



# MINICURSO

## Criando Websites Dinâmicos com Node.js

---

Vitor Bruno de Oliveira Barth INSTRUTOR  
André Geraldo Guimarães Pinto MONITOR  
Yuri Willians dos Santos MONITOR

Prof. Dr. Ruy de Oliveira ORIENTADOR

# OBJETIVO DO MINICURSO

Ensinar a teoria e a prática do desenvolvimento de aplicações web dinâmicas utilizando Node.js

---

# CRONOGRAMA

## Dia 1 – Criação de APIs com Node.js

### TEORIA

Estrutura de Aplicações Web

Conceitos de API REST

Node.js e Express.js

### LABORATÓRIO

Trabalhar com APIs

## Dia 2 – Criação de Páginas Web

### TEORIA

Revisão do Dia 1

Estrutura Básica do Bootstrap

Gráficos com C3.js

Requisições HTTP com Fetch

### LABORATÓRIO

Criar Páginas Dinâmicas

# TÓPICO 1

## Estrutura de Aplicações Web

---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

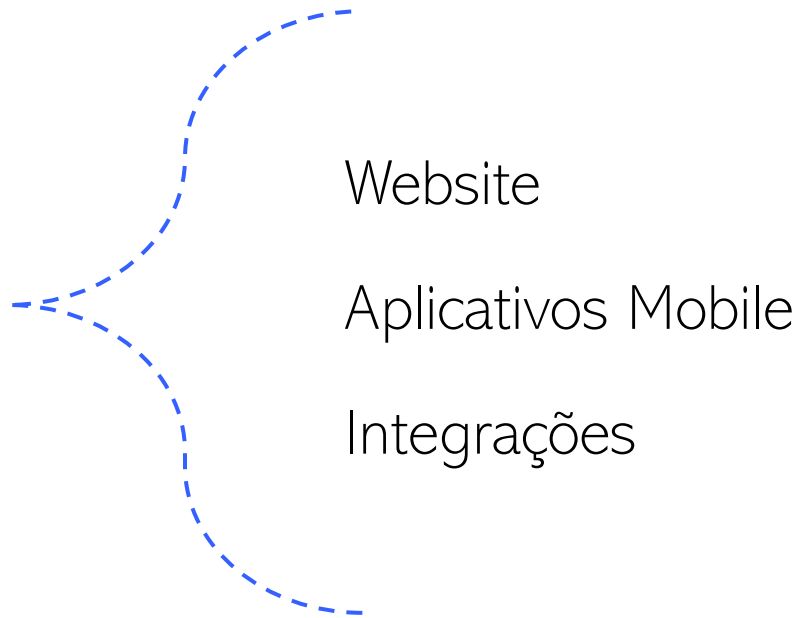
# TÓPICO 1

## Estrutura de Aplicações Web

---

ESTUDO DE CASO

**facebook**



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

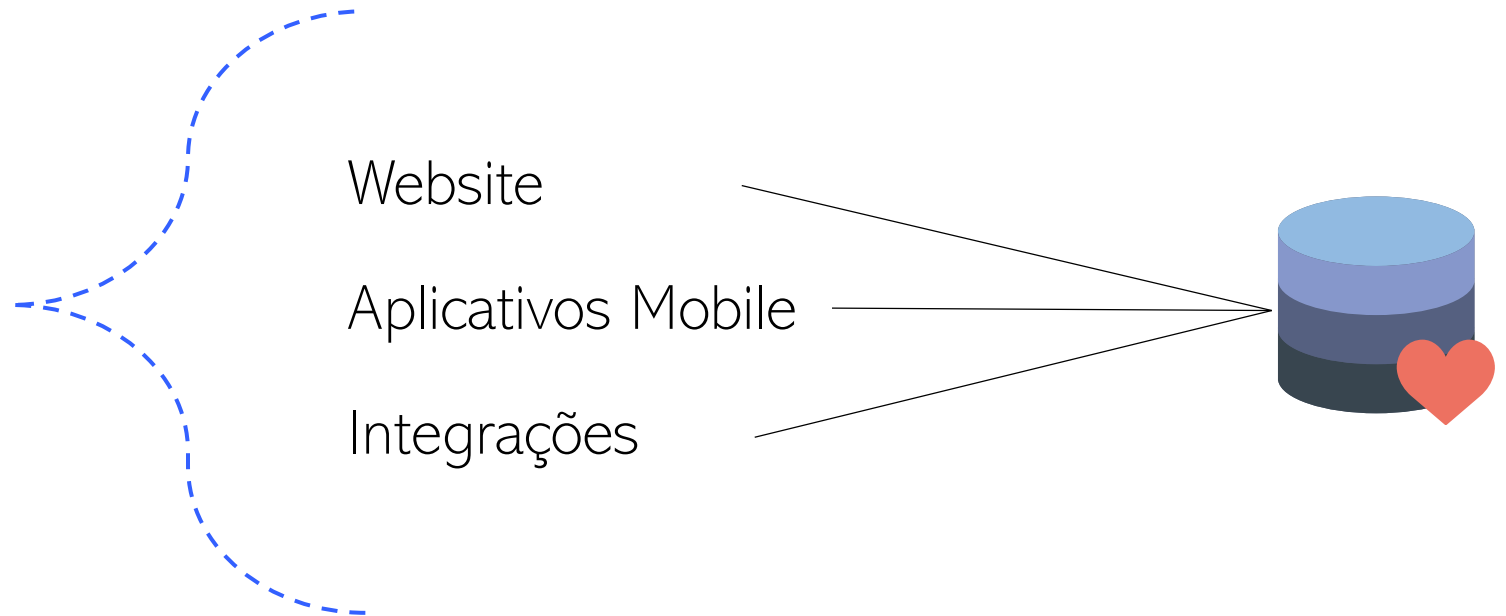
# TÓPICO 1

## Estrutura de Aplicações Web

---

ESTUDO DE CASO

**facebook**



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

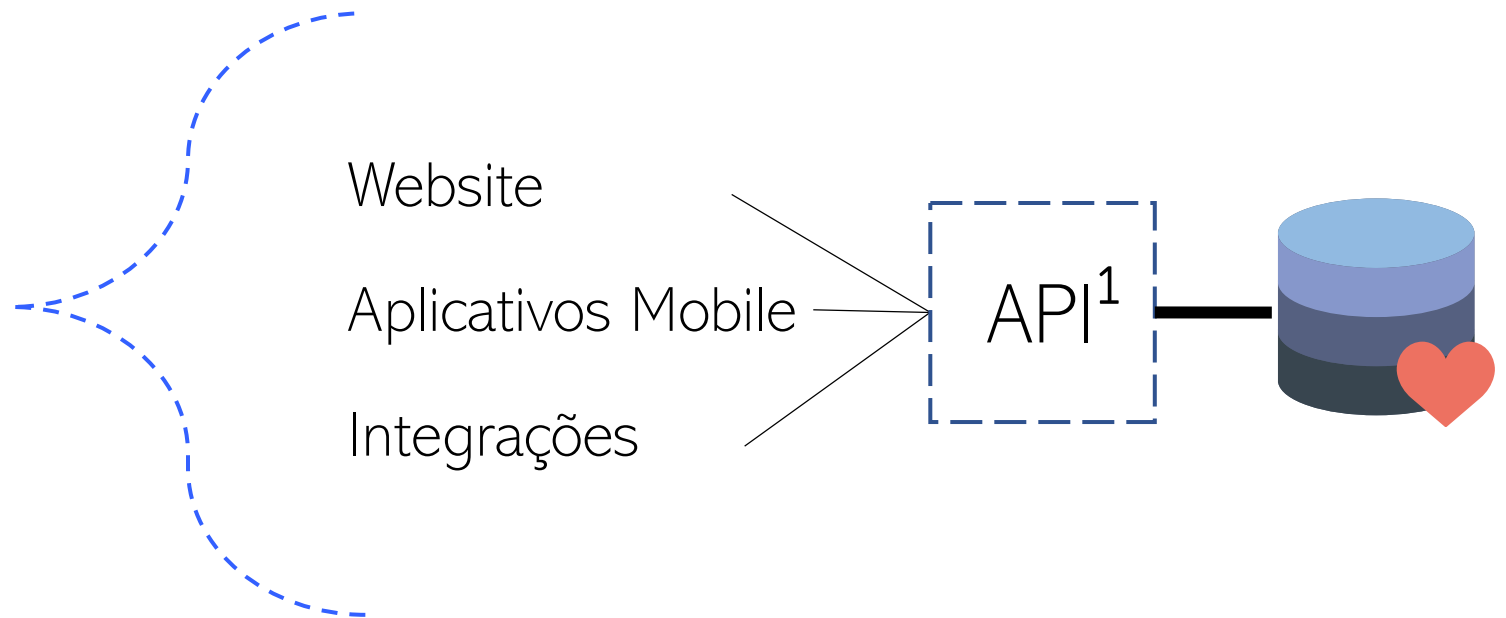
# TÓPICO 1

## Estrutura de Aplicações Web

---

ESTUDO DE CASO

**facebook**

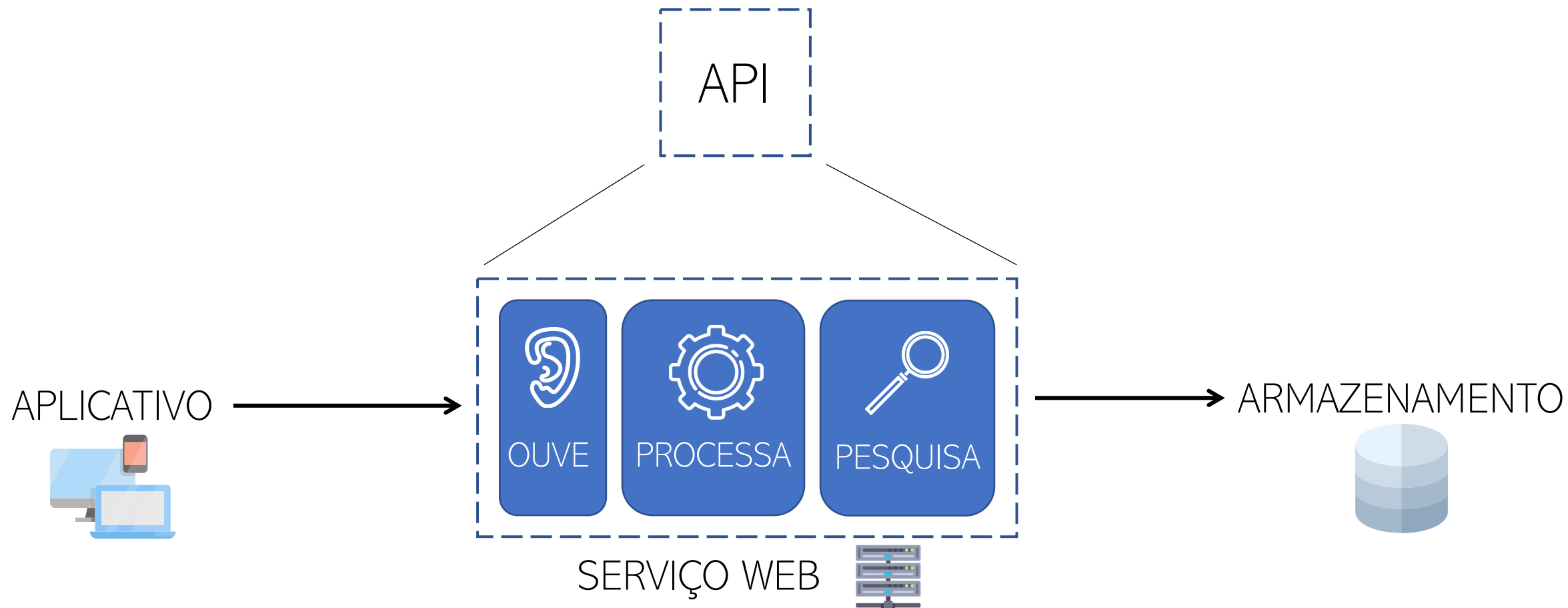


<sup>1</sup> API é acrônimo para *Application Programming Interface* (Interface de Programação de Aplicativos), e se refere à qualquer estrutura que permite a troca de informações entre *softwares* independentes.

