Programação & Serviços Web
React – Aula 1
2019/2020

INTRODUÇÃO AO EXPRESSJS

Aula 1 – Introdução ao ExpressJS

- Módulos Node
- Node e o Http
- Estrutura MVC
- Introdução ao ExpressJS
- Ficha 1

Relembrar Arquitetura 3 Camadas

Camada de Apresentação	
Camada de Lógica do Negócio	
Camada de Acesso de Dados	

Este é apenas uma possibilidade a nível de tecnologias.

Node.Js

JAVASCRIPT, BUT BETTER

Node.JS

- Ambiente Javascript programado no motor Javascript Chrome V8
- Usa um modelo baseado em eventos que não bloqueia o I/O
  - Faz com que seja bastante leve e eficiente
- Neste momento, só vamos falar da utilização do Node a nível de cliente-side, utilizando o React
  - A nível de server-side vai ser feito na UC TDW

Node Package Manager (NPM)

- Gere o ecossistema dos módulos / pacotes do node.

- Cada um destes pacotes contem:
  - Ficheiros JS
  - Package.json (manifest)

- Resumidamente, um módulo/pacote é algo **feito por terceiros** que pode **adicionar ao seu projeto e utilizar**. Por exemplo:
  - Um input do tipo calendário
  - Um loader para mostrar quando carrega as páginas
  - Sistema de Traduções
  - Entre outros

Node.JS / NPM

- O ficheiro package.json serve para:
  - Documentação dos pacotes que o projeto precisa
  - Permite especificar a versão dos módulos a ser usados projeto
  - Faz com que a build seja reproduzível, de forma a ser mais fácil de partilhar com o resto da equipa

- **Build** – Processo de compactar todos os módulos/código do projeto num só ficheiro de forma a poder ser colocado num site (PDF 6)

Instalação do Node.JS / NPM

- Fazer o download e instalação do node.JS (LTS) em
  - https://nodejs.org/en/download/

- Para verificar a instalação do Node, faça na linha de comandos

```
node -v
```

Inicializar package.json

- Para criar o **package.json** no projeto, basta abrir a linha de comandos na pasta e fazer:

```
npm init
```

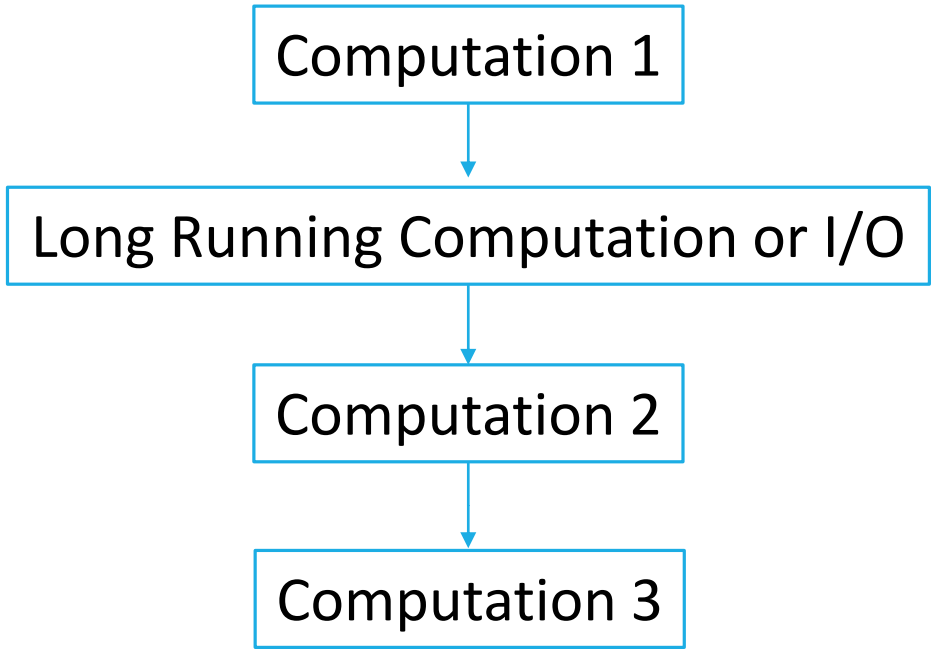
- São-lhe feitas umas perguntas relativamente ao projeto, informação essa que vai ficar gravada no ficheiro.
  - Se não souber o que introduzir em alguns dos campos (por exemplo do test), deverá deixar o campo vazio

