

TRABALHO

Projeto Mongo

Artur José Gomes Pereira

Nº 2040415

João José da Costa Cabral

Nº 2020919

**Tecnologias e Programação de Sistemas de
Informação**

UNIDADE CURRICULAR:

Sistemas Gestores de Bases de Dados I

DOCENTE:

Magno Andrade

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Índice

Introdução	2
Problema	2
Funcionalidades	3
Estrutura Da Base De Dados	3
Esquema De Validação	4
<i>Validações Games</i>	4
<i>Validações Seller</i>	4
<i>Validações Game Developers</i>	4
<i>Validações Game Types</i>	4
<i>Validações Game Publishers</i>	4
<i>Validações Users</i>	4
Queries De Inserção/Criação	4
<i>Criação Da Base De Dados</i>	4
<i>Criação Da Coleção Games</i>	5
<i>Criação Da Coleção Seller</i>	5
<i>Criação Da Coleção Game Developers</i>	5
<i>Criação Da Coleção Game_Types</i>	5
<i>Criação Da Coleção Game Publisher</i>	5
<i>Criação Da Coleção Users</i>	5
Updates Da Base De Dados	5
<i>Alterando O Nome De Um Estudio</i>	5
<i>Alterando Todos O Nome Dos Campos</i>	6
<i>Adicionando Ao Array</i>	6
<i>Alterando Os Tipos Do Jogo Para Ingles</i>	6
Removes Da Base De Dados	6
<i>Remove Os Documentos Que Não Tem Stock</i>	6
<i>Removendo O Id Do Vendedor Do Jogo</i>	6
<i>Remove O Pais Dos Publishers</i>	6
<i>Remove Os Países Que Têm "U"</i>	6
<i>Remove Os Jogos Do Tipo Aventura</i>	6
<i>Apaga Todos Os Jogos Lançados Depois Da Data</i>	6
<i>Apaga Todos As Vendas Que O Valor É Maior Que 10</i>	6
Queries De Pesquisa	7
<i>Procura Todos Os Jogos Com Data Posterior Ao Indicado</i>	7
<i>Indica Todos Os Jogos Que Tem O Vendedos Com O Id 2</i>	7
<i>Indica Todos Os Jogos Que Um Vendedor Tem</i>	8
<i>Indica Quantos Vendedores Estam Atualmente A Vender Jogos</i>	8
<i>Pesquisa E Indica O Jogo Procurado Se Têm Na Base De Dados</i>	8
<i>Pesquisa E Apresenta Os Jogos Que Tem No Minimo Uma Key A Venda</i>	8
<i>Pesquisa Os Utilizadores</i>	9
Analise De Dados	9

<i>Indica O Publisher Do Jogo E O Seu Nome</i>	<i>9</i>
<i>Todos Os Jogos De Um User Que Estam A Venda E O Jogo.</i>	<i>10</i>
<i>Mostrar Apenas Of Filmes Que São De O Publisher De Um Destino.....</i>	<i>11</i>
<i>Indica O Todos Os Jogos Que Um Respetivo User Está A Vender Com O Valor Superior A 10€.....</i>	<i>11</i>
<i>Indica O Valor Médio De Um Jogo Que Está A Venda</i>	<i>11</i>
Mean Stack Part 2 Projeto.....	12
<i>Alteração À Base De Dados.....</i>	<i>12</i>
Nova Estrutura Da Base De Dados	12
Angular.Js	13
<i>Views</i>	<i>13</i>
<i>Routes</i>	<i>14</i>
<i>Controllors.....</i>	<i>15</i>
Express.Js	19
<i>Vantagens / Desvantagens Da Solução Desenvoldida</i>	<i>21</i>
Conclusão	22
Anexos.....	22

Introdução

Este relatório é referente a primeira parte do projeto solicitado pelo professor Magno Andrade. O trabalho tem como objetivo aplicar as principais técnicas de modelação de dados, bem como implementar numa base de dados criada pelo nosso grupo os conhecimentos adquiridos na unidade de Sistemas Gestores de Base de Dados 2.

Os seus principais objetivos consistem na análise de um problema, e aplicar vários métodos para a resolução desse mesmo problema.

Tomamos a decisão de criar uma base de dados para uma pagina web de venda de keys de jogos, uma vez que achamos o tema interessante e que seria um desafio.

A ideia por detrás da base de dados e da pagina web é dar a um vendedor a capacidade de vender um produto, e dar ao comprador a possibilidade de comprar jogos a preços mais apelativos do que nas principais plataformas de vendas de jogos online.

Problema

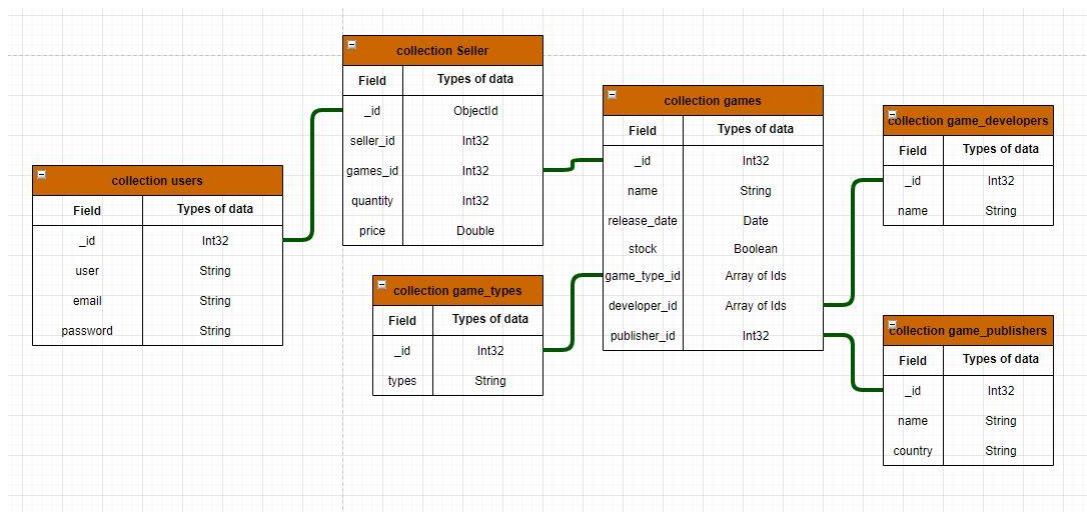
O nosso problema consiste em ter um website com AngularJS que permite aos nossos clientes poderem visualizar os jogos que estão a venda e toda a informação que consideramos relevante desses jogos.

Funcionalidades

Inicialmente tivemos de criar uma base de dados com validações para os campos inseridos, o objetivo do projeto é criar um site onde apresenta informação sobre os jogos, se o utilizador for o administrador do website consegue efetuar as operações CRUD, caso seja um utilizador normal consegue visualizar toda a informação sobre os jogos, no entanto não consegue apagar/criar e editar os registos.

Discussão – Análise da solução proposta

Estrutura da base de dados



A coleção "games" vai guardar informação para conseguir verificar o nome dos jogos, a data em que os jogos foram lançados, se tem ou não stock, qual é o id do tipo de jogo, o id do publisher e o grupo de developers.

