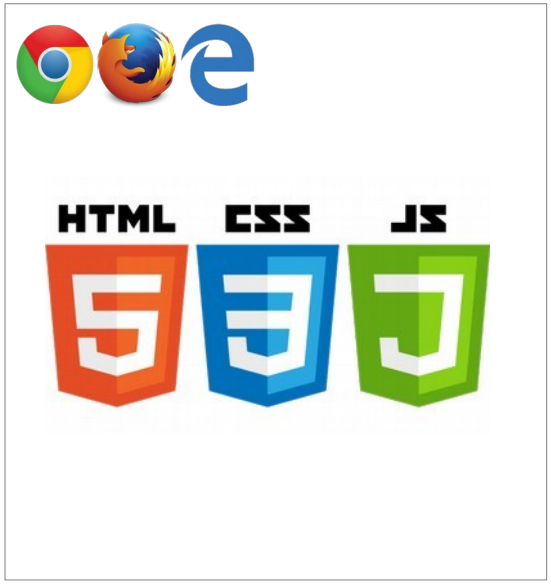
# Express

**Programação para Internet I**

IFB - Tecnologia em Sistemas para Internet

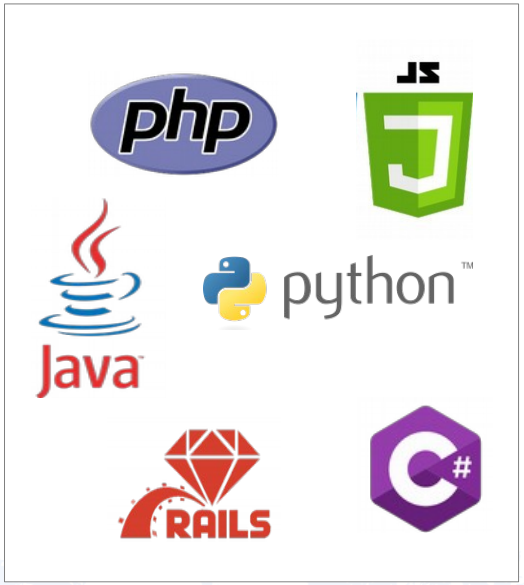
Marx Gomes van der Linden

# Arquitetura da web




Requisição HTTP

Resposta HTTP




**Cliente (Front end)**

**Servidor (Back end)**

# Express

- Framework que permite construir o **back-end** de uma aplicação web em Javacript
  - » Javascript executa no servidor (não depende do navegador)
  - » Parte do Node.js
- Para instalar (linha de comando):

```
npm install express --save
```

- Para usar no código Javascript:

```
const express = require('express');
```

# Nodemon

- Use o **nodemon** no lugar do **node** para executar o Express durante o desenvolvimento
  - » Reinicia automaticamente a cada modificação
- Para instalar:

```
npm install -g nodemon
```

- Preste atenção onde o executável foi salvo!

# Conceitos básicos

- Objeto **app**
  - Inicializado com **app = express()**
  - Ponto central da aplicação em Express
- Para inicializar o servidor:
  - `app.listen(porta, callback)`
    - Inicia o servidor na porta especificada e executa a função de callback quando estiver tudo pronto
    - Acesso em **http://localhost:porta/**

# Exemplo básico

```
const express = require('express');
const app = express();

app.listen(
    3000,
    function () {
        console.log('Inicialização OK!');
    }
);
```

- Acessar em:
  - » http://localhost:3000/

# Recebendo requisições

- `app.get(caminho, funcao)`
  - Quando uma requisição GET chega na URL com caminho (depois de “localhost:porta”), deve ser executada **`funcao`**
- Formato da função passada:
  - `function(req, res) { … }`
    - `req` → Dados da requisição (para ler)
    - `res` → Dados da resposta (para escrever)
      - `res.send(…)` → Escreve algo na resposta

# Recebendo GET em /

```
const express = require('express');
const app = express();

app.get('/', function (req, res) {
    res.send("Funciona!");
});

app.listen(3000,
    function () { console.log('Inicialização OK!'); }
);
```

- Acessar em:
  - » http://localhost:3000/

# Recebendo GET em /teste

```
const express = require('express');
const app = express();

app.get('/',
    (req, res) => { res.send("Funciona!"); }
);

app.get('/teste',
    (req, res) => { res.send("Teste funciona!"); }
);

app.listen(3000,
    () => console.log('Inicialização OK!')
);
```

- http://localhost:3000/
- http://localhost:3000/teste

# Cabeçalho CORS

- Para que o seu servidor seja capaz de responder a requisições AJAX de outros domínios, é necessário acrescentar um cabeçalho CORS
  - » Cross-origin resource sharing
- Para instalar:

```
npm install cors
```

- Para usar no código:

```
const cors = require('cors');
app.use(cors());
```

# Outras ações HTTP

- `app.get(caminho, funcao)`
- `app.post(caminho, funcao)`
- `app.put(caminho, funcao)`
- `app.delete(caminho, funcao)`
- `app.patch(caminho, funcao)`

# Propriedades da requisição (`req`)

- O objeto `req` tem várias propriedades que trazem informações sobre a requisição
  - `req.hostname`
  - `req.path`
  - `req.method`
  - `req.headers`
- A propriedade `headers` traz o cabeçalho HTTP, conforme enviado pelo cliente:
  - `req.headers['user-agent']`
  - `req.headers.host`
  - `req.headers['accept-language']`

# Parâmetros da URL

- Para obter parâmetros a partir da URL, use **`req.params`**

```
app.get('/usuario/:nome/:idade',
    (req, res) => {
        res.send(
            `Você é o usuário <strong>${req.params.nome}</strong>`
            + ` e tem ${req.params.idade} anos`
        );
    }
);
```

# Retornando um objeto JSON

- Para retornar um objeto JSON, basta usar `res.json(obj)` no lugar de `res.send(texto)`
- O Express automaticamente formata **`obj`** como uma string JSON válida e configura o cabeçalho apropriado na resposta

```
const express = require('express');
const app = express();

app.get('/dino',
    (req, res) => {
        res.json(
            {
                nome: 'Tyrannosaurus rex',
                periodo: "Cretáceo",
                peso: 14000,

                especimens:
                     [ 15, 13.2, 12.3],

                ref: {
                    id: 1245,
                    cod: 'trex'
                }
            }
        );
    }
);

app.listen(3000,
    () => console.log('Inicialização OK!')
);
```

# Requisição JSON - Cliente

- Para garantir o envio de JSON com o cliente, é necessário usar **`JSON.stringify`** nos dados enviados e mudar o **`contentType`**:

```
$.ajax({
    type: 'POST',
    url: 'http://localhost:3000/inserir',
    data: JSON.stringify(
        {
            usuario: 'fulano',
            nome: 'Fulano de Tal'
        }
    ),
    contentType: 'application/json',
    success: d => {
        // ...
    }
});
```

# Requisição JSON - Servidor

- Para receber parâmetros JSON no corpo da requisição, primeiro é necessário usar o middleware **`body-parser`**:

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

- Depois, o objeto JSON enviado pelo cliente pode ser recuperado em **`req.body`**

# Requisição JSON - Servidor

```
app.post('/inserir', (req, res) => {
    console.log(req.body);

    res.json({
        mensagem: 'Usuário inserido com sucesso',
        dados: {
            usuario: req.body.usuario,
            nome: req.body.nome
        },
        novoid: 123
    });
});
```