Modulos Javascript

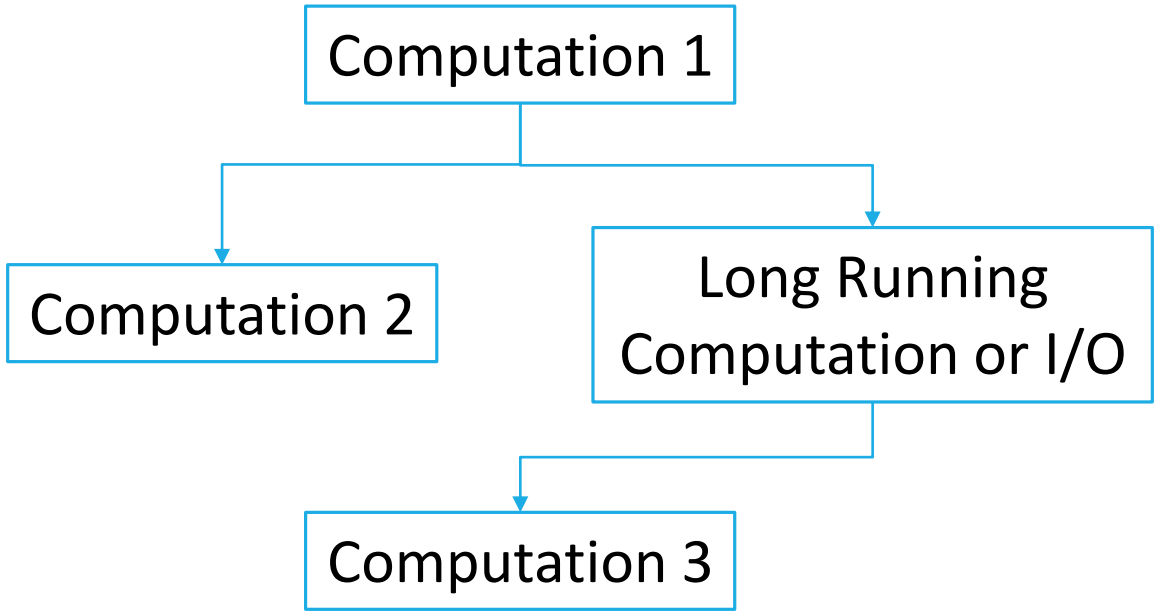
- O Javascript não é uma biblioteca
- Existem diversos módulos que oferecem funcionalidades extra / código já feito para que se possa utilizar no projeto

- Num projecto que utilize o Node, pode-se ir buscar módulos extra para utilizar de forma rápida e fácil.