# TÓPICO 1

## Estrutura de Aplicações Web

---

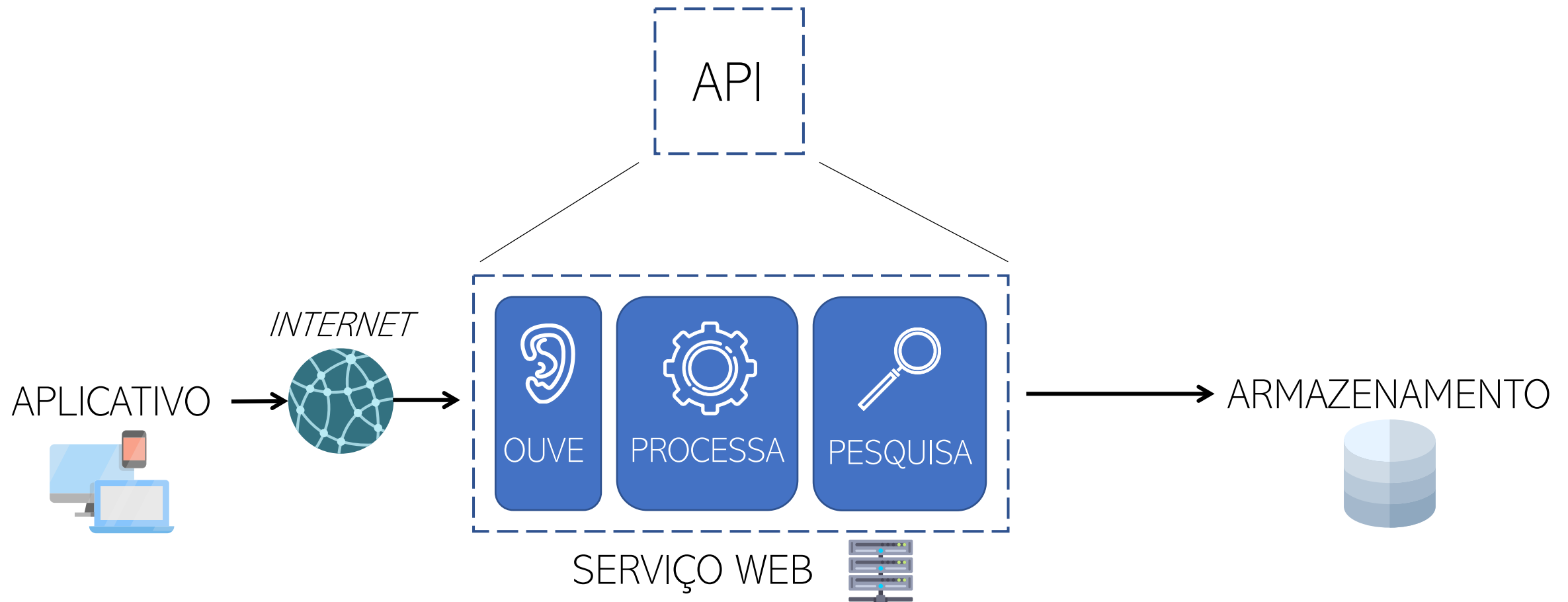




# TÓPICO 1

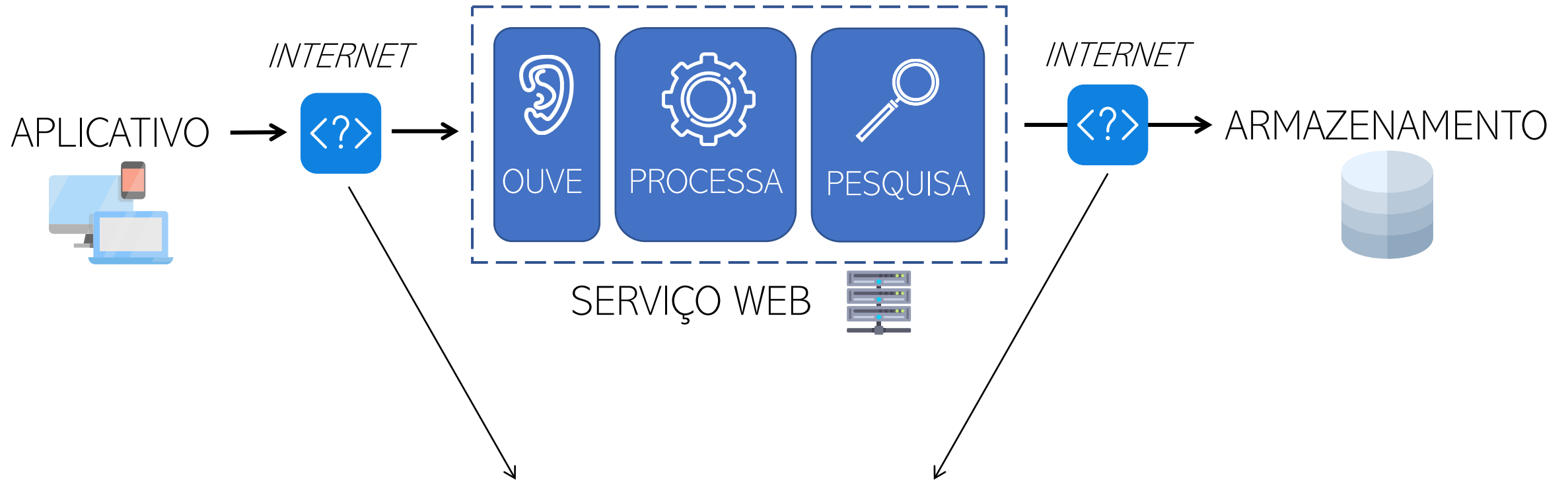
## Estrutura de Aplicações Web

---



# TÓPICO 1

## Estrutura de Aplicações Web

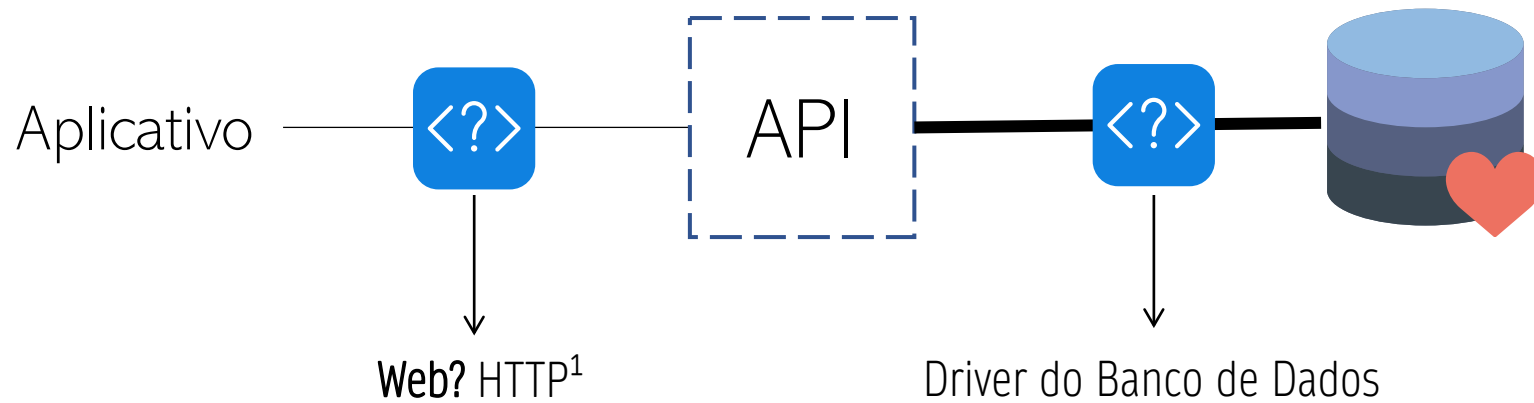


Para que aplicações independentes se comuniquem, é necessário estabelecer uma estrutura de dados que ambas conheçam

# TÓPICO 1

## Estrutura de Aplicações Web

---

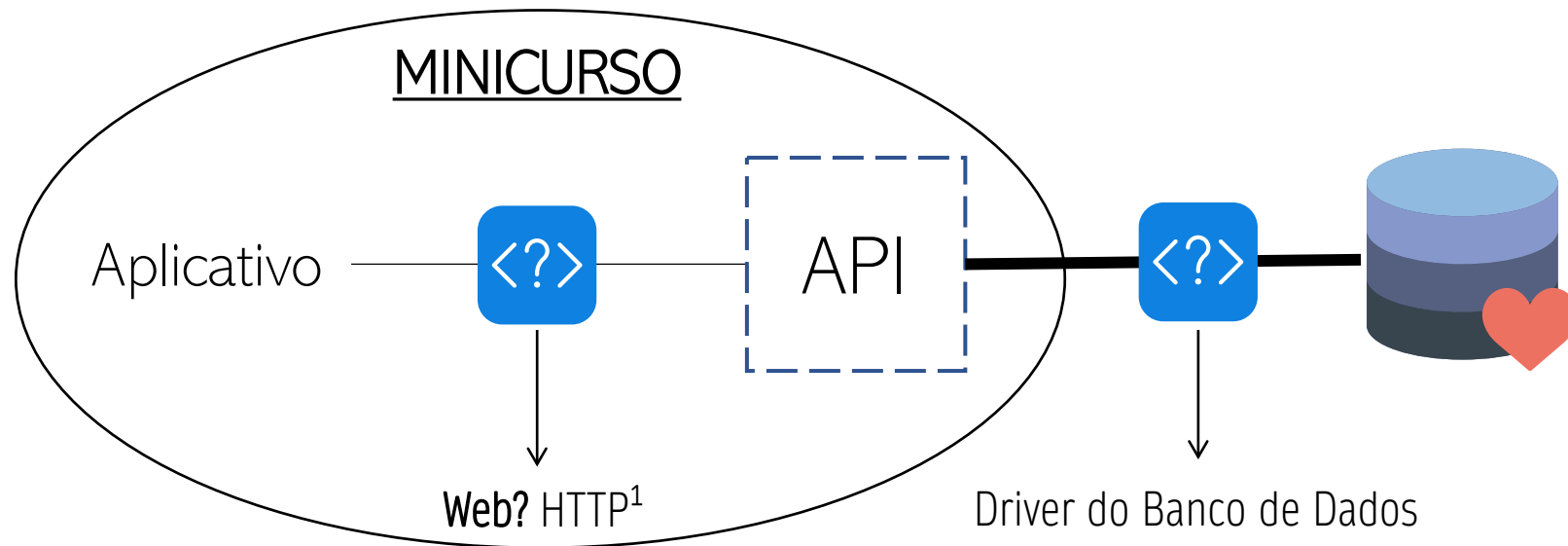


