Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Aplicação BORGA (Board Games Application)

Autores: 48280 André Jesus
48287 Nyckollas Brandão
48309 André Santos

Relatório para a Unidade Curricular de Introdução à Programação na Web da Licenciatura em Engenharia Informática e de Computadores

Professor: Engenheiro João Trindade

22– Janeiro – 2022

<< Esta página foi intencionalmente deixada em branco >>

Índice

1 Introdução

No âmbito da disciplina de Introdução de Programação na Web, foi-nos pedida a realização de uma API desenvolvida com tecnologia Node.js, que maioritariamente é capaz de gerir grupos de jogos para vários usuários.

O processo de desenvolvimento desta aplicação implicou a sua testagem através de testes unitários e de integração. A respetiva documentação está presente neste documento. Este trabalho foi divido em 3 partes distintas, tornando necessário a reformulação de código e boas práticas de programação.

A API inclui também uma interface web e faz uso de *HTTP* para tal e para o armazenamento dos dados usando *ElasticSearch*.

2 Estrutura da Aplicação

A aplicação forneçe uma API que segue os princípios REST, através de uma interface web, que suporta várias funcionalidades. Toda a informação fornecida por tais funcionalidades é recolhida a partir do website Board Game Atlas, fazendo uso da sua Web API para o efeito.

As funcionalidades suportadas são as seguintes:

- Obter uma lista dos jogos mais populares;
- Pesquisar jogos pelo nome;
- Gerenciar grupos de jogos:
    - Criar grupos, fornecendo o seu nome e descrição;
    - Editar grupos, alterando o seu nome e descrição;
    - Listar todos os grupos do utilizador;
    - Apagar um grupo;
    - Obter os detalhes de um grupo (nome, descrição e nomes dos jogos do mesmo);
    - Adicionar um jogo a um grupo;
    - Remover um jogo de um grupo;
- Criar um novo utilizador.

Tal como referido anteriormente, ao longo do trabalho foram adicionadas novas funcionalidades, e como tal foi necessário ter uma boa organização do software.

A aplicação tem assim vários módulos, que comunicam entre si, sendo uns dependentes de outros. De aqui em seguida serão referidos cada um desses módulos e uma breve explicação sobre o seu funcionamento.

A Figura 1 exemplifica as dependências entre os vários módulos.

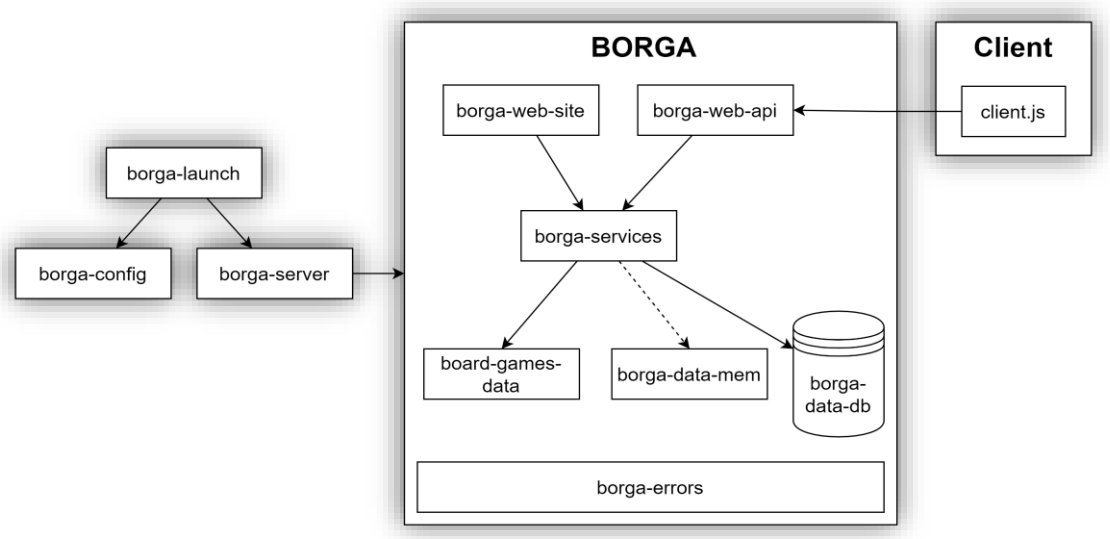

Figura 1 – Diagrama de módulos do projeto

2.1 Board-games-data

O módulo ***board-games-data*** é o ponto de partida para toda a aplicação. É a partir deste que é feito o acesso à *Board-Games-Atlas API*, que fornece toda a informação necessária sobre os jogos para a execução da aplicação.