A coleção "seller" reúne os ids relativos aos "users" e ao "games". Também tem informação sobre a quantidade disponível e o preço.

A coleção "users" tem as informações sobre o utilizador, o email assim como a sua password e o seu id, foi criada uma relação de muitos-para-muitos com a coleção games uma vez que vários utilizadores podem estar a vender vários jogos em simultâneo.

Na coleção "game_publishers" também temos o nome do publisher do jogo e ainda o país de origem do mesmo. Tem uma relação de muitos-para-muitos com a coleção "games" uma vez que múltiplos publishers publicam múltiplos jogos.

Na coleção "game_developers" é onde o nome da team de developers esta guardada

A coleção "game_types" para que os jogos também possam ser identificados pelos seus géneros.

Esquema de Validação

VALIDAÇÕES GAMES

```
db.createCollection("games",{validator:{$jsonSchema:{bsonType:"object",required:["_id","name","game_type_id","developer_id","publisher_id","release_date","stock"],properties:{_id:{bsonType:"int",description:"requiredandmustbeastring"},name:{bsonType:"string",description:"requiredandmustbeastring"},game_type_id:{bsonType:"array",description:"requiredarray"},developer_id:{bsonType:"array",description:"requiredarray"},publisher_id:{bsonType:"int",description:"requiredandmustbeinteger"},release_date:{bsonType:"date",description:"requiredandmustbeinteger"},stock:{bsonType:"bool",description:"requiredandmustbeinteger"},}}})
```

VALIDAÇÕES SELLER

```
db.createCollection("seller",{validator:{$jsonSchema:{bsonType:"object",required:["price","quantity","games_id","seller_id"],properties:{price:{bsonType:"double",description:"mustbeadoubleiftherefieldexists"},quantity:{bsonType:"int",description:"requiredandmustbeinteger"},games_id:{bsonType:"int",description:"requiredandmustbeinteger"},seller_id:{bsonType:"int",description:"requiredandmustbeinteger"},}}})
```

VALIDAÇÕES GAME DEVELOPERS

```
db.createCollection("game_developers",{validator:{$jsonSchema:{bsonType:"object",required:["name"],properties:{name:{bsonType:"string",description:"requiredandmustbeastring"},}}})
```

VALIDAÇÕES GAME TYPES

```
db.createCollection("game_types",{validator:{$jsonSchema:{bsonType:"object",required:["types"],properties:{types:{bsonType:"string",description:"requiredandmustbeastring"},}}})
```

VALIDAÇÕES GAME PUBLISHERS

```
db.createCollection("game_publishers",{validator:{$jsonSchema:{bsonType:"object",required:["name","country"],properties:{name:{bsonType:"string",description:"requiredandmustbeastring"},country:{bsonType:"string",description:"requiredandmustbeastring"},}}})
```

VALIDAÇÕES USERS

```
db.createCollection("users",{validator:{$jsonSchema:{bsonType:"object",required:["user","email","password"],properties:{user:{bsonType:"string",description:"requiredandmustbeastring"},email:{bsonType:"string",pattern:"^.+\\@.+\\.",description:"requiredandmustbevalidemailaddress"},password:{bsonType:"string",description:"requiredandmustbeastring"},}}})
```

Queries de Inserção/Criação

CRIAÇÃO DA BASE DE DADOS

USE PressPlay

CRIAÇÃO DA COLEÇÃO GAMES

```
db.games.insertMany([{"_id":1,"name":"GrandTheftAutoV:PremiumOnlineEdition","release_date":newDate("2013-09-13"),"stock":true,"game_type_id":[1,2,7],"publisher_id":9,"developer_id":[1,2,3,4]},{ "_id":2,"name":"Assassin'sCreed:Valhalla","release_date":newDate("2020-11-10"),"stock":false,"game_type_id":[1,2],"publisher_id":1,"developer_id":[5]},{ "_id":3,"name":"ARK:SurvivalEvolved","release_date":newDate("2015-06-02"),"stock":true,"game_type_id":[1,2,7,4],"publisher_id":2,"developer_id":[7,8,9,10]},{ "_id":4,"name":"TheElderScrollsV:SkyrimSpecialEdition","release_date":newDate("2011-11-11"),"stock":true,"game_type_id":[1,2,4,9],"publisher_id":3,"developer_id":[11]},{ "_id":5,"name":"DeadbyDaylight","release_date":newDate("2016-06-14"),"stock":true,"game_type_id":[4,7,9],"publisher_id":4,"developer_id":[12]},{ "_id":6,"name":"TheWitcher3:WildHuntGOTYEdition","release_date":newDate("2015-05-18"),"stock":true,"game_type_id":[4,9],"publisher_id":5,"developer_id":[13]},{ "_id":7,"name":"ForzaHorizon4StandardEdition","release_date":newDate("2018-09-28"),"stock":true,"game_type_id":[6,7],"publisher_id":6,"developer_id":[14,15]},{ "_id":8,"name":"Cyberpunk2077","release_date":newDate("2020-12-10"),"stock":true,"game_type_id":[1,2,4],"publisher_id":5,"developer_id":[13]},{ "_id":9,"name":"Borderlands3","release_date":newDate("2019-09-13"),"stock":true,"game_type_id":[1,2,7,9],"publisher_id":7,"developer_id":[16,17]},{ "_id":10,"name":"CallOfDutyBlackOps:ColdWar","release_date":newDate("2020-11-13"),"stock":true,"game_type_id":[1,2,4,7,9],"publisher_id":8,"developer_id":[18,19,20]},{ "_id":11,"name":"RedDeadRedemption2","release_date":newDate("2018-10-26"),"stock":true,"game_type_id":[1,2,4],"publisher_id":5,"developer_id":[1,2,3,4]})
```

CRIAÇÃO DA COLEÇÃO SELLER

```
db.seller.insertMany([{"seller_id":1,"quantity":3,"price":9.95,"games_id":1},{ "seller_id":1,"quantity":5,"price":11.90,"games_id":1},{ "seller_id":2,"quantity":1,"price":20.49,"games_id":1},{ "seller_id":3,"quantity":5,"price":10.49,"games_id":3},{ "seller_id":3,"quantity":20,"price":12.99,"games_id":3},{ "seller_id":2,"quantity":1,"price":7.99,"games_id":4},{ "seller_id":3,"quantity":100,"price":14.99,"games_id":5},{ "seller_id":4,"quantity":6,"price":28.99,"games_id":6},{ "seller_id":4,"quantity":8,"price":13.99,"games_id":7},{ "seller_id":4,"quantity":6,"price":28.99,"games_id":8},{ "seller_id":1,"quantity":15,"price":54.99,"games_id":9},{ "seller_id":2,"quantity":1,"price":49.99,"games_id":10}])
```

