**RELATÓRIO**

**Índice**

[Exercício 1 3](#_Toc27053839)

[Exercício 2 3](#_Toc27053840)

[Exercício 3 3](#_Toc27053841)

[Exercício 4 3](#_Toc27053842)

[Exercício 5 4](#_Toc27053843)

[Exercício 6 4](#_Toc27053844)

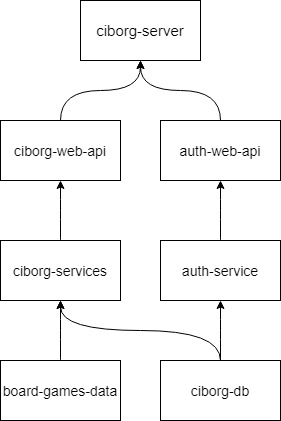
[Exercício 7 7](#_Toc27053845)

### Estrutura

A aplicação CIBORG é uma aplicação web que nos permite efectuar a pesquisa de informação sobre jogos de tabuleiro tendo como fonte de informação a *API Board Game Atlas*. Para além da pesquisa é possível organizar os jogos por grupos sendo para tal utilizada uma base de dados local *ElasticSearch*. Os grupos são criados por cada utilizador e como tal os utilizadores apenas tem acesso aos grupos por si criados, sendo isto assegurado através de um sistema de login.

Estrutura – Servidor

A aplicação é iniciada a partir do servidor e é nesta parte que toda a informação é obtida, seja informação vinda da *API* ou informação vinda da base de dados local.

 O servidor, iniciado em *ciborg-server*, recebe a informação da API da aplicação, *ciborg-web-api*. Para tal é passada a este módulo uma instância do *router* do módulo de *express* que é importado.

Em *ciborg-web-api* são mapeadas as rotas no *router* que lhe é passado. As rotas estão ligadas às várias funções que se encontram neste módulo que efectuam os pedidos ao módulo *ciborg-services*. As respostas são tratadas em *ciborg-web-api*, quer sejam respostas em que o pedido tenha sido efectuado com sucesso ou insucesso.

O módulo *ciborg-services­* serve como um ponto de junção de dados e gestão dos pedidos, pois é aqui que os pedidos são encaminhados para a base de dados ou para *API* *Board Game Atlas*. Este módulo encontra-se ligado a outros dois módulos de onde obtém a informação, *board-games-data* referente à informação que obtida através da *API Board Games Atlas* e *ciborg-db* referente à informação da base de dados local.

Os módulos *ciborg-db* e *board-games-data* servem para efectuar os pedidos às bases de dados. Os pedidos são efectuados utilizando *Promises*, através da importação do módulo npm referente.

Para além da obtenção da informação o servidor também efectua a gestão de autenticação dos utilizadores.  
O módulo *auth-web-api* efectua esta gestão utilizando o módulo *npm passport*. Aqui é gerido as sessões dos utilizadores, o acesso e saída e o registo de utilizadores. Este módulo tem efectua o registo das rotas referentes à autenticação no *router* que lhe é passado pelo servidor. Para além do *router* também recebe uma instância de *auth-service* e do módulo *npm express*.

Em *auth-service* é feito os pedidos de registo e obtenção de informação na base de dados local, sendo estes pedidos feitos ao módulo já referenciado anteriormente, *ciborg-db*.

O servidor também inicia toda a estrutura referente ao lado do cliente que é explicada abaixo.

Estrutura – Cliente

A estrutura do cliente é iniciada em *entry* onde o processo do cliente é iniciado. Aqui é carregado toda a parte visual da página utilizando o módulo *bootstrap*, o ficheiro css da aplicação, os *templates* referentes às várias páginas da aplicação bem como imagens.   
Neste módulo é tratado toda a lógica da página do cliente sendo que a partir de *hashes* presentes na página inicial, a página é reconstruída conforme o que utilizador pretender visualizar e tendo em conta os botões criados na página inicial. Todas as páginas acabam por utilizar esta restruturação da parte visual através de *hashes* em vez de âncoras.

Este módulo utiliza ainda os módulos *games-data* para obtenção de alguns dados directamente da *API* *Board Game Atlas*. É utilizado aqui também os vários *handlers* criados, *games* que efectua a gestão do botão de procura e listagem dos jogos. Os *handlers login* e *logout* efectuam o tratamento dos pedidos de autenticação ao servidor tanto para acesso como para saída.

No módulo *games-data* que *entry* também utiliza, é efectuado os pedidos de dados ao servidor e passada a informação a *entry* conforme necessário.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

### Documentação API

## Obtain most popular games

* Request:
  + Method: GET
  + Path: /games/top
  + Body: none
* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: aplication/json
    - Body example:
  + {
  + "games": [
  + {
  + "id": 1,
  + "name": "game1",
  + "description": "description of game 1"
  + },
  + {
  + "id": 2,
  + "name": "game2",
  + "description": "description of game 2"
  + },
  + ...
  + ]

}

## Obtain a specific game

* Request:
  + Method:GET
  + Path: /games/:name
    - Path parameters:
      * name - game name
  + Body: none
* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: application/json
    - Body:
  + {
  + "id": 1,
  + "name": "game1",
  + "description": "description of game 1"
  + "games": [
  + {
  + ...
  + }
  + ]

}

* Errors:
  + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/games/game1"

}

## Obtain all groups

* Request:
  + Method: GET
  + Path: /groups
  + Body: none
* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: aplication/json
    - Body example:
  + {
  + "groups": [
  + {
  + "id": 1,
  + "name": "group1",
  + "description": "description of group 1"
  + },
  + {
  + "id": 2,
  + "name": "group2",
  + "description": "description of group 2"
  + },
  + ...
  + ]