Também é neste módulo que se apresentam as seguintes funcionalidades/métodos:

- ***doFetch***: realiza um pedido GET para o URI especificado;
- ***getPopularGames***: procura os 20 jogos de tabuleiro mais populares, de acordo com a API anteriormente referida;
- ***searchGamesByName***: faz uma pesquisa dos jogos existentes na API para um certo nome introduzido pelo utilizador.

Este módulo contém também algumas constantes referentes a URI´s de acesso à *Board-Games-Atlas API*, bem como outros métodos de auxílio para a criação de objetos *JavaScript*, que simulam a informação que um jogo deverá conter.

2.2 Armazenamento de dados

Para o armazenamento e manipulação de dados, foram criados dois módulos, ***borga-data-mem*** e ***borga-data-db***, sendo que um que usa armazenamento de dados em memória, e o outro armazenamento numa base de dados *ElasticSearch*, respetivamente.

Ambas implementam exatamente os mesmos métodos, sendo possível escolher qual das duas é utilizada pelo módulo ***borga-services***, e por sua vez, pela aplicação.

Os métodos implementados são os seguintes:

- ***createNewUser****:* cria um novo utilizador, passando o seu nome, ID e *password*;
- ***tokenToUserId****:* retorna o ID do utilizador associado a um certo *token*;
- ***createGroup****:* adiciona um novo grupo a um utilizador;
- ***editGroup****:* edita o nome ou a descrição de um grupo;
- ***listUserGroups****:* lista todos os grupos de um utilizador;
- ***deleteGroup****:* elimina um grupo;
- ***getGroupDetails****:* retorna o nome, a descrição, o ID e a lista de jogos de um grupo;
- ***addGameToGroup****:* adiciona um jogo a um grupo;
- ***removeGameFromGroup****:* remove um jogo de um grupo;

2.2.1 Borga-data-mem

Este módulo contém duas variáveis globais que representam objetos que contêm informação acerca dos utilizadores criados, e dos jogos adicionados pelos mesmos. Existe ainda uma terceira variável global que garante a associação entre *tokens* e Ids dos utilizadores. Todos os métodos implementados inserem ou obtêm dados destas três variáveis.

2.2.2 Borga-data-db

O armazenamento na base de dados *ElasticSearch* é realizado através de *HTTP*. Para inserção e atualização de dados são utilizados os métodos POST e PUT e para a remoção destes é utilizado o método DELETE. Qualquer consulta de informação será realizada através de um GET.

Uma base de dados *ElasticSearch* é organizada em vários índices que contêm documentos. Antes de cada índice existe um dos dois prefixos, "*prod*" para uma normal utilização da API e "*test*", para a execução dos testes de integração.

Os índices base (quando não há nenhum grupo criado) são apenas "*users*" e "*tokens*".

No índice "*tokens"*, cada documento contém o ID do utilizador e é identificado pelo valor do *token*.

No índice "*users"*, cada documento contém o nome do utilizador e a *password* (em *hash*) e é identificado pelo ID do mesmo.

No momento de criação do primeiro grupo de um utilizador, é criado um novo índice apenas para todos os grupos do mesmo, como por exemplo: "prod_guestid_groups" é criado para o utilizador com ID "guestid".

Neste índice, cada documento corresponde a um grupo, que contém um nome e descrição e é identificado pelo ID do mesmo.

Quando se adicionar um jogo a um grupo, é criado um novo índice apenas para todos os jogos do grupo, como por exemplo: "prod_users_guestid_groups_db70a043-d73b-4f82-b922-f259cd1d73eb_games" é criado para o grupo com ID "db70a043-d73b-4f82-b922-f259cd1d73eb" do utilizador com ID "guestid".