<sup>1</sup> HTTP é acrônimo para *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto), e é o conjunto de regras que define a forma de comunicação entre aplicações por meio da *web*.

# TÓPICO 1

## Estrutura de Aplicações Web

---



<sup>1</sup> HTTP é acrônimo para *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto), e é o conjunto de regras que define a forma de comunicação entre aplicações por meio da *web*.

# TÓPICO 2

## Conceitos de API REST

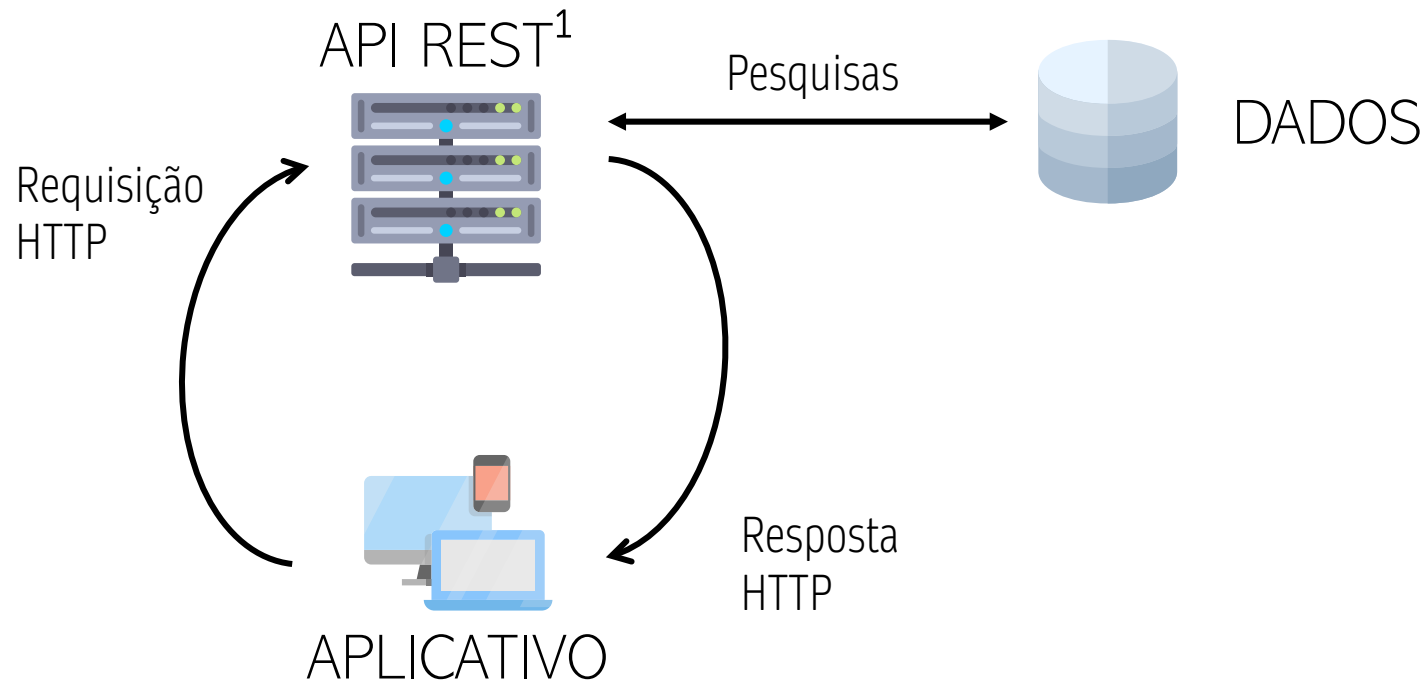
---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

