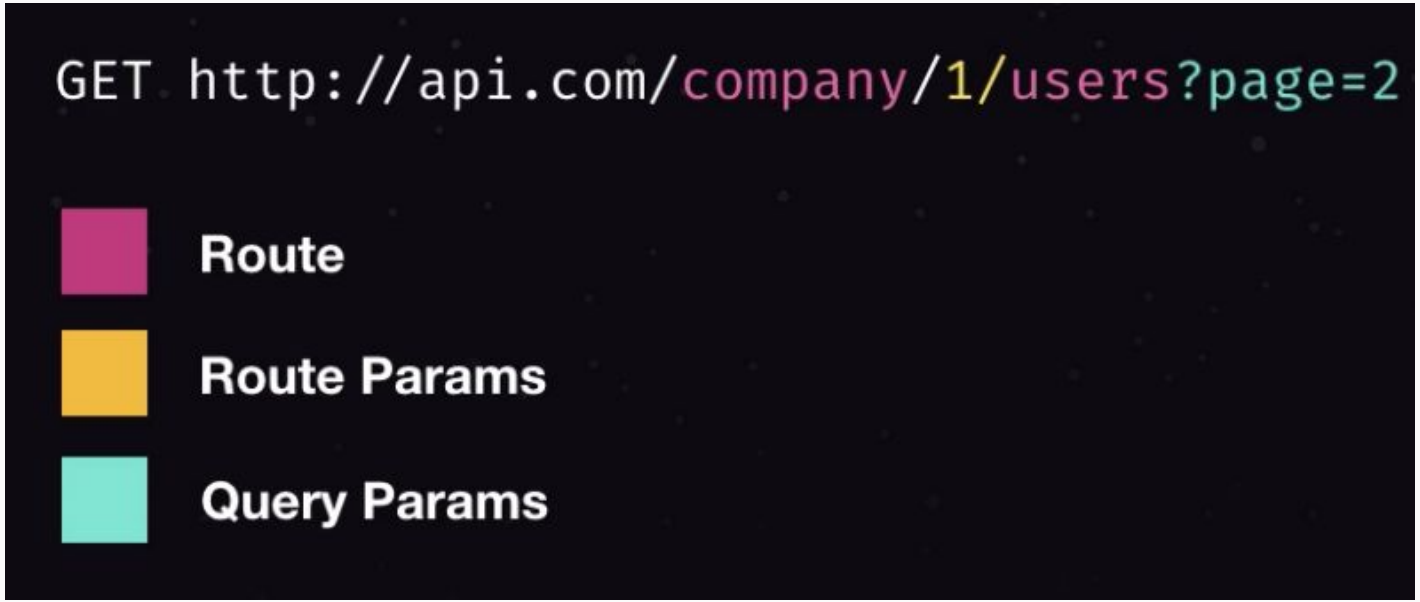
- Fluxo da requisição e resposta
  - Requisição feita por um cliente
  - Resposta retornada através de uma estrutura de dados
  - Cliente recebe resposta e processa resultado
- As rotas utilizam os métodos
  - GET http://minhaapi.com/users
  - POST http://minhaapi.com/users
  - PUT http://minhaapi.com/users/1
  - DELETE http://minhaapi.com/users/1

# Benefícios

- Multiplus Clientes (frontend), mesmo backend
- Protocolo de comunicação padronizado
  - Mesma estrutura para mobile / web / API pública
  - Comunicação com serviços externos

# JSON ( Javascript object notation)

# Conteúdo da requisição



# HTTP codes

- 1xx - Informational
- 2xx - Success
  - 200 Success
  - 201 Created
- 3xx - Redirection
  - 301 Moved Permanently
  - 302 Moved

# HTTP codes

- 4xx - Client Error
  - 400 Bad Request
  - 401 Unauthorized
  - 404 Not found
- 5xx - Server Error
  - 500 Internal Server Error

# Rota básica GET

```
var express = require('express')
var app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
})
```

# Exemplos de métodos

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage')
})

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage')
})
```

# Variações Rotas GET

```
app.get('/', function (req, res) {
  res.send('root')
})
```

```
app.get('/about', function (req, res) {
  res.send('about')
})
```

# Router Params

Os params são denominados segmentos de URL que são usados para capturar os valores especificados em sua posição no URL. Os valores obtidos são preenchidos no objeto req.params, com o nome do parâmetro de rota especificado no caminho como suas respectivas chaves.

# Router Params

Para definir rotas com parâmetros de rota, basta especificar os parâmetros de rota no caminho da rota, como mostrado abaixo. Vamos adicionar o seguinte código ao nosso arquivo app.js:

```
app.get('/users/:username', (req, res, next) => {
  res.send(req.params);
})
```

# Router Params

```
http://localhost:3000/users/ironhack
```

```
{"username":"ironhack"}
```

# Router Params

Observe que o nome de usuário: é preenchido no objeto req.params como uma chave e a string que enviamos nessa posição da URL é o valor dessa chave. Isso significa que podemos substituir o nome de usuário por qualquer coisa que desejarmos. Por exemplo, vamos receber um bookId dessa vez:

# Router Params

```
app.get('/books/:bookId', (req, res, next) => {
  res.send(req.params);
})
```

**http://localhost:3000/books/131l2kj3h$j3h1jk2**

```
{"bookId":"131l2kj3h$j3h1jk2"}
```

# Mais de um Router Params

uma vez que precisaremos enviar mais de um parâmetro, nesses casos, podemos fazer algo como o seguinte:

```
app.get('/users/:username/books/:bookId', (req, res, next) => {
  res.send(req.params)
})
```

# Mais de um Router Params

```
http://localhost:3000/users/ironhack/books/8989
```


```
{"username":"ironhack","bookId":"8989"}
```

# Query Strings

Às vezes, achamos útil enviar parâmetros na URL como uma string e, para isso, os query strings são fantásticos.

Em termos simples, a string de consulta faz parte de um URL após o ponto de interrogação ? e geralmente contém pares de valores-chave separados por & e =. Esses pares de chave / valor podem ser usados pelo servidor como argumentos para consultar um banco de dados ou talvez para filtrar resultados.

# Query Strings

Você receberá as informações sobre o objeto req.query de nossas rotas. Adicione o seguinte código no arquivo app.js:

```
app.get('/search', (req, res, next) => {
  res.send(req.query)
})
```

# Query Strings

Assim como os router params, as query strings são objetos de pares de chave-valor. Tudo o que vem depois do ? será emparelhado como chave-valor usando o = como separador.

```
http://localhost:3000/search?city=Barcelona
```

```
console.log(req.query);
// => { "city" : "Barcelona" }
```

# Mais Query Strings

Quando estamos fazendo algumas pesquisas ou filtros, é comum termos mais informações para enviar ao servidor. Nesse caso, podemos usar o & para anexar mais parâmetros.

http://localhost:3000/search?city=Barcelona&start-date=2018-01-18


```
{ "city" : "Barcelona", "start-date" : "2018-01-18" }
```

# E então....

Dada a URL: http://localhost:3000/products/1345?Show=reviews, e a rota /products/:id, podemos desestruturar o objeto req nos seguintes campos:

| **HTTP Request** | **Express `req` object** |
| --- | --- |
| Method | req.method // `GET` |
| URL Path | req.path // `/products/1345` |
| URL Params | req.params.id // `1345` |
| URL Query String | req.query.show // `reviews` |