Neste índice, cada documento corresponde a um jogo, que apenas contém o nome do jogo e é identificado pelo ID do mesmo. Para obter uma lista dos jogos do grupo, basta fazer uma pesquisa neste índice.

2.3 Borga-errors

O módulo ***borga-errors*** é responsável pela criação e listagem dos vários erros que poderão ocorrer durante a utilização da API.

Os erros possíveis são:

- ***FAIL***: tipo de erro geral, lançado quando algo não corre como esperado;
- ***BAD_REQUEST***: lançado quando existe um pedido incorreto, por parte do utilizador;
- ***NOT_FOUND***: lançado quando algum recurso não existe em memória ou quando algum URI não é reconhecido;
- ***ALREADY_EXISTS***: lançado em tentativas de criação de dados já existentes;
- ***EXT_SVC_FAIL***: lançado quando existem erros exteriores ao servidor, na conexão à Board Game ATLAS API;
- ***UNAUTHENTICATED***: lançado quando ocorre uma falha na autenticação do utilizador, devido a dados mal introduzidos por parte deste;
- ***MISSING_PARAM***: lançado quando, durante o processo de autenticação, o utilizador não especifica password ou ID.

Cada erro é composto pelos seguintes parâmetros:

- ***code***: código do erro;
- ***name***: nome do erro;
- ***info***: breve descrição do tipo de erro;
- ***message***: descrição do erro que ocorreu ou em que elemento o erro ocorreu.

2.4 Borga-data-services

O módulo ***borga-data-services*** é o módulo central da API. É a partir deste que se torna possível verificar depêndencias entre os vários módulos.

A este módulo é passado como parâmetro o módulo de armazenamento de dados a usar, sendo possível alternar entre tipos de armazenamento sem ter de alterar a lógica de funcionamento deste módulo.

Todos as funcionalidades da aplicação, ou seja, dos módulos da API e do WebSite, são implementados neste módulo, onde é feita a união dos dados armazenados e dos obtidos a partir do exterior.

A essas funcionalidades, este módulo também fornece métodos de controlo de erros:

- ***checkBadRequest****:* valida o pedido enviado, verificando os parâmetros e o corpo desse pedido;
- ***checkCredentials****:* verifica se um ID de um utilizador e uma *password* estão associados;
- ***checkAuthentication****:* verifica se um ID de um utilizador e um *token* estão associados.

2.5 Borga-web-api

O módulo ***borga-web-api*** implementa todos os *end-points*/rotas HTTP da aplicação, através do uso do módulo *Express.js*.

É este módulo e o módulo ***borga-web-site*** que recolhem todos os dados fornecidos pelo utilizador, seja a partir dos parâmetros ou do corpo do pedido, e invoca os vários métodos que a aplicação suporta e que se encontram presentes no módulo *borga-data-services*.

2.6 Borga-server

O módulo ***borga-server*** é responsável pela junção das módulos *borga-web-api*, *borga-web-site* e *borga-services* e pela criação e configuração da aplicação *Express*.

2.7 Borga-web-site

O módulo ***borga-web-site*** implementa a *User Interface* através do uso de *HTML* e *CSS*.

Este contém todas as operações e rotas *HTTP* disponiveis no *WebSite* da aplicação, renderizando os ficheiros Handlebars criados para cada página do mesmo.

Para obter uma melhor apresentação, recorreu-se a *CSS* e a *Bootstrap*.

2.8 Borga-launch

O módulo ***borga-launch*** inicializa a aplicação, acedendo às configurações das mesma através do módulo *borga-config*, e iniciando o servidor.

2.9 Borga-config

O módulo ***borga-config*** contém a informação relevante para a aplicação, como o URL da base de dados *ElasticSearch* e a informação do utilizador *guest* (*ID*, *nome*, *password* e *token*), criado com o propósito da realização de testes.

3 Utilização da aplicação

Este capítulo contém as instruções necessárias para usar a aplicação e os respetivos testes.

Inicialmente é necessário ter o seguinte software instalado:

1. Node.js
2. JDK (é aconselhado a utilização do *Amazon Correto JDK*) e *ElasticSearch*

É também necessário ter uma variável de ambiente com o nome 'ATLAS_CLIENT_ID', contendo um *token* disponibilizado pela API do *Board Game ATLAS*, quando criada uma conta.