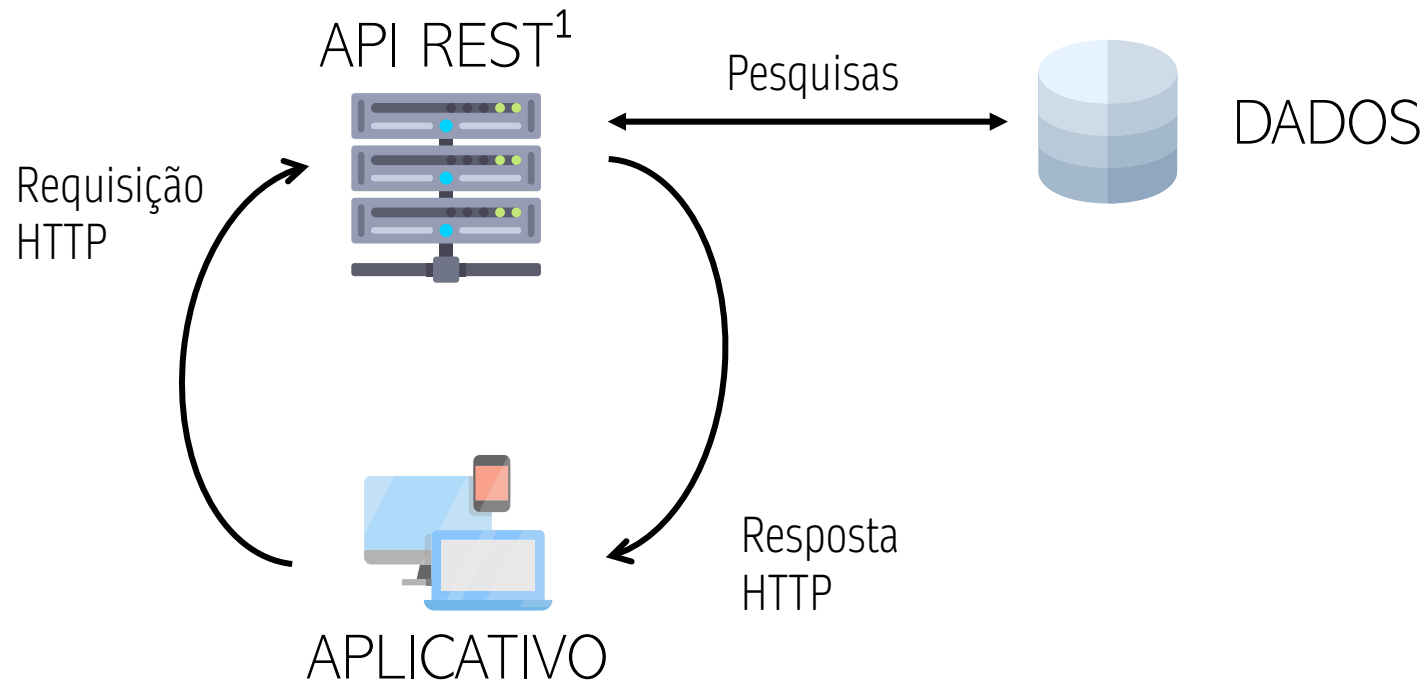


<sup>1</sup> REST é acrônimo para *Representational State Transfer* (Transferência Representacional de Estado), o qual define uma linguagem padrão de comunicação entre serviços web.

# TÓPICO 2

## Conceitos de API REST

---



O padrão REST institui diversos métodos, destacando-se:

- > **GET** para buscar
- > **POST** para salvar
- > **PUT** para sobrescrever
- > **PATCH** para editar
- > **DELETE** para remover

<sup>1</sup> REST é acrônimo para *Representational State Transfer* (Transferência Representacional de Estado), o qual define uma linguagem padrão de comunicação entre serviços web.