CRIAÇÃO DA COLEÇÃO GAME DEVELOPERS

```
db.game_developers.insertMany([{"_id":1,"name":"RockstarNorth"}, {"_id":2,"name":"RockstarSanDiego"}, {"_id":3,"name":"RockstarLeeds"}, {"_id":4,"name":"RockstarToronto"}, {"_id":5,"name":"UbisoftMontreal"}, {"_id":7,"name":"VirtualBasement"}, {"_id":8,"name":"EfectorStudios"}, {"_id":9,"name":"InstinctGames"}, {"_id":10,"name":"Abstraction"}, {"_id":11,"name":"BethesdaGameStudios"}, {"_id":12,"name":"BehaviourInteractive"}, {"_id":13,"name":"CDProjektRED"}, {"_id":14,"name":"PlaygroundGames"}, {"_id":15,"name":"Turn10Studios"}, {"_id":16,"name":"GearboxSoftware"}, {"_id":17,"name":"GearboxStudioQuebec"}, {"_id":18,"name":"Treyarch"}, {"_id":19,"name":"Beenox"}, {"_id":20,"name":"RavenSoftware"}])
```

CRIAÇÃO DA COLEÇÃO GAME_TYPES

```
db.game_types.insertMany([{"_id":1,"types":"Ação"}, {"_id":2,"types":"Aventura"}, {"_id":3,"types":"Estratégia"}, {"_id":4,"types":"RPG"}, {"_id":5,"types":"Desporto"}, {"_id":6,"types":"Corrida"}, {"_id":7,"types":"Jogoonline"}, {"_id":8,"types":"Simulação"}, {"_id":9,"types":"Outros gêneros"}])
```

CRIAÇÃO DA COLEÇÃO GAME PUBLISHER

```
db.game_publishers.insertMany([{"_id":1,"name":"Ubisoft","country":"France"}, {"_id":2,"name":"WildCard","country":"UnitedStates"}, {"_id":3,"name":"BethesdaGameStudios","country":"UnitedStates"}, {"_id":4,"name":"BehaviourInteractive","country":"Canada"}, {"_id":5,"name":"CDProjekt","country":"Poland"}, {"_id":6,"name":"Turn10Studios","country":"UnitedStates"}, {"_id":7,"name":"GearboxSoftware","country":"UnitedStates"}, {"_id":8,"name":"Activision","country":"UnitedStates"}, {"_id":9,"name":"RockstarGames","country":"UnitedStates"}])
```

CRIAÇÃO DA COLEÇÃO USERS

```
db.users.insertMany([{"_id":1,"user":"ArturPereira","email":"arturpe95@gmail.com","password":"2040415"}, {"_id":2,"user":"JoaoCabrai","email":"naoseioteuEmail@gmail.com","password":"2030919"}, {"_id":3,"user":"MagnoAndrade","email":"magno.andrade@staff.uma.pt","password":"trabalhoMerece20"}, {"_id":4,"user":"JuanMartin","email":"elmacholatino@gmail.com","password":"randomUser12345"}])
```

UPDATES DA BASE DE DADOS

ALTERANDO O NOME DE UM ESTUDIO

```
db.game_publishers.update(
```

```
{"name":"Bethesda Game Studios"},"{"name":"Bethesda Game Studio"}}
```

ALTERANDO TODOS O NOME DOS CAMPOS

```
db.games.updateMany({},{$rename:{"game_type_id":"type_id"}})
```

ADICIONANDO AO ARRAY

```
db.games.update({'_id':11},{$push: {'type_id': 3}})
```

ALTERANDO OS TIPOS DO JOGO PARA INGLES

```
db.games_types.update({},
```

```
 {"type":"Ação"},{$set:{"type":"Action"}})
```

```
db.games_types.update({},
```

```
 {"type":"Aventura"},{$set:{"type":"Adventure"}})
```

```
db.games_types.update({},
```

```
 {"type":"Estratégia"},{$set:{"type":"Strategy"}})
```

```
db.games_types.update({},
```

```
 {"type":"Desporto"},{$set:{"type":"Sports"}})
```

```
db.games_types.update({},
```

```
 {"type":"Corrida"},{$set:{"type":"Running"}})
```

```
db.games_types.update({},
```

```
 {"type":"Jogo online"},{$set:{"type":"Online Game"}})
```

```
db.games_types.update({},
```

```
 {"type":"Simulação"},{$set:{"type":"Simulation"}})
```

```
db.games_types.update({},
```

```
 {"type":"Outros gêneros"},{$set:{"type":"Other genders"}})
```

REMOVES DA BASE DE DADOS

REMOVE OS DOCUMENTOS QUE NÃO TEM STOCK

```
db.games.remove({'stock':false})
```

REMOVENDO O ID DO VENDEDOR DO JOGO

```
db.seller.remove({'seller_id':3})
```

REMOVE O PAIS DOS PUBLISHERS

```
db.game_publishers.remove({'country':'United States'})
```

REMOVE OS PAISES QUE TÊM "U"

```
db.game_publishers.remove({'country': /. *U.*/})
```

REMOVE OS JOGOS DO TIPO AVENTURA

```
db.games_types.remove({'type': 'Aventura'})
```

APAGA TODOS OS JOGOS LANÇADOS DEPOIS DA DATA

```
db.games.deleteMany({'release_date':{'$gt': new Date('2000-01-01')}})
```

APAGA TODAS AS VENDAS QUE O VALOR É MAIOR QUE 10

```
db.seller.remove({'price':{'$gt': 10 }})
```

QUERIES DE PESQUISA

PROCURA TODOS OS JOGOS COM DATA POSTERIOR AO INDICADO

```
db.games.find({"release_date":{"$gt: new Date('2000-01-01')}}).pretty()
```

- ➔ A query procura os jogos que tem um release_date superior ao indicado e apresenta os mesmos.

```
{
  "_id" : 11,
  "name" : "Red Dead Redemption 2",
  "release_date" : ISODate("2018-10-26T00:00:00Z"),
  "stock" : true,
  "game_type_id" : [
    1,
    2,
    4
  ],
  "publisher_id" : 5,
  "seller" : [
    {
      "seller_id" : 2,
      "quantity" : 1,
      "price" : 49.99
    }
  ]
}
```

INDICA TODOS OS JOGOS QUE TEM O VENDEDORES COM O ID 2

```
db.games.find({"seller.seller_id":2}).pretty()
```

- ➔ A query vai a tabela jogos, vai procurar no subdocumento o ID do vendedor indicado e apresenta-o.

```
{
  "_id" : 8,
  "name" : "Cyberpunk 2077",
  "release_date" : ISODate("2020-12-10T00:00:00Z"),
  "stock" : true,
  "game_type_id" : [
    1,
    2,
    4
  ],
  "publisher_id" : 5,
  "seller" : [
    {
      "seller_id" : 2,
      "quantity" : 1,
      "price" : 49.99
    }
  ]
}
```

INDICA TODOS OS JOGOS QUE TEM O VENDEDOR COM O ID 1

```
db.seller.aggregate([{$lookup:{from:"games",localField:"games_id",foreignField:"_id",as:"CurrentlySelling"}},{ $match: {'seller_id':1}}]).pretty()
```

Usamos um aggregate para juntas ambas as tabelas após usamos um match para definir uma condição.


```

{
  "_id" : ObjectId("5fd9fd1897955d4c2c1dc72e"),
  "seller_id" : 1,
  "quantity" : 3,
  "price" : 9.95,
  "games_id" : 11,
  "currentlySelling" : [
    {
      "_id" : 11,
      "name" : "Red Dead Redemption 2",
      "release_date" : ISODate("2018-10-26T00:00:00Z"),
      "stock" : true,
      "game_type_id" : [
        1,
        2,
        4
      ],
      "publisher_id" : 5,
      "developer_id" : [
        1,
        2,
        3,
        4
      ]
    }
  ]
}

