Programação na Internet

LI52D – Miguel Gamboa

Relatório da Terceira Fase

André Gaudêncio nº42204

Nuno Conceição nº42195

Guilherme Arede nº41548

Índice

Introdução

Este trabalho foi realizado na cadeira de Programação na Internet. Tem como objetivo principal aprender a trabalhar na área *Back End* com a ajuda do Node.js, um interpretador de código JavaScript que funciona do lado do servidor, e também *Front End*.

Para isso, foi-nos pedido a realização de um projeto que dá pelo nome de COIMA (Chelas Open Internet Movie Application) com a duração de três fases. Aplicação esta que visa apresentar informação sobre filmes, atores, realizadores, entre outros provenientes da API Movie Data Base(https://developers.themoviedb.org/).

As fases propostas dividem-se nas seguintes operações e funcionalidades:

Fase 1

- Requisitos Funcionais
  - Pesquisa
  - Detalhes de um filme
  - Lista de personagens de um filme
  - Participações de um ator
  - Detalhes de uma pessoa
- Requisitos Não Funcionais
  - Request – Realização de pedidos HTTP
  - Handlebars – View Engine
  - Debug – Mensagens de debug
  - Nodeunit – Testes unitários

Fase 2

- Requisitos Funcionais
  - Registo e autenticação de utilizadores.
  - Gestão de listas de filmes:
- Requisitos Não Funcionais
  - Persistência de Dados – CouchDB
  - Refactored – Estruturar melhor o projeto

Fase 3

- Requisitos Funcionais
  - Comentários a filmes
  - A página HTML que apresenta as listas filmes de um utilizador deve suportar paginação.

- Requisitos Não Funcionais
  - Adição e remoção de filmes a uma lista
  - Alteração do nome da lista na página que mostra todas as listas de filmes de um utilizador
  - Outras situações que considere adequadas para usar esta técnica.

Projeto

FASE 1

Numa primeira fase a implementação, foi necessário organizar a estrutura da aplicação de modo a ter uma boa separação do código consoante a tarefa a que cada ficheiro ou conjunto de ficheiros se destinam.

Ficou decidido que a separação iria incidir sobre o modelo de negocio, o domínio, a vista, e posteriormente, seria adicionada mais uma pasta relativa aos testes.

Começando pelo modelo de negocio, eram necessárias 3 funcionalidades principais que incidiam sobre a pesquisa de filmes, a procura de um filme particular a partir do seu id e a procura de um ator a partir do seu id.

Para qualquer dessas operações é necessário realizar pedidos à API que neste caso, resultam num documento no formato JSON. Por essa razão implementámos uma função de request que transforma o resultado em JSON e envia-o no callback.

Para a obtenção da lista de filmes no search é recebida a query que o utilizador inseriu e a pagina à qual se quer aceder, para motivos de paginação. É feito um pedido à API e retornado o resultado.

O mesmo acontece para o getActor e getMovie, mas estes têm ainda uma característica que é a necessidade de fazer mais do que um pedido, e foi decidido por motivos de performance que esses pedidos fossem feitos em paralelo, e para isso foi usada uma função.

A função que realiza pedidos em paralelo recebe um array de URLs e faz os pedidos, chamando o callback quando recebe todas as respostas, sendo a verificação feita por numero de respostas obtidas.

Temos ainda uma funcionalidade de cache que é usada para evitar pedidos repetidos quando é pesquisado algum filme ou ator por id.

Essa cache baseia-se num objeto que guarda o resultado dos pedidos, e quando for feito um pedido para o mesmo URI é retornado o objeto que foi guardado como propriedade do objeto cache, ao invés de fazer o pedido de novo.

O domínio é usado para representar, com objetos mais simplificados, os dados retornados pela API.

As vistas são ficheiros HTML que representam os objetos do domínio e/ou as suas manipulações, ou representam outro conteúdo relevante para a aplicação.

Fase 2

Na fase 2 um dos primeiros objetivos foi usar o módulo do Express para a organização da aplicação.

E para isso foi usado numa primeira instancia o Express Generator, que gera o template do projeto, e adaptar o nosso código para este novo desenho, não foi necessário modificar a vista, o modelo de negocio, nem o domínio, sendo apenas necessário redefinir as rotas, e obviamente adicionar os requisitos do Express, como alguns módulos que este requere.

O segundo requisito era a adição de autenticação e utilizadores, e para isso foi usado o passport, que é um middleware de autenticação. E esse middleware adiciona ao req uma propriedade que representa o utilizador que esta autenticado no momento.

Para registarmos ou autenticarmos utilizadores foi usada a base de dados CouchDb, que funciona com pedidos HTTP para modificar ou obter o seu conteúdo.