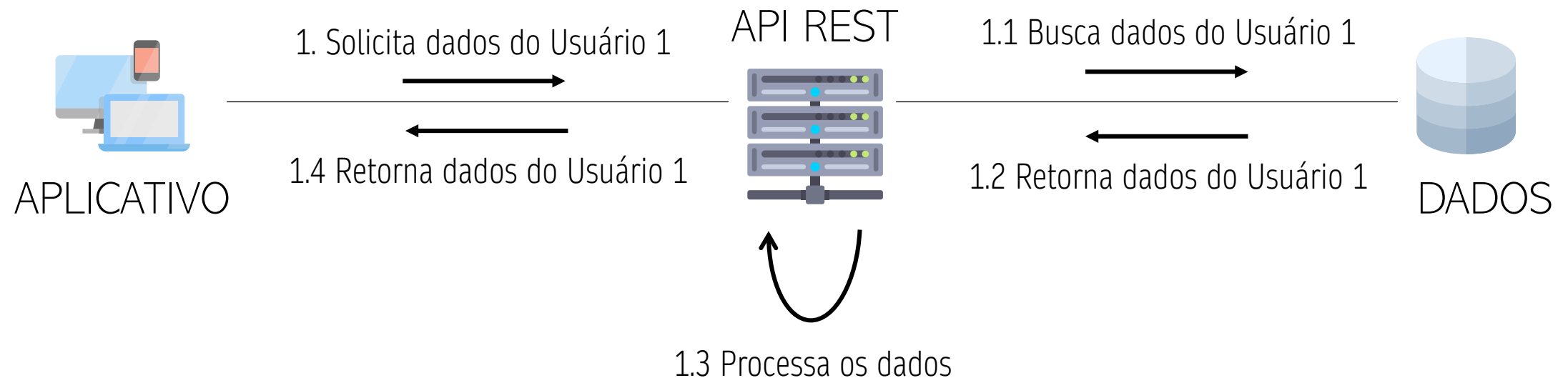
# TÓPICO 2

## Conceitos de API REST

---

### ESTUDO DE CASO

Buscar um Usuário



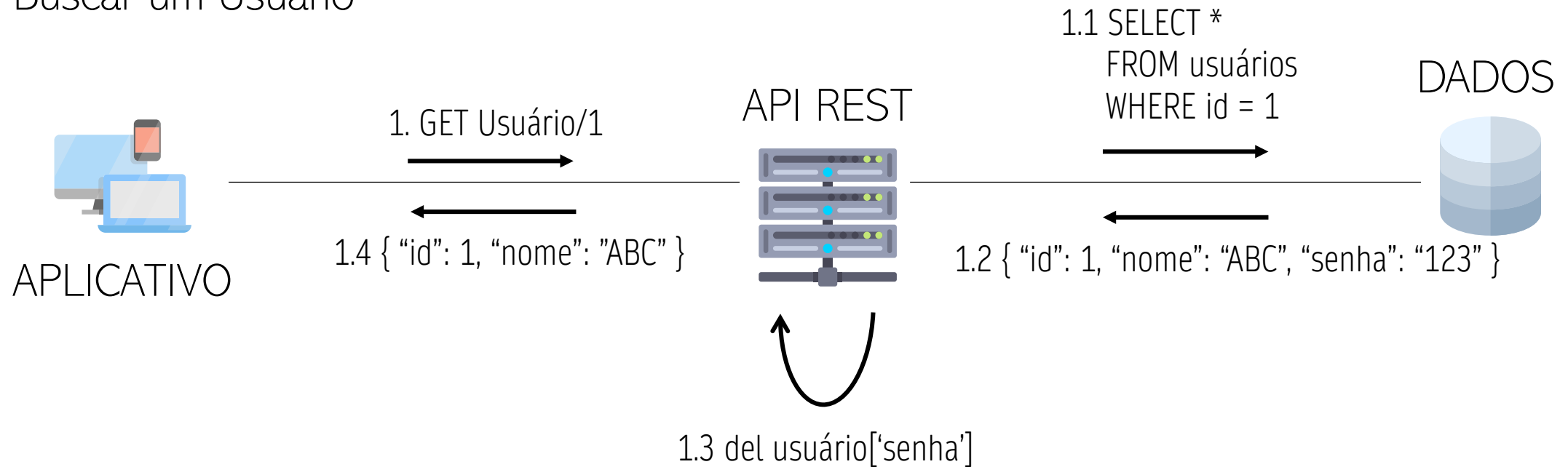


# TÓPICO 2

## Conceitos de API REST

### ESTUDO DE CASO

Buscar um Usuário

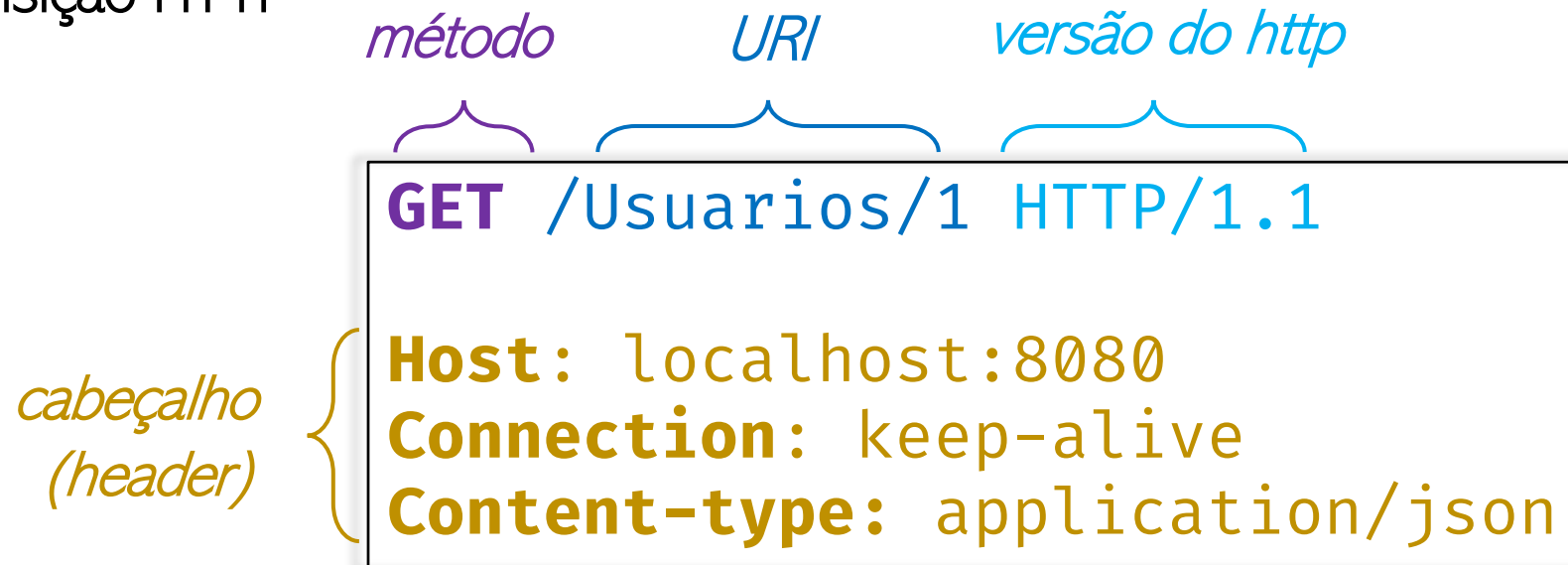


# TÓPICO 2

## Conceitos de API REST

---

### Requisição HTTP



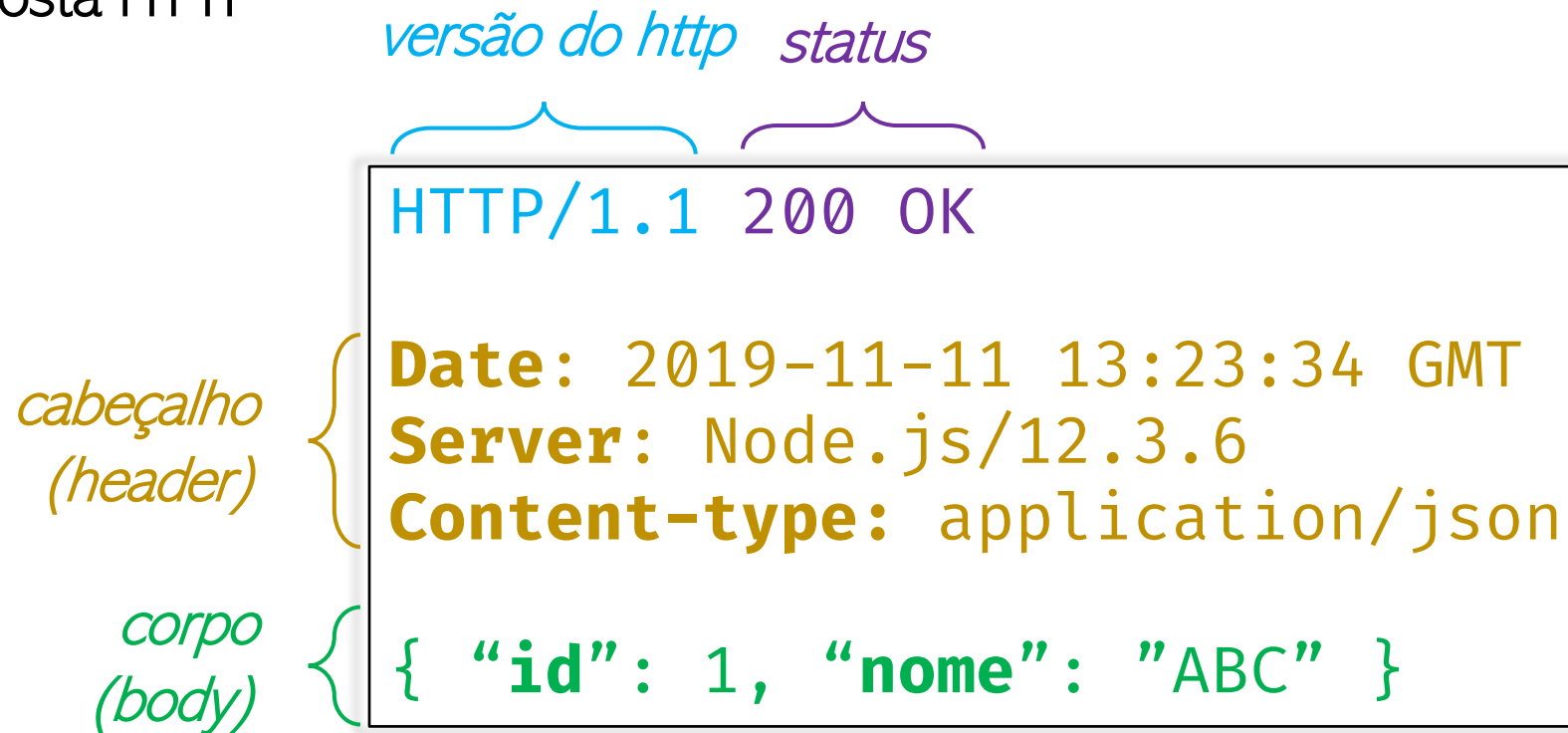
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### Resposta HTTP