Programação Assíncrona



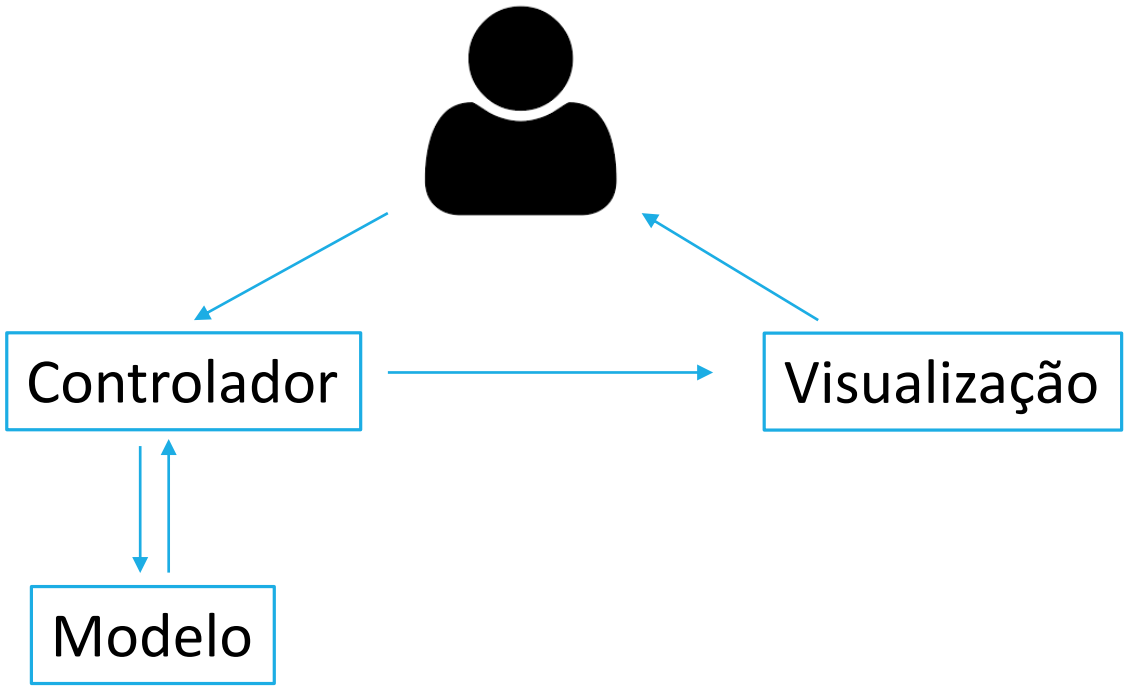
Computação Síncrona



Computação Assíncrona

Padrão MVC

- Arquitetura de engenharia de software
  - Isola a parte lógica da interface do utilizador
  - Permite o desenvolvimento, o teste e a manutenção em separado



Controlador

- Recebe o input do utilizador, gere a informação e envia para o modelo.