```

INDICA TODOS OS JOGOS QUE UM VENDEDOR TEM

```
db.seller.find({'seller_id':2}).count()
```

- ➔ Pesquisa o nome do utilizador e depois utiliza o método count para apresentar o valor.

```

> db.seller.find({"seller_id":2}).count()
3

```

INDICA QUANTOS VENDEDORES ESTAM ATUALMENTE A VENDER JOGOS

```
db.seller.distinct("seller_id").length
```

- ➔ A query pesquisa na tabela seller utilizando a função distinct() os diferentes ids que estam no subdocumento associado a tabela e indica a sua dimensão

```

> db.seller.distinct("seller_id").length
4

```

PESQUISA E INDICA O JOGO PROCURADO SE TÊM NA BASE DE DADOS

```
db.games.find({'name':"Red Dead Redemption 2"}).pretty()
```

- ➔ A query vai a tabela dos jogos e procura pelo nome o jogo que for solicitado e indica toda a informação disponível desse jogo que se encontra na base de dados

```

> db.games.find({'name':"Red Dead Redemption 2"}).pretty()
{
  "_id" : 11,
  "name" : "Red Dead Redemption 2",
  "release_date" : ISODate("2018-10-26T00:00:00Z"),
  "stock" : true,
  "game_type_id" : [
    1,
    2,
    4
  ],
  "publisher_id" : 5,
  "developer_id" : [
    1,
    2,
    3,
    4
  ]
}

```

PESQUISA E APRESENTA OS JOGOS QUE TEM NO MINIMO UMA KEY A VENDA

```
db.games.aggregate([{$match:{"stock":{$eq: true}}])
```

- ➔ Procura todos os jogos que tem o stock como "true" e depois apresenta os mesmos.

```
{
  "_id" : 10,
  "name" : "Call of Duty Black Ops: Cold War",
  "release_date" : ISODate("2020-11-13T00:00:00Z"),
  "stock" : true,
  "game_type_id" : [
    1,
    2,
    4,
    7,
    9
  ],
  "publisher_id" : 8,
  "developer_id" : [
    18,
    19,
    20
  ]
}
```

PESQUISA OS UTILIZADORES

db.users.find()

- ➔ Apresenta todos os utilizadores assim como as suas informações

```
{ "_id" : 1, "user" : "Artur Pereira", "email" : "arturpe95@gmail.com", "password" : "2040415" }
{ "_id" : 2, "user" : "Joao Cabral", "email" : "naoseioteuEmail@gmail.com", "password" : "2030919" }
{ "_id" : 3, "user" : "Magno Andrade", "email" : "magno.andrade@staff.uma.pt", "password" : "trabalhoMerece20" }
{ "_id" : 4, "user" : "Juan Martin", "email" : "elmacholatino@gmail.com", "password" : "randomUser12345" }
```

ANALISE DE DADOS

INDICA QUAIS OS JOGOS QUE OS UTILIZADORES TAO A VENDER E O SEU PREÇO

```
db.seller.aggregate([{$lookup:{from:"users",localField:"seller_id",foreignField:"_id",as:"User_id"},{$lookup:{from:"games",localField:"games_id",foreignField:"_id",as:"Games"}},{$project":{"_id":0,"email":1,"User_id.user":1,"Games.name":1,"price":1}}]).pretty()
```

- ➔ Utilizamos dois lookups com aggregate para juntas a coleção users e games ao dos sellers e depois utilizamos um project para apenas apresentar a informação necessária.

```
{
  "price" : 49.99,
  "User_id" : [
    {
      "user" : "Joao Cabral"
    }
  ],
  "Games" : [
    {
      "name" : "Call of Duty Black Ops: Cold War"
    }
  ]
}
```

INDICA O PUBLISHER DO JOGO E O SEU NOME

```
db.game_publishers.aggregate([{$lookup:{from:"games",localField:"_id",foreignField:"publisher_id",as:"Publisher_Game"}},{$project":{"_id":1,"name":1,"Publisher_Game.name":1}}]).pretty()
```

- ➔ Utilizamos o lookup novamente para criar um leftjoin das coleções e ter acesso a informação de ambas as coleções após criamos o campo com

o nome onde queremos ver a informação que ambos têm em comum e utilizamos o project para apresentar a informação que queremos

```
{
  "_id" : 5,
  "name" : "CD Projekt",
  "Publisher_Game" : [
    {
      "name" : "The Witcher 3: Wild Hunt GOTY Edition"
    },
    {
      "name" : "Cyberpunk 2077"
    },
    {
      "name" : "Red Dead Redemption 2"
    }
  ]
}
```

INDICA O NOME DO JOGO, O PREÇO QUE O JOGO ESTÁ A SER VENDIDO O SEU PUBLISHER O SEU TIPO E AINDA OS SEUS GAME DEVELOPERS

```
db.games.aggregate([{$lookup:{from:"seller",localField:"_id",foreignField:"games_id",as:"Seller"}},{$lookup:{from:"game_publishers",localField:"publisher_id",foreignField:"_id",as:"Publisher"}},{$lookup:{from:"game_developers",localField:"developer_id",foreignField:"_id",as:"Developers"}},{$lookup:{from:"game_types",localField:"game_type_id",foreignField:"_id",as:"Type_of_Game"}},{$project:{ "_id":0,"name":1,"Seller.price":1,"Seller.seller_id":1,"Publisher.name":1,"Developers.name":1,"Type_of_Game.type":1}}]).pretty()
```

- ➔ Neste caso foi necessário utilizar quatro lookups de uma só vez visto que temos quatro coleções distintas que temos de unir à nossa coleção principal.
- ➔ Depois simplesmente utilizamos o project de forma a não aparecer tudo o que encontra mas apenas a informação que pedimos.

```
{
  "name" : "Red Dead Redemption 2",
  "Seller" : {
    {
      "seller_id" : 1,
      "price" : 9.95
    }
  },
  "Publisher" : [
    {
      "name" : "CD Projekt"
    }
  ],
  "Developers" : [
    {
      "name" : "Rockstar North "
    },
    {
      "name" : "Rockstar San Diego"
    },
    {
      "name" : "Rockstar Leeds"
    },
    {
      "name" : "Rockstar Toronto"
    }
  ],
  "Type_of_Game" : [
    {
      "type" : "Outros géneros"
    },
    {
      "type" : "Aventura"
    },
    {
      "type" : "RPG"
    }
  ]
}
```

TODOS OS JOGOS DE UM USER QUE ESTAM A VENDA E O JOGO.

```
db.seller.aggregate([{$lookup:{from:"users",localField:"seller_id",foreignField:"_id",as:"Users"}},{$lookup:{from:"games",localField:"games_id",foreignField:"_id",as:"Games"}},{$match:{ "seller_id":1}},{$project:{ "_id":0,"Users.user":1,"Games.name":1}}]).pretty()
```