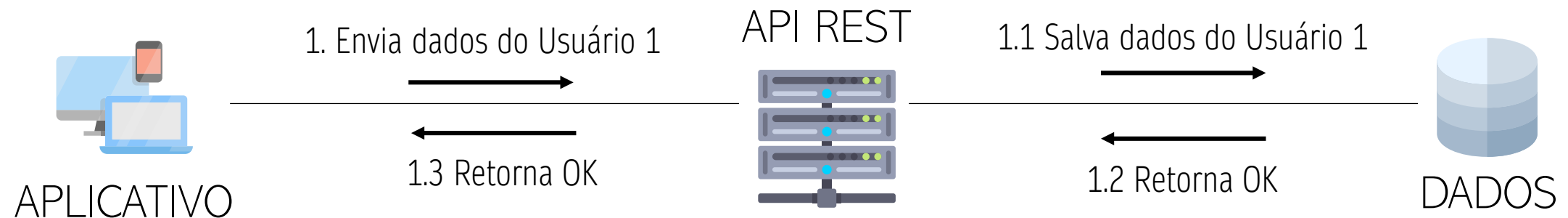
# TÓPICO 2

## Conceitos de API REST

---

### ESTUDO DE CASO

Cadastrar um Novo Usuário



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

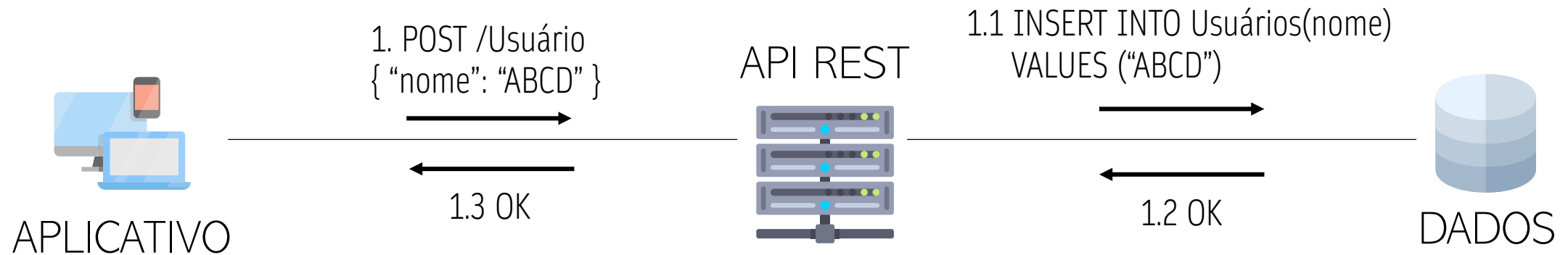
# TÓPICO 2

## Conceitos de API REST

---

### ESTUDO DE CASO

#### Cadastrar um Novo Usuário



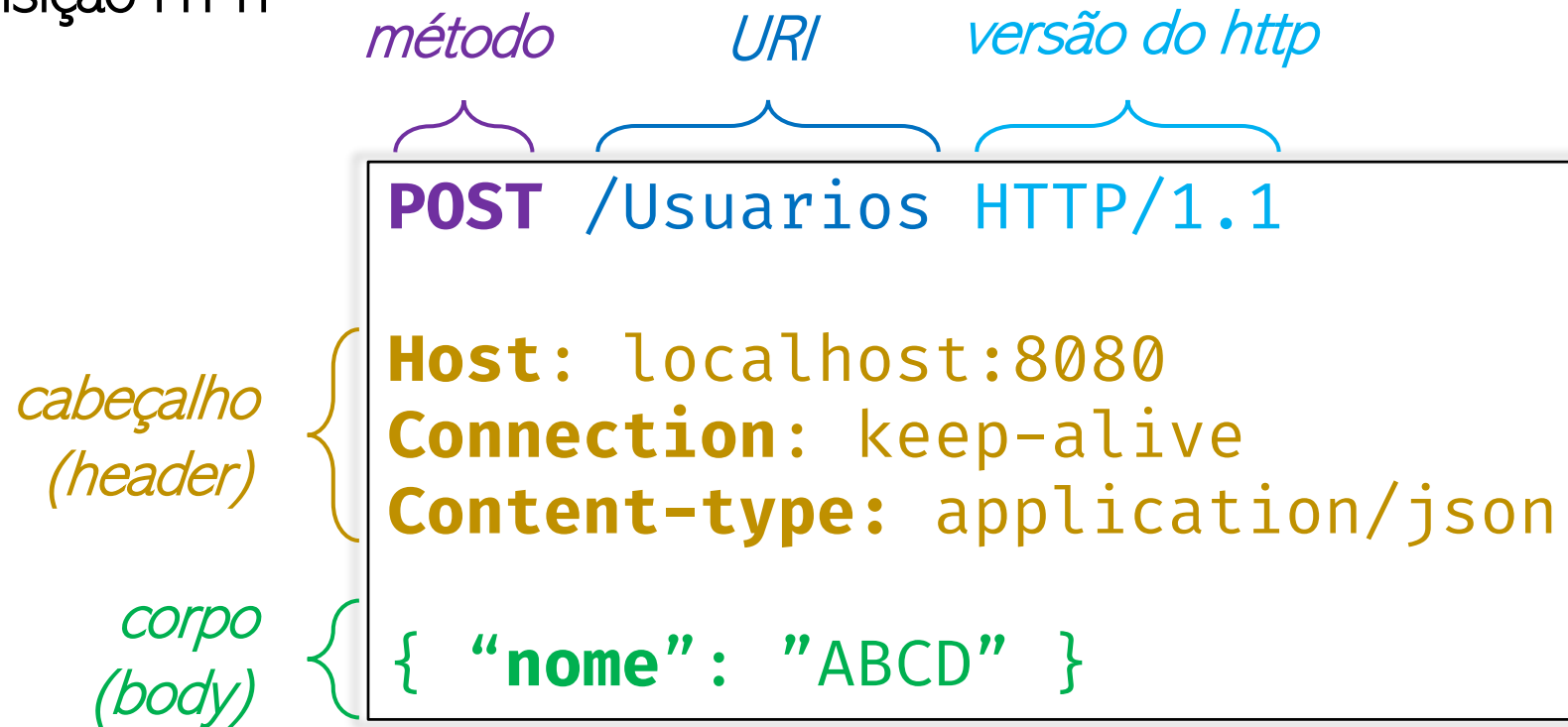
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### Requisição HTTP



# TÓPICO 2

## Conceitos de API REST

---

### Resposta HTTP

*versão do http*   *status*

**HTTP/1.1 200 OK**

*cabeçalho (header)* { **Date:** 2019-11-11 13:23:34 GMT  
**Server:** Node.js/12.3.6

O padrão REST institui diversos códigos de status:

> 100 - Informação

> 2XX – Sucesso  
200 OK

> 3XX - Redirecionamento  
301 Moved Permanently

> 4XX – Erro de Cliente  
400 Bad Request  
403 Forbidden  
404 Not Found

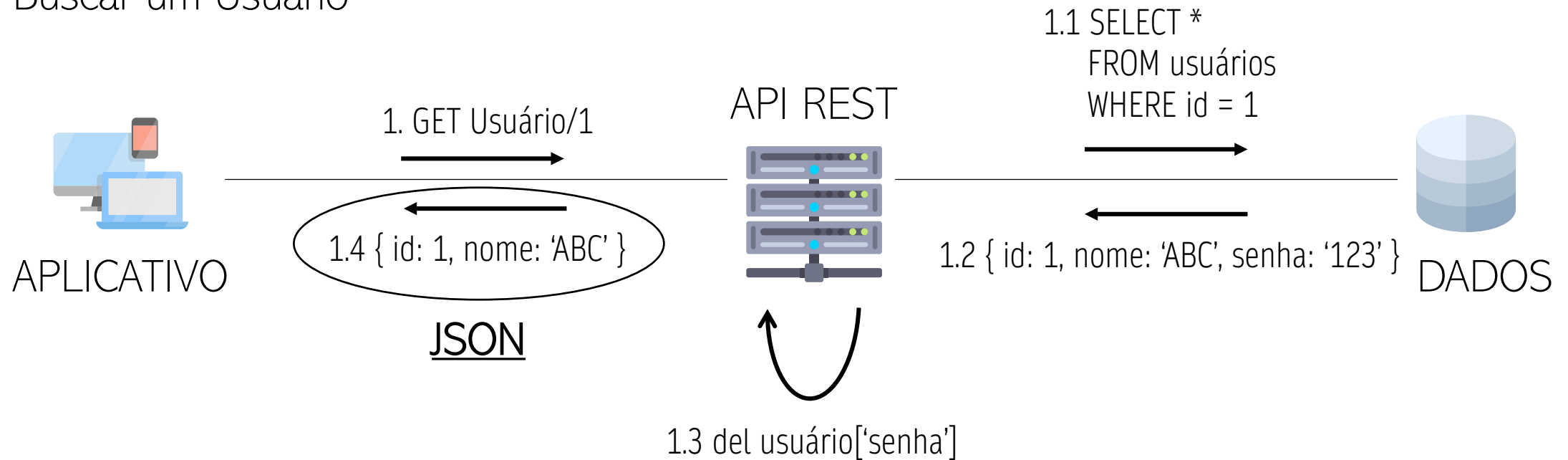
> 5XX – Erro de Servidor  
500 Internal Server Error