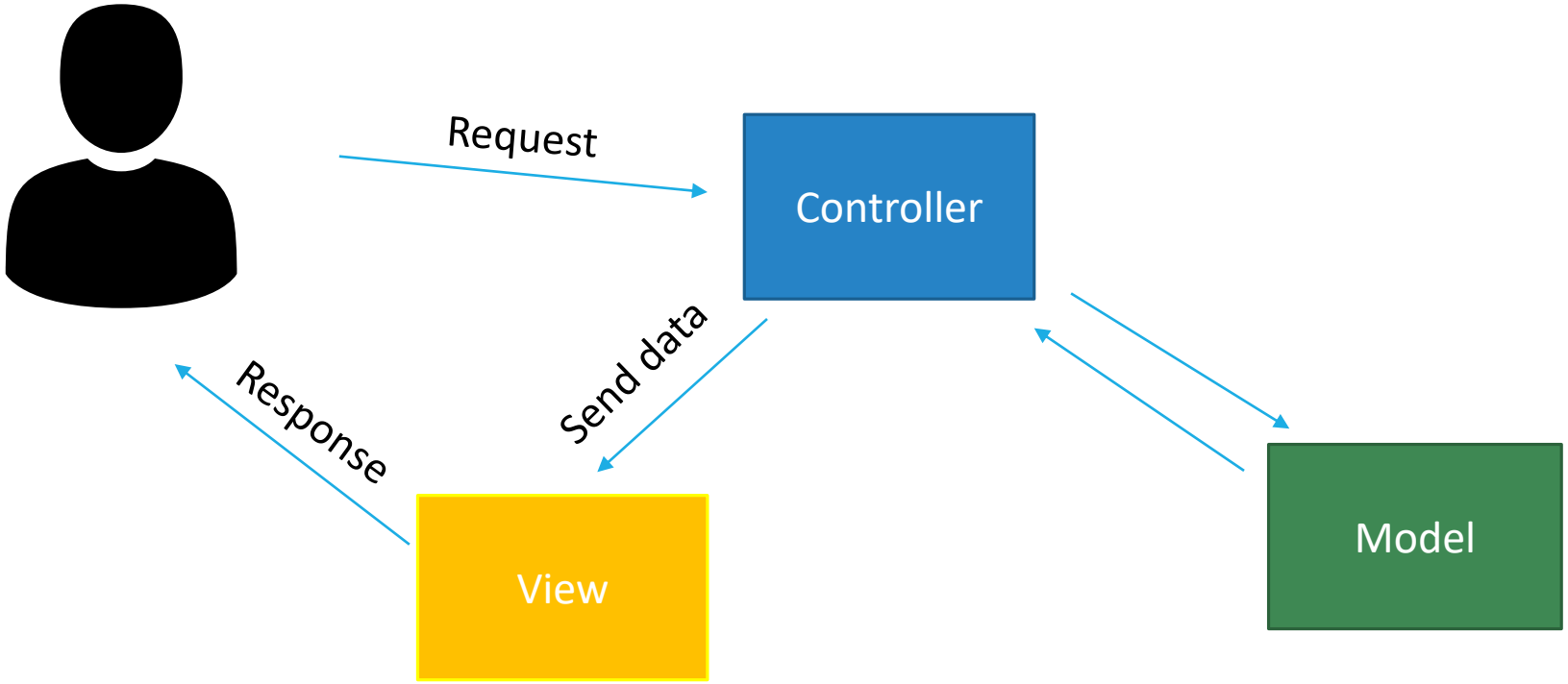
Modelo

- Gere a informação da aplicação.
- Responde aos requests para a informação sobre o seu estado
- Responde a instrução de mudança de estado (do controlador)

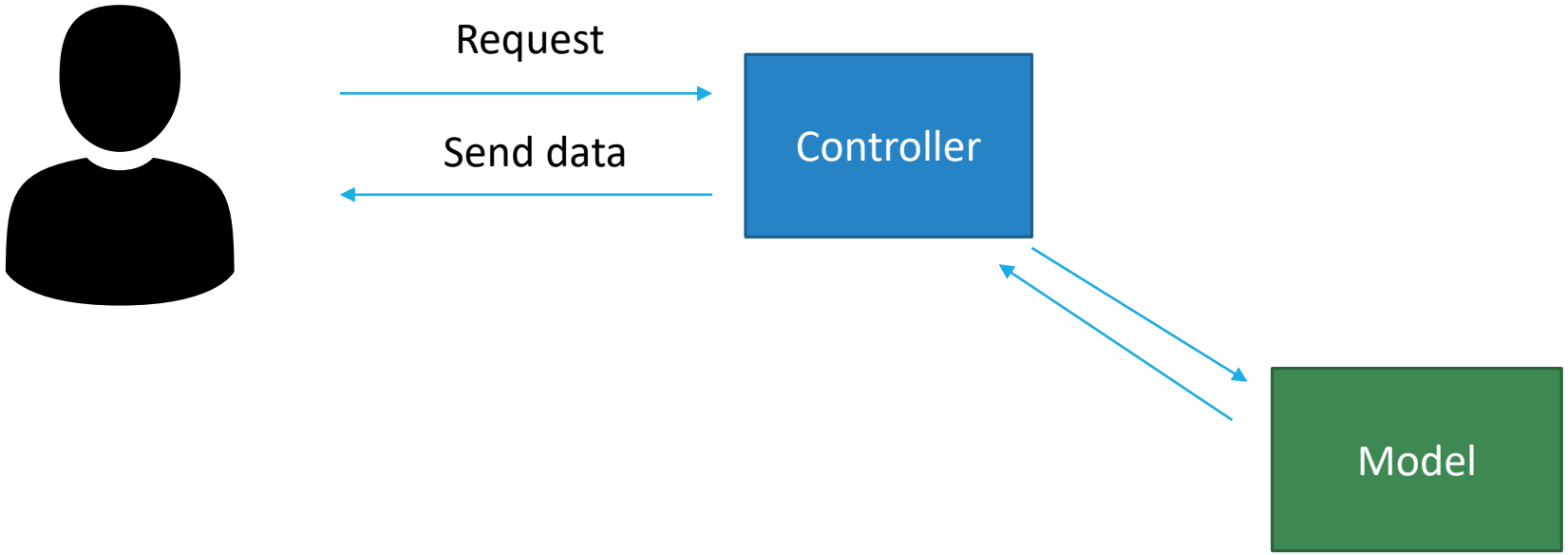
View

- Renderiza o modelo numa forma fácil de interação, tipicamente uma interface para o utilizador
- Várias views podem existir para um único controlador/modelo, com diferentes objetivos

