Trabalho prático PI

45365 David Fidalgo

45367 José Coelho

Índice

Introdução

Neste trabalho fizemos, com recurso à API Board Games Atlas, uma API que fornece uma lista de jogos populares, pesquisa de jogos por nome e gestão de listas de jogos a que chamamos grupos. Cada utilizador tem os seus próprios grupos onde pode remover e adicionar jogos. Para utilizar a API o utilizador tem que se inscrever como utilizador, caso não o tenha feito, e autentificar-se fazendo login.

Fizemos ainda como complemento à API uma aplicação que utiliza os serviços da API e as apresenta de uma forma mais "*user friendly*".

API

Para iniciar o servidor é apenas necessário correr o comando **npm start**. A API está dividida em módulos:

Server

O server comunica com o servidor HTTP. É utilizado o modulo express para fazer o routing e definir recursos estáticos. Associado a cada rota está uma função de um outro modulo chamado web-api.

Web-api

Este modulo faz a abstração das respostas e pedidos HTTP para os restantes módulos, recebe os argumentos de cada rota e envia-os para um outro modulo, Service, após resolução dos pedidos por esse modulo, interpreta-os e envia a resposta, seja ela bem-sucedida ou não.

Service

O Service possui a logica da API, caso algo não seja valido este invalida-o e retorna uma resposta de erro imediatamente, caso contrário utiliza um modulo que interaja com a base de dados ou com a API BGG.

Bases de dados

Neste projeto usamos bases de dados ellastic para guardar, utilizadores e grupos. Os utilizadores ficam num índex chamado users e os grupos num chamado group. Para a aplicação funcionar **é necessário tem o ellastic a correr em background.** É ainda importante mencionar que para cada interação com a base de dados de um grupo é feito uma verificação para ver se o utilizador pode aceder a esse grupo, ou seja se o grupo lhe pertence.

Data

Este módulo faz pedidos à API BGG para obter listas de jogos, filtra-a a informação desejada e retorna-a ao Service.

Web App

Para fazer a app utilizamos o modelo MVC.

Model

O model faz fetch à API que desenvolvemos e retorna a promise com o valor desejado.

View

A view atualiza os campos previamente carregados no índex conforme a rota corrente.

Controller

O controller simplesmente chama o model com os parâmetros desejados.

Após desenvolvimento destes módulos desenvolvemos um modulo chamado routes que contem diversos objetos cujo nome é uma route e que liga cada view a um controller.
Para fazer o routing da web app é utilizado um hash router que devolve uma route e executa o seu controller.

Conclusão

A realização deste trabalho foi bastante consolidadora da matéria lecionada das aulas e deu uma perspetiva do que é necessário para o desenvolvimento de uma Web app e uma API. Percebemos a importância da modelação de um projeto melhor, sendo que por vezes a falta disto nos complicou um bocado o trabalho. O relatório ficou um pouco fraco de modo que vamos ter que trabalhar nisso.

É importante referir que para correr a aplicação é necessário:

- **Porto 8000**
- **Ter o ellastic a correr em background**
- **Executar o comando npm start**