# TÓPICO 2

## Conceitos de API REST

### ESTUDO DE CASO

Buscar um Usuário





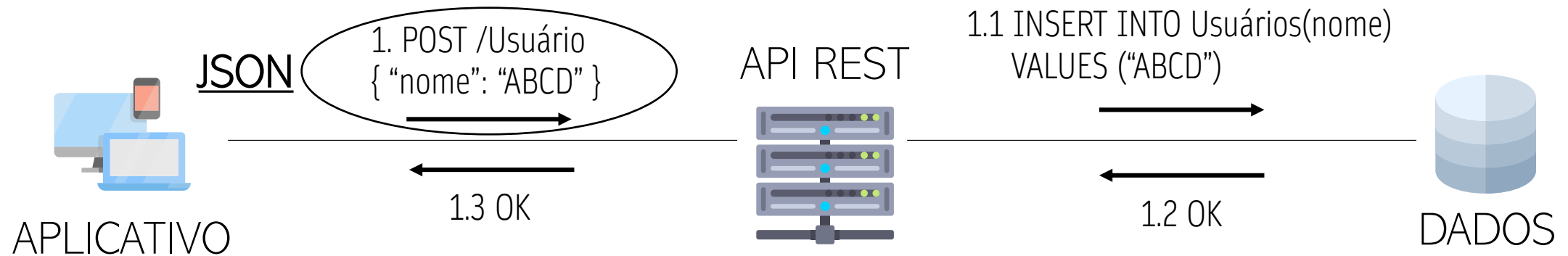
# TÓPICO 2

## Conceitos de API REST

---

### ESTUDO DE CASO

Cadastrar um Novo Usuário

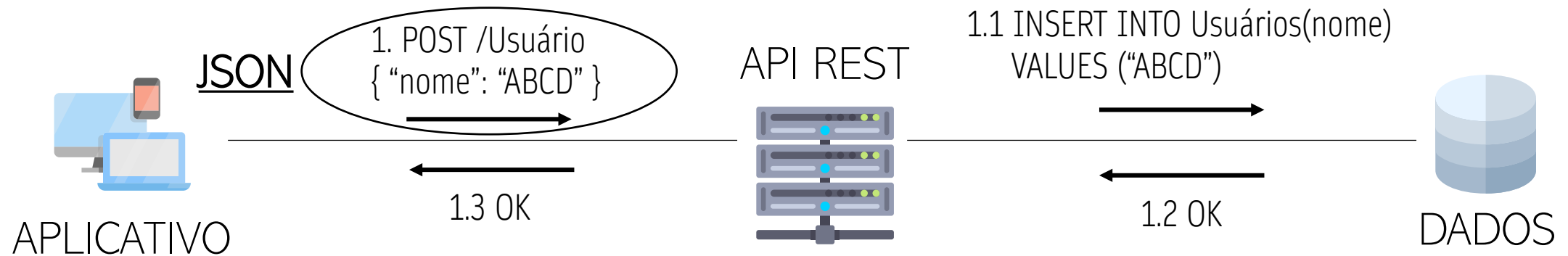


# TÓPICO 2

## Conceitos de API REST

### ESTUDO DE CASO

Cadastrar um Novo Usuário



JSON é a linguagem padrão entre a Aplicação e o Serviço Web<sup>1</sup>

<sup>1</sup> Qualquer estrutura de dados pode ser utilizada, mas JSON é o padrão adotado pela maioria das aplicações Web

# TÓPICO 2

## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
}
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{ ← Delimitado por chaves  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
} ←
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
}
```

*variável: valor*



# TÓPICO 2


## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
}
```

Virgulas separam os elementos



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
}
```

← Aspas duplas indicam *strings*

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2


## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Usuário 1",  
  "senha": "a1b2c3",  
  "criadoEm": "11/11/2019 13:23:34"  
}
```

Permite valores inteiros



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>



# TÓPICO 2


## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Produto 1",  
  "preço": 13.32  
}
```

Permite valores reais



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2


## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Sensor 1",  
  "amostras": [0, 1, 2, 1, 0, -1, -2, -1, 0]  
}
```

Permite vetores



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 2

## Conceitos de API REST

---

### JSON (JavaScript Object Notation)

```
{  
  "id": 1,  
  "nome": "Instituto Federal de Mato Grosso",  
  "salas": [{  
    "numero": "E014",  
    "professor": "Professor 1",  
  }, {  
    "numero": "E015",  
    "professor": "Professor 2",  
  }]  
}
```

← Permite objetos aninhados

# INTERVALO

## 15 MINUTOS

---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---



- Interpretador de **JavaScript**
- Desenvolvido em 2009 por Ryan Dahl
- Baseado no motor JavaScript V8, da Google
- Possui 500.000+ bibliotecas disponíveis
- Pode ser usado como **Serviço Web**

# TÓPICO 3

## Criando APIs com Node.js e Express

---

express

- Framework para Node.js
- Facilita a criação de Serviços Web REST

# TÓPICO 3

## Criando APIs com Node.js e Express

---

Instalar o Express e o Body Parser

A terminal window with a dark gray background and a green border. The title bar says "Terminal". The command `npm install express body-parser --save` is displayed in a monospaced font. The word "npm" is blue, "install" is white, "express" is white, "body-parser" is white, and "--save" is purple.

```
Terminal  
npm install express body-parser --save
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>



# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Criar um serviço com Express

```
index.js

let express = require('express')
let app = express()

app.listen(8080, function (err) {
  if (err) throw err
  console.log('Servidor ouvindo em http://localhost:8080')
})

app.get('/', function (req, res) {
  res.send('Semana da Informática IFMT 2019')
})
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

Iniciar a aplicação



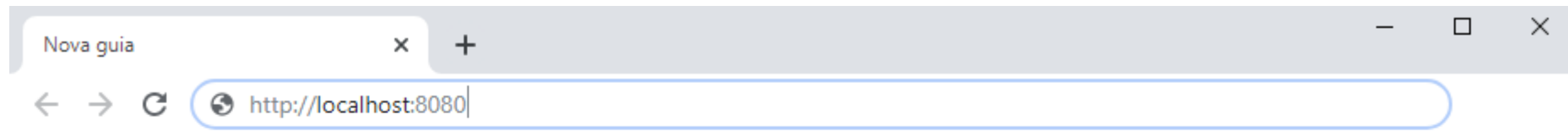
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Testando a aplicação no Navegador



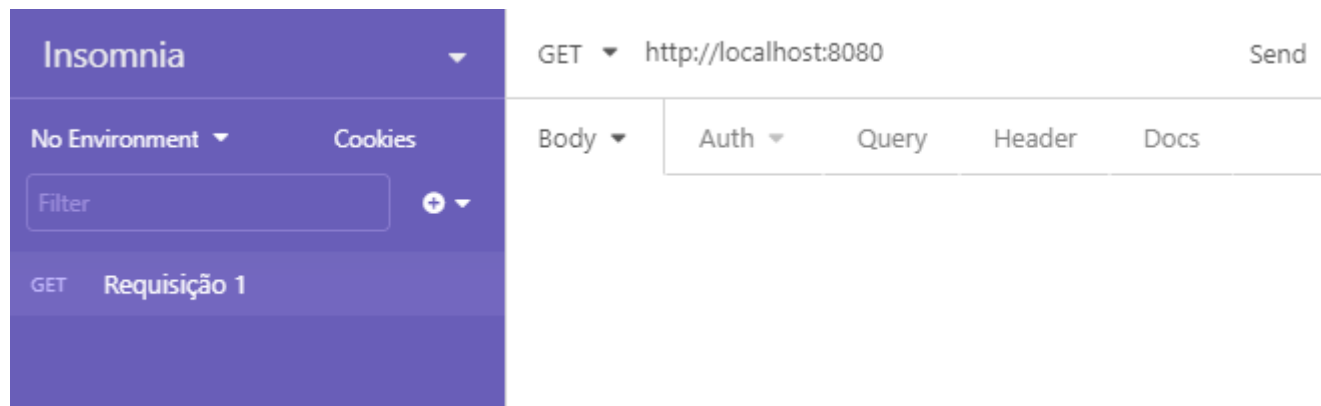
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Testando a aplicação no Insomnia ®