Padrão MVC



Neste caso do ExpressJS



O servidor apenas devolver informação
Não existe nenhuma **View**

ExpressJS

ExpressJS

- **Express**: Framework minimalista de Node.Js, muito rápida
- Oferece um conjunto de características robustas
- Existem muitos middlewares (3rd party) para estender as funcionalidades
- Para instalar o ExpressJS,
  1) criar uma pasta;
  2) criar o package.json — npm init
  3) Instalar express — npm install express

```
"dependencies": {
  "express": "^4.17.1"
}
```

Ao abrirem o package.json, vêm pelo menos isto

Middleware para o Express

- Os middlewares oferecem uma série de funcionalidades do estilo plug-in (ligar e usar)
- Por exemplo:
  - **Morgan**, para fazer *logging*

```
var morgan = require("morgan");
app.use(morgan("dev"));
```

- Se pretender servir recursos estáticos a nível web

```
app.use(express.static(__dirname + "/public"));<
```

__filename e **__dirname** devolve o caminho completo do ficheiro e pasta do módulo actual

Começar um projeto

- Cada projeto deverá ter **o ficheiro index.js**
  - Este ficheiro é um ficheiro **JavaScript**. Este ficheiro **não faz** com que seja um **projeto ExpressJS.**
- Para transformar a pasta num projeto Express deverá

1) Fazer o require da biblioteca expressJS (não esquecer de instalar)

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

2) Declarar variável que utilizará o express

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

Começar um projeto

3) Indicar a porta

```
const port = 4000;
```

O react usava a porta 3000, portanto é melhor evitar essa porta

4) Definir um serviço standard (para a raiz, por exemplo)

```
app.get('/', (req, res) => { res.send('Olá Mundo!')} )
```

2 Parâmetros (objetos):
Req – Request
Res – Resposta

O **res.send** só devolve **strings**. No caso de querer enviar um número, tem de converter para **string**

5) Meter a aplicação à escuta

```
app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Iniciar Servidor

6) Ligar o Servidor (usando o Terminal)

```
node index.js
```

```
Example app listening on port 4000!
```

Se, neste momento, abrirem o url em **localhost:4000**, vão ver a mensagem "Olá Mundo!"

Na realidade, estão a fazer um GET ao URL de raiz

Cada endpoint no expressJS é uma função e podemos fazer o que quisermos lá (usar, declarar variáveis, etc)

Ao contrário do React, o ExpressJS não abre o browser automaticamente

POST, PATCH, DELETE

- O código para estes 3 tipos de requests é muito semelhante.

```
app.get('/', (req, res) => { res.send('Olá Mundo!')} )
```

Em vez de **.get**, temos **.post**, **.patch**, **.delete**

No caso de querermos utilizar os valores enviados no url (/**:id**), podemos acede-los via req.params

```
app.get('/:id', (req, res) => { console.log(req.params.id) });
```

```
app.get('/:private_Id', (req, res) => { console.log(req.params.private_id) });
```

Submeter com o POST

- Para os valores do body, acede-se via req.body

- Visto que, para já, se vai usar o JSON, é necessário adicionar o seguinte código (logo após a declaração do app)

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

Submeter com o POST

- Imagine que queria enviar para o servidor a sua idade. Um exemplo do JSON a enviar é:

```
{
   “idade": “18"
}
```

- Para receber este campo no POST, usa-se o body:

```
app.post('/', (req, res) => {
   console.log(req.body.idade);
   return res.send("Idade Recebida: " +
req.body.idade);
})
```

Não esquecer de enviar nos headers:

Content-Type: application/json

Ficha 1 - Moodle

Para submissão no final da aula