- ➔ Foi necessário criar dois lookups com o aggregate de forma a conectar as coleções necessárias, depois utilizamos o match de forma a inserir condições específicas para a pesquisa neste caso apenas queremos os jogos que um vendedor está a vender.

```
{
  "Users": [
    {
      "user": "Artur Pereira"
    }
  ],
  "Games": [
    {
      "name": "Red Dead Redemption 2"
    }
  ]
},
{
  "Users": [
    {
      "user": "Artur Pereira"
    }
  ],
  "Games": [
    {
      "name": "Grand Theft Auto V: Premium Online Edition"
    }
  ]
},
{
  "Users": [
    {
      "user": "Artur Pereira"
    }
  ],
  "Games": [
    {
      "name": "Borderlands 3"
    }
  ]
}
```

MOSTRAR APENAS OS FILMES QUE SÃO DE O PUBLISHER DE UM DESTINO

```
db.game_publishers.aggregate([{$lookup:{'from':'games','localField':'_id','foreignField':'publisher_id','as':'Games'}},$match:{'country':'UnitedStates'}},$project:{'_id':0,'name':1,'country':1,'Games.name':1}}]).pretty()
```

```
{
  "name": "Rockstar Games",
  "country": "United States",
  "Games": [
    {
      "name": "Grand Theft Auto V: Premium Online Edition"
    }
  ]
}
```

INDICA O TODOS OS JOGOS QUE UM RESPETIVO USER ESTÁ A VENDER COM O VALOR SUPERIOR A 10€

```
db.game_publishers.aggregate([{$lookup:{'from':'games','localField':'_id','foreignField':'publisher_id','as':'Games'}},$match:{'country':'UnitedStates'}},$project:{'_id':0,'name':1,'country':1,'Games.name':1}}]).pretty()
```

- ➔ Juntamos a coleção users com a dos games de forma a ter acesso aos utilizadores e aos jogos depois utilizamos um match como critério de pesquisa e restringimos a pesquisa ao nome do utilizar e aos jogos que têm um valor mais elevado que 10€

```
{
  "price": 54.99,
  "User": [
    {
      "user": "Artur Pereira"
    }
  ],
  "Game": [
    {
      "name": "Borderlands 3"
    }
  ]
}
```

INDICA O VALOR MÉDIO DE UM JOGO QUE ESTÁ A VENDA

```
db.seller.aggregate([{$lookup:{'from':'games','localField':'games_id','foreignField':'_id','as':'Game'}},$match:{'Game.name':'ARK:SurvivalEvolved'}},$group:{'_id':'$User.games_id','Valormediodojogo':{'$avg':'$price'}}}).pretty()
```

- ➔ Usamos novamente um lookup com o aggregate de forma a juntar ambas as coleções, depois criamos um critério de pesquisa com o match de forma a apenas aparecer o jogo que queremos verificar, utilizamos o group() para utilizar a função avg() que vai calcular a media do valor indicado.

```
{ "_id" : null, "Valor medio do jogo" : 11.74 }
```

MEAN STACK PART 2 PROJETO

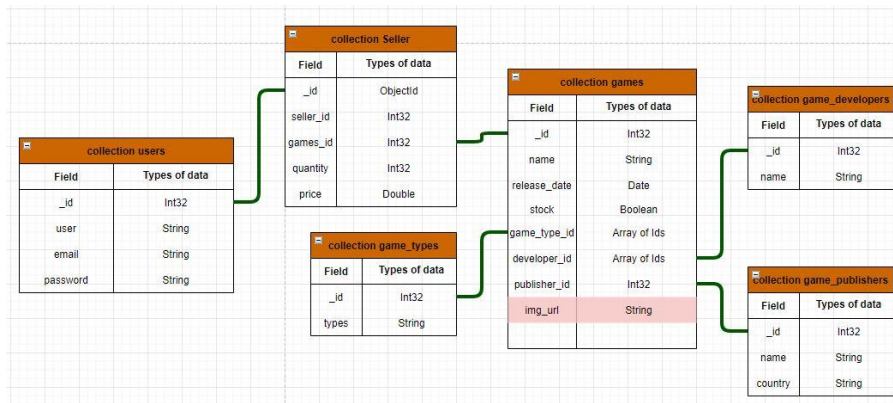
ALTERAÇÃO À BASE DE DADOS

```
db.games.update({},{$set:{"img_url":"1"}},false,true)
```

- ➔ De forma a podermos apresentar imagens no front end tivemos de alterar a nossa base de dados para guardar o URL das imagens e depois fizemos um update nos campos já existentes para inserir a imagem.

```
db.games.update({_id:1},{ $set:{"img_url":"https://upload.wikimedia.org/wikipedia/pt/8/80/Grand_Theft_Auto_V_capa.png"}})db.games.update({_id:2},{ $set:{"img_url":"https://store.ubi.com/dw/image/v2/ABBS_PRD/on/demandware.static/-/Sites-masterCatalog/default/dw75a90933/images/large/5e84a5065cdf9a21c0b4e737.jpg?sw=341&sh=450&sm=fit"}})db.games.update({_id:3},{ $set:{"img_url":"https://store-images.s-microsoft.com/image/apps.30459.13817182746640445.1152921504738442637.a405c13b-cd2b-4d17-8184-4957c880e4ee"}})db.games.update({_id:4},{ $set:{"img_url":"https://s1.gaming-cdn.com/images/products/1512/orig/the-elder-scrolls-v-skyrim-special-edition-cover.jpg"}})db.games.update({_id:5},{ $set:{"img_url":"https://cdn.cdkeys.com/500x706/media/catalog/product/d/e/dead-by-daylight-pc-get-cheap-cd-key_6.jpg"}})db.games.update({_id:6},{ $set:{"img_url":"https://s1.gaming-cdn.com/images/products/1497/orig/the-witcher-3-wild-hunt-goty-cover.jpg"}})db.games.update({_id:7},{ $set:{"img_url":"https://store-images.s-microsoft.com/image/apps.53613.14397339579473373.33c025e6-b311-42fd-a5d3-702031988979.354a28ca-1fd5-42ce-9d0a-7882e6884d43"}})db.games.update({_id:8},{ $set:{"img_url":"https://upload.wikimedia.org/wikipedia/pt/f/f7/Cyberpunk_2077_capa.png"}})db.games.update({_id:9},{ $set:{"img_url":"https://cdn.cdkeys.com/500x706/media/catalog/product/b/o/borderlands-3-pc-buy-now-cd-key_1.jpg"}})db.games.update({_id:10},{ $set:{"img_url":"https://store-images.s-microsoft.com/image/apps.25992.14107985044965209.e8fed65f-093d-40d1-849f-6c564d2ad876.89955624-8043-4f20-8620-4381e11a3546"}})db.games.update({_id:11},{ $set:{"img_url":"https://cdn.cdkeys.com/500x706/media/catalog/product/r/e/red-dead-redemption-2-ultimate-edition-cdkeys-pc.jpg"}})
```

Nova estrutura da base de dados



➔ O campo adicionado colocamos a vermelho para que seja mais fácil a sua visualização.