Para iniciar a aplicação e os respetivos testes basta seguir os seguintes passos:

1. Clonar o repositório no GitHub para o armazenamento local do computador;
2. Abrir o terminal na diretoria principal do projeto e executar o seguinte código para instalar os módulos utilizados pela aplicação;

```
npm install
```

3. Iniciar a base de dados *ElasticSearch*;
4. Para iniciar a aplicação executa-se o seguinte comando:

```
npm start
```

5. Para executar os testes unitários executa-se o seguinte comando:

```
npm test
```

6. Para executar os testes de integração executa-se o seguinte comando:

```
npm run integration-tests
```

4 Conclusão

Em suma, podemos concluir que todos os módulos do projeto foram implementados com sucesso, superando algumas dificuldades graças à pesquisa realizada sobre o problema e à ajuda do professor.

Esta projeto proporcionou-nos a oportunidade de utilizar vários conhecimentos que adquirimos nas aulas, implementando diversos aspetos do desenvolvimento de software na Web.

Adquirimos conhecimentos relativos ao protocolo *HTTP*, à construção de uma aplicação *Express.js* e à programação em *JavaScript*, tanto ao nível do servidor, como ao nível do cliente. Também aprendemos a utilizar *HTML*, *CSS*, *Bootstrap*, *Handlebars* e outros softwares já enumerados neste documento.

A área do desenvolvimento de software cativa-nos imenso e nesta unidade curricular pudemos a aprender e a desenvolver conhecimentos que serão úteis tanto no nosso futuro académico, como profissional, sendo que o desenvolvimento na Web é uma das áreas em que temos mais interesse.

5 Software utilizado

Segue-se uma lista do software utilizado na realização deste trabalho, juntamente com uma breve descrição sobre a sua utilização:

- ***Visual Studio Code***: edição de código;
- ***Git/GitHub***: controlo de versões e armazenamento do projeto num repositório;
- ***diagrams.net***: criação de diagramas;
- ***Heroku***: *deploy* da aplicação;
- ***Postman***: criação de uma coleção de pedidos *HTTP* para testar funcionalidades da API e manuntenção manual da base de dados *ElasticSearch*;
- ***JavaScript*:** principal linguagem utilizada;
- ***Node.js***: ambiente de execução de *JavaScript* em *back-end*;
- ***Swagger:*** documentação da API;
- ***Handlebars (HTML), CSS e Bootstrap***: linguages utilizadas para a construção da parte visual do *WebSite.*

6 Referências

[1] *Board Game Atlas. (n.d.). GET /API/search. Board Game Atlas. Retrieved January 21, 2022, from* https://www.boardgameatlas.com/api/docs/search

[2] *Free and open search: The creators of Elasticsearch, Elk & Kibana. Elastic. (n.d.). Retrieved January 21, 2022, from* https://www.elastic.co/

[3] *Moodle 2021-2022. pt. (n.d.). Retrieved January 21, 2022, from* https://2122moodle.isel.pt/

[4] *5.X API. Express 5.x - API Reference. (n.d.). Retrieved January 21, 2022, from* http://expressjs.com/en/5x/api.html

[5] *Node.js. (n.d.). Documentation. Node.js. Retrieved January 21, 2022, from* https://nodejs.org/en/docs/

[6] *ISEL-LEIC-IPW. GitHub. (n.d.). Retrieved January 21, 2022, from* https://github.com/isel-leic-ipw

[7] *Handlebars. (n.d.). Retrieved January 21, 2022, from* https://handlebarsjs.com/api-reference/

[8] *Eloquent javascript3rd edition (2018). Eloquent JavaScript. (n.d.). Retrieved January 21, 2022, from* https://eloquentjavascript.net/

[9] *Heroku. Cloud Application Platform. (n.d.). Retrieved January 21, 2022, from* https://www.heroku.com/

[10] *Getting started · jest. Jest RSS. (n.d.). Retrieved January 21, 2022, from* https://jestjs.io/docs/getting-started

[11] *Documentation. Swagger Documentation. (n.d.). Retrieved January 21, 2022, from* https://swagger.io/docs/