}

## Obtain a specific group

* Request:
  + Method:GET
  + Path: /groups/:groupId
    - Path parameters:
      * groupId - group identifier
  + Body: none
* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: application/json
    - Body:
  + {
  + "id": 1,
  + "name": "group1",
  + "description": "description of group 1",
  + "games": [
  + {
  + ...
  + }
  + ]

}

* Errors:
  + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/groups/1"

}

## Create a Group

* Request:
  + Method:POST
  + Path: /groups
  + Content-Type: application/json
  + Body:

{

"name": "group1",

"description": "description of group 1"

}

* Response:
  + Success:
    - Status code: 201 Created
    - Headers:
      * Location: /groups/1
    - Content-Type: application/json
    - Body example:
  + {
  + "status" : "Group created",
  + "uri": "/groups/1"

}

* Errors:
  + Status Code: 400 Bad Request
  + Body:
  + {
  + "error": "Bad request",
  + "errorDetail": "Parameter x error"
  + "uri": "/groups/1"

}

## Update a Group

* Request:
  + Method: PUT
  + Path: /groups/:groupId
    - Path parameters:
      * groupId - group identifier
  + Content-Type: application/json
  + Body:

{

"name": "group1",

"description": "description of group 1"

}

* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: application/json
    - Body example:
  + {
  + "status" : "Group updated" ,
  + "uri": "/groups/1 "

}

* Errors:
  + Status Code: 400 Bad Request
  + Body:
  + {
  + "error": "Bad request",
  + "errorDetail": "Parameter x error"
  + "uri": "/groups/1"

}

* + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/groups/2"}

## Add a game to a group

* Request:
  + Method: PUT
  + Path: /groups/:groupId/games/:gameId
    - Path parameters:
      * groupId - group id
      * gameId - game id
  + Content-Type: application/json
  + Body: none
* Response:
  + Success:
    - Status code: 200 OK
    - Content-Type: application/json
    - Body example:
  + {
  + "status" : "Group updated" ,
  + "uri": "/groups/1/games/1 "

}

* Errors:
  + Status Code: 400 Bad Request
  + Body:
  + {
  + "error": "Bad request",
  + "errorDetail": "Parameter x error",
  + "uri": "/groups/1/games/1"

}

* + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/groups/1/games/1"

}

## Delete a game from a group

* Request:
  + Method:DELETE
  + Path: /groups/:groupId/games/:gameId
    - Path parameters:
      * groupId - group identifier
      * gameId - game identifier
  + Content-Type:
  + Body: none
* Response:
  + Success:
    - Status code: 200
    - Content-Type: application/json
    - Body example:
  + {
  + "status" : "Game removed",
  + "uri": "/groups/1/games/1"

}

* Errors:
  + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/groups/1/games/1"

}

## Get games from group that have an determined interval duration

* Request:
  + Method: GET
  + Path: /groups/:groupId/games/:min&:max
    - Path parameters:
      * groupId - group id
      * min - minimum duration
      * max - maximum duration
  + Content-Type:
  + Body: none
* Response:
  + Success:
    - Status code: 200
    - Content-Type: application/json
    - Body example:
  + {
  + "games":[
  + {
  + "name":"game1",
  + "min\_playtime":"45",
  + "max\_playtime":"90",
  + "description":"Description of game 1"
  + }
  + {
  + "name":"game2",
  + "min\_playtime":"45",
  + "max\_playtime":"90",
  + "description":"Description of game 2"
  + }
  + ]

}

* Errors
  + Status Code: 404 Not Found
    - Body:
  + {
  + "error": "Resource not found" ,
  + "uri": "/groups/1"

}

### Implementação da aplicação

Implementação - Instalação

Deverá instalar a base de dados Elastic Search e na mesma correr os comandos abaixo para criação da tabela de grupos e da tabela de utilizadores.

curl -X PUT http://localhost:9200/groups  
curl -X PUT http://localhost:9200/users

Com a base de dados instalada deverá descarregar a aplicação através da página do github. Poderá fazer clone com o comando abaixo.

git clone <https://github.com/antmps/PI-1920i-LI51N-G09.git>

Na pasta da aplicação deverá fazer a instalação dos módulos npm utilizando o comando abaixo.

npm install

Para correr a aplicação deverá correr o comando abaixo

npm run -script build&npm start

Implementação – Testes

A página base tem os menus da aplicação no topo da página. Abaixo segue a explicação de cada menu.

* *Home*: página inicial apenas com o título da aplicação;
* *Search* *Games*: página para pesquisa de jogos;
* *Most* *Popular*: os jogos que se encontram no topo;
* *Groups*: permite visualizar os grupos e criar novos grupos;
* *Login*: Permite fazer autenticação ou criação de um novo utilizador;
* *Logout*: Permite a saída do utilizar *logado.*

Na página *Search Games* após efectuar a pesquisa de um jogo pode clicar no nome do mesmo para aceder a mais detalhes do jogo.

Em *Most Popular* teremos a informação dos jogos que se encontram no top 10 e aqui é novamente possível aceder ao detalhe de cada jogo clicando no nome do jogo.

Na página de detalhes de cada jogo, denominada *GameDetails* é possível adicionar o jogo a um grupo, desde que o utilizador já se tenha autenticado.

A página *Groups* permite a criação e a visualização dos grupos que temos criado. Tal como para adicionar um jogo a um grupo, o acesso a esta página está limitado a utilizadores autenticados.

A página de *Login* permite a criação de um utilizador e a autenticação.