Angular.Js

Como solicitado pelo professor o nosso front end está como SPA na pagina principal app.html é onde temos a nossa navegação e a localização para os nossos controladores o angular assim como o express usa também o MVC.

```
<head>
<title>Press Play</title>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link rel="stylesheet" href="css/app.css" />
<!-- to use angular, ui-router, and our custom js file -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.7.5/angular.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular-ui-router/1.0.28/angular-ui-router.js"></script>

<script src="/routes/app.js"></script>
<script src="/controllers/gamesController.js"></script>
<script src="/controllers/userController.js"></script>
<script src="/controllers/createGameController.js"></script>
<script src="node_modules/angular-cookies/angular-cookies.js"></script>
</head>

<body ng-app="routerApp">
<div>
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
<div class="container">
<a class="navbar-brand" ui-sref="home">Press Play</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive"
aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarResponsive">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" ui-sref="home" ui-sref-active="active">Home
</a>
</li>
<li ng-show="loggedin" class="nav-item">
<a class="nav-link" ui-sref="register" ui-sref-active="active">Register</a>
</li>
<li ng-show="loggedin" class="nav-item">
<a class="nav-link" ui-sref="login" ui-sref-active="active">Login</a>
</li>
<li ng-show="loggedin" class="nav-item">
<a class="nav-link" ui-sref="dashboard" ui-sref-active="active">Dashboard</a>
</li>
<li ng-show="loggedin" class="nav-item">
<a class="nav-link" ui-sref="createGames" ui-sref-active="active">Create Games</a>
</li>
<li ng-show="loggedin" class="nav-item">
<a class="nav-link" ui-sref="logout" ui-sref-active="active">Logout</a>
</li>
</ul>
</div>
</div>
</nav>
<div>
<div class="container">
<div ui-view class="mainContainer"></div>
</div>
</div>
</body>
```

VIEWS

Estas são as viés que foram criadas e utilizadas no nosso front end.

createGames.html	U
dashboard.html	U
details.html	U
editGames.html	U
index.html	U
login.html	U
registerUser.html	U

ROUTES

O nosso ficheiro de rotas vai estar todas as rotas que estão disponíveis para o utilizador.

Exemplo de rota para utilizador:

```
.state("dashboard", {
  url: "/dashboard",
  controller: "userController",
  templateUrl: "views/dashboard.html",
  resolve: {
    Auth: function ($rootScope, $state) {
      if (!$rootScope.loggedIn) {
        $state.go("home");
      }
    },
  },
},
})
```

Exemplo de rota apenas disponível para o Admin:

```
.state("editGame", {
  url: "/edit/:id",
  controller: "createGameController",
  templateUrl: "views/editGames.html",
  resolve: {
    Auth: function ($rootScope, $state) {
      if (!$rootScope.isAdmin) {
        $state.go("home");
      }
    },
  },
},
})
```

O resolve o que vai fazer é antes de carregar o controlador ou o template da página, vai primeiramente verificar se o utilizador não está autenticado, se não estiver é redirecionado para a página inicial, caso esteja com o login feito ele deixa visualizar o conteúdo.

A autenticação funciona da seguinte forma, o utilizador ao criar uma conta automaticamente cria uma cookie e faz com que o boolean "loggedIn" seja true, redireciona para a página dashboard, se for o admin, ambos os boolean são true o que permite acesso a qualquer página do site, também temos definidos botões com o ng-show apenas para mostrar informação caso seja admin ou utilizador.

Para ter acesso à cookie em qualquer parte do site utilizamos a função run do angular em que em qualquer página que seja acedida no site vai verificar o role do user que está no nosso site.

Utilizadores que pode utilizar para verificar (**user/password**): teste/teste(ou um novo user) e admin/admin(para ter acesso às paginas protegidas)

```
routerApp.run([
  "$rootScope",
  "$location",
  "$cookies",
  function ($rootScope, $location, $cookies) {
    $rootScope.cookie = $cookies.getObject("loginCookie");
    if ($cookies.getObject("loginCookie") == null) {
      $rootScope.loggedIn = false;
    } else {
      $rootScope.loggedIn = true;
      if ($cookies.getObject("loginCookie").user == "admin") {
        $rootScope.isAdmin = true;
      }
    }
  },
]);
```

CONTROLLERS

Os nossos controllers chamam os end points da API de forma a ter ligação com o back end.

```
JS createGameController.js    U
JS gamesController.js         U
JS userController.js          U
```

```
JS userController.js
```

Controlador utilizado para criar novos utilizadores e onde se encontra o sistema de autenticação.

```
JS createGameController.js
```

Este é o controlador mais importante é onde podemos criar e editar os jogos que se encontram disponíveis no nosso site.

Consome a API e cria um POST da informação inserida no front end.

```
$scope.saveData = function (game) {
  if ($state.current.name === "createGames") {
    game.game_type_id = $scope.arr; //recebe array game type
    game.developer_id = $scope.arrDeveloper_id; //recebe array developer id
    $http({
      method: "POST",
      url: globalConfig.apiAddress + "/games",
      data: game,
    }).then($state.go("home"));
  }
};
```

Apresenta no front end os tipos de jogos, os developers e os publishers que temos disponíveis na base de dados.


```

$http({
  method: "GET",
  url: globalConfig.apiAddress + "/game_types",
}).then(
  function mySuccess(response) {
    $scope.game_types = response.data;
  },
  function myError(response) {
    console.log("erro");
  }
);

$http({
  method: "GET",
  url: globalConfig.apiAddress + "/game_developers",
}).then(
  function mySuccess(response) {
    $scope.game_developers = response.data;
  },
  function myError(response) {
    console.log("erro");
  }
);

$http({
  method: "GET",
  url: globalConfig.apiAddress + "/game_publishers",
}).then(
  function mySuccess(response) {
    $scope.game_publishers = response.data;
  },
  function myError(response) {
    console.log("erro");
  }
);

```

Depois de selecionar a opção carrega no botão que está disponível no front end em que corre a função push ou pushDev, vai adicionar ao array que criamos para depois enviar para o backend e gravar na base de dados.

```

$scope.arr = [];
$scope.push = function () {
  var inputVal = $scope.arrInput;
  $scope.arr.push(inputVal);
};

$scope.arrDeveloper_id = [];
$scope.pushDev = function () {
  var inputVal = $scope.arrInput2;
  $scope.arrDeveloper_id.push(inputVal);
};

```

Para editar um jogo existente a ideia é a mesma.

```

// array dos devs
$http({
  method: "GET",
  url: globalConfig.apiAddress + "/games/devs/" + $stateParams.id,
}).then(function mySuccess(response) {
  var devDB = response.data;
  Object.keys(devDB).forEach((element) => {
    devDB[0].Developers.forEach((element) => {
      $scope.devArray.push(element._id);
    });
  });
});

// array dos tipos
$http({
  method: "GET",
  url: globalConfig.apiAddress + "/games/type/" + $stateParams.id,
}).then(function mySuccess(response) {
  var devType = response.data;
  Object.keys(devType).forEach((element) => {
    devType[0].Type_of_Game.forEach((element) => {
      $scope.typeArray.push(element._id);
    });
  });
});

```

Procuramos pelos parâmetros o jogo existente e a sua informação e apresenta essa informação ao Admin para saber as opções de escolha disponíveis para editar.

```

$http({
  method: "GET",
  url: globalConfig.apiAddress + "/games/info/" + $stateParams.id,
}).then(
  function mySuccess(response) {
    $scope.games = response.data;
  },
  function myError(response) {}
);

$http({
  method: "GET",
  url: globalConfig.apiAddress + "/games/infoSeller/" + $stateParams.id,
}).then(
  function mySuccess(response) {
    $scope.sellers = response.data;
  },
  function myError(response) {}
);