P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Análise do Código

A code editor window with a green border and a dark background. It has three colored window control buttons (red, yellow, green) in the top left corner. The title bar reads 'index.js'. The code inside is:

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

Importa a biblioteca *Express* e cria um aplicativo

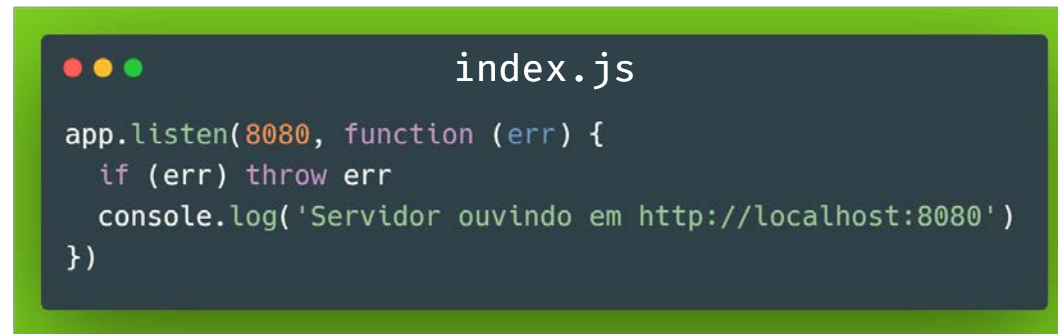
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Análise do Código



```
index.js

app.listen(8080, function (err) {
  if (err) throw err
  console.log('Servidor ouvindo em http://localhost:8080')
})
```

Cria um serviço web que ouve na porta 8080, que após iniciado escreve uma mensagem no console

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Análise do Código



```
index.js

app.get('/', function (req, res) {
  res.send('Semana da Informática IFMT 2019')
})
```

Fala ao servidor ouvir por requisições do tipo GET na URI '/', e quando receber, retornar uma mensagem

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Servindo páginas HTML com Express



```
index.js

[ ... ]

app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html')
})

[ ... ]
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>



# TÓPICO 3

## Criando APIs com Node.js e Express

---

## Servindo Arquivos Estáticos com Express

```
index.js

let express = require('express')
let app = express()

app.use('/static', express.static('static'))

app.listen(8080, function (err) {
  if (err) throw err
  console.log('Servidor ouvindo em http://localhost:8080')
})

app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html')
})

app.get('/texto', function (req, res) {
  res.send('Semana da Informática IFMT 2019')
})
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Requisições POST com Express

```
index.js

let express = require('express')
let bodyParser = require('body-parser')

let app = express()
app.use(bodyParser.json())

let cache = ''

app.listen(8080, function (err) {
  if (err) throw err
  console.log('Servidor ouvindo em http://localhost:8080')
})

app.get('/cache', (req, res) => {
  res.json({ value: cache })
})

app.post('/cache', (req, res) => {
  cache = req.body.value
  res.json({ success: true })
})
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

## Tratamento de Erros com Express

A screenshot of a code editor window titled 'index.js'. The code is written in JavaScript and demonstrates error handling for a POST request to '/cache'. It uses a try-catch block to handle the request body. If the body is not valid, it throws an error, which is then caught and a 400 status code is returned with a message 'Estrutura mal-formada'.

```
index.js

[ ... ]

app.post('/cache', (req, res) => {
  try {
    if (req.body.value) {
      cache = req.body.value
      res.json({ success: true })
    }

    else throw Error()
  }

  catch {
    res
      .status(400)
      .json({ success: false, message: 'Estrutura mal-formada' })
  }
})
```

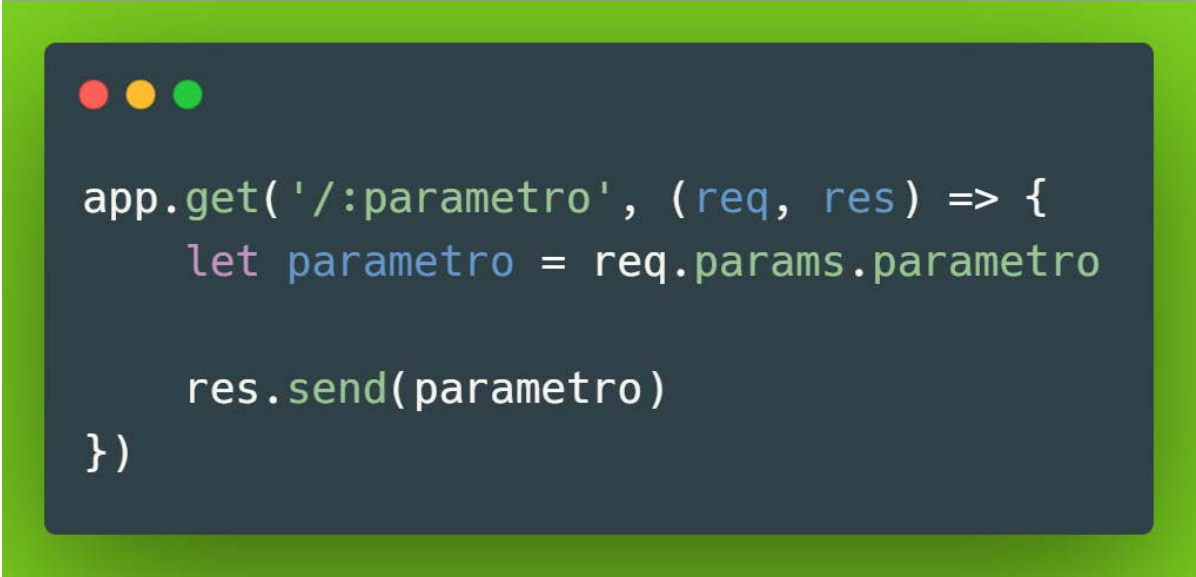
P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# TÓPICO 3

## Criando APIs com Node.js e Express

---

### Rotas Dinâmicas com Express



```
app.get('/:parametro', (req, res) => {  
  let parametro = req.params.parametro  
  
  res.send(parametro)  
})
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# ATIVIDADE 1

## Criar uma API que gerencia alunos

---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# ATIVIDADE 1

Criar uma API que gerencia alunos

---

```
index.js

class Aluno {
  constructor(id, nome, endereco, email) {
    this.id = id
    this.nome = nome
    this.endereco = endereco
    this.email = email
  }
}

let alunos = []

let vitor = new Aluno(1, 'Vitor Barth', 'Rua Zulmira Canavarros, 95', 'vitor.barth@gmail.com')
alunos.push(vitor)

/* SUGESTÃO DE PASSOS:
1. Importar Express
1.1 Iniciar o servidor na porta 8080

2. Importar Body Parser
3. Criar uma rota GET que retorna o vetor alunos
4. Criar uma rota GET que retorna alunos por ID
5. Criar uma rota GET que retorna alunos por Email
6. Criar uma rota POST que salva no vetor alunos
7. Criar uma rota PUT que sobrescreve aluno por ID
*/
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

# ATIVIDADE 2

Criar uma API que gera números aleatórios

---

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>

## ATIVIDADE 2

Criar uma API que gera números aleatórios

---

```
let timeSeries = {  
  1: [],  
  2: [],  
  3: []  
}  
  
// PASSOS:  
// 1. Importar o Express  
// 2. Iniciar o servidor na porta 8080  
// 3. Criar uma rota GET que retorna todas as TimeSeries  
// 4. Criar uma rota GET que retorna uma TimeSeries por ID  
// 5. Criar uma rota POST que salva um novo valor em uma TimeSeries por ID  
// 6. Criar uma rota POST que adiciona um valor aleatório em uma TimeSeries por ID
```

P.S.: Todo o material utilizado está disponível em <http://pastebin.com/PFawQgtc>



Todo o material utilizado está disponível em  
<http://pastebin.com/PFawQgtc>

# FIM DO DIA 1

## Obrigado pela presença! Até Amanhã!

---

### Realização



### Agradecimentos



### Contato

André Guimarães – [andregp019@gmail.com](mailto:andregp019@gmail.com)

Yuri Santos – [yuriduli12@gmail.com](mailto:yuriduli12@gmail.com)

Vitor Barth – [vitor.barth@gmail.com](mailto:vitor.barth@gmail.com)

Prof. Dr. Ruy de Oliveira – [ruy@cba.ifmt.edu.br](mailto:ruy@cba.ifmt.edu.br)