Foi para tal adicionada uma nova rota às rotas da aplicação, e também um novo serviço, que neste caso, recebe o objeto de utilizador e envia-o para a base de dados. Também faz a autenticação do utilizador, que neste caso se baseia em verificar se o username e password correspondem.

Foi ainda pedido que o utilizador pudesse ter listas de filmes, por isso foram criadas novas rotas com esse objetivo, e correspondentemente adicionado um serviço. Novamente para os dados serem persistentes foi usada a base de dados CouchDb. A implementação do serviço é parecida com os outros serviços na medida em que realiza pedidos a um servido HTTP, que neste caso é a base de dados, no entanto tem algumas diferenças importantes, que são do âmbito do problema em si.

Uma referencia para a lista fica guardada no objeto do user, mas não o seu conteúdo, este fica guardado noutro documento por motivos de performance. O facto de não carregar todos os elementos de todas as listas sempre que o utilizador realizar alguma operação que envolva autenticação ou interação com alguma lista, foi o que de facto nos fez implementar desta forma.

Fase 3

Nesta fase foi necessário novamente adicionar novos serviços e rotas para realizar as operações necessárias.

Neste caso eram comentários de filmes que deveríamos suportar, e para isso novamente foi necessário a CouchDb para guardar os comentários de forma persistente.

Uma das dificuldades foi o facto de um comentário poder ter respostas e essas respostas terem outras respostas e assim sucessivamente. Foi necessário organizar os dados num array "recursivo" porque não sabíamos a profundidade que cada comentário iria ter, e o objeto representante da resposta iria ter os mesmos campos, e iria ter a possibilidade de ter uma resposta.

Para realizar a vista também foi necessário pesquisar um pouco mais sobre Handlebars porque novamente não sabíamos a profundidade de cada comentário por isso não poderíamos usar um helper de iteração. A resolução desse problema foi novamente a recursão, ou seja, na vista de comentários usar partials para representar as respostas, e esses partials chamarem-se a eles próprios se essa resposta tivesse uma resposta e assim sucessivamente.

Quanto à pagina do utilizador, esta é privada, por isso cada vez que o utilizador faz um comentário (não uma resposta) a um filme, o objeto correspondente a esse utilizador é populado com os dados dessa resposta. Outra solução seria o utilizador ter uma referencia para um documento que teria as suas respostas, mas por simplicidade não foi feito dessa maneira.

Após a aplicação estar concluída enviámo-la para um servidor externo, neste caso escolhemos os serviços do cloudno.de porque fornecem um serviço gratuito para uma aplicação pequena, como a nossa, e também bastante importante, o facto de ter suporte interno para o couchDB que é a base de dados que usamos.

Para fazer o deploy da aplicação para o servidor foi necessário criar um ficheiro de boot, e mudar os caminhos referentes à base de dados, e aos seus documentos.

O URL da aplicação é http://iselpi20172018coima.cloudno.de/

Conclusão

Para a realização deste trabalho foi necessário adquirir várias aptidões de programação.

Primeiro que tudo, foi necessário ganhar conhecimento a nível da linguagem JavaScript, bem como a sua enorme facilidade e capacidade em termos de programação.

Com maior relevância, foi bastante importante também aprender o conceito do Node.js como interpretador de código do lado do servidor, ou seja, *Back End*. Entender como é possível criar rotas para ajudar a criar um servidor bem estruturado, funcional e eficiente.

Foram ganhas habilidades também a nível visual, por parte do Handlebars e CSS. Estes foram de grande utilidade para ser possível embelezar a página HTML aos olhos do utilizador. E ao mesmo tempo facilitar a construção da mesma para o programador.

Outra noção muito marcante neste projeto foi a persistência dos dados. Ao utilizar este conceito, foi possível manter estado da aplicação para cada conta de utilizador, tornando possível cada um ter os seus favoritos, comentários e página de perfil. No fim de contas, torna-se como uma comodidade adicional bastante utilizada e que visa tornar privado a informação do utilizador.

Aprendemos também a trabalhar com o CouchDB. Uma base de dados online que nos permitiu guardar dados persistentes na aplicação em forma de objetos JSON, visando facilitar guardar objetos JavaScript.

Por fim aprendemos a trabalhar com pedidos AJAX. Útil para realizar alterações dinâmicas às páginas HTML através de código JavaScript cliente.

Para complementar isto tudo, este projeto teve ser muito bem organizado. Podendo dizer assim que cada componente aqui descrito teve de ser colocada no seu package respetivo, como pode ser visto na descrição feita acima no capítulo Projeto.