```

Com o método PUT introduz na base de dados a nova informação que foi inserida.

```

$scope.saveEditedGame = function (editedGame) {
  if ($state.current.name === "editGame") {
    editedGame.developer_id = $scope.devArray; //recebe array developer id com os dados da BD mais os adicionados
    editedGame.game_type_id = $scope.typeArray; //recebe array game type

    $http({
      method: "PUT",
      url: globalConfig.apiAddress + "/games/" + $stateParams.id,
      data: editedGame,
    }).then(
      function mySuccess(response) {
        $state.go("home");
      },
      function myError(response) {
        console.log("something went wrong");
      }
    );
  }
};

```

Para o admin saber quais são developers e os tipos de jogo associados, temos um array que recebe a informação que os jogos já têm e depois adiciona a esse array os novos tipos/developers.

```

$scope.typeArray = [];
$scope.pushDevType = function () {
  var inputVal = $scope.game_type_id;
  $scope.typeArray.push(inputVal);
};

$scope.devArray = [];
$scope.pushDevNew = function () {
  var inputVal = $scope.developer_id;
  $scope.devArray.push(inputVal);
};

```

Depois é tudo enviado para o backend onde é validado e posteriormente adicionado a base de dados.

JS gamesController.js

Se o estado atual da rota for home consome a API e apresenta todos os jogos que estão na nossa base de dados depois temos um ng-repeat no front end de forma a mostrar todos os jogos.

```

if ($state.current.name === "home") {
  $http({
    method: "GET",
    url: globalConfig.apiAddress + "/games",
  }).then(
    function mySuccess(response) {
      $scope.games = response.data;
    },
    function myError(response) {}
  );
}

```

Em baixo podemos verificar se o estado atual for gameInfo apresenta a pagina detalhes em que podemos ver toda a informação que se encontra disponível na base de dados sobre o jogo que está a ser visualizado.

Temos tres endpoints em que no primeiro, procura a informação sobre o jogo, o segundo vai apresentar os vendedores atuais que o jogo têm, também temos o endpoint com o method DELETE para que, se o admin pretende apagar um jogo

carrega no botão que se encontra no front end e com um ng-click corre a função associada. (Opção está apenas disponível para o admin)

```
if ($state.current.name === "gameInfo") {
  $http({
    method: "GET",
    url: globalConfig.apiAddress + "/games/info/" + $stateParams.id,
  }).then(
    function mySuccess(response) {
      $scope.games = response.data;
    },
    function myError(response) {}
  );
  $http({
    method: "GET",
    url: globalConfig.apiAddress + "/games/infoSeller/" + $stateParams.id,
  }).then(
    function mySuccess(response) {
      $scope.sellers = response.data;
    },
    function myError(response) {}
  );
  /////////////////////////////////////////////////// DELETE GAME
  $scope.deleteGame = function () {
    if ($cookies.getObject("loginCookie").user == "admin") {
      $http({
        method: "DELETE",
        url: globalConfig.apiAddress + "/games/del/" + $stateParams.id,
      }).then(
        function mySuccess(response) {
          $state.go("home");
        },
        function myError(response) {}
      );
    }
    else {
      $state.go("home");
    }
  };
}
```

Esta função primeiro vai buscar o endpoint para criar um novo utilizador depois caso não ocorra erros redireciona o utilizador para a pagina inicial.

```
$scope.saveData = function (user) {
  if ($state.current.name === "register") {
    $http({
      method: "POST",
      url: globalConfig.apiAddress + "/users",
      data: user,
    }).then(
      function mySuccess(response) {
        $state.go("home");
      },
      function myError(response) {
        console.log("something went wrong");
      }
    );
  }
};
```

Nesta função temos um end point para ir buscar o objecto de todos os utilizadores, depois utilizamos um for para correr os resultados e verificar se o que está a ser inserido no form do login corresponde a um utilizador que está registado na base de dados, se for o caso ele redireciona o utilizador para a pagina dashboard, é criado um cookie e vai buscar esse cookie, verifica também se o utilizador inserido é o admin, se for o caso nós depois temos definidos no front end algumas questões em que apenas o admin consegue realizar.

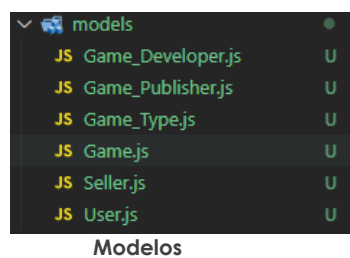
```
$scope.tryLogin = function (user) {
  if ($state.current.name === "login") {
    $http({
      method: "GET",
      url: globalConfig.apiAddress + "/users",
    }).then(function mySuccess(response) {
      for (i = 0; i < response.data.length; i++) {
        if ((response.data[i].user == user.user) && (response.data[i].password == user.password)) {
          $state.go("dashboard"); //redireciona para dash
          var expireDate = new Date(); //cria um obj de tempo nv
          expireDate.setDate(expireDate.getDate() + 1); //data de hj +1 dia
          $rootScope.loggedIn = true;
          $cookies.putObject('loginCookie', user, {'expires': expireDate}); //guarda a cookie com os dados
          $rootScope.cookie = $cookies.getObject("loginCookie"); //vai buscar o obj guardado na cookie
          if ($cookies.getObject("loginCookie").user == "admin") {
            $rootScope.isAdmin = true;
          }
        }
        else {
          $scope.errorMessage = "Wrong email or password";
        }
      }
    });
  }
};
```

Quando o utilizador carrega no botão de logout corre esta função, em que expira o cookie e recarrega a página, o utilizador automaticamente será redirecionado para uma página que não precisa de permissões (para isto acontecer foi definido previamente no app.js rotas com autorização e sem autorização)

```
$scope.logout = function() {  
  document.cookie =  
    "loginCookie=; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";  
  location.reload(true);  
}
```

Express.Js

O express foi utilizado como o nosso servidor, o express adopta o padrão MVC, foi necessário definir os modelos das nossas coleções da base de dados de forma a ter acesso aos documentos para as views utilizamos o angularJS.



O modelo principalmente utilizado foi o Modelo Game.js uma vez que foi o modelo principal para as operações CRUD, apesar de termos também criado algumas operações no modelo dos Users.js de forma a ter novos utilizadores no nosso site, no modelo Game.js existem varias agregações de \$lookup de forma a posteriormente poderemos usar o método populate() do mongoose.

```
gamesSchema.virtual("Seller", {  
  ref: "Seller",  
  localField: "_id",  
  foreignField: "games_id",  
});  
  
gamesSchema.virtual("Publisher", {  
  ref: "Game_Publisher",  
  localField: "publisher_id",  
  foreignField: "_id",  
});  
  
gamesSchema.virtual("Developers", {  
  ref: "Game_Developer",  
  localField: "developer_id",  
  foreignField: "_id",  
});  
  
gamesSchema.virtual("Type_of_Game", {  
  ref: "Game_Type",  
  localField: "game_type_id",  
  foreignField: "_id",  
});
```

Nas nossas rotas é onde encontra os endpoints utilizadas para procurar/criar/ler e editar a informação em baixo vamos mostrar um pouco acerca do que foi feito.

```
// Get All
router.get("/", (req, res, next) => {
  Game.find({}, (error, results, fields) => {
    if (error) {
      res.status(400).json("400- Bad Request");
    } else if (results.length == 0) {
      res.status(404).json("404- Not Found");
    } else {
      res.json(results);
    }
  });
});
```

- ➔ Utilizamos este endpoint no front end de forma a mostrar todos os jogos existentes na nossa base de dados.

```
router.get("/info/:id", (req, res, next) => {
  var id = req.params.id;
  Game.find({ _id: id }, (error, results, fields) => {
    res.json(results);
  })
  .populate("Seller")
  .populate("Publisher")
  .populate("Developers")
  .populate("Type_of_Game");
});
```

- ➔ Foi criado para a pagina de informação acerca dos jogos em que o populate vai preencher a informação que anteriormente já se encontrava no modelo Game.js e vai enviar a informação para o front end em que depois selecionamos o que vamos utilizar.

```
router.get("/devs/:id", (req, res, next) => {
  var id = req.params.id;
  Game.find({ _id: id }, (error, results, fields) => {
    res.json(results);
  })
  .populate("Developers");
});
```

- ➔ Apresenta os developers do jogo

```
router.get("/type/:id", (req, res, next) => {
  var id = req.params.id;
  Game.find({ _id: id }, (error, results, fields) => {
    res.json(results);
  })
  .populate("Type_of_Game");
});
```

- ➔ Apresenta os tipo de jogos associados ao jogo em questão

```
router.post("/", (req, res) => {
  Game.findOne({}, {}, { sort: { _id: -1 } }, function (err, post) {
    var model = {
      _id: (post._id + 1),
      name: req.body.name,
      stock: req.body.stock,
      game_type_id: req.body.game_type_id,
      publisher_id: req.body.publisher_id,
      developer_id: req.body.developer_id,
      img_url: req.body.img_url,
      release_date: new Date(req.body.release_date),
    };
    // console.log(model);
    Game.insertMany(model);
    res.send("Criado com Sucesso");
  }).select({ _id: 1 });
});
```

- ➔ Endpoint que nós utilizamos para criar um jogo, uma vez que utilizamos números inteiros em vez do ObjectId no modelo, foi necessário realizar uma

querie para primeiro procurar o ultimo ID que foi inserido, o ID assim é introduzido automaticamente sem repetidos, e o resto da informação é passada pelo body.

```
router.put("/:id", (req, res) => {
  Game.findByIdAndUpdate(req.params.id, req.body, (error, result) => {
    console.log(req.body);
    if (error) {
      res.status(400).json("400- Bad Request");
    } else if (result == null) {
      res.status(404).send("ID inexistente");
    } else {
      res.send("Atualizado com sucesso");
    }
  });
});
```

- Vai buscar por parâmetro o ID do jogo que pretendemos dar update, a informação também é introduzida pelo body.

```
router.delete("/del/:id", (req, res) => {
  var id = req.params.id;
  Game.findByIdAndDelete(id, (error, result, fields) => {
    if (error) {
      res.status(400).json("400- Bad Request");
    } else if (result == null) {
      res.status(404).send("ID inexistente");
    } else {
      res.json("Deleted");
    }
  });
});
```

- Recebe por parâmetro o ID do jogo que é para dar delete e depois apaga.

No ficheiro app.js consta todas as nossas dependências assim como o URI do nossos endpoints.

```
app.use('/', indexRouter);
app.use('/game_developers', game_developersRouter);
app.use('/game_publishers', game_publishersRouter);
app.use('/game_types', game_typesRouter);
app.use('/games', gamesRouter);
app.use('/users', userRouter);
app.use('/seller', sellerRouter);
```

VANTAGENS / DESVANTAGENS DA SOLUÇÃO DESENVOLVIDA

Uma das desvantagens de termos utilizado **Arquitetura Modelo-Visão-Controle**,

É a necessidade de um tempo maior para explorar e modelar o sistema, no entanto as suas vantagens compensam uma vez que é mais simples para transformar o interface sem alterar a camada de negócio e tem melhor desempenho e produtividade por causa da sua estrutura modular.

CONCLUSÃO

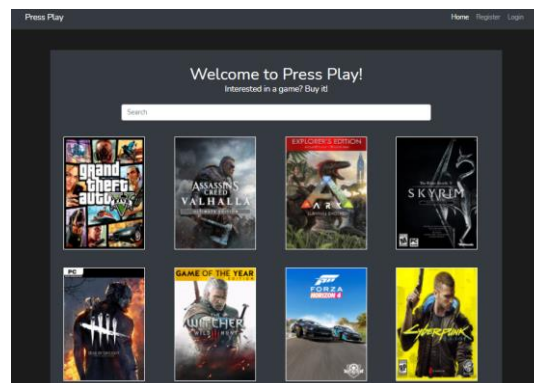
Após concluirmos a nossa base de dados, chegamos a conclusão que de forma a ser eficiente é necessária organização e uma definição de objetivos a atingir para poder trabalhar em equipa é necessária comunicação entre os dois membros da equipa e partilha de informação entre ambos, apesar de algumas dificuldades iniciais em definir os nossos objetivos conseguimos realizar a base de dados assim como organizar toda a informação que terá sido solicitado pelo professor.

Temos a agradecer ao professor Magno Andrade a informação transmitida durante o período de aulas uma vez que apesar de já termos utilizado o Express anteriormente na cadeira de Back-End a informação transmitida durante o semestre ajudou a melhorar alguns aspetos e ajudou-nos a relembrar alguns conteúdos.

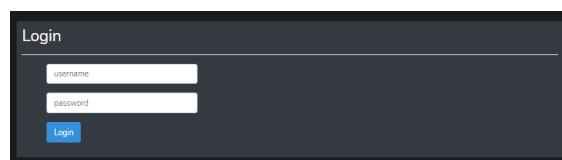
Finalizamos este relatório com a certeza de ter aumentado os nossos conhecimentos sobre o Mongo, AngularJS e o ExpressJS, que com certeza será útil durante toda a nossa futura jornada como programadores.

Anexos

→ [Pagina inicial](#)



→ [Login](#)



→ [Register](#)

Registration

Name

Email

Password

[Save](#) [Cancel](#)

→ Create Games

Press Play Home Dashboard Create Games Logout

Insert a new Game

Game Name mm/dd/yyyy

Current Genres = []

Current Devs = []

Stock


Select Game Publisher

[Insert Movie](#)

→ Details Page

Your selected game

[Go Back](#) [Edit Game](#)



Assassin's Creed: Valhalla

Game Developers:

- Ubisoft Montreal

Game Publisher:

- Ubisoft

Game Type:

- Acao
- Aventura

Currently Selling:

There are no sellers currently selling this game.


Release date:

2020-11-10T00:00:00.000Z

[DELETE GAME](#)

→ Edit Page

EDIT



New IMC link

Assassin's Creed: Valhalla

New Name

Game Developers:

- Ubisoft Montreal

Game Publisher:

- Ubisoft

Current Devs = [5]

Game Type:

- Acao
- Aventura

Release date:

2020-11-10T00:00:00.000Z

Current Types = [1,2]

[Save Movie](#) [Go Back](#)